

Allegro®

PCB SI User Guide

Product Version 23.1

September 2023

Last Updated: February 2022

© 2023 Cadence Design Systems, Inc. All rights reserved.

Portions © Apache Software Foundation, Sun Microsystems, Free Software Foundation, Inc., Regents of the University of California, Massachusetts Institute of Technology, University of Florida . Used by permission. Printed in the United States of America.

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

Allegro PCB SI contains technology licensed from, and copyrighted by: Apache Software Foundation, 1901 Munsey Drive Forest Hill, MD 21050, USA © 2000-2005, Apache Software Foundation. Sun Microsystems, 4150 Network Circle, Santa Clara, CA 95054 USA © 1994-2007, Sun Microsystems, Inc. Free Software Foundation, 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA © 1989, 1991, Free Software Foundation, Inc. Regents of the University of California, Sun Microsystems, Inc., Scriptics Corporation, © 2001, Regents of the University of California. Daniel Stenberg, © 1996 - 2006, Daniel Stenberg. UMFPACK © 2005, Timothy A. Davis, University of Florida, (davis@cise.ul.edu). Ken Martin, Will Schroeder, Bill Lorensen © 1993-2002, Ken Martin, Will Schroeder, Bill Lorensen. Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, Massachusetts, USA © 2003, the Board of Trustees of Massachusetts Institute of Technology. vtkQt, © 2000-2005, Matthias Koenig. All rights reserved.

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.

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.

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. Cadence is committed to using respectful language in our code and communications. We are also active in the removal and/or replacement of inappropriate language from existing content. This product documentation may however contain material that is no longer considered appropriate but still reflects long-standing industry terminology. Such content will be addressed at a time when the related software can be updated without end-user impact.

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>About this Manual</u>	15
<u>Intent</u>	15
<u>Audience</u>	15
<u>Where to Find Information</u>	15
<u>Conventions</u>	17
<u>Special Terms</u>	17
<u>How to Contact Technical Support</u>	18
 <u>1</u>	
<u>Introduction</u>	19
<u>What is Allegro PCB SI?</u>	19
<u>Why Use PCB SI?</u>	20
<u>How PCB SI Works</u>	20
<u>In the Front End</u>	20
<u>In the Back End</u>	20
<u>The PCB SI Toolset</u>	22
<u>SI</u>	23
<u>SigXplorer</u>	24
<u>SigNoise</u>	26
<u>Spectre®</u>	28
<u>Allegro Constraint Manager</u>	29
<u>SigWave</u>	30
<u>Allegro PCB PI</u>	31
<u>EMControl</u>	32
<u>Device Modeling Language (DML)</u>	33
<u>Allegro Design Entry HDL High Speed Option</u>	34
 <u>2</u>	
<u>The High-Speed Design Flows</u>	35
<u>Understanding the Flows</u>	35

Allegro PCB SI User Guide

<u>PCB SI Flow Overview</u>	35
<u>PCB SI Flow – Model Development and Verification</u>	36
<u>PCB SI Flow – Pre-Route Constraints Development</u>	37
<u>PCB SI Flow – Critical Component Pre-placement</u>	38
<u>PCB SI Flow – Solution Space Analysis</u>	39
<u>PCB SI Flow – Constraint-Driven Placement</u>	40
<u>PCB SI Flow – Constraint-Driven Routing</u>	41
<u>PCB SI Flow – Post-Route Design Rule Checking</u>	42
<u>PCB SI Flow – Post-Route Verification</u>	43
<u>MGH Flow Overview</u>	44
<u>MGH Flow – System-level Design</u>	45
<u>MGH Flow – Block-Level Interconnect Design</u>	45
<u>MGH Flow – System-Level Interconnect Evaluation</u>	46
<u>MGH Flow – Channel Analysis and Pre-emphasis Optimization</u>	47
<u>MGH Flow – Detailed Time Domain Verification</u>	47
<u>Getting Started with the PCB SI Flow</u>	49
<u>Model Development and Management</u>	49
<u>Pre-route Constraints Development</u>	53
<u>Critical Component Pre-Placement</u>	55
<u>Logic</u>	58
<u>Design Audit and Setup</u>	60
<u>Managing Models and Libraries</u>	63
<u>Managing Models</u>	67
<u>Model Verification and Source Management</u>	71

3

Model and Library Management

<u>Managing Model Libraries</u>	75
<u>Introduction to Model Libraries</u>	75
<u>Working with SI Model Browser</u>	75
<u>Performing Library Management</u>	77
<u>Basic Library Management</u>	78
<u>Advanced Library Management</u>	79
<u>Signal Integrity Model Libraries</u>	82
<u>Managing Models</u>	85

Allegro PCB SI User Guide

<u>Introduction to Simulation Models</u>	85
<u>Where to Obtain Device Models</u>	85
<u>Model Data Verification</u>	85
<u>The Cadence Sample Model Libraries</u>	86
<u>Available Models</u>	87
<u>Basic Model Development</u>	89
<u>Model Setup in Allegro Design Entry HDL or Third-Party Libraries</u>	94
<u>Advanced Model Development</u>	95
<u>Guidelines for Specifying Parasitic Values</u>	105
<u>Specifying Parasitics</u>	105
<u>Adding and Changing Pin Definitions</u>	107
<u>Adding or Editing Buffer Delay Information</u>	109
<u>More About Buffer Delays for an IBIS Device Model</u>	111
<u>Editing IBIS IOCell Models</u>	112
<u>Editing V/I Curve Data</u>	120
<u>Editing V/T Curve Data</u>	122
<u>Editing Espice Device Models</u>	123
<u>Editing PackageModels</u>	123
<u>Editing Analog Output Models</u>	123
<u>Editing and Regenerating Interconnect Models</u>	126
<u>Managing Models Resident in a Design</u>	127
<u>Auditing Models and Libraries</u>	132
<u>Model Translation</u>	135
<u>Model Translation Using SI Model Browser</u>	135
 <u>4 Transmission Line Simulation</u>	
<u>Overview</u>	137
<u>About the PCB and Package SI Simulator</u>	137
<u>Analysis Results</u>	138
<u>Simulation Setup</u>	141
<u>Setup Options</u>	141
<u>Initializing the Simulation Environment</u>	142
<u>Assigning Device Models</u>	145
<u>Setting Environment Variables</u>	151

<u>Auditing Simulation Setup</u>	157
<u>SI Audit Errors Report</u>	157
<u>Design Audit Checks</u>	160
<u>Simulation Run Directory Structure</u>	162
<u>Simulation Message Window and Log File</u>	168

5

<u>Floorplanning</u>	169
<u>Introduction</u>	169
<u>Board Setup</u>	170
<u>Cross-Section Stackup and Materials</u>	170
<u>Board Outline</u>	174
<u>Room Outlines</u>	174
<u>Plane Outlines</u>	176
<u>Keepouts</u>	177
<u>Importing Setup Data</u>	178
<u>Defining Logic</u>	182
<u>Drawing Logic Scenarios at the Board Level</u>	182
<u>Component Creation and Placement</u>	183
<u>Device Model Creation and Assignment</u>	184
<u>Netlist Creation</u>	185
<u>Logic Scenario Mock-up Example - A Look at Self-Coupling</u>	187

6

<u>Topology Extraction</u>	205
<u>Overview</u>	205
<u>Extraction Prerequisites</u>	205
<u>Extraction Setup</u>	206
<u>Unrouted Interconnect</u>	207
<u>Topology Template Formats</u>	208
<u>Physical and Extended Nets</u>	210
<u>Physical Net</u>	210
<u>Extended Net (Xnet)</u>	210
<u>Topology Template Extraction</u>	211
<u>Probing a Net to Extract a Topology Template</u>	211

<u>Using Constraint Manager to Extract a Topology Template</u>	213
<u>Benefits of Topology Templates with Routed Interconnect</u>	215
<u>Topology Simulation</u>	216

7

Determining and Defining Constraints 217

<u>Overview</u>	217
<u>Solution Space Analysis</u>	218
<u>What is Solution Space Analysis?</u>	218
<u>Solution Space Analysis – Stage 1</u>	219
<u>Solution Space Analysis – Stage 2</u>	220
<u>Solution Space Analysis – Stage 3</u>	222
<u>Solution Space Analysis – Stage 4</u>	225
<u>Solution Space Analysis – Stage 5</u>	225
<u>Solution Space Analysis – Stage 6</u>	227
<u>Parametric Sweeps</u>	228
<u>Specifying Part Parameter Values for Sweeping</u>	229
<u>Controlling Sweep Sampling and Coverage</u>	230
<u>Sweep Results</u>	231
<u>Saving and Restoring Sweep Cases</u>	232
<u>Defining High-Speed Constraints</u>	234
<u>What is a Constraint?</u>	234
<u>What is a Constraint Set?</u>	234
<u>Creating ECSets</u>	234
<u>Referencing ECSets</u>	237
<u>Setting Nets to Check Themselves for Crosstalk and Parallelism</u>	238
<u>Version Compatibility</u>	238

8

Signal Integrity Analysis 239

<u>Setting Simulation Preferences</u>	239
<u>DeviceModels Tab</u>	241
<u>InterconnectModels Tab</u>	242
<u>Simulation Tab</u>	244
<u>S-Parameters Tab</u>	247

Allegro PCB SI User Guide

<u>Units Tab</u>	249
<u>EMI Tab</u>	250
<u>More on Setting Preferences and Parameters</u>	252
<u>Pre-Route Analysis</u>	254
<u>Critical Net Analysis</u>	258
<u>Post-Route Verification</u>	259
<u>Interactive Simulation</u>	261
<u>Batch Simulation</u>	264
<u>Crosstalk Analysis</u>	264
<u>EMI Analysis</u>	266
<u>Multi-Board Analysis</u>	266
<u>Source Synchronous Bus Analysis</u>	267
<u>Analysis Results</u>	280
<u>Enhanced Bus Simulation Report</u>	280
<u>Analyzing to Generate Text Reports</u>	284
<u>Reflection Summary Report</u>	288
<u>Simulating with Custom Stimulus</u>	292
<u>Delay Report</u>	306
<u>Ringing Report</u>	311
<u>Single Net EMI Report</u>	317
<u>Parasitics Report</u>	320
<u>SSN Report</u>	322
<u>Segment Crosstalk Report</u>	327
<u>Crosstalk Summary Report</u>	333
<u>Crosstalk Detailed Report</u>	336
<u>Signal Quality Screening</u>	341
<u>Analyzing to Generate Waveforms</u>	343
<u>Specifying the Simulation Type</u>	346
<u>Common Tab Areas</u>	347
<u>Optional Tab Sections</u>	348
<u>Optional Tab Selection</u>	349
<u>Displaying and Interpreting Waveforms</u>	349
<u>Conductor Cross Sections</u>	350

9

Analyzing for Static IR-Drop	353
<u>Static IR Drop</u>	354

10

Post-Route Signal Integrity Analysis Using the 3D Field Solver	
355	

<u>Introduction</u>	355
<u>What is Sentinel-NPE?</u>	356
<u>Whole Package Modeling</u>	356
<u>PCB-Level Simulation</u>	361
<u>Supported Technologies</u>	361
<u>3D Field Solver Functional Differences</u>	361
<u>3D Modeling and Simulation Setup</u>	362
<u>Pre-Checking Your Design</u>	365
<u>Important Setup Guidelines</u>	365
<u>Illegal Bonding Wire Checks</u>	365
<u>Performing 3D Signal Integrity Simulation</u>	366
<u>Pre-simulation Checklist</u>	366
<u>3D Field Solution Progress and Control</u>	367
<u>3D Package and Interconnect Model Device Files</u>	369
<u>Package Model Formats</u>	370
<u>Model Parasitics Report</u>	370
<u>Multiport Net Support</u>	371
<u>S-Parameter Model Support</u>	378
<u>3D Field Solver Setup Guidelines</u>	379
<u>Interpreting 3D Modeling Messages</u>	383
.....	385

11

Dynamic Analysis with the EMS2D Full Wave Field Solver	387
<u>EMS2D Operating Parameters</u>	387
<u>Coplanar Waveguide Characterization</u>	387

<u>Library Enhancements</u>	392
<u>Dispersive Dielectric Material Support</u>	393
<u>S-Parameter Extraction</u>	395
<u>Lossy Transmission Line Modeling in HSPICE</u>	396
<u>Enhanced Etch Factor Support</u>	397
<u>Algorithm-Based Modeling</u>	401
<u>Using EMS2D</u>	406
 A	
<u>Constraint-Driven Layout</u>	407
<u> Introduction</u>	407
<u> Constraint-Driven Placement</u>	407
<u> Placement Stages</u>	408
<u> Constraint-Driven Routing</u>	410
<u> Routing Stages</u>	410
<u> Constraints that Affect Routing</u>	411
 B	
<u>System-Level Analysis</u>	413
<u> Introduction</u>	413
<u> What is a System Configuration?</u>	413
<u> What is a DesignLink?</u>	414
<u> What is a Cable Model?</u>	415
<u> Modeling Strategies</u>	415
<u> Working with System Configurations</u>	417
<u> New System Configurations</u>	417
<u> System Configuration Editor Controls</u>	419
<u> Existing System Configurations</u>	420
<u> Setting Constraints at the System Level</u>	423
<u> System-Level Simulation</u>	424
 C	
<u>Working with Crosstalk</u>	425
<u>Crosstalk DRCs</u>	426

<u>Crosstalk Simulations</u>	427
<u>Crosstalk Timing Windows</u>	428
<u>Crosstalk Methodology</u>	429
<u>Database Setup</u>	429
<u>Crosstalk Timing Window Definition</u>	429
<u>Crosstalk Table Generation</u>	430
<u>Exporting and Importing Crosstalk Tables</u>	434
<u>Constraints and Crosstalk-Driven Routing</u>	435
<u>Post-route Crosstalk Simulation</u>	440
<u>Crosstalk Troubleshooting</u>	445

D

<u>Working with Timing</u>	451
<u>Introduction</u>	451
<u>Timing Analysis Basics</u>	451
<u>Static Timing Analysis</u>	451
<u>Modern System Design</u>	452
<u>Flight Time</u>	453
<u>Synchronous Design Issues</u>	454
<u>Impact of Crosstalk on Bus Timing</u>	455
<u>SI Analysis Basics</u>	457
<u>The Signal Integrity Model</u>	457
<u>Measuring Interconnect Delay</u>	458
<u>Minimum and Maximum Delays</u>	459
<u>Component Timing</u>	460
<u>The Double-Counting Problem</u>	461
<u>Making The Pieces Fit Together</u>	463
<u>Determining the Buffer Delay</u>	463
<u>Measuring Flight Time</u>	464
<u>About Device Modeling</u>	465
<u>Integrating Timing and SI Analysis</u>	467
<u>Manual Approach</u>	467
<u>General Approach</u>	468
<u>Bus-level Timing Approach</u>	469
<u>Bus Timing Model</u>	470

E

<u>Working with Multi-GigaHertz Interconnect</u>	473
<u>Introduction</u>	473
<u>Serial Data Links</u>	473
<u>Inter-Symbol Interference (ISI)</u>	474
<u>Advanced Solutions for Multi-GigaHertz Signal Design</u>	474
<u>Channel Analysis</u>	474
<u>The Serial Data Channel</u>	476
<u>Macro Modeling</u>	477
<u>Building MacroModels</u>	480
<u>Via Modeling</u>	482
<u>Via Model Formats</u>	484
<u>Via Model Types</u>	485

F

<u>Modeling in the Interconnect Description Language</u>	487
<u>Overview</u>	487
<u>IDL Interconnect Line Segment Models</u>	487
<u>RLGC Matrix Values in Interconnect Models</u>	488
<u>Example Line Segment Model</u>	492
<u>IDL Via Models</u>	501
<u>Example Via Model</u>	501
<u>Coupled Multiple Vias</u>	505
<u>IDL Shape Models</u>	510
<u>Example Shape Model</u>	511

G

<u>DML Syntax</u>	515
<u>Overview</u>	515
<u>About DML files</u>	515
<u>Cadence Sample Device Model Library</u>	515
<u>File Structure</u>	515
<u>DML Syntax</u>	516
<u>Comments</u>	516

<u>ModelTypeCategory Keywords</u>	516
<u>Tokens</u>	518
<u>Parameters</u>	518
<u>Sub-parameters of PackageModel</u>	519
<u>Example of PackageModel</u>	522
<u>Sub-parameters of Cable</u>	523
H	
<u>Computations and Measurements</u>	525
<u>Overview</u>	525
<u>Pre-Analysis Requirements</u>	525
<u>Device Models</u>	525
<u>Stack Up Definition</u>	526
<u>Modeling Unrouted Interconnect</u>	526
<u>Signal Integrity Simulations and Computations</u>	527
<u>The tIsim Simulator and Simulations</u>	527
<u>Reflection Simulations</u>	527
<u>Segment-Based Crosstalk Estimation</u>	528
<u>Crosstalk Simulations</u>	528
<u>Timing-Driven Crosstalk Analysis</u>	529
<u>Simultaneous Switching Noise (SSN) Simulations</u>	530
<u>Comprehensive Simulations</u>	530
<u>Delay Computations</u>	530
<u>Distortion Computations</u>	534
<u>Simultaneous Switching Noise Measurements</u>	536
I	
<u>Cadence ESpice Language Reference</u>	537
<u>Overview</u>	537
<u>About the Input Format</u>	537
<u>Statements</u>	537
<u>Node Names</u>	539
<u>Using Numbers</u>	539
<u>Datapoint Sections</u>	540
<u>Statement Types</u>	540

Allegro PCB SI User Guide

<u>ESpice Syntax Fundamentals</u>	542
<u>Learning about DC Path to Ground</u>	545
<u>Using Parameters and Expressions</u>	545
<u>Using Subcircuits</u>	546
<u>Supported Circuit Elements</u>	548
<u>Describing Basic Elements</u>	549
<u>Describing Controlled Source Elements</u>	554
<u>Describing IBIS Behavioral Model Elements</u>	570
<u>Using Multi-Conductor Transmission Line Models</u>	583
<u>Using Control Statements</u>	587
<u>Creating ESpice Models for Use with Allegro® SI</u>	590
<u>Creating an ESpice Packaged Part</u>	590
<u>Using SigXplorer Sources and Functions with ESpice Devices</u>	593

About this Manual

Intent

The intent of this User Guide is to provide conceptual as well as high-level procedural information regarding the design and analysis of high-speed printed circuit boards and systems using Allegro® Sigrity SI.

Note: Although Sigrity SI and PCB Editor share the same database and graphic user interface with many commands in common, this manual focuses on functionality unique to the Allegro Sigrity SI environment.

For details on using the board layout functionality available in both Allegro Sigrity SI and PCB Editor, refer to the [*Allegro® PCB and Package User Guide*](#).

Audience

This manual is written for both signal integrity and electrical engineers who are familiar with current methods and practices used to design and analyze high-speed printed circuit boards and systems. It is intended for novice and intermediate users who need basic information regarding the Allegro X high-speed PCB and MGH design flows and operations within the Allegro Sigrity SI environment.

Where to Find Information

The Allegro Sigrity SI User Guide contains the following chapters and appendices.

- Chapter 1, [Introduction](#), explains what Allegro Sigrity SI is, how it works, as well as an overview of the PCB SI toolset.
- Chapter 2, [The High-Speed Design Flows](#), presents an overview of the Allegro X high-speed PCB and MGH design flows.
- Chapter 3, [Model and Library Management](#), describes how to obtain, manage and maintain simulation models and model libraries.
- Chapter 4, [Transmission Line Simulation](#), presents an overview of the transmission line simulator (TLsim) as well as details on how to perform simulation setup and audit.

Allegro PCB SI User Guide

About this Manual

- Chapter 5, Floorplanning, describes common board setup tasks and how to define logic from scratch.
- Chapter 6, Topology Extraction, describes extraction prerequisites, extraction set up, and how to extract a net topology into SigXplorer for simulation and analysis via SI or Constraint Manager.
- Chapter 7, Determining and Defining Constraints, discusses how to determine high-speed constraints through solution space analysis and how to define constraints in the Allegro X database.
- Chapter 8, Signal Integrity Analysis, discusses how to set simulation preferences, perform pre-route analysis, post-route verification, and describes the various types of analysis results that can be generated.
- Chapter 9, Analyzing for Static IR-Drop, discusses how to perform Static IR-Drop analysis.
- Chapter 10, Post-Route Signal Integrity Analysis Using the 3D Field Solver, discusses how to perform post-route signal integrity analysis using the 3D Field Solver.
- Chapter 11, Dynamic Analysis with the EMS2D Full Wave Field Solver, discusses how to perform dynamic analysis with the EMS2D Full Wave Field Solver.
- Appendix A, Constraint-Driven Layout, provides an overview of the constraint-driven placement and constraint-driven routing processes.
- Appendix B, System-Level Analysis, describes how to work with PCB system configurations (DesignLinks) and perform system-level simulation.
- Appendix C, Working with Crosstalk, describes how to set up, simulate and analyze a design for crosstalk mitigation.
- Appendix D, Working with Timing, discusses timing analysis basics.
- Appendix E, Working with Multi-GigaHertz Interconnect, provides an overview on MGH signal design and also discusses macro and via modeling.
- Appendix F, Modeling in the Interconnect Description Language, provides an overview on modeling in the Interconnect Description Language.
- Appendix G, DML Syntax, provides a summary of the syntax and structure of Device Model Library (DML) files and their use within package modeling.
- Appendix H, Computations and Measurements, provides an overview of the analytical approach used in signal integrity simulation.
- Appendix I, Computations and Measurements, describes the native language of the simulator *tlsim* and its input file format, ESpice.

Conventions

The following fonts, characters, and styles have specific meanings throughout this manual.

- Courier font identifies text that you type exactly as shown, such as command names, keywords, and other syntax elements.

For example:

`(average_pair_length [on | off])`

- Italic type identifies menu paths or dialog box buttons in the graphic user interface (GUI), titles of books, and may also be used to emphasize portions of text.

For example:

Choose *File – Quit*.

Click *Apply*.

To learn more about generating eye diagrams, refer to the *SigWave User Guide*.

- Italicized labels enclosed in angle brackets (<>) are placeholders for keywords, values, filenames, or other information that you must supply.

For example:

`<directory_path_name>`

Special Terms

The following special terms are used in this manual.

- *Click* means press and release the left mouse button.
- *Click-right* means press and release the right mouse button.
- *Double-click* means press and release the left mouse button twice, in rapid succession.
- *Drag* means press and hold the left mouse button while you move the pointer.
- *Select* means to click on (highlight) objects in the design (such as nets, or components) or click on items in a list (such as net names) within a dialog box for exclusive processing by a command.
- *Choose* means to navigate the menu system to highlight and click on a menu option in order to execute a command.

How to Contact Technical Support

If you have questions about installing or using Allegro PCB SI, contact the Cadence Customer Response Center at:

<http://support.cadence.com/wps/myportal/cos/psa/contacts>

Introduction

What is Allegro PCB SI?

Allegro PCB SI is an integrated design and analysis environment for electrical engineers who create high-speed digital printed circuit boards and systems. It allows you to explore and resolve electrical performance related issues at all stages of the design cycle. By exploring various design scenarios and making trade-offs between timing, signal integrity, crosstalk, power delivery and EMI, you can optimize electrical performance and reliability before committing a design for manufacture. You can perform high-speed analysis at the board, multi-board, or system level – across multiple system design configurations.

Why Use PCB SI?

High-speed system engineers must contend simultaneously with the issues of timing analysis, signal integrity, crosstalk, power delivery and EMI. While these issues are often addressed separately (using a variety of analysis tools), modern high performance designs require that these issues be addressed both collectively and continuously throughout the design process. As design rules are created and subsequently refined, they must be stored within the design database to drive the PCB design process. Changes made to the design to optimize signal integrity are almost certain to impact system timing and radiated EMI.

A design process that optimizes any high-speed design characteristic without assessing its impact on other high-speed issues can only prolong the design cycle and increase the chance of error. However, by enabling engineers to analyze and address all high speed design issues *concurrently*, PCB SI reduces both the time and risk required to bring a new high-speed system to market.

How PCB SI Works

In the Front End

PCB SI works with any front-end schematic capture system capable of outputting a packaged netlist. When the schematic is finished and a netlist is available, you simply read it into PCB SI. In the meantime, you can use SigXplorer to explore net topologies and develop electrical constraints (ECSets).

Allegro Constraint Manager lets you associate the topology templates you create in SigXplorer with critical nets at the schematic level in Allegro Design Entry HDL. This ensures that electrical constraints are implemented automatically once the netlist is read into either PCB SI or PCB Editor.

Electrical engineers can use Design Entry HDL SI to determine optimal constraints for non-critical nets in the front end. This frees up SI engineers to focus on new chip sets and very critical nets, saving time and money.

In the Back End

PCB SI is designed to work with PCB Editor and PCB Router. It shares a common database and constraint system with PCB Editor, providing seamless integration. This lets you perform high-speed analysis at any stage of the design cycle with your board partially or fully placed, partially or fully routed, or even in situations where a netlist is not yet available.

Allegro PCB SI User Guide

Introduction

The ability of PCB SI to directly read and write the PCB Editor database and its unique ability to create constraints that drive the placement and routing processes ensures that high-speed design rules are quickly and accurately implemented throughout the design.

The PCB SI Toolset

The PCB SI environment is comprised of several high-speed software tools. Each tool performs a specific design or analysis task. The actual toolset available within your environment is determined by the PCB SI product series that you have purchased from Cadence. To determine which PCB SI product you have, check with your CAD Systems Administrator.

The following tools comprise the PCB SI environment. For further details on a tool, click on its name.

Floorplanning

[SI](#)

Power Distribution Network Analysis

[Allegro PCB PI](#)

Topology Exploration and Development

[SigXplorer](#)

EMI Analysis

[EMControl](#)

Simulation Subsystem

SigNoise

Device Model Development

[Device Modeling Language \(DML\)](#)

[TLsim](#)

[Sigxsect](#)

[Spectre®](#)

Schematic-Level Constraints Integration

[Allegro Design Entry HDL High Speed Option](#)

Constraints Management

[Allegro Constraint Manager](#)

Waveform Analysis

[SigWave](#)

SI

SI provides a *physical view* of the board and lets you simulate and edit your PCB design.

Using SI you can:

- quickly and easily evaluate the effects of different placement strategies on design behavior.
- run board simulations from the PCB database without the need to translate data into another format.
- perform test routing using proposed electrical constraints to ensure that high-speed design rules are achievable before passing them on to the layout designer.

For further details, see [Chapter 5, “Floorplanning.”](#)

SigXplorer

SigXplorer is a graphical environment for exploring, analyzing and defining interconnect strategies. It provides an *electrical view* of the physical interconnect on the board and lets you explore different placement and routing strategies. Solution space analysis lets you quickly develop and capture a comprehensive set of design rules.

Using SigXplorer you can perform the following tasks:

- extract electrical views of nets from both placed (pre-route) and finished (post-route) databases.
- analyze and edit the net model in detail.
- set up and run single parameter topology simulations.
- capture design constraints to drive the physical design process.
- set up and run a series of related topology simulations (sweeps) where you vary topology element characteristics such as part parameter values and driver slew rates.
- extract electrical views of nets with associated (closed-form) via models from both placed (pre-route) and finished (post-route) databases.
- upgrade Closed Form via models interactively to other advanced via model formats (such as S-Parameter or Wide Band) for multi-gigahertz exploration.
- perform multi-gigahertz channel analysis.
- S-parameter capabilities.

SigXplorer Topology Editor

The Topology Editor is included with PCB SI and Design Entry HDL. It is particularly useful for the Allegro layout designer looking to modify the implementation of custom net scheduling that presumably came from an external source (such as a text document from a design engineer or a vendor design guide). You invoke the Topology Editor from the Constraint Manager nets worksheet. It is also presented by default in cases where you attempt to invoke SI SigXplorer and an appropriate Allegro SI license is unavailable. Otherwise, you are presented with a dialog box that allows you to select the SI SigXplorer tool to use.

Using SigXplorer Topology Editor you can:

- graphically define or edit custom net scheduling for an electrical constraint set when no simulation or analysis capability is available.
- experiment with the circuit topology by:

Allegro PCB SI User Guide

Introduction

- repositioning parts.
- viewing circuit parameters.
- printing the contents of the topology canvas.
- rescheduling the topology.
- redefining constraint rules.
- applying net schedule and constraint changes (update) back to Constraint Manager and your design database.

For further details on SigXplorer, refer to the [*Allegro SI SigXplorer User Guide*](#).

SigNoise

SigNoise is the Allegro simulation environment for signal integrity, crosstalk and optional EMI analysis. Using SigNoise, you can quickly examine or scan one or more signals by performing reflection simulations and crosstalk estimations on entire designs or on large groups of signals. You can also probe individual signals or small groups of signals where you want to delve into specific signal behaviors in detail through the generation of discrete text reports or waveforms.

SigNoise includes the following components.

- TLsim (Transmission line simulation) engine
- DML (Device Modeling Language)
- SigWave
- Sigxsect

TLsim

TLsim is a Cadence proprietary SPICE-based simulation engine that combines the advantages of traditional structural modeling with the speed of behavioral analysis. TLsim includes an IBIS-style behavioral driver that models I/O behavior based on the V-I and V-T data provided by behavioral modeling techniques. By combining both structural and behavioral modeling techniques, TLsim allows you to model complex device behavior accurately and efficiently.

For further details, see [Chapter 4, “About the PCB and Package SI Simulator.”](#)

Sigxsect

The Sigxsect (signal cross section) window allows you to view the geometry of interconnect models and the equipotential field lines between the cross sections of interconnect. SigNoise generates models for the interconnect in your design. The field solver generates the parasitic values in the model. The Sigxsect window displays a three-dimensional view of the interconnect and its parasitic values.

When the simulator writes a model, it includes all the trace segments that fall within the geometry window distance specified in the simulator’s Analysis Preferences dialog box. If another trace segment is sufficiently close to the trace segment you selected in the design window, you see two trace cross sections in the geometry display. When you display the sigxsect window, its geometry display shows a geometric cross-sectional representation of the interconnect model that you have selected.

Allegro PCB SI User Guide

Introduction

For further details, see [Chapter 8, “Conductor Cross Sections.”](#)

Spectre®

Spectre is a general-purpose circuit simulator that uses direct methods to simulate analog and digital circuits at the differential equation level. The Spectre simulation interface in SI (and SigXplorer) supports Spectre transistor-level simulation models. Spectre is similar in function and application to SPICE, but does not descend from SPICE. Spectre uses the same basic algorithms (implicit integration methods, Newton-Raphson, direct matrix solution, and so on), but the algorithms are implemented in new ways. These new algorithms make Spectre faster, more accurate, more reliable, and more flexible than previous SPICE-like simulators.

Using Spectre you can:

- develop DML macro models for Spectre IO buffer sub circuits.
- perform PCB-level simulations in PCB SI using transistor-level IO buffer circuit models.

Simulator

Allows you to choose a simulator for models. Choices are Tlsim, Hspice, and Spectre.

Allegro Constraint Manager

Allegro Constraint Manager provides you with a real-time display of high-speed rules and their status based on the current state of your design. It employs a spreadsheet-like interface that lets you capture, manage, and validate design rules in a hierarchical fashion. The interface presents two views of the constraint information in the database. One view allows you to see the different electrical constraint sets (ECSets) present in the database and their associated constraint values. The other view presents the different nets contained in the system, the names of the ECSets associated with those nets (if any), and their associated constraint values.

Using Constraint Manager you can:

- group all of your high-speed constraints for a collection of signals to create an ECSet.
- associate ECSets with nets to manage their actual implementation.
- display color-coded results of design analysis in real time alongside the constraint values in the spreadsheet to indicate success or failure.
- use SigXplorer with Constraint Manager to graphically create, edit and review electrical constraint sets as graphical topologies that act as an electronic blueprint of an ideal implementation strategy.

For further details, refer to the [*Allegro Constraint Manager User Guide*](#).

SigWave

SigWave is a waveform viewer. It displays waveforms based on data generated by simulation tools - emulating the way an oscilloscope works. It is closely integrated with PCB SI and provides board-level signal integrity analysis. SigWave supports the display of time domain, bus, frequency, and eye diagram graphs, as well as the application of fast fourier transforms (FFT).

Using SigWave you can:

- display the simulation results.
- load one or more previously saved waveform files in order to superimpose and compare the waveforms.
- view and edit spreadsheet data for a displayed waveform.
- measure and annotate displayed waveforms to prepare precise documentation of waveform analysis.

For further details, refer to the [*Allegro SI SigWave User Guide*](#).

Allegro PCB PI

Using SigWave, PCB PI presents a family of curves that describe impedance as a function of frequency at each cell on the PCB. These curves are plotted along with the power delivery system's target impedance. You can correct those areas on the board where the power delivery system impedance exceeds the target impedance by placing de-coupling capacitors whose resonant frequencies effectively lower the system impedance to within the allowable target impedance or by decreasing the inter-plane dielectric thickness for frequencies too high for de-coupling capacitors to be effective.

Using PCB PI you can:

- set up the board database for analysis.
- define the target impedance.
- perform single-node analysis to validate and refine your capacitor selection.
- perform multi-node analysis to refine your placement.

EMControl

Systems can adversely impact each other due to electromagnetic interference (EMI), or due to unwanted coupling of energy between conductors, components, and systems.

Electromagnetic compatibility (EMC) is the ability of electronic systems to function as expected within their intended environment without adversely affecting other systems.

EMControl allows you to detect and resolve EMC problems early in the design cycle by enabling you to repeatedly check your design against selected sets of rules. EMControl includes several default rule sets. You can also write your own rules to verify specific design, environment, and regulatory requirements. Running EMControl early in the design cycle often helps to detect potential EMC problems before they can significantly impact product development.

Using EMControl you can:

- identify and setup critical EMC components, nets, and regions on your board.
- select the EMC rules to be checked.
- execute rule checking and view results and reports.
- cross-probe EMC violations.

[EMControl User Guide.](#)

Device Modeling Language (DML)

DML is a Cadence proprietary device modeling language. All device and buffer models are stored in DML format. You can use the DML format for many model types, such as IBIS and SPICE. DML also includes a behavioral modeling syntax and extensions to the SPICE syntax. DML files are ASCII files and may contain one or more SPICE-like sub-circuits.

The following is a simple example of DML syntax – a bi-directional S model for a single inductor with a value of 15 nH:

```
(PackagedDevice
  ("inductor15nH"
    (PinConnections
      (1 2 )
      (2 1 ) )
    (ESpice
      ".subckt inductor15nH 1 2
      R1 1 2 1e-6 L=1.5e-8
      .ends inductor15nH") ) )
```

For further details, refer to the [Allegro SI Device Modeling Language User Guide](#).

Allegro Design Entry HDL High Speed Option

The number of constrained nets on a typical high-speed board design has jumped from 25 to 75 percent (or more) of the design's total nets. The role of SI engineers within a design team is to analyze these nets. However, this job is growing rapidly as the number of nets on a board that require analysis (and the complexities of new chip sets) increase dramatically.

The design methodology supported by Allegro Design Entry HDL High Speed option improves design team productivity and minimizes the impact on overloaded SI engineering resources. It controls added costs by enabling electrical engineers to develop and manage constraints on their designs without having to depend on SI engineers to analyze all of the constrained nets.

Today, engineering teams must identify constrained nets and divide them in to two groups: those that are imperative to the design cycle and need to be verified quickly by SI engineers, and those that are not critical, and are not verified. This practice has often meant that these less-critical nets were either over constrained to assure a functional design or not managed at all, driving up the cost of the board. The risk of board failure increases when critical nets are managed in this manner, forcing potentially avoidable and expensive re-spins. Design Entry HDL High Speed option enables electrical engineers to determine optimal constraints for those not-so-critical nets at the front end, while SI engineers are able to focus on new chip sets and very critical nets, saving time and money.

The High-Speed Design Flows

Understanding the Flows

There are currently two different design flows used in Allegro high-speed PCB design.

- The PCB SI flow for high-speed printed circuit board design.
- The PCB MGH (Multi-GigaHertz) flow for high-speed serial data link design.
See the [MGH Flow Overview](#) on page 44 for further details.

PCB SI Flow Overview

The PCB SI flow can include up to eight different phases.

- Model Development and Verification
- Pre-Route Constraints Development
- Critical Component Pre-placement
- Solution Space Analysis
- Constraint-Driven Placement
- Constraint-Driven Routing
- Post-Route Design Rule Checking (DRC)
- Post-Route Verification

Note: The actual flow phases you use are largely determined by your corporate PCB design process and the characteristics of your design.

PCB SI Flow – Model Development and Verification

Procuring, developing, and verifying simulation models up front in the high-speed flows is crucial to the success of your design. Today's models come in many different styles and formats. Allegro SI DML (Device Modeling Language) enables you to accurately describe all devices and advanced behaviors.

A DML model refers to a single specific entity. That entity can be a package model, an interconnect model, an Espice model, or a translated IBIS model. It should be noted that an IBIS model can contain a package model within one translated file.

A DML file contains one or more models written in the DML language and is identified by its `.dml` extension. These model files are used in circuit simulation by analysis tools such as PCB SI and SigXplorer. Models are procured or developed in advance of simulation and used to characterize manufactured components such as ICs, discrete components, and connectors. The Allegro SI simulator requires that simulation models be in DML format for successful simulation. For further details on DML, refer to the [Allegro SI Device Modeling Language User Guide](#).

You can use Model Integrity to streamline the model development process. Model Integrity offers extended functionality and a unique environment that lets you create, edit, and test DML model files. You can also translate 3rd party model files to DML format.

For further details on model development, see [Model Development and Management](#) on page 49.

PCB SI Flow – Pre-Route Constraints Development

In order to shorten the design cycle and improve quality and performance, electrical and SI engineers must work concurrently to develop and effectively manage logic constraints in the early stages of the high-speed design flow.

Often times, engineering teams identify constrained nets by dividing them into two groups:

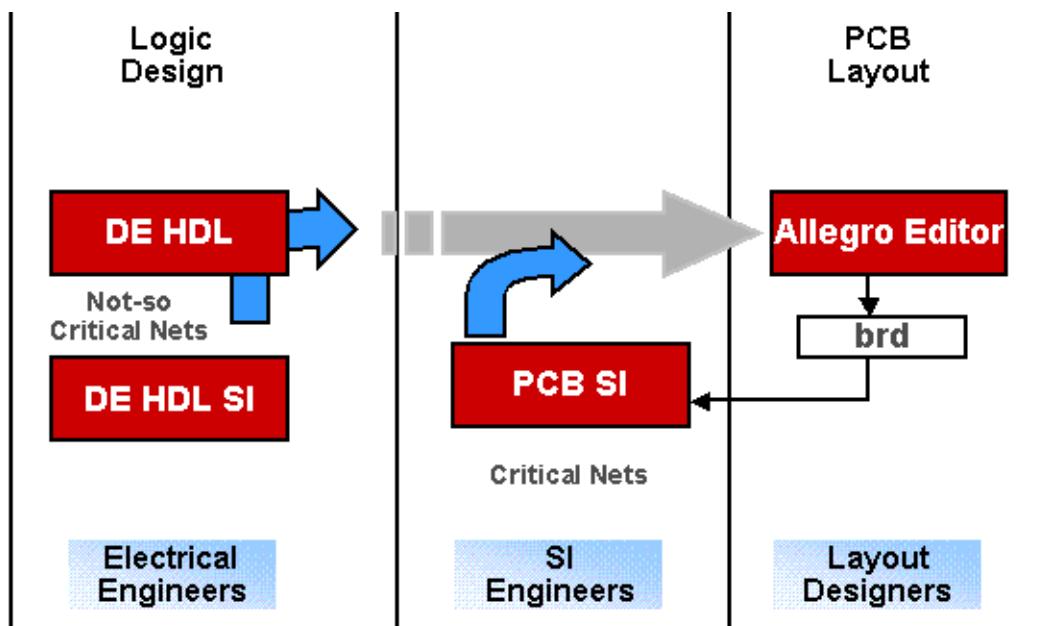
- those that are imperative to the design cycle and need to be verified quickly by SI engineers.
- those that are not critical, and are not verified.

Using this practice means that these less-critical nets are either over constrained to assure a functional design or not managed at all, driving up the cost of the board. The risk of board failure increases when non-critical nets are managed in this manner, forcing expensive re-spins that are avoidable.

Using Allegro Design Entry HDL SI during this phase enables electrical engineers to determine optimal constraints for those not-so-critical nets at the front end. At the same time, SI engineers are able to focus more on new chip sets and very critical nets using PCB SI, saving time and money. Design Entry HDL SI is a separately licensed product.

For details on using DE HDL SI to perform pre-route constraints development, see [Pre-route Constraints Development](#) on page 53.

Figure 2-1 Pre-route Constraints Development Flow



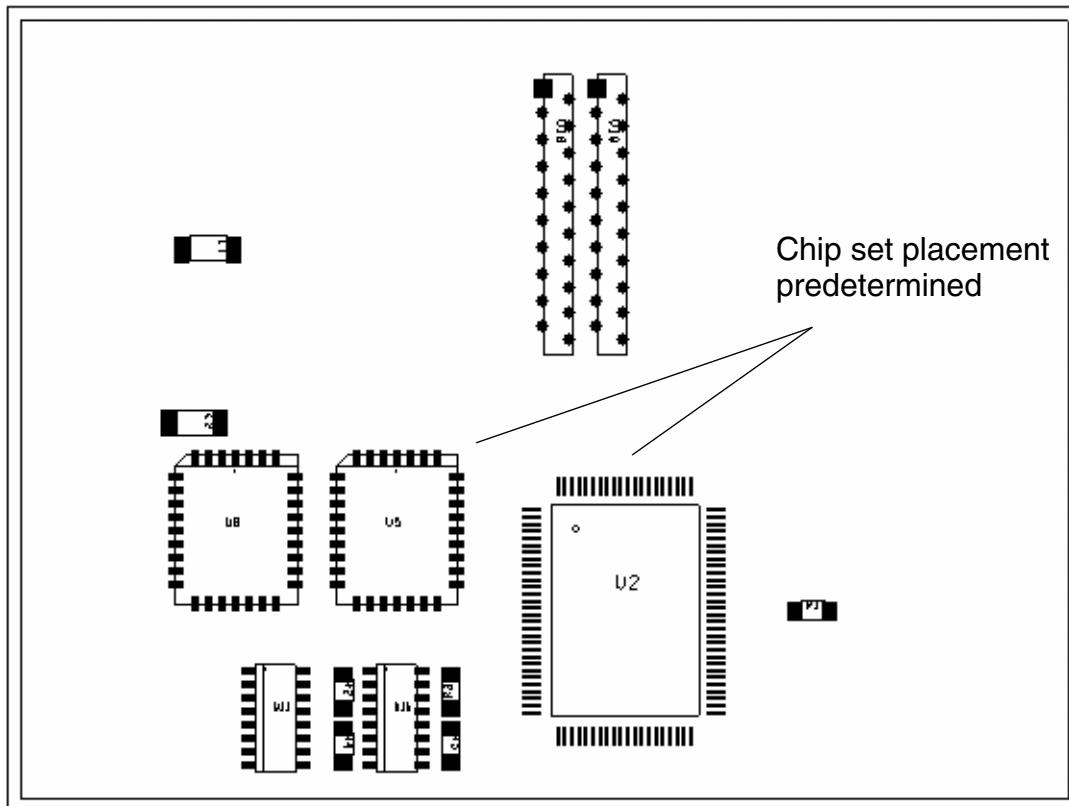
PCB SI Flow – Critical Component Pre-placement

The placement of key components and connectors is predetermined in many system designs. This is commonly seen in PC designs where the cabinet style often determines where you locate the processor, memory, and PCI / ISA slots.

Pre-placement data is a useful starting point for solution space analysis because high-speed signals often involve these components. Once these components are pre-placed and your board properly set up, you can extract topologies for critical signals into SigXplorer to begin the next phase of the high-speed design flow.

For details on performing critical component pre-placement, see [Getting Started with the PCB SI Flow](#) on page 49.

Figure 2-2 Component Pre-placement



PCB SI Flow – Solution Space Analysis

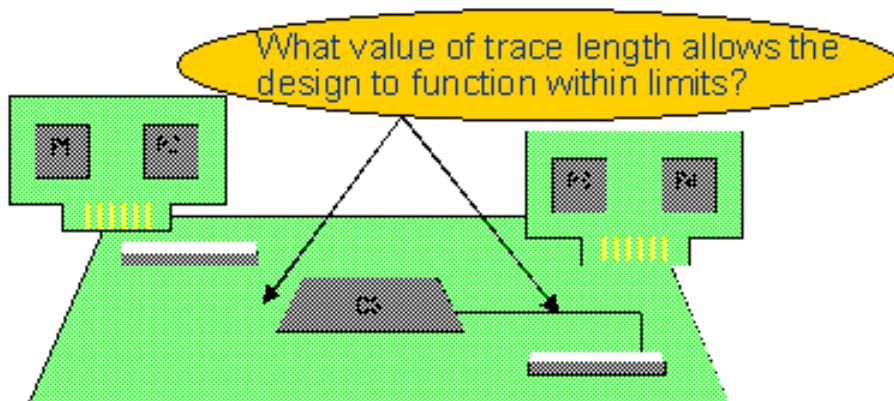
The goal of solution space analysis is to define a set of pin ordering and routing rules (topology templates) that allow the design to operate reliably. In order to begin this phase of the flow - you must set up your board properly. For further details on setup, see [Setting up the Design](#) on page 61.

You use SigXplorer to perform solution space analysis by sweeping all possible combinations of conditions under which your design must operate. These conditions may include:

- Manufacturing Variances
 - component speed
 - trace impedance
 - terminator value
- Design Variances
 - segment lengths

For details on performing solution space analysis, see [Chapter 7, “Solution Space Analysis”](#).

Figure 2-3 Signal Exploration



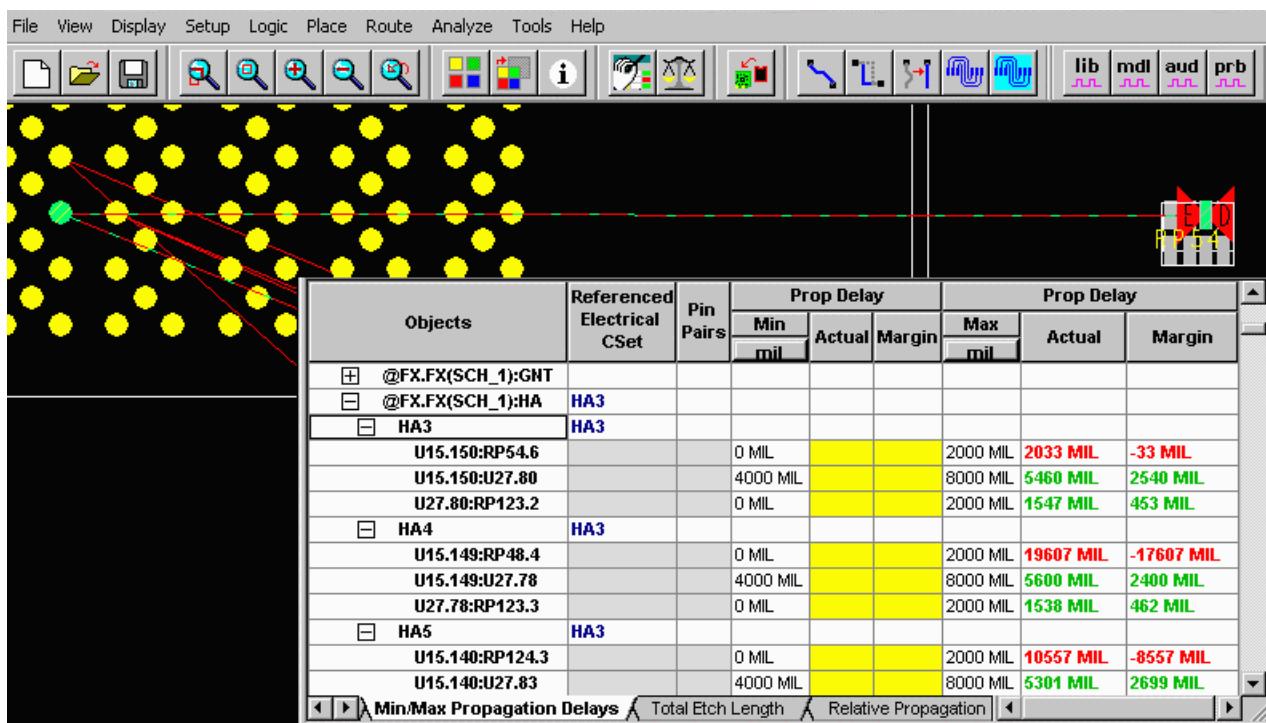
PCB SI Flow – Constraint-Driven Placement

Constraint-driven placement uses established rules to guide the process of defining locations for the remaining components. The design rules are derived from the solution space analysis phase and used to guide the placement process.

Constraint Manager plays a key role in guiding and evaluating component placement. Note in the following figure how the Margin column shows length over and above the manhattan connection distance available for routing. This provides fast feedback on routability. Zero or negative margin (in red) is undesirable.

For further details on constraint-driven placement, see [Appendix A, “Constraint-Driven Layout”](#).

Figure 2-4 Constraint-Driven Placement



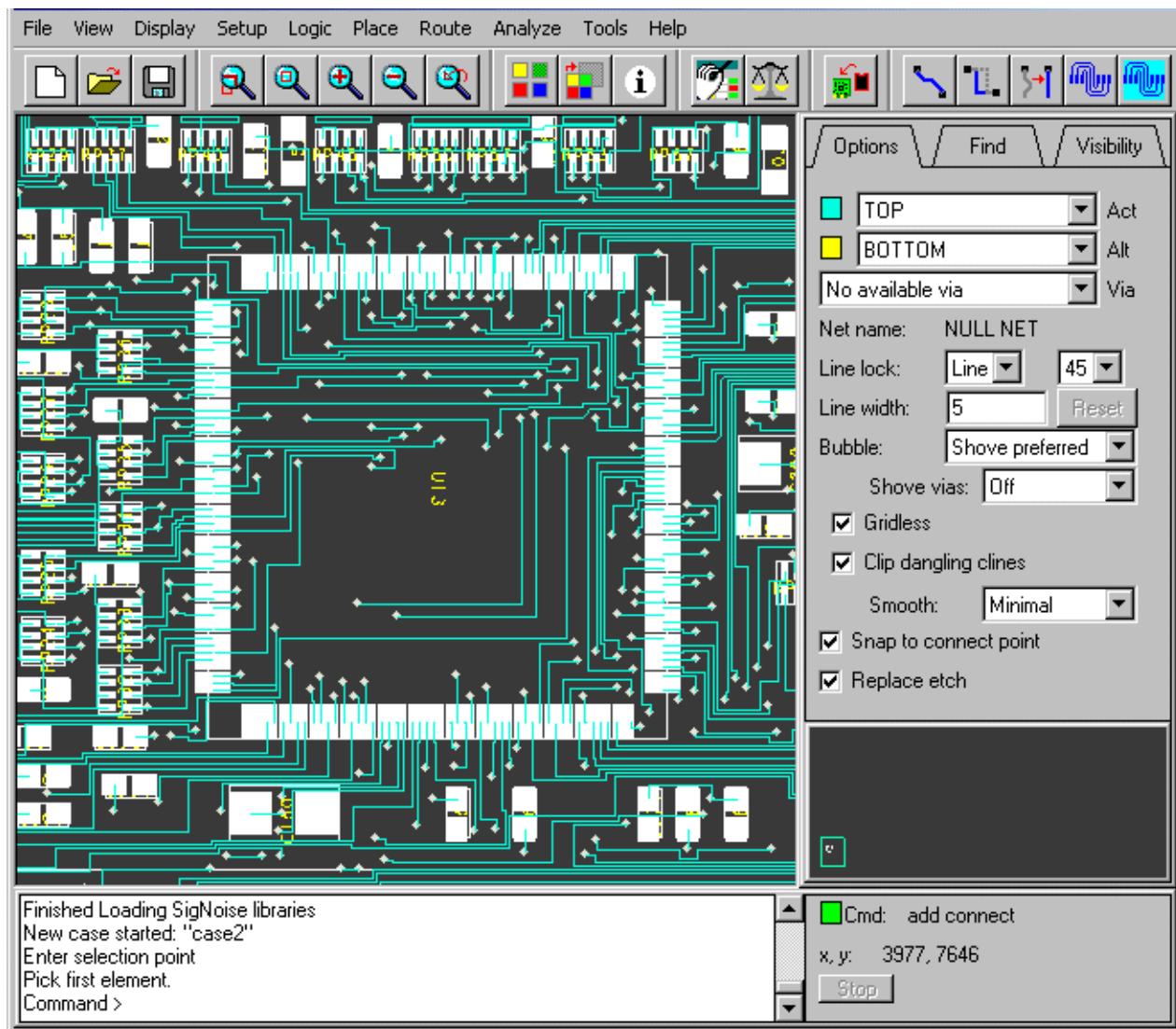
PCB SI Flow – Constraint-Driven Routing

Constraint-driven routing uses established design rules to drive the automatic and interactive routing process. PCB Router adheres to those design rules. Design rule violations during interactive routing are identified in real time.

Once you define a set of conditions under which the nets are known to work in the solution space analysis phase, chance of first-pass routing success is high. You can rip up and re-route nets, as long as they still adhere to the design rules.

For further details on constraint-driven routing, see Appendix A, “[Constraint-Driven Layout](#)”.

Figure 2-5 Constraint-Driven Routing

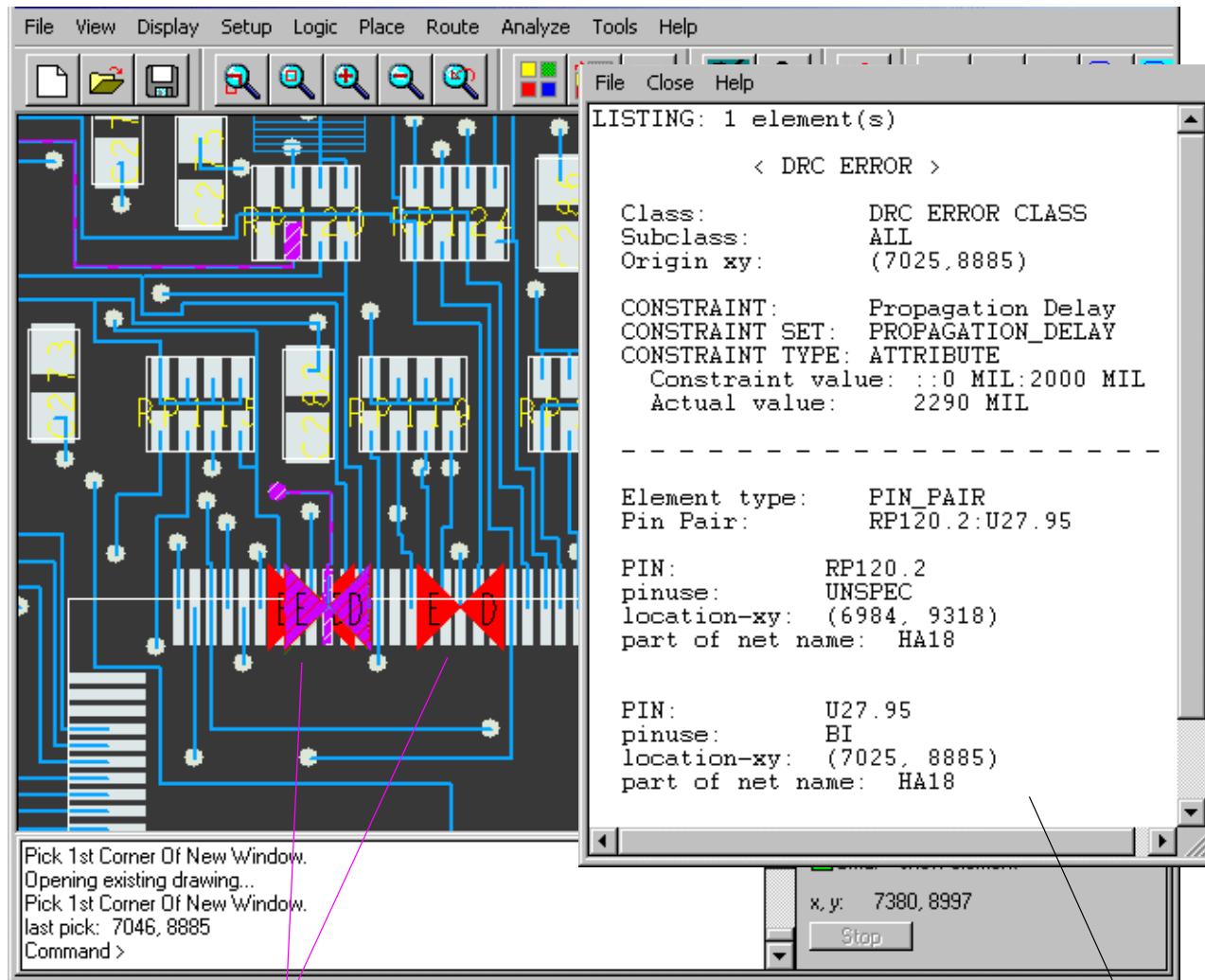


PCB SI Flow – Post-Route Design Rule Checking

DRC checks identify areas that do not comply with design rules. Nets are marked visually to identify the constraints that were violated. DRC provides a *first pass* check that is faster than simulation. You can also apply design rules, without ripping up etch, to pinpoint problems in boards routed before design rules were available.

For further details on performing post-route design rule checking, refer to the *Allegro PCB and Package User Guide*.

Figure 2-6 Post-Route Design Rule Checking



DRC markers quickly identify areas where routing violates physical or electrical rules.

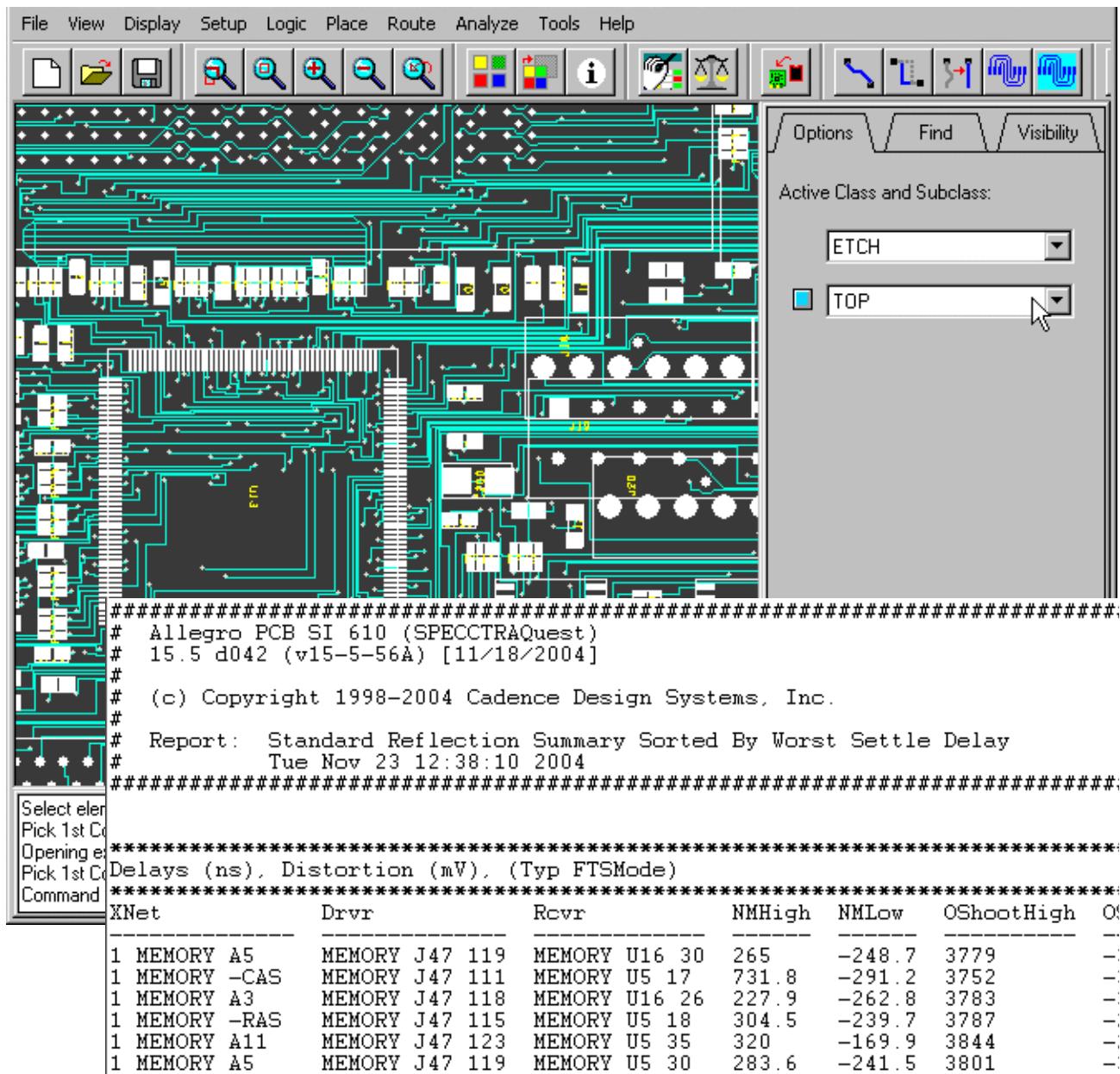
Description of the error

PCB SI Flow – Post-Route Verification

Post-route verification is a signal integrity *sign off* process. Chances of first-time success are high if a thorough solution space analysis has been performed. You can extract nets individually into SigXplorer and analyze them in-depth if problems are found.

See [Chapter 8, “Post-Route Verification,”](#) for further details.

Figure 2-7 Post-Route Analysis



MGH Flow Overview

The PCB SI MGH flow for designing high-speed serial links comprises five different phases.

- System-Level Design
- Block-Level Interconnect Design
- System-Level Interconnect Evaluation
- Channel Analysis and Pre-emphasis Optimization
- Detailed Time Domain Verification

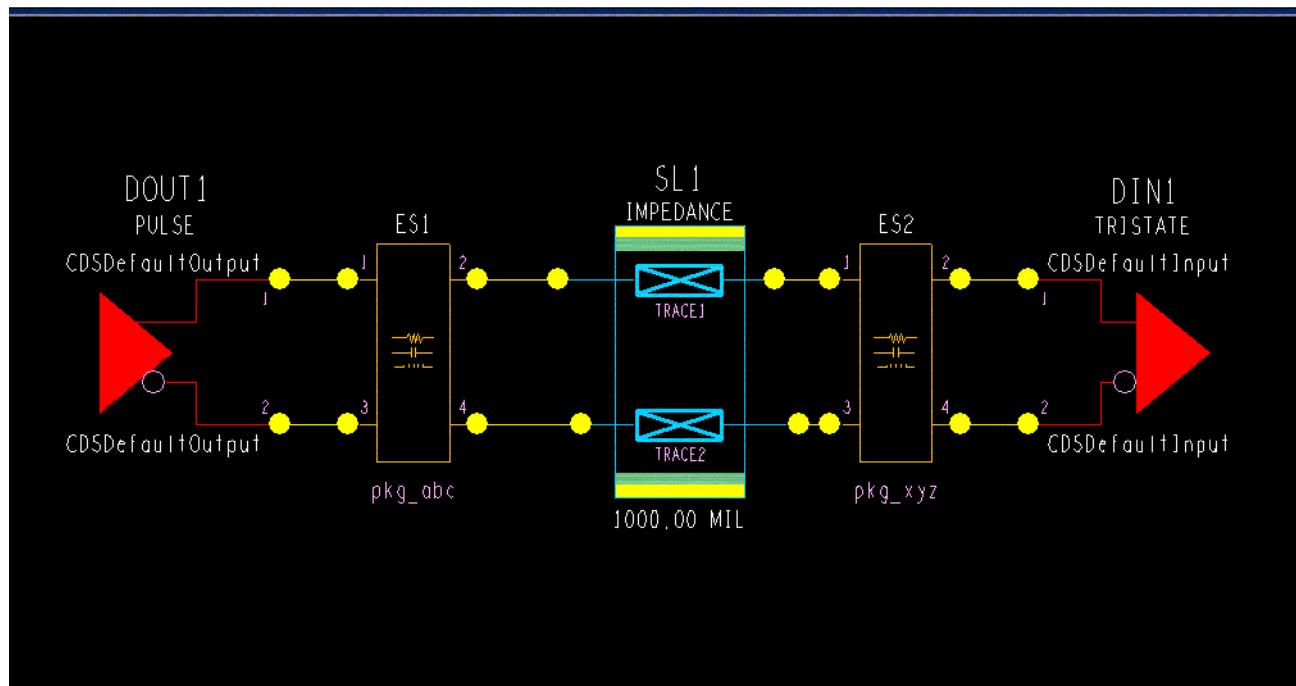
The general approach is to use frequency domain analysis (FDA) as the primary design technique, and the more computationally expensive time domain analysis (TDA) as the verification technique. To help address the massive TDA simulations that need run in order to predict bit error rate (BER); accelerated proprietary TDA techniques are required.

For further details on designing high-speed serial links, see [Appendix E, “Working with Multi-GigaHertz Interconnect”](#)

MGH Flow – System-level Design

The system-level design of a serial data link begins with planning out the overall data path in SigXplorer. This represents a mock-up of the entire data channel. The following figure shows an example for a very simple on-card (no backplane, no via) case.

Figure 2-8 Serial data channel in SigXplorer



The package model shown in Figure 2-8 could contain detailed SPICE sub circuits, or it could be shown as actual trace and via models.

You model the characteristics of the transmitter output and the requirements of the receiver, place them on the SigXplorer canvas, and define a system-level loss and jitter specification. This defines the maximum amount of overall loss (in dB) and jitter (in ps) that can be tolerated through the channel at the receiver. From this system-level spec, a loss budget is partitioned out to the various interconnect blocks that make up the channel. Jitter needs to be verified later at the system level.

MGH Flow – Block-Level Interconnect Design

Using the loss budgeting determined in the previous phase, interconnect design can be accomplished for the various blocks. This is close to the traditional PCB design we tend to think of. For example, one of the blocks may be a PCB that was budgeted a max loss spec of

4dB over a certain frequency range. The design engineer (DE) or SI Engineer would be tasked with designing the physical geometries for the serial data differential pair such that it would meet the block-level loss spec. Some of the parameters involved here are:

- PCB material and stack-up
- trace thickness
- differential line width and spacing (and other differential pair parameters)
- impedance
- via geometry
- selection of routing layers
- spacing to other traces

A detailed topology for the proposed PCB-level interconnect is built in SigXplorer. S-parameter generation takes place and the result is displayed directly in SigWave. You can do this multiple times and overlay several waveforms in SigWave to compare results.

If the spec is not met, then you need to go back to the original topology circuit, modify existing circuit parameters, and repeat the process until the loss budget is met. In cases where this is not possible, you may need to re-budget the system accordingly, allocating additional budget to the problematic PCB and tightening in other areas.

When the block-level spec is met then the final topology is stored in the library. Topology templates are then generated to provide wiring rules for physical layout.

MGH Flow – System-Level Interconnect Evaluation

When the previous phase has been repeated for the multiple blocks in the channel, and the associated loss budgets have been met, you append the multiple blocks from the library into SigXplorer, and evaluate the entire interconnect path in the frequency domain. The results of this analysis are evaluated against the system-level loss budget.

If the budget is not met, there are a number of things that you may want to do:

- Plot the frequency responses of the multiple blocks as an overlay of one another to gauge the relative losses and pinpoint which block is the main culprit of the non-compliance.
- Go back to the block-level and re-work a particular portion of the interconnect circuit before returning to the system-level.

When the system-level loss budget is eventually met, move on to the next phase.

MGH Flow – Channel Analysis and Pre-emphasis Optimization

When there is confidence that the frequency domain approach has optimized the interconnect to the degree possible, channel analysis is commenced. This consists of the following steps.

1. Open the detailed circuit in SigXplorer.

The interconnect can use S-parameter or circuit model format. Driver/receivers are either structural transistor-level models or behavioral macro models.

2. Run a quick test simulation in the time domain, making sure models compile.
3. Characterize the circuit.
4. Set up stimulus generation for the bit stream from the graphic user interface. You can pre-define stimulus generation from popular data formats or dynamically generate a long pseudo-random bit sequence (PRBS). It is possible to select multiple bit streams.
5. Run channel analysis, which produces the following outputs:
 - Eye pattern contours
 - Voltage and jitter distribution plots
 - Reports
6. If requirements are not met, you can modify items such as pre-emphasis what-if's, data rate, driver, receiver or topology; and then rerun the analysis.
7. When results are satisfactory for typical silicon, verify the silicon corner cases (process/voltage/temperature).

Once the requirements are met, move on to the next phase.

MGH Flow – Detailed Time Domain Verification

The pre-emphasis and equalization settings suggested in the previous flow phase are the optimal settings, which may or may not be directly achievable with existing silicon. For this reason, it is desirable to run full time domain circuit simulation in SigXplorer using the system-level S-parameter interconnect model, together with the detailed driver and receiver models. The driver and receiver models use actual settings in the models, staying as close as possible to the parameters recommended by the channel analysis described in the previous flow phase.

For post-route verification, if an actual routed PCB is available, you can extract directly from SI into SigXplorer, rather than using the system-level S-parameter model generated previously.

Once the circuit is built in SigXplorer, you should:

- define the stimulus to use through the graphic user interface.
- execute simulation.
- plot time domain results in SigWave.
- define the start time in the simulation for the eye diagram plots, allowing operating levels to stabilize.
- display eye diagrams in SigWave.
- possibly import channel analysis results and overlay for comparison purposes.

Getting Started with the PCB SI Flow

Model Development and Management

Developing and Testing Models using Model Integrity

Model Integrity streamlines the model development process by enabling you to:

- work with multiple open files for model editing efficiency (copy and paste between files).
- translate third-party model files to DML format.
- parse IBIS or DML files to check for syntax errors.
- view data curves associated with buffer models in SigWave.
- simulate buffer models to pre-check their behavior against I/O characteristics.
- qualify DML files for use in TLsim.

For step-by-step procedures on performing these tasks, refer to the [Model Integrity Command Reference](#). For further information about Model Integrity features and its user interface, refer to the [Model Integrity User Guide](#).

To start Model Integrity

- Choose *Tools - Model Integrity* from the PCB SI.
The Model Integrity window appears.

Managing and Maintaining Models using the Library and Model Browsers

The SI Model Browser available from within PCB SI and SigXplorer enables you to:

- create and manage libraries of device and interconnect models.
- specify which device and interconnect libraries you want SigNoise to access, as well as the order of library access.
- create model files on the fly and add them to the working library.
- edit and maintain model files.

For step-by-step procedures on performing these tasks, see [Chapter 3, “Model and Library Management.”](#)

To access the SI Model Browser

- Choose *Analyze – Model Browser* from the PCB SI.
- or -
- Choose *Analyze – Model Browser* from SigXplorer.

The SI Model Browser dialog box appears as shown in [Figure 3-1](#) on page 76.

Models and Simulation

During analysis, SigNoise develops simulation circuits using models of the devices and interconnect in your design. Prior to analysis, you must associate device models with the components in your design and point SigNoise to the device model libraries (where the device models are stored).

The simulation circuits are created on an as-needed basis by SigNoise in the Interconnect Description Language (IDL) and stored in the Interconnect Model Library that you specify. The stored models are used later to avoid repeating the field solution of the same physical interconnect configuration. You may examine the interconnect models and modify them. See [“Modeling in the Interconnect Description Language”](#) on page 487 for more information on IDL.

Device Models

The different types of device models let you choose between varying levels of detail to more accurately model and simulate your design. The following device models are available:

- IBIS device models that behaviorally model active devices.
These models list all the pins on a device, associate individual pins with IOCell models, and define power and ground pins. They also list the package parasitics associated with each pin and which power or ground bus each signal pin references. This information is used for Simultaneous Switching Noise (SSN) analysis. Additionally, IBIS device models can also define differential pair pairing.
- IBIS IOCell models that describe the behavior of an IO buffer (that is, drivers and receivers).

- IOCell models represent individual drivers and receivers for specific pins on a device. They contain behavioral information about an IO buffer, such as its voltage thresholds and I-V curves. You can assign default IOCell models so that any pin that doesn't have a component with a SIGNAL_MODEL property associated with it will match up with these defaults according to its pin use. You can also determine whether or not SigNoise will use default IOCell models.
- Package models that are matrices of resistance, inductance, conductance, and capacitance values for each conductor and between conductors in a package. Package models can optionally contain within them SPICE sub circuits for more detailed package modeling.
- Espice models that contain within them a SPICE description of a device. Espice device models are used for passive devices such as resistors. Espice models represent simple SPICE sub circuits and represent discrete components such as resistors and capacitors. For example, a 50 ohm resistor may be represented as follows:

```
.subckt resistor50 1 2  
R1 1 2 50  
.ends resistor50
```

The software can create Espice models automatically, based on device data setup for discrete components.

- System Configurations that describe a connection between multiple printed circuit boards (PCBs). System Configuration models are used in system-level simulation.
- Cable models that describe the parasitics between multiple PCBs. You use Cable models in multi-board systems where the Cable models typically represent cables and connectors.

Interconnect Models

During simulation, SigNoise automatically creates the interconnect models by field solving geometries and stores them in the interconnect model library that you specify. SigNoise writes the models for the interconnect in the Interconnect Design Language (IDL). You can also use IDL to model passive devices as you would simple SPICE sub circuits.

You do not have to route designs prior to simulation. The unrouted interconnect modeling information (a percent Manhattan distance between pins and user-defined assumptions for the characteristic impedance and propagation velocity) allows you to run pre-route

Allegro PCB SI User Guide

The High-Speed Design Flows

simulations from the rats nest information. Using the results of these simulations, you can evaluate items such as reflections, termination, and delays.

For routed connections, you can simulate using actual routed interconnect models. In these cases, the unrouted interconnect modeling information is ignored.

Pre-route Constraints Development

Design Entry HDL SI

DE HDL SI integrates the following Allegro tools, bringing PCB SI technology to the Electrical Engineer's desktop.

- Allegro Design Entry HDL
- SigXplorer
- SigWave
- Allegro Constraint Manager
- Model Integrity

Use Model

The general sequence of events for using DE HDL SI is as follows:

1. Launch DE HDL SI and open your schematic.
2. Assign device models to certain non-critical nets and Xnets in the schematic.
3. Launch Constraint Manager from DE HDL SI to check current net constraints in worksheet (based on assigned device models).
4. Optional: Launch Model Integrity and edit device model syntax.
5. Optional: Check Constraint Manager to verify constraint changes.
6. Launch SigXplorer from Constraint Manager to extract the net topology.
7. View, edit, and simulate the net topology in the SigXplorer canvas.

Note: You can append other *canned* topologies (added into the canvas as a test harness) by wiring them to the net topology before you simulate. Simulation sweeps are also possible to help determine optimum solution space.

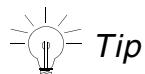
8. Analyze the simulation waveforms in SigWave and the simulation data in the SigXplorer Results spreadsheet.
9. If necessary, adjust topology constraints in SigXplorer (*Set – Constraints*) based on results.
10. Repeat the last three steps as required until net constraints are optimized and a desired solution space is achieved.

Allegro PCB SI User Guide

The High-Speed Design Flows

11. Update Constraint Manager (*File – Update Constraint Manager*) with the final constraints for the net.

Critical Component Pre-Placement



Tip

If you have a PCB design with critical components already pre-placed, skip this task and proceed with setting your board up for solution space analysis. Proceed to [Setting up the Design](#) on page 61.

The *Place – Manually* command in PCB SI supports timing-guided placement of critical components driven by previously defined delay rules. Delay rule violations are immediately highlighted with visual markers that provide instant feedback. This feedback makes immediate compensation possible when components are being placed. Components must already be defined.

Note: In cases where an unplaced component is not defined in the package library, a confirmmer asks whether you want to create a new (temporary) package symbol.

To place critical components on your board

1. Choose *Place – Manually* from the PCB SI menu bar.

The Placement dialog box appears as shown in [Figure 2-9](#) on page 56.

2. Make sure the *Type filters* field is set to *Any* and the Advanced Settings tab controls are set accordingly so that all components are displayed.
3. Choose the component names you want to place.

Note: Only unplaced components are listed.

The first component that you select is attached to the cursor by its symbol origin. The symbol reflects the rotated and mirrored position specified on the *Options* tab. Rubberbanding ratsnest lines indicate the connections of the component with any components already placed.

4. If necessary, right-click to change the orientation of the component using the *Rotate* and *Mirror* options on the context-sensitive menu.

See [Rotating During Placement](#) and [Mirroring During Placement](#) on page 57 for details.

5. Choose a location for the component by doing one of the following:

- a. Place the component at the desired location, then click to place it in the design.
- b. Type the X, Y coordinates of the component location on the command line.

Specify the coordinates as follows:

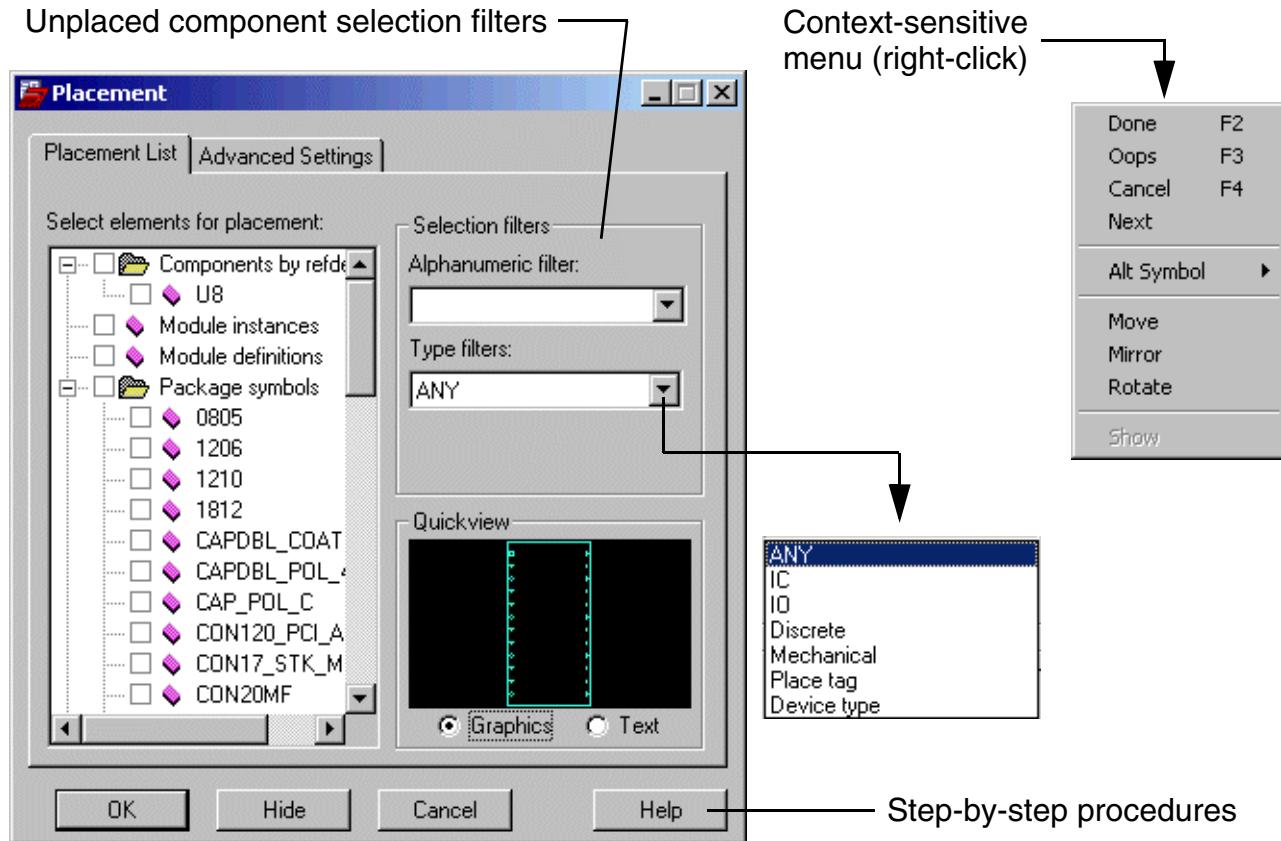
x <x coordinate> <y coordinate>

An example of specifying coordinates is: x 450 3600

As soon as you place the component, a P indicating *placed*, is superimposed on the icon for the component in the *Select elements for placement* list.

6. If *AutoNext* is enabled in the *Advanced Settings* tab and you have checked more than one component (or the entire group), continue placing components until all your selections have been placed.
7. When you are finished placing components, choose *Done* from the context-sensitive menu.

Figure 2-9 The Placement Dialog Box and Related Menus



Rotating During Placement

The *Rotate* option in the context-sensitive menu enables you to rotate the component attached to the cursor until the correct rotation is shown before you position it in the design.

You can rotate the component by any angle increment, in either a clockwise or a counter-clockwise direction.

To rotate the component attached to the cursor during a placement operation

1. Right-click to display the Placement context-sensitive menu.
2. Choose the *Rotate* option.

A line appears connecting the cursor to the component allowing you to control rotation.

Note: If required, change the angle increment at which rotation occurs in the *Angle* field on the *Options* tab of the control panel.

3. Move the cursor in a clockwise or counter-clockwise direction to rotate the component by the increment specified in the *Angle* field.
4. When the design element is at the required angle, click.

The rotated symbol reappears attached to the cursor. You can now continue placing the component.

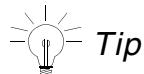
Mirroring During Placement

The *Mirror* option in the context-sensitive menu enables you to mirror the component attached to the cursor until the correct orientation is shown before you position it in the design.

Note: If the *Mirror* option is selected on the *Options* tab, placement automatically uses a component in its mirrored state.

Logic

You derive logic for your design by importing a netlist from either a Cadence or a third-party source. Additionally, you can create a netlist from scratch within PCB SI. See [Chapter 5, “Defining Logic”](#) for further details.

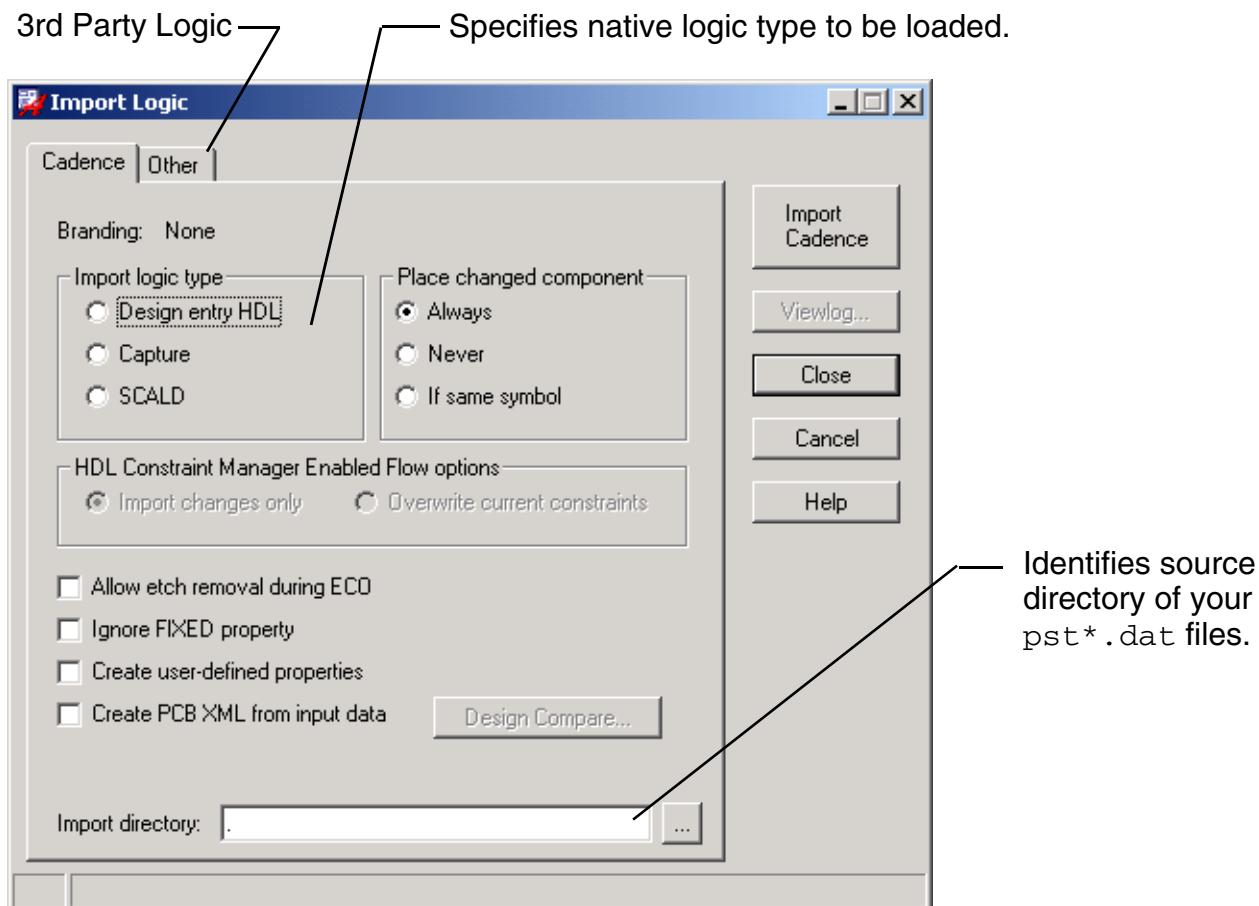


If you have received a PCB design with a netlist already resident in the database, skip this task and proceed directly to [Setting up the Design](#) on page 61

Importing Logic

To import a netlist into your design, choose *File – Import – Logic*. The Import Logic dialog box appears as shown in the following figure.

Figure 2-10 Import Logic Dialog Box



Allegro PCB SI User Guide

The High-Speed Design Flows

Use this dialog box to load the logic for your design into the design's database and establish the operating characteristics for the netrev utility. Logic is derived natively (that is, from a Cadence source) or from a third-party netlist. Choose the appropriate tab to set the parameters for loading logic into your design.

Design Audit and Setup

SI tools require tool-specific information that must be available for the tool to function correctly. You must provide the correct input to PCB SI before you can extract signals into SigXplorer, derive topology templates, and drive them back into your layout.

The *SI Design Audit* and *SI Design Setup* commands provide wizards that walk you through the steps required to set up the tool and perform audit on the designs.

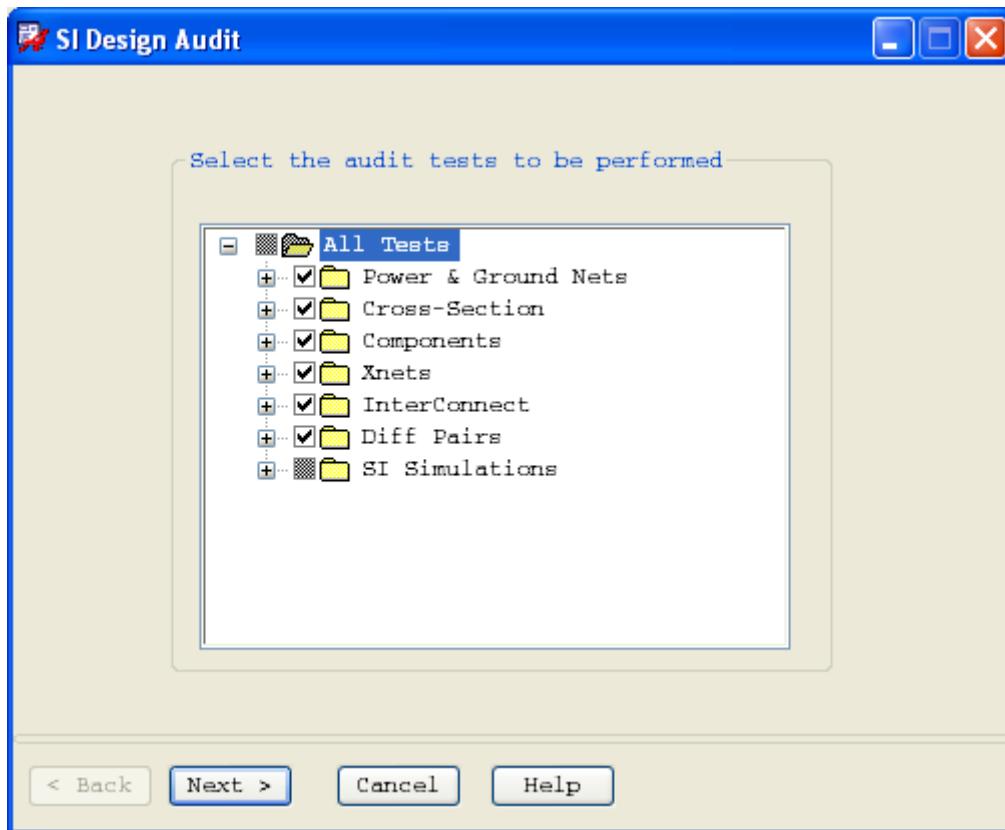
Auditing a Design

The SI Design Audit command runs an audit on all or selected nets in a design. The wizard helps you audit specific nets in the layout to verify that they are set up properly for extraction and simulation.

To invoke the SI Design Audit wizard,

- Choose *Setup – SI Design Audit*.
OR
- Run *signal audit* in the console window.
OR
- Choose *Tools – Utilities – Keyboard Commands* and choose *signal audit* in Command Browser.
OR
- Right-click the canvas and choose *SI Design Audit* from the pop-up menu, provided the Signal Integrity application mode is on.

The SI Design Audit wizard is displayed.



In this wizard you can perform an audit on selected nets and Xnets and check for any missing models. A report is displayed for that net indicating the current status. The SI Design Audit wizard walks you through the steps to:

- Control which tests are to be performed
- Select the Xnets and nets to be audited
- Detect and resolve the errors encountered

For information on auditing nets, see [Allegro PCB and Physical Layout Command Reference: S Commands](#).

Setting up the Design

The existing Setup Advisor utility is replaced with the new *SI Design Setup* command. This command launches the Setup Category Selection wizard, which helps you set up the design to perform SI simulations. The *SI Design Setup* command *assists you in making your*

board ready to run high-speed analysis. It simplifies the setup by guiding you through the required steps.

To invoke the Setup Category Selection wizard:

- Choose *Setup – SI Design Setup*.

You can also right-click the canvas and choose SI Design Setup from the pop-up menu.

The Setup Category Selection wizard is displayed. You can perform the following design setup tasks using the wizard:

- Selecting Setup Categories
- Selecting Xnets and Nets to Setup
- Setting Up Search Directories and File Extensions
 - Library Search Directories
 - Library File Extensions
 - Working Libraries
- Setup Power and Ground Nets
- Setup Design Cross-Section
- Setup Component Classes
- Assign Models to Components
- Setup Diff Pairs
- Setup SI Simulations
- Setup Complete

For information on design setup, see [Allegro PCB and Physical Layout Command Reference: S Commands](#).

Managing Models and Libraries

Use the Signal Model Assignment dialog to assign models and Signal Model Browser to manage model libraries.

Assigning Models

The simulator uses device models to create circuit simulation models for the nets in your design. This means that you must assign a device model to each component that you simulate.

Use the Signal Model Assignment dialog box to assign signal models to design components. You can use the *Auto Setup* option in the Signal Model Assignment dialog box for all 2-pin components with a VALUE property and no previous model assignment.

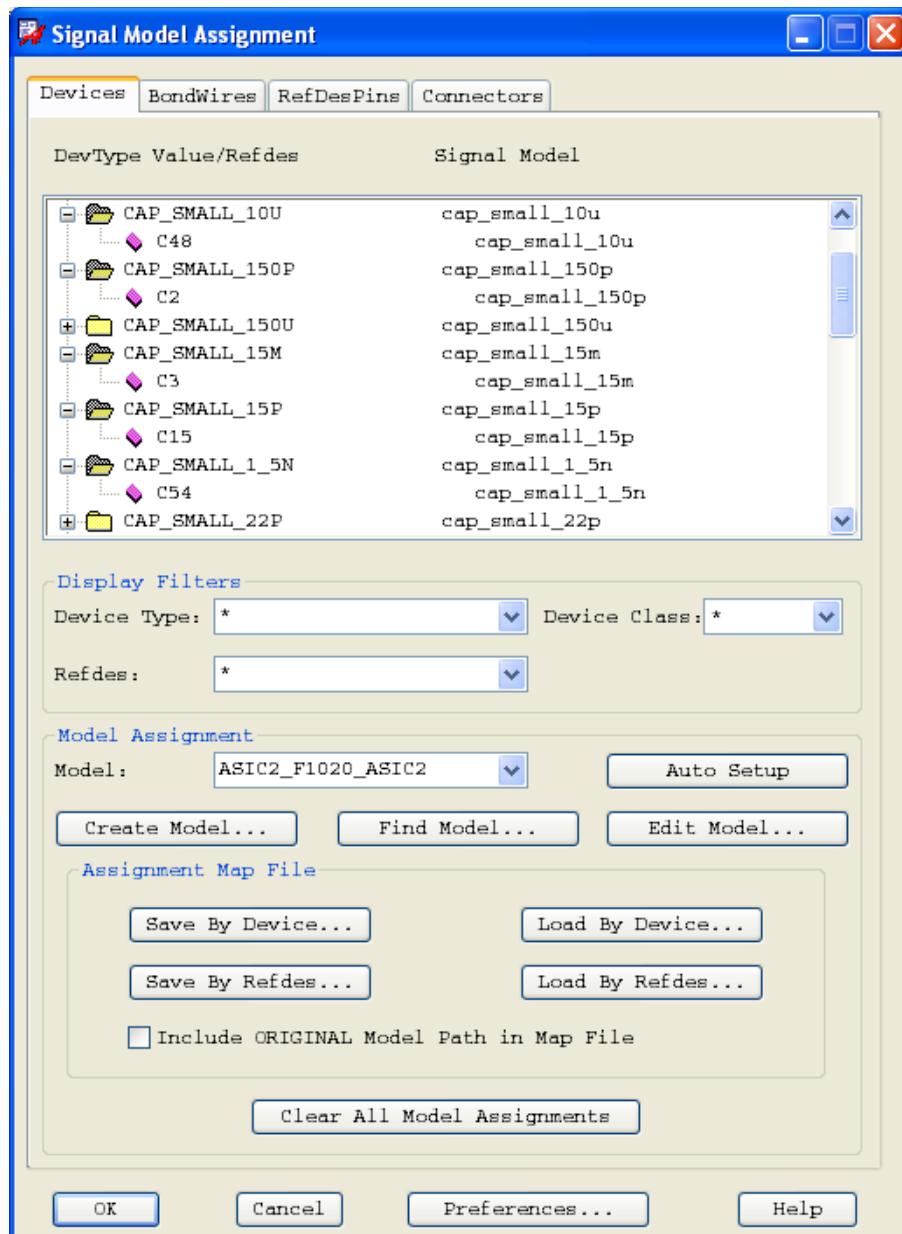
The simulator looks first in open libraries for existing model names, using the device type prefix as a reference. If no model is found, the simulator creates a new model in the working device library and names the model after the device type (with an underscore replacing each non-alphanumeric character).

The following table describes the signal model creation schema.

Table 2-1 Signal Model Creation Schema

When the . . .	this model is created.
component VALUE property is > 1.0	Resistor
component VALUE property is < .001	Capacitor
component reference designator starts with L	Inductor
component reference designator starts with C	Capacitor
component reference designator starts with R	Resistor
<hr/>	
For all other reference designators. - and -	Resistor
For all other VALUE property values.	

Figure 2-11 Signal Model Assignment Dialog Box



When you finish edits to model assignments, a report is displayed indicating the changes.

Devices Tab

Use the *Devices* tab to assign device models to components; automatically or manually. You can access the Model Browser to find device models, modify existing models before

assigning them, and create new models. You can also load and save the Assignment Mapping file for the design.

Automatic Model Assignment

During automatic model assignment, the simulator attempts to assign models to all two-pin components having VALUE property and no previous model assignment.

The simulator looks first in open libraries for existing model names, using the device type prefix as a reference. If no model is found, the simulator creates a new model in the working device library and names the model after the device type (using underscores to replace each non-alphanumeric character). Models are created using the signal model creation schema. See [Table 2-1](#) on page 63 for details.

BondWires Tab

Use the *BondWires* tab to locate and assign trace models to bondwire connections. You can also modify trace models using the Model Browser.

RefDesPins Tab

Use the *RefDesPins* tab to assign IOCell models to specific pins. You can also assign models to pins that have a selection of programmable buffer models.

Connectors Tab

Use the *Connectors* tab to assign coupled connector models to components such as male/female connectors, PCI slots, and other components that connect one design to another.

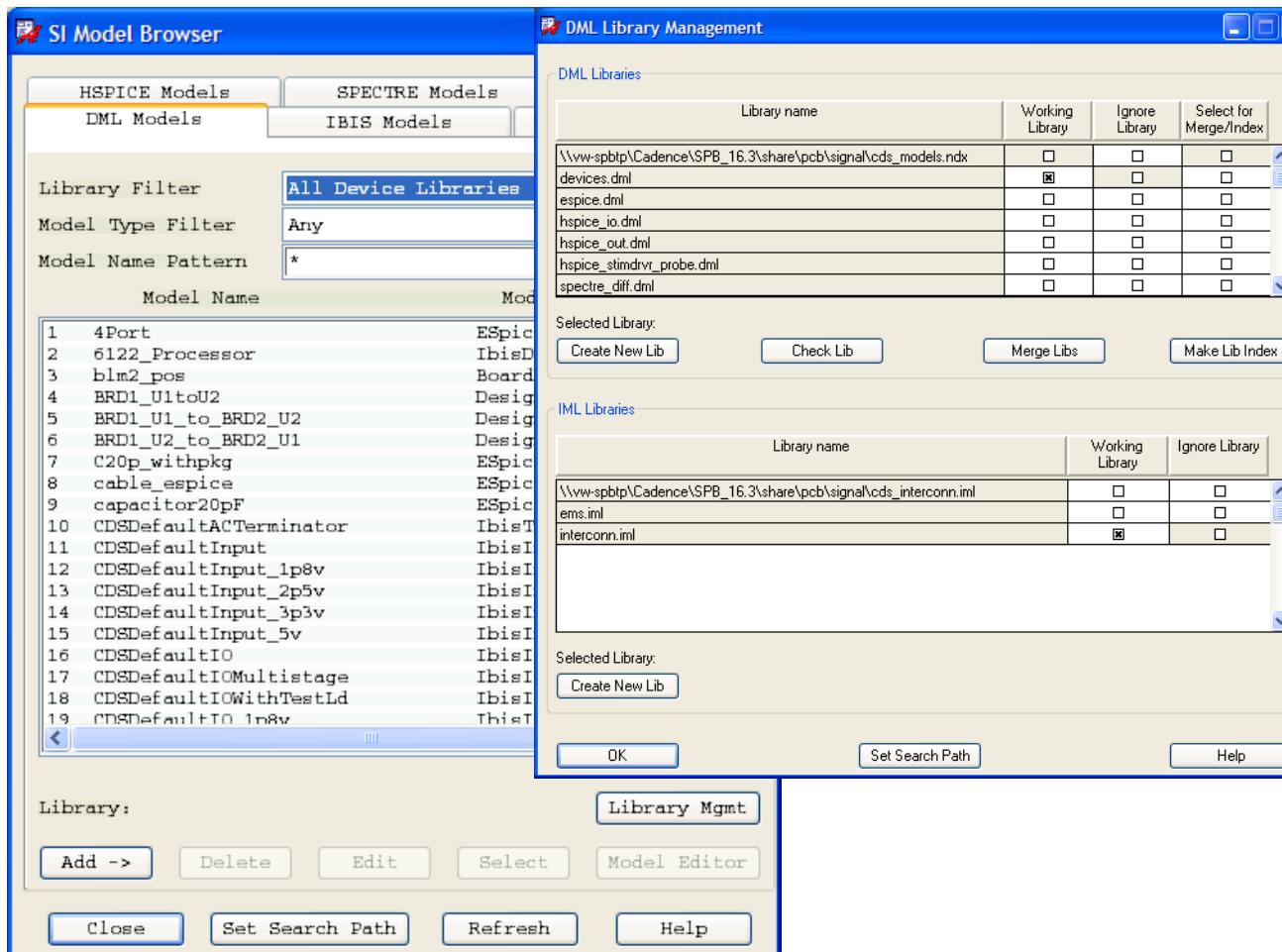
Managing Model Libraries

You manage device and interconnect model libraries by choosing *Analyze – Model Browser* from the PCB SI menu bar. The SI Model Browser appears as shown in the following figure.

Allegro PCB SI User Guide

The High-Speed Design Flows

Figure 2-12 SI Model Browser - DML Library Management



Unless you choose to simulate using the default device models, you must have your device model libraries loaded in the SI Model Browser so the simulator can access the models. Device and interconnect libraries can be located anywhere on the system as long as an absolute pathname is specified. SigNoise searches the libraries in the order they appear in the browser library lists (top to bottom).

Setting the Working Library

SigNoise stores new models in the current working libraries. To set a library as a working library:

1. Click *Library Mgmt* in the SI Model Browser dialog box.

DML Library Management dialog box displays. The name of the working library for device models and interconnect models is displayed in the *Working Library* column in the DML

Libraries or IML Libraries section, respectively. The default working library for device models is `devices.dml` while the default working library for interconnect models is `interconn.iml`.

2. Click the *Working Library option* in the DML Libraries or IML Libraries section.
3. Click *OK* to close the Device Library Management dialog box.

Translating and Adding Libraries

You can translate HSPICE, SPECTRE, IBIS, and SPICE models to the PCB SI device model library (DML) format in the SI Model Browser dialog box.

To translate a model:

1. Select the model in SI Model Browser.

As soon as you select a model, its complete path appears SI Model Browser.

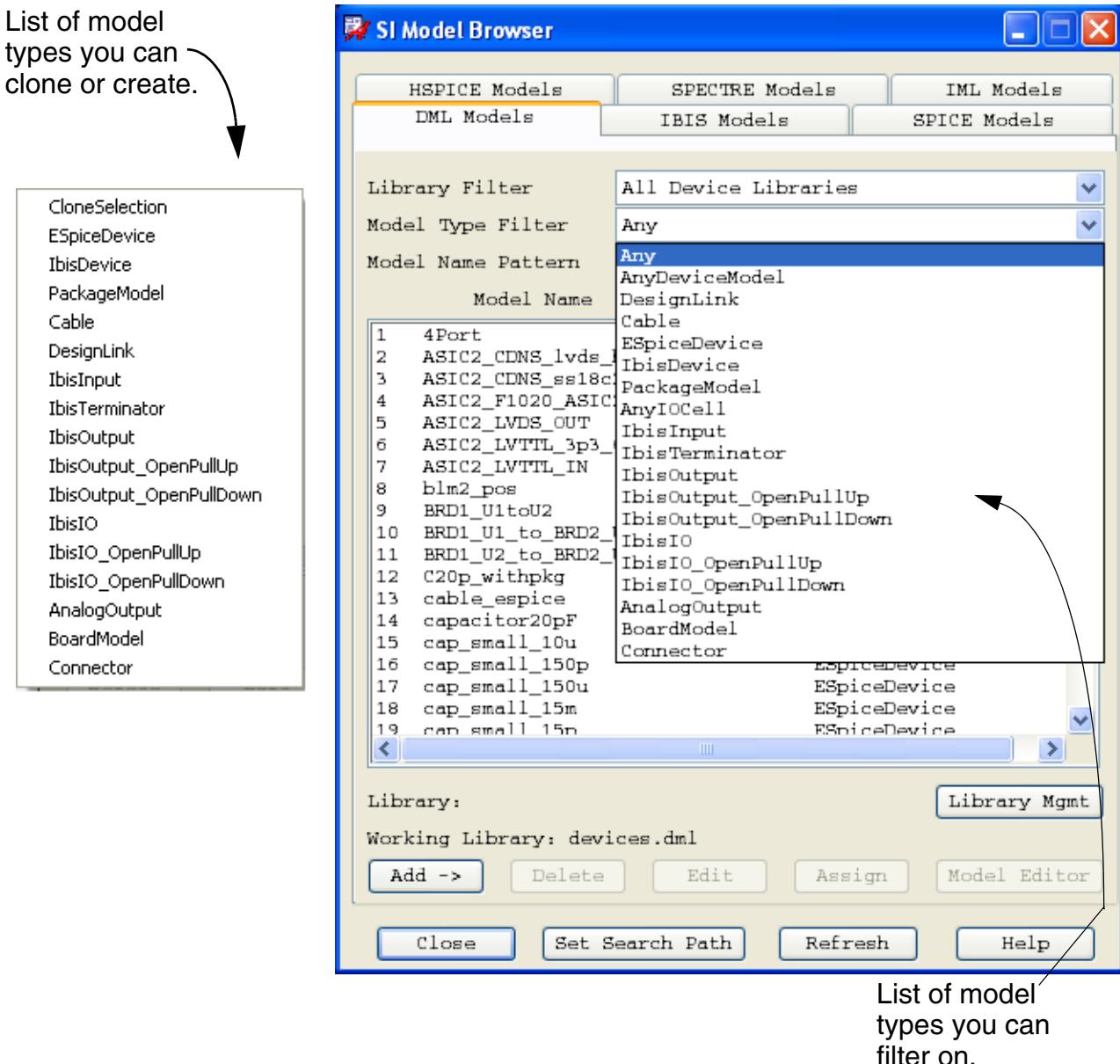
2. Click *Translate*.

PCB SI reads the original file and creates a `.dml` file using a name you assign. The new file is then filtered through a syntax checker, (`dmlcheck`) to check the syntax and integrity of library files. If no errors are detected, the output file is created and added to the list of DML files in the DML Models tabbed page.

Managing Models

You can browse and manage models in SI Model Browser, by applying the appropriate library or model type filter. All the models are displayed in SI Model Browser as shown in [Figure 2-13](#) on page 68.

Figure 2-13 SI Model Browser - DML Models



Use SI Model Browser to add, delete, edit, and list models in the device library.

Device models contain:

- a list of all the pins on the device.
- IOCell models
- power and ground pins.

- the package parasitics (R, L, G, C values) associated with the device pins
- the power and ground pins that each signal pin references (used for simultaneous switching noise).

Interconnect models contain:

- matrices of resistance, inductance, conductance, and capacitance values.
The PCB SI field solvers calculate and write these values into SPICE sub circuit models.

Table 2-2 Device Models Supported by Signal Integrity Tools

Model Type	Use and Relationship
<i>Design Link</i>	Used to specify system-level connectivity, like multi-board or advanced package-on-board scenarios.
<i>Cable</i>	Referenced from a system configuration. Models cables interconnecting multiple boards. Can be an RLGC model or SPICE sub-circuits.
<i>ESpiceDevice</i>	Assigned to discrete parts like resistors and capacitors. Contains SPICE sub-circuits.
<i>IbisDevice</i>	Assigned to ICs and connectors. An IbisDevice model for a connector has package parasitics but no IOCell models.
<i>PackageModel</i>	Referenced from an IbisDevice model. Models the package parasitics of the entire component package. Can be an RLGC matrix model or SPICE sub-circuits.
<i>AnyIOCell</i>	Referenced from an IBISDevice model. An IOCell model is used to model driver and receiver buffers at the pin level.
<i>IbisInput</i>	A type of IOCell model. It is a receiver model.
<i>IbisTerminator</i>	Models termination internal to the device pin
<i>IbisOutput</i>	A type of IOCell model. It is a driver model.
<i>IbisOutput_Open</i> <i>PullUp</i>	Driver model with no pullup resistor
<i>IbisOutput_Open</i> <i>PullDown</i>	Driver model with no pulldown resistor
<i>IbisIO</i>	Bidirectional buffer model, which can drive or receive

Model Type	Use and Relationship
<i>IbisIO_OpenPull Up</i>	Bidirectional buffer model with no pullup resistor
<i>IbisIO_OpenPull Down</i>	Bidirectional buffer model with no pulldown resistor
<i>AnalogOutput</i>	Models the behavior of an analog device pin
<i>BoardModel</i>	Referenced from a system configuration. Models entire boards for situations in which the physical Allegro database is not available. Contains SPICE sub circuits.

Table 2-3 Interconnect Models Supported by Signal Integrity Tools

Model Type	Use and Relationship
<i>Trace</i>	Geometry-based model that represents a single transmission line with no coupling. A Trace can have frequency-dependent loss.
Coupled Traces	Geometry-based model representing coupled lossy transmission lines.
<i>Any CPW</i>	Any model representing a coplanar waveguide (CPW) structure.
<i>Single CPW</i>	Represents a single CPW. It is a two-pin symbol containing no dielectrics.
<i>Diff Pair CPW</i>	Represents a differential pair CPW. It is a four-pin symbol containing no dielectrics.
<i>Any Via</i>	Models the parasitics of a via providing z-axis connectivity between traces.
Closed Form Via	<ul style="list-style-type: none"> ■ Fast Closed Form This is the most basic form of a via model. It is a static method, formula-generated model. ■ Detailed Closed Form A more accurate, static method, formula-generated model created by FSvia.
Narrow/Wide Band Via	An FSvia-generated narrow band (single frequency-point) or wide band (multiple frequency-points) model containing RLGC values for a range of frequencies, as specified by the user. When generating a Wideband model, FSvia first generates S-Parameters and then creates RLGC values based on those S-Parameters.

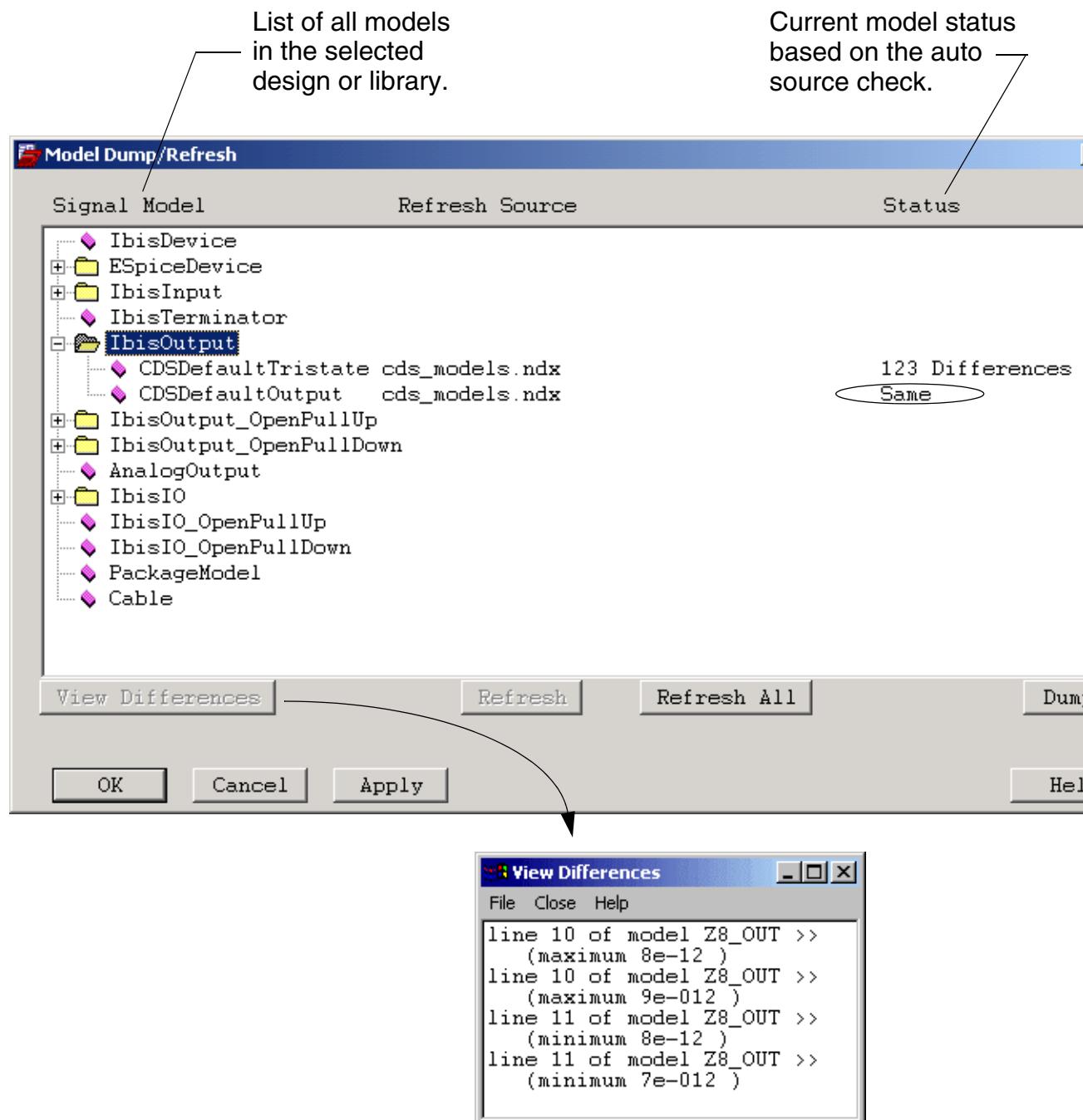
Model Type	Use and Relationship
S-Parameter Single Via	An FSvia-generated S-Parameter model containing values for a range of frequencies, as specified by the user
Signal/Signal Coupled Via	Represents a via model between two signals.
Signal/Ground Coupled Via	Represents a via model between a signal and a ground component.
Signal/Power Coupled Via	Represents a via model between a signal and a power component.
<i>Shape</i>	Models a copper shape encountered in a physical design.

Note: All the model types supported in the highest tier of Allegro PCB SI are displayed in Model Browser of every version of the product. However, access to model types depends on the version of SI for which you are licensed. For example, if an `.iml` library contains a S-Parameter Via model type, it is available in Model Browser for *Allegro PCB SI Multi-Gigabit Option*.

Model Verification and Source Management

You perform verification and source management operations on the device models in a selected design or library by choosing *Analyze – Model Dump/Refresh* from the PCB SI menu bar.

Figure 2-14 Model / Dump Refresh Dialog Box



Use this dialog box to perform verification and source management operations on the device models in a selected design or library. Upon displaying the dialog box, models resident in the current design are checked against their original source. Once the check is completed, the

Allegro PCB SI User Guide

The High-Speed Design Flows

dialog box displays a list of all models in the design and shows related source and status information for each model.

For further details, see [Chapter 3, “Managing Models Resident in a Design.”](#)

Allegro PCB SI User Guide

The High-Speed Design Flows

Model and Library Management

Managing Model Libraries

Introduction to Model Libraries

When analyzing a design, PCB SI builds simulation circuits using the device models that have been stored in your design (.brd file). These *resident* device models are associated with devices in your design. During simulation, SigNoise automatically constructs the required interconnect models. The actual source files for these device and interconnect models are stored and organized in either device model libraries (DML's) or interconnect model libraries (IML's) that are external from the design database.

Working with SI Model Browser

You use the *Signal Model Browser* and the *DML Library Management dialog boxes* to create and manage your libraries of device and interconnect models, and launch Model Editor. You can also use it to specify which device and interconnect libraries you want SigXplorer to access, as well as the order of library access.

To access the **SI Model Browser** dialog box from PCB SI

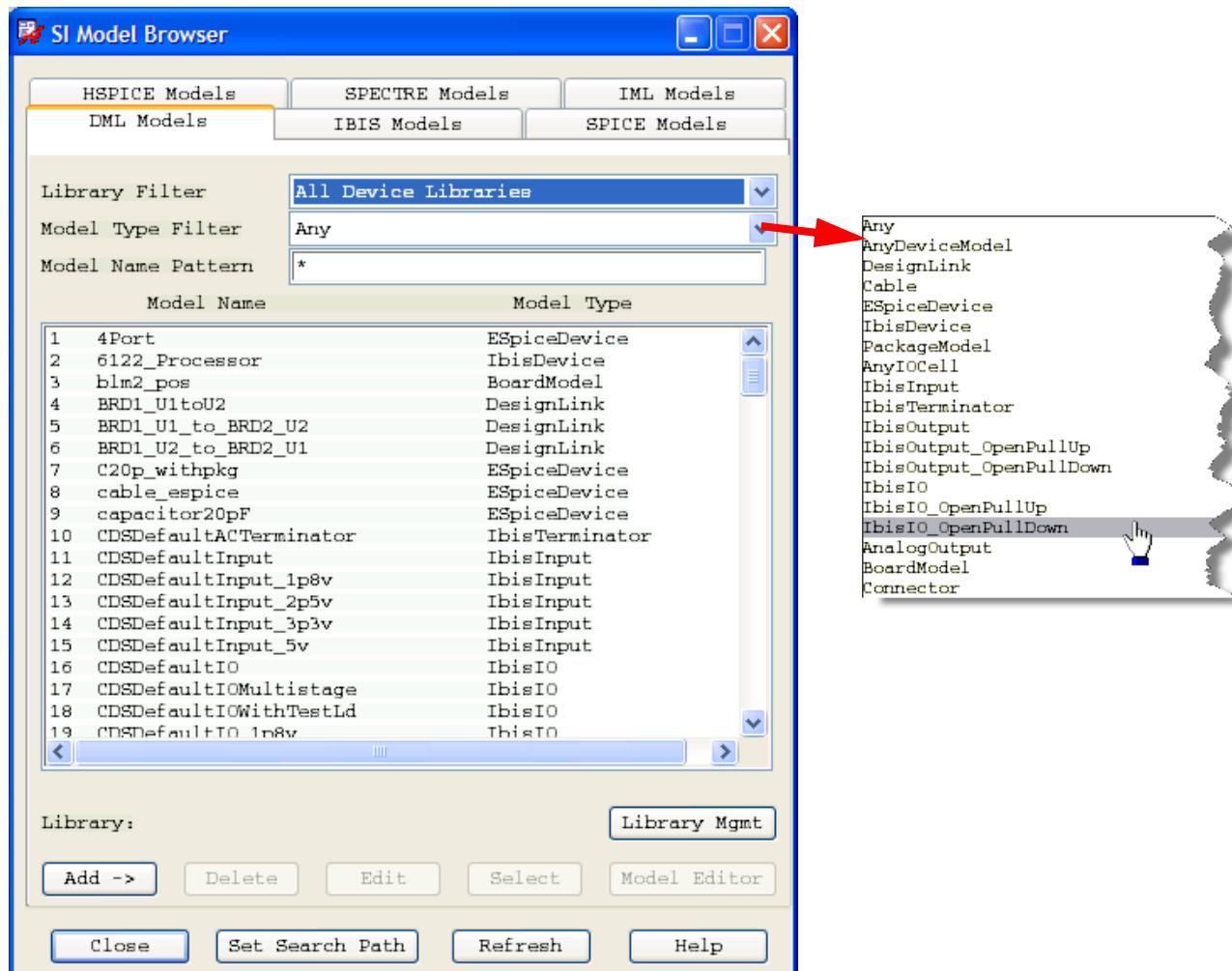
- Choose *Analyze – Model Browser*.

The SI Model Browser appears as shown in [Figure 3-1](#) on page 76.

Allegro PCB SI User Guide

Model and Library Management

Figure 3-1 SI Model Browser



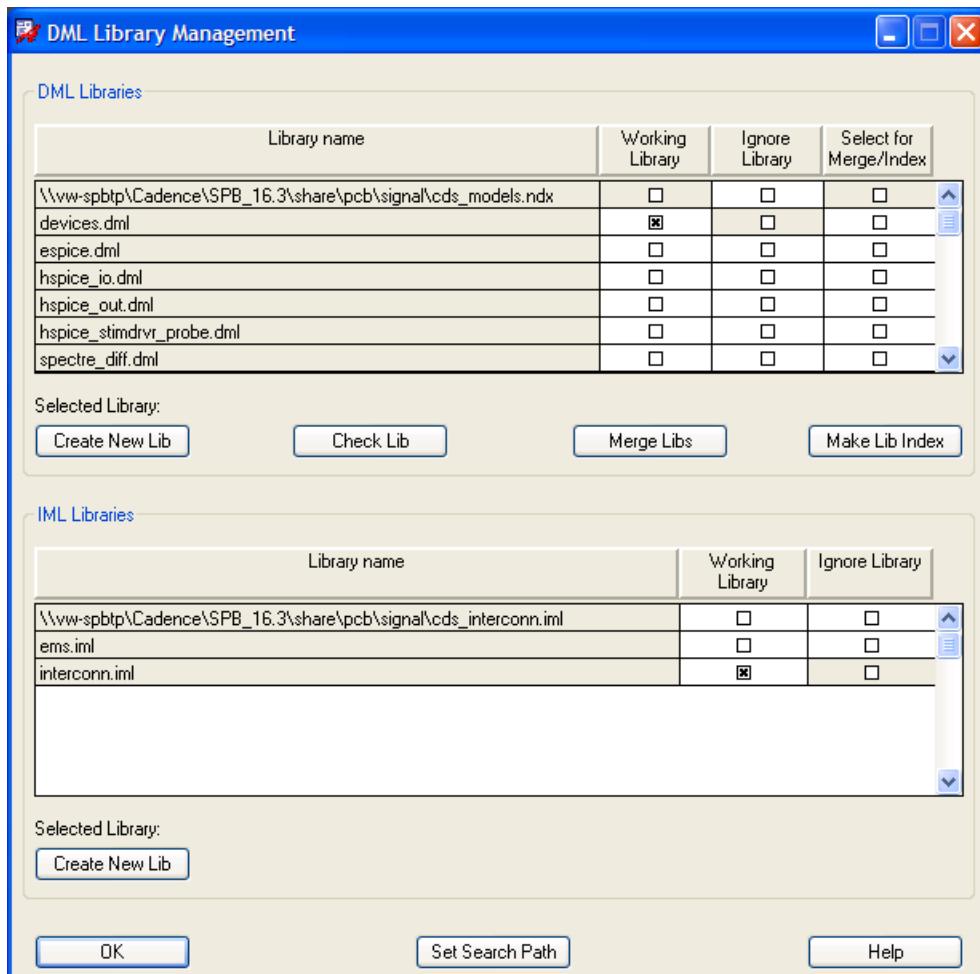
The SI Model Browser's tabbed interface accommodates the model type that you want to translate, be it IBIS, Spectre, Spice, IML, DML, or HSPICE. You need to select the appropriate tab, click the model, and click the *Translate* button to translate it. From these tabs, you can also edit a model directly in its native format. Once translated, these models also appear under the DML tab.

Each tab contains a field for filtering the listed models, as well as a button to set the model's library search path and to set its associated file extensions.

Note: You can access the SI Model Browser in PCB SI from *Analyze – Model Browser*.

Performing Library Management

You use the DML Library Management dialog box to manage DML libraries. To access the DML Library Management dialog box, click the *Library Mgmt* button on the *DML Models* or the *IML Models* tab of SI Model Browser dialog.



The DML Library Management dialog box provides controls to set the working library, ignore libraries, and create indices.

Using the DML Library Management dialog box, you can:

- create a new library and add it to the list of libraries
- Perform a syntax check on a DML library
- specify the working libraries
- create an index for a device model library

- merge two or more device model libraries

Basic Library Management

Adding and Removing Libraries in a Search List

You cannot access device and interconnect models unless their libraries are first added to the library search list. Conversely, if a library is no longer in use, you can remove it from the search list thereby improving overall library access performance.

Reordering the libraries in a Search List

Libraries are searched starting at the top of the list. In cases where a model is included in two or more libraries, you can use the search order to determine which library SigNoise searches first. SigNoise uses the first model found.

Specifying the Working Device and Interconnect Libraries

SigNoise only adds models to the working libraries. If you want to add a model to a library that is not the working library, you must first make it the working library before you start the process. You can have one working device model library and one working interconnect model library active at one time.



The name of the working library for device or interconnect models appears in the *Working:* field above the library search list.

Advanced Library Management

Adding Device Library Indices to a Search List

To improve access performance with large device model libraries, you can add a library index (in place of the library file) to the Device Library search list. An index is a `.ndx` file that contains *pointers* to the device models in the DML. The index requires fewer resources because only models required for analysis are loaded from the index into memory, as opposed to loading the entire library. This utility performs the same checks as `dmlcheck`, to ensure that the indexed models are syntactically correct. Only models that pass `dmlcheck` will be indexed.

Note: Index files are read-only. For this reason, you cannot index any library currently designated as the working library to which SigNoise automatically writes any edits.

Use the `mkdeviceindex` utility to create a library index for one or more device model library files. You can access the `mkdeviceindex` utility from the Library Browser (*Make Index* button) or the command line.

To access the `mkdeviceindex` utility from the command line, enter the following command:

```
mkdeviceindex [-d] [-o <index_filename>] <library_filename>...
```

mkdeviceindex Command Arguments

Argument	Function
<code>-d</code>	Checks model dependencies. For example, an IBIS Device model is indexed only when all required IOCell and PackageModel models are present and pass the checks.
<code>-o</code>	Names the device model library index.
<code>index_filename</code>	Names the device model library index for the IBIS input file you want to translate.
<code>library_filename</code>	Names one or more device model libraries. Note: You can also specify directories which are hierarchically searched for <code>*.dml</code> files.

Merging Device Model Libraries

The *mergedml* utility enables you to combine one or more model libraries into a single library. You can access the *mergedml* utility from the Library Browser (Merge DML button) or the command line.

To access the *mergedml* utility from the command line, enter the following command:

```
mergedml <library_filename>... -o name
```

mergedml Command Arguments

Argument	Function
<i>library_filename</i>	Names one or more device model libraries to combine.
-o	Names the merged output library.

Protecting Device Model Libraries

Use the *dmlcrypt* program to produce encrypted versions of DML files to protect model data. You can use encrypted files for simulation, but you cannot view them in plain text. The program requires the name of an existing DML file and the name of a new encrypted DML file to write. For example:

```
dmlcrypt devices.dml devices_e.dml
```

An encrypted copy of `devices.dml` is saved as `devices_e.dml`.



Once a file is encrypted there is no way to convert it back to plain text. Be careful to retain a copy of the original plain text DML file.

For further protection, add a SPICE comment to your EspiceDevice, MacroModels, or PackageModels as follows:

```
* |protect_simulation_files: comps.spc ibis_models.inc
```

Doing so will remove these files at simulation ensuring that no plain text copy of your data remains.

You can recognize an encrypted DML file by the characters at the beginning:

```
FILE_FMT=SYENCRYPT2      &<qpBi#48tk]OzP) :^"7) [C ...
```

The remainder of the file contains undecipherable binary characters.

Auditing Device Model Libraries

You can use the *dmIcheck* utility to check the syntax of one or more library files. For further details, see [Auditing Models and Libraries](#) on page 132.

Signal Integrity Model Libraries

The Cadence Libraries

The models in the Cadence Libraries fall into three categories:

- DIG_LIB Library Models

Standard digital logic families containing 217 unique parts in a format directly compatible with the SigNoise simulator with their corresponding IBIS files. These models were created using spice files from Signetics (except for one device).

- DEFAULT_LIB Library Models

Default set of IOCell models for gtl, pci, and asic types. You can use these IOCell models to select the right buffer for a given application.

- PACKAGES Library Models

Library Structure

The Cadence Libraries are located in:

```
<install>/cds/share/pcb/signal/SignalPartLib
```

The directory contains the following sub-directories corresponding to the three categories:

- DIG_LIB

Contains subdirectories corresponding to four digital logic families. Each digital logic family subdirectory contains files for IBIS Device models and IOCell models (.dml files). There is one device model in each file. The corresponding IBIS files (.ibs files) also exist. There is one `DIGlib_assump.txt` file giving the test setups used to validate each family.

- DEFAULT_LIB

Contains signoise .dml files and their corresponding .ibs files. The corresponding `assumption.txt` file lists the assumptions, approximations and validation test setup used.

- PACKAGES

Note: In the directory above the `SignalPartLib` directory, SigNoise uses the device model index file `cds_partlib.ndx` to quickly load groups of models.

The DIG_LIB Library Models

The digital device model library supports models for parts from four types of digital logic families (or technologies). A part's digital logic family refers to the different processes and implementations used in the manufacture of the parts used in integrated circuits. For example, you can use bipolar transistor-transistor-logic, CMOS, bipolar emitter coupled logic, or a combination of several technologies. Each technology has different input and output parameters. The library includes the digital logic families described in the following table.

Digital Device Model Library

Technology	Description
<i>ABT</i>	Advanced BiCMOS Technology for bus interfaces with high drive, lower power consumption, and fast propagation.
<i>ALS</i>	Advanced Low Power Schottky for low power consumption in non-speed critical circuits.
<i>ALVC</i>	Advanced Low Voltage CMOS for low power consumption.
<i>FTTL</i>	Fast Transistor-Transistor Logic for speed critical circuits.

The DEFAULT_LIB Library Models

The following table describes the default set of library models.

Technology	Number of IOCell Models	Description
<i>GTL</i>	01	IOCell models derived from Texas gtl spice files.
<i>PCI</i>	04	Compatible IOCell models for both 3v and 5v derived from Intel pci spice files.
<i>ASIC</i>	22	IOCell models for different current capability both for 3v and 5v, with and without slew made from suitable approximations for the ASIC CMOS technology.

The PACKAGES Library Models

The default package models include:

- 20dip
- 14soic
- 16soic
- 20soic
- 16ssop
- 20ssop
- 28plcc
- 44plcc

For a complete list of models, refer to the `<install>/share pcb/signal/cds_partlib.ndx` file. You can search this file using the UNIX `grep` command or the Windows NT `Find` command.

Managing Models

Introduction to Simulation Models

There are two basic categories of models used to build circuits for simulation.

- Device models
- Interconnect models

Device models must be obtained in advance of simulation. You use them to characterize manufactured components such as ICs, discrete components, and connectors. They are stored in files with a `.dml` extension. A device model library consists of a `.dml` file that contains one or more device models.

Interconnect models are extracted directly from the physical design database and synthesized on demand. Interconnect models cover such items as traces and vias, and are stored in files with a `.iml` extension.

Where to Obtain Device Models

You can procure or create device models to suit your design requirements. Choose from a standard parts library of Cadence models, procure models from the device manufacturer, translate IBIS, SPICE, or Quad models to SigNoise's native format, or create your own models through cloning, physical test measurements, or information gleaned from databooks. You can also work with Cadence's consulting services to obtain custom models based on your unique requirements.

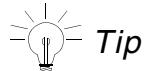
Model Data Verification

Use the Model Editor and SI Model Browser to easily manage the integrity of the model data required for high-speed circuit simulations. For more information, see the [Allegro Signal Explorer User Guide](#).

The Cadence Sample Model Libraries

The simulation models are stored in device and interconnect model libraries. The Cadence sample device model libraries are named `cds_*.dml` and are located in:

```
<install>/share pcb/signal
```



Tip

You can examine these files for important details regarding the format and syntax of device models as well as the structure of the device model library. The libraries contain examples of different types of device models and are well commented as shown in the following example.

Example

```
(IbisPinMap
  (13)                                ; Pin 13 of this device.
  (signal "RAS0#")                    ; Signal name, not used.
  (signal_model "CDSDefaultOutput")   ; The IOCell model for this pin.
  (ground_bus gndbus)                ; Attach IOCell ground pin to this bus.
  (ground_clamp_bus gndbus)          ; If not present this defaults to the
                                    ; ground_bus value.
  (power_bus pwrbus)                 ; Attach IOCell power pin to this bus.
  (power_clamp_bus pwrbus)           ; If not present this defaults to the
                                    ; power_bus value.
  (R 200m)                           ; Resistance, Inductance, and
  (L 5n)                             ; Capacitance for the pin-lead.
  (C 2p)                            ; These are ignored if PackageModel is
                                    ; used, and defaulted to the
                                    ; EstimatedPinParasitics if not specified
                                    ; and no PackageModel is set.
  (WireNumber 13)                   ; The WireNumber is only needed if a
                                    ; PackageModel is set and the pin-names
                                    ; are not numeric. The WireNumber maps
                                    ; a pin to an index in the PackageModel
                                    ; matrix. If the pin names are numeric
                                    ; and the WireNumber is absent then the
                                    ; pin-name is also assumed to be the
                                    ; WireNumber. In this example the
                                    ; WireNumbers are not needed, but
                                    ; included to show the proper syntax.)
```

Available Models

The following tables describe the available device and interconnect models, their contents, and how they are used.

Device Models

Model Type	Use and Relationship
<i>IbisDevice</i>	Assigned to ICs and connectors with the SIGNAL_MODEL property. (An IbisDevice model for a connector has package parasitics but no IOCell models.)
<i>IbisIOCell</i>	Referenced from an IBISDevice model. Used to model driver and receiver buffers at the pin level. The different types of IOCell models are: IbisOutput – Driver model. IbisOutput_OpenPullUp – Driver model with no pullup resistor. IbisOutput_OpenPullDown – Driver model with no pulldown resistor. IbisInput – Receiver model. IbisIO – Bidirectional buffer model, which can drive or receive. IbisIO_OpenPullUp – Bidirectional buffer model with no pullup resistor. IbisIO_OpenPullDown – Bidirectional buffer model with no pulldown resistor. AnalogOutput – Models the behavior of an analog device pin. IbisTerminator – Models termination internal to the device pin.
<i>PackageModel</i>	Referenced from an IbisDevice model. Models the package parasitics of the entire component package. Can be an RLGC matrix model or SPICE sub-circuits.
<i>ESpiceDevice</i>	Assigned to discrete parts like resistors and capacitors with the SIGNAL_MODEL property. Contains SPICE sub-circuits.
<i>System Configuration</i>	Used to specify system-level connectivity, like multi-board or advanced package-on-board scenarios. Note: System configurations created prior to release 14.0 of PCB SI appear as DesignLink models.
<i>Cable</i>	Referenced from a system configuration. Models cables interconnecting multiple boards. Can be an RLGC model or SPICE sub-circuits.

Model Type	Use and Relationship
<i>BoardModel</i>	Referenced from a system configuration. Models entire boards for situations in which the physical Allegro database is not available. Contains SPICE sub circuits.

Interconnect Models

Model Type	Use and Relationship
<i>Trace</i>	Geometry-based model that represents a single transmission line with no coupling. A Trace can have frequency-dependent loss.
<i>MultiTrace</i>	Geometry-based model representing multiple, coupled lossy transmission lines. A MultiTrace can have a frequency-dependent loss.
<i>Shape</i>	Models a copper shape encountered in a physical design.
<i>Via</i>	Models the parasitics of a via providing z-axis connectivity between traces.

Use the Model Browser to create, display, manage, and edit the models in your libraries. The Model Browser dialog box functions and basic model development tasks are discussed in the following section. See [Advanced Model Development](#) on page 95 for information on using the model editors and on performing more complex model development tasks.

Basic Model Development

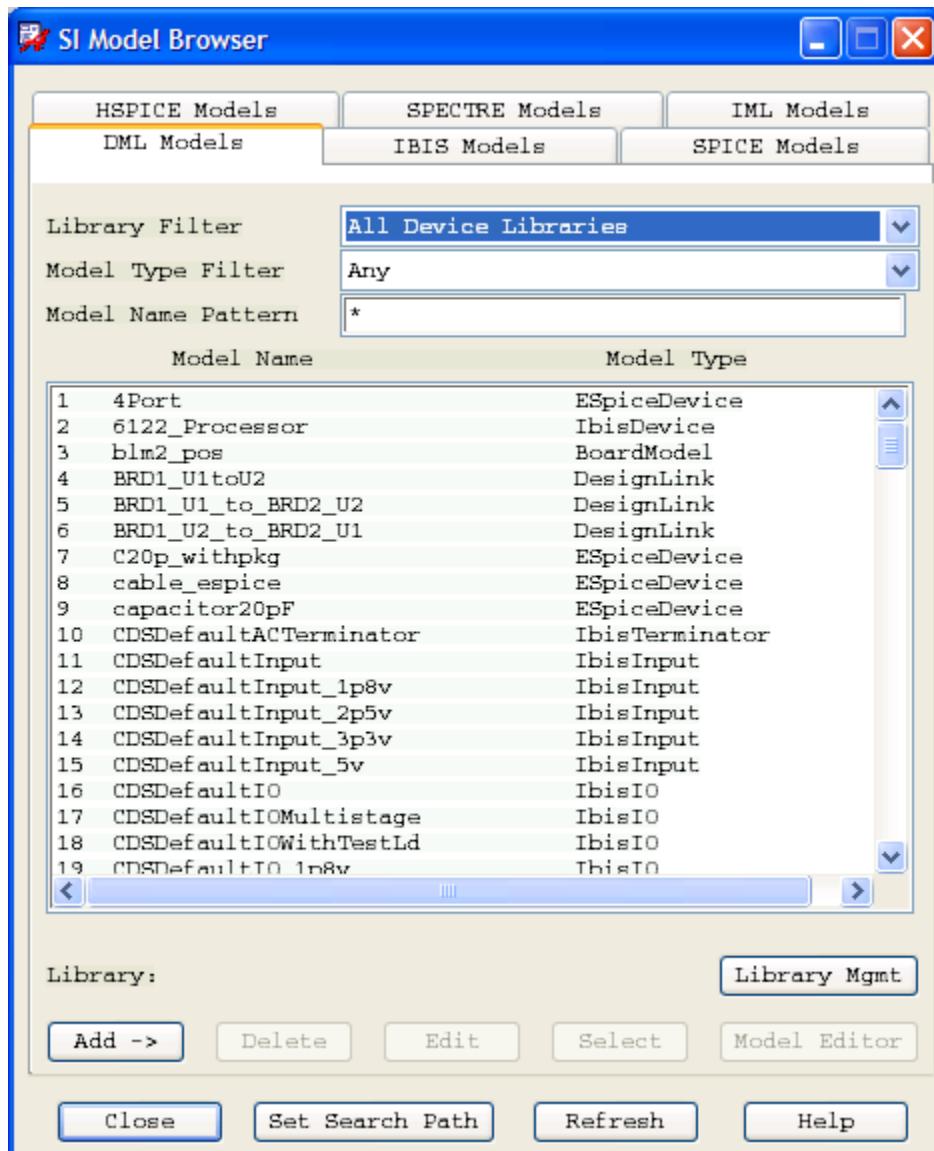
Basic model development tasks are handled through the use of the Model Browsers for Device Library files and Interconnect Library files.

To access the SI Model Browser from the PCB SI

1. Choose *Analyze – Model Browser*.

The SI Model Browser dialog box appears.

Figure 3-2 Model Browser Dialog Box



Using the Model Browsers you can perform the following basic model development tasks.

- List the models in a library.
- Create a device or interconnect model with default values or clone an existing device model and add the newly created model to the working library.
- Delete a model from the working library.
- Select a Model Field Solver (for Interconnect Models only).

Displaying a List of Models

Filter fields at the top of the Model Browser control which models are displayed in the Model Browser list box. You can specify which models are listed in the model search list by library, by model type, or by characters in the model name.

Creating Models and Adding Them to a Working Library

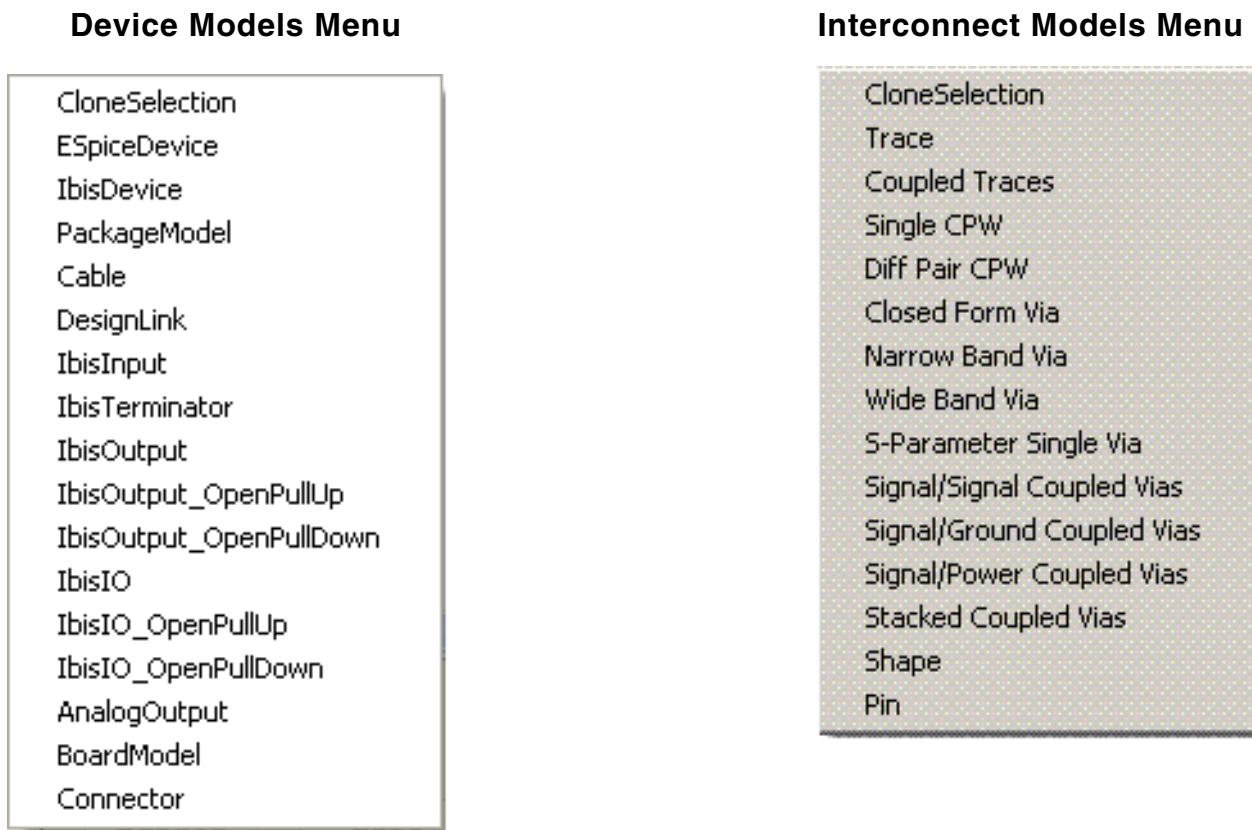
You can add a device or interconnect model to the working device or interconnect model library by copying (or cloning) an existing model or by creating a new model with default values. You must first create a device model before you can edit it to characterize a particular device.

To display a menu of models to add

- Click *Add Model* in the Library area of the Model Browser dialog box.

A pop-up menu of options is displayed for the currently selected model library type as shown in the following figure.

Figure 3-3 Add Model Menus



The following table describes the *Add Model* menu options for device model libraries.

Option	Description
<i>Clone Selection</i>	Copies the selected model from the Model Browser list box and adds the clone to the working library. You specify the name the clone.
<i>EspiceDevice</i>	Displays the Create Espice Device Model dialog box.
<i>IBISDevice</i>	Displays the Create IBIS Device Model dialog box.
<i>PackageModel thru Connector</i>	Opens a dialog box that prompts you to specify a name for the new model type. Clicking <i>OK</i> then adds a template file for the model to the working library that you must edit to complete.

The following table describes the *Add Model* menu options for interconnect model libraries.

Option	Description
<i>Clone Selection</i>	Copies the selected model from the Model Browser list box and adds the clone to the working library. You specify the name the clone.
<i>Trace thru Diff Pair CPW</i>	Opens a dialog box that prompts you to specify a name for the new model. Clicking <i>OK</i> then adds a template file for the model to the working library that you must edit to complete.
<i>Closed Form Via thru Stacked Coupled Vias</i>	Opens the Via Model Generator that lets you create a new model or modify an existing one. For detailed information, view the online Help for the dialog box.
<i>Shape and Pin</i>	Opens a dialog box that prompts you to specify a name for the new model. Clicking <i>OK</i> then adds a template file for the model to the working library that you must edit to complete.

Note: When a new device model is added to the working library, the library check program (*dmlcheck*) verifies the validity of the entry.

Deleting a Model

Click the *Delete* button to remove the previously selected model from the model list box.

For further details on this dialog box or for additional procedures regarding model management, refer to the [signal_library](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

Model Setup in Allegro Design Entry HDL or Third-Party Libraries

You specify the model setup for components in Design Entry HDL libraries. When SigNoise finds that a component has no SIGNAL_MODEL property, it checks to see if it has a SIGNAL_MODEL property on the device definition.

You can attach SIGNAL_MODEL properties to device definitions by setting the SIGNAL_MODEL property in either the:

- *chips_prt* file for that device.
- *phys_prt.dat* file for your schematic.
- Allegro device file (if you are using *netin*).

The value of the SIGNAL_MODEL property must be the name of an IBISDevice or ESpiceDevice model. Furthermore, SigNoise validates all model assignments based on the PINUSE property.

The SIGNAL_MODEL property assigned to components using the Signal Model Assignment dialog box (instances) overrides those in the device definition, if they exist.

SigNoise uses the following precedence to determine which model gets assigned to a device.

1. An instance-specific SIGNAL_MODEL assignment made in the Signal Model Assignment dialog box (stored in the *.brd* file).
2. A SIGNAL_MODEL property on the component definition (Design Entry HDL PPT file).
3. A VOLT_TEMP_MODEL property on the component definition (Design Entry HDL PPT file).
4. A DEFAULT_SIGNAL_MODEL property on the component definition (Design Entry HDL PPT file).

A common use of the DEFAULT_SIGNAL_MODEL property is to establish a model name for the device before the actual model is developed. The simulator warns you when a part with a SIGNAL_MODEL property does not have an associated model; however, if a default model name is attached to a part, as directed by having checked the *Use Defaults For Missing Components Models* in the DeviceModels tab of the Analysis Preferences dialog box, the simulator does not report an error when a model is not yet available.

You can use a default model name pattern as a placeholder for a to-be-procured library of models or for implementing model names based on your internal model naming conventions.

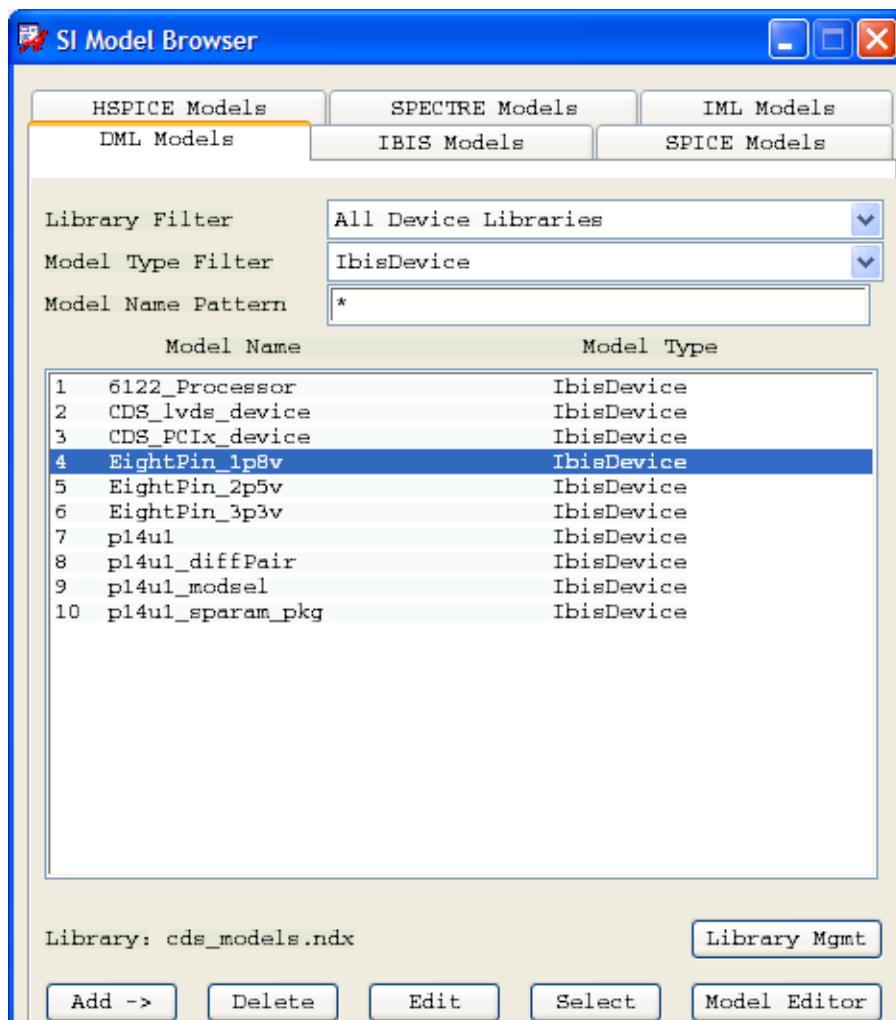
Advanced Model Development

Editing IBIS Device Models

You can edit any existing IBIS device model (except Cadence default models) that has been created and added to a library. If you create the model by cloning (copying) an existing model, you need to edit the cloned model so that it characterizes the device you are modeling. If you create the model from scratch, it contains default values that you may want to edit.

IBIS device models are modified using the IBIS Device Model Editor.

Figure 3-4 Selecting an IBIS Device Model for Editing

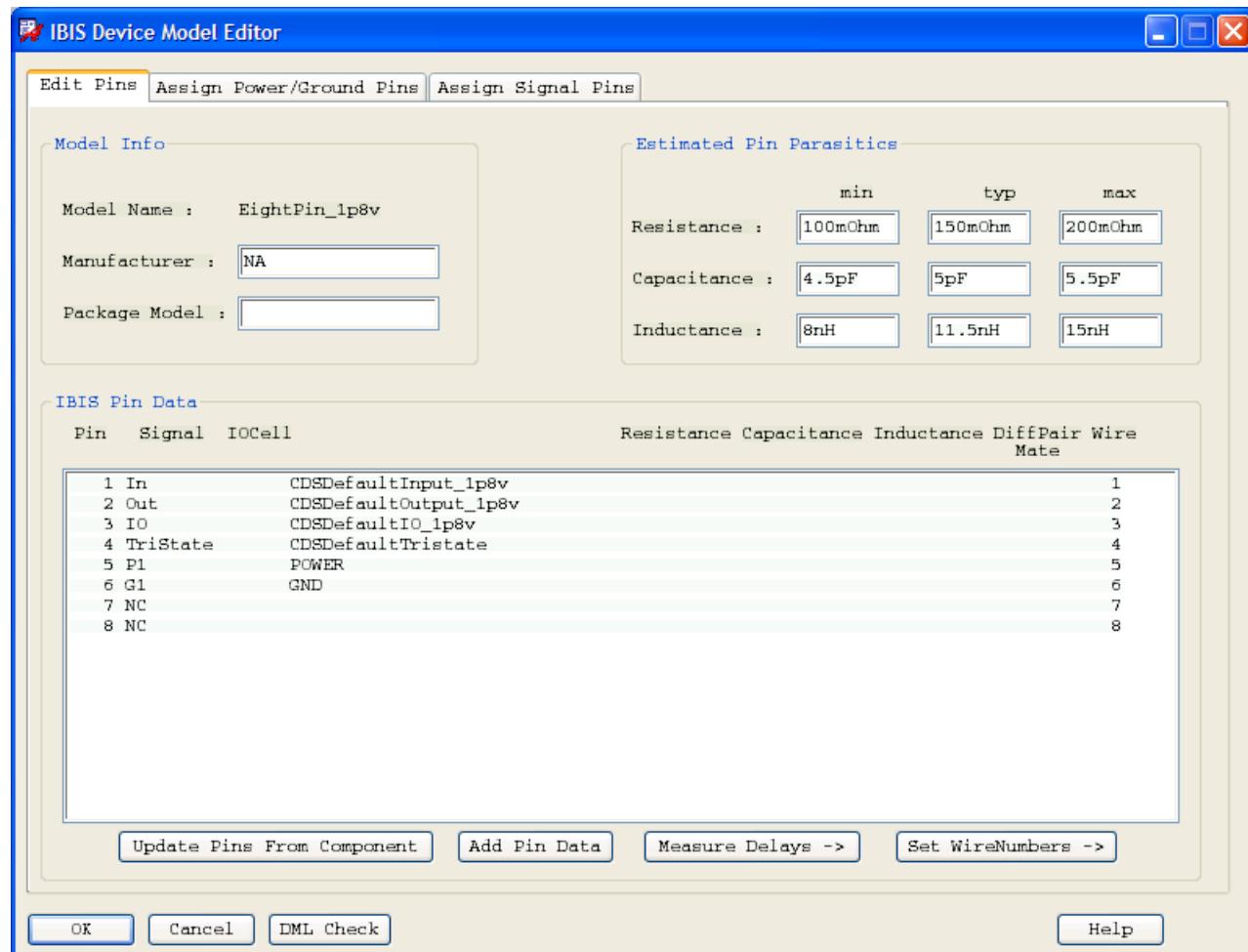


To access the IBIS Device Model Editor

1. Select the *IbisDevice* filter from the *Model Type Filter* drop-down list in the SI Model Browser.
2. Select the appropriate model from the list.
3. Click *Edit*.

The IBIS Device Model Editor dialog box is displayed as shown in the following figure.

Figure 3-5 IBIS Device Model Editor - Edit Pins Tab



The IBIS Device Model Editor dialog box contains three tabs that you can use to:

- edit information for the pins associated with the IBIS device model.
- group power and ground pins and assign them to power and ground buses.

- group signal pins and assign IOCell models and IOCell supply buses.

Edit Pins Tab

Use this tabbed page of the IBIS Device Model Editor to:

- specify package model parasitics for the device.
- specify estimated pin parasitics for the device in terms of minimum, typical, and maximum values for resistance, capacitance, and inductance for the package.

These values are used for pins that have no individual pin parasitics (when the IBIS Device model has no assigned PackageModel). See [Guidelines for Specifying Parasitic Values](#) on page 105 for further details.

- update the IO cell models and diff pair data in the IBIS device model you are editing to match the pin uses defined in a selected component.
- add or modify pin data including individual pin parasitics and buffer delays.
- measure buffer delays.
- add, edit, or display buffer delay information.
- set wire numbers.

IBIS Pin Data Area

Use the *IBIS Pin Data* area to view and edit the data, including buffer delays, for each pin associated with the IBIS device model. Pins are listed by wire number order. Pins with no wire number are listed in alphanumeric order.

When you select a pin in the list box, the IBIS Device Pin Data dialog box appears displaying data for that pin. For further details, see [“Adding or Editing Data for a Pin”](#) on page 108.

For further details on this tab, or for procedures regarding its usage, refer to the [signal_library](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

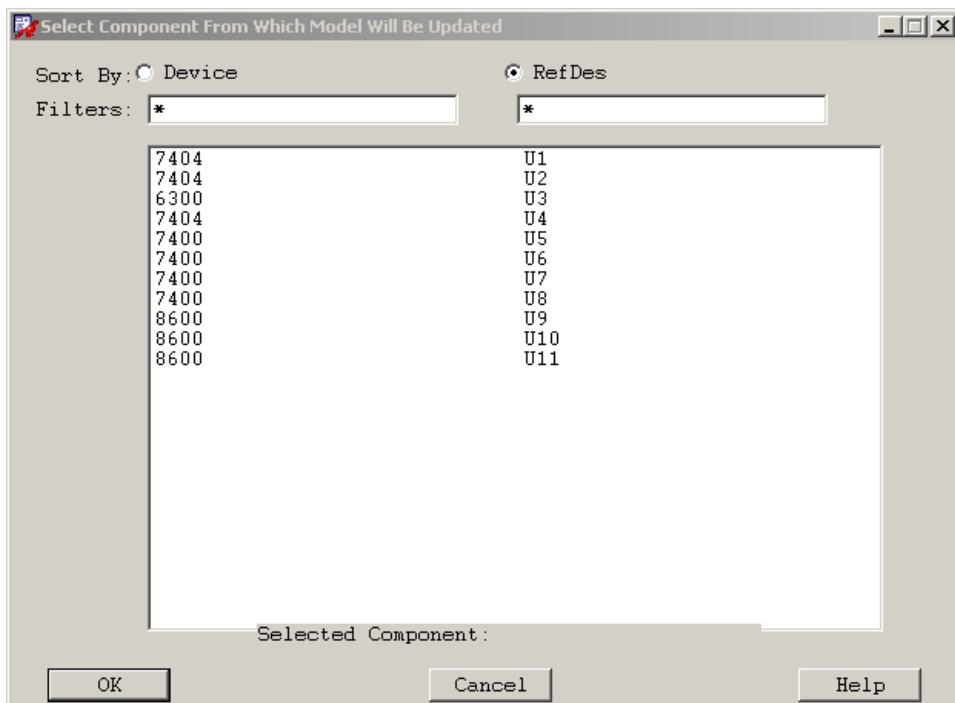
Update process for component pins in the IBIS device model being edited

You can dramatically expedite the process of updating IO cell models and diff pair data in IBIS device models to match the pin uses of a specified component in your design. You will find this procedure particularly useful for devices that contain very large pin counts.

Note: This functionality assumes that the pin uses of components in your design are correct. Also, this feature does not replace or affect the model assignment functionality addressed by the `signal model` command (*Analyze – Model Browser*) that updates the pins of a component based on the buffer model of the IBIS device model assigned to the component.

When you select *Update Pins From Component* in the IBIS Device Model Editor, the Select Component dialog box appears, as shown in Figure 3-6.

Figure 3-6 Select Component... Dialog Box



From here you can select a component, the pins of which will be updated in the following manner:

- ❑ Pins not found in the IBIS device model and whose use is anything other than NC or UNSPEC are added to the model with a unique wire number and an IO cell model name based on the pin use of the component pin. The mapping convention is:

Pin Use	Model Name
IN	<IBISDevicemodelName>IN
OUT	<IBISDevicemodelName>OUT
BI	<IBISDevicemodelName>IO

TRI	<IBISDevicemodelName>OUT
OCA	<IBISDevicemodelName>OUT
OCL	<IBISDevicemodelName>OUT
POWER	POWER
GROUND	GND

- ❑ A search is conducted for IO cell models referenced by pins found in the IBIS device model. If the models are not found, a warning is generated and the pin is not updated in the model. If the models *are* found, pin updates will be governed by the type of IC cell model referenced by the pin, as described below:

IO Cell Type	Component Pin Use	Update
Input	IN	Not updated
	OUT,BI, TRI, OCA, OCL	Outputs error message to log file
	POWER	Sets signal_model to POWER
	GROUND	Set signal_model to GND
	NC	No signal model; set signal to NC
Output	OUT, TRI, OCA, OCL	Not updated
	IN, BI	Outputs error message to log file
	POWER	Sets signal_model to POWER
	GROUND	Set signal_model to GND
	NC	No signal model; set signal to NC

IO	BI	Not updated
	IN	Create IC cell model of type Input
	OUT, TRI, OCA, OCL	Create IC cell model of type Output
	POWER	Sets signal_model to POWER
	GROUND	Set signal_model to GND
	NC	No signal model; set signal to NC

- ❑ When the IO cell model/component pin use combination is IO/IN, an IO cell model of type Input is created, named <IOCellmodelName>_IN (where <IOCellmodelName> is the name of the model of type IO), and added to the working DML library.
- ❑ When the IO cell model/component pin use combination is IO/OUT, TRI, OCA, or OCL, an IO cell model of type output is created, named <IOCellmodelName>_OUT (where <IOCellmodelName> is the name of the model of type IO), and added to the working DML library.
- ❑ Pins found in the IBIS device model but do not exist in the selected component are removed from the model.
- ❑ Pairs of diff pair nets connected to the selected component are updated in the following manner:
 - Where the IBIS device model defines two pins as a diff pair, the pins are not updated.
 - Where the IBIS device model does not define two pins as a diff pair, the information is added to the model. An attempt is made to determine which are the inverting and non-inverting pins from the names of the nets assigned to the pins. The naming formats searched for is:

Non-Inverting Net Name Prefix or Suffix	Inverting Net Name Prefix or Suffix
P	N
+	-
POS	NEG

T

Non-Inverting Net Name Format

<name>

<name>

R

Inverting Net Name Format

<name>_

<name>_N

For example, if two diff pair pins are connected to nets ABC_POS and ABC_NEG, ABC_POS is identified as the non-inverting net and ABC_NEG the inverting net. Additionally, if two nets exist named ABC and ABC_, ABC identifies the non-inverting pin and ABC_ identifies the inverting pin.

Where a polarity of the diff pair pins cannot be established from their net names, a polarity is assigned randomly.

- Where either of the two pins have been defined as part of a different diff pair in the model, the diff pair is deleted.

When pin updating is completed, a `pinUpdate.log` file is created describing all the actions taken during the update process. the information is displayed automatically in the manner shown below.

```
*****Updating pins of model 7404 from component U1 device type 7404
*NOTE: Pin 1 exists in both the component and the model. The component pin
is of type input. The model references buffer model CDSDefaultIO which
is of type IO. A new buffer model named CDSDefaultIO_IN of type input
will be created from the IO buffer model.
```

```
*WARNING: Pin 2 exists in the component but not in the model.
It will be added to the model with a buffer model of 7404_IO
```

```
*NOTE: Pin 3 exists in both the component and the model. The component pin
is of type input. The model references buffer model CDSDefaultIO which
is of type IO. A new buffer model named CDSDefaultIO_IN of type input
will be created from the IO buffer model.
```

```
*WARNING: Pin 4 exists in both the component and the model. The component pin
is of an unspecified type so no changes will be made to the model pin.
```

```
*WARNING: Pin 5 exists in the component but not in the model.
It will be added to the model with a buffer model of 7404_IN
```

*WARNING: Pin 6 exists in both the component and the model but no buffer model is defined. A reference to buffer model `7404_IO` will be added.

*NOTE: Pin 7 exists in both the component and the model. This pin is correctly defined as a GROUND in the model so no changes are required.

*WARNING: Pin 8 exists in both the component and the model. The component pin is of an unspecified type so no changes will be made to the model pin.

*NOTE: Pin 9 exists in both the component and the model. The component pin is of type output. The model references buffer model `CDSDefaultIO` which is of type IO. A new buffer model named `CDSDefaultIO_OUT` of type output will be created from the IO buffer model.

*NOTE: Pin 10 exists in both the component and the model. The component pin has a pin use of GROUND and the model pin references buffer model `CDSDefaultIO`. This buffer model will be changed to GND.

*NOTE: Pin 11 exists in both the component and the model. The component pin has a pin use of NC and the model pin references buffer model `CDSDefaultIO`. This buffer model will be changed to NC.

*NOTE: Pin 12 exists in both the component and the model. The component pin has a pin use of POWER and the model pin references buffer model `CDSDefaultIO`. This buffer model will be changed to POWER.

*NOTE: Pin 13 exists in both the component and the model. The component pin is of type output. The model references buffer model `CDSDefaultIO` which is of type IO. A new buffer model named `CDSDefaultIO_OUT` of type output will be created from the IO buffer model.

*NOTE: Pin 14 exists in both the component and the model. This pin is correctly defined as a POWER in the model so no changes are required.

*WARNING: Pin 15 exists in the model but not in the component. It will be removed from the model.

*NOTE: A diff pair between pins 1 and 3 exists in the component so will be added to the model.

Allegro PCB SI User Guide

Model and Library Management

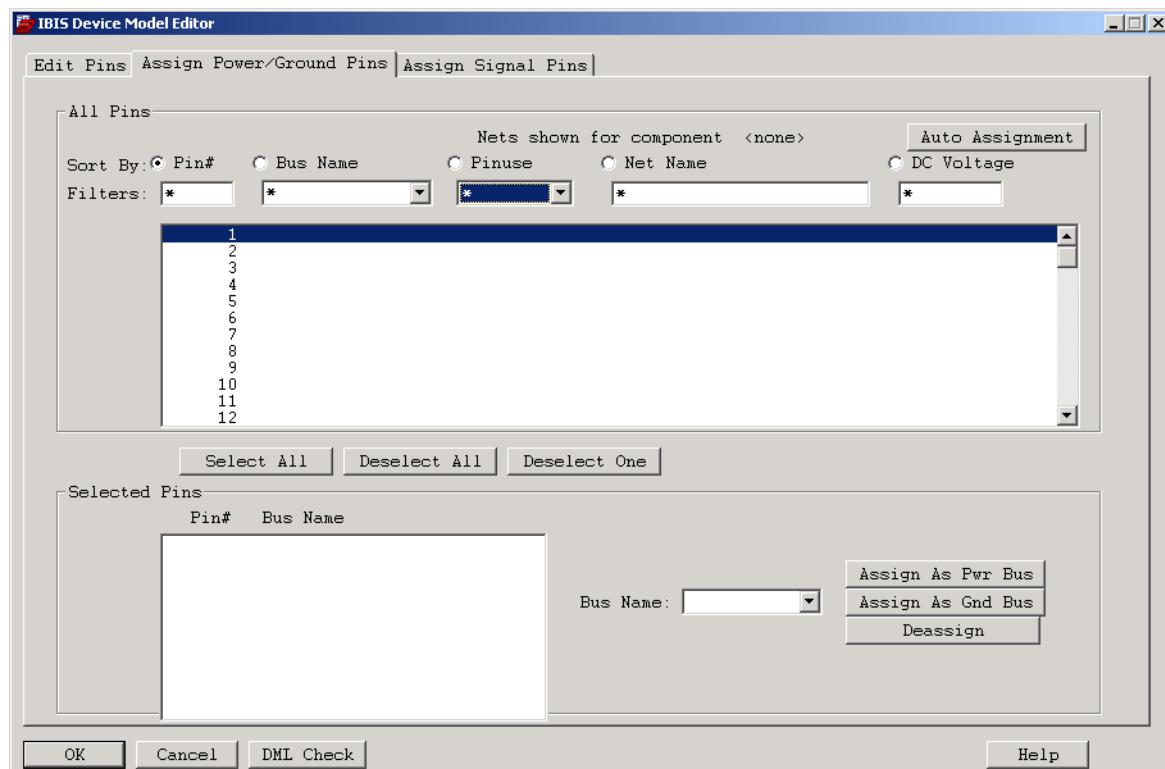
*NOTE: A diff pair between pins 6 and 8 exists in both the model and the component so it will be saved.

*WARNING: A diff pair between pins 5 and 2 exists in the model but not in the component. It will be removed from the model..

Assign Power/ Ground Pins Tab

Use this tab to group the power and ground pins of a device into named power and ground buses.

Figure 3-7 IBIS Device Model Editor - Assign Power/Ground Pins Tab

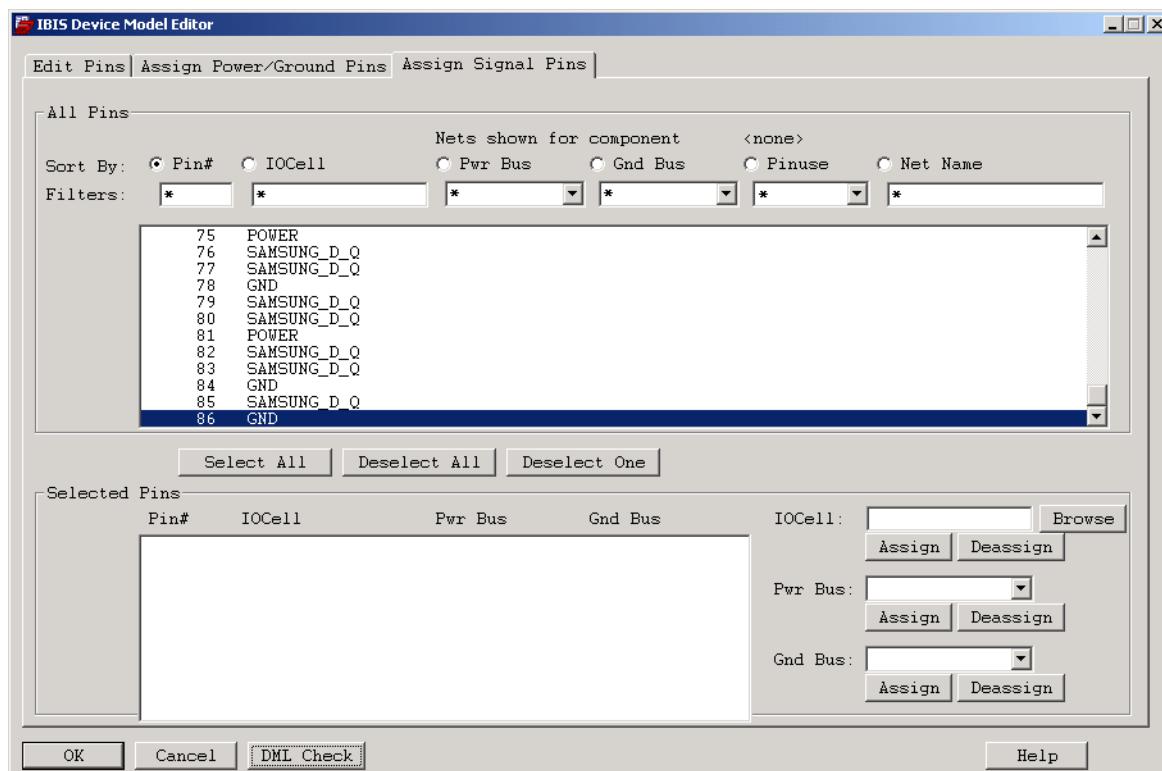


For further details on this tab, or for procedures regarding its usage, refer to the [signal library](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

Assign Signal Pins Tab

Use this tab to group the signal pins of a device and assign a power or ground bus name or an IOCell model to the group.

Figure 3-8 IBIS Device Model Editor - Assign Signal Pins Tab



For further details on this tab, or for procedures regarding its usage, refer to the [signal library](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

Guidelines for Specifying Parasitic Values

Parasitic values for an IBIS device are represented with differing levels of detail.

- Estimated pin parasitics are the most general.

Estimated pin parasitic values are defined globally for the package. The same set of resistance, inductance, and capacitance values are used for all pins on the device. Estimated pin parasitics consist of minimum, typical, and maximum values for package resistance, inductance, and capacitance. See [Specifying Estimated Pin Parasitics](#) on page 105 for instructions.

- Individual pin parasitics are more specific.

Individual package resistance, inductance, and capacitance values are defined on a pin-by-pin basis. When they are supplied, these individual pin parasitic values override estimated pin parasitic values, if any exist.

- Package model parasitics are most specific.

A detailed RLGC that can specify mutual coupling is defined for the entire package. Optionally, a package model can contain an arbitrary passive SPICE circuit. When a package model is present, it takes precedence over both estimated and individual pin parasitic values, if any exist. See [Specifying Package Model Parasitics](#) on page 105 for instructions.

Specifying Parasitics

To specify parasitics for an IBIS device, first select the *Edit Pins* tab of the IBIS Device Model Editor dialog box.

Specifying Estimated Pin Parasitics

Use the *Estimated Pin Parasitics* area of the *Edit Pins* tab to enter minimum, typical, and maximum values for resistance, capacitance, and inductance. Delete any listed package model from the *Package Model* field in the *Model Info* section of the *Edit Pins* tab.

Specifying Package Model Parasitics

Use the *Package Model* area of the *Edit Pins* tab to specify a package model for the IBIS device model. Select a package model name in the Model Browser or click the *Package Model* field and type the package model name.

Allegro PCB SI User Guide

Model and Library Management

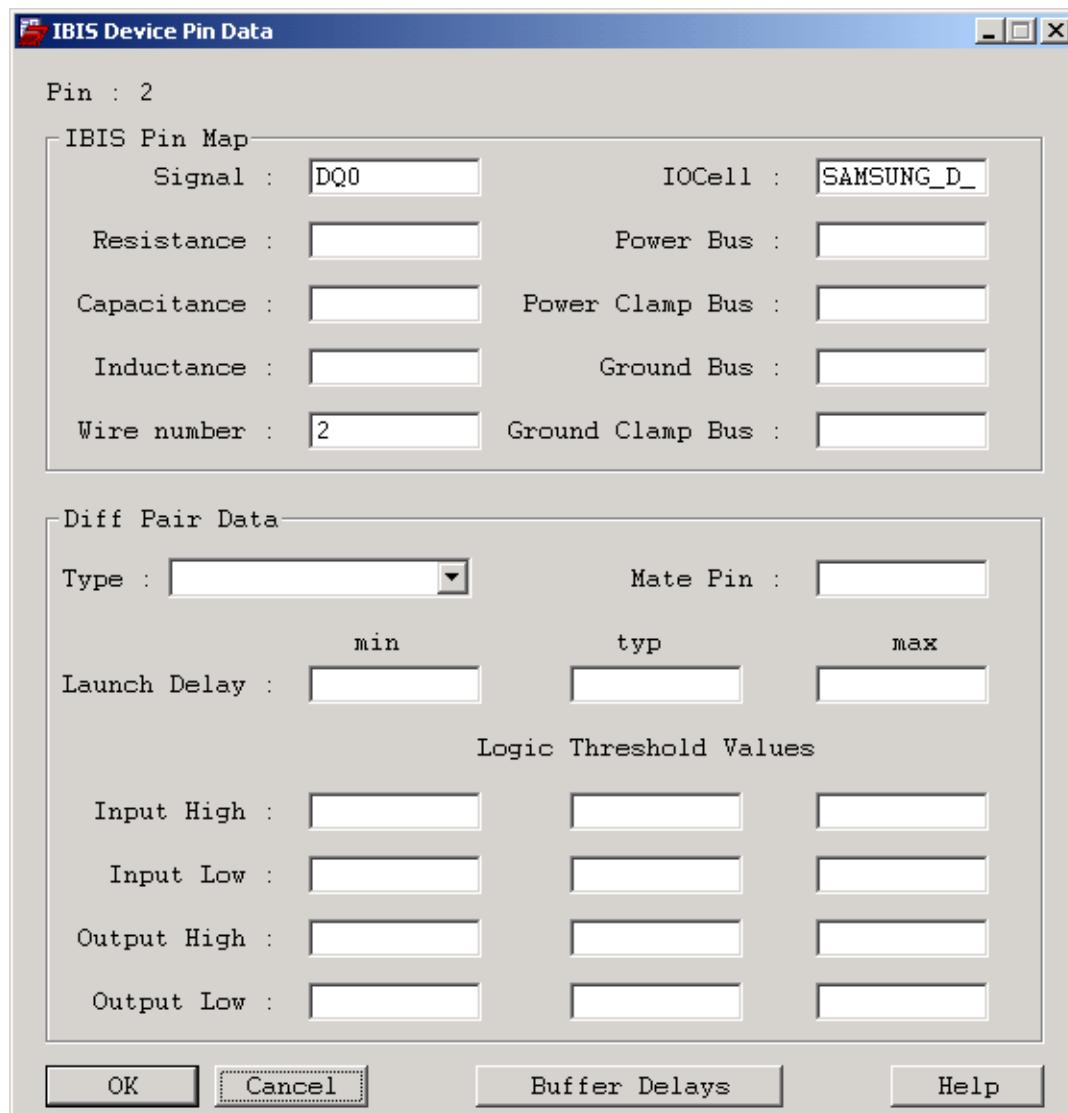
Note: SigNoise uses this specified package model in the simulation circuit model of the layout and ignores any values in the *Estimated Pin Parasitics* fields. Also, ensure that you re-measure buffer delays after changing parasitics.

Adding and Changing Pin Definitions

The IBIS Device Model Editor contains a list box that displays the pins in the device and describes the data defined for each; for example, the pin's IOCell model and parasitics. From the IBIS Device Model Editor, you can display the IBIS Device Pin Data dialog box to:

- add or edit data (including individual pin parasitics) for the pins in the IBIS device model.
- add or edit buffer delay information for the pins in the IBIS device model. See [“Adding or Editing Buffer Delay Data for a Pin”](#) on page 109.

Figure 3-9 IBIS Device Pin Data dialog box



Adding or Editing Data for a Pin

To access the IBIS Device Pin Data dialog box with data for a specified pin.

- ▶ Click to select the pin in the IBIS Pin Data list box in the IBIS Device Model Editor.
The IBIS Device Pin Data dialog box appears as shown in [Figure 3-9](#) on page 107.

To display data for a different pin.

- ▶ Click to select another pin in the IBIS Pin Data list box.
The IBIS Device Pin Data dialog box changes to display data for that pin.

To display the IBIS Device Pin Data dialog box and add data for a new pin.

- ▶ Click *Add Pin Data* on the Edit Pins tabbed page and specify the pin name.
For further details on this tab, or for procedures regarding its usage, refer to the [signal_library](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

Adding or Editing Buffer Delay Information

Use the Buffer Delays dialog box to add or edit buffer delay information for the IOCell models associated with the pins in an IBIS device model.

Note: Whenever any IOCell model data changes, it is important to recalculate buffer delay values in all drivers mode using either the *Measure Delays* in the Buffer Delays dialog box or *Measure Delays — All Drivers* from the IBIS Device Model Editor.

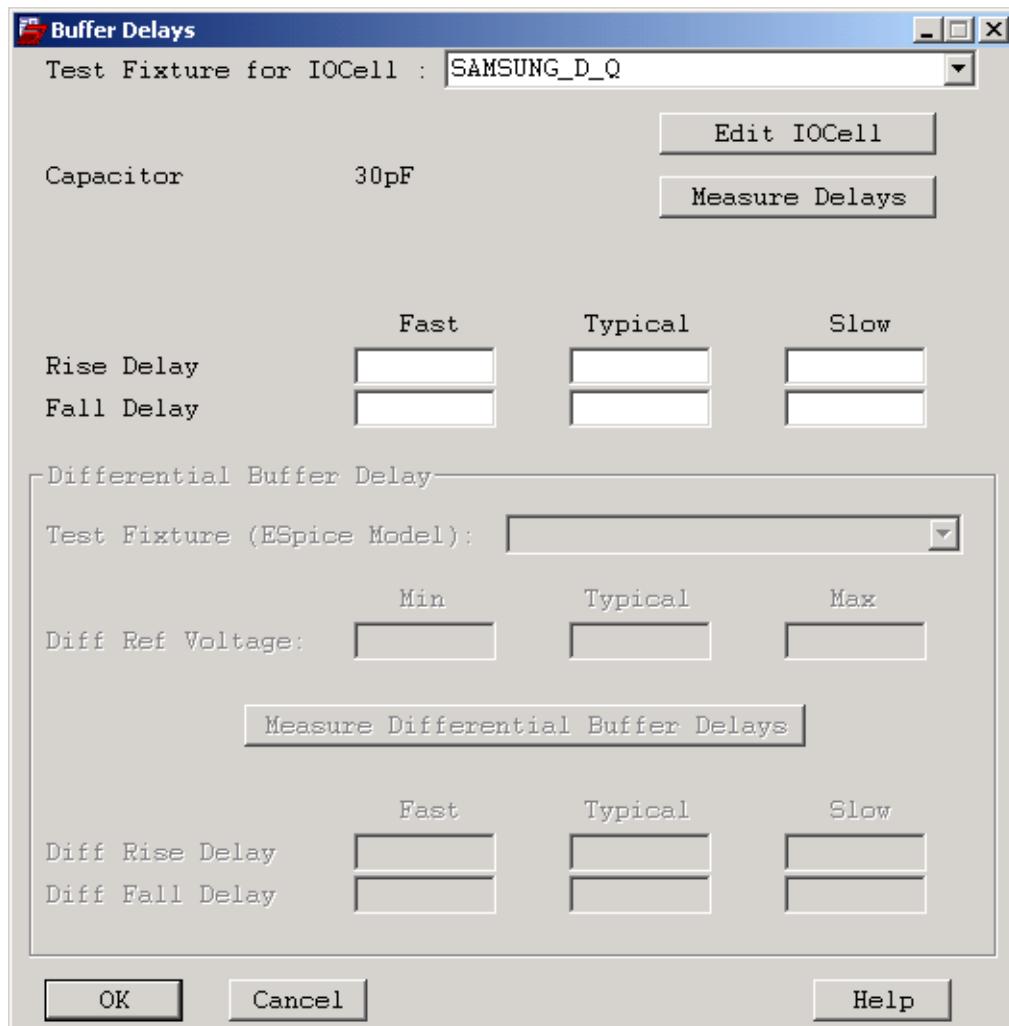
Adding or Editing Buffer Delay Data for a Pin

To access the Buffer Delays dialog box for a specific pin.

- Click *Buffer Delays* on the IBIS Device Pin Data dialog box.

The Buffer Delays dialog box appears as shown in [Figure 3-10](#) on page 110.

Figure 3-10 Buffer Delays dialog box



For further details on this tab, or for procedures regarding its usage, refer to the [signal_library](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

More About Buffer Delays for an IBIS Device Model

When buffer delay values exist for the pins on an IBIS device, they are used to compensate the switch and settle delay values that appear in reports. The pin's associated buffer delay value (rise/fall or fast/typical/slow) is subtracted from the absolute switch and settle delay waveform measurements to produce the compensated switch delay and settle delay values.

Note: If no buffer delay values exist for a particular driver pin, the reported absolute switch and settle delay values are left uncompensated.

When measuring buffer delay values for a device, SigNoise performs only one simulation set for each group of pins having the same IOCell model, pin parasitics, and test fixture. Because of the complex nature of package models, SigNoise always simulates all pins for any device having an assigned package model.

Measuring Buffer Delay Values in the IBIS Device Model Editor

Click on *Measure Delays* to display the delay measurement options: *Unmeasured Drivers*, *All Drivers*, and *Clear All Delays*.

- Use *Measure Delays – Unmeasured Drivers* to calculate all six measured delay values: slow, typical, and fast buffer delays for all rising and falling drivers that currently have no buffer delay values. In unmeasured drivers mode, SigNoise will reuse existing simulation results, even from other devices, in order to maximize performance.
- Use *Measure Delays – All Drivers* to calculate all six measured delay values: slow, typical, and fast buffer delays for every rising or falling driver. These new values override any previous buffer delay values that exist in the model. The new values are saved in the buffer delay section for each driver. In all drivers mode, all delay values are re-measured. Existing simulation results are not used. Use all drivers mode to regenerate buffer delay values for a device whenever any changes are made to the IOCell models for that device.
- Use *Measure Delays – Clear All Delays* to reset to 0 all buffer delay values.

These new (or deleted) values override any previous buffer delay values that exist in the model. The new values are saved for each driver.

These new values override any previous buffer delay values that exist in the model. The new values are saved for each driver.

Using Buffer Delay Compensation

There are several ways to use buffer delay compensation.

- Generate Reflection or Delay reports which have buffer delay compensated switch and settle time values.
- Examine switch and settle delay values using the SigXplorer results spreadsheet.
- Create and examine compensated delay values for a driver using the IBIS Device Model Editor.

If buffer delay values exist for a pin on an IBIS device, the delay values are extracted from the library data for the pin's IOCell model and subtracted from the simulated times of threshold crossing delays to produce compensated first switch and final settle delays.

When buffer delay values do not exist for an IOCell model, no buffer delay is subtracted and buffer delay appears in reports as 0.0.

Measuring Compensated Buffer Delay Values from the IBIS Device Model Editor

You can use the IBIS Device Model Editor to simulate, measure, and edit buffer delay values and associated data for drivers associated with a selected IBIS Device model.

Editing IBIS IOCell Models

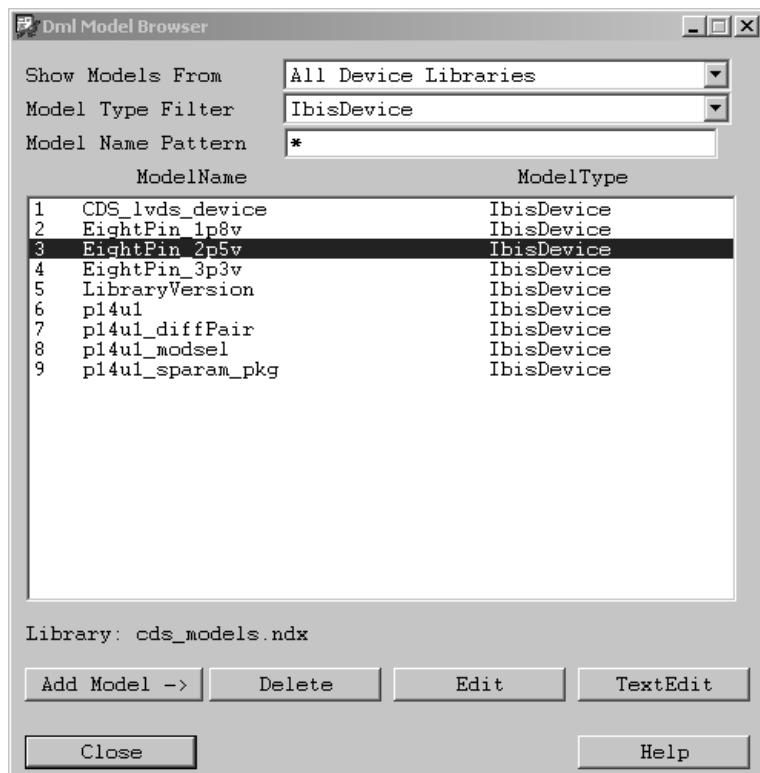
You can edit any existing IOCell model that has been created and added to a library. If you created the model by cloning (or copying) an existing model, you need to edit the cloned model so that it characterizes the device you are modeling. If you created the model from scratch, it will contain default values that you may want to edit. See “[Introduction to Simulation Models](#)” on page 85 for general information on creating device models and adding them to a library.

Note: Whenever any changes are made to the IOCell models for a device, regenerate the buffer delay values for a device using the *All Drivers* mode. See “[Adding or Editing Buffer Delay Data for a Pin](#)” on page 109 for more information.

IBIS IOCell models are modified using the IBIS IOCell Editor. Using the IOCell Editor you can modify:

- general information about the model.
- high and low logic thresholds for an input buffer.
- rise and fall times and high and low logic thresholds for an output buffer.
- delay measurement test fixture data for the model.

Figure 3-11 Selecting an IBIS IOCell model for editing



To access the IBIS IOCell editor

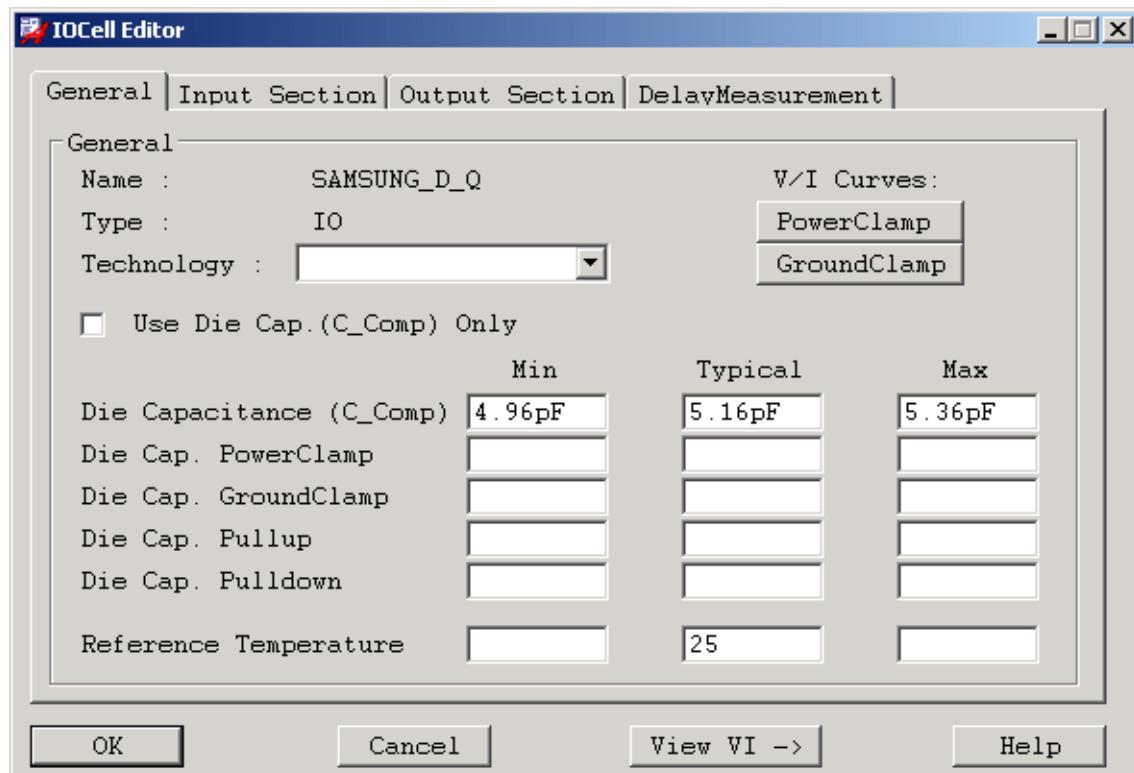
- Double-click on an IOCell name in the Dml Model Browser list box

- or -

Select an IBIS IOCell model from the Model Browser list box, then click *Edit*.

The IBIS IOCell Editor dialog box appears as shown in the following figure.

Figure 3-12 IOCell Editor - General Tab



General Tab

Use the *General* tab of the IOCell Editor to perform the following tasks.

- List the name, model type, and technology family of the IOCell model you are editing.
- Access the V/I curve editors.
- List various minimum, typical, and maximum die capacitance values as well as reference temperature values for the IOCell model you are editing.

General Tab Usage Notes

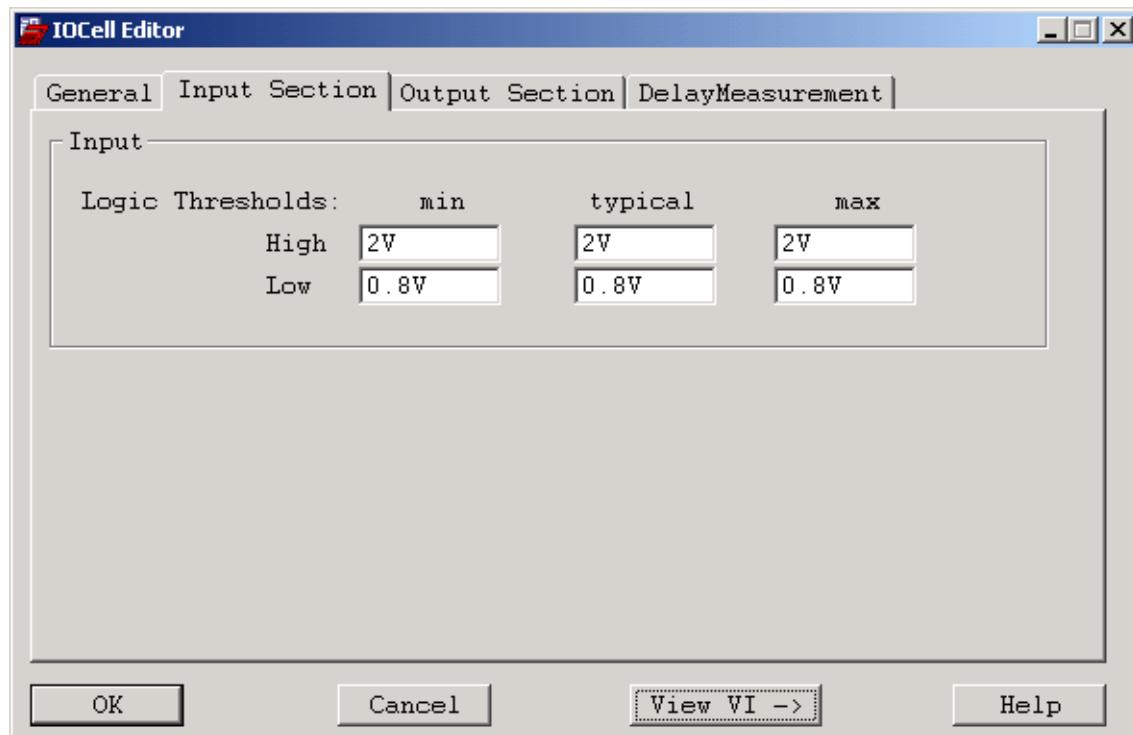
- You can verify the minimum, typical, and maximum values for the die capacitance in the IBIS data file; the die capacitance values are listed under c_comp.
- The reference temperature is typically 50 degrees, with a minimum of 0 degrees and a maximum of 100 degrees. The voltage and current values for the Power Clamp, Ground Clamp, Pull Up, and Pull Down VI curves that you enter elsewhere in the IOCell editor correspond to the reference temperature you enter here.

For further details on this tab, or for procedures regarding its usage, refer to the [signal_library](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

Input Section Tab

Use this tab to enter high and low thresholds for voltage-in.

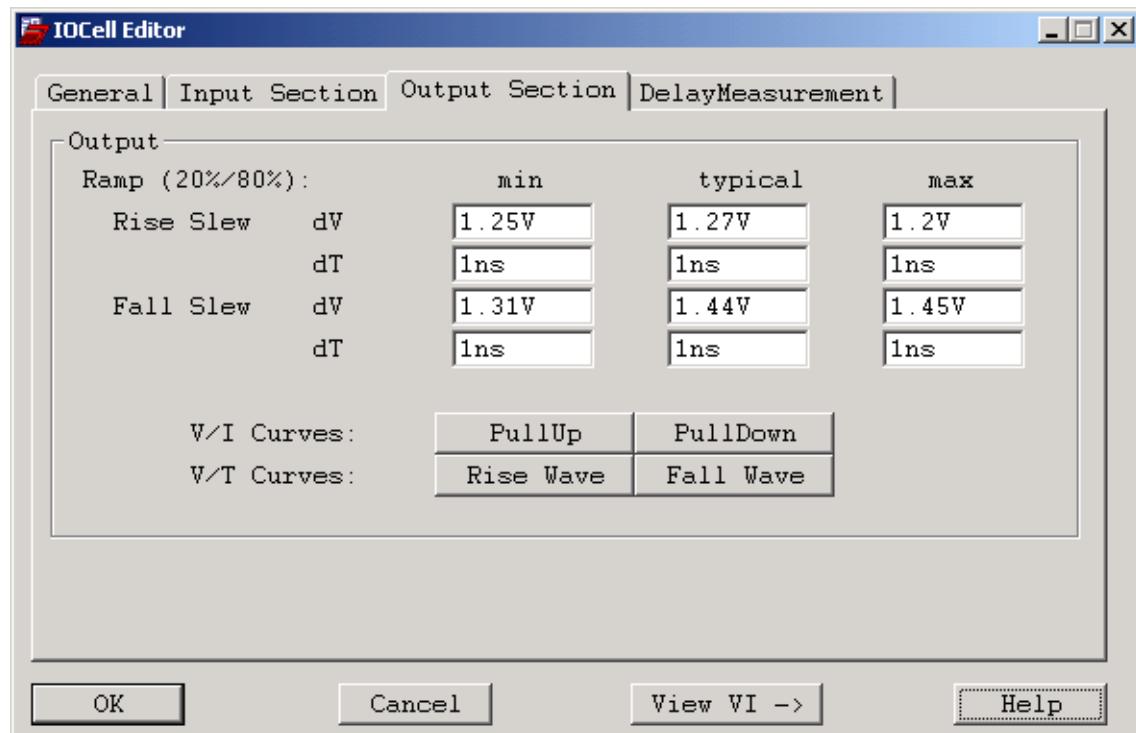
Figure 3-13 IOCell Editor - Input Section Tab



Output Section Tab

Use this tab to enter minimum, typical, and maximum voltage and time delta values for rise and fall slew rate. You can also access the VI and VT curve editors from this tab.

Figure 3-14 IOCell Editor - Output Section Tab



Output Section Tab Usage Notes

- Use the *Rise Slew dV* and *dT* fields and the *Fall Slew dV* and *dT* fields to specify minimum, typical, and maximum values for ramp rates associated with the IOCell model. By IBIS convention, the ramp rate values (the dV/dT values or slew rates) are required to be 20%/80% values. This implies that the rise time and the fall time are defined as the time it takes the output buffer to go from 20% of its final value to 80% of its final value. The *dV* value represents the difference between 20% and 80% of the actual voltage swing. The *dT* value is the actual time taken for the 20%/80% voltage swing. You must also correctly categorize ramp rate values as minimum, typical, and maximum slew rates. The minimum value is the slowest slew rate and the maximum value is the fastest.
- Use *PullUp* and *PullDown* to start a VI Curve Editor and enter high and low output voltage and current data point values
 - *PullUp*—High state V/I curve
 - *PullDown*—Low state V/I curve

- The *View VI* button displays the following curves in SigWave.
 - Sum of Pullup, PowerClamp, and GroundClamp (hidden)
 - Sum of Pulldown, PowerClamp, and GroundClamp (hidden)
 - Pullup
 - Pulldown
 - PowerClamp
 - GroundClamp
- The Pullup and PowerClamp curves are offset on the voltage scale by the respective Reference Voltage. For a 3.3V part, for example, 1.0V on the Pullup curve is summed at $3.3V - 1.0V = 2.3V$ on the curve display. The composite sum curves are initially hidden in SigWave.
- Use *Rise Wave* and *Fall Wave* to start a VT Curve Editor and enter high and low output voltage and time data point values
 - Rise Wave*—Rising V/T curve
 - Fall Wave*—Falling V/I curve

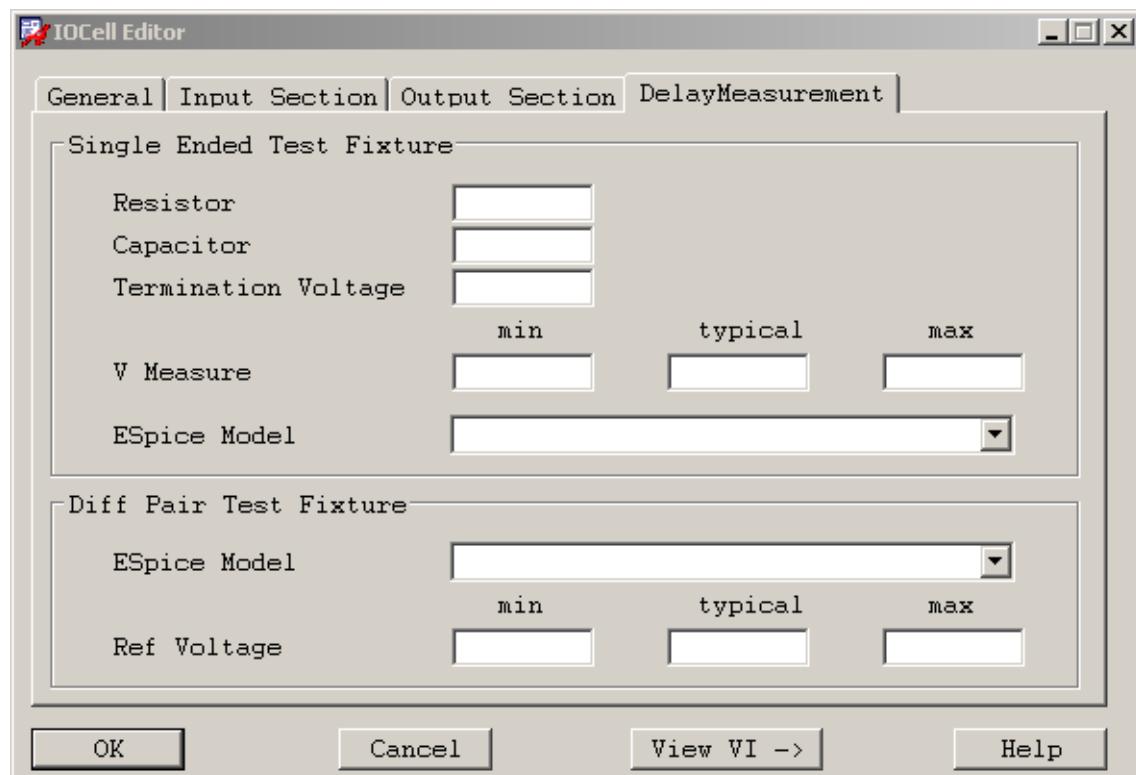
For further details on this tab, or for procedures regarding its usage, refer to the [signal_library](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

Delay Measurement Tab

This tab contains the test fixture data SigNoise uses when measuring buffer delays for rising and falling drivers (output buffers) and to optionally define an ESpice model as the test fixture.

The test fixture values specify the loading conditions under which SigNoise measures buffer delays. Note that the test fixture circuit illustrated in [Figure 3-16](#) on page 119 includes the package parasitics for the specific pin being measured.

Figure 3-15 IOCell Editor - Delay Measurement Tab



Defining Diff Pair and Single-Ended IO Buffer Test Fixture Information for Delay Measurement

You must typically compute buffer delays for single-ended and diff pair outputs in a DML IbisDevice model. You do this by defining a test fixture for each output. The procedures used for defining information for diff pair and single-ended test fixtures is similar but not identical. Additionally, the methods for doing so vary according to whether a test fixture is defined in the selected output model of your library. Your Allegro platform tool searches for test fixture data in the following sequence:

For a single-ended pin:

1. An ESpice model defined for IO cell model associated with the pin. If not found:
2. The parameters for the IO cell model's resistor, capacitor, termination voltage, and Vmeasure.

For a diff pair:

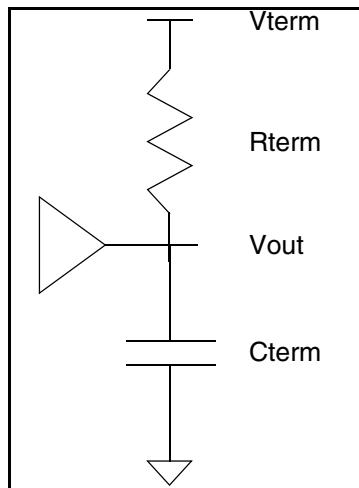
1. The test fixture that is defined in the IBISDevice model. If not defined:
2. The test fixture in the IO cell model for the non-inverting pin of the diff pair. If not found there:
3. The IO cell model for the inverting pin.

If you are defining information for single-ended test fixtures but your output model does not contain that information, you need to populate the *Resistor*, *Capacitor*, and *Termination Voltage*, and *V Measure* fields. If your output model *does* specify a test fixture, use the *ESpice Model* field to select a specific model from a loaded DML library as the test fixture.

If you are defining information for diff pair test fixtures but your output model does not contain that information, you cannot use the model for the test fixture. Instead, you must enter the necessary buffer delay information in the IBIS Device Pin Data > Buffer Delay dialog box.

For further details on this tab, or for procedures regarding its usage, refer to the [signal library](#) command.

Figure 3-16 Delay Measurement Test Fixture Circuit



Editing V/I Curve Data

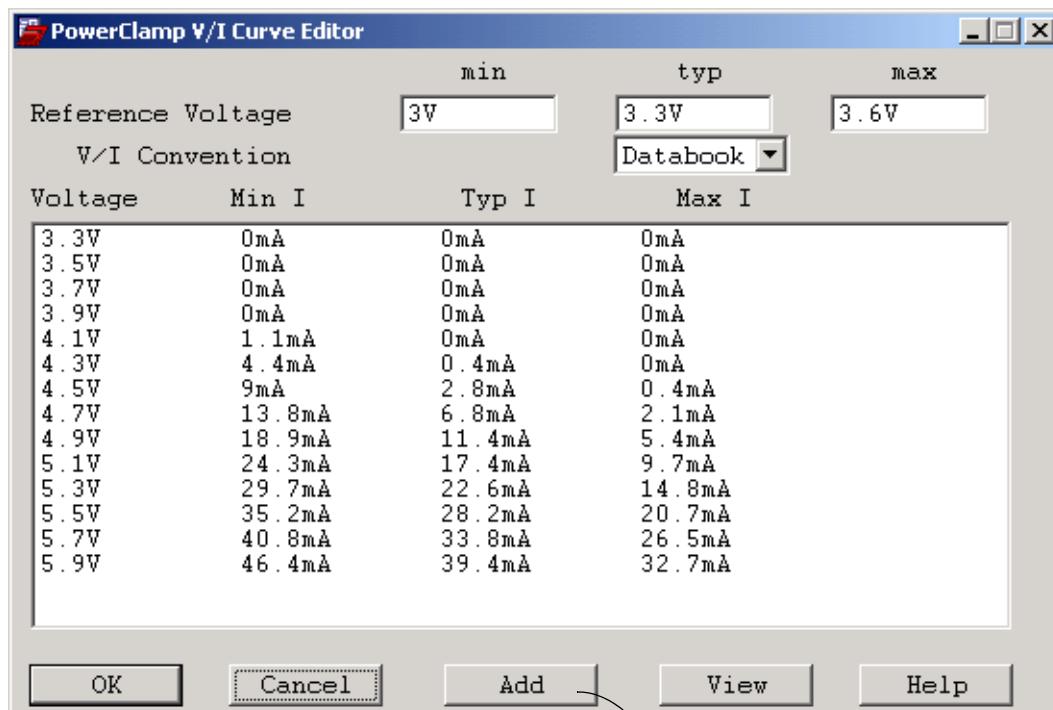
Depending on the behavior for which you intend to edit VI curve data, access the VI curve editor from either the *General* or *Output Section* tabs of the IOCell Editor dialog box.

To access the VI curve editor

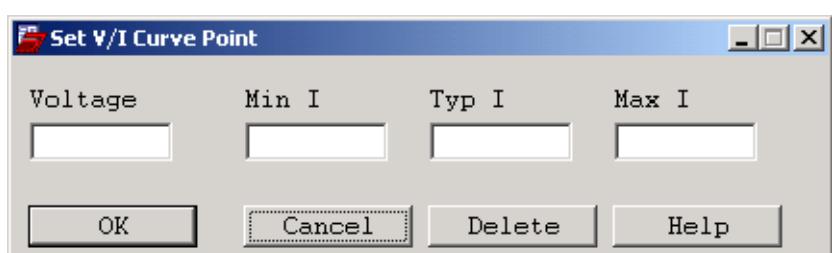
- ▶ From the *General* tab of the IOCell Model Editor, click:
 - *PowerClamp* to edit the V/I curve for a diode clamping high
 - *GroundClamp* to edit the V/I curve for a diode clamping low
- ▶ From the *Output Section* tab of the IOCell Model Editor, click:
 - *PullUp* to edit the high state V/I curve
 - *PullDown* to edit the low state V/I curve

A V/I Curve Editor is displayed for the specified behavior. [Figure 3-17](#) on page 121 shows the Power Clamp V/I Curve Editor dialog box.

Figure 3-17 V/I Curve Editor dialog box



Add, modify, or delete a curve point.



For further details on these dialog boxes, or for procedures regarding editing V/I curve data, refer to the [signal_library](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

Editing V/T Curve Data

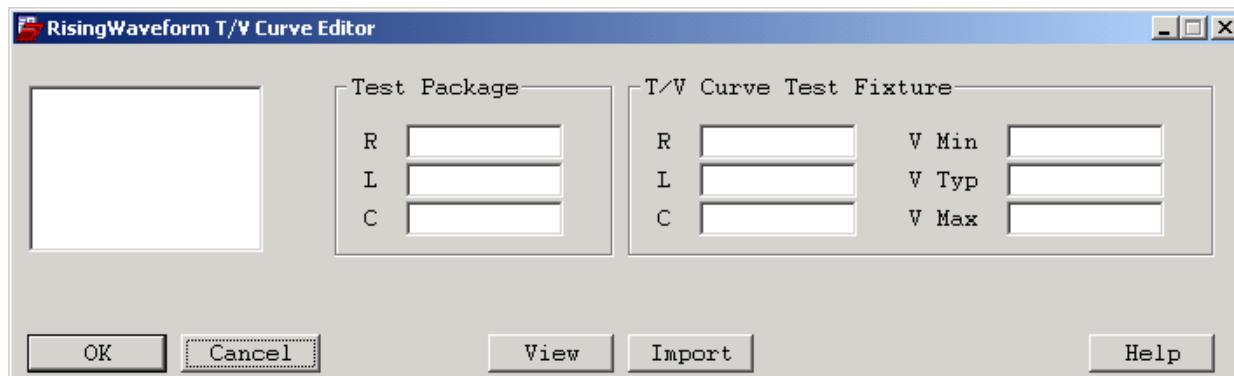
An IOCell model can have any number of V/T curves. Each curve corresponds to one or more measured waveforms for a given test fixture. SigNoise compares this test fixture data against existing circuit conditions, analyzes the waveforms, and then adjusts the IOCell model's switching characteristic to mimic the measured waveforms as closely as possible.

You can access the V/T curve editor from the *Output Section* tab of the IOCell editor. To display a V/T curve editor:

- From the *Output Section* tab of the IOCell Model Editor, click:
 - ❑ *Rise Wave* to edit the rising V/T curve
 - ❑ *Fall Wave* to edit the falling V/T curve

A V/T curve editor is displayed for the specified VT curve. [Figure 3-18 on page 122](#) shows the RisingWaveform V/T Curve Editor dialog box.

Figure 3-18 V/T Curve Editor dialog box



V/T Curve Editor Usage Notes

- Use the *V/T Curve Test Fixture* fields to modify the test fixture for the V/T curve. Select a test fixture and edit the values.
- Use *View* to open the SigWave window and display the waveform for a selected test fixture.
- The SigWave window is displayed with the waveform for the test fixture. See the [*SigWave User Guide*](#) for further details.
- Use *Import* to add a V/T curve from an AWB file to the IOCell model.

If the IOCell model you are editing does not have falling_waveform or rising_waveform sections, you can import the waveform from a carefully designed AWB simulation. Specify the path to the AWB wave file that contains the measured waveform and a name for the text fixture.

For further details on this dialog box, or for procedures regarding the modification of V/T curve data, refer to the [signal_library](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

Editing Espice Device Models

You can edit any existing Espice device model that has been created and added to a Library. If you created the model by cloning (or copying) an existing model, you need to edit the cloned model so that it characterizes the device you are modeling. If you created the model from scratch, it will contain default values that you may want to edit. See “[Introduction to Simulation Models](#)” on page 85 for information on creating device models and adding them to a library.

Use *Edit* in the Model Browser to modify a selected ESpiceDevice model. Your default text editor is opened with the contents of the Espice model.

Editing PackageModels

You can edit any existing PackageModel that has been created and added to a Library. If you created the model by cloning (or copying) an existing model, you need to edit the cloned model so that it characterizes the device you are modeling. If you created the model from scratch, it will contain default values that you may want to edit. See “[Introduction to Simulation Models](#)” on page 85 for information on creating device models and adding them to a library.

Use *Edit* in the Model Browser to modify a selected PackageModel. Your default text editor is opened with the contents of the PackageModel.

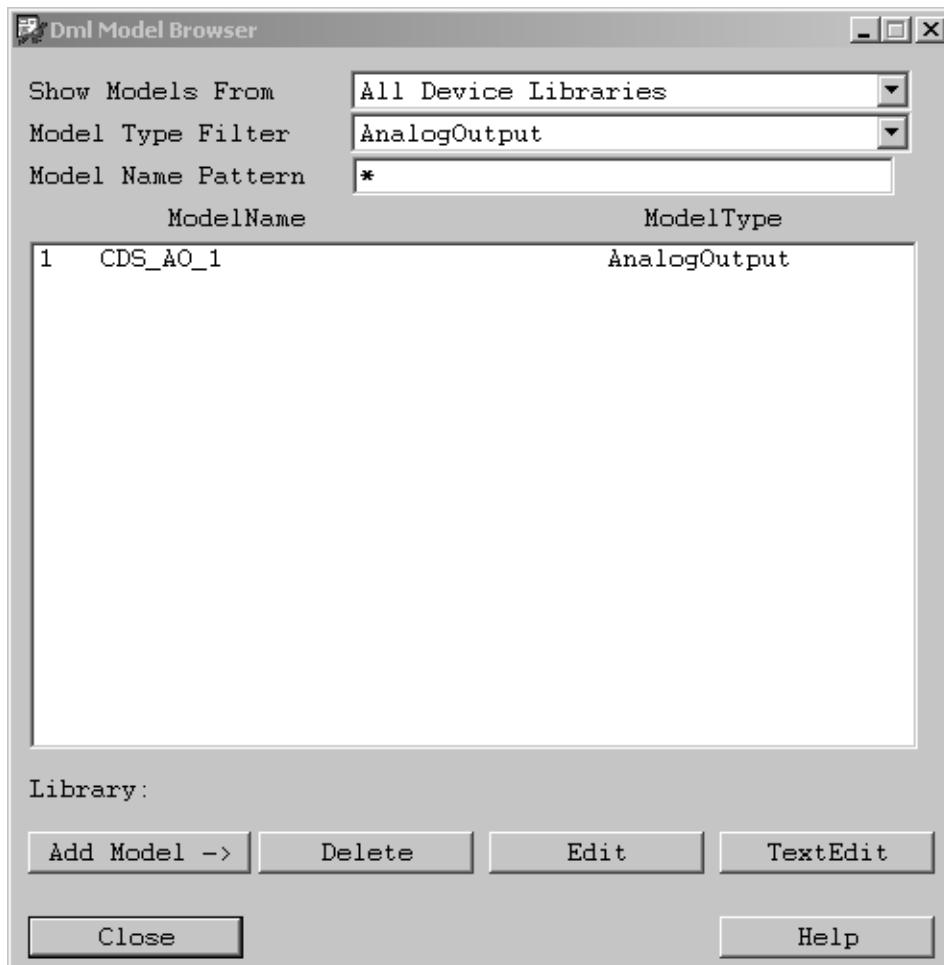
Editing Analog Output Models

You can edit any existing analog output IOCell model that has been created and added to a library. If you created the analog output model by cloning (or copying) an existing model, you need to edit the cloned and renamed copy so that it characterizes the device you are modeling. If you created the model from scratch, it will contain default values that you may want to edit. See “[Introduction to Simulation Models](#)” on page 85 for information on creating device models and adding them to a library.

An analog output model represents a driver pin on an analog device. In analog output models, you specify Cadence Analog Workbench (AWB) wave files for rising and falling edges, pulses, and inverted pulses to describe the behavior of the driver pin.

Analog Output models are modified using the Analog Output Model editor.

Figure 3-19 The Model Browser

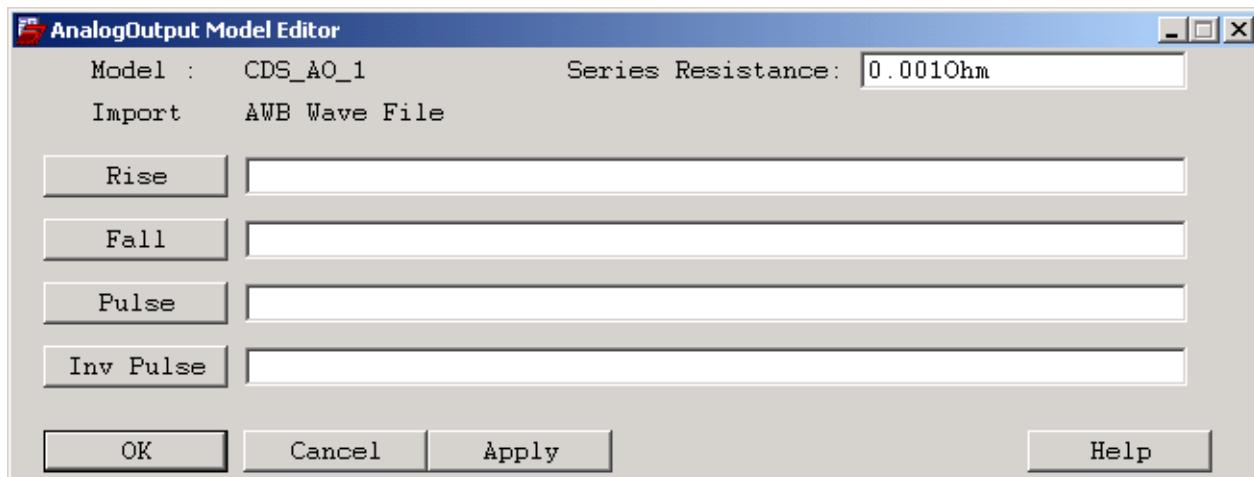


To access the Analog Output Model Editor

- Double-click on an analog output model name in the Dml Model Browser list box.
 - or -
 - Click to select an analog output model from the Model Browser list box, then click *Edit*.
- If necessary, see ["To access the SI Model Browser from the PCB SI"](#) on page 89.

The Analog Output Model Editor dialog box appears as shown in [Figure 3-20](#).

Figure 3-20 Analog Output Model Editor



Using the Analog Output Model Editor you can perform the following tasks.

- Use the *Series Resistance* field to specify a resistance value.
- Use the *Rise*, *Fall*, *Pulse*, and *Inv Pulse* buttons and fields to specify the paths to one or more AWB files and import the files. (Use the button with an empty field to display a file browser.) SigWave displays the selected Analog Workbench file.

Editing and Regenerating Interconnect Models

All interconnect models are written in the Interconnect Description Language (IDL).

To edit an interconnect model

1. Click to select the interconnect model in the Iml Model Browser list box.
2. Click *TextEdit*.

Your default text editor containing the interconnect model is displayed.

Once you have modified the geometry portion of an interconnect model using *TextEdit*, you can regenerate the model's electrical data using the *Solve* button on the Model Browser as explained in the following procedure.

Managing Models Resident in a Design

For improved performance and design portability, device models used for signal integrity analysis are stored directly in the design database (.brd file). Whenever you edit model source within a Device Model Library, you will need to *refresh* the design database in order to incorporate your changes into the models within your current design.

Models are refreshed by searching DML files based on their pre-defined order in the SigNoise library list. The *first* model of the same name and type that is found is used for the refresh. This scheme allows you to move libraries or add new ones as desired.

When a model in the database is re-loaded from a Device Model Library, any Xnets that are affected by the model are updated. Additionally, pin-use codes of all components affected by the refresh are re-checked. If they are inconsistent with the new model, they are updated and a text window is displayed with a report listing the pins whose pin-use codes were updated.

When required, you can also dump device models in the current design to a new signal integrity model library.

You may want to dump the models in order to:

- display the data for the device models in your design.
- generate a new DML to enable the portability of the device models (independent of the design database) that are resident in the current design.

Model Dump/Refresh Dialog Box

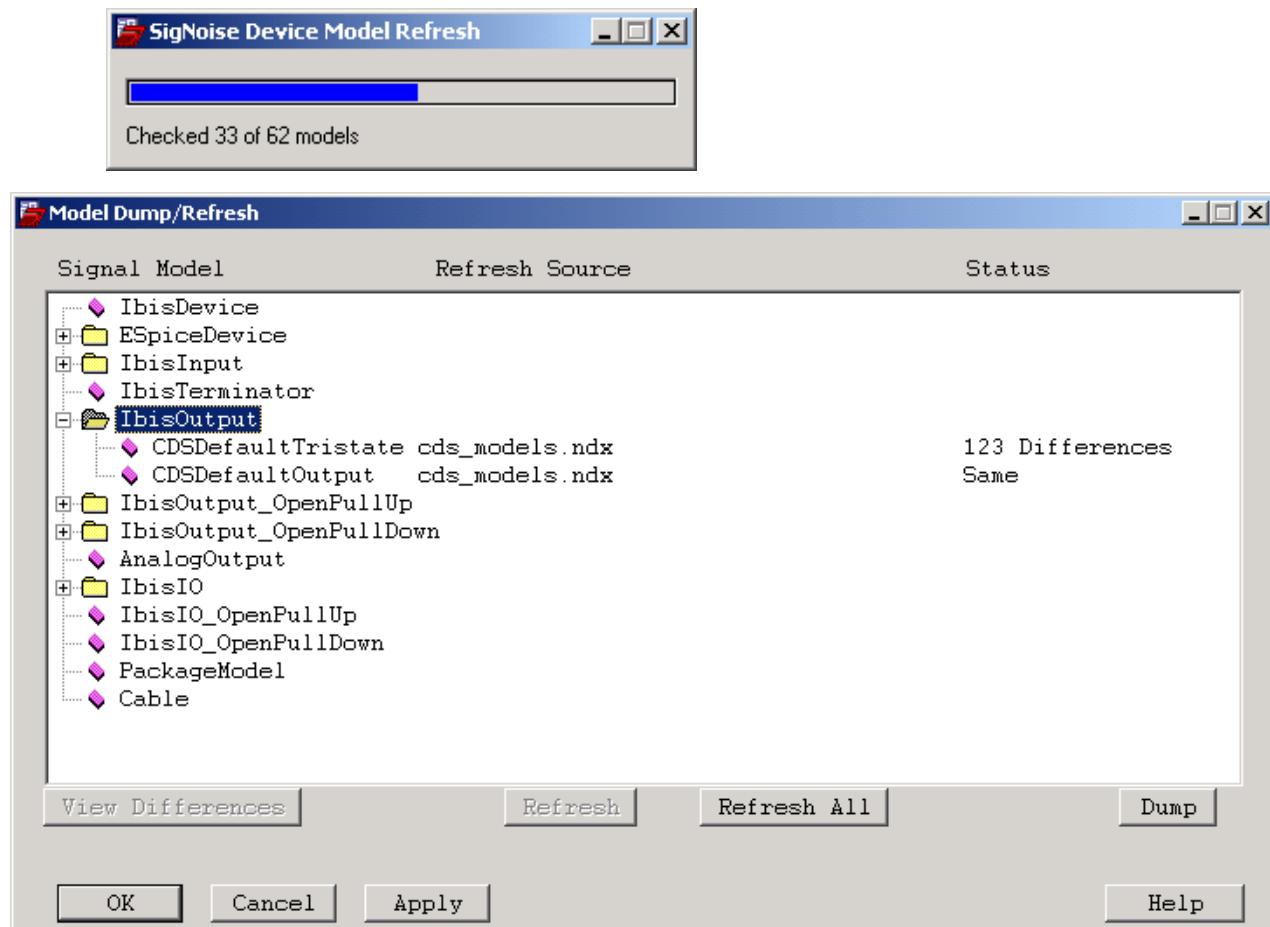
The Model Dump/Refresh dialog box contains functions that let you perform verification and source management operations on the device models in your current design. Once this dialog box is accessed, models are checked against their source while a meter is displayed showing the progress of the task. Upon completion of the check, a list box displays all models resident in the current design.

To access the Model Dump/Refresh dialog box from the PCB SI

- Choose *Analyze – Model Dump/Refresh*

Models are checked, a progress meter is displayed, and the Model Dump/Refresh dialog box appears as shown in [Figure 3-21](#) on page 128.

Figure 3-21 The Model Dump/Refresh Dialog Box



For further details on this dialog box, or for procedures regarding dumping or refreshing models in your design, refer to the [signal_model_refresh](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

About Model Status Messages

There are two possible status messages that can appear against a device model in the Signal Model window. They are described in the following table.

Message	Meaning
<i>Same</i>	The code of the model resident in the current design is identical to the source code of the model in the library. Note: For all models from Cadence standard libraries and Zeelan libraries (which users cannot change), the Status field value will always be <i>Same</i> .
<i><integer> differences</i>	There are <i><integer></i> differences between the code of the model resident in the board and its source code within the library shown in the <i>Refresh Source</i> field.

Reports

Model Refresh Summary

Through the use of the *Refresh* and *Apply* buttons in the dialog box, you can refresh the models in the current design individually and apply changes without having to close (*OK*) the form. When the *Apply* button is selected, a Model Refresh report displays providing verification on the models refreshed thus far. The following figure shows a sample report.

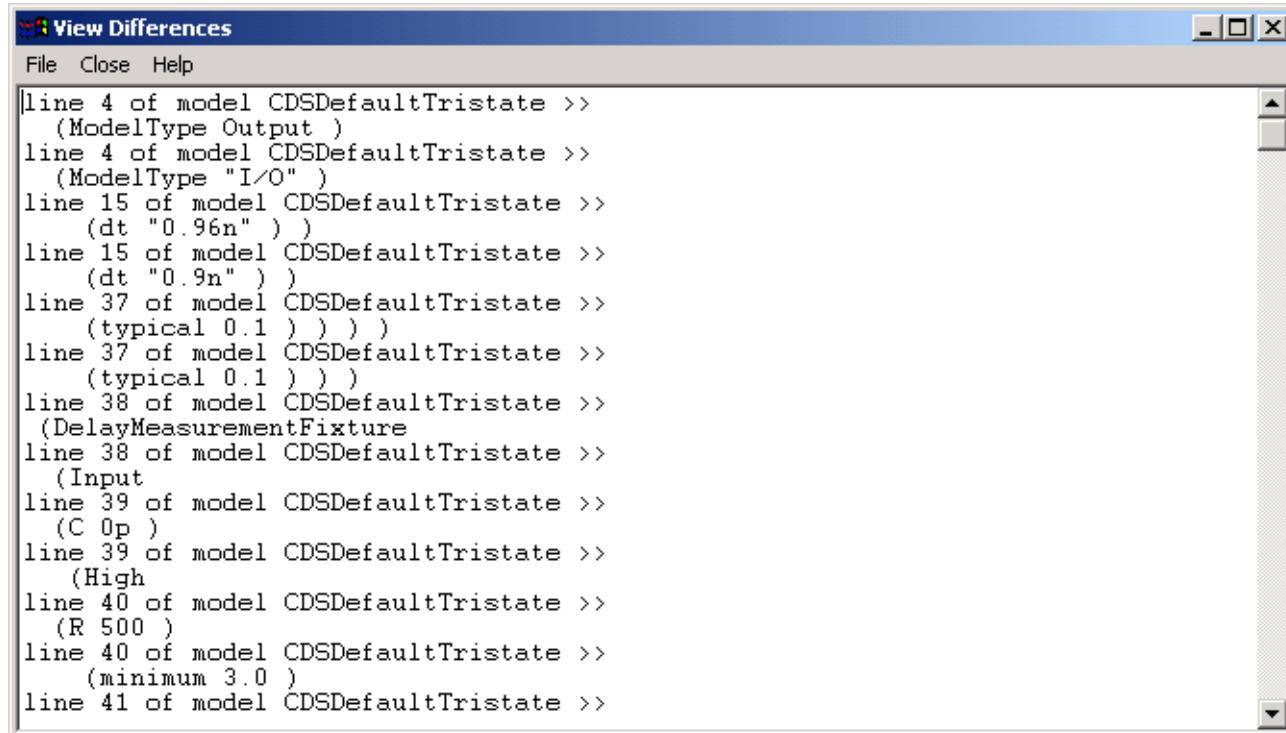
Figure 3-22 Sample Model Refresh Report

Model Refresh Report	
Model	Library Where New Model Found
The following model(s) have been refreshed :	
x7r_0805_1n	power_integrity.dml
npo_0805_82p	power_integrity.dml
npo_1206_560p	power_integrity.dml
npo_0805_680p	power_integrity.dml
x7r_1206_56p	power_integrity.dml
x7r_1206_6_8n	power_integrity.dml
x7r_0805_22n	power_integrity.dml
npo_0805_100p	power_integrity.dml
x7r_1206_100n	power_integrity.dml
x7r_1206_3_3n	power_integrity.dml
x7r_1206_47n	power_integrity.dml
npo_0805_220p	power_integrity.dml
x7r_0805_10n	power_integrity.dml
x7r_0805_150n	power_integrity.dml
x7r_0805_100n	power_integrity.dml
x7r_1206_1_5n	power_integrity.dml
x7r_1206_10n	power_integrity.dml
x7r_1206_47p	power_integrity.dml
npo_0603_1n	power_integrity.dml
npo_0805_2_2n	power_integrity.dml
npo_1206_4_7n	power_integrity.dml

View Differences

When the status of a device model is listed as an integer, there are differences between the model code in the current design and its source. You can check these differences by first selecting the model and then clicking *View Differences* on the Model Dump/Refresh dialog box. The following figure shows a sample report.

Figure 3-23 Sample View Differences Report



The screenshot shows a window titled "View Differences". The menu bar includes "File", "Close", and "Help". The main area displays a text-based comparison report between a model's data in the current design and its source. The report lists several lines of code, mostly starting with "line" followed by a line number and a model identifier like "CDSDefaultTristate >>". The code includes various parameters such as "ModelType Output", "ModelType "I/O"" (with values "0.96n" and "0.9n"), "typical 0.1", "DelayMeasurementFixture", "Input", "C 0p", "High", "R 500", and "minimum 3.0". The report ends with "line 41 of model CDSDefaultTristate >>".

```
line 4 of model CDSDefaultTristate >>
(ModelType Output )
line 4 of model CDSDefaultTristate >>
(ModelType "I/O" )
line 15 of model CDSDefaultTristate >>
(dt "0.96n" )
line 15 of model CDSDefaultTristate >>
(dt "0.9n" )
line 37 of model CDSDefaultTristate >>
(typical 0.1 ) ) )
line 37 of model CDSDefaultTristate >>
(typical 0.1 ) ) )
line 38 of model CDSDefaultTristate >>
(DelayMeasurementFixture
line 38 of model CDSDefaultTristate >>
(Input
line 39 of model CDSDefaultTristate >>
(C 0p )
line 39 of model CDSDefaultTristate >>
(High
line 40 of model CDSDefaultTristate >>
(R 500 )
line 40 of model CDSDefaultTristate >>
(minimum 3.0 )
line 41 of model CDSDefaultTristate >>
```

The report shows a line by line comparison of the differences between the selected model's data within the current design and its source. If no differences are detected, a message is displayed.

Auditing Models and Libraries

The *dmlcheck* Utility

Use the *dmlcheck* utility to check the syntax of one or more library files or models. There are actually several ways to invoke *dmlcheck*. As you recall, the IBIS model editors have DML Check buttons which enable you to check the syntax of new models as they are created. You can check a group of libraries (.dml files) using a command line entry. Also, in some cases, the *dmlcheck* utility is invoked automatically.

Still another way to invoke *dmlcheck* is to use one of the following Library Audit options.

To check device model and library syntax from the PCB SI

1. Run signal lib audit

- or -

Choose *Tools – Utilities – Keyboard Commands*.

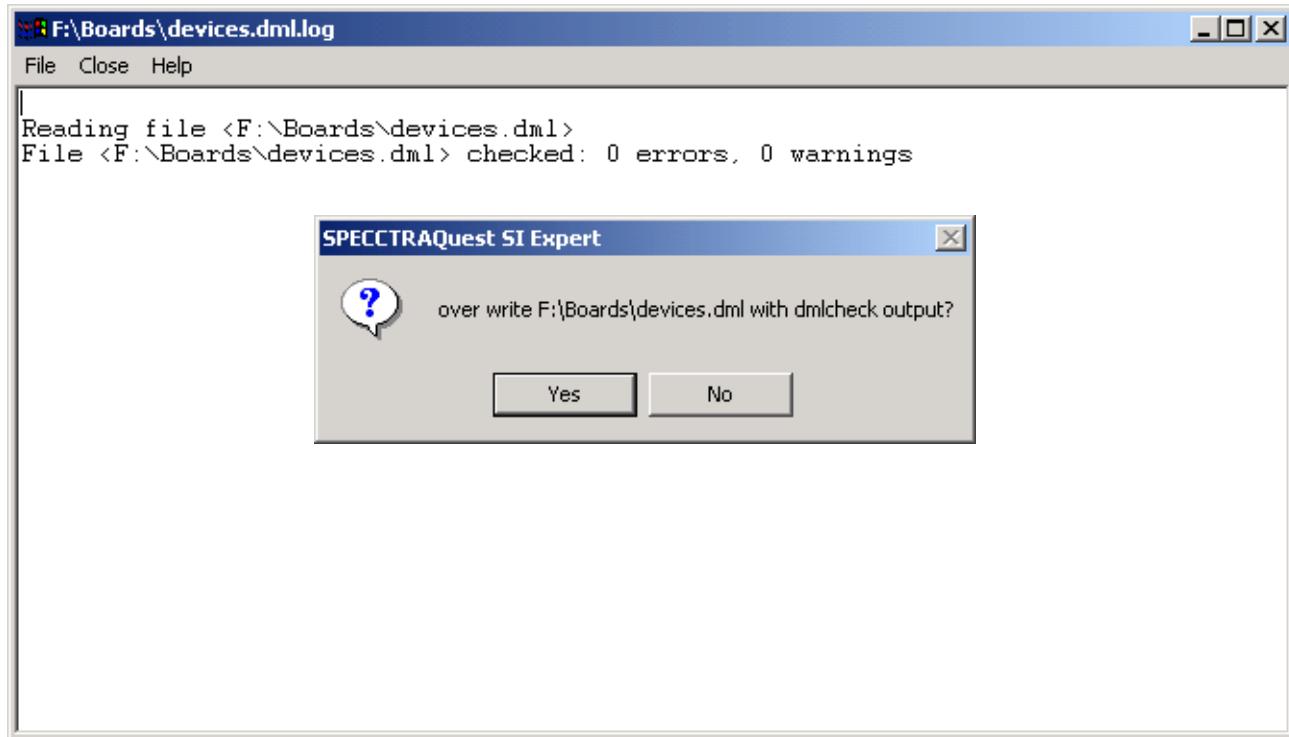
A file browser appears.

2. Select or enter the name of the library (.dml) or library list (.lst) file to be checked, then click *Open*.

Upon completion of the library syntax check, the Dmlcheck Message log window appears with a description of any warnings or errors found within a library or model format.

In cases where *dmlcheck* has modified a curve to fix some problem, a confirm pop-up is displayed asking to overwrite the original library with the output from *dmlcheck*.

Figure 3-24 Sample dmlcheck Log and Overwrite Confirm



To check library syntax from the UNIX command line

- At the operating system prompt, enter the `dmlcheck` command with the following arguments:

```
dmlcheck [options] <library_filename>...
```

dmlcheck Command Arguments

Options	Function
<code>-bufferdelay</code>	Calculate the buffer delay for each IBIS Device pin. You must also specify the <code>-o</code> option with this option.
<code>-curvedir <directory></code>	Creates the specified directory into which each VI and VT curve creates a SigWave waveform file.
<code>-o <extension></code>	Appends the specified extension to the output file created by <code>dmlcheck</code> .

Options	Function
<i>library_filename</i>	One or more device model libraries to be checked.

The *dmlcheck* utility checks the syntax of each file in turn. It prints errors and warning messages as necessary to standard text output and also reports when a file checks out okay.

Dmlcheck Command Line Examples

- The following example checks all library files in the current directory.
`dmlcheck *.dml`
- The following example checks all library files in the current directory, and writes converted data into files with the `.new` extension.
`dmlcheck -o new *.dml`
- The following example simulates to measure BufferDelay, which is stored in the `.new` output file.
`dmlcheck -bufferdelay -o new *.dml`
- The following examples creates a "curves" directory into which is placed a `.sim` waveform file for each V/I and V/T curve.
`dmlcheck -curvedir curves *.dml`

Each waveform file contains the following curves:

- The minimum, typical, and maximum curves in the input file.
- The same three curves after `dmlcheck` fixing.

The files are named with the IOCell name and curve name, separated by an underscore. For example, `CDSDefaultIO_Pullup.sim`. Use SigWave to view the curves.

Model Translation

Translating Third-party Models to DML

You can use translation utilities to translate models from third-party formats to DML files used by SigNoise. The following table shows which translator to use for each third-party model format supported by Cadence.

Model Format to be Translated to DML	Translator to use
IBIS	ibis2signoise
QUAD	quad2signoise
Touchstone	ts2dml

Refer to the [*Allegro SI Device Modeling Language User Guide*](#) for further details on translating third-party models and for information regarding error and warning messages.

Translating Espice Models to Generic SPICE Formats

You can use the *spc2spc* utility to read a named SigNoise netlist and generate SPICE or Spectre formatted output files. This utility has options that allow flattening, node renaming, and ladder network generation. An option to allow the generation of Hspice elements is also available. Enabling this option also causes the creation of Hspice RLG specification files for each different coupled transmission line topology.

Refer to the [*Allegro SI Device Modeling Language User Guide*](#) for further details on translating Espice files and for information regarding error and warning messages.

Model Translation Using SI Model Browser

You translate model files into DML format quickly and more efficiently using the SI Model Browser. To display SI Model Browser:

- Choose *Analyze – Model Browser* in PCB SI.
- or-
- Choose *Analyze – Model Browser* in SigXplorer.

Allegro PCB SI User Guide

Model and Library Management

For further details, refer to the [*Allegro Signal Explorer User Guide*](#).

Transmission Line Simulation

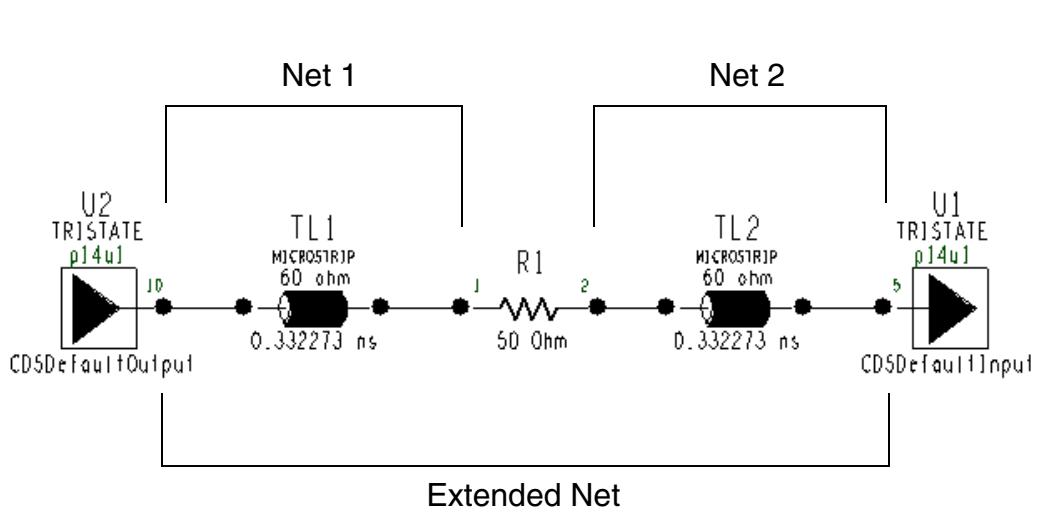
Overview

Transmission line simulation helps you resolve high-speed interconnect problems that often accompany higher density designs, shorter cycle times, higher clock frequencies, shorter rise and fall times, and decreasing ratios of rise time to propagation delay. You can analyze a design for delay, distortion, parasitic, crosstalk effects, and design rule violations. You can review analysis results in both waveform and text report formats.

About the PCB and Package SI Simulator

SigNoise (TLsim) is the transmission line simulator employed by Allegro PCB SI. When you analyze a design for signal integrity, SigNoise develops models of your design and simulates the behavior of one or more *extended nets*. An extended net (or Xnet) is a set of connected and coupled nets.

Figure 4-1 An Extended Net



There are two different ways in which you can use transmission line simulation. You can screen entire designs or large groups of nets for problem areas. Based on the results of these initial analysis, you can then analyze specific individual signals or small groups of signals in order to troubleshoot signal integrity issues.

You can use SigNoise throughout the development of a design:

- During critical component placement.
- After component placement and before you route any connections.
- After you route the critical nets.
- After you route the entire design.

Simulations

Once the interconnect parasitics are derived and the appropriate device models are retrieved and plugged in, SigNoise builds the simulation circuit based on the type of simulation you require. You can distinguish the different simulation types by what is included in the circuit, and how stimulus is applied.

The following types of simulation are available.

- Reflection
- Comprehensive
- Crosstalk
- SSN
- EMI Single Net

Batch Simulation

In addition to performing signal integrity analysis interactively from the user interface, you can also use SigNoise in batch mode. See [Chapter 8, “Analyzing to Generate Text Reports”](#) for more information.

Analysis Results

SigNoise provides its analysis results in the form of:

- ten types of standard analysis text reports.

- custom designed text reports.
- waveforms and accompanying data.
- conductor cross-section diagrams.
- ground bounce movies.

Standard Analysis Reports

Descriptions for the different types of standard analysis text reports generated by SigNoise are provided in the following table.

Table 4-1 Standard Analysis Reports

Report Type	Description
Reflection Summary	Gives delay and distortion data in a concise, summary format.
Delay	Gives propagation delays, switch delays (rising and falling edge), settle delays (rising and falling edge), and reports a pass or fail status for first incident rise and fall and monotonic rise and fall heuristics for selected nets.
Ringing	Gives overshoot and noise margin values for selected nets.
Single Net EMI	Gives essential EMI data for the net in a concise, single-line format.
Parasitics	Gives total self capacitance, impedance range, and transmission line propagation delays for selected nets.
SSN report	Gives noise levels induced on a component's power and ground busses when drivers on that bus switch simultaneously.
Segment Crosstalk	Gives estimated peak and total crosstalk for selected nets. Crosstalk values are derived from closed form algorithms using tables produced from time domain simulation.
Crosstalk Summary	Gives peak and total crosstalk for selected nets in a concise, summary format. Crosstalk values are derived from multi-line simulations.
Crosstalk Detailed	Gives total crosstalk on selected nets for all cases. Crosstalk values are derived from multi-line simulations.

Custom Reports

You can define and then generate text reports with a specific format that you define using the *Custom Report* tab in the *Report Generator* dialog box.

Waveforms and V/I Curves

The waveform data shows the waveform of a signal on a driver-receiver pair. SigWave can display waveforms for all pins in a simulation circuit as well as the VI curves for IOCell models.

Conductor Cross Sections

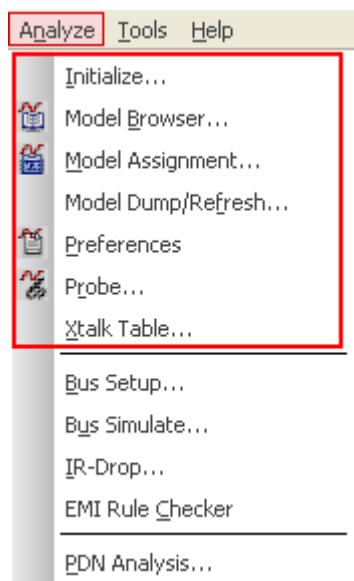
SigNoise generates models for the interconnect in your design. The field solvers generate the parasitic values in the model. The Sigxsect window shows you a three-dimensional view of the interconnect and its parasitic values.

Simulation Setup

Setup Options

To set up for simulation

- Select any of the menu options shown in the following figure from the main menu of your PCB or Package editor.



The following table describes the Analyze menu options relevant for simulation.

Table 4-2 SI/EMI Simulation Menu Option Descriptions

Option	Description
<i>Initialize</i>	Displays the Signal Analysis Initialization dialog box. See “ User Directed Initialization ” on page 143.
<i>Model Browser</i>	Displays the SI Model Browser dialog for working with libraries and models. See Chapter 3, “Working with SI Model Browser.”
<i>Model Assignment</i>	Displays the Signal Model Assignment dialog box for assigning models to components. See “ Auditing Simulation Setup ” on page 157.

Table 4-2 SI/EMI Simulation Menu Option Descriptions

Option	Description
<i>Model Dump / Refresh</i>	Displays the Model Dump/Refresh dialog box for dumping signal integrity models (stored) in the current design to a library or refreshing models in the current design with changes made to their source files in the library. See Chapter 3, “Managing Models Resident in a Design.”
<i>Preferences</i>	Displays the SigNoise Preferences dialog box for specifying the analysis parameters. For details, see Chapter 8, “Setting Simulation Preferences.”
<i>Probe</i>	Displays the Signal Analysis dialog box for detailed analysis. For details, see Chapter 8, “Interactive Simulation.”
<i>Xtalk Table</i>	Displays the Signal Analysis Crosstalk Table dialog box from which you can specify a crosstalk table or create one. See Appendix C, “Crosstalk Timing Windows,” for more information.

Initializing the Simulation Environment

Automatic Initialization

You do not have to perform any manual initialization tasks before you simulate. When you initiate a signal integrity or EMI simulation from your PCB or Package editor, SigNoise automatically takes the following initialization actions.

- Assumes single board analysis mode.
- Opens or writes the `signoise.log` file in the start directory.
- Uses or creates the `signoise.run` directory structure. See [Figure 4-9](#) on page 164.
- Runs the most recently used case and clears simulation data.
- Uses the system configuration last used in the current case.

Each case runs in its own sub-directory (within the `signoise.run` directory). On the first run SigNoise creates the `case1` sub-directory containing the current (active) case. Subsequently, new (consecutively numbered) case sub-directories are created as needed.

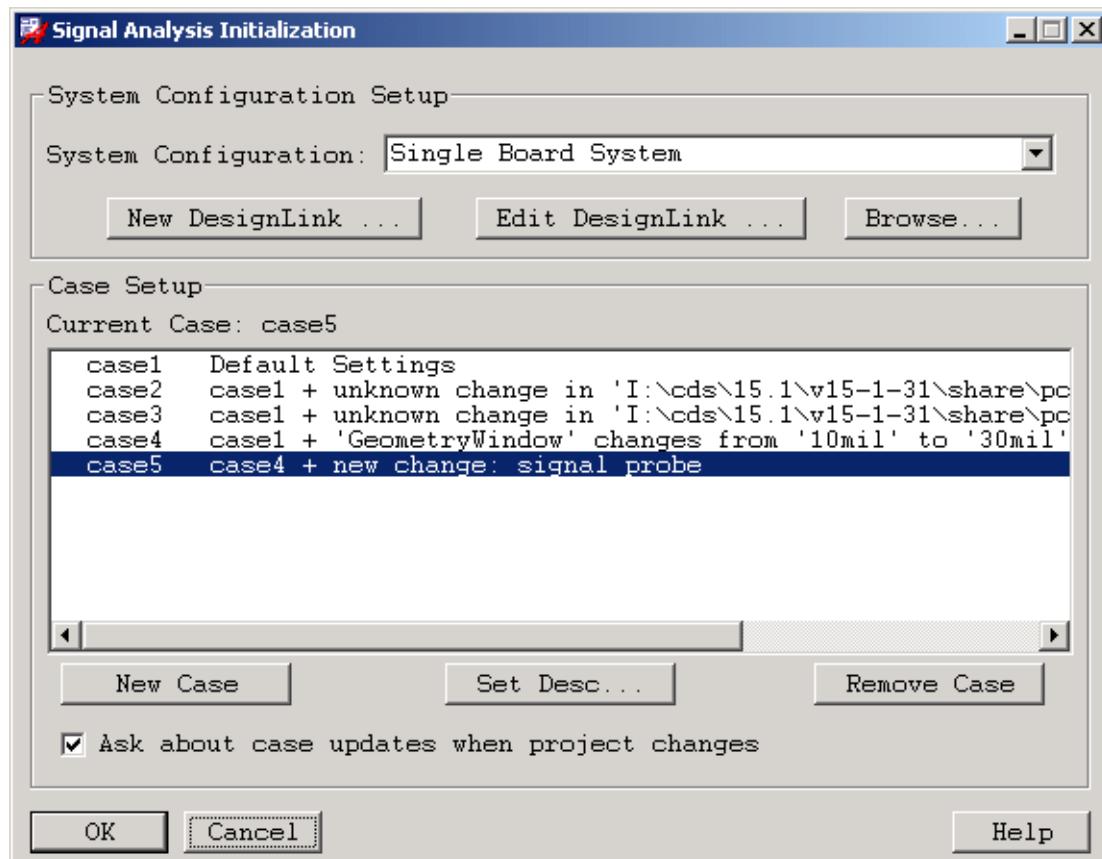
User Directed Initialization

You may also choose to initialize your simulation environment manually.

To initialize SigNoise manually

- Choose *Analyze – Initialize* from the main menu of either your PCB or Package editor.
The Signal Analysis Initialization dialog box appears as shown in the following figure.

Figure 4-2 Signal Analysis Initialization Dialog Box



Using this dialog box you can:

- elect to perform signal integrity analysis on a single board or analyze a multi-board system using a System Configuration model.
- perform case management tasks such as:
 - changing the current case.

- creating a new case or delete an existing case.
- editing the comments associated with a case.

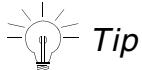
For further details on this dialog box or for procedures regarding simulation initialization, refer to the [signal_init](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

Single Board and Multi-Board Setup

By default, when you perform a simulation operation, SigNoise runs in single board mode on the current design. From the Initialization dialog box you can select to operate SigNoise in multi-board mode. Multi-board mode enables you to analyze a system of more than one printed circuit board using a System Configuration model.

Case Management

Setup and analysis data are partitioned into cases with one case being operated on at a time. You can use the initialization dialog box to manually create and delete cases, and to switch from the current case to another existing case. When you change to a different case using the initialization dialog box, upon confirmation (*OK*), all other dialog boxes are updated to reflect the case data file (`case.cfg`) for the new current case.



By default, *Always ask me about case updates when the project changes* is unchecked, and the *Keep the current case, clearing simulation data* executes in the background. To change this behavior, choose *Analyze – Initialize* and check *Always ask me about case updates when the project changes*. Subsequent parameter changes and simulations will then invoke the *Case Update* dialog box (see [Figure 4-2](#) on page 143), where you can change the case management settings.

Note: You can also access the *Case Update* dialog box by choosing *Analyze – Probe* and clicking *Reports* or *Waveforms*.

When you initiate an operation which would change the case data file, `case.cfg`, in a way that could invalidate simulation data in the current case directory, you are notified of the change by the display of the Update Case dialog box, shown in [Figure 4-2](#) on page 143.

You can choose to:

- create and name a new case based on a copy of the configuration information in the existing `case.cfg` file, plus the proposed change.

Make the new case the current case and begin working there. The new case includes no existing simulation data.
- add the proposed change to the current case, clear all existing simulation data from the case, and continue your work there. This is the default behavior.
- add the proposed change to the configuration of the current case, and continue your work there.

All existing simulation details are retained. Do this only when you are certain that new simulation data, based on the modified configuration information, will be compatible with existing simulation data.

Assigning Device Models

SigNoise uses device models, IOCell (or buffer) models, Espice models, and trace models to create complete simulation circuits for nets in your design. You can assign device models to discrete devices automatically.



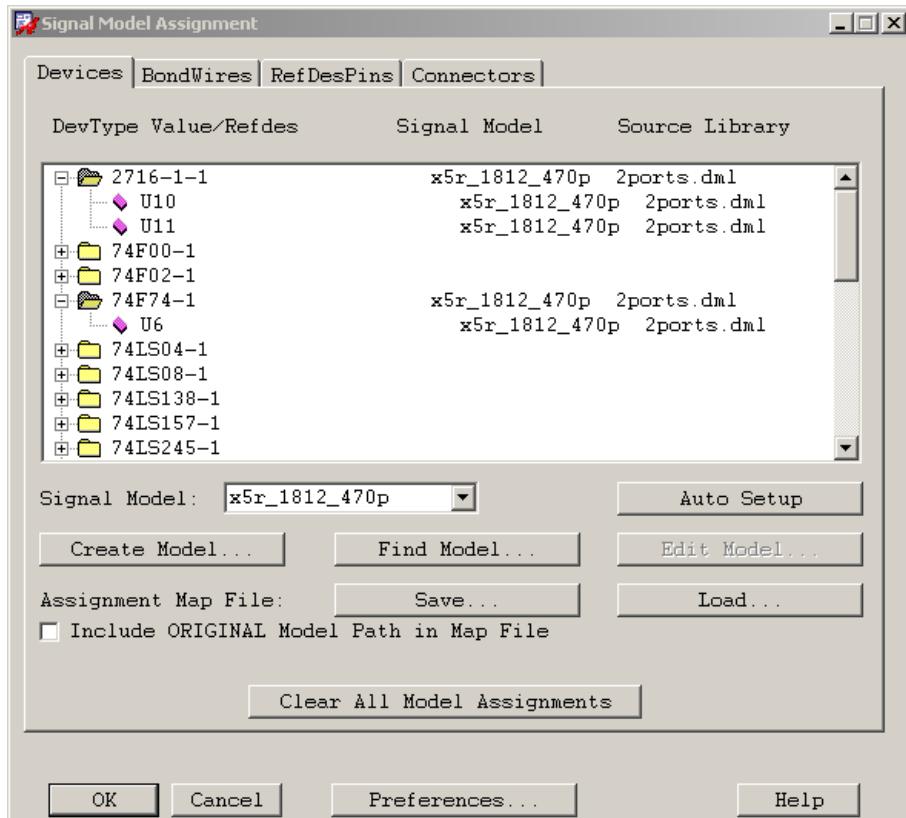
Before performing simulations, choose *Setup – SI Design Audit* to verify that each component has a model assigned to it. For further details, see [Auditing Simulation Setup](#) on page 157. When necessary, you are required to manually assign device models.

To assign device and interconnect models to design objects

- Choose *Analyze – SI/EMI Sim – Model* from the main menu of your PCB or Package editor.

The Signal Model Assignment dialog box is displayed as shown in the following figure.

Figure 4-3 Signal Model Assignment - Devices Tab



Using this dialog box you can perform the following tasks.

- Choose to have SigNoise automatically assign models to capacitors, resistors, and inductors.
- Manually assign models to components and bond wires.
- Disassociate models from design objects.
- Save model assignments to a Model Assignment mapping file or load an existing mapping file.

Devices Tab

Using the *Devices* tab of the Signal Model Assignment dialog box, you can assign device models to components in the design either manually or automatically. Note the *Auto Setup* button in [Figure 4-3](#) on page 146.

Devices Tab Usage Notes

- Use *Preferences* at anytime while using the Signal Model Assignment dialog box to display the SigNoise Preferences dialog box. Through this dialog box, you can change the characteristics of the default device and interconnect models.
- Use *Auto Setup* to automatically assign device models to simple components such as capacitors and resistors using the device type prefix as a reference. In order for automatic model assignment to succeed, components must have reasonable *value* property data in the design database.
- For manual model assignments, either assign a single device model to all components having the same device file or assign individual device models to individual components specified by reference designator. When you specify a model you can enter a model name in the *Signal Model* field or select a model in the Model Browser.

Use *Find Model* to display a set of models appropriate for the selected DevType or RefDes.

- You can also use *Create Model* to invoke a model editor and create a model from scratch. Depending on the *value* property data associated with the component, either the Create Espice Device Model dialog box or the Create IBIS Device Model dialog box is invoked.

For further details on this tab, refer to the [signal model](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

BondWires Tab

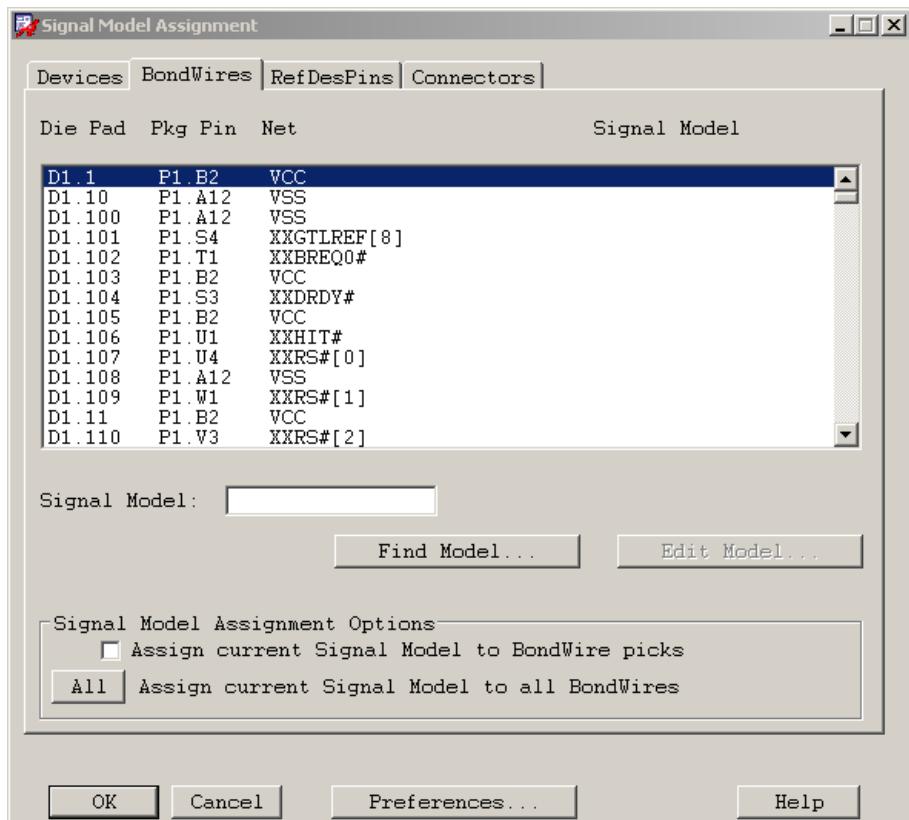
Bond wires are connect lines (clines) on wire bond layers. Use the *BondWires* tab to assign trace models or Espice models to individual bond wire connections in the design or to modify these models. When the Model Browser is open along with the Signal Model Assignment dialog box, the name of a trace model or Espice model selected in the Model Browser also displays in the *Signal Model* field.

Note: Espice models assigned to the bond wires are not used in parasitic reports and DRC.



When assigning an Espice model, map the first Espice subvariety node to the die pin and the second node to the bond finger.

Figure 4-4 Signal Model Assignment - BondWires Tab



BondWires Tab Usage Notes

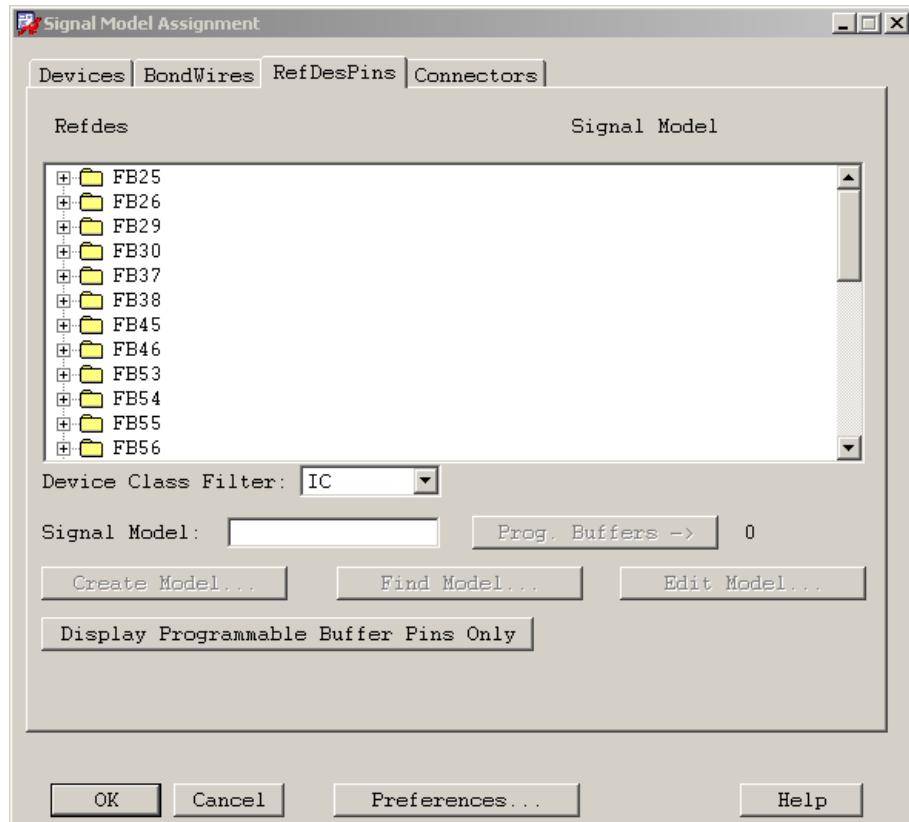
- To manually assign trace and Espice models, use *Find Model* to view and select existing models in the selected interconnect library.

For further details on this tab, refer to the [signal_model](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

RefDesPins Tab

Use the *RefDesPins* tab to assign IOCell models and programmable buffer models to individual pins identified by reference designator. You can also modify existing models during assignment. When the model Browser is open along with the Signal Model Assignment dialog box, the name of the model selected in the Model Browser also displays in the *Signal Model* field.

Figure 4-5 Signal Model Assignment - RefDesPinsTab



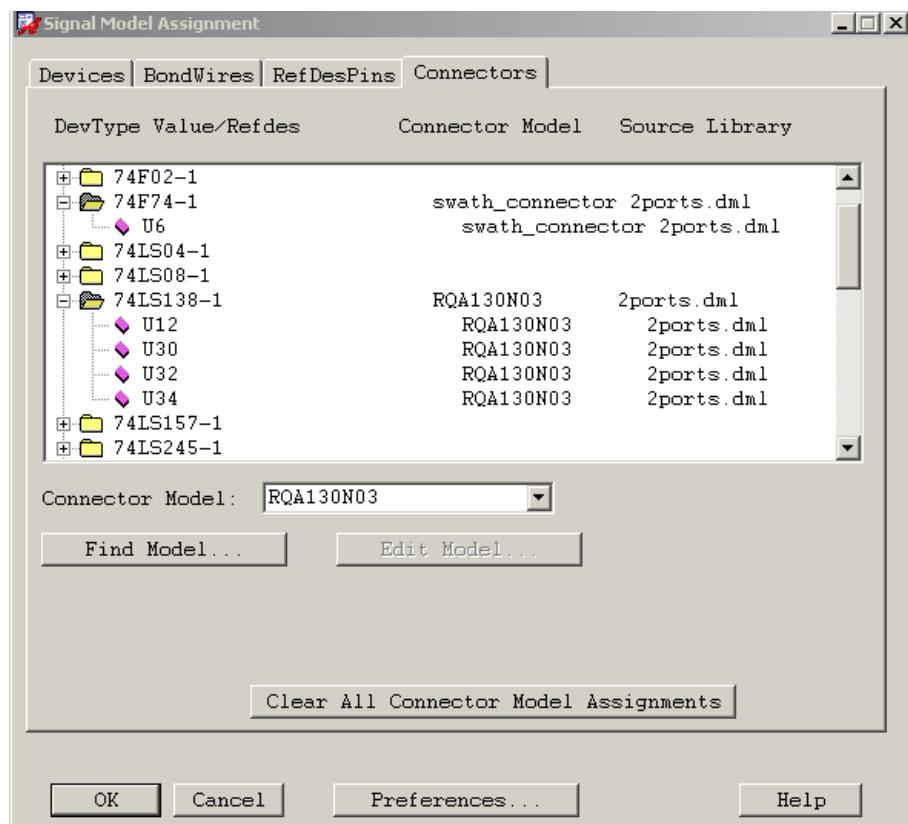
RefDesPinsTab Usage Notes

- If necessary, use the *Preferences* button to modify default data values.
- To assign an existing IBIS Device model, you can enter a model name in the *Signal Model* field, or select a model in the Model Browser. Use the *Find Model* button to display models appropriate for the selected reference designator.
- To activate the *Prog. Buffers* command button, first select a Programmable Buffer model from the working library.

Connectors Tab

Use the Connectors tab to assign coupled connector models to components such as male/female connectors, PCI slots, and other components that connect one design to another.

Figure 4-6 Signal Model Assignment - Connectors Tab



Connectors Tab Usage Notes

- Use the *Connector Model* field to type in an existing connector model name for specified components. Connector model names previously entered in the field appear as selection entries in the drop-down. You can clear individual model assignments by choosing the *No Model* entry.
- To manually assign connector models, use *Find Model* to view and select existing models in the selected device library.
- If necessary, you can modify the chosen model with *Edit Model*.
- To quickly clear all connector models assigned to components in the design, use *Clear All Connector Model Assignments*.

For further details on this tab, refer to the [signal_model](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

Setting Environment Variables

SigNoise references both general and PCB SI simulation-related environment variables that you can set to customize your simulation environment. Simulation-related variables are best handled using a local environment (`env`) file, which is read by PCB SI at start-up. See [Setting Allegro® PCB SI Simulation Variables](#) on page 154.

General System Variables

General system variables can be set in either your local `.cshrc` file (UNIX platforms) or in the System Properties dialog box (Windows platform).

Setting General System Variables on UNIX Platforms

The variables in the following table can be set by adding its command line as an entry in your local `.cshrc` file.

Table 4-3 General System Variables on UNIX Platforms

Variable Name	Purpose	Command Line
EDITOR	Sets the default text editor.	<code>setenv EDITOR 'xterm -e /usr/ucb/vi'</code>
PRINTER	Sets the default printer to use when a print command is given.	<code>setenv PRINTER printer_name</code>

Note: If the EDITOR variable is not set, SigNoise will run `xterm vi` when you text edit a file. You may omit the `xterm` for editors that open a new window of their own.

Setting General System Variables on the Windows Platform

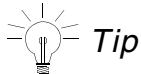
You can set the variables that are described in the following table by adding an entry within the User Variables section of the System Properties dialog box.

Table 4-4 General System Variables on the Windows Platform

Variable Name	Purpose	Value
EDITOR	Sets the default text editor.	editor_application_name
PRINTER	Sets the default printer to use when a print command is given.	printer_name

Allegro® PCB SI Environment Variables

The PCB SI environment variables for both the UNIX and Windows platforms are set in your local env file (`<home>/pcbenv/env`). They apply to the Transmission Line Simulator, and other analysis tools which reference them.



Tip
You are not required to set these environment variables unless you choose to customize your simulation environment. Setting these environment variables is optional.

To determine your local pcbenv directory

1. In your PCB SI or SigXplorer command window, type: `set`

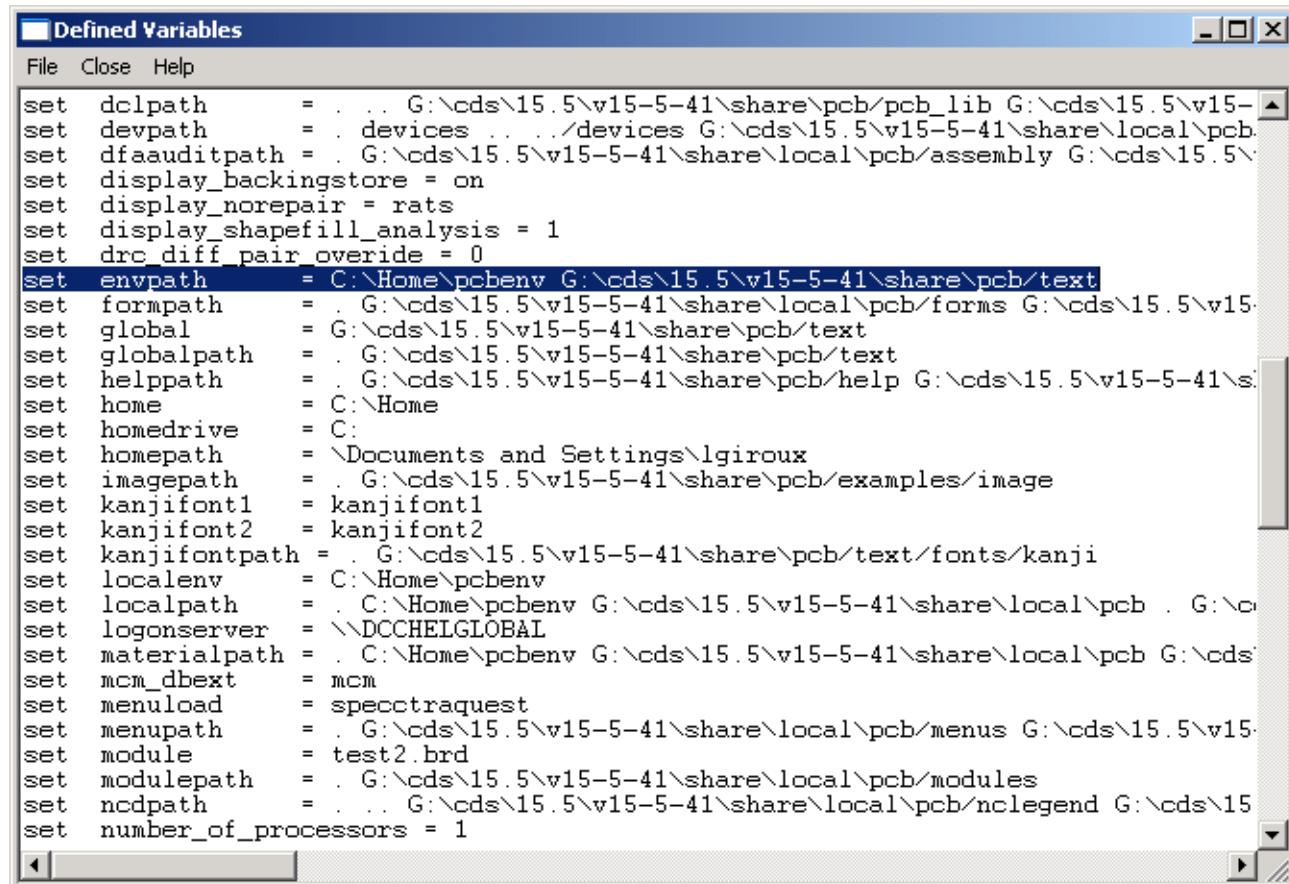
The Defined Variables window appears as shown in the following figure.

2. Check the value of the ENVPATH variable.

Typically, this shows a path to a local directory followed by a path to a common directory in your software installation.

Note: For local environment variable settings, the best strategy is to put them in a local env file (`<home>/pcbenv/env`) and leave the settings in the software installation as-is.

Figure 4-7 Defined Variables Window



The screenshot shows a Windows-style application window titled "Defined Variables". The menu bar includes "File", "Close", and "Help". The main area displays a list of environment variable definitions. Most variables point to paths within the Allegro installation directory (G:\cds\15.5\v15-5-41\share\pcb). The "envpath" variable is set to C:\Home\pcbenv. Other variables include "formpath", "global", "globalpath", "helppath", "home", "homedrive", "homepath", "imagepath", "kanjifont1", "kanjifont2", "kanjifontpath", "localenv", "localpath", "logonserver", "materialpath", "mcm_dbext", "menuload", "menupath", "module", "modulepath", "ncdpath", and "number_of_processors".

```
set dclpath      = . . . G:\cds\15.5\v15-5-41\share\pcb\lib G:\cds\15.5\v15-
set devpath      = . devices ...../devices G:\cds\15.5\v15-5-41\share\local\pcb
set dfauditpath = . G:\cds\15.5\v15-5-41\share\local\pcb\assembly G:\cds\15.5\
set display_backingstore = on
set display_norepair = rats
set display_shapefill_analysis = 1
set drc_diff_pair_override = 0
set envpath      = C:\Home\pcbenv G:\cds\15.5\v15-5-41\share\pcb\text
set formpath     = . G:\cds\15.5\v15-5-41\share\local\pcb\forms G:\cds\15.5\v15-
set global       = G:\cds\15.5\v15-5-41\share\pcb\text
set globalpath   = . G:\cds\15.5\v15-5-41\share\pcb\text
set helppath    = . G:\cds\15.5\v15-5-41\share\pcb\help G:\cds\15.5\v15-5-41\s
set home        = C:\Home
set homedrive   = C:
set homepath    = \Documents and Settings\lgiroux
set imagepath   = . G:\cds\15.5\v15-5-41\share\pcb\examples\image
set kanjifont1  = kanjifont1
set kanjifont2  = kanjifont2
set kanjifontpath = . G:\cds\15.5\v15-5-41\share\pcb\text/fonts/kanji
set localenv    = C:\Home\pcbenv
set localpath   = . C:\Home\pcbenv G:\cds\15.5\v15-5-41\share\local\pcb . G:\cd
set logonserver  = \\DCHELGLOBAL
set materialpath = . C:\Home\pcbenv G:\cds\15.5\v15-5-41\share\local\pcb G:\cds\
set mcm_dbext   = mcm
set menuload    = spectraquest
set menupath    = . G:\cds\15.5\v15-5-41\share\local\pcb\menus G:\cds\15.5\v15-
set module      = test2.brd
set modulepath  = . G:\cds\15.5\v15-5-41\share\local\pcb\modules
set ncdpath     = . . . G:\cds\15.5\v15-5-41\share\local\pcb\nclegend G:\cds\15
set number_of_processors = 1
```

3. Take a look in your `local` ENVPATH directory to see if an `env` file already exists there. If it does not exist, you may want create one. See [To create a local env file](#) on page 153.

To create a local env file

1. Make a `pcbenv` directory under your home directory if one does not currently exist.
2. Copy the `<install>/share/pcb/text/env_local.txt` file to your `pcbenv` directory and name it `env`. This local environment file is read first when PCB SI starts up.
3. Edit this `env` file to add your environment variable settings, aliases, and so on to the end of the file. See [To edit your local env file](#) on page 155 for further details.

Setting Allegro® PCB SI Simulation Variables

You can set the variables in the following table by adding its command line syntax as an entry in your local `env` file.

Table 4-5 Simulation Environment Variables

Variable Name	Function	Command Line Syntax
TOPOLOGY_TEMPLATE_PATH	Sets the path where topology templates are stored.	<code>set TOPOLOGY_TEMPLATE_PATH = /home/sig6/newboard/tops</code>
SIGNOISEPATH	Sets the path to various items used by SigNoise, such as the default libraries.	<code>set SIGNOISEPATH = ~/cds/tools/pcb/text</code>
DISPLAY_NOHILITEFONT	Useful when trying to highlight a trace on a fully routed board. The highlight is a solid color.	<code>set DISPLAY_NOHILITEFONT</code>
LOG_NET_XTALK	Sets all net-to-net results from the Crosstalk Estimator to be logged in the Allegro message window and the <code>signoise.log</code> file.	<code>set LOG_NET_XTALK</code>
SIGSUPPRESS	Suppresses display of the specified types of messages in SigNoise's Message Log window. Follow the variable name with a <i>space-separated</i> list of message types to suppress. These can be warning and/or error messages. Case is not significant.	<code>set SIGSUPPRESS [list_of_message_types]</code>

Table 4-5 Simulation Environment Variables

Variable Name	Function	Command Line Syntax
SIGNAL_RUN_DIRECTORY	Sets SigNoise to use an existing run directory, <code>signoise.run</code> . When the design software is started, SigNoise will use this run directory.	<code>set SIGNAL_RUN_DIRECTORY /home/sig7/projects/signoise.run</code>
SIGNAL_DEVLIBS	Sets the default device libraries that appear in the library search list.	<code>set SIGNAL_DEVLIBS /home/sig6/new.dml /home/sig5/std.dml</code>
SIGNAL_ICNLIBS	Sets the interconnect model libraries that appears in the library search list.	<code>set SIGNAL_ICNLIBS /home/sig6/intrconn1 /home/sig5/intrconn2</code>
SIG_MAPFILE_ORGPATH	Specifies that the original model path is used in the device model assignment map file when it is saved.	<code>set SIG_MAPFILE_ORGPATH 1</code>

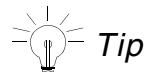
Note: The SIGNAL_DEVLIBS and SIGNAL_ICNLIBS variables do not apply to SigNoise run directories created *before* the variables were set. Also, if you specify a directory, rather than a file, SigNoise will include all libraries (*.dml*) in that directory within the library search list.

To edit your local env file

You can set environment variables simply by editing your local `<home>/pcbenv/env` file using a text editor and then sourcing the file from your command console window.

Note: Do not edit the standard header information in this file.

1. Add environment variable statements to the bottom of your `env` file, preceded by the command `set`, as shown in the [Local env File Example](#) on page 156. Refer to [Table 4-5](#) on page 154 for exact syntax.



A “#” sign is used to precede comment lines and to comment out (turn off) environment variables without removing them from the file.

- 2.** Save the file, then type the following line in your PCB SI or SigXplorer command console window to have the env file read.

```
source <home>/pcbenv/env.txt
```

The new environment variables are now set in your Allegro PCB SI session.

Local env File Example

```
#  
# ALLEGRO local user's environment file  
# # - this indicates a comment  
#  
# read global environment file source $ALLEGRO_INSTALL_DIR/text/env  
#  
# Do not edit the default header above this line.  
#  
# Custom user preferences below  
#  
# enforce lossy Welement algorithms for MGH sims  
set Enforce_Welement_Simulation 1  
#  
# holes in the shield layers are included in delay calculations.  
set USE_ACCURATE_DELAY_CALCULATION  
#  
# system Xnet names can come from any design.  
set SXNET_NAME_FROM_DESIGN = Any  
#  
# define a trapezoidal trace cross-section.  
# value is the acute angle of the sides of the trapezoid.  
# set Trapezoidal_Angle_in_Degree = 60  
#  
# for very fine geometric shapes below 1 mil  
# set BEM2D_BOUNDARY_ELEMENT_SIZE = 5  
#  
# suppress SigWave display of internal i waveforms  
# set SW_HIDE_ALL_I_WAVEFORMS  
#  
# set the default display of T-line delay to be in length.  
set SIGXP_LENGTH_MODE
```

Auditing Simulation Setup

SI Audit Errors Report

The SI Design Audit wizard lets you generate a report about the information discovered during the audit process.

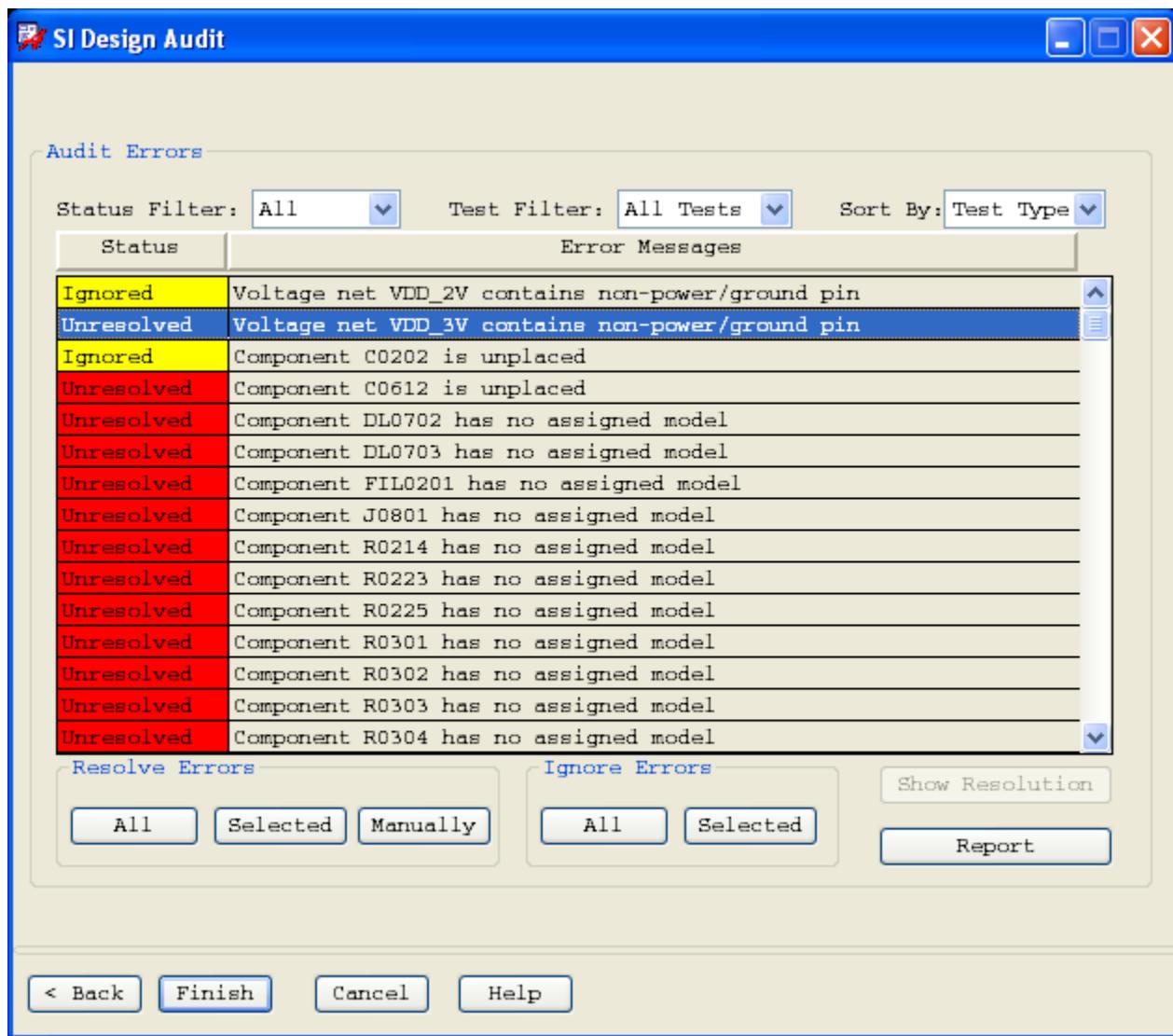
To check your design setup

- Choose *Setup – SI Design Audit* from the main menu of your PCB or Package editor.

Allegro PCB SI User Guide

Transmission Line Simulation

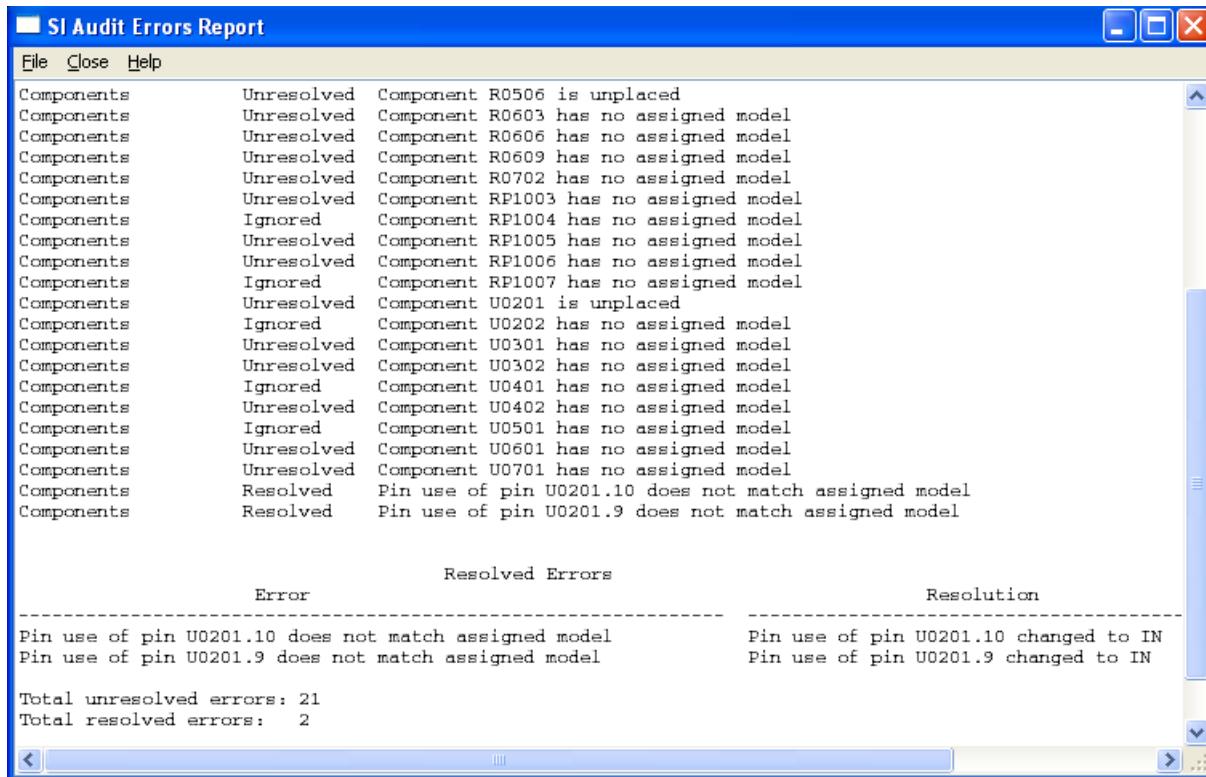
As your design and its libraries are checked, audit errors are listed in the Audit Errors page of the SI Design Audit wizard. This page displays a list of errors encountered, ignored, or resolved during the audit process.



Allegro PCB SI User Guide

Transmission Line Simulation

You can click the *Report* button to generate a report of the errors encountered during audit. The following figure shows the SI Audit Errors Report window with a sample report. Use this window to view and save your setup report as needed.



For information on design audit, see [Allegro PCB and Physical Layout Command Reference: S Commands](#).

Design Audit Checks

To report serious simulation setup problems (Errors), following are checked:

- zero thickness layers in the layer stack.
- unplaced components.
- nets with POWER or GROUND pins, but no VOLTAGE property.
- nets with POWER or GROUND pins or a VOLTAGE property, but no shape or a VOLTAGE_SOURCE pin.
- no VOLTAGE property on any net.
- nets with no drivers or receivers, and no pins attached to a component with an EspiceDevice SIGNAL_MODEL reference.
- C-lines with a SIGNAL_MODEL reference that do not exist in any open interconnect library
- no working interconnect library.
- an active system configuration reference that is not loaded or where a DesignLink model does not exist in any open device library.
- Default IOCells that do not exist in any open device library
- components with a SIGNAL_MODEL reference that do not exist in any open device library.
- model version discrepancies.
- referenced device models that do not pass dmlcheck (Audit Report will list problem models, but actual errors will appear in SigNoise log window).
- pin signal_model parameters in IBISDevice pin map do not match Allegro pin use.
- Allegro component pins not found in IBISDevice pin map (other than NC pins).
- components with the TERMINATOR_PACK property not assigned an EspiceDevice SIGNAL_MODEL property.
- shapes on PLANE layers with no VOLTAGE property on a net.
- layers with improper material for a given layer type.
- nets with improper differential pair connections like a non-inverting driver that is driving an inverting receiver.

- nets with the DIFFERENTIAL_PAIR property and no differential pair signal models on pins, or vice versa.
- DISCRETE components with the wrong PINUSE property.

To find setup problems that may hinder accuracy (Warnings), SigNoise checks for:

- default settings in the layer stack.
- wire bond layers that do not have the SIGNAL_MODEL property attached to clines.
- components that have no SIGNAL_MODEL property.
- nets with a DC VOLTAGE property.
- default IOCELL models that are not set.
- PLANE layers that have no name assigned.

SigNoise also reports:

- layer stack information.
- number of nets and components.
- assigned models, including the library file.

Simulation Run Directory Structure

Creation Scheme

SigNoise is the Allegro® SI simulation engine. It creates the run directory structure required for simulation when it is needed. In most cases, the parent directory of the SigNoise run directory structure is the directory *from where you start* your design software. Your start directory contains a log file (`signoise.log`) which contains information about libraries which are currently loaded as well as a run directory (`signoise.run`).

Your current working directory is always tracked by SigNoise. If you open a design in a different directory, SigNoise switches focus to the new directory, adjusts settings accordingly, and uses the local `signoise.run` directory (if one exists). Otherwise, a new run directory structure is created there and a message is issued announcing the action.

Note: You can use the `SIGNAL_RUN_DIRECTORY` environment variable to establish a *default* run directory. See the [Simulation Run Directory Structure](#) figure on page 164 for more information.

[Figure 4-8](#) on page 163 depicts the files and sub-directories that are contained within a typical working directory.

Figure 4-8 Working Directory Structure

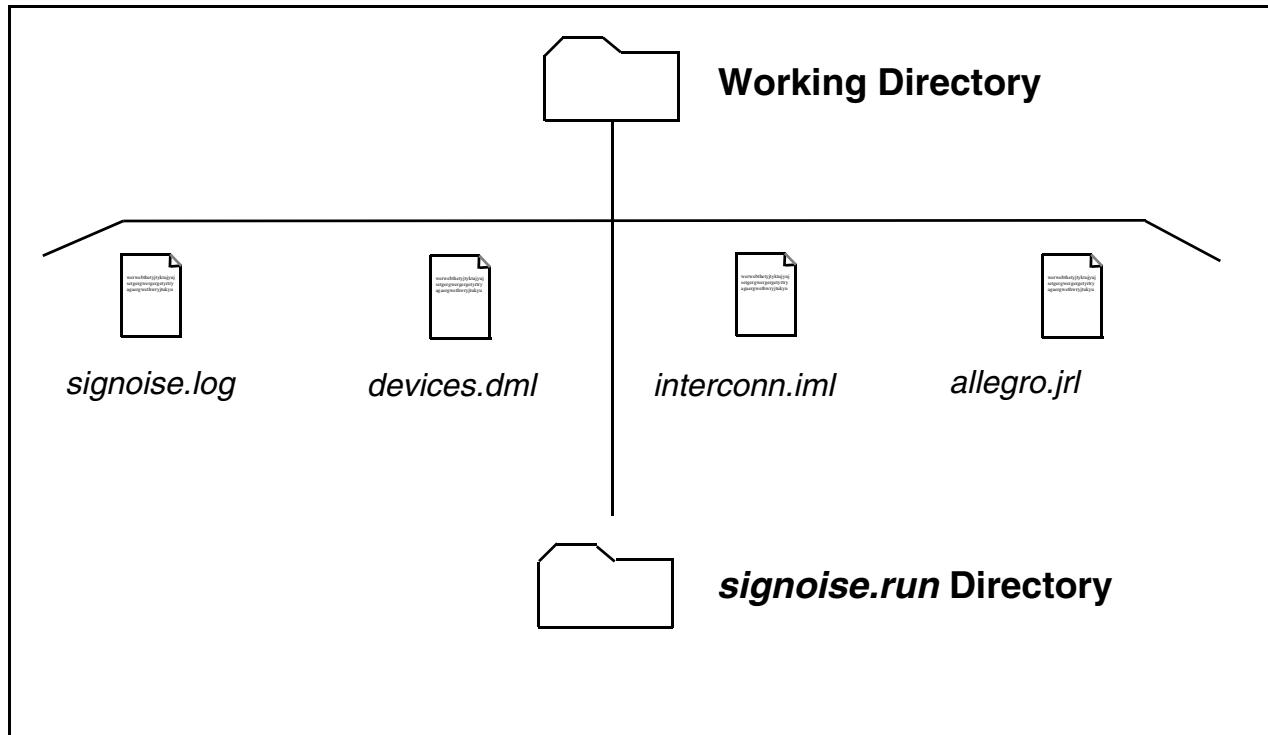
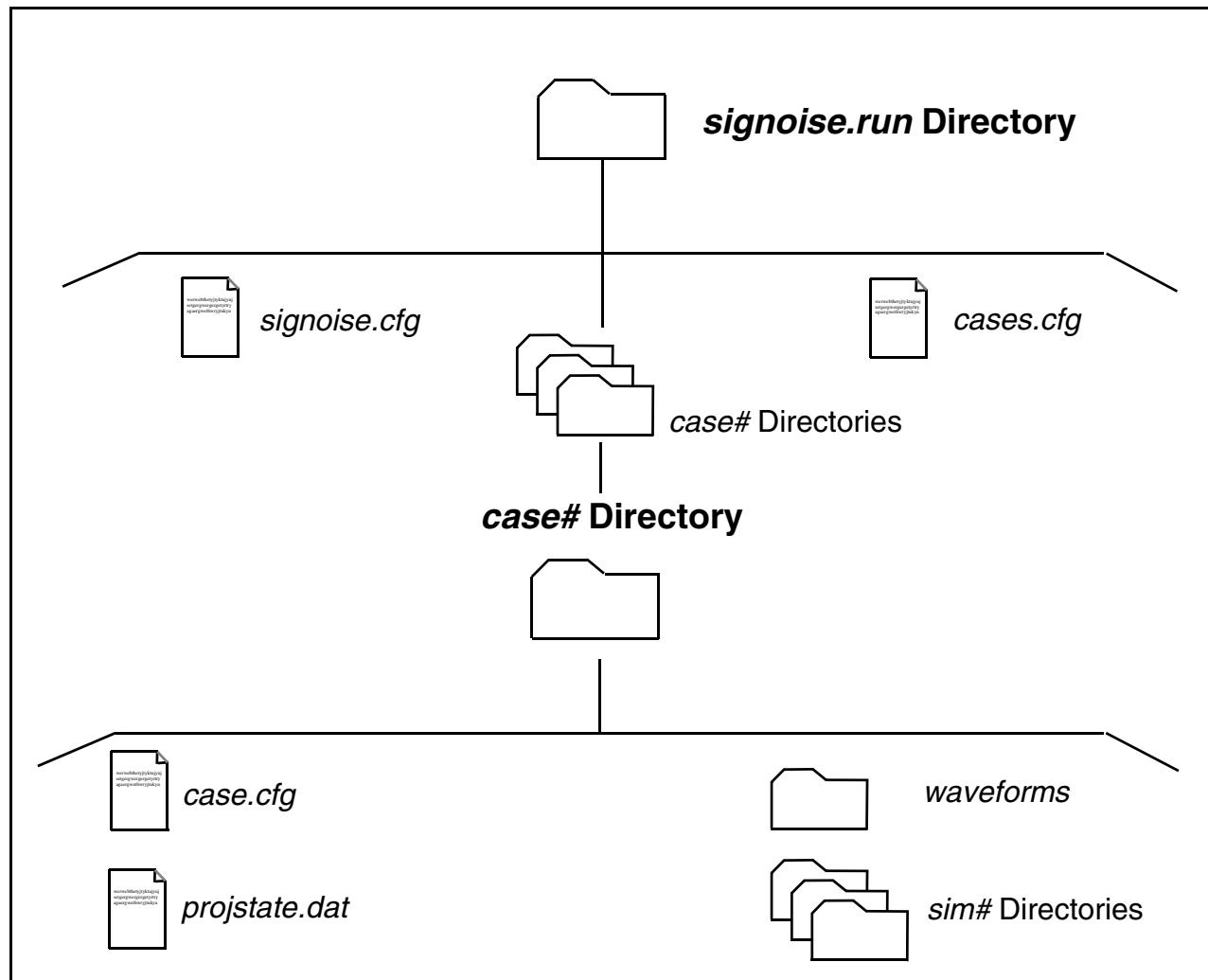


Figure 4-9 on page 164 depicts the files and sub-directories that are contained within the `signoise.run` directory.

Figure 4-9 Simulation Run Directory Structure



signoise.run Directory Contents

- **signoise.cfg** - Contains the configuration information that is general and common to all simulations. For example, information on whether a System Configuration model is active and the name of the current device model library is included here.
- **cases.cfg** - Contains a listing of the case directories and the descriptive text string associated with each case.

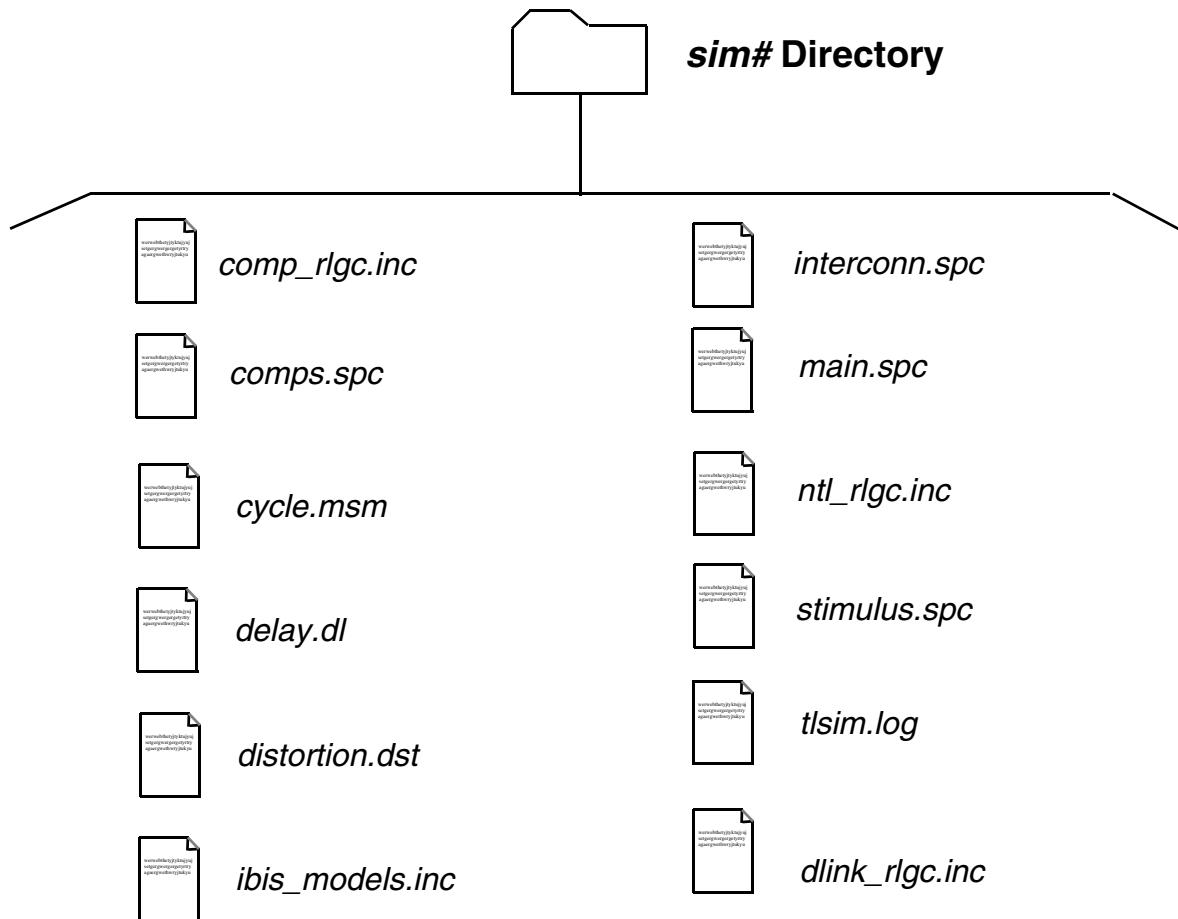
Case# Directory Contents

SigNoise creates a *new* case directory whenever one is required. You also have the option to create case directories manually and to determine the characteristics of the cases that SigNoise creates. See “[Case Management](#)” on page 144 for more information.

- **case.cfg** - Contains the configuration information that is specific to that particular case. This information includes, for example, the text string describing the case and the stimuli to apply when a simulation for this case is run.
- **projstat.dat** - Lists timestamp data for each .brd file in the system as well as each .dml loaded. Use this information to determine when these files have been modified.
- **sim# Directories** - These sub-directories (sim1, sim2, and so on) are created only when Save Circuit Files is on. They contain all input and output files for a specific simulation, except for the .sim waveform files that get saved to the waveforms directory.
- **waveforms** - If you elect to save circuit files (in either the Report Generator or the WaveForm Simulation dialog boxes), SigNoise saves SPICE files in the simulation directories. If you elect to save waveforms (in the Report Generator), SigNoise saves the waveform files corresponding to SigNoise runs (sim1.sim, sim2.sim, and so on) in the waveforms directory.

When you perform an EMI emissions simulation, SigNoise saves both the time voltage waveform files corresponding to SigNoise runs (sim4.sim, sim5.sim, and so on) and the files containing the emission spectrum in the frequency domain (sim4_emi_db.sim, sim5_emi_db.sim, and so on) in the waveforms directory.

Figure 4-10 Simulation Run Directory Structure (cont.)



sim# Directory Contents

- ***comp_rlgc.inc*** - Describes the package parasitic values of package model RLGC matrices.
- ***comps.spc*** - Describes component sub circuits, and power and ground values of the simulation. It also contains package parasitics when the package model contains spice sub circuits.
- ***cycle.msm*** - Lists the simulation results for each node to measure. It contains delay and distortion data.
- ***delay.dl*** - Lists the delay simulation results.
- ***distortion.dst*** - Lists the distortion simulation results.

Allegro PCB SI User Guide

Transmission Line Simulation

- ***ibis_models.inc*** - Describes the parameter values of IbisIOCell model definitions.
- ***interconn.spc*** - Describes sub circuits of interconnect model definitions.
- ***main.spc*** - This is the main SPICE file calling the other SPICE sub circuits.
- ***ntl_rlgc.inc*** - Describes the parameter values of trace model RLGC matrixes.
- ***stimulus.spc*** - This SPICE file describes the stimulus input.
- ***tlsim.log*** - The log file for the Cadence proprietary SPICE simulator.
- ***dlink_rlgc.inc*** - Describes the parameter values of cable model RLGC matrixes.

Simulation Message Window and Log File

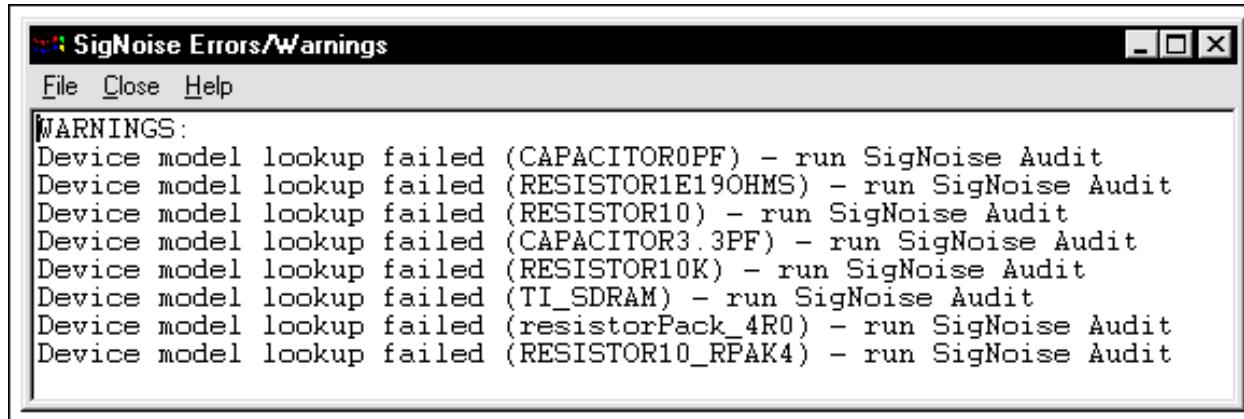
Warning and error messages generated during simulation are displayed in the SigNoise Errors/Warnings window. The window opens when the first message is generated and the display list expands for each subsequent message. When you perform the next simulation action, all existing messages are cleared and the window is closed.

The SigNoise Errors/Warnings window displays warnings below error messages. As the simulator generates new messages, they are *added to the top* of each list. Duplicate messages are filtered out of the list. The associated `signoise.log` file logs all informational messages, warnings, and errors.

Note: You can suppress the display of SigNoise Errors/Warnings window by setting the `SIGSUPPRESS` environment variable. See “[Simulation Run Directory Structure](#)” on page 162 for more information.

The SigNoise Errors/Warnings window with a sample list of warnings is shown in [Figure 4-11](#).

Figure 4-11 The SigNoise Errors/Warnings Window



Floorplanning

Introduction

Floorplanning in PCB SI allows you to bring layout decisions to the forefront of the design cycle. SI provides a physical view of your design and allows you to do system-level topology and floorplanning exploration. It functions as both a pre-route editor and post-route analysis tool that enables you to quickly develop and verify net topologies and constraints for the high-speed circuits in your design.

Using SI you can:

- perform database setup requirements for high-speed circuit simulation. See [Setting up the Design](#) on page 61
- identify rooms for critical component placement. For example, you may want to separate analog components from digital components by confining the analog components to a specific room area on the board. See [Room Outlines](#) on page 174.
- evaluate the effects of different placement strategies on design behavior.
- mock-up, simulate, and analyze logic at the board-level. See the example [Drawing Logic Scenarios at the Board Level](#) on page 182.
- develop logic and constraints for high-speed circuits from scratch without a netlist.
- ensure quality by implementing re-use of design elements and critical components. You can do this easily by importing technology files, board geometry, and setup data from other designs. For further details, see [Importing Setup Data](#) on page 178.
- perform test routing using proposed electrical constraints to ensure high-speed design rules are achievable before passing them on to the Layout Designer.
- ensure design compliance by performing quick post-route verification checks using an extensive set of post-route analysis capabilities. Checks are performed directly from the board database. Results are available with comprehensive waveform viewing and reports. See [Chapter 8, “Post-Route Verification.”](#)

PCB SI Floorplanning focuses on two main design tasks.

- Board Setup
- Logic

Board Setup

Board setup involves defining or editing one or more of the following.

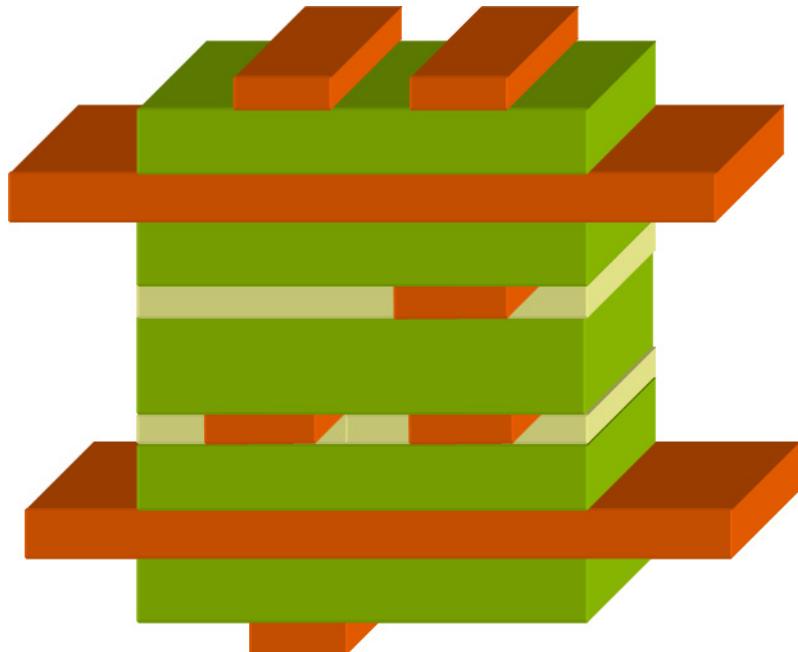
- Cross-section Stackup and Materials
- Board Outline
- Room Outlines
- Plane Outline
- Keepouts

Cross-Section Stackup and Materials

The Cross-Section Stackup Editor presents the ordered layers of your board in a worksheet-like format. Each layer is identified in the worksheet within the *Etch Subclass* column. The attributes of each etch subclass are presented in the cells of the worksheet. Information regarding layer type, thickness, spacing, electrical characteristics, and same layer or layer-to-layer differential impedance are listed. You can define this information initially using the [SI Design Setup](#) command. However, as the Design Engineer, you should check and (if necessary) edit the stackup before commencing with signal integrity simulation on your design.

When you simulate, the etch in the design is processed through a field solver to generate models. Therefore, a proper cross-section is necessary for accurate post-route analysis and for providing information for interconnect in topologies extracted into SigXplorer. The cross-section also determines the propagation velocity of a signal as well. This propagation velocity determines the value of delay when it is expressed as a constraint value in terms of time rather than length. The use of percentage in propagation delay rules is also based on the propagation velocity of the signal.

Figure 5-1 Conceptual View of a Layout Cross-Section



To access the Stackup Editor

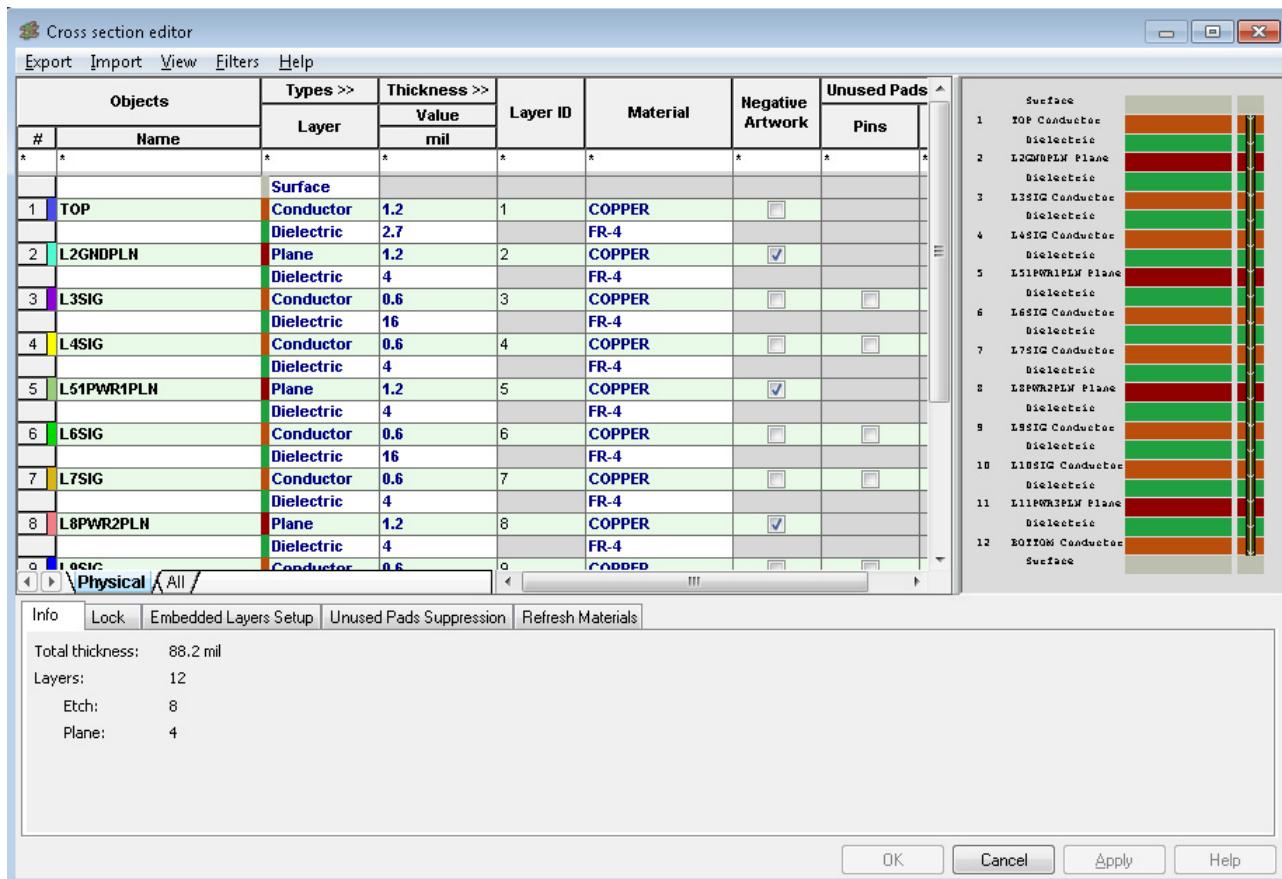
- In SI, choose *Setup – Cross-section*.

The Cross section editor appears as shown in [Figure 5-2](#) on page 172.

Allegro PCB SI User Guide

Floorplanning

Figure 5-2 Cross section editor



You can modify most attributes by entering a new value in the appropriate cell. Exceptions to this include the extreme outer layers, which have a fixed name called SURFACE and no definable attributes, and the extreme outer CONDUCTOR layers, which have a fixed name of TOP and BOTTOM. You cannot change the name TOP and BOTTOM but you can change the attribute values on those layers.

When you change the value of an attribute, other attributes may be re-calculated. For example, if you change the value of the *Line Width*, the *Impedance* changes as well.

The Materials Editor

The Materials Editor presents materials that are currently defined in your Materials file. Each row represents a single material with columns representing the various attributes of the material. You can resize the dialog box to fully display an extended range of materials available in the Materials file (the default size displays twenty materials). It is also possible to reduce the size of the dialog box. However, the scale worksheet itself remains fixed.

Allegro PCB SI User Guide

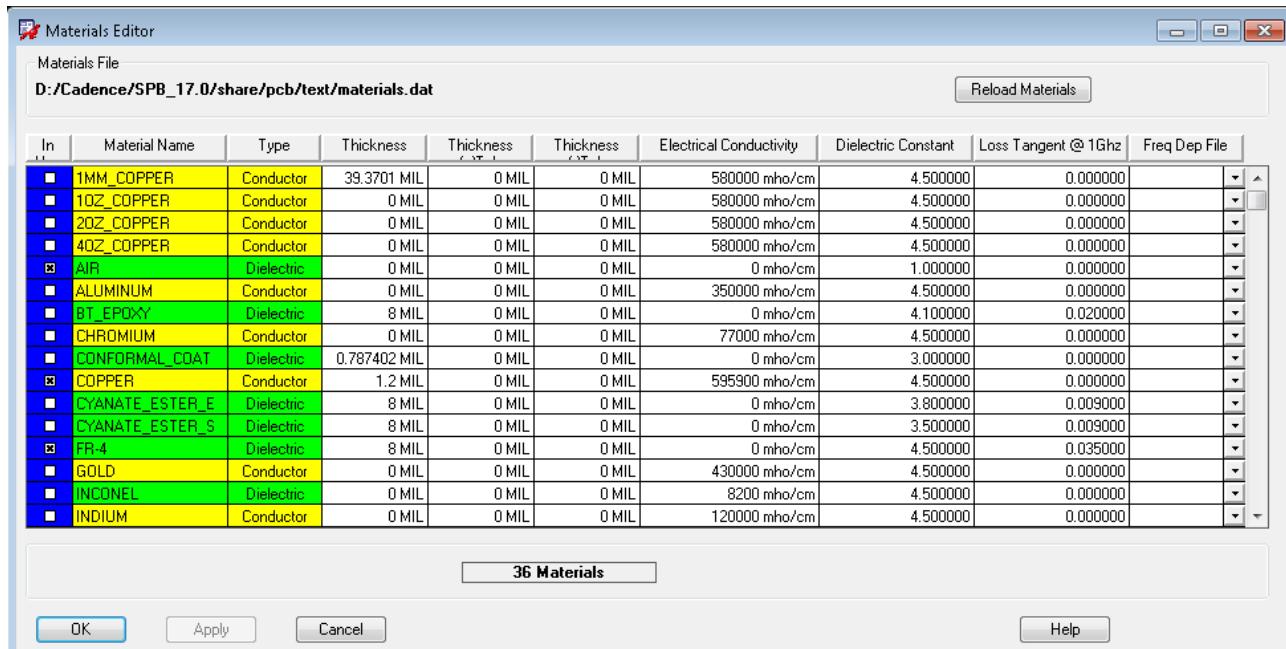
Floorplanning

To access the Materials Editor

- In SI, choose *Setup – Materials*.

The Materials Editor appears as shown in [Figure 5-3](#)

Figure 5-3 The Materials Editor



The Materials Editor automatically displays default values that are in either the `materials.dat` file (PCB SI) or the `mmcmmat.dat` file (IC Packaging Design). These are read-only files provided by Allegro that contain the most common industry fabrication materials. By default, they are located in the following directory within your installation hierarchy.

`$ALLEGRO_INSTALL_DIRECTORY/share/pcb/text`

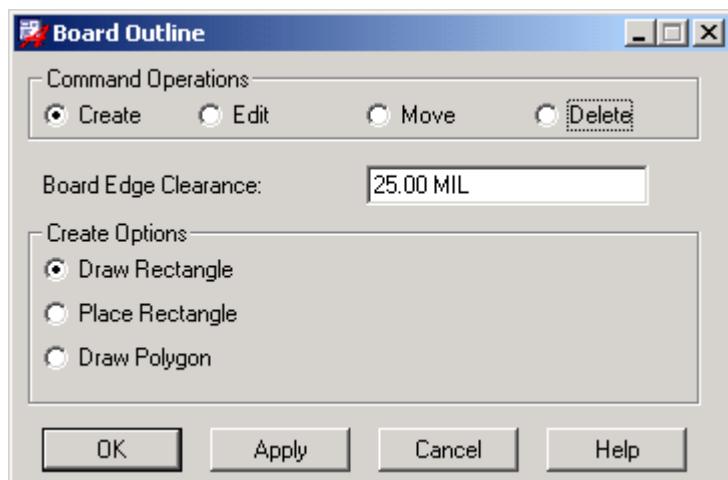
You can modify material names and most other attribute values by entering a new value in the appropriate cell. Two exceptions are *In Use* and *Type* which cannot be changed.

To add or modify materials in your design, refer to the procedures for the [define materials](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

Board Outline

Use the Board Outline dialog box to create a new board outline or modify, move, or delete an existing one. Creating a board outline automatically generates package and route keeppins. Modifying or moving a board outline automatically regenerates those keeppins.

Figure 5-4 The Board Outline Dialog Box



To initiate a board outline task

1. In SI, choose *Setup – Outlines – Board Outline*.

The Board Outline dialog box appears as shown in [Figure 5-4](#)

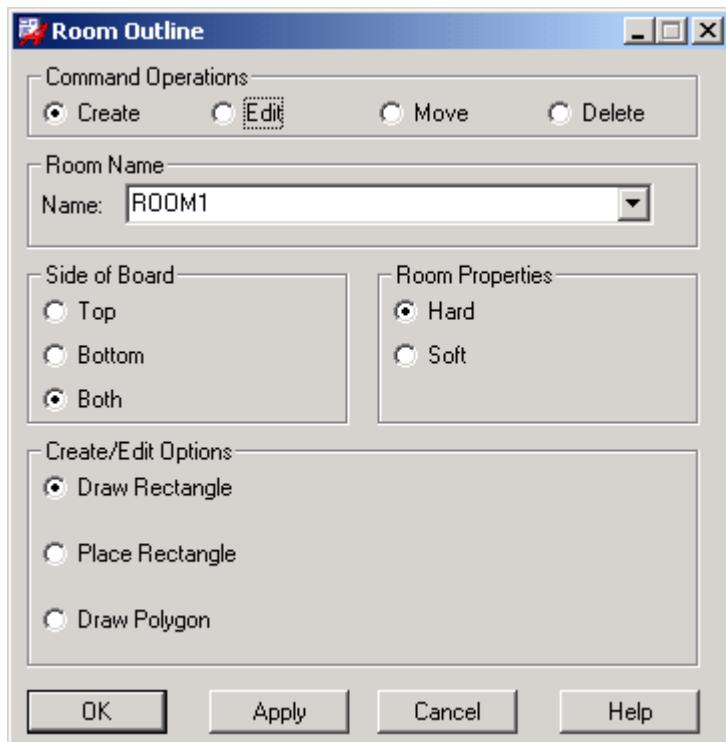
2. Choose the task you want to perform from the *Command Operations* area.

For further steps, refer to the procedures for the [board_outline](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

Room Outlines

Use the Room Outline dialog box to create rooms, specify room names, specify the board layer on which a room is situated, and control DRC errors. Assignment of a physical area to a grouping provides instant feedback during critical placement to assure compliance with grouping constraints.

Figure 5-5 Room Outline Dialog Box



To initiate a room outline task

1. In SI, choose *Setup – Outlines – Room Outline*.

The Room Outline dialog box appears.

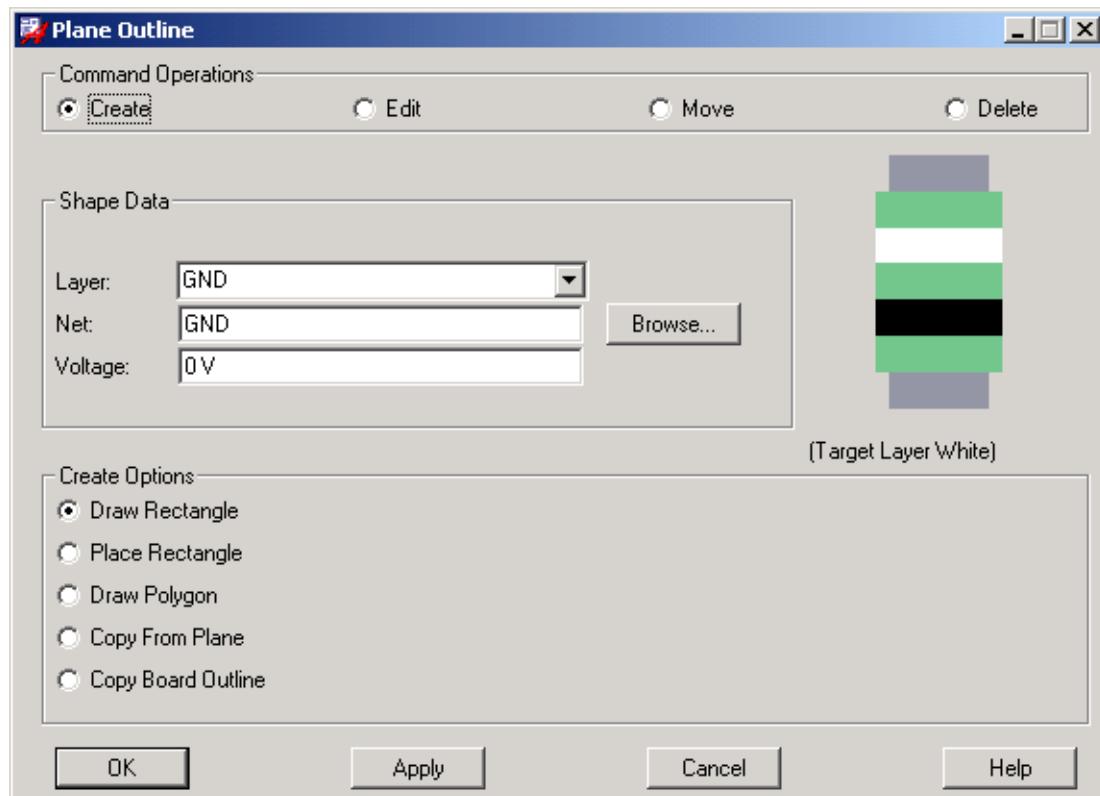
2. Choose the task you want to perform from the *Command Operations* area.

For further steps, refer to the procedures for the room_outline command in the *Allegro PCB and Package Physical Layout Command Reference*.

Plane Outlines

Use the Plane Outline dialog box for creating new plane outlines or modifying, moving, or deleting an existing outline.

Figure 5-6 Plane Outline Dialog Box



To initiate a plane outline task

1. In SI, choose *Setup – Outlines – Plane Outline*.

The Plane Outline dialog box appears.

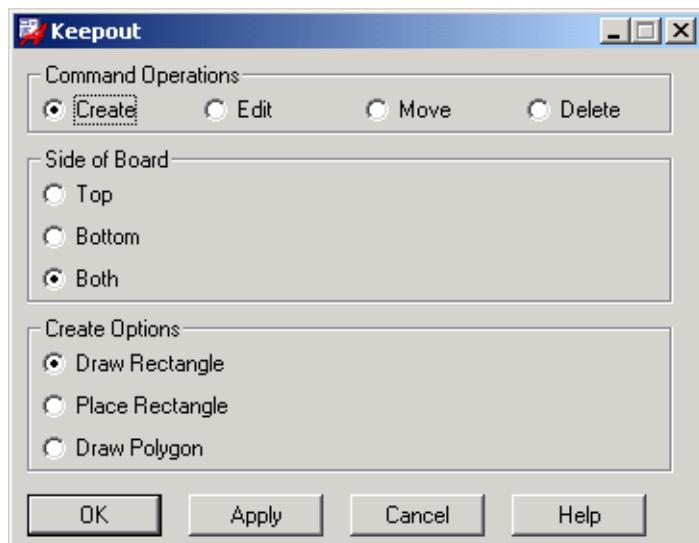
2. Choose the task you want to perform from the *Command Operations* area.

For further steps, refer to the procedures for the `board_plane` command in the *Allegro PCB and Package Physical Layout Command Reference*.

Keepouts

Use the Keepout dialog box, for defining keepout areas to isolate sections within the board outline where component placement is not allowed. You can create, modify, or delete keepout areas. This allows you to define areas of the board without having to use one of the add shape commands.

Figure 5-7 Keepout Dialog Box



To initiate a keepout task

1. In SI, choose *Setup – Outlines – Keepout*.

The Keepout dialog box appears.

2. Choose the task you want to perform from the *Command Operations* area.

For further steps, refer to the procedures for the [board_keepout](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

Importing Setup Data

Rather than setting up your board from scratch, you can also import setup data from external sources such as board files as well as technology files that have been exported from other designs. Importing setup data into your design is one way to implement re-use of board geometry, parameters, constraints, and critical components that have been quality proven in other designs.

The following table lists the setup data that can be imported into your design along with its corresponding source.

Table 5-1 Board Setup Data

Source	Data
Board File (.brd)	Board Outline Cross-section Keepouts Rooms Placed Components Electrical Rules
Technology File (.tech)	User units Drawing parameters Layout cross section parameters DRC modes for constraints Spacing constraint sets Physical constraint sets Electrical constraint sets Design constraints Constraint assignment tables User property definitions

Importing a Technology File

Use the Tech file In dialog box, for importing a technology file into your design. A tech file is in ASCII format. Results of the import appear in the `tf_read.log` file within the current directory. Any errors, warnings, and conflicts are categorized according to their severity.



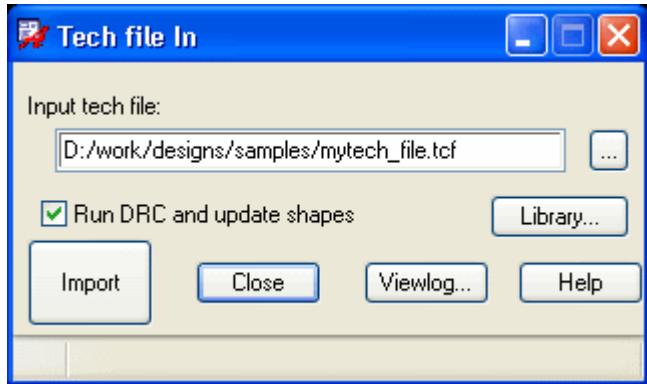
When you read the tech file in, all values in the design are overwritten. If a constraint in the tech file does not exist in the design, it is added. If an error occurs in the tech file, it continues to be read with warning and error messages written to the log file. Your design is not updated if an error occurs.

To access the Tech File In dialog box

- In SI, choose *File – Import – Techfile*.

The Tech File In dialog box appears as shown in [Figure 5-8](#)

Figure 5-8 Tech file In Dialog Box

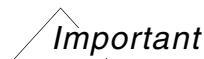


To import a technology file into your design, refer to the procedures for the [`techfile in`](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

For further details, see [*Chapter 6, Creating and Using Technology Files*](#) in *Allegro PCB and Package User Guide: Defining and Developing Libraries*.

Importing a Board File

Use the Import Board dialog box, for selectively importing board design data from another board into your current design. The directory path and file name of the source board appear at the top of the dialog box.



If conflicts are detected between existing data and source data while importing a board into your design, a Conflicts dialog box appears. This dialog box gives you the opportunity to either overwrite existing data with the source data or reject the source data in its entirety. You can also choose to do this on an item-by-item basis.

To access the Import Board dialog box

1. In SI, choose *File – Import – Board*.

The Boardoutline Import dialog box appears as shown in [Figure 5-9](#)

2. Navigate and select the board file that you want to import into your design, then click *Open*.

The Import Board dialog box appears as shown in [Figure 5-10](#) on page 181

Figure 5-9 Boardoutline Import Dialog Box

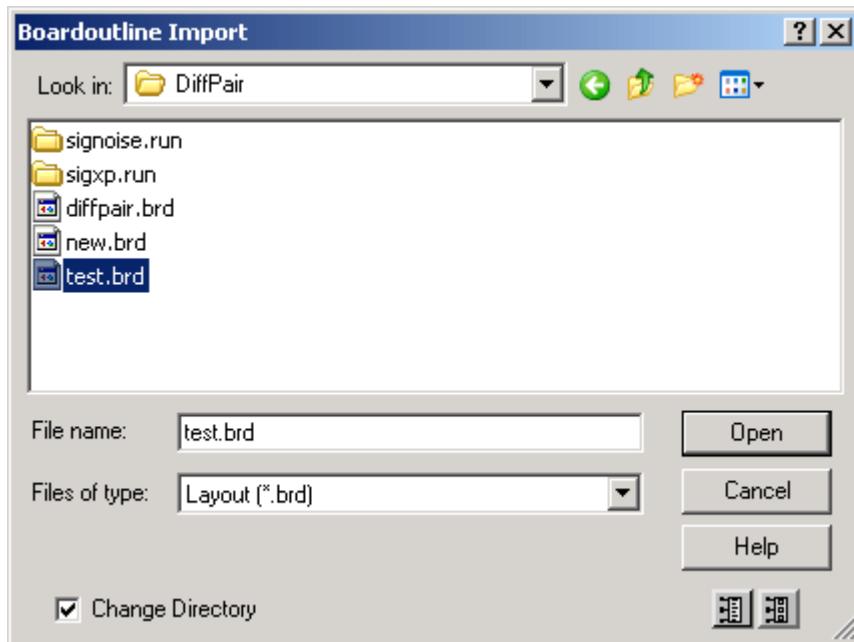
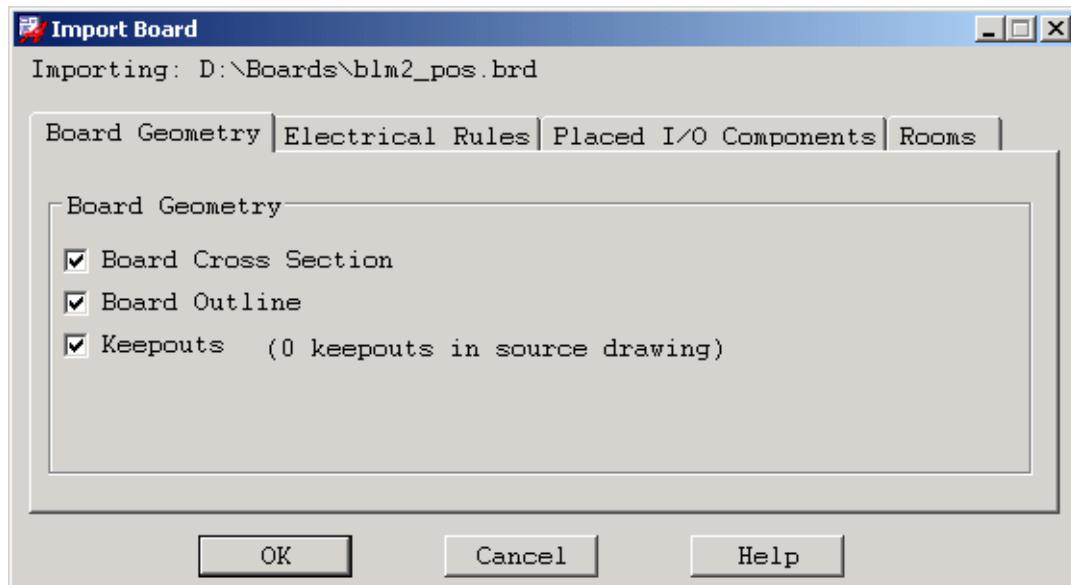


Figure 5-10 Import Board Dialog Box



To import a board file into your design, refer to the procedures for the [boardoutline import](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

Defining Logic

Defining logic involves one or more of the following tasks.

- [Component Creation and Placement](#)
- [Device Model Creation and Assignment](#)
- [Netlist Creation](#)

Drawing Logic Scenarios at the Board Level

You may want to simulate certain layout configurations that are best mocked-up directly at the board level. This often comes up when you want to test a layout geometry that is not part of the standard SigXplorer Interconnect library. Sloffers layout features that provide an easy way to draw and test most any configuration imaginable. For further details and to review an example, see [Logic Scenario Mock-up Example - A Look at Self-Coupling](#) on page 187.

Component Creation and Placement

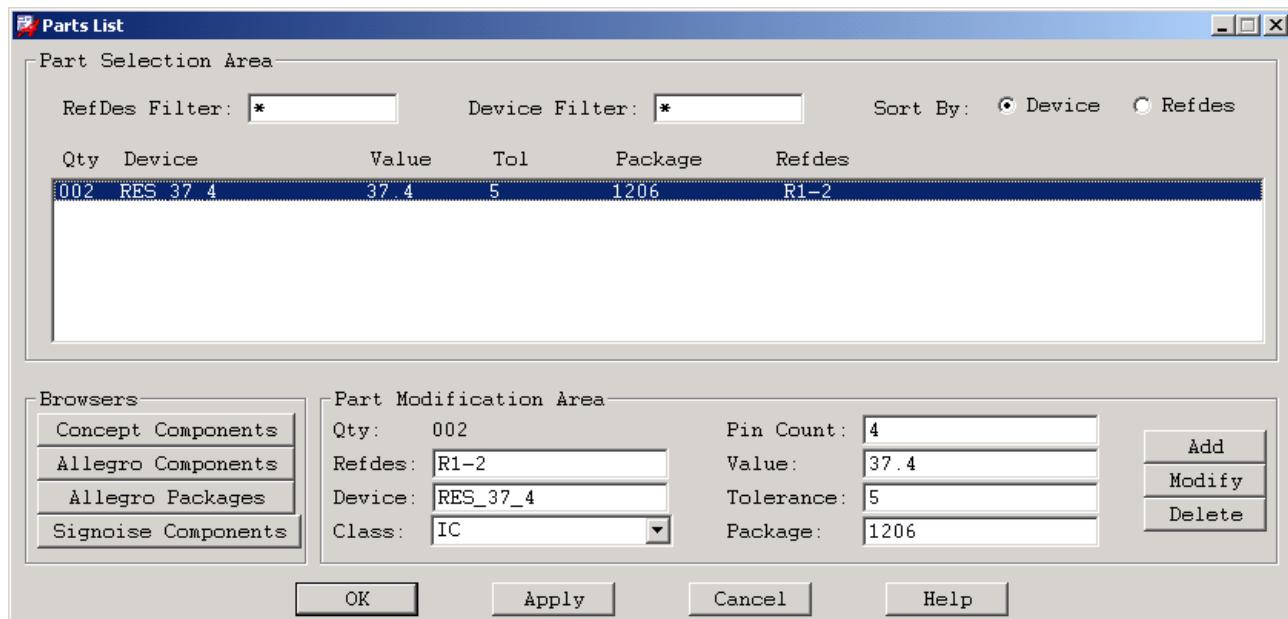
Use the Parts List dialog box to view and edit the parts list of your design or to create temporary component parts from scratch.

To access the Parts List dialog box

- In SI, choose *Logic - Parts List*.

The Parts List dialog box appears as shown in [Figure 5-11](#)

Figure 5-11 Parts List Dialog Box



To edit the parts list in your design or create / modify temporary components, refer to the procedures for the [edit_parts](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

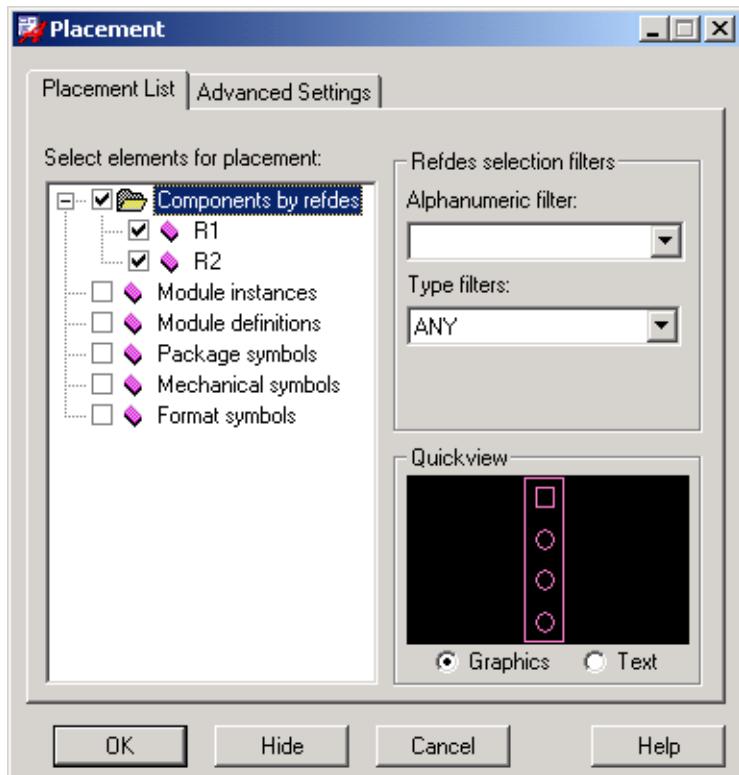
After creating components, use the Placement dialog box to interactively place the components into your design. In addition to components, the tabbed interface allows you to choose placement symbol types and modules.

To access the Placement dialog box

- In SI, choose *Place - Manually*.

The Parts List dialog box appears as shown in [Figure 5-12](#)

Figure 5-12 Placement Dialog Box



To place components in your design, refer to the procedures for the [place manual](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

Device Model Creation and Assignment

Once components are placed, use the Signal Model Assignment dialog box to either create new or select existing device models and assign them to the components in your design.

When you simulate a net, TLsim develops circuit models using the device models and interconnect in your design. This means that you must assign a device model to each component in the design and point TLsim to the device model libraries (where device model files are stored). Model assignments are made to individual components or to all components having the same device file.

During device model assignment, you can select models from the default model library, the standard digital device model library, or from other device model libraries that you have developed and made available through the Library Browser. You can also create Espice and

IBIS device models from scratch directly from the Signal Model Assignment dialog box. IBIS device models are created from IO Buffer Information Sheet (IBIS) standard data, or by editing existing models to accurately characterize devices.

To access the Signal Model Assignment dialog box

- In SI, choose *Analyze – SI/EMI Sim – Model*

The Signal Model Assignment dialog box appears.

To create and assign device models, refer to the procedures for the [signal model](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

Netlist Creation

SI lets you create a netlist without having to draw a schematic. This unique feature enables you to explore various layout geometries at the board level. Once components are placed and device models are assigned, you can use the Edit Nets dialog box to create a netlist that defines the interconnect between the components.

Using the ratsnest of the net, you can perform certain signal integrity simulations on the layout. However, If your simulations require etch (for example, self-coupling analysis), you can use the `add connect` command to route the connections interactively before you simulate.

To access the Edit Nets dialog box

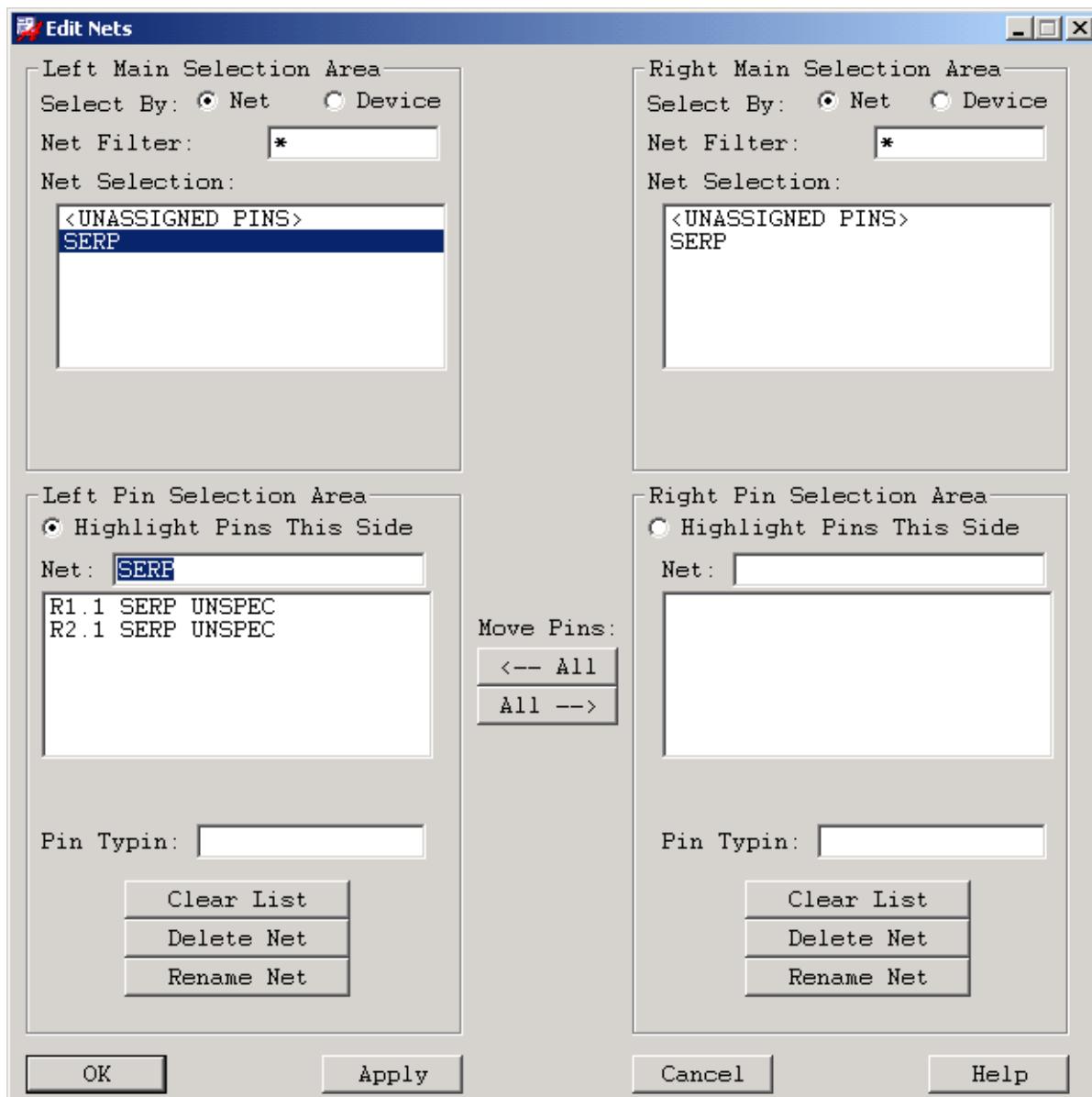
- In SI, choose *Logic – Edit Nets*.

The Edit Nets dialog box appears as shown in [Figure 5-13](#) on page 186.

Allegro PCB SI User Guide

Floorplanning

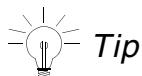
Figure 5-13 Edit Nets Dialog Box



To create a netlist from scratch or to modify an existing netlist in your design, refer to the procedures for the [edit nets](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

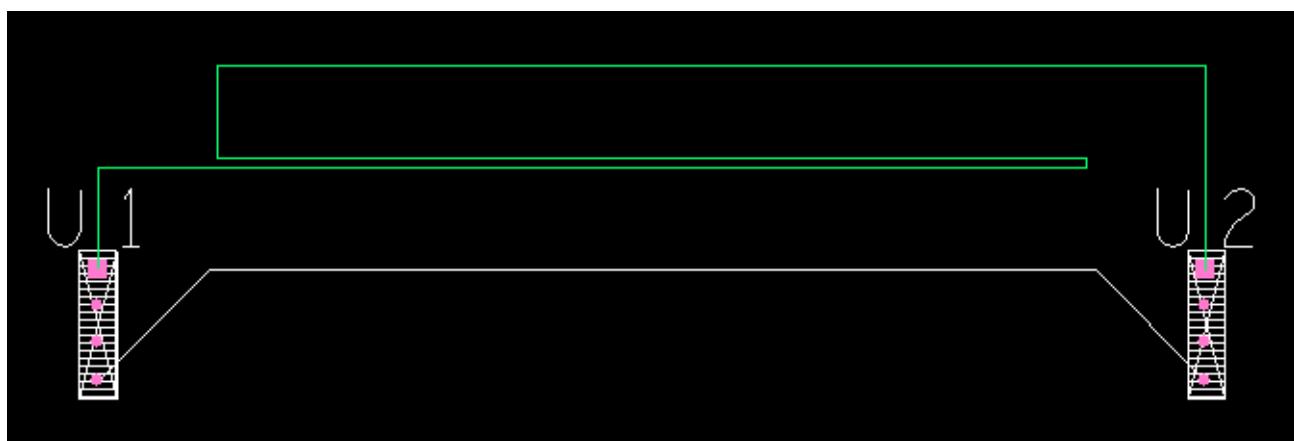
Logic Scenario Mock-up Example - A Look at Self-Coupling

The following example illustrates how you can use the logic features in SI to quickly mock-up, simulate, and analyze a circuit for self-coupling. A net routed in a serpentine pattern is the basis for the analysis as shown in [Figure 5-14](#).



You can use the same steps outlined in this example to invent, mock-up and simulate many other logic scenarios at the board level.

Figure 5-14 Self-coupling Test Scenario

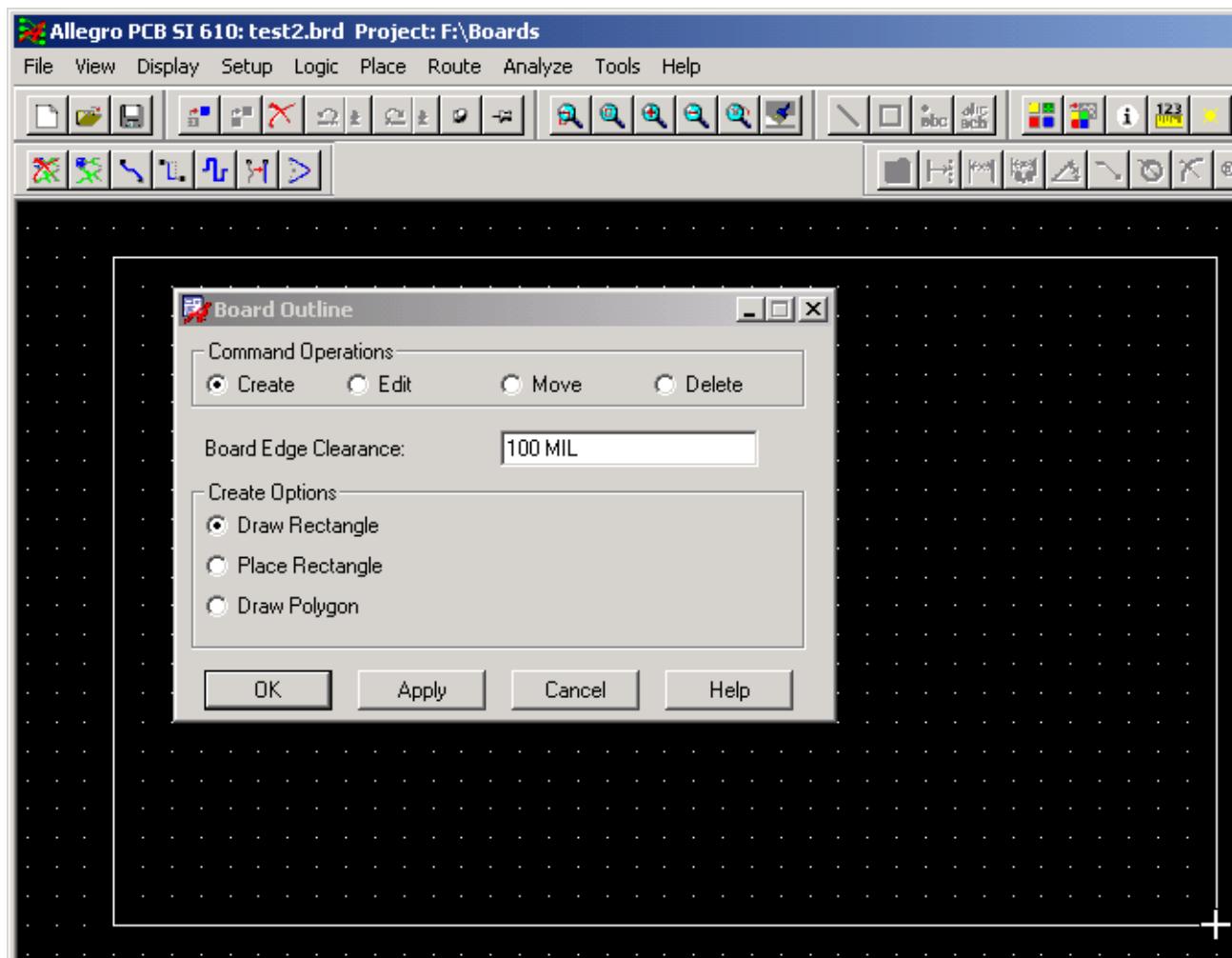


Step 1 - Board Setup

In order to perform self-coupling simulations at the board level, you need to perform the following setup tasks.

- Create a simple rectangular board outline on an empty canvas as shown in [Figure 5-15](#). See [Board Outline](#) on page 174 for details.

Figure 5-15 Drawing the PCB Mock-up Outline

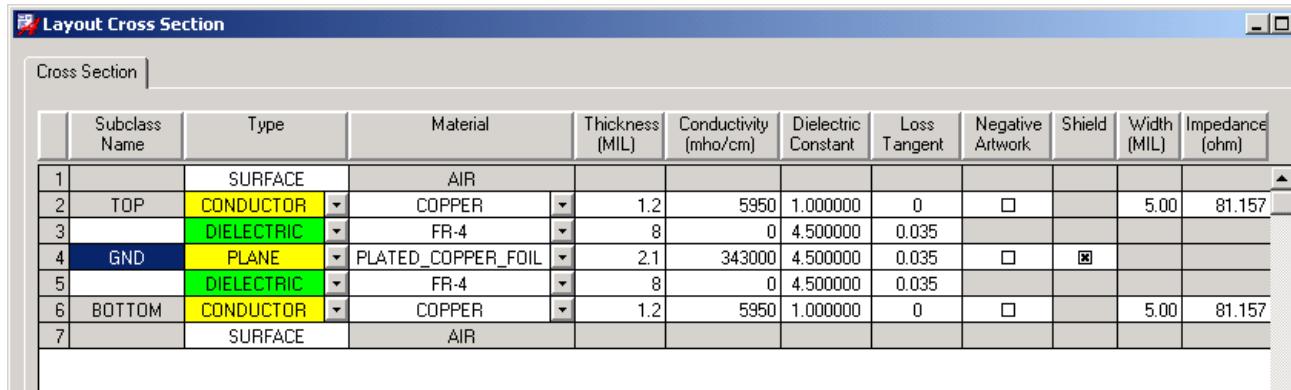


- Edit the default stackup and add a ground layer to provide an impedance for the trace as shown in [Figure 5-16](#) on page 189. Refer to the procedure [The Materials Editor](#) on page 172 for complete details.

Allegro PCB SI User Guide

Floorplanning

Figure 5-16 Adding a Ground Layer



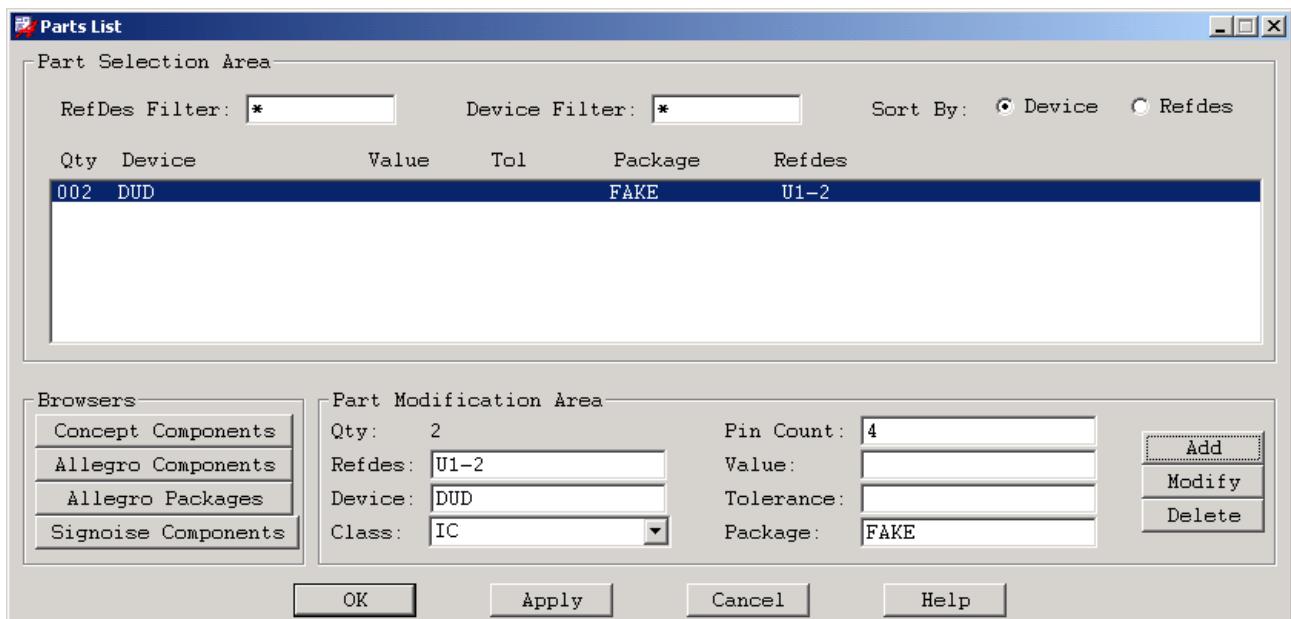
The screenshot shows the 'Layout Cross Section' dialog with a table titled 'Cross Section'. The table has columns for Subclass Name, Type, Material, Thickness (MIL), Conductivity (mho/cm), Dielectric Constant, Loss Tangent, Negative Artwork, Shield, Width (MIL), and Impedance (ohm). The rows represent different layers of the PCB:

Subclass Name	Type	Material	Thickness (MIL)	Conductivity (mho/cm)	Dielectric Constant	Loss Tangent	Negative Artwork	Shield	Width (MIL)	Impedance (ohm)
1	SURFACE	AIR								
2	TOP	CONDUCTOR	COPPER	1.2	5950	1.000000	0	<input type="checkbox"/>	5.00	81.157
3		DIELECTRIC	FR-4	8	0	4.500000	0.035			
4	GND	PLANE	PLATED_COPPER_FOIL	2.1	343000	4.500000	0.035	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
5		DIELECTRIC	FR-4	8	0	4.500000	0.035			
6	BOTTOM	CONDUCTOR	COPPER	1.2	5950	1.000000	0	<input type="checkbox"/>	5.00	81.157
7	SURFACE	AIR								

Step 2 - Component Creation

- Create some temporary components to support the self-coupling scenario as shown in [Figure 5-17](#). Refer to the procedures in the [edit_parts](#) command in the *Allegro PCB and Package Physical Layout Command Reference* for complete details.

Figure 5-17 Creating Temporary Components



Step 3 - Component Placement

- Place the temporary components on the board as shown in [Figure 5-19](#) on page 191. Refer to the procedures for the [place manual](#) command in the *Allegro PCB and Package Physical Layout Command Reference* for complete details.

Note: Due to the fact that the component package (FAKE) specified in the Parts List dialog box does not exist yet, you are prompted to first create a temporary package for the components. Proceed with temporary package creation by clicking *OK* in the Create Temporary Package dialog box that appears as shown in [Figure 5-18](#).

Figure 5-18 Creating a Temporary Package

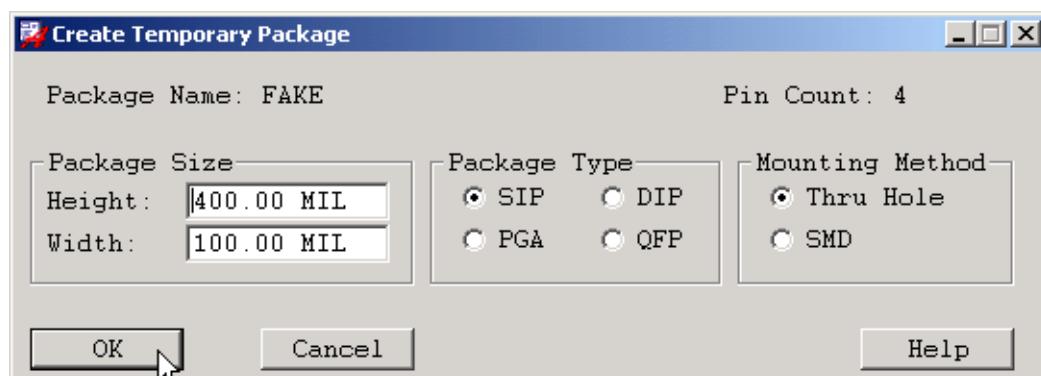
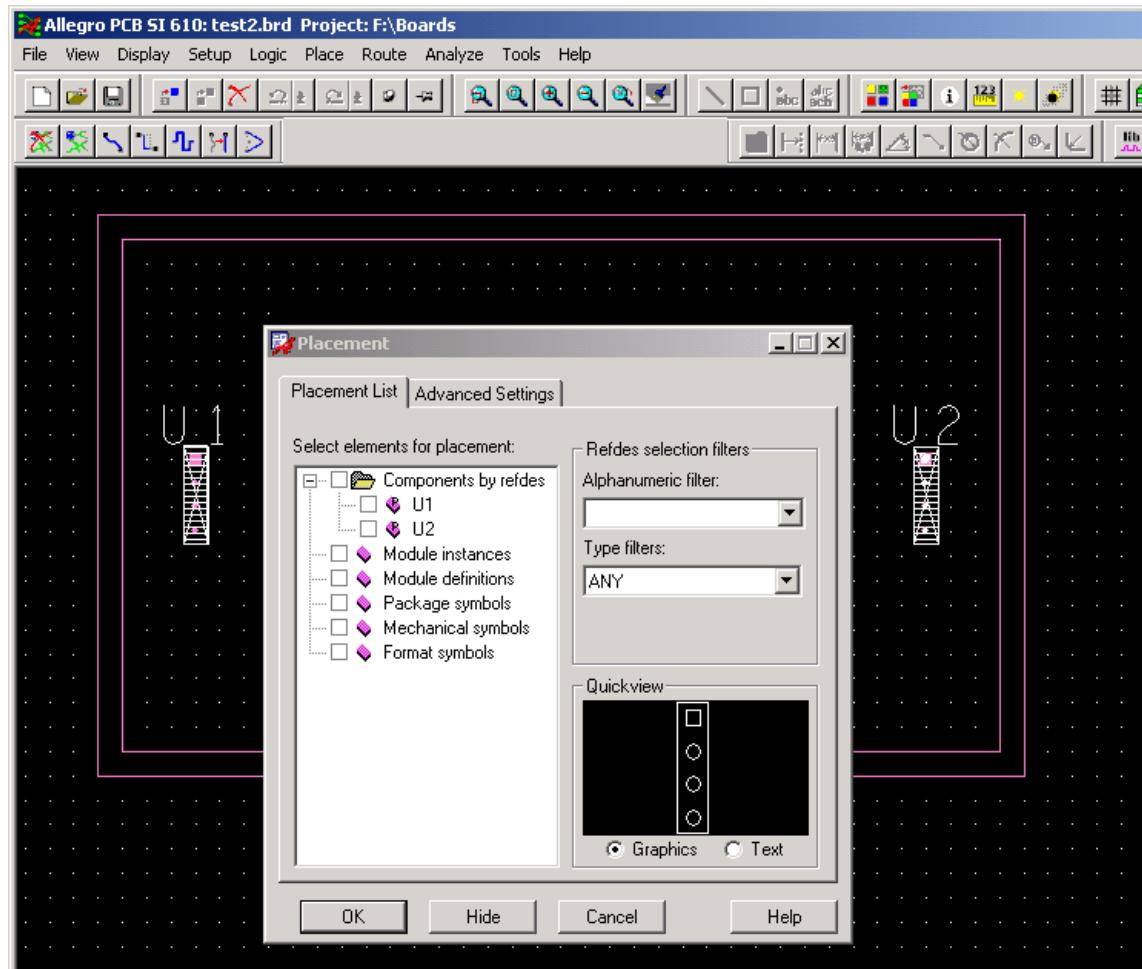


Figure 5-19 Placing the Temporary Components



Step 4 - Device Model Creation and Assignment

- Create an IBIS device model and assign it to the temporary components as shown in [Figure 5-21](#) on page 193, [Figure 5-22](#) on page 193, [Figure 5-23](#) on page 193, and [Figure 5-24](#) on page 194.

Note:

- Click Yes when prompted to add ground and power pins to the model.
- Clicking *OK* in the Signal Analysis dialog box after creating the model automatically assigns it to the DUD components.

Refer to the procedures for the `signal model` command in the *Allegro PCB and Package Physical Layout Command Reference* for complete details.

Allegro PCB SI User Guide

Floorplanning

Figure 5-20 Launching the Create Device Model Dialog Box for the DUD Components

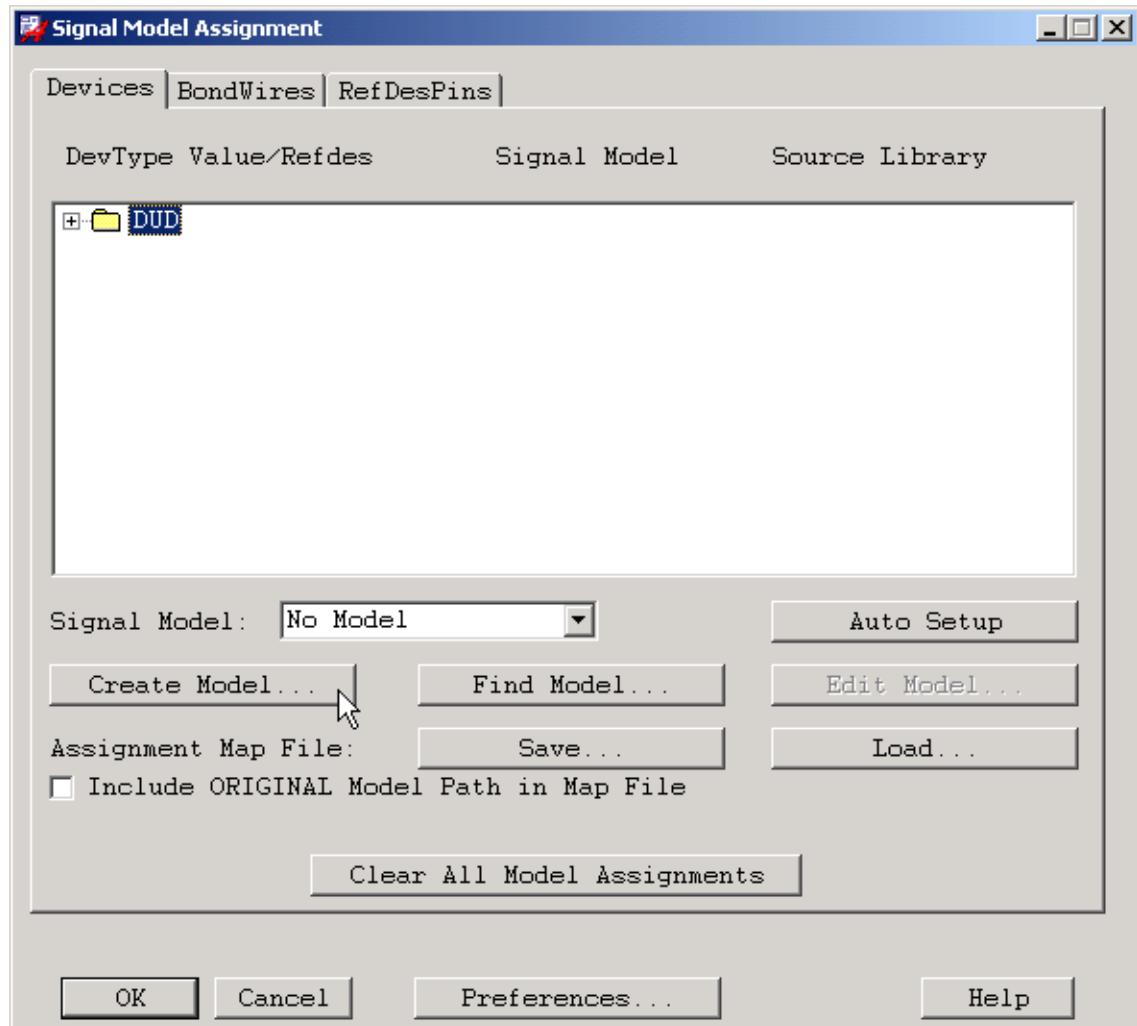


Figure 5-21 Specifying the Device Model Type

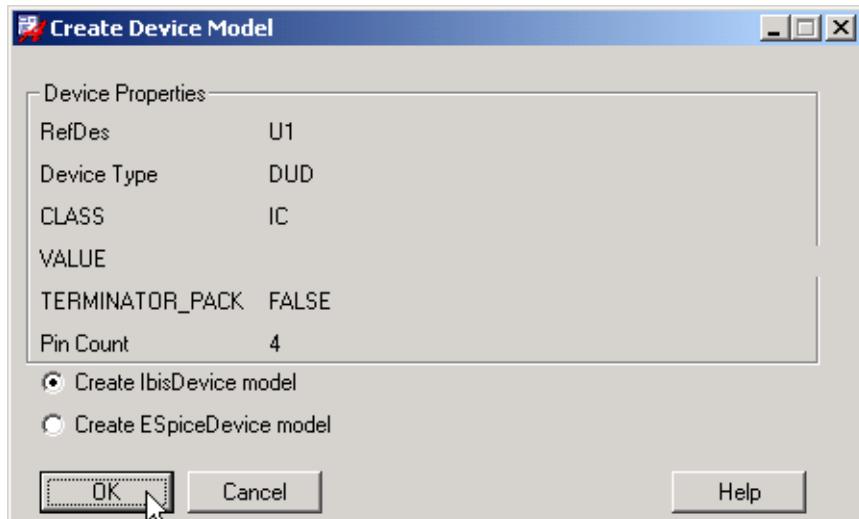


Figure 5-22 Creating the IBIS Device Model

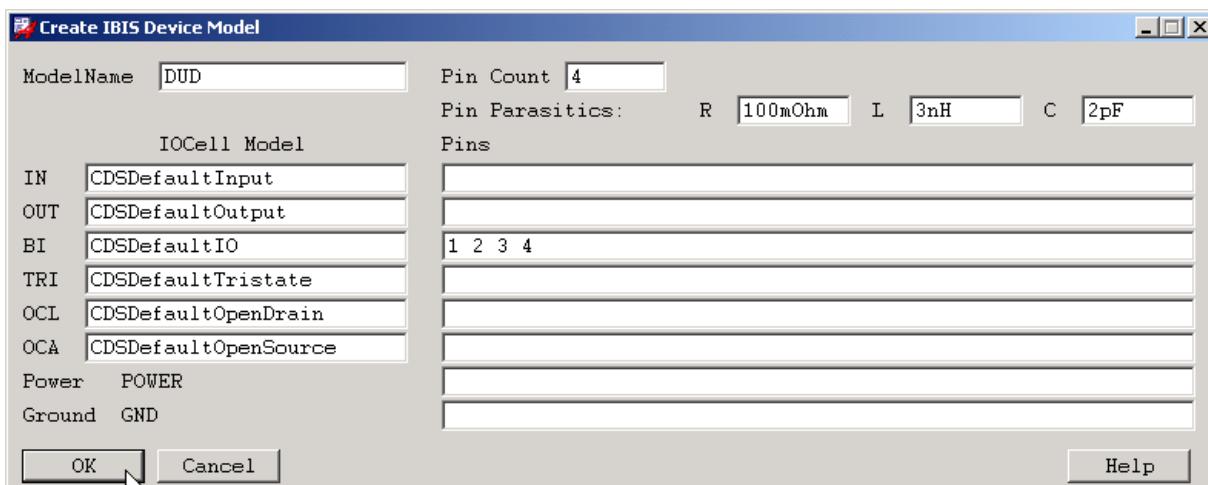
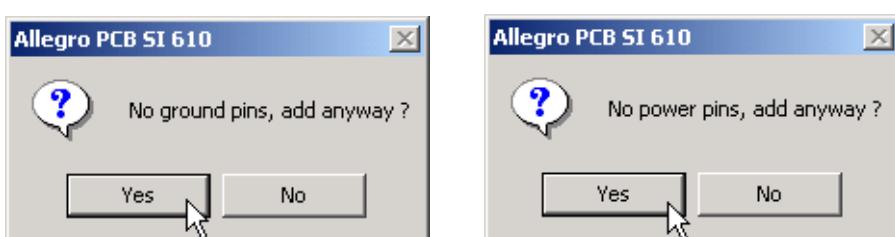


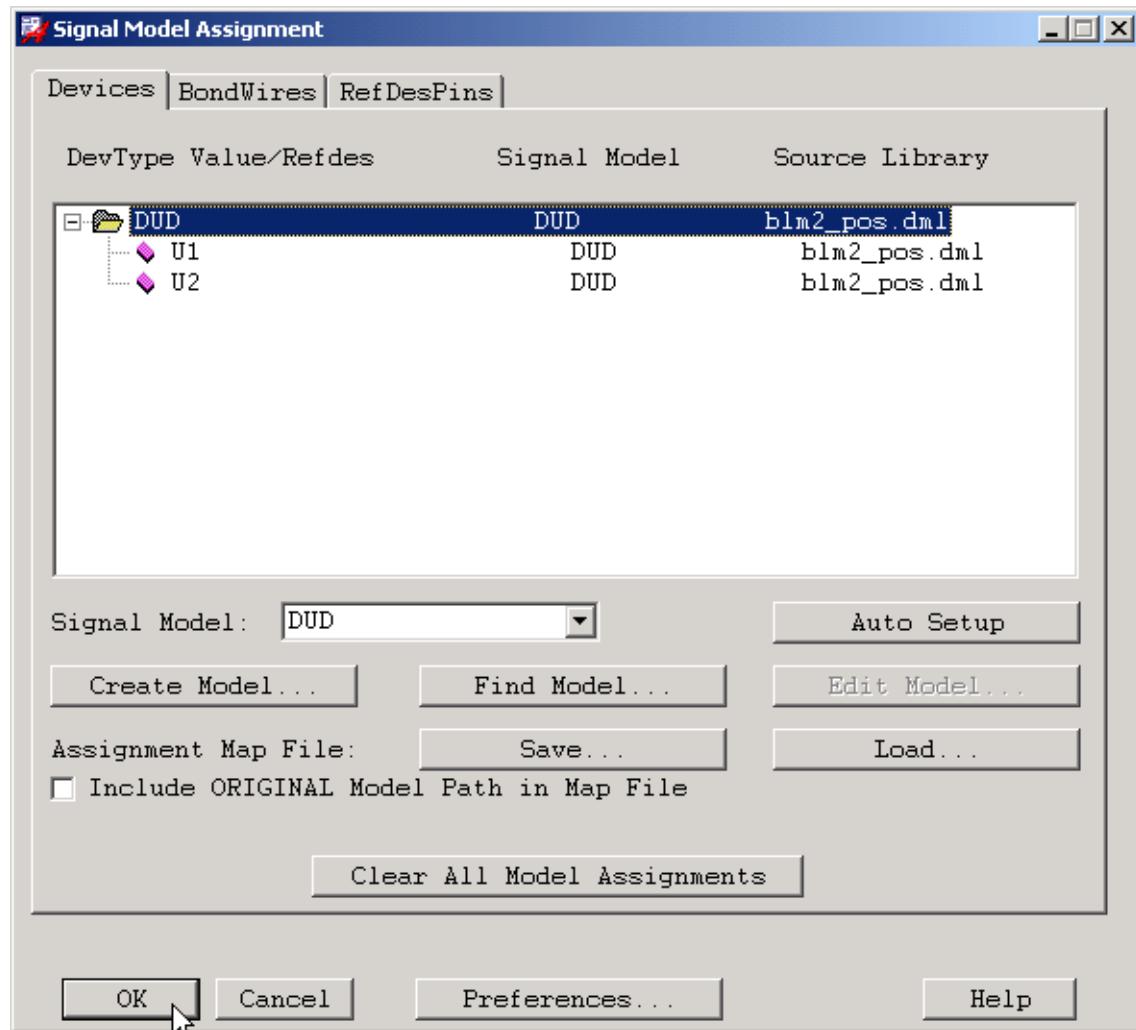
Figure 5-23 Power and Ground Pins Prompts



Allegro PCB SI User Guide

Floorplanning

Figure 5-24 Device Model Assignment



Allegro PCB SI User Guide

Floorplanning

Figure 5-25 Device Model Report

The screenshot shows a window titled "Signal Model Assignment Changes". The menu bar includes "File", "Close", and "Help". The main content area displays two sections of text and a table.

SIGNAL_MODEL_DUD assigned to the following:
component U1 (changed from <none>)
component U2 (changed from <none>)

Component pin PINUSE properties modified to match assigned model types:

COMPONENT	PIN	OLD PINUSE	NEW PINUSE	MODEL TYPE
U2	4	UNSPEC	BI	IbisIO
U2	3	UNSPEC	BI	IbisIO
U2	2	UNSPEC	BI	IbisIO
U2	1	UNSPEC	BI	IbisIO
U1	4	UNSPEC	BI	IbisIO
U1	3	UNSPEC	BI	IbisIO
U1	2	UNSPEC	BI	IbisIO
U1	1	UNSPEC	BI	IbisIO

Step 5 - Netlist Creation

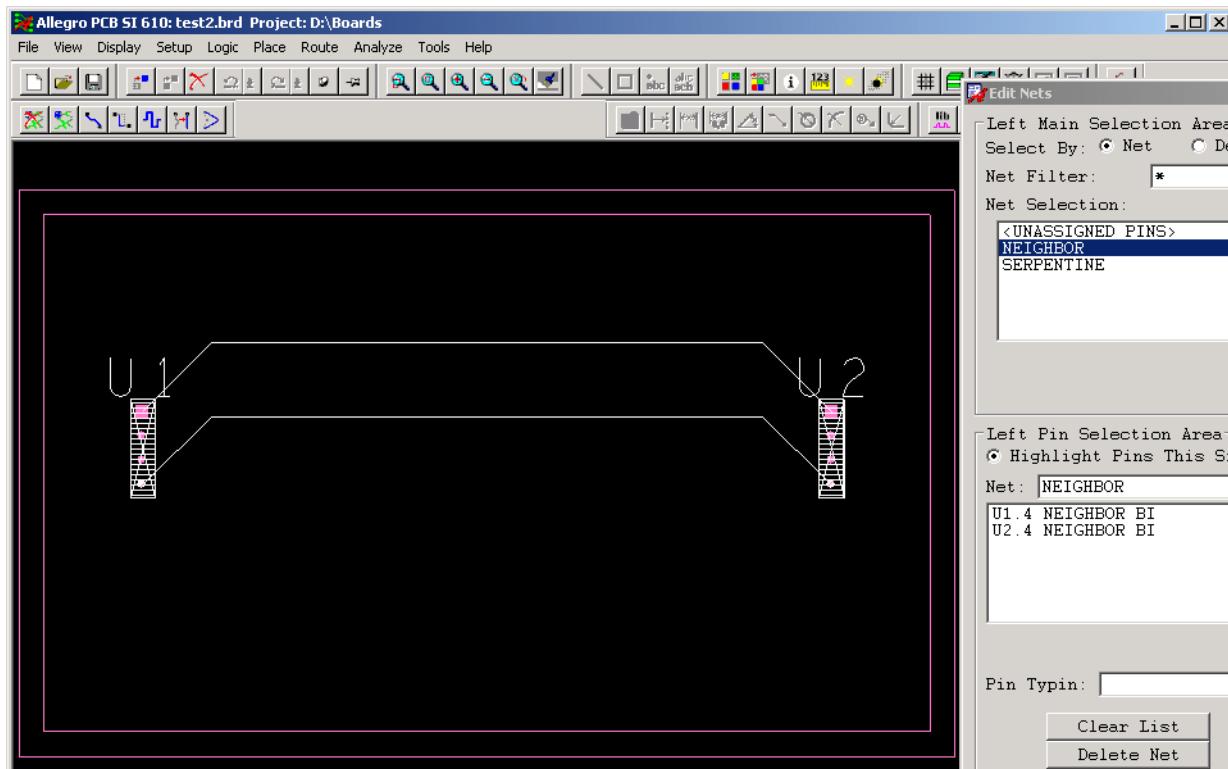
- Create a netlist comprised of two nets (SERPENTINE and NEIGHBOR) that connect the pins of the temporary components as shown in [Figure 5-26](#).

Note:

- Click Yes when presented with the following prompts.
- The net name <netname> doesn't currently exist. Do you want to create a new net?
 - Do you want the pins in this side's pin list added to the new net?
- Once the netlist is created, you are able to perform certain simulations on the layout using the ratsnest. However, in order to perform a self-coupling simulation, you must first route the SERPENTINE net back onto itself. This is addressed in [Step 6 - Routing](#) on page 197.

Refer to the procedures for the [edit nets](#) command in the *Allegro PCB and Package Physical Layout Command Reference* for complete details.

Figure 5-26 Creating a Netlist from Scratch



Step 6 - Routing

- Route the net named SERPENTINE back onto itself for a 3 inch run as shown in [Figure 5-27](#). This step is required for the self-coupling simulation.

Refer to the procedures for the [add_connect](#) command in the *Allegro PCB and Package Physical Layout Command Reference* for complete details.

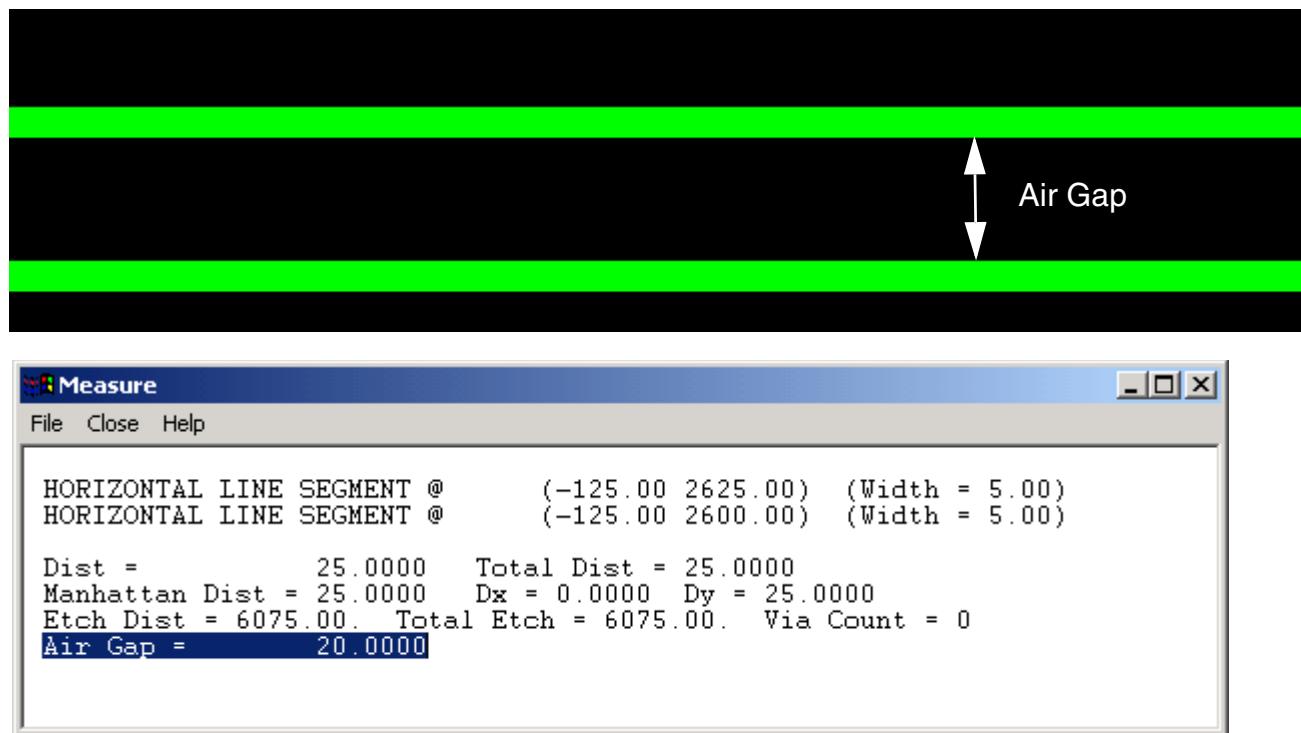
Figure 5-27 Routing a Net to Simulate



Step 7- Simulation and Analysis

- Simulate the SERPENTINE net to analyze for self-coupling.
 - Zoom in on an area where the trace runs parallel and measure the air gap as shown in [Figure 5-28](#). This value is used later to set the size of the geometry window around the net where the coupling is checked.
- Refer to the procedures for the [show_measure](#) command in the *Allegro PCB and Package Physical Layout Command Reference* for complete details.

Figure 5-28 Measuring the Air Gap



- b.** Choose *Analyze – SI/EMI Sim – Probe*.

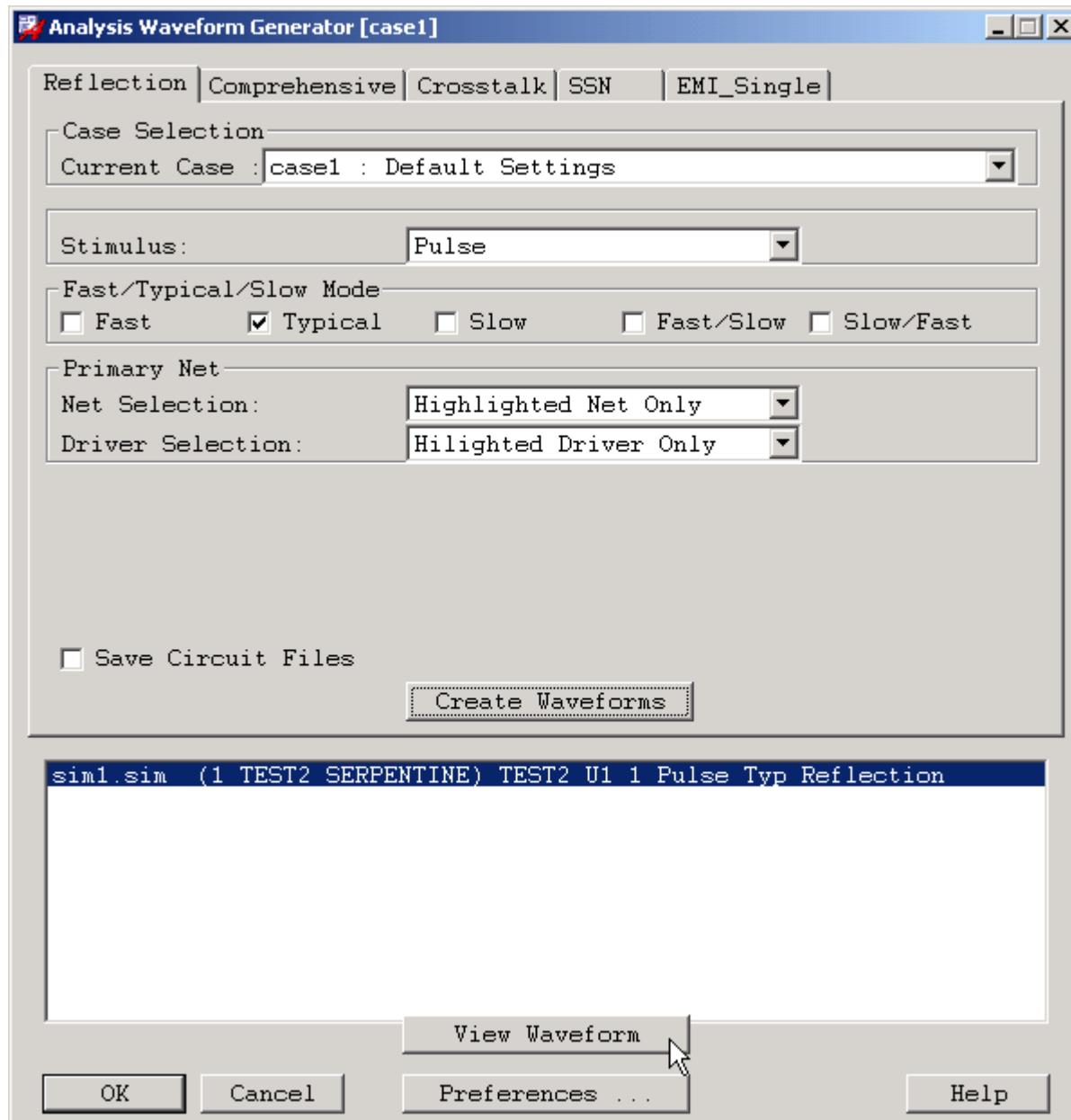
The Signal Analysis dialog box appears.

- c.** Click *Waveforms*.

The Waveform Analysis Generator dialog box appears.

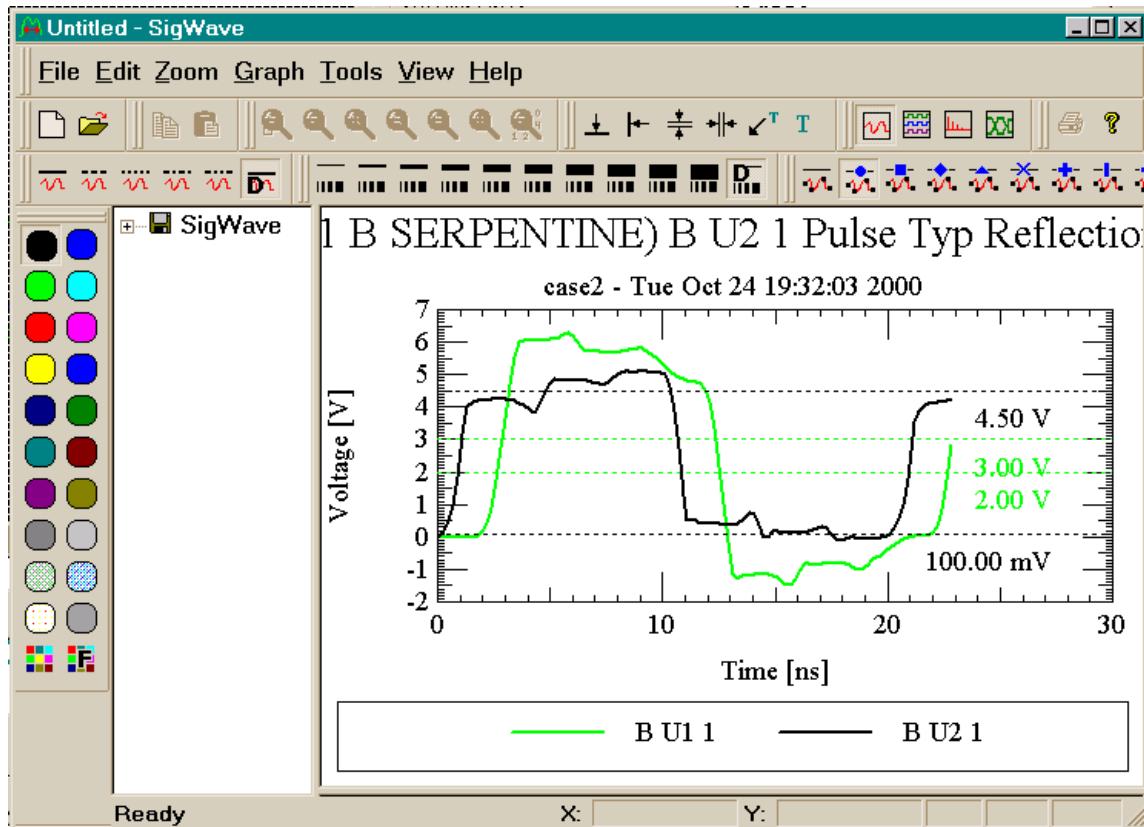
- d.** Set up a Reflection simulation (no coupling) as shown in [Figure 5-29](#) on page 199.

Figure 5-29 Setting Up a Reflection Simulation (no coupling)



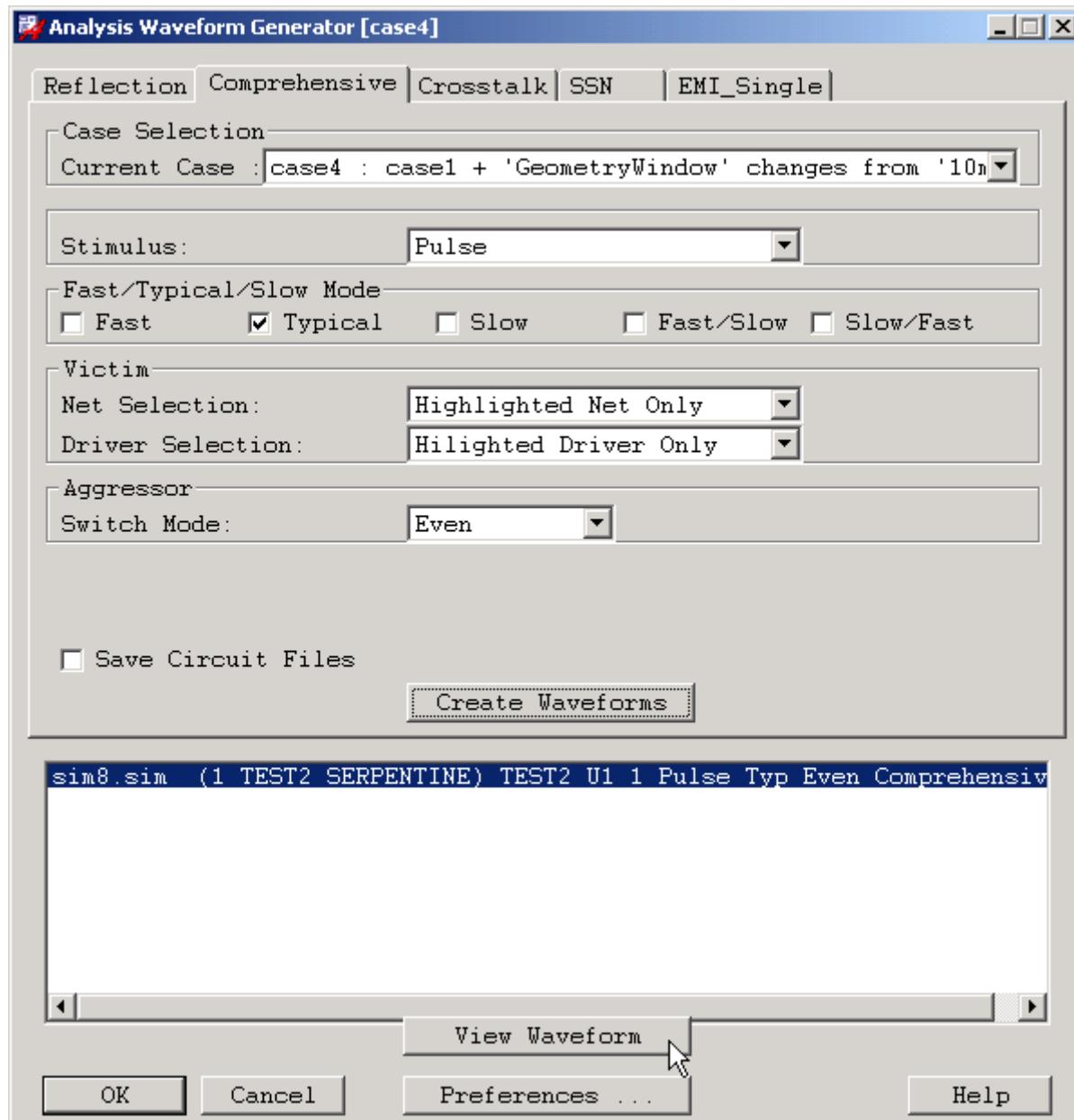
- e. Generate and view the resulting waveforms in SigWave as shown in [Figure 5-30](#) on page 200.

Figure 5-30 Viewing Reflection Simulation Waveforms in SigWave



- f. Repeat the simulation using the *Comprehensive* tab of the Analysis Waveform Generator (to specify coupling) setting the *Aggressor Switch Mode* to *Even* as shown in [Figure 5-31](#) on page 201.

Figure 5-31 Setting up for Coupling Simulation

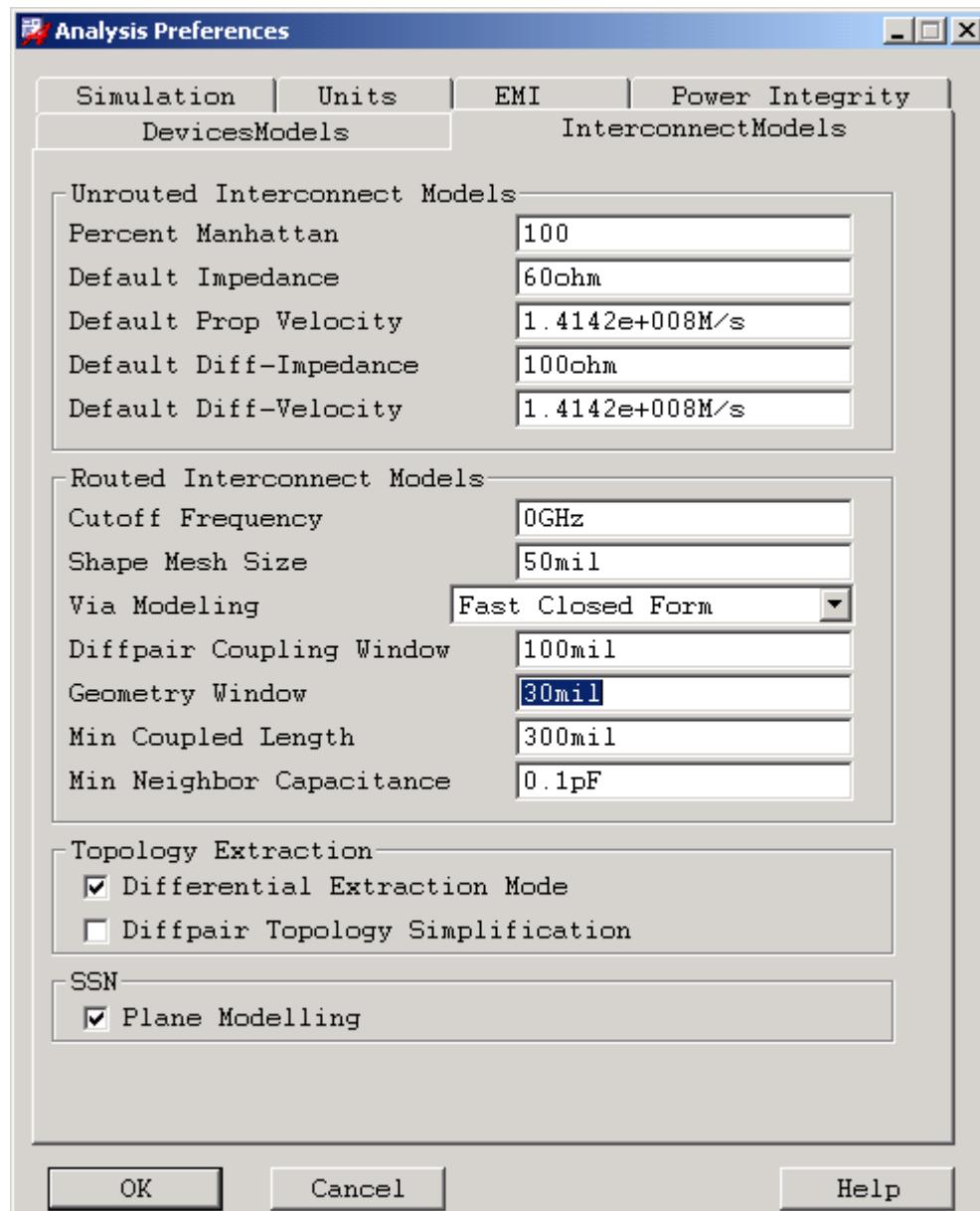


g. Click *Preferences*.

The Analysis Preferences dialog box appears.

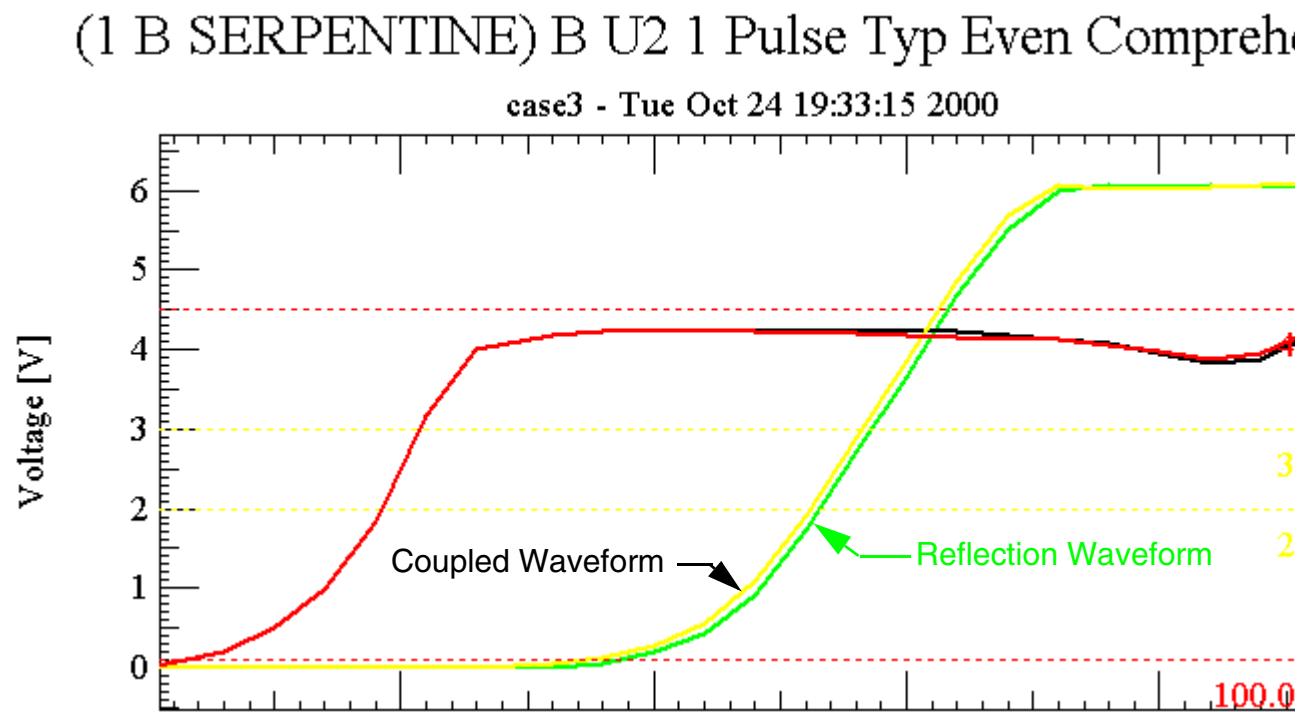
h. Display the *InterconnectModels* tab and set the *geometry window* to 30 mils (10 mils larger than the air gap) as shown in [Figure 5-32](#) on page 202.

Figure 5-32 Setting the Geometry Window Size



- i. Generate and view the waveforms in Sigwave.
- j. Turn on (superimpose) the original waveforms (no coupling) still resident in Sigwave and compare them against the new waveforms (coupling simulated) to analyze the effect of self-coupling on the signal as shown in [Figure 5-33](#) on page 203.

Figure 5-33 Analyzing Self-Coupling in Sigwave (Zoomed In)

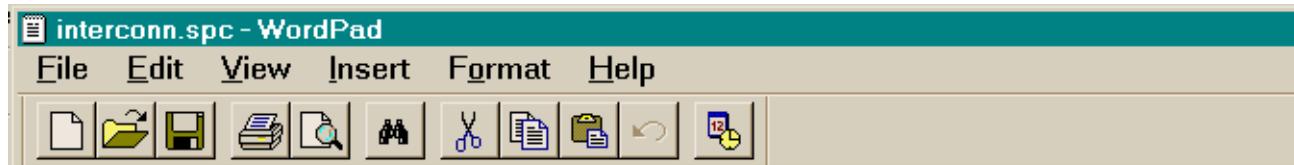


As you can see (zoomed in on the top-left corner of the waveforms) in [Figure 5-33](#), the impedance is skewed a little higher and the edge is moving slightly faster on the coupled waveform.

- Check the `interconn.spc` file generated by `SigNoise`.

In addition to viewing waveforms, you can extend your self-coupling analysis by checking the Spice file `interconn.spc` in your `.../signoise.run>case<#>` directory. Refer to [Figure 5-34](#) on page 204.

Figure 5-34 Coupled Trace Segment



The screenshot shows a Windows WordPad application window titled "interconn.spc - WordPad". The menu bar includes File, Edit, View, Insert, Format, and Help. The toolbar contains standard icons for file operations like Open, Save, Print, and Cut/Paste. The main text area contains K-spice code for a coupled trace segment:

```
*  
* K-spice B_Interconn  
*  
.subckt B_Interconn 1 2  
* External nodes map @@U2.1 @@U1.1  
* The circuits.  
NTL_XMTL_SX3100_SY31600_EX6300_EY31600_L1_P1 ( 3 0 ) ( 4 0 )  
+L=0.00635 rlgc name=STL 1S 1R 7930 file=nt1_rlgc.inc  
NTL_XMTL_SX3100_SY31600_EX6300_EY31600_L1_P2 ( 4 5 0 ) ( 6 7 0 )  
+L=0.07493 rlgc name=MTL_1S_2R_1246 file=nt1_rlgc.inc
```

The coupled, lossy and frequency-dependent line of code highlighted in this figure represents the three inch long section of parallel trace that was routed on the mock-up PCB. See [Figure 5-27](#) on page 197.

Topology Extraction

Overview

You can create a circuit topology for analysis in one of two ways. You can either build it from scratch in SigXplorer, or you can select a net in the design and *extract* its topology into SigXplorer for exploration. Once extracted, you are ready to begin solution space analysis on the topology as default signal models are already assigned.

During solution space analysis, you simulate, examine, compare and refine the net topology to provide the best solution for the design. The result of these analysis is a topology template that represents a set of electrical and physical constraints that are used to control the final placement and routing of the circuit on the board. In other words, the topology template implements your design intent.

Extraction Prerequisites

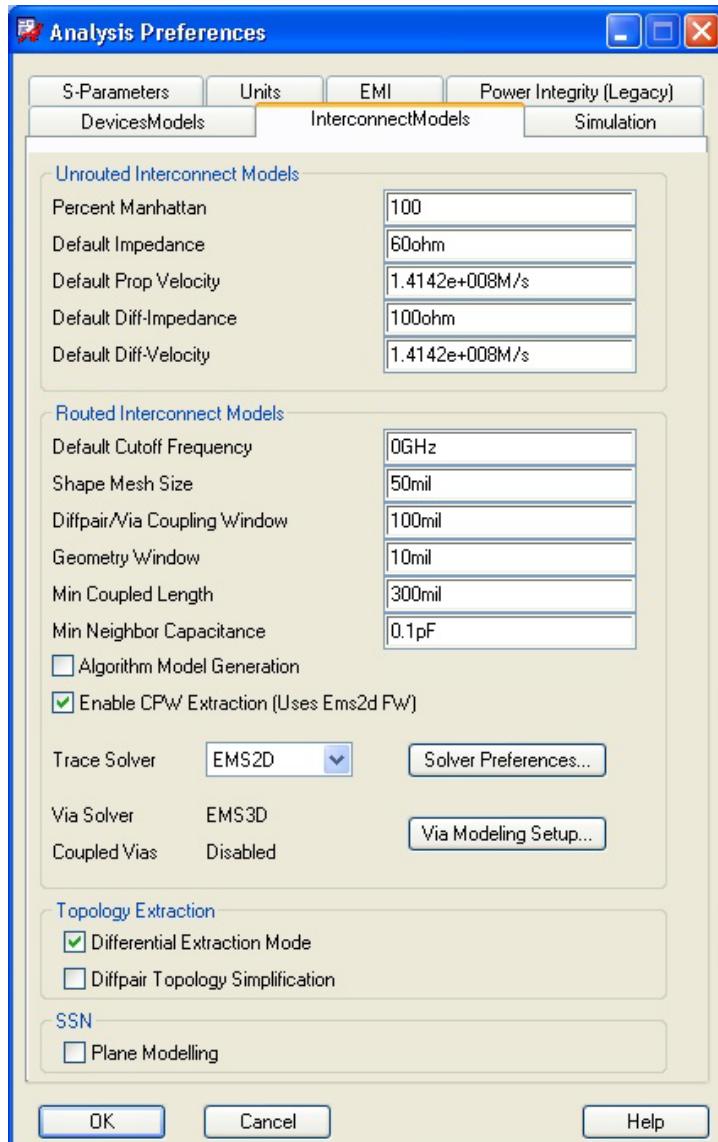
Before you can extract a net topology, you must:

- complete the database setup requirements. See [Setting up the Design](#) on page 61.
- complete the extraction setup requirements. See [Extraction Setup](#) on page 206.

Extraction Setup

Perform your pre-route extraction setup by choosing *Analyze – Preferences* from the PCB SI menu. The Analysis Preferences dialog box appears as shown in the following figure.

Figure 6-1 PCB SI Analysis Preferences Dialog Box – InterconnectModels Tab



Unrouted Interconnect

Use the *Interconnect Models* tab to establish default values that determine how interconnect is modeled during simulation both before and after routing and how crosstalk and SSN analysis is performed. The simulator cannot calculate length or impedance of a ratsnest, even if you have defined a board stack-up. This tab specifies what impedance and what portion of the manhattan distance to use for ideal Tlines produced during net extraction. The *Default Prop Velocity* value is used in conjunction with the manhattan distance to calculate delay in time.

Unrouted Interconnect Models

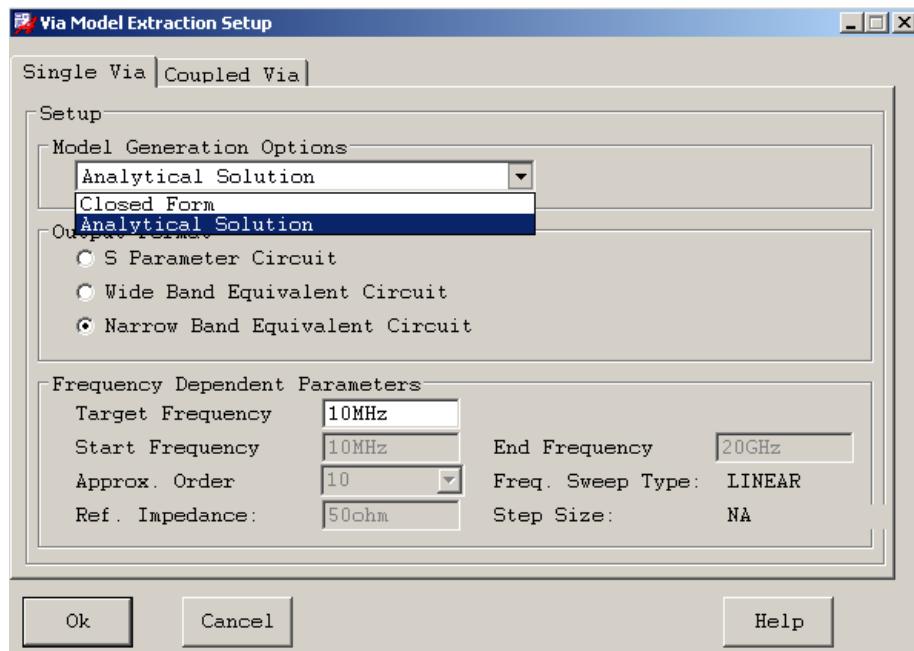
For pre-route signal integrity analysis, SigNoise models hypothetical traces using a percent Manhattan value, a default impedance value, and a default propagation velocity.

Routed Interconnect Models

For post-route signal integrity analysis, you can specify a field solver cutoff frequency and the way that vias are modeled. The field solver cutoff frequency establishes a bandwidth within which interconnect parasitics are solved. This prompts the SigNoise field solver to generate frequency-dependent transmission line models in the interconnect library. The default cutoff frequency of 0GHZ directs the field solver to disregard signal frequencies. This saves computation time, but may not be as accurate as frequency-dependent interconnect modeling.

To define how vias are modeled during simulation, first select whether each single via should have a closed form model, or use the preferred Analytical Solution as configured in the *Via Model Extraction Setup* dialog box shown in Figure 6-2. You can set up coupled vias using only the Analytical Solution; however, in cases such as incorrect stack-ups or field solver limitations that make the Analytical Solution unfeasible for coupled vias, the operation defaults to a single via. This single via may create a closed form, narrow or wide band, or S-Param solution. Coupled vias are detected by way of the settings in the *Diffpair/Via Coupling Window* field of the *Interconnect Models* tab as well as the settings in the Via Model Extraction Setup dialog.

Figure 6-2 Via Model Extraction Setup Dialog Box



Topology Template Formats

You can extract a net that is either routed or unrouted into SigXplorer for exploration. The resulting topology template is capable of providing a routed or an unrouted view of that net on the SigXplorer canvas.

A routed net can be extracted with either a routed or an unrouted topology format. However, unrouted nets are always extracted with an unrouted topology format created using an ideal transmission line to represent the connections (in either manhattan or actual length). Both formats are as shown in [Figure 6-3](#).

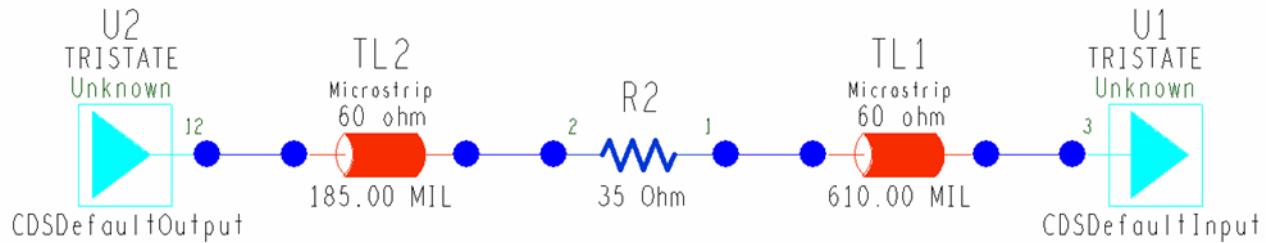
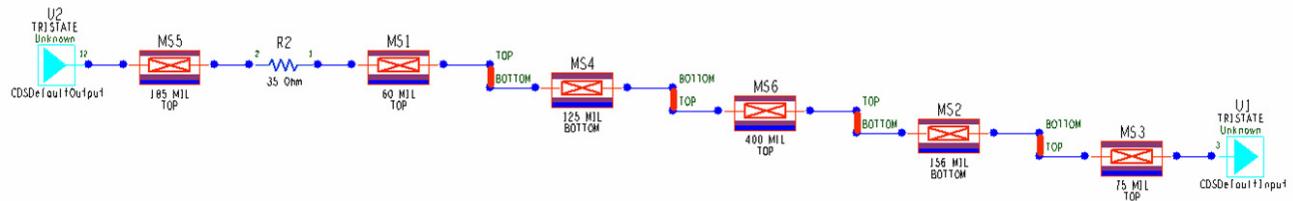
Note: The process of getting the proper topology extracted into SigXplorer is very much dependant on the etch and component information for the net in the database.

Allegro PCB SI User Guide

Topology Extraction

Figure 6-3 Topology Formats as Displayed in SigXplorer

Routed Format



Unrouted Format

Physical and Extended Nets

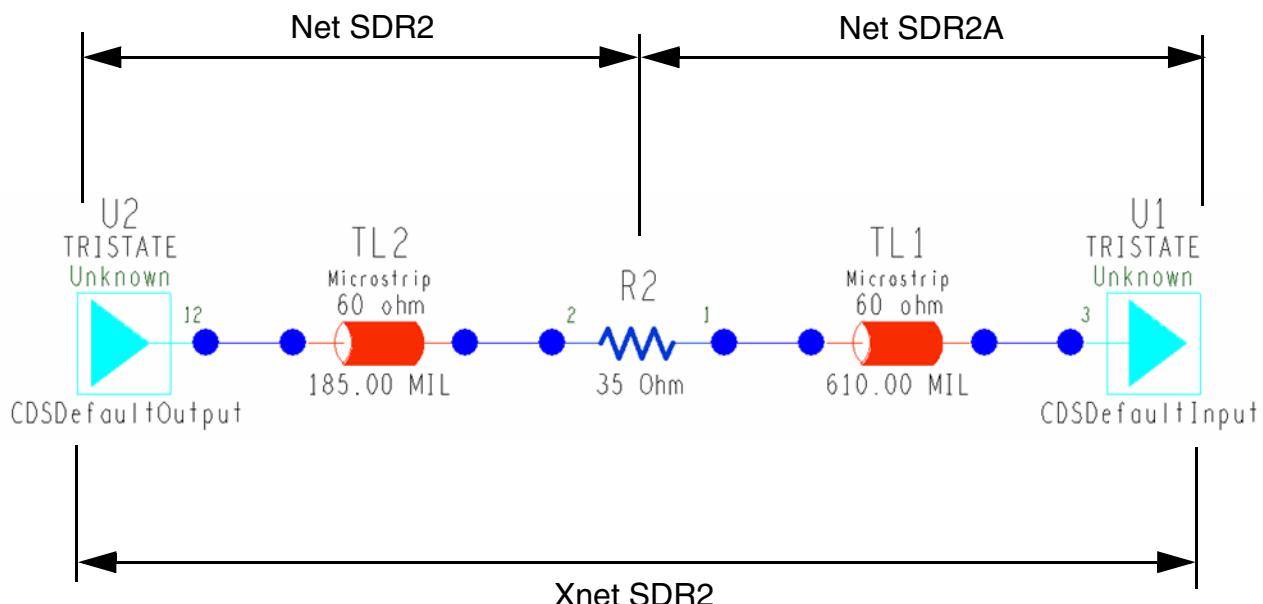
Physical Net

A physical net is a connection between two or more components.

Extended Net (Xnet)

An Xnet is a connection between drivers and receivers. An Xnet includes all the drivers and receivers connected to each other *plus* any discrete components that are connected to the Xnet. Xnets pass through devices such as resistors and capacitors as shown in the following figure.

Figure 6-4 An Xnet containing two physical nets



PCB SI and Allegro Editor use the same design database (.brd file). They both understand physical nets. However, PCB SI also understands Xnets and multi-board connectivity. In other words, it understands the connections between Xnets (from one board to another). This requires the ability to trace through connectors with detailed mapping information for connector pins.

Topology Template Extraction

A topology template provides the ability to capture and save the design intent for a net. The design intent includes the net's schedule, impedance, delay, and termination strategy.

You can extract a topology template for simulation and analysis within SigXplorer by:

- probing a net at the board level within SI.
- using Constraint Manager.

Probing a Net to Extract a Topology Template

This method extracts the routed interconnect of the net as part of the topology template. You do this by accessing the Signal Analysis dialog box in SI.

To extract a topology template by probing a net at the board level

1. In SI, choose *Analyze – Probe*.

The Signal Analysis dialog box appears as shown in [Figure 6-5](#) on page 212.

2. Probe (click) the net in the design whose topology you want to extract.

The name and other information for the selected net is displayed in the dialog box.

Note: You can also browse for a net by clicking *Net Browser*.

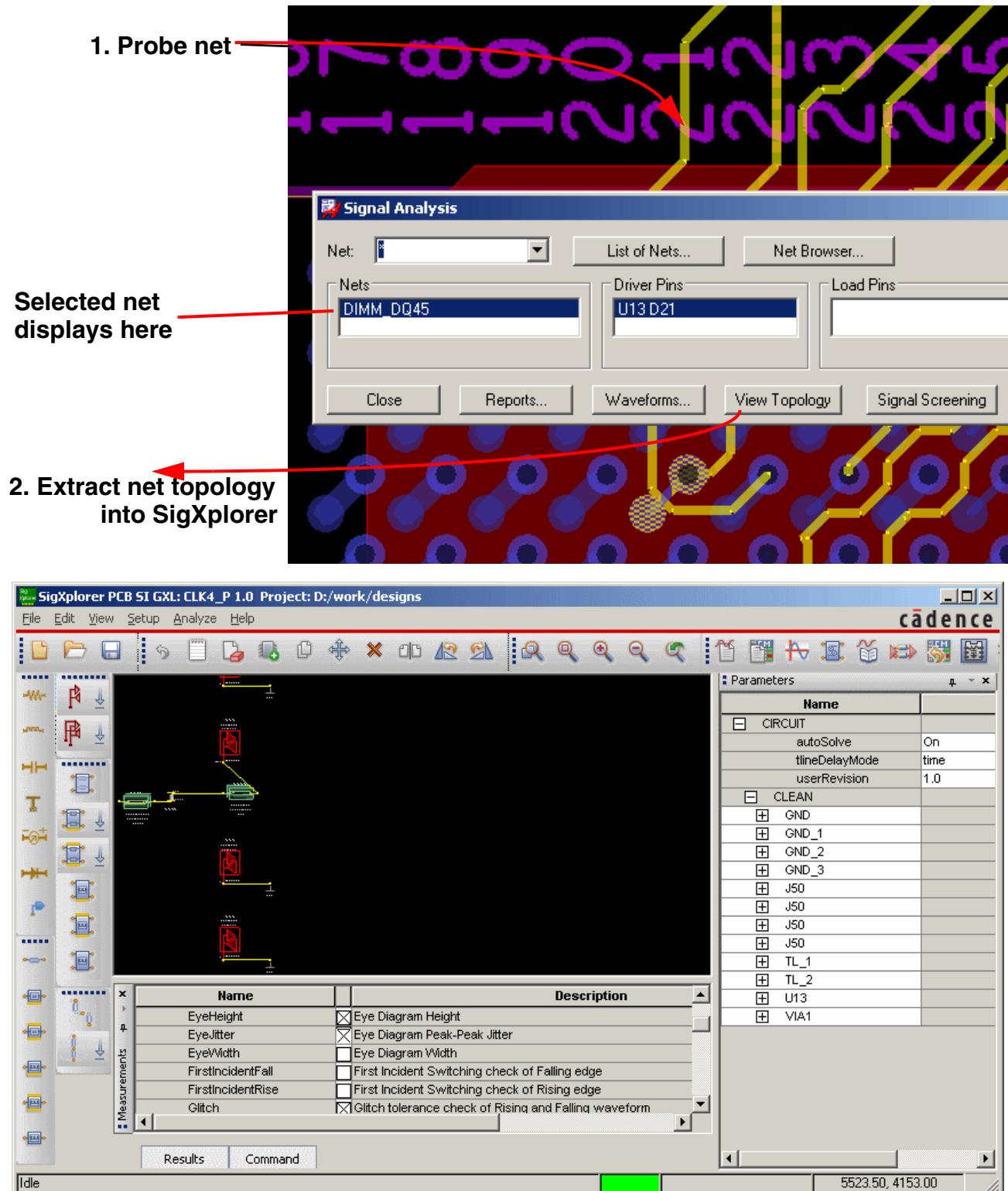
3. Click *View Topology*.

The topology template is extracted, SigXplorer launches and displays the topology in the SigXplorer canvas.

Allegro PCB SI User Guide

Topology Extraction

Figure 6-5 Extracting a Topology at the Board Level



Using Constraint Manager to Extract a Topology Template

Constraint Manager lets you start SigXplorer for a selected Xnet. Once started, you can browse or inspect templates and apply them to the nets of the Xnet. Using SigXplorer enables you to cut down on the time you spend determining the appropriate circuit topologies that meet timing and signal integrity constraints. For topology template extraction in Constraint Manager, the first step is to set up the option to extract.

To set up for topology template extraction

1. In Constraint Manager, choose *Tools – Options*.

The Options dialog box appears as shown in [Figure 6-6](#) on page 214.

2. In the *Electrical CSet Extraction* area, choose whether or not you want to include routed interconnect.

Note: For pre-route net topology extraction, disable the *Include routed Interconnect* option.

3. Click *OK* to dismiss the dialog box.

Note: The interconnect models you extract have ideal transmission line models in accordance with the preferences set in the *Unrouted Interconnect Models* section of the Analysis Preferences dialog box as shown in [Figure 6-1](#) on page 206.

To extract a topology template

1. Right-click on the Xnet object in the spreadsheet that you wish to extract.

A popup menu appears with options as shown in [Figure 6-6](#) on page 214.

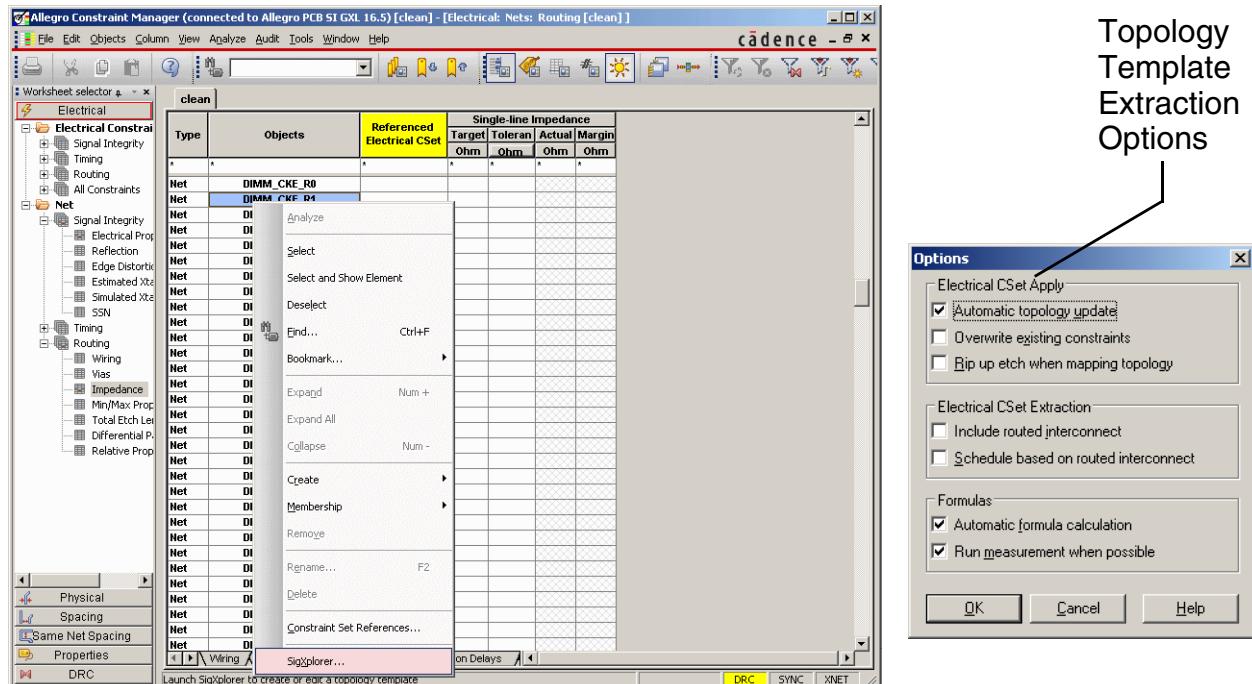
2. Choose *SigXplorer*.

The topology is extracted and is displayed on the SigXplorer canvas.

Allegro PCB SI User Guide

Topology Extraction

Figure 6-6 Constraint Manager Topology Extraction Options



Benefits of Topology Templates with Routed Interconnect

PCB Designers can highlight nets, turn layers on and off, and dim the view to get a look at the way nets are routed in a design. However, using this method may have mixed results for the SI Engineer needing to track down issues such as scheduling DRCs.

By extracting a net topology along with its routed interconnect into SigXplorer, you can view the RefDes and pins, vias, length and impedance information, and what layers the traces are on. It is not necessary to have a deep understanding of the Trace models to do this.

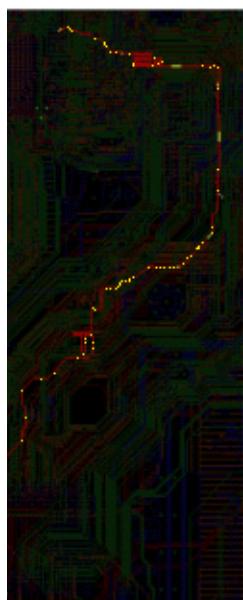
Figure 6-7 compares the view of a routed net in the design window to that of its routed net topology on the SigXplorer canvas.

You extract a net along with its routed interconnect by:

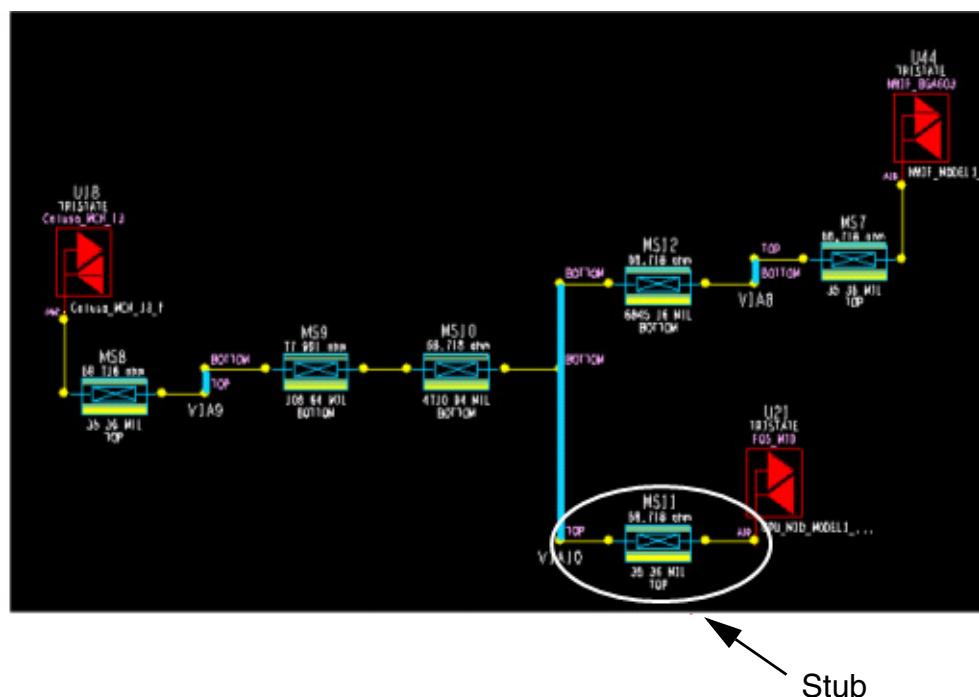
- probing the net from within SI.
- extracting the topology template using Constraint Manager with the *Include routed interconnect* option enabled (see the topology template extraction options in Figure 6-6 on page 214).

Figure 6-7 Viewing a Routed Net vs. Viewing Routed Net Topology

Highlighted in
SI



Extracted and Displayed in SigXplorer



Topology Simulation

Once you extract a net topology into SigXplorer, you are ready to begin exploring (simulating) the topology to determine and define an appropriate set of electrical constraints. This process is known as Solution Space Analysis. For further details on this process, refer to [Chapter 7, “Solution Space Analysis.”](#)

For information on setting simulation preferences and simulating a net topology using SigXplorer, refer to the [*Allegro SI SigXplorer User Guide*](#).

Determining and Defining Constraints

Overview

Using SigXplorer you define, examine, modify, and compare net topologies to determine the best solution for a design. You create topology templates that represent physical wiring strategies and component selection based upon the rules and constraints of your design. You create constraint sets to define both electrical and physical constraints that drive printed circuit board routing.

As the signal integrity engineer, your job is to simulate topologies and analyze circuits for propagation delays, switch/settle times, minimum and maximum flight times, overshoot and undershoot times, and signal monotonicity. You can run a single simulation or a set of parametric sweeps to cover a variety of conditions.

Constraint sets are assigned to a net or groups of nets to quickly provide a foundation for correct-by-design floorplanning and implementation. You can make modifications to constraint sets for an entire group of nets.

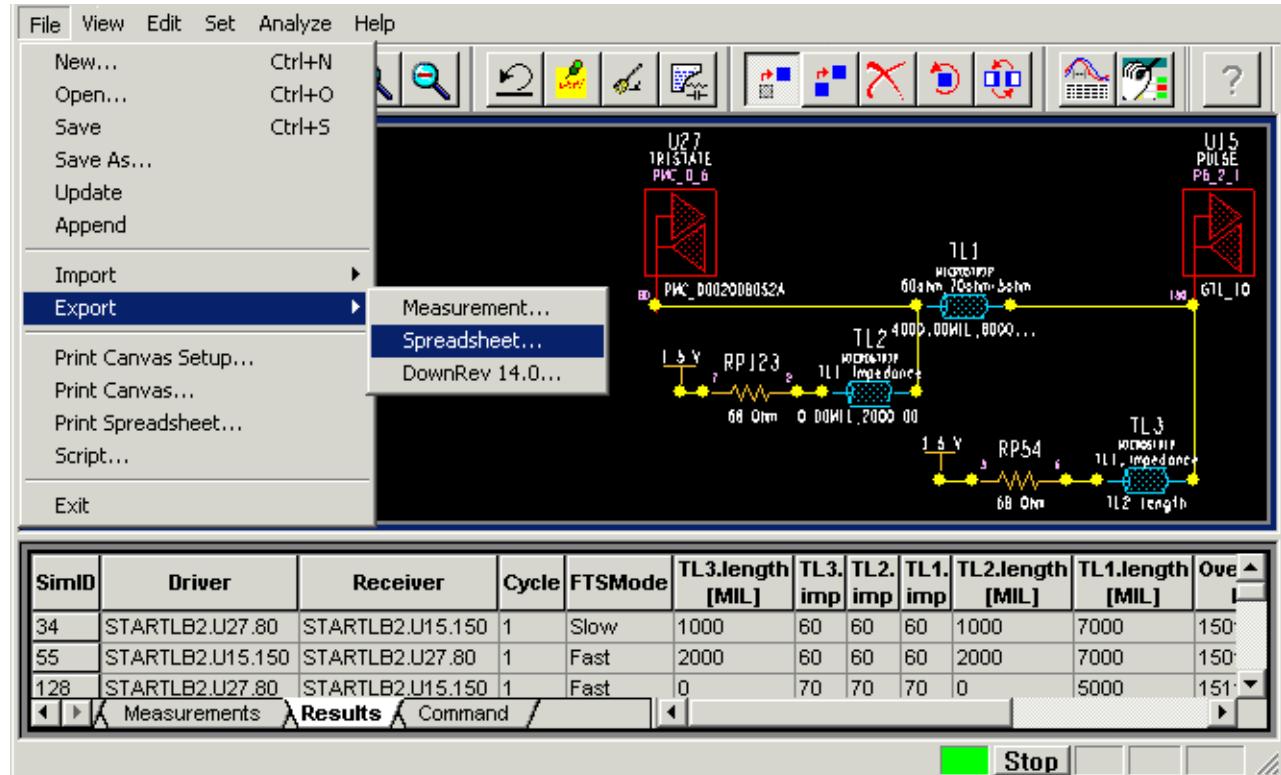
Solution Space Analysis

What is Solution Space Analysis?

Solution Space Analysis is a *methodology* used to model, analyze, and compare all possible combinations of conditions under which your design must operate (such as design and manufacturing variances). Results are collected and analyzed using the SigXplorer Results spreadsheet as shown in [Figure 7-1](#). You can also save data in a Microsoft® Excel compatible format.

The ultimate goal of Solution Space Analysis is to find a routing solution for high-speed nets that is robust - one that allows you to work under all conditions of fast and slow devices, high and low impedance, etc. Once that goal is achieved, use the results data is used to drive the routing rules (minimum and maximum lengths, and so forth) for the layout designer. By performing a thorough analysis to define rules before routing, you minimize the chance of re-routing the design and maximize the chance of first-pass success.

Figure 7-1 Solution Space Analysis – Comparing Results



Solution Space Analysis – Stage 1

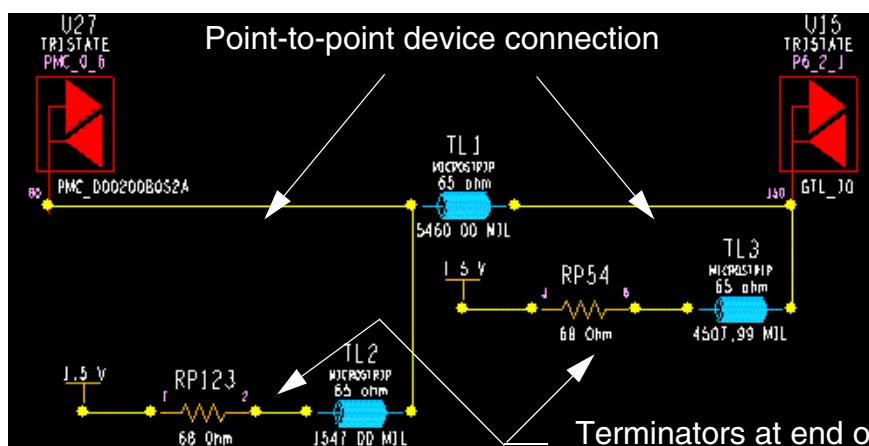
The first stage of Solution Space Analysis involves:

- extracting or creating topology to analyzed determine:
 - pin ordering
 - termination strategies
 - rat-T positions (if any)
- identifying and entering nominal values for all parameters such as:
 - board impedance
 - trace velocity
 - terminator value
 - segment lengths

Solution Space Analysis is based on the premise that all of the signals on a bus have the same timing and signal quality constraints. As such, finding an electrical and physical solution for one bit of the bus constitutes finding a solution for the whole bus. At this point in the design process, you can enter any electrical constraints (such as flight time, overshoot, undershoot) for the bus being analyzed. Any solution space simulations that violate the specified constraints should be flagged as errors.

You can extract the net from a partially or fully placed board design, or create it from scratch using SigXplorer. The initial step is to define discrete devices, intended pin ordering (connectivity), as well as the nominal (ideal) values for all topology properties.

Figure 7-2 Solution Space Analysis – Pin Ordering



Solution Space Analysis – Stage 2

The second stage of Solution Space Analysis involves:

- identifying manufacturing variances that you want to include in the analysis such as:
 - trace impedance
 - trace velocity
 - fast / slow components
 - device values (terminators)
 - power supplies (if applicable)
- identifying initial ranges for design rule parameters such as min/max lengths.

During stage 2, you begin to make some educated guesses about the target topology and the minimum/maximum conditions of length, impedance, driver speed, and so forth, under which the circuit operates. You then run min/max simulations, or small simulation sweeps, to see the circuit's behavior under a range of conditions. The swept variables usually fall into the following two categories:

Manufacturing Variances

Driver speeds, TLine impedances, resistor tolerances, and so on must be accounted for, and the circuit must work under all possible combinations of conditions. The designers may have some control over these variances (for instance, specifying that board impedance must be 65 ohms +/- 5%), but the variances must still be accounted for. The effect of the manufacturing variances are simulated, but are usually not passed into the layout process via a topology template.

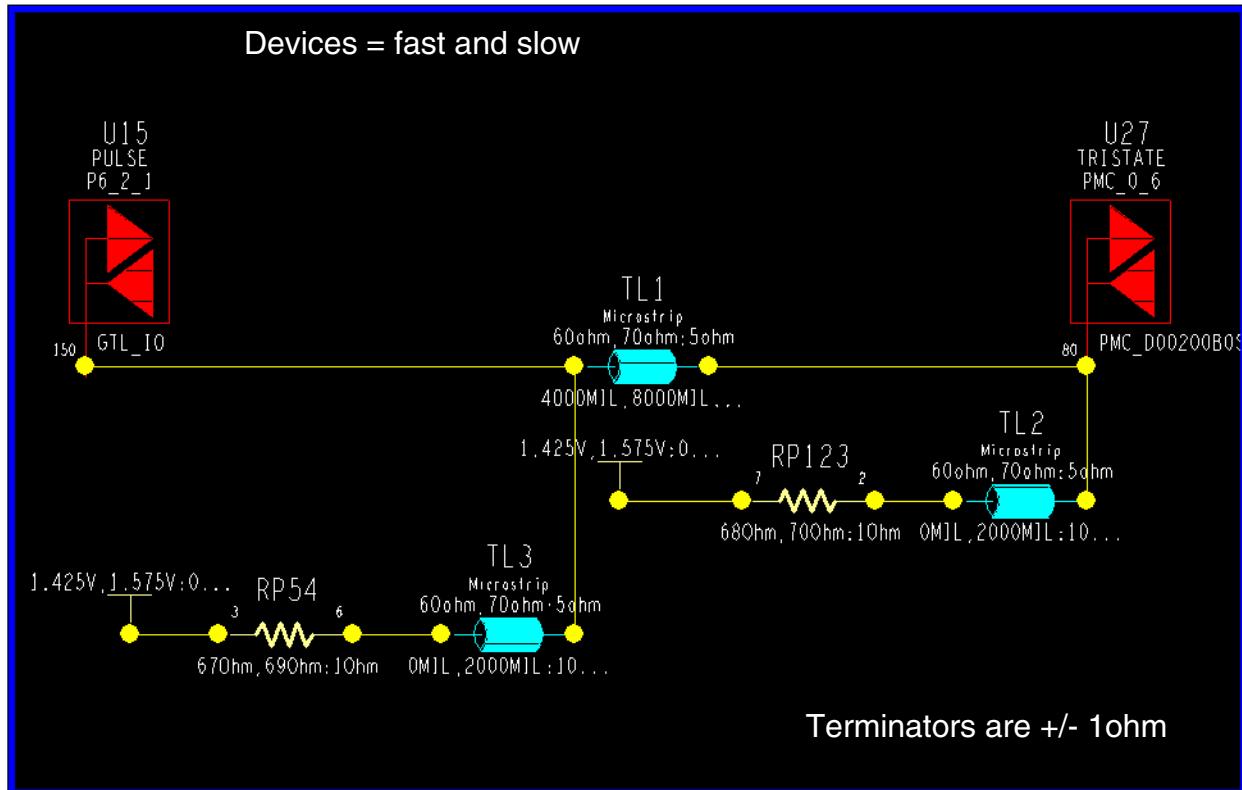
Design Variances

Once the manufacturing variances are accounted for, the designer's task is to find the widest possible range of design variances (such as routing lengths) within which the design will function. The resulting constraints (pin ordering, min/max routing lengths, length matching requirements) are passed downstream to the physical design process as design constraints. These are usually contained in a topology template (.top) file and applied to the design database through PCB SI or Allegro Editor.

Allegro PCB SI User Guide

Determining and Defining Constraints

Figure 7-3 Solution Space Analysis – Simulation Sweeps



Solution Space Analysis – Stage 3

The third stage of Solution Space Analysis involves:

- composing a master list of all variables including their ranges for analysis. See [Figure 7-4](#).
- identifying dependencies between these variables, based on how the design is to be implemented.

For example:

- traces on the same layer will have identical characteristics.
- resistors in the same RPAK will match closely.

- developing a simulation strategy based on combinations you plan to analyze.

The following figure shows an example of a master variable list that you could compose from scratch.

Figure 7-4 Master Variable List Example

Parameter	Min	Typ	Max	# Steps
P6 Speed	Fast		Slow	2
440FX_Speed	Fast		Slow	2
TL1 Impedance	60 ohms		70 ohms	2
TL1 Velocity	5400 mils/ns		6600 mils/ns	2
TL1 Length	4000 mils		8000 mils	2
TL2 Impedance		TL1 Impedance		1
TL2 Velocity		TL1 Velocity		1
TL2 Length	0 mils		2000 mils	2
TL3 Impedance		TL1 Impedance		1
TL3 Velocity		TL1 Velocity		1
TL3 Length	0 mils		2000 mils	2
RP A Impedance	67 ohms		69 ohms	2
RP B Impedance	67 ohms		69 ohms	2
Total Combinations				512

The assumption is that board impedances are 65 ohms +/- 5 ohms and the traces are routed on a surface layer where velocities are expected to vary between 5400 mil/ns and 6600 mil/ns. The terminators used are 68 ohms +/- 1 ohm, and the termination voltage supply is specified to 1.5V +/- 5% (1.425 - 1.575V). The processor and the chipset are either fast or slow (the information for fast/slow buffer behavior and min/ max package parasitics is contained in the IBIS signal models provided by the semiconductor manufacturers). For this

small example, we have $2^{**}11$ (2048) permutations of worst-case conditions alone, and this is a simple case!

If you know that all of the segments of a trace will be routed on the same layer, then varying the impedance and velocity of TL1, TL2, and TL3 independently is pessimistic—the values of impedance and velocity for all three segments should be swept in lock step. Doing this would reduce the 6 variables for velocity and impedance down to 2—a savings of $2^{**}4$ (16) in edge case sweeps.

Sensitivity analysis is useful to determine if any of the remaining variables have a small enough effect on circuit behavior to ignore — thus reducing the total number of sweep simulations that you need to run. For the moment, assume that you can ignore the tolerance on the termination resistors and voltages. Minimizing the number of variable sweeps leaves you with $2^{**}9$ (512) simulations.

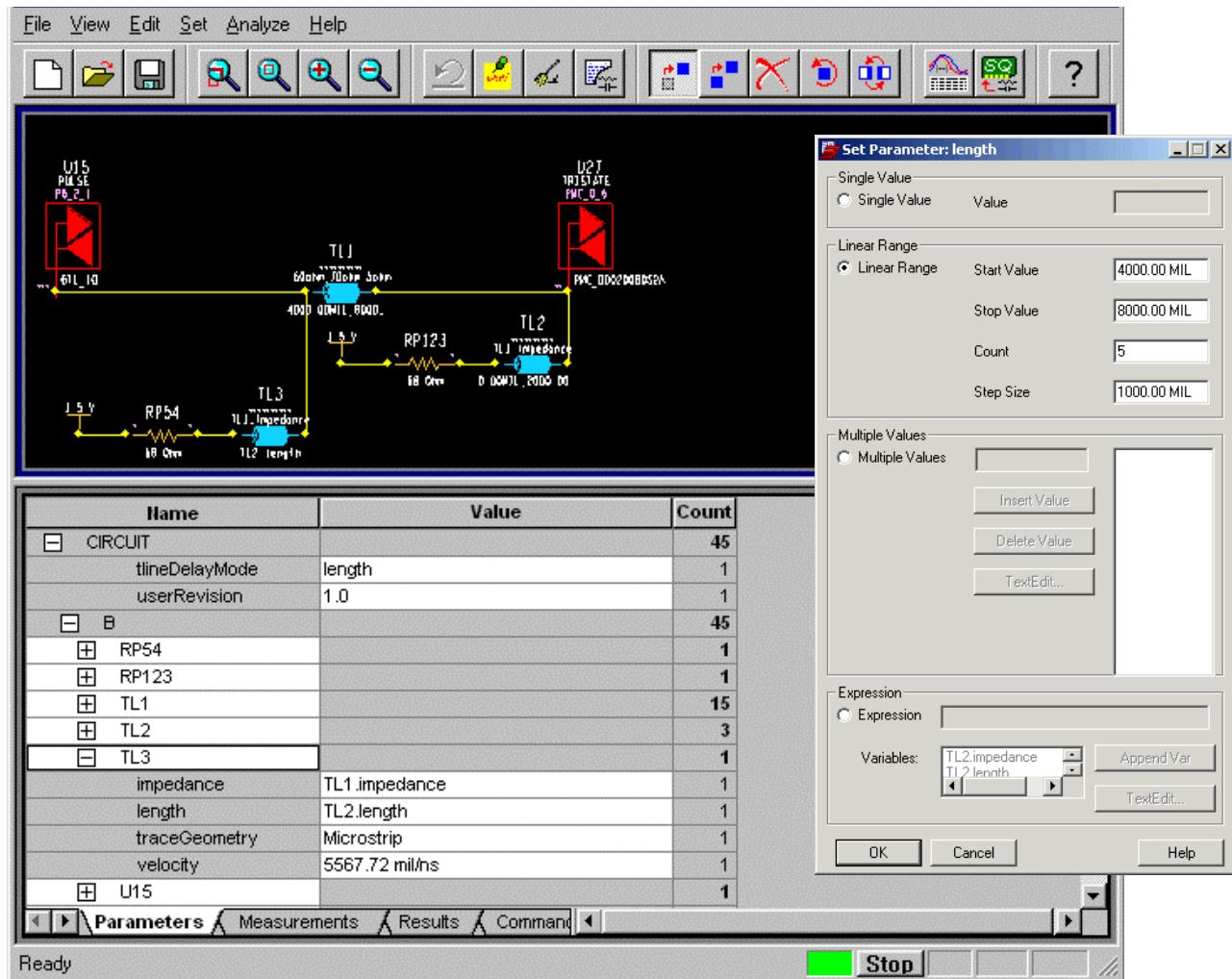
For simpler topologies, it may be possible to perform worst-case condition analysis by directly modeling the cases that will cause the best- and worst-case switching behavior. This is traditionally how worst-case analysis has been performed, with swept parameter analysis reserved for complex topologies.

Even when you are to perform swept parameter analysis, directly modeling and analyzing select cases allows you to identify and resolve problems before running large amounts of simulations.

Allegro PCB SI User Guide

Determining and Defining Constraints

Figure 7-5 Worst-case Analysis



For further details on simulation sweeping, see [Parametric Sweeps](#) on page 228.

Solution Space Analysis – Stage 4

The fourth stage of Solution Space Analysis involves:

- running simulations and gathering results from the SigXplorer spreadsheet.
- evaluating results and identifying cases (combinations of variables) that cause topology to fail (not meet design goals).
- simulating individual cases, analyzing, correcting the design if needed, and iterating.

You then open up the design range and repeat the process until you have determined the maximum allowable range of routing lengths has been determined.

Solution Space Analysis – Stage 5

The fifth stage of Solution Space Analysis involves:

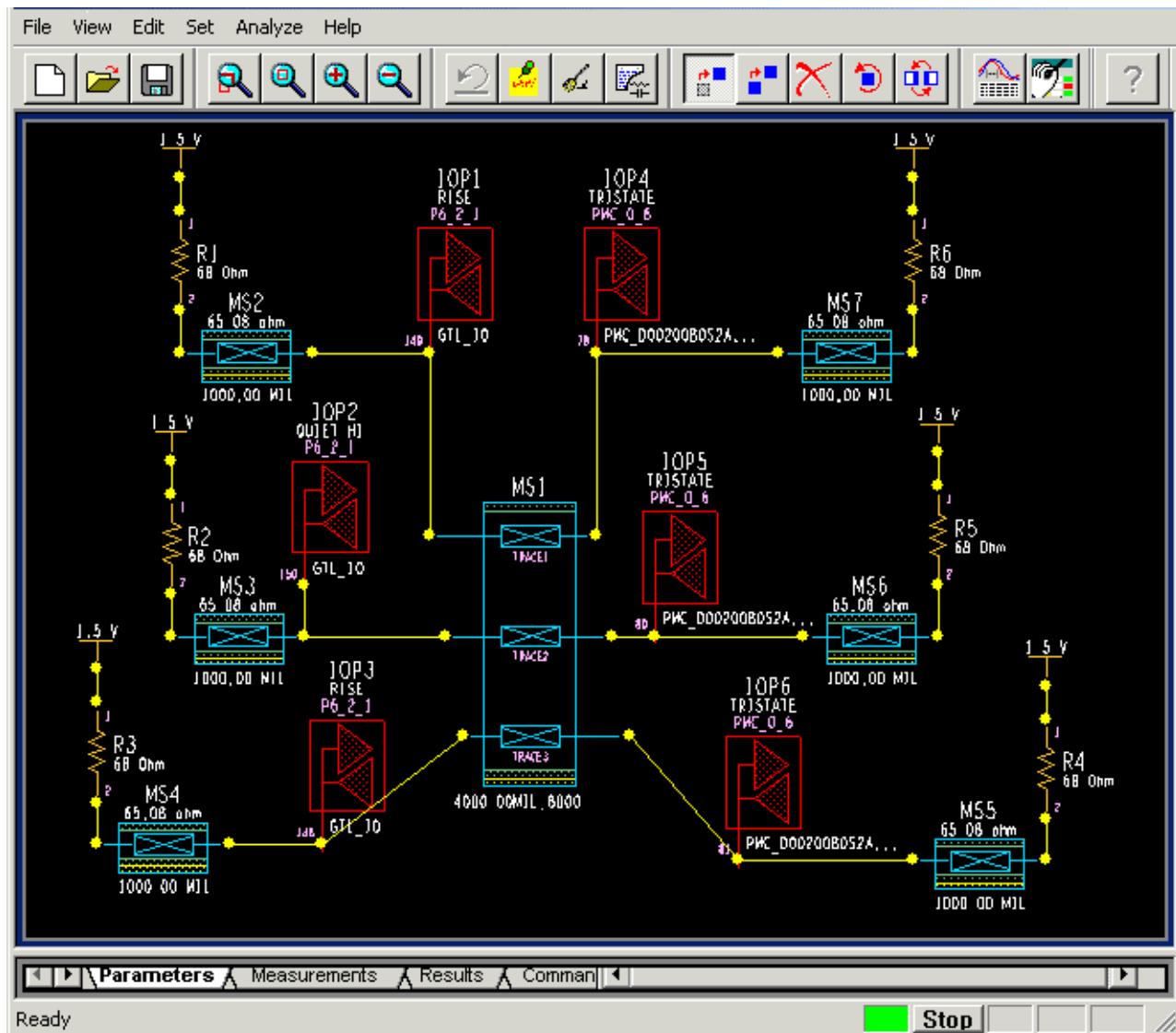
- ensuring that crosstalk timing shifts are within crosstalk budget.
- modifying single-line topology to model coupling where appropriate.
- evaluating different line width / spacing rules for timing impact.

The solution thus far meets the parameters for minimum and maximum flight time.

The timing budget for high-speed buses typically includes a maximum timing shift budget due to crosstalk between the bus bits. You can use Signal Explorer Expert to model the realistic worst-case scenario (longest coupled length, fastest drivers, and so forth) to measure the crosstalk-induced timing shifts for a given spacing rule.

Allegro PCB SI User Guide
Determining and Defining Constraints

Figure 7-6 Solution Space Analysis - Single Line Solution



Solution Space Analysis – Stage 6

The last stage of Solution Space Analysis involves:

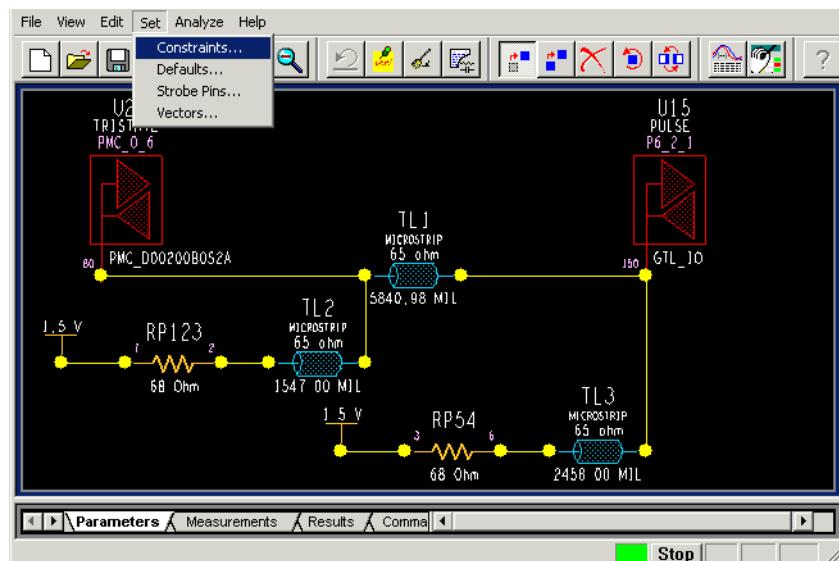
- creating a final topology template from the following analysis results.
 - Segment min/max lengths
 - Parallelism rules

Note: Do not include all variances in the final topology template. Trace min/max impedances are often not included in the router's rules.

Once Solution Space Analysis is complete, you can create the topology template (electrical and physical constraint set) to be driven into placement and routing. This is typically a subset of the information used for simulation.

For example, if the target impedance of the board is 65 ohms, and +/- 10% impedance simulation is performed, you would not want the 58- and 62-ohm values driven into the router. You should design the board for 65 ohms, with the understanding that 10% impedance control will result in 58- to 62-ohm impedance.

Figure 7-7 Solution Space Analysis - Final Topology Template



Parametric Sweeps

Simulation sweeping is based on varying combinations of the following criteria.

- Varying part parameter values
- Varying driver slew rates
- Sequencing active drivers

Sweeping by part parameter values entails traversing a set or range of values (sweep count points) that you specify for eligible sweep parameters through a set of simulations. SigXplorer calculates the total number of simulations based on the number of sweep count points required for each sweep parameter.

Sweeping by driver slew rate is accomplished by selecting a set of FTS Mode target rates from the Simulation Mode section of the *Simulate* tab in the Analysis Preferences dialog box.

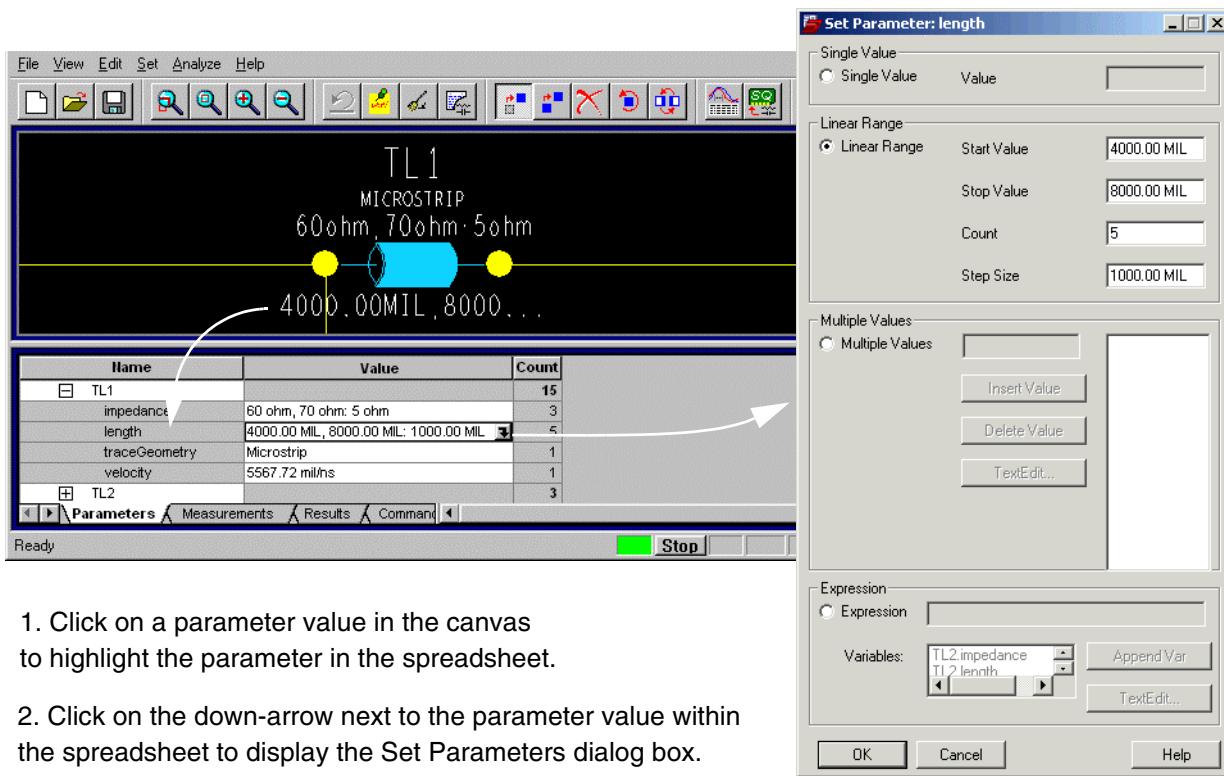
Sweeping by sequencing active drivers is accomplished by selecting *All Drivers* sweep mode to sequence through eligible IOCells with each one, in turn, driving a simulation.

When you specify multiple sweep criteria, SigXplorer employs a hierarchical ordering when performing the simulations. For example, if you select multiple FTS Modes as well as several part parameter values for sweeping, then all part parameter sweeps are executed for each selected FTS Mode. Additionally, if you also select *All Drivers*, then part parameter sweeps for each selected FTS Mode will execute as each driver activates in sequence.

Allegro PCB SI User Guide

Determining and Defining Constraints

Note: Simulation sweeping is only available in certain versions of SigXplorer. Sweep Parameter Setup



1. Click on a parameter value in the canvas to highlight the parameter in the spreadsheet.
2. Click on the down-arrow next to the parameter value within the spreadsheet to display the Set Parameters dialog box.

Specifying Part Parameter Values for Sweeping

All parameter attribute values, including parameter values that you can sweep, are accessible for viewing and editing through the *Parameters* tab of the *SigXplorer* spreadsheet.

Parameter values that you can sweep include:

- a single number value.
- a linear range of number values specified as start and stop values and a step size for iterating from start (the minimum value) to stop (the maximum value).
- a list of discrete number values.
- an expression string composed of operators, functions, and references to other parameters.

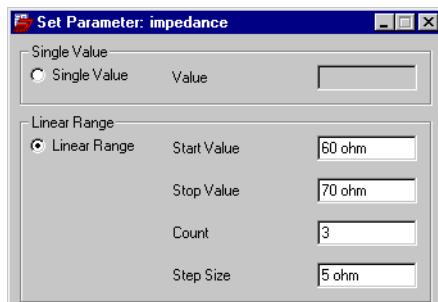
When you use an expression to define a parameter attribute value that references a second parameter attribute value that is defined as a range or list, the first parameter tracks the

second parameter as it changes during simulation sweeping. By defining an expression that references another parameter value and adds a constant, you can track the first parameter value with an offset.

When you delete a part, any references to that part's parameter values are invalidated and display in red within the spreadsheet.

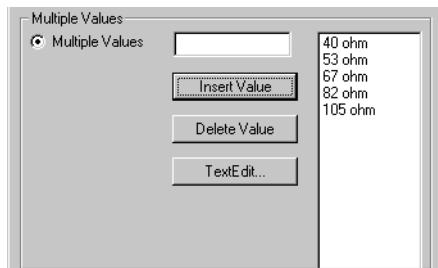
Note: You can save a topology that contains invalid references, but you cannot simulate it.

Figure 7-8 Setting Sweep Parameters using the Set Parameters Dialog Box



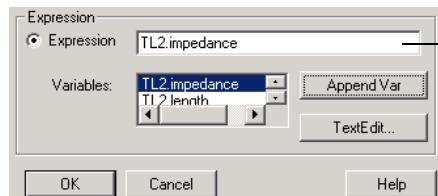
Using a Linear Range

Count value determines the number of sweep count points.



Using Multiple Values

Discrete values determine the number of sweep count points.



Using an Expression

Expression listed, as well as parameters referenced in the expression determine the number of sweep count points.

Controlling Sweep Sampling and Coverage

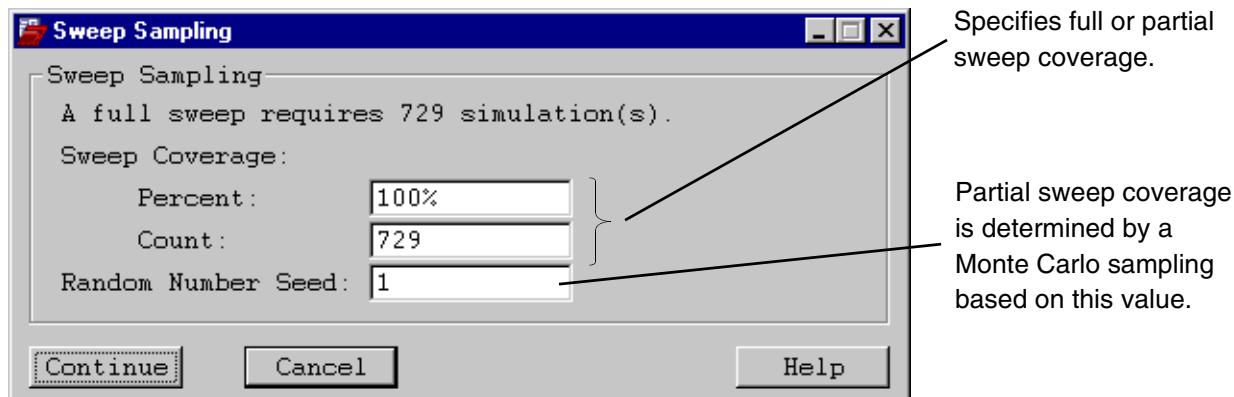
SigXplorer lets you control sweep sampling. After you are set up to perform simulation sweeps, choose *Analyze — Simulate*. The Sweep Sampling dialog box appears before an active sweep begins.

You can specify full or partial sweep coverage in this dialog box by:

- defining sweep samples as a percentage of full coverage.
- specifying an explicit number of simulations.
- specifying a seed number for random sampling.

Partial sweep coverage is obtained by randomly sampling the full solution space using Monte Carlo methods. To vary sample point sets, SigXplorer selects sweep count points based on the specified random number seed.

Figure 7-9 The Sweep Sampling Dialog Box



Sweep Results

When parametric sweeping is invoked, SigXplorer initializes SigNoise which sweeps through the required series of simulations. Sweep results display the *Results* tab of the SigXplorer spreadsheet.

The sweep report contains information on topology, swept elements, driver and load names, impedance and delay variables. You can choose *File — Export Spreadsheet* to save simulation sweep results in a tab-delimited text file. The contents of this file are suitable for import into an external spreadsheet program.

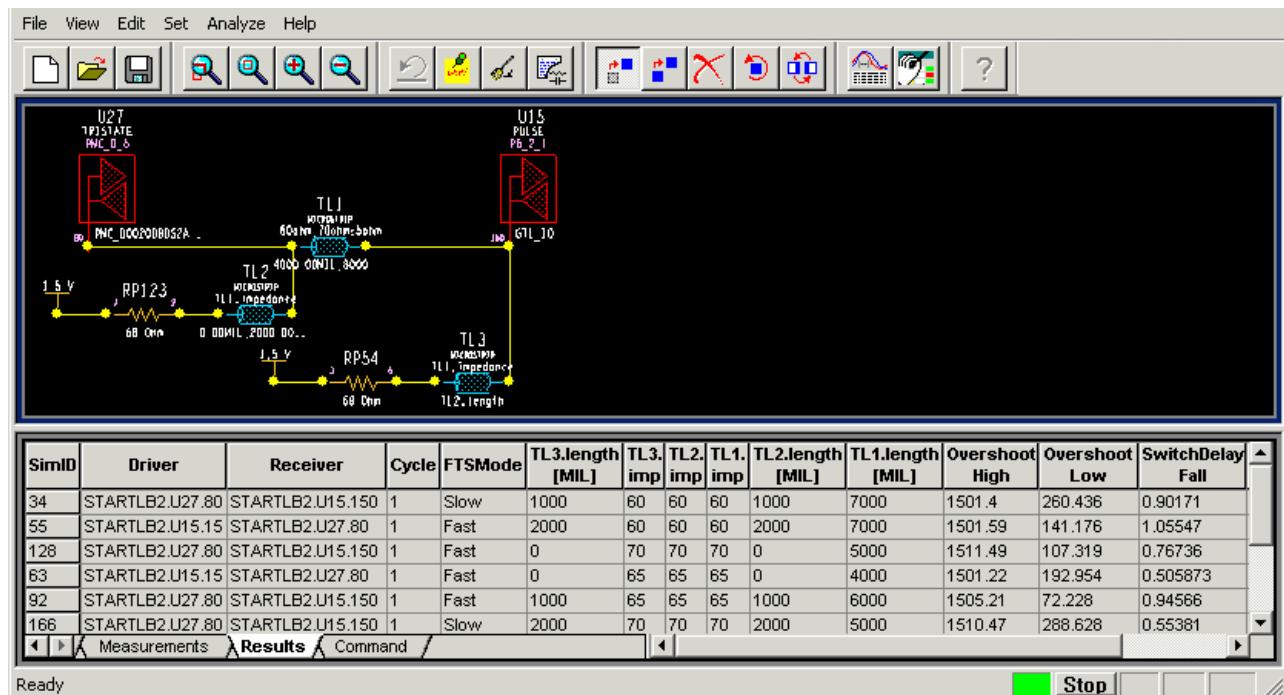
Waveforms

The parametric sweep function does not produce waveforms directly. However, when viewing the sweep results, you can click to select a row in the spreadsheet which re-runs that single simulation and opens SigWave to display the resulting waveforms.

Allegro PCB SI User Guide

Determining and Defining Constraints

Figure 7-10 The Sweep Report



Saving and Restoring Sweep Cases

You can save and restore sweep simulation data; this enables you to view waveforms from any prior sweep iteration. This eliminates the need of having to manually reset simulation parameters and perform re-simulations to view waveforms in *SigWave* resulting from prior sweep simulations. When you elect to save sweep cases, the waveforms as well as the environment details which existed at the time of the simulation are saved in a case directory. When the sweep case is restored, you return to the same state ensuring the data accuracy of your waveforms.

Note: Saving waveforms from sweeps can consume large amounts of disk space.

Sweep Case Data

Saved sweep cases are comprised of the following data.

- Waveforms
- Topology file
- *SigNoise* preferences

Allegro PCB SI User Guide

Determining and Defining Constraints

- Results spreadsheet data

For further details on parametric sweeps, refer to the *SigXplorer User Guide*.

Defining High-Speed Constraints

What is a Constraint?

A constraint is a *user-defined restriction* applied to an element in a design. When you define a constraint and apply a value, PCB SI adheres to that constraint in both automatic and interactive processing of the design element. When constraint violations are detected, they are flagged graphically in the design with DRC (Design Rule Checking) markers as well as in the Constraint Manager worksheet cells using the color red.

You define high-speed constraints using the results from Solution Space Analysis (see [Solution Space Analysis – Stage 6](#) on page 227) to prepare your design for layout and routing.

What is a Constraint Set?

A constraint set is a *collection of constraints* which defines rules that are applied as you create the interconnections of an individual net. Within the PCB SI high-speed environment, you create electrical constraint sets (ECSets) and assign them to nets in your design to control their electrical behavior.

Although an ECSet is applied to an individual net, other nets may contribute to the measurement of a constraint (for example, crosstalk). The ECSet applies to all parts of the net, regardless of subclass or layout area.

Certain electrical constraints are actually properties that can be assigned in Design Entry HDL and are sent directly from the schematic to the board file. Some examples are:

- MIN_LINE_WIDTH
- MAX_VIA_COUNT
- PROPAGATION_DELAY

Creating ECSets

Create and assign ECSets to net objects using Constraint Manager. Within the PCB SI environment, you can start Constraint Manager by:

- Clicking  on the toolbar in either PCB SI or SigXplorer.
- or -

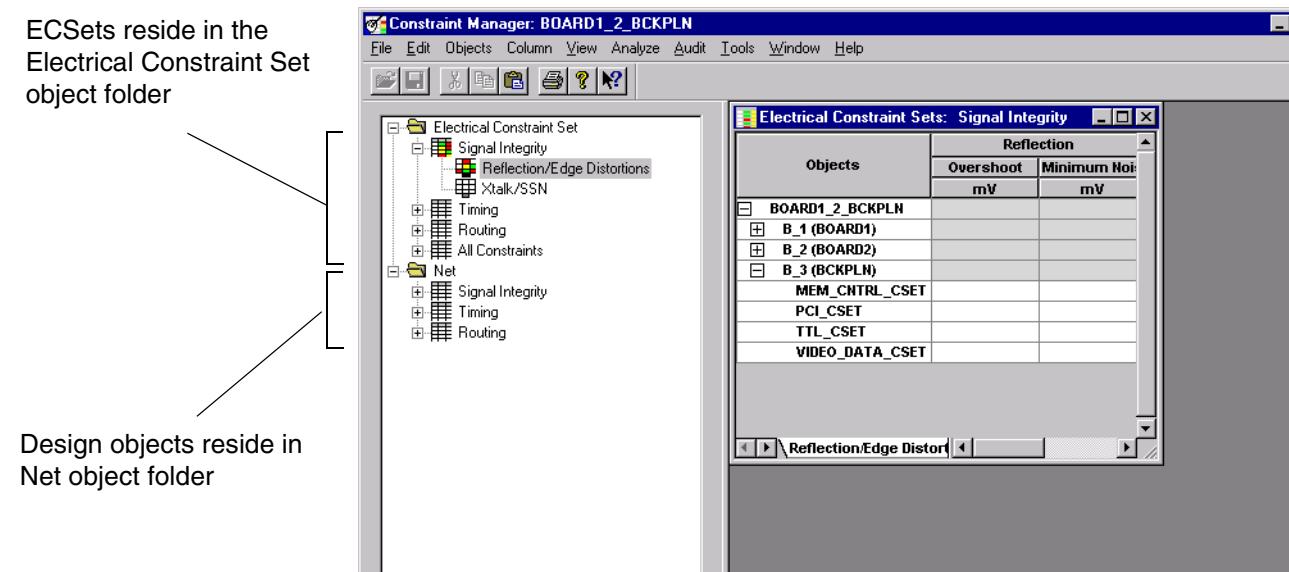
Allegro PCB SI User Guide

Determining and Defining Constraints

- Choosing *Setup – Electrical Constraint Spreadsheet* from the PCB SI menu bar.

The Constraint Manager graphic user interface appears as shown in the following figure.

Figure 7-11 Constraint Manager ECSet Folder Object Hierarchy



Within Constraint Manager, you can capture any or all electrical constraints, including topology-related information, in an ECSet. When you access an ECSet worksheet, objects are presented hierarchically. The *System* is the top-most object with lowest precedence and pin-pairs is at the bottom of the hierarchy with the highest precedence.

The *Signal Integrity* worksheet at the Electrical Constraint Set-level depicted in [Figure 7-11](#) shows the following objects and ECSets:

System	BOARD_2_BCKPLN
Designs	B_1, B_2, B_3
Electrical CSets	MEM_CNTRL_CSET, PCI_CSET, TTL_CSET, VIDEO_DATA_CSET

You define ECSets under the *Electrical Constraint Set* object folder. You can apply constraints subsequently to net-related objects.

As design requirements change, you can:

- edit the ECSet constraints. All net-related objects that reference the ECSet will automatically inherit these changes.
- assign a different ECSet, one that reflects a different rule-set, to the net-related object.
- define override properties on individual net-related objects. Cells with overrides are colored blue.

You can also define an ECSet based on the characteristics of a net or Xnet. Defining net-derived rules lets you create (or clone) rules based on the electrical characteristics of the physical net in your design.

Note: An ECSet also acts as a container for custom constraints, custom measurements, and custom stimulus. See Custom Constraints, Custom Measurements, and Custom Stimulus in the [*Constraint Manager User Guide*](#) for more information on these unique constraint types.

For further details on creating ECSets, see [*Objects – Create – Electrical CSet*](#) in the [*Allegro Constraint Manager Reference*](#).

The following tables show electrical constraints listed in the ECSet worksheets within Constraint Manager and the electrical behavior that they control.

Table 7-1 Electrical Constraints Affecting Automatic Routing

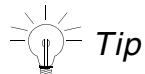
This constraint . . .	controls	
Propagation Delay	Delay in time or connection length restrictions between any two pins on a net or between any pin and a point.	
Relative Propagation Delay	Connections that are part of a match group. Note: Assigning a propagation delay constraint to one of the connections in a match group restricts all other connections in the group. Be careful not to assign conflicting propagation delay constraints within a group which might cause unpredictable results.	
Stub Length	The maximum length allowed for a stub.	
Max Via Count	The maximum via count for a net.	
Max Parallel	Coupled Length	The allowable length for selected signals to run parallel.
	Gap	The allowable gap between selected signals running parallel.

Table 7-2 Electrical Constraints Not Affecting Automatic Routing

This constraint . . .	controls
Min First Switch	The time a signal takes to reach the input low threshold voltage at the receiver.
Max Final Settle	The time a signal takes to reach the input high threshold.
Overshoot	The maximum overshoot value allowed on a net.
Max Xtalk	The limit of total crosstalk on a victim net from all aggressor nets.
Max Peak Xtalk	The limit of crosstalk on a victim net from a single aggressor net.
Max SSN	The maximum noise from simultaneous switching that is allowed on a net.
Min Noise Margin	The minimum noise margin tolerated for a net.

Referencing ECSets

When an ECSet is referenced from a net-related object, certain constraints are inherited while others are actually *applied* to the objects. For example, you must apply topology information since objects cannot simply inherit it due to the mapping that occurs between the ECSet and the net objects.



When you click in a worksheet cell, the source of the information, the ECSet name if inherited, appears in the status bar.

When an ECSet is updated from importing a topology template, the characteristics of the net-related objects must match those of the topology template. Otherwise, Constraint Manager will not refresh the ECSet with this new constraint information.

- Choose *Audit – Constraints* from the Constraint Manager menu bar to view a report of constraints that have net-related overrides.
- Choose *Audit – Electrical CSets* from the Constraint Manager menu bar to view a report of all objects referenced by each ECSet and to learn about inconsistencies between the ECSet and the Nets or Xnets that reference the ECSet.

For further details on how Constraint Manager maps ECSets to candidate nets, see [Objects – Electrical CSet References](#) in the *Allegro Constraint Manager Reference*.

ECSet Reference Rules

- All ECSets are presented under the appropriate Design or System and are referenced only by objects within the same Design or System.
- ECSets are referenced by any number of net-related objects (bus, differential pair, Xnet, or net) but an object may reference only one ECSet.

Setting Nets to Check Themselves for Crosstalk and Parallelism

To enable the nets in your design to perform a design rule self-check for crosstalk and parallelism (in addition to the checks the net makes against all other nets), turn on the feature by way of the Options tab in the Electrical Constraints dialog box. ([cns electrical](#) command).

Same net modifies the Segment Crosstalk report to include estimated crosstalk values of each net to itself (self crosstalk). The value of the control is printed in the simulations preferences section of the report, which you can create through the Probe command's (*Analyze – Probe*) Signal Analysis > Reports interface, or by way of the signoise batch command ([signoise](#)) from your system command prompt.

Version Compatibility

Enabling this command creates same net crosstalk records in your design database. Because such data is not supported in releases prior to 15.5.1, you must perform a database down rev in later releases to remove these objects. For releases earlier than 15.5.1, attempts to open designs containing same net DRC data will produce an error message and the design will not open.

Signal Integrity Analysis

Setting Simulation Preferences

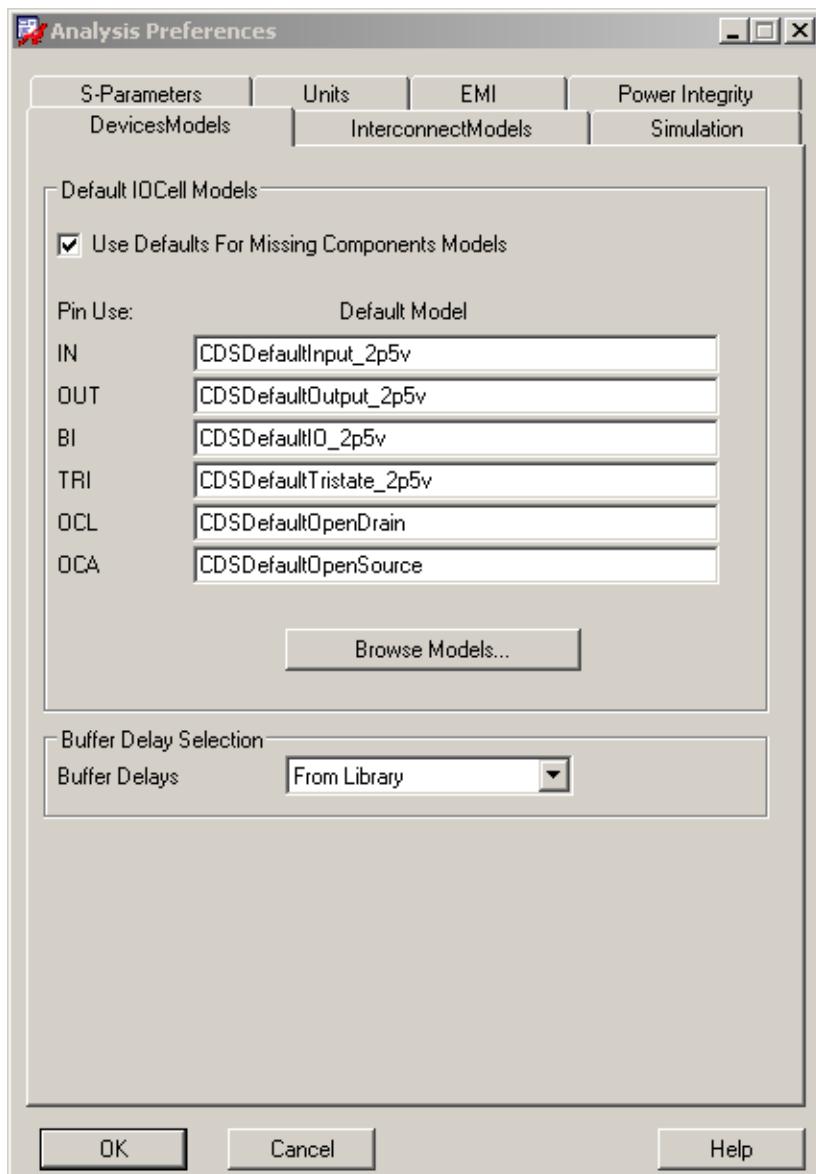
Before analyzing a design for signal integrity and EMI, you should set up SigNoise to perform simulation analysis according to your preferences.

To set simulation preferences

- Choose *Analyze – Preferences*.

The Analysis Preferences dialog box is displayed as shown in [Figure 8-1](#).

Figure 8-1 Analysis Preferences Dialog Box - Device Models Tab



Using this dialog box you can specify:

- IOCell model defaults
- Interconnect model defaults
- Crosstalk specific defaults
- Simulation mode defaults

- EMI specific defaults
- Default simulation units

DeviceModels Tab

Using the *DeviceModels* tab in the Analysis Preferences dialog box, you can choose whether or not SigNoise will use a default IOCell model when it encounters a driver or receiver pin without an associated IOCell model for six specific pin use types: IN, OUT, BI, TRI, OCL, and OCA. Your Cadence Signal Integrity product is shipped with the following IO cell models:

- CDSDefaultOutput
- CDSDefaultInput
- CDSDefaultIO
- CDSDefaultTristate

Each of the IO cell models listed above is available in four voltages: 5V, 3.3V, 2.5V, and 1.8V. The voltage amount is appended to each model name; for example, the default output IO cell model with 2.5V is CDSDefaultOutput_2p5v.

- CDSDefaultOpenDrain
- CDSDefaultOpenSource

The open drain and open source IO cell models are available only in 5V, therefore no voltage indicator is indicated.

When you choose the `Use Defaults` option for missing component models in the Analysis Preferences dialog box, you are setting up your simulation to use the 2.5V version of the default IO cell models. (These defaults are located in the index file `cds_models.ndx` at `share pcb signal` in your installation directory and accessed by way of the Signal Analysis Library Browser's `Add existing library > Standard Cadence Library` option.)

You do not have to modify design databases created with pre-16.0 versions of Cadence's default IO cell models (CDSDefaultOutput, CDSDefaultInput, CDSDefaultIO, and CDSDefaultTristate), all of which were 5V versions. These models are still supported.

For further details on this tab, refer to the `signal_prefs` command in the *Allegro PCB and Package Physical Layout Command Reference*.

InterconnectModels Tab

From the *InterconnectModels* tab on the Analysis Preferences dialog box, you can establish default values to determine how interconnect is modeled during simulation both before and after routing and how crosstalk and SSN analysis is performed.

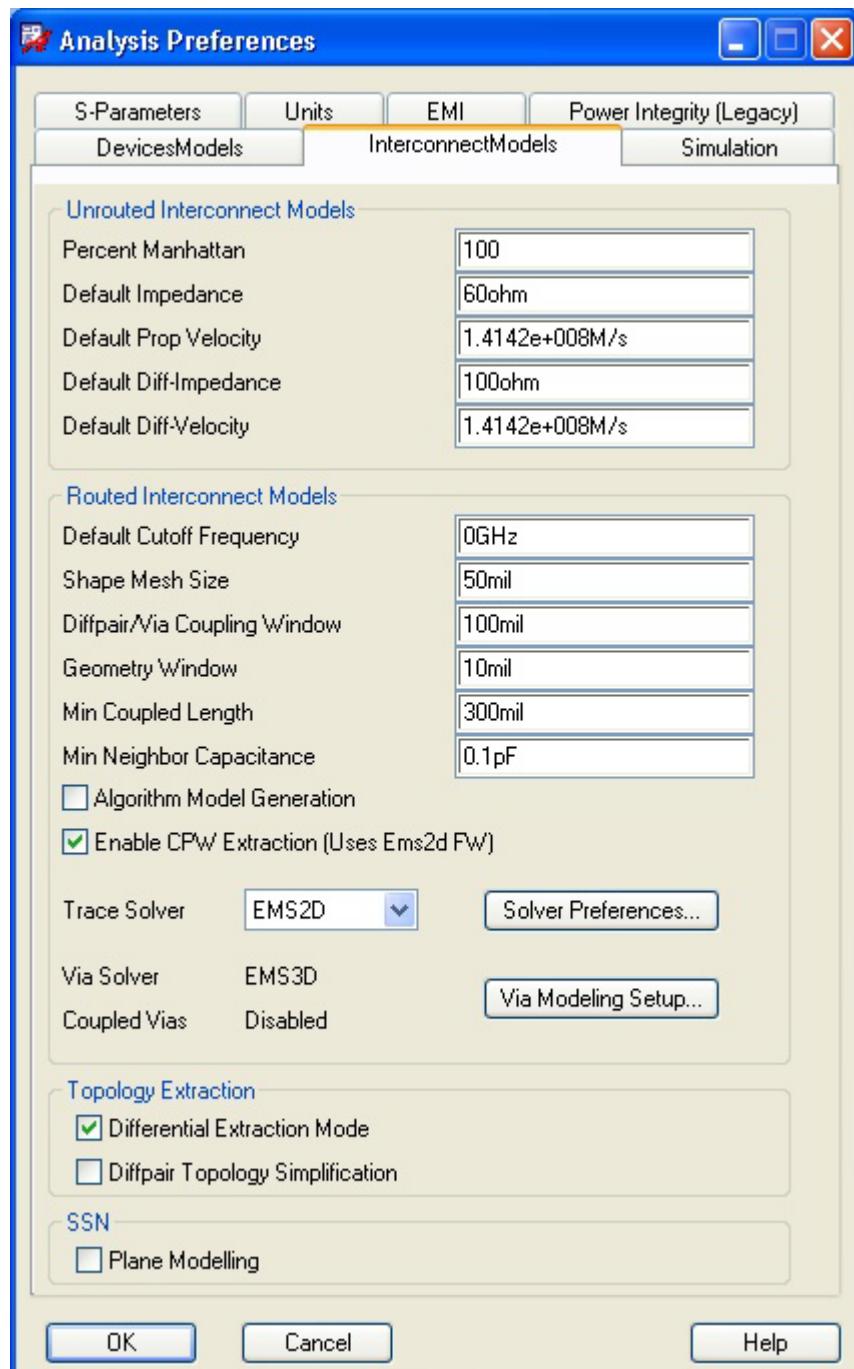


Tip
You can increase simulation performance by limiting the number of .iml files saved during simulation (defaults to 50). Choose *Setup – User Preferences*, and click the *Signal_analysis* folder. Then specify a value for `NUM_NEW_IML_MODELS_BEFORE_SAVE`.

For further details on this tab, refer to the `signal_prefs` command in the *Allegro PCB and Package Physical Layout Command Reference*.

Allegro PCB SI User Guide
Signal Integrity Analysis

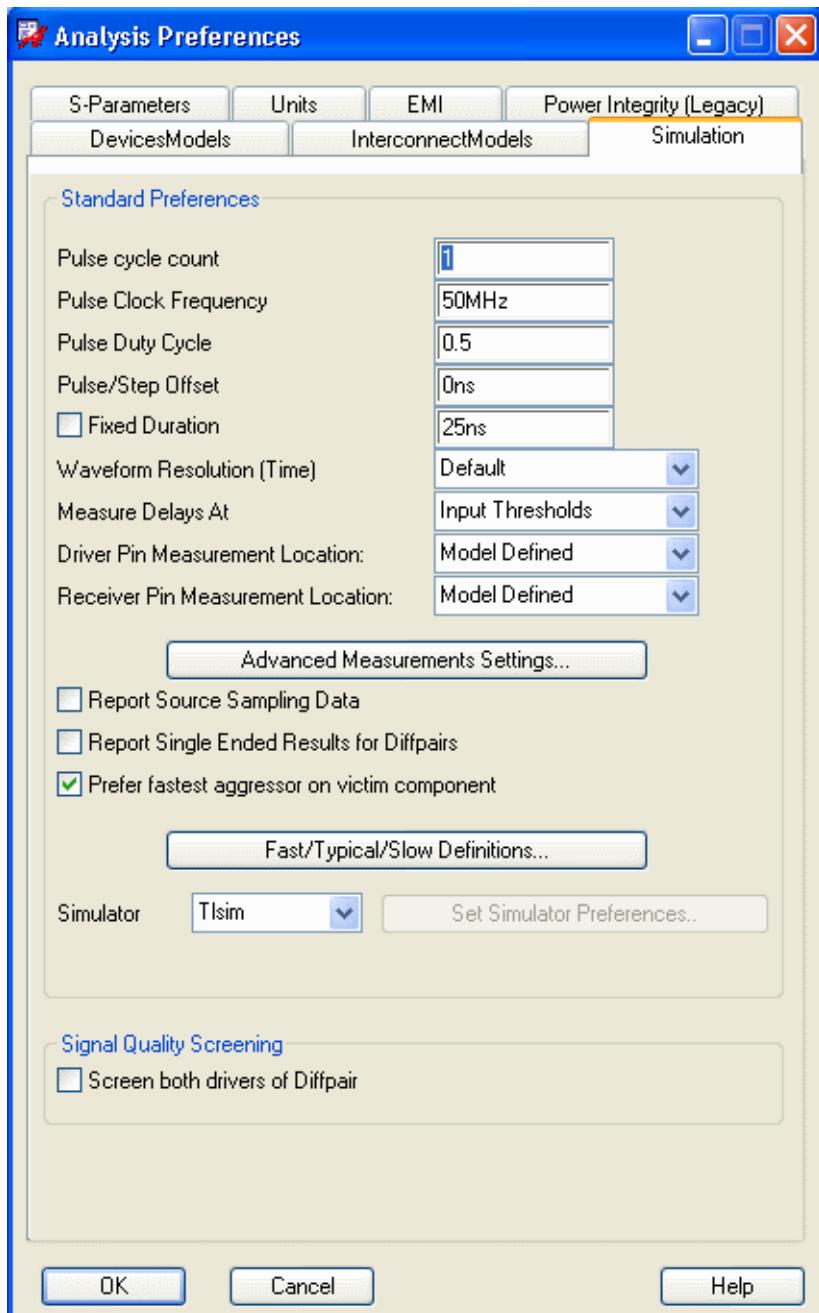
Figure 8-2 Analysis Preferences Dialog Box - Interconnect Models Tab

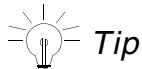


Simulation Tab

From the *Simulation* tab on the Analysis Preferences dialog box, you can determine how simulations are performed by default, and define glitch settings and fast, typical, and slow simulation modes. You can also set driver and receiver pin measurement locations.

Figure 8-3 Analysis Preferences Dialog Box - Simulation Tab



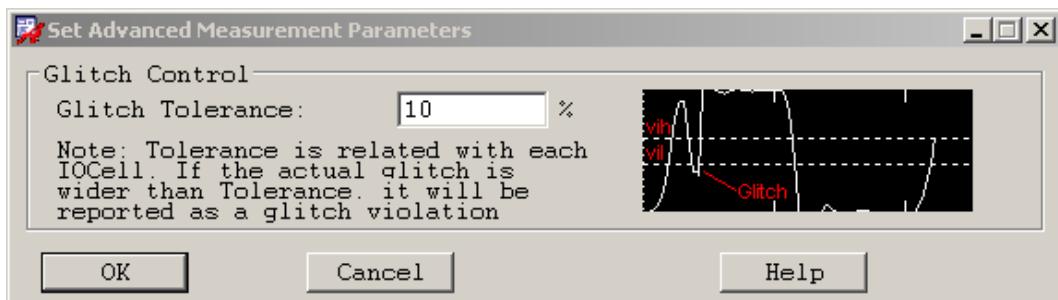


You can increase simulation performance by limiting the number of .iml files saved during simulation (defaults to 50). Choose *Setup – User Preferences*, and click the *Signal_analysis* folder. Then specify a value for `NUM_NEW_IML_MODELS_BEFORE_SAVE`.

Advanced Measurement Settings

Click the *Advanced Measurements Settings* button to display the Set Advanced Measurement Parameters dialog box shown in [Figure 8-4](#).

Figure 8-4 Set Advanced Measurement Parameters Dialog Box



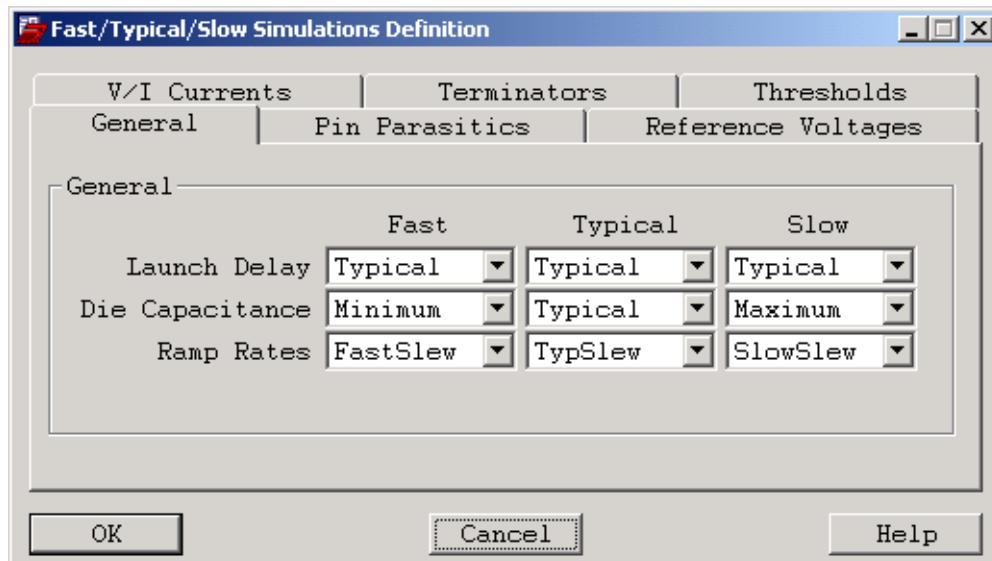
From here you can set measurement parameters that govern glitch controls that can assist you in finding correct cycles in your waveform. The glitch tolerance setting is a relative percentage of the faster of the rising and falling edges of each IO cell buffer model you need to measure. When a glitch occurs between the starting and ending points of a cycle, a glitch violation is reported if the value of the glitch exceeds the tolerance percentage entered in the Glitch Tolerance field. The glitch is *not* reported as a cycle. For information on how glitch settings are established in SigXplorer, see the [SigXplorer Command Reference](#).

Fast/Typical/Slow Definitions

Click the *Fast / Typical / Slow Definitions* button to display the Fast/Typical/Slow dialog box shown in [Figure 8-5](#).

For further details on this tab, refer to the `signal_prefs` command in the *Allegro PCB and Package Physical Layout Command Reference*.

Figure 8-5 Fast/Typical/Slow Dialog Box - General Tab



You can represent device operating conditions by simulating in Fast, Typical, and Slow modes. The device model data is given as minimum, typical, and maximum values. The Fast/Typical/Slow dialog box shown in Figure 8-5 controls the selection of model values for each simulation mode. For example, minimum Die Capacitance usually results in the fastest operating mode.

Each tab on this dialog box lets you define fast, typical, and slow mode for a list of related properties. Properties are listed in a column on the left. Each property is followed by an array of pulldown menus, one each for slow, typical, and fast mode. These choices refer to the minimum, typical, and maximum values given in the IOCell model.

In most cases the menu choices are minimum, typical, and maximum. On the *General* tab, Ramp Rate choices are *FastSlew*, *TypSlew*, and *SlowSlew*. On the *V/I Currents* tab, all the choices are *TempCntl*, *Typ-Z*, *Low-Z*, and *High-Z*.

If the simulation type is Temperature Controlled, the options in the Typical column of the form are used, except for the V/I currents. In this case, the V/I curve used is interpolated between the three given curves based on temperatures for each IOCell and the *VIReferenceTemperature* parameter.

For further details on this dialog box, refer to the [*signal_prefs*](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

S-Parameters Tab

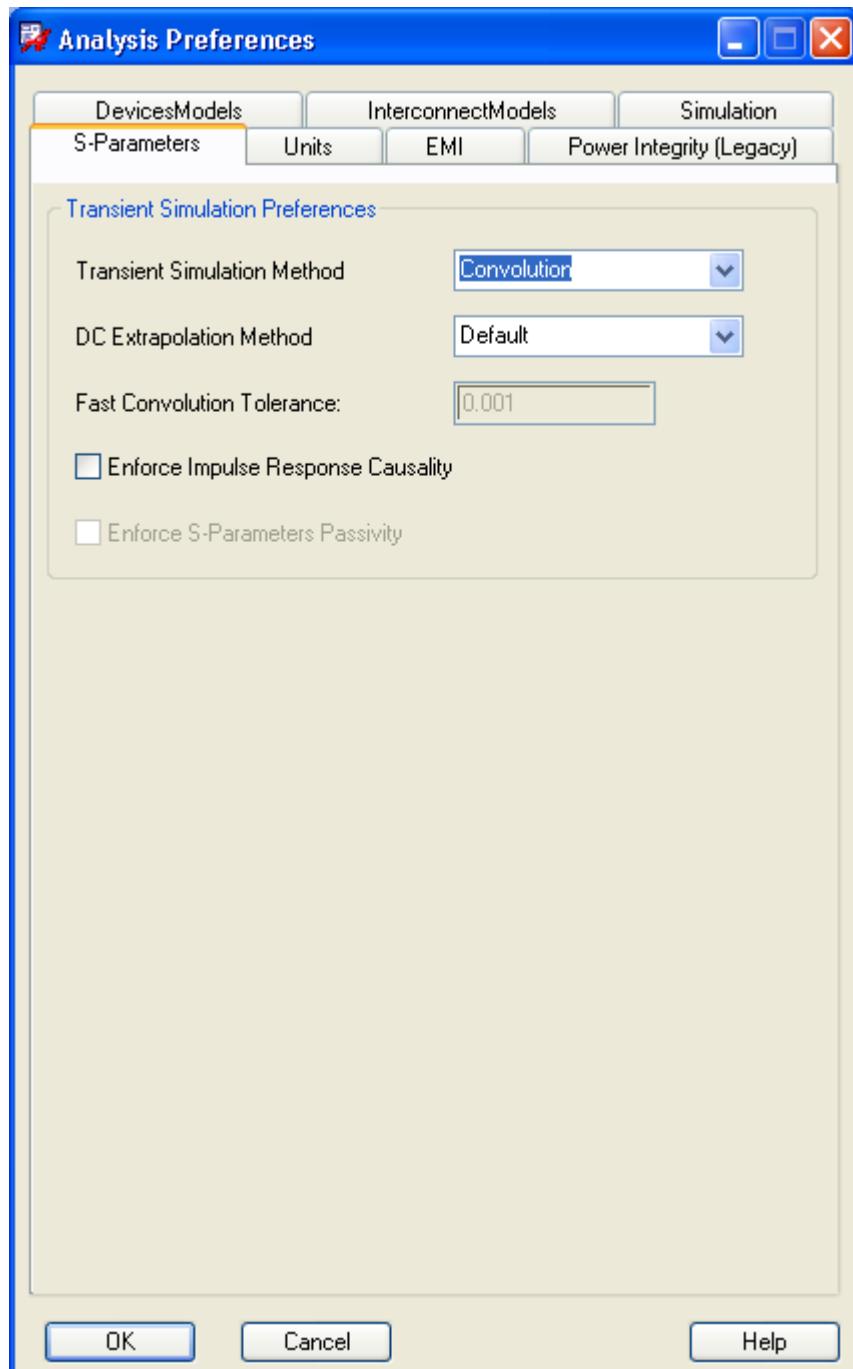
From the *S-Parameters* tab on the Analysis Preferences dialog box, you can

- set various S-Parameter transient simulation options
- perform extrapolation of low frequency points down to the DC level of the S-Parameter, and
- enforce impulse response causality for physical systems

This functionality is available in higher tiers of Allegro PCB SI and in SigXplorer PCB SI. In post-layout designs, the functionality is dependent on ESpice models containing S-Parameters. For details on configuring the controls in this dialog box, refer to the [signal_prefs](#) topic in the *Allegro PCB and Package Physical Layout Command Reference* (for Allegro PCB SI) or the [Analyze – Preferences](#) topic in the *SigXplorer Command Reference*.

Allegro PCB SI User Guide
Signal Integrity Analysis

Figure 8-6 Analysis Preferences Dialog Box - S-Parameters Tab

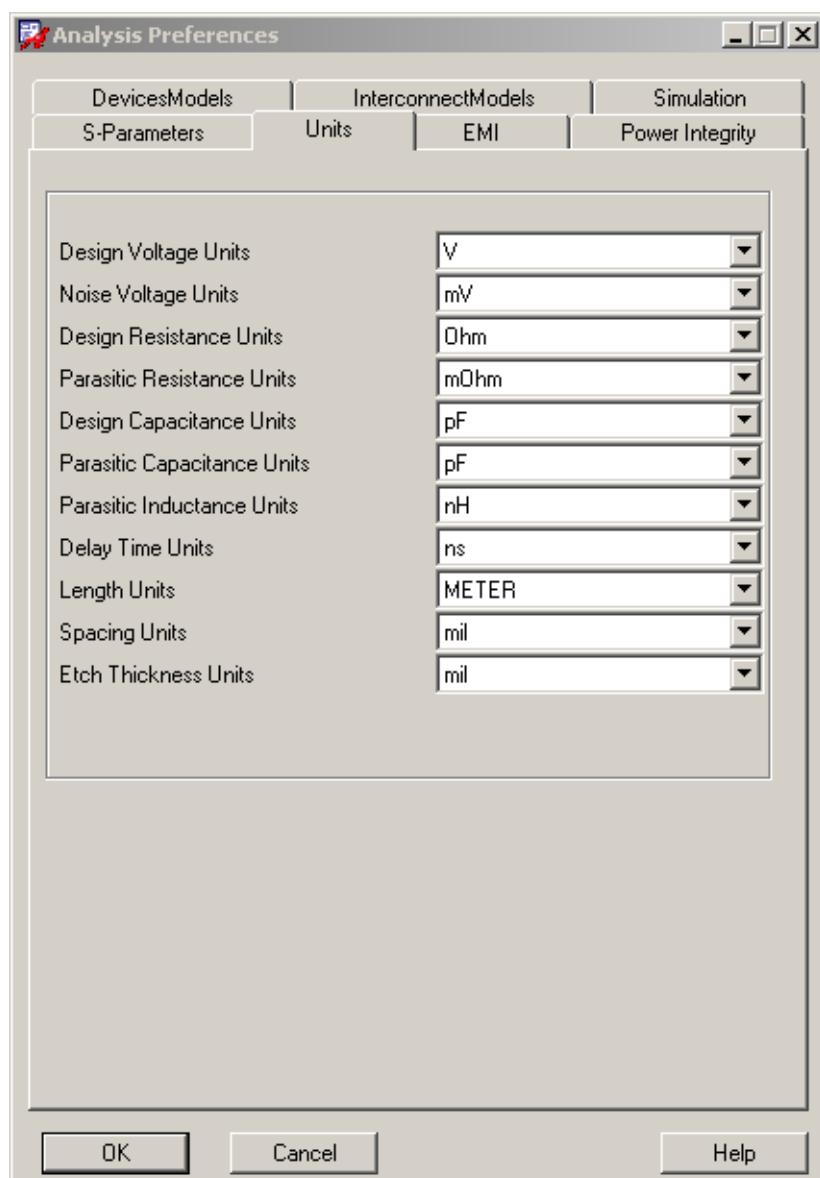


Units Tab

From the *Units* tab on the Analysis Preferences dialog box, you can determine the units in which certain parameters are presented in dialog boxes and reports.

For further details on this tab, refer to the [signal_prefs](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

Figure 8-7 Analysis Preferences Dialog Box - Units Tab



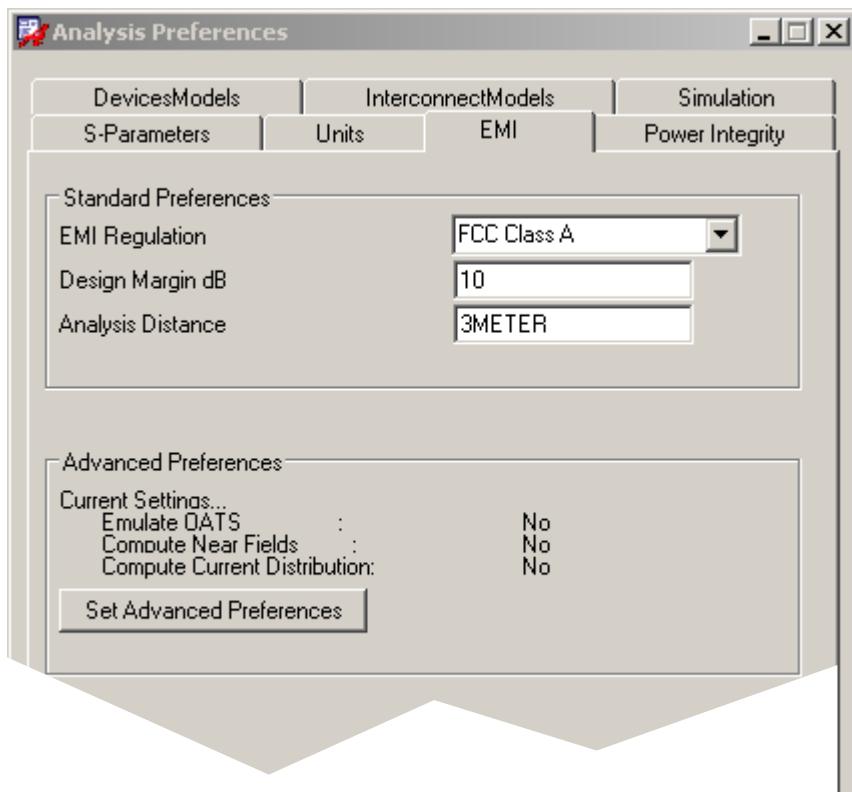
EMI Tab

From the *EMI* tab on the Analysis Preferences dialog box, you can establish basic setup information for EMI single net simulation. Use the *Standard Preferences* to establish an environment appropriate for EMI simulation during design.

Use the information in the *Advanced Preferences* area to view whether advanced EMI simulations are selected and to establish advanced preferences for EMI single net simulation. The advanced EMI preferences specify general control settings for EMI computations, establish an OATS test environment appropriate for evaluation of an experimental setup, and define values for computation of near field EMI effects.

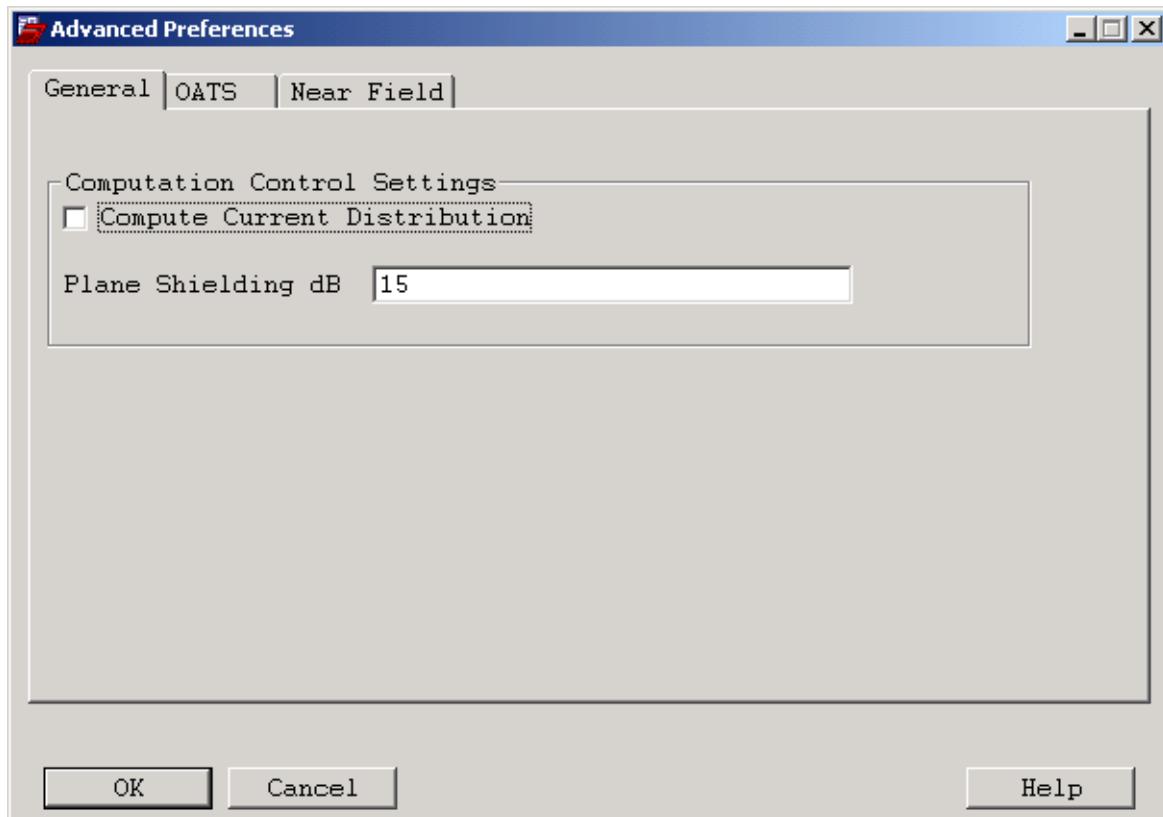
For further details on this tab, refer to the [signal_prefs](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

Figure 8-8 Analysis Preferences Dialog Box - EMI Tab



Allegro PCB SI User Guide
Signal Integrity Analysis

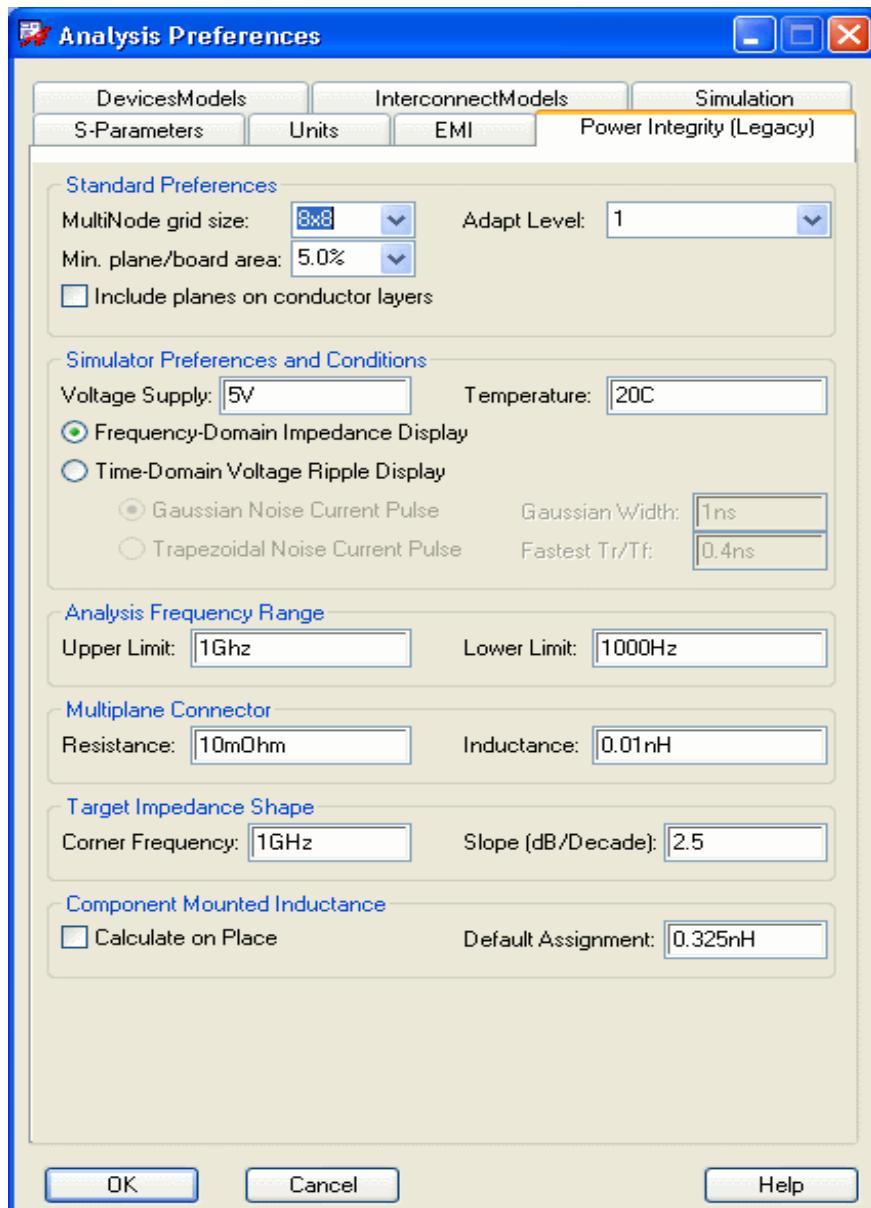
Figure 8-9 Advanced Preferences Dialog Box



Allegro PCB SI User Guide

Signal Integrity Analysis

Figure 8-10 Analysis Preferences Dialog Box - Power Integrity Tab



More on Setting Preferences and Parameters

Unrouted Interconnect Models

For pre-route signal integrity analysis, SigNoise models hypothetical traces using a percent Manhattan value, a default impedance value, and a default propagation velocity.

Routed Interconnect Models

For post-route signal integrity analysis, you can specify a field solver cutoff frequency and the way that vias are modeled. The field solver cutoff frequency establishes a bandwidth within which interconnect parasitics are solved. This prompts the SigNoise field solver to generate frequency-dependent transmission line models in the interconnect library. The default cutoff frequency of 0GHZ directs the field solver to disregard signal frequencies. This saves computation time, but may not be as accurate as frequency-dependent interconnect modeling.

To define how vias are modeled during simulation, first select whether vias are to be ignored or whether each via should have a closed-form model. If you specified that closed-form models are to be used, you can further specify whether SigNoise should save via models in the interconnect library and search the interconnect library for via models.

Crosstalk

For crosstalk analysis, you can specify the size of the area that SigNoise will search for neighbor nets and the minimum mutual capacitance value for a net to be considered a neighbor net. The geometry window specifies the axial distance from the edge of the trace that SigNoise will search for neighbor nets. The minimum coupled length is the minimum distance that two traces must run parallel to each other within the geometry window distance for SigNoise to consider the adjacent trace to be a neighbor net. The minimum mutual capacitance value is the minimum amount of capacitive coupling between traces for SigNoise to look for crosstalk. The capacitance value is read from the RLGC matrix inside the package model.

Traces falling within the geometry window distance of the interconnect, traveling parallel to it for more than the minimum coupled length, and having more than the minimum amount of capacitive coupling will be regarded as neighbor net for the purpose of crosstalk calculations.

Note: This will take the z-axis into account until it hits a plane shield.

Pre-Route Analysis

Pre-route signal integrity analysis comes after preliminary placement and before routing. It is very beneficial to perform this analysis from a time-to-market standpoint. Many signal integrity and timing problems can be quickly identified and corrected before any time and effort is invested in routing the design. It can be increasingly costly and time-consuming to address these issues later on in the design cycle.

Unrouted interconnect is modeled based on your assumptions for percent Manhattan distance, characteristic impedance, and propagation velocity. You can quickly simulate the entire layout and compare it against the electrical constraints to identify the signals that are marginal or failing. This determination should include signals that span multiple layouts. An example of this is a signal running from a connector on one board through a cable to a connector on another board. Rapid simulation is a key time saver when a layout has thousands of nets. This allows you to focus attention on problem nets first and avoid wasting time on signals that are initially within constraints.

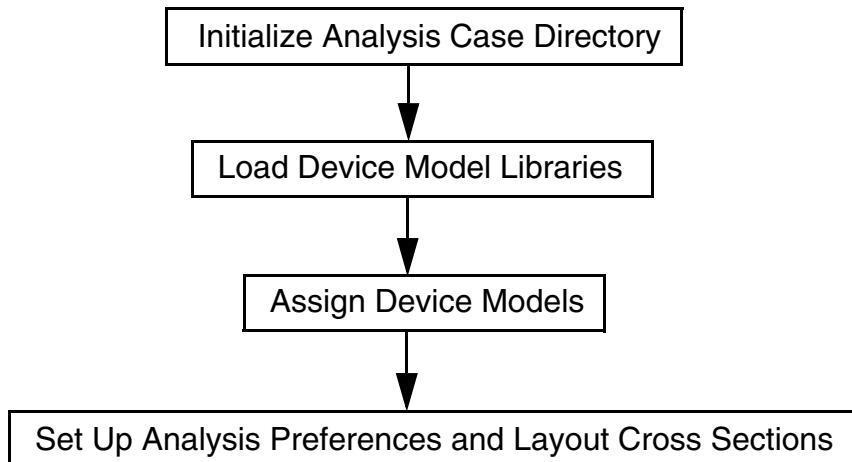
In pre-route signal integrity analysis you look for the following.

- How placement affects critical delays and reflections in the design.
- How net scheduling affects delays and reflections.
- The need for terminators on nets in the design.
- An early evaluation of the power distribution system.

Pre-route Analysis Setup

The following figure and instructions describe the procedure for setting up SigNoise for pre-route signal integrity analysis.

Figure 8-11 Pre-Route Setup Flow Diagram



To set up for pre-route analysis

1. Optionally, initialize an analysis directory to tell SigNoise where to write signal analysis data files.

After placement, SigNoise can provide you with delay and distortion data that comes from hypothetical traces. SigNoise develops these hypothetical traces based on a percent Manhattan distance between pins and user-defined assumptions for the characteristic impedance and propagation velocity. This information is specified on the *InterconnectModels* tab in the Analysis Preferences dialog box.

2. Load device model libraries.
3. Assign device models from these libraries to components in the design.
4. Set simulation preferences and set up the layout cross section.

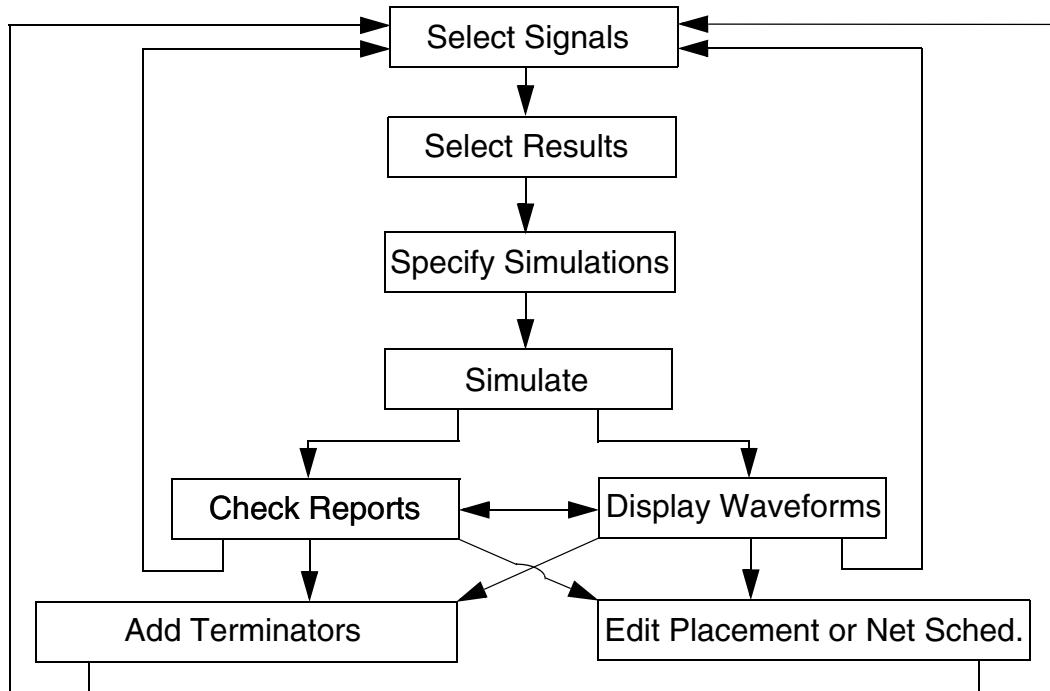
The preferences specify, for example, the default IOCell models and the units of measurements for reports. When you set up the cross section, you define, for example, how the layers stack up and what materials and thicknesses you use for these layers.

Performing Pre-route Signal Integrity Analysis

After you set up your device models and device model libraries and make IOCell model assignments, you can perform simulations and generate analysis data.

The following figure and instructions describe the procedure for performing pre-route signal integrity analysis.

Figure 8-12 Pre-Route Analysis Flow Diagram



Procedure:

1. Select signals for simulation by:

clicking to select a ratsnest line or a pin in the design window.

—or—

specifying a net by name in the Signal Analysis dialog box.

—or—

specifying a netlist file by name in the Signal Analysis dialog box or selecting the nets through the Net Browser dialog box.

2. Select the type of analysis results to create.

Click *Reports* in the Signal Analysis dialog box to present the analysis results as text reports. This opens the Report Generator.

—or—

Click *Waveforms* in the Signal Analysis dialog box to present the analysis results as waveform files.

The Analysis Waveform Generator dialog box opens.

3. Specify the type of simulation you want SigNoise to run.

Select the appropriate options in the Analysis Report Generator or Analysis Waveform Generator dialog box.

Note: To use the Power Plane Designer to analyze your power and ground plane design, you must enable the *Plane Modeling* option on the *Interconnect Models* tab of the Analysis Preferences dialog box. You then perform an SSN simulation.

4. Trigger the simulation.

Click *Create Report* or *Create Waveforms*.

SigNoise performs the requested simulations based on your specifications.

5. Following simulation, you can look at the delay and distortion data in text reports or view time domain waveform displays at driver and receiver pins.

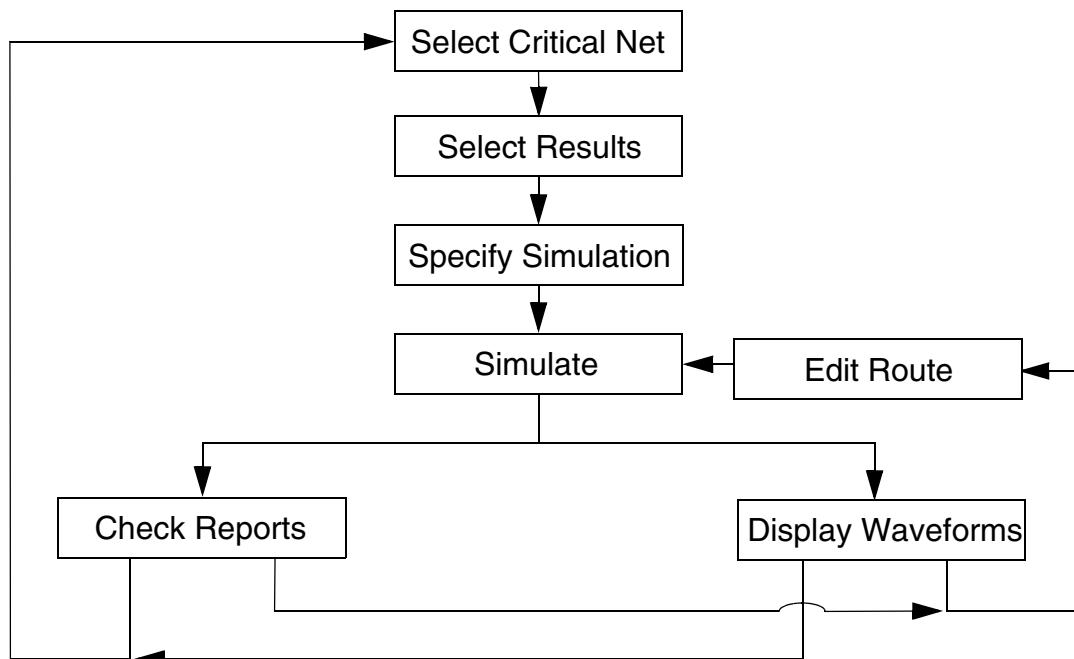
The simulation results may lead you to edit the placement of components, modify net schedules, or to experiment with terminators to suppress distortion.

Note: Crosstalk data is not applicable when running pre-route analysis and is listed as either N/A or 0 in text reports. For details on running meaningful post-route crosstalk simulations, see [Crosstalk Analysis](#) on page 264.

Critical Net Analysis

After pre-route analysis you might want to interactively route critical nets and then analyze them for signal integrity. The following figure and instructions describe the procedure for performing critical net analysis during routing.

Figure 8-13 Critical Net Analysis Flow Diagram



During pre-route analysis, SigNoise built a simulation circuit model. It used the device models that you specified and the hypothetical interconnect models that it approximated from the percent Manhattan distance, the default impedance, and the default propagation velocity that you specified. Now that the critical nets are routed, you can analyze them more precisely, this time using the actual etch instead of the Manhattan-based estimates.

Procedure:

1. You can begin critical net analysis with interconnect library setup to specify where you want SigNoise to save the interconnect models it creates. You might also create a Parasitics report for a critical net.
2. You can also scan the design for problem areas using the same steps you followed in pre-route analysis.

3. You then select a net for simulation and look at the results as waveform displays and text reports.
4. After you examine your results you can edit the routing for that critical net and perform another analysis. The process of analyzing and editing the traces is an iterative process that you can continue until you see satisfactory simulation results.

Post-Route Verification

During post-route verification, you generate your final simulations and create reports using PCB SI. These reports enable you to verify and confirm that your design is performing as originally intended.

Rather than being the primary vehicle for identifying SI issues, post-route verification is intended to serve as a signal integrity *sign-off*. Due to constraint-driven design, problems uncovered during this design phase tend to be isolated and correctable. You simply extract the problem nets individually into SigXplorer, analyze them in-depth, then make the necessary adjustments to the design.

You use the SigNoise simulator to perform post-route analysis for reflection, crosstalk, and SSN (simultaneous switching noise). SigNoise is the simulation engine used by PCB SI. You can also perform all of these analysis across multiple printed circuit boards using a special library model called a DesignLink. As you perform these simulations, you save the waveforms in the current simulation directory along with any reports that you create. This lets you organize your results for archival and future reference.

During post-route signal integrity analysis, you look for:

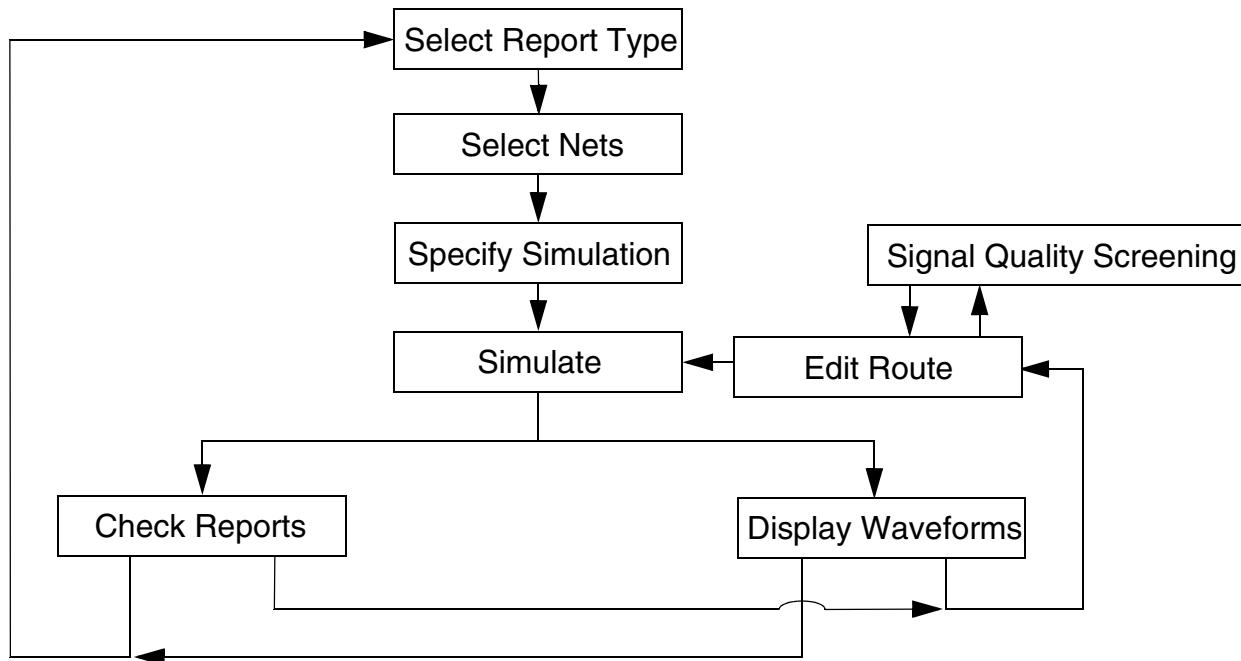
- the effect of the routed interconnect on signal integrity and EMI.
- the effect that the routed interconnects have on each other.

That is, you can now see the effect of couplings between interconnect segments and how they create crosstalk and affect signal integrity and EMI.

- the effect of neighboring interconnects that you have added since critical nets were routed.

The following figure and instructions describe a typical procedure for post-route analysis.

Figure 8-14 Post-Route Verification Flow Diagram



Procedure:

1. Begin with the parasitic analysis.
2. After parasitic analysis you can scan the design for problem areas or proceed to detailed analysis of individual nets.

You can run single or multi-line simulations depending on whether you want to take neighboring nets into account.

3. After signal integrity simulation you can perform one or more of the following tasks.
 - ❑ Run source synchronous reflection and comprehensive bus analysis for all Xnets of a selected bus and their strobe/clock Xnets. See [Source Synchronous Bus Analysis](#) on page 267 for details.
 - ❑ Look at the Delay, Ringing, Crosstalk, SSN, and EMI Single Net reports. See [“Analyzing to Generate Text Reports”](#) on page 284 for details.
 - ❑ Use the Conductor Cross Section window (sigxsect) to look at geometric displays of the models SigNoise writes for interconnect segments. See [“Conductor Cross Sections”](#) on page 350 for further details.
 - ❑ Use [Signal Quality Screening](#) to determine signal quality of a system and perform focused analysis resulting in improved designs in a shorter time.

- Use EMControl to analyze the design for EMI performance.

[EMControl User Guide](#)

Note: Because of the high volume of simulations often performed for post-route analysis, you have the option to run post-route analysis in batch mode rather than from the UI. See “[Batch Simulation](#)” on page 264 for further details.

Interactive Simulation

Using SigNoise interactively, you can quickly examine or scan one or more signals by performing Reflection simulations and Crosstalk estimations on the entire design or on large groups of signals. You can also probe individual signals, or small groups of signals, where you want to delve into specific signal behaviors in detail through the generation of discrete text reports or waveforms.

Text Reports

There are several pre-formatted text reports available to choose from or you can generate your own custom reports based on specific criteria.

Waveforms

SigWave displays waveform data for all pins in a simulation circuit. The waveform data shows the waveform of a signal on a driver-receiver pair with both the package pin and the internal die location (denoted by the suffix i after the pin number) being displayed. This allows you to view the effects of the package parasitics. If the parts on the SigXplorer canvas do not have package parasitics (indicated by a box surrounding the element), then only the waveforms at the pins are displayed.

Conductor Cross Sections

SigNoise generates models for the interconnect in your design. The SigNoise field solvers generate the parasitic values in the model. The Conductor Cross Section window shows you a three-dimensional view of the interconnect and its parasitic values.

If two interconnect segments are within the distance specified in the geometry window parameter and if you are running multi-line simulations, SigNoise writes a model that includes both interconnect segments. You can see both segments in the Conductor Cross Section Window. You can also display equipotential field lines between interconnects in the Conductor

Cross Section window. Slide interconnect segments to see how they change both the field lines and the RLGC matrix of the model.

Simulation Process

SigNoise can locate problem areas in your design. Use the following steps to diagnose and resolve signal integrity problems:

1. First quickly examine, or scan, large groups of nets, or the entire design, for problem areas.
2. Based on the waveforms and text reports resulting from these initial analysis, analyze small groups of signals, or extract and analyze specific individual signals using SigXplorer in order to troubleshoot signal integrity or EMI issues.

Color Highlighting During Analysis

SigNoise will highlight pins and connect line segments on nets when you analyze a design. When SigNoise analyzes the pin-to-pin connections in a design, it highlights the objects shown in the following table.

Table 8-1 Object Highlight Colors

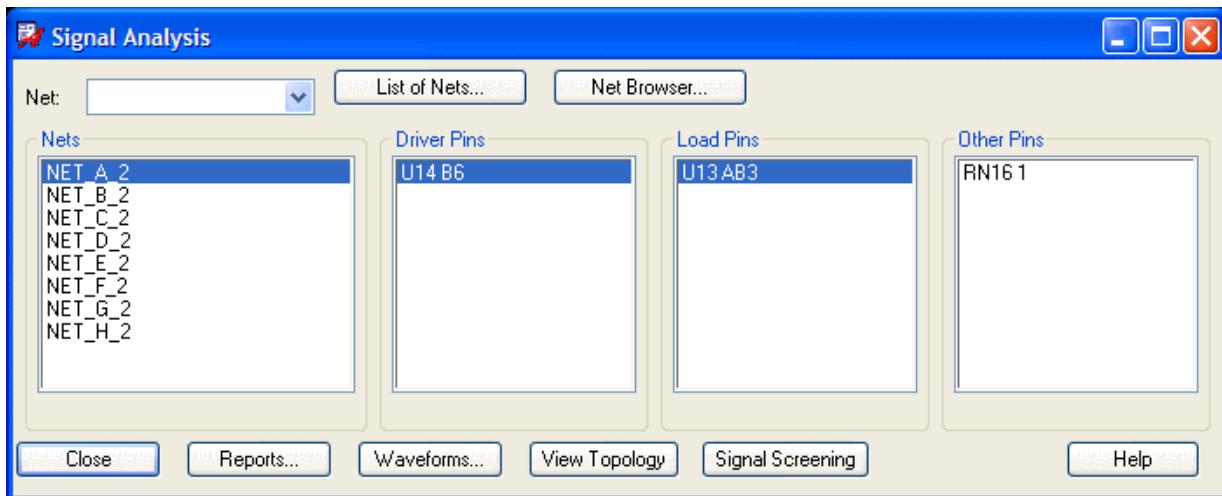
Object	Assigned Highlight Color
The connect line segment that contributes the most crosstalk to the pin-to-pin connection.	Temp highlight
The driver and receiver pins, the connect line between them, and all components between these pins such as a series resistor.	Perm highlight

To begin interactive signal integrity and EMI analysis within PCB SI

- Choose *Analyze – Probe*.

The Signal Analysis dialog box displays as shown in the following figure.

Figure 8-15 Signal Analysis dialog box



Use the Signal Analysis dialog box as the starting point for performing signal integrity and EMI emissions simulations. The Signal Analysis dialog box enables you to select nets and driver-receiver combinations for analysis.

You can also display the Signal Analysis Waveform and Report Generator dialog boxes from the Signal Analysis dialog box. In these dialog boxes, you specify which waveforms or reports to generate. SigNoise performs the necessary simulations accordingly.

You can also run the Signal Quality Screening process from this dialog box.

The *SigXplorer* topology editor and the *sigxsect* interconnect cross-section viewer are also launched from the Signal Analysis dialog box. Use *SigXplorer* to perform what-if studies on different driver and receiver combinations and transmission line scenarios. Use *sigXsect* to display cross-sections of routed interconnect segments.

For details on specific options and buttons in the Signal Analysis dialog box or for procedures regarding interactive analysis, refer to the signal_probe command in the *Allegro PCB and Package Physical Layout Command Reference*.

Selecting Nets and Pins for Simulation

In the Signal Analysis dialog box, you can select nets for analysis in several different ways:

- Click on single ratsnest lines, routed etch, etch, or pins in the design window.
 - or -

Drag a window around groups of ratsnest lines, routed etch, etch, or pins in the design window.

- Click *Net Browser* in the Signal Analysis dialog box to select groups of nets by browsing for netlist files.

- or -

Create a netlist file with a text editor.

- At the PCB SI or PCB Editor command line, type:

```
net <name>
```

Upon selection of a net or a pin pair, the names of the nets and driver, and receiver pins appear in the *Nets*, *Driver Pins*, and *Load Pins* list boxes in the Signal Analysis dialog box. Also, the PCB Editor message line or the PCB SI message log window display messages that tell you SigNoise is gathering extended net information for the nets that you have selected.

Batch Simulation

In addition to performing signal integrity analysis interactively from the UI, you can also use SigNoise in batch mode. See the information on *Batch Generation* in each of the text report sections within [“Analyzing to Generate Text Reports”](#) on page 284 for more information.

Crosstalk Analysis

You can choose between two modes of crosstalk analysis: estimated and simulated.

- Estimated crosstalk lets you quickly scan your design to identify problem areas for further, detailed, crosstalk simulations. Estimated crosstalk constructs a table of crosstalk data based on a series of crosstalk simulations performed on the specified traces at various trace spacings.
- Detailed crosstalk analysis uses multiline simulations for more detailed and accurate analysis.

Both crosstalk estimation and detailed crosstalk simulation can be timing-driven. Performing timing-driven crosstalk analysis using crosstalk timing windows greatly increases real-world accuracy.

Timing-Driven Crosstalk Analysis

SigNoise lets you perform timing-driven crosstalk analysis using crosstalk timing windows. Timing-driven crosstalk analysis can both minimize crosstalk false alarms and reduce the overall pessimism of crosstalk results, thus helping you to increase the density of your designs.

Crosstalk timing windows use crosstalk timing properties to determine when nets are active and sensitive. Only aggressor nets that have an active time overlap with the victim nets are sensitive.

The following crosstalk timing properties can be applied to nets:

- XTalk_ACTIVE_TIME
- XTalk_SENSITIVE_TIME
- XTalk_IGNORE_NETS

You can assign the XTalk_ACTIVE_TIME property to a net to specify the times during which that net can generate crosstalk on a neighbor net. If a net has no attached XTalk_ACTIVE_TIME property, SigNoise assumes that the net can generate crosstalk at all times.

You can assign the XTalk_SENSITIVE_TIME property to specific nets for even greater accuracy to indicate times when that net is susceptible to crosstalk and when it is not.

You can use the XTalk_IGNORE_NETS property to tell a net or a net group to disregard other nets or net groups as a source of crosstalk. For example, use this property when you want to disregard crosstalk between bits on a synchronous bus.

A Simple Example

In Crosstalk simulations, the XTalk_ACTIVE_TIME, XTalk_SENSITIVE_TIME, and XTalk_IGNORE_NETS crosstalk properties can be used to determine how to stimulate multi-line circuits for crosstalk analysis.

For example, assume a victim net being analyzed for crosstalk had 2 aggressor nets, and the following properties.

- victim net - XTalk_SENSITIVE_TIME = 5-10
- neighbor #1 - XTalk_ACTIVE_TIME = 7 - 15
- neighbor #2 - XTalk_ACTIVE_TIME = 20-25

Neighbor #2 is not stimulated in the circuit since its active time does not overlap with the victim net's sensitive time. In this case, stimulating both aggressor nets together would be overly pessimistic and not indicative of real-world behavior.

See [Appendix C, “Working with Crosstalk”](#) for further details.

EMI Analysis

Simulation

SigNoise provides EMI single net simulations which allow you to compute differential mode radiated electric field emissions from traces. Simulation results include a graphical display of the emission spectrum and a text report summarizing emission details and compliance results.

Using EMControl with SigNoise

You can use SigNoise in conjunction with EMControl to perform EMI analysis. Some of the signal routing and signal quality rules provided with EMControl employ SigNoise simulations and SigNoise device models during analysis for EMI. Using EMControl enables design engineers to begin evaluating their designs for EMI early in the design process with increased accuracy throughout design development.

[*EMControl User Guide*](#)

Multi-Board Analysis

SigNoise lets you perform multi-board (or system level) simulation for a design that is made up of more than one printed circuit board (PCB). When a net extends to more than one PCB, SigNoise can analyze and report the behavior of a signal as it propagates from a driver on one PCB to a receiver on another.

Nets that span multiple PCBs are analyzed using a multi-board system configuration. A multi-board system configuration contains a pin map to hook up connector pins on one PCB to connector pins on another. When a circuit is built for a system extended net (SXnet) that spans multiple PCBs, SigNoise traces out the interconnect to the connector pin, then finds the system connection in the device library and jumps to the next PCB to continue tracing out the circuit. A system configuration can contain a model to represent the mated connector or cable that physically connects the two PCBs. One circuit, spanning the multiple PCBs, is generated for the entire Xnet, allowing full system-level simulations to be done.

See [Appendix B, “System-Level Analysis”](#) for further details.

Source Synchronous Bus Analysis

PCB SI supports post-layout source synchronous reflection and comprehensive bus analysis for all Xnets of a selected bus, as well as their strobe and/or clock Xnets. This functionality lets you:

- assign Xnets to multiple buses that are used only for bus simulations
- set different on-die termination (ODT) settings for different dual in-line memory modules (DIMMs)
- create derating values for all input signals to your calculated setup and hold times

This functionality complements pre-layout source synchronous bus analysis, as described in [*“Working with Source Synchronous Custom Measurements”*](#) in the *Allegro SI SigXplorer User Guide*.

Strobe/Clock Nets in Source Synchronous Designs

Current high-speed designs often incorporate source synchronous bus interfaces instead of common clock buses (though a single design can incorporate both types). In source synchronous schemes, the driving chip sends both the clock (strobe) signal and the data signal to the receiver chip, rather than having both chips share a common external clock. This is illustrated in Figures [8-16](#) and [8-17](#).

Figure 8-16 Common Clock Bus Interface

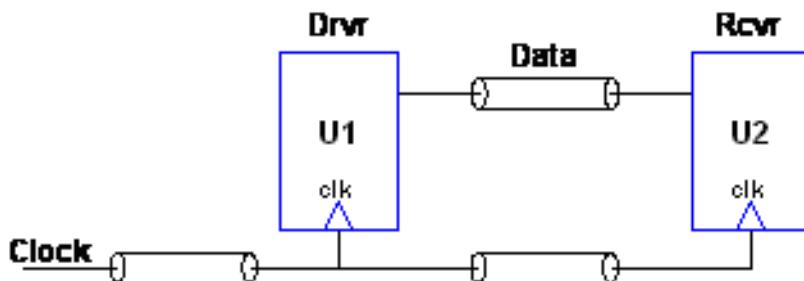
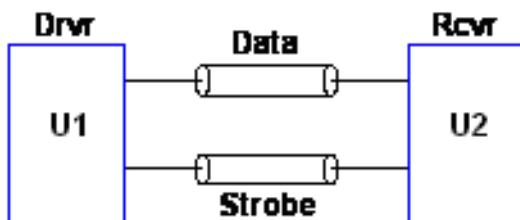


Figure 8-17 Source Synchronous Bus Interface



Because source synchronous bus performance depends on relative delays, which tend to be much smaller than the absolute delays associated with common clock buses, a much higher performance can be achieved using source synchronous buses. This enhanced performance becomes critical to high-bandwidth memory systems using upwards of 800MHz, because common clock buses can typically accommodate only 150 MHz.

Note: Simulations that you perform in the context of a source synchronous flow do not support Odd/Even/Static aggressor modes. Rather, custom stimuli that you have assigned to the victim and aggressor Xnets are referenced.

Support for Address Bus Topology

Address topologies are as important to timing simulation as data topologies. The bus analysis setup helps you associate a strobe or clock net with each bit of the bus being simulated. As the address Xnet connects to all the devices (DRAMs) in the memory banks (DIMMs), multiple clocks need to be associated with each address signal. The source synchronous bus analysis functionality supports assignment of multiple clocks to one address signal. When a specific clock/strobe is selected, the clock Xnets assigned to another address signal are available in the *Unassigned Bus Xnets* list. As a result, these clocks can be assigned to multiple signals.

Note: The bus analysis solution is applicable to all source synchronous buses and not limited to memory interfaces.

Derating Tables for Input Signals

To maximize the reliability of your simulations, you can create derating tables to establish maximum values for your calculated input setup and input hold times for all input signals. You can also create separate derating tables for address signals. Your derating values will depend on the respective signals' nominal or tangential slew rate, as well as the slew rates of your clocks/strobes.

You must create your derating tables in a format recognizable by your Cadence tool. This is a text-based CSV (comma-separated values) format, a sample of which is shown in [Figure 8-18](#).

Earlier in the bus simulation model, both clock and data slew values were required in descending order in the derating tables. The majority of the memory vendors publish this data with the values ascending on both axes. As a result you needed to reorder the data about both the axes before arranging it in the derating table. This time-consuming and error-prone practice to specify values in the derating table is discontinued from the current release.

Release 16.5 onwards, SI provides support for sorting of data in the derating tables in both ascending and descending orders. The sorting order is decided automatically based on the content of the derating file.

File Format of a Derating Table

A derating file must be created as a plain text file with no special formatting characters and must adhere to a pre-defined format. If the file is not formatted correctly, the simulation continues without warning and ignores the derating capabilities. Similarly, no checking or reporting occurs if the file is not found. To be accessible via the browser, the name must end with a .dat file extension. The format must be strictly adhered to but may contain comment lines and blank lines for clarity. All comment lines must contain the "#" character as the first non-white space character. Blank lines and comments can exist throughout the file for better readability.

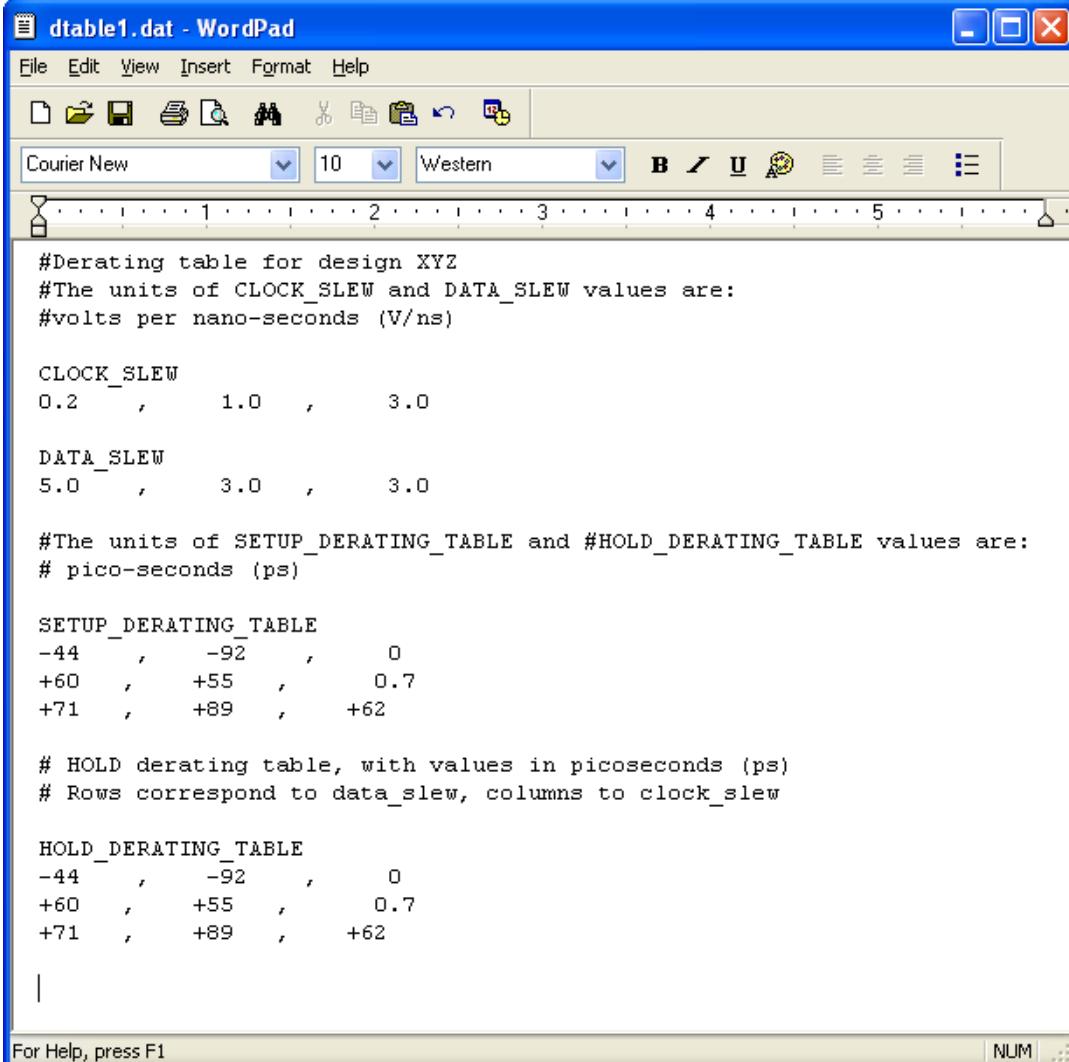
When specifying numbers, leading "+" signs are optional for positive numbers. Additionally a leading "0" may be omitted for decimal values greater than -1 but less than 1.

Example:

+7 is equivalent to 7

0.7 is equivalent to .7

Figure 8-18 Sample Derating File



The screenshot shows a Microsoft WordPad window with the title bar 'dtable1.dat - WordPad'. The menu bar includes 'File', 'Edit', 'View', 'Insert', 'Format', and 'Help'. The toolbar contains icons for file operations like Open, Save, Print, and Find. The font is set to 'Courier New' at size 10, and the encoding is 'Western'. The text content is a sample derating file:

```
#Derating table for design XYZ
#The units of CLOCK_SLEW and DATA_SLEW values are:
#volts per nano-seconds (V/ns)

CLOCK_SLEW
0.2 , 1.0 , 3.0

DATA_SLEW
5.0 , 3.0 , 3.0

#The units of SETUP_DERATING_TABLE and #HOLD_DERATING_TABLE values are:
# pico-seconds (ps)

SETUP_DERATING_TABLE
-44 , -92 , 0
+60 , +55 , 0.7
+71 , +89 , +62

# HOLD derating table, with values in picoseconds (ps)
# Rows correspond to data_slew, columns to clock_slew

HOLD_DERATING_TABLE
-44 , -92 , 0
+60 , +55 , 0.7
+71 , +89 , +62
```

Note: You can find a sample derating file, `derating_table_file.dat`, at the following location:

`<your_install_dir>\share\pcb\examples`

A derating file consists of the following four sections: `CLOCK_SLEW`, `DATA_SLEW`, `SETUP_DERATING_TABLE`, and `HOLD_DERATING_TABLE`. Each section must exist in every derating file and must be presented in the order shown in the sample file.

Descriptions of the various elements of the derating table are:

CLOCK_SLEW

The first non-comment, non-blank line must contain the Data Header `CLOCK_SLEW` followed by a line representing the delineated values in the vendor supplied derating table. `CLOCK_SLEW` is clock or strobe slew rates in Volts per Nanosecond (V/ns). The `CLOCK_SLEW` values represent the column headings in the setup and hold tables. The values are comma separated and white space tolerant.

The order of the setup and hold derating table columns is the same as the order of the `CLOCK_SLEW` data.

Example

```
CLOCK_SLEW      0.4,    0.5,    0.6,    0.7,    0.8,    0.9,  
1.0,    1.5,    2.0
```

DATA_SLEW

The next non-comment, non-blank line must contain the Data Header `DATA_SLEW` followed by a line representing the delineated values in the vendor supplied derating table. `DATA_SLEW` is address or data slew rates in V/ns. The `DATA_SLEW` values represent the row headings in the setup and hold tables. The values are comma separated and white space tolerant.

The order of the setup and hold derating table ROWS is the same as the order of the `CLOCK_SLEW` data.

Example

```
DATA_SLEW      0.4 , 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.5, 2.0
```

Note: The numerical values of clock/strobe slew and data slew rates must be in the same line with the data headers `CLOCK_SLEW` and `DATA_SLEW`, respectively. Else, the output report file will report N/A for all derating values.

SETUP_DERATING_TABLE The next non-comment, non-blank line begins the Setup Data Matrix. This entry must contain the keyword SETUP_DERATING_TABLE. Each line contains a sequence of numeric values representing the derating numbers as defined in the vendor table for setup.

The number of entries in each row must be equal to the number of entries in the CLOCK_SLEW. Similarly, the number of rows must be equal to the number entries in the DATA_SLEW list.

The values are comma separated and white space tolerant and are expressed in Picoseconds (ps). For each “0” value defined in the vendor table, a “0” must be specified in the matrix. A NULL value or white spaces are not acceptable. The matrix must be fully populated.

Example:

SETUP_DERATING_TABLE

-33,	-43,	-53,	-63,	-78,	-78,	-78,	-78,	-78
17,	7,	-3,	-13,	-28,	-28,	-28,	-28,	-28
38,	28,	18,	8,	-7,	-7,	-7,	-7,	-7
50,	40,	30,	20,	5,	5,	5,	5,	5
61,	52,	42,	32,	17,	17,	17,	17,	17
70,	60,	50,	40,	25,	25,	25,	25,	25
75,	65,	55,	45,	30,	30,	30,	30,	30
142,	132,	122,	112,	97,	97,	97,	97,	97
175,	165,	155,	145,	130,	130,	130,	130,	130

HOLD_DERATING_TABLE The next non-comment, non-blank line begins the Hold Data Matrix. This entry must contain the keyword HOLD_DERATING_TABLE. Each line contains a sequence of numeric values representing the derating numbers as defined in the vendor table for hold.

The number of entries in each row must be equal to the number of entries in the CLOCK_SLEW list. Similarly, the number of rows must be equal to the number entries in the DATA_SLEW list.

The values are comma separated and white space tolerant and are expressed in Picoseconds (ps). For each “0” value defined in the vendor table, a “0” must be specified in the matrix. A NULL value or white spaces are not acceptable. The matrix must be fully populated.

Example:

HOLD_DERATING_TABLE

```
-213, -210, -206, -203, -198, -198, -198, -198, -198  
-150, -147, -143, -140, -135, -135, -135, -135, -135  
-108, -105, -102, -98, -93, -93, -93, -93, -93  
-79, -75, -72, -69, -64, -64, -64, -64, -64  
-56, -53, -49, -46, -41, -41, -41, -41, -41  
-39, -36, -32, -29, -24, -24, -24, -24, -24  
-25, -22, -18, -15, -10, -10, -10, -10, -10  
17, 20, 24, 27, 32, 32, 32, 32, 32  
38, 41, 45, 48, 53, 53, 53, 53, 53
```

SETUP DERATING TABLE Δt_s is a *Setup Derating Values* matrix of size $M \times N$ where M is the number of rows corresponding to the DATA_SLEW list values and N is the number of columns corresponding to the CLOCK_SLEW list values.

Hold Derating Table Δt_H is a *Hold Derating Values* matrix of size $M \times N$ where M is the number of rows corresponding to the DATA_SLEW list values and N is the number of columns corresponding to the CLOCK_SLEW list values.

To complement the pre-layout source synchronous bus analysis functionality in SigXplorer, PCB SI offers post-layout GUI-based analysis tools.

Allegro PCB SI User Guide

Signal Integrity Analysis

- **Analyze – Bus Setup** lets you identify the source synchronous buses in your layout, create buses for simulation purposes only, and provide data required for you to perform the analysis. You enter this data by way of the Signal Bus Setup and the Stimulus Setup dialog boxes.
- **Analyze – Bus Simulate** lets you perform the actual simulation of the source synchronous bus

You perform each step in this setup-and-simulate flow using the dialog boxes illustrated in Figures 8-19 and 8-20.

Figure 8-19 Signal Bus Setup Dialog Boxes

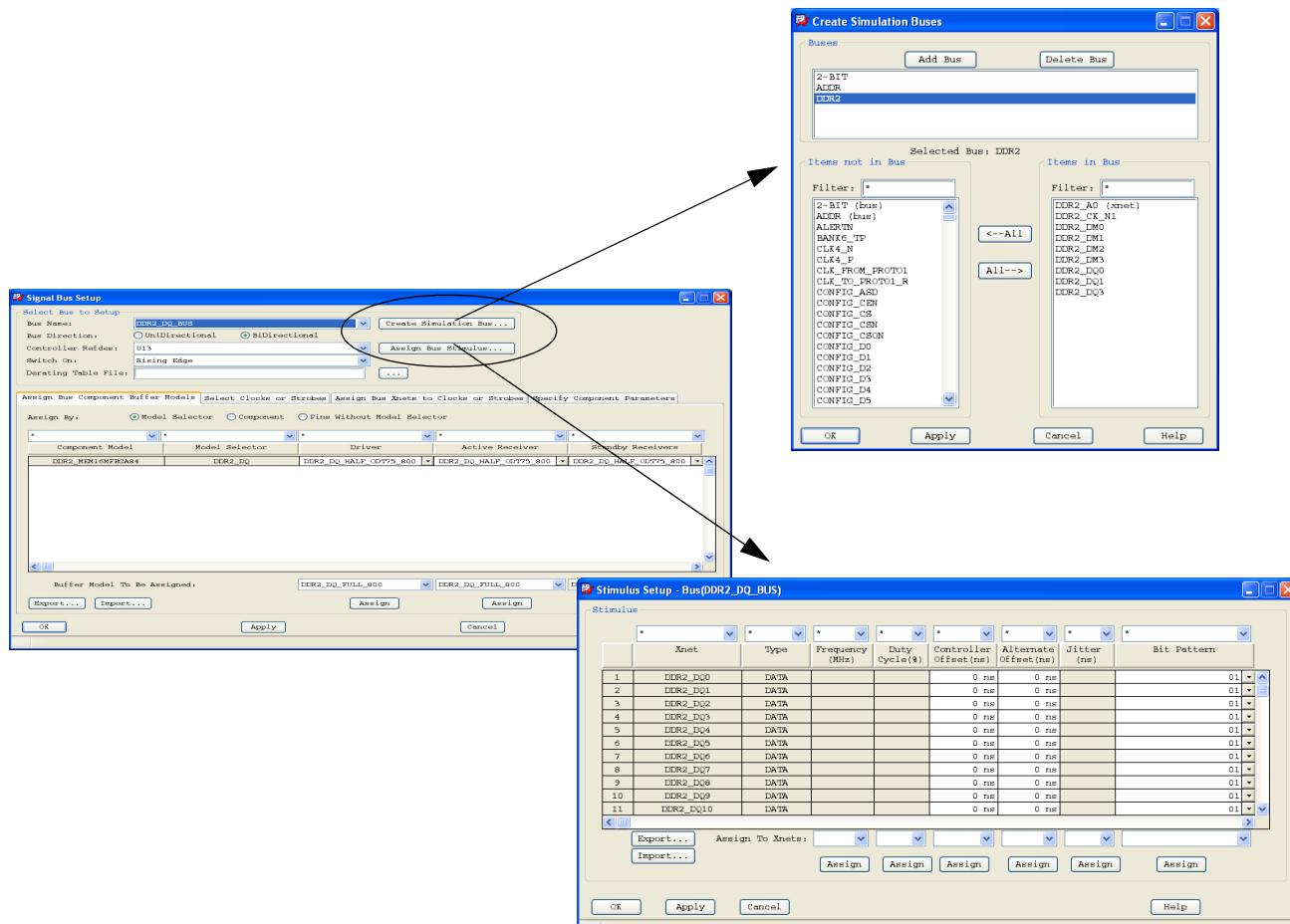
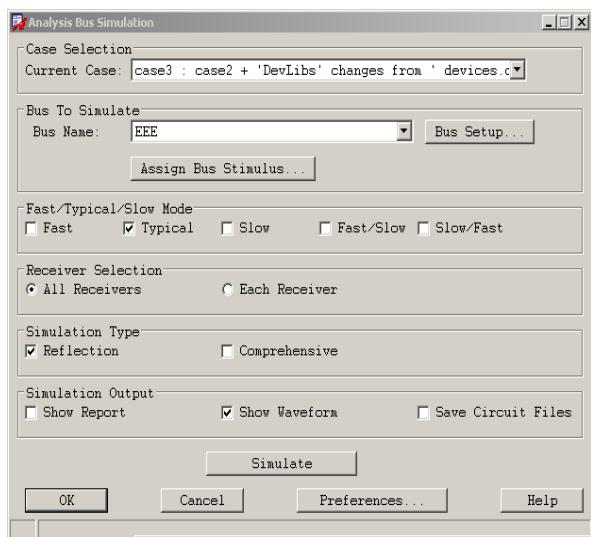


Figure 8-20 Analysis Bus Simulation Dialog Box



The Signal Bus and Stimulus Setup dialog boxes also support Import/Export functionality that allows you to import bus values from a .csv file in spreadsheet format into the dialog boxes and to export values set up in the dialog boxes to a .csv file. You can view the results of source synchronous bus analysis in the form of standard reflection summary reports, waveforms, and circuit files.

Information generated in the reports include

- worst case values for setup/hold, noise margin, and overshoot within a cycle
- data and clock/strobe slew rates
- derating table values
- AC/DC input thresholds for IO models

For complete information on the controls in all these dialog boxes as well as the recommended procedure for performing setup and simulation of source synchronous buses, see [signal bus setup](#) and [signal bus sim](#) in the *Allegro PCB and Package Physical Layout Command Reference*.

Calculating Time Margins

The bus analysis report contains all the raw data needed to determine timing closure for a source synchronous interface. The required calculations are performed to arrive at a pass/fail timing test and to report the time margin information in the bus analysis report file. The report

file includes the adjustment for derating and a total simulated (data) setup and hold time margins.

To calculate data setup and hold time margin, the following equations are used by the tool:

Equations 1 and 2

```
Tsetup_margin = Tsetup_simulated - (Tsetup_req + Tsetup_derated) - (Toffset - Tvb) ...1  
  
Thold_margin = Thold_simulated - (Thold_req + Thold_derated) + (Toffset + Tva - UI) ...2
```

Since:

(Equations 3 and 4

```
Tvb = Toffset - Tskew_max ...3
```

```
Tva = Tskew_min - Toffset + UI ...4
```

Therefore, equations 1 and 2 can also be written as:

```
Tsetup_margin = Tsetup_simulated - (Tsetup_req + Tsetup_derated) - Tskew_max ...5  
  
Thold_margin = Thold_simulated - (Thold_req + Thold_derated) + Tskew_min ...6
```

where:

Tsetup_margin	Setup time margin
Thold_margin	Hold time margin
Tsetup_simulated/ Thold_simulated	The simulated results that contain offset (SetupHigh, SetupLow, HoldHigh, HoldLow).
Tsetup_req	The minimum time the signal is required to be valid at the receiving components before the sampling edge of the strobe.
Thold_req	The minimum time the signal is required to be valid at the receiving components after the sampling edge of the strobe.
Tsetup_derated/ Thold_derated	Setup or hold time compensation according to the waveform slew rate.

Allegro PCB SI User Guide

Signal Integrity Analysis

T_{offset} Data internal offset referenced to its clock or strobe signal.

For controller pins:

```
Toffset = Tclockcontrolleroffset(ns) - Tdatacontrolleroffset(ns) ..7
```

For other pins,

```
Toffset = Tclockalternateoffset(ns) - Tdataalternateoffset(ns) ..8
```

You may set Tclockcontrolleroffset, Tdatacontrolleroffset, Tclockalternateoffset and Tdataalternateoffset in the Bus Stimulus Setup dialog box.

Tskew_max/
Tskew_min Maximum and Minimum launch skew. Launch skew is the delay skew between data output transition at die pad and clock output transition at die pad.

- Tskew_max/min is a positive quantity if data is valid after transition of the strobe.
- Tskew_max/min is a negative quantity if data is valid prior to transition of the strobe.

Tvb Signal valid time at the driving components before the sampling edge of the strobe. Tvb is a positive quantity if data is valid prior to transition of the strobe. Tvb is a negative quantity if data is valid after transition of the strobe.

Tva Signal valid time at the driving components after the sampling edge of the strobe. Tva is a positive quantity if data is valid after transition of the strobe. Tva is a negative quantity if data is valid prior to transition of the strobe.

UI Data bit duration. If the bus data is latched on both edges of the clock,

```
UI = 1 / (2* Clock or strobe Frequency) ..9
```

If the bus data is latched on single edge of the clock,

```
UI = 1 / (Clock or strobe Frequency) ..10
```

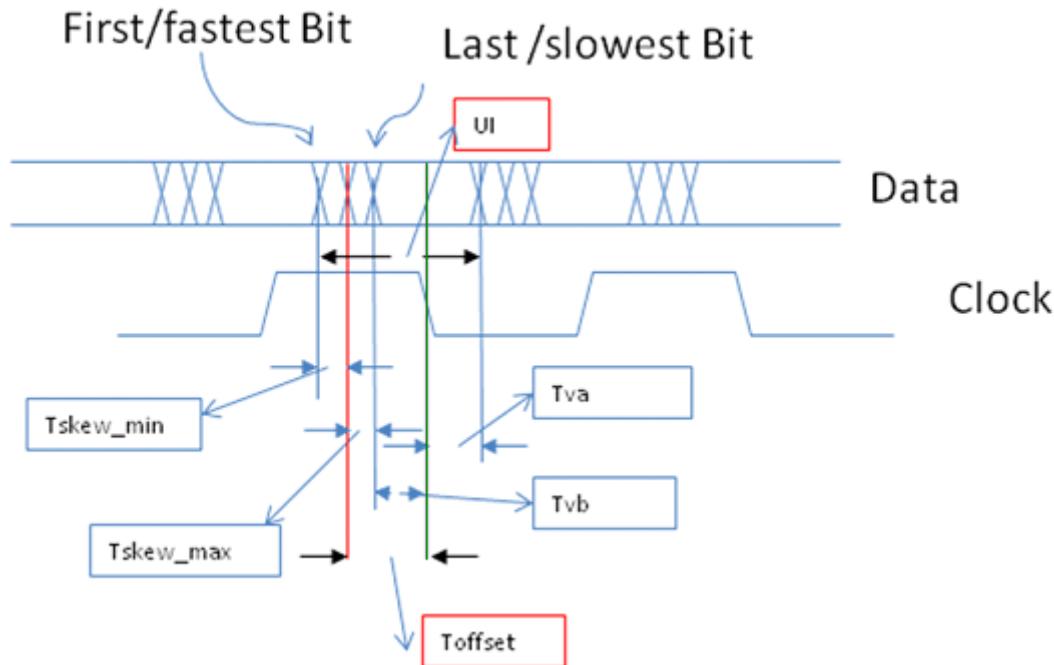
The universal formula for source synchronous timing calculation (equations [1](#) and [2](#)) depend on T_{va}/T_{vb} , setup/hold requirements, and offsets that are all specified in the GUI. You can derive T_{va} and T_{vb} from the vendor datasheet.

Equations [3](#) and [4](#) determine the relationship between `skew_min/max` and T_{va}/T_{vb} .

By default, $T_{skew_max} = 0$, $T_{skew_min} = 0$, so default T_{vb} and T_{va} are set to:

`Tvb = Toffset ..11`

`Tva = UI - Toffset ..12`

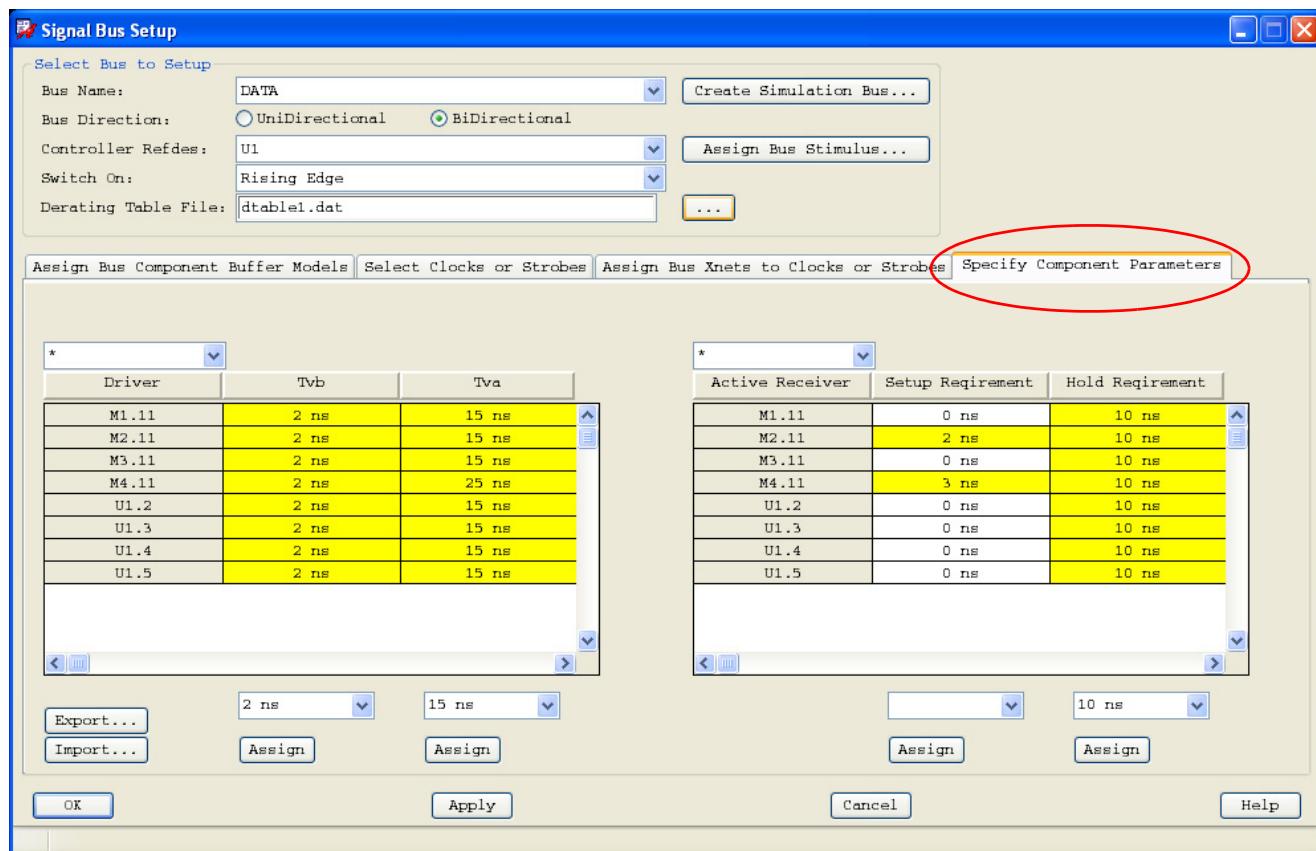


You specify the T_{vb} and T_{va} values for each driver to perform source synchronous timing calculation in the *Specify Component Parameters* tab of the Signal Bus Setup dialog. Additionally, you can assign the Setup and Hold requirements for each active receiver in this dialog box. By default, the Setup Requirement and Hold Requirement values are set to zero.

Allegro PCB SI User Guide

Signal Integrity Analysis

Figure 8-21 Signal Bus Setup



Option

Description

Filters

Use filters to speed up the process of setting values by filtering out target driver and receiver pins. In the filter field above the *Driver* and the *Active Receiver* columns, you can select or specify the component names to filter out targeted pins.

Assign

Use the drop-down fields to specify a single unique *Tvb*, *Tva*, *Setup* and *Hold requirements* value for all the target pins, and click the corresponding *Assign* button.

You can also assign unique values in the grid cell for each pin. If a cell value is updated, the back color of the changed cell turns to yellow to notify that the value has changed.

Export/Import

Export or import parameter settings in a comma separated file (CSV).

Analysis Results

SigNoise provides analysis results in the following forms.

- Ten different types of standard analysis text reports.
- Custom designed text reports.
- Waveforms and accompanying data.
- VI curves and accompanying data.
- Conductor Cross Section diagrams.

Enhanced Bus Simulation Report

Bus simulations produce a large amount of data. Analysis of the complete set of data is necessary to determine if a source synchronous sub-system meets the necessary timing requirements for a particular configuration. To assist with this task the data must be grouped in such a way that the simulation output is consistent with the way the data will be analyzed.

In the new comprehensive report for source synchronous bus, signals are grouped by strobe and then subdivided by direction (read/write). Both setup and hold are reported on the same line. Rising edge and falling edge data is reported on sequential lines.

The report is organized for easier reading and indicates whether timing margin values pass or fail based on new user input for setup and hold requirements.

Allegro PCB SI User Guide

Signal Integrity Analysis

Figure 8-22 A sample Standard Comprehensive Report for Source Synchronous Bus

```

#####
# Allegro PCB SI GXL
# 16.4 B013 (v16-4-64C) [1/21/2011]
#
# (c) Copyright 1998-2010 Cadence Design Systems, Inc.
#
# Report: Standard Comprehensive Report for Source Synchronous Bus
# Tue Feb 08 13:31:50 2011
#####

*****
Setup/Hold Times Worst Case Results: Delays in ns, SlewRate in (V/ns), Typ FTGMode, Preferred Measurement Location: Model
*****
XNet      StrobeXNet        Drvr       Rcvr      StrobePin   DataBitState  SetupMargin  HoldMargin
-----
CLEAN DDR2_DQ0  NA          CLEAN U13 AL20  CLEAN U28 G8  NA          High         0            0
CLEAN DDR2_DQ0  NA          CLEAN U13 AL20  CLEAN U28 G8  NA          Low          0            0
CLEAN DDR2_DQ0  CLEAN CONFIG_CS  CLEAN U28 G8  CLEAN U13 AL20  CLEAN U13 AC22  High         NA           NA
CLEAN DDR2_DQ0  CLEAN CONFIG_CS  CLEAN U28 G8  CLEAN U13 AL20  CLEAN U13 AC22  Low          NA           NA
CLEAN DDR2_DQ1  CLEAN CONFIG_CS  CLEAN U28 G2  CLEAN U13 AH19  CLEAN U13 AC22  High         NA           NA
CLEAN DDR2_DQ1  CLEAN CONFIG_CS  CLEAN U28 G2  CLEAN U13 AH19  CLEAN U13 AC22  Low          NA           NA
CLEAN DDR2_DQ1  NA          CLEAN U13 AH19  CLEAN U28 G2  NA          High         0            0
CLEAN DDR2_DQ1  NA          CLEAN U13 AH19  CLEAN U28 G2  NA          Low          0            0
CLEAN DDR2_DQ2  NA          CLEAN U13 AJ19  CLEAN U28 H7  NA          High         0            0
CLEAN DDR2_DQ2  NA          CLEAN U13 AJ19  CLEAN U28 H7  NA          Low          0            0
CLEAN DDR2_DQ2  CLEAN CLK_FROM_PROTO1  CLEAN U28 H7  CLEAN U13 AJ19  CLEAN U13 T32  High         NA           NA
CLEAN DDR2_DQ2  CLEAN CLK_FROM_PROTO1  CLEAN U28 H7  CLEAN U13 AJ19  CLEAN U13 T32  Low          NA           NA
CLEAN DDR2_DQ3  CLEAN BANK6_TP  CLEAN U28 H3   CLEAN U13 AH20  CLEAN U13 Y10  High         NA           NA
CLEAN DDR2_DQ3  CLEAN BANK6_TP  CLEAN U28 H3   CLEAN U13 AH20  CLEAN U13 Y10  Low          NA           NA

```

The standard comprehensive report contains the following headers:

Table 8-2 Standard Comprehensive Report for Source Synchronous Bus

Column Header	Description
Xnet	All the Xnets in the design.
StrobeXNet	The Xnets identified as strobe Xnets during the bus analysis setup.
Drv	The driver pin.
Rcvr	The receiver pin.
StrobePin	The Strobe pin.

Table 8-2 Standard Comprehensive Report for Source Synchronous Bus

Column Header	Description
DataBitState	The state of the data bit. The values are High and Low.
SetupMargin	The margin which is left after subtracting SetupRequirement from SetupTime which you get from the simulation result.
HoldMargin	The margin which is left after subtracting HoldRequirement from HoldTime which you get from the simulation result.
SetupTime	The simulated time which you get by subtracting the time when data signal reaches the high threshold voltage value (before the sampling edge of the strobe) from the time when strobe reaches reference voltage level.
SetupCycle	The clock cycle at which setup time is measured.
SetupDataSlew	The rate of change of data signal that is voltage w.r.t time and it is denoted by V/ns.
SetupClkSlew	The rate of change of clock signal that is voltage w.r.t time and it is denoted by V/ns.
SetupDerVal	The setup derate value that needs to be compensated from the base SetupRequirement as per the slew rate. If the slew rate is slow, the SetupRequirement reduces. If the slew rate is fast, the SetupRequirement increases. <i>Setup-time requirement = Base-setup time requirement + delta SetupDerVal</i> where <i>delta</i> can be +ve or -ve depending on the slew rate

Allegro PCB SI User Guide
Signal Integrity Analysis

Table 8-2 Standard Comprehensive Report for Source Synchronous Bus

Column Header	Description
SetupRequirement	The minimum time the signal is required to be valid at the receiving components before the sampling edge of the strobe.
Tvb	The signal valid time at the driving components before the sampling edge of the strobe.
HoldTime	The simulated time obtained by subtracting the time when strobe reaches reference voltage level from the time when data signal reaches the DC threshold voltage value after the sampling edge of the strobe.
HoldCycle	The clock cycle at which hold time is measured.
HoldDataSlew	The rate of change of data signal that is voltage w.r.t time and it is denoted by V/ns .
HoldClkSlew	The rate of change of clock signal that is voltage w.r.t time and it is denoted by V/ns .
HoldDerVal	The hold derate value that needs to be compensated from the base hold requirement as per the slew rate. If the slew-rate is slow, the hold requirement reduces. If the slew rate is fast, hold time requirement increases. <i>Hold-time requirement = Base-Hold time requirement + delta HoldDerVal</i> where <i>delta</i> can be +ve or -ve as per slew rate
HoldRequirement	The minimum time the signal is required to be valid at the receiving components after the sampling edge of the strobe.

Table 8-2 Standard Comprehensive Report for Source Synchronous Bus

Column Header	Description
Tva	The signal valid time at the driving components after the sampling edge of the strobe.

Analyzing to Generate Text Reports

When you perform analysis for signal integrity or EMI emissions by generating text reports, SigNoise performs the necessary simulations based on the selections you make in the Signal Analysis and Analysis Report Generator dialog boxes.

Having both dialog boxes open together, you can switch back and forth between them, selecting nets and pins for analysis from the Signal Analysis dialog box, and specifying report and simulation details from the Analysis Report Generator dialog box.

After examining the report data, you can then refine your net / pin selection and simulation details, change simulation preferences (if necessary), and perform more specific analysis to pinpoint problem signals.

The following table describes the different text reports that are available.

Table 8-3 Standard Analysis Reports

Type	Description
Reflection Summary	Gives delay and distortion data in a concise, summary format. See “Reflection Summary Report” on page 288.
Delay	Gives propagation delays, switch delays (rising and falling edge), settle delays (rising and falling edge), and reports a pass or fail status for first incident rise and fall and monotonic rise and fall heuristics for selected nets. See “Delay Report” on page 306.
Ringing	Gives overshoot and noise margin values for selected nets. See “Ringing Report” on page 311.
Single Net EMI	Gives essential EMI data for the net in a concise, single-line format. See “Single Net EMI Report” on page 317.
Parasitics	Gives total self capacitance, impedance range, and transmission line propagation delays for selected nets. See “Parasitics Report” on page 320.

Table 8-3 Standard Analysis Reports

Type	Description
SSN	Gives noise levels induced on a component's power and ground busses when drivers on that bus switch simultaneously. See " SSN Report " on page 322.
Segment-based Crosstalk Estimation	Presents detailed <i>segment-based coupling</i> information derived from xtalk tables. See " Segment Crosstalk Report " on page 327.
Crosstalk Summary	Gives peak and total crosstalk for selected nets in a concise, summary format. Crosstalk values are derived from multi-line simulations. See " Crosstalk Summary Report " on page 333.
Crosstalk Detailed	Gives total crosstalk on selected nets. For all cases simulated, crosstalk values are derived from multi-line simulations. See " Crosstalk Detailed Report " on page 336.

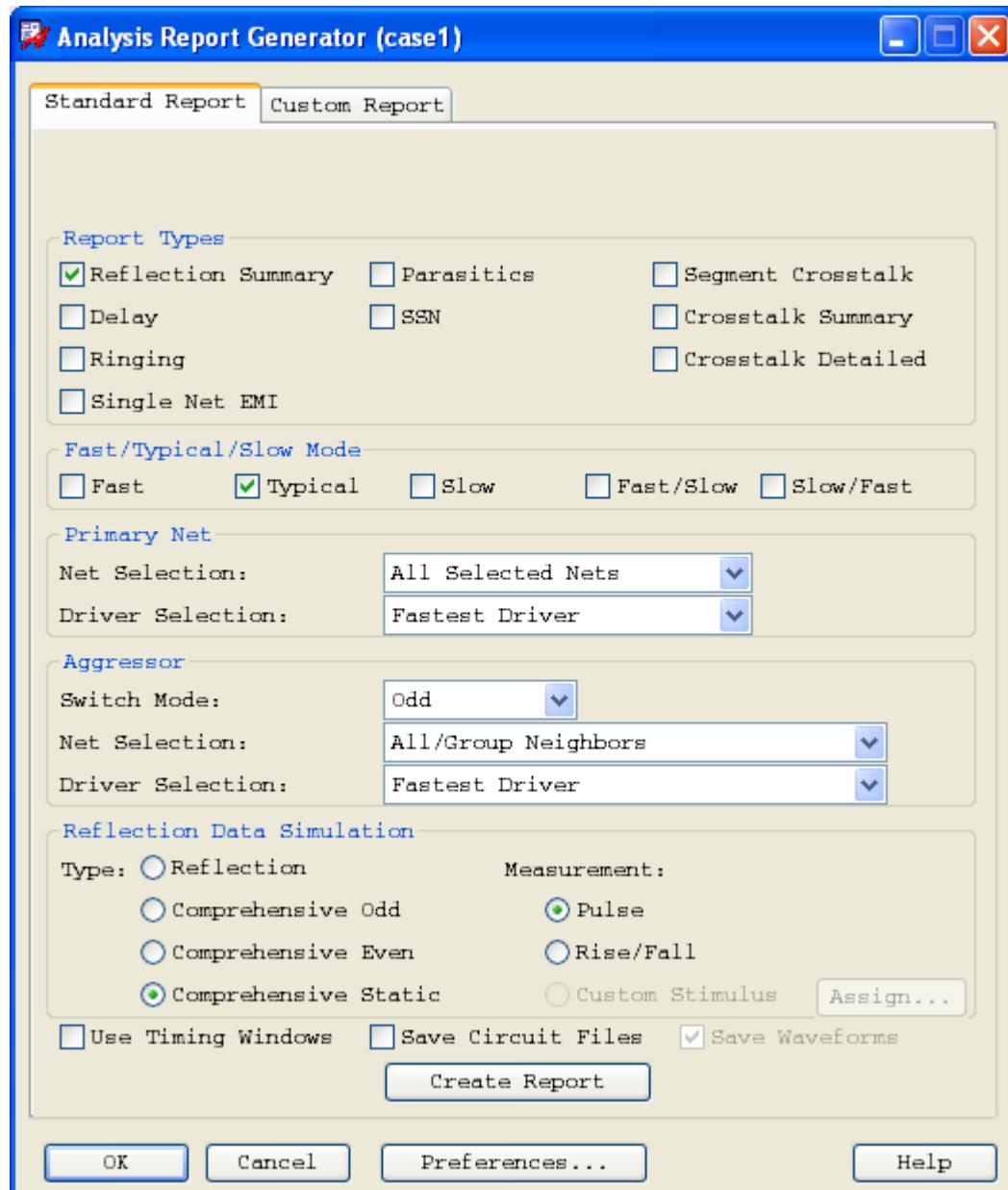
To begin text report based analysis from the PCB SI

- Click *Reports* in the Signal Analysis dialog box. See [Figure 8-15](#) on page 263.
The Analysis Report Generator dialog box displays as shown in [Figure 8-23](#) on page 286.

Allegro PCB SI User Guide

Signal Integrity Analysis

Figure 8-23 The Analysis Report Generator Dialog Box - Standard Report Tab

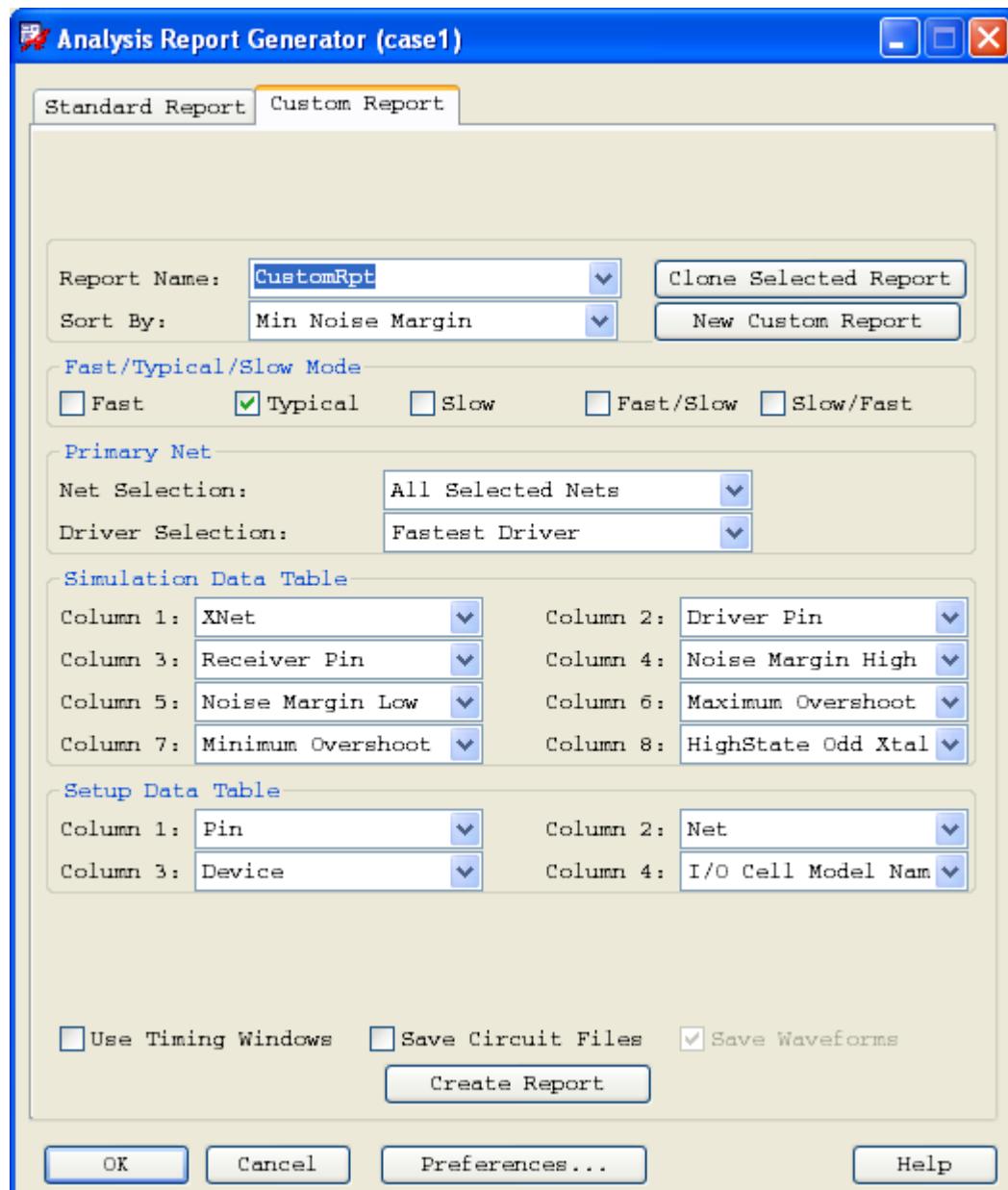


The Custom Report tab is shown in [Figure 8-24](#) on page 287.

Allegro PCB SI User Guide

Signal Integrity Analysis

Figure 8-24 The Analysis Report Generator Dialog Box - Custom Report Tab



In the Analysis Report Generator dialog box you can:

- select one of ten standard report types.
- specify the format for a custom report.
- select simulation details common to both Standard and Custom reports.

- select whether or not to use timing windows and save circuit files and waveform files.
- display the Analysis Preferences dialog box to modify simulation preferences.
- display the Stimulus Setup dialog box to assign custom stimulus parameters.
- run simulations and generate reports based on the selection criteria.

For details on specific options and buttons in the Analysis Report Generator dialog box or for a list of procedures regarding simulation text report generation, refer to the [signal probe](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

Reflection Summary Report

The Reflection summary report presents simulation results for propagation delays, switch delays (rising and falling edge), and settle delays (rising and falling edge). It also reports a pass/fail status for first incident rise and fall and monotonic rise and fall for selected nets.

In the case of multiple receivers on a net, the Reflection summary report shows only the worst case. While the delay report shows data for *all* receivers, the Reflection summary is generally a good first cut when you analyze the entire board (simulating all nets) as it limits the data to one line per receiver net.

The Reflection report can be generated either in batch mode, or interactively from the Analysis Report Generator dialog box.

Batch Generation

Following is an example of a batch command which would generate a Reflection summary report:

```
signoise -b Die/Die -f my_nets.txt -r ReflectionSummary -s reflection -o  
ref_rpl.txt my.brd
```

Table 8-4 Batch Command Switches - Reflection Summary Report

Switch	Description
-b	Location from which to measure results (model/pin/die)
-f	A list-of-nets file
-r	Report type to generate
-s	Simulation type to perform (Reflection or Comprehensive)
-o	Name of the output file to create

Interactive Generation

Selection of the *Reflection Summary* option in the *Analysis Report Generator* dialog box specifies a Reflection summary report for selected nets. See [Figure 8-23](#) on page 286.

Sample Reflection Summary Report

Note: Some report sections are split to fit the page.

```
#####
# Allegro PCB SI 15.5
# (c) Copyright 2004 Cadence Design Systems, Inc.
#
# Report: Standard Reflection Summary Sorted By Worst Settle Delay
#       Wed Feb  9 14:13:48 2004
#####
*****  

Delays (ns), Distortion (mV), (Typ FTSMode) Preferred Measurement Location: Pin
*****
```

XNet	Drvrv	Rcvr	NMHigh	NMLow	OShootHigh
-----	-----	-----	-----	-----	-----
1 memory A4	memory J47 35	memory U18 29	678.7	-84.46	4001
1 memory A4	memory J47 35	memory U3 29	991	819.1	3991
1 memory -CAS	memory J47 111	memory U14 17	-30.5	769.2	3984
1 memory A0	memory J47 33	memory U14 23	643.1	-39.78	3994

Allegro PCB SI User Guide

Signal Integrity Analysis

1 memory -CAS	memory J47 111	memory U7 17	-24.21	777.6	3976
1 memory BA0	memory J47 122	memory U18 20	624.8	-29.29	3979

....

Sample Report (continued)

OShootLow	SwitchRise	SwitchFall	SettleRise	SettleFall	Monotonic
334.6	4.667	2.01	8.236 *	3.169 *	FAIL
313.4	4.689	2.027	8.183 *	3.155 *	FAIL
253.1	4.828	2.037	8.17 *	3.145 *	FAIL
250.6	4.769	1.98	8.159 *	3.115 *	FAIL
248.4	4.843	2.04	8.135 *	3.136 *	FAIL
273.8	4.687	1.923	8.133 *	3.151 *	FAIL

....

Pulse Data Per Xnet

XNet	PulseFreq	PulseDutyCycle	PulseCycleCount
1 memory WP	50MHz	0.5	1
1 memory UN4CAP226PA0	50MHz	0.5	1
1 memory UN4CAP225PA0	50MHz	0.5	1
1 memory SDA	50MHz	0.5	1
1 memory SCL	50MHz	0.5	1

Allegro PCB SI User Guide

Signal Integrity Analysis

1 memory SA2 50MHz 0.5 1

....

Description of column abbreviations

Column	Description
XNet	Extended net
Drvrv	Driver Pin
Rcvr	Receiver Pin
NMHigh	Noise Margin High
NMLow	Noise Margin Low
OShootHigh	Maximum Overshoot
OShootLow	Minimum Overshoot

Measurement Location

Pin and/or die measurement location for driver and receiver can be determined from the DML model defined in your setup, from the external pin node, or from the internal die node, if present. (Die pad measurements are relevant only to Reflection, Delay and Ringing reports as well as related Custom and Comprehensive reports.) You can set these choices in the signoise batch command or by way of the Analysis Preferences dialog box.

Note: Editing measurement locations by way of the defined DML model entails manually changing the DML file by adding or deleting the appropriate keywords using the correct syntax in the proper section. Pin and die measurement locations are made at the external pin node and internal die node, respectively.

To distinguish in the report whether the measurement is being made at the pin pad or the die pad, the following convention is used:

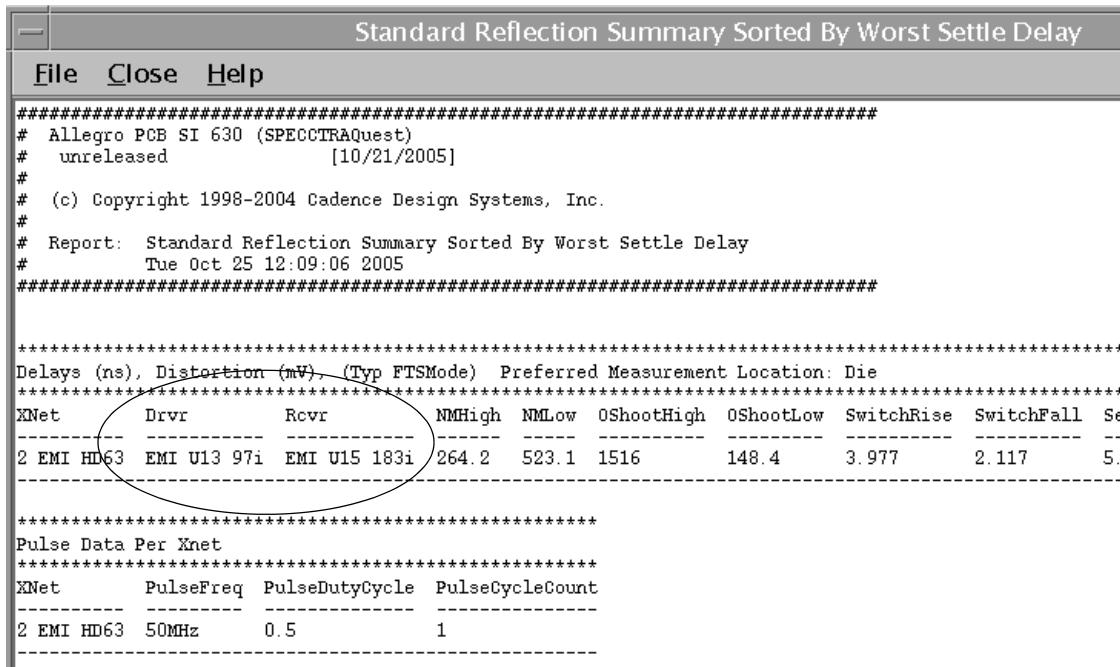
- If taken at the pin pad, the pin pad measurement name is identical to the pin name (for example, PIN5).
- If taken at the die pad location, the pin name is displayed with an i appended to it (for example, Pin5i).

Allegro PCB SI User Guide

Signal Integrity Analysis

The following figure illustrates a reflection summary report displaying die pad location results.

Figure 8-25 Reflection Summary Report with Die Location Specification



```
Standard Reflection Summary Sorted By Worst Settle Delay
File Close Help
#####
# Allegro PCB SI 630 (SPECCTRAQuest)
# unreleased [10/21/2005]
#
# (c) Copyright 1998-2004 Cadence Design Systems, Inc.
#
# Report: Standard Reflection Summary Sorted By Worst Settle Delay
# Tue Oct 25 12:09:06 2005
#####

*****
Delays (ns), Distortion (mV), (Typ FTSMode) Preferred Measurement Location: Die
*****
XNet Drvr Rcvr NMHigh NMLow OShootHigh OShootLow SwitchRise SwitchFall Se
-----
2 EMI HD63 EMI U13 97i EMI U15 183i 264.2 523.1 1516 148.4 3.977 2.117 5.

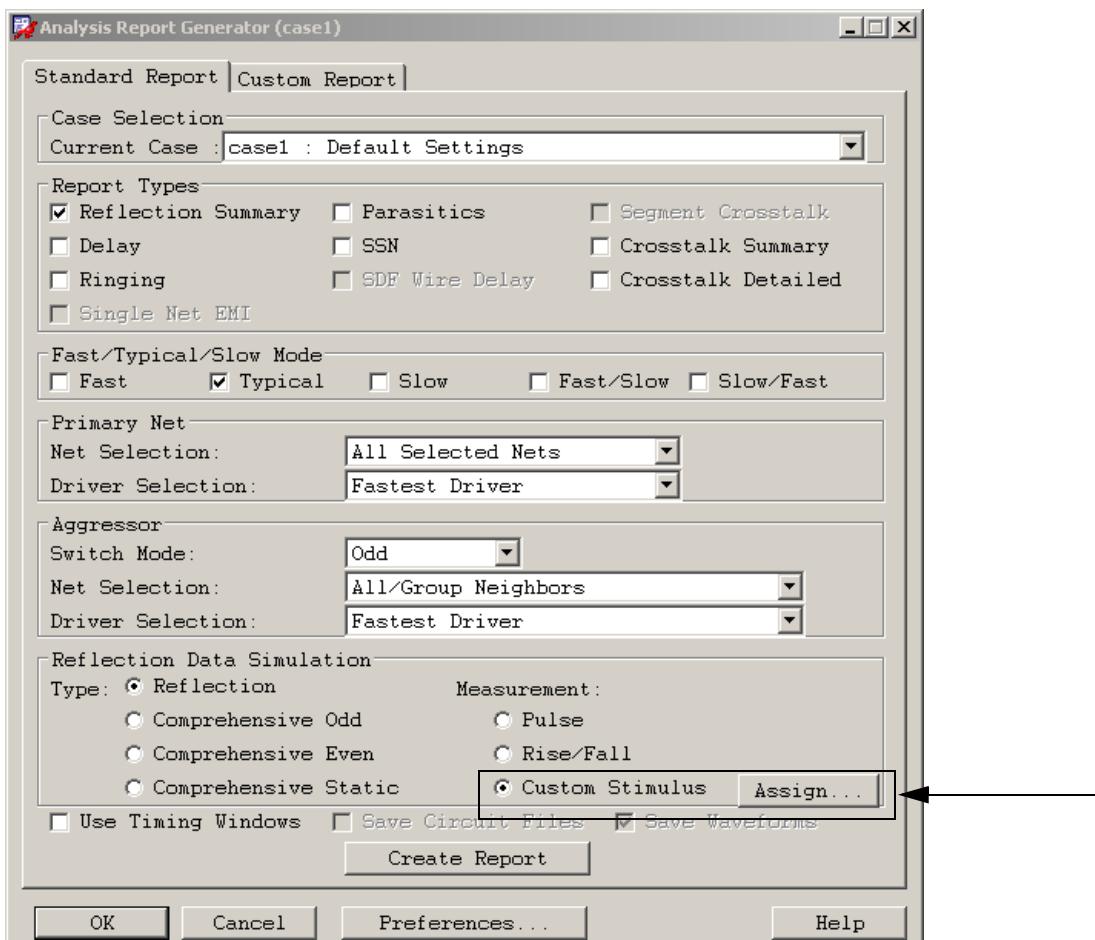
*****
Pulse Data Per Xnet
*****
XNet PulseFreq PulseDutyCycle PulseCycleCount
-----
2 EMI HD63 50MHz 0.5 1
```

Simulating with Custom Stimulus

You can drive analysis at the board level with the *Custom Stimulus* option. This attaches the PULSE_PARAM property that defines pulse parameter data of nets and Xnets. You access the feature from two areas of the GUI when you run *Analyze – Probe*:

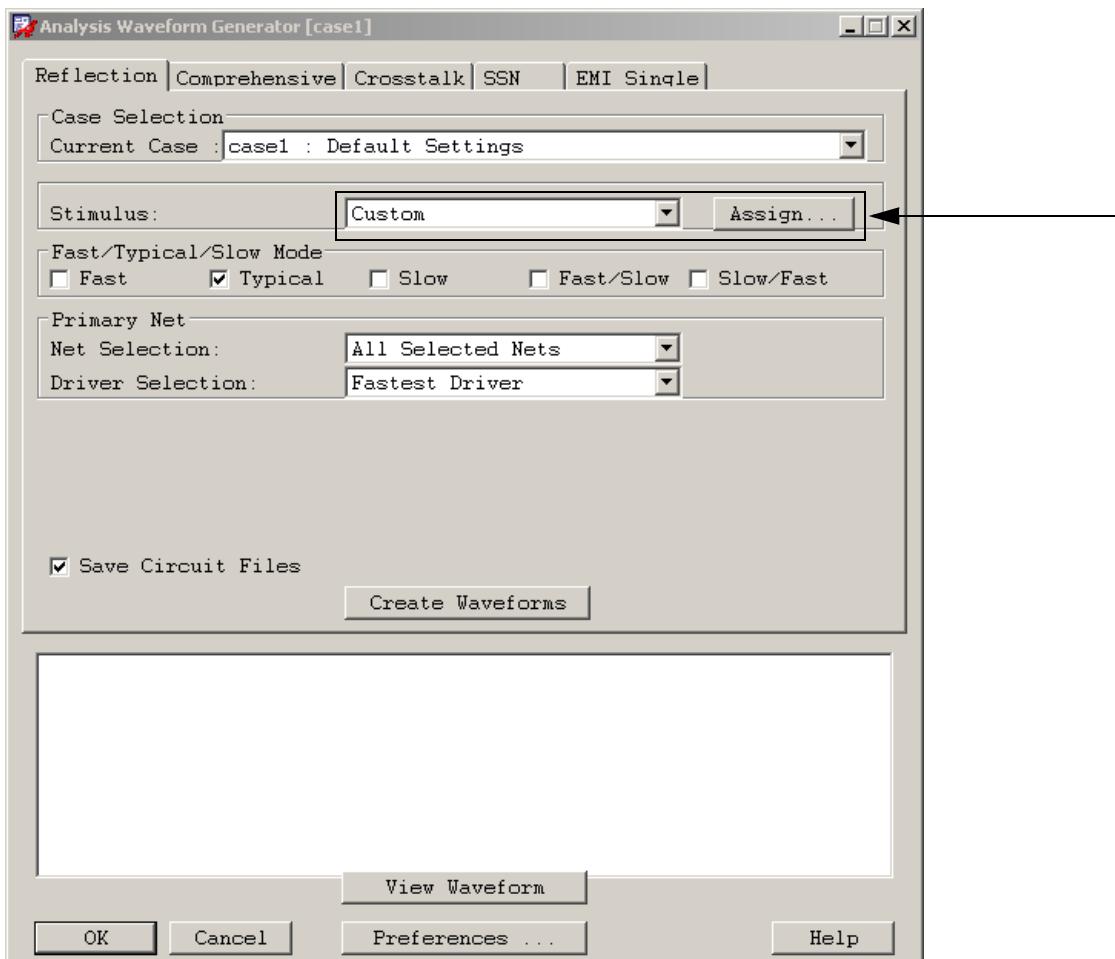
- The Reflection Data Simulation section of the Analysis Report Generator for Standard reports (Figure 8-26)

Figure 8-26 Analysis Report Generator



- The Stimulus selections section of the Analysis Waveform Generator for Reflection waveforms (Figure 8-27)

Figure 8-27 Analysis Waveform Generator



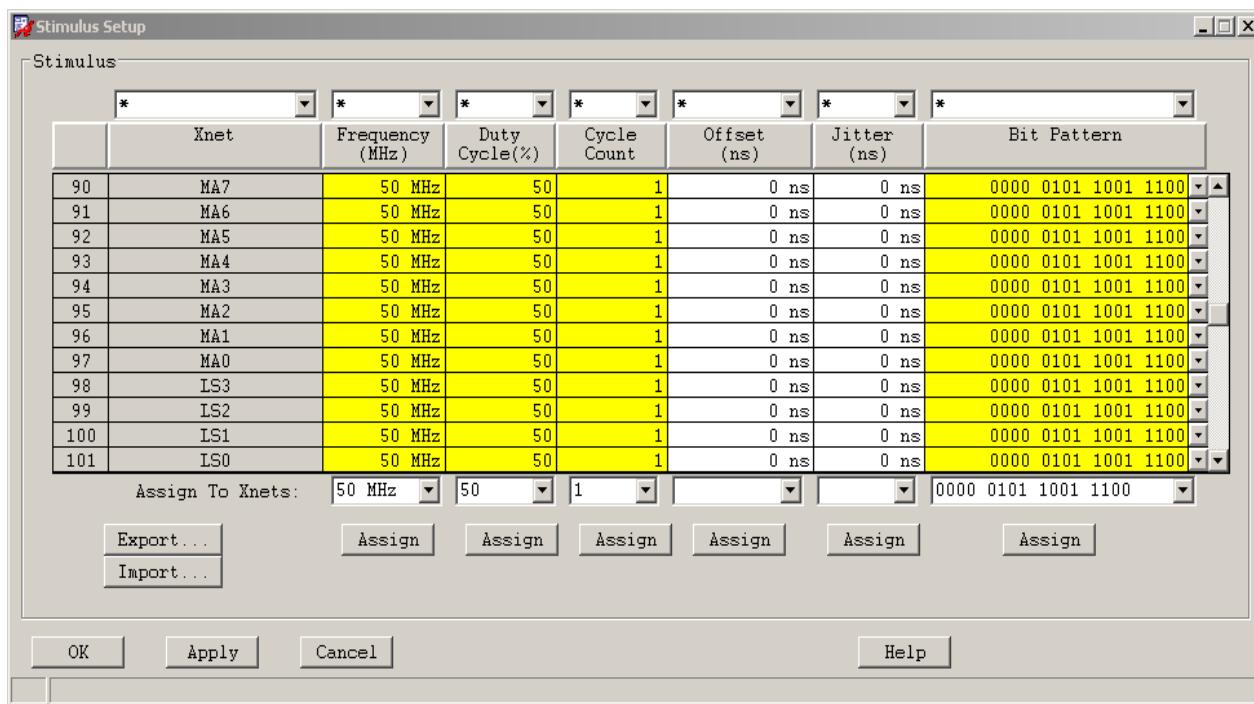
The Stimulus Setup dialog box (Figure 8-28) allows you to assign predefined custom stimuli to all drivers in your current simulation through. From there, pre-loaded nets and extended nets that you have selected from your board can be assigned frequency, cycle count, offset, jitter, and bit pattern values. You can save these settings to a .csv-formatted spreadsheet file. Modifications that you make in the spreadsheet for existing nets can then be imported back into SI.

Note: This functionality supersedes the .inc custom stimulus files which continue to be supported, but will be overridden with the values you set in the Stimulus Setup dialog.

Allegro PCB SI User Guide

Signal Integrity Analysis

Figure 8-28 Stimulus Setup Dialog Box



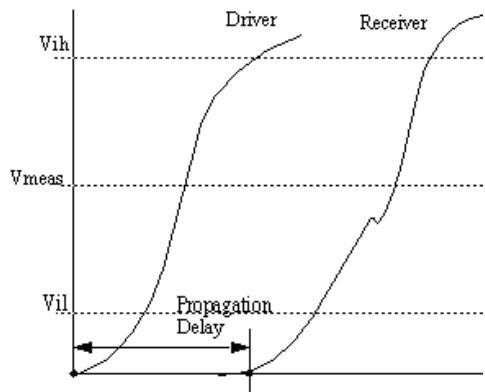
Report Computations

Delay Criteria

Propagation Delay

Propagation delay is the summation of all calculated transmission line delays along the shortest path between two points. Although propagation delay is a calculated value, Tlsm (the simulator) performs the calculation since it is the only tool that has a system level view of the transmission line paths.

Figure 8-29 Propagation Delay Measurement Points



Propagation Delay Simulation

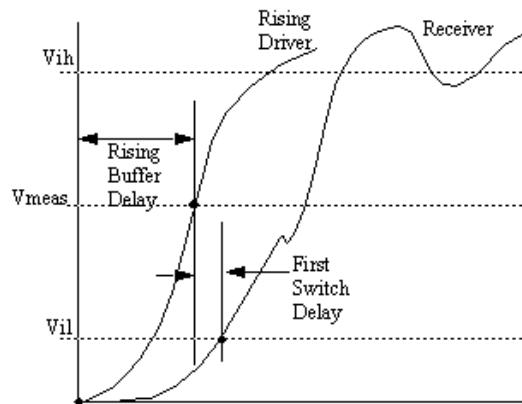
Propagation delay is measured from any simulation available. SigNoise performs a Reflection simulation with pulse stimulus if no simulation results are available. Propagation delay is used for the `DELAY_RULE` and `MATCHED_DELAY` constraints.

First Switch Delay

For a rising edge, the simulation measurement is from time zero to when the receiver first crosses V_{il} , the low voltage switching threshold. The associated rising buffer delay for the driving IOCell is subtracted from this measurement value to produce the reported first switch delay.

For a rising edge: First switch delay = time to reach V_{il} - buffer delay

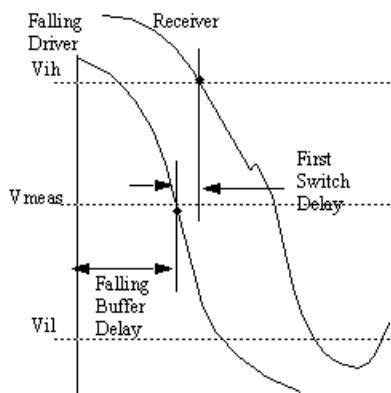
Figure 8-30 Rising Edge Switch Delay Measurement Points



For a falling edge, the simulation measurement is from time zero to when the receiver first crosses V_{ih} , the high voltage switching threshold. The associated falling buffer delay of the driving IOCell is subtracted from this measurement value to produce the reported first switch delay.

For a falling edge: First Switch = time to reach V_{ih} - buffer delay

Figure 8-31 Falling Edge Switch Delay Measurement Points



Switch Delay Simulation

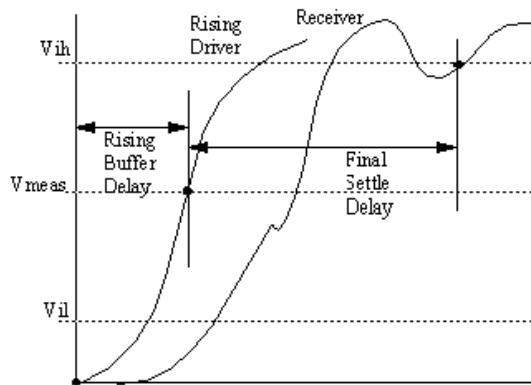
SigNoise performs either a Reflection or Comprehensive simulation with pulse stimulus to collect a first switch delay measurement which is used for the MIN_FIRST_SWITCH constraint.

Final Settle Delay

Final settle delay is the time to reach the second threshold voltage encountered and stay above or below it, minus the Buffer Delay for the driver.

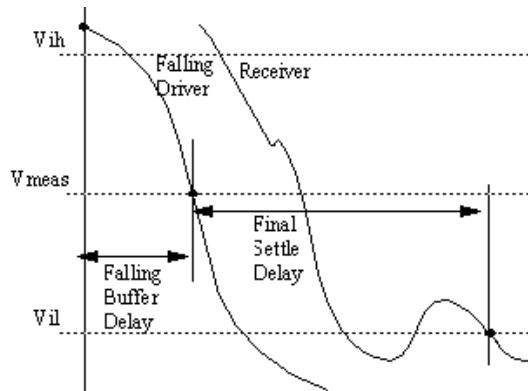
For a rising edge, the simulation measurement is from time zero to when the receiver crosses V_{ih} , the high threshold voltage, the final time and settles into the high logic state. The associated rising buffer delay for the driving IOCell is subtracted from this measurement value to produce the reported final settle delay as shown in [Figure 8-32](#).

Figure 8-32 Rising Edge Settle Delay Measurement Points



For a falling edge, the simulation measurement is from time zero to when the receiver first crosses V_{ih} , the high voltage switching threshold. The associated falling buffer delay of the driving IOCell is subtracted from this measurement value to produce the reported first switch delay as shown in [Figure 8-33](#) on page 299.

Figure 8-33 Falling Edge Delay Measurement Points



Settle Delay Simulation

SigNoise performs either a Reflection or a Comprehensive simulation with pulse stimulus to collect a final switch delay measurement which is used for the MAX_FINAL_SETTLE constraint.

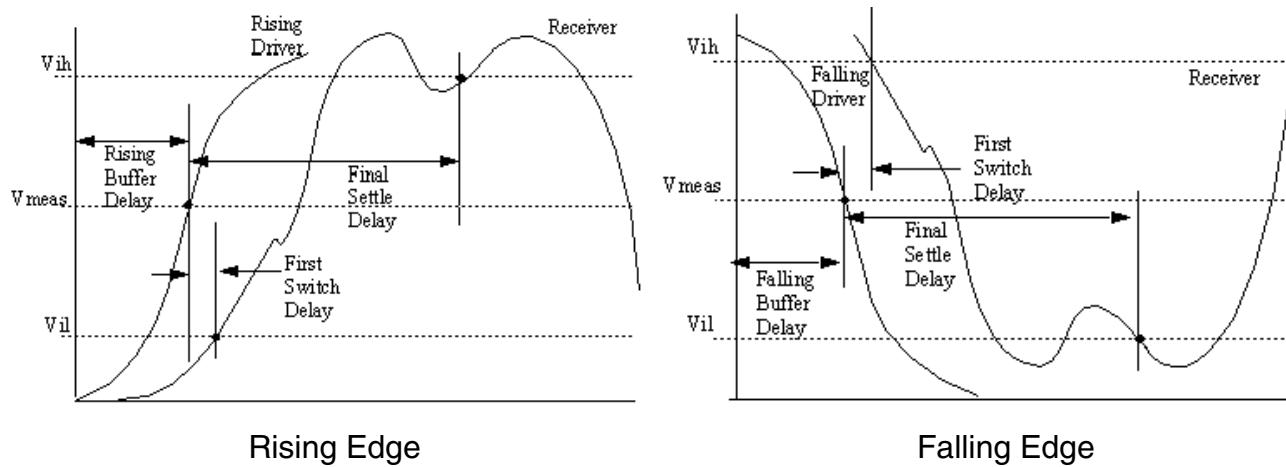
Buffer Delay

Buffer delay is the time it takes the voltage of a driver to reach a predefined measurement voltage, V_{meas} , when driving a standard test load. Buffer delay is subtracted from the absolute time for a receiver waveform to reach a logic threshold. The difference between these two measurements represents the portion of the delay attributable to interconnect effects. Buffer delay is measured for both rising and falling edges.

When measuring waveforms at a receiver against time zero in a simulation, the buffer delay, or driving IOCell delay, is included as well as the delay contributed by the interconnect. For the purpose of timing analysis, the buffer delay is already accounted for in the overall component delay. In order that the buffer delay is not counted twice, the assumed buffer delay is subtracted from the simulation results when reporting first switch and final settle delays.

Since the actual topology to which each pin is attached is not available for up-front timing analysis, a test load (or test fixture) is assumed to be attached to the buffer in order to derive the component delay. SigNoise hooks up the IOCell to its corresponding test load circuit and runs simulations to capture the slow, typical, and fast buffer delay values, measured at V_{meas} for rising and falling edges.

Figure 8-34 Buffer Delay Measurement Points



Buffer Delay Simulation

When a simulation is run for a design, the appropriate buffer delay is subtracted to properly compensate switch and settle delays so that these delay measurements represent interconnect contribution only.

The buffer delay selection information you enter in the PCB SI, Allegro Editor, or SigXplorer *Analysis Preferences* dialog box allows you to specify how SigNoise should obtain the buffer delay values to use during the simulation. You can instruct SigNoise to retrieve stored buffer delay values from the device model or to measure buffer delay at the start of the simulation.

Note: Use the Buffer Delay Selection options on the *Device Models* tab to set *From Library* or *On-the-Fly*, or to *No Buffer Delay*.

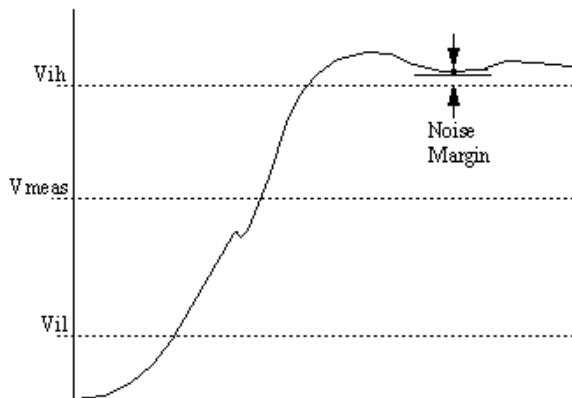
Once measured, buffer delay is stored with the pin data for the individual pins of an IBIS Device Model (unless you selected *No Buffer Delay*, which assumes 0ns buffer delays). Buffer delay values for a pin are found on the Buffer Delays dialog box which is accessible from the IBIS Device Pin Data dialog box of the IBIS Device Model Editor. When buffer delay values are not available in the IBIS device model, buffer delays are not subtracted from the reported first switch and final settle delay values. You may use the *On-the-Fly* buffer delay method to compute the buffer delays along with other simulation results in that case.

Distortion Criteria

Noise Margin

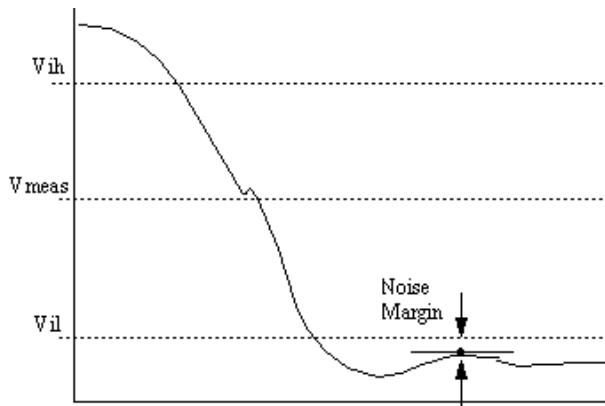
For a rising edge, high state noise margin measures how close the high state receiver waveform comes to the high state switching threshold, V_{ih} . This measurement, V_{min} , is taken after the waveform crosses V_{ih} and before the onset of a falling transition that crosses both thresholds (falling side of the pulse).

Figure 8-35 Rising Edge Noise Margin Measurement Points



For a falling edge, low state noise margin measures how close the low state signal comes to the low switching threshold. This measurement is taken after crossing the low switching threshold, and before the onset of a rising transition that crosses both thresholds (rising side of the pulse).

Figure 8-36 Falling Edge Noise Margin Measurement Points



Noise Margin Simulation

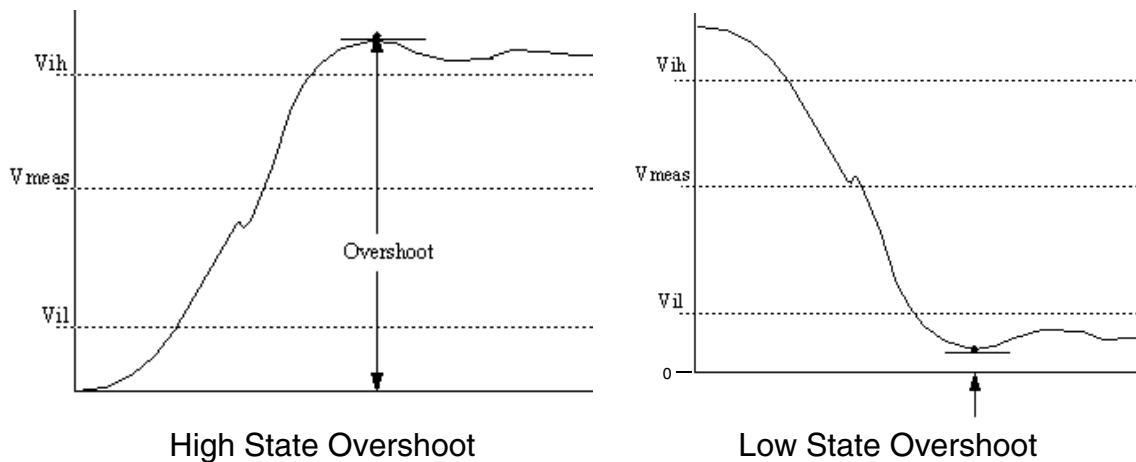
SigNoise performs either a Reflection or a Comprehensive simulation to collect the noise margin measurement which is used for the MIN_NOISE_MARGIN constraint.

Overshoot

Overshoot is the maximum voltage excursion of a signal measured in absolute voltage units. Note that the overshoot voltages are measured relative to the zero volt ideal ground, not the steady state value of the signal, V_{ss} .

For a rising edge, high state overshoot is the highest voltage seen. For a falling edge, low state overshoot is the lowest voltage seen.

Figure 8-37 Overshoot Measurement Points



Overshoot Simulation

SigNoise performs either a Reflection or a Comprehensive simulation to collect overshoot measurements which are used for the MAX_OVERSHOOT constraint. Note that for rising edges, the high state overshoot is greater than MAX_OVERSHOOT and for falling edges, the low state overshoot is less than MAX_OVERSHOOT.

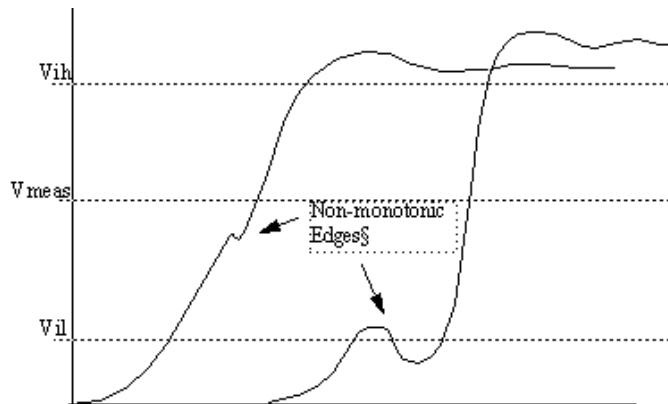
Non-Monotonic Edge

Non-monotonic edge is a PASS or FAIL status value indicating whether an edge is monotonic or not. A rising edge is monotonic if each next point in time has a greater voltage value than the previous point until it crosses V_{vh} . A falling edge is monotonic if each next point in time has a smaller voltage value than the previous point until it crosses V_{vl} .

A non-monotonic edge is considered significant for clock signals. The presence of a non-monotonic edge is regarded as non-monotonic switching.

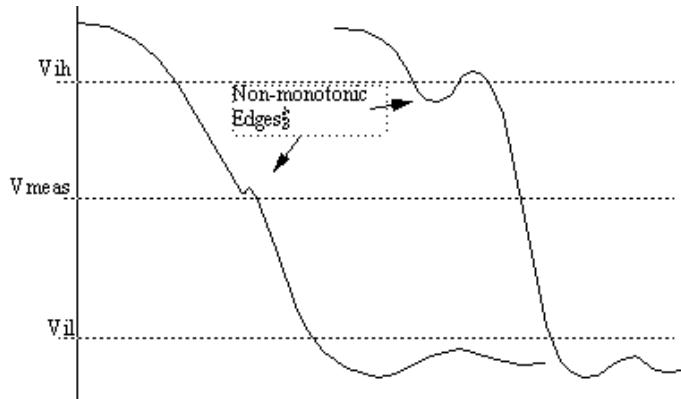
For a rising edge, a non-monotonic edge is a signal reversal that occurs after crossing the low voltage threshold, V_{il} , but before the signal reaches the high voltage threshold, V_{vh} .

Figure 8-38 Rising Edge Non-Monotonic Edge Measurement Points



For a falling edge, a non-monotonic edge is a signal reversal that occurs after crossing the high voltage threshold, V_{ih} , but before the signal reaches the low voltage threshold, V_{il} .

Figure 8-39 Falling Non-Monotonic Edge Measurement Points



Non-Monotonic Edge Simulation

SigNoise performs either a Reflection or a Comprehensive simulation to collect the non-monotonic edge data which is used for the EDGE_SENS constraint.

Electrical Constraints

The following tables show constraints that you can set on a net or on a pin-to-pin connection for evaluation during delay analysis.

Note: These constraints are only checked by SigNoise.

Table 8-5 Measure Delay Value Comparisons

Constraint	Definition	Format
MAX_OVERSHOOT	Limits the minimum and maximum absolute voltage for a receiver.	maximum value:minimum value
MIN_NOISE_MARGIN	Limits the amount of ringback as compared to the receiver's voltage thresholds.	maximum value:minimum value
MIN_FIRST_SWITCH	Limits the time to reach and stay above/below the low/high switching threshold voltage.	maximum value:minimum value
MAX_FINAL_SETTLE	Limits the time to reach and stay above/below the high/low switching threshold voltage.	maximum value:minimum value
EDGE_SENS	Flags whether or not an Xnet/Net is sensitive to non-monotonicity in the receiver waveform. If this constraint is not set, the net is insensitive.	rising:falling:both

Table 8-6 Timing Rule Constraint Checks

Constraint	Item
MIN_FIRST_SWITCH	Minimum first switch delay
MAX_FINAL_SETTLE	Maximum final settle delay

Delay Report

The delay report presents simulation results for propagation delays, switch delays (rising and falling edge), and settle delays (rising and falling edge). It also reports a pass or fail status for first incident rise and fall and monotonic rise and fall for selected nets. This report is good for checking clock nets, particularly to detect non-monotonic rise or fall.



Important

You can use the delay values and First Incident Switch heuristic to check data nets, but they are *not* a substitute for full-path-based timing analysis which is recommended. It is possible that adjusting interconnect lengths or terminating to achieve first incidence switching will solve problems found by the delay report.

You can generate the delay report in either batch mode or from the Analysis Report Generator dialog box.

Batch Generation

Following is an example of a batch command which would generate a delay report on a list of nets with comprehensive odd simulations using typical FTS mode:

```
signoise -f my_nets.txt -r Delay -n Odd -s Comprehensive -o delay_rpt1.txt my.brd
```

Table 8-7 Batch Command Switches - Delay Report

Switch	Description
-b	Location from which to measure results (model/pin/die)
-f	A list-of-nets file
-r	Report type to generate
-n	Neighbor switching mode for Comprehensive simulation
-s	Simulation type to perform (Reflection or Comprehensive)
-o	Name of output file to create

Interactive Generation

Selecting the *Delay* option in the *Analysis Report Generator* dialog box specifies a delay report for selected nets. See [Figure 8-23](#) on page 286.

Allegro PCB SI User Guide

Signal Integrity Analysis

Sample Delay Report

Note: Some report sections are split to fit the page.

```
#####
# Allegro PCB SI 15.5
# (c) Copyright 2004 Cadence Design Systems, Inc.
#
# Report: Standard Delay Report
# Thu Feb 10 14:39:17 2004
#####
```

```
*****
Delays (ns) (Typ FTSMode) Preferred Measurement Location: Pin
*****
```

XNet	Drvrv	Rcvr	PropDly	SwitchRise
-----	-----	-----	-----	-----
1 memory A4	memory J47 35	memory U20 29	0.7089	5.188
1 memory -CAS	memory J47 111	memory U9 17	0.6992	5.376
1 memory A3	memory J47 118	memory U12 26	0.6884	5.192
1 memory CLKE1	memory R4 2	memory U12 37	0.6876	3.189
1 memory CLKE1	memory R4 2	memory U20 37	0.6875	3.197
1 memory A2	memory J47 34	memory U12 25	0.6869	5.225
....				

```
*****
```

```
*****
```

SwitchFall	SettleRise	SettleFall
-----	-----	-----
2.415	7.847 *	2.749 *
2.404	7.89 *	2.8 *

Allegro PCB SI User Guide

Signal Integrity Analysis

2.353	7.786 *	2.681 *
2.293	5.881 *	2.649 *
2.308	5.88 *	2.641 *
2.379	7.86 *	2.716 *
....		

Sample Report (continued)

```
*****  
Monotonicity (Typ FTSMode)  
*****
```

XNet	Rcvr	FirstIncRise	FirstIncFall	MonotonicRise
-----	-----	-----	-----	-----
1 memory WP	memory U11 7	PASS	PASS	PASS
1 memory WP	memory U10 7	PASS	PASS	PASS
1 memory CLKE1	memory U20 37	PASS	PASS	PASS
1 memory CLKE1	memory U19 37	PASS	PASS	PASS
1 memory CLKE1	memory U12 37	PASS	PASS	PASS
1 memory CLKE1	memory U14 37	PASS	PASS	PASS
....				

```
*****
```

```
*****
```

MonotonicFall

PASS

Allegro PCB SI User Guide

Signal Integrity Analysis

PASS

....

Driver I/O Characteristics (Typ FTSMode) RiseSlew/FallSlew in (mV/ns)

Drvrv	IOModel	Volmax	Vohmin	RiseSlew	FallSlew
memory R6 1	CDSDefaultOutput	100 mV	4500 mV	3333	3333
memory C23 1	CDSDefaultOutput	100 mV	4500 mV	3333	3333
memory R3 1	CDSDefaultOutput	100 mV	4500 mV	3333	3333
memory RP5 2	CDSDefaultOutput	100 mV	4500 mV	3333	3333
memory U5 11	CDSDefaultOutput	100 mV	4500 mV	3333	3333
memory U5 47	CDSDefaultOutput	100 mV	4500 mV	3333	3333

....

Sample Report (continued)

Load I/O Characteristics:

Rcvr	IOModel	Vilmax	Vihmin
memory U11 7	CDSDefaultInput	2000 mV	3000 mV
memory U10 7	CDSDefaultInput	2000 mV	3000 mV
memory U20 37	CDSDefaultInput	2000 mV	3000 mV
memory U19 37	CDSDefaultInput	2000 mV	3000 mV
memory U12 37	CDSDefaultInput	2000 mV	3000 mV
memory U14 37	CDSDefaultInput	2000 mV	3000 mV

....

Allegro PCB SI User Guide

Signal Integrity Analysis

Pulse Data Per Xnet

XNet	PulseFreq	PulseDutyCycle	PulseCycleCount
1 memory WP	50MHz	0.5	1
1 memory UN4CAP226PA0	50MHz	0.5	1
1 memory UN4CAP225PA0	50MHz	0.5	1
1 memory SDA	50MHz	0.5	1
1 memory SCL	50MHz	0.5	1
1 memory SA2	50MHz	0.5	1
....			

Description of column abbreviations

Column	Description
DefImp	Default Impedance
DefPropVel	Default Propagation Velocity
DiffPairMate	Differential Pair Mate
Drvr	Driver Pin
FTSMode	Fast/Typical/Slow Mode
FallDly	Fall Buffer Delay
FallSlew	Fall Slew : 20%/80% dV/dT
FirstIncFall	First Incident Switch Fall
FirstIncRise	First Incident Switch Rise
GeomWin	Geometry Window
IOModel	I/O Cell Model Name

JTemp	Pin Junction Temperature
MhtPercent	Percent Manhattan Distance
PropDly	Propagation Delay
Rcvr	Receiver Pin
RiseDly	Rise Buffer Delay
RiseSlew	Rise Slew : 20%/80% dV/dT
SettleFall	Settle Fall Delay
SettleRise	Settle Rise Delay
SwitchFall	Switch Fall Delay
SwitchRise	Switch Rise Delay
Vihmin	High State Logic Input Threshold
Vilmax	Low State Logic Input Threshold
Vohmin	High State Logic Output Threshold
Volmax	Low State Logic Output Threshold
XNet	Extended Net

Report Computations

The Delay report computations are the same as those for the [Reflection Summary Report](#) on page 288. For further information, see [“Report Computations”](#) on page 295.

Ringing Report

The ringing report shows noise margin as well as overshoot high and low values for all selected nets. It also identifies IOCell characteristics that you have applied. This report requires either a Reflection or a Comprehensive simulation for each driver pin.

Use this report to detect impedance discontinuities that are significant due to the high slew rates of the drivers. Usually these problems are corrected by changing terminations, topology, or driver characteristics.

The ringing report also includes the Extended Net Distortion section containing the following:

- A subsection for each driver on the extended net and the noise margin and overshoot information for the driver-receiver pair.
- Pins on the extended net and buffer model information about the pins.

You can generate the ringing report either in batch mode or interactively from the *Analysis Report Generator* dialog box.

Batch Generation

Following is an example of a batch command which would generate a ringing report on a list of nets with Fast/Slow FTS mode:

```
signoise -f my_nets.txt -r Ringing -m Fast/Slow -o ring_rpt1.txt my.brd
```

Table 8-8 Batch Command Switches - Ringing Report

Switch	Description
-b	Location from which to measure results (model/pin/die)
-f	A list-of-nets file
-r	Report type to generate
-m	Mode to use while simulating for reports or waveforms
-o	Name of output file to create

Interactive Generation

Selecting the *Ringing* option in the Analysis Report Generator dialog box specifies a ringing report for selected nets. See [Figure 8-23](#) on page 286.

Sample Report

Note: Some report sections are split to fit the page.

```
#####
# Allegro PCB SI 15.5
# (c) Copyright 2004 Cadence Design Systems, Inc.
#
# Report: Standard Ringing Report Sorted By Worst Noise Margin
# Mon Feb 14 16:59:38 2004
#####

*****
Distortion (mV) (Typ FTSMode) Preferred Measurement Location: Pin
*****
```

Allegro PCB SI User Guide

Signal Integrity Analysis

XNet	Drvr	Rcvr	NMHigh	NMLow	OShootHigh
1 memory SDA	memory U10 5	memory U11 5	1960	1943	5007
1 memory DQM1	memory J47 29	memory U5 39	1936	1936	5012
1 memory -CS1	memory J47 114	memory U17 19	1934	1954	5003
1 memory -CS3	memory J47 129	memory U14 19	1932	1936	5013
1 memory SCL	memory J47 83	memory U10 6	1930	1931	5013
1 memory DQM1	memory J47 29	memory U3 39	1964	1928	5002
....					

OShootLow

-471
-595.8
-682
-695.6
-138.7
-626.7

....

Driver I/O Characteristics (Typ FTSMode) RiseSlew/FallSlew in (mV/ns)

Allegro PCB SI User Guide

Signal Integrity Analysis

Drvrv	Device	IOModel	Volmax
memory R6 1	RES_603-0,1A,603,E-603	CDSDefaultOutput	100 mV
memory C23 1	CAP_603-10PF,5%,50V,603,E-603	CDSDefaultOutput	100 mV
memory R3 1	RES_603-10,5%,603,E-603	CDSDefaultOutput	100 mV
memory RP5 2	RPAK4C-4R_SM-10,5%,A1206-SM	CDSDefaultOutput	100 mV
memory U5 11	SDRAM2MX4BX8_SSOP-SDRAM2MX4BX8A	CDSDefaultOutput	100 mV
memory U5 47	SDRAM2MX4BX8_SSOP-SDRAM2MX4BX8A	CDSDefaultOutput	100 mV
....			

Vohmin	RiseSlew	FallSlew	JTemp	DiffPairMate
4500 mV	3333	3333	NA	NA
4500 mV	3333	3333	NA	NA
4500 mV	3333	3333	NA	NA
4500 mV	3333	3333	NA	NA
4500 mV	3333	3333	NA	NA
4500 mV	3333	3333	NA	NA
....				

Load I/O Characteristics

Rcvr	Device	IOModel	Vilmax
....

Allegro PCB SI User Guide

Signal Integrity Analysis

memory U11 7	NM24C03_SSOP-UNKNOWN, E-SSOP	CDSDefaultInput	2000 mV
memory U10 7	EPROMSERIAL_SOI8-624664-301-SOA	CDSDefaultInput	2000 mV
memory U20 37	SDRAM2MX4BX8_SSOP-SDRAM2MX4BX8A	CDSDefaultInput	2000 mV
memory U19 37	SDRAM2MX4BX8_SSOP-SDRAM2MX4BX8A	CDSDefaultInput	2000 mV
memory U12 37	SDRAM2MX4BX8_SSOP-SDRAM2MX4BX8A	CDSDefaultInput	2000 mV
memory U14 37	SDRAM2MX4BX8_SSOP-SDRAM2MX4BX8A	CDSDefaultInput	2000 mV

....

Vihmin DiffPairMate

3000 mV	NA

....

Pulse Data Per Xnet

XNet	PulseFreq	PulseDutyCycle	PulseCycleCount
-----	-----	-----	-----
1 memory WP	50MHz	0.5	1
1 memory UN4CAP226PA0	50MHz	0.5	1
1 memory UN4CAP225PA0	50MHz	0.5	1
1 memory SDA	50MHz	0.5	1

Allegro PCB SI User Guide

Signal Integrity Analysis

1 memory SCL	50MHz	0.5	1
1 memory SA2	50MHz	0.5	1

....

Description of column abbreviations

Column	Description
DefImp	Default Impedence
DefPropVel	Default Propagation Velocity
DiffPairMate	Differential Pair Mate
Drvrv	Driver Pin
FTSMode	Fast/Typical/Slow Mode
FallSlew	Fall Slew : 20%/80% dV/dT
GeomWin	Geometry Window
IOModel	I/O Cell Model Name
JTemp	Pin Junction Temperature
MhtPercent	Percent Manhattan Distance
NMHigh	Noise Margin High
NMLow	Noise Margin Low
OShootHigh	Maximum Overshoot
OShootLow	Minimum Overshoot
Rcvr	Receiver Pin
RiseSlew	Rise Slew : 20%/80% dV/dT
Vihmin	High State Logic Input Threshold
Vilmmax	Low State Logic Input Threshold
Vohmin	High State Logic Output Threshold
Volmax	Low State Logic Output Threshold
XNet	Extended Net

Report Computations

The Ringing report computations are the same as those for the [Reflection Summary Report](#) on page 288. For further information, see “[Report Computations](#)” on page 295.

Single Net EMI Report

EMI simulation computes the differential mode radiated emission arising from clock signals propagating on all fully routed nets, taking one net (or Xnet) at a time. EMI Simulations simulate only the victim net and none of the neighboring aggressor nets. EMI simulation does not account for the parasitics of power and ground pins.

In EMI simulations, SigNoise performs the following tasks.

1. Traces out the extended net (Xnet).
2. Characterizes the interconnect cross sections.
3. Obtains the relevant device models.
4. Builds a single-line circuit (disregards neighbor nets).
5. Runs a Reflection simulation using a pulse stimulus one cycle in length.
6. Obtains and stores geometrical information for EMI computations (for example, interconnect coordinates and circuit board orientation).

The pulse stimulus is applied to the driver pin on the Xnet. In the case of multiple drivers on a net, multiple simulations are run with one active driver stimulated in each simulation. Other drivers on the Xnet are inactive during the simulation.

The transient simulation output is time domain voltage and current waveforms at a set of predetermined nodes. As a minimum, all driver and receiver pins on the Xnet are treated as nodes. You can include additional nodes at transmission line branch points for increased accuracy.

You can generate the single net EMI report either in batch mode or interactively from the Analysis Report Generator dialog box.

Batch Generation

Following is an example of a batch command which would generate a single net EMI report.

```
signoise -f nets.txt -r SingleNetEMISummary -o emi.txt my.brd
```

Allegro PCB SI User Guide

Signal Integrity Analysis

Table 8-9 Batch Command Switches - Single Net EMI Report

Switch	Description
-f	A list-of-nets file
-r	Report type to generate
-o	Name of output file to create

Interactive Generation

Selecting the *Single Net EMI* option in the Analysis Report Generator dialog box specifies a single net EMI report for selected nets. See [Figure 8-23](#) on page 286.

Sample Report

Note: Some report sections are split to fit the page.

```
#####
# Allegro PCB SI 15.5
# (c) Copyright 2004 Cadence Design Systems, Inc.
#
# Report: Single Net Emissions Report
#           Tue Feb 15 14:16:08 2004
#####
```

```
*****
*****
```

Voltage (V), Time (ns), Emission (dBuV/m), (Typ FTSMode)

```
*****
```

XNet	Drvrv	PulseFreq	VoltageSwing	RiseTime
-----	-----	-----	-----	-----
1 memory WP	memory R6 1	50MHz	4.4	0.9
1 memory UN4CAP226PA0	memory C23 1	50MHz	4.4	0.9
1 memory UN4CAP225PA0	memory R3 1	50MHz	4.4	0.9
1 memory SDA	memory U10 5	50MHz	4.4	0.6

Allegro PCB SI User Guide

Signal Integrity Analysis

1 memory SCL	memory J47 83	50MHz	4.4	0.9
1 memory SA2	memory J47 167	50MHz	4.4	0.9

....

PeakEmission	PeakFrequency	EMIStatus
-----	-----	-----
28.86	850Mhz	Pass
NA	NA	NA
NA	NA	NA
36.05	1250Mhz	Pass
34.19	950Mhz	Pass
32.69	950Mhz	Pass

....

Pulse Data Per Xnet

XNet	PulseFreq	PulseDutyCycle
-----	-----	-----
1 memory WP	50MHz	0.5
1 memory UN4CAP226PA0	50MHz	0.5
1 memory UN4CAP225PA0	50MHz	0.5
1 memory SDA	50MHz	0.5
1 memory SCL	50MHz	0.5
1 memory SA2	50MHz	0.5

....

Parasitics Report

The parasitics report shows total self capacitance, impedance range, and transmission line propagation delays for selected nets. The total net self capacitance includes capacitance from the transmission lines, via padstacks, pin padstacks, and IOCell die. Delay values are compared against delay constraints, if any exist. Information about the pins of the selected nets is included.

You can use the parasitics report to identify nets that are either overloaded or have excessive impedance discontinuities. The net parasitics report is a good choice for analyzing analog nets. This report does not use crosstalk estimations.

You can generate the parasitics report either in batch mode or interactively from the Analysis Report Generator dialog box.

Batch Generation

Following is an example of a batch command which would generate a Parasitics report.

```
signoise -f my_nets.txt -r Parasitics -o parasitics_rpt1.txt my.brd
```

Table 8-10 Batch Command Switches - Parasitics Report

Switch	Description
-f	A list-of-nets file
-r	Report type to generate
-o	Name of output file to create

Interactive Generation

Selecting the *Parasitics* option in the Analysis Report Generator dialog box specifies a parasitics report for selected nets. See [Figure 8-23](#) on page 286.

Sample Report

Note: This report has been split to fit the page.

```
#####
# Allegro PCB SI 15.5
# (c) Copyright 2004 Cadence Design Systems, Inc.
```

Allegro PCB SI User Guide

Signal Integrity Analysis

```
#  
# Report: Standard Parasitics Report  
# Thu Feb 17 10:08:16 2004  
#####
```

```
*****
```

XNet Parasitics

```
*****
```

XNet	MinImpedance	MaxImpedance	Capacitance	Inductance
1 memory WP	70.72	70.72	2.035e-12	NA
1 memory UN4CAP226PA0	70.72	70.72	4.607e-13	2.525e-09
1 memory UN4CAP226PA0	70.72	70.72	5.447e-13	2.946e-09
1 memory SDA	70.72	70.72	1.639e-1	NA
1 memory SCL	70.72	70.72	1.45e-12	NA
1 memory SA2	70.72	70.72	1.426e-12	NA
....				

```
*****
```

```
*****
```

Resistance

```
-----
```

NA

0.005502

0.008619

NA

NA

NA

```
....
```

SSN Report

The SSN report shows noise levels induced on the power and ground busses of a component when all drivers deriving power from that bus switch simultaneously. These noise levels are used as an approximation of the distortion effects that will be seen at the signal pins. The power bus noise is used as the basis for Rise distortion and ground bus noise is used for Fall distortion. The power and ground busses are identified in this report.

You are required to route power and ground nets to yield accurate results for the SSN report, although package parasitics are accounted for even without power and ground routing. Component placement adjustment, decoupling, power and ground net reassignment, and power/ground plane rearrangement are techniques used for solving SSN noise problems. It may be useful to use this report during both placement and routing phases, taking care to update the SimulSwitch simulations.

The SSN report also includes the Extended Net SSN section containing:

- A subsection for each driver pin on the extended net and the Rise SSN and Fall SSN information for the driver.
- Pins on the extended net and information about the pins.

You can generate the SSN report either in batch mode or interactively from the Analysis Report Generator dialog box.

Batch Generation

Following is an example of a batch command which would generate an SSN report.

```
signoise -f my_nets.txt -r SSN -o ssn_rpt1.txt my.brd
```

Table 8-11 Batch Command Switches - SSN Report

Switch	Description
-f	A list-of-nets file
-r	Report type to generate
-o	Name of the output file to create

Interactive Generation

Selection of the *SSN* option in the Analysis Report Generator dialog box specifies a simultaneous switching noise report for selected nets. See [Figure 8-23](#) on page 286.

Allegro PCB SI User Guide

Signal Integrity Analysis

Sample Report

Note: Some sections of this report have been split to fit the page.

```
#####
# Allegro PCB SI 15.5
# (c) Copyright 2004 Cadence Design Systems, Inc.
#
# Report: Standard SSN Report
#           Mon Mar 27 17:57:23 2004
#####
```

```
*****
Simultaneous Switching Noise (mV) for XNet `2 ssn NET1` (Typ FTSMode)
*****
```

Drvrv	Net	PowerBus	SSNRise	GroundBus	SSNFall
ssn U1 2	ssn NET1	pwrbus	505.5	gndbus	944.2

```
*****
Driver I/O Characteristics (Typ FTSMode) RiseSlew/FallSlew in (mV/ns)
*****
```

Drvrv	IOModel	Volmax	Vohmin	RiseSlew	FallSlew
ssn U1 2	CDSDefaultIO	100 mV	4500 mV	5000	5000

```
*****
Load I/O Characteristics
*****
```

Allegro PCB SI User Guide

Signal Integrity Analysis

Rcvr	IOModel	Vilmax	Vihmin
-----	-----	-----	-----
-----	-----	-----	-----

Pulse Data Per Xnet

XNet	PulseFreq	PulseDutyCycle	PulseCycleCount
-----	-----	-----	-----
2 ssn NET1	50MHz	0.5	1
-----	-----	-----	-----

Simulation Preferences

Variable	Value
-----	-----
Percent Manhattan	100
Default Impedance	60ohm
Default Prop Velocity	1.4142e+08M/s
Geometry Window	10mil
Min Neighbor Capacitance	0.1pF
Cutoff Frequency	0GHz
Pulse Clock Frequency	50MHz
Pulse Duty Cycle	0.5
Pulse Step Offset	0ns

Description of abbreviations

Allegro PCB SI User Guide

Signal Integrity Analysis

Abbr	Abbreviations
DefImp	Default Impedence
DefPropVel	Default Propagation Velocity
Drvrr	Driver Pin
FTSMode	Fast/Typical/Slow Mode
FallSlew	Fall Slew : 20%/80% dV/dT
GeomWin	Geometry Window
IOModel	I/O Cell Model Name
MhtPercent	Percent Manhattan Distance
Rcvr	Receiver Pin
RiseSlew	Rise Slew : 20%/80% dV/dT
SSNFall	Simultaneous Switching Noise on Ground Bus
SSNRise	Simultaneous Switching Noise on Power Bus
Vihmin	High State Logic Input Threshold
Vilmax	Low State Logic Input Threshold
Vohmin	High State Logic Output Threshold
Volmax	Low State Logic Output Threshold
XNet	Extended Net

Net name syntax

Net: <design name> <net name>
XNet: <number of nets> <design name> <first net name>

Allegro PCB SI User Guide

Signal Integrity Analysis

Pulse Data Per Xnet

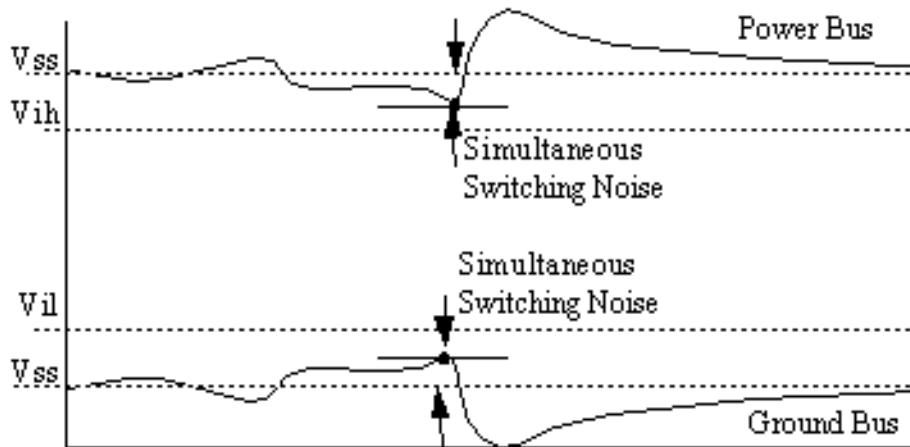
XNet	PulseFreq	PulseDutyCycle	PulseCycleCount
1 memory WP	50MHz	0.5	1
1 memory UN4CAP226PA0	50MHz	0.5	1
1 memory UN4CAP225PA0	50MHz	0.5	1
1 memory SDA	50MHz	0.5	1
1 memory SCL	50MHz	0.5	1
1 memory SA2	50MHz	0.5	1
....			

Report Computations

For the high state, the magnitude of the largest negative excursion of the power bus voltage is measured. All driver pins are simultaneously switched for the rising edge and waveforms are generated at the die for the driver device's power bus. The rising edge simultaneous switching noise is taken as the magnitude of difference between the steady state voltage of the power bus minus the lowest excursion of the power bus waveform.

For the low state, the magnitude of the largest positive excursion of the ground bus voltage is measured. All driver pins are simultaneously switched for the falling edge and waveforms are generated at the die for the driver device's ground bus. The falling edge simultaneous switching noise is taken as the magnitude of difference between the highest excursion of the ground bus waveform minus the steady state voltage of the ground bus.

Figure 8-40 Simultaneous Switching Noise Measurement Points



Electrical Constraints

You can set the following constraints on a net or on a pin-to-pin connection for evaluation during delay analysis.

Table 8-12 Measured Delay Value Comparisons

Constraint	Definition	Format
MAX_SS	Limits noise due to simultaneous switching outputs.	maximum value:minimum value

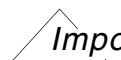
Segment Crosstalk Report

The segment crosstalk report presents detailed *segment-based coupling* information derived from post-route verification. Included in the report is a segment-by-segment listing of coupled lengths, indication of layers involved, along with x-y coordinates of where coupling occurs, and estimated crosstalk for coupled segments. The segment crosstalk report also includes the information on crosstalk received from each single aggressor neighbor Xnet and total crosstalk received from all aggressor neighbor nets.

This report is actually a super set of both crosstalk and parallelism reports. This means that segments within the geometry window that are parallel, but have no crosstalk effects on each other, will still appear in the table with crosstalk values set to zero. Also, when there are no

crosstalk values available, only the parallelism data is shown in the report with crosstalk values set to *NA*. The report data is supplied in a delimited text format suitable for spreadsheet applications.

Segment-based crosstalk estimation is intended to support an *interactive* crosstalk debugging or etch editing use model, where rapid results and quick identification of worst offenders are required. Rather than performing costly coupled-line time domain simulation in real time, this technique performs the time domain simulation up front to sweep representative crosstalk circuits for the design, generating tables of crosstalk data. In addition to being available in the report, net to net results are flagged by Design Rule Checks (DRCs) in PCB SI and PCB Editor, and enforced by PCB Router.

 *Important*

Segment-based crosstalk estimation does not take into account detailed topological effects like reflections, wave cancellation, or the plethora of unique stimulus combinations that can occur in actual designs. To analyze these effects, you should utilize full time domain crosstalk simulation.

You can run the segment crosstalk report either in batch mode or interactively from the Analysis Report Generator dialog box.

Batch Generation

Following is an example of a batch command which would generate a segment crosstalk report:

```
signoise -f nets.txt -g 100 -l 500 -r SegmentXtalk -a Each -o test_xtalk.rpt  
test.brd
```

Table 8-13 Batch Command Switches - Segment Crosstalk Report

Switch	Description
-f	A list-of-nets file
-g	Geometry window size
-l	Minimum coupled length to consider
-r	Report type to generate
-a	Which aggressors to include in the crosstalk estimations (You should specify each to generate a segment report)
-o	Name of the output file to create

Interactive Generation

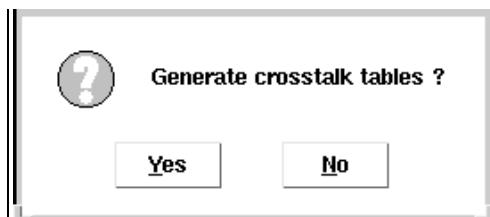
Selecting the *Segment Crosstalk* option in the Analysis Report Generator dialog box specifies a Segment Crosstalk report for selected nets. See [Figure 8-23](#) on page 286.

Note: To generate an appropriate report, be sure to select the *Each Neighbor* option (for Net Selection) in the Aggressor section of the dialog box.

Important

To obtain segment-based crosstalk results, crosstalk tables must be available in your board file. If not, the dialog box shown below will appear after clicking the *Create Report* button asking if you would like to have them generated. If you select *No*, your report will include parallelism data only with all crosstalk values set to *NA*.

Figure 8-41 Crosstalk Tables Dialog Box.



Sample Report

```
#####
# Allegro PCB SI 15.5
```

Allegro PCB SI User Guide

Signal Integrity Analysis

```
# (c) Copyright 2004 Cadence Design Systems, Inc.  
#  
# Report: Standard Segment Crosstalk Report  
# Thu Feb 03 14:07:53 2004  
#####
```

```
*****  
Segment Crosstalk (mV)  
*****
```

Victim	Aggressor	Layer:Layer	XYCoord	Gap	Length	SegXtalk
2 dimm128 DQ47	all_neighbor_xnets	NA	NA	NA	NA	18.1
2 dimm128 DQ47	dimm128 DQR45	NA	NA	NA	729	12.69
2 dimm128 DQ47	dimm128 DQR45	BOTTOM:BOTTOM	1672:243	10	233	7.823
2 dimm128 DQ47	dimm128 DQR45	BOTTOM:BOTTOM	1842:290	10	116	3.895
2 dimm128 DQ47	dimm128 DQR45	BOTTOM:BOTTOM	1544:232	13	32	0.8844
2 dimm128 DQ47	dimm128 DQR45	BOTTOM:BOTTOM	2006:473	50	267	0.8566
2 dimm128 DQ47	dimm128 DQR45	BOTTOM:BOTTOM	1895:353	18	19	0.3371
2 dimm128 DQ47	dimm128 DQR45	BOTTOM:BOTTOM	1903:370	23	23	0.2855
2 dimm128 DQ47	dimm128 DQR45	BOTTOM:BOTTOM	1532:207	25	18	0.207
2 dimm128 DQ47	dimm128 DQR45	BOTTOM:BOTTOM	1889:337	38	17	0.09458
2 dimm128 DQ47	dimm128 DQR45	BOTTOM:BOTTOM	1532:218	16	4	0.0868
2 dimm128 DQ47	dimm128 DQR43	NA	NA	NA	448	11.27
2 dimm128 DQ47	dimm128 DQR43	BOTTOM:BOTTOM	1663:275	10	237	7.958
2 dimm128 DQ47	dimm128 DQR43	BOTTOM:BOTTOM	1813:307	10	88	2.955
2 dimm128 DQ47	dimm128 DQR43	BOTTOM:BOTTOM	1844:380	21	76	1.013
2 dimm128 DQ47	dimm128 DQR43	BOTTOM:BOTTOM	1523:275	32	43	0.357
2 dimm128 DQ47	dimm128 DQR43	BOTTOM:BOTTOM	1844:340	36	4	0.02591
2 dimm128 DQ47	dimm128 DQR42	NA	NA	NA	421	3.919
2 dimm128 DQ47	dimm128 DQR42	BOTTOM:BOTTOM	1659:291	26	230	2.54
2 dimm128 DQ47	dimm128 DQR42	BOTTOM:BOTTOM	1801:318	26	74	0.8171
2 dimm128 DQ47	dimm128 DQR42	BOTTOM:BOTTOM	1827:381	38	74	0.4117

Allegro PCB SI User Guide

Signal Integrity Analysis

2 dimm128 DQ47	dimm128 DQR42	BOTTOM:BOTTOM	1523:291	48	43	0.1504
2 dimm128 DQ47	dimm128 DQR46	NA	NA	NA	367	3.876

....

Report Format

Column	Data Description
Victim	The net receiving the coupling from the neighbor nets.
Aggressor	The net contributing the coupling to the victim net.
Layer:Layer	Specifies for this coupled segment, the layer the victim and aggressor nets are routed on respectively. For example, if the coupled segments of both nets are routed on the TOP layer, it would read TOP:TOP. If the coupled segment had the victim net on INT1 and the neighbor net on INT2, then this column would read INT1:INT2. Layer to layer information is only applicable to coupled segments.
XYCoord	Specifies the xy coordinate for the <i>middle</i> of the aggressor segment coupling, enabling crossprobing, troubleshooting and debugging. The xy coordinate is available only for segment-to-segment couplings. It is not applicable to net-to-net coupling in the table.
Gap	Specifies the distance between the segments on the victim and neighbor nets. For cases where the coupling occurs on the same layer, this is simply the spacing between the traces. For layer-to-layer coupling cases, this gap represents a combination of the dielectric spacing between the layers and the offset between the victim and neighbor (Pythagorean Theorem). The gap only applicable to coupled segments.
Length	The length of the coupled segment. Note that the table gives a <i>segment-by-segment</i> breakdown, and also an overall <i>net-to-net</i> value, and an overall <i>all-neighbors-to-victim-net</i> value.
SegXtalk	Amount of voltage coupled over from the aggressor to the victim in that particular coupled segment. Note: The initial value in this column represents the total crosstalk received from <i>all</i> neighbors. The next set of values consists of a single neighbor crosstalk total followed by a <i>segment-by-segment</i> crosstalk breakdown for the same neighbor. This pattern is then repeated for each neighbor listed in the report.

Report Computations

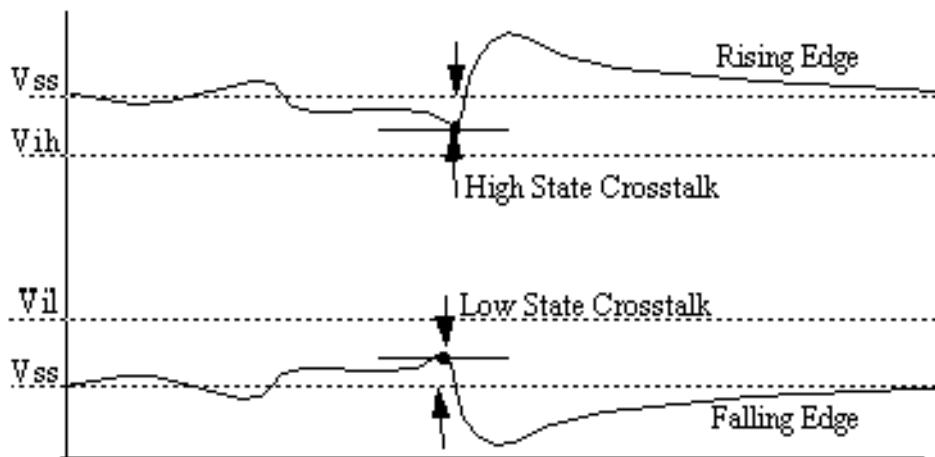
The all-neighbors-to-victim-net value is calculated based on the root-sum-squared (RSS) summation method.

Crosstalk is the magnitude of the voltage change seen at a pin on a victim net where the voltage change is induced by signals on coupled neighboring aggressor Xnets when drivers on aggressor nets are switching simultaneously.

The crosstalk measurement for a victim net held in the high state is taken as the magnitude of the difference between the lowest excursion of the receiver waveform minus the victim net's steady state voltage. The crosstalk measurement for a victim net held in the low state is measured as the magnitude of the difference between the highest excursion of the receiver waveform minus the victim net's steady state voltage.

Crosstalk measurements for victim nets in both the high and low states are shown in the following figure.

Figure 8-42 Crosstalk Measurement Points



Crosstalk Summary Report

The crosstalk summary report delivers an abbreviated crosstalk report, identifying only the selected victim Xnet and driver, and reporting on high and low state crosstalk for odd and even stimulus.

You can generate the crosstalk summary report either in batch mode or interactively from the Analysis Report Generator dialog box.

Batch Generation

Following is an example of a batch command which would generate a crosstalk summary report with *All Neighbor* crosstalk simulations on the entire board using fast mode.

```
signoise -f my_nets.txt -a All -n Odd,Even -r XtalkSummary -m Fast  
-o xtlksum_rpt1.txt my.brd
```

Table 8-14 Batch Command Switches - Crosstalk Summary Report

Switch	Description
-f	A list-of-nets file
-a	Aggressors specified for crosstalk simulations Choices are: Each, All, Inter, or Intra
-n	Neighbor switching mode specified for crosstalk and comprehensive simulations. Choices are: Even, Odd or Odd, Even
-r	Report type to generate
-m	Mode to use while simulating for reports or waveforms
-o	Name of output file to create

Interactive Generation

Selecting the *Crosstalk Summary* option in the Analysis Report Generator dialog box specifies a Crosstalk Summary report for selected nets. See [Figure 8-23](#) on page 286.

Sample Report

Note: This report has been split to fit the page.

```
#####
# Allegro PCB SI 15.5
# (c) Copyright 2004 Cadence Design Systems, Inc.
#
# Report: Standard Crosstalk Summary Report Sorted By Worst Case Crosstalk
#           Thu Feb 17 12:02:03 2004
#####
```

Allegro PCB SI User Guide

Signal Integrity Analysis

All Neighbors Crosstalk (mV) (Typ FTSMode)

Victim XNet	Victim Drvr	HSOddXtalk	HSEvenXtalk	LSOddXtalk
-----	-----	-----	-----	-----
1 memory A7	memory J47 120	401.5	NA	214.6
1 memory A0	memory J47 33	387.9	NA	229.8
1 memory A3	memory J47 118	384.2	NA	207.3
1 memory A4	memory J47 35	351.4	NA	200.5
1 memory A9	memory J47 121	306.4	NA	168.7
1 memory A9	memory J47 39	297.8	NA	159.5
....				

LSEvenXtalk

NA
....

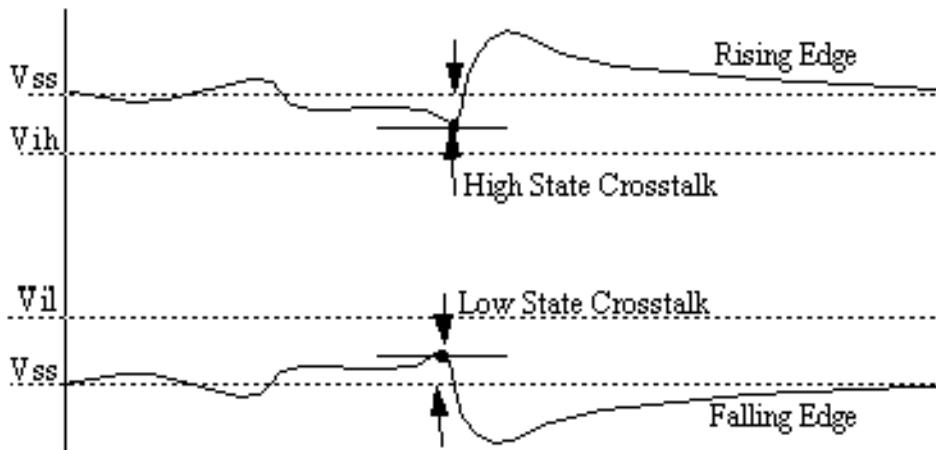
Report Computations

Crosstalk is the magnitude of the voltage change seen at a pin on a victim net where the voltage change is induced by signals on coupled neighboring aggressor Xnets when drivers on aggressor nets are switching simultaneously.

The crosstalk measurement for a victim net held in the high state is taken as the magnitude of the difference between the lowest excursion of the receiver waveform minus the victim net's steady state voltage. The crosstalk measurement for a victim net held in the low state is measured as the magnitude of the difference between the highest excursion of the receiver waveform minus the victim net's steady state voltage.

Crosstalk measurements for victim nets in both the high and low states are shown in the following figure.

Figure 8-43 Crosstalk Measurement Points



Crosstalk Detailed Report

The crosstalk detailed report identifies the selected victim Xnet, drivers, and all receivers, and reports on high and low state crosstalk for odd and even stimulus. It also provides information on the devices and models used, their electrical characteristics, the default simulation settings, and a full glossary of abbreviations.

The crosstalk detailed report can be generated either in batch mode or interactively from the Analysis Report Generator dialog box.

Batch Generation

Following is an example of a batch command which would generate a crosstalk detailed report with *All Neighbor* crosstalk simulations on the entire board using fast mode.

```
signoise -f my_nets.txt -a Each -D All -n Odd,Even -r XtalkDetailed  
-m Fast -o xtlkdet_rpt1.txt my.brd
```

Table 8-15 Batch Command Switches - Crosstalk Detailed Report

Switch	Description
-f	A list of nets file.
-a	Aggressors specified for crosstalk simulations. Choices are: Each, All, Inter, or Intra.
-D	Use either the fastest driver or the all drivers on the neighboring aggressor nets. Choices are: Fastest or All.
-n	Neighbor switching mode specified for crosstalk and comprehensive simulations. Choices are: Even, Odd or Odd, Even.
-r	Report type to be generated.
-m	Mode to be used while simulating for reports or waveforms.
-o	Name of output file to be created.

Interactive Generation

Selection of the *Crosstalk Detailed* option in the Analysis Report Generator dialog box specifies a Crosstalk Detailed report for selected nets. See [Figure 8-23](#) on page 286.

Sample Report

Note: Some sections of this report have been split to fit the page.

```
#####
# Allegro PCB SI 15.5
# (c) Copyright 2004 Cadence Design Systems, Inc.
#
# Report: Standard Crosstalk Report Sorted By Worst Case Crosstalk
#           Thu Feb 17 16:42:21 2004
```

Allegro PCB SI User Guide

Signal Integrity Analysis

All Neighbors Crosstalk at Receivers (mV) (Typ FTSMode)

Victim XNet	Victim Drvr	Victim Rcvr	HSOddXtalk	HSEvenXtalk
1 memory A7	memory J47 120	memory U17 32	401.5	NA
1 memory A7	memory J47 120	memory U4 32	391	NA
1 memory A0	memory J47 33	memory U12 23	387.9	NA
1 memory A3	memory J47 118	memory U12 26	384.2	NA
1 memory A0	memory J47 33	memory U17 23	380.2	NA
1 memory A3	memory J47 118	memory U9 26	376.4	NA

LSDoddXtalk	LSEvenXtalk
214.6	NA
209.3	NA
229.8	NA
207.3	NA
210.1	NA
202	NA

— ■ ■ ■ —

* * * * *

Pulse Data Per Xnet

Allegro PCB SI User Guide

Signal Integrity Analysis

XNet	PulseFreq	PulseDutyCycle	PulseCycleCount
1 memory WP	50MHz	0.5	1
1 memory UN4CAP226PA0	50MHz	0.5	1
1 memory UN4CAP225PA0	50MHz	0.5	1
1 memory SDA	50MHz	0.5	1
1 memory SCL	50MHz	0.5	1
1 memory SA2	50MHz	0.5	1
....			

Description of abbreviations

Column	Description
HSEvenXtalk	Crosstalk, Rise Stimulus to Neighbors and Victim held High
HSOddXtalk	Crosstalk, Fall Stimulus to Neighbors and Victim held High
LSEvenXtalk	Crosstalk, Fall Stimulus to Neighbors and Victim held Low
LSOddXtalk	Rise Stimulus to Neighbors and Victim held Low
Victim Drvr	Victim Driver Pin
Victim Rcvr	Victim Receiver Pin
Victim XNet	Victim Extended Net

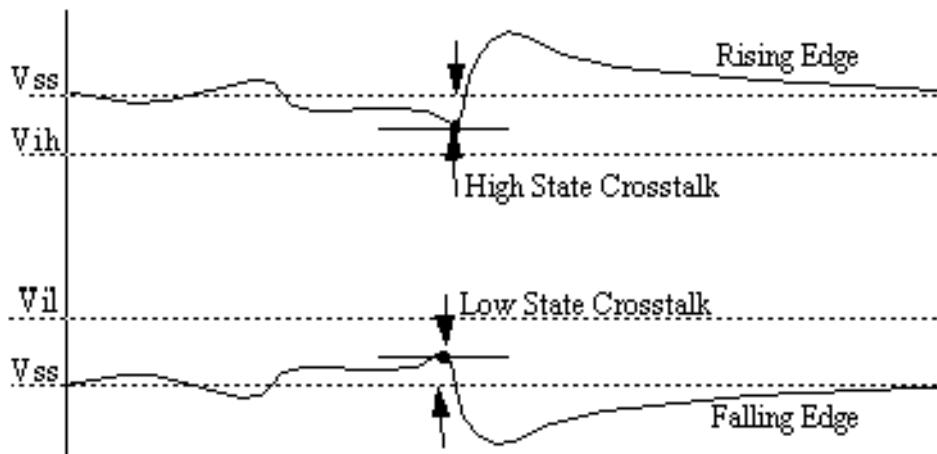
Report Computations

Crosstalk is the magnitude of the voltage change seen at a pin on a victim net where the voltage change is induced by signals on coupled neighboring aggressor Xnets when drivers on aggressor nets are switching.

The crosstalk measurement for a victim net held in the high state is taken as the magnitude of the difference between the lowest excursion of the receiver waveform minus the victim net's steady state voltage. The crosstalk measurement for a victim net held in the high state is taken as the magnitude of the difference between the victim net's steady state voltage minus the lowest excursion of the receiver waveform.

Crosstalk measurements for victim nets in both the high and low states are shown in the following figure.

Figure 8-44 Crosstalk Measurement Points



Electrical Constraints

You can set the following constraints on a net or Xnet for evaluation during crosstalk analysis.

Table 8-16 Measures Delay Value Comparisons

Constraint	Definition	Format
MAX_XTALK	Limits the total crosstalk on a victim net from all aggressor nets.	maximum value:minimum value
MAX_PEAK_XTALK	Limits the crosstalk on a victim net from a single aggressor net.	maximum value:minimum value

Signal Quality Screening

Signals are subject to degradation when they are transmitted through a channel. High speed signals are especially susceptible to degradation as loss and distortions tend to be frequency-dependent. Channel characterization and multi-million bit simulations can be used to investigate these issues in greater detail. However, such investigations are time consuming for PCBs with several high speed nets.

What is Signal Quality Screening

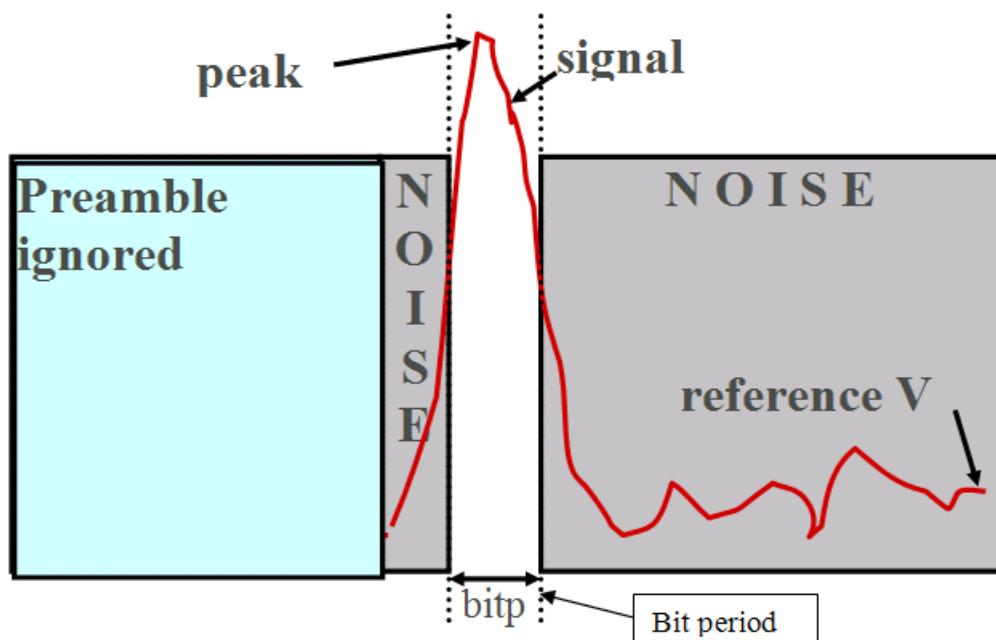
Signal Quality Screening is used to determine signal quality of a system. It determines the signal and background noise intervals introduced by the system or channel. Extensive simulation is most beneficial when applied to the noisiest channel(s) in a given group. Signal quality screening determines the noisiest channel that needs further investigation. This focused analysis results in improved designs in a shorter time.

You can perform signal quality screening on a set of nets which could either be single-ended or part of a differential pair.

How Signal Quality Screening Works

The signal quality screening engine uses the frequency domain simulation capability of TLSim. A clean bit is sent to a channel to be analyzed. Its response is measured at the end of the channel, or input of a receiver. The response spreads in a much wider time interval than the original bit period. In [Figure 8-45](#), a window is placed on the time interval, in which the white portion represents signal while the gray portion represents noise.

Figure 8-45 Signal Quality Screening Process



The energy in the signal portion and the noise portions is then quantified by the areas under the waveform curve. The energy is calculated by root-square summations of the difference between the signal voltages and the reference voltage at every sampling point, or time step. The reference voltage is taken to be the final value at the end of the simulation when the signal is finally settled.

Calculating the Signal-to-Noise Ratio Value

The Signal-to-Noise Ratio (SNR) value, which determines signal quality of a system, is calculated by taking the ratio of the energy of the signal and the rest of the waveform. Signal energy is derived by taking the sum of the square of voltage differences between the voltage instance and the reference voltage.

$$SNR_Value = \frac{Signal_Energy}{Total_Energy * Signal_Energy}$$

For detailed information on the signal quality screening procedure, see [Performing Signal Quality Screening](#).

Analyzing to Generate Waveforms

When signal integrity problems are detected in a critical net, waveforms are often used to diagnose the root causes. SigWave is an oscilloscope emulator that you can use to display the waveforms resulting from your simulations.

When you perform analysis for signal integrity or EMI emissions by creating and viewing waveforms, SigNoise performs the necessary simulations based on specifications you make in the Signal Analysis and Analysis Waveform Generator dialog boxes.

Having both dialog boxes open together, you can move back and forth between them, selecting nets and pins for analysis in the Signal Analysis dialog box and specifying simulation details in the Analysis Waveform Generator dialog box.

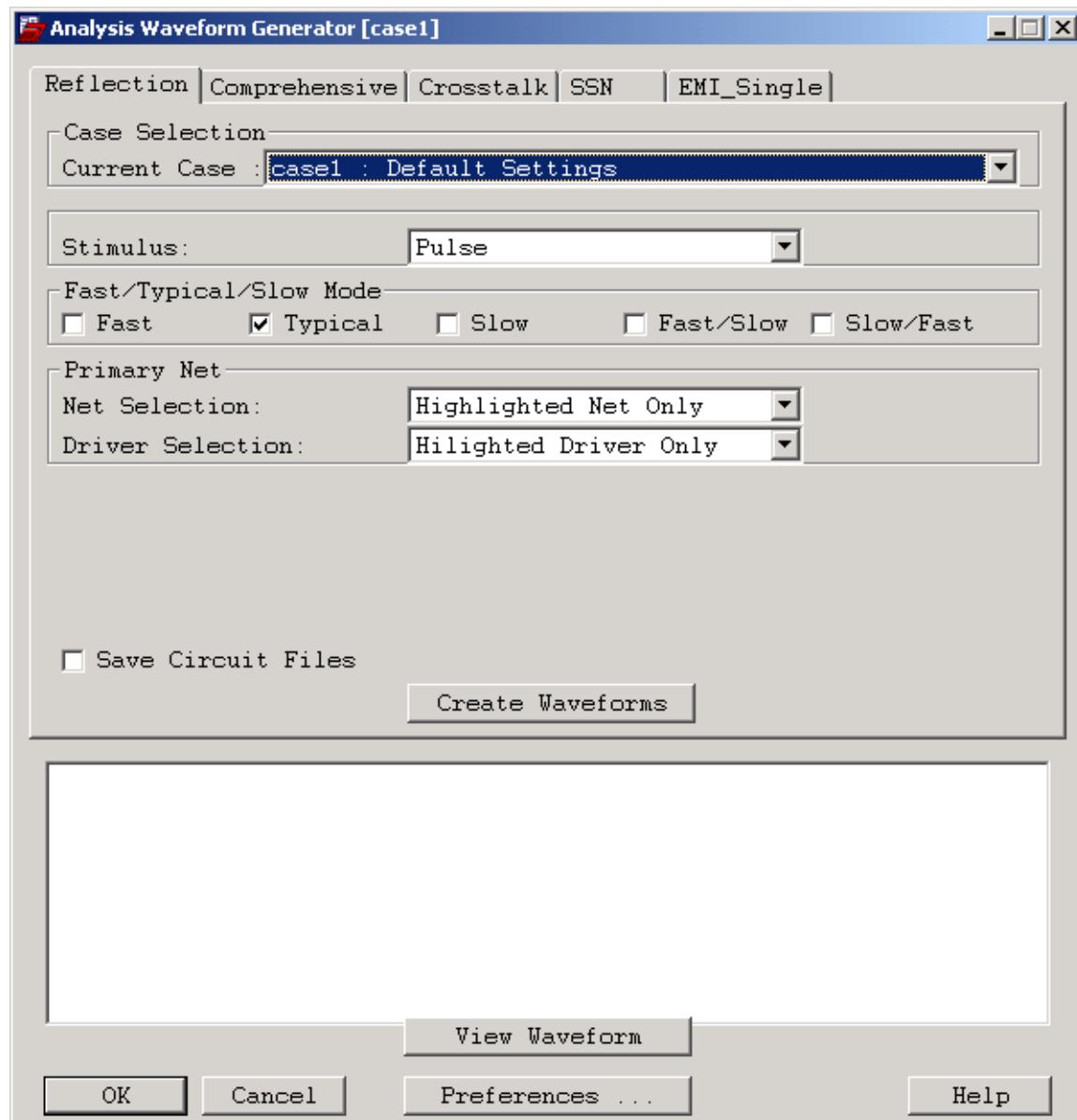
After examining the resulting waveforms in the SigWave window, you can refine your net and pin selections and simulation details, change simulation preferences (if necessary), and perform more specific analysis to pinpoint problem signals for a more refined analysis.

To begin waveform based analysis

- Click *Waveforms* in the Signal Analysis dialog box. See [Figure 8-15](#) on page 263.

The Analysis Waveform Generator dialog box is displayed, as shown in [Figure 8-46](#) on page 344.

Figure 8-46 Analysis Waveform Generator Dialog Box



In the Analysis Waveform Generator dialog box you can:

- select a simulation tab to perform one of five types of simulations: Reflection, Comprehensive, Crosstalk, SSN, or EMI Single.
- specify simulation details such as:
 - select a simulation case.

- ❑ select the stimulus (choices vary between simulation types).
- ❑ select the Fast/Typical/Slow Simulation speed mode.
- ❑ select the Victim nets and drivers.
- ❑ select the Aggressor net switching mode (for Crosstalk and Comprehensive simulations) and Aggressor nets and drivers (for Crosstalk simulations).
- ❑ select whether or not to save circuit files or use timing windows (for Crosstalk simulations).
- create the waveforms.
- open the Analysis Preferences dialog box to modify simulation preferences after reviewing waveforms to further refine the waveforms resulting from subsequent simulations.
- view selected waveforms.

Specifying the Simulation Type

Select one of the five tabs in the Analysis Waveform Generator dialog box to determine the type of simulation performed to generate the waveforms.

- Use the *Reflection* tab to perform reflection simulations.
- Use the *Comprehensive* tab to perform comprehensive simulations.
- Use the *Crosstalk* tab to perform crosstalk simulations.
- Use the *SSN* tab to perform simultaneous switching noise simulations.
- Use the *EMI_Single* tab to perform EMI single net simulations.

Reflection Simulations

Reflection simulations simulate only the victim net and none of the neighboring aggressor nets. Reflection simulation does not take the parasitics of power and ground pins into account.

Comprehensive Simulations

Comprehensive simulations simulate the specified victim net and its neighboring aggressor nets at the same time. In Comprehensive simulation, SigNoise applies the stimulus type you select to the victim net and either the same or the opposite stimulus to the neighboring aggressor nets depending on the switch mode you specify. Comprehensive simulation takes power and ground parasitics into account. It also shows glitches in the victim net that are produced by activity on the aggressor nets.

Crosstalk Simulations

Crosstalk simulations simulate one or more specified victim nets and one or more neighboring aggressor nets at the same time.

Specifying *All/Group Neighbors* for aggressor nets shows how activity on the specified aggressor nets can cause crosstalk on the victim nets. With a stimulus type of *Rise* or *Pulse*, SigNoise holds the victim nets high and applies a *Fall* or *Inverted Pulse* stimulus to the neighboring aggressor nets. With a stimulus type of *Fall* or *Inverted Pulse*, SigNoise holds the victim nets low and applies a *Rise* or *Pulse* stimulus to the neighboring aggressor nets.

Specifying *Each Neighbor* isolates crosstalk contributions from individual neighboring aggressor nets. In each neighbor crosstalk analysis, SigNoise runs multiple simulations where in each simulation, a single aggressor net is active while the other neighboring nets

are passive (held in the same state as the victim net). With a stimulus type of *Rise* or *Pulse*, SigNoise holds the victim net high and applies a *Fall* or *Inverted Pulse* stimulus to the neighboring aggressor net. With a stimulus type of *Fall* or *Inverted Pulse*, SigNoise holds the victim net low and applies a *Rise* or *Pulse* stimulus to the neighboring aggressor nets.

SSN Simulations

SSN simulations examine what happens when all of the drivers on a device that use the same power and ground bus as the driver in the selected victim nets trigger simultaneously, causing power and ground bounce. SigNoise monitors the ripple at the internal power and ground buses and takes them into account in the analysis of the extended net (or Xnet).

EMI Single Simulations

EMI single simulations perform a reflection simulation for a single net to evaluate the differential mode radiated emissions for the net.

Common Tab Areas

The following tables describe the common sections and buttons for all tabs.

Table 8-17 Common Tab Sections

Section	Function
<i>Current Case</i>	The name of the current case including a pulldown menu of available cases from which you can select to report on.
<i>Stimulus</i>	The current stimulus type including a pulldown menu of available stimulus choices. Choices are <i>Pulse</i> , <i>Rise</i> , <i>Fall</i> , <i>Rise/Fall</i> , <i>InvertedPulse</i> .
<i>Fast/Typical/Slow Mode</i>	Check boxes to specify one or more speeds at which SigNoise will run simulations. <i>Fast/Slow</i> and <i>Slow/Fast</i> refers to speeds of driver/receiver combinations.
<i>Primary Net</i>	Pull down menus for selecting the victim nets to monitor during simulation and selecting victim net driver to stimulate when victim nets are held in the high state.
<i>Save Circuit Files</i>	A check box to indicate if SigNoise will save resulting TLsim and SPICE circuit files in the case directory for each simulation performed.

Table 8-18 Common Tab Buttons

Button	Function
<i>Create Waveforms</i>	Starts waveform generation using the selections you specified in both the Signal Analysis and Analysis Waveform Generator dialog boxes.
<i>View Waveform</i>	Opens the SigWave window and loads the waveform file selected from the list box.
<i>Preferences</i>	Displays the Analysis Preferences dialog box enabling you to modify simulation preferences as you specify simulations and generate waveforms.

Optional Tab Sections

The following table describes the optional sections found only on the *Comprehensive* and *Crosstalk* tabs.

Table 8-19 Optional Tab Sections

Section	Function
<i>Aggressor</i>	Pull down menus for selecting switch mode, net selection, and driver selection.
<i>Use Timing Windows</i>	A check box to specify that SigNoise use timing window properties to refine the crosstalk simulations to account for received crosstalk that is insignificant due to the timing of signals.

Optional Tab Selection

The following table describes the optional stimulus selection found only on the *Comprehensive* tab.

Table 8-20 Optional Tab Sections

Section	Function
<i>Stimulus</i>	The current stimulus type includes <i>Custom</i> , to apply and/or edit custom stimuli to nets and xnets. Selection of <i>Custom</i> activates the <i>Assign</i> button which opens the Stimulus Setup dialog box. From there, you can set custom stimulus parameters for nets/ xnets. This functionality is described at length in the <u>signal probe</u> Help topic in the <i>Allegro PCB and Package Physical Layout Command Reference</i> .

Displaying and Interpreting Waveforms

SigWave is an interactive waveform viewer. Its primary purpose is to display response waveforms from circuit simulations. SigWave is also used by other applications that require waveform display. For example, using SigWave you can display IBIS VI and VT transfer curves.

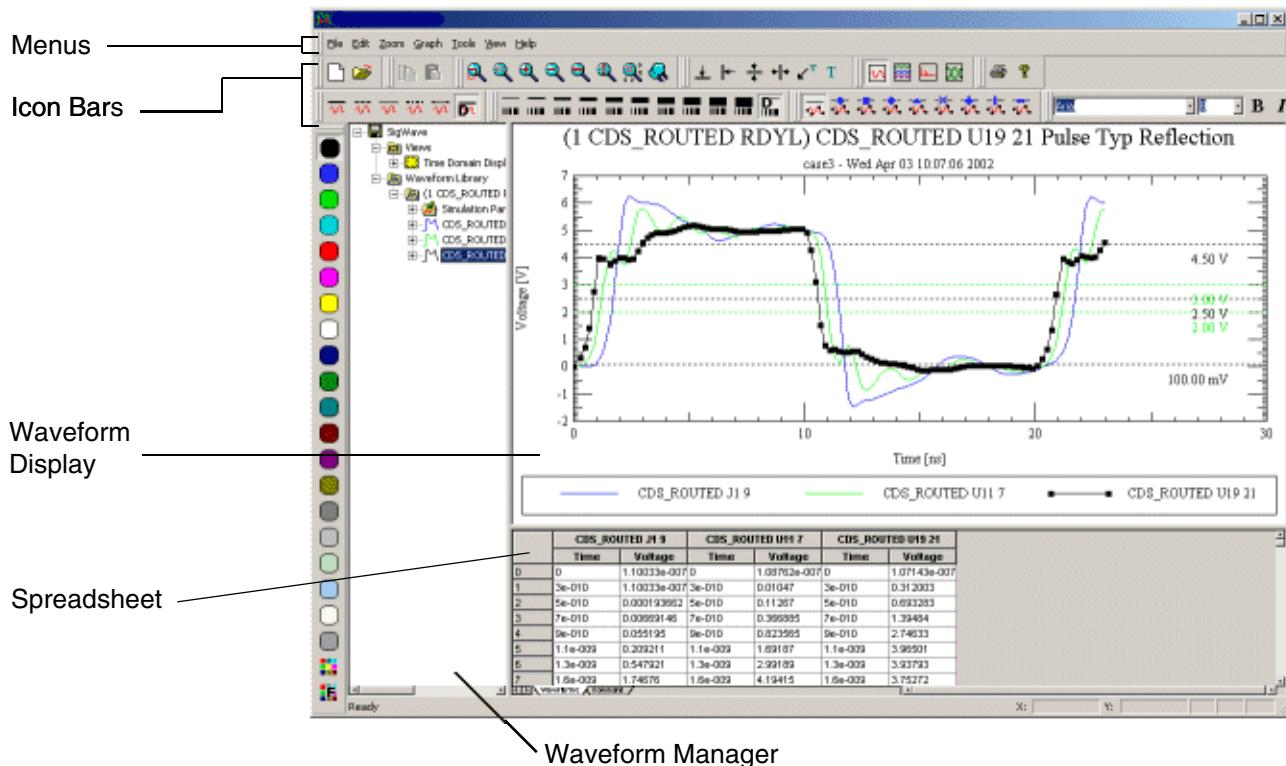
To display a waveform in the SigWave window

1. Select a waveform file from the list box in the Analysis Waveform Generator dialog box.
2. Click *View Waveform*. The SigWave window appears displaying the selected waveform as shown in the following figure.

Allegro PCB SI User Guide

Signal Integrity Analysis

Figure 8-47 The SigWave Window



Refer to the [SigWave User Guide](#) for further details on displaying and interpreting waveforms.

Conductor Cross Sections

The Conductor Cross Section window (sigxsect) allows you to view the geometry of Interconnect Models and the equipotential field lines between the cross sections of interconnect.

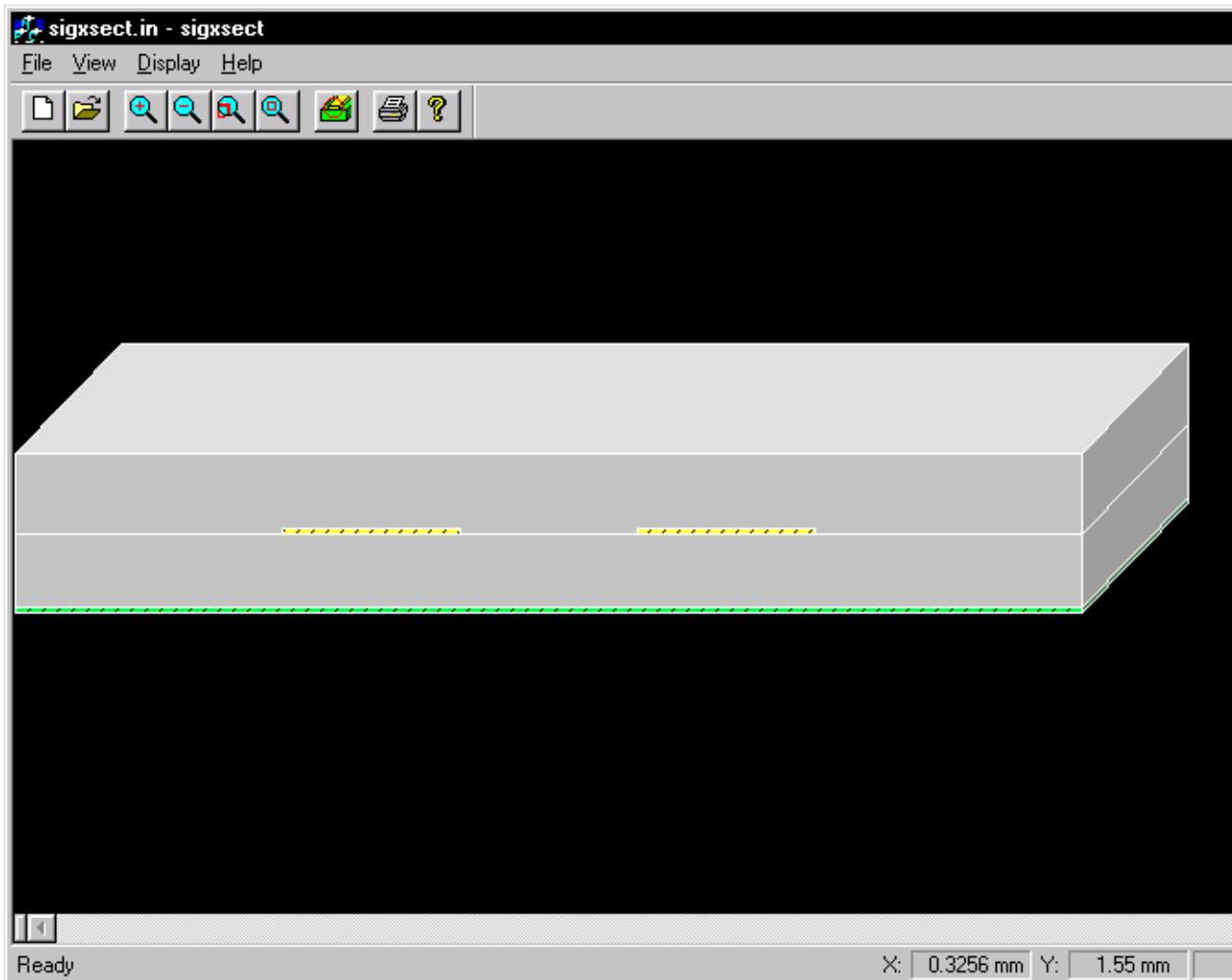
To view interconnect cross section geometry

1. Select a net from the list box in the Signal Analysis dialog box.
2. Click *View Geometry* in the Signal Analysis dialog box. See [Figure 8-15](#) on page 263.

The sigxsect window appears displaying the conductor cross-section as shown in [Figure 8-48](#) on page 351.

Allegro PCB SI User Guide
Signal Integrity Analysis

Figure 8-48 The sigxsect Window



Allegro PCB SI User Guide
Signal Integrity Analysis

Analyzing for Static IR-Drop

Increasingly complex circuit designs call for greater power consumption while requiring ever-greater reductions of overall power supply voltages. Resulting voltage fluctuations can translate to significant timing inaccuracies and circuit failure if reliable analysis of power distribution networks cannot be accomplished early in the design process. A critical component of such analysis is determining IR (resistive voltage) drop for nets. You can perform IR-Drop analysis on both DC and signal nets, though you would typically use it on power/ground nets.

The IR-Drop analysis functionality obtains voltage drop data by analyzing the nets to calculate the resistance of each meshed cell, via, and cline on one or more selected nets. With a simple mouse click you can then view accurate voltage drops across power planes; on the clines, vias, and pins of the simulated net. You can also view current on clines vias, and shapes as well as temperatures rises on clines and shapes. As an additional aid to setting up your design for analysis, you can select specific material types for padstack plating.

Static IR Drop

Static IR-drop describes the DC voltage that develops across a conductor as a result of its electrical resistance. This voltage is proportional to the current that flows through the conductor ($V=I \cdot R$) and results in a drop in voltage available at the load devices ($V_{load} = V_{supply} - V_{drop}$).

Allegro PCB PI highlights potential problems in power delivery paths, providing visibility for both IR-drop and ‘hot-spotting’ issues. PCB PI helps to accurately design high-current power connections by quantifying the amount of voltage drop and temperature rise that are to be expected.

The Static IRDrop analysis helps you assess the following:

- Voltage drop - When maximum current is drawn, is the voltage at the load within specification?
- Temperature rise – is the power path capable of delivering the maximum supply current without excessive temperature rise? The analysis reveals local pockets of high current-density, where a risk of excessive heating exists. The temperature analysis helps you ensure that a sufficient number of parallel vias have been used in power paths.

The IR-Drop analysis functionality obtains voltage drop data by analyzing the nets to calculate the resistance of each meshed cell, via, and cline on one or more selected nets. With a simple mouse click you can then view accurate voltage drops across power planes; on the clines, vias, and pins of the simulated net. You can also view current on clines vias, and shapes as well as temperatures rises on clines and shapes. As an additional aid to setting up your design for analysis, you can select specific material types for padstack plating.

Note: You need to specify source and sink current information on a per-device or per-pin basis before you run the analysis on the selected power nets.

You perform IR-Drop analysis from PCB PI.

Post-Route Signal Integrity Analysis Using the 3D Field Solver



Important

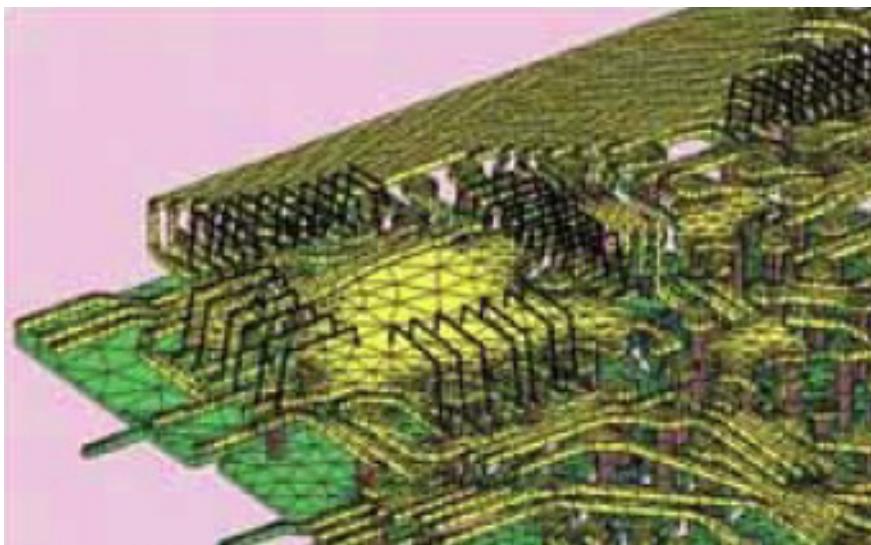
The 3D Field Solvers described in this chapter are supplied and supported by third-party vendors. You must ensure that you have the required field solver installed on your operating system to convert package design data to full 3D finite element (RLGC) models.

Introduction

With designs running at multi-GHz frequencies, it is crucial to understand and accurately model three-dimensional structures when you perform package-level signal integrity analysis.

Note: Allegro platform products recognize different geometry window settings in multiple designs in a system configuration or design link, resulting in a detailed crosstalk report that considers the different geometry window settings in each of the .mcm files. However, these multiple geometry windows apply only to 2D modeling; coupling algorithms in the 3D Field Solver which are *nearest neighbor*- based supersede any multiple geometry window settings.

Figure 10-1 Conceptual View of a 3D Wire bond MCM Package Model



What is Sentinel-NPE?

Sentinel-NPE (referred to as the *3D Field Solver* from this point forward) is a high-capacity, high performance, quasi-static electromagnetic modeling tool for IC Packaging and System-in-Package (SiP) designs. With its easy-to-use graphical user interface and direct import of package design database, package designers, and Signal Integrity (SI) engineers can efficiently build a physically intuitive RLGC model for the entire package. This tool lets you aggressively design packages while reducing or eliminating the risk of design conflicts with electrical performance specifications.

Whole Package Modeling

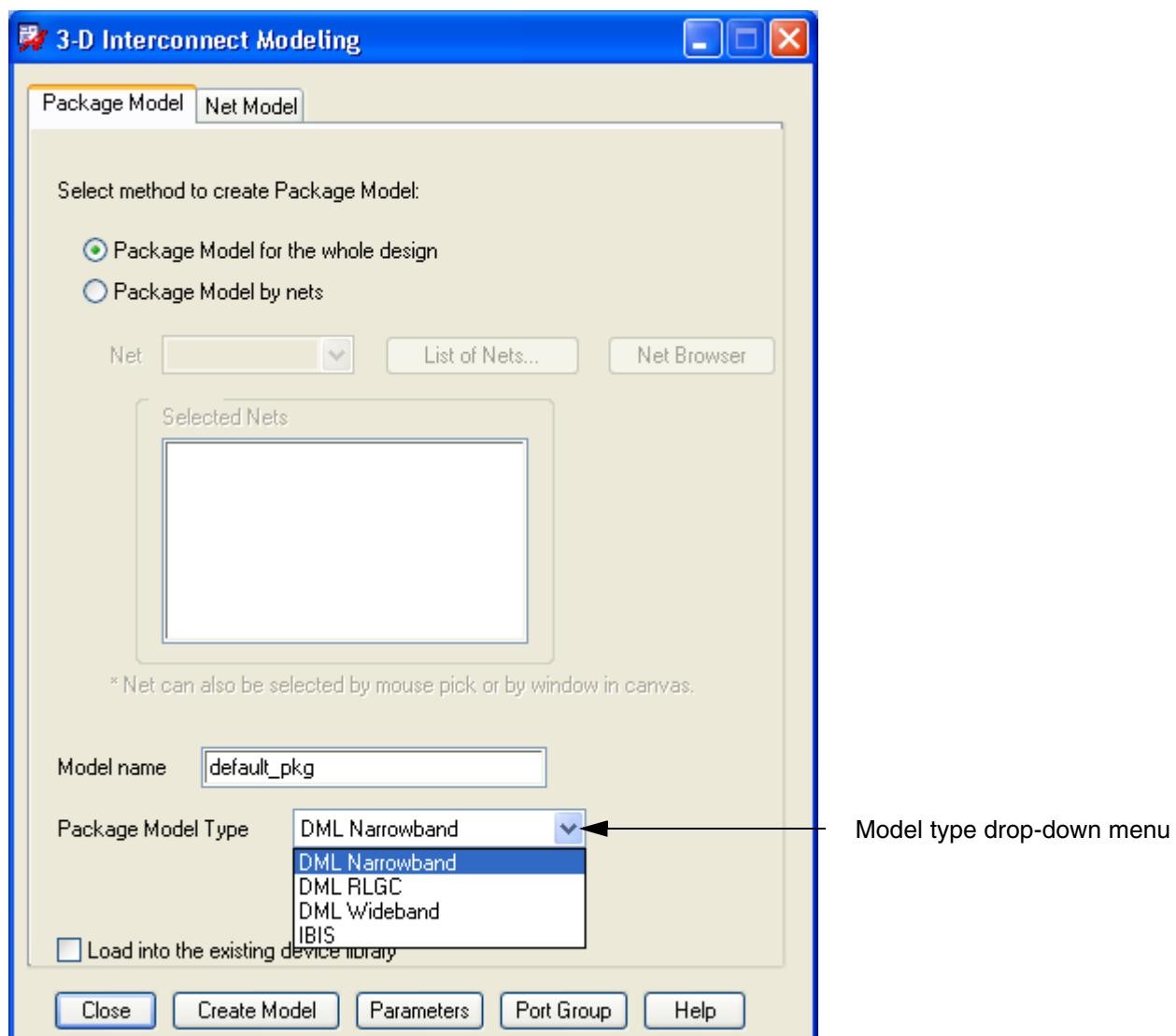
The 3D Field Solver can model the whole package or, in the case of S-Parameters, individual nets. Output can be specified as:

- Spice circuit models for single or coupled nets
 - Multi-section narrow band models
 - Wide band models
- IBIS package format
- DML format
 - RLGC

- Subcircuit wrapped
- S-Parameter in Touchstone format

These model types are displayed in the Package Model Type drop-down menu in the 3-D Interconnect Modeling dialog box. The dialog appears when you select *Analyze – 3-D Modeling* from Package SI L.

Figure 10-2 Model Types



Once you select 3D package modeling in Package SI L and select nets to simulate, the 3D Field Solver runs to generate the required models and uses them to analyze the signal integrity of the package.

Note: The 3D Field Solver outputs RLGC matrices at a single frequency point producing a narrowband circuit model. This is the default model.

Unrouted and Partially Routed Signal Nets

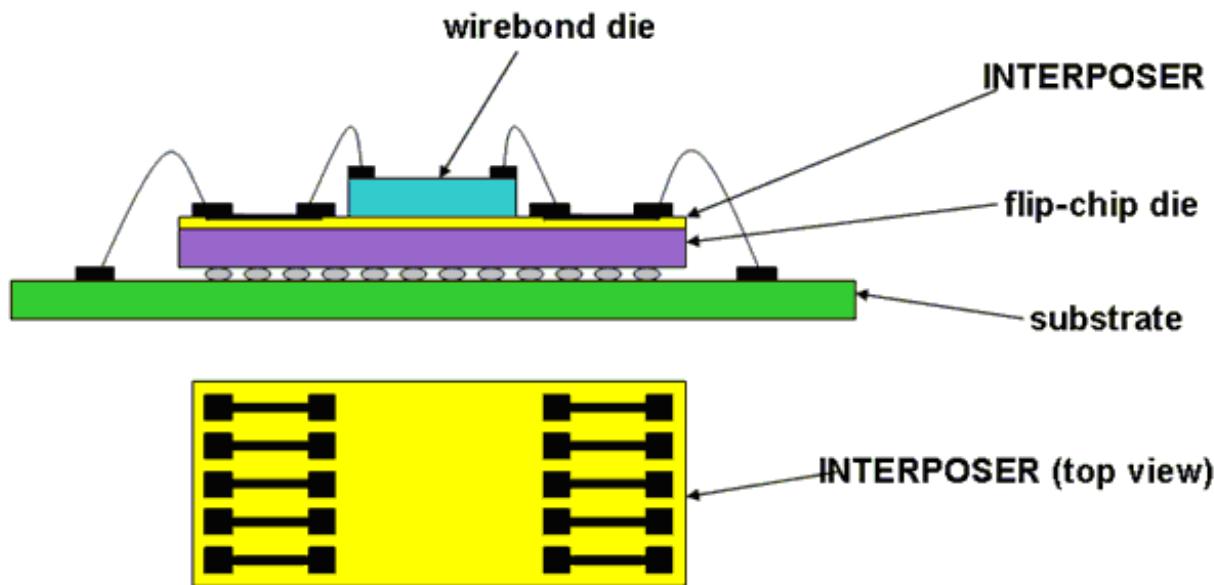
You cannot run simulations on unrouted signal nets. However, you can run simulations with partially routed nets in the design with the following limitations.

- Single Model Case
 - The 3D Field Solver puts a *skip* attribute on the partially routed net and does not generate an RLGC model for that net.
- Coupled Model Case
 - The 3D Field Solver puts a *skip* attribute on the partially routed net. However, the section that is routed is treated as metal and influences the results of the nearby nets. There is no coupling to the net that is partially routed given the *skip* tag.

Single-layer Wirebond Connections

When capturing net connectivity, 3D Field Solver recognizes single-layer wirebond connections on interposer (bonding wire) layers between dies. Bond pads and clines can reside on interposer layers, as illustrated in Figure 10-3.

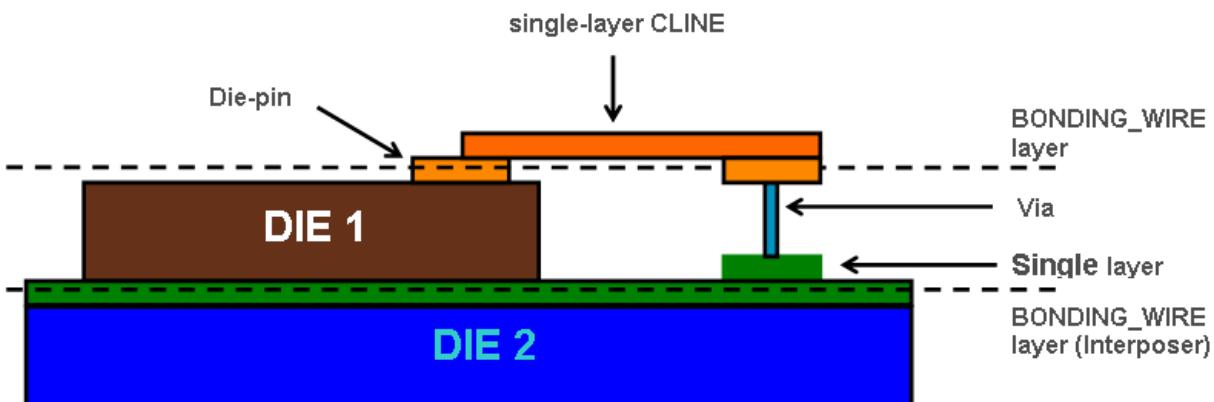
Figure 10-3 Interposer Layer



Three connection types are supported. Types 2 and 3 engender specific responses by 3D Field Solver:

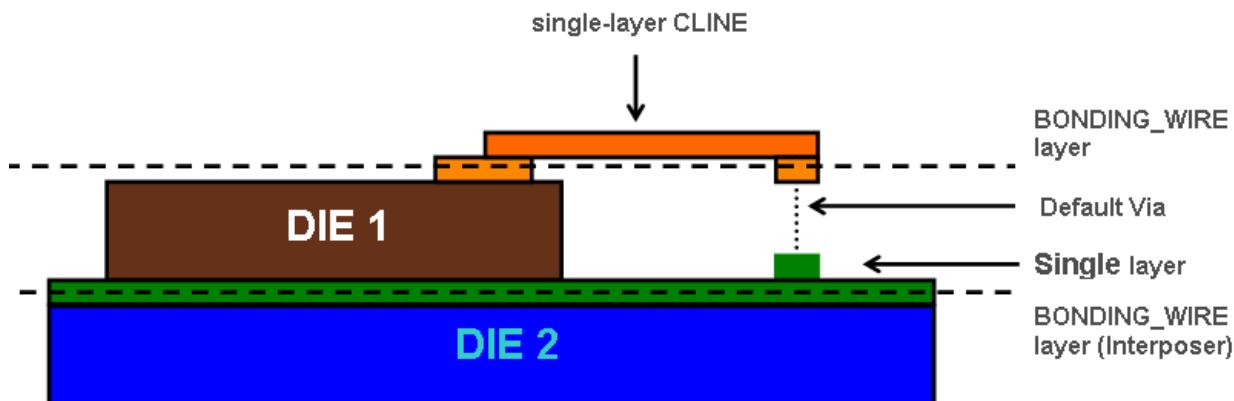
- A wirebond cline and a die pin on the same interposer layer with a design via connecting the cline to the interposer layer, as shown in Figure 10-4.

Figure 10-4 Connection Type 1



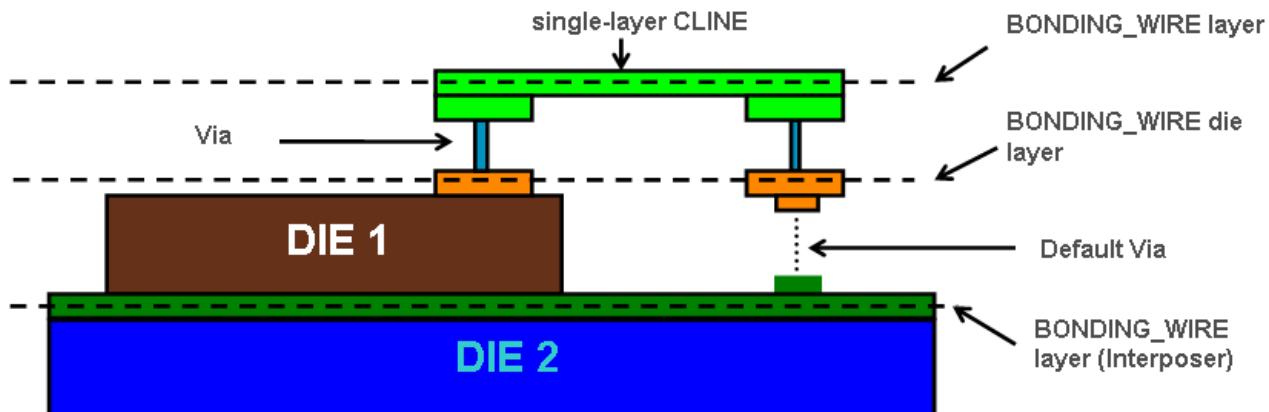
- A wirebond cline and a die pin on the same interposer layer. For this connection type, 3D Field Solver detects the condition and inserts a default via that connects the end of the cline to the interposer layer, as shown in Figure 10-4.

Figure 10-5 Connection Type 2



- A wirebond cline and a die pin are on different layers. Two design vias connect the each end of the cline to the die pin layer, as shown in Figure 10-6. For this connection type, 3D Field Solver inserts a default via that completes the connection to the interposer layer.

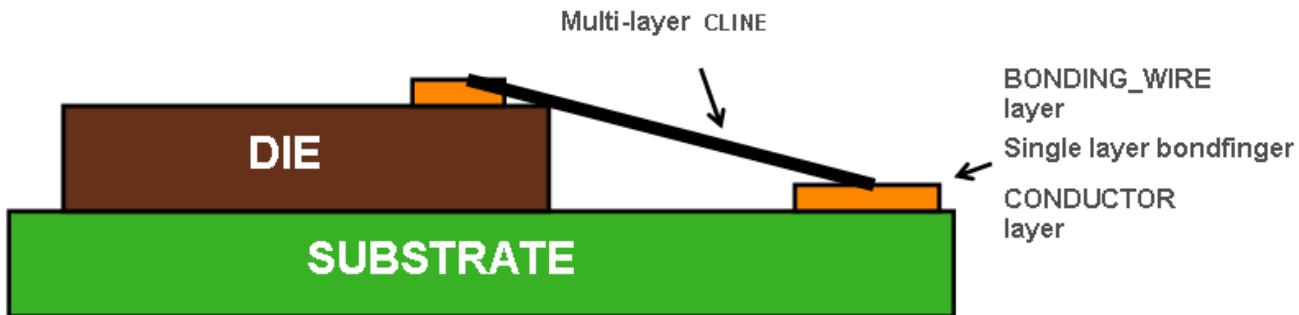
Figure 10-6 Connection Type 3



Multi-Layer Wirebond Connections

3D Field Solver also supports direct wirebond cline connections between single-layer bondpads on different layers. The conditions of this connection type are that the wirebond is of subclass ETCH_WIRE and that its two endpoints are on different etch layers.

Figure 10-7 Connection Type 4



PCB-Level Simulation

You can also use the 3D Field Solver to generate package model device files that can be automatically loaded into the SigNoise device model library and used for PCB-level simulation. For further details, see [3D Package and Interconnect Model Device Files](#) on page 369.

Supported Technologies

The 3D Field Solver supports the following configurations

- Lead Frame
- MCM
- SiP
- Stacked Die
- Complex plane shapes with perforations
- Shielding traces/coplanar shapes
- Interposer (subclass BONDING_WIRE) layers between dies

Note: DIE2DIE wire bonding configurations are not currently supported.

3D Field Solver Functional Differences

The following table lists the functional differences for the 3D Field Solver, depending on the product you use.

Table 10-1 3D Field Solver Functional Differences

Using this product . . .	you can do this....
Allegro Package SI L	Perform 3D package modeling and signal integrity analysis on a package and its interconnect.
<i>Allegro Package SI L (using SiP SI XL)</i>	
Cadence SiP Digital Architect GXL	
Cadence SiP Digital SI XL	Generate 3D package model device files suitable for signal integrity analysis. Generate 3D package interconnect model device files suitable for signal integrity analysis.
Allegro Package Designer L	Generate 3D package model device files suitable for signal integrity analysis.
Allegro Package Designer L (<i>using SiP Layout XL</i>)	
Allegro Package Designer XL	
Cadence SiP Digital Layout GXL	
Cadence SiP RF Layout GXL	

3D Modeling and Simulation Setup

To select 3D package and interconnect modeling:

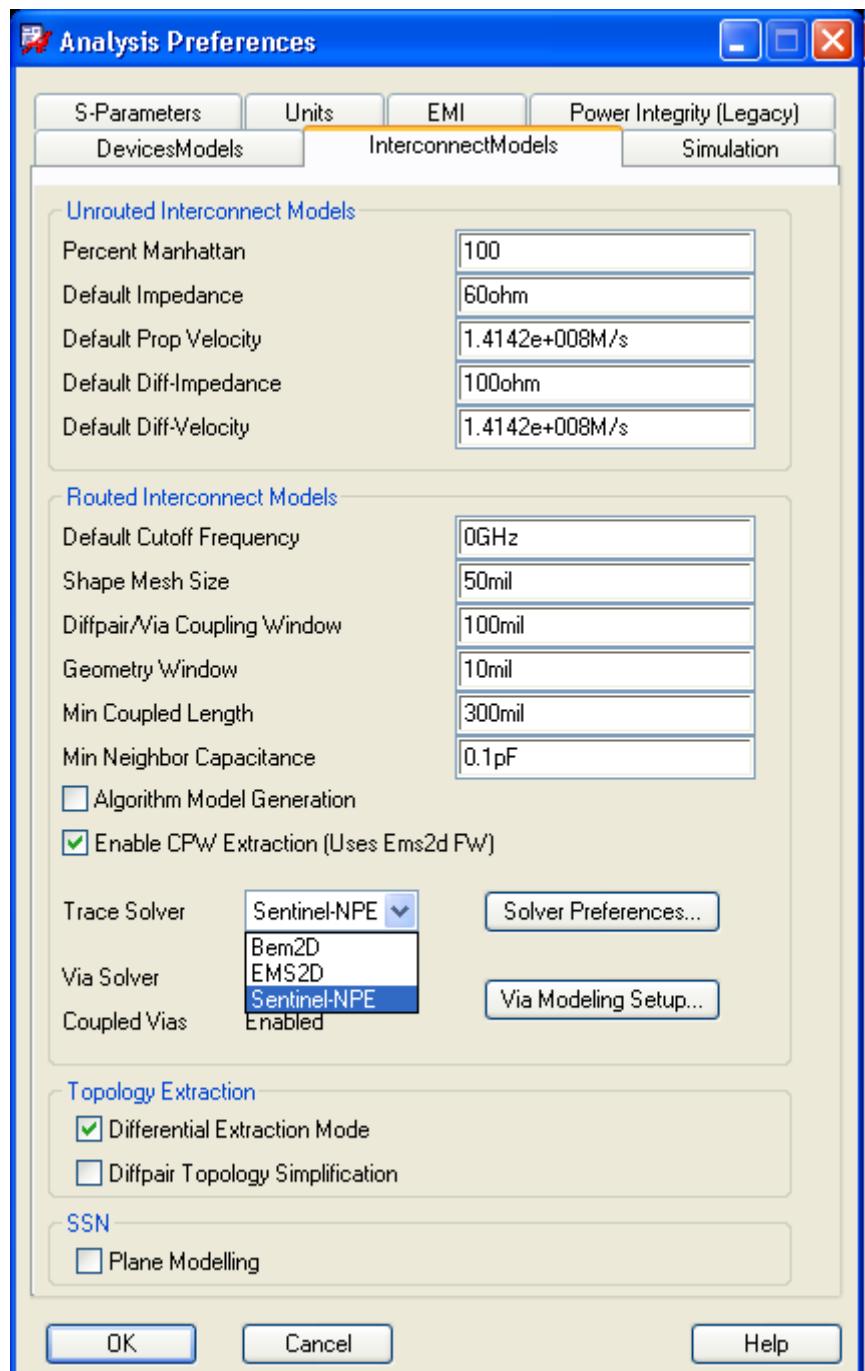
1. Choose *Analyze – Preferences*.

The Analysis Preferences dialog box appears.

2. Click on the *Interconnect Models* tab.

Allegro PCB SI User Guide
Post-Route Signal Integrity Analysis Using the 3D Field Solver

3. Select *Sentinel-NPE* from the Trace Solver drop-down list and click *OK*.



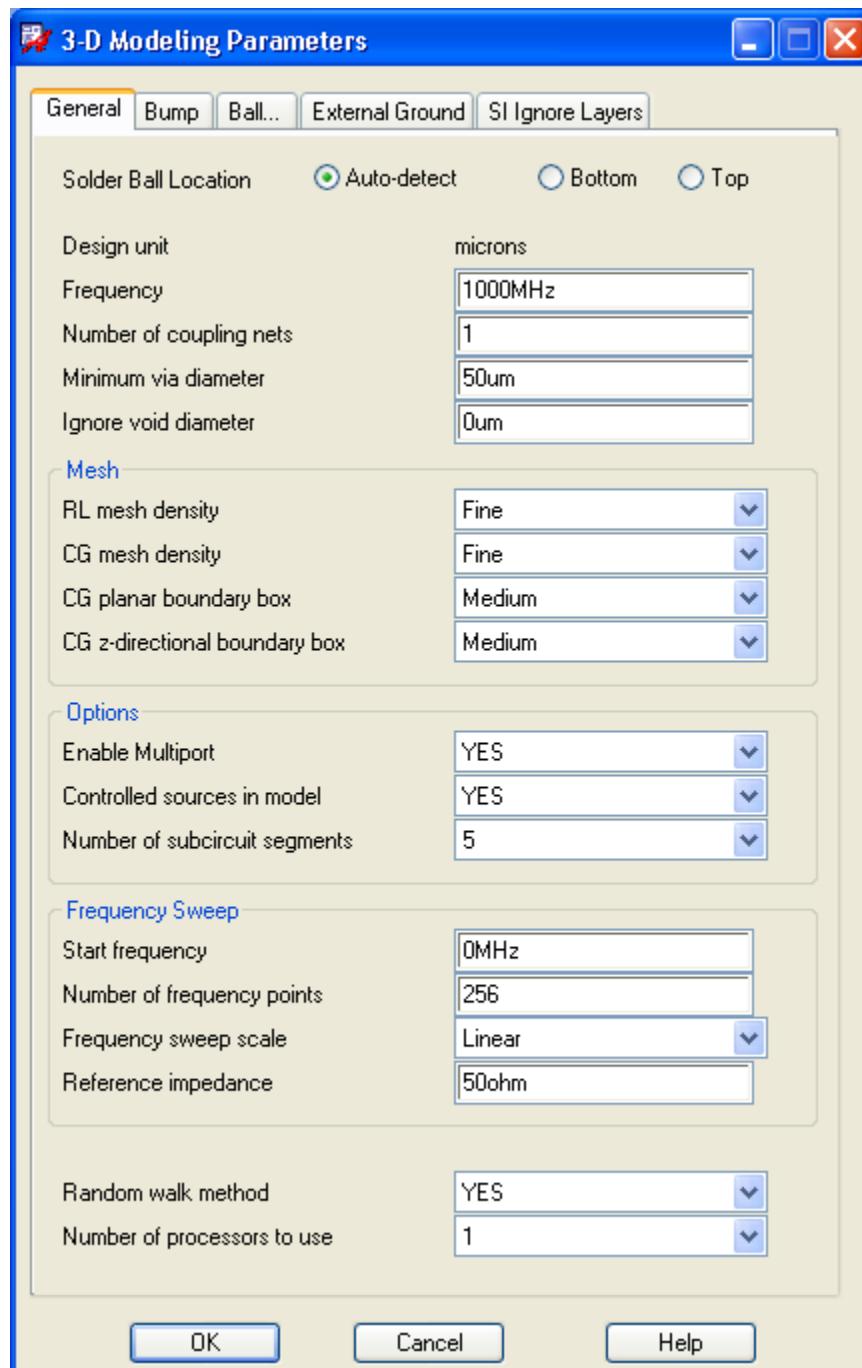
Allegro PCB SI User Guide
Post-Route Signal Integrity Analysis Using the 3D Field Solver

To set 3D modeling parameters:

1. Choose *Analyze – 3-D Modeling*.

The 3-D Interconnect Modeling dialog box appears.

2. Click *Parameters* to open the 3-D Modeling Parameters dialog.



3. Set the parameters on each tab as required and click *OK*.

For a complete description of the parameter options in the 3-D Modeling Parameters dialog box, see the [signal_prefs](#) command.

Pre-Checking Your Design

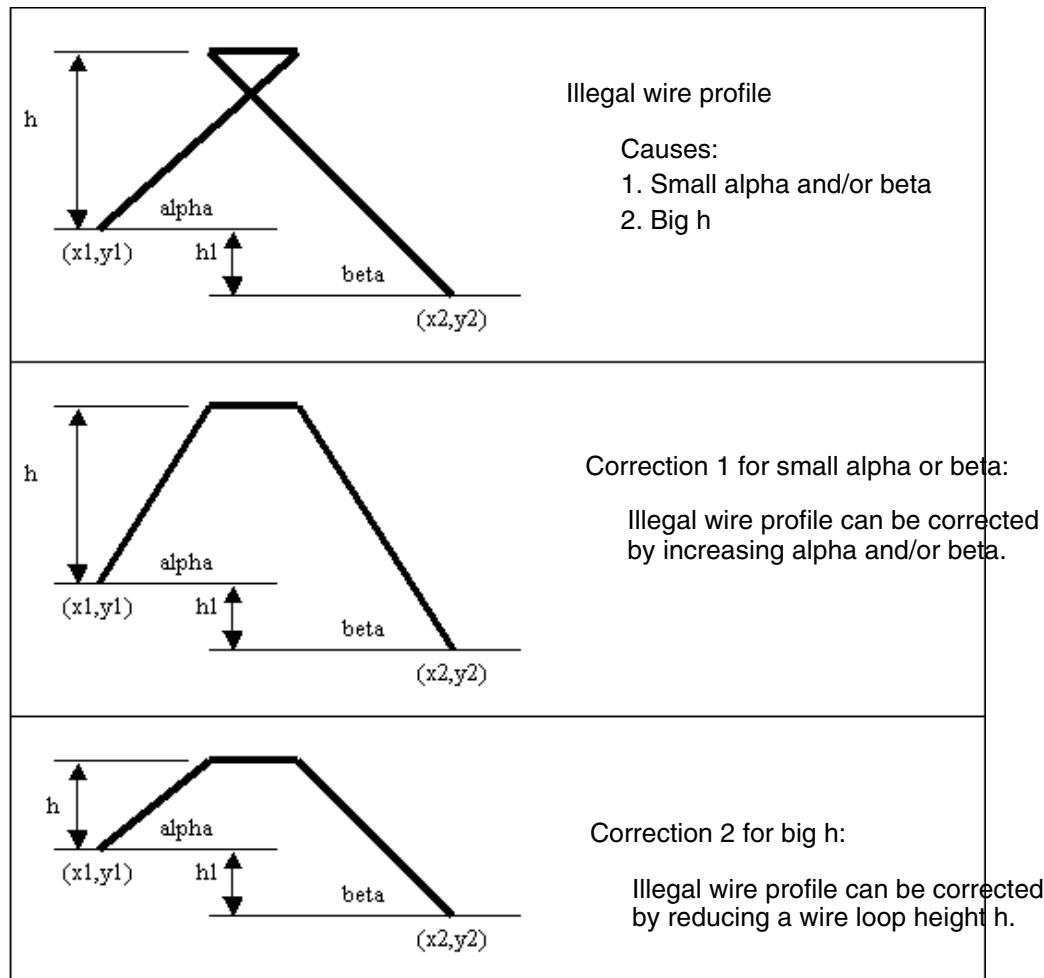
Important Setup Guidelines

During 3D model generation, there may be cases where modeling fails due to improper setup. It is strongly recommended that you check your design against the [3D Field Solver Setup Guidelines](#) on page 379 and make any necessary adjustments to ward off potential problems *before* you initiate a 3D signal integrity simulation.

Illegal Bonding Wire Checks

The 3D Field Solver automatically checks for illegal bond wire profiles. If violations are detected, a list of illegal wires is presented so that you can make adjustments. Illegal profile causes and corrections are listed in [Figure 10-8](#) on page 366.

Figure 10-8 Illegal Bonding Wire Profile Causes and Corrections



Performing 3D Signal Integrity Simulation

Pre-simulation Checklist

To perform 3D signal integrity simulation on your package successfully, you must:

- Be licensed for Allegro Package SI L.
- Adhere to the [3D Field Solver Setup Guidelines](#) on page 379 and make adjustments to your design accordingly.
- Elect 3D package modeling and simulation. For details, see [To select 3D package and interconnect modeling](#) on page 362.

Upon completion of the checklist above, you can commence with signal integrity simulation. For complete details on initiating a signal integrity simulation, refer to chapter 8 of this user guide.

3D Field Solution Progress and Control

As simulation proceeds, 3D Field Solver's progress is reported to you through the progress panel shown in figure [Figure 10-9](#) on page 368.

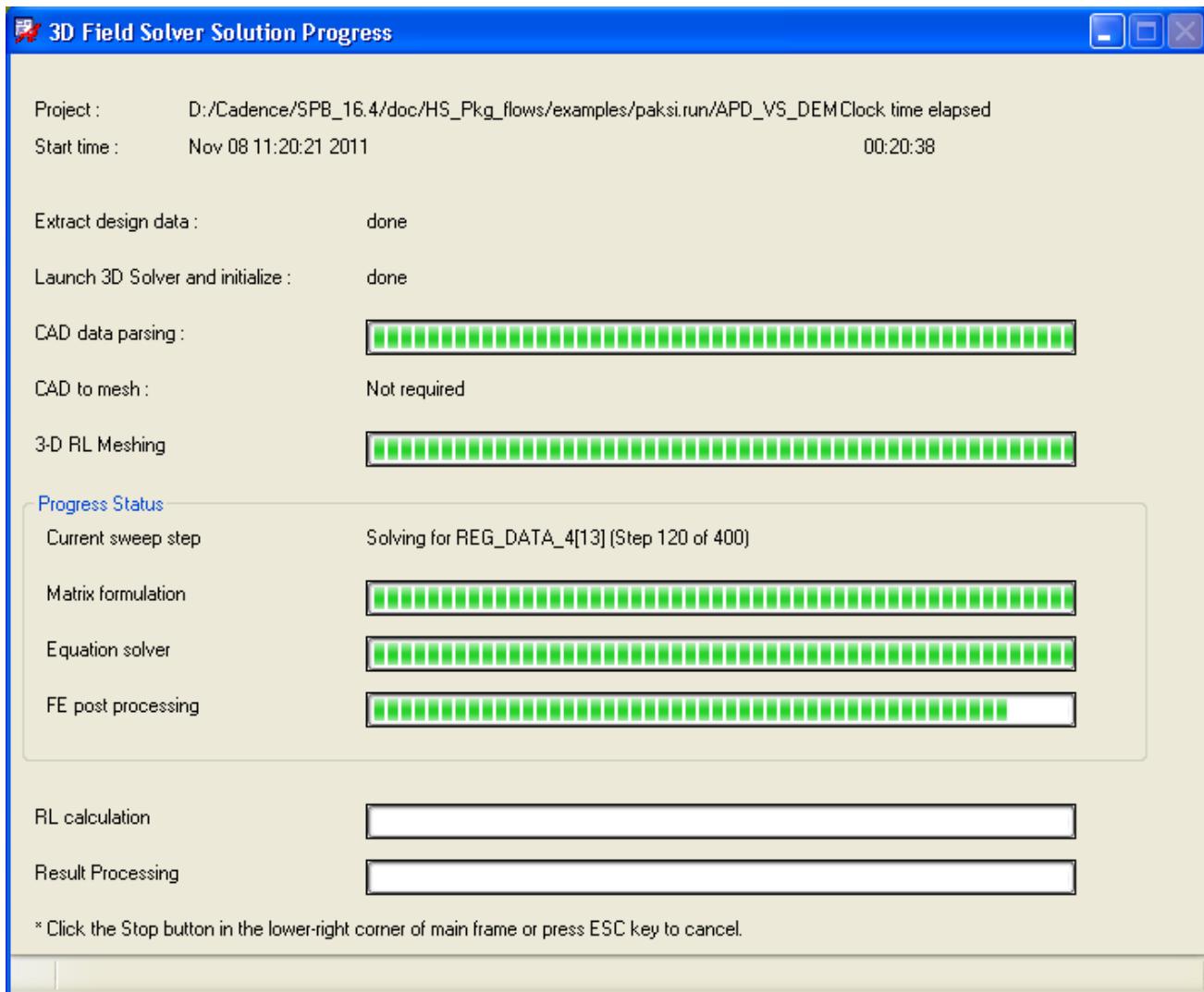
To abort a lengthy or stalled simulation

- Click the *Stop* button in the lower right corner of your design window (see [Figure 10-9](#) on page 368).
 - or -
- Press the `Esc` key on your keyboard.

A confirmation dialog box appears.

Allegro PCB SI User Guide
Post-Route Signal Integrity Analysis Using the 3D Field Solver

Figure 10-9 3D Field Solution Progress Panel



The following table describes the field solution calculations reported in the Progress panel.

Calculation	Description
<i>Extract design data:</i>	done
<i>Launch 3D Solver and initialize</i>	
<i>Cad data parsing</i>	
<i>CAD to mesh</i>	

Calculation	Description
<i>3-D RL/CG Meshing</i>	
<i>Current sweep step</i>	
<i>Matrix formulation</i>	Form, populate Finite Element Model (FEM) Matrix.
<i>Equation solver</i>	Solve the actual FEM matrix.
<i>FE post processing</i>	After solving, prepare files for RL, CG computation.
<i>RL/CG calculation</i>	Compute R, L, C, G values.
<i>Result Processing</i>	Prepare results for viewing (text files and update UI).

Intermediate Files Generated by the 3D Field Solver

It is not necessary to look at the intermediate files generated by the 3D Field Solver. However, there are several files that have good information regarding the 3D Field Solver engine. They are located in your `paksi.run` directory.

Files of interest include:

- `paksi.log`
- `corestatus.txt`
- `design.apf` (port based sink and source assignments)
- `design.agf` (die stack information)
- `design.awf` (wire endpoint information)
- `design.anf` (net display)
- `design.bon` (wire bond designs only, display wire bond information)
- `subckt` (useful to show mapping of nodes to ports)

3D Package and Interconnect Model Device Files

Generating package and interconnect model device files involves:

- Selecting the method for creating your package model.
- Selecting the method for creating your interconnect models (Package SI L only).

- Selecting a name and type for your models.
- Specifying whether you want DML models to load automatically into the SigNoise library.

For details on how to create 3D package and interconnect model files, refer to the *Allegro PCB and Physical Layout Command Reference: S Commands*.

Package Model Formats

The 3D Field Solver currently outputs package model files in the following formats.

- Spice circuit models for single or coupled nets
 - Multi-section narrow band models
 - Wide band models
- IBIS package format
- DML format
 - RLGC
 - Subcircuit wrapped
- S-Parameter in Touchstone format

For further details on translating and loading IBIS models, refer to Appendix D of this user guide.

For further details on the DML formats along with DML package model examples, refer to Appendix B of this user guide.

Model Parasitics Report

When you generate a 3D package or interconnect device model, a Parasitics report is automatically generated. You can access this report by opening the file <model_name>.csv located in your current working directory. The file is written in a tab or blank space-separated format and can be easily loaded into an Microsoft Excel® spreadsheet.

The head record line is in the following format with units specified within parentheses.

Net i,Net j,Rij (mOhm),Lij (nH),Cij (pF),Gij (uMho), TD(rs)

The data record line is in the following format.

Neti Netj Rij (mOhm) Lij (nH) Cij (pF) Gij (uMho) Td (ns) TD (rs)

Note: If <net_name_1> and <net_name_2> are identical, the RLGC are self-coupling parasitic values. Otherwise, they are mutual-coupling parasitic values.

Sample Report

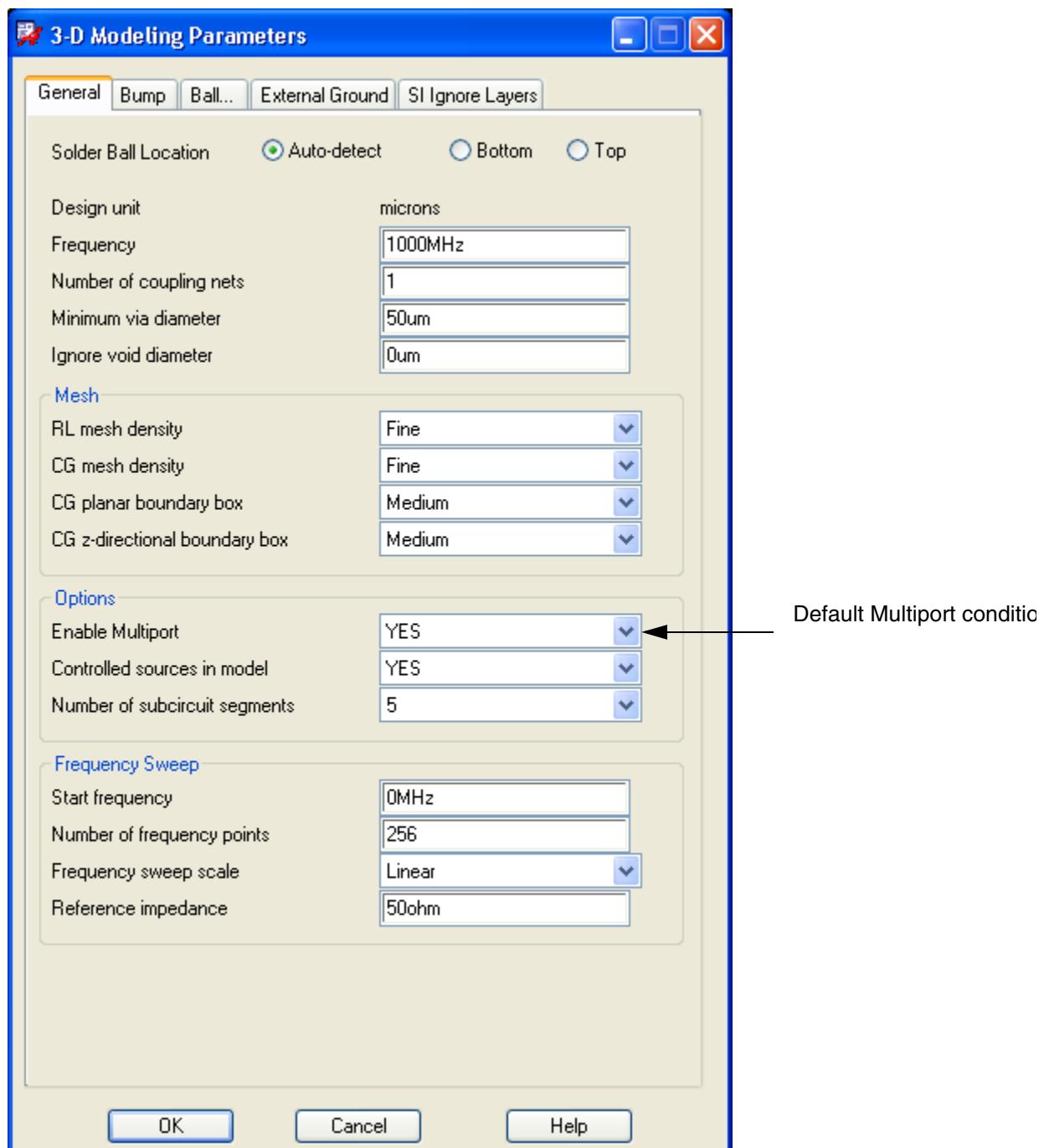
	A	B	C	D	E	F	G	H
1	Parasitic Extraction Results ----- (performed on March 22, 2007)							
2								
3	Neti	NetJ	Rij (mOhm)	Lij (nH)	Cij (pF)	Gij (uMho)	Td (ns)	
4	B	B	1.86E+02	1.70E+00	2.64E-01	1.07E+02	2.12E-02	
5								
6								

Multiport Net Support

A multiport net option allows you to specify whether multi-pin circuits will generate an equivalent lumped circuit representing all ports in the circuit in the post-processed model. If you choose not to employ this option, a multiport solution is generated for all ports; however, the post-processed model is collapsed into a two-node (input and output) lumped model. The default selection for Multiport option is YES, as illustrated in Figure 10-10.

Allegro PCB SI User Guide
Post-Route Signal Integrity Analysis Using the 3D Field Solver

Figure 10-10 Multiport Option





The multiport option is intended to model signal nets with 3 ports. While you can use this feature to help in the extraction of models of power or ground nets, it requires significant computing time and resources due to the typically large number of pin ports in power/ground nets. We recommend you exercise caution in using this feature when modeling power/ground nets.

If your designs contain multiple T-points, 3-D Field Solver will contain correct extraction results. However, be aware that the field solver does not recognize more than one T-point. It processes only the first T-point it encounters; subsequent ones are ignored. Information concerning data to and from other Ts are not reported.

Subcircuits for multiport nets use H and V sources, as shown in this sample subcircuit file.

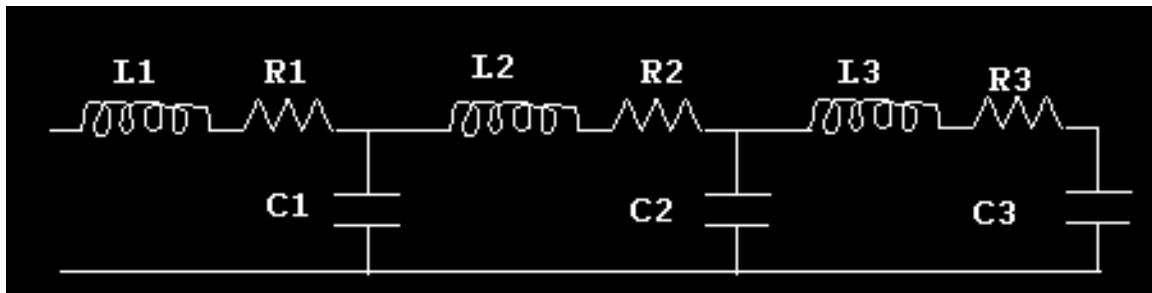
```
.subckt paksi_interconn I1 I2 I3 O1
R1 NIH1_3 M1 0.282668
L1 M1 O1 5.67408e-009
V1 I1 NI1 0
H1_2 NI1 NIH1_2 V='V2*0.000451862'
H1_3 NIH1_2 NIH1_3 V='V3*0.000409859'
R2 NIH2_3 M2 0.207849
L2 M2 O1 5.07899e-009
V2 I2 NI2 0
H2_1 NI2 NIH2_1 V='V1*0.000451862'
H2_3 NIH2_1 NIH2_3 V='V3*0.0766999'
V3 I3 NI3 0
H3_1 NI3 NIH3_1 V='V1*0.000409859'
H3_2 NIH3_1 NIH3_2 V='V2*0.0766999'
R3 NIH3_2 M3 0.348358
L3 M3 O1 7.02795e-009
CI1 I1 0 9.79101e-014
RG1I1 I1 0 3.83793e+006
CI2 I2 0 9.79101e-014
RG1I2 I2 0 3.83793e+006
CI3 I3 0 9.79101e-014
RG1I3 I3 0 3.83793e+006
CO1 O1 0 9.79101e-014
RG0I1 O1 0 3.83793e+006
K1_2 L1 L2 0.00225405
K1_3 L1 L3 0.00222227
K2_3 L2 L3 0.13538
```

```
.ends paksi_interconn
```

You can control the number of distributed subcircuits generated for a narrowband model transmission line by entering a value in the *Number of subcircuit segments* fields. Be aware that higher numbers of segments will yield more accurate models, but may increase computation time.

You can control the number of distributed subcircuits generated for a narrowband model transmission line by entering a value in the *Number of subcircuit segments* fields. Be aware that higher numbers of segments will yield more accurate models, but may increase computation time. A distributed circuit with three segments is shown in Figure 10-11.

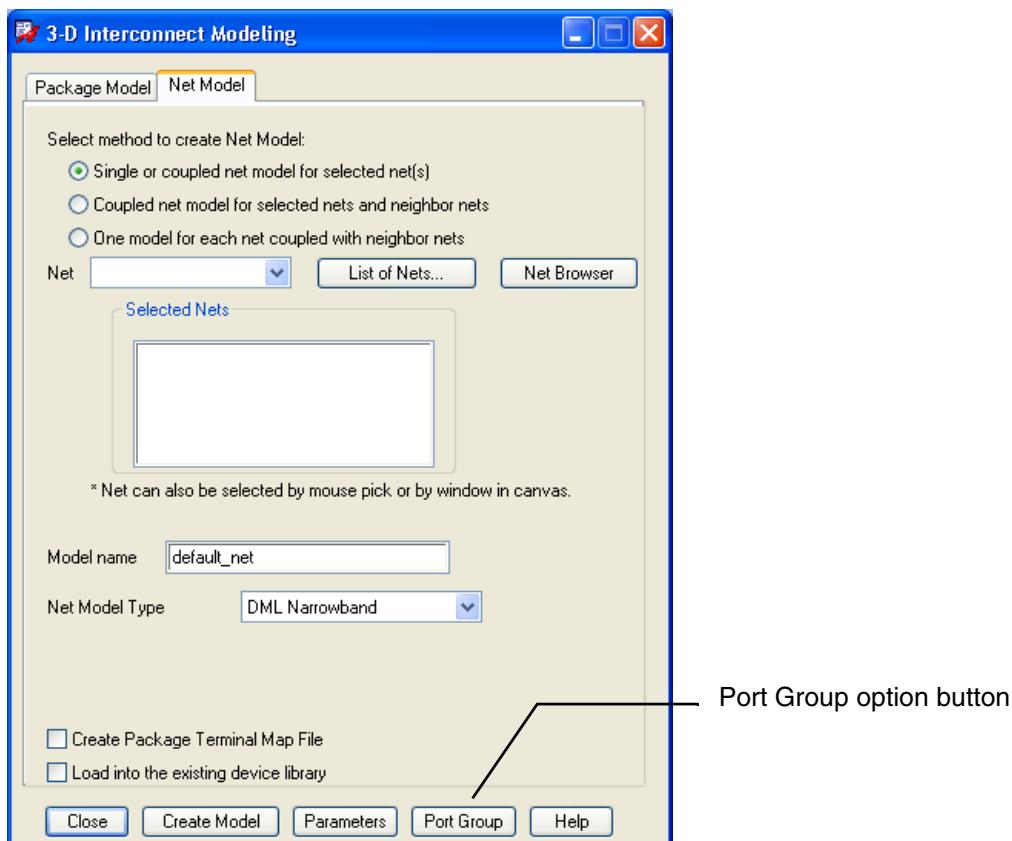
Figure 10-11 Distributed Circuit



3-D Modeling Port Grouping

Port grouping lets you group source pins and sink pins in a multiport net. Port grouping gives you the capability of setting up a partition-based extraction by enclosing ports of source and sink pins in a specified portion of your design. This eliminates the limitation of having to extract the entire design with each pin identified.

Figure 10-12 Port Group Option



For purposes of simulation, you must assign at least one source pin and one sink pin to each net. You must also designate one group as the reference group to avoid generating an error message. Otherwise, you can designate any pin (port) as either source or sink. You can also include source and sink pins in a single group. In every instance, float pins are ignored during simulation.

When you group ports in a net, the group numbers will be appended to the port name of its associated DML model, as shown in this example:

```
(“net_B_3port.dml”
(PackagedDevice
(“net_B_3port”
(ESpice “.subckt net_B_3port BGA_A5_gp1 DIE_A3_gp3 DIE_C1_gp2
Xnet_B_3port_wrap BGA_A5_gp1 DIE_A3_gp3 DIE_C1_gp2 net_B_3port
.subckt net_B_3port BGA_A5 DIE_A3 DIE_C1
* -----
* O1 = BGA-A5
* I1 = DIE-A3
```

Allegro PCB SI User Guide
Post-Route Signal Integrity Analysis Using the 3D Field Solver

```
* I2 = DIE-C1
(PinConnections
(BGA_A5_1 DIE_A3_gp3 )( BGA_A5_gp1 DIE_C1_gp2)
(DIE_A3_3 BGA_A5_gp1 )( DIE_A3_gp3 DIE_C1_gp2)
(DIE_C1_2 BGA_A5_gp1 )( DIE_C1_gp2 DIE_A3_gp3)) ) )
```

When a net contains port groups for its pins, the port names will be shown when you display the model in SigXplorer, as shown in Figure 10-13.

Figure 10-13 Port Group in SigXplorer Canvas

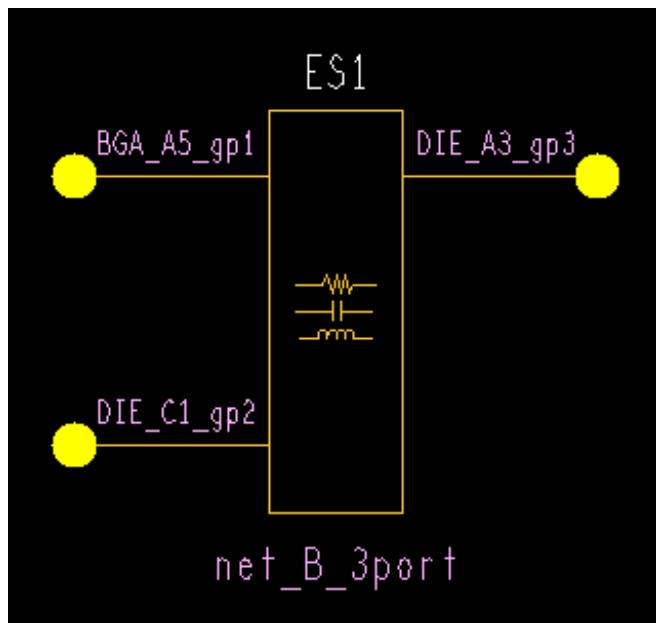
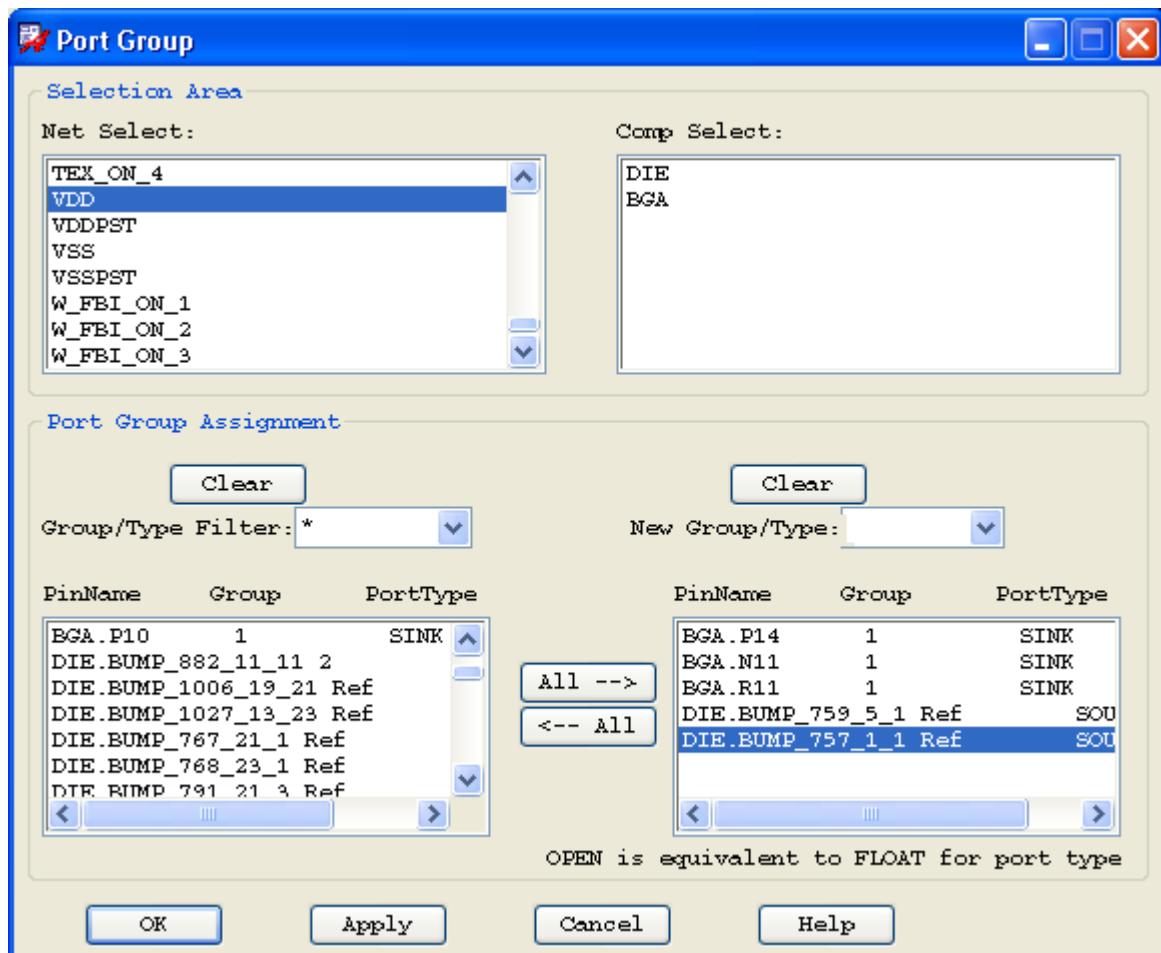


Figure 10-14 Port Group Dialog Box

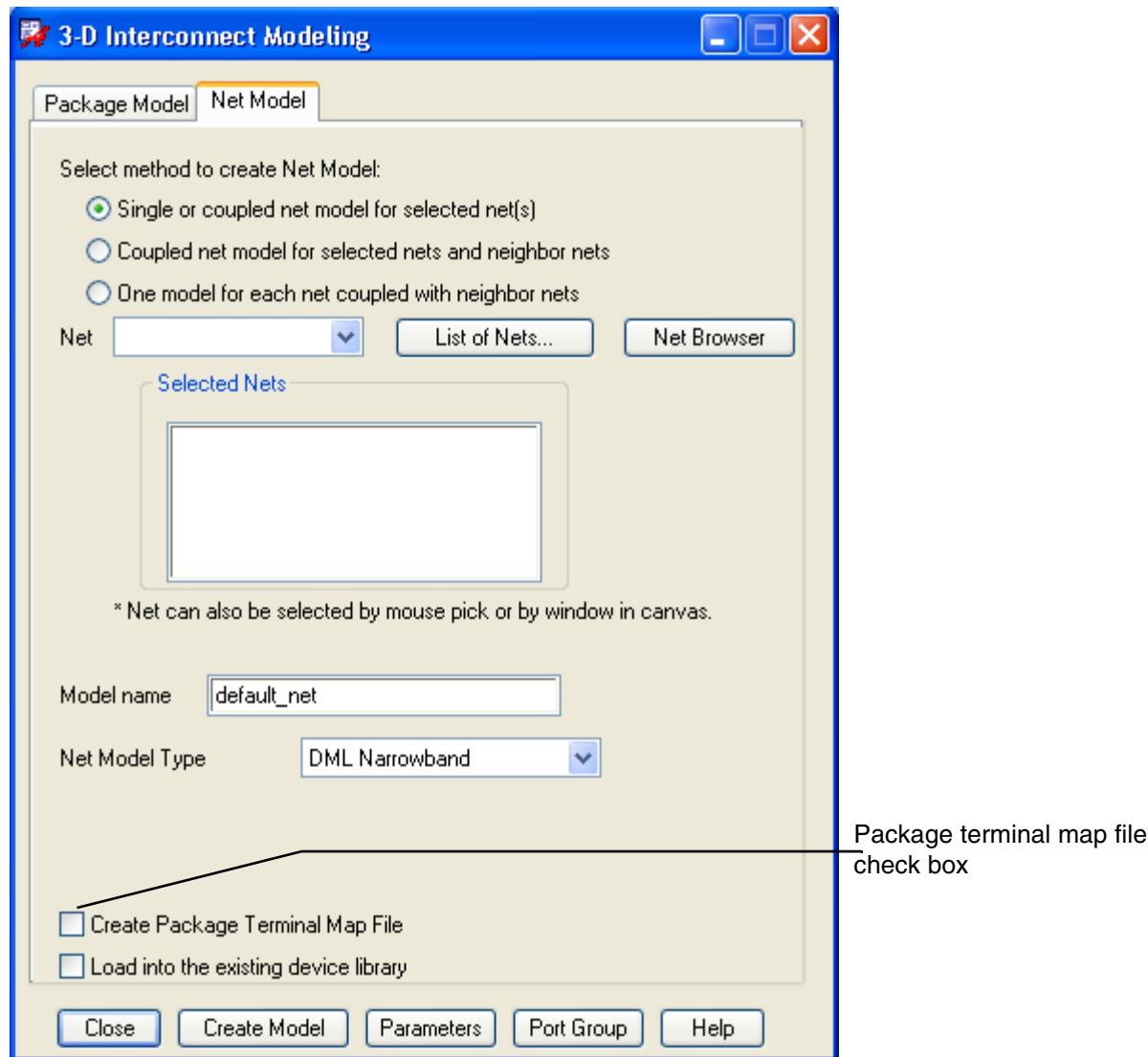


Use this dialog box to group source pins and sink pins in a multiport net. Port grouping gives you the capability of setting up a partition-based extraction by enclosing ports of source and sink pins in a specified portion of your design. This eliminates the limitation of having to extract the entire design with each pin identified.

Creating package terminal map files

You can generate a text file that maps the nodes in the 3D field solver subcircuit file to the bump pad names on the die by selecting the *Create Package Terminal Map File* option in the 3-D Interconnect Modeling dialog box. This allows IC power analysis tools to link the power/ground model in the package to the power grid circuit of the silicon in order to perform post-route simulation with package effects.

Figure 10-15 Package Terminal Map File Option



S-Parameter Model Support

3-D Field Solver can generate S-Parameters (in Touchstone format) for selected signal nets. When you choose the S-Parameter model option, you must specify the extraction parameters by filling in the *Frequency* field and the *Frequency Sweep* section in the 3-D Modeling Parameters dialog box (shown in Figure 10-10).

Note: Long extraction times may be experienced if you use a wide frequency range and a large number of frequency points. The default for these parameters are 1GHz and 256, respectively.

3D Field Solver Setup Guidelines

The following setup guidelines help you prepare Allegro Package Designer designs for use with the 3D Field Solver.

The main focus of the Package Designer and Package SI physical design environment is placement, routing, and generating layer-based artwork for manufacturing. The database is flexible and allows you to do things that are not physically possible (from a manufacturing standpoint). Given that, be aware of the potential gap that can exist between a typical MCM design and real 3D geometry.



Following these guidelines is strongly recommended by Cadence. Failure to do so may result in increased processing time, inaccurate results, or 3D modeling errors.

1. Identify DC nets

This is part of the standard high-speed setup. Failing to identify the DC nets in your design results in very long processing time to generate a field solution. The Cadence 3D Field Solver performs multiport analysis on your design which is essential for net-based simulation. It is different from Apache's standalone Field Solver in this regard. The latter performs two-port source/sink analysis by default, and performs multiport analysis only when you specifically ask for it.

Generally, a DC net is a complicated network with many pins. Multiport analysis on such a net is extremely slow and thus impractical. Therefore, you need to identify your DC nets by attaching a DC voltage property to them. By doing so, DC nets are eliminated from net-based modeling and serve as return paths for signal nets.

nets

identify

2. Set BOND_PAD property

Make sure the `BOND_PAD` property of all nets is set to YES before performing 3D model extraction.

3. Set component class properly

This is also part of the standard high-speed setup. Currently, a component with a class of IC is a die, and a component with a class of IO is a BGA ball. As you create die or BGA balls

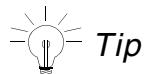
using APD or SiP SI, this convention is followed. However, if you are using PCB Editor to design your .mcm, you must double-check your component classes in advance to ensure that they are set properly.

For details on how to check component classes in your design using a Bill of Material report, see the procedures for the `reports` command in the [Allegro Package Physical Layout Command Reference](#).

4. Set cavity-up / cavity-down

5. Set package position relative to the PCB

Both Package Designer and Package SI provide an *Auto-detect* feature that analyzes the layer stackup in your design and sets package position parameter for you. However, in rare cases, *Auto-detect* may not be able to derive this information from the design. When this situation occurs, you are presented with a warning message and prompted to specify the Top or Bottom condition directly. Failure to do so produces inaccurate results or unexpected modeling errors.

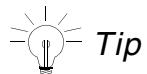


If the die and ball pins are on the same side of the package substrate, it is a Bottom package position.

For details on how to set package position, refer to the General Tab Options for the 3-D Modeling Parameters dialog box in the [Allegro Package Physical Layout Command Reference](#).

6. Set bondwire profiles

If your design is a stack bondwire or one with multiple wire tiers, you need to classify the wires into different groups with different loop height (h) values. Wires in a group have the same profile parameters (alpha, beta, and loop height). For a stack die, die on different layers should have different die elevation ($h1$) values.



You need to make sure there is no contact between wires. Currently, the 3D Field Solver has limited checking for this geometry violation.

For details on how to set bondwire profile parameters for 3D modeling, refer to the Wire Bond Editor in the Ball Tab of the [3-D Modeling Parameters dialog box](#).

7. Make sure balls or bumps do not contact each other

Before setting ball or bump parameters (*Die Component*, *Dmax*, *D1*, *D2*, and *HT*), check the pad size as well as the spacing between balls or bumps. Set these parameters using *reasonable* values to ensure that they do not contact each other. Specifically, *Dmax* should not be set so large that adjacent balls or bumps touch or overlap each other.

For details on how to set ball / bump parameters for 3D modeling, refer to the Ball Tab Options and the Bump Tab Options for the 3-D Modeling Parameters dialog box in the [Allegro Package Physical Layout Command Reference](#).

8. Set SI Ignore layers

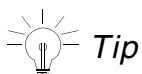
Both Package Designer or Package SI allow conductive layers for masking purposes. You can place etch and shapes on these layers as long as they have subclass names. These *fictitious* layers can cause problems for the 3D Field Solver because they introduce illegal metal contact that short the nets together.

To avoid this situation, you must mark these layers to be ignored by the 3D Field Solver. The dielectric layers directly above and below them may also need to be marked. Failing to do so may cause inaccurate 3D modeling and faulty SI analysis results.

For details on how to set layers to be ignored by the 3D Field Solver, see the procedures for the [signal prefs](#) command.

9. Make sure that the ball pad is on the most external metal layer

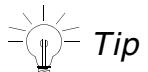
In certain cases, although the ball pad is on an external conductor layer, the 3D Field Solver may display an error. This is because bondwire layers are not classified as metal layers.



This error is usually fixed by checking for fictitious layers in the stackup and making sure they are set to be ignored by the 3D Field Solver. See guideline [3. Set component class properly](#) on page 379 for further details.

10. Do not route plating bars

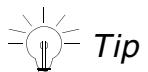
A plating bar is occasionally used in a package design (for manufacturing purposes). However, if the bar is routed, it shorts all the signal nets producing unexpected modeling errors. This may prevent the 3D Field Solver from completing a field solution.



Tip
The simplest way to prepare designs that contain routed plating bars for 3D SI analysis is to back up the original design and delete the plating bar.

11. Make sure that there are no vias or through-hole pins at the same location

Some package designs may have two or more vias / through-hole pins placed at the same x-y location with padstacks overlapped. Most likely, they are on the same net. The 3D Field Solver considers this a geometry violation. Unless you relocate overlapping vias or through-hole pins, unexpected modeling errors may occur.

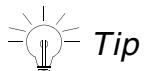


Tip
To check for this condition, invoke a via-to-via spacing DRC with the Same Net Checking option turned on.

cns space values

12. Check the spacing clearance between etch objects

We recommend a minimum 10-micron spacing between separate etch objects. In the early design stage, when there are no basic spacing DRC rules set, you may use 1 database unit (minimum spacing) for clearance between a via and a positive shape. With such a small gap, the 3D Field Solver may generate overlapped meshing cells due to numerical error accumulation, and erroneously connects the etch.



Tip
To check this, invoke basic spacing DRCs such as line-to-line, line-to-shape, and via-to-shape with reasonable rules. Use of the Same Net checking option is not necessary.

cns space values

13. Check your padstacks

Make sure that the drill hole size is smaller than the minimum pad diameter. Otherwise, the following error appears when you initiate 3D modeling.

ERROR in design

The drill hole size is equal or larger than the smallest pad size in padstack <name>.

14. Do not route a bondwire with more than one segment

This condition is not currently supported by the 3D Field Solver.

15. Do not route four or more bondwires onto the same point of a bond pad

This condition is not currently supported by the 3D Field Solver.

Note: This rule does not include DC nets.

Interpreting 3D Modeling Messages

When 3D modeling problems occur during simulation, error and warning messages appear. [Table 10-2](#) on page 383 and [Table 10-3](#) on page 384 contain descriptions of some of the warning and error messages that may appear in the console window during the 3D modeling process.

Table 10-2 Warning Messages

Message	Cause
Could not open file <filename>.	Error when opening a file that is not critical for program continuation.
No neighbor nets found.	There were no neighbor nets to the reference net.
Couldn't remove temp file.	Could not remove a temporary FEA model file (.HDO).
DC-reduction for Eisenstat algorithm can not apply!	Warning during equation solving.

Message	Cause
File jobname.ELE may have data integrity problems.	Some data conflicts occurred during equation solving.
Warning: Cannot C-reduce non-sorted matrix!	Warning during equation solving.
Warning: Cannot DC-reduce non-sorted matrix!	Warning during equation solving.
Warning: Cannot D-reduce non-sorted matrix!	Warning during equation solving.
Warning: Cannot factorize non-sorted matrix!	Warning during equation solving.
Warning: DRIC can only apply to Column Index Sorted Matrix!	Warning during equation solving.

Table 10-3 Error Messages

Message	Cause
Error: Temporary file rewind error!	Could not rewind a temporary file (.TP1 or .TP2).
Incorrect Constraint Equation number!	Incorrect constraint equation number.
Incorrect coupled DOF set number!	Incorrect coupled DOF set number.
Incorrect element ID number in FEA model file!	Incorrect element ID number in FEA model file (.HDR).
Incorrect FEA model file towards end!	FEA model file (.HDR) is incorrect.
Incorrect file version of FEA model file!	Version number of an FEA model file (.HDR) is incorrect.

Allegro PCB SI User Guide
Post-Route Signal Integrity Analysis Using the 3D Field Solver

Message	Cause
Incorrect number of DOFs per node!	Number of DOFs per node was <= 0 or > 32.
No net is selected for analysis.	No net has been selected for analysis.
The iFE solver log file can not be open!	A finite element solver log file iFESolver.log could not be opened.
Zero DOF remaining after elimination!	Zero DOF remaining after finite element matrix elimination.

Allegro PCB SI User Guide
Post-Route Signal Integrity Analysis Using the 3D Field Solver

Dynamic Analysis with the EMS2D Full Wave Field Solver

High density interconnect on PCB and packaging designs with signal switch rates over 5 Gpbs require model characterizations that can support frequency ranges from DC up to THz. Within this wide spectrum, electrical resonance, oscillation, signal dispersion and EM radiation are all likely and must be accounted for. Static or Quasi-static characterization such as Bem2d is not able to address these high frequency issues. Skin effect and dielectric loss are analyzed by simple formulation or empirical equations. Therefore, a full-wave solution is needed to handle these electromagnetic interaction effects.

The Electromagnetic Solution 2D Full Wave field solver—EMS2D—provides the full-frequency range analysis from DC, through the middle frequency range which covers the skin effect, to the THz range of the electromagnetic interactions which address resonances, radiations and EM signal integrity issues.

EMS2D is implemented using the finite element method (FEM), which complements Allegro's moment-based BEM2D field solver. EMS2D combines multiple EM computation modules, static, quasi-TEM, and full-wave analysis. Additionally, EMS2D is able to analyze arbitrary transmission line-type and waveguide structures over PCB cross-sections and provide characterized models in table format.

EMS2D Operating Parameters

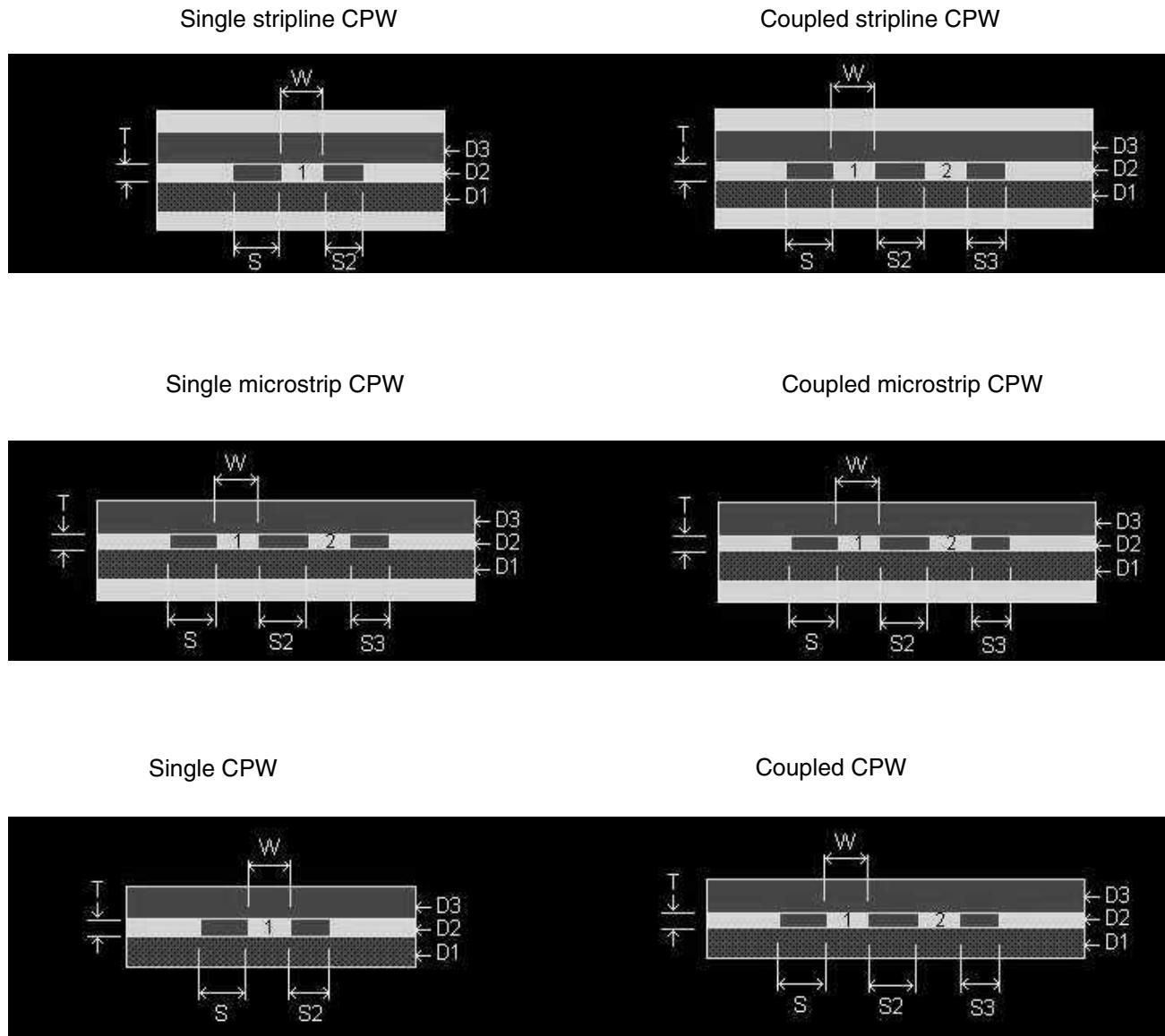
The EMS2D full wave field solver supports the following capabilities:

Coplanar Waveguide Characterization

EMS2D supports analysis of coplanar waveguide (CPW) structures, including single and differential coupled CPWs in differential pair, microstrip, or stripline types. CPW examples are shown in Figure [11-1](#).

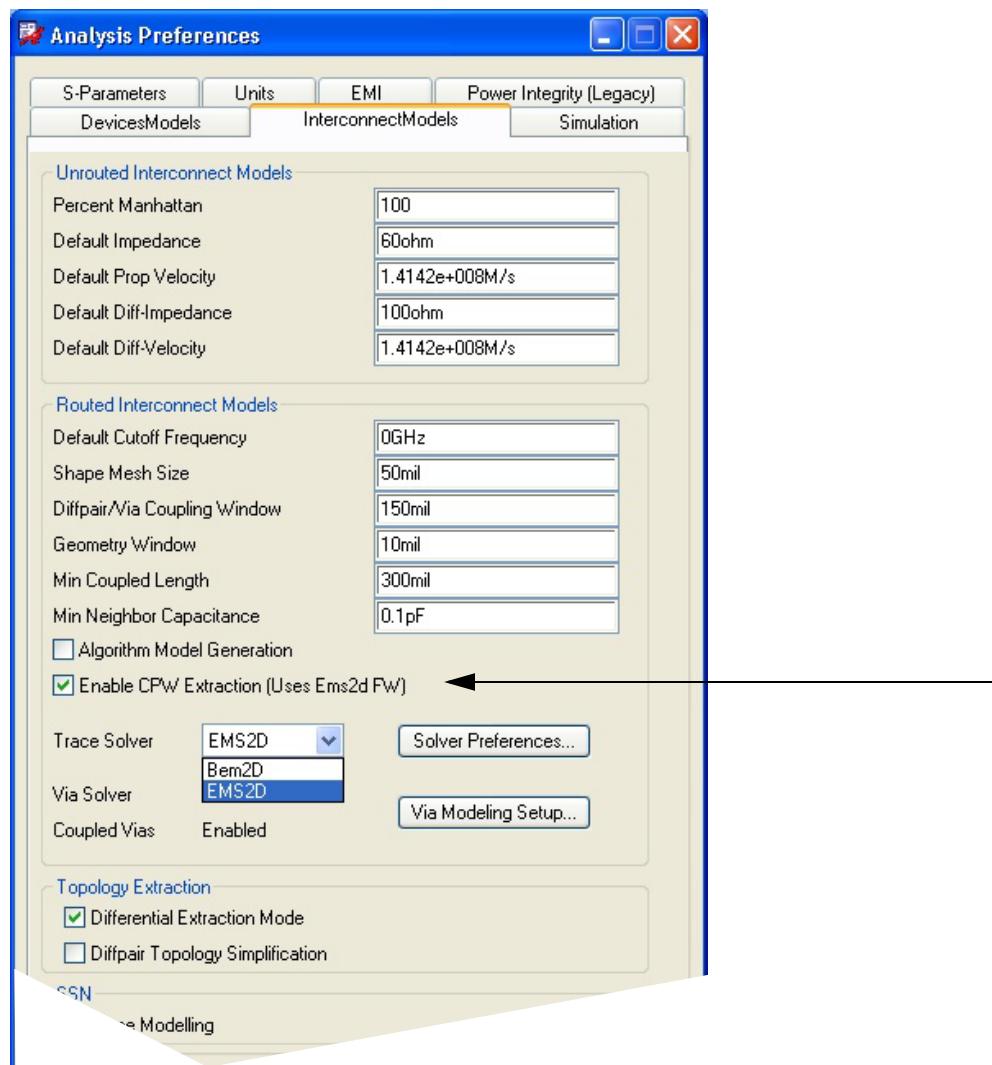
Allegro PCB SI User Guide
Dynamic Analysis with the EMS2D Full Wave Field Solver

Figure 11-1 CPW Structures



In Allegro PCB SI, you can extract CPWs for model generation by enabling the CPW extraction option in the InterconnectModels tab of the Analysis Preferences form, shown in Figure 11-2.

Figure 11-2 CPW Extraction Option



With CPW extraction enabled, EMS2D determines whether a single net should be handled as a CPW based on the presence of two shapes adjacent to the cline (shown in the following illustration). Each shape is searched using a window equal to the geometry window setting. The presence of adjacent nets between the net you are extracting and adjacent shapes is *not* considered.

To set and detect coplanar waveguides

1. Choose *Analyze – Preferences*.

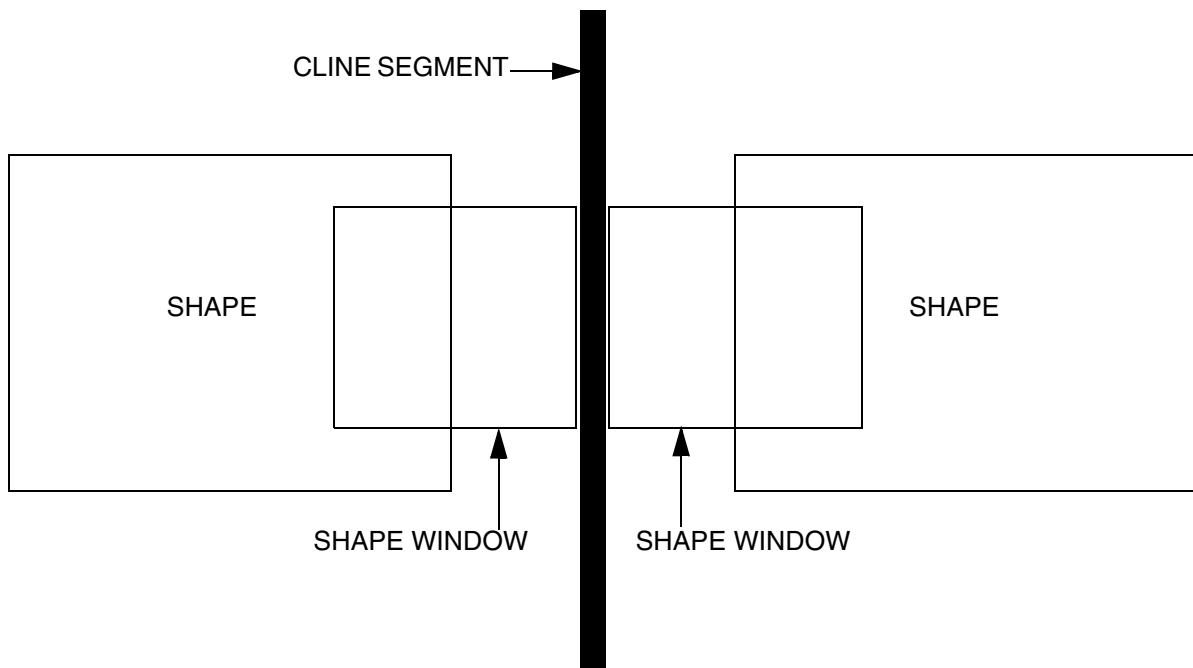
The Analysis Preferences dialog box appears.

2. Click the InterconnectModels tab.
3. Select *Enable CPW Extraction* and *Ems2d FW* to enable coplanar waveguides in the entire design.
4. If you wish to disable CPW for specific nets, do the following for each selected net, otherwise proceed to step 5.
 - a. Right-click on the net you want to apply the *CPW_DISABLED* property to.
 - b. Choose *Property Edit* from the pop-up menu.

The Edit Property dialog box opens.

 - c. Select *Cpw_Disabled* from the Available Properties list and click *Apply*.

The selected net will now be handled during analysis as a non-CPW net. If you have selected only the *Ems2dFW* option (without *Enable CPW Extraction*), non-CPW nets will be generated with Bem2d.
5. Set the Geometry Window parameter to accommodate the configuration of DC shapes surrounding the cline segment, as shown in the graphic.



For each segment of the cline, Ems2d will use the dimensions set in the Geometry Window to check for shapes on either side of the cline.

6. Click the *Preferences* button to open the EMS2D Preferences dialog box.

Allegro PCB SI User Guide
Dynamic Analysis with the EMS2D Full Wave Field Solver

7. Choose the frequency settings and other options appropriate for your analysis. These settings are explained in the [EMS2D Preferences Dialog Box](#) section.
8. Click *OK*.

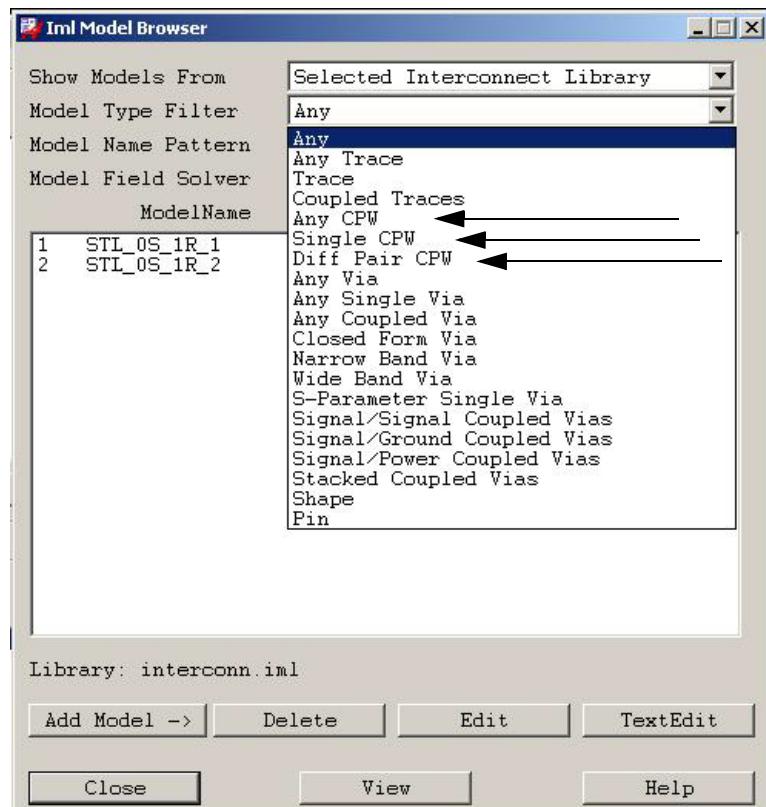
Library Enhancements

Interconnect libraries in Allegro products that support EMS2D contain a number of enhancements. They include:

- CPW structures (as described in the previous section)

CPW structures are represented by interconnect models in IML libraries. You can filter model displays in the Iml Model Browser, as shown in Figure 11-3. You can also create new or cloned models for single and/or differential pair CPWs.

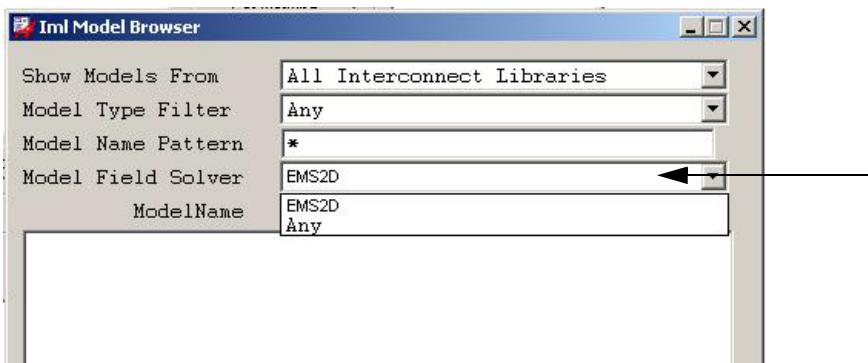
Figure 11-3 CPW Models in the IML Model Browser



- Field solver identification

Interconnect models created by specific field solvers are identified as such in the model syntax (`FieldSolver`). This information is then used to “tag” models that can be reused by the field solver that created the model. Models created by specific field solvers are displayed in the Iml Model Browser when the appropriate field solver is selected from the *Model Field Solver* drop-down menu, as shown below.

Figure 11-4 Model Field Solver Selector



Dispersive Dielectric Material Support

The EMS2D field solver is capable of accurately modeling the delay and dispersive behavior of arbitrary materials, thus allowing rigorous analysis of frequency-dependent material properties.

Frequency-dependent materials are assumed to be correctly defined in the frequency-dependent material (.material) file, an example of which is shown below. This type of file contains electrical properties for individual materials (for example, copper) defined over a range of frequencies. If the material data in the file is available only over a *specified* frequency range, EMS2D will use the lowest available frequency point for DC extrapolation and will use the highest available frequency point for asymptotic analysis. For any other points, piecewise linear (PWL) interpolation will be used by default.

Sample of a frequency-dependent material file

```
! material file used in Electromagnetic Solution
! limited one material per file
! freq parameter valuetype      material_type      material_name version

# GHz      CEr      Complex      Anisotropic      ML3      0.1
! dimension = 1 for isotropic, dimension = 3 for anisotropic
!freq   er(1,1)  er(1,2)  er(1,3)  er(2,1)  er(2,2)  er(2,3)  er(3,1)  er(3,2)
er(3,3)
0.0001  3.5 0.1    0 0    0 0    0 0    3.5 0.1    0 0    0 0    0 0    3.5 0.1
0.0800998 3.4 0.2    0 0    0 0    0 0    3.4 0.2    0 0    0 0    0 0    3.4 0.2
0.1601  3.4 0.3    0 0    0 0    0 0    3.4 0.3    0 0    0 0    0 0    3.4 0.3

! omh*meter
```

Allegro PCB SI User Guide

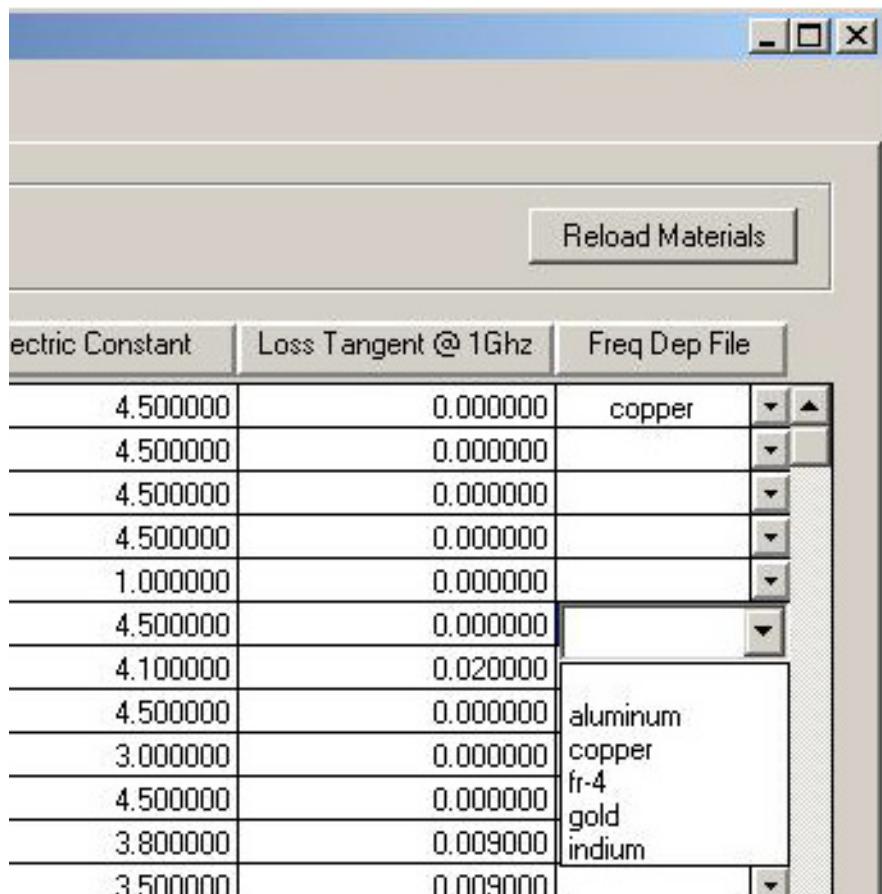
Dynamic Analysis with the EMS2D Full Wave Field Solver

```
# DcConductivity=1.e-6
```

#	GHz	LossTangent	Real	Anisotropic				ML3	0.1
0.0001	0.1	0	0	0	0.1	0	0	0	0.1
0.0800998	0.2	0	0	0	0.2	0	0	0	0.2
0.1601	0.3	0	0	0	0.3	0	0	0	0.3

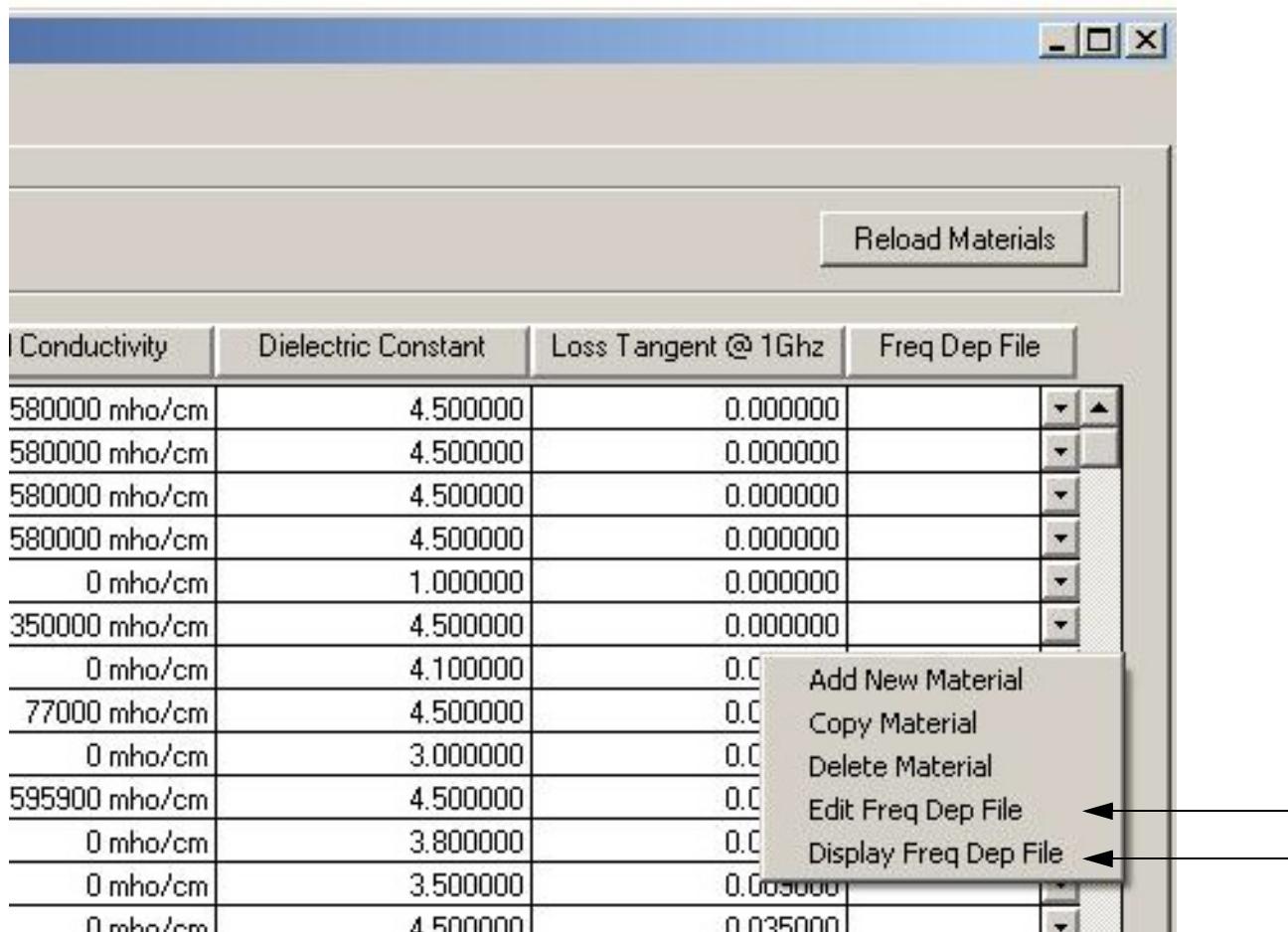
Frequency-dependent material files for specific materials and/or layers are defined graphically in the Material Properties form and Cross section editor in the Allegro tools. In either form, you can select a frequency-dependent file from the files residing in your MATERIALPATH directory, //<install_directory>/share/pcb/test/materials, as illustrated in Figure 11-5. All Allegro products that support EMS2D will include a set of default material files in that location.

Figure 11-5 Material File Selectors in Material Properties Form



In addition, you can edit (in a text file) or display (in SigWave) the frequency-dependent file associated with a material or layer by way of the right-button pop-up menu, as shown in Figure 11-6.

Figure 11-6 Right Mouse Button Option Menu



S-Parameter Extraction

EMS2D extracts S-Parameters when the segment length of interconnect is specified. In such cases, the S-Parameter is output in Touchstone file format (.snp) that you can view in SigWave. The associated frequency points will be specified in the frequency point (.frequency) file. The command line option for this feature is

```
-sparam <filename.snp> -length <a_number_in_meters> -frequencypointfile
<filename.frequency>
```

These parameters can also be set in the EMS2D Preferences form, accessed from the Analysis Preferences dialog boxes in PCB SI and SigXplorer.

Lossy Transmission Line Modeling in HSPICE

EMS2D analyzes lossy transmission line models with skin effects when provided with sufficient multiple frequency points to cover the skin effect range. The command line option for this feature is

```
-HspiceRlgcFile <filename.rlc>
```

Sample lossy transmission line model

```
* RLGC parameter for a 3-conductor lossy
* frequency-dependent line
3
* L0
2.7234e-007
6.90478e-008 2.82732e-007
2.33805e-008 6.90478e-008 2.7234e-007
* C0
4.35837e-011
-1.03689e-011 4.44211e-011
-1.10836e-012 -1.03689e-011 4.35837e-011
* R0
44.2087
-1.00052e-011 44.2087
-1.77929e-012 -1.00052e-011 44.2087
* G0
0
0 0
0 0 0
* Rs
0.00466516
0.000378297 0.00482829
7.18185e-005 0.000378297 0.00466516
* Gd
3.25144e-016
2.28853e-016 1.64409e-016
3.12163e-016 2.28853e-016 3.25144e-016
```

Enhanced Etch Factor Support

EMS2D supports an enhanced version of the functionality that allows you to define a trapezoidal cross-section for clines on conductor and plane layers. This functionality differs from the existing `trapezoidal_angle_in_degree` environment variable in the following respects:

- Allows different degree of angle for clines on different layers
- Allows you to set the degree of angle for either the top or the bottom of the cline

If you select BEM2D, it uses the average of all etch factors (for all layers) in the cross section when determining trace models.

If you select EMS2D, it uses the etch factor for each layer of the cross section when determining trace models.

Note: SigXplorer writes these trace models to the interconnect library.

Layer Specification

You can set a different degree of angle for every cline on a specific conductor/plane layer. You do this in the Layer Cross Section form (*Setup – Cross-section*). As shown in Figure 11-7, the Etch Factor column displays the default setting (90 degrees), for each conductor and plane layer in your design. To change the default, you simply enter a new value in the field on the appropriate row. To maintain viable angles, values are restricted to within 45 degrees of vertical, thus between 45 to 135 degrees or between 225 to 315 degrees.

Allegro PCB SI User Guide
 Dynamic Analysis with the EMS2D Full Wave Field Solver

Figure 11-7 Layer Cross Section

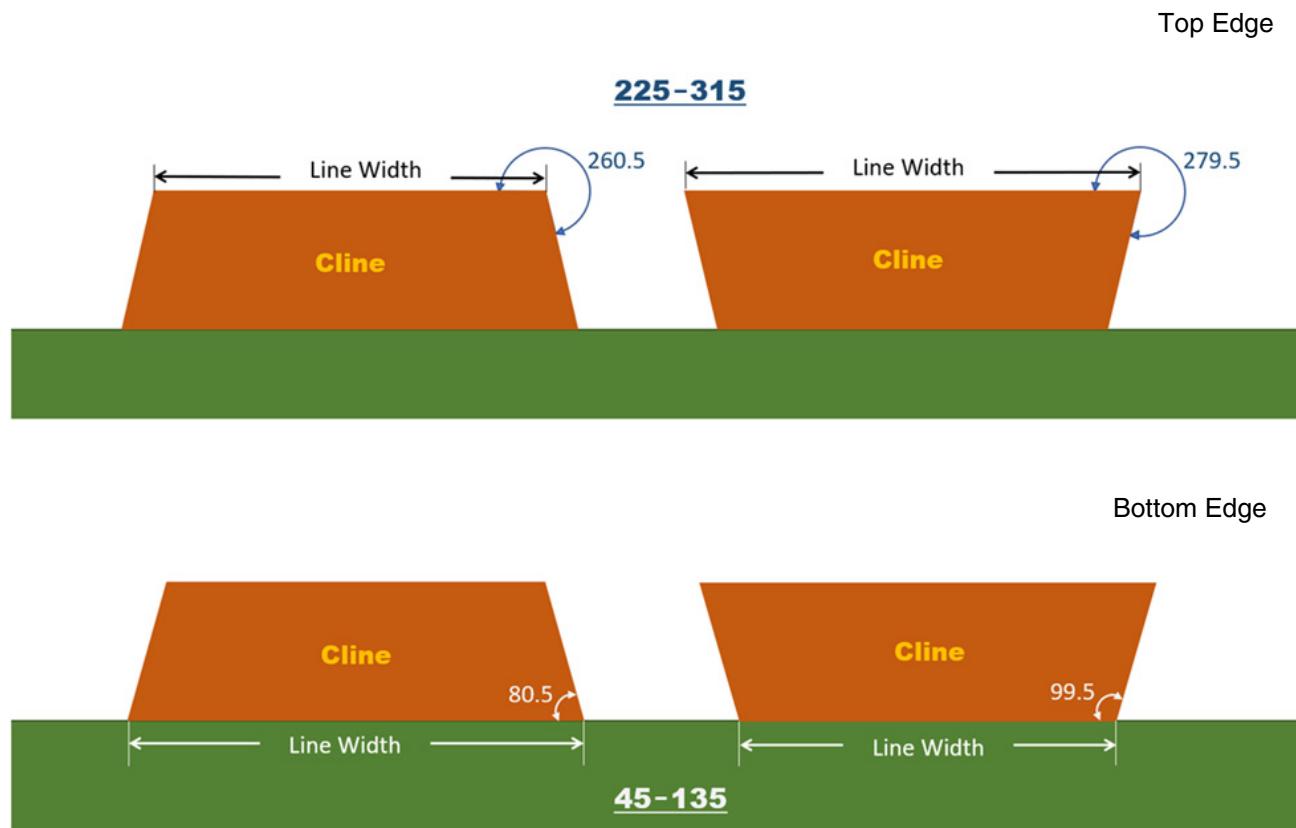
Type	Material	Thickness (μm)	Conductivity (mho/cm)	Dielectric Constant	Loss Tangent	Freq Dep File	Negative Artwork	Shield	Width (μm)	Etch Factor (degrees)	Impedance (ohm)
SURFACE	AIR			1	0						
DUCTOR	COPPER	36.576	595900	1	0				150.00	90	50.589
ELECTRIC	FR-4	101.6		0	5.2						
LAYER	COPPER	30.48	595900	5.2	0						90
ELECTRIC	FR-4	203.2		0	4.5						
LAYER	COPPER	36.576	595900	4.5	0						90
ELECTRIC	FR-4	250		0	5.2						
DUCTOR	COPPER	36.576	595900	5.2	0				100.00	90	60.016
ELECTRIC	FR-4	25.4		0	5.2						
DUCTOR	COPPER	36.576	595900	5.2	0				100.00	90	60.016
ELECTRIC	FR-4	250		0	5.2						
LAYER	COPPER	36.576	595900	5.2	0						90
ELECTRIC	FR-4	250		0	5.2						

Top Edge or Bottom Edge Specification

Etch factoring lets you set line width for either the top edge or the bottom edge of the cline. You do this by selecting a value within one of the two valid ranges of values 45-135 or 225-315.

When you set a value for the top or bottom edge, angles less than 180 degrees (45-135) indicate that the bottom edge of the cline is defined as the line width. Angles more than 180 degrees (225-315) indicate that the top edge of the cline is defined as the line width. This is depicted in Figure 11-8 where the etch factor value for the cline for the top etch is set at 260.5 degrees and 279.5 degrees, implying that the Top edge is defined as the line width. The etch factor value for the cline for the bottom edge is set at 80.5 degrees and 99.5 degrees, implying that the Bottom edge is defined as the line width.

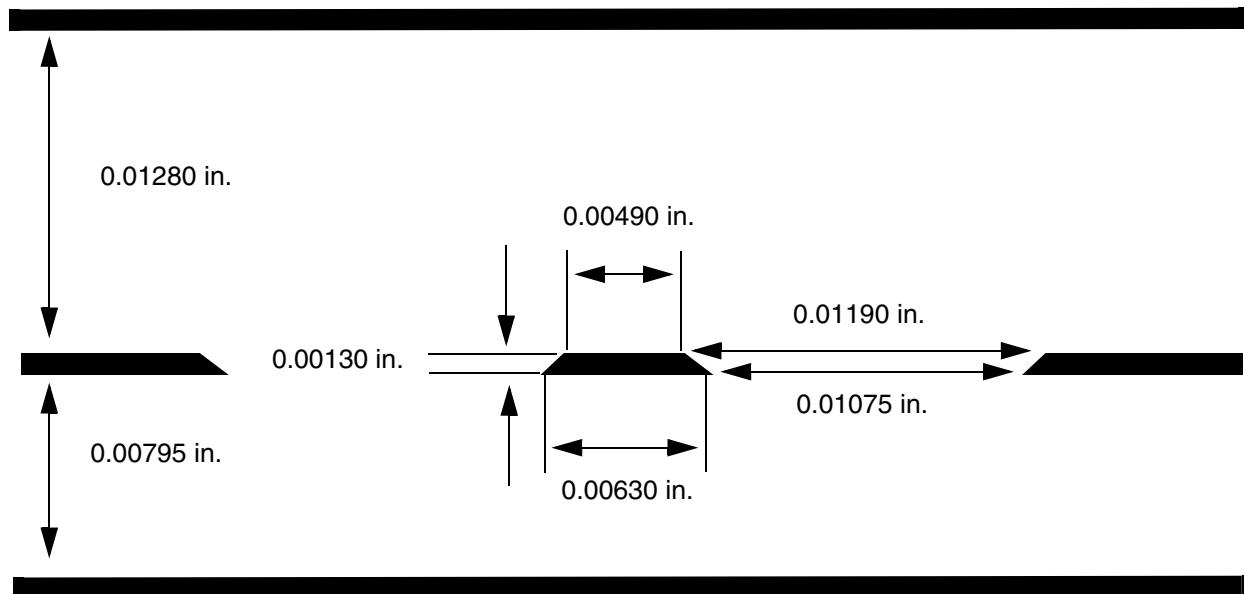
Figure 11-8 Top Edge and Bottom Edge Line Widths



Coplanar Waveguide Support

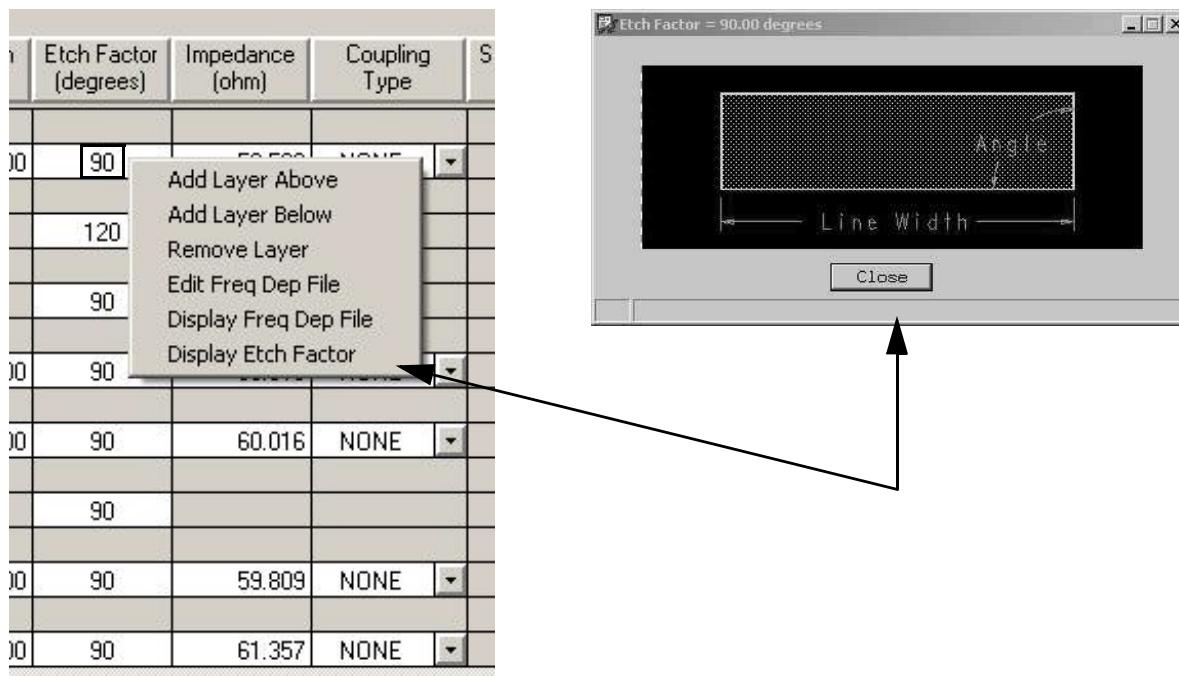
If you specify an etch angle factor for a cline within a CPW structure, EMS2D automatically applies the specified angle value to the bottom edges of the surrounding ground shapes when the angle is something other than 90 degrees. Figure 11-9 illustrates how the spacing between the cline and ground shapes are defined.

Figure 11-9 Etch Factor Cline/Shape Effect on a CPW Structure



To help you better determine the proper etch factor settings, you can display a graphical representation of the cline by clicking the right mouse button on the Etch Factor field of interest and selecting *Display Etch Factor* from the pop-up menu, as shown in Figure 11-10.

Figure 11-10 Etch Factor Display Window for 90-Degree Angle



Based on the visual feedback, you can then adjust your values as required.

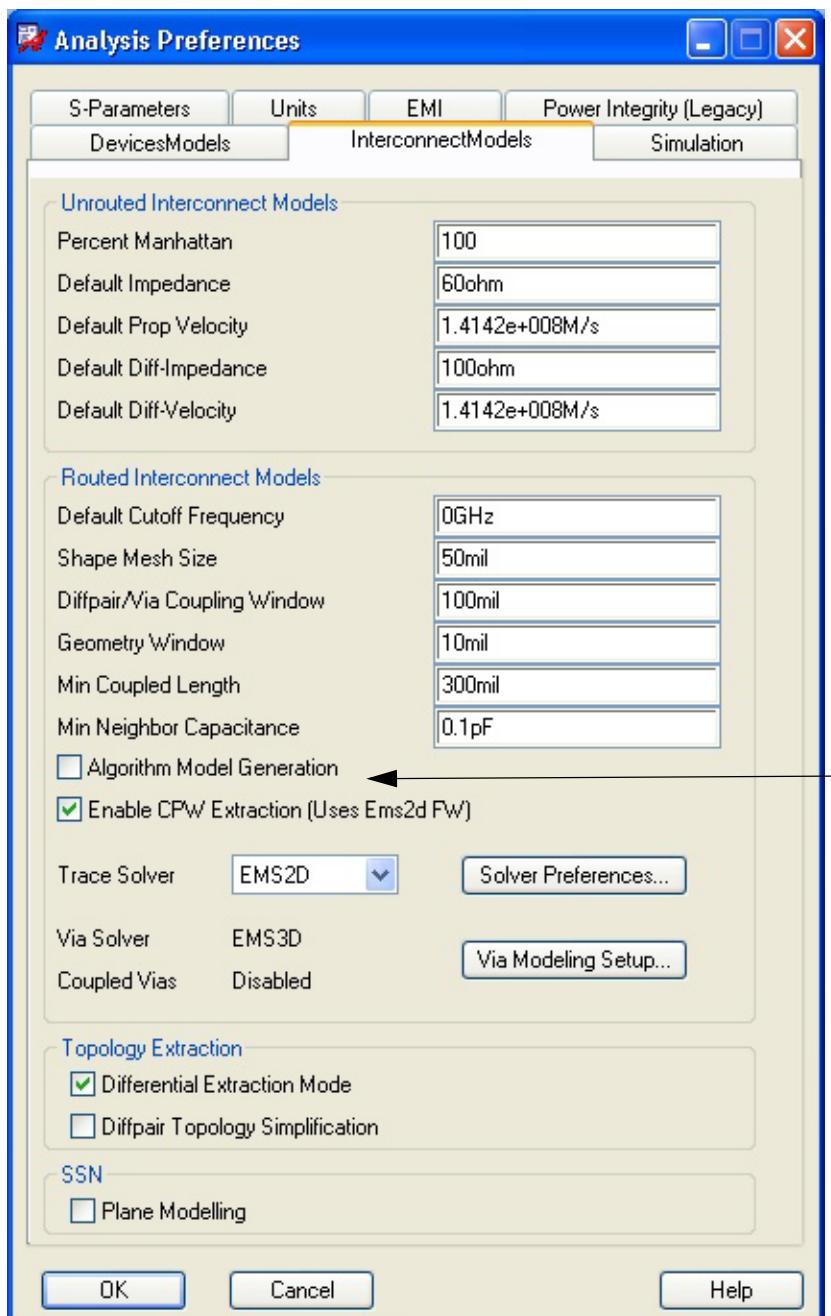
Algorithm-Based Modeling

Algorithm-based interconnect models (ABIML) are designed to greatly enhance simulation times when interconnect models that match simulation criteria cannot be found in existing traditional models. Algorithm model generation lets you create accurate interconnect models off-line that exactly match not only shield, dielectric, trace and physical geometry layer information but also entire frequency spectrums. These models are then integrated into libraries for reuse in multiple simulations.

Algorithm-based modeling is optional. You can enable/disable it from the InterconnectModels tab of the Analysis Preferences dialog box in Allegro PCB SI or the Simulation Parameters tab in SigXplorer.

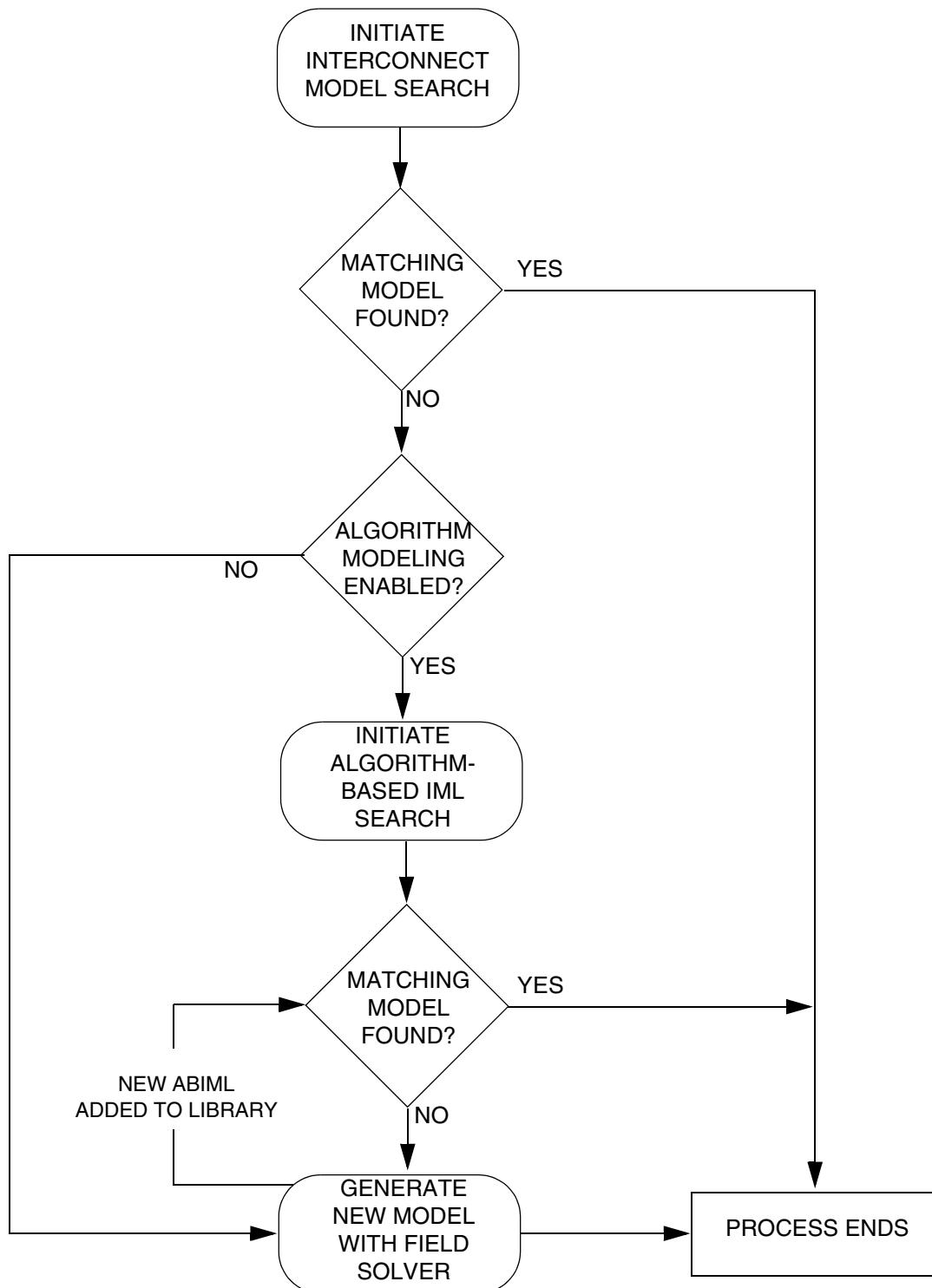
Allegro PCB SI User Guide
Dynamic Analysis with the EMS2D Full Wave Field Solver

Figure 11-11 ABIML Control in PCB SI Analysis Preferences



In both products, the default condition is On. If you turn off algorithm modeling, your Allegro tools will not search for algorithm-based models. Instead, it will directly engage the field solver to create the required model. This process is illustrated in the flow chart below.

Figure 11-12 Interconnect Library Search/Create Flow with ABIML Enabled



Allegro PCB SI User Guide

Dynamic Analysis with the EMS2D Full Wave Field Solver

Algorithm-based models can be cloned and/or edited in the same fashion as other interconnect models. The syntax of the model contains two sections:

- Model information such as parameter range, interpolation type, and sweep step type
- Multiple RLGC data used in model generation

The following is a simple example of the file format.

```
[Model] abiml_test
[Model Info]
    [Field_Solver_Used] ems2d
    [ABIML_Version] 1.0
    [Model_Type] sinlgetrace
    [Num_of_Port] 2
    [Num_of_DielectricLayer] 1
    [Num_of_ShieldLayer] 1

    [Parameter Info]
        [LayerStack]
            [Layer] 1
                *           min     max     step   step_type   interp_type   ID
                [Thickness] 1.0     1.0
                [Constant]  1.0     1.0
                [Losstangent] 0.0    0.0
                [IsShield]    YES

            [Layer] 2
                *           min     max     step   step_type   interp_type   ID
                [Thickness] 1.0     2.0     5      linear      linear       1
                [Constant]  4.4     4.6     3      log         2_order_poly 2
                [Losstangent] 0.0    0.0
                [IsShield]    NO

            [End LayerStack]
        [CrossSection]
            [conductor] 1
                *           min     max     step   step_type   interp_type   ID
                [Thickness] 1.0     1.5     5      linear      linear       3
                [Width]      1.0     10.0    20     linear      2_order_poly 4
                [Losstangent] 0.0    0.0

            [end CrossSection]
    [End Model Info]
```

Allegro PCB SI User Guide

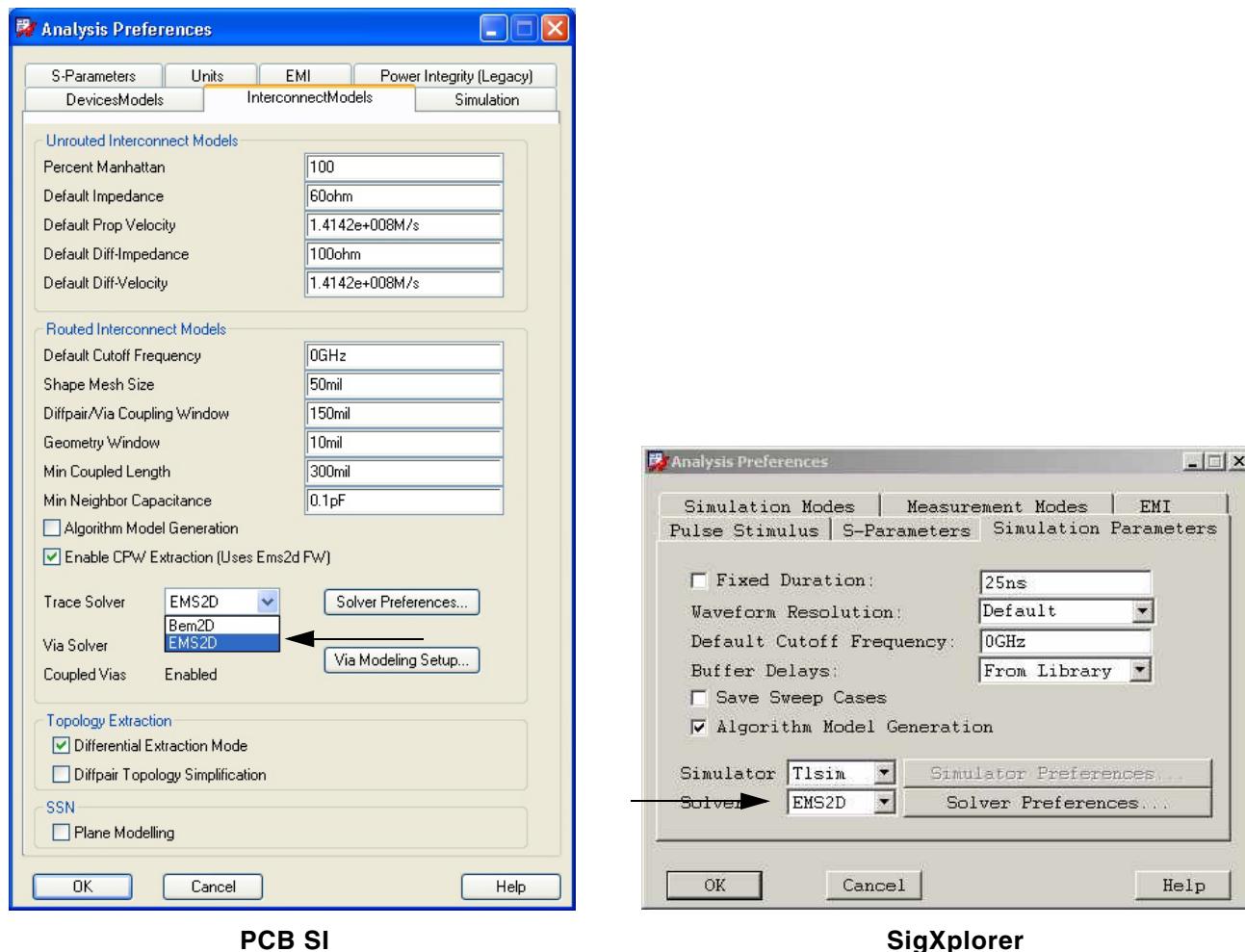
Dynamic Analysis with the EMS2D Full Wave Field Solver

```
[Model Data]
[Data] 1
[R] 1.459500e+01
[L] 6.088700e-07
[G] 0.000000e+00
[C] 5.162900e-11
[Data Condition]
*
      ID      value
      1       1.0
      2       4.4
      3       1.0
      4       1.0
.....
[Data] 4725
[R] 3.630000e+00
[L] 4.567500e-07
[G] 0.000000e+00
[C] 7.440100e-11
[Data Condition]
*
      ID      value
      1       2.0
      2       4.6
      3       1.5
      4      10.0
[End Model Data]
[End Model]
```

Using EMS2D

You run EMS2D from the Analysis Preferences forms in Allegro PCB SI and SigXplorer.

Figure 11-13 EMS2D Access in PCB SI and SigXplorer



For specific information on how to run EMS2D from these tools, see the online documentation accessed from the Help buttons on the forms.

Constraint-Driven Layout

Introduction

The Allegro system interconnect platform stores a common set of constraints directly in the design database. Once constraints are assigned or inherited by design elements, they are adhered to by all tools across the entire design flow.

SI analysis and constraint-driven layout helps you create more robust designs by optimizing your design with respect to timing and noise. This method reduces place-route-verify iterations and ultimately accelerates your time to market.

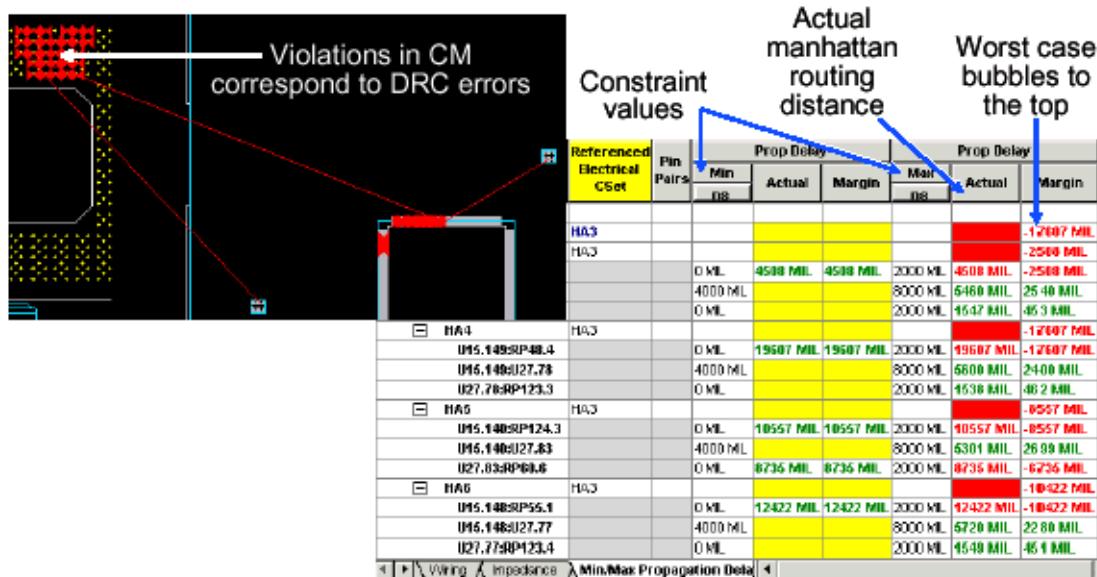
Constraint-Driven Placement

Once you have electrical constraints (ECSets) applied to your nets, you can begin the task of constraint-driven placement of your components. The SI engineer and the PCB layout designer both can perform component placement. In each case, delay (length) constraints must be met, or DRC (design rule check) errors are produced and displayed in the SI design window. DRCs are identified by a bow tie marker.

You can use Constraint Manager in conjunction with SI (see [Figure A-1](#) on page 408) to help identify components in violation and to assist in guiding their relocation.

For further details on relocating components, see the [move](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

Figure A-1 Delay Violations in SI and Constraint Manager



Placement Stages

Constraint-driven placement is typically comprised of three stages.

- Preliminary Placement
- Routability Analysis
- 90% Placement

Preliminary Placement

In the preliminary placement stage, all devices must be placed within the PCB outline, staying clear of keepouts and mounting holes, to validate that the PCB size and shape is sufficient. Critical component placement is driven by constraints and must be handled accordingly. If all devices do not fit on top and bottom, you need to revise the mechanical assumptions.

Routability Analysis

With the netlist loaded and the ratsnest on, the placement (positions and orientations) is adjusted to simplify the route process by studying the basic flow and crossing of signals. Powerplanes and copper areas need to be studied along with decoupling capacitors and their placement.

90% Placement

Constraint-driven placement is complete and nearly all placements are locked in. However, some freedom to nudge components is desirable during this stage to solve congestion problems that were not anticipated during the previous stage.

Constraint-Driven Routing

Constraint-driven routing does not suggest that an SI engineer route an entire board using PCB SI. However, it may be important for you to route critical nets for analysis of actual routed traces. These routes use the actual board stackup and include vias; items that were not available when you created the topology file.

For further details on manually routing critical nets, see the [add connect](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

Once routed, analyze a critical net by extracting its topology into SigXplorer. The topology includes the actual routed trace models and via models from the board. You can then execute a series of simulations based upon the extracted parameters and, if necessary, modify the topology file that contains the constraints (thereby modifying the Electrical CSet). You can then re-apply the Electrical CSet to the net and embed the constraint changes in all related nets.

For further details on extracting a routed trace into SigXplorer, see the [signal probe](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

Routing Stages

Constraint-driven routing is typically comprised of three stages.

- Route Critical Nets
- Route Sensitive Nets
- Route Remaining Nets

Route Critical Nets

Regardless of the PCB being interactively or automatically routed, there are usually some nets that need more attention than others. These nets may have certain restrictions defined by the design team. You must document these restriction in some way to provide clarity. Nets in this class may even require special widths or gaps between them and adjacent traces.

Route Sensitive Nets

Nets in this category are not critical but may be susceptible to certain electrical effects such as coupling. Once all critical nets are routed, you should then route sensitive nets.

Route Remaining Nets

This stage involves finishing all routes, testpoints, powerplanes, and any other special requirements. Design rule checks (DRCs) are normally flagged throughout this stage for physical and electrical constraint violations.

Constraints that Affect Routing

The constraints you set in PCB SI are passed to PCB Router through the design file (.dsn). Examples of constraints that are passed are:

- PROPAGATION_DELAY_RULE
- RELATIVE_PROPAGATION_DELAY
- Net Scheduling with T-points
- Parallelism
- Differential Pair
- Max Stub Length
- Max Via Count
- All Spacing Rules (wire to wire, wire to via, and so forth)
- Route Priority

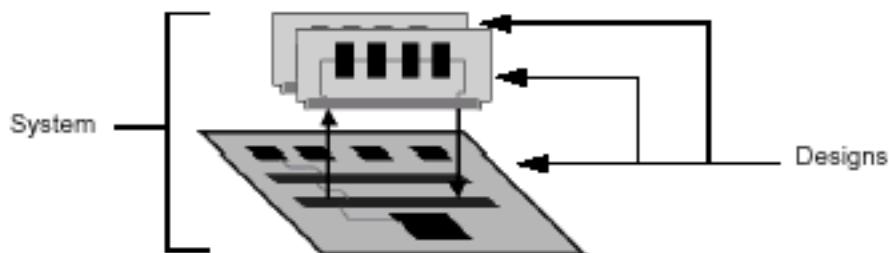
Allegro PCB SI User Guide

Constraint-Driven Layout

System-Level Analysis

Introduction

A *design* is a board (.brd) or MCM (.mcm) file in PCB SI and the PCB Editor. A *system* consists of all participating *designs*, along with the interconnecting cables and connectors. For example, a system may consist of a motherboard, a power supply, a cooling fan, and several plug-in cards such as a video controller, a sound card, and RAM modules.



With system-level simulation, you analyze an extended net (system-level Xnet) that spans more than one design. The simulation takes into account the path through all the separate designs in the system and the connectors and cables that connect these designs. Separate designs of various types that constitute a system are called a *design link* and may include board (.brd), package (.mcm) and system-in-package (.sip) designs. Design links as described above should not be confused with *DesignLink*, a type of signal model keyword within a DML file that specifies both a set of connections and the other designs to which you make connections.

What is a System Configuration?

A system configuration file (.scf) is a database representation of all the participating designs, including interconnecting cables and connectors, that comprise the system. The system configuration also includes the Xnets and pin-pairs that traverse a system and their assigned constraint values.

A system configuration represents the electrical characterization of a system. For example, a motherboard may have four slots for memory modules. If two of these slots are populated, then that is considered a system configuration. If four of these slots are populated, then that is considered an entirely different system configuration.

As another example, if only the first memory slot is populated, moving the memory module to the fourth slot is considered a change in the system configuration because the trace to the fourth slot is further from the signal source than the trace to the first slot.

For each permutation of memory modules (1 through 4, populated or vacant) there is a different system configuration. You can constrain each of these unique configurations as a different system configuration database. A final example is swapping out the cable that connects one design to another. If it is swapped for another cable, perhaps a longer one, then this is considered a change in the system configuration.

Eventually the design or board file (`.brd`) is constrained such that any system configuration meets the system level constraint requirements.

With a system configuration database, you can:

- switch between various system configurations without loss of information.
- use a single system configuration database for multiple participating design databases.
- use Constraint Manager to manage system-level constraints.
- preserve system-level constraints, system pin-pairs, and system Xnets from one project session to the next.

What is a DesignLink?

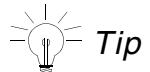
To simulate a system of designs, you must specify the designs that comprise the system (the design links) as well as information regarding how to connect the designs together. To do this, you use a *DesignLink*. A DesignLink is a type of signal model within a DML file that specifies both a set of connections and the other designs to which you make connections. Each system configuration file is established (*seeded*) from a DesignLink.

Once the system configuration is established (or activated), it is saved to a system configuration database file (`.scf`) which then maintains the system connectivity in the database format. At this point, the DesignLink is stored in the `.scf` file and no longer used.

What is a Cable Model?

A cable model represents the parasitics of the cable running between multiple designs. It is similar to a PackageModel; both contain RLGC matrices (or circuit models). However, you insert a cable model into a DesignLink whereas a PackageModel is inserted into an IbisDevice model.

Note: Cable models are very rarely used. You cannot generate them using PCB SI; you must obtain from external sources.-



Tip

Rather than attempting to build a cable from scratch, Cadence recommends that you create a new cable model by cloning (and then editing) an existing cable model from the sample library to characterize the cable that you need to model.

Modeling Strategies

A typical system configuration might consist of two printed circuit boards, PCB1 and PCB2, two mated connectors and a cable. A connector is typically made up of a plug and a receptacle. In this scenario, one receptacle is mounted on PCB1, the other receptacle is mounted on PCB2, and the two plugs are mounted on the ends of the cable.

The following approach was taken to model this multi-board system:

- A PackageModel is used to represent the parasitics of a mated connector (plug and receptacle).
You assign a PackageModel on the board and include the parasitics of both the receptacle and the plug.
- An IbisDevice model is generated for each connector on the board, then a PackageModel is inserted into each IbisDevice model.
In this scenario, identical connectors are used on both PCBs.
- An RLGC model is used to represent the parasitics of the cable running between the two boards.
- A DesignLink model is generated to specify the system-level connections between the two boards.
The RLGC model is inserted into the DesignLink model.

Allegro PCB SI User Guide

System-Level Analysis

Use the guidelines in the following table when employing a modeling strategy.

Modeling Common Multi-Board Configurations

To model . . .	Do this . . .
PCBs connected through a cable, with a connector on the PCBs at each end of the cable	Represent the connectors on the PCBs as a PackageModel within an IBIS device model. Represent the cable as an RLGC model within a DesignLink model.
daughter boards with gold fingers plugged into a card-edge connector on a backplane	Represent the connector as an IBIS device with package parasitic information.
an MCM in a Pin-Grid-Array (PGA) package plugged into a PCB	Model the pins of the PGA in an IBIS device with package parasitic information.

Working with System Configurations

New System Configurations

A new system configuration is established with a DesignLink model. It specifies both the participating designs and how they are connected.

To establish a new system configuration

1. From PCB SI, choose *Analyze – Initialize*

The Signal Analysis Initialization dialog box appears.

2. In the Signal Analysis Initialization dialog box, click *New DesignLink*.

A dialog box appears for you to name the new DesignLink.

3. Enter a name for the DesignLink.

4. Click *Edit*.

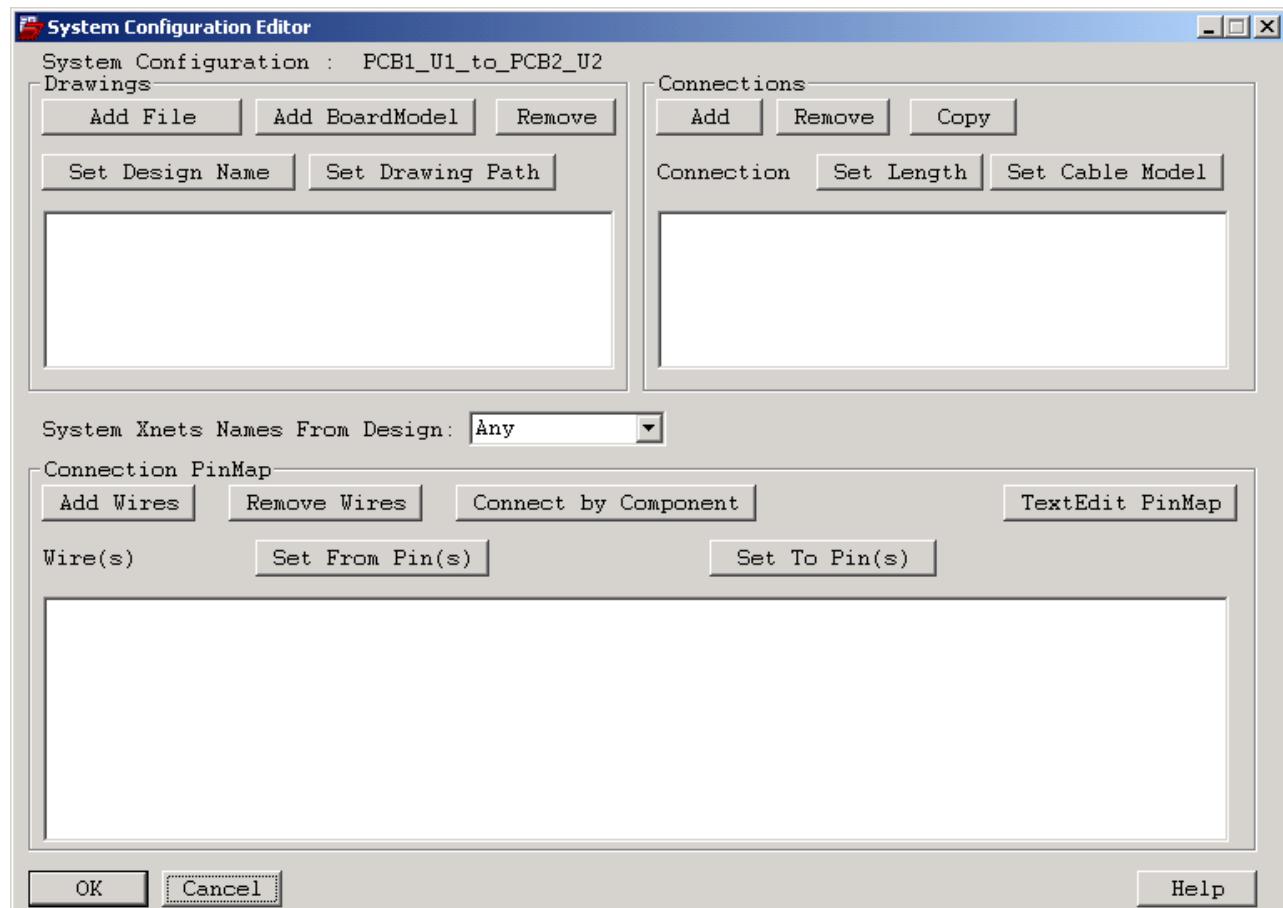
The System Configuration Editor appears with all fields empty as shown in [Figure B-1](#) on page 418.

5. Complete the system configuration. For details on the dialog box options, see [System Configuration Editor Controls](#) on page 419.

6. Once the DesignLink model is established, you can use it to create a new system configuration. See [“Existing System Configurations”](#) on page 420 for more information.

Allegro PCB SI User Guide
System-Level Analysis

Figure B-1 System Configuration Editor dialog box



System Configuration Editor Controls

Table B-1 Drawings Area Buttons

Click this button	To perform this operation
<i>Add File</i>	Add a board file (design) to the system configuration
<i>Add BoardModel</i>	Add a BoardModel to the system configuration (you can represent a design in the abstract as an electrical BoardModel)
<i>Remove</i>	Remove a design from the system configuration
<i>Set Design Name</i>	Add a label to the selected design name
<i>Set Drawing Path</i>	Specify a path to the selected design name. You can, in effect, instantiate the same design many times with reference to a design (.brd or .mcm file). Give each derivative a unique label. For example, you can instantiate a memory module of the same design in a system.

Table B-2 Connection Area Buttons

Click this button	To perform this operation
<i>Add</i>	Name a connection between participating designs in the system configuration.
<i>Remove</i>	Remove a connection between participating designs in the system configuration.
<i>Copy</i>	Clone an existing connection between participating designs in the system configuration.
<i>Set Length</i>	Specify a cable length in meters. Use zero for plug-in boards.
<i>Set Cable Model</i>	Specify a cable model from the library.

Table B-3 Connection Pinmap Area Buttons

Click this button	To perform this operation
<i>Add</i>	Add new pin connections to the selected cable connection. Presents a series of dialog boxes to collect information on the first wire number, the number of wires, and the starting pin (at either end of the connections).
<i>Remove</i>	Remove the selected pin connections from the selected cable connection.
<i>Set From Pins</i>	Presents a dialog box for you to select the starting pin at one end of the cable connection
<i>Set to Pins</i>	Presents a dialog box for you to edit the starting pin at the other end of cable connection
<i>TextEdit PinMap</i>	Opens the entire pin connection map in a text editor for further manipulation.
<i>Connect by Component</i>	Opens a dialog box that enables you to select two components on specific designs to quickly form the pin to pin connections. A pin to pin connection is established between each pin on one component to a pin on the other component with the same pin number.

Note: The system configuration name is saved as a property (SYS_CONFIG_NAME) within the board database. If the board is opened again in PCB SI, the system configuration is auto-started in single-board mode.

Existing System Configurations

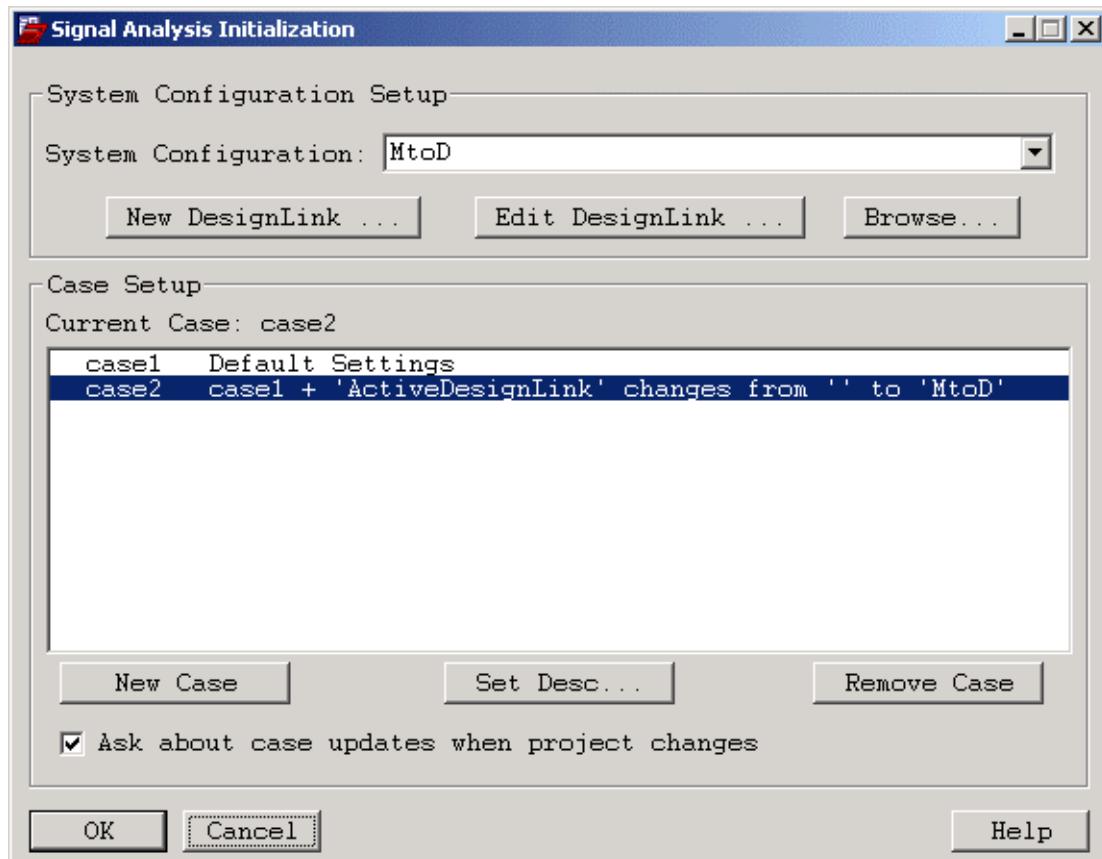
You can choose among existing system configurations using the Signal Analysis Initialization dialog box.

To choose an existing system configuration

1. From PCB SI, choose *Analyze – Initialize*.

The Signal Analysis dialog box appears as shown in the following figure.

Figure B-2 Multiple System Configurations



2. Click the *System Configuration* down-arrow to display a menu that shows all available system configuration (.scf) files as well as all available DesignLink models that use the current board (.brd) file.

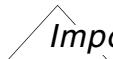
Note: Of the available system configurations there is the *Single Board System*. This is the default system configuration. Single board system uses a single design or board file (without other participating designs). A system configuration database file is *not* produced when choosing *Single Board System*.

3. Choose a system configuration and click *OK*.

The system configuration is checked. If there are any errors with the chosen system configuration, the previous system configuration is restored.

When a DesignLink model is chosen (and activated by clicking *OK*), with only one participating design, the DesignLink is ignored and the previous system configuration is restored. If the DesignLink model contains multiple participating designs, a new system

configuration is created (with the same name as the DesignLink model). You can still clone DesignLink model or use it to seed a different system configuration.



The only way to create a new system configuration is to create a new DesignLink model (either from scratch or by cloning an existing DesignLink model), and then choosing the DesignLink model, as a system configuration, from the pull-down menu in the Signal Analysis Initialization dialog box (see [“Multiple System Configurations” on page 421](#)).

The system configuration path is determined by the environment variable `SCFPATH` which is, by default, set to the current working directory.

To modifying an existing system configuration

1. Select a system configuration from the *System Configuration* drop-down menu in the System Analysis Initialization dialog box (see [“Multiple System Configurations” on page 421](#)).
2. Click *Edit System Configuration*.

Note: You must activate the system configuration before you edit it.

3. Make changes to the selected system configuration through the System Configuration Editor as described in the [System Configuration Editor Controls](#) on page 419.

Setting Constraints at the System Level

You can set a constraint directly on a system Xnet in your layout or you can define the constraint in Constraint Manager as an ECSet and then reference it to a system Xnet.

When a board is removed from a system configuration and accessed as a single board, it is desirable to maintain the constraints that were set at the system-level. Therefore, when a constraint is set on a system Xnet, the relevant system constraints are copied to each net and design Xnet within the system Xnet. The source of these design-level constraints are marked as system-level for the purpose of deleting them when the board is included in a different system configuration. When the board is opened as a single board, the constraints that were copied from the system-level constraints remain in the design.

Electrical constraints are assigned to a system Xnet as follows.

- Each non pin-pair based constraint is copied to every net and design Xnet in the system Xnet as a property override unless the net or design Xnet already has a defined constraint value. In this case, the worst-case constraints from the net or Xnet and system Xnet is chosen.
- For pin-pair based constraints (Min First Switch, Max Final Settle, Propagation Delay, Relative Propagation Delay, Impedance), all auto-generated pin-pair rules are copied to each net and design Xnet within the system Xnet. Any user-defined pin-pair rules on a single design are copied and flattened to the design that it references. Those pin pairs that reference pins on different designs are saved as system pin-pairs.

Most of the constraints that are set on system Xnets are copied to the member nets and design Xnets, and are shown in a Constraint Manager worksheet on both system and design Xnets. The actual values displayed by the worksheet correspond to the object. For example, the net level Max Vias constraint show an actual value that is the total number of vias in the net. The actual value displayed for a System Xnet is the total number of vias in the system Xnet.

Note: For any design Xnet that is part of a bus or a differential pair object, its system Xnet is also be part of that bus or differential pair. You do not have to define same bus or differential pair objects in all designs.

System-Level Simulation

After you have created your system configuration, you can proceed to simulate nets that span multiple designs. When you model for multi-board analysis, consider these basic concepts:

- The simulator automatically models the parasitics for interconnect structures produced during routing (for example, traces and vias).
- Interconnect structures that are not created during routing (for example, packages, connectors, and cables), are represented by RLGC matrices or circuit models, either directly in an RLGC model (as in a DesignLink model) or in a PackageModel (as in a IBIS device model).

Working with Crosstalk

Your Allegro tools support two types of crosstalk (xtalk) checking: estimated and simulated.

- Estimated crosstalk is based on pre-computed tables of crosstalk data. This type of crosstalk data is used for crosstalk design rule checks (DRCs) and in SI Segment Crosstalk reports.
- Simulated crosstalk is similar to estimated crosstalk but is more accurate, thus taking a longer time to generate. This type of crosstalk is used in SI Crosstalk Summary and Crosstalk Detailed reports.

Crosstalk values are calculated from tables of crosstalk data that are stored in the database of your design. These tables are built by capturing the time domain simulation data produced by sweeping multiple crosstalk scenarios derived from the Allegro database. Data is captured for each driver model in the design, multiple impedance values, line-to-line spacing, layer-to-layer combinations, and simulation mode (Fast, Typical, or Slow). If one of these tables is exported to a file, it is formatted as shown below.

```
.data
CDSDefaultOutput 0.000127 5 5 Typical 1.50904 0.127278 0.0960336
CDSDefaultOutput 0.000127 2 2 Typical 1.57598 0.127278 0.100294
CDSDefaultOutput 0.000127 2 3 Typical 1.26156 0.127278 0.0802844
CDSDefaultOutput 0.000127 3 2 Typical 1.26156 0.127278 0.0802844
CDSDefaultOutput 0.000127 3 3 Typical 1.57598 0.127278 0.100294
CDSDefaultOutput 0.000127 0 0 Typical 1.50904 0.127278 0.0960336
CDSDefaultOutput 0.000254 5 5 Typical 0.71302 0.127278 0.0453759
CDSDefaultOutput 0.000254 2 2 Typical 0.553263 0.127278 0.0352091
CDSDefaultOutput 0.000254 2 3 Typical 0.510822 0.127278 0.0325082
CDSDefaultOutput 0.000254 3 2 Typical 0.510822 0.127278 0.0325082
CDSDefaultOutput 0.000254 3 3 Typical 0.553263 0.127278 0.0352091
CDSDefaultOutput 0.000254 0 0 Typical 0.71302 0.127278 0.0453759
CDSDefaultOutput 0.000508 5 5 Typical 0.270644 0.127278 0.0172235
CDSDefaultOutput 0.000508 2 2 Typical 0.0768282 0.127278 0.00488927
CDSDefaultOutput 0.000508 2 3 Typical 0.0759594 0.127278 0.00483398
```

Allegro PCB SI User Guide

Working with Crosstalk

```
CDSDefaultOutput 0.000508 3 2 Typical 0.0759594 0.127278 0.00483398
CDSDefaultOutput 0.000508 3 3 Typical 0.0768282 0.127278 0.00488927
CDSDefaultOutput 0.000508 0 0 Typical 0.270644 0.127278 0.0172235
CDSDefaultOutput 0.000762 5 5 Typical 0.143255 0.127278 0.0091166
CDSDefaultOutput 0.000762 2 2 Typical 0.0109948 0.127278 0.000699696
CDSDefaultOutput 0.000762 2 3 Typical 0.0109769 0.127278 0.000698556
CDSDefaultOutput 0.000762 3 2 Typical 0.0109769 0.127278 0.000698556
CDSDefaultOutput 0.000762 3 3 Typical 0.0109948 0.127278 0.000699696
CDSDefaultOutput 0.000762 0 0 Typical 0.143255 0.127278 0.0091166
.classes
.class CDSDefaultOutput
K9 K8 K7 K6 K5 K4 K3 K2 K10 K1 J9 J8 J7 J6 J5 J4 J3 J2 J10 J1
H9 H8 H3 H2 H10 H1 G9 G2 G10 G1 F9 F2 F10 F1 E10 E1 D10 D1 C9
C8 C3 C2 C10 C1 B9 B8 B7 B6 B5 B4 B3 B2 B10 B1 A9 A8 A7 A6 A5
A4 A3 A2 A10 A1
```

The columns in the table (reading left to right) are:

- IOCell
- Line-to-Line Gap
- Aggressor Layer
- Victim Layer
- Simulation Mode
- Line Impedance
- XtalkPerUnitLength
- Saturation Length

The noise units are in Volts and the length units are in meters.

Crosstalk DRCs

Crosstalk DRCs use the table-driven approach to quickly calculate the amount of coupled noise that you can transmit onto a victim signal. This functionality is leveraged through:

- crosstalk avoidance during interactive etch editing with PCB Editor.
- crosstalk avoidance during autorouting with PCB Router.
- crosstalk troubleshooting post-route.

For a specific coupling scenario on the board, crosstalk DRC looks up relevant entries in the table, interpolates if necessary, and calculates the amount of noise coupled from the aggressor onto the victim net. Crosstalk is calculated by estimating the distance between the aggressor net and the victim net within the dimensions of the geometry window. It checks the value against the constraint and produces a DRC error if the constraint is violated. You can also output actual values in signal integrity reports or in Constraint Manager.

There are 2 different types of crosstalk DRCs that you can use.

- MAX_PEAK_XTALK – limits the maximum amount of noise in millivolts that can be coupled over from any single aggressor.
- MAX_XTALK – limits the total (as defined by RSS summation of individual aggressor contributions) amount of noise in millivolts that can be coupled to a victim net from all aggressors.

Utilizing these DRCs is covered later in this appendix.

When you perform a crosstalk DRC in a design that does not contain a crosstalk table, an out-of-date state is set for the net you are checking. The check will be performed when you add a table later. If you update an existing table, existing DRCs are set to out-of-date and automatically recomputed.

Crosstalk Simulations

Crosstalk simulations extract full coupled-line circuits from the layout, walking the victim net and searching within the geometry window for aggressors. Two simulations are then run with specific stimuli applied. The default configuration is as follows.

- a driver on the victim is held low, fastest driver on each aggressor is stimulated with a rising edge, and the receiver pins on the victim are monitored for maximum voltage excursion from the low state.
- a driver on the victim is held high, the fastest driver on each aggressor is stimulated with a falling edge, and the receiver pins on the victim are monitored for maximum negative voltage excursion from the high state.

You can modify these defaults to stimulate each individual neighbor at a time, each driver on the aggressor, even mode rather than odd mode switching, and so forth. The defaults are typically a good choice for most applications.

Crosstalk Timing Windows

Crosstalk timing windows enable you to apply intelligent summing to crosstalk DRCs, and intelligent stimuli to neighboring Xnets for crosstalk simulations. With no crosstalk timing windows defined, PCB SI assumes that the victim is always sensitive to crosstalk and aggressors are always active. That is, crosstalk is always generated by all present neighbor Xnets.

This can be overly pessimistic in many cases, and can cause a very conservative layout, sacrificing density. You can define the following crosstalk timing window parameters to control which neighbors are to be considered as aggressors for crosstalk analysis.

- **XTALK_ACTIVE_TIME** – time slots in which aggressor Xnets can switch, and cause crosstalk (for example 1-5).
- **XTALK_SENSITIVE_TIME** – time slots in which victim Xnets are sensitive to crosstalk from aggressors (for example 3-7, 9-10).
- **XTALK_IGNORE_NETS** – nets for a victim to ignore as sources of crosstalk (can specify nets or Electrical Constraint Sets).

Consider the following example as a case in point.

Victim XNet “DATA1” has the attributes:

`XTALK_ACTIVE_TIME = 1-5, XTALK_SENSITIVE_TIME = 6-10`

Aggressor XNet “DATA2” has the attributes:

`XTALK_ACTIVE_TIME = 1-5, XTALK_SENSITIVE_TIME = 6-10`

Aggressor XNet “FSB_A5” has the attributes:

`XTALK_ACTIVE_TIME = 4-8, XTALK_SENSITIVE_TIME = 1-20`

When crosstalk analysis is run for the victim DATA1, its sensitive time (6-10) is compared with the active times for aggressors DATA2 (1-5) and FSB_A5 (4-8). Since DATA2’s active time *does not* overlap with the victim’s sensitive time, its crosstalk contributions are not considered during analysis. Since FSB_A5’s active time *does* overlap with the victim’s sensitive time (4-8 overlaps with 6-10), its crosstalk contributions are considered during analysis.

Therefore, crosstalk timing windows provide a simple means for controlling whether or not a signal (or a group of signals) is a valid source of crosstalk for another signal (or group of signals). Using this control wisely significantly decreases the amount of analysis required, eliminate false crosstalk DRCs, and improving routing density on the PCB.

Crosstalk Methodology

The following methodology is recommended to mitigate crosstalk and is broken down into the following sections.

- Database setup
- Crosstalk timing windows definition
- Crosstalk table generation
- Constraints and crosstalk-driven routing
- Post-route crosstalk simulation
- Crosstalk troubleshooting

Database Setup

To predict crosstalk accurately, the board database needs to be set up properly, as it is essentially an input to the analysis. There are a number of things to set up, all of which you can drive from the Database Setup Advisor.

For complete details on setting up a design, [Setting up the Design](#) on page 61.

Crosstalk Timing Window Definition

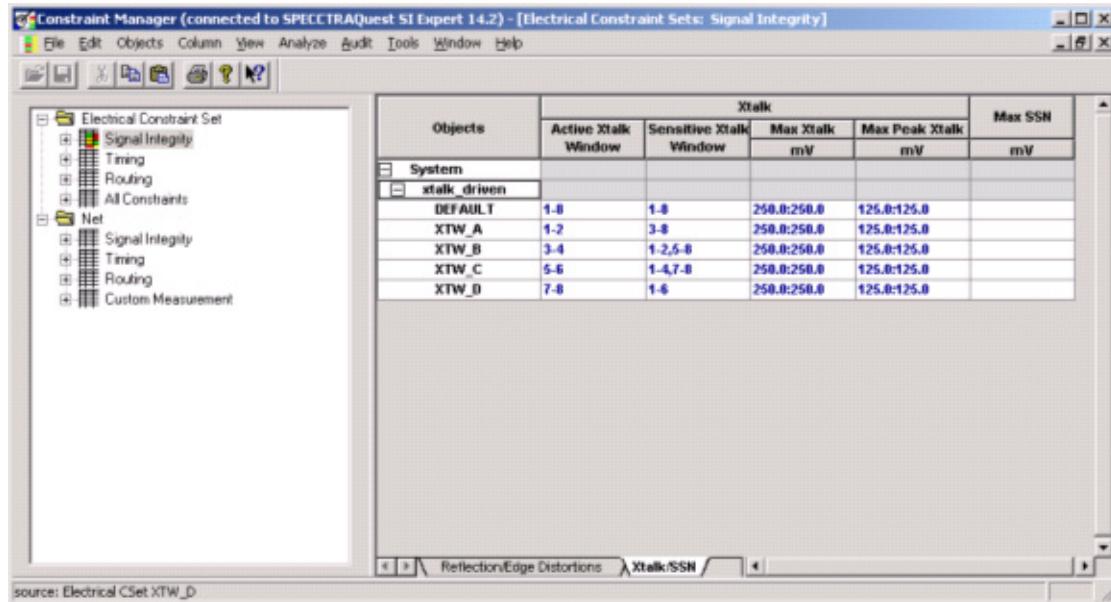
A good approach for applying Crosstalk Timing Windows to signals in the design is to group them by Electrical Constraint Set (ECSet) and define the Crosstalk_Active_Time and Crosstalk_Sensitive_Time ranges at the ECSet level. This allows it to be done quickly and has advantages downstream in the process that are described later in this appendix.

The goal is to identify groups of signals that will not be actively switching while other groups are being sampled (during their setup and hold window). The most common case of this is a synchronous bus, where intra-bus crosstalk has time to settle out before data is clocked in at the receivers. In this scenario, you consider only crosstalk coupled from signals outside the bus, and ignore bits within the bus as aggressors. This enables dense routing to be done. The following figure shows an example of how you set this up in Constraint Manager:

Allegro PCB SI User Guide

Working with Crosstalk

Figure C-1 Crosstalk Timing Windows Definition in Constraint Manager



In this example, there are 4 buses defined, XTW_A, B, C, and D. Each of these have been set up to ignore crosstalk from bits within their own bus.

Crosstalk Table Generation

Assuming the database is set up, there are two other parameters that must be set up before you can generate the crosstalk tables.

- Minimum line-to-line spacing
- Crosstalk geometry window

Note: Allegro platform products recognize different geometry window settings in multiple designs in a system configuration or design link. The result is a detailed crosstalk report that considers the different geometry window settings in each of the .brd and .mcm files. See [To set geometry windows at the drawing level](#) for details.

To set up the minimum line-to-line spacing

1. Choose *Setup – Constraints*.

The Constraints System Master dialog box appears.

2. Click Set Values in the *Spacing Rule Set* section.

The Spacing Rule Set dialog box appears.

3. Enter the desired value in the *Line to Line* text box, then click *OK*.

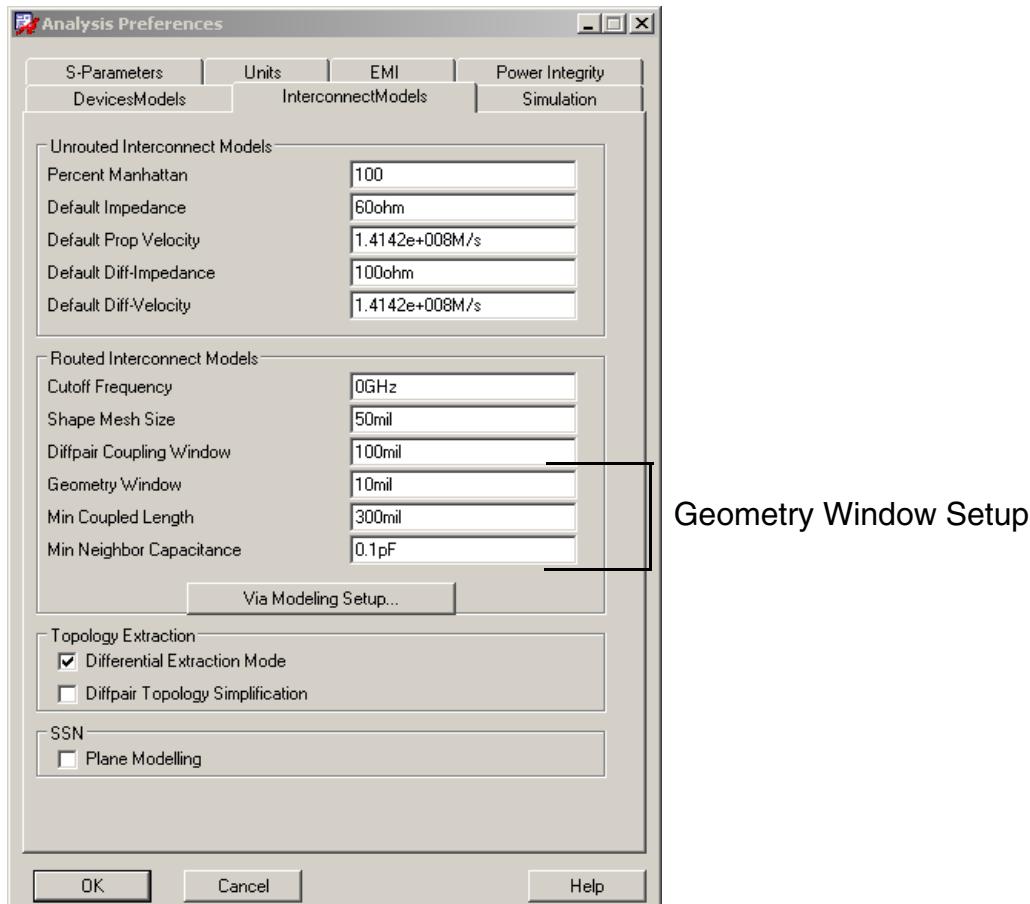
To set up the geometry crosstalk window

1. Choose *Analyze – Preferences*.

The Analysis Preferences dialog box appears.

2. Click on the *Interconnect Models* tab, enter the desired value in the *Geometry Window* text box, then click *OK*.

Figure C-2 Setting the Geometry Window in the Analysis Preferences Dialog Box



When crosstalk sweeps are run to generate the crosstalk tables, spacing is swept from the smallest line-to-line spacing value in the design to the geometry window value. A good value to use for the geometry window is about three times the dielectric thickness from the

conductor to its reference plane. For example, if 5 mil dielectrics are used in the stack-up, setting the geometry window for 16 mils would take into account aggressor nets up to three times away. This should encompass the majority of significant coupling. Once these parameters are set, you are ready to generate crosstalk tables.

To set geometry windows at the drawing level

1. Open a .brd or .mcm database file.

2. Choose *Analyze – Preferences*.

The Analysis Preferences dialog box appears.

3. Click on the *Interconnect Models* tab, enter the desired value in the *Geometry Window* text box, then click *OK*.

4. Save the database file.

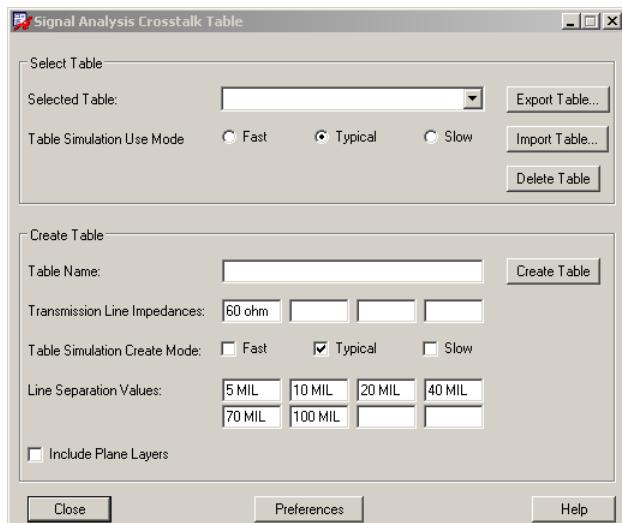
The file now contains a drawing-level GW value. This is the value that will be used for this package or board when a SI analysis is run for a system incorporating multiple .mcm and/or .brd files.

To generate the crosstalk tables

1. Choose *Analyze – Xtalk Table*.

The Signal Analysis Crosstalk Table dialog box appears as shown in the following figure.

Figure C-3 Generating Crosstalk Tables



2. In the Create Table section of the dialog, enter the name of the new crosstalk table.
3. Enter up to four transmission line impedances to use for generating the table.

Note: If your design contains impedance rules, four of the impedance values specified in the rules will be displayed as the default values in these fields. If no impedance rules are defined in the design, the impedance value that has been set in the *Default Impedance* field of the Analysis Preferences dialog box is the default selection.

4. Set the Simulation Create modes (*Fast/Typical/Slow*) to select the driver speeds you wish to use. You can set any or all of them.
5. Define up to eight line-to-line separation values for simulation. Default values for these fields are:
 - The smallest value of all the line-to-line spacing values defined in all of the spacing constraint sets
 - The geometry window

Note: If the difference between the above two values is very large, intermediate values are automatically created. If a line separation value is larger than the geometry window, the program generates an error message.

The following limitations apply when you are setting up separation values:

- You must define at least two values.
- One of the values must be the geometry window.
- You cannot enter a value greater than that of the geometry window.
- You cannot enter a value less than or equal to zero.
- Duplicate values will be deleted.

6. Select *Include Plane Layers* to add rows to the table that define crosstalk between lines on power and ground planes and other lines on either the same layer or on adjacent conductor layers.
7. Click *Create Table* to run the simulation, generate the table, and store it in the design database.

A generated message displays how many simulations will be run and provides you the opportunity to cancel the simulations.

Generating the crosstalk table enables you to use crosstalk DRCs as well as automatically seed the PCB Router rules .do file with the appropriate crosstalk data, enabling it to mitigate crosstalk during autorouting.

To use an existing crosstalk table

1. Choose *Analyze – Xtalk Table*.

The Signal Analysis Crosstalk Table dialog box appears

2. In the Select Table section of the dialog, choose from the Selected Table drop-down, one of the crosstalk tables stored in the design database.
3. Select a Simulation Use Mode value. You can select only one of the values that were specified when the table was created.

Exporting and Importing Crosstalk Tables

Exporting a crosstalk table from one drawing and importing it into another drawing lets you share the same crosstalk table across multiple drawings.

To export a crosstalk table

1. Choose *Analyze – Xtalk Table*.

The Signal Analysis Crosstalk Table dialog box appears

2. In the Select Table section of the dialog, choose from the Selected Table drop-down, one of the crosstalk tables stored in the design database.
3. Click *Export Table* to export the selected crosstalk table to a file. A default extension of .xtb is used for this type of file.

To import a crosstalk table

1. Choose *Analyze – Xtalk Table*.

The Signal Analysis Crosstalk Table dialog box appears

2. Click *Import Table* to import a crosstalk table into the design database. The table file is now available as a selection choice in the drop-down menu.

To delete a crosstalk table

1. Choose *Analyze – Xtalk Table*.

The Signal Analysis Crosstalk Table dialog box appears

2. Click *Delete Table* to delete the selected crosstalk table from the design database.

Constraints and Crosstalk-Driven Routing

You can perform crosstalk-driven routing either automatically, interactively, or some combination of both. To use crosstalk-driven routing, crosstalk constraints must be set.

To set crosstalk constraints

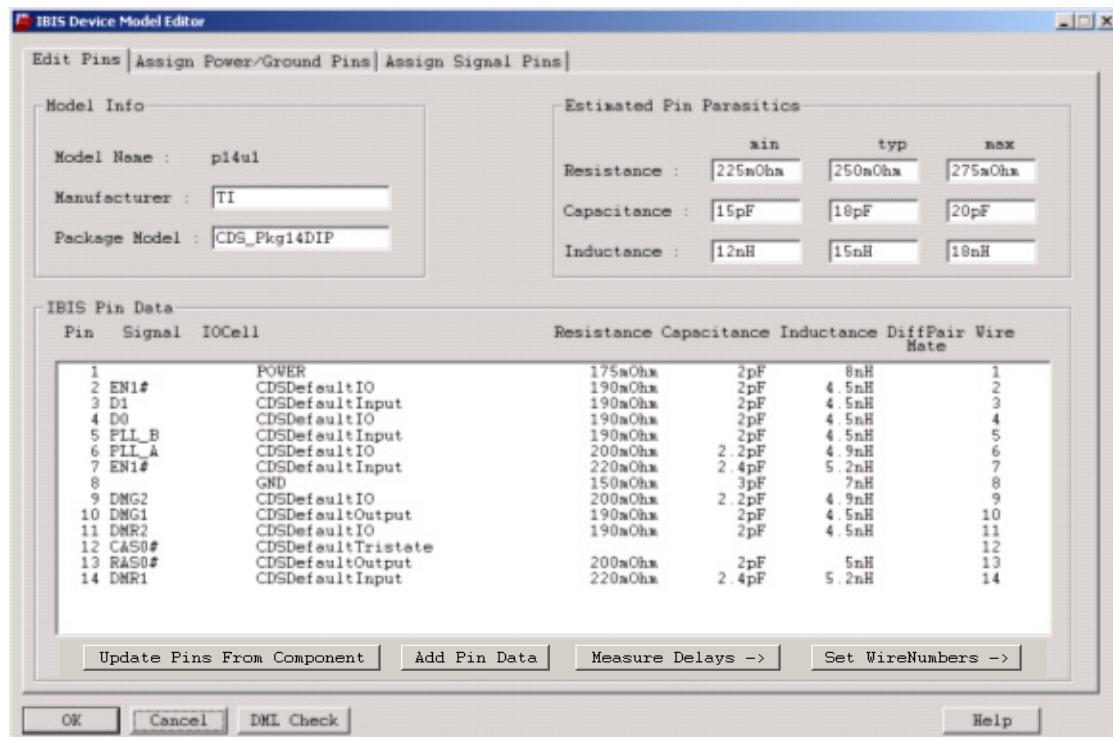
1. For a given class of signals, look at the IOCell model for the most sensitive receiver.
2. Choose *Analyze – Model Browser*.

The SI Model Browser appears.

3. Select the relevant IbisDevice model in the list and click *Edit*.

The IbisDevice Model Editor appears as displayed in the following figure.

Figure C-4 IBIS Device Model Editor



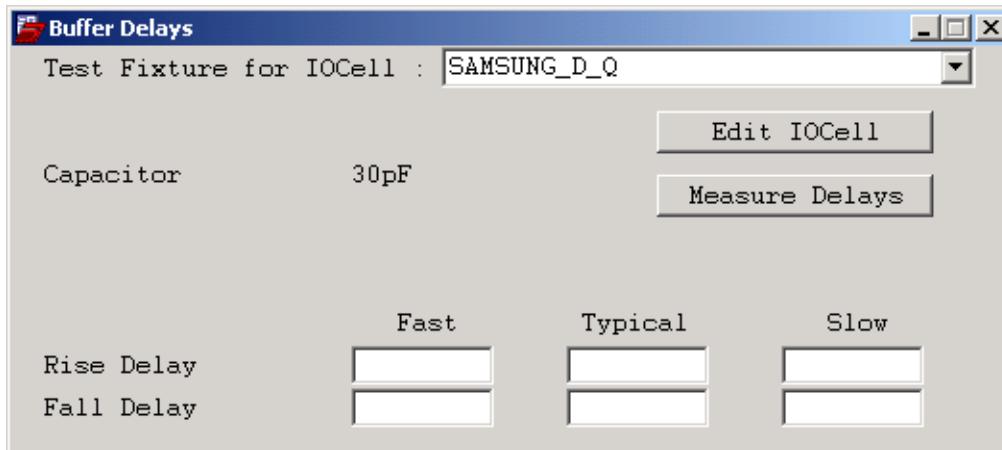
4. Click the most critical receiving signal on the part,

The Ibis Device Pin Data dialog box appears.

5. Click *Buffer Delays*.

The Buffer Delays dialog box appears as shown (top portion) in the following figure.

Figure C-5 Buffer Delays Dialog Box

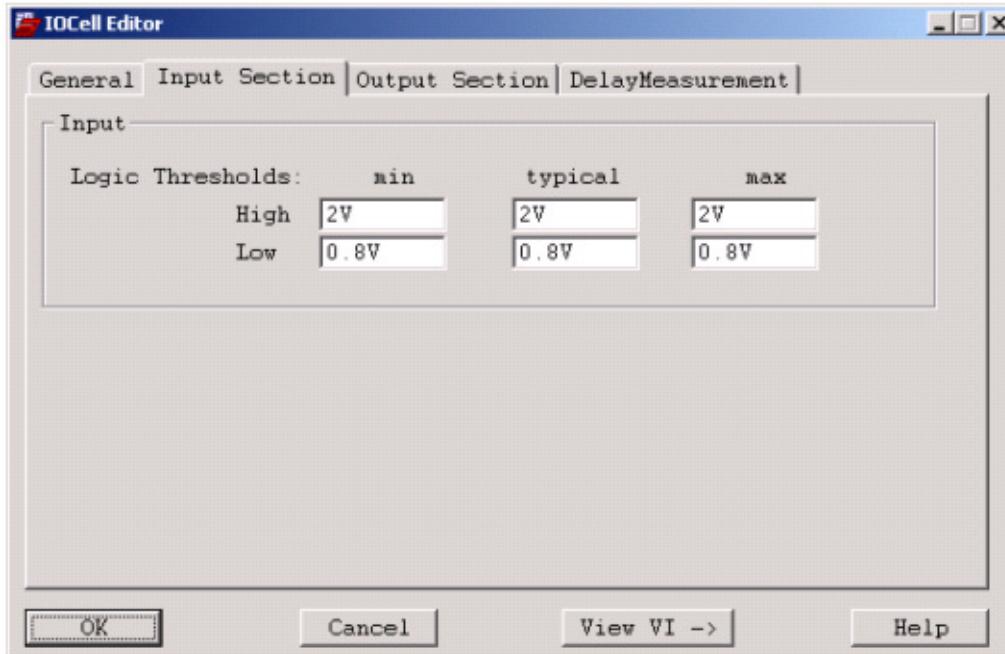


6. Click *Edit IOCell*.

The IOCell appears in the IOCell Editor.

7. Click on the *Input* tab and take a look at the input thresholds. See [Figure C-6](#) on page 436.

Figure C-6 Input Thresholds in IOCell Editor



In this example, we have thresholds of 2V and 0.8V (typical TTL levels) for a 3.3V part. That gives us inherent noise immunities of $3.3 - 2 = 1.3\text{V}$ in the high state and $0.8 - 0 = 800\text{mV}$ for the low state.

Crosstalk DRCs calculate only the amount of noise that initially gets coupled over from aggressors, and doesn't account for what happens afterwards. Coupled noise propagates around the victim net and reflect at impedance discontinuities (just like regular signals), and voltage doubling can occur. The actual simulated crosstalk results can often be significantly higher than DRC predictions. Because of this, it is good practice to always initially set `max_xtalk` constraints for < half the noise immunity. This can be relaxed later on when routing is completed and simulated results are available.

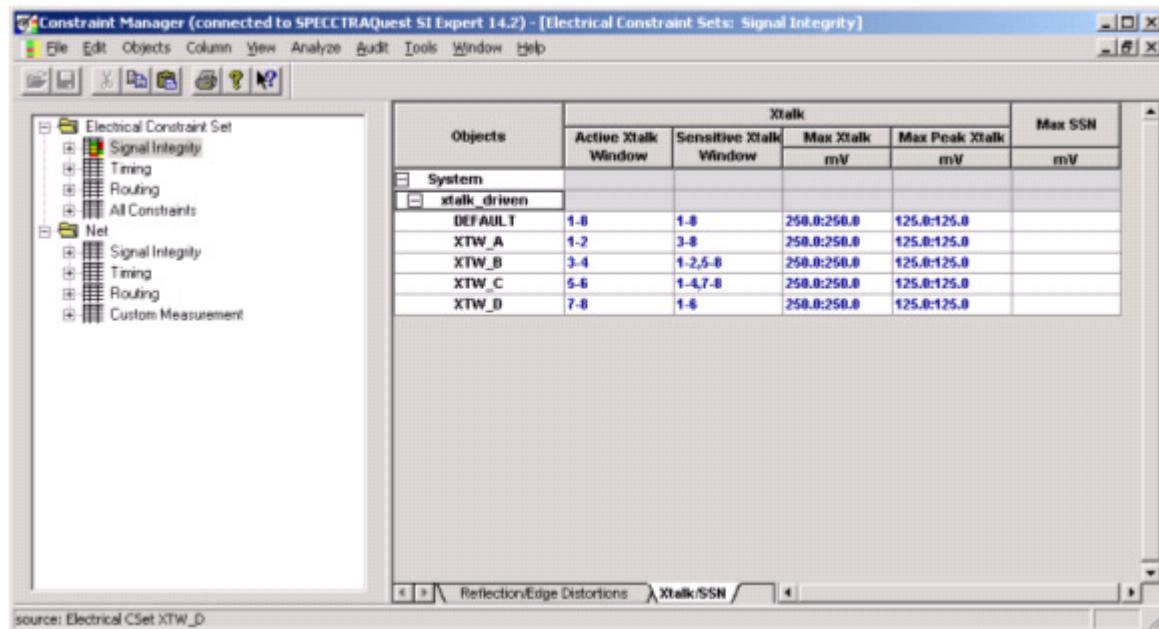
One way to derive an initial crosstalk budget (and a value for the `max_xtalk` constraint), is to take the inherent noise immunity, subtract the potential voltage ripple the IOCell could see, then divide the remaining noise margin in half. In our case of a 3.3V supply with a $\pm 5\%$ ripple tolerance, this would be $(0.8 - (.05 \cdot 3.3))/2 = 317.5\text{ mV}$. So setting `max_xtalk` for 250 or 300mV would be pretty realistic.

Setting `max_peak_xtalk` is a bit more subjective. One general rule of thumb is that you don't want any single aggressor to eat up more than half of your overall crosstalk budget. In this case, if we conservatively set `max_xtalk = 250mV`, we can set `max_peak_xtalk = 125mV`. These are just recommendations, and can be adjusted for your individual design and desired noise budget. These constraint values are set in Constraint Manager by right clicking in the field and selecting *Change* from the context-sensitive menu. An example is shown in the following figure.

Allegro PCB SI User Guide

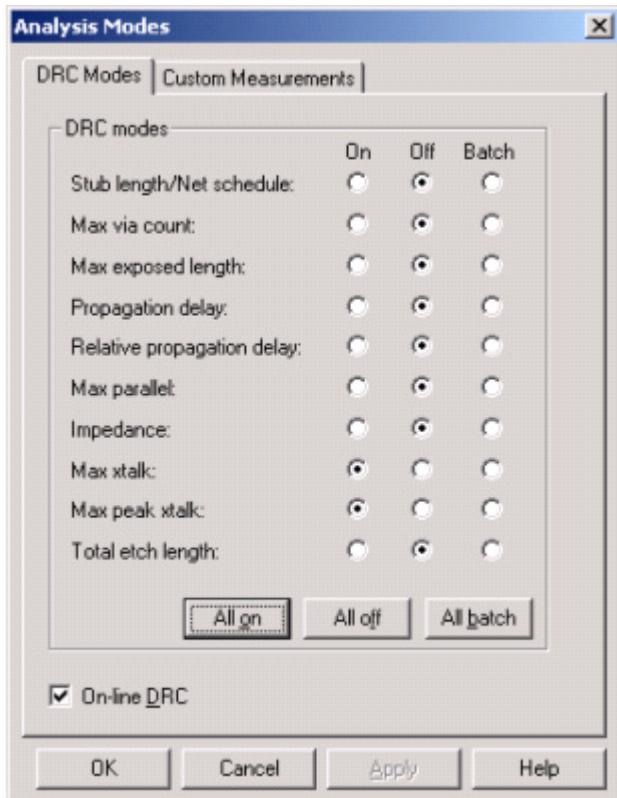
Working with Crosstalk

Figure C-7 Setting Crosstalk Constraints



To enable crosstalk-driven autorouting with PCB Router, you must turn on the `Max_xtalk` DRC. Do this by choosing *Analyze – Analysis Modes* in Constraint Manager. The Analysis Models dialog box appears as shown in [Figure C-8](#) on page 439.

Figure C-8 Setting Crosstalk DRC Modes



Note that you can set both *max_xtalk* and *max_peak_xtalk*. The *max_xtalk* DRC mode must be turned *On* for the crosstalk information to be written by SPIF into the *rules.do* file. *Max_peak_xtalk* has no effect in PCB Router, but is a useful DRC in Allegro post-route. This is discussed later in the appendix.

Setting the *max_xtalk* constraint and turning on the DRC in PCB SI makes the following appear in the resulting *.dsn* and *rules.do* files for PCB Router:

- in the *<brdname>.dsn* file
 - (noise_accumulation RSS)
 - (noise_calculation linear_interpolation)
 - (crosstalk_model CCT1A)
- in the *<brdname>_rules.do* file
 - max_noise
 - parallel_noise

- tandem_noise
- layer_noise_weight
- threshold (min coupled length)
- saturation_length
- switch_window
- sample_window

With these parameters in place, PCB Router works to minimize coupling and avoid violating crosstalk constraints during routing. Note that since PCB Router routes to `max_xtalk` constraints, but has no concept of `max_peak_xtalk` (max from any single aggressor), you may see some `max_peak_xtalk` DRCs appear after autorouting, and possibly some `max_xtalk` DRCs as well, depending on the routing density.

For threshold, the PCB SI default `MinCoupledLength` of 300 mils is used as the default value for PCB Router. In PCB SI release 14.2, you can control this with the environment variable `SPIF_XTALK_THRESHOLD`.

For example, typing:

```
set SPIF_XTALK_THRESHOLD 700
```

on the PCb Editor or PCB SI command line causes the threshold value in the `rules.do` file to be set at 700 mils. You can also set this in the `pcbenv\env` file.

Whether the design is fully autorouted or interactively routed, on-line crosstalk DRCs enforce the crosstalk constraints. Calculations are run real-time during etch editing to help identify potential crosstalk problems. If the crosstalk constraints are set intelligently, you should treat these like any other type of DRC and work to eliminate them.

Post-route Crosstalk Simulation

You should view crosstalk DRCs as a means to an end. The basic purpose they serve is to minimize significant coupling and reduce the probability of potentially dangerous crosstalk scenarios from occurring in the layout. The most accurate way to verify this is with full post-route crosstalk simulation. As described earlier, this builds full coupled-line circuits and stimulates them in the worst-case manner, accounting for reflections, wave addition and cancellation, skin effect, dielectric loss, and so forth. Waveform measurements are taken at the receiver pins to determine the maximum crosstalk value found in the high state and in the low state. Both are reported.

If DesignLinks are not set up to define system-level connectivity, backplane connector pins should be set with a pinuse of `IN`, so that a driver-to-receiver connection is found on the signal, ensuring that it is stimulated as an aggressor during crosstalk simulation.

To set up backplane connector pins in PCB SI

1. Choose *Logic – Pin Type*.

The Logic - Pin Type dialog box appears.

2. In the *Refdes Filter* text box, type in the pattern followed by the backplane connectors (for example: `J*`), then click on the resulting components in the list to select their pins.
3. In the *New Pin Type* text box, choose `IN` from the pulldown menu and move over the list of pins to become inputs.
4. Click *OK* to dismiss the dialog box.

To set up crosstalk simulation preferences

1. Choose *Analyze – Preferences*.

The Analysis Preferences dialog box appears.

2. Click the *Interconnect* tab and set the *Geometry Window* and *Min Coupled Length* parameters as desired.

Note: Cadence recommends to set the latter for 200 – 300 mils to ignore the small parallelism found during pin escaping and minimize simulation time.

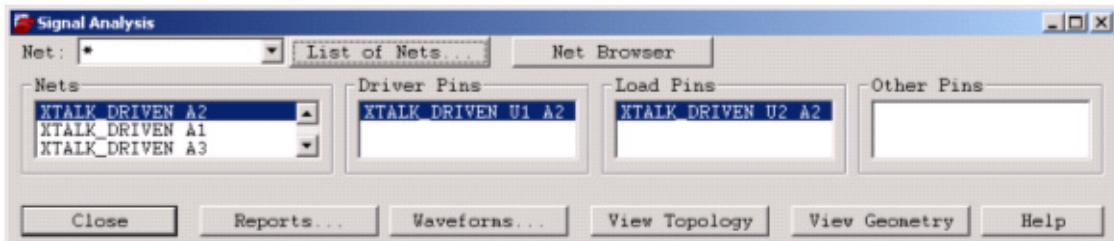
To commence crosstalk simulation and generate reports

1. Choose *Analyze – Probe*.

The Signal Analysis dialog box appears.

2. Enter `*` in the *Net* text box to search for all the Xnets in the design as shown in [Figure C-9](#) on page 442.

Figure C-9 Signal Analysis Probe Dialog Box

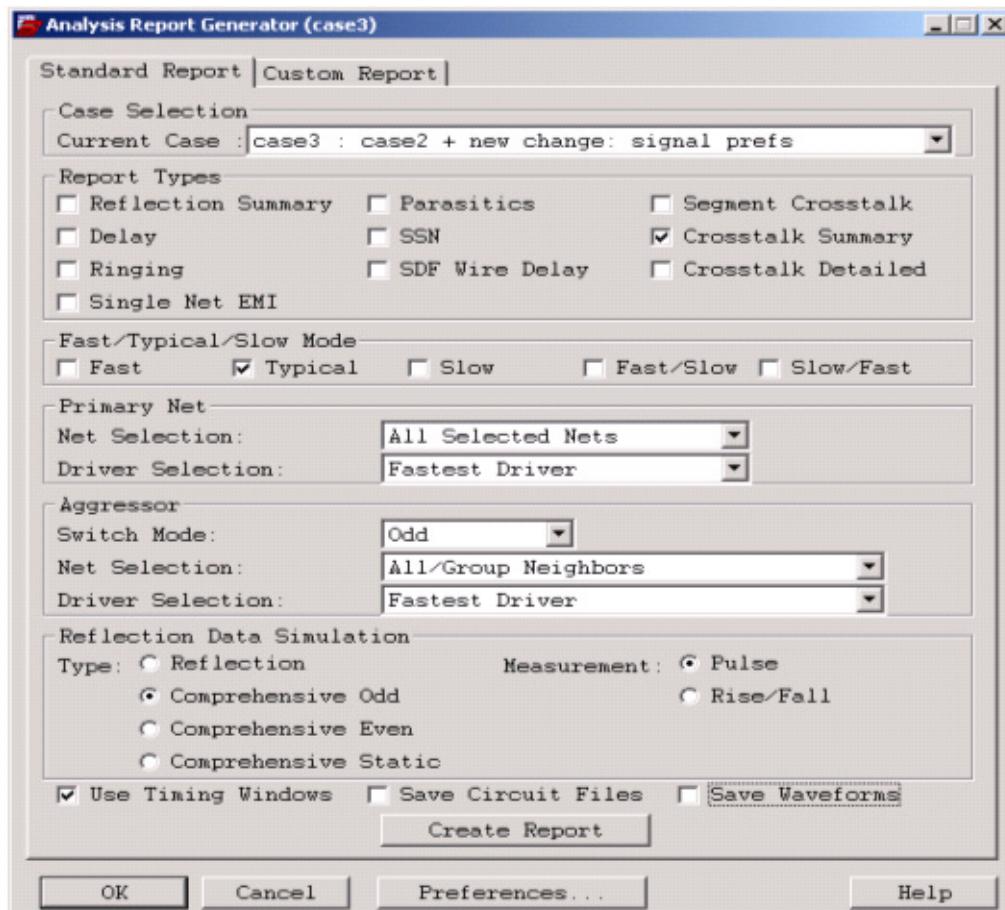


3. Once the Xnets appear in the list box, click *Reports*.

The Analysis Report Generator dialog box appears.

4. In the Analysis Report Generator dialog box, set the parameters as shown in [Figure C-10](#).

Figure C-10 Analysis Report Generator



5. Turn off all reports except for *Crosstalk Summary*.

6. Set the Simulation mode (*Fast/Typ/Slow*) as desired.

Note: Generally *Typical* is used, but the same mode should probably be used as was used to generate the crosstalk tables to drive DRCs.

7. Set the controls in the *Primary Net* and *Aggressor* sections as shown for most board-level crosstalk runs.

Note: Modifying these settings can cause very large numbers of simulations to run, and should be used carefully.

8. Turn on the *Use Timing Windows* option if they are defined in the design. This helps minimize false alarms and simulation time.

9. Check all the parameters settings carefully, then click *Create Report* to start the simulations.

When the simulations are finished, the crosstalk report appears in a Report window. It is often convenient to save this as a text file and bring it up in Microsoft® Excel where you can clean up, re-format, or sort as you wish. An example of this is shown in the following figure.

Figure C-11 Crosstalk Simulation Report

Microsoft Excel - xtalk_sim_report.xls

A4

Crosstalk Simulation results

All values in mV

	Victim	XNet	H50OddXtalk	L50OddXtalk	Group
7	K2		677.4	380	Group 5
8	B9		671.6	345.7	Group 7
9	F9		668.1	332.8	Group 1
10	B5		630.7	385.2	Group 1
11	C2		618.6	322.6	Group 1
12	A10		589.3	314.5	Group 1
13	G1		569.9	389.3	Group 1
14	K10		562.2	298	Group 1
15	G2		560	276	Group 3
16	A1		548.3	319.2	Group 7
17	D1		541.7	331.2	Group 1
18	A4		470.9	311.7	Group 3
19	B4		467.4	300.5	Group 1
20	H10		449.5	291.7	Group 7
21	J1		445.2	288.7	Group 1
22	F10		444	294.5	Group 1
23	H9		443.7	291.5	Group 3

As mentioned before, we are now looking at simulated crosstalk results, which include the effects of reflections, so the values are higher than what is seen with crosstalk DRCs (which only account for initial coupled noise).

Revisiting the crosstalk budget, we had calculated an initial `max_xtalk` constraint value (for our 3.3V supply with a +/-5% ripple tolerance) of: $(0.8 - (0.05 \cdot 3.3)) / 2 = 317.5 \text{ mV}$

Since we are now looking at simulated results, this constraint can now actually be relaxed now to $2 \times 317.5 = 635\text{mV}$.

You can build in some margin and round down to an even 600mV to keep things simple. Based on the report shown in [Figure C-11](#) on page 444, you are left with 5 signals that violate the budget and which you need to address.

Crosstalk Troubleshooting

Based on the Simulated Crosstalk report, create a file containing a list of the signals that you need to address for crosstalk.

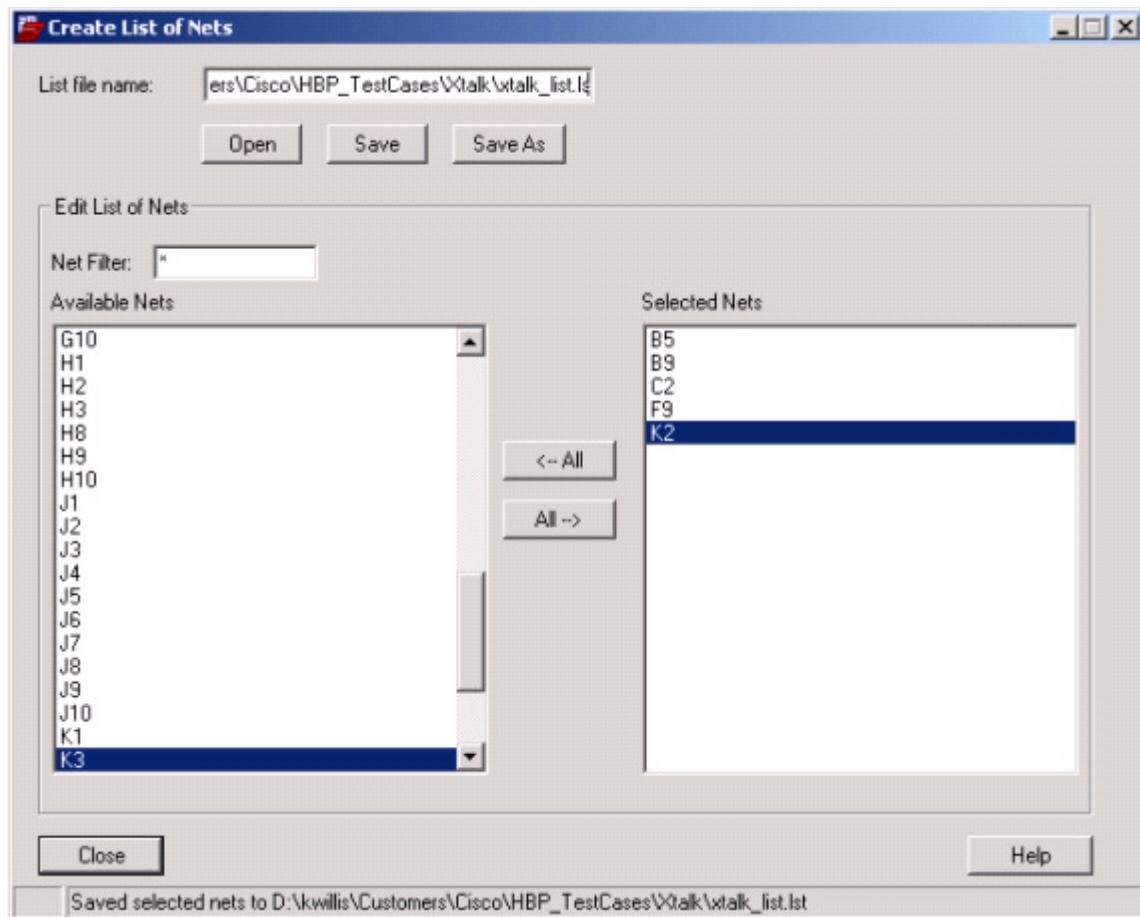
To create a list of nets

1. Choose *Logic – Create List of Nets*.

The Create List of Nets dialog box appears as shown in the following figure.

2. In the List file name text box, enter a filename for the net list.
3. Based on the Crosstalk report, select the nets to be in the list, then click *Save*.
4. Click *Close* to dismiss the dialog box.

Figure C-12 Creating a List of Nets



To determine which signals are coupling onto these nets

1. Choose *Analyze – Probe*.

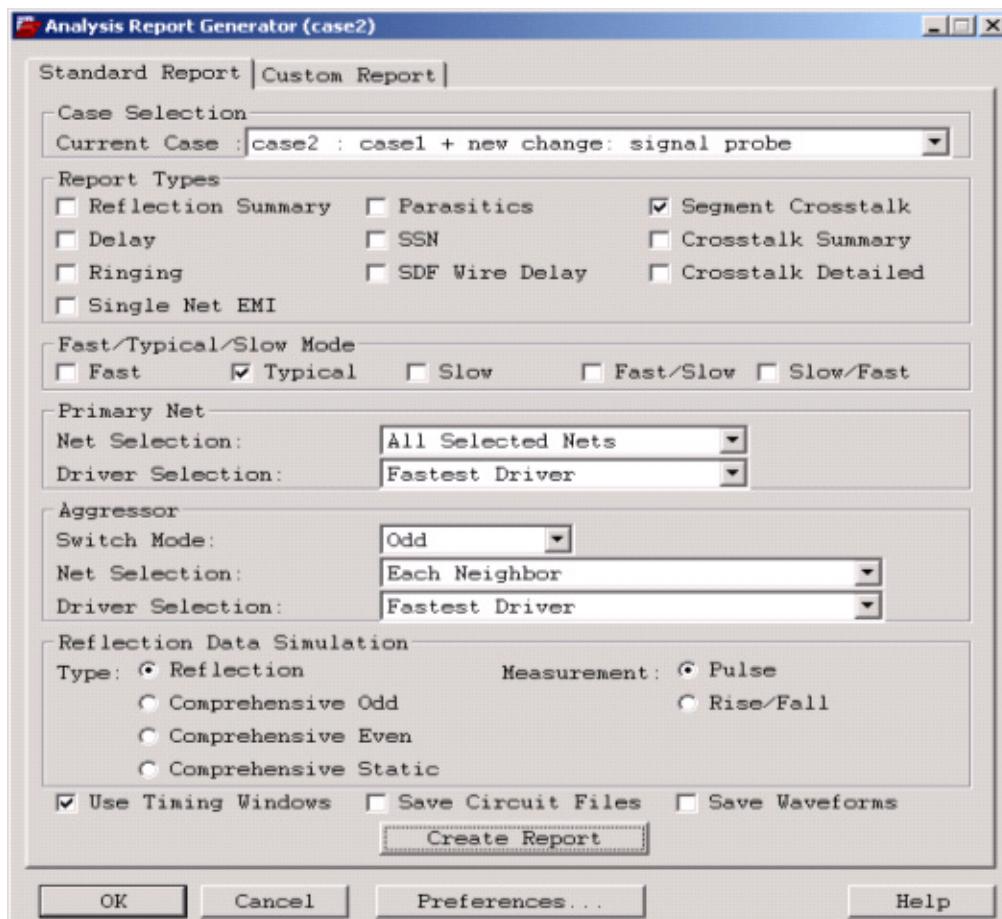
The Signal Analysis dialog box appears.

2. Click *List of Nets* and read in the netlist file that was created.

3. Determine the signals in the design that are coupling onto the signals in the list, and what the levels look like.

You can do this quickly by choosing a *Segment Crosstalk* report, selecting *Each Neighbor* in the *Aggressor* section of the Report Generator (as shown in the following figure), and then clicking *Create Report*.

Figure C-13 Creating a Segment Crosstalk Report



Allegro PCB SI User Guide

Working with Crosstalk

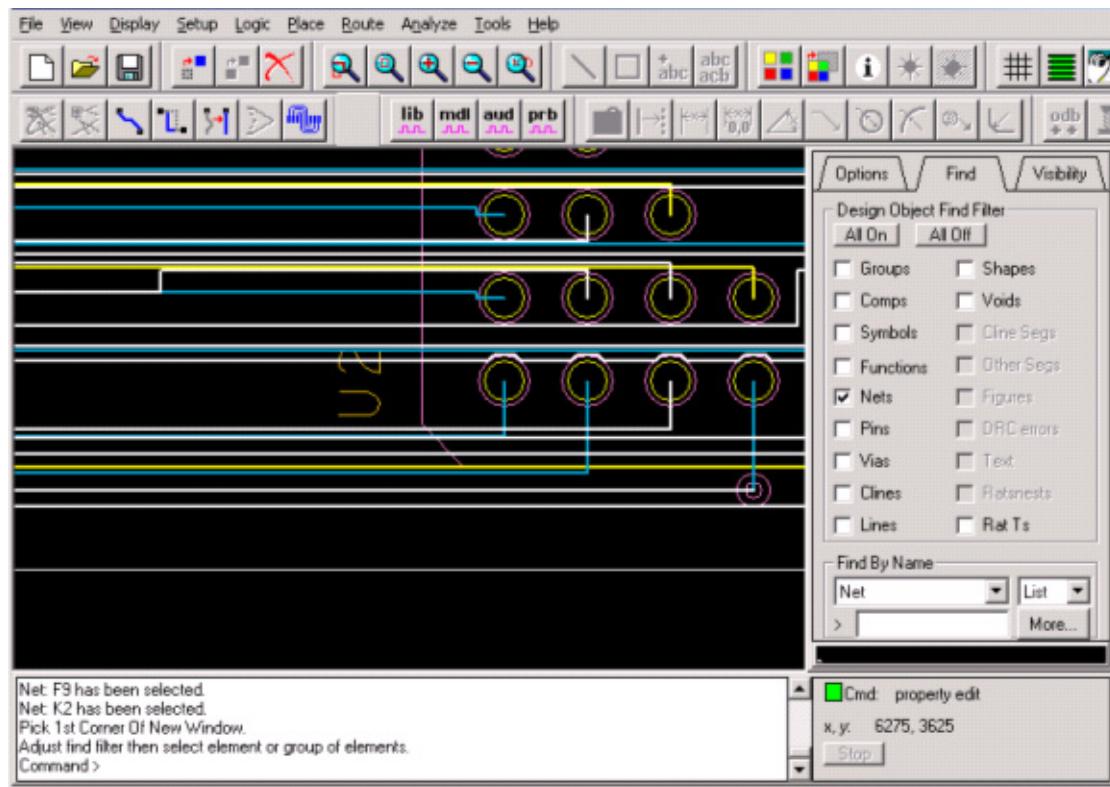
Figure C-14 Resulting Segment Crosstalk Report

Segment Crosstalk (mV)							
Victim	Aggressor	Layer	Layer	XCoord	Gap	Length	SegXtalk
1 XTALK_DRIVEN_ROUTE B9	XTALK_DRIVEN_ROUTE F9	SIG2	SIG1	4900:3863	0	3000	96.13
1 XTALK_DRIVEN_ROUTE B9	all_neighbor_xnets	NA	NA	NA	NA	NA	96.1
1 XTALK_DRIVEN_ROUTE B9	XTALK_DRIVEN_ROUTE A10	SIG2	SIG1	6777:3865	0	47	1.506
1 XTALK_DRIVEN_ROUTE F9	XTALK_DRIVEN_ROUTE B9	SIG1	SIG2	4900:3863	0	3000	96.13
1 XTALK_DRIVEN_ROUTE F9	all_neighbor_xnets	NA	NA	NA	NA	NA	96.1
1 XTALK_DRIVEN_ROUTE B5	XTALK_DRIVEN_ROUTE G2	SIG2	SIG1	4050:4263	0	2900	92.93
1 XTALK_DRIVEN_ROUTE B5	all_neighbor_xnets	NA	NA	NA	NA	NA	92.9
1 XTALK_DRIVEN_ROUTE K2	all_neighbor_xnets	NA	NA	NA	NA	NA	87.2
1 XTALK_DRIVEN_ROUTE K2	XTALK_DRIVEN_ROUTE C2	SIG1	SIG2	4650:4663	0	2700	86.52
1 XTALK_DRIVEN_ROUTE K2	XTALK_DRIVEN_ROUTE H1	SIG1	SIG2	2977:4658	0	353	11.31
1 XTALK_DRIVEN_ROUTE C2	all_neighbor_xnets	NA	NA	NA	NA	NA	86.6
1 XTALK_DRIVEN_ROUTE C2	XTALK_DRIVEN_ROUTE K2	SIG2	SIG1	4650:4663	0	2700	86.52
1 XTALK_DRIVEN_ROUTE C2	XTALK_DRIVEN_ROUTE A1	SIG2	SIG1	6477:4683	15	447	3.331

This report shows the main aggressors for these problem nets and what the initial coupled noise values look like.

Now you need to take a look at these cases in the layout and see if you can address them. One way to do this is to tighten up the `max_peak_xtalk` constraints for these individual signals in Constraint Manager, and use the on-line DRCs to find the areas of coupling.

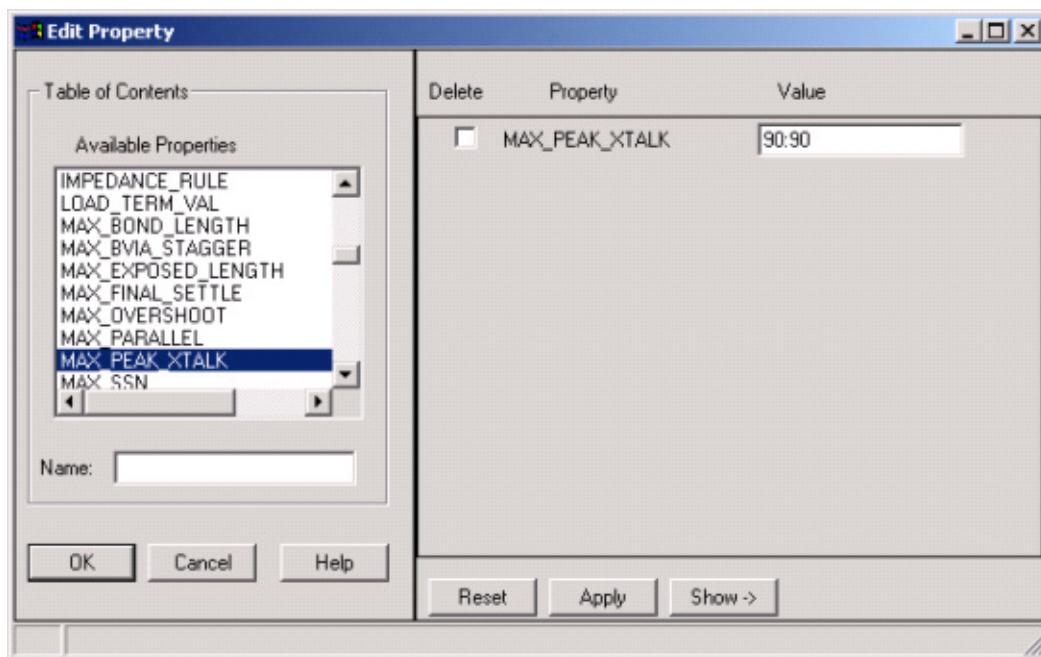
Figure C-15 Grabbing the Signals in the Crosstalk List



To change the max_peak_xtalk constraints

1. Choose *Setup – Edit Properties*, click *All Off* in the Find Filter and turn on just *Nets*.
 2. Set the *Find By Name* fields for *Net* and *List*, and enter the name of the list file, *xtalk_list.1st* in this case.
- The Edit Property dialog box appears.
3. Select **MAX_PEAK_XTALK** and enter the value for 90mV (for high and low state), as shown in the following figure.

Figure C-16 Setting MAX_CROSSTALK Constraint Overrides



If you look in Constraint Manager, you can see that the 90mV constraint value shows up as an override on these signals, *in blue*. At the board level, new DRCs show up, that you can now address.

To Highlight the Problem Nets

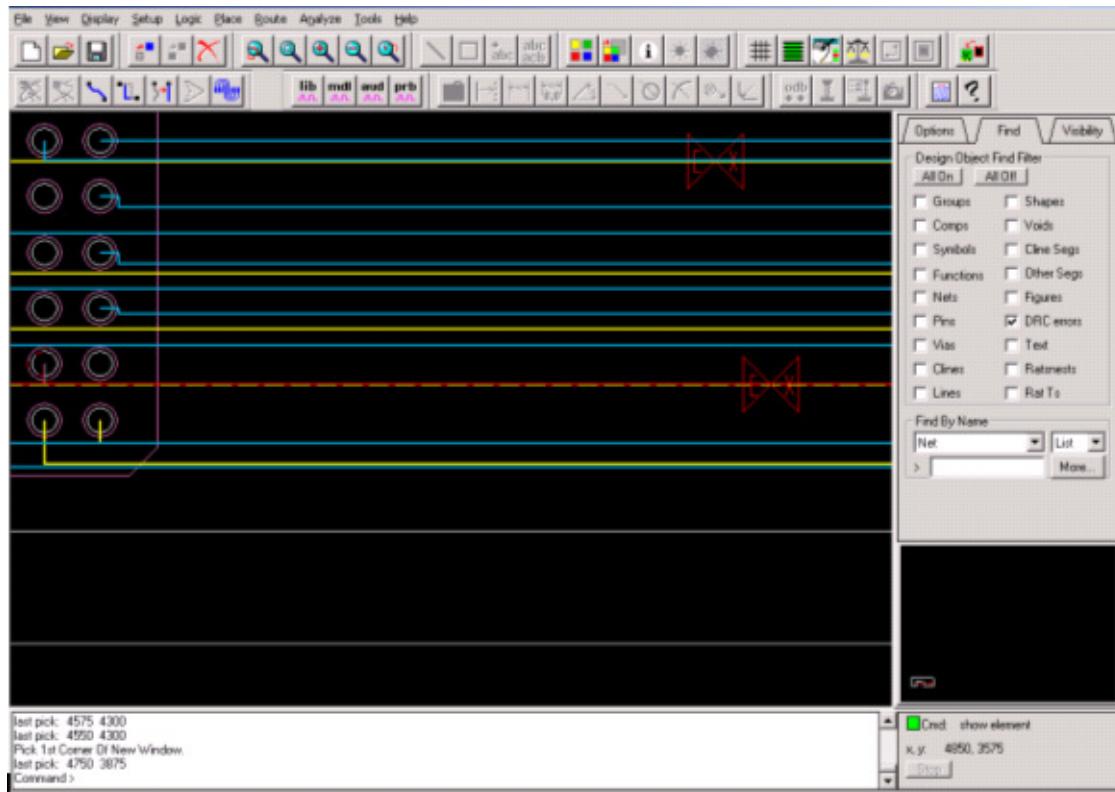
- Choose *Display – Element*, select *DRC errors* in the Find Filter, then click on the DRC marker.

The problem nets are highlighted and are ready to move.

Allegro PCB SI User Guide

Working with Crosstalk

Figure C-17 Crosstalk DRC Display



In the case above, traces are on top of each other, creating some significant layer-to-layer coupling. You can slide traces and move them around until the DRCs disappear, then you can re-run the simulations. Constraint Manager is used to modify override values for individual signals using the *Nets – Estimated Crosstalk* worksheet. Once again, these show up in blue, as seen in the following figure.

Allegro PCB SI User Guide

Working with Crosstalk

Figure C-18 Crosstalk Constraint Overrides in Constraint Manager

The screenshot shows the Allegro PCB Constraint Manager interface. The left pane displays a tree view of constraint sets: Electrical Constraint Set, Signal Integrity, Timing, Routing, All Constraints, Net, and Signal Integrity. Under Signal Integrity, there are sub-options for Timing, Routing, and Custom Measurement. The right pane is a table titled 'Xtalk' and 'Peak Xtalk'. The 'Xtalk' table has columns for Objects (B8 through G9), Max (mV), Actual (mV), and Margin (mV). The 'Peak Xtalk' table has columns for Max (mV), Actual (mV), and Margin (mV). In the 'Actual' column of the 'Xtalk' table, several values are highlighted in red: B9 (90.00), C1 (80.00), C2 (80.00), F9 (90.00), and F10 (90.00). The table also includes tabs for Edge Distortions, Estimated Xtalk, and Simulate.

Objects	Xtalk			Peak Xtalk		
	Max	Actual	Margin	Max	Actual	Margin
	mV	mV	mV	mV	mV	mV
B8	250.0	250.0	125.0	125.0	125.0	125.0
B9	250.0	250.0	125.0	90.00	90.00	125.0
B10	250.0	250.0	125.0	125.0	125.0	125.0
C1	250.0	250.0	125.0	125.0	125.0	125.0
C2	250.0	250.0	125.0	80.00	80.00	125.0
C3	250.0	250.0	125.0	125.0	125.0	125.0
C8	250.0	250.0	125.0	125.0	125.0	125.0
C9	250.0	250.0	125.0	125.0	125.0	125.0
C10	250.0	250.0	125.0	125.0	125.0	125.0
D1	250.0	250.0	125.0	125.0	125.0	125.0
D10	250.0	250.0	125.0	125.0	125.0	125.0
E1	250.0	250.0	125.0	125.0	125.0	125.0
E10	250.0	250.0	125.0	125.0	125.0	125.0
F1	250.0	250.0	125.0	125.0	125.0	125.0
F2	250.0	250.0	125.0	125.0	125.0	125.0
F9	250.0	250.0	125.0	90.00	90.00	125.0
F10	250.0	250.0	125.0	125.0	125.0	125.0
G1	250.0	250.0	125.0	125.0	125.0	125.0
G2	250.0	250.0	125.0	125.0	125.0	125.0
G9	250.0	250.0	125.0	125.0	125.0	125.0

You can repeat this process of updating constraints, addressing DRCs, and time domain simulation until verification results for all signals meet the desired crosstalk levels.

Working with Timing

Introduction

Both timing and signal integrity analysis are critical aspects of ensuring that a design will work *at speed*. You must integrate the results of both analysis to get the complete picture of system timing behavior. Each type of analysis assumes certain conventions about how delays are computed. As a high-speed PCB designer, you must understand these conventions and have the ability to check and validate the design data for conformity.

PCB SI provides a *bus timing model* capability for integrating signal integrity and timing analysis. See [Bus Timing Model](#) on page 470 for further details.

Timing Analysis Basics

Static Timing Analysis

Static timing analysis is a systematic analysis of a synchronous ASIC, PCB or system design, that identifies:

- logic hazards.
- clocked timing paths.
- timing errors.

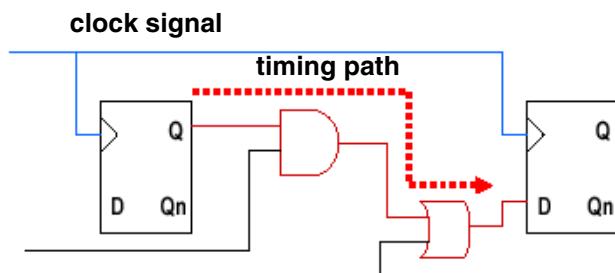
The inputs required for static timing analysis are:

- a functional description of the circuit (a nattiest).
- component-level timing data.
- circuit operating (clock) speeds.

What is a clocked timing path?

A clocked timing path consists of all of the logic between two clocked elements that operate off the same clock signal. The path is analyzed to ensure that setup and hold requirements are met at the input of each clocked element. You can also use the slack (delay margin) in the path to derive SI flight time constraints. See [Figure D-1 on page 452](#).

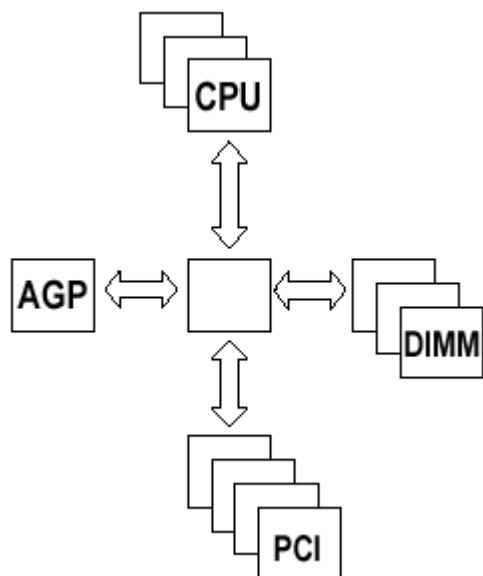
Figure D-1 Clocked Timing Path



Modern System Design

Modern systems are dominated by high speed bus interconnections. You can perform timing analysis for data buses using a simplified bus-level timing model.

Figure D-2 Modern System Design



Flight Time

Flight time accounts for the electrical delay of interconnect (PCB etch) between the driving device and receivers. You can estimate for slow speed circuits but must simulate (signal integrity) for high speed designs.

Figure D-3 Flight Time

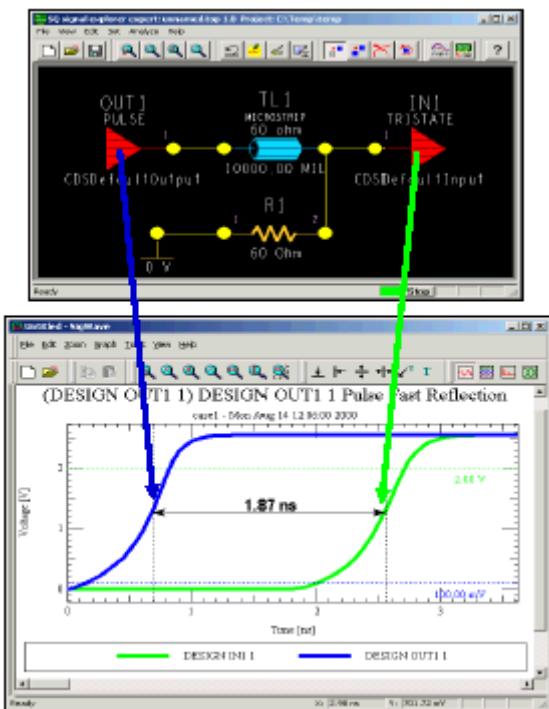
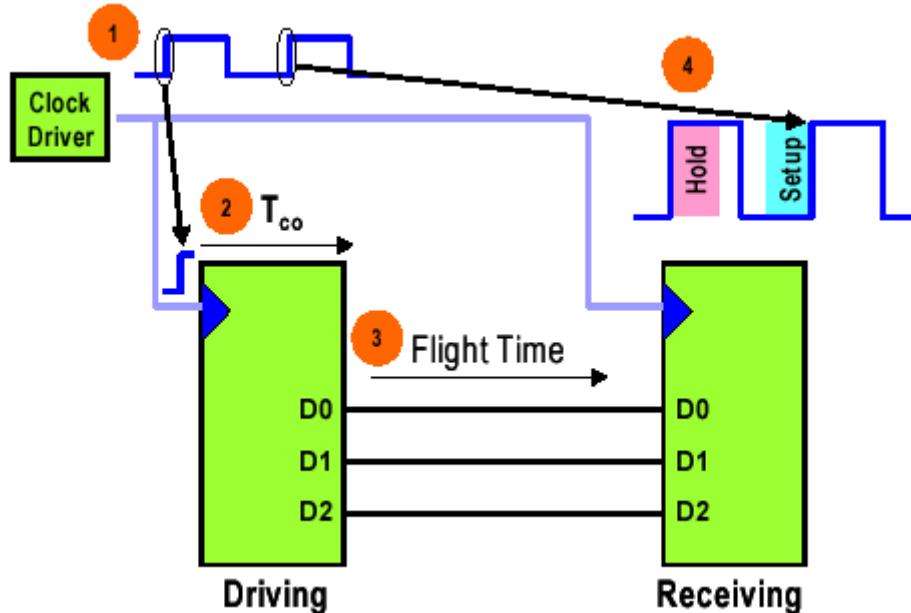
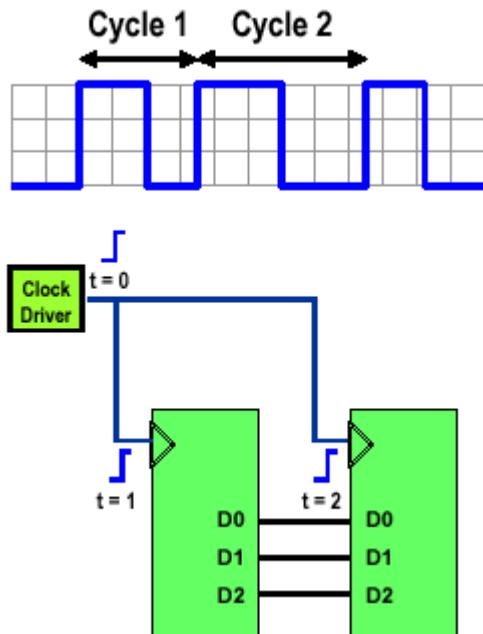


Figure D-4 Standard Synchronous Data Transfer

Synchronous Design Issues

Clock jitter increases or decreases the individual clock cycle, decreasing the time left for data transfer. The clock skew changes the effective clock period depending on which devices are driving or receiving.

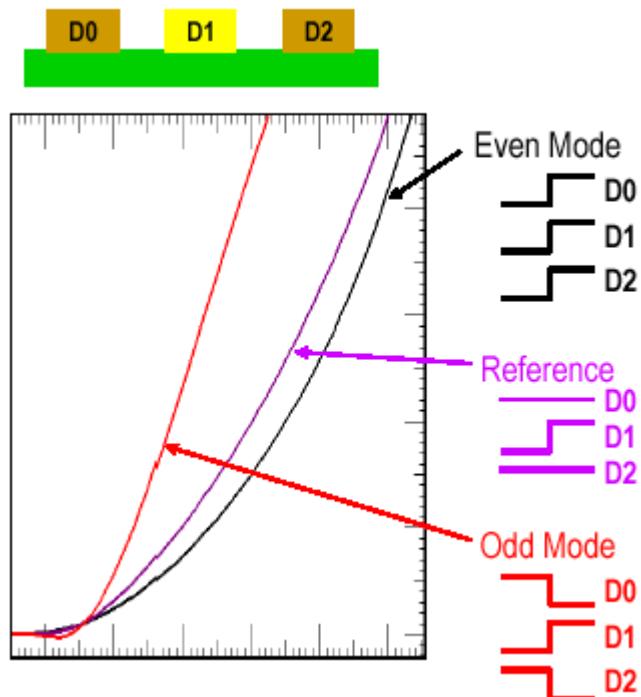
Figure D-5 Synchronous Design Issues



Impact of Crosstalk on Bus Timing

Crosstalk between adjacent bus bits affects edge speed (and therefore, flight time). Denser routing makes better use of board space, but usually at the expense of larger variations in flight time. Pre-route crosstalk analysis helps you make the best trade-off between routing density and signal integrity.

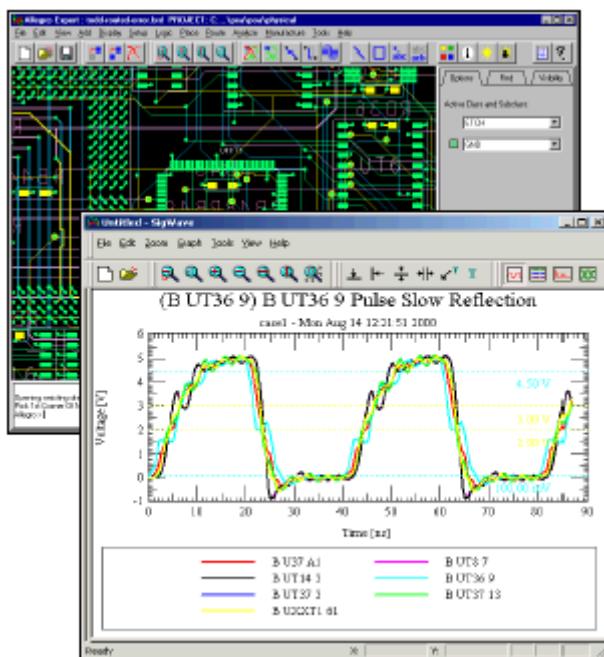
Figure D-6 Crosstalk Effects on Bus Timing



SI Analysis Basics

Signal integrity analysis is analog analysis of digital switching behavior. It uses special analog models to represent device inputs and outputs.

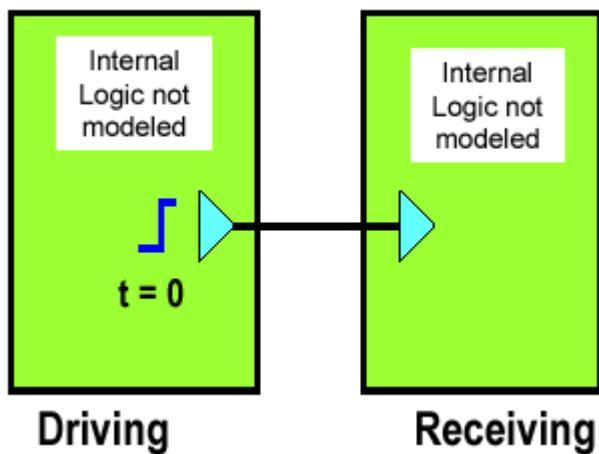
Figure D-7 Signal Integrity Analysis



The Signal Integrity Model

SI models represent only the behavior of the device output and input buffers. The internal component functions and associated timing are not modeled.

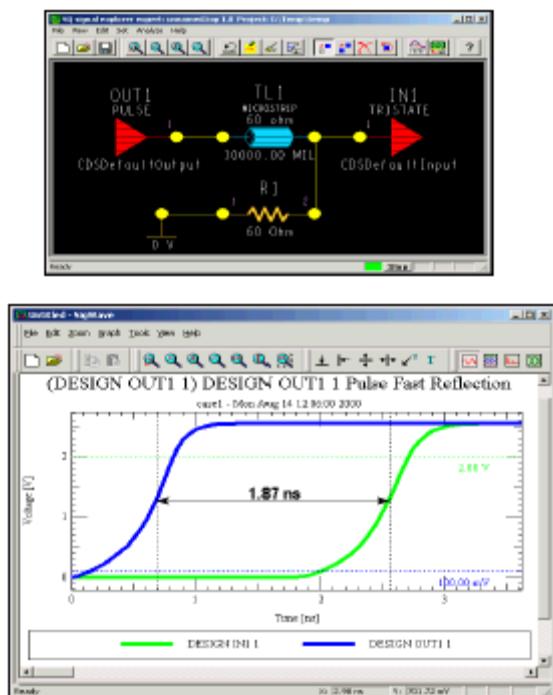
Figure D-8 SI Models



Measuring Interconnect Delay

Interconnect delay accounts for electrical delay caused by the interconnect (PCB etch) between the driving device and each receiver on the net. This is usually different for each driver-receiver combination; use signal integrity analysis to determine the delay.

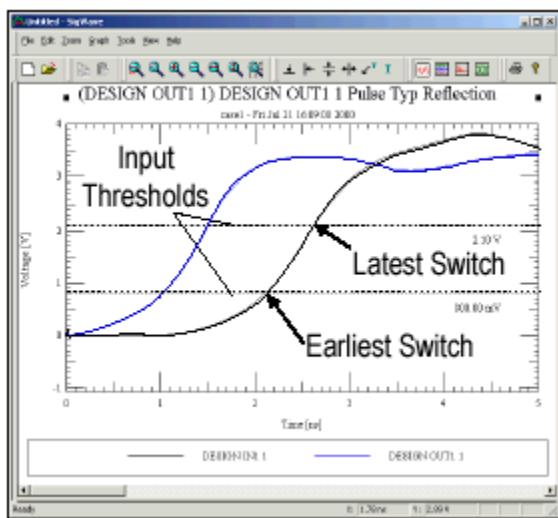
Figure D-9 Measuring Interconnect Delay



Minimum and Maximum Delays

Use the receiver's input thresholds to determine the earliest and latest times that the input change can be detected. This information is then used to determine minimum & maximum flight time data for each driver/receiver combination.

Figure D-10 Minimum and Maximum Delays



Component Timing

Figure D-11 A closer look at T_{co}

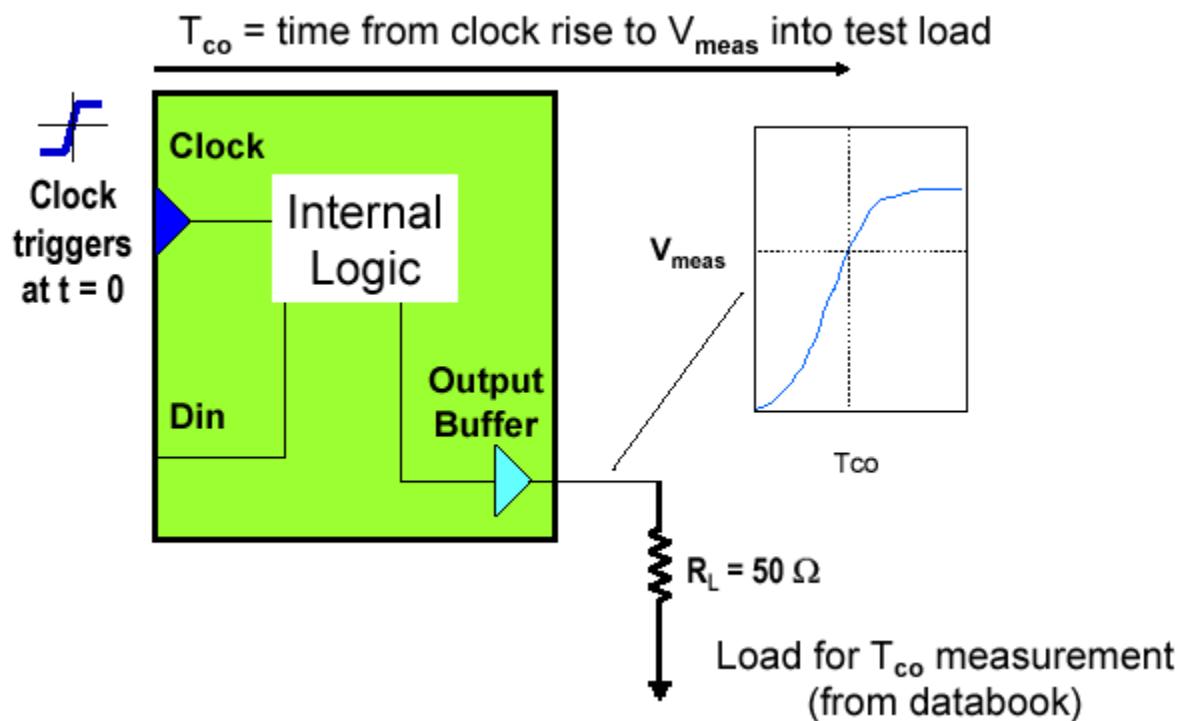
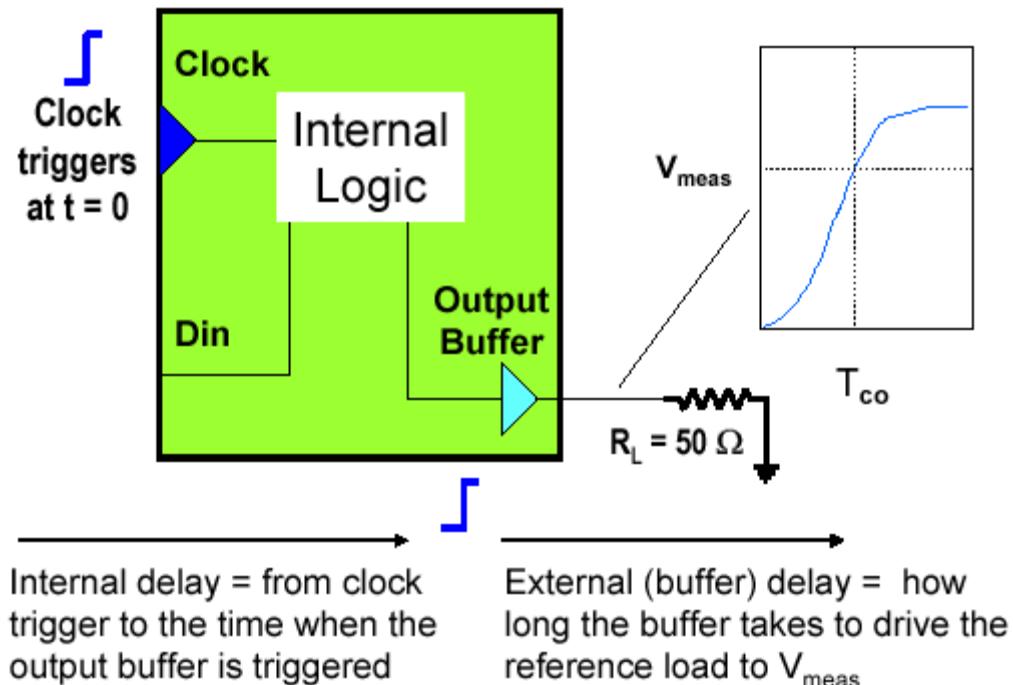


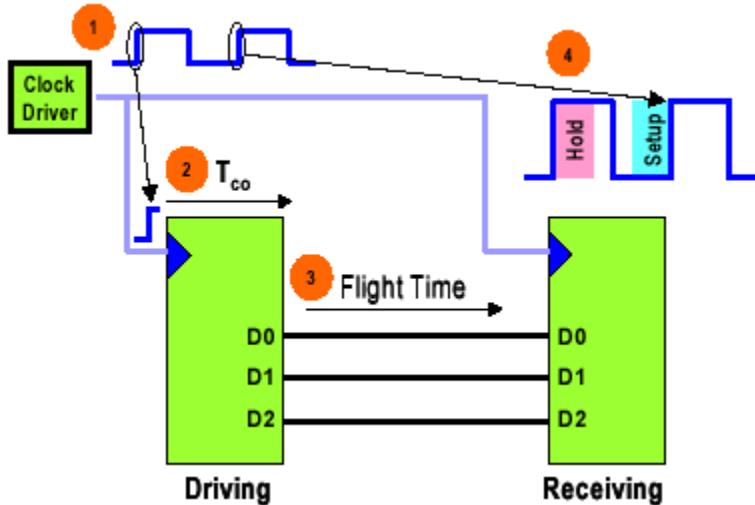
Figure D-12 Components of T_{co}



The Double-Counting Problem

You want to know at what point in the clock period signals arrive and stabilize at the receiver input. This is then compared to your setup and hold constraints. You find this by combining the component timing data (T_{co}) with flight time data from signal integrity analysis.

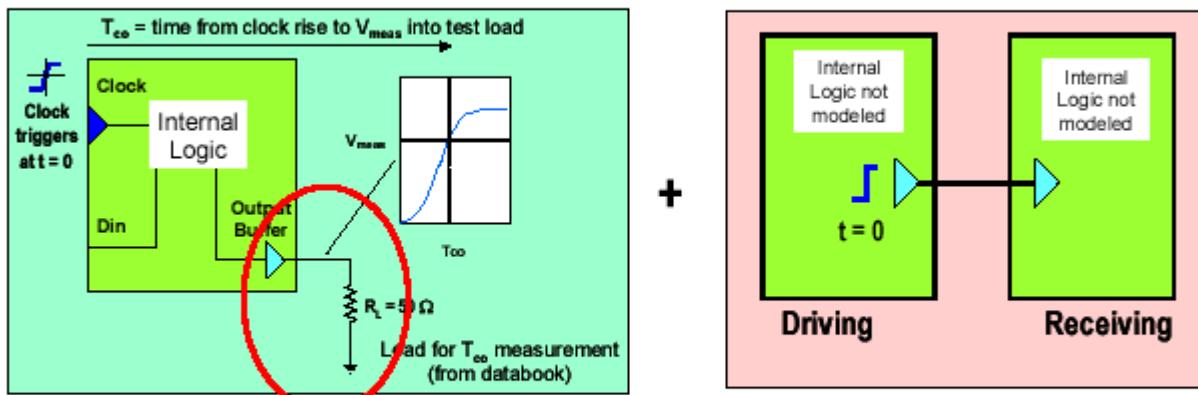
Figure D-13 Combining T_{co} with flight time



However, if you simply add the T_{co} from the databook to the simulated delay, the external buffer delay portion of the T_{co} gets counted twice as illustrated in [Figure D-14](#).

Figure D-14 Double-Counting Problem

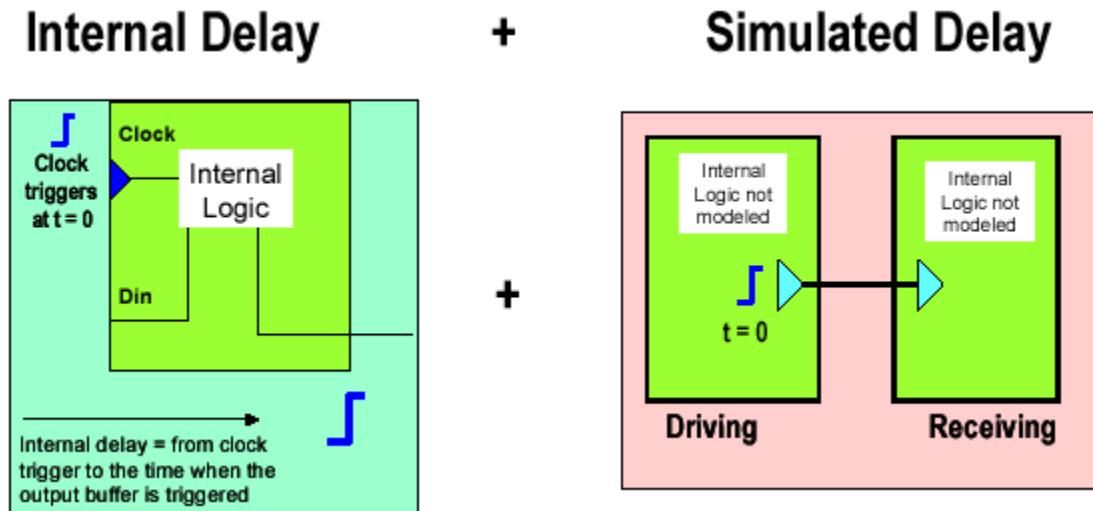
T_{co} (from Databook) + **Simulated Delay**



The external buffer delay portion of T_{co} gets double-counted !!

What you really want to do is add the internal delay and the simulated delay as shown in [Figure D-15](#) on page 463.

Figure D-15 Combining internal and simulated delays



Making The Pieces Fit Together

You can solve this discrepancy by:

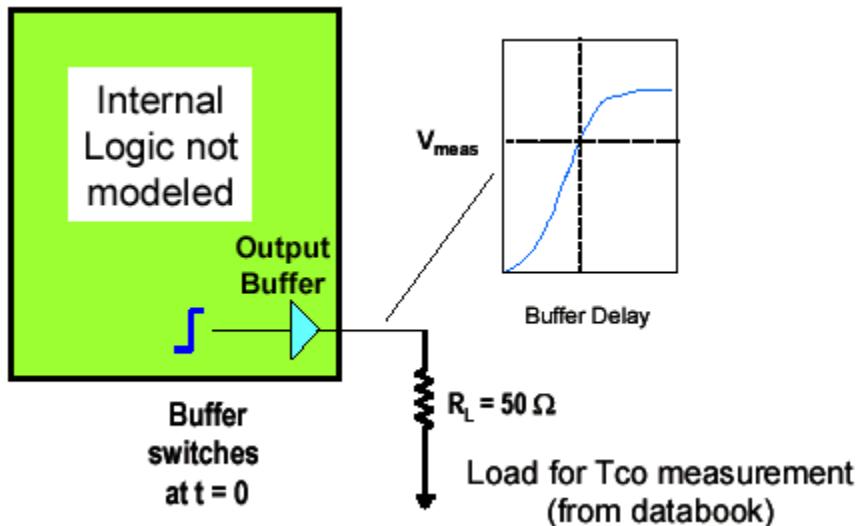
- adjusting the value of Tco used for timing analysis by subtracting out the time attributed to Tco buffer delay.
- subtracting the time attributed to the Tco buffer delay from the input receiver switching times predicted by simulation.

By convention, Cadence recommends using the latter method.

Determining the Buffer Delay

The output buffer model used for signal integrity analysis is connected to the Tco test load and simulated. The delay is measured at the point where the output pin crosses Vmeas. The corresponding delay is then saved and used in flight time computations.

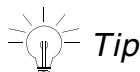
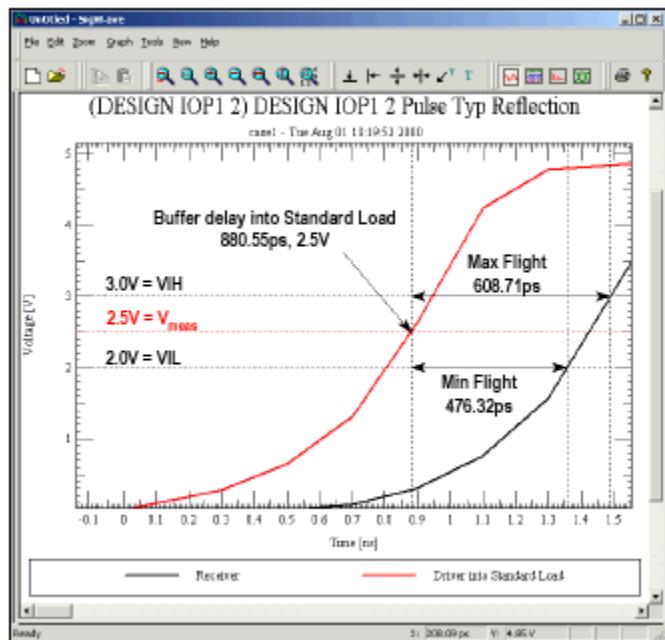
Figure D-16 Determining the Buffer Delay



Measuring Flight Time

Flight time is always measured with respect to the delay into the standard load. This is accomplished by determining the T_{CO} buffer delay and subtracting that value from simulation results. See [Figure D-17](#) on page 465.

Figure D-17 Measuring Flight Time



You cannot directly measure output-to-input delay to determine flight time. The loading condition used to compute buffer delay and the conditions under which T_{CO} is measured must be identical.

About Device Modeling

Quality problems in SI models are not unusual. You should always strive to check the model syntax quality and buffer delay information before using a device model. An efficient way to do this is to use the Model Integrity tool available with PCB SI. For further details on checking models using Model Integrity, refer to the [Model Integrity User Guide](#).

There are different device models, each supporting a specific purpose.

- Pre- layout models support min/max package parasitics only.
- Post- layout models support detailed per-pin parasitic data.

Verifying Standard Loading Conditions

IBIS provides specific keywords to define the conditions under which you should simulate and measure buffer delays. The measurement and loading conditions in the IBIS file should be the same as the conditions under which T_{CO} is specified in the device's datasheet.

Figure D-18 Verifying Loading Conditions

IBIS Model File

```
...
Model_type I/O_open_drain
Polarity Non-Inverting
Enable Active-Low
Vinl = 0.8
Vinh = 1.2
Vmeas = 1.00
Cref = 0.00p
Rref = 25.00
Vref = 1.50
...
...
```

Integrating Timing and SI Analysis

There are several ways to integrate timing analysis and signal integrity results.

- Manual Approach
- General Approach
- Bus-level Timing Approach

Manual Approach

You can determine allowable min/max flight times using component timing data and a spreadsheet. You then use signal integrity analysis to verify that the design meets the computed flight time requirements.

For common-clock buses, you can compute allowable min/max flight times from bus speeds, system budgets, and component timing data as shown in [Figure D-19](#) on page 468. Timing equations are programmed into a spreadsheet and allowable flight times computed.

While not elegant, this method is fast, flexible, and reliable when you need to determine the timing for a small number of buses.

Allegro PCB SI User Guide

Working with Timing

Figure D-19 Determining Flight Times from Component Timing Data

Budgeted Parameters						
Clock Skew	0.2 ns	Device Timing Information				
Clock Jitter	0.4 ns	T _{co} _{min}	T _{co} _{max}	Setup	Hold	
Crosstalk	0.4 ns	Pentium Pro	0.55 ns	4.40 ns	2.20 ns	0.45 ns
Clock Freq.	66 MHz	440FX	1.25 ns	7.25 ns	5.00 ns	0.00 ns
Clock Period	15.15 ns					

Pentium Pro to 440FX						
T _{flight} _{max} =	ClockPeriod -	T _{co} _{max} -	Skew -	Jitter -	Crosstalk -	Receiver(Setup)
4.75 ns	15.15 ns	4.40 ns	0.20 ns	0.40 ns	0.40 ns	5.00 ns
T _{flight} _{min} =	Receiver(Hold) -	T _{co} _{min} +	Skew +		Crosstalk	
0.05 ns	0.00 ns	0.55 ns	0.20 ns		0.40 ns	

440FX to Pentium Pro						
T _{flight} _{max} =	ClockPeriod -	T _{co} _{max} -	Skew -	Jitter -	Crosstalk -	Receiver(Setup)
4.70 ns	15.15 ns	7.25 ns	0.20 ns	0.40 ns	0.40 ns	2.20 ns
T _{flight} _{min} =	Receiver(Hold) -	T _{co} _{min} +	Skew +		Crosstalk	
-0.20 ns	0.45 ns	1.25 ns	0.20 ns		0.40 ns	

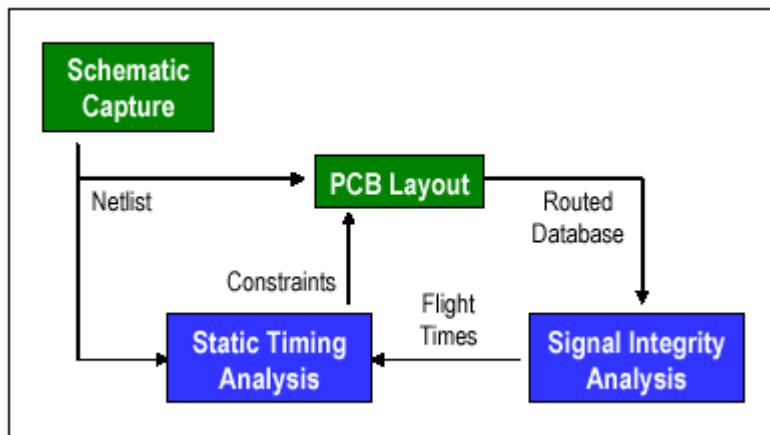
- $T_{flight_{max}} = 4.70 \text{ ns}$
- $T_{flight_{min}} = 0.05 \text{ ns}$

General Approach

You can use static timing analysis to evaluate system timing and signal integrity analysis to compute flight times. You then feed flight time data back into the static timing tool.

Timing analysis, layout, and SI analysis are run as separate processes as shown in [Figure D-20](#) on page 469. Flight time data from signal integrity analysis is fed back into timing analysis to complete the loop and integrate the two sets of data. Changing the design requires re-running the complete loop.

Figure D-20 Timing Analysis and SI Analysis Flow

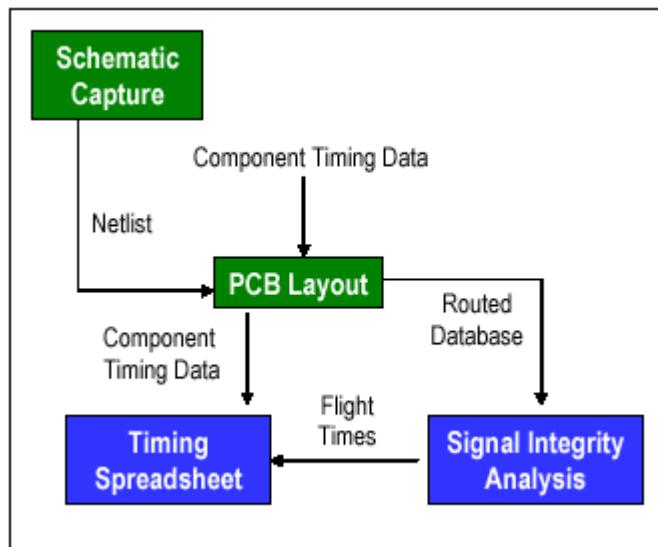


Bus-level Timing Approach

You can use standard timing equations and component timing data to perform spreadsheet-based timing analysis. You then feed flight times from signal integrity analysis back into the spreadsheet to compute design margins.

Component timing, bus speeds and clock jitter /skew budgets are captured as part of the PCB database. Signal integrity analysis is run from the PCB database, then a spreadsheet containing bus- level timing equations is used to compute the design margins based on simulation results as shown in [Figure D-21](#) on page 470.

Figure D-21 Timing Spreadsheet and SI Analysis Flow



Bus Timing Model

The bus timing model provided in PCB SI offers the following advantages:

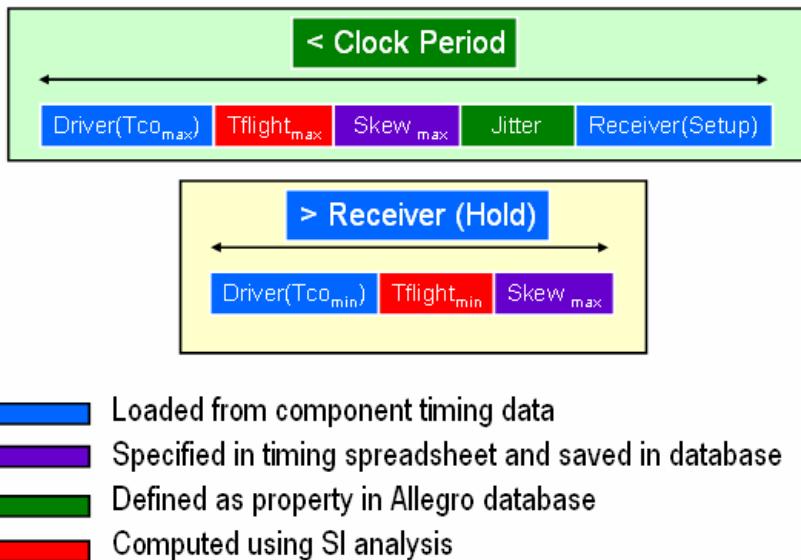
- Bus- level timing and signal integrity analysis is integrated in a single tool.
- You can run analysis interactively as parts are moved or nets are routed.
- All information is kept in a single design database.

[Figure D-22](#) on page 471 illustrates the bus timing model. [Figure D-23](#) on page 472 illustrates the timing flow that you should use within PCB SI.

Allegro PCB SI User Guide

Working with Timing

Figure D-22 PCB SI Timing Model



The Setup/Hold calculations should be documented as follows:

```
Setup Margin = ClockPeriod - clockJitter - clockSkewMax - Clock2OutMax -  
SettleDelay - minSetup
```

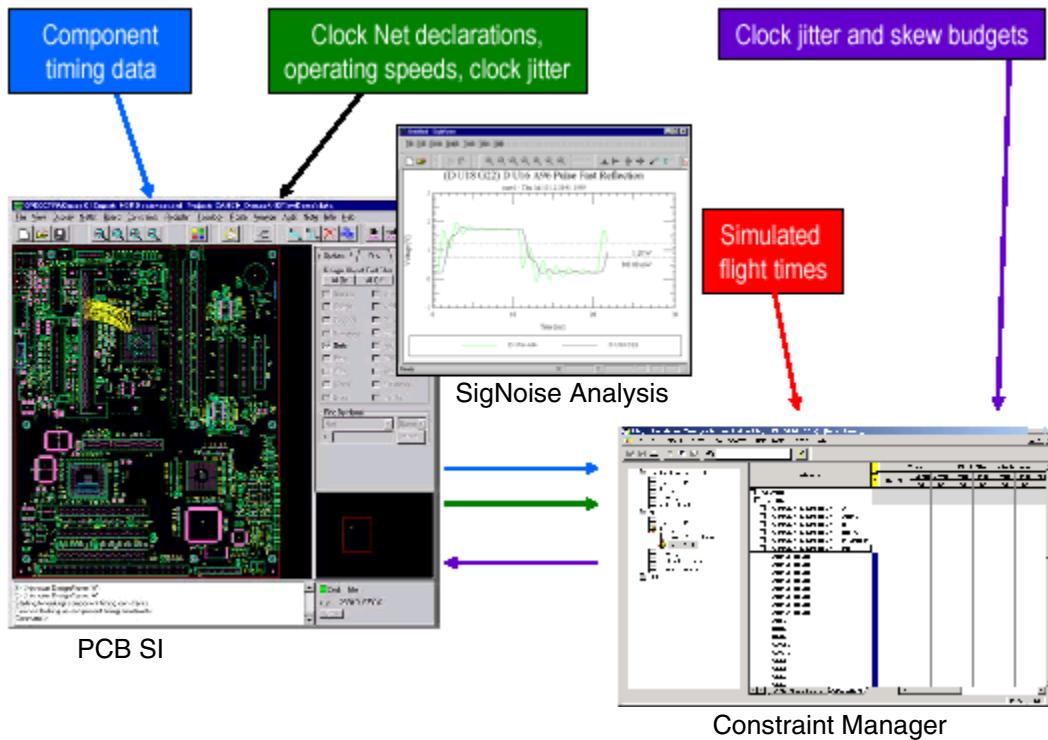
```
HoldMargin = Clock2OutMin + switchDelay - clockSkewMax - minHold
```

Note: The ClockSkewMin column remains on the worksheet but is not used.

Allegro PCB SI User Guide

Working with Timing

Figure D-23 PCB SI Timing Flow



Working with Multi-GigaHertz Interconnect

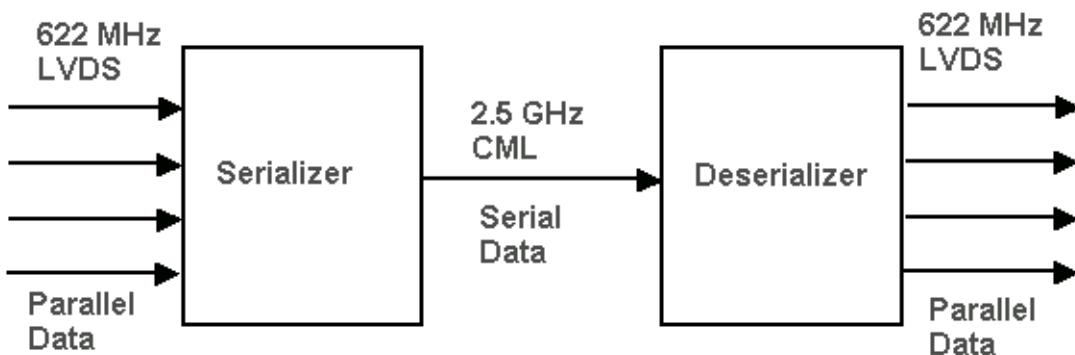
Introduction

Serial Data Links

A common design approach to meet today's bandwidth requirements in the telecom and datacom markets is to use serial data links. You can scramble multiple parallel data streams of lower frequency buses together to form a single higher frequency serial data stream, transmit it from one place to another, then subsequently unscramble it back down into the lower frequency parallel data streams again. This allows you to move mass quantities of data around with lower physical density.

In the following figure, four parallel 622MHz LVDS buses run to a serializer chip (or serializer block embedded in a large ASIC), mux'd up to a serial 2.5GHz CML differential pair, and sent over a backplane, where it is then received and demux'd on another PCB.

Figure E-1 Parallel and Serial Data



Multi-gigahertz (MGH) serial data links have unique design challenges. As opposed to sub-GHz source synchronous buses, where the end game is setup and hold margins, the ultimate

requirement for serial data links is to meet a specific bit error rate (BER). This is derived through analysis of the data's eye pattern, which is heavily influenced by the loss and ISR generated along the total signal path or channel (due primarily to skin effect, dielectric loss, and impedance discontinuities). This causes the Eye pattern to close and jitter to increase. This is difficult to handle at today's data rates. As the market moves to higher frequencies, traditional lab validation techniques using oscilloscopes becomes difficult or impossible. The bit error rate tester (BERT) becomes the final validation metric for these types of interfaces.

Inter-Symbol Interference (ISI)

From a design standpoint, predicting BER through simulation requires running a huge stream of tens of thousands of bits and analyzing the resulting eye pattern. Doing this through traditional time domain circuit simulation becomes too computationally intensive for the design phase, and requires a new and alternate methodology. Also, frequency domain analysis becomes necessary to characterize the losses seen through the different sections of the data channel.

Advanced Solutions for Multi-GigaHertz Signal Design

Signals operating in the MGH range require a new generation of design tools to properly design system interconnect. These tools must model each element of the signal's path accurately and quickly. At high frequencies, the losses on a signal increase as the signal travels through different discontinuities such as vias, connectors and different layers in one or more printed circuit boards. At GigaHertz frequencies, the loss in a transmission line can be approximately 0.25+ dB/inch, thus creating challenges for longer interconnects on PCB systems. Ensuring that losses in critical signals are acceptable is an important step in the design of MGH signals. To accomplish this, you need a way to do loss budget trade-offs quickly and iteratively. You need a way to change MGH signal topology and within seconds be able to view the expected loss through the system interconnect.

Allegro PCB SI Multi-Gigabit Option provides an integrated, complete environment for loss budget trade-offs using industry-standard format S-parameters. With Allegro PCB SI, generating an S-parameter from an extracted topology (pre-route or post-route) to view losses on an interconnect takes seconds compared to hours with standalone tools. This ability to do quick iterative loss-budget trade-offs shortens the design time for MGH signals and more importantly, allows system designers to optimize the performance of such signals.

Channel Analysis

For differential signals that are used in serial link designs, you must ensure that timing and voltage margins are met. This is also referred to as *acceptable eye opening*.

Allegro PCB SI User Guide

Working with Multi-GigaHertz Interconnect

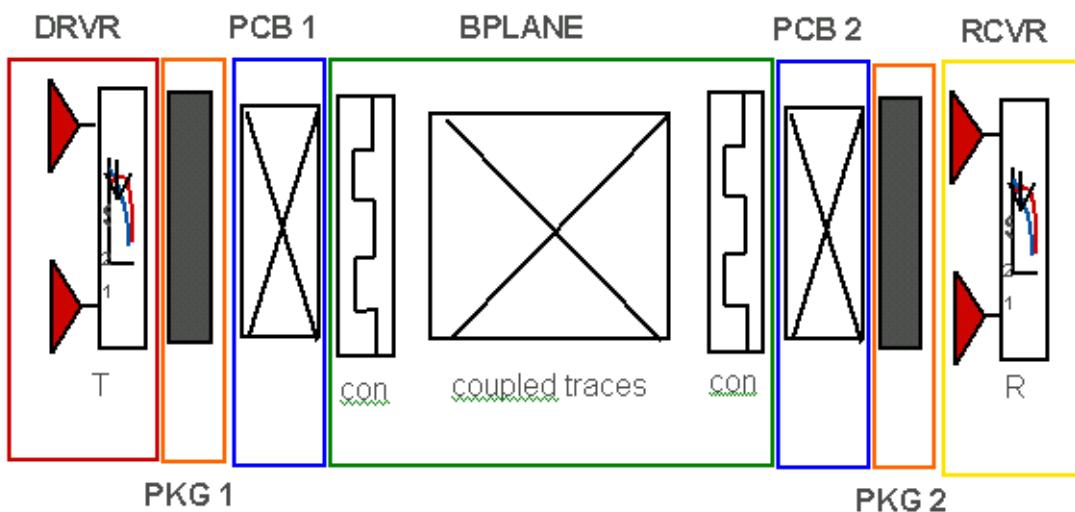
Traditional circuit simulators are limited to about 1024 bits of custom stimulus pattern length. This means that the effect of Inter-Symbol Interference (ISI) is not adequately modeled by traditional simulation solutions. To accurately predict the eye opening, you need simulation tools that can simulate stimulus patterns of 10,000 bits or more, and in some cases over 1 million bits.

The channel analysis technology within Allegro PCB SI Multi-Gigabit Option can simulate 10,000 bits in seconds on a typical Windows desktop platform. Such capability saves system designers the need to build multiple fully configured physical prototypes of the system to verify performance of the MGH interconnect in the lab.

The Serial Data Channel

Serial data links are generally comprised of 7 main building blocks as shown in the following figure.

Figure E-2 Block-level diagram of the serial data channel



The blocks represent the following:

- DRVR_IC - transistor level or behavioral differential driver, with configurable pre-emphasis settings
- PKG1 and PKG2 - detailed package parasitics, including die-to-C4 bump path and balls for BGAs
- PCB1 and PCB2 - pin escape traces, vias, and coupled traces on the PCBs
- Backplane - coupled connectors, large PTHs, coupled traces on the backplane (possibly pre-existing design)
- RCVR_IC - transistor level or behavioral differential receiver, with configurable equalization settings

Macro Modeling

Macro modeling has been a feature of Allegro PCB SI for many years. It offers nodal, behavioral and spice-like (E Spice) syntax that includes special elements unique to high-speed PCB design.

The advantages to using MacroModels for MGH applications are fairly clear. They ease the task of what-if analysis considerably. Verification of board level traces is practically impossible with transistor-level models, due to performance and convergence issues. Yet verification is feasible with MacroModels. The challenge is building or obtaining them.

The easiest way to develop MacroModels is to start with a well-documented working example of the type of model you need. It is much easier to edit an existing working model than create one from scratch. A library of MacroModel templates such as those available in Allegro PCB SI eases this task considerably. Well constructed and qualified MacroModel templates are essential to the success of the MacroModeling technology.

MacroModels provide a powerful behavioral device modeling capability. MacroModels can be used to model various elements in a serial link such as:

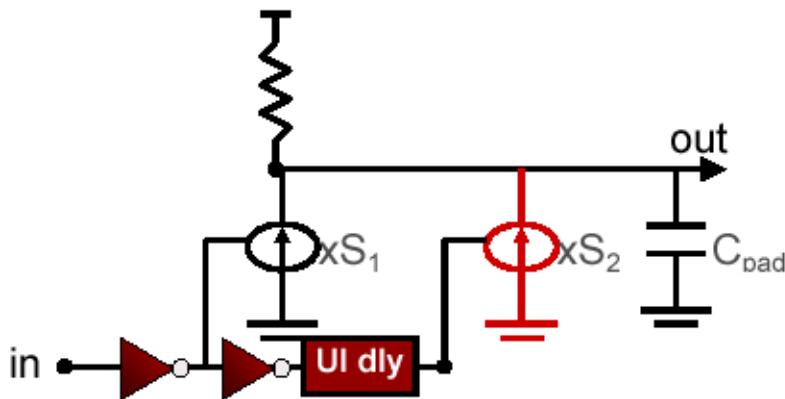
- active or passive driver pre-emphasis.
- active or passive receiver equalization.
- receiver amplification.

Carefully constructed MacroModels are capable of matching transistor-level device model accuracy while simulating at several hundred times the speed. For example it is not unusual to go from simulations of hours to just few minutes. The productivity gains you receive with quick turn around in both ‘what-if’ and verification type simulations cannot be understated.

Behavioral Model for a Driver with Pre-Emphasis

The guiding principle when developing a MacroModel is to keep the model as simple as possible. Simple models have much fewer adjustable parameters. These adjustable parameters are the key to success with MacroModels.

Figure E-3 Schematic for a driver with one tap of pre-emphasis



The model elements are composed of the behavioral controlled current sources, inverter, delay line, pad capacitance and the built-in die termination. The total current at the output is the combined current from the main current source and the delayed current source.

The simplest of these models will have just two classes of parameters; scaling factors “S” for the behavioral current sources and “C_{pad}”, for the pad capacitance. Variants of this model can include other transistor level effects like Miller capacitance. The more knowledge you have about the part, the more likely you will be able to develop simpler behavior models with fewer parameters.

An important part of the model development process is tuning these parameters to match the response observed for the vendor-supplied transistor level models. Model accuracy is certainly enhanced by employing parameter optimization techniques.

It should be noted that you can tune behavioral models equally well to measurements made on real silicon, circumventing the transistor level matching. Measurement-based techniques have the potential to yield both increased accuracy and efficiency in your models.

Correlation to Transistor-Level Models

Do not expect to replace transistor-level models with MacroModels. Detailed transistor-level models are useful for characterization and final verifications, using pre-defined worst-case bit patterns. However, MacroModels provide enormous productivity benefits in terms of simulation time, usability, and use model. They are also extremely useful when concurrent IC and PCB development is going on, especially when final silicon layout is not yet been frozen.

Obviously, correlation is very important for any model used to generate data for design decisions. You must correlate a MacroModel well to its associated transistor-level model to

use it reliably. And just like with transistor-level models, it is important to correlate measured lab results with the MacroModel to verify that the simulation it is producing is indicative of real-world behavior (although these measurements become increasingly challenging at Gbps rates).

Developing MacroModels for your serial drivers provides significant productivity gains. The MacroModel templates provided by Cadence are parameterized so that you can optimize the parameters for your particular application and technology.

You can achieve very close correlation with the behavioral MacroModels matching eye openings predicted by the transistor-level models within 1.5%. Sample comparisons are shown below. The transistor-level results are in red, and the MacroModel results with PCB SI are in blue.

Figure E-4 Overlay of MacroModel results vs. transistor-level Hspice waveforms

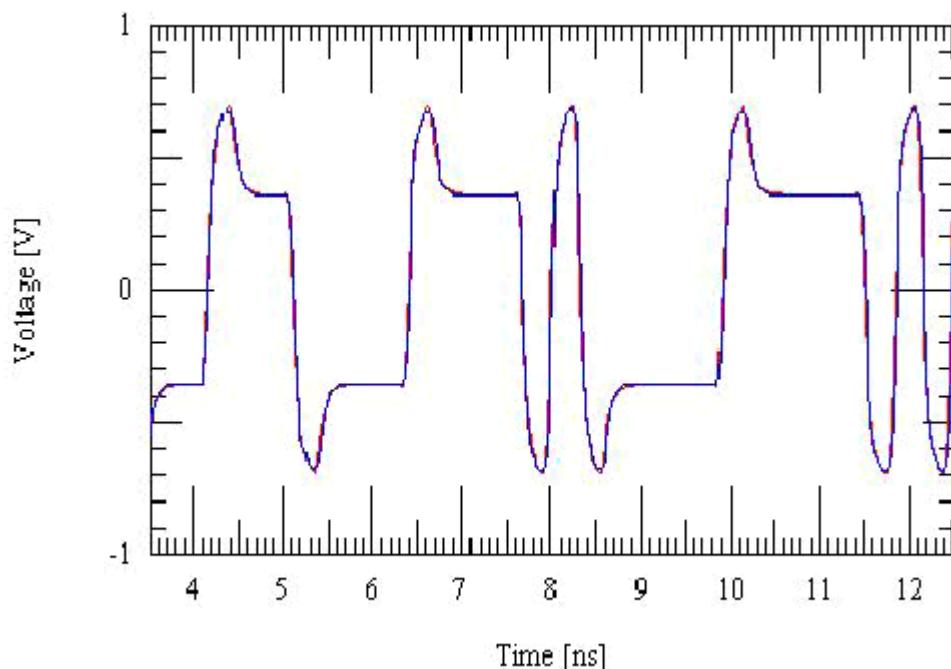
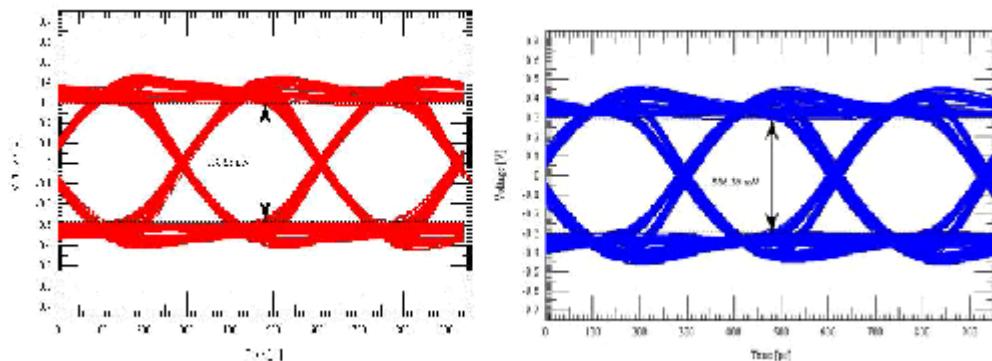


Figure E-5 Comparison of transistor-level Hspice eye pattern (red) and Allegro PCB SI MacroModel results (blue)



MacroModels also simulate up to 400 times faster than their transistor-level counterparts, enabling large bit streams to be easily run. As an added bonus, behavioral MacroModels are adjusted easily to match the behaviors of the actual silicon measured in the lab.

Building MacroModels

To build a MacroModel, you must collect the following data. Refer to [Figure E-6](#) on page 481 and [Table E-1](#) on page 481.

Normal IBIS data

- V_{tt}
- R_t
- Pulldown VI Curve
- Ramp rate (N)
- C_{comp}

Additional data

- Unit interval (UI dly)
- Pre-emphasis dB, or
 - Scale factor (x)

- If correlating, get a waveform of your silicon model into a known load.

Figure E-6 Driver Schematic

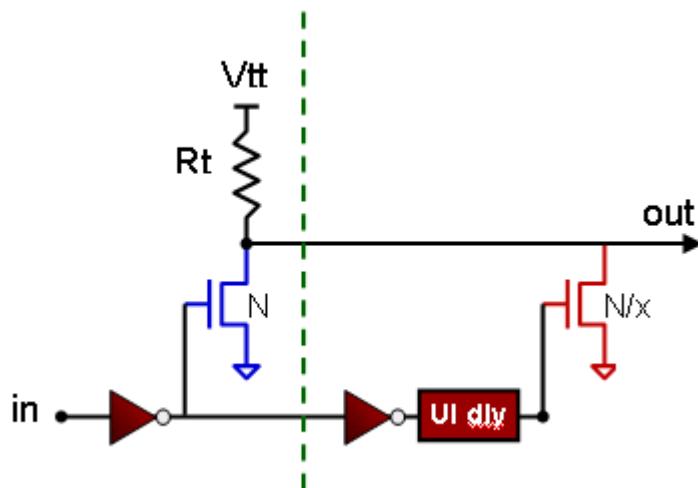


Table E-1 Where to Place Data in a Cadence MacroModel Template

Put this data . . .	in this section of the template
Vtt	(Pullup (ReferenceVoltage
Rt	rt
Pulldown Vi Curve	(Pulldown (VICurve
Ramp rate	(Ramp (dt
C_comp	(C_comp and/or padcap
Unit interval	bitp
Pre-emphasis dB	eqdb
- or -	
Scale factor (x)	cf1

Technical Notes

- Use [Ramp] data instead of VT curves.

- Works fine for these CML (non-push/pull) drivers.
 - Eliminates charge storage and over-clocking issues.
 - Much simpler to adjust and correlate.
- Some Tx designs exhibit miller capacitance effects and may cause slight miscorrelation when pulldown is on.

Note: You can use the models in both the Allegro® PCB SI and Package SI analysis environments.

It is important to verify your serial link channel design using simulation. In non-ideal channels, simulations need to comprehend tens and even hundreds of thousands of bit variations. You can build faster models to perform this kind of simulation by downloading MacroModel templates and modifying them as required, leveraging the various resources and examples mentioned previously.

For further information on creating MacroModels, refer to the [Allegro SI Device Modeling Language User Guide](#).

Via Modeling

Modeling via structures accurately over a very high frequency range is critical in MGH applications. Vias often represent some of the most significant discontinuities that you can find on PCB, package, and IC structures. Given their inherent 3D nature, they can cause severe signal integrity and EMI issues. In addition to signal degradation on the host net, via excitation of waveguide modes can propagate and radiate energy to neighbor nets and into space as well.

The via modeling capability is accurate well into the GHz frequency range. Since the modeling is analytical in nature, the computational cost is minimal compared with general-purpose 3D full-wave solvers. The electrical via model formats include narrowband, wideband, and scattering parameters (S parameters). You can easily create via models in PCB SI, add them as parts to a layout, perform what-if simulations, and perform channel analysis using SigWave. A distinct advantage to using via models is the ability to remove the vias from the topology quickly and easily (unlike with hardware prototypes) to see the impact the vias have on the channel.

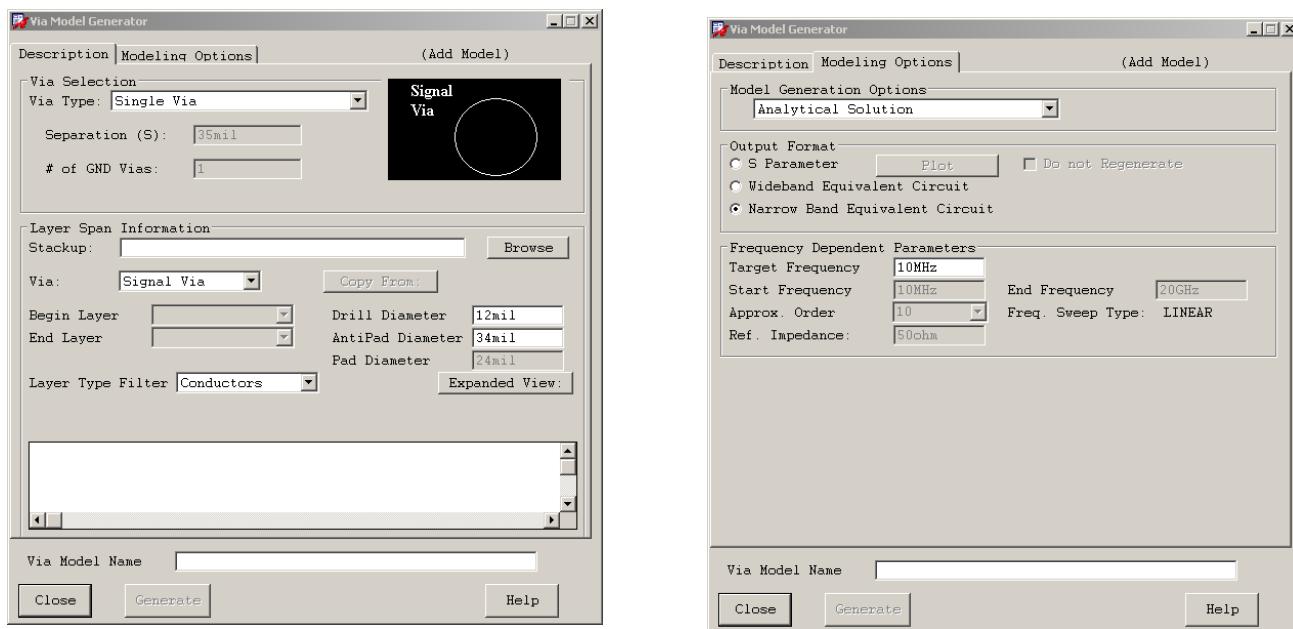
You can modify an existing via model or create one from scratch using the Via Model Generator dialog box shown in [Figure E-7](#).

Note: Before you create a new via model, be sure that the library you want to add it to is designated as the *working* library.

Allegro PCB SI User Guide

Working with Multi-GigaHertz Interconnect

Figure E-7 Tabbed Via Model Generator Dialog Box



To access the Via Model Generator dialog box

1. Choose *Analyze – Model Browser*.

The SI Model Browser appears.

2. Click the *IML Models* tab.

3. In the *Library Filter* field, select the required interconnect library.

A list of existing models in the selected interconnect library is displayed.

4. Click *Add* and choose one of the following from the list:

Closed Form Via
Narrow Band Via
Wide Band Via
S-Parameter Single Via
Signal/Signal Coupled Vias
Signal/Ground Coupled Vias
Signal/Power Coupled Vias
Stacked Coupled Vias

The Via Model Generator dialog box appears.

Via Model Formats

S-Parameter Format Details

- This is the most accurate via format. It should accurately capture the via behavior over the entire frequency range.
- Expect slower performance compared the circuit-based formats as more processing is required.
- *Start Frequency* for MGH applications is recommended at 10MHz. If DC convergence issues occur, you can drop to 1MHz (but no lower than 0.1MHz).
- *End Frequency* should be about 2/t_rise (1/t_rise minimum). Go up to 5/t_rise for greater accuracy, similar to when you use a fine waveform resolution like 5ps or 10ps.
- *No. of Freq Points* should be 128 points for most via models (this is the default value)

Note: If you include S-Parameter via models in larger S-Parameter circuits, their accuracy must be similar to that of the final circuit.

S-Parameter Settings Example

Edge Rate	Start Freq.	End Freq.	Bandwidth	No. of Freq. Points
100 ps	10 MHz	20GHz	20 GHz	128

Wideband Equivalent Circuit Details

- *Start Frequency* for MGH applications is recommended at 0MHz.
- *End Frequency* should be about 20 GHz.
- Leaving *Approx Order* set to 10 is recommended. You can increase it to 12 if *End Frequency* goes beyond 20GHz for improved accuracy.

Note: Setting *Approx Order* greater than 15 is not recommended.

- There is some loss of accuracy compared to the S Parameter format. However, simulation time is significantly faster.
- Convergence issues are possible if the frequency range is stretched too far.

Narrowband Equivalent Circuit Details

- The narrowband model is derived from the *Target Frequency*.
 - Use a target frequency that is near the middle of the energy content.
 - A good rule of thumb is $1/(1000 \times \text{risetime})$. For a driver with 100ps rise times, a target frequency of 10MHz is recommended.
 - If the *Target Frequency* is too high, then low frequency (DC losses) are dramatically overestimated.
 - If the *Target Frequency* is too low, then high frequency effects (skin effect and dielectric loss) are underestimated. However, these are small effects in a via.
- This is the least accurate of the via model formats. However, it is very stable and simulates very quickly.

Via Model Types

SI lets you generate and add both single and coupled vias:

```
Single Via
Coupled Signal Vias
Coupled Signal and GND Vias
Coupled Signal and PWR Vias
```

Note: Power-and-signal vias require an external voltage source.

Coupled via symbols are distinguished from single signal via symbols, as shown below.

Figure E-8 Single Signal Via Symbol

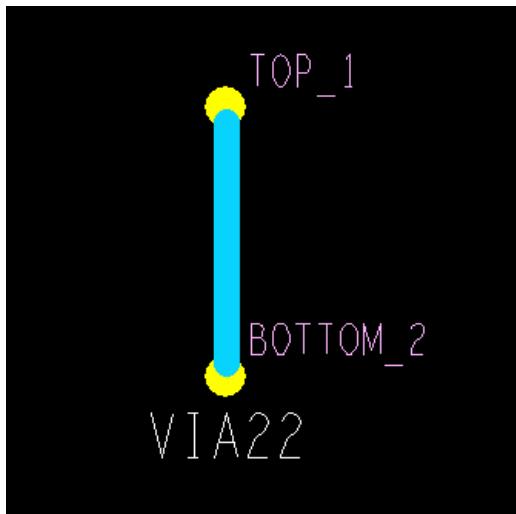
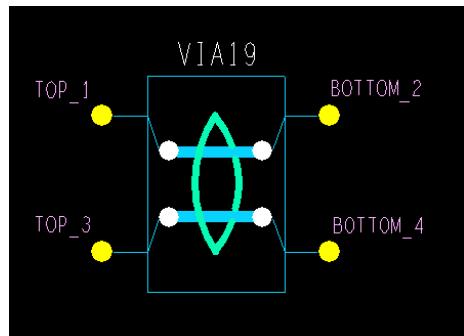
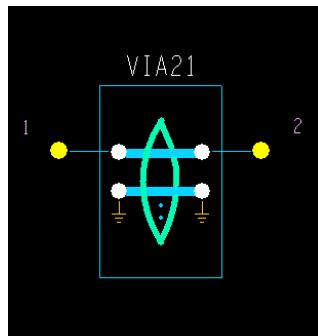


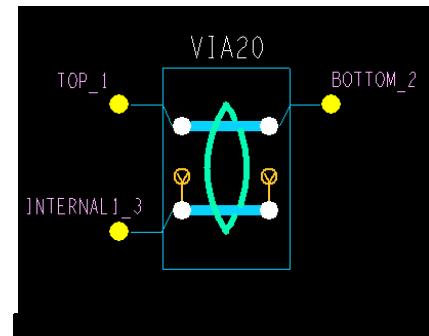
Figure E-9 Coupled Via Symbols



Signal-and-signal Via



Ground-and-signal Via



Power-and-signal Via

Modeling in the Interconnect Description Language

Overview

The Interconnect Description Language (IDL) is the language used by both SigNoise and device model developers. The language is an extension of SPICE and consists of control characters, keywords, and values. SigNoise uses IDL to write models for the connect line segments, vias, shapes, and pins in designs. You can modify these models. Device model developers use IDL to write models for

- Packages. Package models describe the parasitics between a component's pads and the die of the device within.
- RLGC matrices. RLGC models (matrices of resistance, inductance, conductance, and capacitance values) specify the parasitics in the connections used in system design links.
- Passive components. For example, resistors and capacitors.

This appendix describes IDL interconnect models for connect line segments, shapes, pins, and vias, and shows how you might modify such models.

IDL Interconnect Line Segment Models

IDL interconnect line segment models can describe one or more of the following:

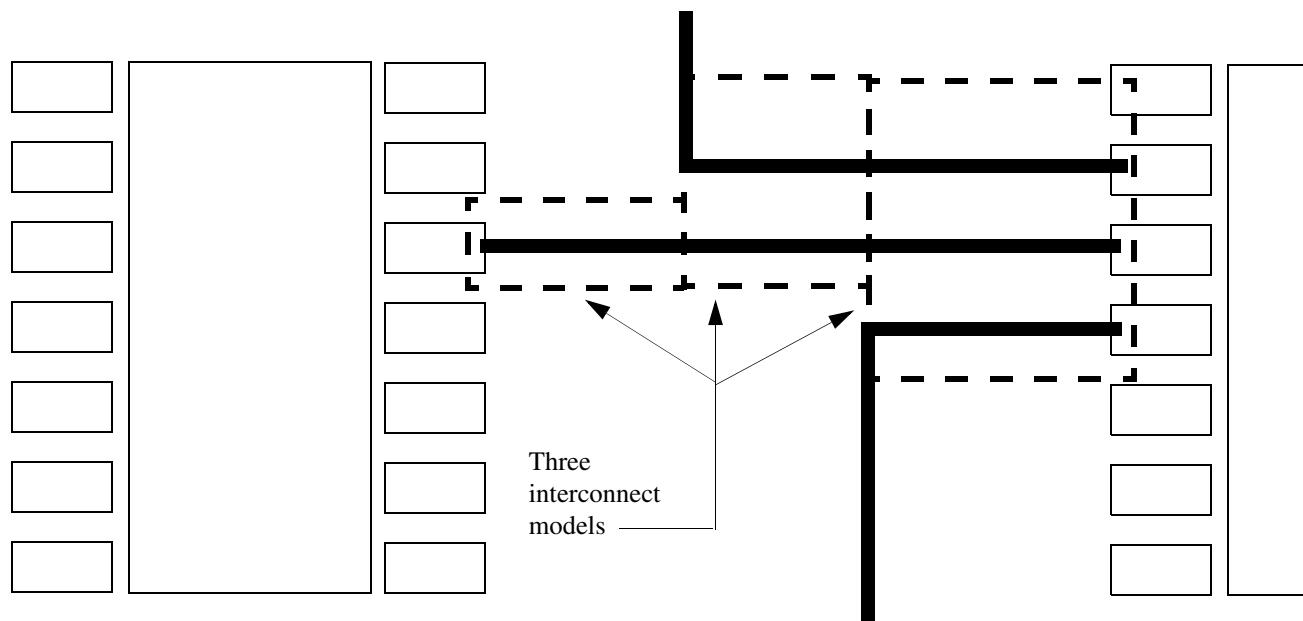
- Connect lines
- Connect line segments
- Parts of connect lines
- Parts of connect line segments

When you route connect lines within the distance of the geometry window, the resulting interconnect models describe more than one connect line, connect segment, or part of a connect segment.

The following figure shows the three interconnect models that SigNoise writes for the horizontal connect lines and horizontal connect line segments in the illustration.

- The model on the left describes the left section of the middle connect line.
- The model in the middle describes both the left section of the connect line segment on top and the central section of the middle connect line.
- The model on the right describes the right section of the connect line segment on top, the right section of the middle connect line, and the connect line segment on the bottom.

Figure F-1 Interconnect Models for Connect Line Segments



RLGC Matrix Values in Interconnect Models

Interconnect models contain matrices of resistance, inductance, conductance, and capacitance (or RLGC) values. The SigNoise field solvers calculate and write these values into the model.

Allegro PCB SI User Guide

Modeling in the Interconnect Description Language

You might want to edit the RLGC values in line segment models to change how SigNoise simulates the corresponding connect line segments. To do this, edit the RLGC matrix values in the `.rlgc` declaration in the line segment model.

The `.rlgc` declaration is located toward the end of the model, and contains parasitic value matrices. The following is an example of an `.rlgc` declaration.

```
.rlgc RLGCMTL_1S_2R_2914 ( Length=length N=2 )
.C 0
+ 6.625200e-11 -4.567200e-12
+ -4.567500e-12 5.729800e-11
.L 0
+ 4.834800e-07 7.706100e-08
+ 7.705900e-08 4.388100e-07
.G 0
+ 0.000000e+00 -0.000000e+00
+ -0.000000e+00 0.000000e+00
.R 0
+ 3.586500e+00 0.000000e+00
+ 0.000000e+00 1.793200e+00
.endrlgc RLGCMTL_1S_2R_2914
```

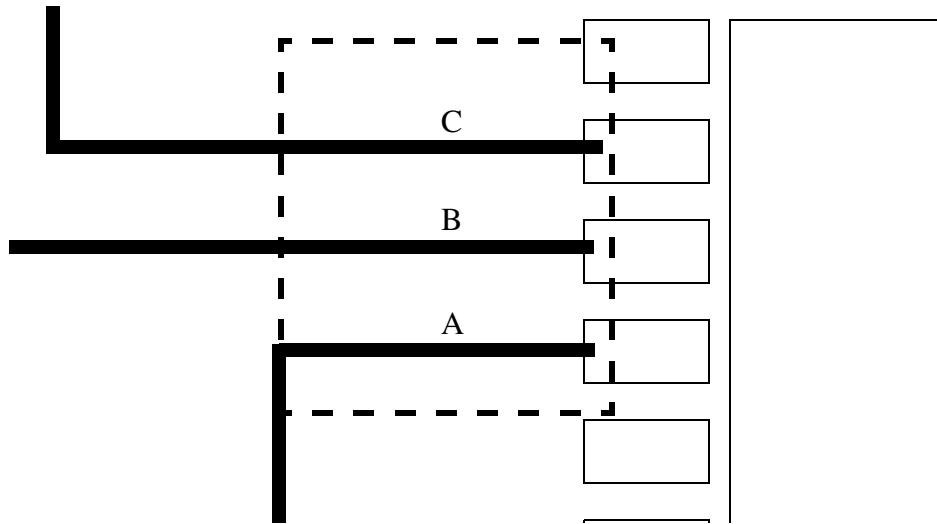
The `.rlgc` declaration is a type of subcircuit inside a subcircuit that is written into the model by the field solvers.

The declaration includes matrices that specify the *self* and *mutual* parasitic values for the connect lines, segments, and parts in the model and between the connect lines, segments, and parts.

- A self value is the parasitic value of an individual connect line, segment, or part with respect to some reference, such as a ground plane.
- A mutual value is the parasitic values between connect lines, segments, and parts.
- The capacitance matrix is a Maxwell Capacitance Matrix. In this form, the c11 and c22 values are not the self capacitance, but the loaded capacitance. The C12 and C21 values are the negative mutual capacitance values.

The more connect lines, segments, and parts in a model the larger the size of the matrices in the `.rlgc` declaration. A model for a single connect line, segment, or part contains matrices that specify only the self value of that connect line, segment, or part. A model for two connect lines, segments, or parts has matrices of two self values and two mutual values. A model for three connect lines, segments, or parts, such as that in the figure below, has matrices of three self values and six mutual values.

Figure F-2 Model for Three Interconnect Lines



The model illustrated in the previous figure describes connect line segments or parts *A*, *B*, and *C*. The self values in the matrices have the following arrangement in the matrix:

A

B

C

A matrix with the self and mutual values has the following arrangement:

A	A-B	A-C
B-A	B	B-C
C-A	C-B	C

In this matrix *A-B* is the mutual value between *A* and *B*, *C-A* is the mutual value between *C* and *A*. Mutual values between the same lines, segments, or parts are identical, so the *B-C* mutual value equals the *C-B* mutual value.

.rlgc Declaration Syntax

The syntax of the `.rlgc` declaration for a model with two lines, segments, or parts is:

```
.rlgc subcircuit_name (Length=value N=value)
.C frequency_value
+ loaded_value      neg.mutual_value
+ neg.mutual_value  loaded_value
.L frequency_value
+ self_value        mutual_value
+ mutual_value     self_value
.G frequency_value
+ self_value        mutual_value
+ mutual_value     self_value
.R frequency_value
+ self_value        mutual_value
+ mutual_value     self_value
.endrlgc subcircuit_name
```

The following table shows the keywords and values used in the `.rlgc` declaration syntax

Table F-1 Keywords and Values

<code>.rlgc</code>	A keyword that specifies the beginning of the declaration that contains the matrices of parasitic values.
<code>subcircuit_name</code>	The SigNoise field solvers generate the matrix values as a special kind of subcircuit of the model and names this subcircuit. The name is a derivative of the model name.
<code>Length=value</code>	A keyword and value statement that specifies the length of the connect lines, segments, or parts. The field solvers can enter the keyword <i>length</i> for this value to tell SigNoise to take the length value from the layout instead of from the model.
<code>N=value</code>	A keyword and value statement that specifies both the number of lines, segments, or parts modeled and the size of the matrix. A value of 2, for example, specifies two lines, segments, or parts, a 2x2 matrix value specifies two self values and two mutual values.
<code>.C</code>	Specifies a matrix of capacitance values.
<code>.L</code>	Specifies a matrix of inductance values.
<code>.G</code>	Specifies a matrix of conductance values.

Table F-1 Keywords and Values, *continued*

.R	Specifies a matrix of resistance values.
frequency_value	Specifies the minimum frequency to which SigNoise applies the parasitic values in the following matrix.
+	A continuation character that indicates that a line is part of a declaration started on a previous line.
self_value	The parasitic value of a connect line, segment, or part.
mutual_value	The parasitic value between two connect lines, segments, or parts.
loaded_value	The parasitic value of a connect line, segment, or part including loading effects from neighboring lines.
neg. mutual_value	Negative mutual parasitic value.

Example Line Segment Model

The following example shows an IDL model for two neighboring connect line segments. The sections following the example describe the model according to its functional parts.

```
.subckt MTL_1S_2R_2914
+X1250Y800L1
+X1225Y800L1
+0
+X1250Y2650L1
+X1225Y2650L1
+0
.material sml6 dielectric=4.5 losstangent=0.001
.material sml5 conductivity=5.959e+07 losstangent=0
.material sml4 dielectric=4.5 losstangent=0.001
.material sml3 conductivity=5.959e+07 losstangent=0
.material sml2 dielectric=4.5 losstangent=0.001
.material m17 conductivity=3.43e+07
.material m16 dielectric=4.5
.material m15 conductivity=5.959e+07
.material m14 dielectric=4.5
.material m13 conductivity=5.959e+07
.material m12 dielectric=4.5
.material m11 conductivity=3.43e+07
.layerstack Layerstack3
```

Allegro PCB SI User Guide

Modeling in the Interconnect Description Language

```
+shield( 3.048e-05 1 0 )
+dielectric( 0.0003048 4.5 0.001 )

.crosssection
+rectangle ( 3.43e+07 0 0.0003048 0.0001524 0.00035814 )
+rectangle ( 3.43e+07 0.0005588 0.0003048 0.0008636 0.00035814 )
+Length=length
.rlgc RLGCMTL_1S_2R_2914 ( Length=length N=2 )
.C 0
+ 6.625200e-11 -4.567200e-12
+ -4.567500e-12 5.729800e-11
.L 0
+ 4.834800e-07 7.706100e-08
+ 7.705900e-08 4.388100e-07
.G 0
+ 0.000000e+00 -0.000000e+00
+ -0.000000e+00 0.000000e+00
.R 0
+ 3.586500e+00 0.000000e+00
+ 0.000000e+00 1.793200e+00
.endrlgc RLGCMTL_1S_2R_2914
```

**The Characteristic Modal Delay, Admittance and
**Impedance Matrices of these Transmission Lines:

*Delay Matrix.
*Td 0 (n=2)
* 5.706200e-09 0.000000e+00
* 0.000000e+00 4.889800e-09

*Admittance Matrix.
*Y 0 (n=2)
* 1.185600e-02 -1.415100e-03
* -1.415100e-03 1.158100e-02

*Impedance Matrix.
*Z 0 (n=2)
* 8.559500e+01 1.045800e+01
* 1.045800e+01 8.762400e+01
.ends T_1S_2R_291

The following sections describe IDL control characters, keywords, and values as you encounter them from beginning to end in the model:

- *.subckt*
- *.material*
- *.layerstack*
- *.crosssection*
- *.rlgc*
- *Delay, Admittance, and Impedance*

.subckt Declaration

The sample interconnect model begins with the *.subckt* declaration:

```
.subckt MTL_1S_2R_2914
+X1250Y800L1
+X1225Y800L1
+0
+X1250Y2650L1
+X1225Y2650L1
+0
```

This declaration specifies that the model is a subcircuit of a circuit that SigNoise simulates. It specifies the following:

- Name of the subcircuit, *MTL_1S_2R_2914*

- External nodes of the circuit

```
+X1250Y800L1
+X1225Y800L1
+0
+X1250Y2650L1
+X1225Y2650L1
+0
```

The plus sign (+) is a continuation control character that specifies that a line is a continuation of a declaration in the previous line.

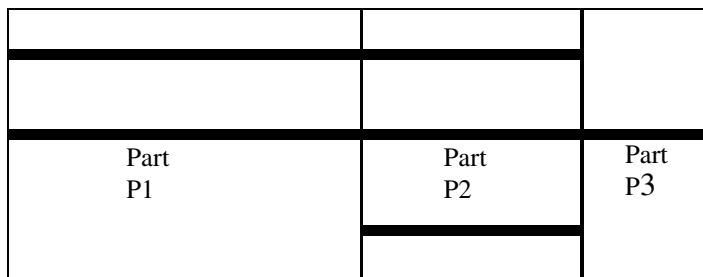
The zeroes in this list of external nodes indicate a reference to a ground plane and separate the input external nodes from the output external nodes.

To name the subcircuit, SigNoise used

- The *MTL* prefix identifying this as a multi-trace model
- The *1S* component indicating one shield layer found in the trace model
- The *2R* component indicating two rectangular conductors found in the trace model
- The *2914* component as an arbitrary number to differentiate this model from other trace models

The following figure shows how SigNoise can write models for a part of a connect line segment.

Figure F-3



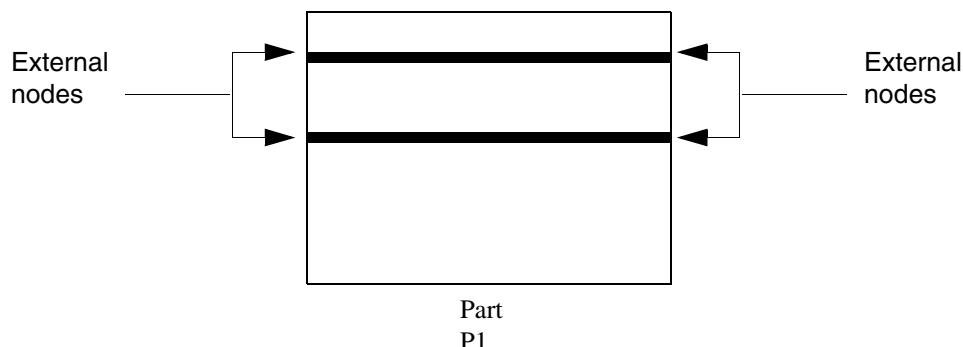
In this figure, SigNoise writes a model for the following:

- *Part P1* of the middle connect line segment and part of the top connect line segment
- *Part P2* of the middle connect line segment as well as part of the top and bottom connect line segments
- *Part P3* of the middle connect line segment

The external nodes of a model show where the model connects to other parts of the circuit. The following figure shows the external nodes in the model for *Part P1*.

Allegro PCB SI User Guide
Modeling in the Interconnect Description Language

Figure F-4

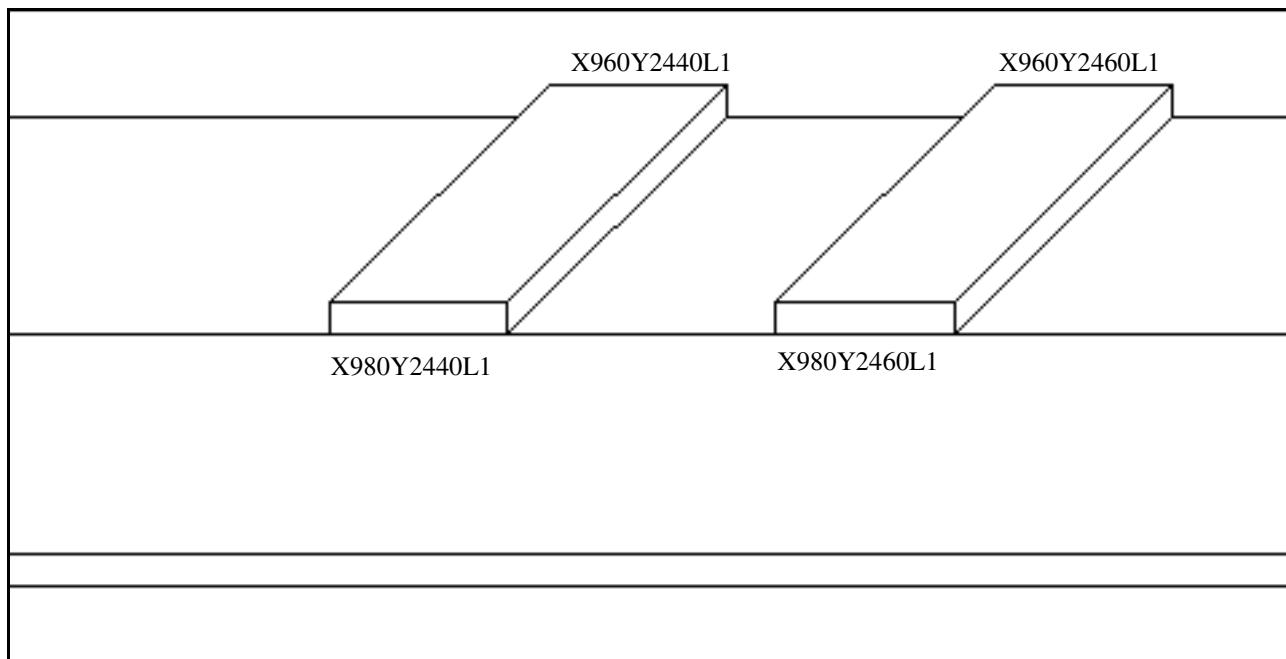


To name the external nodes of the subcircuit, SigNoise combined

- The left most X coordinate of the external node
- The lowest Y coordinate of the external node
- The layer of the external node

Figure A-5 shows how the model appears in the Sigssect Cross-Section window, and labels the external nodes.

Figure F-5



.layerstack Declaration

The next part of the sample model is the `.layerstack` declaration.

```
.layerstack Layerstack3
+shield( 3.048e-05 1 0 )
+dielectric( 0.0003048 4.5 0.001 )
```

A layerstack is a stack of layers in the design bounded by the surface of the board, a power plane, or a ground plane. A board can have more than one layerstack. The first layerstack, *Layerstack1*, is the stack of layers that begins with the bottom surface of the board and ends at a power plane, ground plane, or the top surface of the board. The `.layerstack` declaration in this model specifies a model of connect lines in the third layerstack from the bottom, *Layerstack3*.

This layerstack consists of one shield layer and one dielectric layer because the declaration contains only one instance of the keywords `shield` and `dielectric`. If, for example, the layerstack contained more than one dielectric layer, the model would have more than one line beginning with the plus continuation control character (+) and the keyword `dielectric`.

The values in parentheses that follow the keywords `shield` and `dielectric` specify in the following order:

- The thickness of the layer
- The dielectric constant of the layer
- The loss tangent of the layer

Note: All layers need these three values. SigNoise applies a dielectric constant value of 1 to a shield layer as a placeholder even though a shield layer has no dielectric constant.

.crosssection Declaration

After the `.layerstack` section, the next part of the sample model is the `.crosssection` declaration.

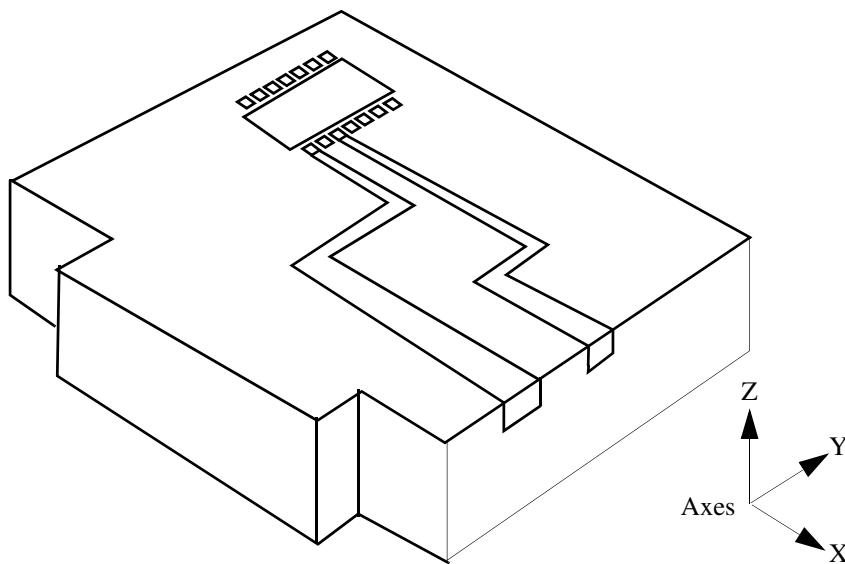
```
.crosssection
+rectangle ( 3.43e+07 0 0.0003048 0.0001524 0.00035814 )
+rectangle ( 3.43e+07 0.0005588 0.0003048 0.0008636 0.00035814 )
```

The `.crosssection` declaration specifies the geometry of a cross section of the parts on connect line segments in the model. For each part of a segment in the model there is a line beginning with `+rectangle` and, in parentheses, values for

- The conductivity of the part of the segment

- The four coordinates of the lower left and upper right corners of cross sections of these parts of segments

These coordinates are for the X and Z axes in models for vertical segments and the Y and Z axes in horizontal segments, as shown in the following figure.



These coordinates show the positions of the parts of the segments relative to each other. One X or Y coordinate value is always 0 and the other coordinate values specify the relative distances of the other lower left or upper right corners to that 0 value.

The line `+Length=length` specifies that `SigNoise` takes the lengths of the parts of the segments from the layout instead of from the model. Specifying a value instead of `length` is valid syntax for an IDL model.

.rlgc Declaration

After the `.crosssection` section, the next part of the sample model is the `.rlgc` declaration.

```
.rlgc RLGCMTL_1S_2R_2914 ( Length=length N=2 )
.C 0
+ 6.625200e-11 -4.567200e-12
+ -4.567500e-12 5.729800e-11
.L 0
+ 4.834800e-07 7.706100e-08
+ 7.705900e-08 4.388100e-07
.G 0
```

Allegro PCB SI User Guide

Modeling in the Interconnect Description Language

```
+ 0.000000e+00 -0.000000e+00
+ -0.000000e+00 0.000000e+00
.R 0
+ 3.586500e+00 0.000000e+00
+ 0.000000e+00 1.793200e+00
.endrlgc RLGMTL_1S_2R_2914
```

The `.rlgc` declaration begins with a name for the subcircuit. This name is provided by the field solver. The field solver replaces the *STL* or *MTL* prefix for the model subcircuit name with the *RLGC* prefix. In this model, the name of the `.rlgc` subcircuit is `RLGMTL_1S_2R_2914`.

After the name of the `.rlgc` subcircuit there is, in parentheses, a reiteration of the *Length=length* statement from the `.crosssection` declaration and the size statement *N=2*. The *N=2* statement specifies that the `.rlgc` subcircuit provides parasitic values for two connect lines, segments, or parts and that the matrices that specify these values in the `.rlgc` subcircuit have a dimension of 2 by 2 values.

The remainder of the `.rlgc` subcircuit contains `.C`, `.L`, `.G`, and `.R` declarations of capacitance, inductance, conductance, and resistance value matrices.

All of these matrices have the same format for specifying self and mutual parasitic values. The matrix lists self values diagonally from the top left to the bottom right of the matrix. The self value on the top of the matrix is the value of the left-most part of a connect line segment in a cross sectional view of the design. The matrix lists mutual values to the left or right of the self values.

For example, the following section of code is the capacitance value matrix:

```
.C 0
+ 6.625200e-11 -4.567200e-12
+ -4.567500e-12 5.729800e-11
```

In this matrix, the part of a segment that appears to the left in a cross section has a self capacitance of `6.625200e-11`. The capacitance between the left segment part and the right segment part is `-4.567200e-12`. The right part has a self capacitance of `5.729800e-11` and the capacitance between the right and left segment parts is once again `-4.567500e-12`.

In this capacitance value matrix, the value to the right of the `.C` keyword is the frequency value. In this model, it specifies that SigNoise apply the matrix values for all frequencies greater than zero.

Characteristic Modal Delay, Admittance, and Impedance Matrices

The last part of the example line segment model is the Characteristic Modal Delay, Admittance, and Impedance section declaration.

```
**The Characteristic Modal Delay, Admittance and
**Impedance Matrices of these Transmission Lines:

*Delay Matrix.
*Td 0 (n=2)
* 5.706200e-09 0.000000e+00
* 0.000000e+00 4.889800e-09

*Admittance Matrix.
*Y 0 (n=2)
* 1.185600e-02 -1.415100e-03
* -1.415100e-03 1.158100e-02

*Impedance Matrix.
*Z 0 (n=2)
* 8.559500e+01 1.045800e+01
* 1.045800e+01 8.762400e+0
*Odd Mode Impedances, 2*(z11-z12).
* Z(odd) = 1.073500e+002
*Even Mode Impedances, (z11+z12)/2.
* Z(even) = 3.826600e+001
**The Near-End Crosstalk Coefficient of these Transmission
**Lines based on Near-End Resistance=50 ohm assumption:
*Near-End Crosstalk Coefficient Matrix, Z0*Inv(R+Z0).
* 0 (n=2)
* 5.612800e-001 4.355700e-002
* 4.355700e-002 5.612800e-001
.ends MTL_1S_2R_4413
")
("KSPICE"
"DATAPOINTS RLGC MTL_1S_2R_4413
FREQUENCY=0
CMATRIX
8.922200e-011 -1.048100e-011
-1.048100e-011 8.922200e-011
LMATRIX
3.742200e-007 8.696900e-008
```

```
8.696900e-008 3.742200e-007
GMATRIX
0.000000e+000 0.000000e+000
0.000000e+000 0.000000e+000
RMATRIX
4.335200e+000 0.000000e+000
0.000000e+000 4.335200e+000
END RLG
")
("Frequency" "0")
)
```

IDL Via Models

SigNoise generates IDL models for single and coupled vias. You can edit these models if you want to change the values specified in them by SigNoise algorithms.

Example Via Model

The following example shows an IDL model for a single via. The sections following the example describe the statements in a single via model and where you might want to change the SigNoise values in the model.

```
.subckt VIA_POAR_VIA_L1A0W600L7A135W600
+L1A0W600 L7A135W600 SL3 SL9
.material sml9 conductivity=5.959e+07 losstangent=0
.material sml8 dielectric=4.5 losstangent=0.001
.material sml6 dielectric=4.5 losstangent=0.001
.material sml4 dielectric=4.5 losstangent=0.001
.material sml3 conductivity=5.959e+07 losstangent=0
.material sml2 dielectric=4.5 losstangent=0.001
.material sml10 dielectric=4.5 losstangent=0.001
.material m19 conductivity=5.959e+07
.material m18 dielectric=4.5
.material m17 conductivity=5.959e+07
.material m16 dielectric=4.5
.material m15 conductivity=5.959e+07
.material m14 dielectric=4.5
.material m13 conductivity=5.959e+07
```

Allegro PCB SI User Guide

Modeling in the Interconnect Description Language

```
.material m12 dielectric=4.5
.material m111 conductivity=3.43e+07
.material m110 dielectric=4.5
.material m11 conductivity=3.43e+07
.layerstack LayerStackAll
+dielectric( sml10 0.00019304 )
+shield( SL9 sml9 3.048e-05 )
+dielectric( sml8 0.0001524 )
+dielectric( sml6 0.00079248 )
+dielectric( sml4 0.00018288 )
+shield( SL3 sml3 3.048e-05 )
+dielectric( sml2 0.0001397 )
.Via L1A0W600 L7A135W600 SL3 SL9
+ pad( 0.00152146 0.0015748 ellipse(m11 0.0 0.0 0.0014224 0.0014224))
+ void( 0.00135128 0.00138176 ellipse(m13 0.0 0.0 0.0014224 0.0014224) )
+ pad( 0.0011684 0.00119888 ellipse(m15 0.0 0.0 0.0014224 0.0014224))
+ pad( 0.00037592 0.0004064 ellipse(m17 0.0 0.0 0.0014224 0.0014224))
+ void( 0.00019304 0.00022352 ellipse(m19 0.0 0.0 0.0014224 0.0014224) )
+ pad( 0 5.334e-05 ellipse(m111 0.0 0.0 0.0014224 0.0014224))
+ drill(5.334e-05 0.00152146 ellipse(m11 0.0 0.0 0.0003429 0.0003429))
+ trace( L1A0W600 0 rectangle(sml1 0.0 0.00152146 0.0001524 0.0015748))
+ trace( L7A135W600 135 rectangle(sml7 0.0 0.00037592 0.0001524 0.0004064))
*There is a PIN named U23.1 here.
*There is a VIA here.
* Closed-form formula solution
C0 L7A135W600 0 3.04105e-14
C1 L1A0W600 0 3.04105e-14
R01 L7A135W600 L1A0W600 1e-7 L= 1.37277e-09
.ends VIA_POAR_VIA_L1A0W600L7A135W600
```

.subckt Declaration

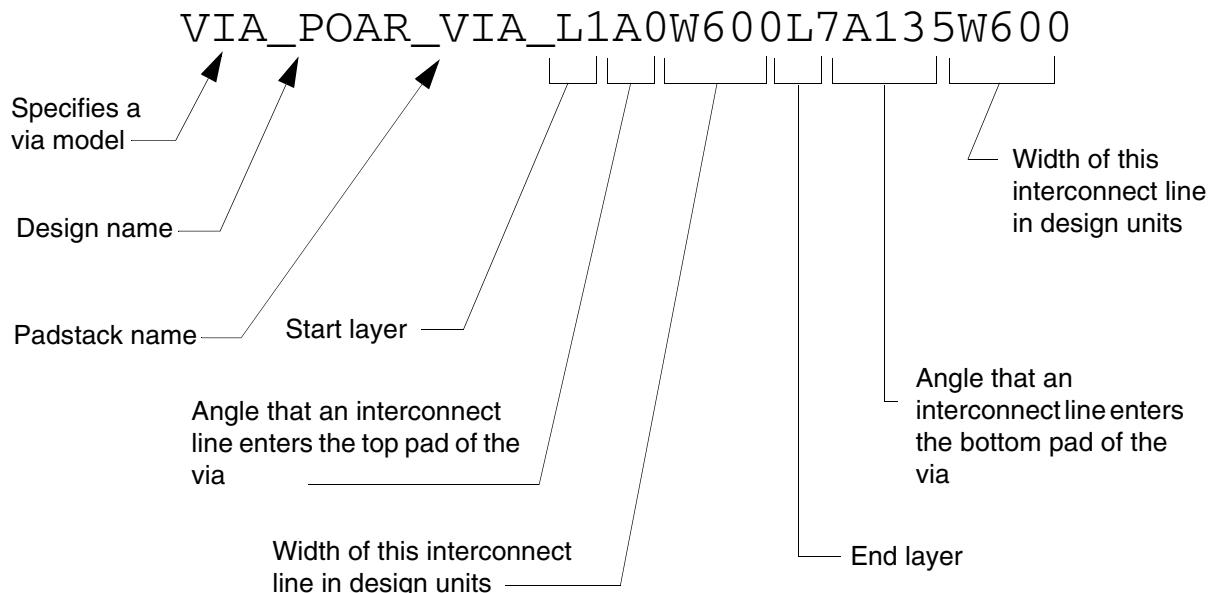
The via model begins with a **.subckt** declaration:

```
.subckt VIA_POAR_VIA_L1A0W600L7A135W600
+L1A0W600 L7A135W600 SL3 SL9
```

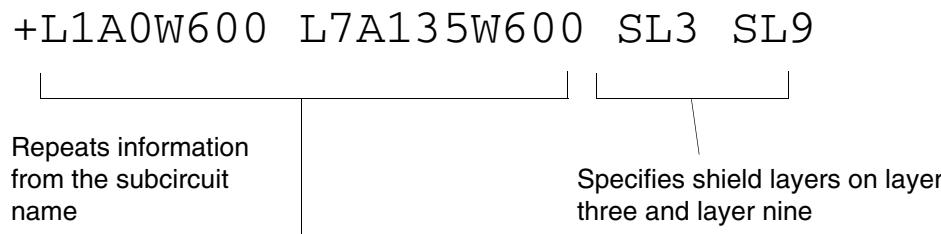
Allegro PCB SI User Guide

Modeling in the Interconnect Description Language

The subcircuit name specifies the following information:



The following information follows the subcircuit name:



.layerstack Declaration

Via models use a special type of layer stack called `LayerStackAll` that includes all the layers in the design. In the following `.layerstack` declaration, each line is for a layer of the design.

```
.layerstack LayerStackAll
+dielectric( sml10 0.00019304 )
+shield( SL9 sml9 3.048e-05 )
+dielectric( sml8 0.0001524 )
```

Allegro PCB SI User Guide

Modeling in the Interconnect Description Language

```
+dielectric( sml6 0.00079248 )
+dielectric( sml4 0.00018288 )
+shield( SL3 sml3 3.048e-05 )
+dielectric( sml2 0.0001397 )
```

The first shield layer line contains the following information:

```
+shield( SL9 sml19 3.048e-05 )
```

Specifies a shield layer

Shield layer notation from the .subckt declaration

Thickness of the layer

Material layer placeholder from the .material declaration

.via Declaration

The `.Via` declaration specifies geometry information about via elements such as pads, voids, drill holes, and the interconnect lines that connect to the via. In the following `.Via` declaration, each line is for a pad or void in the via or for the drill hole or the interconnect lines that connect to the via.

```
.Via L1A0W600 L7A135W600 SL3 SL9
+ pad( 0.00152146 0.0015748 ellipse(ml1 0.0 0.0 0.0014224 0.0014224) )
+ void( 0.00135128 0.00138176 ellipse(ml3 0.0 0.0 0.0014224 0.0014224) )
+ pad( 0.0011684 0.00119888 ellipse(ml5 0.0 0.0 0.0014224 0.0014224) )
+ pad( 0.00037592 0.0004064 ellipse(ml7 0.0 0.0 0.0014224 0.0014224) )
+ void( 0.00019304 0.00022352 ellipse(ml9 0.0 0.0 0.0014224 0.0014224) )
+ pad( 0 5.334e-05 ellipse(ml11 0.0 0.0 0.0014224 0.0014224) )
+ drill(5.334e-05 0.00152146 ellipse(ml1 0.0 0.0 0.0003429 0.0003429) )
+ trace( L1A0W600 0 rectangle(sml1 0.0 0.00152146 0.0001524 0.0015748) )
+ trace( L7A135W600 135 rectangle(sml7 0.0 0.00037592 0.0001524
                                         0.0004064) )
```

The first pad line contains the following information:

```
+ pad( 0.00037592 0.0004064 ellipse(m17 0.0 0.0 0.0014224 0.0014224))
```

The diagram shows the following annotations:

- An arrow points to the opening parenthesis of the `pad` function, with the text: "Specifies that the geometry information is for a pad".
- An arrow points to the word `ellipse`, with the text: "Specifies that the pad is an ellipse".
- An arrow points to the parameter `m17`, with the text: "Specifies the material layer from the .material declaration".
- An arrow points to the four coordinate values at the end of the line, with the text: "Specifies the four coordinates of a bounding box around the ellipse of the pad".
- A bracket under the Z-axis coordinates `0.00037592` and `0.0004064` is labeled "The Z axis coordinates for the pad".

Closed-Form Solution

At the end of the via model is a SPICE circuit description that represents the behavior of the via. SigNoise first uses a closed-form method to generate values for this circuit description.

The following is an example of a closed-form solution:

```
* Closed-form formula solution
C0 L7A135W600 0 3.04105e-14
C1 L1A0W600 0 3.04105e-14
R01 L7A135W600 L1A0W600 1e-7 L= 1.37277e-09
```

In this example, two capacitors and an inductor are used to represent the behavior of the via. You can change these values when you edit a via model.

The capacitance and inductance values in this solution automatically come from a closed-form solution. You might want to use the more accurate values that come from the SigNoise three-dimensional field solver.

Coupled Multiple Vias

The following examples show IDL models for coupled vias. The statement descriptions for coupled via models are similar to those for single via models.

.subckt Declaration

Vias are recognized as coupled if they are located in the same subcircuit, as shown below.

```
.subckt subcircuit_name node_1 node_2
```

```
.via via_node_list  
.via via_node_list  
. .  
.end subcircuit_name
```

.layerstack Declaration

In the layerstack declaration section of a coupled via model, connections to ground and/or power planes are identified through the mapping of via nodes to plane terminals, as shown below.

```
+shield ( Plane_1_terminal_name Plane_1_conductivity Plane_1_thickness )  
+shield ( Plane_2_terminal_name Plane_2_conductivity Plane_2_thickness )
```

Additional Differentiators

In addition to the declarations addressed above, the following conditions apply to coupled via modeling:

- Vias are connected to planes when their antipad diameters are less than the via drill diameters
- Via terminals are allowed inside the antipad regions as long as the terminal is identified on the subcircuit node list
- Antipads in individual vias may duplicate one another in instances where multi-drills go through the same antipad

Allegro PCB SI User Guide

Modeling in the Interconnect Description Language

Example Edge Coupled Differential Via Pair

```
*There is a VIA here.  
.subckt VIA_DP  
+L15A0W4500 L27An45W6000 L15A180W4500 L27An135W6000  
.layerstack LayerStackAll  
+dielectric( 2.2 0.000119634 )  
+shield( SL23 5.959e+07 1.8034e-05 )  
+dielectric( 2.2 0.001524 )  
+shield( SL17 5.959e+07 1.8034e-05 )  
+dielectric( 2.2 0.000119634 )  
.Via L15A0W4500 L27An45W6000  
+ pad( -1.8034e-5 0.0 ellipse(5.959e+07 0.0 0.0 0.000924 0.000924))  
+ void( 0.000119634 1.37668e-4 ellipse(5.959e+07 0.0 0.0 0.003048 0.003048))  
+ void( 0.001661668 0.001679702 ellipse(5.959e+07 0.0 0.0 0.003048 0.003048) )  
+ pad( 0.001799336 0.00181737 ellipse(5.959e+07 0.0 0.0 0.000924 0.000924))  
+ drill( 0.0 0.001799336 ellipse(m11 0.0 0.0 0.000914 0.000914))  
+ trace( L15A0W4500 0 rectangle(5.959e+07 0.0 0.001799336 0.0001143 0.00181737))  
+ trace( L27An45W6000 45 rectangle(3.43e+07 0.0 -1.8034e-05 0.0001524 0.0))  
.Via L15A180W4500 L27An135W6000  
+ pad( -1.8034e-5 0.0 ellipse(5.959e+07 0.003048 0.0 0.003972 0.000924))  
+ void( 0.000119634 1.37668e-4 ellipse(5.959e+07 0.003048 0.0 0.006096 0.003048))  
+ void( 0.001661668 0.001679702 ellipse(5.959e+07 0.003048 0.0 0.006096 0.003048) )  
+ pad( 0.001799336 0.00181737 ellipse(5.959e+07 0.003048 0.0 0.003972 0.000924))  
+ drill( 0.0 0.001799336 ellipse(m11 0.003048 0.0 0.003962 0.000914))  
+ trace( L15A180W4500 0 rectangle(5.959e+07 0.0 0.001799336 0.0001143 0.00181737))  
+ trace( L27An135W6000 135 rectangle(3.43e+07 0.0 -1.8034e-05 0.0001524 0.0))  
.ends VIA_DP
```

Example Broadside Coupled Differential Via Pair

```
*There is a VIA here.  
.subckt VIA_DP  
+L15A0W4500 L27An45W6000 L15A180W4500 L27An135W6000  
.layerstack LayerStackAll  
+dielectric( 2.2 0.000119634 )  
+shield( SL23 5.959e+07 1.8034e-05 )  
+dielectric( 2.2 0.001524 )  
+shield( SL17 5.959e+07 1.8034e-05 )  
+dielectric( 2.2 0.000119634 )  
.Via L15A0W4500 L27An45W6000  
+ pad( -1.8034e-5 0.0 ellipse(5.959e+07 0.0 0.0 0.000924 0.000924))
```

Allegro PCB SI User Guide

Modeling in the Interconnect Description Language

```
+ void( 0.000119634 1.37668e-4 ellipse(5.959e+07 0.0 0.0 0.003048 0.003048))
+ void( 0.001661668 0.001679702 ellipse(5.959e+07 0.0 0.0 0.003048 0.003048) )
+ pad( 0.001799336 0.00181737 ellipse(5.959e+07 0.0 0.0 0.000924 0.000924))
+ drill( 0.0 0.001799336 ellipse(ml1 0.0 0.0 0.000914 0.000914))
+ trace( L15A0W4500 0 rectangle(5.959e+07 0.0 0.001799336 0.0001143 0.00181737))
+ trace( L27An45W6000 45 rectangle(3.43e+07 0.0 -1.8034e-05 0.0001524 0.0))
.Via L15A180W4500 L27An135W6000
+ pad( 3.2766e-5 5.08e-5 ellipse(5.959e+07 0.003048 0.0 0.003972 0.000924))
+ void( 0.000119634 1.37668e-4 ellipse(5.959e+07 0.003048 0.0 0.006096 0.003048))
+ void( 0.001661668 0.001679702 ellipse(5.959e+07 0.003048 0.0 0.006096 0.003048) )
+ pad( 1.850136e-3 1.86817e-3 ellipse(5.959e+07 0.003048 0.0 0.003972 0.000924))
+ drill( 5.08e-5 1.850136e-3 ellipse(ml1 0.003048 0.0 0.003962 0.000914))
+ trace( L15A180W4500 0 rectangle(5.959e+07 0.0 1.850136e-3 0.0001143 1.86817e-3))
+ trace( L27An135W6000 135 rectangle(3.43e+07 0.0 3.2766e-5 0.0001524 5.08e-5))
.ends VIA_DP
```

Example One Signal Via Coupled with One Ground Via

*There is a VIA here.

```
.subckt VIA_GND
+L15A0W4500 L27An45W6000
*** +L15A180W4500 L27An135W6000
.layerstack LayerStackAll
+dielectric( 2.2 0.000119634 )
+shield( SL23 5.959e+07 1.8034e-05 )
+dielectric( 2.2 0.001524 )
+shield( SL17 5.959e+07 1.8034e-05 )
+dielectric( 2.2 0.000119634 )
.Via L15A0W4500 L27An45W6000
+ pad( -1.8034e-5 0.0 ellipse(5.959e+07 0.0 0.0 0.000924 0.000924))
+ void( 0.000119634 1.37668e-4 ellipse(5.959e+07 0.0 0.0 0.003048 0.003048))
+ void( 0.001661668 0.001679702 ellipse(5.959e+07 0.0 0.0 0.003048 0.003048) )
+ pad( 0.001799336 0.00181737 ellipse(5.959e+07 0.0 0.0 0.000924 0.000924))
+ drill( 0.0 0.001799336 ellipse(ml1 0.0 0.0 0.000914 0.000914))
+ trace( L15A0W4500 0 rectangle(5.959e+07 0.0 0.001799336 0.0001143 0.00181737))
+ trace( L27An45W6000 45 rectangle(3.43e+07 0.0 -1.8034e-05 0.0001524 0.0))
.Via SL23 SL17
+ pad( 0.000119634 1.37668e-4 ellipse(5.959e+07 0.003048 0.0 0.003972 0.000924))
+ void( 0.000119634 1.37668e-4 ellipse(5.959e+07 0.003048 0.0 0.006096 0.003048))
+ void( 0.001661668 0.001679702 ellipse(5.959e+07 0.003048 0.0 0.006096 0.003048) )
+ pad( 0.001661668 0.001679702 ellipse(5.959e+07 0.003048 0.0 0.003972 0.000924))
```

Allegro PCB SI User Guide

Modeling in the Interconnect Description Language

```
+ drill( 0.0 0.001799336 ellipse(m11 0.003048 0.0 0.003962 0.000914))
***+ trace( L15A180W4500 0 rectangle(5.959e+07 0.0 1.850136e-3 0.0001143 1.86817e-3)
***+ trace( L27An135W6000 135 rectangle(3.43e+07 0.0 3.2766e-5 0.0001524 5.08e-5)
.ends VIA_GND
```

Example One Signal Via Coupled with One Power Via

```
*There is a VIA here.
.subckt VIA_GND
+L15A0W4500 L27An45W6000
*** +L15A180W4500 L27An135W6000 SL23
.layerstack LayerStackAll
+dielectric( 2.2 0.000119634 )
+shield( SL23 5.959e+07 1.8034e-05 )
+dielectric( 2.2 0.001524 )
+shield( SL17 5.959e+07 1.8034e-05 )
+dielectric( 2.2 0.000119634 )
.Via L15A0W4500 L27An45W6000
+ pad( -1.8034e-5 0.0 ellipse(5.959e+07 0.0 0.0 0.000924 0.000924))
+ void( 0.000119634 1.37668e-4 ellipse(5.959e+07 0.0 0.0 0.003048 0.003048)
+ void( 0.001661668 0.001679702 ellipse(5.959e+07 0.0 0.0 0.003048 0.003048) )
+ pad( 0.001799336 0.00181737 ellipse(5.959e+07 0.0 0.0 0.000924 0.000924))
+ drill( 0.0 0.001799336 ellipse(m11 0.0 0.0 0.000914 0.000914))
+ trace( L15A0W4500 0 rectangle(5.959e+07 0.0 0.001799336 0.0001143 0.00181737))
+ trace( L27An45W6000 45 rectangle(3.43e+07 0.0 -1.8034e-05 0.0001524 0.0))
.Via SL23 SL17
+ pad( 0.000119634 1.37668e-4 ellipse(5.959e+07 0.003048 0.0 0.003972 0.000924))
+ void( 0.000119634 1.37668e-4 ellipse(5.959e+07 0.003048 0.0 0.006096 0.003048)
+ void( 0.001661668 0.001679702 ellipse(5.959e+07 0.003048 0.0 0.006096 0.003048) )
+ pad( 0.001661668 0.001679702 ellipse(5.959e+07 0.003048 0.0 0.003972 0.000924))
+ drill( 0.0 0.001799336 ellipse(m11 0.003048 0.0 0.003962 0.000914))
***+ trace( L15A180W4500 0 rectangle(5.959e+07 0.0 1.850136e-3 0.0001143 1.86817e-3)
***+ trace( L27An135W6000 135 rectangle(3.43e+07 0.0 3.2766e-5 0.0001524 5.08e-5)
.ends VIA_GND
```

Example One Signal Via Coupled with Multiple Power Vias

```
*There is a VIA here.
.subckt VIA_GND
+L15A0W4500 L27An45W6000
```

Allegro PCB SI User Guide

Modeling in the Interconnect Description Language

```
*** +L15A180W4500 L27An135W6000
.layerstack LayerStackAll
+dielectric( 2.2 0.000119634 )
+shield( SL23 5.959e+07 1.8034e-05 )
+dielectric( 2.2 0.001524 )
+shield( SL17 5.959e+07 1.8034e-05 )
+dielectric( 2.2 0.000119634 )
.Via L15A0W4500 L27An45W6000
+ pad( -1.8034e-5 0.0 ellipse(5.959e+07 0.0 0.0 0.000924 0.000924))
+ void( 0.000119634 1.37668e-4 ellipse(5.959e+07 0.0 0.0 0.003048 0.003048))
+ void( 0.001661668 0.001679702 ellipse(5.959e+07 0.0 0.0 0.003048 0.003048) )
+ pad( 0.001799336 0.00181737 ellipse(5.959e+07 0.0 0.0 0.000924 0.000924))
+ drill( 0.0 0.001799336 ellipse(m11 0.0 0.0 0.000914 0.000914))
+ trace( L15A0W4500 0 rectangle(5.959e+07 0.0 0.001799336 0.0001143 0.00181737))
+ trace( L27An45W6000 45 rectangle(3.43e+07 0.0 -1.8034e-05 0.0001524 0.0))
.Via SL23 SL17
+ pad( 0.000119634 1.37668e-4 ellipse(5.959e+07 0.003048 0.0 0.003972 0.000924))
+ void( 0.000119634 1.37668e-4 ellipse(5.959e+07 0.003048 0.0 0.006096 0.003048))
+ void( 0.001661668 0.001679702 ellipse(5.959e+07 0.003048 0.0 0.006096 0.003048) )
+ pad( 0.001661668 0.001679702 ellipse(5.959e+07 0.003048 0.0 0.003972 0.000924))
+ drill( 0.0 0.001799336 ellipse(m11 0.003048 0.0 0.003962 0.000914))
***+ trace( L15A180W4500 0 rectangle(5.959e+07 0.0 1.850136e-3 0.0001143 1.86817e-3))
***+ trace( L27An135W6000 135 rectangle(3.43e+07 0.0 3.2766e-5 0.0001524 5.08e-5))
.ends VIA_GND
.Via SL23 SL17
repeat ...
```

IDL Shape Models

SigNoise also generates IDL models for shapes. You might need to edit these models if you want to change the values specified in them by the field solver.

You can create your own shape model using the field solver. If so, you need to understand the structure of these models.

Example Shape Model

The following example shows an IDL model for a shape. The sections following the example describe the statements in a shape model and where you might want to change the SigNoise values in the model.

```
.subckt SHAPE_BLM_X980Y2660L1
+X1260Y2620L1 X1040Y2580L1
.material sml6 dielectric=5.2 losstangent=0.001
.material sml5 conductivity=5.959e+07 losstangent=0
.material sml4 dielectric=5.2 losstangent=0.001
.material sml3 conductivity=5.959e+07 losstangent=0
.material sml2 dielectric=5.2 losstangent=0.001
.material ml7 conductivity=5.959e+07
.material ml6 dielectric=5.2
.material ml5 conductivity=5.959e+07
.material ml4 dielectric=5.2
.material ml3 conductivity=5.959e+07
.material ml2 dielectric=5.2
.material ml1 conductivity=5.959e+07

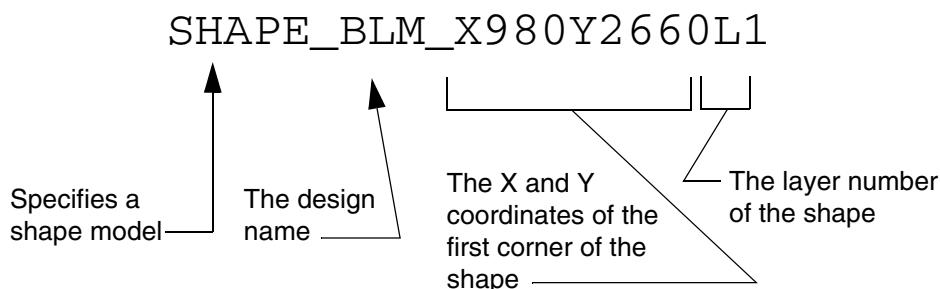
C1 0 1 5.92089e-12
R1 1 X1260Y2620L1 1e-6 L=1e-13
R2 1 X1040Y2580L1 1e-6 L=1e-13
.ends SHAPE_BLM_X980Y2660L1
```

.sbckt Declaration

The shape model begins with a `.subckt` declaration:

```
.subckt SHAPE_BLM_X980Y2660L1
+X1260Y2620L1 X1040Y2580L1
```

The subcircuit name specifies the following information:



The first corner of a shape is the location of the first point you selected when you drew the shape.

The following information follows the subcircuit name:

+X1260Y2620L1 X1040Y2580L1



The X and Y coordinates
and the layer number
where an interconnect line
attaches to the shape

The X and Y coordinates
and the layer number where
another interconnect line
attaches to the shape

.material Declaration

The shape model contains a .material declaration for each layer in the design. These declarations contain dielectric coefficients, conductivity, and loss tangent data from the technology file.

The .material declaration in the model specifies the following information:

.material sml6 dielectric=5.5 losstangent=0.001



A placeholder for material layer number
nine. The prefix "s" for this placeholder
indicates that this .material declaration
specifies two values.

Values for the dielectric
coefficient and loss
tangent.

Closed-Form Solution

At the end of the shape model is a SPICE circuit description that represents the behavior of the shape. SigNoise first uses a closed-form method to generate values for this circuit description.

The following is an example of a closed-form solution:

Allegro PCB SI User Guide

Modeling in the Interconnect Description Language

```
C1 0 1 5.92089e-12
R1 1 X1260Y2620L1 1e-6 L=1e-13
R2 1 X1040Y2580L1 1e-6 L=1e-13
```

In this example a capacitor and two inductors are used to represent the behavior of the shape. As with via models, you can change these values when you edit a shape model.

Allegro PCB SI User Guide
Modeling in the Interconnect Description Language

DML Syntax

Overview

This appendix is a summary of the syntax and structure of Device Model Library (DML) files and their use within package modeling. Most of the examples here are extracted from existing library files.

About DML files

A DML file is a library that contains models and sub-circuits used in circuit simulation.

Cadence Sample Device Model Library

There are a number of sample device model libraries located in this directory:

```
/install_dir/share/pcb/signal
```

Look at the files for additional details of the structure of a Device Model Library file, as well as for examples of the different types of devices.

You can use the Model Browser to list the models in the library, and to edit, add, and delete models. For details on using the Model Browser to work with the models in a library, see [Basic Library Management](#) on page 78.

File Structure

The top 3 levels of a DML file are organized according to the following structure:

```
("filename.dml" ; Name of this DML file.  
(ModelTypeCategory      ; A keyword. See "ModelTypeCategory Keywords" on page 516.  
  ("modelname"          ; A token, The name the user will know the model by.
```

```
<model subparams> ; The data is different for each model type.  
) ; End of one model.  
<more models>  
) ; End of this model type category.  
<more model types>  
) ; End of the data in this library.
```

DML Syntax

A DML file contains lists and sub-lists of tokens enclosed within pairs of parentheses. Comments are also included in the file for documentation purposes.

Comments

A comment consists of all characters on a code line that follow a semi-colon.

Examples:

```
; This is a comment line.  
; End of the package model
```

ModelTypeCategory Keywords

The following table contains keywords that are used to specify the model type.

Table G-1 ModelTypeCategory Keywords

Keyword	Usage
PackageDevice	Defines part models.
PackageModel	Describes package parasitics which may apply to many parts.
IbisIOCell	Describes I/O buffer model information, also known as a <i>buffer</i> .
AnalogOutput	Define a buffer simply by a waveform.
DesignLink	Describes connectivity between modules.

Allegro PCB SI User Guide

DML Syntax

Table G-1 ModelTypeCategory Keywords

Keyword	Usage
Cable	Defines RLGC parasitics matrices and also used by System Configurations to represent cables.

Tokens

A token is a string of characters enclosed in double-quotes(" ").

Example:

```
"Dip14Pin"  
"LogicThresholds"  
"Ramp"
```

Note: If the token contains only alphabetic, numeric, and underscore characters; the double-quotes may be omitted. However, within double-quotes, a token may span multiple lines, with the line endings retained as part of the token. Semi-colons within double-quotes are semi-colons, not the beginning of a comment.

The first token after each left parenthesis is the name of a piece of data.

The first token of a DML file is always the name of the library file.

Example:

```
("filename.dml" ...)
```

Parameters

The tokens following the first one, up to the balancing right parenthesis, represent the value of the data. This collection is called a *parameter*.

Example:

```
(example  
  (type example_type)  
  (name "this is a name with spaces and even a newline,  
        which must be quoted")  
  (timing  
    (setup "1.1")  
    (hold "0.6"))  
)  
)
```

A parameter may have any number of sub-parameters, implementing a data hierarchy. A parameter may instead have a value token. Never does a DML parameter have multiple value tokens, or a value token *and* sub-parameters.

The order of sub-parameters does not matter. It can be in any order as long as it is present.

Sub-parameters of PackageModel

Sub-parameters of PackageModel include:

- PinNameToNumber
- R, G, L, C
- frequency
- format
- data

PinNameToNumber

The PinNameToNumber parameter maps pin-names to corresponding wire-numbers, thus associating each pin with an index in the RLGC matrix.

Note: PinNameToNumber parameter is not necessary if the pin-names are numeric. This information essentially duplicates the WireNumber (if specified) in the PackagedDevice models. If both are present in a given circumstance then the WireNumber from the PackagedDevice overrides the information here. This allows special cases and pin-renaming.

R G L C Matrices

These represent the parasitics matrices at different frequencies.

Note: RLGC data in a PackageModel will be used only if CircuitModels do not exist.

Frequency Value

The frequency point (in Hertz) at which RGLC are extracted.

Format

- BandedSymmetricMatrix

BandedSymmetricMatrix is used to describe the coupling relationship between pins. Two values are associated with this format:

- band (band *B_NUMBER*)

B_NUMBER must always be odd. It indicates that each pin has mutuals with *B_NUMBER* - 1 neighbors.

Example:

(band 3) means that pin 5 has neighbor pins 4 and 6; pin 9 has neighbor pins 8 and 10, etc.

- dimension (dimension *D_NUMBER*)

dimension is defined as the number of pins. It is also the matrix dimension of RLGC parasitics.

■ SymmetricMatrix

SymmetricMatrix format is the same as

BandedSymmetricMatrix with band = 2 * dimension - 1.

■ SparseSymmetricMatrix

SparseSymmetricMatrix format describes a R/G/L/C matrix in SPARSE matrix format.

- dimension (dimension *D_NUMBER*)

dimension is defined same as the one in BandedSymmetricMatrix format.

■ CircuitModels

The CircuitModels format describe a package model using SPICE subcircuit syntax. The RLGC section does not exist when CircuitModels is defined.

Note: CircuitModels is used for large package model description. It can also embed arbitrary SPICE models for packages.

CircuitModels may contain the following two parameters:

- SingleLineCircuits

(SingleLineCircuits (*pin_number* (*SubCircuitName name_of_subckt*))

SingleLineCircuits describes a corresponding single line subcircuit (without coupling to adjacent lines) for each pin in the package.

pin_number maps the subcircuit to corresponding pins.

name_of_subckt gives the name of a SPICE subcircuit in the format:

subckt name_of_subckt input output

where input and output are the subcircuit port nodes; input is to the buffer side for a package model; and output is to the package side.

Note: The port order cannot be altered.

Example:

```
(SingleLineCircuits
  (1 (SubCircuitName longsingle_dc_wire))
  (2 (SubCircuitName longsingle_dc_wire))
  (4-14 (SubCircuitName longsinglewire)))
  ...
)
```

Example:

```
(CircuitModels
  (SingleLineCircuits
    (1-5 (SubCircuitName (typical longsinglewire))))...
  SubCircuits"
    .subckt longsinglewire 1 2
    r13 1 3 0.002 l=3n
    t32 3 0 2 0 z0=70 td=0.1n
    .ends longsinglewire"
  )
)
```

Note: Sub-parameter SingleLineCircuits is REQUIRED in defining CircuitModels.

CoupledLineCircuits

CoupledLineCircuits are needed only for Crosstalk and Comprehensive simulations.

Data

```
(data "data_section")
```

data_section includes the matrix entries of R/L/G/C.

Each entry is separated with space.

■ Data for BandedSymmetricMatrix

The *data_section* of BandedSymmetricMatrix has total number of data as

```
SUM (dimension - i), where i = 0, 1, 2, ... K, K = (1 + (band -1 )/2).
```

Example:

```
(BandedSymmetricMatrix (band 27) (dimension 14)
  ...
)
```

It has $K = (1 + (27 - 1)/2) = 14$, therefore, the total number of data in the data section is equal to $14 + 13 + 12 + \dots + 1 = 105$

Example:

```
(BandedSymmetricMatrix (band 1) (dimension 14)
  ...
```

It has K = 0, and the total number of data is 14.

- Data for SymmetricMatrix

Same as for BandedSymmetricMatrix with band = 2 * dimension - 1.

- data for SparseSymmetricMatrix

```
(data " r_num c_num value...")
```

r_num and *c_num* are the row and column numbers of a non-zero entry.

value is the entry's value.

Note: The order of the entries is unimportant, but it is an error to have row > column for any entry. Give only the upper triangle and the diagonal of the matrix. The lower triangle is assumed equal to the transpose of the upper triangle.

Example:

```
(C
  SparseSymmetricMatrix
    (dimension 14)
    (data " 1 1 643.0f
          2 3 -55.6f
          ...
    )
```

Example of PackageModel

```
(PackageModel
  ("Ex_14Pin"
    (BinNameToNumber ; Maps pin names to numbers
      ("A1" 1)
      ("A2" 2)
      ("A3" 3)
      ("A4" 4)
      ("A5" 5)
      ("A6" 6)
      ("A7" 7)
      ("A8" 8)
```

```
("A9" 9)
("A10" 10)
("A11" 11)
("A12" 12)
("A13" 13)
("A14" 14)
) ; End of PinNameToNumber
(RLGC           ; Set of R, L, G, & C matrices per frequency.
 (0            ; The frequency (in Hertz)
 (R            ; Begin Resistance matrix
 (BandedSymmetricMatrix
 (band 27) (dimension 14) ; That makes this a full matrix.
 (data "0.0006919231 0 0
 ...
") ; End of data section of R
) ; End of BandedSymmetricMatrix
) ; End of R description
(L ; Begin Inductance matrix
(BandedSymmetricMatrix
 (band 27) (dimension 14)
 (data "2.631284E-09 9.014512E-10 5.038304E-10
       3.486299E-10 2.649051E-10 2.16809E-10
       ...
") ; End of data section of L
) ; End of BandedSymmetricMatrix
) ; End of L description
) ; End of RLGC at frequency = 0
...
) ; End of RLGC description
) ; End of Dip14Pin
) ; End of PackageModel
```

Sub-parameters of Cable

Sub-parameters of Cable include PinNameToNumber, RLGC matrix, and CircuitModels.

- **PinNameToNumber**

Same as defined in PackageModel.

- **RLGC**

RLGC indicates the start of the R/L/G/C matrix section.

There are R, L, G, C sub-parameters under this section. They are defined as those in PackageModel.

Example:

```
(Cable
  ("FourWireCable"
    (RLGC ; RLGC here is the sub-parameter of Cable to describe matrices.
      (0
        (R
          ...
        )
        (L
          ...
        )
      ) ; End of frequency 0
    )
  ); End of FourWireCable
) ; End of Cable
```

■ **CircuitModels**

Same as defined in PackageModel.

Computations and Measurements

Overview

This appendix presents an overview of the analytical approach used in signal integrity simulation. The intent is to provide a condensed explanation of how analysis results are derived.

Using SigNoise, you can quickly examine, or scan, one or more signals by performing reflection simulations and crosstalk estimations on entire designs or on large groups of signals. You can also probe individual signals, or small groups of signals, where you want to delve into specific signal behaviors in detail, to generate and view waveforms and text reports. When generating text reports, you can sort the results by specific criteria (for example, undershoot, noise margin, or crosstalk) in order to rapidly identify signals that violate electrical constraints.

Pre-Analysis Requirements

Before running SigNoise, the design needs to be properly prepared with regards to properties, constraints, stack-up definition, and device models. For details, see [“Simulation Setup” on page 141](#). Assuming the board is set up correctly, it should pass the Design Audit procedure in a satisfactory manner. Following is a brief summary of some of the more important items.

Device Models

The libraries containing the required device models must be available in the Library Browser. There are several different types of models that can be present:

- IBIS device models - Assigned to IC's and connectors.
- ESpice models - Assigned to discrete passive components such as resistors and capacitors.

Stack Up Definition

The stack up definition is very important. This provides the geometries SigNoise needs to derive interconnect parasitics. The conductor/dielectric thicknesses and dielectric constants give the Z-axis information, and the X- and Y-axis information are obtained from the routed conductors in the design. This allows SigNoise to have a “2.5 D” database for characterizing interconnect during signal analysis.

Modeling Unrouted Interconnect

You do not have to route the design to run SigNoise. You can define how to model unrouted interconnect in the Parameters form, including percent Manhattan distance, characteristic impedance, and propagation velocity. This allows simulations to be run prior to routing, right from the rats nest. Items such as reflections, termination, and delays can be evaluated at this stage. For routed connections, the actual routed traces are used instead of the criteria described above.

Field Solutions

When you select a net for simulation, a detailed geometry extraction is performed, including the primary net, its neighbor nets (per the user-defined geometry search window), and the mutual coupling between them. This generates a coupled network of multiple conductors.

The interconnect geometry comprising the network is broken up into its unique individual cross sections. For example, if the primary net had 6 mil spacing to a neighbor net at one point, and 12 mil spacing to a neighbor net at another point, these would be broken up into 2 different unique cross sections. Then SigNoise looks in the specified interconnect libraries for the required trace models, based on geometry. If it finds the ones it needs, it plugs them into the circuit. The various trace models are spliced together to build the complete simulation circuit.

If a required trace geometry hasn't been characterized, it is sent to the field solver. The field solver uses 2-dimensional boundary-element techniques to break up the cross section into a fine mesh. Charge distribution over the mesh is then solved to derive a capacitance matrix for the conductors in the cross section. The capacitance matrix is then used to derive the inductance matrix. Resistance and conductance matrices are also generated to take into account conductor loss such as skin effect, and as dielectric losses.

The output of the field solver is frequency-dependent RLGC matrixes (resistance, inductance, conductance, and capacitance) per unit length, which get stored along with the geometry as a trace model in the interconnect library. Therefore, these results can be re-used across multiple designs, so the number of field solutions required to characterize a particular layout

are minimized. When a trace model is called from the interconnect library, its matrices are then multiplied by the length associated with that particular cross section to generate a unique interconnect model for that specific instance.

When a section of an interconnect network is represented in a simulation circuit, the length of the section is used together with the RLGC matrix in the trace model to derive the impedance and propagation delay of the section, along with mutual couplings to other sections.

Vias are automatically modeled through closed-form algorithms, based on their padstack geometry in the layout database. Vias can optionally be modeled in greater detail utilizing the 3D field solver in SigNoise (if available).

Signal Integrity Simulations and Computations

After the interconnect parasitics are derived, the appropriate device models are retrieved and plugged in. A complete simulation circuit is built based on the type of analysis that was requested. The different analysis types are distinguished by what is included in the circuit, and how stimulus is applied, as discussed below.

The *tlsim* Simulator and Simulations

SigNoise uses a proprietary simulator, *tlsim*. The *tlsim* simulator is essentially a clean subset of public domain SPICE with proprietary enhancements and extensions developed by Cadence to optimize it for PCB and MCM applications. For example, *tlsim* uses proprietary macromodels for transmission lines and IOCells.

Reflection Simulations

For a reflection simulation, SigNoise traces out the extended net (xnet), characterizes the trace geometries, obtains the relevant device models, builds a single-line circuit (disregards neighbor nets), and runs a transmission line simulation. The stimulus is applied to the driver pin on the xnet. In the case of multiple drivers on a net, multiple simulations are run, with one active driver stimulated in each simulation. Other drivers on the xnet are inactive during the simulation.

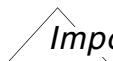
The output of these transient simulations are waveforms and delay and distortion report data. Waveforms are produced for all driver and receiver pins on the xnet. The waveform is a plot of voltage vs. time, so the values for noise margin, overshoot, first switch delay, and final settle delay are taken directly from the waveform data. The receiver waveforms are checked for

monotonicity during the low-to-high and high-to-low transition period. Non-monotonic edges are flagged as violations.

Segment-Based Crosstalk Estimation

Segment-based crosstalk estimation is intended to support an interactive crosstalk debugging/etch editing use model where rapid results and quick identification of worst offenders are required. Rather than performing costly coupled-line time domain simulation in real time, this technique performs the time domain simulation up front to sweep representative crosstalk circuits for the design. This generates tables of crosstalk data. Closed-form algorithms are then employed that intelligently leverage these pre-existing tables of crosstalk data in conjunction with the unique routed geometries of specific nets. This rapidly predicts the magnitude of coupled crosstalk on a segment-by-segment basis in real time. The results are available in report format and are also enforced by on-line DRC's (design rule checks) in Allegro® PCB SI as well as by the Allegro® PCB auto-router.

To obtain segment-based crosstalk results, the crosstalk tables need to be available. These tables are automatically generated based on the IOCell's, stackup, and spacing rules in the design. Based on this information, crosstalk circuits are built and swept through all relevant combinations using full-time domain simulation. Crosstalk magnitudes and saturation lengths are recorded and crosstalk lookup tables generated for use during estimation. When a crosstalk estimate is requested for a given signal, an *Xnet walk* is performed where each coupled segment on the Xnet is identified along with the aggressor. Based on the geometry for each of these unique coupled segments, and the fastest IOCell on the aggressor, the magnitude of the crosstalk imposed on the victim for that specific coupled segment is rapidly predicted. In addition to the individual coupled segment contributions, overall aggressor Xnet contributions and total victim crosstalk data are algorithmically derived and presented in report format.



Important

Segment-based crosstalk estimation does not take into account detailed topological effects like reflections, wave cancellation or the plethora of unique stimulus combinations that can occur in actual designs. To analyze these sorts of effects, full-time crosstalk simulation should be utilized.

Crosstalk Simulations

In Crosstalk simulations, a multi-line circuit is built that includes the victim net, neighboring aggressor nets and mutual coupling. The victim net is held at either the high or low state and the aggressor nets are stimulated. Crosstalk is simulated in the time domain, producing waveforms and report data.

The user can choose to stimulate the aggressor nets in groups, or one aggressor net at a time to isolate the contributions from individual nets, identifying worst offenders. Crosstalk timing properties can also be used to set up neighbor stimuli, as discussed in the following section, [“Timing-Driven Crosstalk Analysis.”](#)

Note: Crosstalk simulation is intended to support a post-route verification use model.

Timing-Driven Crosstalk Analysis

Both segment-based crosstalk estimation and time domain crosstalk simulation in SigNoise are capable of being timing-driven, which greatly increases real-world accuracy and helps eliminate pessimistic crosstalk false alarms. The estimation algorithms use crosstalk timing properties to determine when nets are active and sensitive, and only considers the aggressor nets that have an active time overlap with the victim nets sensitive time as crosstalk contributors. The `XTALK_IGNORE_NETS` property can be used to tell a net or a group of nets to disregard other nets or group of nets as a source of crosstalk. An example would be when you want to disregard crosstalk between bits on a synchronous bus.

The following crosstalk timing properties can be applied to nets in your PCB- or package-editor:

```
XTALK_ACTIVE_TIME  
XTALK_SENSITIVE_TIME  
XTALK_IGNORE_NETS
```

`XTALK_SENSITIVE_TIME` properties can also be assigned to specific receiver pins for an even greater level of accuracy.

For Crosstalk simulations, the crosstalk properties above are used to determine how to stimulate multi-line circuits for crosstalk analysis. For example, assume a victim net being analyzed for crosstalk had 2 aggressor nets, and they had the following properties:

```
victim net - XTALK_SENSITIVE_TIME = 5-10  
neighbor #1 - XTALK_ACTIVE_TIME = 7 - 15  
neighbor #2 - XTALK_ACTIVE_TIME = 20-25
```

Neighbor #2 would not be stimulated in the circuit, since its active time does not overlap with the victim net's sensitive time. In this case, stimulating both aggressor nets together would be overly pessimistic, and not indicative of real-world behavior.

Simultaneous Switching Noise (SSN) Simulations

To understand how a circuit is built for SSN simulations, assume a simple net is selected that has only one driver and one receiver. When the driver pin is found, SigNoise examines the device model for that pin's component and determines which power and ground buses the driving IOCell references. It identifies all other driver pins in the package that share the same power and ground buses. Assume that SigNoise finds four other drivers like this in the package, connected to nets A, B, C, and D respectively.

A circuit is then built to include the following:

- The device model containing the five driver pins, the power and ground pins on the buses involved, and their package parasitics
- Any power and ground pin escape traces and vias
- The original net selected and nets A, B, C, and D

Voltage sources are applied in the circuit corresponding to the nodes at which the pin escape vias contact the power or ground plane. A simulation will then be run in which the outputs at the driving package are switched simultaneously. This produces waveforms at the internal power and ground buses, (at the die itself) for the driving package.

Plane modeling can optionally be included for SSN simulations to account for power and ground planes and decoupling capacitors. In this case, the power and ground planes in the stackup are characterized in an LC mesh. Decoupling capacitors are extracted from the design and attached to the appropriate locations in the mesh. Voltage sources in the circuit are inserted into the circuit based on explicit VOLTAGE_SOURCE_PIN properties attached to connector pins in the design. This detailed analysis not only increases accuracy, but enables the user to view 3D animation of the electromagnetic wave propagation through the design and explore *what if* scenarios with stackup and decoupling.

Comprehensive Simulations

In a Comprehensive simulation, a multi-line circuit is built similar to that used for crosstalk analysis. Power and ground parasitics are taken into account similar to basic SSN analysis. The victim net and the aggressor nets are all switched simultaneously. The user can control whether odd or even mode switching between the victim and aggressors is used.

Delay Computations

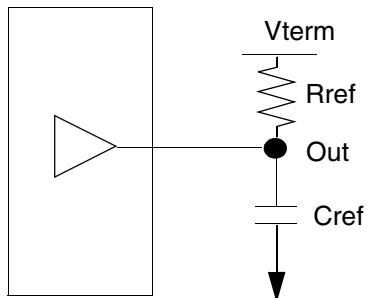
During delay analysis, SigNoise performs the calculations and checks described in this section. These include timing rule checks and pass/fail waveform checks.

Compensated Interconnect Delays

When measuring waveforms at a receiver against time zero in a SigNoise simulation, you include the driving IOCell delay as well as the delay contributed by the interconnect. For the purpose of timing analysis, the IOCell delay is already accounted for in the overall component delay. In order that the IOCell delay is not counted twice when reporting first switch and final settle delays, the assumed IOCell delay must be backed out of SigNoise simulation results.

Since the actual topology to which each pin is attached is not available for up-front timing analysis, a test load (or test fixture) is assumed to be attached to the IOCell in order to derive the component delay. This test load appears as follows:

Figure H-1



To derive the contributed IOCell delay, SigNoise hooks up the IOCell to its corresponding test load circuit and runs simulations to capture the slow, typical, and fast IOCell delay values, measured at a predefined measurement voltage threshold ($V_{measure}$) for rising and falling edges.

These values are stored with the model in the device library. When a SigNoise simulation is run on the design, the appropriate IOCell (or buffer) delay is backed out to properly compensate first switch and final settle delays to represent interconnect contribution only.

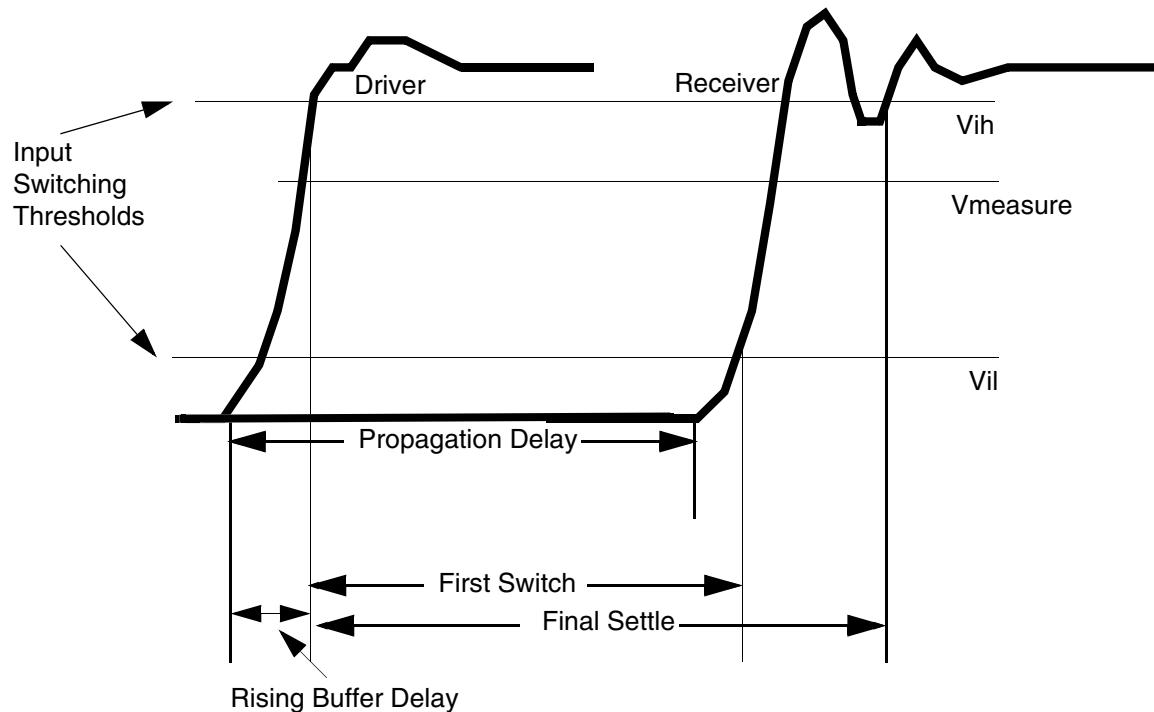
SigNoise determines the delays described in the following table.

Delay	Description
Propagation Delay	Transmission line delay, the time required for wave propagation from driver to receiver.

Delay	Description
First Switch Delay	This is determined by subtracting the associated buffer delay from a simulation measurement. For example, on a rising (falling) edge, the simulation measurement is from time zero to when the receiver first crosses its V_{il} (V_{ih}) switching threshold. The associated rising (falling) buffer delay of the driving IOCell is subtracted from this measurement value to produce the reported first switch delay.
Final Settle Delay	This is determined by subtracting the associated buffer delay from a simulation measurement. For example, on a rising (falling) edge, the simulation measurement is from time zero to when the receiver crosses its V_{ih} (V_{il}) switching threshold the final time and settles into the high (low) logic state. The associated rising (falling) buffer delay of the driving IOCell is subtracted from this measurement value to produce the reported final settle delay.

The following diagram illustrates buffer delay, propagation delay, first switch delay, and final settle delay for a rising signal.

Figure H-2 Buffer Delay, Propagation Delay, First Switch Delay, and Final Settle Delay for a Rising Signal



Note: First switch and final settle delays can be optionally measured to the associated IOCells $V_{measure}$ thresholds, as opposed to the receiver's logic thresholds.

Timing Rule Checks

SigNoise checks the measured delay values against the following constraints:

Constraint	Item
DELAY_RULE	minimum propagation delay maximum propagation delay
MIN_FIRST_SWITCH	minimum first switch delay
MAX_FINAL_SETTLE	maximum final settle delay

Note: You can set these constraints on a net or on a pin-to-pin connection.

Pass/Fail Checks

During delay analysis, the following applies:

- First Incident Rule

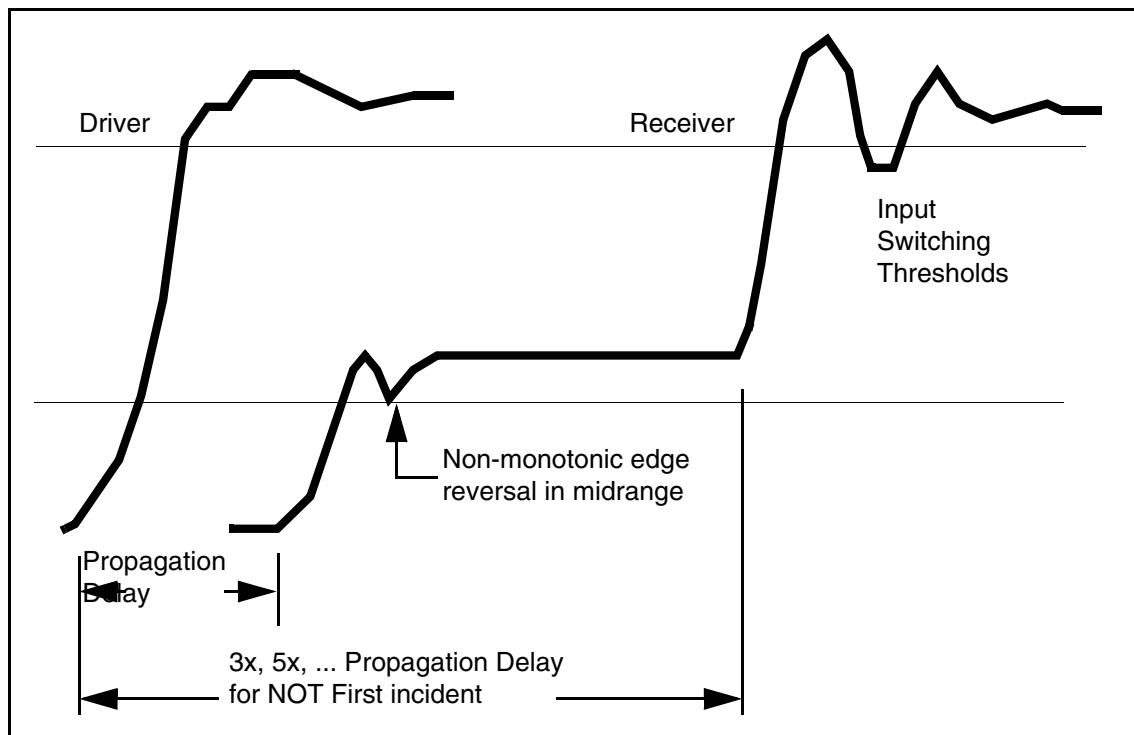
Verifies that the receiver pin transition occurs approximately in sync with the propagation delay plus the actual rise/fall time observed at the driver. Typically, if the transition does not occur on this first transmission of the edge, it occurs much later with reflections matching some odd multiple of the propagation delay (for example, 3x, 5x, and so on, the length delay). The failure is flagged if the time is greater than 1.5 times the expected time of propagation plus the rise or fall time.

- Monotonic Rule

Checks for a smooth transition through the mid-range. This check is especially important for nets which drive edge-triggered devices which could otherwise *double clock* due to a nonmonotonic edge. Ringing that occurs exclusively outside of the mid-range, that is, above the high logic threshold or below the low, is not a violation of this monotonic rule.

The following illustration shows non-monotonic and non-first Incident rules violations.

Figure H-3 Non-monotonic and Non-first Incident Rules Violations



Distortion Computations

SigNoise analysis is controlled by user-defined criteria that establish the thresholds for delay and distortion analysis. SigNoise includes both timing-oriented rules and the following checks for voltage-related signal noise. You can limit each source of signal noise to a maximum voltage level to detect high levels of a particular source of noise. Distortion criteria measured by SigNoise include:

- Overshoot
- Noise Margin
- Crosstalk
- Simultaneous switching noise (SSN)
- Differential mode electromagnetic interference (EMI) in near and far field.

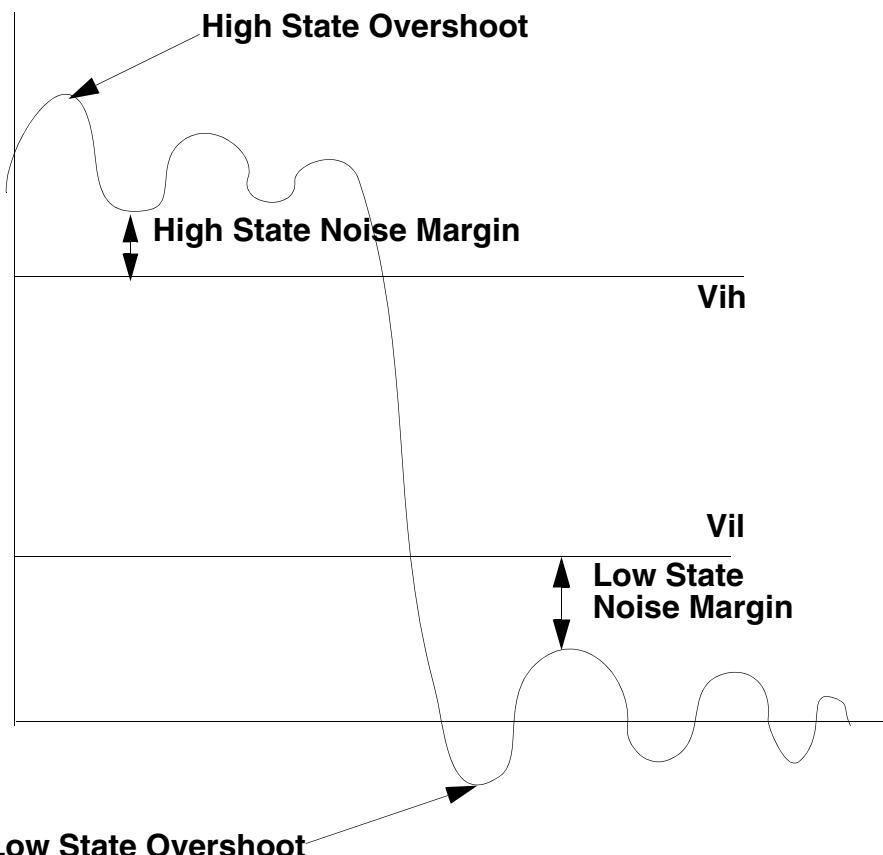
Reflection Measurements

Reflections become prevalent when the signal rise or fall time is relatively small compared to the propagation delay of the signal through the interconnecting etch or when

$$t_r < 2t_d$$

where t_r is the rise/fall time and t_d the one-way propagation delay through the etch. When this is the case, the driver does not see the load. Consequently the proportion between electric and magnetic energy (between voltage and current) is primarily determined by the impedance of the transmission line connected to the driver. As the signal encounters a load or another transmission line with different impedance, the proportion between voltage and current is readjusted. This readjustment is the type of signal distortion called reflection. [“Reflection Measurements” on page 535](#) illustrates waveform reflection measurements.

Figure H-4 Reflection Measurements



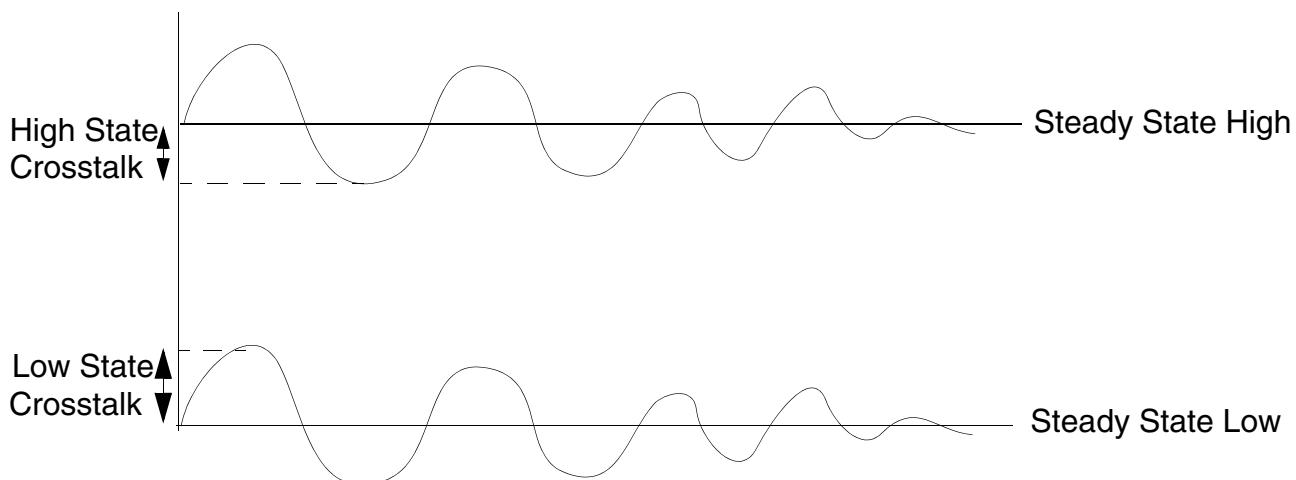
Reflection analysis in SigNoise produces overshoot and noise margin values. For reflection analysis, a transient simulation is performed, and the voltage waveform at each receiver pin

is analyzed producing overshoot and noise margin data as shown in the previous figure. Overshoot values (OV) are reported in reference to ground.

Crosstalk Measurements

When crosstalk simulations are run, multi-line circuits are built. The field solver extracts self and mutual parasitics for routed interconnect, allowing coupled networks to be simulated in the time domain. Waveforms for receiver pins on the victim net are analyzed to produce crosstalk report data. Measurements are taken as follows based on the logic state of the victim. Crosstalk for the victim held in the low (high) state is taken as the magnitude of the delta between the highest (lowest) excursion of the receiver waveform minus the victim's steady state voltage. This is shown in the figure below.

Figure H-5 Crosstalk Measurements



Simultaneous Switching Noise Measurements

To measure the simultaneous switching noise associated with the net, SigNoise will:

- Use the waveforms generated at the die for the driver device's ground node and measure the low state SSN value in the same manner as described for low state crosstalk in "[Crosstalk Measurements](#)" on page 536.
- Use the waveforms generated at the die for the driver device's power node and measure the high state SSN value in the same manner as described for high state crosstalk in "[Crosstalk Measurements](#)" on page 536.

Cadence ESpice Language Reference

Overview

This appendix describes the native language of the simulator *tlsim* and its input file format, ESpice. As a time domain circuit simulator, *tlsim* uses algorithms similar to generic SPICE.

For greater flexibility than available using standard DML syntax, describe the behavior of a special device by embedding ESpice models in the DML model used by your PCB- and package-editor, and SigXplorer.

ESpice (formerly *Kspice*) syntax resembles the generic SPICE syntax. Unlike generic SPICE, ESpice does not yet support transistor level models. ESpice supports model types for devices described by IBIS behavioral data. ESpice also supports novel types of controlled sources and latches. Using ESpice, you can easily, accurately, and efficiently model timing, signal integrity, and electro-magnetic incompatibility (EMI) circuits entirely in the analog domain.

The first sections of this appendix introduce ESpice and *tlsim*, and describe the syntax for basic elements (resistors, capacitors, sources), as well as advanced elements specific to ESpice (IBIS behavioral models). The last section describes how to embed ESpice models in a DML model. ESpice is Cadence proprietary SPICE. IBIS is Avanti's SPICE and is the industry standard.

About the Input Format

ESpice syntax is a relaxed form of the Berkeley Standard SPICE simulator syntax.

Statements

A set of statements composes an ESpice netlist. An ESpice netlist consists of a set of statements, each starting on a separate logical line. A logical line comprises a physical line and continuation lines that follow.

ESpice interprets a + in the first position of a line as a continuation character, and in any other position as the addition operator. Similarly, ESpice interprets an * in the first position of a line as the beginning of a comment statement, and in any other position, as the multiplication operator.

The following example shows a statement on a single physical line:

```
R1 2 3 4.7k
```

The following example uses a continuation line to show the same statement:

```
R1 2 3  
+ 4.7k
```

Statements consist of text elements separated by space and tab characters. ESpice is not case-sensitive. For example, the following are equivalent:

GND, Gnd, and gnd

Using ESpice, you can indent text in statements with one or more spaces to improve readability, unlike using Standard SPICE which requires you begin statements at the first position on the physical line.

You can insert blank lines and comment statements for readability, but may not insert them before continuation lines.

The following example shows a continuation line and a blank line used correctly:

```
R1 2 3  
+ 4.7k  
  
R2 4 5  
+ 6.8k
```

The following example shows a blank line incorrectly used before a continuation line:

```
R1 2 3  
  
+ 4.7k
```

Node Names

Statements may contain node names. ESpice uses alphanumeric node names.

The following example uses alphanumeric node names:

```
R1 first_node second_node 4.7k
```

The special node names 0 and gnd are interchangeable as the absolute reference node.

Note: Leading zeros and trailing letters on node numbers are recognized by ESpice.

Each of the following is a distinct node name:

0, 00, 000, 0a

Using Numbers

ESpice recognizes numbers in integer, floating point, scientific or engineering format as shown in the following table.

Table I-1 ESpice Numeric Formats

Format Type	Examples
Integer	1, 99
Floating Point	3.14, 0.998
Scientific	1.23e-12, 5e12
Engineering	1.23m, 1.0G

ESpice recognizes the engineering scale factors shown in the following table.

Table I-2 Engineering Scale Factors

Scale Factor	Value
T	1e12
G	1e9
Meg	1e6
K	1e3
M	1e-3

Table I-2 Engineering Scale Factors, *continued*

Scale Factor	Value
U	1e-6
N	1e-9
P	1e-12
F	1e-15



Engineering scale factors are not case-sensitive. The standard S.I. notation conventions are not followed. In particular, M is interpreted as mili (i.e. 1e-3, not Meg = 1e6).

Datapoint Sections

Datapoint sections are tables of values used for controlled sources, IBIS models and lossy transmission lines.

Datapoint sections start with a DATAPOINT statement and end with an END statement. ESpice syntax rules do not apply within a datapoint section. Continuation characters (+) are not allowed.

See “[Datapoints Statements](#)” on page 561 for more detail on datapoint sections.

Statement Types

The first significant letter of a statement is generally used to determine the statement type. Table G-3 describes the recognized statement types.

Table I-3 ESpice Statement Types

Statement Type	Description
B	Behavioral I/O buffer model.
C	Capacitor instance.
D	Diode instance (unless start of Datapoints).
Datapoints	Start of datapoint section.

Table I-3 ESpice Statement Types, *continued*

Statement Type	Description
E	Voltage controlled voltage source (expression, piecewise linear, rational function or non-linear inductor model).
F	Current controlled current source (expression, piecewise linear, rational function or non-linear capacitor model).
G	Voltage controlled current source instance (expression, piecewise linear, rational function or non-linear capacitor model).
H	Current controlled voltage source instance (expression, piecewise linear, rational function or non-linear inductor model).
K	Mutual Inductor instance.
L	Inductor instance.
N	Single or multi-conductor lossy transmission line instance.
R	Resistor instance (simple or skin effect).
T	Single (lossless) transmission line instance.
V	Voltage source instance (simple, pulse, sinusoidal, exponential or piecewise linear).
X	Subcircuit instance.
.	Control Statement (see below).
*	Comment (unless followed by 'l').
*	Proprietary simulator directive (do not use).

Note: ESpice does not support the standard SPICE transistor elements.

ESpice supports the control statements described in the following table.

Table I-4 ESpice Control Statements

Control Statement	Description
.END	End of main circuit definition or datapoint section.
.ENDS	End of subcircuit definition.
.INCLUDE	Include contents from specified file to complete circuit.

Table I-4 ESpice Control Statements, *continued*

Control Statement	Description
.MODEL	Specify common model parameters that can be applied to one or more (diode) elements. See “ Diodes ” on page 554 for an example.
.PARAM	Create a parameter. See “ NODE_PARAM Statement ” on page 587 for an example.
.NODE_PARAM	Assign measurements, logic thresholds, reference node, node printing and node name for reports. See “ NODE_PARAM Statement ” on page 587 for an example.
.NODE_PARAM_DIFFERENTIAL	Set node parameters for differential pairs. See “ NODE_PARAM_DIFFERENTIAL Statement ” on page 589 for an example.
.PRINT	Print voltage or current at node (You can use fewer statements by including the print option as part of the .node_param control statement.)
.REF_NODE	Mark named node as the reference node for timing measurements. (You can use fewer statements by including the refnode option as part of the .node_param control statement.)
.SUBCKT	Start subcircuit definition.

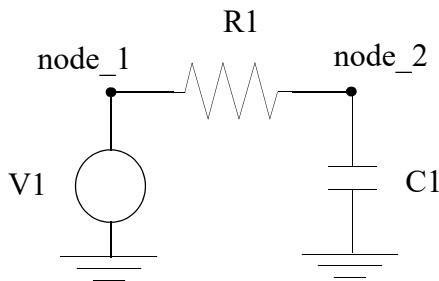
ESpice Syntax Fundamentals

An ESpice circuit contains the following two types of objects representing the electrical circuit:

- Nodes
- Elements

Elements represent the electrical components. Nodes represent how the electrical components are connected as shown in the example.

Figure I-1 Simple Circuit



```
* Simple circuit
* 5 Volt DC voltage source with negative terminal grounded
V1 node_1 gnd 5

* 10uF capacitor with one terminal grounded
C1 gnd node_2 10u

* 50 Ohm resistor between the non-grounded terminals on
* the capacitor and voltage source
R 1 node_1 node_2 50

* Mark the end of the main circuit
.end
```

You can learn the full syntax for the simple resistor, capacitor and voltage source in later chapters. This example shows one syntax for these two terminal devices:

<element identifier (type given by first character)>

<first node name>

<second node name >

<element value>

Where two or more elements have matching node names, assume the nodes are connected.

ESpice partitions circuit functions into modules called *subcircuits* to represent a complex electrical circuit as a flat netlist. The following example shows a subcircuit within a main circuit:

```
* The same simple circuit
* 5 Volt DC voltage source with negative terminal grounded
V1 node_1 gnd 5

* An instance of a subcircuit containing the RC elements
Xsub1 gnd node_1 my_rc_subcircuit

* Define the subcircuit
.subckt my_rc_subcircuit node_a node_b

* 10uF capacitor with one terminal grounded
C1 node_a node_c 10u

* 50 Ohm resistor between the internal terminals on
* the capacitor and the second external port
R 1 node_c nodeb 50

* Mark the end of the subcircuit
.ends

* Mark the end of the main circuit
.end
```

In this case, the subcircuit instance syntax is:

<element identifier (starting with an X)>
<list of nodes connecting subcircuit with main circuit>
<name of subcircuit that this element is an instance of>

In this case, the subcircuit definition syntax is:

.subckt
<name of subcircuit>
< list of nodes connecting subcircuit with main circuit>

Note that nodes in the subcircuit instance are connected to those in the subcircuit definition by order, rather than by name. You can learn the complete syntax for subcircuit instances and definitions in later chapters.

Learning about DC Path to Ground

In a top-level circuit, a DC path must be connected between each node and ground. You can use parameters, expressions, and subcircuits to represent the DC path to ground in ESpice. Include elements in a subcircuit instance in a DC path to ground. The DC path cannot pass through a capacitor or a controlled current source.

Note: *tlsim* checks connectivity to confirm there is a current path flow to ground. Consequently, the circuit cannot contain a loop of voltage sources and/or inductors.

Using Parameters and Expressions

In ESpice, you can assign names to constants. You can make circuit descriptions easier to understand and update by referencing constants by name. You can use the `.param` statement to create parameters as shown in the following example.

```
.param name=value
```

Example

```
* Simple circuit
.param Rval=50
.param Cval=10u
V1 node_1 gnd 5
R1 node_1 node_2 Rval
C1 gnd node_2 Cval
.end
```

Note: The *preferred* syntax for the `.param` statement is `.param key=value`. Although the `=` is optional, use it in all new models.

You can use expressions in ESpice. Expressions must be **enclosed in single quotation marks**.

Example

```
*Simple resistor
.param Rbase=50
.param Rscale=1.25
```

```
R1 node_1 node_2 'Rbase*Rscale'
```

The next section shows how to create parameters within subcircuit calls and definitions.

Using Subcircuits

Using subcircuits, you can name and easily reuse portions of circuit. ESpice, unlike standard SPICE, includes syntax for *parameterized* subcircuits. There are no limits on the size and complexity of subcircuits. Subcircuits can be global (beyond the circuit's `.end` statement) or they can be nested (hierarchical).

You can partition complex circuits and can use subcircuits in various levels of hierarchy.

General Form

```
Xyyyyyyy n1 n2 ... nk Subckt_Name param1=val1 param2=val2 ...
.....
.....
.END /* main or subckt definition ends here */
.SUBCKT Subckt_Name n1 n2 ... nk param1=default_val1 param2=default_val2
subckt description
.ENDS Subckt_Name
```

The key letter X identifies the element as a subcircuit with an arbitrary number of external terminals, for example, n1, n2 ...nk. The X statement and the `.subckt` statement must have the same number of nodes, matched by position.

You can best use parameters by defining them in the calling X statement. ESpice uses the following rules to determine parameter value:

1. If the parameter is defined in the calling X statement (or in a `.param` statement), ESpice assigns that value.

A simple example:

```
.subckt top 1 2
x1 node1 nested param1=60
.subckt nested 1
r1 1 0 param1
.ends nested
.ends top
```

A more complex example:

```
.subckt top 1 2
.param param1=50
```

```
x1 node1 nested param1=60
.subckt nested 1 param1=70
r1 1 0 param1
.ends nested
.ends top
```

The value of the resistor `r1` in both examples is 60 ohms.

2. If the parameter is not defined in the calling X statement (or in a .param statement), ESpice assigns the value from the first definition of the parameter in the calling sequence.

For example:

```
.subckt top 1 2
.param param1=50
x1 node1 nested
.subckt nested 1 param1=70
r 1 0 param1
.ends nested
.ends top
```

Here `param1` is defined in `subckt top` and is not defined in the nested subcircuit `nested`. Therefore, the value of the resistor `r1` is 50 ohms. ESpice assigns the parameter value from the parent subcircuit `top`.

3. If the parameter is not defined in the calling X statement, a .param statement, or in the calling sequence, ESpice assigns the default value defined in the `.subckt` statement.

For example:

```
subckt top 1 2
x1 node1 nested
.subckt nested 1 param1=70
r1 1 0 param1
.ends nested
.ends top
```

The resistor `r1` in this example will be 70 ohms.

Parameterization of Subcircuits

You can pass parameters through subcircuits using `tlsim` as shown in the following example of a macromodel. You choose the parameter name. Then you assign a value. In the example, the parameter name chosen is `ibis_file` and the value assigned to it is `ibis_models.inc`, a DML file translated from an IBIS file. The DML model file contains parameterized model characteristics such as Typ, Min, and Max.

Note: Within a subcircuit, use the absolute path to reference a file.

```
.subckt macromodel 1 2 3 4 5 6 7 ibis_file=ibis_models.inc  
+ibis_model=default_IO  
. . . .  
bmydrives 1 2 3 4 5 6 7 model=ibis_model file=ibis_file  
. . . .  
.ends
```

You can also pass text strings as parameters as shown in the following example.

```
.subckt macromodel out pw gnd pc gc input enable ibis_file=ibis_models.inc  
+ibis_model=default_IO  
. . . .  
bmydrives out pw gnd pc gc input enable model=ibis_model file=ibis_file  
. . . .  
.ends
```

Supported Circuit Elements

The major difference between generic SPICE and ESpice is that ESpice does not yet support transistor-level devices. However, ESpice supports efficient behavioral models, novel controlled sources and latches, and frequency-dependent, multi-conductor transmission lines. Using these additional elements, you can perform timing and signal integrity simulation entirely in the analog domain. ESpice does not support simple current sources.

Following are the major classes of devices described in ESpice and supported in *t/sim*:

- Basic elements
 - Passive elements: resistors, capacitors, inductors and mutual inductors
 - Single lossless transmission lines
 - Voltage sources: DC voltage sources, pulse voltage sources, PWL voltage sources, sinusoidal voltage sources, and exponential voltage sources
 - Diodes
- Controlled source elements
 - Voltage controlled current source (VCCS)
 - Voltage controlled voltage source (VCVS)
 - Current controlled voltage source (CCVS)

- Current controlled current source (CCCS)
- Voltage and current controlled non-linear capacitors and inductors
- Behavioral active devices defined by IBIS data
- Frequency-dependent multi-conductor transmission lines
- Latches: time controlled and voltage controlled latches and hysteresis type latches
- Linear controlled sources driven by rational transfer functions
- Controlled sources defined by expressions

Describing Basic Elements

You can use ESpice to describe Basic SPICE elements.

Passive Elements: Resistor, Capacitor, Inductor, Mutual Inductor

General Forms

```
Rxxxxxxx n1 n2 value <L=value>
Cxxxxxxx n1 n2 value
Lxxxxxxx n1 n2 value
Kxxxxxxx Lxxx1 Lxxx2 value
```

The key letters R, C, L, and K signify resistor, capacitor, inductor and mutual inductor, respectively. The node numbers are *n1* and *n2*. <...> indicates optional parameters

Examples

5 ohm resistor connected between nodes 1 and 2:

```
RS 1 2 5
```

5 pico farad capacitor connected between nodes 1 and 2:

```
C1 1 2 5PF
```

5 nano henry inductor connected between nodes 1 and 2:

```
L1 1 2 5NH
```

0.1 ohm resistor and 3n inductor connected between nodes 1 and 2:

```
R 1 2 0.1 L=3n
```

Mutual Inductor driven by inductors LA and LB. The coupling coefficient should be $0 \leq k \leq 1$.

K LA LB 0.8

Single Lossless Transmission Line

General Form

Txxxxxxx n1 n2 n3 n4 Z0=value TD=value

The key letter T signifies the single lossless transmission line.

n1 and n2 are the nodes at port 1 and n3 and n4 are the nodes at port 2. The reference nodes, n2 and n4, must be the same for both ends. In signal applications, the reference nodes are usually set to 0, the absolute ground.

Z0 is the characteristic impedance in ohms. TD is the propagation delay of the transmission line.

Example

A 50 ohm, 0.5 nano second delay transmission line connected between 2 and 3 referenced to same ground.

T1 2 0 3 0 Z0=50 TD=0.5NS

Voltage Sources

ESpice supports DC, pulse, PWL, sinusoidal, and exponential voltage sources.

DC Voltage Source

General Form

Vxxxxxxx n1 n2 value

The key letter V signifies a DC voltage source. In this case, the DC voltage source is connected between the positive node n1 and the negative node n2.

Example

A -2 volt DC source connected between nodes 5 and 0.

VDC 5 0 -2

Pulse Voltage Source

General Form

```
Vxxxxxxx n1 n2 PULSE(V1 V2 TD TR TF PW PER)
```

The keyword PULSE signifies a pulsed AC voltage source with the following parameters:

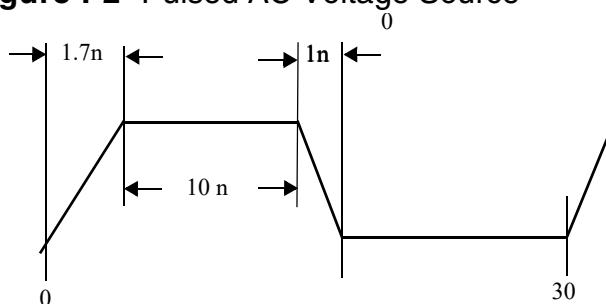
- V1 - Initial Value
- V2 - Final Value
- TD - Delay Time to Pulse Start
- TR - Rise Time
- TF - Fall Time
- PW - Pulse Width
- PER - Period

Example

```
VIN 4 0 PULSE(0 3.5 0 1.7NS 1NS 10NS 30NS)
```

A 3.33 MHz 3.5 volt pulsed source connected between positive terminal 4 and negative terminal 0, with a rise time of 1.7ns and a fall time of 1ns. See the following figure.

Figure I-2 Pulsed AC Voltage Source



PWL Voltage Source

General Form

```
Vxxx n1 n2 PWL (T1 V1 <T2 V2 T3 V3...>)
```

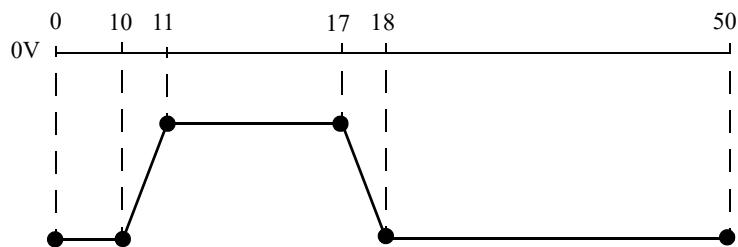
Examples

```
vclk 7 5 PWL (0 -7 10n -7 11n -3 17n -3 18n -7 50n -7)
```

```
Vperiodic 7 5 PWL (0 -7 10n -7 11n -3 17n -3 18n -7 50n -7 ..)
```

The second statement shows that adding two periods (...) at the end makes the PWL statement periodic. See the following figure. Also see [“Describing Controlled Source Elements”](#) on page 554 for a discussion of PWL (Piecewise Linear Tables).

Figure I-3 PWL Voltage Source



Sinusoidal Voltage Source

ESpice supports a transient sinusoidal voltage source.

General Form

```
Vin n1 n2 SIN(VO VA FREQ TD THETA)
```

Example

```
VSIG 10 5 SIN (0 .01V 100KHz 1mS 1E4)
```

- VO (offset) Volts
- VA (amplitude) Volts
- FREQ (frequency) Hertz
- TD (delay) seconds
- THETA (damping factor) 1/seconds

The shape of the waveform is given by:

$$\begin{aligned} \text{0 to } Td, V &= V_0 \\ Td \text{ to } TSTOP, V &= V_0 + V_A \exp(-((t-Td)\theta)) \sin(2\pi FREQ(t+TD)) \end{aligned}$$

Exponential Voltage Source

General Form

```
VIN n1 n2 EXP (V1 V2 TD1 TAU1 TD2 TAU2)
```

Example

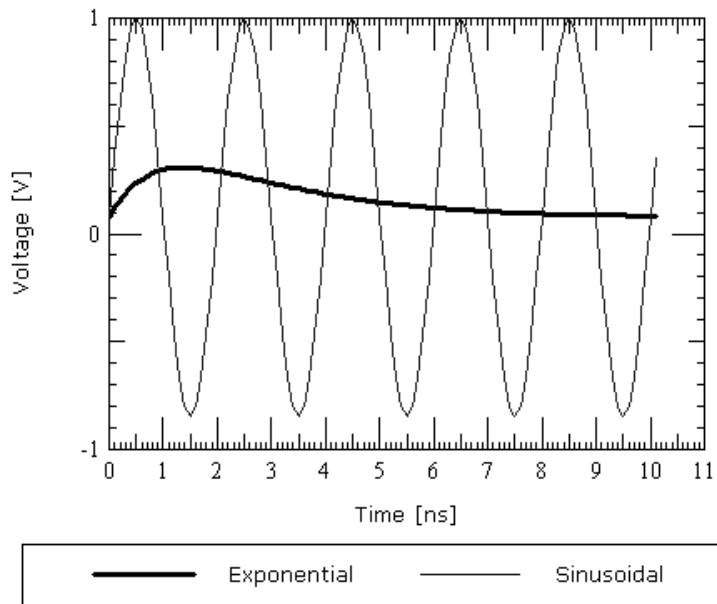
```
VRAMP 10 5 EXP (0V .2V 2uS 20us 40uS 20uS)
```

- V1 (initial value) Volts
- V2 (pulsed value) volts
- TD1 (rise delay time) seconds
- TAU1 (rise time constant) seconds
- TD2 (fall delay time) seconds
- TAU2 (fall time constant) seconds

The shape of the waveform is given by:

$$\begin{aligned} & 0 \text{ to } TD1, V = V1 \\ & TD1 \text{ to } TD2, V = V1 + (V2 - V1)(1 - \exp(-(t - TD1)/TAU1)) \\ & TD2 \text{ to } TSTOP, V = V1 + (V2 - V1)(1 - \exp(-(t - TD1)/TAU1)) \\ & + (V1 - V2)(1 - \exp(-(t - TD2)/TAU2)) \end{aligned}$$

Figure I-4 Sinusoidal and Exponential Voltage Sources



Diodes

General Form

```
Dxxxxxxxx n1 n2 Model_Name  
...  
...  
.MODEL Model_Name Dyyyyyyy IS=value
```

The key letter D signifies the diode element statement. It specifies the nodes marking the start and end points of the diode in the circuit and identifies the associated diode model.

You need both an element statement and a model statement to specify a diode. The model control statement starts with the keyword MODEL and identifies a diode model. You can define IS, the saturation current parameter of the diode.

Examples

A diode connected between positive node 3 and negative node 4 with a saturation current of 1e-14 amperes.

```
D6 3 4 CUTOFF0V7  
...  
...  
.MODEL CUTOFF0V7 D IS=1e-14
```

Describing Controlled Source Elements

Controlled source elements are fundamental to flexible, advanced multi-stage buffer models, and macromodels. A skillful modeler can develop macromodels to accurately represent devices containing hundreds or thousands of transistors. Macromodel computations are more efficient than transistor-level model computations.

ESpice has three major classes of controlled sources. Almost any engineering system can be modeled using classes:

- Controlled sources based on expressions
- Controlled sources based on expression piecewise linear tables (PWL)
- Controlled sources based on rational functions with real coefficients

The following sections describe these controlled sources.

Expression Driven Controlled Sources

You can use expression driven controlled sources with *t/sim* for added flexibility. The expressions can include arbitrary terminal voltages, element currents, and symbolic parameters which can be passed through subcircuits. In addition, expressions can include *derivatives of any order* with respect to time. Expressions can also contain special variables like **TIME**, **PrevTime**, and **TimeStep**.

You can also include complex equations within expressions using *IF ELSEIF ELSE* conditional statements.

General Form

```
Gxxx pos neg i='<expression>'  
Gxxx pos neg v='<expression>'
```

In this case the key letter can be G, E, H or F. The key letter i or v in *i='<expression>'* determines the type of the source.

Be sure to wrap the expression in single quotes.

This example examines a simple harmonic oscillator.

Harmonic Oscillator - the Expression Implementation

Using the expression format, you can directly describe the second-order differential equation:

```
e 2 0 v='v(2) - L * C * ddt_2(v(2)) - R * C * ddt_1 (v(2))'
```

ddt_i (<expr>) is the *i*th derivative of the expression *<expr>*.

Following is a list of syntactical elements allowed in expressions.

Special Function *ddt*

You can calculate derivatives of expressions with *t/sim* using the special function *ddt*. The syntax is:

```
ddt_n (<expr>)
```

The index 'n' stands for the nth derivative.

General Form:

```
ddt_n ((<expr>) (ddtval_n, ddtval_n-1, ... ddtval_1))
```

Examples

```
ddt(v(pos3,neg))  
ddt_2(v(2))  
ddt(i(vp))
```

Special Function prev

prev_n gives the nth previous value. Omitting the index '-n' gives the value of prev_1.

General Form

Prev_n ((<expr>)(prevval_n, prevval_n-1,..prevval_1))

Examples

```
prev(v(out))
```

You can implement latches using special function prev as shown in the following example:

```
vclk clk 0 pulse (0 1 0 1n 1n 4n 10n)  
eout out 0 volt='if (v(clk) > 0.1 && v(clk) < 0.9) (v(in)) else  
+ (prev (v(out))))'
```

The latch in the example outputs a value only during transitions.

Operators

You can use both arithmetic and logical operators in expressions.

Table I-5 Arithmetic Operators

Operator	Function
+	Sum
-	Difference
/	Divide
*	Multiply
^	Exponent
%	Percent

Table I-6 Logical Operators

Operator	Function
&	And
=	Equal
!=	Not equal
	Or
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal

Charge Storage Effects in IBIS Behavior Models

IBIS 3.2 includes a charge storage transit time parameter. You can use a macromodel subcircuit to retrofit the parameter to the existing IBIS models. The following example shows a generic implementation using the transit time parameters TTpwr and TTgnd.

```
* This is general subcircuit for bhvr model which implements charge storage
* effect defined by transit time parameters TTpwr and TTgnd

.subckt chargestoragebuff 1 2 3 4 5 6 7 ibis_file=ibis_models.inc
BUFF=myibisbuff TTpwr=0 TTgnd=0

* introduce current probes at the pwrclamp and gndclamp terminals

vpwrcl 60 6 0
vgndcl 7 70 0

* now attach the bhvr drvr - notice it is attached to nodes 60 and 70
bdrv 1 2 3 4 5 60 70 Model=BUFF File=ibis_file

* now introduce charge storage capacitor effect
gpwrcl 2 6 i='TTpwr * ddt(vpwrcl)'

ggndcl 7 2 i='TTgnd * ddt(vgndcl)'

.ends chargestoragebuff
```

Functions

Expressions can contain any of the following functions:

abs	asinh	cosh	sin	tanh
acos	atan	exp	sinh	
acosh	atanh	ln	sqrt	
asin	cos	log	tan	

Special Variables

The variables **TIME** and **PrevTime** return the current time and the previous time, respectively. The variable **TimeStep**, which returns the difference between **Time** and **PrevTime**, is a short form for the expression (**Time** - **PrevTime**).

The following example produces a sinusoidal source:

```
e pos neg v='sin (2 * PI * Frequency * TIME)'
```

Conditional Expressions

General Form

```
IF (<expr1>) (<expr2>) ELSEIF (<expr3>) (<expr4>) ELSE (<expr5>)
```

If expr1 evaluates to greater or equal to 1, then expr2 is evaluated and so on. Expr1 and expr3 are logic expressions and C-like logic statements (i.e. IF, ELSE, ELSEIF) are allowed.

Examples

The following examples give a flavor of expressions. You can use expressions to solve general circuit problems and to solve engineering and mathematical problems. For example, you can use expressions to solve nonlinear equations.

The examples use IF ELSEIF conditional expressions.

You can instruct the simulator to start the circuit partitioning algorithm by using the current variable $i(v)$.

Simple NMOS Transistor

```
* simple long channel mos device
* from Digital Integrated Circuits by Jan M. RabAey Prentice Hall 1996

.subckt mos_1 dr ga sr bu VT0=0.743 PHI=0.6 LAMDA=0.06 KP=2e-5 W=1u L=1u
GAMMA=0

.eVT tvt 0 v='if (GAMMA <= 0) (VT0) else (VT0 + GAMMA * (sqrt (abs(v(sr,bu)-
PHI)) - sqrt(PHI)))'

.gds dr sr i='if(v(ga,sr) < v(tvt)) (0) elseif(v(dr,sr) > (v(ga,sr) - v(tvt)))
((KP * W/(2 * L)) * ((v(ga,sr) - v(tvt)) ^ 2) * (1 + LAMDA * v(dr,sr))) else ((KP
* W/L) * (((v(ga,sr) - v(tvt)) * v(dr,sr)) - ((v(dr,sr) ^ 2)/2)))'
modifynewton=diodenewton

.ends mos_1
```

Numerical Integrator

```
.subckt integ3 in out starttime=0 endtime=10e-9 out 0
+v='if (time <= starttime) (0)
+elseif ((endtime > 0) && (time > endtime)) (v(out))
+else (prev(v(out)) + ((prev(v(in)) + v(in))/2.0) * timestep))'.
ends integ3
```

Table Driven Controlled Sources

You can use table driven controlled sources with t/sim . These sources use Piecewise Linear Tables (PWL) that capture arbitrary non-linearities.

The type of PWL data is captured in the *DATAPORTS* statement. Using the datapoints statement you can represent data points such as V-I (voltage-current), general v-v (voltage-voltage, current-voltage, and current-current), voltage-coefficient, and voltage-impedance tables.

General Form

VCCS (voltage controlled current source):

```
Gxxxx pos neg PWL[SUM] node1 node2 ...
```

VCVS (voltage controlled voltage source):

```
Exxxx pos neg PWL[SUM] node1 node2 ...
```

CCVS (current controlled voltage source):

```
Hxxxx pos neg PWL[SUM] V1 V2 ...
```

CCCS (current controlled current source):

```
Fxxxx pos neg PWL[SUM] V1 V2 ...
```

Note: When PWLSUM is used instead of PWL in an equation, the multiplication in that equation is replaced by summation.

The key letters G, E, F, and H identify VCCS, VCVS, CCCS and C CVS respectively. These sources are connected between a positive terminal (pos) and a negative terminal (neg). The keyword PWL identifies a *piecewise linear* source.

For the voltage controlled sources (VCCS and VCVS, key letters G and E respectively), the node pairs following the PWL keyword provide the controlling voltages.

For the current controlled sources, (CCVS and CCCS, key letters H and F respectively), the voltage sources V1 and V2 provide the controlling currents.

For the PWL keyword, the value of the voltage or current is given by:

$I/V = f1(v(c1, c2)) * f2(v(c3, c4)) * \dots$ for G and E elements

or

$I/V = f1(I(v1)) * f2(I(v2)) * \dots$ for F and H elements

Non-linear Capacitors

General Form

```
Gxxxx pos neg PWLSUM c1 c2...capacitor=1
```

```
Fxxxx pos neg PWL V1 V2...capacitor=1
```

You can model non-linear capacitors by appending the parameter, *capacitor=1*, to the G or F statement. The PWL table, in this case, describes the capacitance value.

Non-linear Inductors

General Form

```
Exxxxx pos neg PWL c1 c2...inductor=1  
Hxxxxx pos neg PWL V1 V2...inductor=1
```

You can model non-linear inductors by adding the parameter, *inductor=1* to the E or H statement.

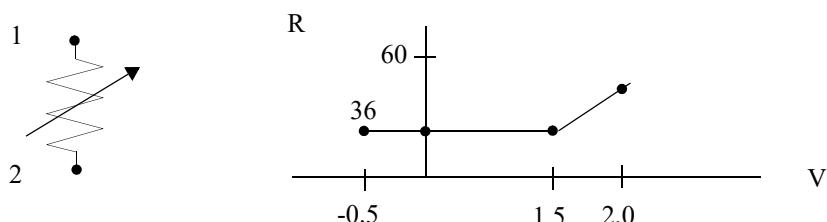
Datapoints Statements

Many circuit elements use tabular data. You use datapoints statements to embed the tabular data in the circuit file as shown in the examples.

One Dimensional Voltage Controlled Impedance

```
GVII 1 2 PWL 1 2  
DATAPOINTS IMPED  
-0.5 36  
0 36  
1.5 36  
2.0 60  
END DATAPOINTS IMPED
```

Figure I-5 One Dimensional Voltage Controlled Impedance



The variable source is controlled by the voltage across its terminals. The datapoints statement specifies the V-I behavior of the source.

```
GVI 1 2 PWL 1 2  
DATAPOINTS VI  
-5 142ma  
0 -132ma  
2.5 -110ma  
5 0  
6 64ma  
8 118ma  
END DATAPOINTS VI
```

Two Dimensional Voltage Controlled Impedance

```
GVII 1 2 PWL 1 2 4 0  
DATAPOINTS IMPED  
-0.5 36  
0 36  
1.5 36  
2.0 60  
END IMPED  
DATAPOINTS COEFF  
1 0  
1.25 0.3  
5 1.0  
END COEFF
```

General Form

```
DATAPOINTS datapoints_type param1=val1 param2=val2 ..  
...  
END [DATAPOINTS datapoints_type]
```

Note: The number of datapoints must equal the number of controlling voltages or currents.

Non-embedded Datapoints

You can keep a very large number of datapoints in a separate file rather than in the main circuit description.

```
e 5 0 pwl 1 0 2 0  
datapoints vv VV file=dtapts.inc  
datapoints coeff Cf file=dtapts.inc
```

The VCVS uses datapoints stored by the names VV and Cf in the file dtapts.inc.

```
datapoints vv VV
....
end vv
datapoints coeff Cf
..
..
end Cf
```

The datapoints_type can be:

Table I-7 Datapoints_Type Descriptions

datapoints_type	Description
IMPED	impedance data
COND	admittance = 1.0/IMPED
VI	Voltage-current data
COEFF	coefficient table
VV	General voltage-voltage, current-voltage, current-current tables
TIMECOEFF	New dynamic time dependent coefficients which can be specified for rising and falling states. Determine the rise and fall states using voltage threshold parameters <i>vlohi</i> and <i>vhilo</i> . Use this instead of the following obsolete latches.

Building Dynamic Latches and Effects Using Timecoeff and Threshold Controlled Sources

tlsim supports new time controlled coefficients which you can use to build any kind of dynamic latch. Implement latches and other dynamic effects using these time controlled coefficients.

Timecoeff

```
DATAPOINTS timecoeff vlohi=0.5 vhilo=3
rising
```

```
0 1  
2n 2  
falling  
0 2  
4n 1  
end timecoeff
```

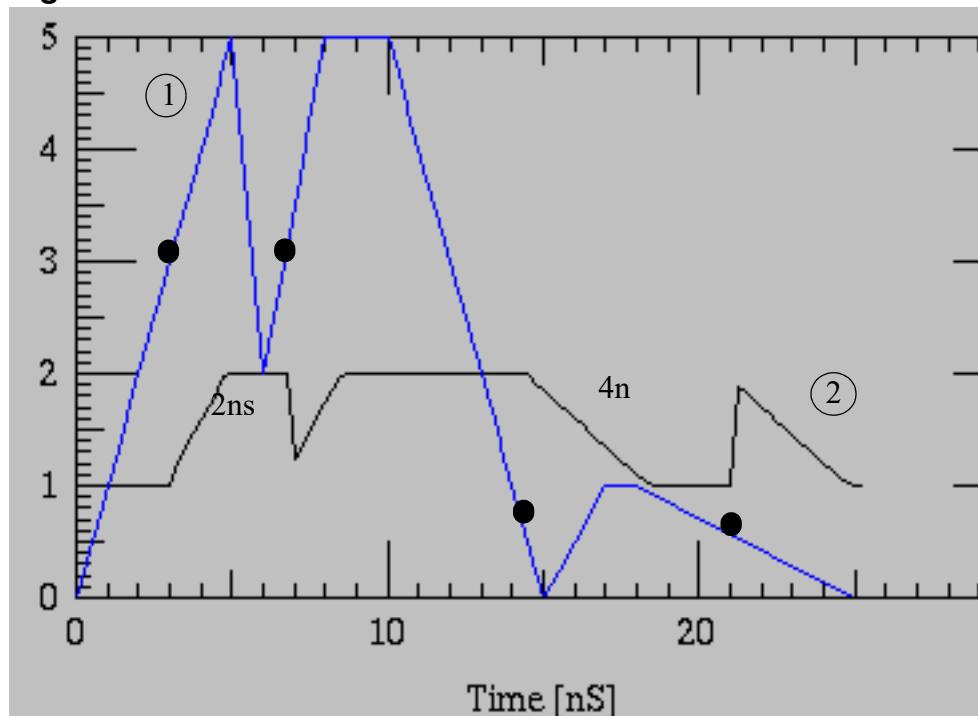
Timecoeff is the key word identifying the time controlled coefficient. You must use at least one of the two supported parameters, *vlohi* or *vhilo*, with this keyword.

Note: The *vlohi* and *vhilo* parameters are not defined in the IBIS standard.

If either *vlohi* or *vhilo* is not present, timecoeff does not transition in the missing direction.

The controlling voltage, crossing *vhilo* with a positive slope, marks the beginning of the rising edge. The controlling voltage, crossing *vlohi* with a negative slope, marks the beginning of the falling edge. Glitches (for example, crossing of *vlohi* followed by yet another crossing of *vlohi*) cause the time to be reset. The following figure shows the timecoeff at node 2 controlled by the input voltage at node 1.

Figure I-6 Timecoeff



Threshold Controlled Source

Use the threshold modifiers *vlohi* and *vhilo* in datapoints statements to model hysteresis effects. The controlled source has two sets of tables: one for rising and another for falling.

See the following examples for use of *vlohi* and *vhilo* in datapoints statements.

Example 1

```
datapoints coeff vlohi=0.5 vhilo=0.5
rising
0 0
0.5 0
1 1
falling
1 1
0.5 0.5
0 0
end coeff
```

Example 2

```
* voltage controlled voltage source
* exhibiting hysteresis
vcntrl cntrl 0 pulse (0 1 0 2n 2n 3n 10n)
e 2 0 pwl cntrl 0
datapoints vv vlohi=0.2 vhilo=0.8
rising
0 -1
0.5 0
1 3
falling
1 2
0.5 -1
0 -1
end vv
```

Other Types of Latches

Threshold controlled sources supersede latches, which are an older style of datapoints. Latches are documented only for backward compatibility. In a voltage (current) controlled latch, the latch turn-on (off) points are controlled by both the instantaneous value and the slope. Additional parameters further control the shape of the latch.

```
DATAPOINTS COEFF_ON(OFF)_LATCH <SWITCHONTIME=val> <SWITCHOFFTIME=val2>
<TRANSITIONINDEPENDENTLATCH=1 or 0> <DURATION=val3>
[Falling OR Rising]
* The first column specifies the controlling value at which the latch comes on.
The second on specifies the width of the on time
col1 col2
END COEFF_ON_LATCH
e_latch 1 2 pwl 3 4
DATAPOINTS COEFF_ON_LATCH SWITCHONTIME=0.1n SWITCHOFFTIME=0.2n
* This latch comes on only during falling transition
falling
* The first column specifies the controlling value at which the latch comes on.
The second on specifies the width of the on time
0.7 1n
end coeff_on_latch
```

This datapoint statement describes a latch which comes on when the controlling voltage drops below 0.7 volt and which has a width of 1 nanosecond. SWITCHONTIME, SWITCHOFFTIME, and DURATION are optional parameters.

If a duration is specified, then the time value in the second column of the data is ignored. If the flag TRANSITIONINDEPENDENTLATCH is set, the latch does not turn off when the slope changes to its opposite value.

Time Controlled Latch

The time controlled latch is a special class of latch which turns on or off during falling and rising transitions. A time controlled latch is sensitive only to the digital slope (that is rising or falling) of the controlling voltage. For example:

```
datapoints coeff_on_latch_timecontrolled
rising
0 5n
end coeff_on_latch_timecontrolled
```

Both values following the rising statement specify time. The relative time is reset to zero once the controlling voltage starts falling.

Hysteresis Latch

The hysteresis latch is a VCVS (a voltage controlled voltage source). See “[General Form](#)” on page 560 for more information.

In this example, the hysteresis latch comes on when the controlling voltage crosses 0.2 volts and is rising. The VCVS reaches its maximum value of 1 volt after 2ns. It starts falling when the controlling voltage goes below 0.8 volt during the falling transition and eventually reaches its minimum voltage of 0 after 0.5ns.

```
Ehyst 2 0 PWL 1 0
DATAPOINTS COEFF_HYSTLATCH RISETIME=2n FALLTIME=0.5n RISEVAL=1.0 FALLVAL=0.0
0.2 0.8
END DATAPOINTS COEFF_HYSTLATCH
```

The simple flag parameter specifies a simple voltage controlled hysteresis. The datapoints have both rising and falling sections. When the control value reaches the maximum in the rising section, the controlling curve becomes a falling curve. When the control value reaches the minimum in the falling section, the controlling curve becomes a rising curve.

Analog Hysteresis Latch

In this example, during the rising section, the transition from 0 to 2V is triggered at 0 volts. During falling, the transition is triggered at -0.5v.

```
ESS 2 0 PWL 3 0
DATAPOINTS VV HYSTERESIS=1
Rising
-2 0
-1 0
0 0
0.499 2
0.5 2
Falling
2 2
0 2
-0.499 0
-2 0.5 0
END DATAPOINTS VV
* controlling voltage
vd 3 0 pwl(0 1 50n 1 60n 1m 80n 1m 90n 1 100n 1 120n -1 140n -1 150n -0.4 160n
-1 180n 0 200n -0.4 230n 1)
```

Rational Function Controlled Source

You can represent filters and other solutions of linear circuit problems in the frequency domain. With *tlsim*, you can model rational functions of the following form:

$$I(s) = H(s) * V(s)$$

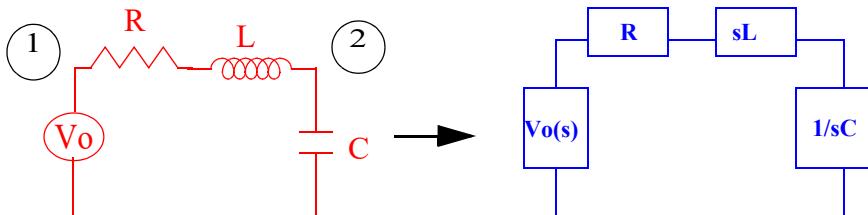
For example, [Figure I-7](#) on page 568 shows the modelling of rational functions using a harmonic oscillator.

Harmonic Oscillator

The following example illustrates a second order harmonic oscillator that contains resistor R, inductor L, and capacitor C connected in series. A second order differential equation (in Laplace algebraic form) describes the transfer function between the input voltage and the output voltage at the capacitor.

Figure I-7 Harmonic Oscillator

$$V_C(s)/V_O(s) = 1/(1 + sRC + s^2LC)$$



The syntax for RATIONAL is very similar to PWLSUM with the keyword RATIONAL replacing PWLSUM. The implementation of the oscillator is:

```
e2 2 0 rational 1 0
datapoints rational
numerator
1
denominator
1 'R*C' 'L*C'
end datapoints rational
```

Note: Datapoints rational can accept parameters like R, L and C.

Each element of the matrix **H(s)** can be represented as a *rational* function.

$$H(s) = \sum a_i s^i / \sum b_i s^i$$

In this equation, a_i and b_i are real coefficients. The time domain simulator uses the rational function in its current form.

General Form

```
Gxxxx pos neg RATIONAL c1 c2 c3 c4 ...
Exxxx pos neg RATIONAL c1 c2 c3 c4 ...
Hxxxx pos neg RATIONAL v1 v2 ...
Fxxxx pos neg RATIONAL v1 v2 ...
```

The controlled value is a sum. For example, look at the k^{th} element of the current vector in our general equation.

$$I_k = H_{k1} V_1 + H_{k2} V_2 + \dots$$

In this equation H_{ki} is the transfer function in the k^{th} row and i^{th} column of the H matrix. $[V1 V2 \dots]$ is the voltage vector. I_k can be directly implemented with a G element. Just like the pwl/pwlsum, each controlling quantity must have a datapoints statement.

```
DATAPOLNTS RATIONAL
  NUMERATOR
    a0 a1 ...
  DENOMINATOR
    b0 b1 b2 ...
END DATAPOLNTS RATIONAL
```

RATIONAL, NUMERATOR and DENOMINATOR are key words.

Note: The harmonic oscillator example follows the general form for Rational and uses the values shown:

$$a0 = 1$$

$$a1 = 0 \text{ (not shown)}$$

$$b0 = 1$$

$$b1 = 'R*C'$$

$$b2 = 'L*C'$$

Describing IBIS Behavioral Model Elements

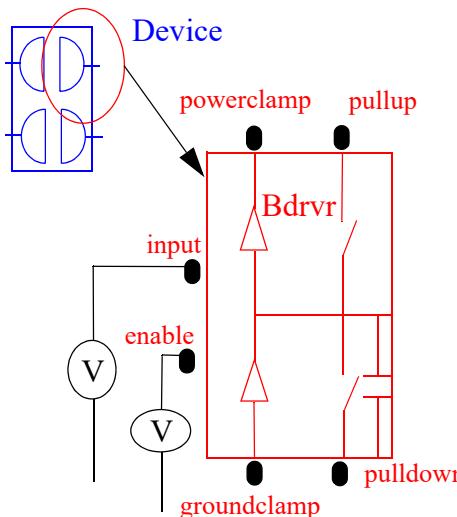
IBIS is an industry standard that captures essential performance data of buffers. Devices containing hundreds of transistors are accurately characterized in a few tables of data. The data tables are:

- Four DC I-V tables, one each for pulldown, pullup, power clamp and ground clamp.
- V-t tables to capture transient behavior.

Note: In DML files and in Allegro PCB SI, V-t and I-V tables are referred to as TV and VI tables respectively.

The basic behavioral model has 7 external terminals. You can use a more advanced 11 terminal model in conjunction with macromodels. This black box behavioral model is internally implemented using the controlled sources. The following figure shows a schematic of the 7 terminal model

Figure I-8 The 7 Terminal IBIS Buffer Model



The pullup and pulldown sources are *switched* and turned on or off, depending on the control voltages at the input and enable terminals. The power clamp and ground clamp are always on. You model switches from pullup to pulldown or from pulldown to pullup, using a switching function.

Without voltage-time data, the shape of the switching function is fixed. Otherwise, the model derives the shape of the function from the voltage-time data. See “[Learning About TV Curves and Switching](#)” on page 581 for more information about using TV curves.

Behavioral Model Syntax

This section describes the operation of 7-terminal behavioral driver (receiver) models. You can follow these model statements with data which describes the switching behavior: DC VI data, rise/fall time data, logic low and high reference voltages, logic low and high state thresholds (the crossing of these thresholds is tracked by the simulator) and TV data. Rather than embed this much data in the main circuit file, store the model data in a separate file. Specify the file name directly in the *Bdrvrv* statement. This differs from standard SPICE practice where you insert the parameters for a model in a separate Model statement.

Note:

- The VI tables are not required. Without VI tables, the device is a simple capacitor or an open circuit.
- Any number of TV tables are allowed. You should generate TV curves for resistive fixtures only and not for reactive elements.
- ESpice allows you a maximum of 1000 datapoints for both VI and TV tables.

General Form

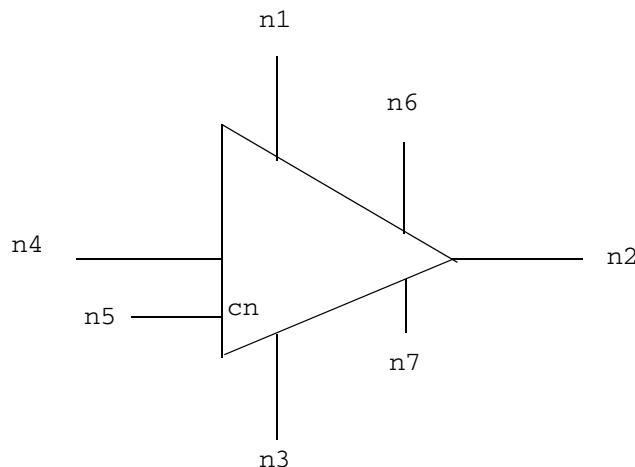
```
BDRVVR (terminals) Model=model_name  
File=file_name
```

Note that when Allegro® PCB SI generates `file_name` from the library data, the file is called `ibis_models.inc`. Run any simulation, open the circuit file, and you can see an example of the file's format and structure.

Both the driver and receiver models start with the keyword *Bdrvrv*. Model receivers with enable off which disables the pullup and pulldown sources.

```
BDRVRxxxxxxxxx n1 n2 n3 n4 n5 n6 n7 [Model=Model_Name File=data_file]
* n1 - Vcc terminal
* n2 - device output
* n3 - Vgnd terminal
* n4 - device input
* n5 - device enable
* n6 - power clamp reference
* n7 - Ground clamp reference
```

Figure I-9 The 7 Terminal BDRVR Element Nodes



Optionally, you can store the behavioral driver data under the *Model_Name* in a data-file using the following syntax:

```
* start the driver data
DATAPOINTS BDRV MODEL_NAME
* the device statement has to be followed by data section
* Required version number MODEL_VERSION=2.0
* Required AC parameters
DATAPOINTS AC_PARAM
VHI_REF=value
VLO_REF=value
RISE_TIME=value
FALL_TIME=value
ENABLE_ON_VOLT=value
ENABLE_OFF_VOLT=value
DVGATEOPEN=value
```

```
DVGATECLOSE=value
* optional threshold switching parameters
VIN_LO_THRESHOLD=value
VIN_HI_THRESHOLD=value
END AC_PARAM
* Logic parameters (optional)
DATAPOINTS LOGIC_PARAM
VHI_MIN=value
VHI_MAX=value
VLO_MIN=value
VLO_MAX=value
END LOGIC_PARAM
```

The following example also demonstrates storing behavioral driver data under the Model_Name in a data-file:

```
* Optional data which specifies the reference voltages for the DC VI curves
PullUpReference=5.0
PullDownReference=0.0
PowerClampReference=5.0
GroundClampReference=0.0
* Required pullup data
* the syntax is
*DATAPOINTS data_points_type ibis-data-keyword
DATAPOINTS VI PULL_UP
-10 -0.142
-2 -0.098
-0.5 - 0.038
0 0
1 0.064
5 0.137
...
END VI PULL_UP
*optional power clamp data
DATAPOINTS VI POWER_CLAMP
...
...
END VI POWER_CLAMP
*optional pull_up switching function
DATAPOINTS COEFF PULL_UP
*optional rising statement
```

Allegro PCB SI User Guide

Cadence ESpice Language Reference

```
RISING
...
...
*optional falling statement
FALLING
...
...
END COEFF PULL_UP
* Required pulldown data since PullUpReference is used earlier
DATAPOINTS VI PULL_DOWN
-1 -0.07
0 0
0.5 0.07
1 0.127
...
END VI PULL_DOWN
*optional ground clamp data
DATAPOINTS VI GROUND_CLAMP
...
...
END VI GROUND_CLAMP
*optional pull_down switching function DATAPOINTS COEFF PULL_DOWN
...
...
END COEFF PULL_DOWN
* optional parasitics data
DATAPOINTS PARASITICS
* component parasitics
C_COMP=value
END PARASITICS
* IBIS 3.2 V-t specifications.
* IBIS 3.2 version allows the insertion of non-linear ramp data .
* These data can be included as
* a set of V-t curves generated in a test fixture. The generation
* of V-t data is done for fixed set of parameters for the
* Device Under Test (DUT) and the test fixture. These parameters are
* Rdut, Ldut and Cdut for the DUT and R_fixture L_fixture, C_fixture
* and V_fixture for the fixture. The parameters R_fixture and V_fixture
* are required while the others are optional.

DATAPOINTS RISINGWAVEFORM R_fixture=60 V_fixture=3.3
```

```
0.0ns 0.608
2.5ns 0.607
...
...
6.5ns 3.3
END RISINGWAVEFORM
DATAPOINTS RISINGWAVEFORM R_fixture=100 V_fixture=3.3
...
...
END RISINGWAVEFORM
DATAPOINTS FALLINWAVEFORM R_fixture=60 V_fixture=3.3
...
...
END FALLINGWAVEFORM
```

Adding Terminators

You can add shunt terminators as described in IBIS 3.2.

```
datapoints bdrvrv mybuffer
..
Rpower=100000
Rgnd=100000
Rac=1000000
Cac=2
...
```

For a complete description refer to the IBIS specification.

Following is an example of a *bdrvrv* model expressed in ESpice. When using a PCB- or package-editor version 14.0 or later, you can use this model in *SigXplorer* as an N-terminal black box model or directly with an independent voltage source.

Example

```
"e:\more_es_bdrvrs\es_bdrivers.dml"
(LibraryVersion 136.2 )
(PackagedDevice
(JB_ES_FAST
(ESpice ".subckt JB_ES_FAST 1 2 3 4 5 6 7
.node_param 1 print
```

Allegro PCB SI User Guide

Cadence ESpice Language Reference

```
.node_param 2 print
.node_param 3 print
.node_param 4 print
.node_param 5 print
.node_param 6 print
.node_param 7 print

*****
* driver
bdrvrvr2 101 102 103 104 105 106 107 Model=JB_IO_Fast thresholdswitch=1 vlohi=2.8
vhi=2.0
.model JB_IO_Fast bdrvrv
DATAPOINTS BDRVVR JB_IO_Fast
MODEL_VERSION=2.0
PullUpReference=5
PullDownReference=0
PowerClampReference=5
GroundClampReference=0

DATAPOINTS AC_PARAM
dVGateOpen=1
dVGateClose=0
ENABLE_ON_VOLT=1
ENABLE_OFF_VOLT=0
RISE_TIME=8e-10
FALL_TIME=8e-10
dVdtr=6.25e+09
dVdtf=6.25e+09
END AC_PARAM

DATAPOINTS LOGIC_PARAM
VHI_MIN=3.1
VLO_MAX=2.1
OUTPUT_VHI_MIN=4.5
OUTPUT_VLO_MAX=0.1
END LOGIC_PARAM

DATAPOINTS VI POWER_CLAMP
0 0
0.1 0
0.4 0.0001
```

Allegro PCB SI User Guide

Cadence ESpice Language Reference

```
0.5 0.0006  
0.6 0.0012  
0.7 0.0024  
0.8 0.006  
0.9 0.013  
1 0.025  
5 0.293  
END VI POWER_CLAMP
```

```
DATAPoints VI ground_clamp  
0 0  
-0.1 0  
-0.4 -0.0001  
-0.5 -0.0005  
-0.6 -0.0012  
-0.7 -0.0024  
-0.8 -0.006  
-0.9 -0.013  
-1 -0.025  
-5 -0.293  
END VI ground_clamp
```

```
DATAPoints VI PULL_UP  
-10 -0.147  
-5 -0.142  
-4.5 -0.138  
-4 -0.133  
-3.5 -0.128  
-3 -0.123  
-2.5 -0.115  
-2 -0.103  
-1.5 -0.088  
-1 -0.069  
-0.5 -0.043  
0 0  
1 0.069  
2 0.103  
3 0.123  
4 0.131  
5 0.142  
END VI PULL_UP
```

Allegro PCB SI User Guide

Cadence ESpice Language Reference

```
DATAPOINTS VI pull_down
-5 -0.225
-4 -0.217
-3 -0.212
-2 -0.193
-1 -0.075
0 0
0.5 0.075
1 0.132
1.5 0.169
2 0.193
2.5 0.208
3 0.212
3.5 0.215
4 0.217
4.5 0.219
5 0.22
10 0.225
END VI pull_down
```

```
DATAPOINTS PARASITICS
C_COMP=3e-12
END PARASITICS
END BDRVR JB_IO_Fast
```

```
Rpu 101 1 0.001
Rpkg 102 2 0.001
Rpd 103 3 0.001
Rin 104 4 0.001
Ren 105 5 0.001
Rpc 106 6 0.001
Rgc 107 7 0.001
```

```
*****
.ends JB_ES_FAST
" ) ) ) )
```

Charge Storage Parameters

Include the IBIS 3.2 charge storage parameters as shown:

```
TTpower=1n  
TTgnd=1
```

See the IBIS 3.2 standard for a description of these parameters.

Using Advanced 11 Terminal Behavior Models

You can specify 11 terminals for behavioral *bdrv* models. The voltages at the extra 4 terminals directly multiply the pullup, pulldown, power clamp and ground clamp V-I tables, respectively, so you can dynamically control and scale these current sources.

For example, you can boost pullup by 20% and pulldown by 30% as shown:

```
Bdrv 1 2 3 4 5 6 7 8 9 10 11 Model=buff File=ibis_file  
v_pullupx 8 0 1.2  
v_pulldownx 9 0 1.3  
v_powerclamp 10 0 1  
v_groundclamp 11 0 1
```

Modifying the Switching Function in Behavior Models

One important aspect of a behavior model is its switching behavior, for example, how it switches from pullup to pulldown and from pulldown to pullup. The basic model, with default switching and no TV curves, is implemented as a VCCS with the input voltage value controlling the switching function. During switching, the coefficient monotonically increases from 0 to 1 for the duration of rise time for the pullup. Similarly, during switching, the coefficient monotonically decreases from 1 to 0 for the duration of the fall time for the pulldown. In this case, the input voltage zero means the pullup is completely shut off, and the input voltage 1 means the pulldown is completely shut off.

With TV curves, the switching function is based on *timecoeff* rather than input voltage value. A positive slope at voltage 0 and a negative slope at input voltage 1 denote the start of rising. Unfortunately, numerical instability and false switching arise if the input node voltage is not a simple source or is floating, as during simultaneous switching. You can make switching use a truly *threshold driven* *timecoeff* by basing the *bdrv* switching on the input voltage thresholds *vlohi* and *vhilo*, which are defined in the IBIS data.

Note: The parameters *vlohi* and *vhilo* are not defined in the IBIS standard.

In this case, the input voltage has to swing from vgnd to Vcc or the full logic swing, rather than 0 to 1, as in the default case. Specify the thresholdswitch parameter in *bdrvrv* statements to enable threshold switching.

```
bdrvrv 1 2 3 4 5 6 7 Model=buff file=ibis_file  
thresholdswitch=1
```

Change the threshold parameters *vlohi* and *vhilo* on an instance by instance basis using the thresholdswitch parameter.

```
bdrvrv 1 2 3 4 5 6 7 Model=buff file=ibis_file  
thresholdswitch=1 vlohi=0.5 vhilo=0.5
```

You can scale the internal C_comp using the switch C_compX

```
bdrvrv 1 2 3 4 5 6 7 Model=buff file=ibis_file  
C_compX=0
```

Prescaling VI and TV Curves in Behavior Models

You learned how to *dynamically* scale VI curves. Dynamic scaling is important in applications like simultaneous switching where VI degradation takes place. In applications like buffer design you need to *prescale* an existing buffer. You can scale the VI and TV curves in *Bdrvrv* statements on an instance basis.

```
bdrvrv 1 2 3 4 5 6 7 Model=buff file=ibis_file VIScale_pullup=1.2  
TVScale_rise1=0.8
```

The prefix *VIScale* requires the underscore after it. The VI curve scaling keywords are:

- *VIScale_pullup*
- *VIScale_pulldown*
- *VIScale_powerclamp*
- *VIScale_groundclamp*

Note that the scaling is for the current.

TVScale scales or *stretches the time in a TV table*. Unlike scaling VI curves, you can individually scale any number of TV curves. The curves are identified by a *name*.

In this example the modifier stretches the time in the TV table by a factor of 0.8 curve identified as *rise1*.

```
TVScale_rise1=0.8
```

The `rise1` label should be in the `bdrvrv` model data as shown:

```
DATAPOINTS tv rise1 v_fixture=1.5 r(fixture)=50
..
END tv
```

Prescaling VI and TV at the Model Level

Modifying the `Bdrvrv` statement affects the VI and TV curves of the model buffer only for that particular instance. The original model is unaltered. If you need scaling on a more global model level, you can include scaling sections in the `Datapoints Bdrvrv buff` model statement.

```
DATAPOINTS BDRVVR buff
..
datapoints vi_scalingfactors
pullup=0.5
pulldown=0.5
end vi_scalingfactors

datapoints tv_scalingfactors
rise_1=0.5
rise_2=0.5
fall_1=0.2
fall_2=0.2
end tv_scalingfactors
..
end bdrvrv buff
```

Learning About TV Curves and Switching

TV curves are not required by `tlsim`. If you have no TV curves, `tlsim` uses ramp data to determine rise and fall time.

Note: Although the IBIS specification allows up to 100 tables, Allegro® PCB SI uses all available TV tables.

While you may use TV curves with full-swing CMOS drivers, you are less likely to use them with less “linear” driver circuits, such as low voltage differential drivers.

For I/O (pullup and pulldown), if you use TV tables, you need to use 4 to obtain correct switching. For a buffer that has only pullup, or only pulldown, if you use TV tables, you need only use 2. [Table I-8](#) on page 582 shows the number of TV tables you need in relation to the presence of pullup, pulldown or both. You can accurately determine the number of TV tables you need without regard to whether the buffers have a timing offset between pullup and pulldown.

These are the most commonly used TV tables:

- Rising, Rload to signal_low, for example, 0V
- Rising, Rload to signal high, for example, Vcc
- Falling, Rload to signal_low, for example, 0V
- Falling, Rload to signal high, for example Vcc

A TV table load should be near the tline impedance you are trying to drive. This is usually in the 50-75 ohm range.

Table I-8 Number of TV Tables Required for Correct Switching

Pullup	X	X
Pulldown		X X
Required number of TV tables for correct switching	2	2 4

The following *tlsim* command line options control switching for TV curves:

Table I-9 Command Line Options Controlling Switching for TV Curves

Switch	Default	Description
BhvrTvOn(Off)	on	Changing to 'off' turns the TV curves off
BhvrTvincludec_compOn(Off)	on	Changing to 'off' ensures that <i>tlsim</i> does not consider drain current through c_comp during coeff computation

Table I-9 Command Line Options Controlling Switching for TV Curves, continued

Switch	Default	Description
BhvrTvPreferDomtimedepcoeffOn(Off)	on	Changing to ‘off’ makes the switching coeff dependent on both time and output voltage which is usually not desirable
BhvrTvUseT_dVXTableOn(Off)	on	Changing to ‘off’ ensures that <i>tlsim</i> does not consider voltage during coeff computation
BhvrTvUseT_IXTableOn(Off)	off	Changing to ‘on’ causes the switching coeff to depend on current, which can lead to instability
BhvrTvUseModifiedTvCoeffOn(Off)	off	Experimental switching coeff modifier - do not change

In devices with fewer than 4 TV curves and containing both pullup and pulldown, the switching coefficient provides continuous and smooth transitions between pullup and pulldown.

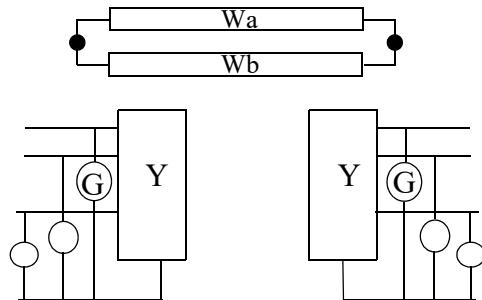
Note: Add the instance specific parameter *pullUpOnRise**pulldownOnFall* to change to switching without continuous transitions between pullup and pulldown.

Adding more than 4 TV tables does not increase switching accuracy and adds simulation overhead. The switching coefficient in these models depends not on load, but on relative rise and fall times. In other words, changes in the gate to source voltage determine the transient behavior.

Using Multi-Conductor Transmission Line Models

The *tlsim* simulator supports advanced, state of the art, multi-conductor, coupled transmission line models. The following schematic shows the circuit model for a multi-conductor line. ESpice supports coupling of multiple conductors.

Figure I-10 Multi-conductor Line Circuit Model



The internal implementation of the model consists of a frequency-dependent admittance matrix Y and delayed controlled current sources (G) at the input and output terminals.

The circuit parameters are derived from frequency-dependent RLGC parameters. These parameters are specified in separate input files. Alternatively, you can embed the RLGC parameters in the ESpice file.

The syntax for a multi-conductor line follows:

```
Nxxxxxxxxx (n1 n2 ....nri) (no1 no2 ..... nro)
L=length-in-meters [file=RLGC_input_file_name rlgc_name=data_name_tag
skin_cutoff_freq=fskin dielecLoss_cutoff_freq=fdielec]
```

The optional parentheses separate the input and output terminals. The words *nri* and *nro* denote the reference nodes. Normally these are node 0. The field *L*= specifies the length of the coupled transmission line system in meters. If *L*=0, then the RLGC matrix is treated as a lumped model.

Use the parameters *skin_cutoff_freq* and *dielectric_cutoff_freq* when only one RLGC matrix (freq=0) is available but you need to model skin effect, or dielectric loss, or both.

Examples of Multi-Conductor Elements

```
NTL_2CONDUCTOR (1 2 0) (3 4 0) L=0.0005 [rlgc_name=Data-name-tag
+file=RLGC_2COND_FILE ]
```

RLGC_2COND_FILE Entries

```
*Specify frequency in Hertz
* Note: If the frequency statement is left out, it will be assumed that the
* rlgc data is *frequency independent. In this case there SHOULD be only ONE
* set of rlgc data
FREQUENCY=100HZ
* Now specify the matrix entries row by row
```

Allegro PCB SI User Guide

Cadence ESpice Language Reference

```
* Specify values per unit length
* for Rmatrix the value is in ohm/meter
* L = henries/meter
* c = farads/meter
* G in siemens (1/ohm)/meter

RMATRIX
1.2 0
0 1.2

LMATRIX
...
...
CMATRIX
...
...
GMATRIX
...
...
FREQUENCY=1000
....
....
```

Embedded NTL Statements

Enter the RLGC data by embedding it in the ESpice file. The embedded data must come after the n-conductor NTL statement.

Note: You omit the file name from the NTL statement.

```
NTL_2CONDUCTOR (1 2 0) (3 4 0) L=0.0005
DATAPOINTS RLGC
FREQUENCY=1000
RMATRIX
1.2 0
0 1.2
LMATRIX
...
...
CMATRIX
...
...
GMATRIX
...
...
FREQUENCY=100000
....
....
END RLGC
```

The mathematical form for the *skin_cutoff_freq* keyword is the same as that in the skin effect resistor as shown:

```
R = Rdc, where f <= fc
R = Rdc sqrt(f/fc), where f > fc
```

For the conductance, use the following form:

```
G = Gdc, where f <= fc
G = Gdc * f/fc, where f > fc
```

Lumped RLGC with L=0

The lumped RLGC matrix is a special case of a multi-conductor line with the length of the multi-conductor set to 0. The syntax is the same as that of a multi-conductor line with L=0.0

```
Nxxxxxxxxx (n1 n2 ....nri) (nol no2 ..... nro)  
L=0.0 file=LC_input_file_name
```

Using Control Statements

tlsim also supports SPICE like control statements. The control statements start with a period (.) character. Following are some of the supported statements:

NODE_PARAM Statement

Use the .NODE_PARAM statement to (re)assign name and logic values to circuit nodes. The syntax is:

```
.node_param nodeid param1=value1 param2=value2
```

Nodeid is the original node id. The allowed parameter keywords are NAME, VHI_MIN, VHI_MAX, VLO_MIN, VLO_MAX, CYCLE, VMEAS.

Note: VHI_MIN and VLO_MAX are important, reserved keywords. Using these values, *tlsim* can define cycle measurements. For nodes which are the output of an IBIS buffer, the VHI_MIN and VLO_MAX are defined in the buffer data. The values defined in the buffer data override the values defined in the node_param statement.

The value of NAME is a string. The other values are floating point values.

Two boolean parameters are supported: PRINT and REFNODE. PRINT specifies the node values must be printed. REFNODE identifies the reference node from which simple “flight” propagation delay measurements are made. The boolean parameters need not be followed by the equal (=) sign.

Cycle Measurements

You can insert special cycle measurement controls in a node_param statement.

The following statement prints cycle measurements for the 3rd cycle:

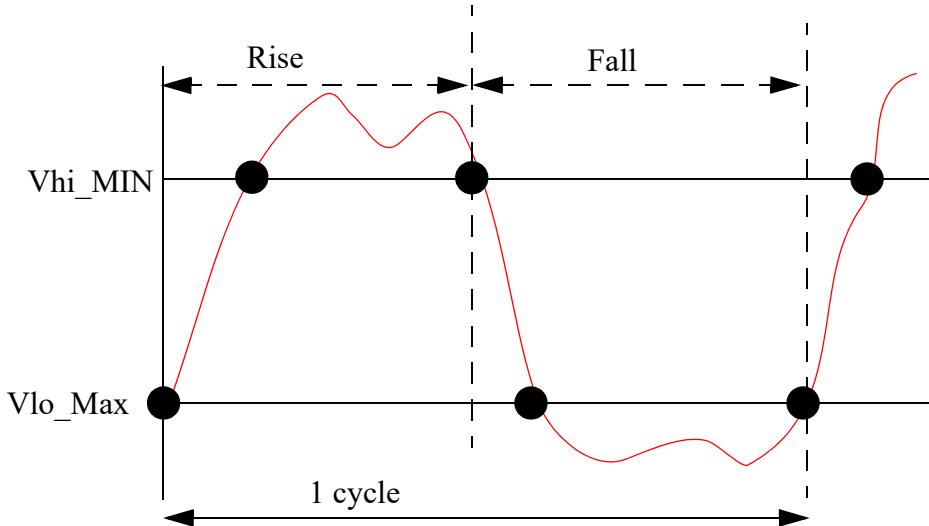
```
.node_param 2 cyclecount=3 print
```

The following statement prints cycle measurements for the 2nd and the 5th cycles:

```
.node_param 2 cycles=(2 5) print
```

With the previous statement, the simulator tries to simulate until node 2 completes its last cycle. It also saves cycle parameter information in the cycle file, `cycle.msm`. The cycle measurement is shown in [Figure I-11 on page 588](#):

Figure I-11 Cycle Measurement



Time Window Measurements

You can measure minimum and maximum voltage in named time windows. The following example outputs the maximum and minimum voltages in window a (9.04 n to 10.72 n) and window b (16.32 n to 17.52 n):

```
.node_param 2 timewindowpairs=(a 9.04n 10.72n b 16.32n 17.52n)
```

Arbitrary Voltage Crossings

You can measure crossing points of arbitrary voltage. These crossings must not have reserved names like `vlo_hi`.

The following statement measures the time crossing points of 1.1 volt. The measurements appear under the label 'v1.1'.

```
.node_param 2 v1.1=1.1
```

Specifying Derived Names in Node Param Statements

You can use node param statements inside a subcircuit. Using the standard syntax, you can specify only one name. When the subcircuit is used multiple times, printed output only includes information for the last node listed.

In the following example, the printed output contains value for only node 2 in the main circuit.

```
x1 1 sckt  
x2 2 sckt  
.subckt sckt 1  
.node_param 1 name=sckt_name print  
.ends sckt
```

Print both nodes by using derived names in node_param statements. In the next example, two names are created: mynode1_derived and mynode2_derived. The statement name() acts like a function; name(1) substitutes the name of node 1. You can append or prepend an additional string, for example, “derived.”

The name function requires the format: name = (name(yournode))

```
x1 1 sckt  
x2 2 sckt  
.node_param 1 name=mynode1  
.node_param 2 name=mynode2  
.subckt sckt 4  
.ends sckt
```

Inheriting Node Parameters

You can also inherit parameters of another node, rather than repeat the same parameters. In this example the inherit function pulls in all the cycle values, and voltage values. Note that nodes 1 and 5 should have been defined first.

```
.node_param 2 inherit=(1)  
.node_param_diff 2 3 inherit=(1 5)
```

NODE_PARAM_DIFFERENTIAL Statement

This statement is similar to .NODE_PARAM. It measures the differential voltage between a primary node and a secondary node.

```
.node_param_differential node1 node2 param1=value1 param2=value2
```

PRINT Statement

```
.PRINT statement
```

Use the standard SPICE print statement to print currents through zero voltage sources, as shown in the following example:

```
V_probe 1 2 0
.print I(V_probe)
```

Creating ESpice Models for Use with Allegro® SI

You need DML format models in your library in order to simulate using Allegro® SI and SigXplorer. You obtain simulation models in DML format from third parties or create your own. If the models are in SPICE format, you can represent them in ESpice. You can also translate IBIS format models to DML using *ibis2signoise*.

You can learn to create models using standard DML features described in “[DML Syntax](#)” on page 516. To create a simulation model that has some atypical behavior, describe that behavior using ESpice.

You can include ESpice simulation models in DML libraries as described next.

Creating an ESpice Packaged Part

Create a packaged part with behavior wholly described in ESpice using the syntax in the following examples:

Example 1 - Simple RC Filter

```
(PackagedDevice
  (MyEspiceRcFilter
    (ESpice
      ". MyEspiceRcFilter 2 1
        R1 1 2 22
        C1 0 2 20p
      .ends MyEspiceRcFilter"
    )
    (PinConnections
      ...
    )
  )
```

)

Example 2 - Simple Series Resistor

```
(PackagedDevice
  (MyESpiceResistor
    (ESpice
      ".subckt MyESpiceResistor 2 1
        R1 2 1 22
      .ends MyESpiceResistor
      "
      )
      (PinConnections
        (1 2)
        (2 1)
      )
    )
  )
)
```

The PinConnections section, which includes pin pairs, designates this part as a series element that can electrically connect two PCB nets to create an extended net (Xnet). Entries are directional from the first pin to the second, so a bi-directional connection has two entries for each pin pair. To eliminate the creation of Xnets from the nets connected to pins defined by the model, you must attach the NO_XNET_CONNECTION to the component you are attaching the ESpice model to.

Note: The external node names in the ESpice subcircuit are used as the pin names in the library component and must match those in the PCB symbol. Also, the entire ESpice description must be within double quotation marks. Write comments inside the double quotations by starting the comment with an asterisk (*).

Use any name for the ESpice subcircuit since Allegro® SI uses the packaged device name to locate the correct library entry. The subcircuit name is set to be the same as the packaged device name.

Example 3 - A Thevenin Terminator Package

```
(PackagedDevice
  (MyEspiceTerminator
    (ESpice
      ".subckt MyEspiceTerminator G1 S1 S2 G2
        R1 S1 G1 220
```

```
R1a S1 G2 330
R2 S2 G1 220
R2a S2 G2 330
.ends MyEspiceTerminator"
)
(PinConnections
    (S1 G1)
    (S1 G2)
    (S2 G1)
    (S2 G2)
    ...
    ...
)
)
)
```

Example 4 - Alternative Version of a Thevenin Terminator Package

```
(PackagedDevice
    (MyEspiceTerminator
        (ESpice
            ".subckt MyEspiceTerminator G1 S1 S2 G2
X1 S1 G1 G2 MyResistors
X2 S2 G1 G2 MyResistors
.ends MyEspiceTerminator
.subckt MyResistors 1 C1 C2
R1 1 C1 220
R2 1 C2 330
.ends MyResistors"
)
(PinConnections
    (S1 G1)
    (S2 G2)
    (S2 G1)
    (S2 G2)
    ...
    ...
)
)
)
```

Note: A DML terminator model provides the same functionality.

Using SigXplorer Sources and Functions with ESpice Devices

Guidelines for Using ESpice Devices

- Ensure that the first SPICE element in the netlist is an R, L, or C. Your netlist will not run without one of these valid initial elements.
- Ensure your syntax is correct and all SPICE elements begin in column 1 since there is no post-checking format on your circuit.
- Ensure that the beginning column for *.subckt* and *.ends* is column 1.

Voltage Source

Use a simple voltage source to drive the system, when studying the behavior of an interconnect. Use an ESpice device and custom stimulus to do this.

Here's how:

1. Build the desired source.
2. Add a disabled driver to the circuit.

You add a disabled driver because the simulation is aborted with no eligible drivers in the topology, and because SigXplorer does not recognize an active Vsource taking the place of a driver. To circumvent this problem, perform these tasks:

1. Add an IO (tri-statable) driver.
2. Using Custom Stimulus, hold the ENABLE line to a zero level until just before the end of the simulation.

Examples of two Vsources you can use implemented as PULSE and PWL follow:

```
("ideal_source"
  ("ESpice"
".subckt ideal_source 1 2
Rdud 1 3 0.00001
V1 3 2 PULSE (0 2.5 0n 1n 1n 8n 20n)
.ends ideal_source
"    )
)

("thevenin_source"
  ("ESpice"
".subckt thevenin_source 1 2
R1 1 3 50
V1 3 2 PWL (0 1 2e-008 1.0 2.1e-008 0 4e-008 0 4.1e-008 1.0 6e-008 1.0 6.1e-008 0
1e-007 0)
.ends thevenin_source
  (PinConnections
    (1,2)
    (2,1)
  )
"    )
)
```

Table-Driven (VI) Diode

Allegro® SI supports Generic Element Diodes that are mostly ideal and hard-coded. You can use a table-driven diode that embeds series resistance and junction capacitance as shown in the following example:

```
("table_diode"
  ("ESpice"
".subckt table_diode 1 2
Rseries 1 3 0.2
Cjunction 3 0 1.2p
G1 anode 2 PWL 1 2
datapoints vi
0      0
0.6    0
0.65   3e-3
0.7    6e-3
1.0    15e-3
```

Allegro PCB SI User Guide

Cadence ESpice Language Reference

```
2.3      300e-3
end datapoints vi
.ends table_diode")
)
```