HSPICE® User Guide: Signal Integrity

Version E-2010.12, December 2010

SYNOPSYS®

Copyright Notice and Proprietary Information

Right to Copy Documentation

The license agreement with Synopsys permits licensee to make copies of the documentation for its internal use only. Each copy shall include all copyrights, trademarks, service marks, and proprietary rights notices, if any. Licensee must assign sequential numbers to all copies. These copies shall contain the following legend on the cover page:

> "This document is duplicated with the permission of Synopsys, Inc., for the exclusive use of _____ and its employees. This is copy number _____."

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Registered Trademarks (®)

Synopsys, AMPS, Astro, Behavior Extracting Synthesis Technology, Cadabra, CATS, Certify, CHIPit, CoMET, Design Compiler, DesignWare, Formality, Galaxy Custom Designer, HAPS, HapsTrak, HDL Analyst, HSIM, HSPICE, Identify, Leda, MAST, METeor, ModelTools, NanoSim, OpenVera, PathMill, Physical Compiler, PrimeTime, SCOPE, Simply Better Results, SiVL, SNUG, SolvNet, Syndicated, Synplicity, the Synplicity logo, Synplify, Synplify Pro, Synthesis Constraints Optimization Environment, TetraMAX, UMRBus, VCS, Vera, and YIELDirector are registered trademarks of Synopsys, Inc.

Trademarks (™)

AFGen, Apollo, Astro-Rail, Astro-Xtalk, Aurora, AvanWaves, BEST, Columbia, Columbia-CE, Confirma, Cosmos, CosmosLE, CosmosScope, CRITIC, CustomExplorer, CustomSim, DC Expert, DC Professional, DC Ultra, Design Analyzer, Design Vision, DesignerHDL, DesignPower, DFTMAX, Direct Silicon Access, Discovery, Eclypse, Encore, EPIC, Galaxy, HANEX, HDL Compiler, Hercules, Hierarchical Optimization Technology, High-performance ASIC Prototyping System, HSIM^{plus}, i-Virtual Stepper, IICE, in-Sync, iN-Tandem, Jupiter, Jupiter-DP, JupiterXT, JupiterXT-ASIC, Liberty, Libra-Passport, Library Compiler, Magellan, Mars, Mars-Rail, Mars-Xtalk, Milkyway, ModelSource, Module Compiler, MultiPoint, Physical Analyst, Planet, Planet-PL, Polaris, Power Compiler, Raphael, Saturn, Scirocco, Scirocco-i, Star-RCXT, Star-SimXT, StarRC, System Compiler, System Designer, Taurus, TotalRecall, TSUPREM-4, VCS Express, VCSi, VHDL Compiler, VirSim, and VMC are trademarks of Synopsys, Inc.

Service Marks (^{sm})

MAP-in, SVP Café, and TAP-in are service marks of Synopsys, Inc.

SystemC is a trademark of the Open SystemC Initiative and is used under license.
ARM and AMBA are registered trademarks of ARM Limited.
Saber is a registered trademark of SabreMark Limited Partnership and is used under license.
All other product or company names may be trademarks of their respective owners.

Contents

Inside This Manual xiii

The HSPICE Documentation Set xiv

Conventions xvi

Customer Support xvii

Acknowledgments xviii

1. Introduction to Signal Integrity 1

Getting Started on Signal Integrity Simulations 2

Signal Integrity Problems 3

Analog Side of Digital Logic 3

System Design Issues 4

Time Domain Reflectometry (TDR) 6

Performing a TDR Simulation in HSPICE 6

Basic TDR Impedance Analysis 7

Testbench Netlist Overview 11

Input Output Ports 11

Selecting the Risetime of the Incident Wave 12

Simulating the Example DUTs 12

Differential TDR Example Netlist 16

Reference 17

Simulating Circuits with IBIS Models in HSPICE 17

Using the B-element to Instantiate Individual Buffers 17

Using the IBIS Component 18

Determining Output and Input Pins in the IBIS Component 19

Power and Ground Pins in the IBIS Component 21

Package and Pin Parasitics 21

Power Supply of the IBIS Buffer 21

Simulating Circuits with Signetics Drivers 22

Simulating Circuits with Xilinx FPGAs 26

Syntax for IOB (xil_iob) and IOB4 (xil_iob4) 26

Ground-Bounce Simulation 28

Coupled Line Noise 31

Example Syntax 34

2. S-parameter Modeling Using the S-element 35

S-parameter Model 36
Notifications and Limitations 37

Mixed-Mode S-parameters 37
Relating Voltage and Current Waves to Nodal Waves 38
Characterizing Differential Data Transfer Systems 40
Deriving a Simpler Set of Voltage and Current Pairs 40
Using the Mixed-Mode S-parameters (S-element) 42
Mixed-Mode S-parameter Netlist Examples 44

Using the Scattering Parameter Element 44

S-element Syntax 45
Node Example 51

S Model Syntax 52

Pre-Conditioning S-parameters 58

Group Delay Handler in Time Domain Analysis 59

Accelerating S-element Time Domain Performance with Recursive Convolution 60
Multithreading Acceleration for S-element on Linux 62
Ensuring Causality in the Rational Function Model 62
Rational Function Matrix (.rfm) File Format 63

S-element Data File Model Examples 65

Multiport Noise Model for Passive Systems 68
Input Interface 68
Output Interface 69

S-element Noise Model 70
Two-Port Noise Parameter Support in Touchstone Files 70
Input Interface 71
Output Interface 72
Notifications and Limitations 73

Small-Signal Parameter Data Frequency Table Model (SP Model) 73
SP Model Syntax 73
Four Valid Forms of the SP Model 76

S Model Data Smoothing 82
Data Smoothing Methods 82

S-model Syntax 83
Predicting an Initial Value for FMAX in S-element Models 84
De-embedding S-parameters 86
S-parameter Standalone Manipulation Utility (SPutil) 87
SPutil Program Features 88
Invoking the Utility 88
SPutil Runset Format 89
Commands 89
Status Messages 92
References 93

3. W-element Modeling of Coupled Transmission Lines 95
Equations and Parameters 96
Frequency-Dependent Matrices 98
Introduction to the Complex Dielectric Loss Model 99
Fitting Procedure Triggered by INCLUDEGDIMAG Keyword 101
Determining Matrix Properties 102
Using the PRINTZO Option 104
Printing Frequency-Dependent Impedance in Mixed Mode 105
File Description for *.wzo 108
Wave Propagation 109
Propagating a Voltage Step 111
Handling Line-to-Line Junctions 113
Using the W-element 115
W-element Capabilities 116
Control Frequency Range of Interest for Greater Accuracy 117
Setting .OPTION RISETIME 118
Using DELAYOPT Keyword for Higher Frequency Ranges 118
Using DCACC Keyword for Lower Frequency Ranges 119
W-element Time-Step Control in Time Domain 119
Time-Step Control 120
Using Dynamic Time-Step Control 120
Input Syntax for the W-element 121
Input Model 1: W-element, RLGC Model 125
Specifying the RLGC Model in an External File 130
Input Model 2: U-element, RLGC Model 132
Using RLGC Matrices 133

Input Model 3: Built-in Field-Solver Model 136
Input Model 4: Frequency-Dependent Tabular Model 136
Notation Used 137
Table Model Card Syntax 137
Examples: 4-Conductor Tx Line and RLGC Model List 139
Introducing Causality Check for W-element RLGC Table Model . . . 141
Input Model 5: S Model 144
S Model Conventions 145
S Model Example 145
Extracting Transmission Line Parameters (Field Solver) 146
Using the Field Solver Model 147
Filament Method 147
Modeling Geometries 149
Solver Limitation 149
Field-Solver-Related Netlist Statements 149
Field Solver Model Syntax 150
Using the Field Solver to Extract a RLGC Tabular Model. 151
Accounting for Surface Roughness Effect in HSPICE W-element . . 155
Accelerating the W-element Field Solver Using an Iterative Solver 157
Visualizing Cross-Sectional Geometric Information 157
Field Solver Examples 159
Example 1: Cylindrical Conductor Above a Ground Plane 159
Example 2: Stratified Dielectric Media 161
Example 3: Two Traces Between Two Ground Planes 164
Example 4: Using Field Solver with Monte Carlo Analysis 165
W-element Passive Noise Model 170
Input Interface 170
Output Interface 171
Using the TxLine (Transmission Line) Tool Utility 172
Invoking the TxLine Tool. 173
Getting Started with TxLine Tool 173
References. 175

4. Modeling Input/Output Buffers Using IBIS Files 177
Verifying IBIS Files with the Golden Parser 178
Using IBIS Buffer 'Models' 179
IBIS Syntax Conventions for I/O Buffers 180
Troubleshooting Signal Propagation Issues 181

Terminology 182
Buffer Types 182
Input Buffer 183
Output Buffer 185
Tristate Buffer 188
Input/Output Buffer 191
Open Drain, Open Sink, Open Source Buffers 194
I/O Open Drain, I/O Open Sink, I/O Open Source Buffers 195
Input ECL Buffer 195
Output ECL Buffer 196
Tristate ECL Buffer 197
Input-Output ECL Buffer 199
Terminator Buffer 200
Series Buffer 202
Series Switch Buffer 202
Multilingual Model Support 204
Specifying Required and Optional Common Keywords 207
file 208
model 208
buffer 209
typ 210
hsp_ver 212
power 212
interpol 212
xv_pu | xv_pd 213
ramp_fwf | ramp_rwf 214
fwf_tune | rwf_tune 215
rwf_pd_dly | fwf_pu_dly 217
pd_scal | pu_scal | pc_scal | gc_scal | rwf_scal | fwf_scal 218
ss_state 219
rm_dly_rwf|rm_dly_fwf|rm_tail_rwf|rm_tail_fwf 219
nowarn 220
c_com_pu | c_com_pd | c_com_pc | c_com_gc 220
detect_oti_mid 222
time_control 222
.OPTION D_IBIS 223
Differential Pins 223
Buffers in Subcircuits 224

Netlist Example with Output Buffer, Transmission Line, and Input Buffer 227

Using the IBIS Component Command 228
How .IBIS Creates Buffers 228
Required Keywords 230
file='*file_name*' 230
component=*'component_name'* 230
Optional Keywords 230
package 230
pkgfile='pkg_file_name' 231
[Model Selector] Support 231
Other Optional Keywords 231
Component Calls for SPICE or Verilog-A Formatted Pins 232
Component Calls for SPICE or Verilog-A Formatted [External Circuit] . . 233
Buffer Power 233
Buffer Power ON 233
Buffer Power OFF 238

Using IBIS Package Modeling 242
Accessing Nets inside a Package Model 243

Using IBIS Board-Level Components 244
.EDB and .IBIS Command Syntax 246
Circuit Topology Created by the .EBD and .IBIS Commands 247
B-element Naming Rules 248
Circuit Topology Created with SPICE or Verilog-A Formatted Pins 251
SPICE or Verilog-A Formatted B-element Naming Rules 252
IBIS Board-Level Component Examples 255

Using IBIS Interconnect Modeling (ICM) 261

5. Ideal and Lumped Transmission Line Models 263

Selecting Ideal or Lossy Transmission Line Elements 264
Source Properties 265
Interconnect Properties 265
Selection of Ideal or Lossy Transmission Line Elements 267
Transmission Lines: Example 269

Ideal T-element Transmission Lines 269
Syntax 269
Lossless Voltage and Current Propagation 271
Ideal Transmission Line Model 273

The Lossy U-element Transmission Line 275

Syntax 275
The U Model for Transmission Lines 276
Selecting U Models 277
Lossy U Model Parameters for Planar Geometries 279
Common Planar Model Parameters 280
Physical Parameters. 281
Loss Parameters 282
Geometric Parameter Recommended Ranges 282
Reference Planes and HSPICE Ground with LLEV 283
Estimating the Skin Effect Frequency. 285
Number of Lumped-Parameter Sections 285
Ringing 286
Geometric Parameters (ELEV=1). 286
Lossy U Model Parameters for Geometric Coax (PLEV=2, ELEV=1) . . . 291
Lossy U Model Parameters Geometric Twinlead (PLEV=3, ELEV=1) . . . 292
Precomputed Model Parameters (ELEV=2) 295
Conductor Width Relative to Reference Plane Width. 297
Alternative Multi-conductor Capacitance/Conductance Definitions . 297
Measured Parameters (ELEV=3) 300
U-element Examples, Models, and Applications 302
Three Coupled Lines, Stripline Configuration 303
Three Coupled Lines, Sea of Dielectric Configuration 308
Simulation Output. 313
IcWire Output Section 314
Capacitance and Inductance Matrices. 316
Five Coupled Lines, Stripline Configuration 318
U Model Applications 319
Data Entry Examples 319
Printed Circuit Board Models 320
Via Modeling for PCBs in HSPICE 321
Coax Models 322
Twinlead Models. 324
Two Coupled Microstrips 324
Solving Ringing Problems with U-elements 326
Oscillations Due to Simulation Errors 326
Timestep Control Error 326
Incorrect Number of Element Lumps 327
Default Computation. 327
Using a Multi-Stage RC Filter to Prevent Ringing 328
Signal Reflections Due to Impedance Mismatch 331

A. Transmission Line Theory 335

Lossless Transmission Line Model 335

Lossy Transmission Line Model 336

Impedance 337

Impedance of Simple Lumped Elements 338

Characteristic Impedance 338

Inductance 340

Mutual Inductance and Self Inductance 340

Operational Definition of Inductance 341

Mutual Inductance 341

Self Inductance 342

Reference Plane Return Paths 342

Crosstalk in Transmission Lines 343

Risetime, Bandwidth, and Clock Frequency 344

Definitions of Transmission Line Terms 346

Relationships and Rules of Thumb 348

Time and Frequency Relationships 348

Transmission Line Effects 349

Intrinsic Properties 349

Reflections 350

Loss and Attenuation 351

Physical Design Quantities 352

Attenuation in Transmission Lines 353

Physical Basis of Loss 354

Skin Depth 355

Dielectric Loss 356

Lossy Transmission Line Model 357

Attenuation Due to Conductor Resistance 358

Attenuation Due to the Dielectric 359

Integrating Attenuation Effects 359

References 361

B. Time Domain Reflectometry (TDR) 363

Optimizing Time Domain Reflectometry (TDR) Packaging 363

Using TDR in Simulation 363
TDR Optimization Procedure 365
Performing a TDR Simulation in HSPICE 368
Application Example: TDR Simulation 368

Index 381

Contents

About This Manual

This manual describes how to use HSPICE to maintain signal integrity in your chip design.

Inside This Manual

This manual contains the chapters described below. For descriptions of the other manuals in the HSPICE documentation set, see the next section, The HSPICE Documentation Set.

Chapter	Description
Chapter 1, Introduction to Signal Integrity	Describes some of the factors that can affect signal integrity in your design.
Chapter 2, S-parameter Modeling Using the S-element	Describes S-parameter and SP modeling as well as other topics related to the S Element
Chapter 3, W-element Modeling of Coupled Transmission Lines	Describes how to use basic transmission line simulation equations and an optional method for computing the parameters of transmission line equations.
Chapter 4, Modeling Input/ Output Buffers Using IBIS Files	Describes how to model input and output buffers using SIBI. Includes information on SIBI conventions, buffers, and the SIBI golden parser.
Chapter 5, Ideal and Lumped Transmission Line Models	Describes how to model ideal and lumped transmission lines.

Chapter	Description
Appendix A, Transmission Line Theory	Relates distributed RLGC values of a transmission line to its characteristic impedance, transmission velocity, and loss; uses the concepts of self and mutual inductance to explain crosstalk; describes rules of thumb for various types of clock pulses; discusses the sources of transmission line attenuation.
Appendix B, Time Domain Reflectometry (TDR)	Discusses using digitized TDR in conjunction with HSPICE to select design components.

The HSPICE Documentation Set

This manual is a part of the HSPICE documentation set, which includes the following manuals:

Manual	Description
HSPICE User Guide: Simulation and Analysis	Describes how to use HSPICE to simulate and analyze your circuit designs, and includes simulation applications. This is the main HSPICE user guide.
HSPICE User Guide: RF Analysis	Describes how to use special set of analysis and design capabilities added to HSPICE to support RF and high-speed circuit design.
HSPICE Reference Manual: Commands and Control Options	Provides reference information for HSPICE and HSPICE RF commands and options.
HSPICE Reference Manual: Elements and Device Models	Describes standard models you can use when simulating your circuit designs in HSPICE, including passive devices, diodes, JFET and MESFET devices, and BJT devices.
HSPICE Reference Manual: MOSFET Models	Describes available MOSFET models you can use when simulating your circuit designs in HSPICE.

Manual	Description
HSPICE Integration to CadenceTM Virtuoso® Analog Design Environment User Guide	Describes use of the HSPICE simulator integration to the Cadence tool.
AvanWaves User Guide	Describes the AvanWaves tool, which you can use to display waveforms generated during HSPICE circuit design simulation.

Searching Across the HSPICE Documentation Set

You can access the PDF format documentation from your install directory for the current release by entering `-docs` on the terminal command line when the HSPICE tool is open.

Synopsys includes an index with your HSPICE documentation that lets you search the entire HSPICE documentation set for a particular topic or keyword. In a single operation, you can instantly generate a list of hits that are hyper-linked to the occurrences of your search term. For information on how to perform searches across multiple PDF documents, see the HSPICE release notes.

Note: To use this feature, the HSPICE documentation files, the Index directory, and the index.pdx file must reside in the same directory. (This is the default installation for Synopsys documentation.) Also, Adobe Acrobat must be invoked as a standalone application rather than as a plug-in to your web browser.

You can also invoke HSPICE and HSPICE RF command help by entering `-help` on your terminal command line when the HSPICE tool is open. This opens a browser-based help system for fast navigation to commands and options used in HSPICE and the HSPICE RF flow.

Known Limitations and Resolved STARs

You can find information about known problems and limitations and resolved Synopsys Technical Action Requests (STARs) in the *HSPICE Release Notes* shipped with this release. For updates, go to SolvNet.

To access the *HSPICE Release Notes*:

1. Go to https://solvnet.synopsys.com/ReleaseNotes. (If prompted, enter your user name and password. If you do not have a Synopsys user name and password, follow the instructions to register with SolvNet.)
2. Select Download Center> HSPICE> version number> Release Notes.

Conventions

The following typographical conventions are used in Synopsys HSPICE documentation.

Convention	Description
`Courier`	Indicates command syntax.
`Italic`	Indicates a user-defined value, such as *`object_name`*.
`Bold`	Indicates user input—text you type verbatim—in syntax and examples.
[ ]	Denotes optional parameters, such as: `write_file [-f filename]`
`...`	Indicates that parameters can be repeated as many times as necessary: *`pin1 pin2 ... pinN`*
|	Indicates a choice among alternatives, such as `low | medium | high`
+	Indicates a continuation of a command line.
/	Indicates levels of directory structure.
Edit > Copy	Indicates a path to a menu command, such as opening the Edit menu and choosing Copy.
Control-c	Indicates a keyboard combination, such as holding down the Control key and pressing c.

Customer Support

Customer support is available through SolvNet online customer support and through contacting the Synopsys Technical Support Center.

Accessing SolvNet

SolvNet includes an electronic knowledge base of technical articles and answers to frequently asked questions about Synopsys tools. SolvNet also gives you access to a wide range of Synopsys online services, which include downloading software, viewing Documentation on the Web, and entering a call to the Support Center.

To access SolvNet:

1. Go to the SolvNet Web page at http://solvnet.synopsys.com.
2. If prompted, enter your user name and password. (If you do not have a Synopsys user name and password, follow the instructions to register with SolvNet.)

If you need help using SolvNet, click Help on the SolvNet menu bar.

The link to any recorded training is
https://solvnet.synopsys.com/trainingcenter/view.faces

Access recent release update training by going to
https://solvnet.synopsys.com/search/advanced_search.faces

Contacting the Synopsys Technical Support Center

If you have problems, questions, or suggestions, you can contact the Synopsys Technical Support Center in the following ways:

- Open a call to your local support center from the Web by going to http://solvnet.synopsys.com/EnterACall (Synopsys user name and password required).
- Send an e-mail message to your local support center.
  - E-mail support_center@synopsys.com from within North America.

  - Find other local support center e-mail addresses at http://www.synopsys.com/support/support_ctr.
- Telephone your local support center.
  - Call (800) 245-8005 from within the continental United States.
  - Call (650) 584-4200 from Canada.
  - Find other local support center telephone numbers at http://www.synopsys.com/support/support_ctr.

Acknowledgments

1

Introduction to Signal Integrity

Describes some of the factors that can affect signal integrity in your design.

The performance of an IC design is not limited to how many million transistors a vendor fits on a single chip. With tighter packaging space and increasing clock frequencies, packaging issues and system-level performance issues (such as crosstalk and transmission lines) become increasingly significant. At the same time, the popularity of multi-chip packages and increased I/O counts is forcing package design to become more like chip design.

Note: The measurement system in this manual always refers to MKS units (meter, kilogram, second measurement), unless otherwise stated. HSPICE expects length and width units of meters. But HSPICE does directly support units of “mil” (.001inch, 25.4e-06 meters) as input. For example, a transmission line is defined with a length of .4 inches:

```
T1 IN 0 OUT 0 Z0=50 f=1meg L=400mil
```

To get the results you expect, use caution when mixing units of measure. For reference, here are other “m” units. Mega, which can be expressed as “meg” or “x”, is often confused with “m” (mili):

1m = 1e-3 (mili)

1meg = 1x = 1e6 (mega)

1u = 1e-6 (micro)

The following topics are discussed in this chapter:

- Getting Started on Signal Integrity Simulations
- Time Domain Reflectometry (TDR)

- Simulating Circuits with IBIS Models in HSPICE
- Simulating Circuits with Signetics Drivers
- Simulating Circuits with Xilinx FPGAs
- Example Syntax

Getting Started on Signal Integrity Simulations

In a signal integrity simulation, you must model the following components:

- Driver cell, including parasitic pin capacitances and package lead inductances
- Transmission lines
- A receiver cell with parasitic pin capacitances and package lead inductances
- Terminations or other electrical elements on the line

Model the transmission line as closely as possible—that is, to maintain the integrity of the simulation, include all electrical elements exactly as they are laid out on the backplane or printed circuit board.

You can use readily-available I/O drivers from ASIC vendors, and the HSPICE device models advanced lossy transmission lines to simulate the electrical behavior of the board interconnect, bus, or backplane. You can also analyze the transmission line behavior under various conditions.

In addition, HSPICE or HSPICE RF preserves the necessary electrical characteristics with full transistor-level library circuits.

HSPICE or HSPICE RF can simulate systems by using:

- System-level behavior, such as local component temperature and independent models to accurately predict electrical behavior.
- Automatic inclusion of library components by using the `SEARCH` option.
- Lossy transmission line models that:
  - Support common-mode simulation.
  - Include ground-plane reactance.
  - Include resistive loss of conductor and ground plane.
  - Allow multiple signal conductors.

- Require minimum CPU computation time.

Signal Integrity Problems

Table 1 lists some of the signal integrity problems that can cause failures in high-speed designs.

Table 1 High-Speed Design Problems and Solutions

Signal Integrity Problem	Causes	Solution
Noise: delta I (current)	Multiple simultaneously-switching drivers; high-speed devices create larger delta I	Adjust or evaluate location, size, and value of decoupling capacitors.
Noise: coupled (crosstalk)	Closely-spaced parallel traces	Establish design rules for lengths of parallel lines.
Noise: reflective	Impedance mismatch	Reduce the number of connectors, and select proper impedance connectors.
Delay: path length	Poor placement and routing; too many or too few layers; chip pitch	Choose MCM or other high-density packaging technology.
Propagation speed	Dielectric medium	Choose the dielectric with the lowest dielectric constant.
Delay: rise time degradation	Resistive loss and impedance mismatch	Adjust width, thickness, and length of line.

Analog Side of Digital Logic

Circuit simulation of a digital system becomes necessary when the analog characteristics of the digital signals become electrically important. The integrity of the digital quality of the signals require careful circuit analysis.

The roadblocks to successful high-speed digital designs are noise and signal delays. Digital noise can originate from several sources. The fundamental digital noise sources are:

- Line termination noise—additional voltage reflected from the load back to the driver, which is caused by an impedance mismatch. Digital output buffers are not designed to accurately control the output impedance. Most buffers have different rising and falling edge impedances.
- Ground bounce noise—noise generated where leadframes or other circuit wires cannot form into transmission lines. The resulting inductance creates an induced voltage in the ground circuit, supply circuit, and output driver circuit. Ground bounce noise lowers the noise margins for the rest of the system.
- Coupled line noise—noise induced from lines that are physically adjacent. This noise is generally more severe for data lines that are next to clock lines.

As system cycle times approach the speed of electromagnetic signal propagation for the printed circuit board, consideration of the line length becomes critical. The system noises and line delays interact with the electrical characteristics of the gates, and might require circuit level simulation.

System Design Issues

Exceeding the noise quota might not cause a system to fail. Maximum noise becomes a problem only when HSPICE accepts a digital input. If a digital systems engineer can decouple the system, HSPICE or HSPICE RF can accept a much higher level of noise.

Some common methods that a digital systems engineer can use to decouple a system include:

- Multiple ground and power planes on the printed circuit board (PCB), multi-chip module (MCM), and pin grid array (PGA).
- Separating signal traces with ground traces.
- Decoupling capacitors.
- Series resistors on output buffer drivers.
- Twisted-pair line driving.

In present systems designs, you must select the best packaging methods at three levels:

- PCB
- MCM
- PGA

Extra ground and power planes are often necessary to lower the supply inductance and to provide decoupling.

- Decoupling capacitors must have very low internal inductance to be effective for high-speed designs.
- Newer designs frequently use series resistance in the output drivers to lower circuit ringing.
- Critical high-speed driver applications use twisted differential-pair transmission lines.

A systems engineer must determine how to partition the logic. The propagation speed of signals on a printed circuit board is about 6 in/ns. As digital designs become faster, wiring interconnects become a factor in how you partition logic.

The critical wiring systems are:

- IC-level wiring
- Package wiring for SIPs, DIPs, PGAs, and MCMs
- Printed circuit-board wiring
- Backplane and connector wiring
- Long lines – power, coax, or twisted pair

If you use ASIC or custom integrated circuits as part of your system logic partitioning strategy, you must make decisions about integrated circuit level wiring. The more-familiar decisions involve selecting packages and arranging packages on a printed circuit board. Large systems generally have a central backplane, which becomes the primary challenge at the system partition level.

Use the following equation to estimate wire length when transmission line effects become noticeable:

```
critical length=(rise time)*velocity/8
```

For example, if rise time is 1 ns and board velocity is 6 in/ns, then distortion becomes noticeable when wire length is 3/4 in. The HSPICE or HSPICE RF circuit simulator contains a field solver to extract full loss transmission line models and the linear analysis (.LIN) feature can extract S-parameter models for packages or complex interconnects.

Time Domain Reflectometry (TDR)

Packaging plays an important role in determining the overall speed, cost, and reliability of a system. With small feature sizes and high levels of integration, a significant portion of the total delay is the time required for a signal to travel between chips.

Time domain reflectometry (TDR) is the closest measurement to actual digital component functions. It provides a transient display of the impedance versus time for pulse behavior. See Performing a TDR Simulation in HSPICE following.

Performing a TDR Simulation in HSPICE

When performing a signal integrity analysis, a bit stream is often the first stimulus used to validate the system. If poor results are obtained, it can be useful to first perform a TDR simulation on either the system or the individual models to find unexpected impedance discontinuities or mismatches. A TDR simulation measures the reflections that result from an ideal step edge traveling through a transmission medium.

The following is an example HSPICE testbench to demonstrate TDR simulations with a variety of classic discontinuities. The included testbench is completely functional and uses single-ended lossless transmission lines and lumped elements to demonstrate the analysis but could be easily adapted to use more complex W-element or S-parameter models. A differential equivalent example netlist is supplied in the last section.

HSPICE provides a number of methods to model transmission lines and media. These include:

- Lumped models with R-, L-, and C-elements connected by lossless (T-element) transmission lines
- S-parameter data (S-element), either from instrument measurements or extracted during previous HSPICE simulations
- The frequency-dependant lossy W-element. The W-element has several different modeling options:
  - The HSPICE field-solver based on materials and geometries
  - RLGC per unit length matrices
  - S-parameter data called by the W-element SMODEL parameter

- A network of behavioral sources

A time domain reflectometry (TDR) simulation measures the reflections that result from an ideal step edge traveling through a transmission medium. This allows you to analyze impedance discontinuities and mismatches. The medium being studied is often an interconnect topology, and could be a combination of circuit board traces, cables, connectors and even the wire and bonding between an IC and its package. The following uses an example HSPICE testbench to demonstrate TDR simulations with a variety of classic discontinuities.

An interconnect simulation often contains a combination of these methods. Singularly or collectively you can refer to them as the Device Under Test (DUT). It is common for the DUT model, or portions of it to be provided by the manufacturer (as in the case of a connector) or be output by a signal integrity analysis tool which creates a model by analyzing the packaging or layout data.

The following sections discuss these topics:

- Basic TDR Impedance Analysis
- Testbench Netlist Overview
- Input Output Ports
- Selecting the Risetime of the Incident Wave
- Simulating the Example DUTs
- Differential TDR Example Netlist
- Reference

Basic TDR Impedance Analysis

The resulting waveform of the TDR simulation is the combination of the incident wave and reflections that occur when the step edge encounters impedance variations.

For this example and netlist, the following names and equations are used:

- Vincident: An ideal copy of the step edge source used as a reference
- Vmeasured: Voltage at the input to the DUT which is the combination of incident and reflected waves
- Vreflect: The reflected portion only which is obtained by subtracting the incident portion from the measured voltage
- Impedance (Z0): Calculated from the above parameters:

$$\frac{Z_{ref} \cdot (V_{incident} + V_{reflect})}{(V_{incident} - V_{reflect})}$$

Using this calculation, you can graph the response of the transmission line in units of characteristic impedance which is often more useful than plotting the voltage of the TDR measured wave. In at least one case, however, the open circuit termination, it is more useful to plot the measured voltage. Note that due to the voltage divider effect, the measured voltage is 1/2 of the incident wave. The following examples demonstrate classic impedance mismatches and discontinuities.

Example 1: Short Circuit Termination

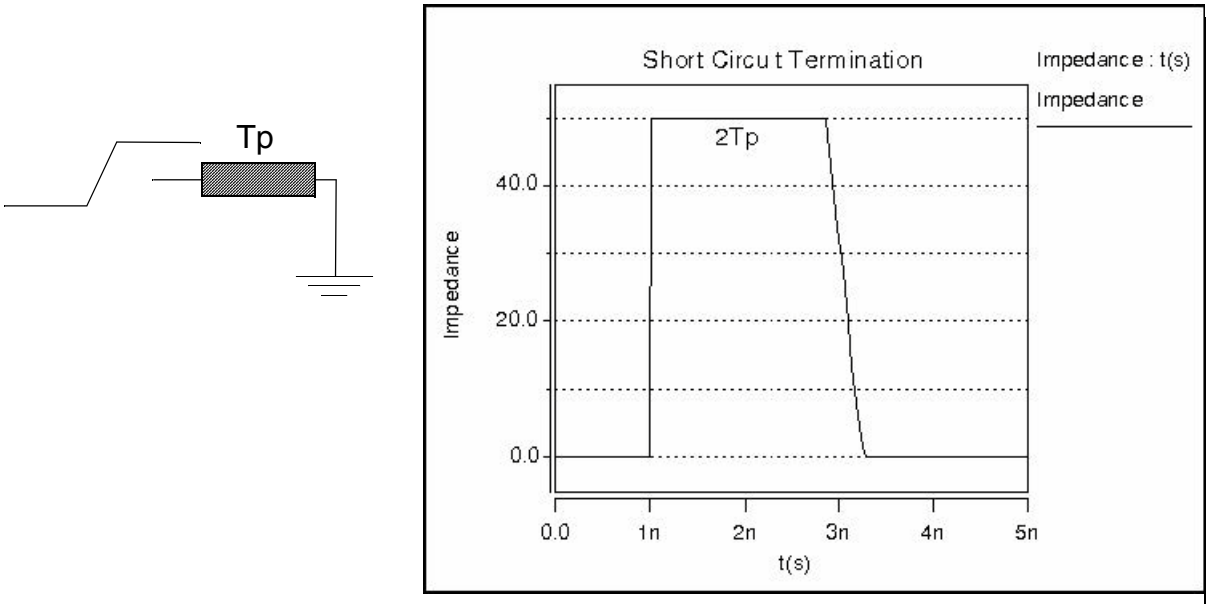


Figure 1 Short Circuit Termination

Example 2: Open Circuit Termination

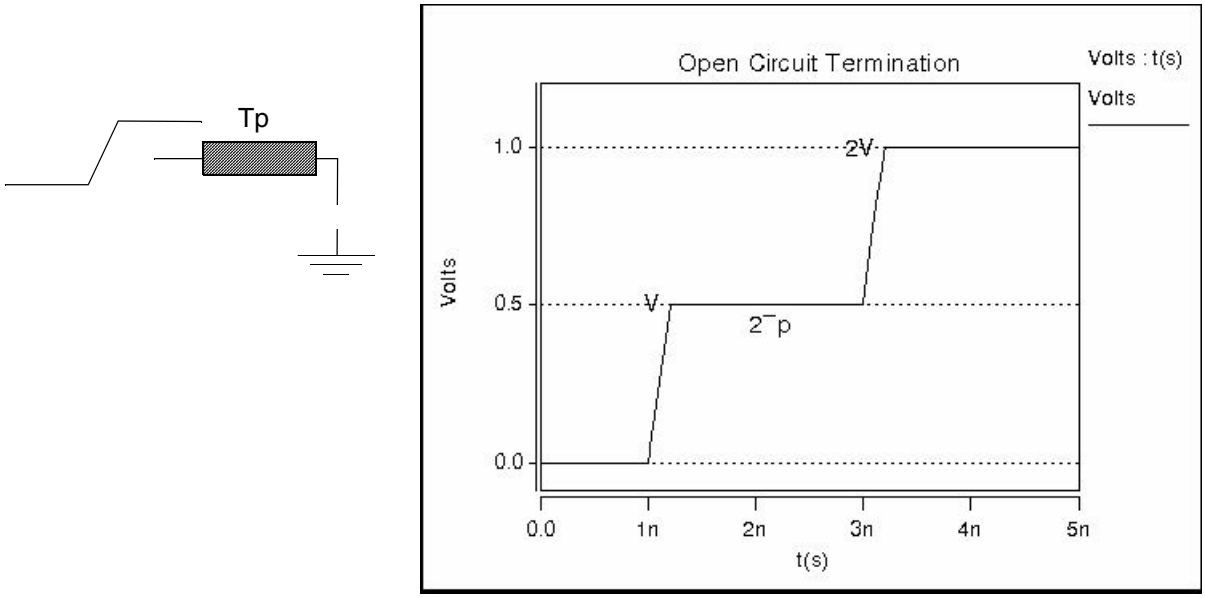


Figure 2 Open Circuit Termination

Example 3: Mismatched Load Termination

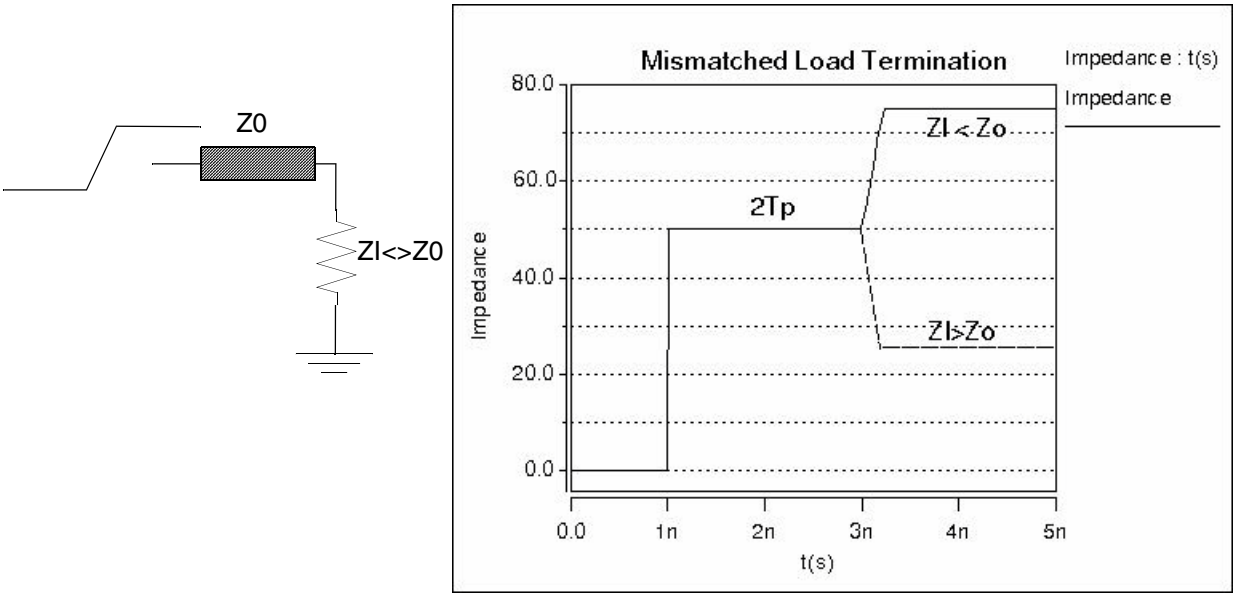


Figure 3 Mismatched Load Termination

Example 4: Shunt Capacitance Discontinuity

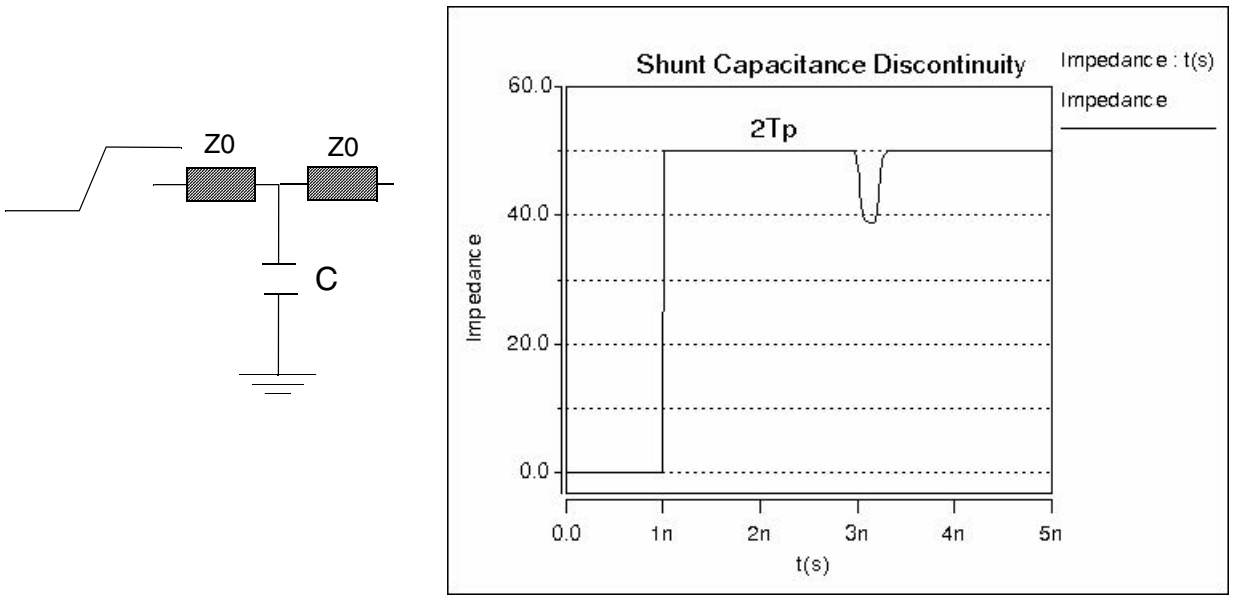


Figure 4 Shunt Capacitance Discontinuity

Example 5: Series Inductance Discontinuity

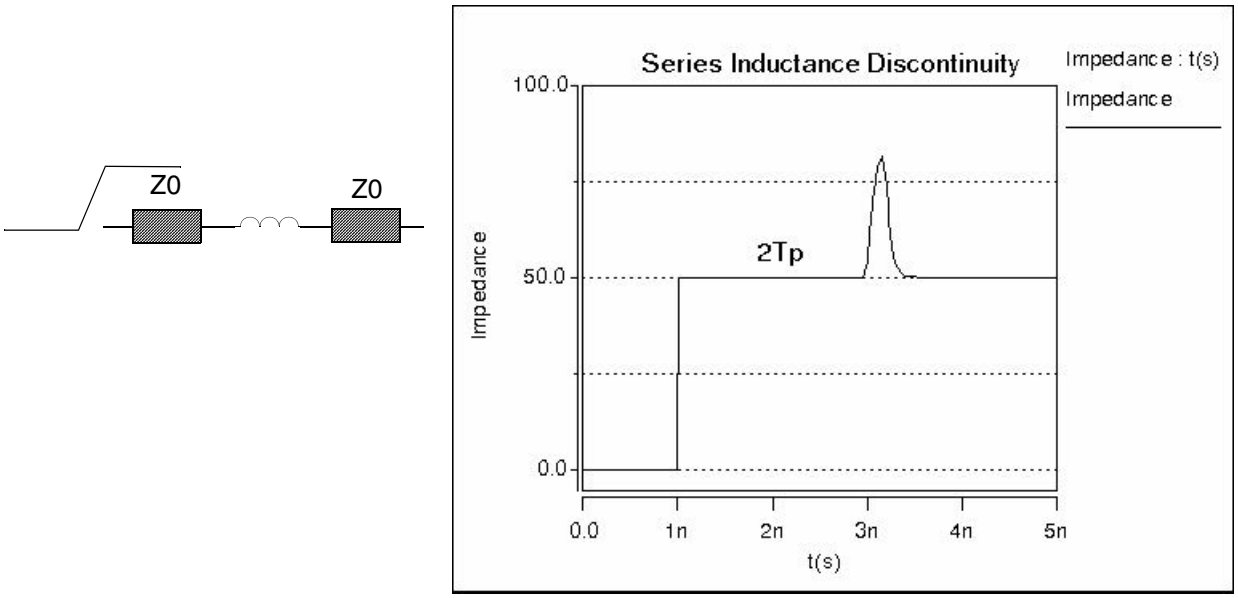


Figure 5 Series Inductance Discontinuity

Testbench Netlist Overview

In creating an HSPICE TDR test bench, this example uses the “P” (port) element for both the input sources and output terminations because it can:

- Represent either single-ended or differential signals, and
- Provide both a source voltage and impedance value in a single construct

The first example simulates a single-ended TDR example. Additional syntax for a differential testbench and DUT is provided below in Differential TDR Example Netlist.

The testbench is composed of four main sections:

1. The port representing the source voltage and impedance for the incident wave
2. The Device Under Test (DUT)
3. An output termination port
4. A reference port and matching termination port

The function of the reference port is to create an ideal version of the incident pulse which is isolated from the DUT and is used to calculate the reflected portion of the TDR measurement.

Input Output Ports

The P element is the basis for the input and output ports. As described above, it can be visualized as an impedance and a voltage source in series:

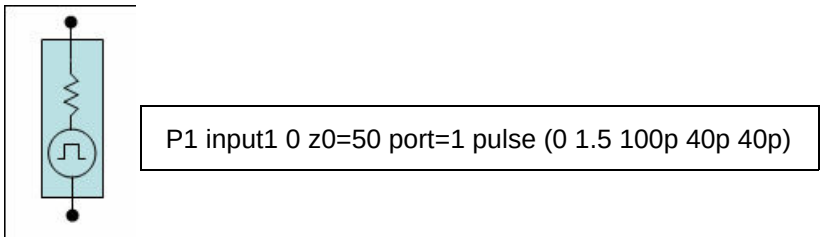


Figure 6 HSPICE Port element

In the example shown in Figure 6, a port named “P1” connects between “input1” and ground and has a characteristic impedance of 50 ohms. It supplies a single 40ps rising edge to input1 after a 100ps delay. If you omit the voltage source argument, the port serves as a simple termination. Each port must have

a unique sequential numerical designation in the “port=” argument. The P-element is also used in the .LIN analysis to extract S-parameters.

For more information on port elements see Port Element in the *HSPICE User Guide: Simulation and Analysis*.

Selecting the Risetime of the Incident Wave

One way to determine the risetime of the incident wave is using the knee frequency. To guarantee reliable operation of a digital system, you should develop and verify the circuit design for frequencies below the knee frequency. It can be shown that most energy in digital pulses concentrates below the knee frequency and that the behavior of a circuit at the knee frequency determines its ability to process a step edge. The knee frequency for any digital signal is related to the rise and fall time of its digital edges rather than its clock rate.

The risetime can be calculated based the desired knee frequency and depends on whether it is based on a 10-90% or 20-80% measurement.

- For a 10-90% measurement, Trise = .5/Fknee
- For a 20-80% measurement, Trise = .35/Fknee

For example, the risetime measured at 20-80% needed for a 10GHz knee frequency:

$$Trise = \frac{.35}{10e9\text{Hz}} = 35\text{ps}$$

In an HSPICE trapezoidal pulse source function, the risetime is specified from 0-100%.

So for a 20-80% risetime, multiply the desired risetime by 1.67 to obtain the “tr” parameter and for a 10-90% measured risetime, multiply the desired risetime by 1.25. For example:

$$(35\text{ps measured at 20-80\% }) \cdot 1.67 = \text{tr parameter} = 58.5\text{ps}$$

Simulating the Example DUTs

The sample netlist includes three separate example DUTs:

1. An inductive discontinuity
2. A capacitive shunt discontinuity

3. A transmission line followed by a shunt capacitance discontinuity, an impedance mismatch in the second transmission line and a series inductance

The following example shows a combination of behaviors. The netlist for simulation is shown below Figure 7. The results are shown in Figure 8 on page 15.

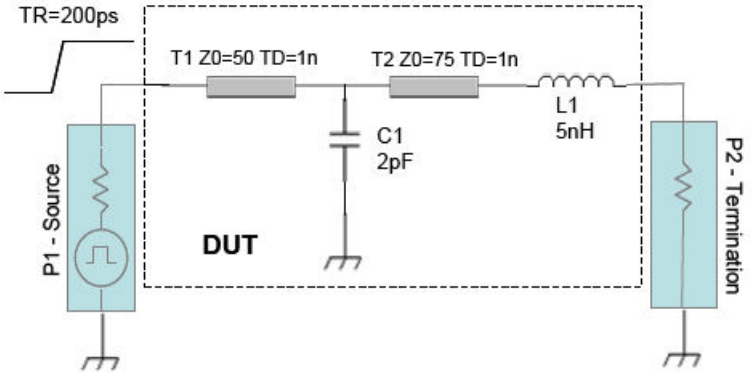


Figure 7 Testbench and DUT

The example test bench below, “HSPICE TDR Netlist with DUT3 Uncommented”, is completely functional and uses simple lossless transmission lines and lumped elements to demonstrate the analysis, but it could just as easily use more complex W-element or S-parameter models. A differential example follows at the end of this section, and contains appropriate modifications to the ports, DUTs and equations.

Example HSPICE TDR Netlist with DUT3 Uncommented

```
** Single-ended TDR example **
*
.opt post probe runlvl=5
.tran 0.1n 20n
*
.param zref=50.0 vlo=0 vhi=1 td=1n tr=.2n tf=.2n pw=50n per=200n
*
* P1 is the input port for the incident wave. It provides
* both an impedance and a pulse source
*
Psource dut_in 0 z0=zref port=1 pulse(vlo vhi td tr tf)
*
* DUT1 - An inductive discountinuity
*T1 dut_in 0 node1 0 Z0=50 td=1n
*L1 node1 node2 5e-9
*T2 node2 0 dut_out 0 Z0=50 td=1n
*
* DUT2 - A capacitive shunt discountinuity
*T1 dut_in 0 node1 0 Z0=50 td=1n
*C1 node1 0 1e-12
*T2 node1 0 dut_out 0 Z0=50 td=1n
*
* DUT3 - Series inductance - shunt capacitance
* with impedance mismatch in the second transmission
* line
T1 dut_in 0 node1 0 Z0=50 td=1n
C1 node1 0 2e-12
T2 node1 0 node2 0 Z0=75 td=1n
L1 node2 dut_out 5e-9
*
* Pterm is the output port of the DUT and provides termination
*
Pterm dut_out 0 z0=50 port=2
*
* Pref and Prefterm are a reference source and termination to use
* as the ideal incident wave in the reflection calculation
*
Pref vref 0 z0=zref port=3 pulse(vlo vhi td tr tf)
Prefterm vref 0 z0=zref port=4
*
* Calculations of the reflected wave and impedance
*
.probe tran vincident = par('v(vref)')
.probe tran vmeasured = par('v(dut_in)')
.probe tran vreflect = par('vmeasured-vincident')
.probe tran Z0 = par('zref*(vincident+vreflect)/(vincident-
vreflect)')
```

```
*
.end
```

Procedural Notes:

1. The default `RUNLVL` should be increased from `3` to `5` to observe the subtleties in the discontinuities of the waveforms.
2. Alternately, you can specify:
   `.option runlvl=0 accurate delmax=10p`
   Since the pulse is a single event (non-repeating), you can omit the pw (pulse width) and `per` (period) arguments, although a warning message results. Since `delmax` forces the maximum time step length, it may sacrifice the efficiency of the dynamic time step control. Therefore, unless you have a specific need for dense time points, normally it is recommended that you do not specify `delmax`.
3. Uncomment only one DUT at a time to observe the example discontinuities and impedance mismatches.

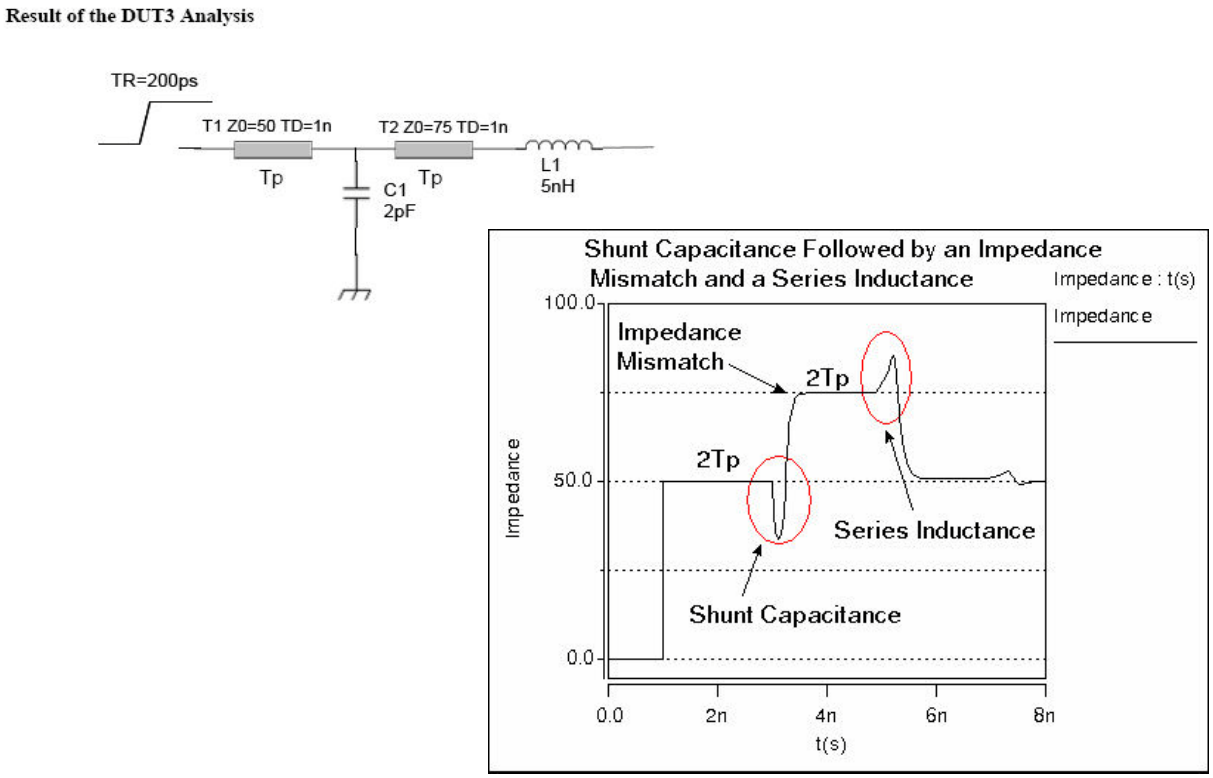


Figure 8 Result of the DUT3 Analysis

Differential TDR Example Netlist

```
** Differential TDR example **
*
.opt post probe runlvl=5
.tran 0.1n 20n
.param zref=50.0 vlo=0 vhi=1 td=1n tr=.2n tf=.2n
* P1 is the differential input port for the incident wave.
* It provides both an impedance and a pulse source
*
Psource dut_inp dut_inn 0 z0=zref port=1 pulse(vlo vhi td tr tf)
*
* DUT1 - An inductive discountinuity
* T1P dut_inp 0 node1p 0 Z0=50 td=1n
* T1N dut_inn 0 node1n 0 Z0=50 td=1n
* L1 node1p node2p 5e-9
* L2 node1n node2n 5e-9
* T2P node2p 0 dut_outp 0 Z0=50 td=1n
* T2N node2n 0 dut_outn 0 Z0=50 td=1n
*
* DUT2 - A capacitive shunt discountinuity
* T1P dut_inp 0 node1p 0 Z0=50 td=1n
* T1N dut_inn 0 node1n 0 Z0=50 td=1n
* C1 node1p 0 1e-12
* C2 node1n 0 1e-12
* T2P node1p 0 dut_outp 0 Z0=50 td=1n
* T2N node1n 0 dut_outn 0 Z0=50 td=1n
*
* DUT3 - A shunt capacitance followed by an impedance mismatch
* and a series inductance
T1P dut_inp 0 node1p 0 Z0=50 td=1n
T1N dut_inn 0 node1n 0 Z0=50 td=1n
C1 node1p 0 2e-12
C2 node1n 0 2e-12
T2P node1p 0 node2p 0 Z0=75 td=1n
T2N node1n 0 node2n 0 Z0=75 td=1n
L1 node2p dut_outp 5e-9
L2 node2n dut_outn 5e-9
*
* Pterm is the dut_output port of the DUT and provides termination
*
Pterm dut_outp dut_outn 0 z0=50 port=2
*
* Pref and Prefterm are a reference source and termination to use
* as the ideal incident wave in the reflection calculation
*
Pref refp refn 0 z0=zref port=3 pulse(vlo vhi td tr tf)
Prefterm refp refn 0 z0=zref port=4
*
```

```
* Calculations of the reflected wave and impedance
*
.probe tran vincident = par('v(refp)-v(refn)')
.probe tran vmeasured = par('v(dut_inp)-v(dut_inn)')
.probe tran vreflect = par('vmeasured-vincident')
.probe tran Zdiff = par('2*zref*(vincident+vreflect)/
(vincidentvreflect)')
*
.end
```

Reference

TDR Impedance Measurements: A Foundation for Signal Integrity Copyright © 2001, Tektronix, Inc.

Simulating Circuits with IBIS Models in HSPICE

There are two ways to instantiate IBIS buffers in your HSPICE testbench. You can instantiate buffers one at a time with the B-element, or you can instantiate all the buffers for a given component using the .IBIS component card. With the IBIS component, package parasitics can be automatically annotated. With the B-element, these must be added as HSPICE circuit elements if desired. For complete discussion of the IBIS model components, see Chapter 4, Modeling Input/Output Buffers Using IBIS Files

The following sections discuss these topics:

- Using the B-element to Instantiate Individual Buffers
- Using the IBIS Component
- Power Supply of the IBIS Buffer

Using the B-element to Instantiate Individual Buffers

If you decide to select and instantiate individual buffers using the B-element syntax, you create the node list and choose what to name the nodes. The list of required nodes for a buffer depends on the buffer type. IBIS models are supported according to the current HSPICE-supported IBIS standard. The complete list of supported buffers can be found in Chapter 4, Buffer Types, but for illustration here are six common types:

- Input buffer:
  B_INPUT nd_pc nd_gc nd_in nd_out_of_in
- Output buffer:
  B_OUTPUT nd_pu nd_pd nd_out nd_in [nd_pc nd_gc]
- Input ECL Buffer:
  B_INPUT_ECL nd_pc nd_gc nd_in nd_out_of_in
- Output ECL Buffer:
  B_OUTPUT_ECL nd_pu nd_out nd_in [nd_pc nd_gc]
- Tri-state buffer:
  B_3STATE nd_pu nd_pd nd_out nd_in nd_en [nd_pc nd_gc]
- Input/Output buffer:
  B_IO nd_pu nd_pd nd_out nd_in nd_en nd_out_of_in [nd_pc nd_gc]

In the preceding examples, the required nodes are listed first, and the optional nodes are in brackets [ ]. The position in the list, not the name, selects the node function. You can name the nodes using any valid HSPICE node name. In these examples: *pu* and *pd* are pull-up and pull-down; *pc* and *gc* are power clamp and ground clamp; *nd* simply stands for node.

The following is a sample B-element instantiation of an output (driver) buffer:

```
b_out1 nd_pu nd_pd out1 in1
+ file = 'at16245.ibs'
+ model = 'AT16245_OUT'
```

This is the minimum syntax to add a B-element. Note that both the file and model names are case-sensitive. There are numerous additional options, many of which are shared with the IBIS component.

See Chapter 4, Specifying Required and Optional Common Keywords for more details.

Using the IBIS Component

To use the .IBIS component card, your IBIS file must have a [Component] section. When you use the IBIS component, HSPICE automatically creates node names for all the buffer pins in the component. These node names are created by concatenating the component name, the pin name, and the pin's function. For node naming purposes, the component name is the user-supplied name (you) give to the .IBIS card—not the component name...

.IBIS ddr256 <--- root of derived node names

For example, the output node of an output buffer (driver) might look like this: ddr256_q1_o where ddr256 is the component name, q1 is the pin name, and _o denotes the driver output.

In that example, “ddr256” is the user-supplied component name, “q1” is the pin name, and “o” means output. HSPICE puts the “_” (underscores) in for readability. You do not have to create these nodes yourself in the netlist. They exist logically, and you can make connections to them simply by using the derived name.

Just as in the B-element node list, the nodes that are created for each buffer depend on the buffer type. For example, a tri-state buffer generates an enable node, where a simple input or output buffer does not.

An easy way to get started with the IBIS component is to create a skeleton netlist with only the IBIS component and a simple analysis. When you probe all signals, you see a list of all the known nodes for each buffer in your waveform tool.

Here is an example of the minimum syntax to instantiate an IBIS component and generate a plotfile:

```
* Instantiate an IBIS component to see the logical nodes created
.ibis ddr256
+ file ='hyb256_400.ibs'
+ component ='hyb25d256'
.tran 1n 5n
.option post probe dcstep=1
.probe v(ddr*)
.end
```

Since all logical nodes of the IBIS component start with ddr256_, using .probe v(ddr256_*) outputs all the nodes created by the IBIS component instantiation. The dcstep option prevents HSPICE from printing numerous warnings about nodes with no path to ground.

The following sections describe these topics:

- Determining Output and Input Pins in the IBIS Component
- Power and Ground Pins in the IBIS Component
- Package and Pin Parasitics

Determining Output and Input Pins in the IBIS Component

How you connect the nodes of a buffer instantiated by an IBIS component depends on if it is an input, output or I/O function. There is a special pin for I/O

buffers named “outofin”. When an IO buffer is in receive mode, the input is external to the component and sits on your SI path. The output of that input buffer (or “outofin”), is inside the component. If you probe the outofin node, you see it is an ideal, behavioral digital waveform. The regular input buffer does not have a separate “outofin” node. Instead, the “_i” node acts as outofin.

Buffer Type	Derived logical node name	Example
Input buffers - receivers		
The input (receiving) node of an input buffer:	'cname'_'pin_name'	ddr256_q1
The output node of an input buffer:	cname'_'pin_name'_i	ddr256_q1_i
Output buffers - drivers		
Input node of an output buffer:	'cname'_'pin_name'_i	ddr256_q1_i
Output node of an output buffer:	'cname'_'pin_name'	ddr256_q1
Die side output node, if package model used:	'cname'_'pin_name'_o	ddr256_q1_o
IO buffers - input/output		
Input of an IO buffer in input mode:	'cname'_'pin_name'	ddr256_q1
Output of an IO buffer in input mode:	'cname'_'pin_name'_outofin	ddr256_q1_outofin
Input of an IO buffer in output mode:	'cname'_'pin_name'_i	ddr256_q1_i
Output node of an IO buffer in output mode:	'cname'_'pin_name'	ddr256_q1
Die side output node of IO buffer in output mode, if package model used:	'cname'_'pin_name'_o	ddr256_q1_o

Buffer Node Names	General Guidelines
Die side output node, if package model used:	'cname'_'pin_name'_o
Output node or input node for input buffers:	'cname'_'pin_name'
Enable node for buffers with enable function:	'cname'_'pin_name'_en
Input node or outofin node for input buffers:	'cname'_'pin_name'_i

Buffer Node Names	General Guidelines
Outofin node for buffers other than input buffers:	'cname'_'pin_name'_outofin

Power and Ground Pins in the IBIS Component

You may wish to externally power your IBIS buffer in certain situations, such as to simulate Simultaneous Switching Noise, or “ground bounce”. Nodes of power and ground pins are created in the “logical” netlist but no buffer is created for them. The pins can also have specific pin or general package parasitics associated with them such as the other IO pins in the component.

Package and Pin Parasitics

The IBIS component gives you the option to automatically annotate the general R_pkg, L_pkg and C_pkg parasitics in the [Package] section, or the pin specific R_pin, L_pin and C_pin parasitics in the [Pin] section, or the complicated package model declared in the [Package Model] section. Please refer the keyword `package` of the .IBIS command.

Power Supply of the IBIS Buffer

For the B-element: The physical pull-up and pull-down nodes are, by default, connected to supply and ground by the *power=on* directive. If you physically connect these nodes to an external voltage source and ground, be sure to set *power=off* in your netlist.

For the IBIS component logical netlist: If there is [Pin Mapping] for this component in the IBIS file, then physical pull-up and pull-down nodes are connected automatically to power and ground pins of the component. You should externally power your buffer from such pins manually. So *power=off* is the default in this situation.

If there is no [Pin Mapping] for this component in the IBIS file, then, like the B-element card, the physical pull-up and pull-down nodes are, by default, connected to the supply and ground by the *power=on* directive. You need not add an external voltage source for them.

Simulating Circuits with Signetics Drivers

HSPICE or HSPICE RF includes a Signetics I/O buffer library in the $*installdir*/parts/signet directory. You can use these high-performance parts in backplane design. An example follows of how to combine these parts with transmission line models.

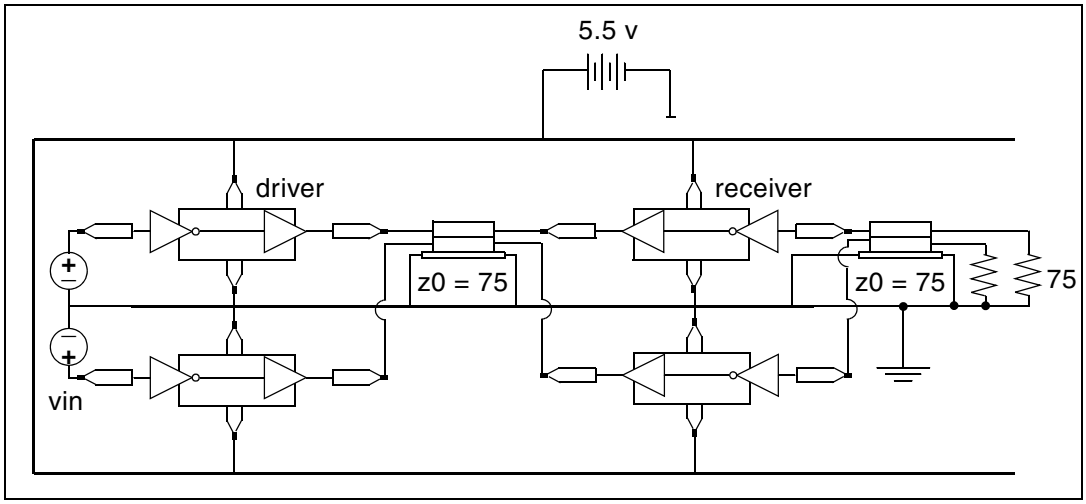


Figure 9 I/O Drivers/Receivers with Package Lead Inductance, Parallel 4" Lossy Microstrip Connectors

The schematic in Figure 9 shows a pair of drivers driving 4 inches of transmission line to a pair of receivers that drive 4 inches of transmission line. A cross-section of the transmission line is shown in Figure 10 on page 23.

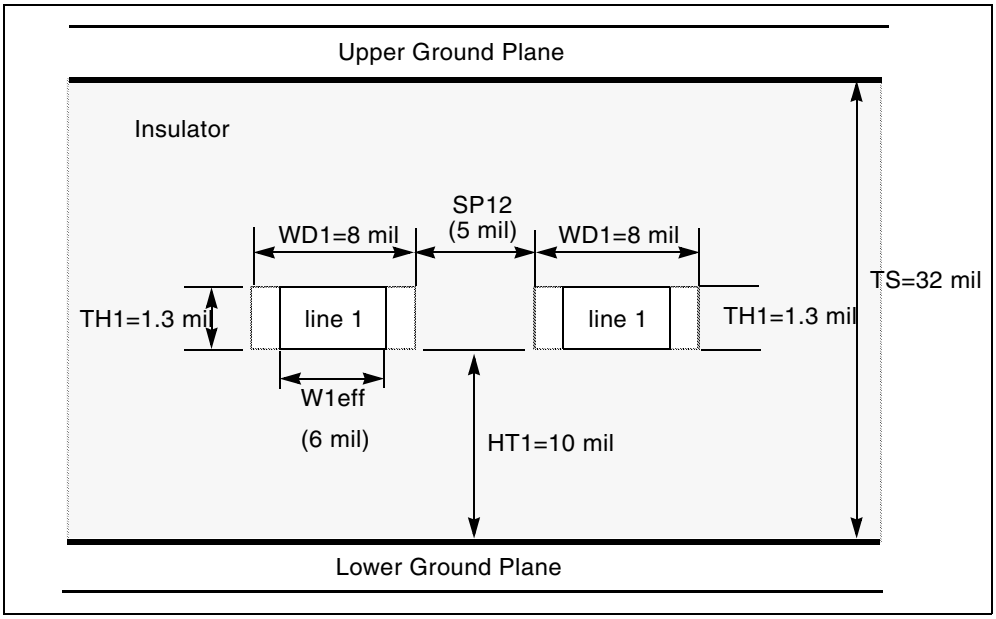


Figure 10 Planar Transmission Line DLEV=2: Microstrip Sea of Dielectric

The following is an example of a chip model with package inductance:

```
.SUBCKT IO_CHIP IN OUT VCC XGND PIN_VCC=7n PIN_GND=1.8n
X1 IN1 INVOUT VCC1 XGND1 ACTINPUT
X2 INVOUT OUT1 VCC1 XGND1 AC109EQ
* Package inductance
LIN_PIN IN IN1 PIN_IN
LOUT_PIN OUT1 OUT PIN_OUT
LVCC VCC VCC1 PIN_VCC
LGND XGND1 XGND PIN_GND
.ENDS
$ TLINE MODEL - 2 SIGNAL CONDUCTORS WITH GND
$ PLANE
.MODEL USTRIP U LEVEL=3 ELEV=1 PLEV=1
+ TH1=1.3mil HT1=10mil TS=32mil KD1=4.5 DLEV=0 WD1=8mil
+ XW=-2mil KD2=4.5 NL=2 SP12=5mil
$ ANALYSIS / PRINTS
.TRAN .1NS 100NS
.PROBE V(STIM1) V(STIM2)       $ Inputs
.PROBE V(TLOUT1) V(TLOUT2) V(TLOUT3) V(TLOUT4)     $ Outputs
.END
```

Simulation results for this model are shown in Figure 11 on page 24.

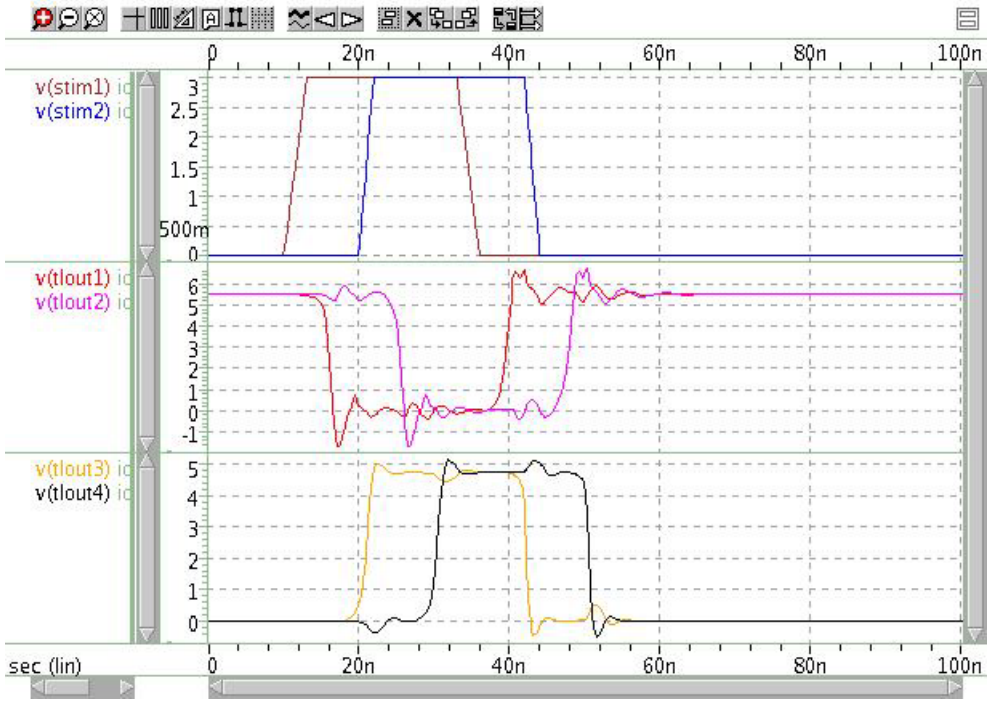


Figure 11 Simulation results of I/O Chips connected with Tlines

Here's a full netlist example of how I/O chips connect with transmission lines:

```
.OPTION SEARCH='$installdir/parts/signet'
.OPTION POST=2 TNOM=27 NOMOD LIST METHOD=GEAR
.TEMP 27
$ DEFINE PARAMETER VALUES
.PARAM LV=0 HV=3 TD1=10n TR1=3n TF1=3n TPW=20n
+ TPER=100n TD2=20n TR2=2n TF2=2n LNGTH=101.6m
$ POWER SUPPLY
VCC VCC 0 DC 5.5
$ INPUT SOURCES
VIN1 STIM1 0 PULSE LV HV TD1 TR1 TF1 TPW TPER
VIN2 STIM2 0 PULSE LV HV TD2 TR2 TF2 TPW TPER
$ FIRST STAGE: DRIVER WITH TLINE
X1ST_TOP STIM1 OUTPIN1 VCC GND IO_CHIP PIN_IN=2.6n
+ PIN_OUT=4.6n
X1ST_DN STIM2 OUTPIN2 VCC GND IO_CHIP PIN_IN=2.9n
+ PIN_OUT=5.6n
U_1ST OUTPIN1 OUTPIN2 GND TLOUT1 TLOUT2 GND USTRIP L=LNGTH
$ SECOND STAGE: RECEIVER WITH TLINE
X2ST_TOP TLOUT1 OUTPIN3 VCC GND IO_CHIP PIN_IN=4.0n
+ PIN_OUT=2.5n
X2ST_DN TLOUT2 OUTPIN4 VCC GND IO_CHIP PIN_IN=3.6n
+ PIN_OUT=5.1n
U_2ST OUTPIN3 OUTPIN4 GND TLOUT3 TLOUT4 GND USTRIP L=LNGTH
$ TERMINATING RESISTORS
R1 TLOUT3 GND 75
R2 TLOUT4 GND 75
$ IO CHIP MODEL - SIGNETICS
.SUBCKT IO_CHIP IN OUT VCC XGND PIN_VCC=7n PIN_GND=1.8n
X1 IN1 INVOUT VCC1 XGND1 ACTINPUT
X2 INVOUT OUT1 VCC1 XGND1 AC109EQ
*Package Inductance
LIN_PIN IN IN1 PIN_IN
LOUT_PIN OUT1 OUT PIN_OUT
LVCC VCC VCC1 PIN_VCC
LGND XGND1 XGND PIN_GND
.ENDS
$ TLINE MODEL - 2 SIGNAL CONDUCTORS WITH GND
$ PLANE
.MODEL USTRIP U LEVEL=3 ELEV=1 PLEV=1
+ TH1=1.3mil HT1=10mil TS=32mil KD1=4.5 DLEV=0 WD1=8mil
+ XW=-2mil KD2=4.5 NL=2 SP12=5mil
$ ANALYSIS / PRINTS
.TRAN .1NS 100NS
.PROBE V(STIM1) V(STIM2)
.PROBE V(TLOUT1) V(TLOUT2) V(TLOUT3) V(TLOUT4)
.END
```

Simulating Circuits with Xilinx FPGAs

Synopsys and Xilinx maintain a library of HSPICE device models and transistor-level subcircuits for the Xilinx 3000 and 4000 series Field Programmable Gate Arrays (FPGAs). These subcircuits model the input and output buffer. See Signal Integrity Examples.

The following simulations use the Xilinx input/output buffer (xil_iob.inc) to simulate:

- Ground bounce, as a function of package, temperature, part speed, and technology
- Coupled noise, both on-chip and chip-to-chip
- Full transmission line effects at the package level and the printed circuit board level
- Peak current and instantaneous power consumption for power supply bus considerations and chip capacitor placement

The following sections discuss these topics:

- Syntax for IOB (xil_iob) and IOB4 (xil_iob4)
- Ground-Bounce Simulation
- Coupled Line Noise

Syntax for IOB (xil_iob) and IOB4 (xil_iob4)

Example of call for 1.2u PART:

```
X1 I O PAD TS FAST PPUB TTL VDD GND XIL_IOB
+ XIL_SIG=0 XIL_DTEMP=0 XIL_SHRINK=0
```

Example of call for 1.08u part:

```
X1 I O PAD TS FAST PPUB TTL VDD GND XIL_IOB
+ XIL_SIG=0 XIL_DTEMP=0 XIL_SHRINK=1
```

Nodes	Description
I (IOB only)	output of the TTL/CMOS receiver
O (IOB only)	input pad driver stage

Nodes	Description
I1 (IOB4 only)	input data 1
I2 (IOB4 only)	input data 2
DRIV_IN (IOB4 only)	
PAD	bonding pad connection
TS	three-state control input (5 V disables)
FAST	slew rate control (5 V fast)
PPUB (IOB only)	pad pull-up enable (0 V enables)
PUP (IOB4 only)	pad pull-up enable (0 V enables)
PDOWN (IOB4 only)	pad pull-up enable (5 V enables)
TTL (IOB only)	CMOS/TTL input threshold (5 V selects TTL)
VDD	5-volt supply
GND	ground
XIL_SIG	model distribution: (default 0) -3==> slow 0==> typical +3==> fast
XIL_DTEMP	Buffer temperature difference from ambient. The default = 0 degrees if ambient is 25 degrees, and if the buffer is 10 degrees hotter than `XIL_DTEMP=10`.
XIL_SHRINK	Old or new part; (default is new): 0==>old 1==>new

All grounds and supplies are common to the external nodes for the ground and VDD. You can redefine grounds to add package models.

Ground-Bounce Simulation

Ground-bounce simulation duplicates the Xilinx internal measurements methods. It simultaneously toggles 8 to 32 outputs. The simulation loads each output with a 56 pF capacitance. Simulation also uses an 84-pin package mode and an output buffer held at chip ground to measure the internal ground bounce.

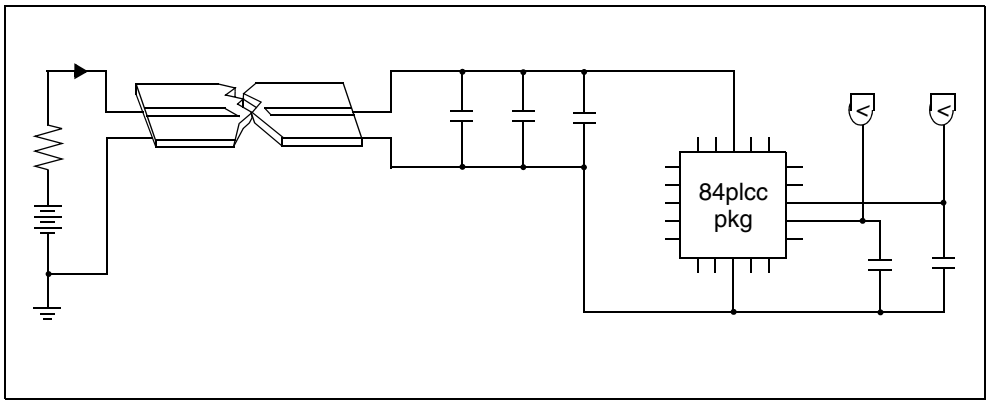


Figure 12 Ground Bounce Simulation

HSPICE or HSPICE RF adjusts the simulation model for the oscilloscope recordings so you can use it for the two-bond wire ground. For example, the following netlist simulates ground bounce:

```
qabounce.sp test of xilinx i/o buffers
.OPTION SEARCH='$installdir/parts/xilinx'
.op
.option post list
.tran 1ns 50ns sweep gates 8 32 4
.measure bounce max v(out1x)
*.tran .1ns 7ns
.param gates=8
.print v(out1x) v(out8x) i(vdd) power
$.param xil_dtemp=-65 $ -40 degrees c
$ (65 degrees from +25 degrees)
vdd vdd gnd 5.25
vgnd return gnd 0
upower1 vdd return iob1vdd iob1gnd pcb_power
+ L=600mil
* local power supply capacitors
xc1a iob1vdd iob1gnd cap_mod cval=.1u
xc1b iob1vdd iob1gnd cap_mod cval=.1u
xc1c iob1vdd iob1gnd cap_mod cval=1u
xgnd_b iob1vdd iob1gnd out8x out1x xil_gnd_test
xcout8x out8x iob1gnd cap_mod m=gates
xcout1x out1x iob1gnd cap_mod m=1
.model pcb_power u LEVEL=3 elev=1 plev=1 nl=1 llev=1
+ th=1.3mil ht=10mil kd=4.5 dlev=1 wd=500mil xw=-2mil
.macro cap_mod node1 node2 cval=56p
Lr1 node1 node1x L=2nh R=0.05
cap node1x node2x c=cval
Lr2 node2x node2 L=2nh R=0.05
.eom
.macro xil_gnd_test vdd gnd outx outref
+ gates=8
* example of 8 iobuffers simultaneously switching
* through approx. 4nh lead inductance
* 1 iob is active low for ground bounce measurements
vout drive chipgnd pwl 0ns 5v, 10ns 5v, 10.5ns 0v,
$+ 20ns 0v, 20.5ns 5v, 40ns 5v R
x8 I8 drive PAD8x TS FAST PPUB TTL chipvdd chipgnd
+ xil_iob xil_sig=0 xil_dtemp=0 xil_shrink=1 M=gates
x1 I1 gnd PAD1x TS FAST PPUB TTL chipvdd chipgnd
+ xil_iob xil_sig=0 xil_dtemp=0 xil_shrink=1 m=1
*Control Settings
rts ts chipgnd 1
rfast fast chipvdd 1
rppub ppub chipgnd 1
rttl ttl chipvdd 1
* pad model plcc84 rough estimates
lvdd vdd chipvdd L=3.0nh r=.02
lgnd gnd chipgnd L=3.0nh r=.02
```

```
lout8x outx pad8x L='5n/gates' r='0.05/gates'
lout1x outref pad1x L=5nh r=0.05
c_vdd_gnd chipvdd chipgnd 100n
.eom
.end
```

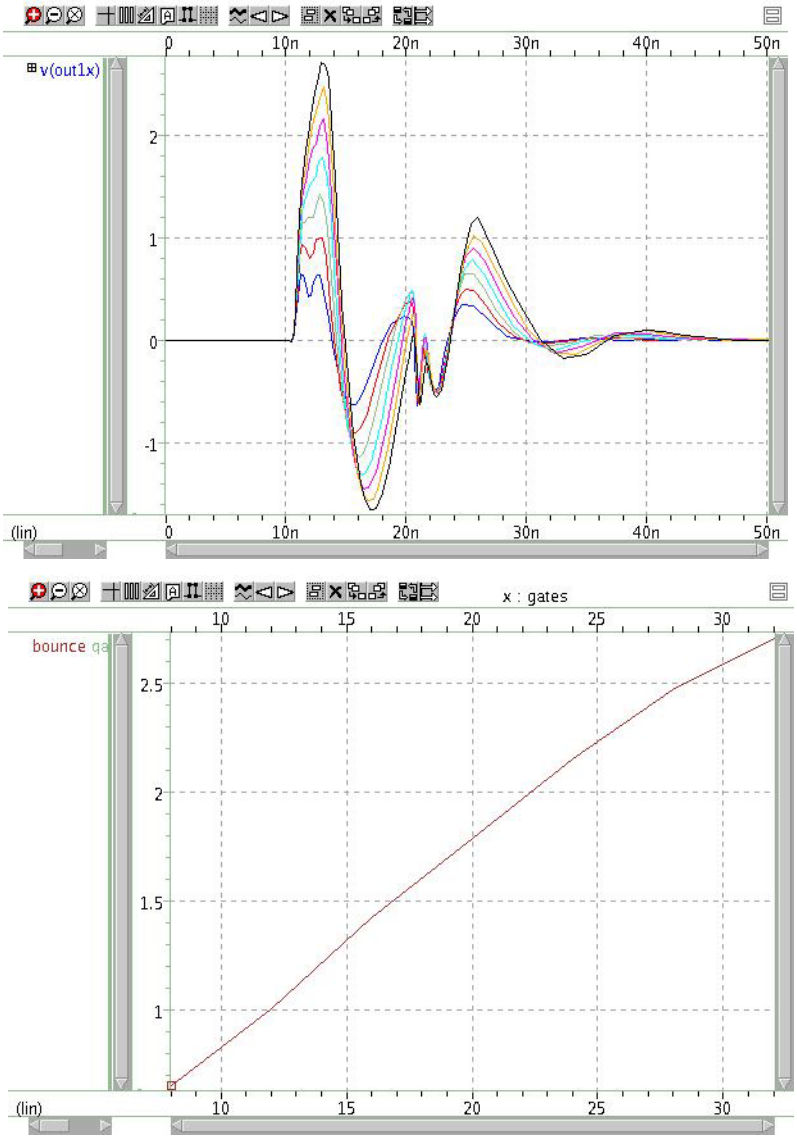


Figure 13 Results of Ground Bounce Simulation

Coupled Line Noise

This example uses coupled noise to separate IOB parts. The output of one part drives the input of the other part through 0.6 inches of PCB. This example also monitors an adjacent quiet line.

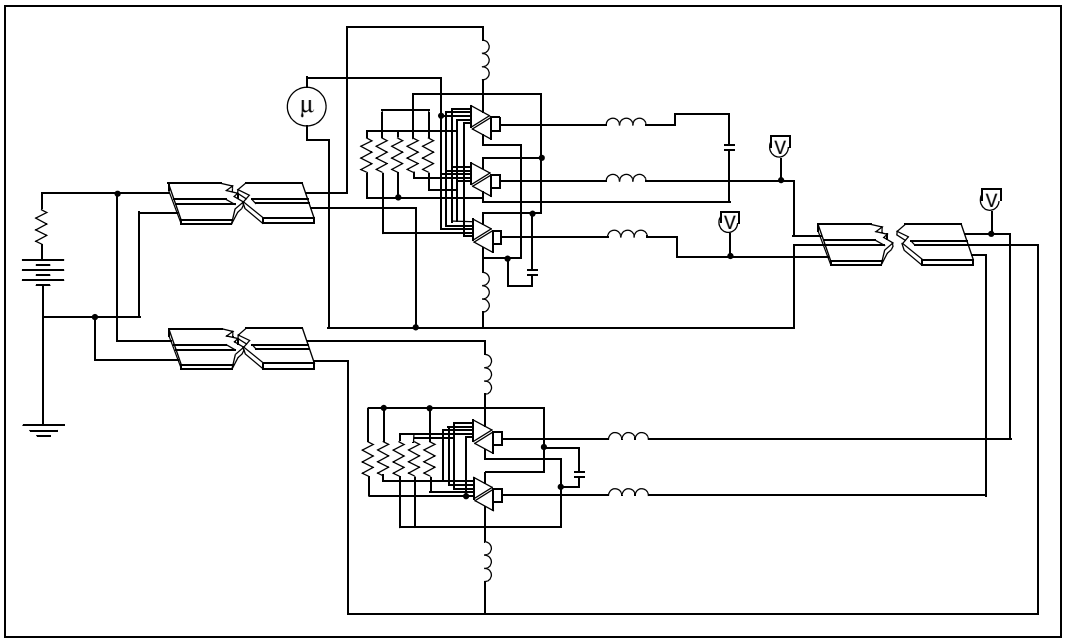


Figure 14 Coupled Noise Simulation

Here is an example netlist for coupled noise simulation:

```
Input File, for qa8.sp test of xilinx 0.8u i/o buffers
.OPTION SEARCH='$installdir/parts/xilinx'
.op
.option nomod post=2
*.tran .1ns 5ns sweep xil_sig -3 3 3
.tran .1ns 15ns
.print v(out1x) v(out3x) i(vdd) v(irec)
vdd vdd gnd 5
vgnd return gnd 0
upower1 vdd return iob1vdd iob1gnd pcb_power L=600mil
upower2 vdd return iob2vdd iob2gnd pcb_power L=600mil
x4io iob1vdd iob1gnd out3x out1x outrec irec xil_iob4
cout3x out3x iob1gnd 9pf
u1x out1x outrec iob1gnd i_o_in i_o_out iob2gnd pcb_top
+ L=2000mil
xrec iob2vdd iob2gnd i_o_in i_o_out xil_rec
.ic i_o_out 0v
.model pcb_top u LEVEL=3 elev=1 plev=1 nl=2 llev=1
+ th=1.3mil ht=10mil sp=5mil kd=4.5 dlev=1 wd=8mil xw=-2mil
.model pcb_power u LEVEL=3 elev=1 plev=1 nl=1 llev=1
+ th=1.3mil ht=10mil kd=4.5 dlev=1 wd=500mil xw=-2mil
.macro xil_rec vdd gnd tri1 tri2
* example of 2 iobuffers in tristate
xtri1 Irec O pad_tri1 TSrec FAST PPUB TTL
+ chipvdd chipgnd xil_iob xil_sig=0 xil_dtemp=0 xil_shrink=1
+ m=1
xtri2 Irec O pad_tri2 TSrec FAST PPUB TTL
+ chipvdd chipgnd xil_iob xil_sig=0 xil_dtemp=0
+ xil_shrink=1 m=1
*Control Setting
rin_output O chipgnd 1
rtsrec tsrec chipvdd 1
rfast fast chipvdd 1
rppub ppub chipgnd 1
rttl ttl chipvdd 1
* pad model plcc84 rough estimates
lvdd vdd chipvdd L=1nh r=.01
lgnd gnd chipgnd L=1nh r=.01
ltri1 tri1 pad_tri1 L=3nh r=0.01
ltri2 tri2 pad_tri2 L=3nh r=.01
c_vdd_gnd chipvdd chipgnd 100n
.eom
.macro xil_iob4 vdd gnd out3x out1x outrec Irec
* example of 4 iobuffers simultaneously switching
* through approx. 3nh lead inductance
* 1 iob is a receiver (tristate)
vout O chipgnd pwl 0ns 0v, 1ns 0v, 1.25ns 4v, 7ns 4v,
+ 7.25ns 0v, 12ns 0v R
```

```
x3 I3 O PAD3x TS FAST PPUB TTL chipvdd chipgnd xil_iob
+ xil_sig=0 xil_dtemp=0 xil_shrink=1 m=3
x1 I1 O PAD1x TS FAST PPUB TTL chipvdd chipgnd xil_iob
+ xil_sig=0 xil_dtemp=0 xil_shrink=1 m=1
xrec Irec O PADrec TSrec FAST PPUB TTL chipvdd chipgnd xil_iob
+ xil_sig=0 xil_dtemp=0 xil_shrink=1 m=1
* control settings
rts ts chipgnd 1
rtsrec tsrec chipvdd 1
rfast fast chipvdd 1
rppub ppub chipgnd 1
rttl ttl chipvdd 1
* pad model plcc84 rough estimates
lvdd vdd chipvdd L=1nh r=.01
lgnd gnd chipgnd L=1nh r=.01
lout3x out3x pad3x L=1nh r=.0033
lout1x out1x pad1x L=4nh r=0.01
loutrec outrec padrec L=4nh r=.01
c_vdd_gnd chipvdd chipgnd 100n
.eom
.end
```

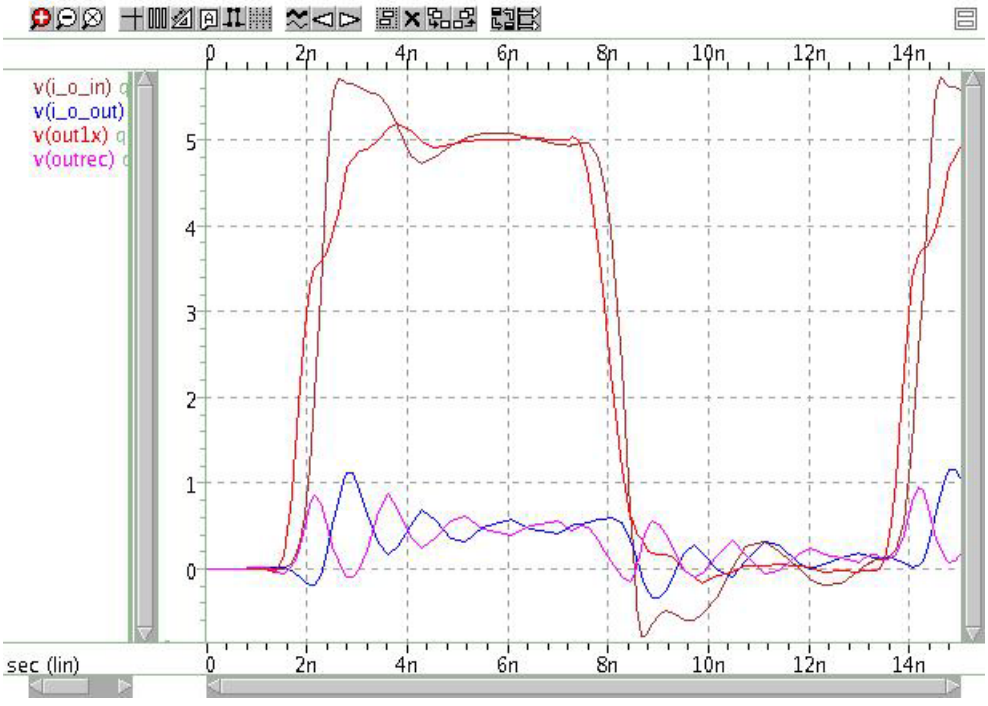


Figure 15 Results of Coupled Noise Simulation

Example Syntax

To copy and paste proven syntax use the demonstration files shipped with your installation of HSPICE (see Listing of Demonstration Input Files). Attempting to copy and paste from the manual or help documentation may present unexpected results, as text used in formatting may include hidden characters, white space, etc. for visual clarity.

HSPICE ships hundreds of examples for your use. Find signal integrity-related demo files under Signal Integrity Examples, S-parameter Examples, IBIS Examples, Transmission (W-element) Line Examples, and Transmission Lines Examples.

For information on statistical eye diagram analysis see Statistical Eye Analysis in the *HSPICE User Guide: Simulation and Analysis*.

2

S-parameter Modeling Using the S-element

Describes S-parameter and SP modeling as well as other topics related to the S-element.

HSPICE numerous examples for your use. Find paths to S-parameter-related demo files under S-parameter Examples.

You can use the S-element to describe a multi-terminal network circuit analyses within most HSPICE and RF analyses. (The exception is Shooting-Newton SN analysis in HSPICE RF.) For more information about using the S-element (S-parameter) for mixed-mode analysis, see S-element (Generic Multiport) in the *HSPICE User Guide: Simulation and Analysis*.

The following sections discuss these topics:

- S-parameter Model
- Mixed-Mode S-parameters
- Using the Scattering Parameter Element
- S-element Syntax
- S Model Syntax
- Pre-Conditioning S-parameters
- Group Delay Handler in Time Domain Analysis
- Accelerating S-element Time Domain Performance with Recursive Convolution
- S-element Data File Model Examples
- Multiport Noise Model for Passive Systems
- S-element Noise Model
- Small-Signal Parameter Data Frequency Table Model (SP Model)

- S Model Data Smoothing
- Predicting an Initial Value for FMAX in S-element Models
- De-embedding S-parameters
- S-parameter Standalone Manipulation Utility (SPutil)
- References

S-parameter Model

You can use small-signal parameters at the network terminals to characterize linear or non-linear networks that have sufficiently small signals. After you set the parameters, you can simulate the block in any external circuit. S-parameters are widely used to characterize a linear network especially among designers of high-frequency circuits.

S-parameters ($\mathbf{S}$) in multiport networks are defined as:

Equation 1 $$\mathbf{b} = \mathbf{S} \cdot \mathbf{a}$$

In the preceding equation, a is an incident wave vector, and b is a reflected wave vector, defined as follows:

Equation 2 $$\mathbf{a} = \mathbf{Y}_r^{1/2} \cdot \mathbf{v}_f = \mathbf{Z}_r^{1/2} \cdot \mathbf{i}_f$$

Equation 3 $$\mathbf{b} = \mathbf{Y}_r^{1/2} \cdot \mathbf{v}_b = \mathbf{Z}_r^{1/2} \cdot \mathbf{i}_b$$

The preceding equations use the following definitions:

- $\mathbf{v}_f$ is the forward voltage vector.
- $\mathbf{v}_b$ is the backward voltage vector.
- $\mathbf{i}_f$ is the forward current vector.
- $\mathbf{i}_b$ is the backward current vector.
- $\mathbf{Z}_r$ is the characteristic impedance matrix of the reference system.

- $\mathbf{Y}_r$ is the characteristic admittance matrix.
- $\mathbf{Z}_r$ and $\mathbf{Y}_r$ satisfy the relationship $\mathbf{Y}_r = \mathbf{Z}_r^{-1}$

The S-parameters are frequency-dependent. When all ports are terminated with impedance matching without a voltage/current source, the forward wave becomes zero. This is because there is no reflection if the ports have no voltage/current source.

Notifications and Limitations

Because the S-element can support two types of noise models, the priority is:

- For multiport (N≠2) S-elements, only passive noise models are considered in noise analysis. If `NOISE=0`, the system is considered as noiseless.
- For two-port S-elements, if two-port noise parameters are provided in a Touchstone file, the noise model is generated from those two-port noise parameters. If two-port noise parameters are not provided and `NOISE=1`, then a passive noise model is triggered. Otherwise, the system is considered as noiseless.
- HSPICE does not support bias-dependent S-parameters.

Mixed-Mode S-parameters

Mixed-mode refers to a combination of Differential and Common mode characteristics in HSPICE linear network (`.LIN`) analysis by using the S-element. It is useful to understand mixed-mode S-parameters since they present a “big picture” big of the wiring channels at first sight by providing a view of inherent behaviors of differential and common mode signal propagation characteristics.

HSPICE accepts both conventional single-ended S-parameters and mixed-mode S-parameters. Internally in HSPICE, since the mixed-mode S-parameters are first converted to the single-ended S-parameters to fit with ground-referenced nodal analyses, there will be no difference in simulation results between single-ended S-parameters and the equivalent mixed-mode representations of them.

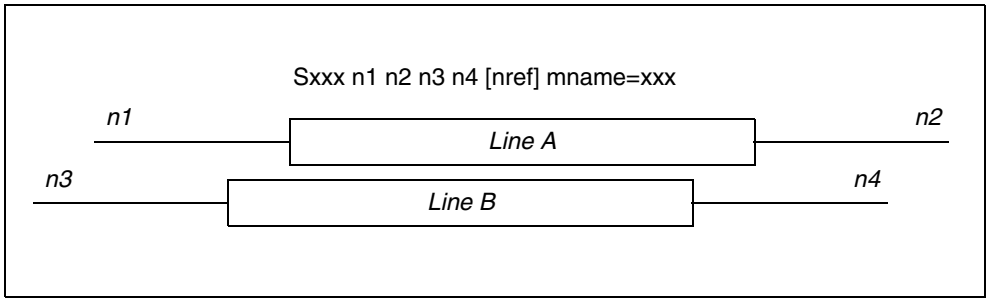


Figure 16 Node Indexing Convention of the Ground Referenced (Single Ended) S-parameter

- Nodes 1 and 3 are the ports for one end of the transmission-line pair.
- Nodes 2 and 4 are the ports for the opposite end of the transmission-line pair.

The following sections discuss these topics:

- Relating Voltage and Current Waves to Nodal Waves
- Characterizing Differential Data Transfer Systems
- Deriving a Simpler Set of Voltage and Current Pairs
- Using the Mixed-Mode S-parameters (S-element)

Relating Voltage and Current Waves to Nodal Waves

The following figure and set of equations include common and differential mode voltage and current waves, relating them to nodal waves. Although you can apply mixed-mode data propagation to an arbitrary number of pairs of transmission lines, a single pair model is used here.

Figure 17 on page 39 shows a schematic of symmetric coupled pair transmission lines commonly used for the differential data transfer system.

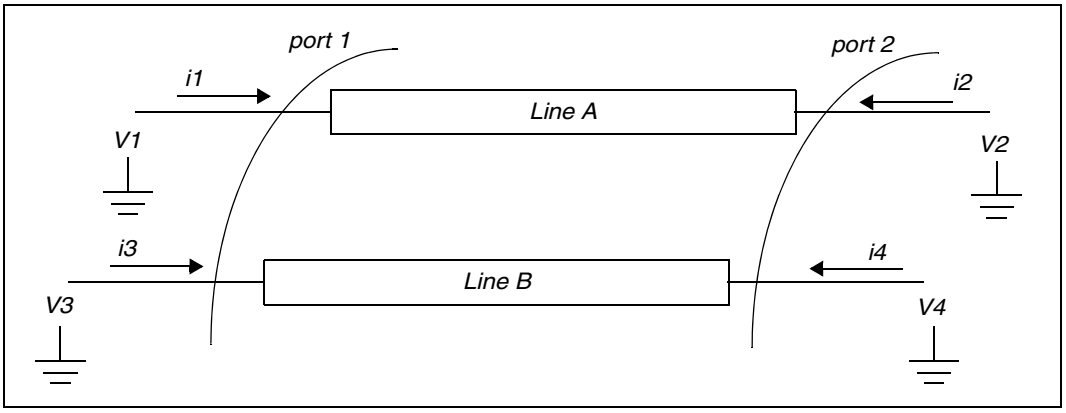


Figure 17 Schematic of Symmetric Coupled-Pair Transmission Line

Solving the telegrapher's equation, you can represent nodal voltage and current waves of the data transfer system as:

Equation 4
$$v_1 = A_1 e^{-\gamma_e x} + A_2 e^{\gamma_e x} + A_3 e^{-\gamma_o x} + A_4 e^{\gamma_o x}$$

Equation 5
$$v_3 = A_1 e^{-\gamma_e x} + A_2 e^{\gamma_e x} - A_3 e^{-\gamma_o x} - A_4 e^{\gamma_o x}$$

Equation 6
$$i_1 = \frac{A_1}{Z_e} e^{-\gamma_e x} - \frac{A_2}{Z_e} e^{\gamma_e x} + \frac{A_3}{Z_o} e^{-\gamma_o x} - \frac{A_4}{Z_o} e^{\gamma_o x}$$

Equation 7
$$i_3 = \frac{A_1}{Z_e} e^{-\gamma_e x} - \frac{A_2}{Z_e} e^{\gamma_e x} + -\frac{A_3}{Z_o} e^{-\gamma_o x} + \frac{A_4}{Z_o} e^{\gamma_o x}$$

Where:

- Υ_e is the propagation constant for even mode waves.
- Υ_o is the propagation constant for odd mode waves.
- Z_e is the characteristic impedance for even mode waves.
- Z_o is the characteristic impedance for odd mode waves.

- A_1 and A_3 represent phasor coefficients for the forward propagating modes.
- A_2 and A_4 represent phasor coefficients for the backward propagating modes.

Each voltage and current pair at each node represents a single propagating signal wave referenced to the ground potential. This type of expression is called nodal wave representation.

Characterizing Differential Data Transfer Systems

The following equations use differential and common mode waves to characterize differential data transfer systems. The difference of the nodal wave defines the voltage and current of the differential wave:

Equation 8 $$v_{dm} \equiv v_1 - v_3$$

Equation 9 $$i_{dm} \equiv \frac{1}{2}(i_1 - i_3)$$

Common mode voltage and current are defined as:

Equation 10 $$v_{cm} \equiv \frac{1}{2}(v_1 + v_3)$$

Equation 11 $$i_{cm} \equiv i_1 + i_3$$

Deriving a Simpler Set of Voltage and Current Pairs

In the following example, substituting equations 2 and 3 into equation 1 derives a simpler set of voltage and current pairs:

Equation 12 $$v_{dm} = 2(A_3 e^{-\gamma_o x} + A_4 e^{-\gamma_o x})$$

Equation 13 $$v_{cm} = A_1 e^{-\gamma_e x} + A_2 e^{\gamma_e x}$$

Equation 14
$$i_{dm} = \frac{A_3}{Z_o}e^{-\gamma_o x} - \frac{A_4}{Z_o}e^{\gamma_o x}$$

Equation 15
$$i_{cm} = 2\left(\frac{A_1}{Z_e}e^{-\gamma_e x} - \frac{A_2}{Z_e}e^{\gamma_e x}\right)$$

You can also relate characteristic impedances of each mode to the even and odd mode characteristic impedances:

$$Z_{dm} \equiv 2Z_o \text{ and } Z_{cm} \equiv \frac{Z_e}{2}$$

Having defined a generalized parameter power wave in this example, you can now define differential normalized waves at port 1 and port 2:

$$a_{dm1} \equiv \left.\frac{v_{dm} + Z_{dm}i_{dm}}{2\sqrt{Z_{dm}}}\right|_{x=0} \text{ and } a_{dm2} \equiv \left.\frac{v_{dm} + Z_{dm}i_{dm}}{2\sqrt{Z_{dm}}}\right|_{x=L}$$

$$b_{dm1} \equiv \left.\frac{v_{dm} - Z_{dm}i_{dm}}{2\sqrt{Z_{dm}}}\right|_{x=0} \text{ and } b_{dm2} \equiv \left.\frac{v_{dm} - Z_{dm}i_{dm}}{2\sqrt{Z_{dm}}}\right|_{x=L}$$

Similarly, you can define common mode normalized waves as:

$$a_{cm1} \equiv \left.\frac{v_{cm} + Z_{cm}i_{cm}}{2\sqrt{Z_{cm}}}\right|_{x=0} \text{ and } a_{cm2} \equiv \left.\frac{v_{cm} + Z_{cm}i_{cm}}{2\sqrt{Z_{cm}}}\right|_{x=L}$$

$$b_{cm1} \equiv \left.\frac{v_{cm} - Z_{cm}i_{cm}}{2\sqrt{Z_{cm}}}\right|_{x=0} \text{ and } b_{cm2} \equiv \left.\frac{v_{cm} - Z_{cm}i_{cm}}{2\sqrt{Z_{cm}}}\right|_{x=L}$$

You can then specify S-parameters for mixed-mode waves as ratios of these waves:

Equation 16
$$\begin{bmatrix} b_{dm1} \\ b_{dm2} \\ b_{cm1} \\ b_{cm2} \end{bmatrix} = S_{mixed}\begin{bmatrix} a_{dm1} \\ a_{dm2} \\ a_{cm1} \\ a_{cm2} \end{bmatrix}, \quad S_{mixed} = \begin{bmatrix} S_{dd} & S_{dc} \\ S_{cd} & S_{cc} \end{bmatrix}$$

Where,

- S_{dd} is the differential-mode S-parameter
- S_{cc} is the common-mode S-parameter
- S_{cd} and S_{dc} represent the mode-conversion or cross-mode S-parameters

Based on these definitions, you can linearly transform nodal wave (standard) S-parameters and mixed mode S-parameters: $\mathbf{M} \cdot \mathrm{S}_{\mathrm{standard}} \cdot \mathbf{M}^{-1} = \mathrm{S}_{\mathrm{mixed}}$

The M transformation matrix is: $\mathbf{M} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$

Using the Mixed-Mode S-parameters (S-element)

The S-element can recognize and parse the mixed-mode S-parameters when the `mixedmode=1` keyword is set. Any keywords besides `mixedmode` and `datatype` remain the same. Use the following syntax for a mixed-mode S-parameter.

```
Sxxx p1+ [p1-] p2+ [p2-] p3+ [p3-]...[n_ref] mname=Smodel
.MODEL Smodel S ...
[+ mixedmode=[0 1]]
[+ datatype=XiYjZk...]
```

Parameter	Description
pn+, pn-	Positive and negative terminals of the port n, respectively. The port numbers must be in increasing order corresponding to the S matrices notation. ▪ If the port is in mixed mode (balanced) one, both positive and negative terminal names are required in series ▪ If the port is single-ended, only one terminal name is required.
mixedmode	When mixedmode=1, the t the element knows that the S-parameters are defined in mixed mode. The default is 0 (standardmode)

Parameter	Description
datatype	A string that determines the order of indices of the incident or reflected vectors (a and b) in Equation 8. The string must be an array of pairs that consists of a letter and a number (for example, Xn), where X= ■ D or d to indicate differential term ■ C or c to indicate common term ■ S, s, G or g to indicate single (grounded) term and n = port number.

The definition `datatype = D1D2C1C2` is the default for a 2-balanced port network and specifies the nodal relationship of the following equation:

Equation 17 $\mathbf{a}_{\text{standard}} = [a_{1+}\ a_{1-}\ a_{2+}\ a_{2-}\ ] \Leftrightarrow \mathbf{a}_{\text{mixed}} = [a_{d1} a_{d2} a_{c1} a_{c2}]^T$

Where:

- a_{1+} is the incident wave goes into positive terminal of the port 1
- a_{1-} is the incident wave goes into negative terminal of the port 1
- a_{2+} is the incident wave goes into positive terminal of the port 2
- a_{2-} is the incident wave goes into negative terminal of the port 2

You can also derive the nodal relationship of the reflection wave in the same way. Nodes are assigned from the given s-matrices to the S-element in the order of $\mathbf{a}_{standard}$. For example, incident and reflected waves at the positive terminal of the $1 a_{1+}, \quad , b_{1+}$ appear at the first node of the S-element.

The definition `datatype=D1C1S2` specifies the nodal relationship of the following equation:

Equation 18 $\mathbf{a}_{\text{standard}} = [a_{1+}\ a_{1-}\ a_2]^T \Leftrightarrow \mathbf{a}_{\text{mixed}} = [a_{d1} a_{c1} a_{s2}]^T$

The default is `datatype=D1D2...DnC1C2...Cn`, which is available for systems with mixed-mode (balanced) ports only.

Mixed-Mode S-parameter Netlist Examples

Example 1: Differential Transmission Line Pair

You can find an example netlist for a differential transmission line pair in the following directory:
$*installdir*/demo/hspice/sparam/mixedmode_s.sp

Example 2: Differential Amplifier

You can find an example netlist for a differential amplifier in the following directory:
$*installdir*/demo/hspice/sparam/diffamp_s.sp

Using the Scattering Parameter Element

The S- (scattering) element gives you a convenient way to describe a multi-terminal network. You can use the S-element in conjunction with the generic frequency-domain model (`.MODEL SP`), or data files that describe frequency-varying behavior of a network, and provide discrete frequency-dependent data such as a Touchstone file and a Common Instrumentation Transfer and Interchange (CITI) file. See the HSPICE User Guide: Simulation and Analysis for more information.

In particular, the S-parameter in the S-element represents the generalized scattering parameter (S) for a multi-terminal network.

The S-parameter and the Y-parameter satisfy the following relationship:

Equation 19
$$\mathbf{Y} = \mathbf{Y}_{rs}(\mathbf{I}-\mathbf{S})(\mathbf{I}+\mathbf{S})^{-1}\mathbf{Y}_{rs}$$

where $\mathbf{Y}_r$ is the characteristic admittance matrix of the reference system. The following formula relates $\mathbf{Y}_r$ to the $\mathbf{Z}_r$ characteristic impedance matrix:

Equation 20
$$\mathbf{Y}_r = \mathbf{Z}_{r'}^{-1}\mathbf{Y}_{rs}\mathbf{Y}_{rs} = \mathbf{Y}_{r'}\mathbf{Z}_{rs}\mathbf{Z}_{rs} = \mathbf{Z}_r$$

Similarly, you can convert the Y-parameter to the S-parameter as follows:

Equation 21
$$\mathbf{S} = (\mathbf{I}+\mathbf{Z}_{rs}\mathbf{Y}\mathbf{Z}_{rs})^{(-1)}(\mathbf{I}-\mathbf{Z}_{rs}\mathbf{Y}\mathbf{Z}_{rs})$$

S-element Syntax

Use the following S-element syntax to show the connections within a circuit:

```
Sxxx nd1 nd2 ... ndN ndRef
+ [MNAME=Smodel_name]
+ [TYPE=s|y] [Z0=value|vector_value]
+ [FBASE = base_frequency] [FMAX=maximum_frequency]
+ [INTERPOLATION=STEP|LINEAR|SPLINE|HYBRID]
+ [INTDATTYP=RI|MA|DBA]
+ [HIGHPASS=1|2|3|4] [LOWPASS=0|1|2|3]
+ [DELAYHANDLE=1|0|ON|OFF] [DELAYFREQ=val]
+ [MIXEDMODE=0|1] [DATATYPE=data_string]
+ [NOISE=[1|0]] [NoiPassiveChk=1|0] [DTEMP=val]
+ [RATIONAL_FUNC=[0|1]] [RATIONAL_FUNC_REUSE=[0|1|2]]
+ [PASSIVE=0|1] [PASSIVE_TOL=val] [ENFORCE_PASSIVE=0|1]
+ [STAMP=S|Y|YSTS|SSTS|DEEMBED] [M=int]
+ [PRECFAC=val] [FQMODEL=sp_model_name]
```

Parameter	Description
nd1 nd2...ndN	Nodes of an S-element (see Figure 18 on page 50) and Node Example. Three kinds of definitions are present: ▪ With no reference node ndRef, the default reference node is GND. Each node ndi (i=1~N) and GND construct one of the N ports of the S-element. ▪ With one reference node, ndRef is defined. Each node ndi (i=1~N) and the ndRef construct one of the N ports of the S-element. ▪ With an N reference node, each port has its own reference node. You can write the node definition in a clearer way as: nd1+ nd1- nd2+ nd2- ... ndN+ ndN- Each pair of the nodes (ndi+ and ndi-, i=1~N) constructs one of the N ports of the S-element.
ndRef	Reference node
MNAME	Name of the S model; Note that string parameters are supported in calling an MNAME.
TYPE	Parameter type: ▪ S: (scattering) (default) ▪ Y: (admittance)

Parameter	Description
Z0 (or Zo)	Characteristic impedance value for the reference line (frequency-independent). For multiple terminals (N>1), HSPICE or HSPICE RF assumes that the characteristic impedance matrix of the reference lines is diagonal, and that you set diagonal values to Z0. Default=50 Ω.
FBASE	Base frequency to use for transient analysis. This value becomes the base frequency point for Inverse Fast Fourier Transformation (IFFT). ▪ If you do not set this value, the base frequency is a reciprocal value of the transient period. ▪ If you set a frequency that is smaller than the reciprocal value of the transient, then transient analysis performs circular convolution, and uses the reciprocal value of FBASE as its base period.
FMAX	Maximum frequency use in transient analysis. Used as the maximum frequency point for Inverse Fast Fourier Transformation (IFFT). See Predicting an Initial Value for FMAX in S-element Models.
INTERPOLATION	The interpolation method: ▪ STEP: piecewise step ▪ SPLINE: b-spline curve fit ▪ LINEAR: piecewise linear (default) ▪ HYBRID: HSPICE combines different interpolation/extrapolation methods, and switches automatically between them to get the best accuracy. If needed, it also does causality correction down to DC. It is most useful for the S-parameters showing local resonances, and provides the proper interpolation and low-frequency extrapolation method for each entry of the S matrix, which shows different behaviors. For best accuracy, low frequency examples should be provided.
INTDATTYP	Data type for the linear interpolation of the complex data. ▪ RI: real-imaginary based interpolation ▪ DBA: dB-angle based interpolation ▪ MA: magnitude-angle based interpolation (default)

Parameter	Description
HIGHPASS	Method to extrapolate higher frequency points. ■ 0: cut off ■ 1: use highest frequency point ■ 2: perform linear extrapolation using the highest 2 points ■ 3: apply the window function to gradually approach the cut-off level (default) ■ 4: Estimates average derivatives of the phase and magnitude from highest 10% of sampling points. Extrapolation is performed using the highest sampling point and these derivatives.
LOWPASS	Method to extrapolate lower frequency points. ■ 0: Cut off. ■ 1: Make use of the S matrix at the magnitude of the lowest given frequency point; Set the magnitude value of each entry as the element of DC matrix. The sign of each value is determined by the real part of the extrapolated value at DC point. (default) ■ 2: Perform linear extrapolation using the magnitude of the lowest two points. ■ 3: Perform rational function approximation based on low end frequency extrapolation.
DELAYHANDLE	DELAYHANDLE in S-element simulation is used to extract a system delay before constructing the system impulse response. This may help to improve transient accuracy when the system does have delay, such as transmission line system. Because S-parameters represent a system which has delay, it is suggested to turn DELAYHANDLE on. When DELAYHANDLE is ON (or 1) the S-element extracts propagation delay to simplify transfer functions, then proceeds to approximation. The extracted delay is handled separately in the time domain. See also, Group Delay Handler in Time Domain Analysis.
DELAYFREQ	Delay frequency for transmission-line type parameters. The default is FMAX. If the DELAYHANDLE is set to OFF, but DELAYFREQ is nonzero, HSPICE still simulates the S-element in delay mode.
MIXEDMODE	Set to `1` if the parameters are represented in the mixed mode.

Parameter	Description
DATATYPE	A string used to determine the order of the indices of the mixed-signal incident or reflected vector. The string must be an array of a letter and a number (*Xn*) where: ▪ X = D to indicate a differential term = C to indicate a common term = S to indicate a single (grounded) term ▪ n = the port number
NOISE	Activates thermal noise. ▪ 1 (default): element generates thermal noise ▪ 0: element is considered noiseless
NoiPassiveChk	Checks S-parameter for passivity in noise analysis (only). ▪ 1 (default): Checks for passivity; if it fails at any frequency, thermal noise is turned off for the specific frequency point. ▪ 0: Disables the passivity checker; thermal noise is always turned on.
DTEMP	Temperature difference between the element and the circuit, expressed in ×C. The default is 0.0. Element temperature is calculated as: T = Element temperature (×K) = 273.15 (×K) + circuit temperature (×C) + DTEMP (×C) Where circuit temperature is specified using either the .TEMP statement, or by sweeping the global TEMP variable in .DC, .AC, or .TRAN statements. When a .TEMP statement or TEMP variable is not used, the circuit temperature is set by .OPTION TNOM, which defaults to 25 ×C unless you use .OPTION SPICE, which raises the default to 27 ×C.
RATIONAL_FUNC	▪ 0: (default) performs the same as conventional S-element. FBASE/FMAX-based linear convolution is performed. ▪ 1: Performs rational function approximation then recursive convolution; also handles non-causal S-parameters.
RATIONAL_FUNC_REUSE	▪ 0: Discard previously extracted rational function data and rerun the rational function approximation. ▪ 1: Reuse rational function data if available. ▪ 2: (default) Reuse rational function data if available and make no change in parameter source file (time stamp), FBASE, FMAX, HIGHPASS, LOWPASS, and passivity enforcement configurations; otherwise rerun the rational function approximation.

Parameter	Description
PASSIVE	Activates the passive checker to help debug passive models. The default is 0 for the S-element where 0=deactivate and 1=activate. Using the tolerance value specified by PASSIVE_TOL keyword, the eigenvalues of matrix (I-S*S'), ev[i], will be checked. If any frequency point violates RE(ev[i]) > -(TOL*0.1), HSPICE issues a warning containing a list of violating frequencies with an “E” flag. Also, the checker verifies potential passivity violations by checking the summation of each S-parameter matrix column. If Sum > (1.0+TOL), HSPICE issues a warning containing a list of those violating frequencies with an “C” flag.
PASSIVE_TOL	Tolerance for eigenvalue checking activated by PASSIVE keyword. Default value is 0.01.
ENFORCE_ PASSIVE	With the ENFORCE_PASSIVE=1 keyword, the S-element checks passivity of all the given frequency sampling points. Once passivity violations are found, the S-element seeks a minimum amount of loss property which restores passivity of all the violated points then adds the loss to all the given frequency points.
STAMP	▪ Y: Conventional admittance based stamp ▪ S: Scattering parameter based stamp (Note 1) ▪ YSTS: Admittance parameter based state space stamp (Note 2) ▪ SSTS: Scattering parameter based state space stamp (Note 2) ▪ DEEMBED: Produces negated stamp to de-embed given a S-parameter block from the adjacent DUT connected in series. (See De-embedding S-parameters in this chapter.) Note 1: Although Y and S stamp types behave mathematically equivalent, when the S type is selected, the S-element activates a procedure to reduce memory consumption by taking matrices’ sparseness into account. Note 2: YSTS and SSTS stamp methods may be activated when RATIONAL_FUNC=1 is used. The state space stamping embeds all the state variables for extracted rational function matrix into the modified nodal analysis (NMA) matrix instead of performing recursive convolution integration. Although this stamping method may incur additional computational cost, since it produces frequency invariant NMA matrix, it enables time domain steady state (so-called .SN in HSPICE RF) analysis to handle frequency-dependent S-parameter blocks.
M	S-element multiplier; replicates element *int* times, in parallel; default is 1. Do not assign a negative value or zero as the M value.

Parameter	Description
PRECFAC	In almost all cases, you do not need to specify a value for this parameter. This parameter specifies the precondition factor keyword used for the precondition process of the S-parameter. A precondition is used to avoid an infinite admittance matrix. The default is 0.75, which is good for most cases. See also, Pre-Conditioning S-parameters.
FQMODEL	Frequency behavior of the parameters. .MODEL statement of sp type, which defines the frequency-dependent matrices array

The nodes of the S-element must come first. If `MNAME` is not declared, you must specify the `FQMODEL`. You can specify all the optional parameters in both the S-element and S model statements, except for `MNAME` argument.

You can enter the optional arguments in any order, and the parameters specified in the element statement have a higher priority.

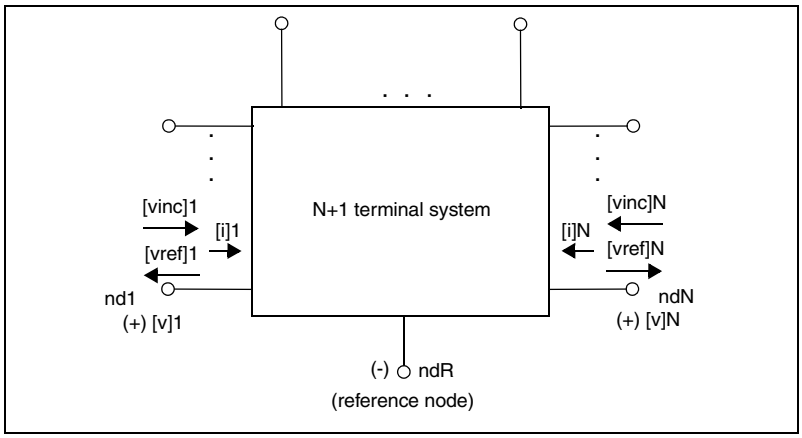


Figure 18 Terminal Node Notation

Procedure to Get Best Accuracy

To achieve the best accuracy, verify these S-parameters settings:

1. Start frequency is close to DC and use best LOWPASS setting.
2. Max frequency is 3x the fastest transition time in the circuit; use the best HIGHPASS setting.
3. Frequency spacing is small enough to capture changes in magnitude and phase.

Then in the netlist, set:

1. FBASE to be equal or smaller than 3 above.
2. FMAX to be the max freq as in 2 above.

These steps should make for an accurate S-parameter simulation. Currently, FBASE is set to 1/TSTOP, so a shorter simulation may be less accurate than a longer one (runlvl has no effect).

For demo files of the S-element usage see S-parameter Examples.

Node Example

The following example illustrates the *nd1 nd2...ndN—no reference*, *single reference*, and multi-reference parameters.

```
**S-parameter example

.opt post
.ac lin 500 1Hz 30MegHz
.tran 0.1ns 10ns

V1 n1 0 ac=1v PULSE 0v 5v 5n 0.5n 0.5n 25n

* no reference
S_no_ref n1 n2 mname=s_model

* single reference
S_one_ref n1 n3 gnd mname=s_model

*multi-reference
S_multi_ref n1 gnd n4 gnd mname=s_model
Rt1 n2 0 50
Rt2 n3 0 50
Rt3 n4 0 50

* 50 ohm resistor
.MODEL s_model S
+ N=2 FQMODEL=SFQMODEL TYPE=S Z0=50 50
.MODEL SFQMODEL SP N=2 SPACING=POI INTERPOLATION=LINEAR
+ MATRIX=NONSYMMETRIC
+ DATA=1
+  1.0  0.333333333 0.0  0.666666667 0.0  0.666666667 0.0
0.333333333 0.0

.end
```

The S-element must have a call to one of the supported S-parameter file formats (Touchstone 1.0/2.0, Citi or .SC#). HSPICE gets the number of ports from the S-parameter file You can also explicitly specify N=n where ‘n’ is the number of ports.

- For n terminals, the S-element assumes no reference node.
- For n+1 terminals, the S-element assumes one reference node.
- For 2n terminals, the S-element assumes signal nodes and n reference nodes. Each pair of nodes is a signal and a reference node.

S Model Syntax

Use the following syntax to describe specific S models:

```
.MODEL Smodel_name S [N=dimension]
+ [TSTONEFILE=filename | CITIFILE=filename |
+ RFMFILE=file_name.rfm | BNPFILE=filename]
+ [TYPE=s|y] [Z0=value | vector_value]
+ [FBASE=base_frequency] [FMAX=maximum_frequency]
+ [INTERPOLATION=STEP|LINEAR|SPLINE|HYBRID]
+ [INTDATTYP=[RI|MA|DBA]]
+ [HIGHPASS=0|1|2|3|4] [LOWPASS=0|1|2|3]
+ [DELAYHANDLE=1|0|ON|OFF] [DELAYFREQ=val]
+ [MIXEDMODE=0|1]
+ [DATATYPE=data_string] [XLINELENGTH=val] [PASSIVE=[0|1]
+ [NoiPassiveChk [1|0]
+ [SMOOTH=val] [SMOOTHPTS=val]
+ [RATIONAL_FUNC=[0|1] [RATIONAL_FUNC_REUSE=0|1|2]
+ [PASSIVE=[0|1] [PASSIVE_TOL=val][ENFORCE_PASSIVE=0|1]
+ [STAMP=S|Y|YSTS|SSTS|DEEMBED]
+ [PRECFAC=val] FQMODEL=sp_model_name
```

Parameter	Description
Smodel_name	Name of the S model.
S	Specifies that the model type is an S model.
N	S model dimension, which is equal to the terminal number of an S-element and excludes the reference node.

Parameter	Description

TSTONEFILE	Specifies the name of a Touchstone file v 1.0/2.0. Data contains frequency-dependent array of matrixes. Touchstone files must follow the *.s#p* file extension rule, where # represents the dimension of the network.
	Note that string parameters are supported for TSTONEFILE.
	Example:
	`.subckt sparam n1 n2 tsfile=str('ss_ts.s2p')`
	`  S1 n1 n2 0 mname=s_model`
	`.model s_model S TSTONEFILE=str(tsfile)`
	`.ends`
	`  x1 A B sparam tsfile=str('ss_ts.s2p')`
	…
	For details, see *Touchstone® File Format Specification* by the EIA/IBIS Open Forum (http://www.eda.org).

CITIFILE	Specifies the name of the CITIfile, which is a data file that contains frequency-dependent data. Note that string parameters are supported for calling a CITIFILE. For details, see *Using Instruments with ADS* by Agilent Technologies (http://www.agilent.com).
RFMFILE	Specifies S-element rational function (RFM) file. See Accelerating S-element Time Domain Performance with Recursive Convolution.
BNPFILE	Specifies Broadband Network Parameter (BNP) file (Sigrity-proprietary). Note that when using the BNPFILE, there is no need for INTERPOLATION or LOW_PASS keywords. For HIGH_PASS, other than HIGH_PASS=4 all options are supported. **Note**: If you get a warning such as `BNP file read failure at f=xx`, this means that the BNP API is returning a read failure. It is likely that this BNP file doesn't cover the f=xx frequency point. Users need to contact Sigrity on this issue. For details on BNP, see http://www.sigrity.com.
TYPE	Parameter type: ▪ S: (scattering) (default). ▪ Y: (admittance).

Parameter	Description
Z0 (or Zo)	Characteristic impedance value of the reference line (frequency-independent). For multi-terminal lines (N>1), HSPICE assumes that the characteristic impedance matrix of the reference lines are diagonal, and their diagonal values are set to Z0. You can also set a vector value for non-uniform diagonal values. Use Z0 to specify more general types of a reference-line system. The default is 50.
FBASE	Base frequency used for transient analysis. HSPICE uses this value as the base frequency point for Fast Inverse Fourier Transformation (IFFT). ■ If FBASE is not set, HSPICE uses a reciprocal of the transient period as the base frequency. ■ If FBASE is set smaller than the reciprocal value of transient period, transient analysis performs circular convolution by using the reciprocal value of FBASE as a base period.
FMAX	Maximum frequency for transient analysis. Used as the maximum frequency point for Inverse Fast Fourier Transform (IFFT). See Predicting an Initial Value for FMAX in S-element Models.
INTERPOLATION	The interpolation method: ■ STEP: piecewise step ■ SPLINE: b-spline curve fit ■ LINEAR: piecewise linear (default) ■ HYBRID: HSPICE combines different interpolation/extrapolation methods, and switches automatically between them to get the best accuracy. If needed, it also does causality correction down to DC. It is most useful for the S-parameters showing local resonances, and provides the proper interpolation and low-frequency extrapolation method for each entry of the S matrix, which shows different behaviors. For best accuracy, low frequency examples should be provided.
INTDATTYP	Data type for the linear interpolation of the complex data. ■ RI: real-imaginary based interpolation. ■ DBA: dB-angle based interpolation. ■ MA: magnitude-angle based interpolation (default).

Parameter	Description
HIGHPASS	Specifies high-frequency extrapolation: ■ 0: Use zero in Y dimension (open circuit). ■ 1: Use highest frequency. ■ 2: Use linear extrapolation with the highest two points. ■ 3: Apply window function (default). ■ 4: Estimates average derivatives of the phase and magnitude from highest 10% of sampling points. Extrapolation is performed using the highest sampling point and these derivatives. This option overrides EXTRAPOLATION in .MODEL SP.
LOWPASS	Method to extrapolate lower frequency points. ■ 0: Cut off. ■ 1: Make use of the S matrix at the magnitude of the lowest given frequency point; Set the magnitude value of each entry as the element of DC matrix. The sign of each value is determined by the real part of the extrapolated value at DC point. (default) ■ 2: Perform linear extrapolation using the magnitude of the lowest two points. ■ 3: Perform rational function approximation based on low end frequency extrapolation. The LOWPASS option overrides EXTRAPOLATION in .MODEL SP.
DELAYHANDLE	DELAYHANDLE extracts a system delay before constructing the system impulse response. This may help to improve transient accuracy when the system does have delay, such as transmission line system. Because S-parameters represent a system which has delay, it is suggested to turn DELAYHANDLE on. When DELAYHANDLE is ON (or 1) the S-element extracts propagation delay to simplify transfer functions, then proceeds to approximation. The extracted delay is handled separately in the time domain. You must set the delay handler, if the delay of the model is longer than the base period specified in the FBASE parameter. If you set DELAYHANDLE=OFF but DELAYFQ is not zero, HSPICE simulates the S-element in delay mode. See also, Group Delay Handler in Time Domain Analysis.
DELAYFREQ	Delay frequency for transmission-line type parameters. The default is FMAX. If the DELAYHANDLE is set to OFF, but DELAYFREQ is nonzero, HSPICE still simulates the S-element in delay mode.
MIXEDMODE	Set to 1 if the parameters are represented in the mixed mode.

Parameter	Description
DATATYPE	A string used to determine the order of the indices of the mixed-signal incident or reflected vector. The string must be an array of a letter and a number (*Xn*) where: ▪ X = D to indicate a differential term = C to indicate a common term = S to indicate a single (grounded) term ▪ n = the port number
XLINELENGTH	The line length of the transmission line system where the S-parameters are extracted. This keyword is required only when the S Model is used in a W-element.
NoiPassiveChk	Checks S-parameter for passivity in noise analysis (only). ▪ 1 (default): Checks for passivity; if it fails at any frequency, thermal noise is turned off for the specific frequency point. ▪ 0: Disables the passivity checker; thermal noise is always turned on.
SMOOTH	An integer value to choose one of following methods ▪ 0: no smoothing (default). ▪ 1: mean. ▪ 2: median. ▪ 3: 2nd order polynomial fit. ▪ 4: 4th order polynomial fit. See S Model Data Smoothing on page 82.
SMOOTHPTS	An integer value to specify width of the smoothing window on each side of the target point. In total, 2*x +1 point is taken at each point calculation.
RATIONAL_FUNC	▪ 0: (default) performs the same as conventional S-element. FBASE/FMAX-based linear convolution is performed. ▪ 1: Performs rational function approximation then recursive convolution; also handles non-causal S-parameters.
RATIONAL_FUNC_REUSE	▪ 0: Discard previously extracted rational function data and re-run the rational function approximation. ▪ 1: Reuse rational function data if available. ▪ 2: (default) Reuse rational function data if available and make no change in parameter source file (time stamp), FBASE, FMAX, HIGHPASS, LOWPASS, and passivity enforcement configurations; otherwise rerun the rational function approximation.

Parameter	Description
PASSIVE	Activates the passive checker to help debug passive models. The default is 0 for the S-element where 0=deactivate and 1=activate. Using the tolerance value specified by PASSIVE_TOL keyword, the eigenvalues of matrix (I-S*S'), ev[i], will be checked. If any frequency point violates RE(ev[i]) > -(TOL*0.1), HSPICE issues a warning containing a list of violating frequencies with an “E” flag. Also, the checker verifies potential passivity violations by checking the summation of each S-parameter matrix column. If Sum > (1.0+TOL), HSPICE issues a warning containing a list of those violating frequencies with an “C” flag.
PASSIVE_TOL	Tolerance for eigenvalue checking activated by PASSIVE keyword. Default value is 0.01.
ENFORCE_ PASSIVE	With the ENFORCE_PASSIVE=1 keyword, the S-element checks passivity of all the given frequency sampling points. Once passivity violations are found, the S-element seeks a minimum amount of loss property which restores passivity of all the violated points then adds the loss to all the given frequency points.
STAMP	▪ Y: Conventional admittance based stamp ▪ S: Scattering parameter based stamp (Note 1) ▪ YSTS: Admittance parameter based state space stamp (Note 2) ▪ SSTS: Scattering parameter based state space stamp (Note 2) ▪ DEEMBED: Produces negated stamp to de-embed given a S-parameter block from the adjacent DUT connected in series (See De-embedding S-parameters in this chapter.) Note 1: Although Y and S stamp types behave mathematically equivalent, when the S type is selected, the S-element activates a procedure to reduce memory consumption by taking matrices’ sparseness into account. Note 2: YSTS and SSTS stamp methods may be activated when RATIONAL_FUNC=1 is used. The state space stamping embeds all the state variables for extracted rational function matrix into the modified nodal analysis (NMA) matrix instead of performing recursive convolution integration. Although this stamping method may incur additional computational cost, since it produces frequency an invariant NMA matrix, it enables time domain steady state (so called .SN in HSPICE RF) analysis to handle frequency-dependent S-parameter blocks.

Parameter	Description
PRECFAC	In almost all cases, you do not need to specify a value for this parameter. This parameter specifies the precondition factor keyword used for the precondition process of the S-parameter. A precondition is used to avoid an infinite admittance matrix. The default is 0.75, which is good for most cases. See also, Pre-Conditioning S-parameters.
FQMODEL	Specifies the name of the Frequency model file behavior of the S,Y, or Z parameters. .MODEL statement of sp type, which defines the frequency-dependent matrices array.

The `FQMODEL`, `TSTONEFILE`, `CITIFILE`, and `RFMFILE` parameters describe the frequency-varying behavior of a network. Only specify one of the parameters in an S model card. If more than one method is declared, only the first one is used and HSPICE issues a warning message.

For full example demo files of the S Model usage see S-parameter Examples.

Pre-Conditioning S-parameters

Certain S-parameters, such as series inductor (2-port), show a singularity when converting S to Y parameters. To avoid this singularity, the S-element adds kR_{ref} series resistance to pre-condition S matrices:

Equation 22
$$\mathbf{S}' = [k\mathbf{I} + (2-k)\mathbf{S}][(2+k)\mathbf{I} - k\mathbf{S}]^{-1}$$

- kR_{ref} is the reference impedance vector.
- k is the pre-conditioning factor.

To compensate for this modification, the S-element adds a negative resistor ($-\mathrm{kR}_{ref}$) to the modified nodal analysis (NMA) matrix in actual circuit compensation. To specify this pre-conditioning factor, use the `PREFAC` keyword in the S model statement. The default pre-conditioning factor is 0.75.

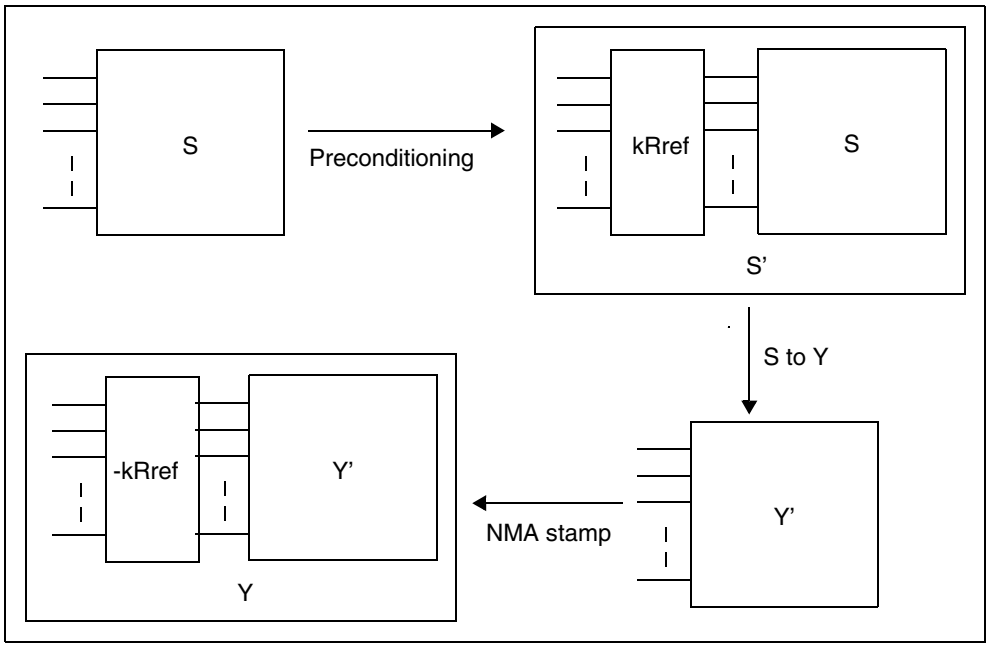


Figure 19 Pre-Conditioning S-parameters

Group Delay Handler in Time Domain Analysis

The S-element accepts a constant group delay matrix in time-domain analysis. You can also express a weak dependence of the delay matrix on the frequency as a combination of the constant delay matrix and the phase shift value at each frequency point. To activate or deactivate this delay handler, specify the `DELAYHANDLE` keyword in the S model statement.

The delay matrix is a constant matrix, which HSPICE RF extracts using finite difference calculation at selected target frequency points. HSPICE RF obtains the $T_{\omega(i,\ j)}$ delay matrix component as:

Equation 23
$$T_{\omega(i,\ j)} = \frac{d\theta_{Sij}}{d\omega} = \frac{1}{2\pi} \cdot \frac{d\theta_{Sij}}{df}$$

- f is the target frequency, which you can set using `DELAYFREQ`. The default target frequency is the maximum frequency point.
- θ_{Sij} is the phase of Sij.

After time domain analysis obtains the group delay matrix, the following equation eliminates the delay amount from the frequency domain system-transfer function:

Equation 24
$$y'_{mn(\omega)} = y_{mn(\omega)} \times e^{j\omega T_{mn}}$$

The convolution process uses the following equation to calculate the delay:

Equation 25
$$i_{k(t)} = (y'_{k1(t)},\ y'_{k2(t)},\ \ldots,\ y'_{kN(t)}) \times (v_{1(t-T_{K1})},\ v_{2(t-T_{K2})},\ \ldots,\ v_{Nt-T_{KN}})^T$$

Accelerating S-element Time Domain Performance with Recursive Convolution

The convolution integral is commonly used to handle frequency-dependent transfer characteristics. To get a system response at time t, the convolution integral can be carried out as shown in Eq. 26 on page 60:

Equation 26
$$y(t) = \int_{-\infty}^{t} x(\tau) \cdot\ h(t-\tau)d\tau$$

where, $x(t),\ h(t),\ y(t)$ are input at the t, system response function in the time domain and output at t, respectively. As observed in Eq. 26, the convolution integral is computationally expensive, especially if t becomes a long transient simulation due to an increasing time window for each time point evaluation. The conventional S-element obtains $h(t)$ by applying IFFT (Inverse Fast Fourier Transform) to the original system function in the frequency domain and performs a discrete linear convolution integral, while Eq. 26 is continuous.

On the other hand, when the frequency domain transfer function, $h(\omega)$ can be described as

Equation 27

$$Hs = \frac{A}{s + \omega_c}$$

Time domain conversion of (27) can be obtained as an exponential decay function,

Equation 28

$$h(t) = Ae^{-\omega_c t}$$

The computational cost of the convolution integral at time point t can be reduced using the convolution result at a previous time point. This technique is called recursive convolution. Since recursive convolution only requires numerical integration from a previous time point to the current time point, it saves computational time as well as storage for input signal history. Recursive convolution can be formulated only when the system response can be represented in certain forms of rational functions, as shown in Eq. 29:

Equation 29

$$s_{row,\ col} \text{ or } y'_{row,\ col} \cong B + sC + \sum_k \frac{Ar_k}{s + \omega r_k} + \sum_l \left(\frac{Ac_l}{s + \omega c_l} + \frac{Ac^*_l}{s + \omega c^*_l} \right)$$

Beginning with the 2007.03 release of HSPICE, when the keyword `RATIONAL_FUNC=1`, the HSPICE S-element generates a rational function matrix based on a given function and performs recursive convolution. Once the rational function is generated, the S-element stores the intermediate data for reuse in the following form: *MODEL_NAME.{yrf/yrfd}*.

When `RATIONAL_FUNC_REUSE=1`, the S-element seeks an available data file and reuses it without running a redundant rational function generation process.

In the current release, HSPICE also accepts rational function data input as external input. The input file syntax is described in the following section, Rational Function Matrix (.rfm) File Format.

In the current release, HSPICE accepts S- or preconditioned Y- parameter matrices as expressions with pairs of poles and residues. In cases of frequency-dependent scattering parameters, S( ), or preconditioned admittance parameter, Y'( ) can be represented as rational function matrix components as,

Equation 30

$$\mathbf{S}' = [\alpha \mathrm{I} + (2 - \alpha)\mathbf{S}][(2 + \alpha)\mathrm{I} - \alpha\mathbf{S}]^{-1}$$

Equation 31

$$\mathbf{Y}' = \mathbf{Y}_c^{\frac{1}{2}} [\mathrm{I} - \mathrm{S}'][\mathrm{I} + \mathbf{S}']^{-1} \mathbf{Y}_c^{\frac{1}{2}}$$

The following sections discuss these topics:

- Multithreading Acceleration for S-element on Linux
- Ensuring Causality in the Rational Function Model
- Rational Function Matrix (.rfm) File Format

Multithreading Acceleration for S-element on Linux

One of the benefits of using the rational function model is that since the function can be modeled independent of the transient simulation configuration, generated rational function data can be reused in subsequent simulation runs. To reduce the computational time for this process, starting in 2009.09, the S-element can take the number of threads specified on the command line invocation of `-mt` as the maximum number of threads to be used. (See Running Multithread/Multiprocess HSPICE Simulations in the *HSPICE User Guide: Simulation and Analysis*. The actual number of threads to be used can be smaller depending on the size of the target S-parameters.

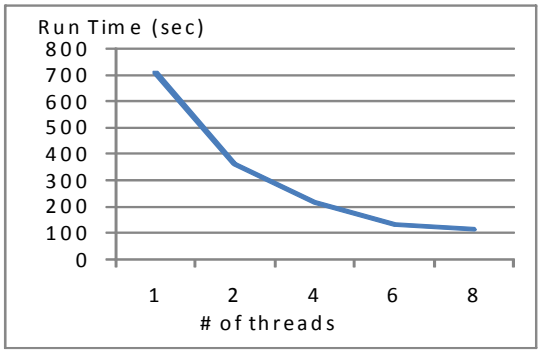


Figure 20 Wall clock time for the rational function generation of a 131 port S-parameter block (Linux system with 8 x 2666 MHz Intel® Xeon®)

Ensuring Causality in the Rational Function Model

When `RATIONAL_FUNC=1` in the S-element statement or the S-model statement, HSPICE enforces the causal behavior. If `RATIONAL_FUNC` is set to 1, the original transfer function is approximated as a summation of the partial rational functions which can be proved to be causal:

$$f_k = \frac{a_k}{\omega_k + s}$$

Therefore, the resulting function must be causal.

In the rational function generation process, potential unstable poles located in the right hand side of the complex plane are automatically filtered out.

Rational Function Matrix (.rfm) File Format

In addition to the rational function matrix (**.yrf/*.yrfd*) file discussed in the previous section, HSPICE provides syntax for users or 3rd parties to create an ASCII representation of the rational function matrix. The resulting **.rfm* file can then be read by the S-element via the S-model `RFMFILE=`*`file_name`*`.rfm` keyword. The **.rfm* file is divided into two parts:

- The header is made up of keywords and setup information for the entire system. This section (first five lines below) contains information about the data that follows, such as number of ports, matrix type, preconditioning factor, and reference impedance.
- The data field consists of rational function coefficients of each matrix component. Each matrix component begins with a `BEGIN` keyword and ends with the `END` keyword.

```
Version 200600
NPORT 2
MATRIX_TYPE Y
PRECFAC 0.75
Z0 50

BEGIN 1 1
CONST 0.0
C 0.0
DELAY 0.0
BEGIN_REAL 2
3.50774e+07 -4.54754e-05
2.37196e+08 -0.00327245
BEGIN_COMPLEX 2
3.81668e+08 3.74508e+08 0.00583496 -2.54387
6.88144e+08 2.08242e+08 6.66955e-06 -2.78498
END
```

A single line can only contain single pairs of pole and residue. Therefore, two numbers must appear in a line for a real pole and four numbers must appear in a line for a complex pole. A single complex pole represents a complex

conjugate pair of poles. An **.rfm* file does not need to include all the matrix components. In case certain terms are not found, the S-element regards these terms as ones with no propagation. The comment special character is an exclamation point. Lines that begin with '!' are ignored

An RFM keyword (with no whitespace) is always the first word on the new line. The table below lists available keywords.

Keyword	Description
VERSION *n*	Version number
NPORT *n*	Number of ports
MATRIX_TYPE [S|Y|Z]	Currently, S and Y are supported.
SYMMETRIC	This keyword indicates symmetric matrix. Only a single declaration must appear in the data field for transposing of pair of non-diagonal matrix components.
Z0 *val(s)* (or) Zo *val(s)*	Reference impedance of ports. Real number impedance only. When a single value is specified, the value is applied to all the ports. A vector of values with the size of the number of port can also be specified. A single line can only contain single number.
PRECFAC *val*	Preconditioning factor; must be between 0.5 and 1.0 (0.5 < < 1.0)
BEGIN *row col*	Beginning of a matrix component specified by row and col. row and col must be 1-based index of the matrix component.
CONST *val*	Constant term of the rational function “B” term of Eq. 29 on page 61; if not specified, equals 0.
C *val*	Reactive term of the rational function “C” term of Eq. 29 on page 61; if not specified, equals 0.
DELAY *val*	Propagation delay from port[col] to port[row]. Must be zero or a positive number. If not specified DELAY=0.
BEGIN_REAL *n*	Pairs of real poles and residues follow. Following each line must contain real pole and real residue in this order. If BEGIN_REAL is not specified, no real pole is constructed. Other keywords must appear before BEGIN_REAL.

Keyword	Description
BEGIN_COMPLEX *n*	Pairs of complex pole and residue follows. Following each line must contain real part and imaginary part of pole, real and imaginary part of residue in this order. Single complex pole and residue pair represents a conjugate pair of poles. If BEGIN_COMPLEX is not specified, no complex pole is constructed. Other keywords must appear before BEGIN_COMPLEX.
END	End of the matrix component.

S-element Data File Model Examples

The S model statement samples shown in Example 1 and Example 2 generate the same results.

Example 1

S model statement code example.

```
s1 n1 n2 n3 n_ref mname=smodel
.model smodel s n=3 fqmodel=sfqmodel z0=50 fbase=25e6 fmax=1e9
s1 n1 n2 n3 n_ref fqmodel=sfqmodel z0=50 fbase=25e6 fmax=1e9
```

Example 2

In this example, the S model statement has the characteristic impedance equal 100 instead of the 50 as defined in `smodel`. The impedance changes because the parameters defined in the S Element statement have higher priority than the parameters defined in the S model statement.

```
s1 n1 n2 n3 n_ref mname=smodel z0=100
.model smodel s n=3 fqmodel=sfqmodel z0=50 fbase=25e6 fmax=1e9
```

Example 3

In this example, `fqmodel`, `tstonefile`, and `citifile` are all declared in `smodel`. HSPICE accepts `tstonefile`, ignores both `fqmodel` and `citifile`, and issues a warning message. It is illegal to define a `tstonefile` and CITIfile `smodel` in the same statement. This prevents conflicts in the frequency-varying behavior description of the network. From the `tstonefile` file extension `.s3p`, you can tell that the network has three ports.

```
s1 n1 n2 n3 n_ref mname=smodel
.model smodel s tstonefile=exp1.s3p fqmodel=sfqmodel
   citifile=exp1.citi0
```

Example 4

In this example, `fqmodel` is declared both in the S-element statement and the S model statement. Each statement refers to a different `fqmodel`, which is not allowed.

```
s1 n1 n2 n3 n_ref mname=smodel fqmodel=sfqmodel_1
.model smodel s n=3 fqmodel=sfqmodel_2
```

Example 5

This example shows a generic S-parameter statement using port elements. For information on port elements see Identifying Ports with the P-element in the *HSPICE User Guide: Simulation and Analysis*.

```
**S-parameter example
.OPTION post
.probe v(n2)
P1 n1 0 port=1 Z0=50 ac=1v PULSE 0v 5v 5n 0.5n 0.5n 25n
P2 n2 0 port=2 Z0=50
.ac lin 500 1Hz 30MegHz
.tran 0.1ns 10ns
* reference node is set
S1 n1 n2 0 mname=s_model
* S parameter
.model s_model S TSTONEFILE = ss_ts.s2p
Rt1 n2 0 50
.end
```

Example 6

This example shows the option line and noise parameters of a Touchstone file.

```
!
! touchstone file example
!
# Hz S MA R 50.0000
0.00000 0.637187 180.000 0.355136 0.00000
0.355136 0.00000 0.637187 180.000
......
! # HZ S DB R 50.0000
! 0.00000 -3.91466 180.000 -8.99211 0.00000
! -8.99211 0.00000 -3.91466 180.000
! ......
!
!# Hz S RI R 50.0000
! 0.00000 -0.637187 0.00000 0.355136 0.00000
! 0.355136 0.00000 -0.637187 0.00000
! ......
!
! 2-port noise parameter
! frequency[Hz] Nfmin[dB] GammaOpt(M) GammaOpt(P) RN/Z0
0.0000 0.29166 0.98916 180.00 0.11055E-03
0.52632E+08 6.2395 0.59071 -163.50 0.32868
0.10526E+09 7.7898 0.44537 175.26 0.56586
! ......
! end of file
```

Example 7

This example shows an S-parameter statement using port elements and its referenced CITI file. For information on port elements see the Identifying Ports with the P-element. in the *HSPICE User Guide: Simulation and Analysis*.

```
**S-parameter
.OPTION post
.probe v(n2)
P1 n1 0 port=1 Z0=50 ac=1v PULSE 0v 5v 5n 0.5n 0.5n 25n
P2 n2 0 port=2 Z0=50
.ac lin 500 1Hz 30MegHz
.tran 0.1ns 10ns
*reference node is set
*S1 n1 n2 0 mname=s_model
* use default reference node
S1 n1 n2 mname=s_model
* S parameter
.model s_model S CITIFILE = ss_citi.citi Z0=50
Rt1 n2 0 50
.end
```

Multiport Noise Model for Passive Systems

Multiport passive and lossy circuits, such as transmission lines and package parasitics, can exhibit considerable thermal noise. The passive noise model is used to present such thermal noise for the S-element representing such circuits. The S-element passive noise model supports normal, two-port and multi-port noise analysis (`.NOISE=1`) and `.LIN noisecalc=1` for two-port and `.LIN noisecalc=2` for N-port]).

The following sections discuss these topics:

- Input Interface
- Output Interface

Input Interface

To trigger a passive multiport noise model, the NOISE and DTEMP keywords in an S-element statement are used:

```
Sxxx n1...nN
+ ...
+ [NOISE=[1|0]] [DTEMP=value]
```

Parameter	Description
NOISE	Activates thermal noise. ■ 1 (default): element generates thermal noise ■ 0: element is considered noiseless
DTEMP	Temperature difference between the element and the circuit, expressed in ×C. The default is 0.0. Element temperature is calculated as: T = Element temperature (° K) = 273.15 (° K) + circuit temperature (° C) + DTEMP (° C) Where circuit temperature is specified using either the .TEMP statement, or by sweeping the global TEMP variable in .DC, .AC, or .TRAN statements. When a .TEMP statement or TEMP variable is not used, the circuit temperature is set by .OPTION TNOM, which defaults to 25° C unless you use .OPTION SPICE, which raises the default to 27° C.

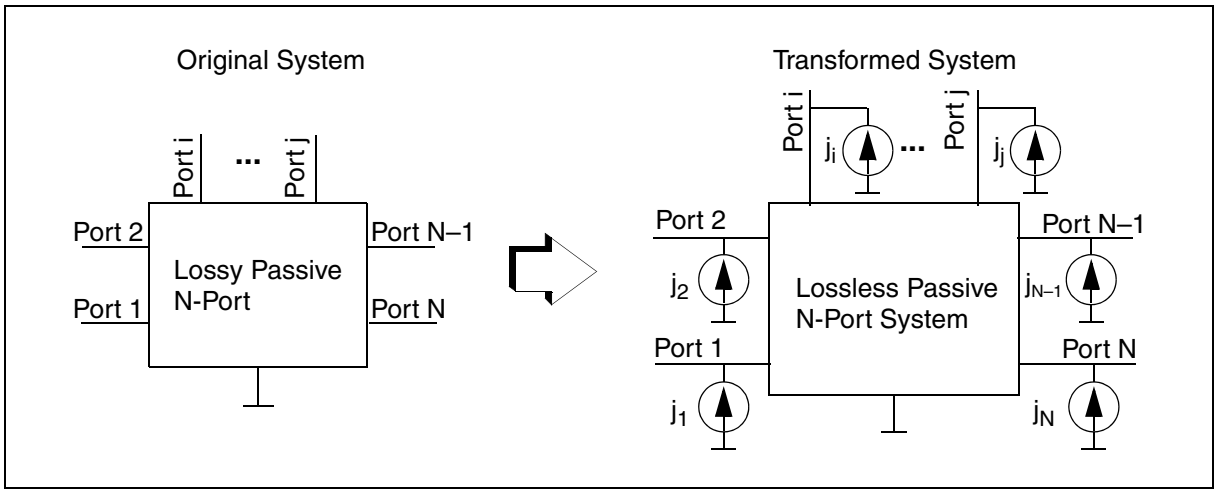
When `NOISE=1`, HSPICE generates a N×N noise-current correlation matrix from the N×N S-parameters according to Twiss' Theorem. The result can be stamped into an HSPICE noise analysis as N-correlated noise current sources: j_i (i=1~N), as shown below:

Equation 32

$$C = 2kT(Y + Y^{*T}) = \begin{bmatrix} \overline{|j_1|^2} & \overline{j_1 j_2^*} & \cdots & \overline{j_1 j_N^*} \\ \overline{j_2 j_1^*} & \overline{|j_2|^2} & \cdots & \overline{j_2 j_N^*} \\ \cdots & \cdots & \cdots & \cdots \\ \overline{j_N j_1^*} & \overline{j_N j_2^*} & \cdots & \overline{|j_N|^2} \end{bmatrix}$$

Where $Y = Y_c(I - S)(I + S)^{-1}$

The noise-current correlation matrix represents the frequency-dependent statistical relationship between N noise current sources, j_i (i=1~N), shown in the following figure.



Output Interface

HSPICE creates a *.lis* output list file that shows the results of a noise analysis just as any other noisy elements. The format is as following:

```
**** s element squared noise voltages (sq v/hz)

    element          0:s1
     N(i,j)          data
   r(N(i,j))          data
    ... i,j = 1~N ...
      total          data
```

Where:

- `N(i,j)` = contribution of $j_i j_j^*$ to the output port
- `r(N(i,j))` = transimpedance of j_i to the output port
- `total` = contribution of total noise voltage of the S-element to the output port.

S-element Noise Model

This section describes how the S-element supports two-port noise parameters and multiport passive noise models.

The following sections discuss these topics:

- Two-Port Noise Parameter Support in Touchstone Files
- Input Interface
- Output Interface
- Notifications and Limitations

Two-Port Noise Parameter Support in Touchstone Files

The S-element is capable of reading in two-port noise parameter data from Touchstone data files and then transform the raw data into a form used for `.NOISE` and `.LIN noisecalc=1[or 2]` analysis.

For example, you can represent a two-port with an S-element and then perform a noise analysis (or any other analysis). The S-element noise model supports normal and two-port (`.NOISE` and `.LIN noisecalc=1`). See Noise Parameters in 2-Port and N-Port Networks.

Note: Because Touchstone files currently provide only two-port noise parameters, this type of noise model only supports two-port S-parameter noise analysis for both passive and active systems.

Input Interface

The frequency-dependent two-port noise parameters are provided in a network description block of a Touchstone data file following the S-parameter data block.

The noise parameter data is typically organized by using the following syntax:

```
frequency[Hz] Nfmin[dB] GammaOpt(M) GammaOpt(P) RN/Z0
{ ...data... }
```

Where:

- `frequency` = frequency in units
- `Nfmin[dB]` = minimum noise figure (in dB)
- `GammaOpt(M)` = magnitude of reflection coefficient needed to realize Fmin
- `GammaOpt(P)` = phase (in degrees) of reflection coefficient needed to realize Fmin
- `RN/Z0` = normalized noise resistance
- `!` = indicates a comment line

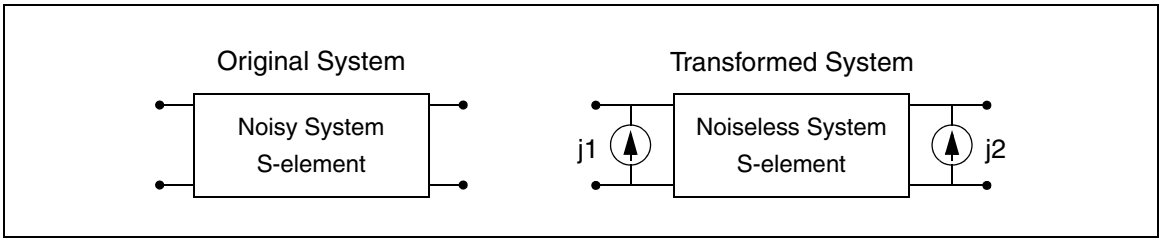
For example:

```
! 2-port noise parameter
! frequency[Hz] Nfmin[dB] GammaOpt(M) GammaOpt(P) RN/Z0
0.0000 0.29166 0.98916 180.00 0.11055E-03
0.52632E+08 6.2395 0.59071 -163.50 0.32868
0.10526E+09 7.7898 0.44537 175.26 0.56586
```

Both `GammaOpt` and `RN/Z0` values are normalized with respect to the characteristic impedance, `Z0`, specified in the header of the Touchstone data file. HSPICE reads this raw data and converts it to a coefficient of the noise-current correlation matrix. This matrix can be stamped into an HSPICE noise analysis as two correlated noise current sources: j_1 and j_2, as shown here:

$$C = \begin{bmatrix} \overline{|j_1|^2} & \overline{j_1 j_2^*} \\ \overline{j_2 j_1^*} & \overline{|j_2|^2} \end{bmatrix}$$

The noise-current correlation matrix represents the frequency-dependent statistical relationship between two noise current sources, j_1 and j_2, as illustrated in the following figure.



Output Interface

HSPICE creates a *.lis* output list file that shows the results of a noise analysis just as any other noisy elements. The format is as following:

```
**** s element squared noise voltages (sq v/hz)
element      0:s1
 N11         data
r(N11)       data
 N12         data
r(N12)       data
 N21         data
r(N21)       data
 N22         data
r(N22)       data
 total       data
```

Where:

- `N11` = contribution of j_1 to the output port
- `r(N11)` = transimpedance of j_1 to the output port
- `N12` = contribution of $j_1 j_2^*$ to the output port
- `r(N12)` = transimpedance of j_1 to the output port
- `N21` = contribution of $j_2 j_1^*$ to the output port
- `r(N21)` = transimpedance of j_2 to the output port
- `N22` = contribution of j_2 to the output port

- `r(N22)` = transimpedance of j_2 to the output port
- `total` = contribution of total noise voltage of the S Element to the output port.

Notifications and Limitations

- Because Touchstone files currently provide only two-port noise parameters, this type of noise model only supports two-port S-parameter noise analysis for both passive and active systems.
- If your Touchstone file includes square brackets in a Z0 definition, HSPICE does not support the square brackets. Acceptable syntax is to list the z0 values without any brackets. For example:

```
.model s_par s tstonefile='tsn.s4p'
+ z0=50 50 50 50
```

For readability, parentheses can be used.

```
.model s_par s tstonefile='tsn.s4p'
+ z0=(50 50 50 50)
```

Small-Signal Parameter Data Frequency Table Model (SP Model)

The small-signal parameter data frequency table model (SP model) is a generic model that describes frequency-varying behavior.

The following sections discuss these topics:

- SP Model Syntax
- Four Valid Forms of the SP Model

SP Model Syntax

```
.MODEL name sp [N=val FSTART=val FSTOP=val NI=val
+ SPACING=val MATRIX=val VALTYPE=val INFINITY=matrixval
+ INTERPOLATION=val EXTRAPOLATION=val DC=val]
+ DATA=(npts ...)|DATAFILE=filename
```

Parameter	Description
name	Model name.
N	Matrix dimension (number of signal terminals). Default is 1. If you use a value other than the default, you must specify that value before you set INFINITY and DATA.
FSTART	Starting frequency point for data. Default=0.
FSTOP	Final frequency point for data. Use this parameter only for the LINEAR and LOG spacing formats.
NI	Number of frequency points per interval. Use this parameter only for the DEC and OCT spacing formats. Default=10.
SPACING	Data sample spacing format: ▪ LIN (LINEAR): uniform spacing with frequency step of (FSTOP-FSTART)/(npts-1). The default. ▪ OCT: octave variation with FSTART as the starting frequency, and NI points per octave. npts sets the final frequency. ▪ DEC: decade variation with FSTART as the starting frequency, and NI points per decade. npts sets the final frequency. ▪ LOG: logarithmic spacing. FSTART and FSTOP are the starting and final frequencies. ▪ POI: non-uniform spacing. Pairs data ▪ (NONUNIFORM) points with frequency points.
MATRIX	Matrix (data point) format: ▪ SYMMETRIC: symmetric matrix. Specifies only lower-half triangle of a matrix (default). ▪ HERMITIAN: similar to SYMMETRIC; off-diagonal terms are complex-conjugates of each other. ▪ NONSYMMETRIC: non-symmetric (full) matrix.
VALTYPE	Data type of matrix elements: ▪ REAL: real entry. ▪ CARTESIAN: complex number in real/imaginary format (default). ▪ POLAR: complex number in polar format. Specify angles in radians.

Parameter	Description
INFINITY	Data point at infinity. Typically real-valued. This data format must be consistent with MATRIX and VALTYPE specifications. npts does not count this point.
INTERPOLATION	Interpolation scheme: ▪ STEP: piecewise step. This is the default. ▪ LINEAR: piecewise linear. ▪ SPLINE: b-spline curve fit.
EXTRAPOLATION	Extrapolation scheme during simulation: ▪ NONE: no extrapolation is allowed. Simulation terminates if a required data point is outside of the specified range. ▪ STEP: uses the last boundary point. The default. ▪ LINEAR: linear extrapolation by using the last two boundary points. If you specify the data point at infinity, then simulation does not extrapolate and uses the infinity value.
npts	Number of data points.
DC	Data port at DC. Normally real-valued. This data format must be consistent with MATRIX and VALTYPE specifications. npts does not count this point. You must specify either the DC point or the data point at frequency=0.
DATA	Data points. ▪ Syntax for LIN spacing: .MODEL name sp SPACING=LIN [N=dim] FSTART=f0 + DF=f1 DATA=npts d1 d2 ... ▪ Syntax for OCT or DEC spacing: .MODEL name sp SPACING=DEC or OCT [N=dim] + FSTART=f0 NI=n_per_intval DATA=npts d1 d2 ... ▪ Syntax for POI spacing: .MODEL name sp SPACING=NONUNIFORM [N=dim] + DATA=npts f1 d1 f2 d2 ...

Parameter	Description
DATAFILE	Data points in an external file. This file must contain only raw numbers without any suffixes, comments or continuation letters. The first number in the file must be an integer value to indicate the number of sampling points in the file. Then, sampling data must follow. The order of sampling data must be the same as in the DATA statement. This data file has no limitation on line length so you can enter a large number of data points.

Note: Interpolation and extrapolation occur after the simulator internally converts the Z and S-parameter data to Y-parameter data.

Four Valid Forms of the SP Model

The four sample files below are valid forms of the SP model.

SP Model 1: Symmetric complex matrices in linear frequency spacing

```
.MODEL fmod SP N=2 FSTOP=30MegHz
+ DATA = 2
* matrix at f=0
+  0.02      0.0
* Re(Y11) Im(Y11)
+ -0.02     0.0              0.02    0.0
* Im(Y21) Im(Y21) (= Y21)   Re(Y22) Im(Y22)
* matrix at f=30MHz
+ 0.02     0.0
* Re(Y11) Im(Y11)
+ -0.02    0.0              0.02    0.0
* Im(Y21) Im(Y21) (= Y21) Re(Y22) Im(Y22)
```

SP Model 2: Non-symmetric complex matrices in linear frequency spacing

```
.MODEL fmod SP N=2 FSTOP=30MegHz MATRIX=NONSYMMETRIC
+ DATA = 2
* matrix at f=0
+ 0.02     0.0     -0.02   0.0
* Re(Y11) Im(Y11) Re(Y12) Im(Y12)
+ -0.02    0.0     0.02    0.0
* Im(Y21) Im(Y21) Re(Y22) Im(Y22)
* matrix at f=30MHz
+ 0.02     0.0     -0.02   0.0
* Re(Y11) Im(Y11) Re(Y12) Im(Y12)
+ -0.02    0.0     0.02    0.0
* Im(Y21) Im(Y21) Re(Y22) Im(Y22)
```

SP Model 3: Symmetric complex matrices in non-uniform frequency spacing

```
.MODEL fmod SP N=2 SPACING=POI
+ DATA = 1
+ 0.0 * first frequency point
* matrix at f=0
+ 0.02      0.0
* Re(Y11) Im(Y11)
+ -0.02     0.0              0.02    0.0
* Im(Y21) Im(Y21) (= Y21)   Re(Y22) Im(Y22)
+ 30e+6 * second frequency point
* matrix at f=30MHz
+ 0.02      0.0
* Re(Y11) Im(Y11)
+ -0.02     0.0              0.02    0.0
* Im(Y21) Im(Y21) (= Y21)   Re(Y22) Im(Y22)
```

SP Model 4: Non-symmetric real matrices in linear frequency spacing

```
.MODEL fmod SP N=2 FSTOP=30MegHz VALTYPE=REAL
+ MATRIX=NONSYMMETRIC
+ DATA = 2
* matrix at f=0
+ 0.02 -0.02
* Y11    Y12
+  -0.02 0.02
* Y21    Y22
* matrix at f=30MHz
+ 0.02 -0.02
* Y11    Y12
+ -0.02 0.02
* Y21    Y22
```

Example 1

```
**S-parameter example
.OPTION post=2
.probe v(n2)
V1 n1 0 ac=1v PULSE 0v 5v 5n 0.5n 0.5n 25n
.op
.ac lin 500 1Hz 30MegHz
.tran 0.1ns 10ns
*S1 n1 n2 0 mname=s_model
S1 n1 n2 0 mname=s_model
.model s_model S fqmodel=fmod Z0=50 50
*.model s_model S fqmodel=fmod2 Z0=50 100
* S parameter for Z0=(50 50)
.MODEL fmod SP N=2 FSTOP=30MegHz DATA = 1
+ 0.333333333 0.0 0.666666667 0.0 0.333333333 0.0
* S parameter for Z0=(50 100)
.MODEL fmod2 SP N=2 FSTOP=30MegHz MATRIX=NONSYMMETRIC
+ DATA = 1
+ 0.5 0.0     0.5 0.0
+ 1.0 0.0     0.0 0.0
Rt1 n2 0 50
.end
```

Example 2

Figure 21 on page 78 illustrates a transmission line that uses a resistive termination, and Table 3 on page 81 shows a corresponding input file listing. In this example, the two outputs from the resistor and S parameter modeling must match exactly.

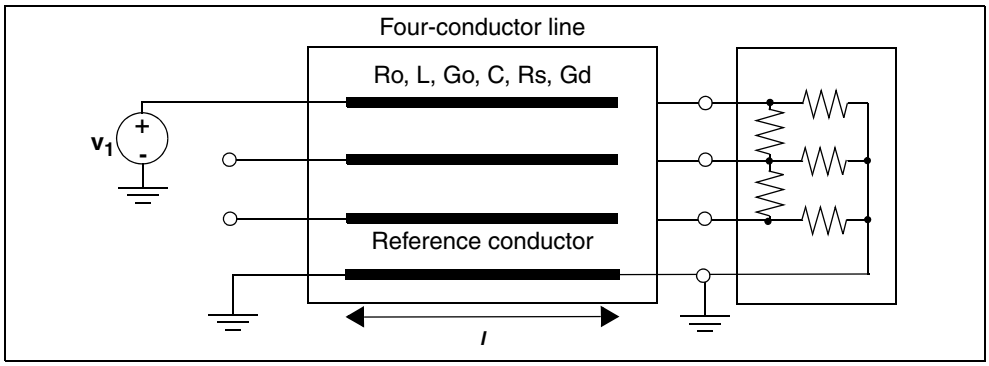


Figure 21 Transmission Line with Resistive Termination

Table 2 Input File Listing

Header, options, and sources	`*S-parameter x-line with a resistive positive termination.OPTION POST V1 i1 0 ac=1v`
Termination	`x1 o1 o2 o3 0 terminator`

Transmission line (W Element)	`W1 i1 i2 i3 0 o1 o2 o3 0 RLGCMODEL=wrlgc N=3`
	`+ L=0.97`
	`.MODEL wrlgc W MODELTYPE=RLGC N=3`
	`+ Lo = 2.78310e-07`
	`+ 8.75304e-08 3.29391e-07`
	`+ 3.65709e-08 1.15459e-07 3.38629e-07`
	`+ Co = 1.41113e-10`
	`+ -2.13558e-11 9.26469e-11`
	`+ -8.92852e-13 -1.77245e-11 8.72553e-11`

Frequency model definition	`.MODEL fmod sp N=3 FSTOP=30MegHz DATA= 1`
	`+ -0.270166 0.0`
	`+ 0.322825 0.0 -0.41488 0.0`
	`+ 0.17811 0.0 0.322825 0.0 -0.270166 0.0`

Resistor elements	`.SUBCKT terminator n1 n2 n3 ref`
	`R1 n1 ref 75`
	`R2 n2 ref 75`
	`R3 n3 ref 75`
	`R12 n1 n2 25`
	`R23 n2 n3 25`
	`.ends terminator`

Analysis	`.AC lin 500 0Hz 30MegHz` `.DC v1 0v 5v 1v`

Equivalent S parameter element	`.ALTER S parameter case`
	`.SUBCKT terminator n1 n2 n3 ref S1 n1 n2 n3 ref`
	`+ FQMODEL=fmod`
	`.ENDS terminator`
	`.END`

Example 3

The transmission line example shown here uses capacitive network termination. The two outputs from the resistor and S-parameter modeling in

Example 4 differs slightly due to the linear frequency dependency relative to the capacitor. To remove this difference, use the linear interpolation scheme in `.MODEL`.

Frequency model definition	`.MODEL fmod sp N=3 FSTOP=30MegHz`
	`+ DATA= 2`
	`+ 1.0 0.0`
	`+ 0.0 0.0 1.0 0.0`
	`+ 0.0 0.0 0.0 0.0 1.0 0.0`
	`+ 0.97409 -0.223096`
	`+ 0.00895303 0.0360171 0.964485 -0.25887`
	`+ -0.000651487 0.000242442 0.00895303`
	`+ 0.0360171 0.97409 -0.223096`

Using capacitive elements	`.SUBCKT terminator n1 n2 n3 ref`
	`C1 n1 ref 10pF`
	`C2 n2 ref 10pF`
	`C3 n3 ref 10pF`
	`C12 n1 n2 2pF`
	`C23 n2 n3 2pF`
	`.ENDS terminator`

Example 4

Figure 22 and Table 3 on page 81 show an example of a transmission line that uses the S-parameter.

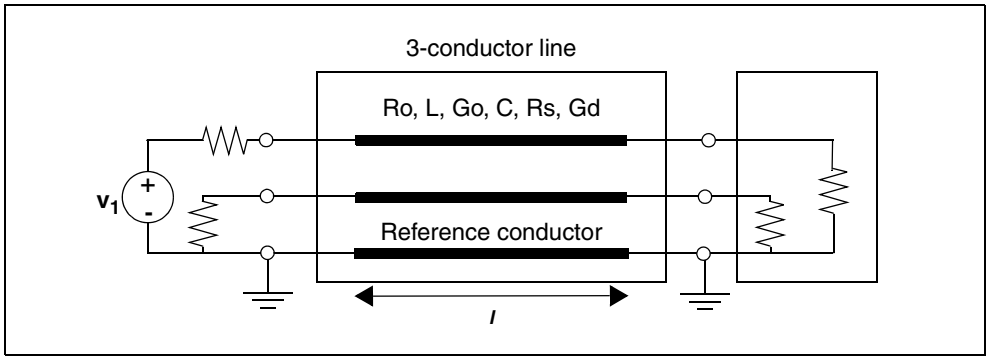


Figure 22 3-Conductor Transmission Line

Table 3 Input File Listing

Header, options, and sources	`*S parameter ex3: modeling x-line by using`
	`+ S parameter`
	`.OPTION POST`
	`vin in0 0 ac=1`

Analysis	`.AC lin 100 0 1000meg` `.DC vin 0 1v 0.2v`
Transmission line	`W1  in1 in2 0 out1 out2 0 N=2 RLGCMODEL=m2`

Termination	`R1 in0   in1   28`
	`R2 in2   0     28`
	`R3 out1  0     28`
	`R4 out2  0      28`

W-element RLGC model definition	`.MODEL m2 W ModelType=RLGC, N=2`
	`+ Lo= 0.178e-6    0.0946e-7    0.178e-6`
	`+ Co= 0.23e-9    -0.277e-11    0.23e-9`
	`+ Ro= 0.97         0            0.97`
	`+ Go= 0            0            0`
	`+ Rs= 0.138e-3    0            0.138e-3`
	`+ Gd= 0.29e-10    0            0.29e-10`

Frequency model definition	`.MODEL SM2 sp N=4 FSTART=0 FSTOP=1e+09`
	`+ SPACING=LINEAR DATA= 60 0.00386491 0`
	`+ 0 0 0.00386491 0 0.996135 0 0 0 0.00386491 0`
	`+ 0 0 0.996135 0 0 0 0.00386491 0`
	`+ -0.0492864 -0.15301`
	`+ 0.00188102 0.0063569 -0.0492864`
	`+ -0.15301 0.926223 -0.307306 0.000630484`
	`+ -0.00154619 0.0492864 -0.15301`
	`+ 0.000630484 -0.00154619 0.926223`
	`+ -0.307306 0.00188102 0.0063569`
	`+ -0.0492864 -0.15301 -0.175236 -0.241602`
	`+ 0.00597 0.0103297 -0.175236 -0.241602`
	`+ 0.761485 -0.546979 0.00093508`
	`+ -0.00508414 -0.175236 -0.241602`
	`+ 0.00093508 -0.00508414 0.761485`
	`+ -0.546979 0.00597 0.0103297 -0.175236`
	`+ -0.241602`
	`+ ...`

Table 3 Input File Listing (Continued)

Equivalent S-parameter element	`.SUBCKT terminator n1 n2 n3 ref`
	`  S1 n1 n2 n3 ref FQMODEL=SM2`
	`.ENDS terminator`
	`.END`

S Model Data Smoothing

Four smoothing functions are provided for the S model. Each of these is available for the S-element and W-element. Scattering parameters are frequently given from measurement instruments such as vector network analyzers (VNA). In measurements, there are many causes of noise injection such as calibration failure, electromagnetic interference (EMI) and so on, especially in high frequency range. For such cases, several data smoothing functions are available to the S-parameter data reader for the purpose of restoring the original noiseless data.

Data smoothing alters the original data to suppress unwanted noise. Therefore, if you are confident of the accuracy of the original data, data smoothing is not recommended.

Data Smoothing Methods

Each smoothed data at *i*th point *S'i* is given as a function of original data Si and its neighbors as,

$$S'\ i\ =\ S_{i-width},\ \ ...,\ S_i...S_{i+width}$$

Four functions for data smoothing are provided:

- Mean: take the average value of $S'\ i\ =\ S_{i-width},\ \ ...,\ S_i...S_{i+width}$
- Median: take the value situated in the middle of $S'\ i\ =\ S_{i-width},\ \ ...,\ S_i...S_{i+width}$

- 2nd order polynomial fit: perform least square fitting of $S'\ i\ =\ S_{i-width},\ ...,\ S_i...S_{i+width}$ with 2nd order polynomial then, compute the value at *i*th frequency.
- 4th order polynomial fit: perform least square fitting of $S'\ i\ =\ S_{i-width},\ ...,\ S_i...S_{i+width}$ with 4th order polynomial then, compute the value at *i*th frequency.

S-model Syntax

```
.model model_name S ....
+ [SMOOTH=val] [SMOOTHPTS=val]
```

Default 0

Keyword	Description
SMOOTH	An integer value to choose one of following methods ■ 0: no smoothing (default) ■ 1: mean ■ 2: median ■ 3: 2nd order polynomial fit ■ 4: 4th order polynomial fit
SMOOTHPTS	An integer value to specify width of the smoothing window on each side of the target point. In total, 2*x +1 point is taken at each point calculation.

Description

Each smoothing function has different characteristics. It is recommended that users observe the original data on the waveform viewer when determining the smoothing filter configuration. Typically, the average function has a strong ability of smoothing but it may lose the necessary bumps in data if they are narrow. Since both the average and median (SMOOTH=1 or 2) takes an intermediate value of the points within the specified window, if these reference points have a wide range of phase difference due to sparse frequency sampling, the smoothing result may lose accuracy.

The Median filter is effective if there are sharp and high noise spikes. These spikes are eliminated by the median filter without changing the offset level. Polynomial fittings are relatively weak in data smoothing but they preserve narrow bumps. Typically, for transmission line type S-parameters, polynomial

fittings are effective since sinusoidal curves (many narrow bumps) are expected in real and imaginary vs. frequency plots due to constant propagation delay.

2nd order and 4th order polynomial smoothing methods preserve the overall waveform trend better than the former two methods but they are relatively weak in strong, narrow range noises.

Example

The plot on the left side of Figure 23 on page 84 shows the original measurement data for the propagation term of the differential pair of transmission lines. With “SMOOTH=3 SMOOTHPTS=5,” second order polynomial fitting with 11 points (5 points from each side in addition to the target point) is applied. The plot on the right side shows smoothed data.

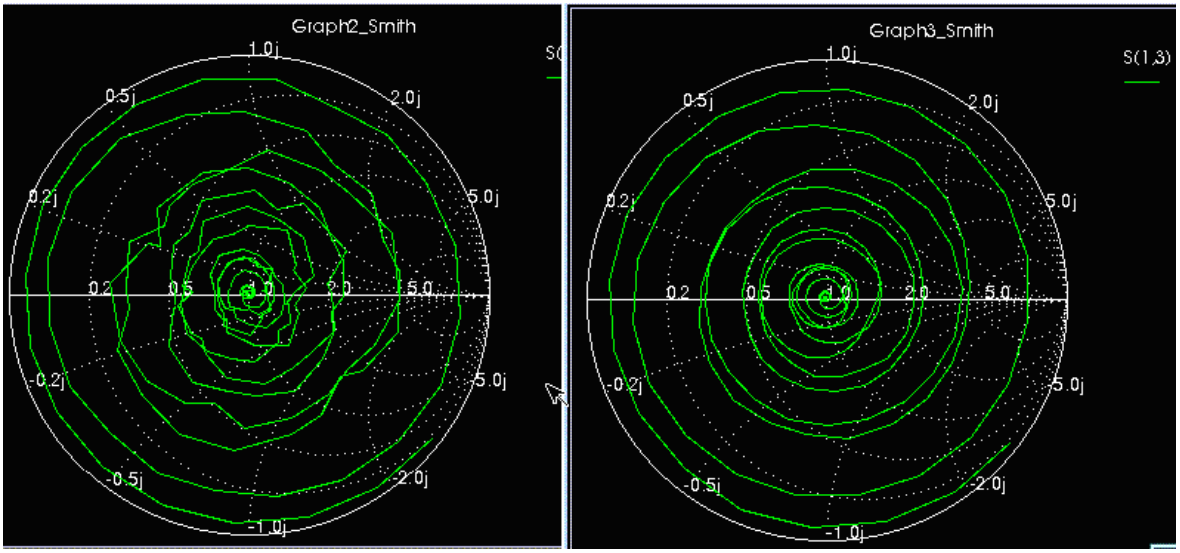


Figure 23 Using the smoothing keywords on S Model data

Predicting an Initial Value for FMAX in S-element Models

When selecting a starting point for the `FMAX` parameter in your S-parameter, it is important to set `FMAX` high enough to account for the fastest edges and higher order harmonics in the input waveforms. Here are two methods to determine a starting point for setting `FMAX`. These methods are only meant to provide an initial value. Always check your results to ensure you are getting the

accuracy you need. Also, setting FMAX without having enough data present in your S-parameter data file may result in extrapolation errors. Refer to this S-parameter application note for complete guidelines:

https://solvnet.synopsys.com/retrieve/017600.html

Method 1: Based on Risetime using the "knee frequency"

This method is handy for TDR type simulations where the incident wave has only one rising or falling edge.

Most energy in digital pulses concentrates below the knee frequency. The behavior of a circuit at the knee frequency determines its processing of a step edge. The knee frequency for any digital signal is related to the rise and fall time of its digital edges, but not its clock rate. If you want to pass a certain rise time with little degradation, you need the medium it propagates through to be about 2x the knee frequency.

The knee frequency can be calculated based on a 10-90% or 20-80% risetime measurement.

For 10-90%, FKNEE = (.35/Trise)

For 20-80%, FKNEE = (.5/Trise)

For example, the FMAX needed for a 25ps risetime measured at 10-90% of the rising edge is $2 \cdot \left(\frac{.35}{25\text{ps}}\right) = 28\text{GHz}$

Method 2: Using FFT

In this method, you run an FFT on the primary data signal and check the frequency at the eleventh harmonic. See the .FFT command in the *HSPICE Reference Manual: Commands and Control Options*. You can use the waveform calculator in Custom WaveView to check the frequency and eleventh harmonic.

In CosmosScope:

1. Select the data signal.
2. Open the Waveform Calculator.
3. "Paste" the waveform into the calculator with the middle mouse button.
4. Click the **WAVE** button and select FFT.
5. Modify the number of points and start/stop times if desired. Click **OK**.
6. Click the **Graph X** button to plot the FFT.

Note: HSPICE usually selects a suitable value of `FBASE` for you that provides a good trade-off between the number of sampling points and performance, so allow `FBASE` to default unless you are not seeing the resolution and accuracy you require.

De-embedding S-parameters

HSPICE's S-element provides an interface to use measured or extracted S-parameters. When obtaining S-parameters by measurement, a common difficulty is that the measured S-parameters may contain characteristics of unwanted peripherals such as probes, connectors, and so on in addition to the target device under test (DUT) characteristics). If the characteristics of these unwanted peripherals are known as S-parameters, you can use HSPICE to de-embed them through common text-based data file formats such as Touchstone, CITIfile, and the netlist model format, **.sc#* by using the existing `.LIN` analysis and S-elements with the following stamping scheme:

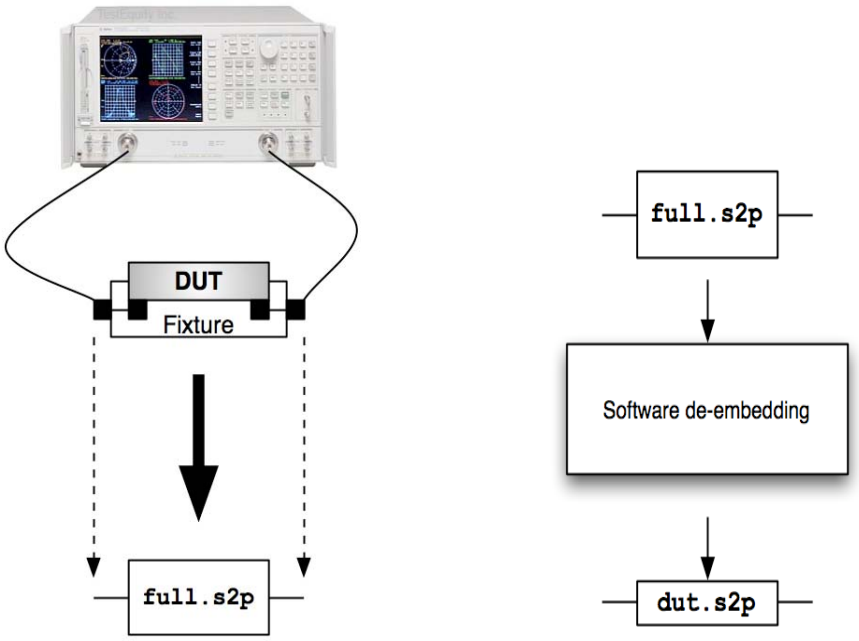


Figure 24 (Left) The measured S-parameter model may include the characteristics of fixtures (connectors, probe, etc.) as well as device under test (DUT) itself. (Right) When the characteristics of fixtures are given, software de-embedding extracts pure DUT characteristics.

S model Syntax

```
S1 1 2 3 4 5 6 7 8 mname=model_name …
+ STAMP=DEEMBED
```

or

```
.model model_name S …
+ STAMP=DEEMBED
```

The `STAMP=DEEMBED` setting produces a negated stamp to de-embed a given S-parameter block from the adjacent DUT connected in series. For example, the following netlist removes 50 ohm resistance represented by the `S1` element from the DUT of 100 ohm resistor, `R1`. As a result, the `.LIN` analysis produces S-parameters of 50 ohm series resistor.

```
P1 1 0 port=1  ac=1
P2 2 0 port=2
* DUT (w/ unwanted 50 ohm)
R1 m 2 100
* de-embed 50 ohm s-parameters
S1 1 m mname=smodel stamp=DEEMBED
.opt post
.ac POI 4 0.0 1.0 1e8 1e9
.lin format=touchstone
.MODEL Smodel S
+ N=2 FQMODEL=SFQMODEL TYPE=S Z0=50
.MODEL SFQMODEL SP N=2 SPACING=POI MATRIX=NONSYMMETRIC
+ DATA=1
+               0
+  0.3333333333     0      0.6666666667     0
+  0.6666666667     0      0.3333333333     0
+
.end
```

Note: Since negated stamp causes non-causal behavior in time domain analysis, `STAMP=DEEMBED` is only supported in frequency domain analyses. For time domain analyses, generate de-embedded S-parameters first using `.LIN` analysis and then reuse them.

S-parameter Standalone Manipulation Utility (SPutil)

HSPICE provides several capabilities to manipulate S-parameters including accepting multiple file formats, inter-/extrapolation schemes, data smoothing,

rational function approximation, and so on. These capabilities become available when an S-element is specified in an HSPICE circuit netlist and passed to HSPICE. The following sections describe a standalone executable (`sputil`), available on Linux, Solaris, and Windows, which allows you to access these S-element features without invoking an HSPICE simulation and additional utility functions. The standalone executable may also be used as a tool to predict and verify the behavior of given S-parameters in advance of HSPICE simulation runs.

The following sections discuss these topics:

- SPutil Program Features
- Invoking the Utility
- SPutil Runset Format
- Commands
- Status Messages

SPutil Program Features

SPutil performs many of the functions which are built into the HSPICE S-element. In addition, utility functions can be added on demand. The following list categorizes the current set of *SPutil* functions.

- Read in and combine s/y parameter files
- Interpolate and/or extrapolate necessary output data points
- Passivity check and enforcement
- Rational function approximation
- Output manipulated S-parameter data

Invoking the Utility

Note: This utility program requires an HSPICE license to be checked out, since it can be switched to check out either HSPICE only or both HSPICE and HSPICE RF depending on your site requirement.

To run a set of S-parameter manipulation commands specified in the input file, type:

```
sputil runset.in
```

SPutil Runset Format

The *SPutil* runset file must contain a list of commands and arguments. The example below shows the basic structure of a *SPutil* runset. Command names must be compatible with HSPICE S-element/model keywords and other HSPICE netlist keywords. *SPutil* directly calls HSPICE's S-element object library in order to realize identical behavior to the corresponding control keyword available in the HSPICE S-element.

This example combines three S-parameter files, *ts1.s4p*, *ts2.s4p* and *citi1.citi*, into one Touchstone file, *output.s4p* with a combined list of frequency points specified by a `FREQ_SWEEP` keyword.

```
NPORT 4
TSTONEFILE ts1.s4p
TSTONEFILE ts2.s4p
CITIFILE citi1.citi
INTERPOLATION LINEAR
HIGHPASS 3
LOWPASS 1
TSTONE_OUT output.s4p
FREQ_SWEEP DEC 10 1.0 10e9
```

Commands

A command keyword must always be the first word on the new line. It is always one continuous word without embedded spaces.

Table 4 Current SPutil Command Set

Keyword	Description
S-element keywords —See also S-element Syntax on page 45 for full keyword descriptions	
NPORT n	Number of ports Note: Multiple declarations of *different* NPORTs are illegal.
MNAME *modelname*	Model name used for output
TYPE [S|Y|Z]	Parameter type. Currently, Y or S are supported. Default is S.
TSTONEFILE *filename*	Specifies a Touchstone file (supports version 2.0)

Table 4 Current SPutil Command Set (Continued)

Keyword	Description
CITIFILE *filename*	Specifies a CITIfile
BNPFILE *filename*	Specifies Broadband Network Parameter (BNP) file (Sigrity-proprietary). For details on BNP, see http://www.sigrity.com.
FBASE *value*	FBASE value for S-element
FMAX *value*	FMAX value for S-element
INTERPOLATION [STEP |LINEAR|SPLINE|HYBRID]	INTERPOLATION setting for S-element
INTDATTYP [RI|DBA|MA]	INTDATTYP setting for S-element
HIGHPASS *n*	HIGHPASS setting for S-element
LOWPASS *n*	LOWPASS setting for S-element
DELAYHANDLE *value*	DELAYHANDLE setting for S-element
DELAYFREQ *value*	DELAYFREQ value for S-element
RATIONAL_FUNC *n*	Rational function setting for S-element
PASSIVE [0|1]	Activates the passive checker to help debug passive models. The default is 0 for the S-element where 0=deactivate and 1=activate. Using the tolerance value specified by PASSIVE_TOL keyword, the eigenvalues of matrix (I-S*S'), ev[i], will be checked. If any frequency point violates RE(ev[i]) > -(TOL*0.1), HSPICE issues a warning containing a list of violating frequencies with an “E” flag. Also, the checker verifies potential passivity violations by checking the summation of each S-parameter matrix column. If Sum > (1.0+TOL), HSPICE issues a warning containing a list of those violating frequencies with an “C” flag.
PASSIVE_TOL *val*	Tolerance for eigenvalue checking activated by PASSIVE keyword. Default value is 0.01.
ENFORCE_PASSIVE [0|1]	When `enforce_passive 1` *SPutil* enforces PASSIVE setting for the S-parameter; default 0 (off).

Table 4 Current SPutil Command Set (Continued)

Keyword	Description
PRECFAC	Preconditioning Factor value for S-element. Non-zero precfac is used only when y-parameter output is requested with S-parameter input.
Control Keywords	
TSTONE_OUT *tstone_name*	Output Touchstone v. 1.0 file name
TSTONE2_OUT	Specifies Touchstone v. 2.0 output file
CITIFILE_OUT *citifile_name*	Output CITIfile name
SELM_OUT *selm_file_name*	Output sc# file name
DATAFORMAT	Specifies the format of the output data file: ▪ RI: real-imaginary ▪ MA: magnitude-phase (default format for Touchstone files) ▪ DB: DB(magnitude)-phase
FREQ_SWEEP *hspice_sweep*	Specifies a type of frequency sweep to allow checking of the manipulated S-parameter matrices. You can specify any of LIN, DEC, OCT, or POI. Specify the nsteps, start, and stop values using the following syntax for each type of sweep: ▪ LIN nsteps start stop ▪ DEC nsteps start stop ▪ OCT nsteps start stop ▪ POI nsteps freq_values If the FREQ_SWEEP keyword is not specified, then HSPICE chooses output frequency points as the combination of the input frequency point. For example: First input file has f=1 2 3 4 Second input file has f=5 6 7 ...then the output frequency points (by default, with no FREQ_SWEEP) are f=1 2 3 4 5 6 7.
SYMMETRY_CHECK [0|1]	Default: `0` When `SYMMETRY_CHECK 1` is specified, the keyword checks symmetry of the S-parameter matrix at each frequency point (especially useful in evaluating differential signaling systems where preserving symmetric S-parameter matrices is key).

Table 4 Current SPutil Command Set (Continued)

Keyword	Description
COLUMN_SUM [0|1]	Default: `0` When `column_sum 1` is specified, the keyword computes the summation of each column of the S-parameter matrix at each frequency point. A summation of each column indicates whether or not corresponding incidents are amplified according to $\sum_k s_{kj} a_j$. Results are printed in an *input_filename.colsum* text file which can be uploaded to the waveform viewer as a Text/PWL data file. The output file contains `real`, `imaginary`, and `magnitude` as functions of frequency for each column summation.
! (comment)	Lines beginning with '!' are ignored

Status Messages

Status messages are written in the command terminal as an *SPutil* runset proceeds. For example, when the following runset is executed:

```
NPORT 4
TSTONEFILE good_tx.s4p
PASSIVE 1
TSTONE_OUT output.s4p
FREQ_SWEEP DEC 10 1e3 1e9
```

... messages such as these might be returned:

```
% sputil test.in
Reading Touchstone File [good_tx.s4p]...

Read Touchstone File [good_tx.s4p]

s-params: 801 pts 300000Hz - 8.5e+09Hz
no noise parameter read

Performing passivity check...

Warning:  s-model [good_tx.s4p] passivity violation detected...
Writing Touchstone File [output.s4p]... done
        See "test.lis" for results and statistics.
```

References

[1] Dmitri Borisovich Kuznetsov and Jose E. Schutt-Aine, “Optimal Transient Simulation of Transmission Lines”, IEE Transaction on Circuits and Systems-I: Fundamental Theory and Applications. Vol. 43, No. 2, February 1996

[2] Bjorn Gustavsen and Adam Semlyen, “Rational Approximation of Frequency Domain Responses by Vector Fitting,” IEEE Transaction on Power Delivery, Vol.14, No.3, pp. 1052-1061, July 1999

3

W-element Modeling of Coupled Transmission Lines

Describes how to use basic transmission line simulation equations and an optional method for computing the parameters of transmission line equations.

HSPICE ships many examples for your use. Find W-element-related demo files under Transmission (W-element) Line Examples.

The W-element is a versatile transmission line model that you can apply to efficiently and accurately simulate transmission lines, ranging from a simple lossless line to complex frequency-dependent lossy-coupled lines. Unlike the U-element, the W-element can output accurate simulation results without fine-tuning optional parameters. For more information on U-elements, see Chapter 5, Ideal and Lumped Transmission Line Models.

A transmission line is a passive element that connects any two conductors, at any distance apart. One conductor sends the input signal through the transmission line and the other conductor receives the output signal from the transmission line. The signal that transmits from one end of the pair to the other end is voltage between the conductors.

Examples of transmission lines include:

- Power transmission lines
- Telephone lines
- Waveguides
- Traces on printed circuit boards and multi-chip modules (MCMs)
- Bonding wires in semiconductor IC packages
- On-chip interconnections

This chapter describes the basic transmission line simulation equations. It explains how to use these equations as an input to the transmission line model,

the W-element. (For more information about the W-element, see Dmitri Kuznetsov, “Optimal Transient Simulation of Transmission Lines,” IEEE Trans., Circuits Syst., vol.43, pp. 110-121, Feb., 1996.)

This chapter also shows you an optional method for computing the parameters of the transmission line equations using the *field solver model*.

Transmission line simulation is challenging and time-consuming, because extracting transmission line parameters from physical geometry requires a significant effort. To minimize this effort, you can use a simple (but efficient and accurate) 2D electromagnetic field solver, which calculates the electrical parameters of a transmission line system, based on its cross-section.

These topics are covered in the following sections:

- Equations and Parameters
- Frequency-Dependent Matrices
- Wave Propagation
- Using the W-element
- Extracting Transmission Line Parameters (Field Solver)
- W-element Passive Noise Model
- Using the TxLine (Transmission Line) Tool Utility
- References

Equations and Parameters

Maxwell’s equations for the transverse electromagnetic (TEM) waves on multi-conductor transmission lines, reduce to the telegrapher’s equations. The general form of the telegrapher’s equation in the frequency domain is:

Equation 33
$$-\frac{\partial}{\partial z}\mathrm{v}(z,\ \omega) = [\mathrm{R}(\omega) + j\omega\mathrm{L}(\omega)]\mathrm{i}(z,\ \omega)$$

Equation 34
$$-\frac{\partial}{\partial z}\mathrm{i}(z,\ \omega) = [\mathrm{G}(\omega) + j\omega\mathrm{C}(\omega)]\mathrm{v}(z,\ \omega)$$

The preceding equations use the following definitions:

- Lower-case symbols denote vectors.
- Upper-case symbols denote matrices.
- v is the voltage vector across the lines.
- i is the current vector along the lines.

For the TEM mode, the transverse distribution of electromagnetic fields at any instant of time is identical to that for the static solution.

From a static analysis, you can derive the four parameter matrices for multi-conductor TEM transmission lines:

- resistance matrix, R
- inductance matrix, L
- conductance matrix, G
- capacitance matrix, C

The telegrapher's equations, and the four parameter matrices from a static analysis, completely and accurately describe TEM lines.

Not all transmission lines support pure TEM waves; some multi-conductor systems inherently produce longitudinal field components. In particular, waves propagating in either the presence of conductor losses or the absence of dielectric homogeneity (but not dielectric losses), must have longitudinal components.

However, if the transverse components of the fields are significantly larger than the longitudinal components, the telegrapher's equations (and the four parameter matrices obtained from a static analysis) still provide a good approximation. This is known as a quasi-static approximation.

Multi-conductor systems in which this approximation is valid are called quasi-TEM lines. For typical micro-strip systems the quasi-static approximation holds up to a few gigahertz.

Frequency-Dependent Matrices

The static (constant) L and C matrices are accurate for a wide range of frequencies. In contrast, the static (DC) R matrix applies to only a limited frequency range, mainly due to the skin effect. A good approximate expression of the R resistance matrix with the skin effect, is:

Equation 35 $$\mathrm{R}(f) \cong \mathrm{R}_o + \sqrt{f}(1+j)\mathrm{R}_s$$

Where:

- R_o is the DC resistance matrix.
- R_s is the skin effect matrix.

The imaginary term depicts the correct frequency response at high frequency; however, it might cause significant errors for low-frequency applications. In the W-element, you can optionally exclude this imaginary term:

```
Wxxx i1 i2 ... iN iR o1 o2 ... oN oR N=val L=val INCLUDERSIMAG=NO
```

In contrast, the G (loss) conductance matrix is often approximated as:

Equation 36 $$\mathrm{G}(f) \cong \mathrm{G}_o + \frac{f}{\sqrt{1+(f/\ f_{gd})^2}}\mathrm{G}_d$$

Where,

- G_o models the shunt current due to free electrons in imperfect dielectrics.
- G_d models the power loss due to the rotation of dipoles under the alternating field (C. A. Balanis, Advanced Engineering Electromagnetics, New York: Wiley, 1989).
- f_{gd} is a cut-off frequency.

If you do not set f_{gd}, or if you set f_{gd} to 0, then G(f) keeps linear dependency on the frequency. In the W-element, the default f_{gd} is zero (that is, G(f) does not use the f_{gd} value).

You can specify an alternate value in the W-element statement:

```
Wxxx i1 i2 ... iN iR o1 o2 ... oN oR N=val L=val fgd=val
```

If you prefer to use the previous linear dependency, set f_{gd} to 0.

Note: Fgd is used to estimate frequency dependent shunt loss conductance described as Equation (33) for the RLGC model without INCLUDEGDIMAG=yes only (see Fitting Procedure Triggered by INCLUDEGDIMAG Keyword).

When you specify INCLUDEGDIMAG=yes, the RLGC model estimates frequency-dependent shunt (C and G) parameters described as Equation Eq. 36 and the fgd value is not be used. Both of these are ways to fit the RLGC model fit with actual measurements.

If you have measured or computationally extracted a tabular RLGC model, it should be more accurate if parameter extraction is accurately done.

The following sections discuss these topics:

- Introduction to the Complex Dielectric Loss Model
- Fitting Procedure Triggered by INCLUDEGDIMAG Keyword
- Determining Matrix Properties
- Using the PRINTZO Option
- File Description for *.wzo

Introduction to the Complex Dielectric Loss Model

When the INCLUDEGDIMAG keyword = yes and there is no wp input, the W-element regards the Gd matrix as the conventional model and then automatically extracts constants for the complex dielectric model.

In conventional use, the HSPICE W-element RLGC model, frequency dependent conductance is approximated as Eq. 36 on page 98.

Where, Eq. 36 represents the increase of shunt conductance due to dielectric loss. These pure real non-constant functions of frequency violate causality[1]. As system operating frequency becomes significantly high even for PCB systems which use high polymer dielectric materials like FR4, the appearance of the dielectric loss becomes significant and significant non-causality of Eq. 36 appears.

The frequency-dependent loss of the shunt conductance in the dielectric is mainly due to dielectric polarization. This polarization loss leads to a complex permittivity, $\varepsilon(\omega)$, for the dielectric material[2].

Equation 37 $$\varepsilon(\omega) = \varepsilon'(\omega) - j\varepsilon''(\omega)$$

And loss tangent of the dielectric material can be specified as the ratio of imaginary part of $\varepsilon(\omega)$ to the real part,

Equation 38 $$\tan\delta(\omega) = \frac{\varepsilon''(\omega)}{\varepsilon'(\omega)}$$

For a single dielectric dipolar moment, complex electric permittivity can be written as,

Equation 39 $$\varepsilon(\omega) = \varepsilon_\infty + \omega_p \frac{\varepsilon_{dc} - \varepsilon_\infty}{j\omega + \omega_p}$$

Where, ε_{dc} and are ε_∞ low and high frequency limits of dielectric permittivity which are real numbers. And ω_p is the angular frequency that corresponds to the polarization time constant of the dielectric material. From Eq. 39, frequency dependent complex shunt loss conductance can be expressed as[3],

Equation 40 $$G(\omega) = Go + Gd\frac{j\omega}{j\omega + \omega_p}$$

Where, the imaginary part of the conductance contributes reactively. In cases of multiple dielectric materials surrounding the system, the complex loss conductance can be extended as linear combinations of multiple dipole moments as,

Equation 41 $$G\omega = Go + \sum_k Gd_k \frac{j\omega}{j\omega + \omega_{pk}}$$

Since Eq. 41 satisfies the Krong-Kramers condition, we can ensure the passivity/causality of the system. Note that when this new model is activated, the definition of Gd changes from conventional [S/m*Hz] to [S/m].

Fitting Procedure Triggered by INCLUDEGDIMAG Keyword

A fitting procedure is provided to generate a complex dielectric model with as close behavior as possible to the conventional pure real loss conductance model while preserving passivity. The `INCLUDEGDIMAG` keyword is the trigger to activate the new complex dielectric loss model. When the model is activated with conventional Go/Gd input with `INCLUDEGDIMAG=yes` without polarization constant (wp) input, the W-element automatically generates the new model by fitting.

In this fitting process, the W-element automatically computes wp and Gd values for the Eq. 41 where the real part of the function fits with conventional pure real dielectric loss model, $\mathrm{G}(f) = \mathrm{Go} + f\mathrm{Gd}$. Then the imaginary part of derived model contributes to the frequency dependency of the capacitance.

Because the model ensures causality, frequency domain and time domain responses maintain better consistency (see Figure 25). Also for passive transfer functions, functional overhead of the `DELAYOPT=3` is reduced. Thus, performance of the `DELAYOPT` function is improved.

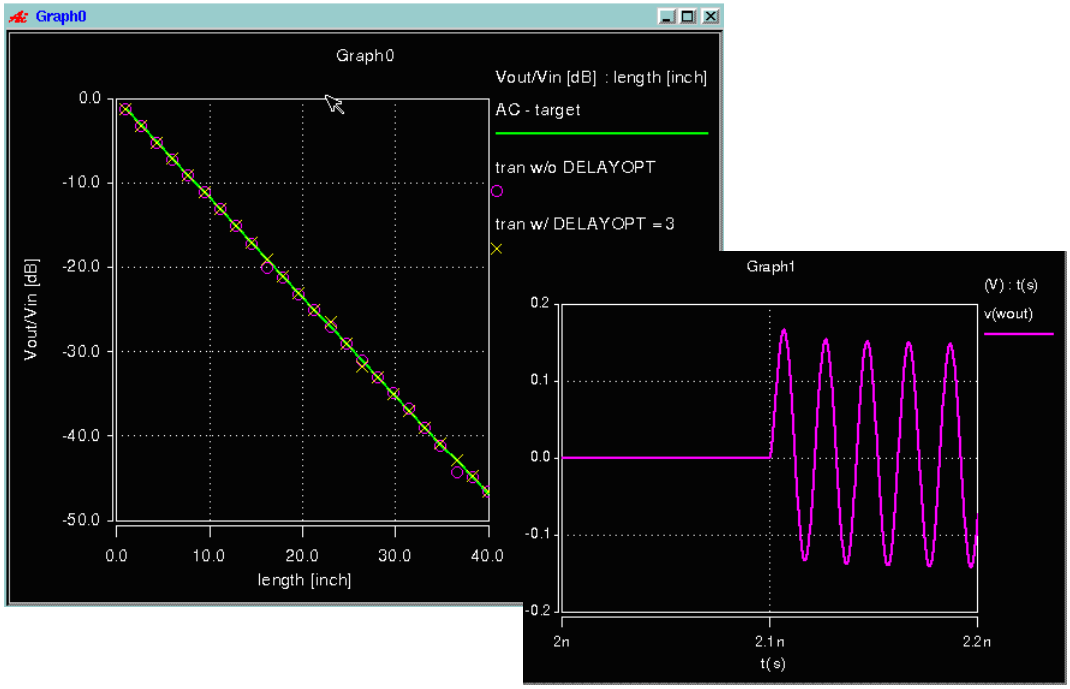


Figure 25 Improved consistency using the INCLUDEGDIMAG keyword

Example 1

This example shows `INCLUDEGDIMAG=yes` with polarization constant (`wp`) input.

```
Wtest win 0 wout 0 N=1 RLGCMODEL=WE1 L=0.3
+ INCLUDEGDIMAG=yes
.MODEL WE1 W MODELTYPE=RLGC, N=1
+ Lo = 3.8e-07
+ Co = 1.3e-10
+ Ro = 2.74e+00
+ Go = 0.0
+ Rs = 1.1e-03
+ Gd = 0.07
+ wp= 0.07
```

Example 2

This example shows `INCLUDEGDIMAG=yes` without polarization constant input.

```
Wtest win 0 wout 0 N=1 RLGCMODEL=WE1 L=0.3
+ INCLUDEGDIMAG=yes
.MODEL WE1 W MODELTYPE=RLGC, N=1
+ Lo = 3.8e-07
+ Co = 1.3e-10
+ Ro = 2.74e+00
+ Go = 0.0
+ Rs = 1.1e-03
+ Gd = 8.2e-12
```

To set this keyword as a global option for all W-elements in a netlist, see .OPTION WINCLUDEGDIMAG in the *HSPICE Reference Manual: Commands and Control Options*

Determining Matrix Properties

All matrices in Frequency-Dependent Matrices on page 98 are symmetric.

- The diagonal terms of L and C are positive, non-zero.
- The diagonal terms of R_o, R_s, G_o, and G_d are non-negative (can be zero).
- Off-diagonal terms of the L, R_o impedance matrices are non-negative.

R_o can have negative off-diagonal terms, but a warning appears. Negative off-diagonal terms normally appear when you characterize R_o at a frequency higher than zero. Theoretically, R_o should not contain negative off-diagonal terms, because these might cause errors during analysis.

- Off-diagonal terms of admittance matrices C, G_o, and G_d are non-positive.
- Off-diagonal terms of all matrices can be zero.

The elements of admittance matrices are related to the self/mutual admittances (such as those that the U-element generates):

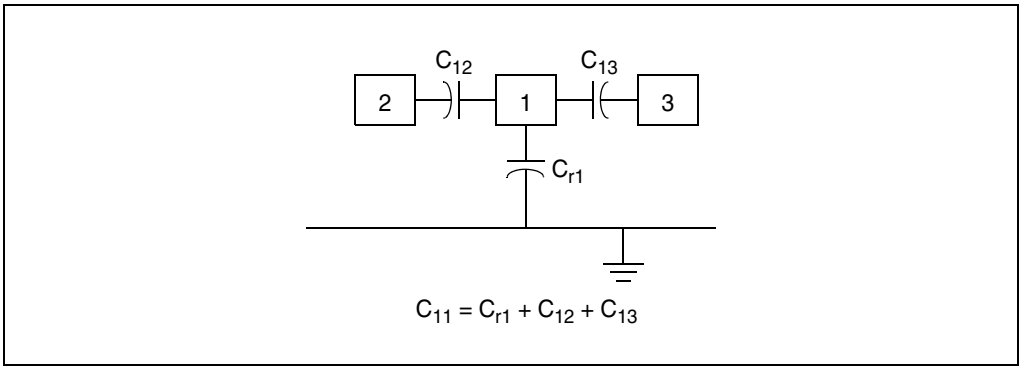
Equation 42
$$Y_{ii} = Y_{ri}^{(self)} + \sum_{k(k \neq i)} Y_{ki}^{(mutual)}$$

Equation 43
$$Y_{ij} = -Y_{ij}^{(mutual)} \quad (i \neq j)$$

In the preceding equations, *Y* stands for either C, G_o, or G_d.

A diagonal term of an admittance matrix is the sum of all self and mutual admittance in this row. This term is larger (in absolute value) than the sum of all off-diagonal terms in its row or column. Admittance matrices are strictly diagonally dominant (except for a zero matrix).

For example, diagonal terms for capacitance matrix can be expressed as shown in this figure:



You can obtain loop impedance matrix terms from the partial impedance matrix:

Equation 44 $$Z_{ij}^{(loop)} = Z_{ij}^{(partial)} - Z_{io}^{(partial)} - Z_{jo}^{(partial)} + Z_{oo}^{(partial)}$$

In the preceding equation, the o index denotes a reference node.

Using the PRINTZO Option

The PRINTZO option outputs the W-element complex characteristic impedance matrix to a *.wzo* file. For simplicity, since the W-element is a symmetric system, the Zo matrix is a symmetric matrix. Therefore, HSPICE only outputs the lower half of the matrix. See File Description for *.wzo and the following sections for discussion of the **.wzo* file. For example, the following frequency sweep example shows the use of the PRINTZO option with the W-element to check for characteristic impedance.

Input:

```
W1 N=2 in1 in2 gnd out1 out2 gnd RLGCMODEL=2_line l=0.1
+PRINTZO=POI 3 1e6 1e9 1e12
```

Output to be stored in *2_line.wzo*

```
* w-element model [2_line] Characteristic Impedance Matrix:
.MODEL ZO SP N=2 SPACING=POI MATRIX=SYMMETRIC
+ DATA=3
+ 1.0e6
+ 175.362 -156.577
+ 3.54758 -2.53246   175.362 -156.577
+ 1.0e9
+ 48.7663 -1.3087
+ 1.69417 -0.0073233 48.7663 -1.3087
+ 1e12
+ 48.9545 0.238574
+ 1.66444 0.0348332  48.9545 0.238574
```

The following example shows a PRINTZO statement with the MIXEDMODE option enabled. The syntax is:

```
Wxxx ni1 ni2...ref_in no1 no2...ref_out
```

Mixed Mode Example

```
W1 N=2 in1 gnd out1 out2 gnd RLGMODEL=2_line 1=0.1
+PRINTZO=POI 3 1e6 1e9 1e12
+MIXEDMODE=1
```

Output is stored in *2_line.wzo*

Printing Frequency-Dependent Impedance in Mixed Mode

This section discusses the HSPICE ability to print out complex characteristic impedance matrix in differential and common mode at given frequency points. This functionality supports high speed network designs, where differential data transfer systems are commonly used to achieve higher data transfer rate with low loss. For such designs, impedance information in mixed (differential and common) mode is more useful than single-ended representation.

Note: To learn more about the SP model syntax which has a complex number matrix by default, refer to Small-Signal Parameter Data Frequency Table Model (SP Model) on page 73.

The following provides a choice to output transmission line characteristic impedance in mixed mode.

For the ideal lossless transmission line system, the characteristic impedance becomes a frequency-independent constant matrix which is given as

Equation 45 $$Zo = (\mathrm{L} \cdot \mathrm{C}^{-1})^{1/2}$$

where, L and $""\mathrm{C}$ are inductance and capacitance matrices of the system, respectively, and in this case, the characteristic impedance is a real matrix. When the system becomes lossy, i.e., the system has non-zero resistance, R, and/or non-zero shunt loss conductance, G. Characteristic impedance becomes a function of frequency, ω, which can be expressed as,

Equation 46 $$Zo = ((\mathrm{R} + j\omega\mathrm{L}) \cdot (\mathrm{G} + j\omega\mathrm{C})^{-1})^{1/2}.$$

In this case, Zo becomes a complex matrix. Knowing characteristic impedance (Zo) matrix of the transmission line system at given frequency point is important for circuit designers to be able to establish well matched signal transfer condition to preserve integrity of the system, especially for high frequency operation. This feature allows users to check the complex characteristic impedance matrix of the system.

Note: The PRINTZO function does not evaluate Eq. 46 but it obtains Zo directly from the W-element's AC model. It does this so you can get $Zo(f)$ from other types of W-element models as well as S-parameter models.

In cases of lossless transmission line structure, the PRINTZO result may differ from Eq. 46 at very low frequencies because he RLGC-based lossless W-element adds a small amount of loss in the very low frequency range when it initializes the AC model. The effect of this on your actual AC or transient simulation is negligible but is important to achieve stable simulation.

The HSPICE W-element creates its own frequency-dependent characteristics when it is constructed based on RLGC parameters (RLGC or RLGC table model), structural (field solver) mode, U-element model, or scattering (S-parameter) model. By using the keyword `PRINTZO` to specify frequency point, users can compute the characteristic impedance not only from the RLGC model but also from any other of W-element configurations.

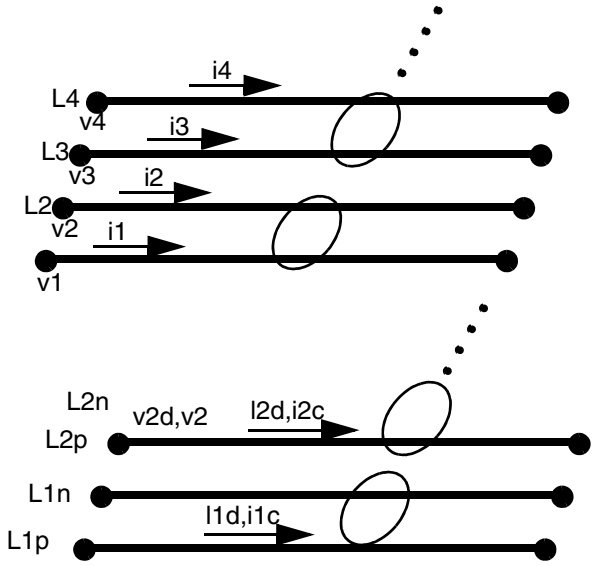


Figure 26 Definition of mixed-mode impedance and derivation from single-ended impedance

Differential and Common voltage are defined as,

Equation 47

where, $\mathbf{M}_V$ and $\mathbf{M}_i$ are voltage and current transformation matrices. Both single-ended and mixed mode representations satisfy relationships of voltage and current vector through characteristic impedance matrices as,

$$\mathbf{V}_{mixed} = \begin{pmatrix} v_{d1} \\ v_{c1} \\ v_{d2} \\ v_{c2} \\ . \\ . \\ . \end{pmatrix} = \begin{pmatrix} v_1 - v_2 \\ (v_1 + v_2)/\ 2 \\ v_3 - v_4 \\ (v_3 + v_4)/\ 2 \\ . \\ . \\ . \end{pmatrix} = \begin{pmatrix} 1 & -1 & 0 & 0 & \ldots \\ 1/\ 2 & 1/\ 2 & 0 & 0 & \ldots \\ 0 & 0 & 1 & -1 & \ldots \\ 0 & 0 & 1/\ 2 & 1/\ 2 & \ldots \\ . & . & . & . & \ldots \\ . & . & . & . & \ldots \\ . & . & . & . & \ldots \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ . \\ . \\ . \end{pmatrix} = \mathbf{M}_V \mathbf{V}$$

$$\mathbf{I}_{mixed} = \begin{pmatrix} i_{d1} \\ i_{c1} \\ i_{d2} \\ i_{c2} \\ . \\ . \\ . \end{pmatrix} = \begin{pmatrix} (i_1 - i_2)/\ 2 \\ i_1 + i_2 \\ (i_3 - i_4)/2 \\ i_3 + v i_4 \\ . \\ . \\ . \end{pmatrix} = \begin{pmatrix} 1/\ 2 & -1/\ 2 & 0 & 0 & \ldots \\ 1 & 1 & 0 & 0 & \ldots \\ 0 & 0 & 1/2 & -1/2 & \ldots \\ 0 & 0 & 1 & 1 & \ldots \\ . & . & . & . & \ldots \\ . & . & . & . & \ldots \\ . & . & . & . & \ldots \end{pmatrix} = \begin{pmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \\ . \\ . \\ . \end{pmatrix} = \mathbf{M}_I \mathbf{I}$$

Equation 48

$$\mathbf{V} = \mathbf{Zo} \cdot \mathbf{I}$$
$$\mathbf{V}_{mixed} = \mathbf{Zo}_{mixed} \cdot \mathbf{I}_{mixed}$$

Substituting Eq. 47 for Eq. 48, mixed mode characteristic impedance can be related to the single-ended one as,

Equation 49

$$\mathbf{M}_V \cdot \mathbf{V} = Zo_{mixed} \cdot \mathrm{M}_I \cdot \mathrm{I}$$
$$\mathbf{V} = \mathbf{M}_V^{(-1)} \cdot \mathbf{Zo}_{mixed} \cdot \mathbf{M}_I \cdot \mathbf{I}$$

thus,

Equation 50

$$\mathbf{Zo}_{mixed} = \mathbf{M}_V \cdot \mathbf{Zo} \cdot \mathbf{M}_I^{-1}$$

For example, for a system with one differential pair of lines, the transformation matrix would be:

Equation 51

$$\mathbf{M}_V = \begin{pmatrix} 1 & -1 \\ 1/\ 2 & 1/\ 2 \end{pmatrix}, \ \mathbf{M}_I = \begin{pmatrix} 1/\ 2 & -1/\ 2 \\ 1 & 1 \end{pmatrix}$$

Therefore, mixed mode characteristic impedance is expressed as,

Equation 52

$$\mathbf{Zo}_{mixed} = \mathbf{M}_V \cdot \mathbf{Zo} \cdot \mathbf{M}_I^{-1}$$

$$= \frac{1}{4}\begin{pmatrix} 2 & -2 \\ 1 & 1 \end{pmatrix} \cdot \mathbf{Zo} \cdot \begin{pmatrix} 2 & 1 \\ -2 & 1 \end{pmatrix}$$

$$= \frac{1}{4}\begin{pmatrix} 2 & -2 \\ 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{pmatrix} \cdot \begin{pmatrix} 2 & 1 \\ -2 & 1 \end{pmatrix}$$

$$= \frac{1}{4}\begin{pmatrix} 4Z_{11}-4Z_{12} & -4Z_{21}+4Z_{22} & 2Z_{11}+2Z_{12}-2Z_{21} & -2Z_{22} \\ 2Z_{11}-2Z_{12} & +2Z_{21}-2Z_{22} & Z_{11}+Z_{12} & +Z_{21} & +Z_{22} \end{pmatrix} = \begin{pmatrix} Z_{dd} & Z_{dc} \\ Z_{cd} & Z_{cc} \end{pmatrix}$$

Here, Z_{dd} is called differential (mode) impedance and Z_{cc} is called common mode impedance. Differential impedance is useful for designers to check matching characteristics of differential signal transfer systems.

Typically, for a symmetric two-line structure with weak coupling, single-ended characteristic impedance matrix components become $Z_{11} = Z_{22} = Z_{self}(\sim 50\Omega)$, $Z_{12} = Z_{21} = Z_{mutual}(\sim 0)$. Therefore, mixed-mode characteristic impedance is,

Equation 53

$$\begin{pmatrix} Z_{dd} & Z_{dc} \\ Z_{cd} & Z_{cc} \end{pmatrix} = \begin{pmatrix} 2Z_{self} - 2Z_{mutual} & 0 \\ 0 & \frac{1}{2}(Z_{self} + Z_{mutual}) \end{pmatrix} = \begin{pmatrix} 100\Omega & 0 \\ 0 & 25\Omega \end{pmatrix}$$

File Description for *.wzo

The **.wzo* file is created when the PRINTZO option is included in a W-element statement. The prefix for this file is the W-element model name.

```
W1 N=2 i1 i2 gnd o1 o2 gnd RLGCMODEL=line l=10m PRINTZO=POI 2
10meg 10g
+ mixedmode=1   $ 1= mixed mode  0= common mode
```

As shown in Eq. 53, the file line *.wzo* is created for this 2 conductor W-element. Two impedance matrices are created, one at 10MHz and the other at 10GHz.

The *.wzo* file contains complex impedances in the form R + jX, where the first term of each pair is R and the second is X. The off-diagonal (negative) terms relate to the interactions between conductors (common mode). For a single-ended analysis, the diagonal terms are identical.

For the mixed-mode configuration, the [1,1] term is the differential mode impedance, and the [2,2] term is the common mode impedance. It is typical for the matrices to show a 3 or 4 to 1 ratio of the [1,1] to the [2,2] terms. When the conductors are loosely coupled, the ratio will be close to 4:1. This is because:

```
Z11 = Z22 = Zself = 50   and   Z12 = Z21 = Zmutual = 0
```

Wave Propagation

To illustrate the physical process of wave propagation and reflection in transmission lines, Figure 27 on page 110 shows lines where the voltage step excites simple termination.

- At time t=$t1$, a voltage step from the e_1 source, attenuated by the $Z1$ impedance, propagates along the transmission line.
- At t=$t2$, the voltage wave arrives at the far end of the transmission line, is reflected, and propagates in the backward direction. The voltage at the load end is the sum of the incident and reflected waves.
- At t=$t3$, the reflected wave arrives back at the near end, is reflected again, and again propagates in the forward direction. The voltage at the source end is the sum of attenuated voltage from the e_1 source, the backward wave, and the reflected forward wave.

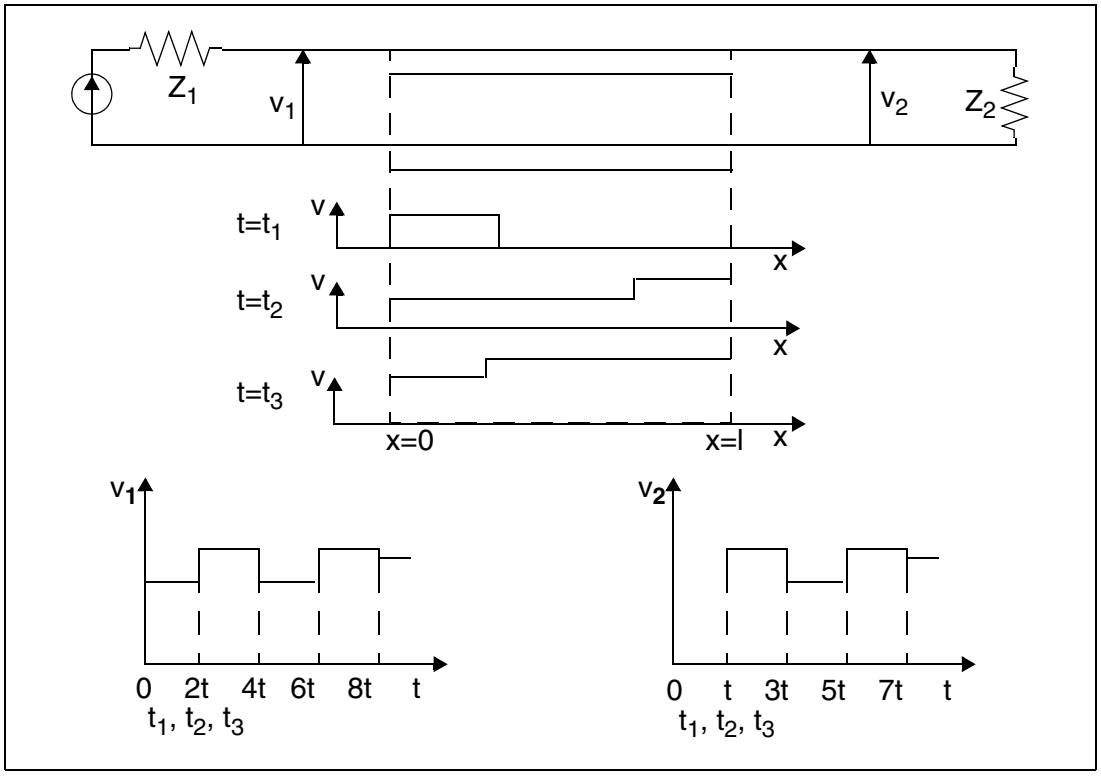


Figure 27 Propagation of a Voltage Step in a Transmission Line

The surface plot in Figure 28 on page 111 shows voltage at each point in the transmission line. The input incident propagates from the left (length = 0) to the right. You can observe both reflection at the end of the line (length = 1), and a reflected wave that goes backward to the near end.

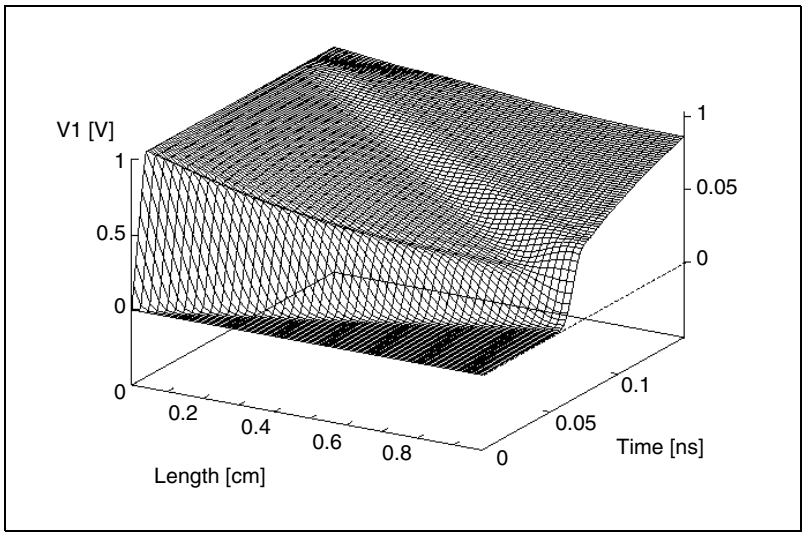


Figure 28 Surface Plot for the Transmission Line Shown in Figure 27 on page 110

You can find more information about transmission lines in this resource: H.B. Bakoglu, *Circuits, Interconnections and Packaging for VLSI*. Reading, MA: Addison-Wesley, 1990.

The following sections discuss these topics:

- Propagating a Voltage Step
- Handling Line-to-Line Junctions

Propagating a Voltage Step

This section is a summary of the process in Figure 27 on page 110 to propagate a voltage step in a transmission line.

- Signals from the excitation source spread-out in the termination networks, and propagate along the line.
- As the forward wave reaches the far-end termination, it does the following:
  - Reflects.
  - Propagates backward.
  - Reflects from the near-end termination.
  - Propagates forward again.

- Continues in a loop.

- The voltage at any point along the line, including the terminals, is a superposition of the forward and backward propagating waves.

Figure 29 on page 112 shows the system diagram for this process, where:

- W_{vr} and W_{vb} are forward and backward matrix propagation functions for voltage waves.
- T_1, T_2 stand for the near-end matrix transmission and reflection coefficients.
- $\Gamma_1 \Gamma_2$, (Gamma_1,Gamma_2) stand for the far-end matrix transmission and reflection coefficients.

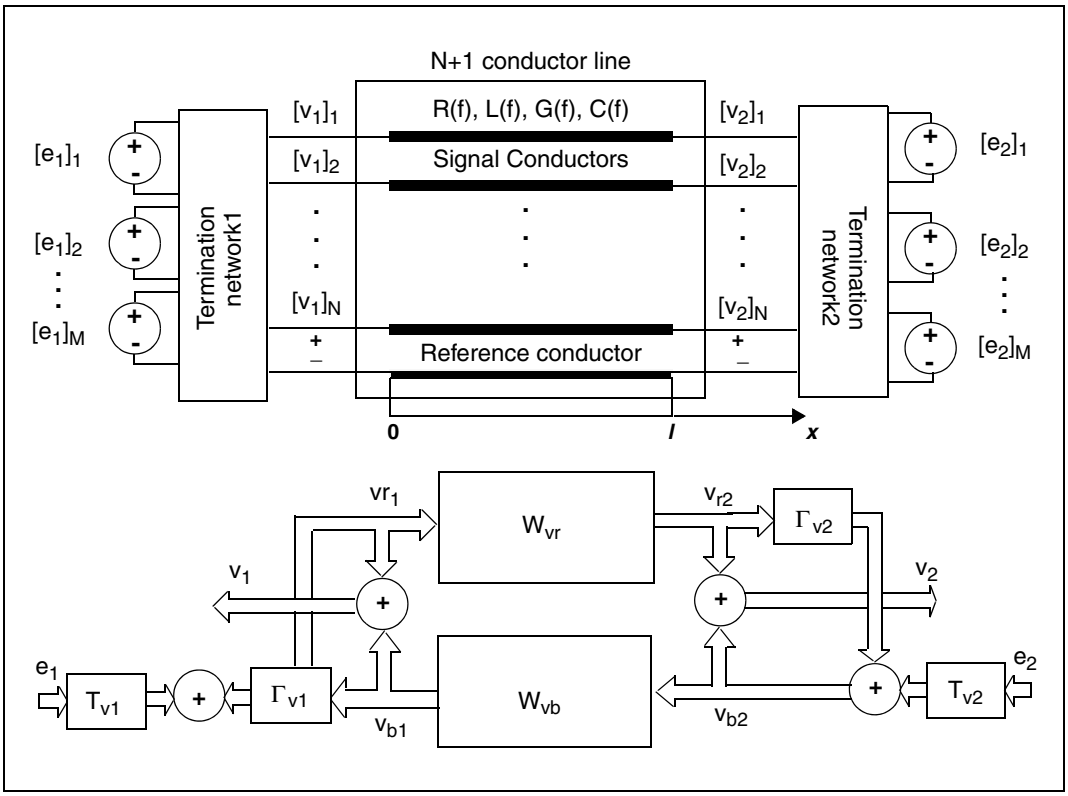


Figure 29 System Model for Transmission Lines

This model reproduces the general relationship between the physical phenomena of wave propagation, transmission, reflection, and coupling in a distributed system. It can represent an arbitrarily-distributed system, such as:

- Transmission line
- Waveguide
- Plane-wave propagation

You can use this model for:

- System analysis of distributed systems, or
- Writing a macro solution for a distributed system without complicated mathematical derivations.

As shown in the figure, transmission lines and terminations form a feedback system. Because the feedback loop contains a delay, both the phase shift, and the sign of the feedback change periodically with the frequency. This causes oscillations in the frequency-domain response of the transmission lines, such as those shown in Figure 35 on page 129.

Handling Line-to-Line Junctions

A special case occurs when the line terminates in another line. Figure 30 shows the system diagram for a line-to-line junction. Use this diagram to:

- Solve multi-layered plane-wave propagation problems.
- Analyze common waveguide structures.
- Derive generalized transmission and reflection coefficient formulas.
- Derive scattering parameter formulas.

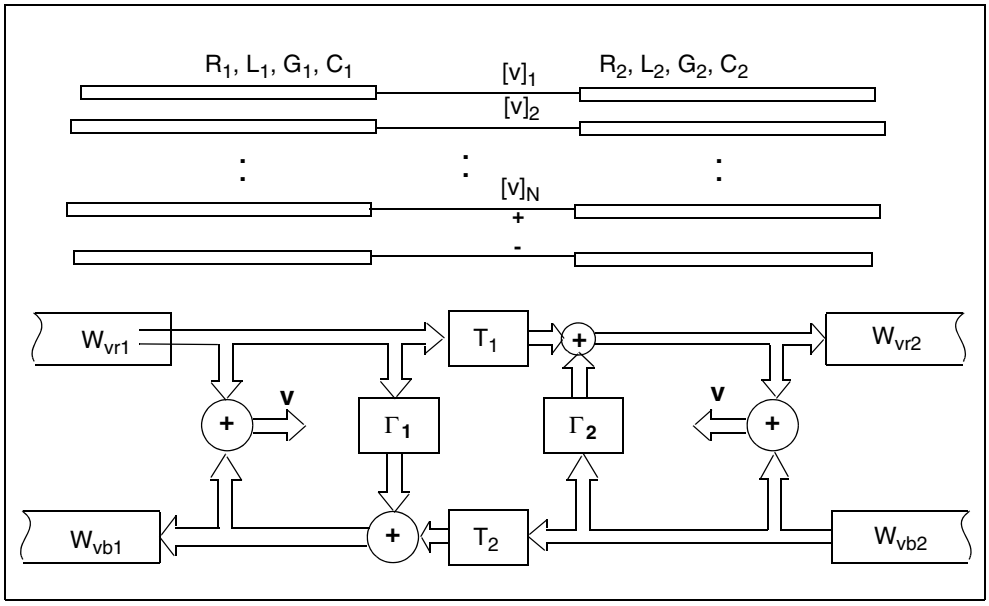


Figure 30 System Model for a Line-to-Line Junction

The W_{vr} and W_{vb} propagation functions describe how propagation (from one termination to another) affects a wave. These functions are equal for the forward (W_{vr}) and backward (W_{vb}) directions. The off-diagonal terms of the propagation functions represent the coupling between conductors of a multi-conductor line.

As a wave propagates along the line, it experiences delay, attenuation, and distortion (see Figure 31). Lines with frequency-dependent parameters (that is, all real lines) do not contain the frequency-independent attenuation component.

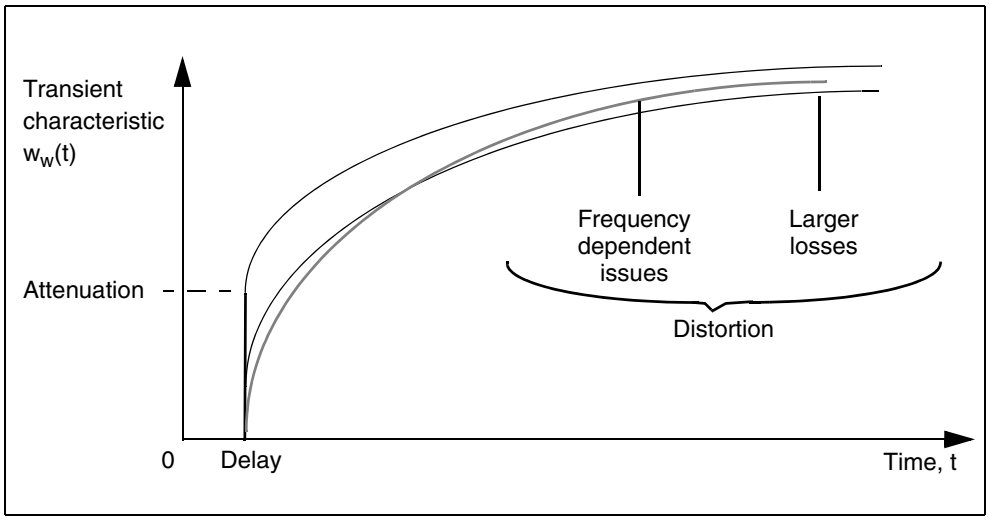


Figure 31 Propagation Function Transient Characteristics (unit-step response)

Using the W-element

The following topics are covered in this section:

- W-element Capabilities
- Control Frequency Range of Interest for Greater Accuracy
- Setting .OPTION RISETIME
- Using DELAYOPT Keyword for Higher Frequency Ranges
- Using DCACC Keyword for Lower Frequency Ranges
- W-element Time-Step Control in Time Domain
- Time-Step Control
- Input Syntax for the W-element
- Input Model 1: W-element, RLGC Model
- Input Model 2: U-element, RLGC Model
- Input Model 3: Built-in Field-Solver Model
- Input Model 4: Frequency-Dependent Tabular Model
- Input Model 5: S Model

The following section discuss these topics:

- W-element Capabilities
- Control Frequency Range of Interest for Greater Accuracy
- W-element Time-Step Control in Time Domain
- Input Syntax for the W-element
- Input Model 1: W-element, RLGC Model
- Input Model 2: U-element, RLGC Model
- Input Model 3: Built-in Field-Solver Model
- Input Model 4: Frequency-Dependent Tabular Model
- Input Model 5: S Model

W-element Capabilities

The W-element is a multi-conductor lossy frequency-dependent transmission line. It provides advanced modeling capabilities for transmission lines. The W-element provides:

- Ability to extract analytical solutions for AC and DC.
- No limit on the number of coupled conductors.
- No restriction on the structure of RLGC matrices; all matrices can be full.
- No spurious ringing, such as is produced by the lumped model. (See Figure 32 on page 117.)
- Accurate modeling of frequency-dependent loss in the transient analysis.
- Built-in 2D field solver, which you can use to specify a physical line shape.

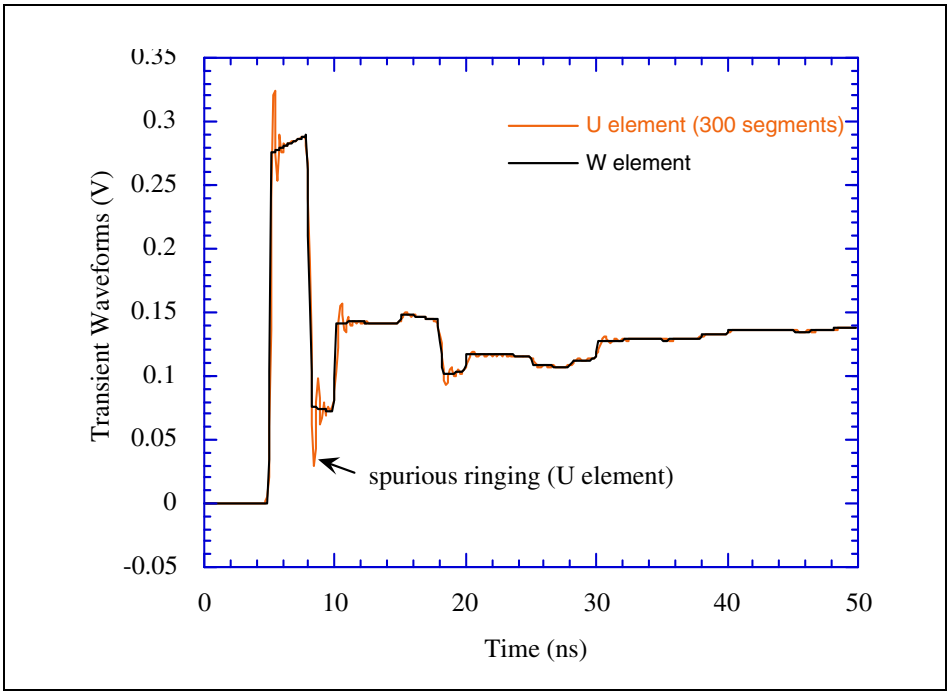


Figure 32 Spurious Ringing in U-element

The W-element supports the following types of analysis:

- DC
- AC
- Transient
- RF analyses (HB, HBAC, HBNOISE, PHASENOISE, LIN)
- Parameter sweeps
- Optimization
- Monte-Carlo

Control Frequency Range of Interest for Greater Accuracy

This section describes the keywords you can use for achieving greater accuracy of the W-element by controlling the frequency of interest. The following sections discuss these topics:

Setting .OPTION RISETIME

The RISETIME option is used to improve accuracy of the W-element analysis by setting the focal frequency. It is applicable to all W-element modeling methods. If not explicitly set, HSPICE automatically determines RISETIME and the focal frequency from independent sources by examining their edge rates and using the fastest of those as the effective RISETIME setting.

If you know the focal frequency, you can explicitly set RISETIME. The focal frequency is user-determined, but the knee frequency of the design is typically a valid approximation. Use the reciprocal of the focal frequency to set RISETIME; e.g. 1/10GHz = 100p. The minimum value for RISETIME is 0.1p and anything smaller defaults to that value. Do not set RISETIME to a value of 0 (zero).

Be cautious when setting RISETIME as it can have a direct impact on transient simulation results. If it is not properly set (too low or too high compared to the maximum frequency contained in the actual signal), responses to the high frequency components might be omitted or inaccurately modeled.

It is generally a good idea to explicitly set RISETIME, especially if you are using behavioral (controlled) sources or Verilog-A sources that do not have edge rates that can be determined by statically parsing the netlist.

You may want to let HSPICE determine RISETIME under the following conditions:

- The sources are independent and not subject to unexpected incidents that modify the predicted edge rates.
- You don't want to risk forgetting to reset RISETIME when you modify your sources.
- You share the netlist others who may use it without knowing that the RISETIME is set such as in the case of intellectual property.
- There is a potential for making a mistake in RISETIME estimation.

Using DELAYOPT Keyword for Higher Frequency Ranges

Long transmission lines fabricated in a high polymer insulator, such as PCB traces, show high losses in high frequencies due to dielectric loss. In such cases, the propagation delay of the system becomes a non-constant function of frequency. To take this phenomenon accurately, beginning with the 2003.09 release of HSPICE, a novel pre-process function was introduced for constructing W-element transient (recursive convolution) model with a higher level of accuracy. To activate this new function, you can add the `DELAYOPT`

keyword to the W-element instance line. You can use `DELAYOPT=0|1|2` to deactivate, activate, and automatically determine, respectively. The default value is `0` (deactivate). If this function is deactivated, the W-element behaves identically to the previous versions.

You can use `DELAYOPT=3` to achieve a level of accuracy up to a tens of GHz operation and involve harmonics up to THz order. With this option, line length limits are removed, which frees the simulation from segmenting, and allows independence in the behavior of the `RISETIME` option setting. A setting of `DELAYOPT=3` automatically detects whether or not frequency-dependent phenomena need to be recorded, which makes it identical to the `DELAYOPT=0` setting if it produces a high enough accuracy.

Note: The `DELAYOPT=3` option activates additional evaluation functions in transient analysis, which might take longer CPU time.

To set this parameter as a global option for all W-elements in a netlist, see .OPTION WDELAYOPT in the *HSPICE Reference Manual: Commands and Control Options*.

Using DCACC Keyword for Lower Frequency Ranges

The W-element can take an additional step in making a time domain model check the accuracy of low frequency and DC coverage. It automatically adds rational function terms if necessary. This process may cause slight additional computational cost and slight difference in element behavior in DC offset. Should you choose to use this conventional behavior, set `DCACC=0` in the W-element instance or model line to deactivate this process.

W-element Time-Step Control in Time Domain

This section describes using static and dynamic time-step controls in the time domain.

The following sections discuss these topics:

- Time-Step Control
- Using Dynamic Time-Step Control

Time-Step Control

The W-element provides accurate results with just one or two time steps per excitation transient (0.1 ns in Figure 32 on page 117). Like the T-element, the W-element supports the `TLINLIMIT` parameter. The `TLINLIMIT=0` default setting enables special breakpoint building, which limits the maximum time step by the smallest transmission line delay in the circuit. This improves transient accuracy for short lines, but reduces efficiency. Setting `TLINLIMIT=1` disables this special breakpoint building.

Longer transmission lines might experience prolonged time intervals when nothing happens at the terminals, while the wave propagates along the line. If you increase the time step, the accuracy of the simulation decreases when the wave reaches the terminal. To prevent this for longer lines excited with short pulses, set `.OPTION DELMAX` to limit the time step to between 0.5 and 1 of the excitation transient.

Using Dynamic Time-Step Control

Static time step control achieves certain accuracy by setting static breakpoints. The `TLINLIMIT=0` parameter limits the maximum time step by the minimum transmission line delay, which results in poor performance for cases with ultra-short delay transmission lines because too many redundant time points are calculated, especially when the transmission line terminal signals do not vary rapidly. The same problem exists with the `DELMAX` option where time steps are evenly set in spite of terminal signal variation. This is inefficient.

In the 2004.09 release, the `WACC` option was added to solve this problem by providing dynamic step control for W-element transient analysis. Setting `WACC` to a positive value removes the static breakpoints and the necessary time points are set dynamically according to the variations in terminal currents and voltages.

The `WACC` option has the following syntax:

`.OPTION WACC=`*`value`*

...where WACC is a non-negative real value between 0.0 and 10.0. HSPICE assigns `WACC -1` if you do not set a `WACC` option, or if you set `.OPTION WACC`. When a value of 1 is specified, HSPICE assigns `WACC` a positive value. If a non-negative value is set in the `.option` line (`.OPTION WACC=XXX`), HSPICE uses the specified `WACC` value for all the W-elements.

When a positive WACC value is set, the dynamic time step control algorithm is activated. When WACC is zero, the conventional static time step control

method is used. Larger WACC values result in less restriction in time point intervals therefore faster simulation), while smaller values result in denser time points with higher accuracy.

Since the 2006.09 release, positive WACC is selected by default to activate the dynamic time step control. HSPICE automatically finds the optimum WACC value based on the netlist properties such as transmission line system delay, risetime, and transient command configurations. Since the W-elements in the netlist may have different properties, each has its own WACC values. If a user-specified positive WACC value is found in the netlist, HSPICE uses the user-defined WACC value for all the W-elements in the netlist. If the user-specified WACC is larger than the automatic estimation, HSPICE outputs a warning message.

For cases containing IBIS, PKG, EBD, or ICM blocks, HSPICE turns WACC off automatically. If you want to use the dynamic time step control algorithm for IBIS related cases, you must set it explicitly in the netlist for example:

```
.option WACC      $ Make HSPICE use automatically generated WACC
                    value for each W element
```

or

```
.option WACC=value      $ Use this value for all the W elements
```

Input Syntax for the W-element

Syntax:

```
Wxxx i1 i2 ... iN iR o1 o2 ... oN oR N=val L=val
+ [RLGCMODEL=name | RLGCFILE=name | UMODEL=name
+ FSMODEL=name | TABLEMODEL=name | SMODEL=name]
+ [ INCLUDERSIMAG=YES|NO FGD=val ] [ DELAYOPT=0|1|2|3 ]
+ [ INCLUDEGDIMAG=YES|NO] [NODEMAP=XiYj[DCACC=[1|0]]
+ [NOISE=[1|0]] [DTEMP=val]
+ [PRINTZO=frequency_sweep MIXEDMODE=0|1]
+ [SCALE_RS=val]
```

Parameter	Description
N	Number of signal conductors (excluding the reference conductor).
i1...iN	Node names for the near-end signal-conductor terminal (Figure 33 on page 125).

Parameter	Description
iR	Node name for the near-end reference-conductor terminal.
o1... oN	Node names for the far-end signal-conductor terminal (Figure 33 on page 125).
oR	Node name for the far-end reference-conductor terminal.
L	Length of the transmission line.
RLGCMODEL	Name of the RLGC model.
RLGCFILE	Name of the external file with RLGC parameters. A RLGC file name must start with an alphabetic letter (not a number). (See Input Model 1: W-element, RLGC Model on page 125.)
UMODEL	Name of the U model. (See Input Model 2: U-element, RLGC Model on page 132.)
FSMODEL	Name of the field solver model. See Using the Field Solver Model on page 147
TABLEMODEL	Name of the frequency-dependent tabular model. See Table Model Card Syntax on page 137
SMODEL	Name of the S model. (See Input Model 5: S Model on page 144.)
INCLUDERSIMAG	Imaginary term of the skin effect to be considered. The default value is YES. (See Frequency-Dependent Matrices on page 98.) This keyword activates the complex dielectric loss model and can operate with the DELAYOPT parameter (see Introduction to the Complex Dielectric Loss Model on page 99).

Parameter	Description
INCLUDEGDIMAG	Activates the complex dielectric loss model (see Fitting Procedure Triggered by INCLUDEGDIMAG Keyword on page 101). Gd: coefficient matrices of the frequency dependency wp: corresponding frequency value of the polarization time constants. If INCLUDEGDIMAG=yes and there is no wp input, the W-element regards the Gd matrix as the conventional model and then automatically extracts constants for the complex dielectric model. The INCLUDEGDIMAG keyword operates with the DELAYOPT parameter. To set this parameter as a global option for all W-elements in a netlist for either HSPICE or HSPICE RF, see .OPTION WINCLUDEGDIMAG in the *HSPICE Reference Manual: Commands and Control Options*.
FGD	Specifies the cut-off frequency of dielectric loss. (See Handling the Dielectric-loss Matrix on page 133.)
DELAYOPT	Deactivates (0), activates (1), determines automatically (2), or high frequency (3). The default is 0. To set this parameter as a global option for all W-elements in a netlist for either HSPICE or HSPICE RF, see .OPTION WDELAYOPT in the *HSPICE Reference Manual: Commands and Control Options*.
DCAAC	Deactivates(0), activates(1). The default is 1. An additional step to check the accuracy of low frequency and DC coverage.
NODEMAP	String that assigns each index of the S-parameter matrix to one of the W-element terminals. This string must be an array of pairs that consists of a letter and a number, (for example, Xn), where ▪ X= I, i, N, or n to indicate near end (input side) terminal of the W-element ▪ X= O, i, F, or f to indicate far end (output side) terminal of the W-element. The default value is NODEMAP = I1I2I3...InO1O2O3...On.
NOISE	Activates thermal noise. ▪ 1 (default): element generates thermal noise ▪ 0: element is considered noiseless

Parameter	Description
DTEMP	Temperature difference between the element and the circuit, expressed in °C. The default is 0.0. Element temperature is calculated as: T = Element temperature (°K) = 273.15 (°K) + circuit temperature (°C) + DTEMP (°C) Where circuit temperature is specified using either the .TEMP statement, or by sweeping the global TEMP variable in .DC, .AC, or .TRAN statements. When a .TEMP statement or TEMP variable is not used, the circuit temperature is set by .OPTION TNOM, which defaults to 25°C unless you use .OPTION SPICE, which raises the default to 27°C.
PRINTZO	Type of frequency sweep to allow checking of the complex characteristic impedance matrix of the system. You can specify any of LIN, DEC, OCT, or POI (see example Using the PRINTZO Option). Specify the nsteps, start, and stop values using the following syntax for each type of sweep: ■ LIN nsteps start stop ■ DEC nsteps start stop ■ OCT nsteps start stop ■ POI nsteps freq_values
MIXEDMODE	■ 0: Single-ended impedance is printed out (default) ■ 1: Output characteristic impedance is printed in mixed mode
SCALE_RS	RS matrix scaling factor, W-element instance

The W-element supports these formats to specify transmission line properties:

- Model 1: RLGC-Model specification
  - Internally specified in a `.MODEL` statement.
  - Externally specified in a different file.
- Model 2: U-Model specification
  - RLGC input for up to five coupled conductors
  - Geometric input (planer, coax, twin-lead)
  - Measured-parameter input
  - Skin effect
- Model 3: Built-in field solver model

- Model 4: Frequency-dependent tabular model.
- Model 5: S model specification
  - S-parameters specified by an S model
  - Valid only for transmission line-based S-parameters.

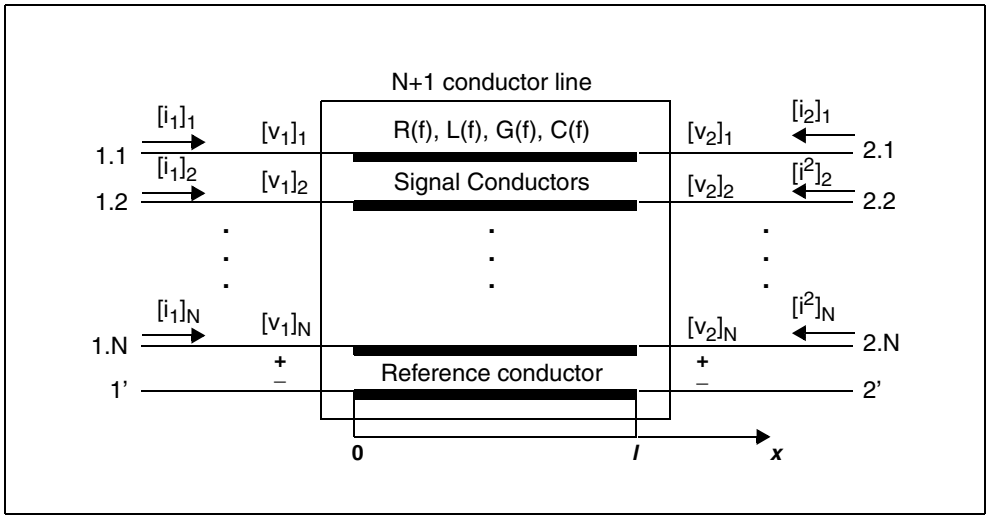


Figure 33 Terminal Node Numbering

Normally, you can specify parameters in the W-element card in any order. Specify the number of signal conductors, N, after the list of nodes. You can intermix the nodes and parameters in the W-element card.

You can specify only one `RLGCMODEL`, `FSMODEL`, `UMODEL`, or `RLGCFILE` in a single W-element card.

For demo files of the S Model usage see Transmission (W-element) Line Examples.

Input Model 1: W-element, RLGC Model

Equations and Parameters on page 96 describes the inputs of the W-element per unit length matrices: R_o (DC resistance), L, G, C, R_s (skin effect), and G_d (dielectric loss)

The W-element does not limit any of the following parameters:

- Number of coupled conductors.
- Shape of the matrices.
- Line loss.
- Length or amount of frequency dependence.

The RLGC text file contains frequency-dependent RLGC matrices per unit length. The W-element also handles frequency-independent RLGC, and lossless (LC) lines. It does not support RC lines.

Because RLGC matrices are symmetrical, the RLGC model specifies only the lower triangular parts of the matrices. The syntax of the RLGC model for the W-element is:

```
.MODEL name W MODELTYPE=RLGC N=val
+ Lo=matrix_entries
+ Co=matrix_entries [Ro=matrix_entries Go=matrix_entries]
+ Rs=matrix_entries wp=val Gd=matrix_entries Rognd=val
+ Rsgnd=val Lgnd=val
```

Parameter	Description
N	Number of conductors (same as in the element card).
L	DC inductance matrix, per unit length $\left[\frac{H}{m}\right]$.
C	DC capacitance matrix, per unit length $\left[\frac{F}{m}\right]$.
Ro	DC resistance matrix, per unit length $\left[\frac{\Omega}{m}\right]$.
Go	DC shunt conductance matrix, per unit length $\left[\frac{S}{m}\right]$.
Rs	Skin effect resistance matrix, per unit length $\left[\frac{\Omega}{m\sqrt{Hz}}\right]$.
Gd	Dielectric loss conductance matrix, per unit length $\left[\frac{S}{m \cdot Hz}\right]$.

Parameter	Description
wp	Angular frequency of the polarization constant [radian/sec] (see Introduction to the Complex Dielectric Loss Model on page 99). When the wp value is specified, the unit of Gd becomes [S/m].
Lgnd	DC inductance value, per unit length for grounds $\left[\frac{H}{m}\right]$ (reference line).
Rognd	DC resistance value, per unit length for ground $\left[\frac{\Omega}{m}\right]$.
Rsgnd	Skin effect resistance value, per unit length for ground $\left[\frac{\Omega}{m\sqrt{Hz}}\right]$.

The following input netlist file shows RLGC input for the W-element:

```
* W-Element example, four-conductor line
W1 N=3 1 3 5 0 2 4 6 0 RLGCMODEL=example_rlc l=0.97
V1 1 0 AC=1v DC=0v pulse(4.82v 0v 5ns 0.1ns 0.1ns 25ns)
.AC lin 1000 0Hz 1GHz
.DC v1 0v 5v 0.1v
.tran 0.1ns 200ns

* RLGC matrices for a four-conductor lossy
.MODEL example_rlc W MODELTYPE=RLGC N=3
+ Lo=
+ 2.311e-6
+ 4.14e-7 2.988e-6
+ 8.42e-8 5.27e-7 2.813e-6
+ Co=
+ 2.392e-11
+ -5.41e-12 2.123e-11
+ -1.08e-12 -5.72e-12 2.447e-11
+ Ro=
+ 42.5
+ 0 41.0 + 0 0 33.5
+ Go= + 0.000609
+ -0.0001419 0.000599
+ -0.00002323 -0.00009 0.000502
+ Rs=
+ 0.00135
+ 0 0.001303
+ 0 0 0.001064
+ Gd=
+ 5.242e-13
+ -1.221e-13 5.164e-13
+ -1.999e-14 -7.747e-14 4.321e-13
.end
```

The following three figures show plots of the simulation results:

- Figure 34 on page 129 shows DC sweep
- Figure 35 on page 129 shows AC response
- Figure 36 on page 130 shows transient waveforms.

These figures also demonstrate that the transmission line behavior of interconnects has a significant and complicated effect on the integrity of a signal. This is why it is very important to accurately model transmission lines when you verify high-speed designs.

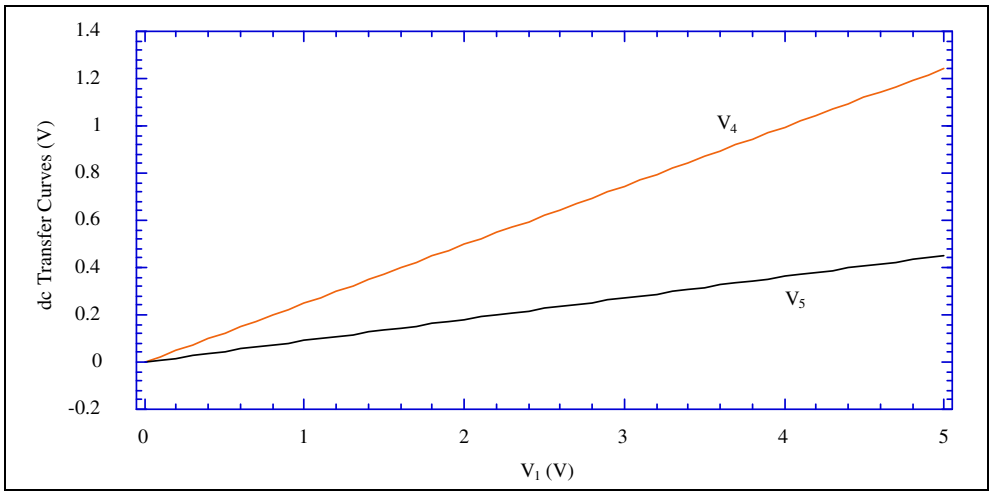


Figure 34 Simulation Results: DC Sweep

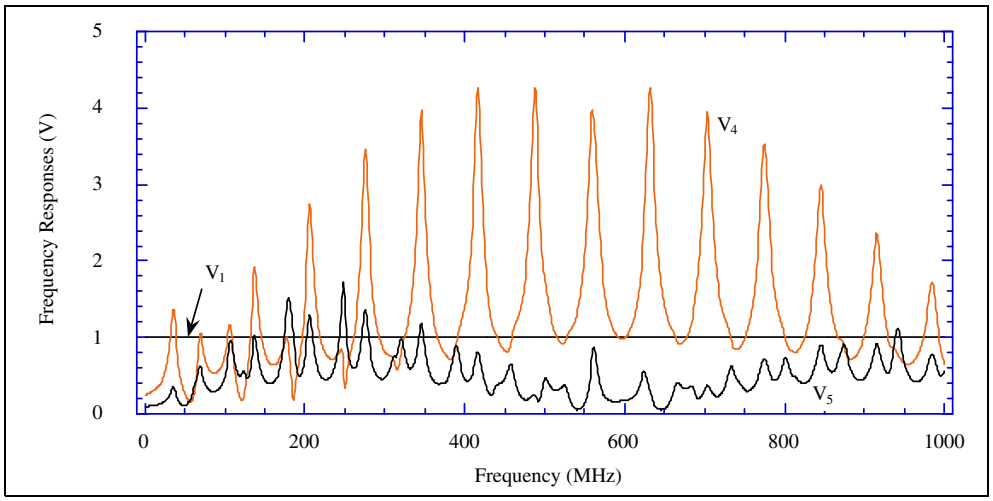


Figure 35 Simulation Results: AC Response

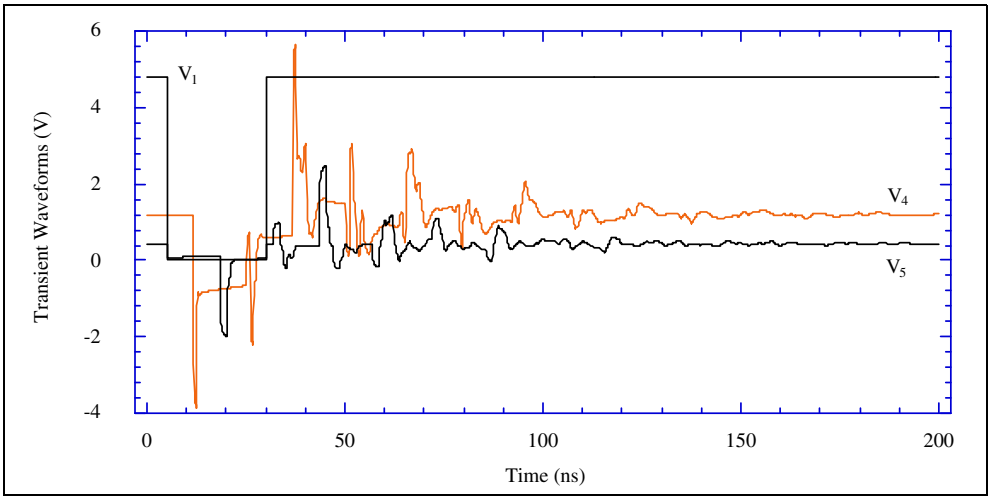


Figure 36 Simulation Results: Transient Waveforms

Specifying the RLGC Model in an External File

You can also specify RLGC matrices in a RLGC file. Its file format is more restricted than the RLGC model; for example:

- You cannot include any parameters.
- The file does not support ground inductance and resistance.

  Note: This format does not provide any advantage over the RLGC model so do not use it unless you already have an RLGC file. It is supported for backward-compatibility.

The RLGC file only specifies the lower-triangular parts of the matrices and is order-dependent. Its parameters are in the following order:

Table 5 Parameters in RLGC File for W-element

Parameter	Description
N	Number of conductors (same as in the element card).
L	DC inductance matrix, per unit length $\left[\frac{H}{m}\right]$.
C	DC capacitance matrix, per unit length $\left[\frac{F}{m}\right]$.

Table 5 Parameters in RLGC File for W-element

Parameter	Description
R_o	(Optional) DC resistance matrix, per unit length $\left[\frac{\Omega}{m}\right]$.
G_o	(Optional) DC shunt conductance matrix, per unit length $\left[\frac{S}{m}\right]$.
R_s	(Optional) Skin effect resistance matrix, per unit length $\left[\frac{\Omega}{m\sqrt{Hz}}\right]$.
G_d	(Optional) Dielectric loss conductance matrix, per unit length $\left[\frac{S}{m \cdot Hz}\right]$.

Note: You can skip the optional parameters, because they default to zero. But if you specify an optional parameter, then you must specify all preceding parameters, even if they are zero.

An asterisk (*) in an RLGC file comments out everything until the end of that line. You can use any of the following characters to separate numbers:

```
space tab newline , ; ( ) [ ] { }
```

This RLGC file is for the same netlist example used for the RLGC model in the previous section:

```
* W- Element example, four-conductor line

W1 N=3 1 3 5 0 2 4 6 0 RLGCfile=example.rlc l=0.97
V1 1 0 AC=1v DC=0v pulse(4.82v 0v 5ns 0.1ns 0.1ns 25ns)

.AC lin 1000 0Hz 1GHz
.DC v1 0v 5v 0.1v
.tran 0.1ns 200ns

.end
```

Calls this *example.rlc* file:

```
* RLGC parameters for a four-conductor lossy
* frequency-dependent line
* N (number of signal conductors)
3
* Lo
2.311e-6
4.14e-7 2.988e-6
8.42e-8 5.27e-7 2.813e-6
* Co
2.392e-11
-5.41e-12 2.123e-11
-1.08e-12 -5.72e-12 2.447e-11
* Ro
42.5
0 41.0
0 0 33.5
* Go
0.000609
-0.0001419 0.000599
-0.00002323 -0.00009 0.000502
* Rs
0.00135
0 0.001303
0 0 0.001064
* Gd
5.242e-13
-1.221e-13 5.164e-13
-1.999e-14 -7.747e-14 4.321e-13
```

The RLGC file format does not support scale suffixes, such as:

```
n (10^-9) or p (10^-12)
```

Input Model 2: U-element, RLGC Model

The W-element accepts the U Model as an input to provide backward compatibility with the U-element. It also uses the geometric and measured-parameter interfaces of the U model.

To use the W-element with the U Model on the W-element card, specify:

`Umodel=U-`*`model_name`*

The W-element supports all U model modes, including:

- geometric, `Elev=1`
  - planar geometry, `Plev=1`

  - coax, `Plev=2`
  - twin-lead, `Plev=3`
- RLGC, `Elev=2`
- measured parameters, `Elev=3`
- skin-effect, `Nlay=2`

The only exception is Llev=1, which adds the second ground plane to the U model. The W-element does not support this. To model the extra ground plane, add an extra conductor to the W-element in `Elev=2`, or use an external lumped capacitor in `Elev=1` or `Elev=3`. For information about the U model, see Chapter 5, Ideal and Lumped Transmission Line Models

Using RLGC Matrices

RLGC matrices in the RLGC model of the W-element are in the Maxwellian format. In the U model, they are in self/mutual format. For conversion information, see Determining Matrix Properties on page 102. When you use the U model, the W-element performs the conversion internally. Table 6 on page 134 shows how the RLGC matrices in the U Model are related to the RLGC matrices in the W-element, and how the W-element uses these matrices.

The following sections discuss these topics:

- Handling the Dielectric-loss Matrix
- Handling the Skin-effect Matrix

Handling the Dielectric-loss Matrix

Because the U model does not input the G_d dielectric loss matrix, the W-element defaults G_d to zero when it uses the U model input.

Handling the Skin-effect Matrix

The U and W-elements use the R_s skin-effect resistance in different ways.

- In a W-element, the R_s matrix specifies the square-root dependence of the frequency-dependent resistance:

Equation 54 $$R(f) \cong R_o + \sqrt{f}(1+j)R_s$$

- In U-elements, R is the value of skin resistance at the frequency:

Equation 55 $R \cong R_c + R_s$

In the preceding equation, the core resistance (R_c) is equivalent to the DC resistance (R_o) in the W-element. The frequency at which the U-element computes the R matrix is:

Equation 56 $f_{skin} = \frac{1}{15 \cdot RISETIME}$

Table 6 RLGC Matrices for U and W elements

For U models with	W-element
RLGC input; Elev=2	Uses the R_s values that you specify in the *U* model.
Geometric input; Elev=1	Divides the R_s (which the U model computes internally), by $\sqrt{f_{skin}}$ to obtain the R_s value. For Elev=1, the R_s value in the U model printout is not the same as the R_s value in the W-element.
Measured-parameter input; Elev=3	Does not support the skin effect.

If you do not specify the `RISETIME` option, the U-element uses `Tstep` from the `.TRAN` card.

Table 7 RLGC Matrices in the W-element and the U Model

W-element Parameters	U Model Parameters
L, C $\begin{bmatrix} L_{11} & & \\ L_{12} & L_{22} & \\ L_{13} & L_{23} & L_{33} \end{bmatrix}$	$\begin{bmatrix} C_{r1}+C_{12}+C_{13} & & \\ -C_{12} & C_{r2}+C_{12}+C_{23} & \\ -C_{13} & -C_{23} & C_{r3}+C_{13}+C_{23} \end{bmatrix}$
Go, Gd $\begin{bmatrix} G_{r1}+G_{12}+G_{13} & & \\ -G_{12} & G_{r2}+G_{12}+G_{23} & \\ -G_{13} & -G_{23} & G_{r3}+G_{13}+G_{23} \end{bmatrix}$	$\begin{bmatrix} 0 & & \\ 0 & 0 & \\ 0 & 0 & 0 \end{bmatrix}$

Table 7 RLGC Matrices in the W-element and the U Model

Nlay=1 (no skin effect)	Nlay=2 (skin effect present)
$Ro\begin{bmatrix} R_{11}+R_{rr} & & \\ R_{rr} & R_{22}+R_{rr} & \\ R_{rr} & R_{rr} & R_{33}+R_{rr} \end{bmatrix}$	$\begin{bmatrix} R_{1c}+R_{rc} & & \\ R_{rc} & R_{2c}+R_{rc} & \\ R_{rc} & R_{rc} & R_{3c}+R_{rc} \end{bmatrix}$
Nlay=1 (no skin effect)	Nlay=2 (skin effect present)
$Rs\begin{bmatrix} 0 & & \\ 0 & 0 & \\ 0 & 0 & 0 \end{bmatrix}\frac{1}{\sqrt{f_{skin}}}$	$\begin{bmatrix} R_{1s}+R_{rs} & & \\ R_{rs} & R_{2s}+R_{rs} & \\ R_{rs} & R_{rs} & R_{3s}+R_{rs} \end{bmatrix}$

The following netlist is for a 4-conductor line as shown in Figure 37.

```
* W Element example, four-conductor line, U model
W1 1 3 5 0 2 4 6 0 Umodel=example N=3 l=0.97
.MODEL example U LEVEL=3 NL=3 Elev=2 Llev=0 Plev=1 Nlay=2
+ L11=2.311uH
+ L12=0.414uH L22=2.988uH
+ L13=84.2nH L23=0.527uH L33=2.813uH
+ Cr1=17.43pF
+ C12=5.41pF Cr2=10.1pF
+ C13=1.08pF C23=5.72pF Cr3=17.67pF
+ R1c=42.5 R2c=41.0 R3c=33.5
+ Gr1=0.44387mS
+ G12=0.1419mS Gr2=0.3671mS
+ G13=23.23uS G23=90uS Gr3=0.38877mS
+ R1s=0.00135 R2s=0.001303 R3s=0.001064
V1 1 0 AC=1v DC=0v pulse(4.82v 0v 5ns 0.1ns 0.1ns 25ns)
.AC lin 1000 0Hz 1GHz
.DC v1 0v 5v 0.1v
.TRAN 0.1ns 200ns
.END
```

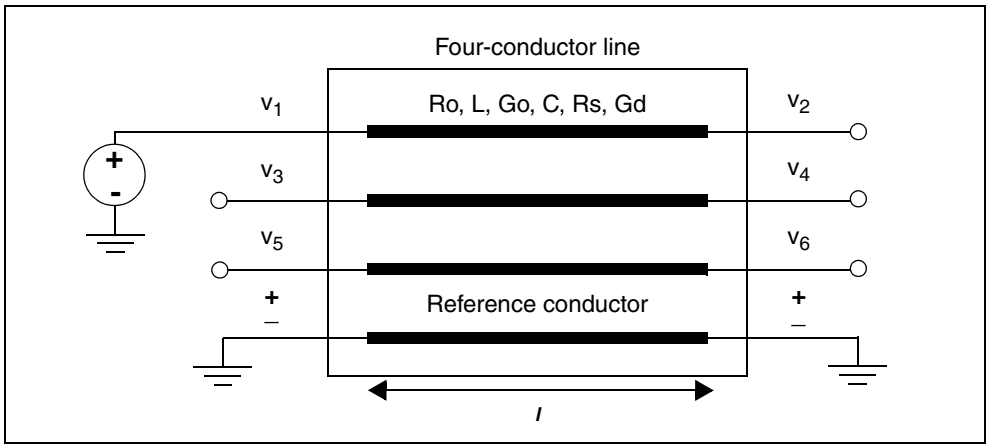


Figure 37 4-Conductor Line

Input Model 3: Built-in Field-Solver Model

Instead of RLGC matrices, you can directly use geometric data with the W-element by using a built-in field solver. To use the W-element with a field solver, specify `FSmodel=`*`model_name`* on the W-element card. For a description of the built-in field solver, see Field Solver Model Syntax on page 150.

Input Model 4: Frequency-Dependent Tabular Model

You can use the tabular RLGC model as an extension of the analytical RLGC model to model any arbitrary frequency-dependent behavior of transmission lines (this model does not support RC lines).

You can use this extension of the W-element syntax to specify a table model (use a `.MODEL` statement of type w). To accomplish this, the `.MODEL` statement refers to .MODEL statements where the “type” is SP (described in Small-Signal Parameter Data Frequency Table Model (SP Model) on page 73), which contain the actual table data for the RLGC matrices.

Note: To ensure accuracy, the W-element tabular model requires the following:

- R and G tables require zero frequency points.
- L and C tables require infinity frequency points as well as zero frequency points.

To specify a zero frequency point, you may use DC keyword or f=0 data entry in the DATA field of the SP model. To specify an infinity frequency point, use the INFINITY keyword of the SP model.

See also, Small-Signal Parameter Data Frequency Table Model (SP Model) on page 73.

The following sections discuss these topics:

- Notation Used
- Table Model Card Syntax
- Examples: 4-Conductor Tx Line and RLGC Model List
- Introducing Causality Check for W-element RLGC Table Model

Notation Used

- Lower-case variable: Scalar quantity
- Upper-case variable: Matrix quantity
- All upper-case words: Keyword
- Parentheses and commas: Optional

Table Model Card Syntax

```
.MODEL name W MODELTYPE=TABLE [FITGC=0|1] N=val
+ LMODEL=l_freq_model CMODEL=c_freq_model
+ [RMODEL=r_freq_model GMODEL=g_freq_model]
```

Parameter	Description
FITCG	Keyword for W Model (w/ MODELTYPE=TABLE) 1=causality check on, 0= causality check off (default)
N	Number of signal conductors (excluding the reference conductor).
LMODEL	SP model name for the inductance matrix array.

Parameter	Description
CMODEL	SP model name for the capacitance matrix array.
RLMODEL	SP model name for the resistance matrix array. By default, it is zero.
GMODEL	SP model name for the conductance matrix array. By default, it is zero.

The following is an example netlist of a two-line system.

```
.MODEL ex1 W MODELTYPE=TABLE N=2 LMODEL=lmod1
+ CMODEL=cmod1 RMODEL=rmod1 GMODEL=gmod1
.MODEL cmod1 sp N=2 SPACING=NONUNIFORM VALTYPE=REAL
+ INFINITY=(5.602360e-11 -7.047240e-12 5.602360e-11)
+ DATA=( 1,
+ (0.000000e+00 5.602360e-11 -7.047240e-12 5.602360e-11)
+ )
.MODEL cmod1 sp N=2 SPACING=NONUNIFORM VALTYPE=REAL
+ INFINITY=(3.93346e-7 4.93701e-8 3.93346e-7)
+ DATA=( 34,
+ (0.000000e+00 3.933460e-07 4.937010e-08 3.933460e-07)
+ (3.746488e+06 4.152139e-07 4.937010e-08 4.151959e-07)
........
+ (4.000000e+09 3.940153e-07 4.937010e-08 3.940147e-07)
+ )
.MODEL lmod1 sp N=2 SPACING=NONUNIFORM VALTYPE=REAL
+ DATA=( 34,
+ (0.000000e+00 8.779530e-02 6.299210e-03 8.779530e-02)
+ (3.746488e+06 6.025640e-01 6.299210e-03 6.021382e-01)
........
+ (4.000000e+09 1.690795e+01 6.299210e-03 1.689404e+01)
+ )
.MODEL rmod1 sp N=2 SPACING=NONUNIFORM VALTYPE=REAL
+ DATA=( 34,
+ (0.000000e+00 8.779530e-02 6.299210e-03 8.779530e-02)
+ (3.746488e+06 6.025640e-01 6.299210e-03 6.021382e-01)
........
+ (4.000000e+09 1.690795e+01 6.299210e-03 1.689404e+01)
+ )
.MODEL gmod1 sp N=2 SPACING=NONUNIFORM VALTYPE=REAL
+ DATA=( 34,
+ (0.000000e+00 5.967166e-11 0.000000e+00 5.967166e-11)
+ (3.746488e+06 1.451137e-05 -1.821096e-06 1.451043e-05)
........
+ (4.000000e+09 1.549324e-02 -1.944324e-03 1.549224e-02)
+ )
```

Examples: 4-Conductor Tx Line and RLGC Model List

Table 8 on page 139 is an example of a four-conductor transmission line system, and Table 9 on page 140 is a list of a tabular RLGC model.

Table 8 Input File Listing

Listing Type	W-element Tabular Model Example

Header, options and sources	`.OPTION POST`
	`V1 7 0 ac=1v dc=0.5v pulse(0.5v 1.5v 0ns 0.1ns)`
	`V2 8 0 dc=1v`

Analysis	`.DC v1 0.5v 5.5v 0.1v SWEEP length POI 2 1.2 2`
	`.AC lin 200 0Hz 1GHz SWEEP Ro POI 3 400 41.6667 400`
	`.TRAN 0.1ns 50ns`

Termination	`R1 7 1 50`
	`R2 4 0 450`
	`R3 5 0 450`
	`R8 6 0 450`
	`R5 4 5 10800`
	`R6 5 6 10800`
	`R7 4 6 1393.5`

Table 8 Input File Listing

Listing Type	W-element Tabular Model Example

Analytical RLGC model (*W*-element)	`.SUBCKT sub 1 2 3 4 5 6 7 8`
	`W1 1 2 3 4 5 6 7 8 l=0.1 fgd=5e6 RLGCMODEL=analymod n=3`
	`.MODEL analymod W MODELTYPE=RLGC N=3`
	`+ Lo=2.41667e-6`
	`+ 0.694444e-6 2.36111e-6`
	`+ 0.638889e-6 0.694444e-6 2.41667e-6`
	`+ Co=20.9877e-12`
	`+ -12.3457e-12 29.3210e-12`
	`+ -4.01235e-12 -12.3457e-12 20.9877e-12`
	`+ Ro=41.6667`
	`+ 0 41.6667`
	`+ 0 0 41.6667`
	`+ Go=0.585937e-3`
	`+ 0 0.585937e-3`
	`+ 0 0 0.585937e-3`
	`+ Rs=0.785e-5`
	`+ 0 0.785e-5`
	`+ 0 0 0.785e-5`
	`+ Gd=0.285e-6`
	`+ 0 0.285e-6`
	`+ 0 0 0.285e-6`
	`.ENDS sub`

Tabular RLGC model (*W*-element)	`.ALTER Tabular Model`
	`.SUBCKT sub 1 2 3 4 5 6 7 8`
	`W1 n=3 1 2 3 4 5 6 7 8 l=0.1 fgd=5e6 tablem odel=trmod`
	`.INCLUDE table.txt`
	`.ENDS sub`

Table 9 Tabular RLGC Model

Listing Type	W-element Tabular Model Example
RLGC table model definition	`.MODEL trmod W MODELTYPE=TABLE N=3` `+ LMODEL=lmod CMODEL=cmod RMODEL=rmod GMODEL=gmod`

C model	`.MODEL cmod sp N=3 VALTYPE=REAL INTERPOLATION=LINEAR`
	`+ DATA=( 1 2.09877e-11 -1.23457e-11 2.9321e-11`
	`+ -4.01235e-12 -1.23457e-11 2.09877e-11)`

Table 9 Tabular RLGC Model

Listing Type	W-element Tabular Model Example

L model	`.MODEL lmod sp N=3 VALTYPE=REAL INTERPOLATION=LINEAR`
	`+ INFINITY= 2.41667e-06 6.94444e-07 2.36111e-06`
	`+ 6.38889e-07 6.94444e-07 2.41667e-06 FSTOP=1e+07`
	`+ DATA=( 25 2.41667e-06 6.94444e-07 2.36111e-06`
	`+ 6.38889e-07 6.94444e-07 2.41667e-06 2.41861e-06`
	`+ 6.94444e-07... 2.41707e-06 6.94444e-07 2.36151e-06`
	`+ 6.38889e-07 6.94444e-07 2.41707e-06 )`

R model	`.MODEL rmod sp N=3 VALTYPE=REAL INTERPOLATION=LINEAR`
	`+ FSTOP=1e+10 DATA=( 200 41.6667 0 41.6667 0 0 41.6667`
	`+ 41.7223 0 41.7223 0 0 41.7223 ...`
	`+ 42.4497 0 42.4497 0 0 42.4497 42.4517 0 42.4517 0 0`
	`+ 42.4517)`

G model	`.MODEL gmod sp N=3 VALTYPE=REAL INTERPOLATION=LINEAR`
	`+ FSTOP=1e+08 + DATA=( 100 0.000585937 0 0.000585937`
	`+ 0 0 0.000585937 0.282764 0 0.282764 0 0 0.282764`
	`+ ... 1.42377 0 1.42377 0 0 1.42377 1.42381 0 1.42381`
	`+ 0 0 1.42381)`

Introducing Causality Check for W-element RLGC Table Model

To improve the accuracy of W-element RLGC table model, you can introduce a complex dielectric coefficient to assure causality for the W-element RLGC table model. You can use the keyword FITGC=[1|0] when using the W-element RLGC table model to turn on or turn off this method. By default, FITGC=0.

Although the dielectric properties have only a slight frequency-dependent character, they have an impact on transmission line simulation accuracy in that not only does that signal appear at the output port before the delay time is reached, but there is also a non-consistency between the segmented lines and an integral line. Figure 38 and Figure 39 on page 142 show that when a long transmission line is separated into 50 series connected segments, the output signal at the far end of the lines has large discrepancies compared to a single line, while they should be the same. When applying DELAYOPT=3, the discrepancy becomes even bigger. This is due to the errors introduced by inter-/extrapolation at a high frequency band which are more accurately fitted, and, therefore, more explicitly exposed in the simulation results.

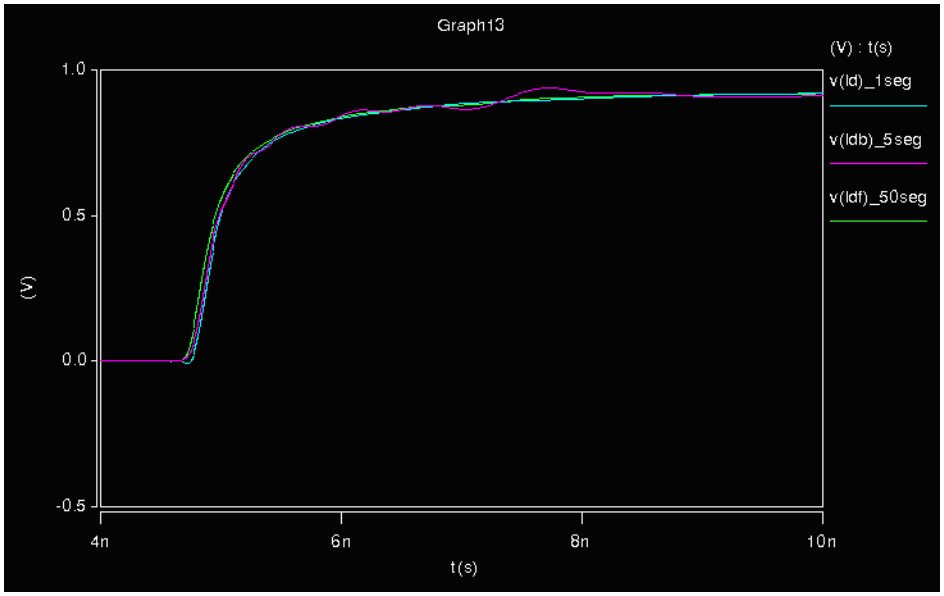


Figure 38 Non-consistency between segmented lines and one integral line; DELAYOPT=0

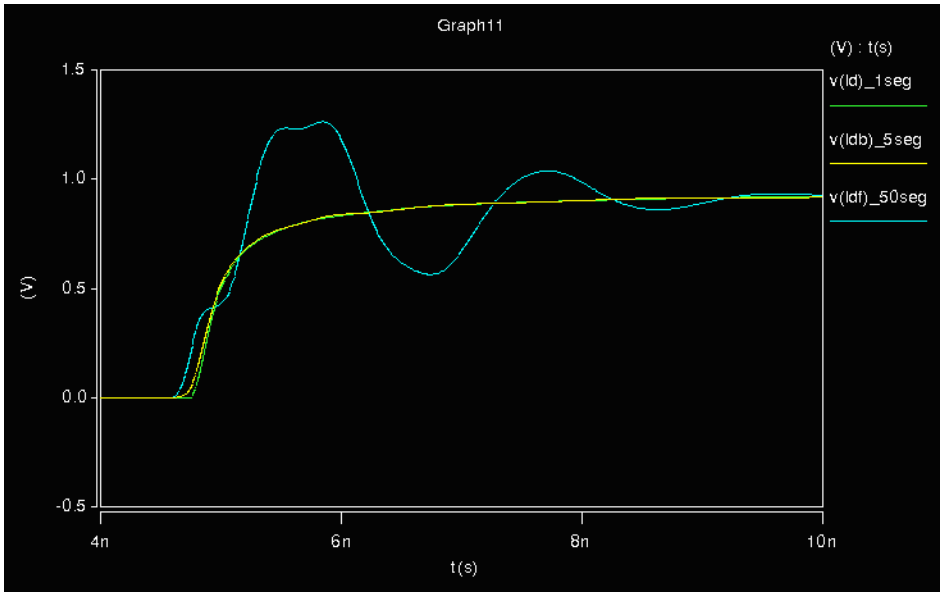


Figure 39 Non-consistency between segmented lines and one integral line; DELAYOPT=3

The solution to this issue lies in the dielectric properties of a transmission line system. In an ideal capacitor, the current that flows through the capacitor is exactly 90 degrees out of phase with the voltage sine wave. If the ideal capacitor were filled with an insulator with a dielectric constant of ε_r, the capacitance would increase to $C = \varepsilon_r C_0$. However, real dielectric materials have some resistivity associated, which leads to leakage current. This current is completely in phase with the voltage. It can be modeled as an ideal resistor. By conventional transmission line theory, it is modeled by conductor G.

Both of C and G are frequency-dependent parameters in a real transmission line system. The frequency-dependent character comes from the dipoles in dielectric material. Actually, both of these two terms relate to the number of dipoles, how large they are and how they are able to move. These characters are described by dielectric constant of the material.

Equation 57

$$\varepsilon_r(\omega) = \varepsilon_r'(\omega) + j\omega\varepsilon_r''(\omega)$$

$$tg\delta = \frac{\varepsilon_r''}{\varepsilon_r'}$$

$$C(\omega) = \varepsilon_{r'}(w)C_0$$

$$C(\omega) = \omega C(\omega) tg\delta = \omega\varepsilon_r''(\omega)C_0$$

The real part of ε corresponds to the motion of the dipoles that are out of phase with the applied field and contributes to increasing capacitance, while the imaginary part corresponds to the motion of the dipoles that are in phase with the applied voltage and contribute to the losses.

Since the frequency-dependent character of both of these terms relates to the motion of dipoles, which can be described by $\varepsilon_r\omega$, the real and imaginary part of ε_r must satisfy the Kramers-Kronig relationship. And therefore, we can fit with rational function.

Equation 58

$$\varepsilon_r(\omega) = D + j\omega E + \sum_{m=1}^{N} \frac{C_m}{j\omega - A_m}$$

Once we get $\varepsilon_r\omega$, we can calculate accurate $C(\omega)$ and $G(\omega)$ interpolation and extrapolation, using Eq. 58.

Once HSPICE gets a successful fitting, we can get a $G(\omega)$ and $C(\omega)$ matrix from the fitting result. Since we only consider the RMS error of the real part during the fitting process, we get only the $C(\omega)$ matrix from the fitting result. For $G(\omega)$, we still use the conventional linear interpolation and extrapolation because of its strong dependency on (ω).

Input Model 5: S Model

The W-element can accept the transmission line-based S-parameters as input. To use the W-element with the S Model on the W-element card specify the following line:

```
SMODEL=Smodel_name NODEMAP=XiYj...
```

Where,

- `Smodel_name` is an S model, which is normally used for an S-element. Use the `XLINELENGTH` keyword in the S Model statement to indicate the line length of the system where the S-parameters are extracted. This keyword is required only when you use an S Model with a W-element. See S-element Syntax on page 45 for more information.
- `NODEMAP` is a string that assign each index of the S-parameter matrix to one of the W-element terminals. This string must be an array of pairs that consists of a letter and a number, (for example, Xn), where
  - `X= I`, `i`, `N`, or `n` to indicate near end (input side) terminal of the W-element
  - `X= O`, `i`, `F`, or `f` to indicate far end (output side) terminal of the W-element.

    For example, `NODEMAP = I1I2O1O2` represents that the
  - 1st port of the s-matrix corresponds to the 1st near end terminal of the W-element.
  - 2nd port of the s-matrix corresponds to the 2nd near end terminal of the W-element.
  - 3rd port of the s-matrix corresponds to the 1st far end terminal of the W-element.
  - 4th port of the s-matrix corresponds to the 2nd far end terminal of the W element.

`NODEMAP = I1I2I3...InO1O2O3...On` is the default setting.

The following sections discuss these topics:

- S Model Conventions
- S Model Example

S Model Conventions

When specifying an S model, you must adhere to the following rules and conventions:

- The size of the NODEMAP array must be the same as twice the line number of the W-elements and also must be the same as the port count of the S-parameter matrices.
- If the W-element input model is SMODEL, an S model definition must accompany that input model.
- S-parameters must have even number of terminals.
- S-parameters must be symmetric.
- S-parameters must be passive.
- Transmission-line based S-parameters can be used with different lengths of a system when the varying length keyword (L) in a W-element instance statement is present.
- The `XLINELENGTH` keyword must be set when used in S Models that use W-elements.

S Model Example

The following input netlist file shows S model input for the W-element:

```
**** W Element Example: S Model ***
rout out 0 50
vin in gnd LFSR (1 0 0 0.1n 0.1n 1g 1 [5,2] rout=50)
*+ pulse(0 1 0 0.1n 0.1n 0.9n 2n)

W1 in gnd out gnd SMODEL=smodel N=1 l=0.3
+ NODEMAP=I1O1
.MODEL smodel S TSTONEFILE=w.s2p
+ XLINELENGTH=0.3

.opt accurate post
.tran .01n 20n

.end
```

Extracting Transmission Line Parameters (Field Solver)

The built-in 2-D electromagnetic field solver is highly-optimized for interconnects in stratified media. This field solver uses the W-element, and it supports optimization and statistical analysis within transient simulation.

The solver is based on:

- An improved version of the boundary-element method, and
- The filament method that is also implemented in the Synopsys product, Raphael.

See K. S. Oh, D. B. Kuznetsov, and J. E. Schutt-Aine, “Capacitance computations in a multi-layered dielectric medium using closed-form spatial Green s functions,” IEEE Trans. Microwave Theory and Tech., vol. 42, pp. 1443-1453, August 1994 for more information on the boundary-element method.

To learn more about BEM and Green’s Function, see the *Raphael Reference Manual*.

The following sections discuss these topics:

- Using the Field Solver Model
- Filament Method
- Modeling Geometries
- Solver Limitation
- Field-Solver-Related Netlist Statements

- Accelerating the W-element Field Solver Using an Iterative Solver
- Visualizing Cross-Sectional Geometric Information
- Field Solver Examples

Using the Field Solver Model

Use the field-solver model to specify a geometry model for the W-element transmission line. In the field-solver model:

- The list of conductors must appear last.
- Conductors cannot overlap each other.
- The Field Solver assumes that floating conductors are electrically disconnected, and does not support non-zero fixed charges. Because the field solver is designed as 2-D, it ignores displacement current in floating conductors.
- The Field Solver treats metal layers in the layer stack as the reference node.
- Conductors defined as `REFERENCE` are all electrically-connected, and correspond to the reference node in the W-element.
- You must place signal conductors in the same order as the terminal list in the W-element statement. For example, the ith signal conductor (not counting reference and floating conductors) is associated with the ith input and output terminals specified in the corresponding W-element.
- Floating and reference conductors can appear in any order.

Note: It is strongly recommended that a separate model be created for complex structures when an internal ground plane is introduced.

Filament Method

This section describes the filament method for the skin-effect resistance and inductance solver. The 2-D filament method uses data about magnetic coupling when it extracts frequency-dependent resistance and inductance. To use this solver, set `COMPUTE_RS=yes` in a `.FSOPTIONS` statement.

The following process explains the filament method:

1. The filament method divides the original conductor system into thin filaments.

2. From the coupling of these filaments, this method then derives the distributed magnetic coupling of the inside and outside of the conductor.
3. After dividing the conductors into thin filaments, this method creates the impedance matrix of the filament system:

   $$\mathbf{Z}_f = \mathbf{R}_f + j\omega\mathbf{L}_f$$

4. This method uses the following equation to solve the current matrix ($\mathbf{i}_f$):

   $$\mathbf{v}_f = Z_f\mathbf{i}_f$$

   In the preceding equation, the vf vector excites the filament system.
5. The filament method uses the result of this equation to calculate the partial current matrix of the conductor system ip as a sum of all filament currents:

Equation 59
$$i_{p(j,\ k)} = \sum_{\text{filaments in conductor j}} i_f \quad (@\ \text{k-th excitation vector})$$

6. The filament method uses the following equation to solve the partial impedance matrix ($\mathbf{Z}_p$):

   $$\mathbf{v}_p = \mathbf{Z}_p\mathbf{i}_p$$ 161

7. From the components of the partial impedance matrix, the filament method uses the following relationship to calculate the components of the loop $Z_{p(j,\ k)}[j,\ k{:}0{\sim}n]$ impedance matrix:

Equation 60
$$z_{l(j,\ k)} = z_{p(j,\ k)} - z_{p(j,\ 0)} - z_{p(k,\ 0)} + z_{p(0,\ 0)}$$

   In the preceding equation, n is the number of signal (non-reference) conductors in the system.

   Note: W-element analysis uses these loop impedance components.

For full discussion of the `.FSOPTIONS` command, see Accelerating the W-element Field Solver Using an Iterative Solver on page 157 or .FSOPTIONS in the *HSPICE Reference Manual: Commands and Control Options*.

Modeling Geometries

In geometry modeling:

- The number of dielectric layers is arbitrary.
- You can arbitrarily shape the conductor cross-section, including an infinitely-thin strip.
- The number of conductors is unlimited.
- The current dielectric region must be planar.
- Conductors must not overlap each other.
- Magnetic materials are not supported.

Geometric modeling outputs the Maxwellian (short-circuit) transmission line matrices: C, L, Ro, Rs, Go, and Gd. (See Equations and Parameters on page 96.)

Solver Limitation

When the field solver computes the conductance matrices (Go and Gd), if the media are not homogeneous, then the solver uses the arithmetic average values of conductivities and loss tangents.

Field-Solver-Related Netlist Statements

Table 10 describes the netlist statements that specifically relate to the field solver. For the syntax and examples of these statements, see the links to the *HSPICE Reference Manual: Commands and Control Options*.

Table 10 Field-Solver Statement Syntax

Statement	Usage
.MATERIAL	Use this statement to define the properties of a material.
.LAYERSTACK	Use this statement to define a stack of dielectric or metal layers.

Table 10 Field-Solver Statement Syntax (Continued)

Statement	Usage
.SHAPE	Use this statement to define a shape. The Field Solver uses these shapes: ■ Rectangle ■ Circle ■ Strip ■ Polygon ■ Trapezoid to describe a cross-section of the conductor.
.FSOPTIONS	Use this statement to set various options for the field solver.
.MODEL W MODELTYPE=FieldSolver	Type of transmission-line model.

The following sections discuss these topics:

Field Solver Model Syntax

Use a `.MODEL` statement to define a field solver.

Syntax

```
.MODEL mname W MODELTYPE=FieldSolver
+ LAYERSTACK=name [FSOPTIONS=name]
+ [RLGCFILE=name] [COORD=0|DESCART|1|POLAR]
+ [OUTPUTFORMAT=RLGC|RLGCFILE]
+ CONDUCTOR=SHAPE=name [MATERIAL=name]
+ [ORIGIN=(x,y)] [TYPE=SIGNAL|REFERENCE|FLOATING] ...
```

Parameter	Description
mname	Model name.
LAYERSTACK	Name of the associated layer stack.
FSOPTIONS	Associated option name. If you do not specify this entry, the Field Solver uses the default options.

Parameter	Description
RLGCFILE	Use the output file for RLGC matrices, instead of the standard error output device. A RLGC file name must start with an alphabetic letter (not a number). If the specified file already exists, then the Field Solver appends the output. To generate output, you must set PRINTDATA in `.FSOPTIONS` to YES; setting it to APPEND, appends the extracted RLGC model to the specified file.
COORD	The polar field solver is invoked only when COORD=1 or COORD=POLAR.
OUTPUTFORMAT	Model syntax format for RLGC matrices in the W-element. Specified in the RLGC file. Default format is an RLGC model.
SHAPE	Shape name.
x y	Coordinates of the local origin.
MATERIAL	Conductor material name. If you do not specify this entry, the Field Solver defaults to the predefined metal name PEC (perfect electrical conductor).
ORIGIN	The (radius, degree) of the polar field solver.
TYPE	One of the following conductor types: ▪ SIGNAL: a signal node in the W-element (the default). ▪ REFERENCE: the reference node in the W-element. ▪ FLOATING: floating conductor, no reference to W-element.

Using the Field Solver to Extract a RLGC Tabular Model

You can use the Field Solver to extract a RLGC tabular model which allows higher flexibility of dependence on frequency. (When `PRINTDATA=APPEND`, RLGC model output is appended to the specified output file.)

Syntax for Extracting RLGC Tabular Model

```
.FSOPTIONS name [ACCURACY=HIGH|MEDIUM|LOW]
+ [GRIDFACTOR=val]
+ [COMPUTE_GO=YES|NO] [COMPUTE_GD=NO|YES]
+ [COMPUTE_RO=YES|NO] [COMPUTE_RS=NO|YES|DIRECT|ITER]
+ [COMPUTE_TABLE=frequency_sweep]
+ [PRINTDATA=NO|YES|APPEND]
```

Note: The forms of the following arguments are interchangeable:

```
COMPUTE_GO : COMPUTEGO
COMPUTE_GD : COMPUTEGD
COMPUTE_RO : COMPUTERO :
COMPUTE_RS : COMPUTERS :
COMPUTE_TABLE : COMPUTETABLE
```

Note: If you only set `COMPUTE_GD=yes` in the `.FSOPTIONS` statement, you do not generate data in the Gd matrix of the RLGC file. Since Gd is the dielectric loss conductance matrix, you also need to define the loss tangent values for each dielectric material you have in your layer stack.

For example:

```
.MATERIAL die1 DIELECTRIC ER=4.1  LOSSTANGENT=.012
```

The Gd matrix is derived from Co matrix and losstangent value as $\mathrm{Gd} = 2 \cdot \pi \cdot \tan\delta \cdot \mathrm{Co}$

when multiple loss tangent values are given to the `.layerstack` structure, the average value (including background material property) is used.

Keyword

`COMPUTE_TABLE` or `COMPUTETABLE`

Frequency sweep	Definition
`COMPUTE_TABLE`	Specifies a type of frequency sweep. You can specify either LIN, DEC, OCT, POI. Specify the nsteps, start, and stop values using the following syntax for each type of sweep: ▪ `LIN` *`nsteps start stop`* ▪ `DEC` *`nsteps start stop`* ▪ `OCT` *`nsteps start stop`* ▪ `POI` *`nsteps freq_values`* Note: To reduce the risk of instability in subsequent simulations, the field solver adds points to ensure that there is at least one frequency point per decade up to 1 THz.

Once a frequency sweep is specified, the W-element computes transmission line parameters at specified frequency points. In addition, since resistance and conductance at DC (zero frequency) and capacitance and inductance at infinite

frequency are essential to ensure the accuracy, these four matrices are automatically computed regardless of the sweep configuration.

In the table model extraction, series impedance, $\mathbf{Z}(\omega) = \mathbf{R}(\omega) + jw\mathbf{L}(\omega)$ is computed directly from the filament method solver. For shunt admittance, $\mathbf{Y}(\omega) = \mathbf{G}(\omega) + j\omega\mathbf{C}(\omega)$, the static capacitance solver is still used. From the static capacitance, C, and corresponding dielectric loss term, $\mathrm{Gd} = 2.\pi.\tan\delta.\mathrm{C}$, a complex dielectric loss model is derived. For further detail about the complex dielectric loss model generation, see Fitting Procedure Triggered by INCLUDEGDIMAG Keyword on page 101.

Default

If the `COMPUTE_TABLE` keyword is not specified, a conventional RLGC model is generated.

Note: When table model output is selected, for computational efficiency, the iterative solver (`COMPUTE_RS=ITER`) is chosen by default.

RLGC Tabular Model Sample Output

```
.MODEL rmod1 sp N=3 MATRIX=SYMMETRIC
+ SPACING=POI VALTYPE=REAL INTERPOLATION=LINEAR
+ DC = 3.850667e+04
+ 7.700000e+03  3.850667e+04
+ 7.700000e+03  7.700000e+03  3.850667e+04
+ INFINITY = 1.444000e+05
+ 3.082000e+04  1.402667e+05
+ 2.389333e+04  3.082000e+04  1.444000e+05
+ DATA = ( 20
+ 0.000000e+00
+ 3.850667e+04
+ 7.700000e+03  3.850667e+04
+ 7.700000e+03  7.700000e+03  3.850667e+04
+ .....
+
.MODEL lmod1 sp N=3 MATRIX=SYMMETRIC
+ SPACING=POI VALTYPE=REAL INTERPOLATION=LINEAR
+ INFINITY =  6.624667e-07
+ 2.603333e-07  7.253333e-07
+ 1.406667e-07  2.603333e-07  6.624667e-07
+ DATA = ( 20
+ 0.000000e+00
+ 9.733333e-07
+ 4.876000e-07  9.806667e-07
+ 3.454000e-07  4.876000e-07  9.733333e-07
+ .....
+
.MODEL gmod1 sp  N=3 MATRIX=SYMMETRIC
+ SPACING=POI VALTYPE=REAL
+ DC = 0.000000e+00
+ 0.000000e+00  0.000000e+00
+ 0.000000e+00  0.000000e+00 0.000000e+00
+ DATA=( 1
+ 0.000000e+00
+ 0.000000e+00
+ 0.000000e+00  0.000000e+00
+ 0.000000e+00  0.000000e+00  0.000000e+00
+ ......
+
.MODEL cmod1 sp  N=3 MATRIX=SYMMETRIC
+ SPACING=POI VALTYPE=REAL+ INFINITY =  3.930057e-17
+ -1.300346e-17   3.969922e-17 + -3.026179e-18  -1.300788e-17
3.929558e-17
+ DATA=( 20+ 0.000000e+00
+ 3.930057e-17 + -1.300346e-17   3.969922e-17
+ -3.026179e-18  -1.300788e-17  3.929558e-17 + .....
```

Accounting for Surface Roughness Effect in HSPICE W-element

In real devices operating at high frequency range, skin effect causes a secondary effect since the surfaces of conductors are not flat but have some roughness. When a majority of current propagates across the rough surfaces, there is a non-negligible increase in series impedance. Since the influence of this phenomenon depends on the dominance of current around the conductor surface, this effect is also frequency-dependent.

HSPICE provides two ways to take this frequency-dependent increase of series impedance into account:

- Scaling RS matrix
- Calculating root mean square (RMS) surface roughness height

To scale the RS matrix use the SCALE_RS keyword with scaling factor and apply it to the skin-effect (Rs) matrix. Thus, the series impedance is expressed as

Equation 61 $$S\,\mathrm{R}(f) = \mathrm{Ro} + (1+j)\cdot\sqrt{f}\cdot SCALE\_RS\cdot \mathrm{Rs}$$

To calculate the RMS surface roughness height, the ratio of resistivity increment, SR, may be empirically estimated as,

Equation 62 $$SR = \frac{2}{\pi}\tan^{-1}\left(1.4\left(\frac{\Delta}{\delta s(f)}\right)^2\right)$$

Where, Δ and δs are the RMS surface roughness height and skin depth of the conductor material. Then, when the W-element filament field solver runs, at each frequency point, metal resistivity, ρ, is re-scaled to be a function of frequency using the surface roughness factor as,

Equation 63 $$\rho'(f) = (1+SR)\rho$$

Syntax for Scaling RS Matrix

```
Wxxx ni1 ni2… ref_in no1 no2… ref_out
+ [SCALE_RS=value]
```

Keyword

`SCALE_RS`: Scaling factor to the RS matrix

Syntax for Taking RMS Surface Roughness of Conductor Materials

```
.material copper metal conductivity=value [roughness=value]
```

Keyword

ROUGHNESS: RMS surface roughness height.

Note: The current release uses the averages of surface roughness factor and skin depth when you specify multiple conductor materials in one field solver system.

Transmission Line Surface Roughness Example

```
*** no surface roughness ***
P1 in1 0 port=1 ac=1
P2 out1 0 port=2
W1 in1 gnd out1 gnd FSmodel=line1 N=1 l=0.1

*** use copper_roughs w/ roughness=2um ***
P3 in2 0 port=3 ac=1
P4 out2 0 port=4
W2 in2 gnd out2 gnd FSmodel=line1_rough N=1 l=0.1

*** use SCALE_RS=1.1 ***
P5 in3 0 port=5 ac=1
P6 out3 0 port=6
W3 in3 gnd out3 gnd FSmodel=line1 N=1 l=0.1 SCALE_RS=1.1

.material diel dielectric er=4.3
.material copper metal conductivity=57.6meg
.material copper_rough metal conductivity=57.6meg
+ROUGHNESS=2e-6

.shape rect rectangle width=400e-6 height=40e-6
.layerstack stack1 background=air
+layer=(copper,10e-6)
+layer=(diel,200e-6)
.fsoptions opt1 printdata=yes  computegd=no computers=yes
.model line1 W Modeltype=fieldsolver,
+layerstack=stack1,
+fsoptions=opt1,
+Rlgcfile=line1.rlgc,
+conductor=(shape=rect,origin=(0,110e-6),material=copper)
.model line1_rough W Modeltype=fieldsolver,
+layerstack=stack1,
+fsoptions=opt1,
+Rlgcfile=line1_rough.rlgc,
+conductor=(shape=rect,origin=(0,110e-6),material=copper_rough)
.opt post
.ac dec 100 1e6 1e10
.end
```

Accelerating the W-element Field Solver Using an Iterative Solver

You can increase the speed of the W-element magnetic coupling field solver with an iterative solver. The skin effect solver employs the filament method which requires discretization of all the area inside of conductors to see frequency-dependent current distribution. Since the filament solver has to solve multiple frequency points to capture frequency dependent effect, it consumes the majority part of field solver run-time. To accelerate field solver by using the iterative solver, declare the `ITER` option.

Syntax

```
FSOPTIONS name [ACCURACY=HIGH|MEDIUM|LOW
+ [GRIDFACTOR=val] [PRINTDATA=NO|YES|APPEND]
+ [COMPUTE_GO=NO|YES] [COMPUTE_GD=YES|NO]
+ [COMPUTE_RO=YES|NO] [COMPUTE_RS=NO|YES|DIRECT|ITER]
+ [COMPUTE_TABLE=frequency_sweep]
```

Keyword

`COMPUTE_RS`

COMPUTE_RS Options	Definition
`YES`	Activate filament solver with direct matrix solver
`NO`	Do not to perform filament solver
`DIRECT`	Activate filament solver with direct matrix solver (same as `YES`)
`ITER`	Activate filament solver with iterative matrix solver

For full discussion of the `.FSOPTIONS` command, see .FSOPTIONS in the *HSPICE Reference Manual: Commands and Control Options*.

Visualizing Cross-Sectional Geometric Information

When HSPICE runs with the W-element field solver model, it generates the *model_name.str* file, which contains Tcl/Tk scripts to display cross-sectional geometrical information of the target structure. Once HSPICE is executed, locate the *model_name.str* file. To invoke Tcl/Tk graphics, at the command prompt, enter either:

```
% wish model_name.str
```

or

```
% chmod +x model_name.str
% model_name.str
```

Example output is seen in Figure 40 on page 158.

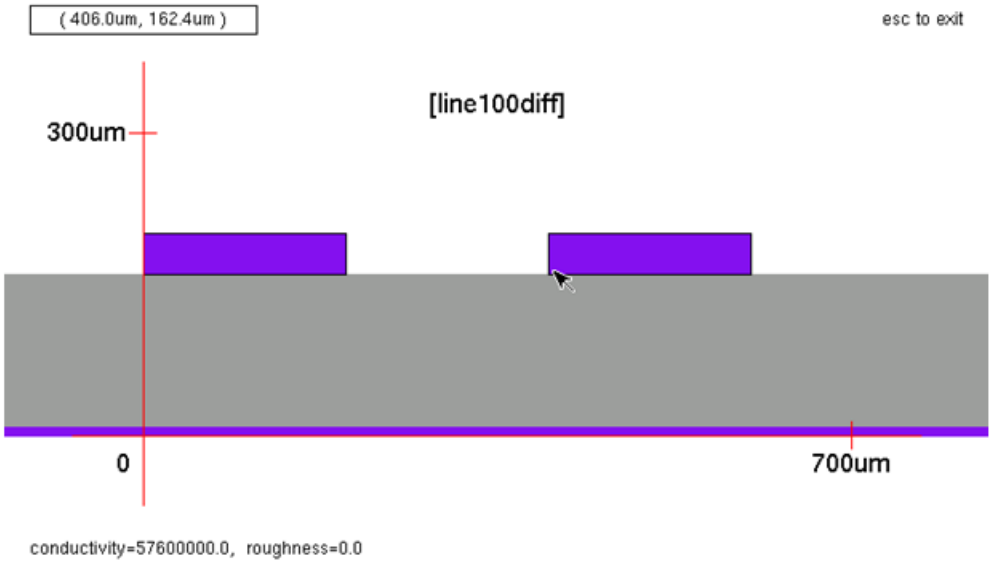


Figure 40 Structural Output Example

The following dynamic functions are available on the structural display window:

- Move the mouse cursor over an object to display corresponding material properties.
- Drag the mouse cursor to display geometrical locations.

Note: To use this function, the Tcl/Tk environment must be installed on users' systems. On a Linux system, Tcl/Tk is included in the default installation package. Manual installations are needed for the other platforms. The Tcl/Tk environment is available for all the platforms supported by HSPICE.

Limitation: the `POLAR` coordinate is not currently supported.

For further information go to: http://www.tcl.tk/software/tcltk/8.0.html

Field Solver Examples

The following examples show you how to use the Field Solver. All of the examples shown in this section run with the `HIGH` accuracy mode and with `GRIDFACTOR = 1`.

- Example 1: Cylindrical Conductor Above a Ground Plane
- Example 2: Stratified Dielectric Media
- Example 3: Two Traces Between Two Ground Planes
- Example 4: Using Field Solver with Monte Carlo Analysis

Example 1: Cylindrical Conductor Above a Ground Plane

This is an example of a copper cylindrical conductor above an ideal (lossless) ground plane.

With these formulas, you can derive the exact analytical formulas for all transmission line parameters:

Equation 64
$$L = \frac{1}{\mu\varepsilon}C^{-1}$$

Equation 65
$$G = \frac{\sigma_d}{\varepsilon}C = \omega \cdot \tan(\delta) \cdot C$$

Equation 66
$$R = \frac{1}{\sigma_c \delta \pi d}\left[\frac{2H/d}{\sqrt{(2H/d)^2 - 1}}\right] = \sqrt{f}\sqrt{\frac{\pi\mu}{\sigma_c}}\frac{1}{\pi d}\left[\frac{2H/d}{\sqrt{(2H/d)^2 - 1}}\right]$$

Figure 41 shows the geometry of a copper cylindrical conductor above an ideal ground plane.

Equation 67
$$C = \frac{2\pi\varepsilon}{\operatorname{acosh}\left(\frac{2H}{d}\right)}$$

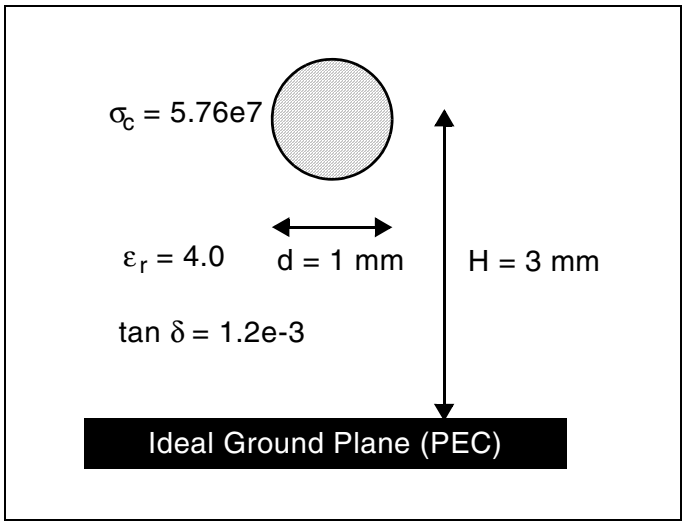


Figure 41 Cylindrical Conductor Above a Perfect Electrical Conductor Ground Plane

Table 11 lists the corresponding netlist.

Table 11 Input File Listing

Listing Type	Field Solver Cylindrical Example

Header, options and sources	`* Example: cylindrical conductor`
	`.OPTION PROBE POST`
	`VIMPULSE in1 gnd PULSE 4.82v 0v 5n 0.5n 0.5n 25n`

W-element	`W1 in1 gnd out1 gnd FSmodel=cir_trans N=1 l=0.5`

Materials	`.MATERIAL diel_1 DIELECTRIC ER=4,`
	`LOSSTANGENT=1.2e-3`
	`.MATERIAL copper METAL CONDUCTIVITY=57.6meg`

Shapes	`.SHAPE circle_1 CIRCLE RADIUS=0.5mm`
Defines a half-space	`.LAYERSTACK halfSpace BACKGROUND=diel_1,` `LAYER=(PEC,1mm)`
Option settings	`.FSOPTIONS opt1 PRINTDATA=YES,` `+ COMPUTE_RS=yes, COMPUTE_GD=yes`

Table 11 Input File Listing

Listing Type	Field Solver Cylindrical Example

Model definition	`.MODEL cir_trans W MODELTYPE=FieldSolver`
	`+ LAYERSTACK=halfSpace, FSOPTIONS=opt1,`
	`RLGCFILE=ex1.rlgc`
	`+ CONDUCTOR=(SHAPE=circle_1, ORIGIN=(0,4mm),`
	`+ MATERIAL=copper)`

Analysis, outputs and end	`.TRAN 0.5n 100n`
	`.PROBE v(out1)`
	`.END`

Compare the computed results with the analytical solutions in Table 12. The Field Solver computes the resistance and conductance at the frequency of 200 MHz, but does not include the DC resistance (Ro) and conductance (Go) in the computed values.

Table 12 Comparison Result

Value	Exact	Computed
C (pF/m)	89.81	89.66
L (nH/m)	494.9	495.7
G (mS/m)	0.1354	0.1352
R (Ω/m)	1.194	1.178

Example 2: Stratified Dielectric Media

This is an example of three traces immersed in a stratified dielectric media (see Figure 42 on page 162).

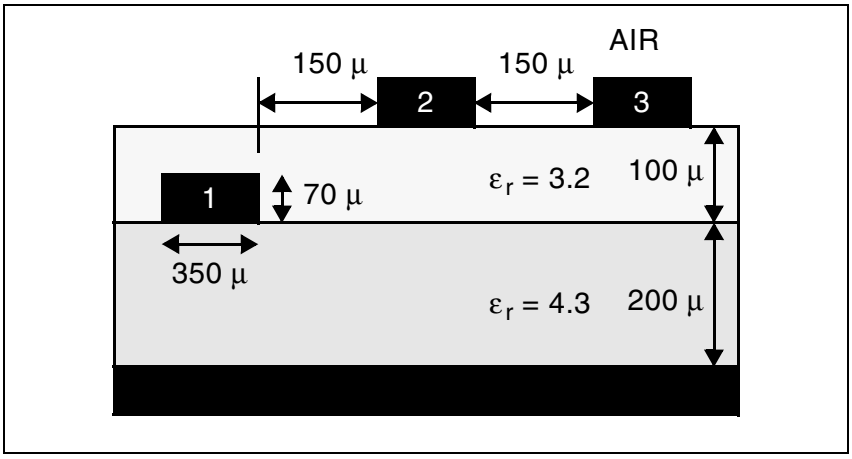


Figure 42 Three Traces Immersed in Stratified Dielectric Media

Table 13 shows the input file.

Table 13 Input File for Three Traces Immersed in Stratified Dielectric Media

Listing Type	Field Solver Stratified Dielectric Example

Header, options and sources	`* Example: three traces in dielectric`
	`.OPTION PROBE POST`
	`VIMPULSE in1 gnd PULSE 4.82v 0v 5n 0.5n 0.5n 25n`

W-element	`W1 in1 in2 in3 gnd out1 out2 out3 gnd` `+ FSmodel=cond3_sys N=3 l=0.5`

Materials	`.MATERIAL diel_1 DIELECTRIC ER=4.3`
	`.MATERIAL diel_2 DIELECTRIC ER=3.2`
	.MATERIAL copper METAL CONDUCTIVITY=57.6meg

Shapes	`.SHAPE rect_1 RECTANGLE WIDTH=0.35mm, HEIGHT=0.07mm`

Uses the default AIR background	`.LAYERSTACK stack_1`
	`+ LAYER=(copper,1um),LAYER=(diel_1,0.2mm),`
	`+ LAYER=(diel_2,0.1mm)`

Option settings	`.FSOPTIONS opt1 PRINTDATA=YES`

Table 13 Input File for Three Traces Immersed in Stratified Dielectric Media (Continued)

Listing Type	Field Solver Stratified Dielectric Example

Three conductors share the same shape	`.MODEL cond3_sys W MODELTYPE=FieldSolver,`
	`+ LAYERSTACK=stack_1, FSOPTIONS=opt1, RLGCFILE=ex2.rlgc`
	`+ CONDUCTOR=(SHAPE=rect_1,MATERIAL=copper, ORIGIN=(0,0.201mm)),`
	`+ CONDUCTOR=(SHAPE=rect_1,MATERIAL=copper,ORIGIN=(0.5mm,0.301mm)),`
	`+ CONDUCTOR=(SHAPE=rect_1,MATERIAL=copperORIGIN=(1mm,0.301mm))`

Analysis, outputs and end	`.TRAN 0.5n 100n`
	`.PROBE v(out1)`
	`.END`

Note: W. Delbare and D. D. Zutter, “Space-domain Green’s function approach to the capacitance calculation of multi-conductor lines in multi-layered dielectrics with improved surface charge modeling,” IEEE Trans. Microwave Theory and Tech., vol. 37, pp. 1562-1568, October 1989.

Figure 43 shows the results of convergence analysis, based on the total capacitance of the first conductor with respect to the `GRIDFACTOR` parameter.

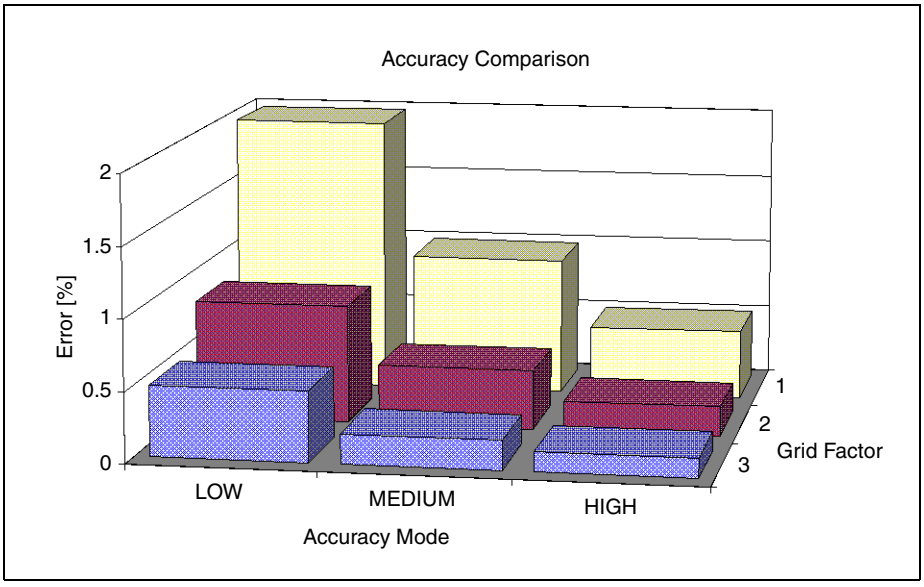


Figure 43 Convergence of Accuracy Modes

Example 3: Two Traces Between Two Ground Planes

This is an example of two traces between two ground planes (in other words, a coupled strip line) (see Figure 44).

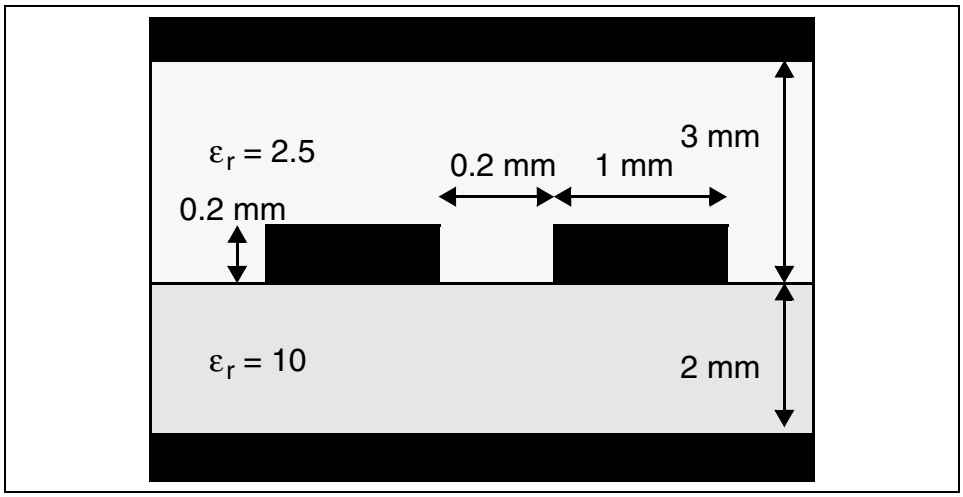


Figure 44 Example of a Coupled Strip Line

Table 14 lists the complete input netlist.

Table 14 Input Netlist for Two Traces Between Two Ground Planes

Listing Type	Field Solver Ground Planes Example

Header, options and sources	`* Example: two traces between gnd planes`
	`.OPTION PROBE POST`
	`VIMPULSE in1 gnd PULSE 4.82v 0v 5n 0.5n 0.5n 25n`

W-element	`W1 in1 in2 gnd out1 out2 gnd FSmodel=cond2_sys` `+N=2 l=0.5`

Materials	`.MATERIAL diel_1 DIELECTRIC ER=10.0`
	`.MATERIAL diel_2 DIELECTRIC ER=2.5`
	.MATERIAL copper METAL CONDUCTIVITY=57.6meg

Shapes	`.SHAPE rect RECTANGLE WIDTH=1mm,` `+ HEIGHT=0.2mm,`

Table 14 Input Netlist for Two Traces Between Two Ground Planes (Continued)

Listing Type	Field Solver Ground Planes Example

Top and bottom ground planes	`.LAYERSTACK stack_1,`
	`+ LAYER=(copper,1mm), LAYER=(diel_1,2mm),`
	`+ LAYER=(diel_2,3mm), LAYER=(copper,1mm)`

Option settings	`.FSOPTIONS opt1 PRINTDATA=YES`

Two conductors share the same shape	`.MODEL cond2_sys W MODELTYPE=FieldSolver,`
	`+ LAYERSTACK=stack_1, FSOPTIONS=opt1`
	`RLGCFILE=ex3.rlgc`
	`+ CONDUCTOR=(SHAPE=rect,MATERIAL=copper, ORIGIN=(0,3mm)),`
	`+ CONDUCTOR=(SHAPE=rect,MATERIAL=copper, ORIGIN=(1.2mm,3mm))`

Analysis, outputs and end	`.TRAN 0.5n 100n`
	`.PROBE v(out1)`
	`.END`

Table 15 compares the computed result with the Finite Element (FEM) solver result.

Table 15 Comparison Between Computed and FEM Solver Results

Computed	$\begin{bmatrix} 214.1 & -105.2 \\ -105.2 & 214.1 \end{bmatrix}$ (pF/m)
FEM Solver	$\begin{bmatrix} 217.7 & -108.2 \\ -108.2 & 217.7 \end{bmatrix}$ (pF/m)

Example 4: Using Field Solver with Monte Carlo Analysis

The following example shows how to use Monte Carlo transient analysis to model variations in the manufacturing of a microstrip.

The transient output waveforms are shown in Figure 45 on page 166.

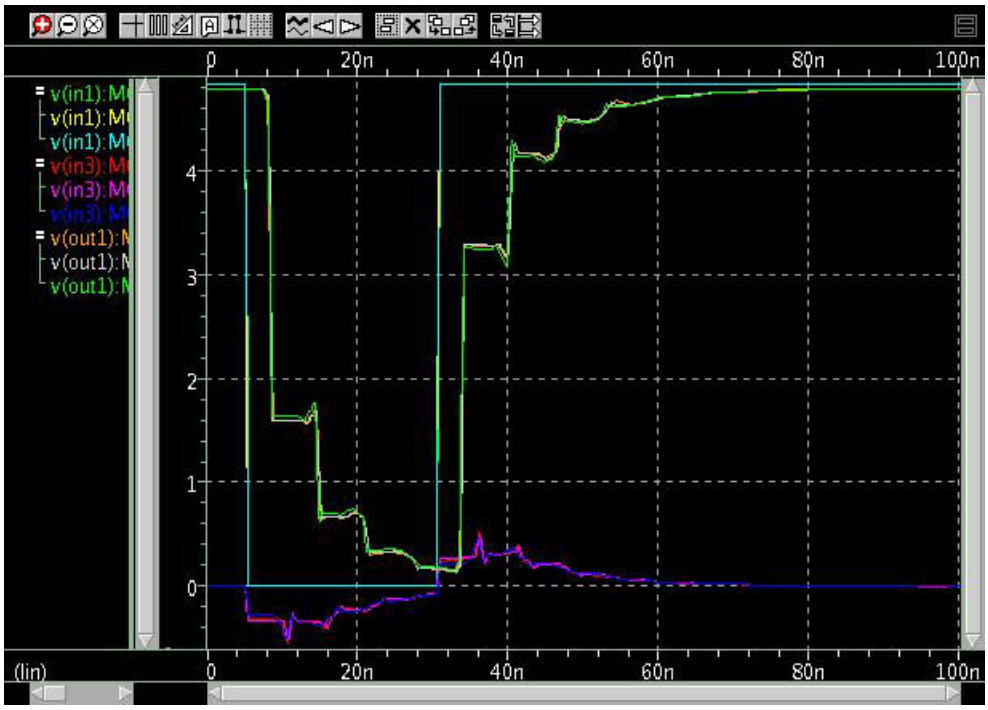


Figure 45 Monte Carlo Analysis with Field Solver and W-element

Table 15 shows the input listing with the W-element.

Table 16 Input File Listing with the W-element

Listing Type	Field Solver Monte Carlo Example

Header, options and sources	`*PETL Example: with Monte Carlo`
	`.OPTION PROBE POST`
	`+ VIMPULSE in1 gnd AC=1v PULSE 4.82v 0v 5ns`
	`+ 0.5ns 0.5ns 25ns`

Parameter definitions	`.PARAM x1=Gauss(0,0.02,1) x2=Gauss(0.5mm,0.02,1)`
	`+ x3=Gauss(1mm,0.02,1)`
	`.PARAM dRef=1u dY1=Gauss(2mm,0.02,1)`
	`+ dY2=Gauss(1mm,0.02,1)`

W-element	`W1 in1 in2 in3 0 out1 out2 out3 0` `+ FSMODEL=cond3_sys N=3 l=0.5`

Table 16 Input File Listing with the W-element (Continued)

Listing Type	Field Solver Monte Carlo Example

Materials	`.MATERIAL diel_1 DIELECTRIC ER=4.3`
	`.MATERIAL diel_2 DIELECTRIC ER=3.2`
	`.MATERIAL copper METAL CONDUCTIVITY=57.6meg`

Shapes	`.SHAPE r1 RECTANGLE WIDTH=0.35mm, HEIGHT=0.070mm`
Uses the default AIR background	`.LAYERSTACK stack_1 LAYER= (copper,dRef),` `+ LAYER=(diel_1,dY1), LAYER= (diel_2,dY2)`

Three conductors share the same shape	`.MODEL cond3_sys W MODELTYPE=FieldSolver,`
	`+ LAYERSTACK=stack1,`
	`+ CONDUCTOR=(SHAPE=r1,MATERIAL=copper,ORIGIN=(x1,'dRef+dY1')),`
	`+ CONDUCTOR=(SHAPE=r1,MATERIAL=copper,ORIGIN=(x2,'dRef+dY1+dY2')),`
	`+ CONDUCTOR=(SHAPE=r1,MATERIAL=copper,ORIGIN=(x3,'dRef+dY1+dY2'))`

Analysis, outputs and end	`.PROBE TRAN v(in1) v(out1) v(in3)`
	`.PROBE AC v(out1) v(out3)`
	`.PROBE DC v(in1) v(out1) v(out3)`
	`.AC LIN 200 0Hz 0.3GHz`
	`.DC VIMPULSE 0v 5v 0.01v`
	`.TRAN 0.5ns 100ns SWEEP MONTE=3`
	`.END`

Example 5: Modeling Coaxial and Shielded Twin-Lead Lines Using the Polar Field Solver

The following examples show how to model a coaxial line and a twin-lead line. The keyword coord=polar (or coord=1) invokes the polar field solver. When the polar field solver is used, the conductor position is defined in polar coordinates (radius, angle in degrees). Only one dielectric is permitted and the dielectric layer is surrounded by ground.

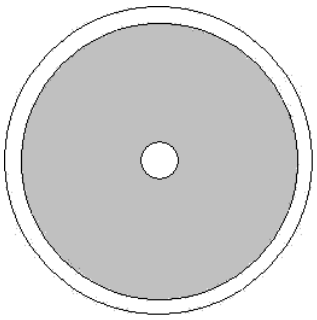


Figure 46 Polar field solver for modeling coaxial lines

Coax Line

```
*PETL Example: Coaxial Line
.OPTION PROBE POST
VIMPULSE in1 gnd AC=1v PULSE 4.82v 0v 5ns
+0.5ns 0.5ns 25ns

*W element
W1 in1 gnd out1 gnd FSMODEL=coax N=1, L=1
R1 out1 gnd 50

* [[ Material List ]]
.MATERIAL diel_1 DIELECTRIC ER=4
.MATERIAL copper METAL CONDUCTIVITY=57.6meg

* [[ Shape List ]]
.SHAPE circle_1 CIRCLE RADIUS=0.5m

* [[ Layer Stack ]]
.LAYERSTACK coaxial LAYER=(diel_1 11m) $ only one

* [[ Field solver option ]]
.FSOPTIONS myOpt printdata=yes compute_rs=yes compute_gd=yes
+ compute_go=yes

* [[ Field solver model ]]
.MODEL coax W MODELTYPE=FIELDSOLVER FSOPTIONS=myOpt COORD=polar
+ LAYERSTACK=coaxial, RLGCFILE=coax.rlgc
+ CONDUCTOR = ( SHAPE=circle_1, MATERIAL=copper, ORIGIN=(0, 0) )

.TRAN 0.5n 100n
.PROBE v(in1) v(out1)
.END
```

Shielded Twin-Lead Line

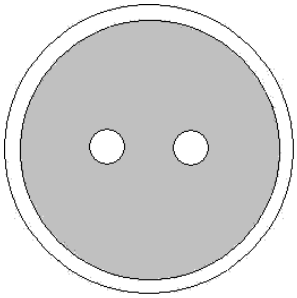


Figure 47 Polar field solver for modeling twin-lead line

```
*PETL Example: Shield twin-lead lines
.OPTION PROBE POST
VIMPULSE in1 gnd AC=1v PULSE 4.82V 0v 5ns
+0.5ns 0.5ns 25ns
*W element
W1 in1 in2 0 out1 out2 0 FSMODEL=twin, N=2, L=1
R1 out1 gnd 50
R2 out2 gnd 50
R3 in2 gnd 50

* [[ Material List ]]
.MATERIAL diel_1 DIELECTRIC ER=4
.MATERIAL copper METAL CONDUCTIVITY=57.6meg

* [[ Shape List ]]
.SHAPE circle_1 CIRCLE RADIUS=0.5m

* [[ Layer Stack ]]
.LAYERSTACK coaxial LAYER=(diel_1 11m)) $ only one

* [[ Field solver option ]]
.FSOPTIONS myOpt printdata=yes compute_rs=yes compute_gd=yes
   compute_go=yes
* [[ Field solver model ]]
.MODEL twin W MODELTYPE=FIELDSOLVER FSOPTIONS=myOpt COORD=polar
+ LAYERSTACK=coaxial, RLGCFILE=twin.rlgc
+ CONDUCTOR = ( SHAPE=circle_1, MATERIAL=copper,
   ORIGIN=(4.5m, 0) )
+ CONDUCTOR = ( SHAPE=circle_1, MATERIAL=copper,
   ORIGIN=(4.5m, 180) )

.TRAN 0.5n 100n
.PROBE v(in1) v(out1) v(out2)
.END
```

W-element Passive Noise Model

The W-element is a passive transmission line model. When the transmission lines are lossy, they generate thermal noise. The W-element passive noise model is used to describe these noise effects. This model supports normal, 2-port and multi-port (`.NOISE` and `.LIN noisecalc=1` [or `2` for N-port]). See Noise Parameters in 2-Port and N-Port Networks.

Input Interface

To trigger a passive noise model, the `NOISE` and `DTEMP` keywords in an W-element statement are used:

```
W i1 i2 ... iN iR o1 o2 ... oN oR N=val L=val
+ ...
+ [NOISE=[1|0]] [DTEMP=val]
```

Parameter	Description
NOISE	Activates thermal noise. ▪ `1 (default)`: element generates thermal noise ▪ `0`: element is considered noiseless
DTEMP	Temperature difference between the element and the circuit, expressed in °C. The default is 0.0. Element temperature is calculated as: T = Element temperature (°K) = 273.15 (°K) + circuit temperature (°C) + DTEMP (°C) Where circuit temperature is specified using either the `.TEMP` statement, or by sweeping the global `TEMP` variable in `.DC`, `.AC`, or `.TRAN` statements. When a `.TEMP` statement or `TEMP` variable is not used, the circuit temperature is set by `.OPTION TNOM`, which defaults to 25 °C unless you use `.OPTION SPICE`, which raises the default to 27 °C.

When `NOISE=1`, HSPICE generates a 2N×2N noise-current correlation matrix from the N-conductor W-element admittance matrix according to Twiss' Theorem. The result can be stamped into an HSPICE noise analysis as 2N-correlated noise current sources: j_i (i=1~2N), as shown below:

$$C = 2kT(Y + Y^{*T}) = \begin{bmatrix} \overline{|j_1|^2} & \overline{j_1 j_2^*} & \cdots & \overline{j_1 j_{2N}^*} \\ \overline{j_2 j_1^*} & \overline{|j_2|^2} & \cdots & \overline{j_2 j_{2N}^*} \\ \cdots & \cdots & \cdots & \cdots \\ \overline{|j_{2N} j_1^*|} & \overline{|j_{2N} j_2^*|} & \cdots & \overline{|j_{2N}|^2} \end{bmatrix}$$

Where,

i=1~N corresponding to N input terminals

i=N+1~2N corresponding to N output terminals.

The noise-current correlation matrix represents the frequency-dependent statistical relationship between 2N noise current sources, j_i (i=1~2N), shown in the following figure.

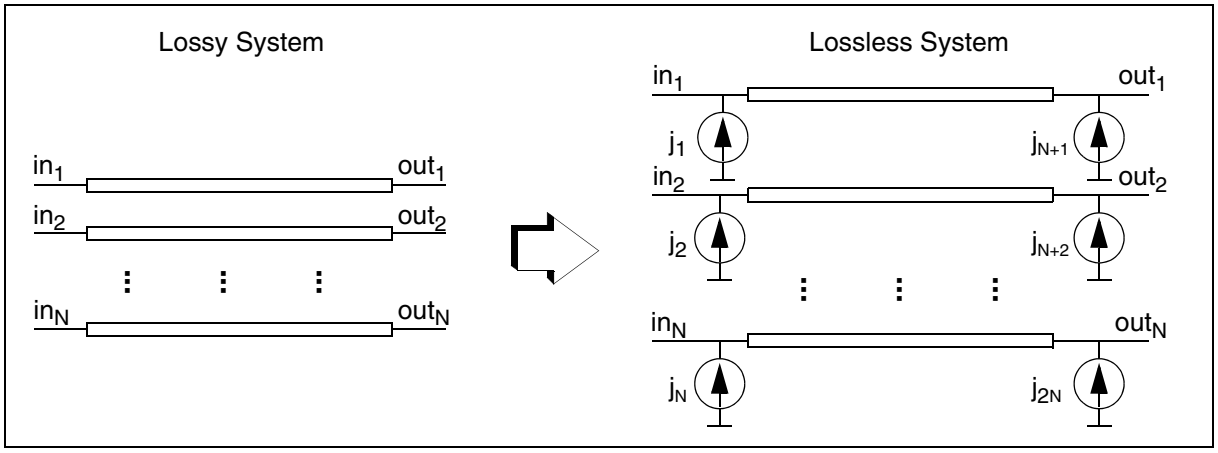


Figure 48 Frequency-dependent relationship, 2N noise current sources

Output Interface

HSPICE creates a *.lis* output list file that shows the results of a noise analysis just as any other noisy elements. The format is as follows:

```
**** w element squared noise voltages (sq v/hz)

    element          0:w1
     N(i,j)          data
   r(N(i,j))          data
    ... i,j = 1~N ...
      total          data
```

Where:

- `N(i,j)` = contribution of $j_1 j_j^*$ to the output port
- `r(N(i,j))` = transimpedance of j_i to the output port
- `total` = contribution of total noise voltage of the W-element to the output port.

Using the TxLine (Transmission Line) Tool Utility

This section describes how to use the W-element GUI utility, TxLine Tool for creating transmission line models. TxTool is built into the CosmosScope binary but HSPICE users do not need a CosmosScope license key to run TxTool.

Note: For an alternative means to visualize field solver results using Tcl/tk, see Visualizing Cross-Sectional Geometric Information on page 157.

The TxLine tool supports GUI-driven creation and characterization of models of systems of coupled transmission lines. The tool allows you to create models of many types of simply connected systems of transmission lines from 2-D geometrical description of the system cross-section, material properties, and length specifications. The TxLine Tool may be used to create models of interacting conductors in a cable or IC interconnect systems. The tool and model solution algorithms provide the essential functional capability of the HSPICE W-element, and allow RLGC model descriptions to be generated that are suitable for simulation in HSPICE.

The following sections discuss these topics:

- Invoking the TxLine Tool
- Getting Started with TxLine Tool

Invoking the TxLine Tool

You can download the latest CosmosScope binary (or use any CosmosScope binary you have previously downloaded) and run the TxLine tool.

To invoke the utility on a UNIX or Linux command-line, enter:

```
% txTool
```

To invoke the utility on Windows OS, double-click the binary *txTool.exe* under the *$INSTALL_CSCOPE*/ai_bin directory.

Getting Started with TxLine Tool

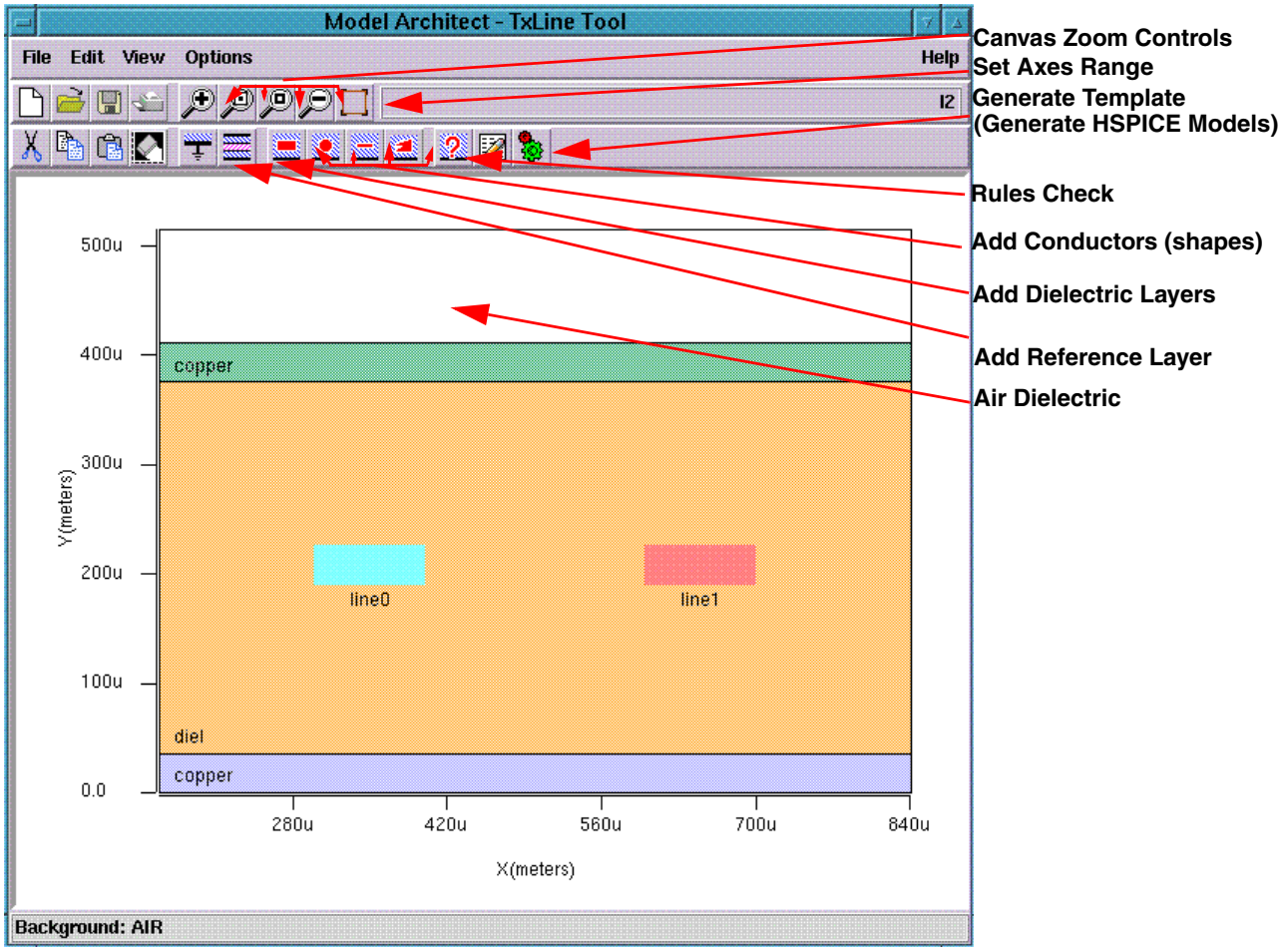


Figure 49 TxLine Tool: controls called out

The TxLine utility has a XY graphical canvas work area where you assemble the 2D geometrical description of your coupled transmission lines.

The tool's free space white background canvas is "air" (dielectric), and is called out in a text status line located in bottom of window, when the mouse cursor rolls over this region. If you mouse over of any inserted dielectric layers and conductors, this status line provides information on their properties. It also reports Rules Check information to verify the validity of your 2-D transmission line system.You must first lay down a reference plane (ground layer), then sequentially add other dielectric layers and conductors in any order. Double-clicking on inserted graphical objects displays a Geometry Attributes form, and a right mouse click pops up a general property menu which contains more options for editing and setting material properties.

The default material for the reference plane and conductors is Cu, and you may specify different material properties such as conductivity, permittivity, and permeability in the Material Property form available under the general property menu.

The Options > Field Solver allows you to set W-element field solver options.

The Options > Instance Parameters allows you to set the transmission line system length (default = 1 meter), name (if not already set in a previous Save or SaveAs operation), and several specialized parameters.

The Options > Model Save Selections allows you to specify your preference for either HSPICE models or MAST templates (models) and symbols to be generated.

You can generate and view HSPICE models and MAST templates by clicking the Generate Template icon button.

The model can then be saved (File > Save As...).

A *.ai_txline file is saved along with all requested files. The 2D transmission line geometry and field solver attributes may be restored by loading this file into the TxLine tool.

For full details about the HSPICE Field Solver, refer to the section "Extracting Transmission Line Parameters (Field Solver)".

To learn about the MAST language, refer to the *Saber® MAST Language User Guide* and *Saber® MAST Language Reference Manual*.

References

[1] Luca Daniel and Joel Philips, “Model Order Reduction for Strictly Passive and Causal Distributed Systems,” proceeding of DAC, June, 2002

[2] [2] Colin Gordon, Thomas Blazeck, and Raj Mittra, “Time-Domain Simulation of Multiconductor Transmission Lines with Frequency-Dependent Losses,” IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN, VOL. I I. NO. 11, pp. 1372-1387, NOVEMBER 1992

[3] [3] Qingiian Yu and Omar Wing, “Computational Models of Transmission Lines with Skin Effects and Dielectric Loss,” IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-I: FUNDAMENTAL THEORY AND APPLICATIONS. VOL. 41, NO. 2, pp. 107-119, FEBRUARY 1994

[4] Eric Bogatin, “Signal Integrity Simplified”, Prentice Hall

[5] Bjorn Gustavsen and Adam Semlyen, “Rational Approximation of Frequency Domain Responses by Vector Fitting,” IEEE Transaction on Power Delivery, Vol.14, No.3, pp. 1052-1061, July 1999

[6] Ravi Kollipara, et al “Practical Design Considerations for 10 to 25 Gbps Copper Backplane Serial Links,” DesignCon 2006, Santa Clara, CA, U.S.A.

[7] E. Hammerstad and O. Jensen, “Accurate models for microstrip computer aided design,” IEEE MTT-S Int. Microwave Symp. Dig., 1980,. pp. 407-409.

4

Modeling Input/Output Buffers Using IBIS Files

Describes how IBIS model components are included in an HSPICE simulation, with usage models for IBIS buffer, component package, board-level components (EBD), and Interconnect Specification (ICM). (HSPICE RF does not support IBIS.)

HSPICE ships numerous of examples for your use. Find paths to IBIS-related demo files under IBIS Examples.

For information about IBIS-Algorithmic Modeling Interface (AMI) usage with HSPICE, see Modeling Complex Linear and Non-Linear Equalizers Using AMI in Chapter 18, Statistical Eye Analysis, *HSPICE User Guide: Simulation and Analysis*.

The Input/Output Buffer Information Specification (IBIS) specifies a standard ASCII format for presenting information describing the behavior of various I/O buffers that send electrical signals outside the silicon chip or receive such signals.

IBIS promotes tool independent I/O models for system level Signal Integrity work and has become:

- The ANSI/EIA-656 and IEC 62014-1 standard for describing the analog behavior of the buffers of digital devices using plain ASCII text-formatted data.
- An alternative to models—IBIS files are not models, they contain the data to be used by the behavioral models and algorithms inside the simulator.

Except for SPICE or Verilog-A formatted buffers, IBIS specifies formats for the following types of information:

- Output I-V curves for output buffers in LOW and HIGH states
- V(t) curves describing the exact form of transitions from LOW to HIGH states and from HIGH to LOW states for a given load
- Values for die capacitance

The IBIS standard was developed by the IBIS Open Forum, affiliated with the Electronic Industries Alliance (EIA). IBIS specifies only the “form” for the information; it does not specify how the information is processed or used by the simulator.

These topics are discussed in the following sections:

- Verifying IBIS Files with the Golden Parser
- Using IBIS Buffer 'Models'
- IBIS Syntax Conventions for I/O Buffers
- Buffer Types
- Specifying Required and Optional Common Keywords
- Differential Pins
- Buffers in Subcircuits
- Netlist Example with Output Buffer, Transmission Line, and Input Buffer
- Using the IBIS Component Command
- Using IBIS Package Modeling
- Using IBIS Board-Level Components
- Using IBIS Interconnect Modeling (ICM)

For introductory information on using IBIS with HSPICE, see Simulating Circuits with IBIS Models in HSPICE in Chapter 1 of this user guide.

Verifying IBIS Files with the Golden Parser

The IBIS standard contains a section devoted to recommendations on how to derive information from either the simulation or silicon measurement. The IBIS Open Forum has sponsored development of a parser for IBIS files — called the golden parser. The golden parser is freely available as an executable and should be used for verification of IBIS files. The golden parser is incorporated into Synopsys circuit simulators. When processing an IBIS file, the golden

parser produces warnings or error messages that appear in the HSPICE output by default.

Using IBIS Buffer 'Models'

This section describes IBIS buffer usage in HSPICE. The I/O buffer element type is called *buffer* and in HSPICE is commonly referred to as the B-element or b-element.

Using buffers is similar to using other simulation elements, such as transistors: specify a name for the buffer, specify a list of nodes that connect the buffer to the rest of the circuit, and specify parameters. Only parameters that specify a model for the buffer (file name and model name) are required. The IBIS model name is limited to 60 characters.

Except for SPICE or Verilog-A formatted buffers, there are two significant differences from the use of other elements:

- You can specify a total of 3 to 8 external nodes depending on the buffer type
- If nodes are supposed to connect to power or ground rails, do not connect them in the netlist because the simulation connects the nodes by default.

This chapter is not intended to introduce the IBIS standard, because it is a large document; familiarity with the standard is assumed. A significant amount of information is available on the Internet.

The official IBIS Open Forum web site is located at:

http://www.eigroup.org/ibis/

This site contains articles introducing IBIS, text of the IBIS standard, examples of IBIS files, and tools such as the golden parser. The site also links to other web sites devoted to IBIS.

Three types of analysis are supported for input/output buffers:

- DC analysis
- Transient analysis
- AC analysis

IBIS Syntax Conventions for I/O Buffers

The general syntax of an element card for I/O buffers is:

```
bxxx node_1 node_2 ... node_N
+ file='filename' model='model_name'
+ keyword_1=value_1 ... [keyword_M=value_M]
+ M=num
```

Parameter	Description
b*xxx*	Buffer element name. The name must begin with B, which can be followed by up to 1023 alphanumeric characters.
node_1 node_2 ... node_N	List of I/O buffer external nodes. The number of nodes and corresponding rules are specific to different buffer types.
file='filename'	Name of the IBIS file.
model='model_name'	Name of the model.
keyword_i=value_i	Assigns the *value_i* value to the *keyword_i* keyword. Specify optional keywords in brackets ([ ]) (see Specifying Required and Optional Common Keywords on page 207 for more information).
M=*num*	Multiplier: Instance parameter where *num* is a positive integer.

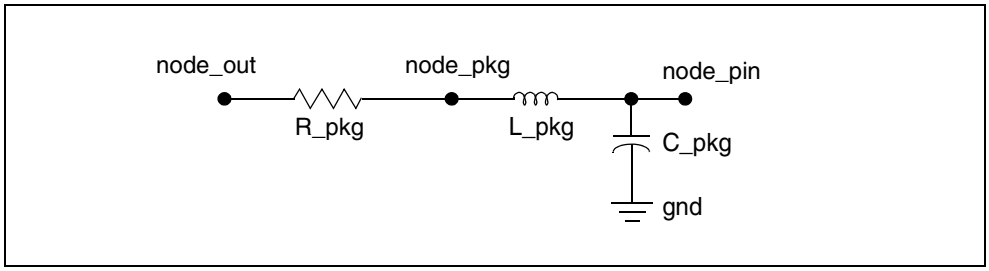


Figure 50 Circuit Diagram for Package

The gnd node on the circuit diagram for buffers denotes the ideal SPICE ground node (the node 0 [zero] notation is also used). This node is always

available in the simulation device models. Do not include this node in the node list on the buffer card. If the gnd node appears on a circuit diagram, simulation connects the node to the ideal ground. The gnd node on circuit diagrams explains the connection of individual parts inside buffers.

In some cases, buffer nodes have different rules than nodes for other elements. Some nodes might already be connected to voltage sources (simulation makes such connections) so do not connect a voltage source to such nodes. Conversely, some nodes should be connected to voltage sources and you need to connect voltage sources to these nodes.

Note: See Specifying Required and Optional Common Keywords on page 207 and the sections about individual buffer types for detailed explanations on how to use these nodes.

Buffers correspond to models in IBIS files and do not include packages. In this case, you need to manually add the corresponding packages. For example, if node_out and node_pin are nodes for output of the output buffer and corresponding pin, then add the following lines to the netlist:

```
R_pkg node_out node_pkg R_pkg_value
L_pkg node_pkg node_pin L_pkg_value
C_pkg node_pin gnd  C_pkgvalue
```

The preceding lines use the IBIS file to find the `R_pkg`, `L_pkg`, and `C_pkg` values (see Figure 50 on page 180 for the circuit diagram).

Troubleshooting Signal Propagation Issues

If you connect a source to your buffer using the correct nodes and do not see any output or have a distorted waveform, your input voltage swings may be out of range, or your input stimulus (digital signal) is switching too fast, such that “overclocking” occurs.

In the case of overclocking, you may see a “jump” when a transition is triggered before the previous one is completed. This shows up as an artificial skew in simulation.

Terminology

The following terms are frequently used in this chapter:

Table 17 Terminology Used in This Chapter

Term	Definition
card, buffer card	Denote lines from the netlist that specifies the buffer name (should begin with the letter b), a list of external nodes, required keyword, and optional keywords.
buffer, I/O buffer, input/output buffer	One of 14 IBIS models as specified in the standard, version 3.2, and implemented in the Synopsys IBIS device models.
RWF, FWF	Rising waveform, falling waveform
I/O	Input/Output
I/V curve	Current-voltage curve
PU, PD	Pullup, pulldown
PC, GC	Power clamp, ground clamp

Buffer Types

This section describes buffers as used in the Synopsys IBIS device models. Refer to Specifying Required and Optional Common Keywords on page 207 for details about how to use keywords that are in the syntax examples in the following sections.

The following sections discuss these topics:

- Input Buffer
- Output Buffer
- Tristate Buffer
- Input/Output Buffer
- Open Drain, Open Sink, Open Source Buffers
- I/O Open Drain, I/O Open Sink, I/O Open Source Buffers

- Input ECL Buffer
- Output ECL Buffer
- Tristate ECL Buffer
- Input-Output ECL Buffer
- Terminator Buffer
- Series Buffer
- Series Switch Buffer
- Multilingual Model Support

Input Buffer

The syntax of an input buffer element card is:

```
B_INPUT nd_pc nd_gc nd_in nd_out_of_in
+ file='filename' model='model_name'
+ [typ={typ|min|max|fast|slow}] [power={on|off}]
+ [buffer={1|input}]
+ [interpol={1|2}]
+ [nowarn]
+ [c_com_pc=c_com_pc_value]
+ [c_com_gc=c_com_gc_value]
+ [pc_scal=pc_scal_value]
+ [gc_scal=gc_scal_value]
```

In the preceding syntax, the total number of external nodes is 4.

If you specify the `power=on` keyword (default), the *nd_pc* and *nd_gc* nodes are connected to voltage sources with values taken from the IBIS file. Such voltage sources are created internally by HSPICE. So do *not* connect the nodes of *nd_pc* and *nd_gc* to other voltage sources. Names for these nodes are provided for you to print out the voltage values, if required.

For example:

```
.PRINT V(nd_pc) V(nd_gc)
```

If you specify the `power=off` keyword, the simulation does not connect these nodes to voltage sources. You need to connect the nodes to voltage sources directly through an RLC network, or through a transmission line.

You can connect node_in to I, E, F, G, and H -elements. The buffer measures and processes the voltage on this node and sends a response to the

nd_out_of_in node. The *nd_out_of_in* node is connected to the voltage source as shown in Figure 51 on page 184. It is an error to connect this node to a voltage source. If `power=off`, you can connect the *nd_pc* and *nd_gc* nodes to the ground, which sets the voltage to zero on these nodes.

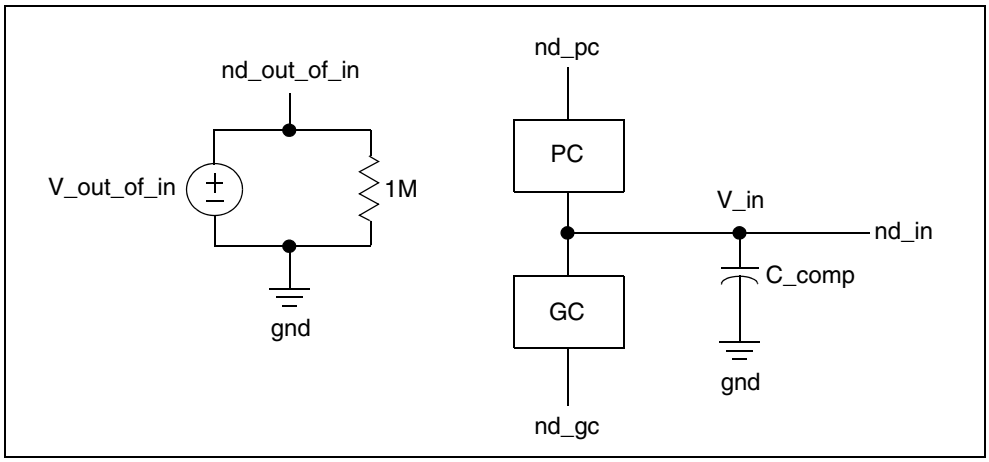


Figure 51 Input Buffer

V_out_of_in is a digital signal that assumes values of either 0, 0.5, or 1 depending on the V_in, Vinh, Vinl, and Polarity voltages. The simulation processes V_out_of_in according to the following rules.

Table 18 IBIS Input Buffer

If:	Then:
Polarity=Non-Inverting	Initially V_out_of_in is set to 0 if V_in < (Vinh+Vinl)/2 and to 1 in the opposite case.
V_in>Vinh	V_out_of_in is set to 1
V_in<Vinl	V_out_of_in is set to 0
else Vinl<=V_in<=Vinh	V_out_of_in is set to 0.5
Polarity=Inverting	Initially V_out_of_in is set to 0 if V_in > (Vinh+Vinl)/2 and to 1 in the opposite case
V_in>Vinh	V_out_of_in is set to 0
V_in<Vinl	V_out_of_in is set to 1

Table 18 IBIS Input Buffer

If:	Then:
else Vinl<=V_in<=Vinh	V_out_of_in is set to 0.5

Figure 51 on page 184 shows a single circuit specified on a single element card. V_out_of_in is a voltage source whose value is a function of V_in (and of Vinl, Vinh thresholds, and Polarity parameter). You can use it to drive other circuits.

If you specify `pc_scal` or `gc_scal` arguments, and *pc_scal_value* or *gc_scal_value* do not equal 1.0, then HSPICE uses the *pc_scal_value* or *gc_scal_value* to adjust the PC or GC I/V curve.

Output Buffer

The syntax for an output buffer element card is:

```
B_OUTPUT nd_pu nd_pd nd_out nd_in [nd_pc nd_gc]
+ file='file_name' model='model_name'
+ [typ={typ|min|max|fast|slow}] [power={on|off}]
+ [buffer={2|output}]
+ [xv_pu=state_pu] [xv_pd=state_pd]
+ [interpol={1|2}]
+ [ramp_fwf={0|1|2}] [ramp_rwf={0|1|2}]
+ [fwf_tune=fwf_tune_value] [rwf_tune=rwf_tune_value]
+ [nowarn]
+ [c_com_pu=c_com_pu_value]
+ [c_com_pd=c_com_pd_value]
+ [c_com_pc=c_com_pc_value]
+ [c_com_gc=c_com_gc_value]
+ [pu_scal=pu_scal_value]
+ [pd_scal=pd_scal_value]
+ [pc_scal=pc_scal_value]
+ [gc_scal=gc_scal_value]
+ [rwf_scal=rwf_scal_value]
+ [fwf_scal=fwf_scal_value]
+ [spu_scal=spu_scal_value]
+ [spd_scal=spd_scal_value]
```

The *nd_pc* and *nd_gc* nodes are optional. However, you can specify both nodes or none of them. The total number of external nodes is either 4 or 6; any

other number is an error. If you do not specify the *nd_pc* and *nd_gc* nodes on the element card, but Power_Clamp or Ground_Clamp I/V curves are present in the model in question, then the simulator simply connects Power_Clamp or Ground_Clamp to the corresponding *nd_pu* (pullup) or *nd_pd* (pulldown).

However, you need the optional *nd_pc* and *nd_gc* nodes if:

- The POWER Clamp Reference and GND Clamp Reference IBIS keywords are present in the IBIS model and have different values than the IBIS keywords Pullup Reference and Pulldown Reference, or
- The Pullup Reference and Pulldown Reference IBIS keywords do not exist and POWER Clamp Reference and GND Clamp Reference have different values than those determined by the Voltage Range IBIS keyword.

If your circuit needs the *nd_pc* and *nd_gc* optional nodes, but they are not in the element card, the simulation connects *nd_pc* to *nd_pu* and *nd_gc* to *nd_pd* and issues this warning:

```
**warning** nodes of pc and pu(or gc and pd) are merged, although
they have different reference voltage!
```

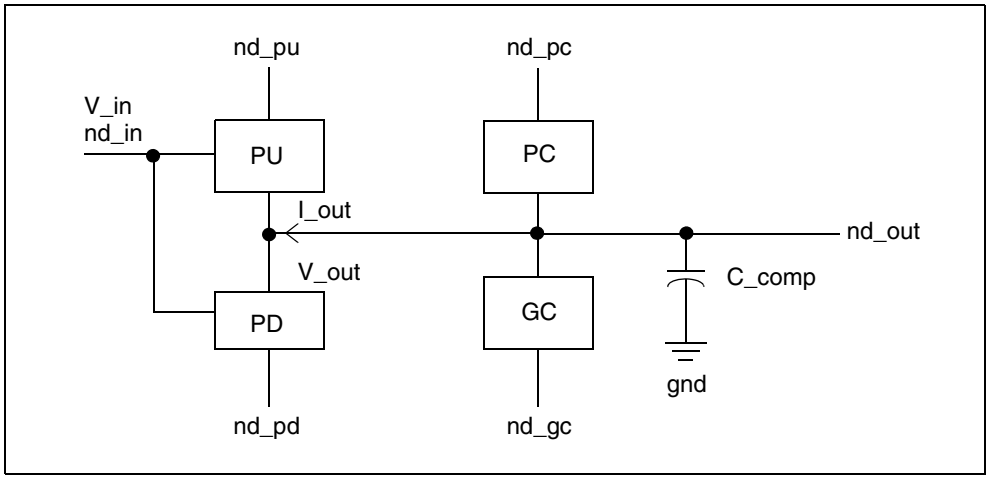


Figure 52 Output Buffer

If you specify the `power=on` (default) keyword, then the *nd_pu* and *nd_pd* nodes, and if specified, the *nd_pc* and *nd_gc* nodes, are connected to voltage sources with values taken from the IBIS file. Such voltage sources are created internally by HSPICE. So do *not* connect the nodes of *nd_pu* and *nd_pd*, (and, if specified, *nd_pc* and *nd_gc*) to other voltage sources. Specify names for these nodes so you can print out

the voltage values if required. For example:

```
.PRINT V(nd_pu) V(nd_pd)
```

If you specify the `power=off` keyword, the simulation does not connect these nodes to voltage sources. Connect the nodes to voltage sources directly, through an RLC network, or through a transmission line.

No special rules apply for the `nd_out` node. The voltage on this node is controlled by the digital signal on the *nd_in* node. You can connect any voltage source, current source, voltage controlled voltage source, voltage controlled current source, current controlled voltage source, or current controlled current source to the *nd_in* node as shown in the following example:

```
V_in nd_in gnd 0V pulse(0V 1V 1n 0.1n 0.1n 7.5n 15n)]
```

If `power=off`, you can connect the *nd_pu*, *nd_pd*, *nd_pc*, and *nd_gc* nodes to the ground if you want zero voltage on these nodes.

V_in is a controlling signal that represents a digital signal with values 0 and 1. However, the simulation can use any signal and process, according to the following rules:

Table 19 IBIS Output Buffer

If:	Then:
Polarity=Non-inverting	At t=0 for transient analysis (or for DC analysis), the buffer goes to HIGH state if V_in > 0.5 and to LOW in the opposite case. Next, if the buffer is in HIGH state, it goes to LOW state if V_in < 0.2. If the buffer is in LOW state, it goes to HIGH state if V_in > 0.8.
Polarity=Inverting	At t=0 for transient analysis (or for DC analysis), the buffer goes to HIGH state if V_in < 0.5 and to LOW in the opposite case. Next, if the buffer is in HIGH state, it goes to LOW state if V_in > 0.8. If the buffer is in LOW state, it goes to HIGH state if V_in < 0.2.

If the `pc_scal` (or `gc_scal`, `pu_scal`, `pd_scal`) argument exists and the `pc_scal_value` (or `gc_scal_value`, `pu_scal_value`, `pd_scal_value`) does not equal 1.0, then the simulation adjusts the PC (or GC, PU, PD) I/V

curve using the pc_scal_value (or gc_scal_value, pu_scal_value, pd_scal_value).

If the rwf_scal (or fwf_scal) argument exists and rwf_scal_value (or fwf_scal_value) does not equal 1.0, then the simulation adjusts the rising and falling vt curves using the rwf_scal_value (or fwf_scal_value).

If the spu_scal (or spd_scal) argument exists and spu_scal_value (or spd_scal_value) is greater than 0 and power=off and V(nd_pu)-V(nd_pd) does not equal the corresponding value in the *ibs* file, then the simulation adjusts the I/V curves of PU (or PD) using the spu_scal_value (or spd_scal_value).

Probing Current Flowing through C_comp when Using a B-element

While you cannot probe the current through a C_comp directly, you can use this workaround:

1. Move C_comp to C_fixture in VT waveforms (i.e., set C_fixture = C_fixture + C_comp and C_comp = 0).
2. Add a capacitor (with value = C_comp) from the output node of the B-element to Ground.
3. Probe the current flowing through the added capacitor.

Tristate Buffer

The syntax for a tristate buffer element card is:

```
B_3STATE nd_pu nd_pd nd_out nd_in nd_en [nd_pc nd_gc]
+ file='file_name' model='model_name'
+ [typ={typ|min|max|fast|slow}] [power={on|off}]
+ [buffer={4|three_state}]
+ [xv_pu=state_pu] [xv_pd=state_pd]
+ [interpol={1|2}]
+ [ramp_fwf={2|1|0}] [ramp_rwf={2|1|0}]
+ [fwf_tune=fwf_tune_value] [rwf_tune=rwf_tune_value]
+ [nowarn]
+ [c_com_pu=c_com_pu_value]
+ [c_com_pd=c_com_pd_value]
+ [c_com_pc=c_com_pc_value]
+ [c_com_gc=c_com_gc_value]
+ [pu_scal=pu_scal_value]
+ [pd_scal=pd_scal_value]
+ [pc_scal=pc_scal_value]
```

```
+ [gc_scal=gc_scal_value]
+ [rwf_scal=rwf_scal_value]
+ [fwf_scal=fwf_scal_value]
+ [spu_scal=spu_scal_value]
+ [spd_scal=spd_scal_value]
```

The *nd_pc* and *nd_gc* nodes are optional, and you can specify either both nodes or none of them. The total number of external nodes is either 5 or 7; any other number is an error. If the element card does not specify the *nd_pc* and *nd_gc* nodes, but the model contains Power_Clamp or Ground_Clamp I-V curves, then the simulator adds Power_Clamp or Ground_Clamp I-V curves data to the corresponding Pull_Up or Pull_Down I-V curves data.

However, you need the *nd_pc* and *nd_gc* optional nodes if:

- The POWER Clamp Reference and GND Clamp Reference IBIS keywords are present in the IBIS model and have different values than the IBIS keywords Pullup Reference and Pulldown Reference, or
- The Pullup Reference and Pulldown Reference IBIS keywords do not exist and POWER Clamp Reference and GND Clamp Reference have different values than those determined by the Voltage Range IBIS keyword.

If your circuit needs *nd_pc* and *nd_gc* optional nodes, but they are not in the element card, the simulation issues a warning and connects *nd_pc* to *nd_pu* and *nd_gc* to *nd_pd*.

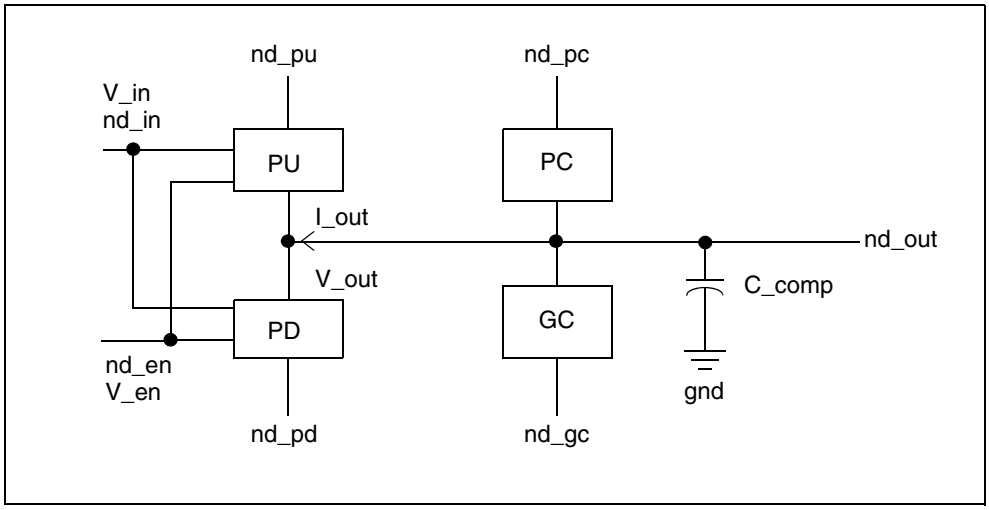


Figure 53 Tristate Buffer

If you specify the `power=on` (default) keyword, then the *nd_pu* and *nd_pd* nodes, (and, if specified, *nd_pc* and *nd_gc*), are connected to voltage sources with values taken from the IBIS file. Do not connect these nodes to voltage sources.

However, specify names for these nodes in the netlist so you can print out the voltage values if required. For example:

```
.PRINT V(nd_pu) V(nd_pd)
```

If you specify the `power=off` keyword, the simulation does not connect these nodes to voltage sources. Connect the nodes to voltage sources directly through an RLC network, or through a transmission line.

No special rules apply for the *nd_out* node. The voltage on this node is controlled by the digital signal on the *nd_in* and *nd_en* nodes. Voltage sources must be connected to the nodes *nd_in*, nd_en as shown in the following example:

```
V_in nd_in gnd 0V pulse( 0V 1V 1n 0.1n 0.1n 7.5n 15n )
V_en nd_en gnd 0V pulse( 0V 1V 3n 0.1n 0.1n 7.5n 15n )
```

You can connect the *nd_pu*, *nd_pd*, *nd_pc*, and *nd_gc* nodes to the ground if you want to have zero voltage on these nodes. The *nd_in* and *nd_en* nodes cannot be connected to the ground.

V_in and V_en are controlling signals representing digital signals with 0 and 1 values. The simulation can use any signal and process according to the rules in Table 20. The enable signal, V_en, supersedes the input signal V_in.

Table 20 IBIS Tristate Buffer

If:	Then:
ENABLE = Active-High	At t=0 for transient analysis (or for DC analysis), the buffer goes to the ENABLE state if V_en > 0.5 and to DISABLE in the opposite case.
ENABLE = Active-Low	At t=0 for transient analysis (or for DC analysis), the buffer goes to ENABLE state if V_en < 0.5 and to DISABLE in the opposite case.
The buffer is in ENABLE state	Begins transition to DISABLE state if V_en < 0.2 (where Enable = Active-High) and if V_en > 0.8 (where Enable = Active-Low).

Table 20 IBIS Tristate Buffer (Continued)

If:	Then:
The buffer is in DISABLE state or in transition from ENABLE to DISABLE state	Begins transition to ENABLE state if V_en > 0.8 (where Enable = Active-High) and if V_en < 0.2 (where Enable = Active-Low).
The buffer is in ENABLE state	Response to the input signal, V_in, is the same as the output buffer.
Polarity=Non-Inverting	At t=0 for transient analysis (or for DC analysis), the buffer goes to HIGH state if V_in > 0.5 and to LOW in the opposite case. Next, if the buffer is in HIGH state, it goes to LOW state if V_in < 0.2. If the buffer is in LOW state, it goes to HIGH state if V_in > 0.8.
Polarity=Inverting	At t=0 for transient analysis (or for DC analysis), the buffer goes to HIGH state if V_in < 0.5 and to LOW in the opposite case. Next, if the buffer is in HIGH state, it goes to LOW state if V_in > 0.8. If the buffer is in LOW state, it goes to HIGH state if V_in < 0.2.

Input/Output Buffer

The syntax of an input/output buffer element card is:

```
B_IO nd_pu nd_pd nd_out nd_in nd_en nd_out_of_in [nd_pc
   nd_gc]
+ file='file_name' model='model_name'
+ [typ={typ|min|max|fast|slow}] [power={on|off}]
+ [buffer={3|input_output}]
+ [xv_pu=state_pu] [xv_pd=state_pd]
+ [interpol={1|2}]
+ [ramp_fwf={2|1|0}] [ramp_rwf={2|1|0}]
+ [fwf_tune=fwf_tune_value] [rwf_tune=rwf_tune_value]
+ [nowarn]
+ [c_com_pu=c_com_pu_value]
+ [c_com_pd=c_com_pd_value]
+ [c_com_pc=c_com_pc_value]
+ [c_com_gc=c_com_gc_value]
+ [pu_scal=pu_scal_value]
```

```
+ [pd_scal=pd_scal_value]
+ [pc_scal=pc_scal_value]
+ [gc_scal=gc_scal_value]
+ [rwf_scal=rwf_scal_value]
+ [fwf_scal=fwf_scal_value]
+ [spu_scal=spu_scal_value]
+ [spd_scal=spd_scal_value]
```

The *nd_pc* and *nd_gc* nodes are optional. However, you can specify either both nodes or none of them. The total number of external nodes is either 6 or 8; any other number is an error. If the element card does not specify the *nd_pc* and *nd_gc* nodes, but the model contains Power_Clamp or Ground_Clamp I-V curves, then the simulator adds Power_Clamp or Ground_Clamp I-V curves data to the corresponding Pull_Up or Pull_Down I-V curves data.

However, you need the *nd_pc* and *nd_gc* optional nodes if:

- The IBIS POWER Clamp Reference and GND Clamp Reference keywords are present in the IBIS model and have different values than the Pullup Reference and Pulldown Reference IBIS keywords, or
- The IBIS Pullup Reference and Pulldown Reference keywords do not exist and POWER Clamp Reference and GND Clamp Reference have different values than those determined by the Voltage Range IBIS keyword.

If you need the *nd_pc* and *nd_gc* optional nodes, and you omitted them from the element card, the simulation issues a warning and connects *nd_pc* to *nd_pu* and *nd_gc* to *nd_pd*.

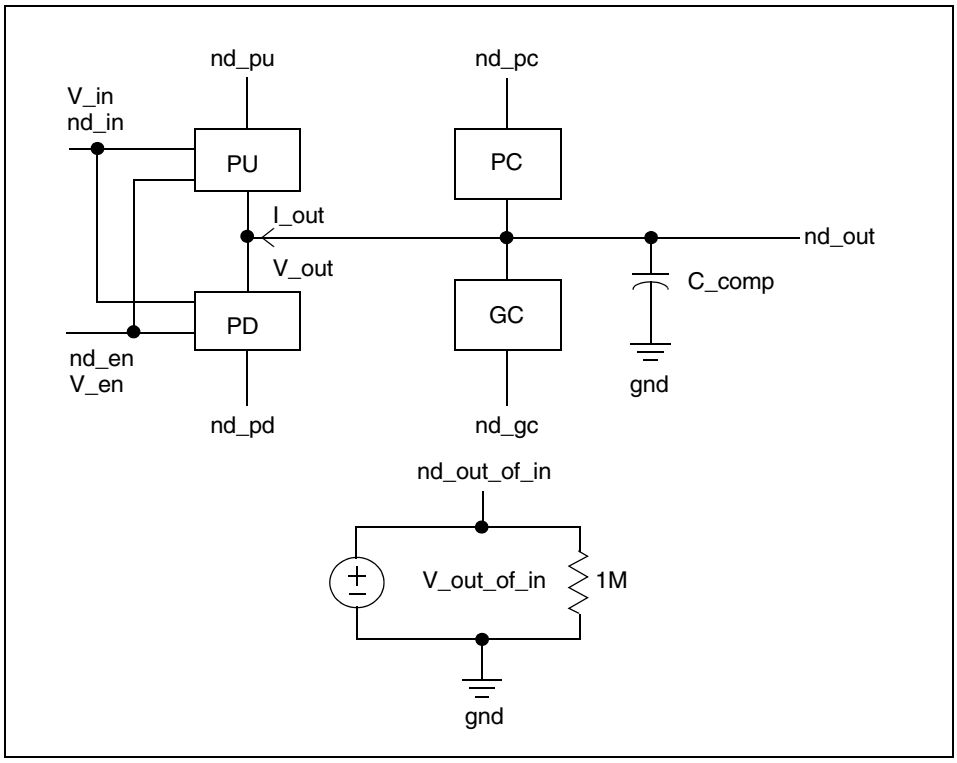


Figure 54 Input-Output Buffer

If you specify the `power=on` (default) keyword, then the *nd_pu* and *nd_pd* nodes, (and if specified, *nd_pc* and *nd_gc*), are connected to voltage sources with values taken from the IBIS file. Such voltage sources are created internally by HSPICE. So do *not* connect the nodes of *nd_pu* and *nd_pd*, (and, if specified, *nd_pc* and *nd_gc*) to other voltage sources. However, you should specify names for these nodes in the netlist so you can print out the voltage values if required. For example:

```
.PRINT V(nd_pu) V(nd_pd)
```

If you specify the `power=off` keyword, the simulation does not connect these nodes to voltage sources. You should connect the nodes to voltage sources directly through an RLC network, or through a transmission line.

No special rules apply for the *nd_out* node. The voltage on this node is controlled by the digital signal on the *nd_in* and *nd_en* nodes. Voltage sources must be connected to the *nd_in* and *nd_en* nodes as shown in the following example:

```
V_in nd_in gnd 0V pulse (0V 1V 1n 0.1n 0.1n 7.5n 15n)
V_en nd_en gnd 0V pulse (0V 1V 3n 0.1n 0.1n 7.5n 15n)
```

You can connect the *nd_pu*, *nd_pd*, *nd_pc*, and *nd_gc* nodes to the ground if you want zero voltage on these nodes.

The *nd_out_of_in* node is connected to a voltage source (see Figure 54 on page 193). Connecting this node to a voltage source or the ground causes an error to occur.

The input-output buffer is a combination of the tristate buffer and the input buffer. See Input Buffer on page 183 and Tristate Buffer on page 188 for more information.

The input-output buffer can function as an input buffer. In this case, the resultant `V_out_of_in` digital signal on the *nd_out_of_in* node is controlled by the `V_out` voltage on the *nd_out* node.

For the input buffer, this controlling voltage is called `V_in` and any corresponding node is called *nd_in*.

The input-output buffer uses `V_in` and *nd_in* notations to denote the controlling voltage and controlling input node for the output part of the buffer.

If the input-output buffer is not in the DISABLE state (this includes ENABLE state and transitions to ENABLE->DISABLE and DISABLE->ENABLE), then it functions as a tristate buffer. If the input-output buffer is in the DISABLE state, it functions as an input buffer.

However, there is a difference in the digital output of the input part of the buffer (voltage `V_out_of_in`). `V_out_of_in` is not always defined (for example, if the buffer is in ENABLE state, or Vinl < V_out < Vinh at the time moment when the transition to DISABLE state is completed). Also, you need to preserve logical LEVELs 0 and 1 for LOW and HIGH states. Therefore, `V_out_of_in` uses the value 0.5 when it is undefined.

Figure 54 on page 193 shows a single circuit specified on a single element card. The `V_out_of_in` is a voltage source whose value is a function of `V_out` (of the Vinl and Vinh thresholds and the Polarity parameter). It can be used to drive other circuits.

Open Drain, Open Sink, Open Source Buffers

Open drain and open sink buffers do not include pullup circuitry. Open source buffers do not include pulldown circuitry. However, the element cards for these three buffers coincide with the element card for the output buffer. Accordingly,

you should always specify names for pullup and pulldown nodes, *nd_pu* and *nd_pd*, even if the buffer does not include pullup or pulldown circuitry.

All rules in Output Buffer on page 185 apply to open drain, open sink, and open source buffers with the following exceptions:

- Because open drain and open sink buffers do not have pullup circuitry, do not specify the `xv_pu=nd_state_pu` option.
- Similarly, because open source buffers have no pulldown circuitry, do not specify the `xv_pd=nd_state_pd` option.

I/O Open Drain, I/O Open Sink, I/O Open Source Buffers

I/O open drain and I/O open sink buffers do not include pullup circuitry. I/O open source buffers do not include pulldown circuitry. However, the element cards for these three buffers coincide with the element card for the input-output buffer. Accordingly, you should always specify names for pullup and pulldown nodes, *nd_pu* and *nd_pd*, even if the buffer does not include pullup or pulldown circuitry.

All rules in Input/Output Buffer on page 191 apply to I/O open drain, I/O open sink, and I/O open source buffers with the following exceptions:

- Because I/O open drain and I/O open sink buffers do not have pullup circuitry, do not specify the `xv_pu=nd_state_pu` option.
- Similarly, because I/O open source buffers do not include pulldown circuitry, do not specify the `xv_pd=nd_state_pd` option.

Input ECL Buffer

The syntax of the input ECL buffer element card is:

```
B_INPUT_ECL nd_pc nd_gc nd_in nd_out_of_in
+ file='file_name' model='model_name'
+ [typ={typ|min|max|fast|slow}] [power={on|off}]
+ [buffer={11|input_ecl}]
+ [interpol={1|2}]
+ [nowarn]
+ [c_com_pc=c_com_pc_value]
+ [c_com_gc=c_com_gc_value]
+ [pc_scal=pc_scal_value]
+ [gc_scal=gc_scal_value]
```

The input ECL buffer is similar to the input buffer. The only difference is in default values for Vinl and Vinh.

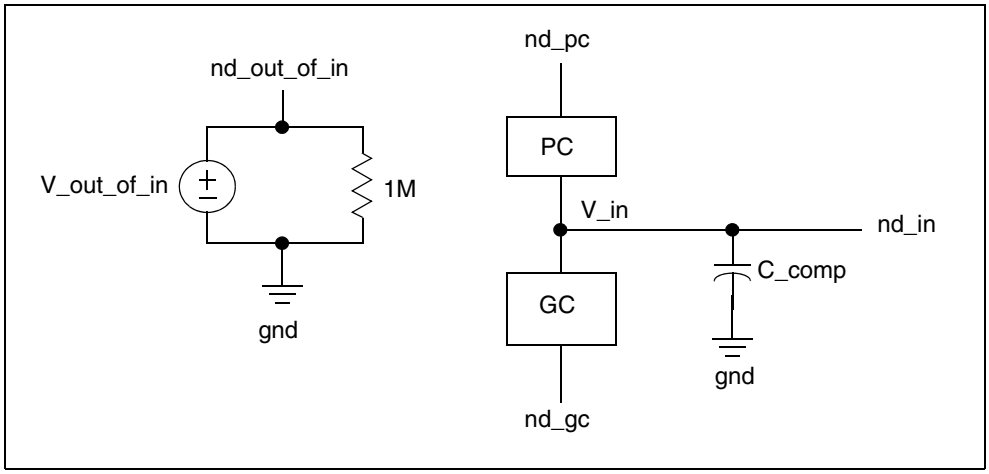


Figure 55 Input ECL Buffer

Output ECL Buffer

The syntax of the output ECL buffer element card is:

```
B_OUTPUT_ECL nd_pu nd_out nd_in [nd_pc nd_gc]
+ file='file_name' model='model_name'
+ [typ={typ|min|max|fast|slow}] [power={on|off}]
+ [buffer={12|output_ecl}]
+ [xv_pu=state_pu] [xv_pd=state_pd]
+ [interpol={1|2}]
+ [ramp_fwf={0|1|2}] [ramp_rwf={0|1|2}]
+ [fwf_tune=fwf_tune_value] [rwf_tune=rwf_tune_value]
+ [nowarn]
+ [c_com_pu=c_com_pu_value]
+ [c_com_pc=c_com_pc_value]
+ [c_com_gc=c_com_gc_value]
+ [pu_scal=pu_scal_value]
+ [pd_scal=pd_scal_value]
+ [pc_scal=pc_scal_value]
+ [gc_scal=gc_scal_value]
+ [rwf_scal=rwf_scal_value]
+ [fwf_scal=fwf_scal_value]
+ [spu_scal=spu_scal_value]
```

```
+ [spd_scal=spd_scal_value]
```

The *nd_pc* and *nd_gc* nodes are optional. However, you can specify both nodes or none of them. The total number of external nodes is either 3 or 5; any other number is an error. The output ECL buffer does not have a pulldown node. The pulldown table in the IBIS file is referenced in respect to pullup voltage.

If you do not specify the *nd_pc* and *nd_gc* nodes on the element card, but the Power_Clamp or Ground_Clamp I-V curves are present in the model in question, then the simulator issues an error message (this simulator behavior is different from that for the output buffer).

In other respects, the output ECL buffer is similar to the output buffer. For more information, see Output Buffer on page 185.

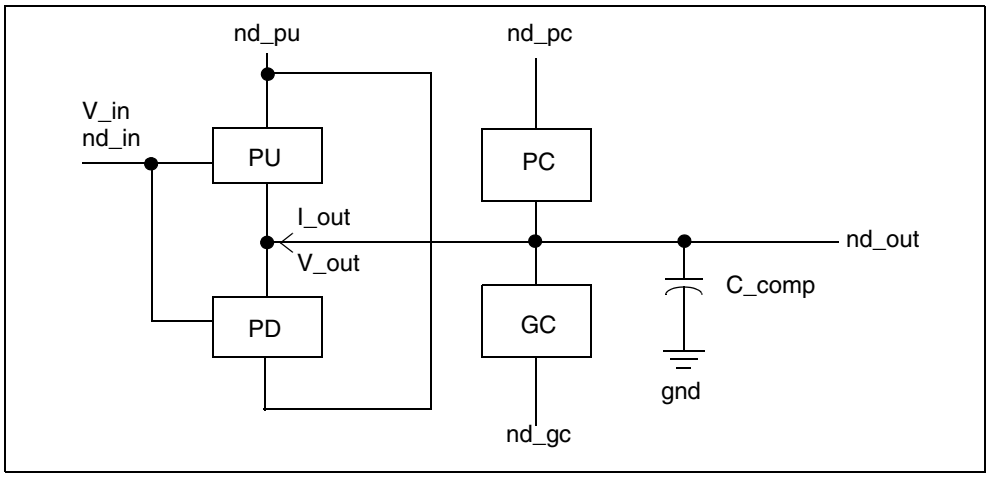


Figure 56 Output ECL Buffer

Tristate ECL Buffer

The syntax for the tristate ECL buffer element card is:

```
B_3STATE_ECL nd_pu nd_out nd_in nd_en [nd_pc nd_gc]
+ file='file_name' model='model_name'
+ [typ={typ|min|max|fast|slow}] [power={on|off}]
+ [buffer={14|three_state_ecl}]
+ [xv_pu=state_pu] [xv_pd=state_pd]
+ [interpol={1|2}]
+ [ramp_fwf={2|1|0}] [ramp_rwf={2|1|0}]
```

```
+ [fwf_tune=fwf_tune_value] [rwf_tune=rwf_tune_value]
+ [nowarn]
+ [c_com_pu=c_com_pu_value]
+ [c_com_pc=c_com_pc_value]
+ [c_com_gc=c_com_gc_value]
+ [pu_scal=pu_scal_value]
+ [pd_scal=pd_scal_value]
+ [pc_scal=pc_scal_value]
+ [gc_scal=gc_scal_value]
+ [rwf_scal=rwf_scal_value]
+ [fwf_scal=fwf_scal_value]
+ [spu_scal=spu_scal_value]
+ [spd_scal=spd_scal_value]
```

The *nd_pc* and *nd_gc* nodes are optional. However, either both or none can be specified. The total number of external nodes is either 4 or 6, any other number is an error. The tristate ECL buffer does not have a pulldown node. The pulldown table in the IBIS file is referenced in respect to pullup voltage.

If you do not specify the *nd_pc* and *nd_gc* nodes on the element card, but the Power_Clamp or Ground_Clamp I-V curves are present in the model in question, then the simulator issues an error message (this simulator behavior is different from that for the tristate buffer).

In other respects, the tristate ECL buffer is similar to the tristate buffer. See Tristate Buffer on page 188 for more information.

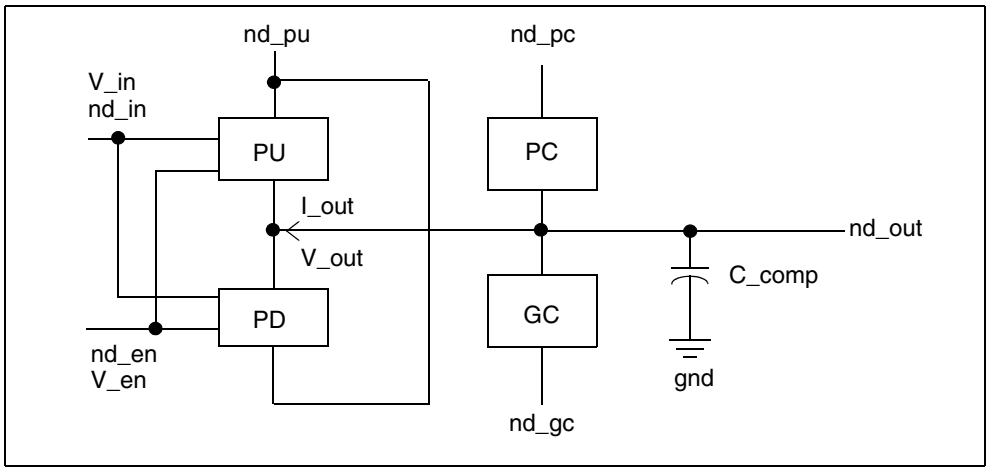


Figure 57 Tristate ECL Buffer

Input-Output ECL Buffer

The syntax for the input-output ECL buffer element card is:

```
B_IO_ECL nd_pu nd_out nd_in nd_en nd_out_of_in [nd_pc nd_gc]
+ file='file_name' model='model_name'
+ [typ={typ|min|max|fast|slow}] [power={on|off}]
+ [buffer={13|io_ecl}]
+ [xv_pu=state_pu] [xv_pd=state_pd]
+ [interpol={1|2}]
+ [ramp_fwf={2|1|0}] [ramp_rwf={2|1|0}]
+ [fwf_tune=fwf_tune_value] [rwf_tune=rwf_tune_value]
+ [nowarn]
+ [c_com_pu=c_com_pu_value]
+ [c_com_pc=c_com_pc_value]
+ [c_com_gc=c_com_gc_value]
+ [pu_scal=pu_scal_value]
+ [pd_scal=pd_scal_value]
+ [pc_scal=pc_scal_value]
+ [gc_scal=gc_scal_value]
+ [rwf_scal=rwf_scal_value]
+ [fwf_scal=fwf_scal_value]
+ [spu_scal=spu_scal_value]
+ [spd_scal=spd_scal_value]
```

The *nd_pc* and *nd_gc* nodes are optional. However, you can specify either both nodes or none of them. The total number of external nodes is either 5 or 7; any other number is an error. The tristate ECL buffer does not include a pulldown node. The pulldown table in the IBIS file is referenced in respect to pullup voltage.

If you specify the *nd_pc* and *nd_gc* nodes on the element card but Power_Clamp or Ground_Clamp I-V curves are present in the model in question, then the simulator issues an error message (this simulator behavior is different from the Input-Output buffer).

In other respects, the input-output ECL buffer is similar to the input-output buffer. See Input/Output Buffer on page 191 for more information.

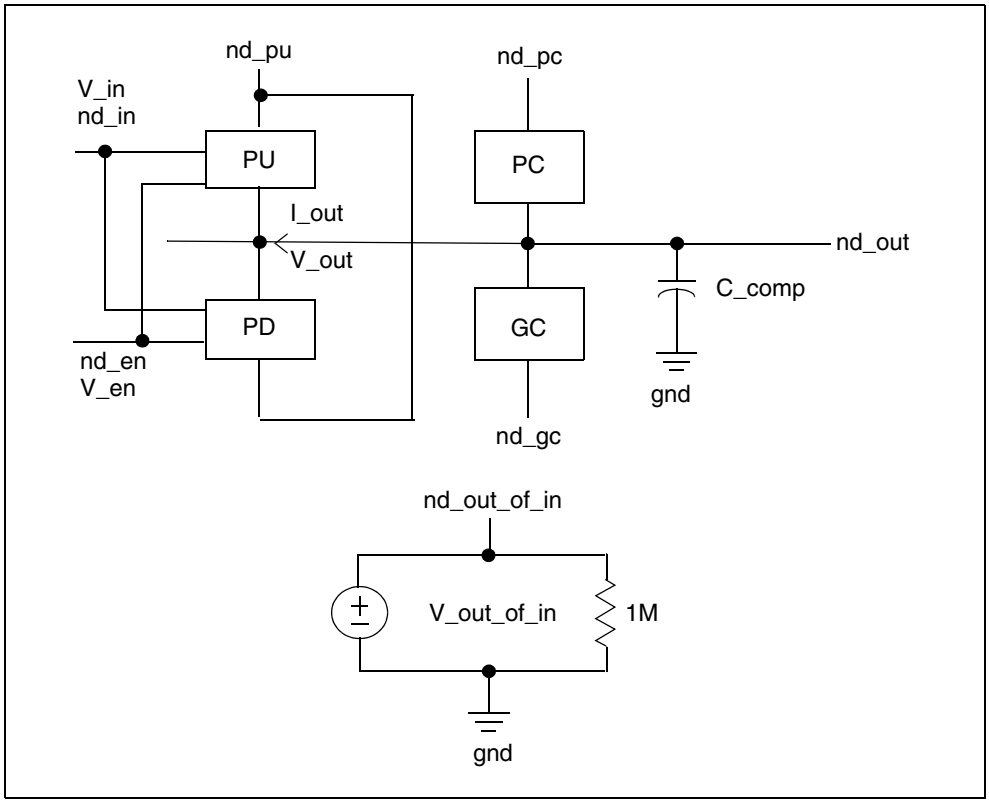


Figure 58 Input-Output ECL Buffer

Terminator Buffer

The syntax for the terminator buffer element card is:

```
b_TERMINATOR nd_pc nd_gc nd_out
+ file='filename' model='model_name'
+ [typ={typ|min|max|fast|slow}] [power={on|off}]
+ [buffer={17|terminator}]
+ [interpol={1|2}]
+ [nowarn]
+ [c_com_pc=c_com_pc_value]
+ [c_com_gc=c_com_gc_value]
+ [pc_scal=pc_scal_value]
+ [gc_scal=gc_scal_value]
```

In the preceding syntax, the total number of external nodes is 3. If you specify the `power=on` keyword (default), the *nd_pc* and *nd_gc* nodes are connected to

voltage sources with values taken from the IBIS file. Such voltage sources are created internally by HSPICE. So do *not* connect the nodes of *nd_pc* and *nd_gc* to other voltage sources. Names for these nodes are provided so you can print out the voltage values if required.

The schematic in Figure 59 shows a terminator buffer:

```
.PRINT V(nd_pc) V(nd_gc)
```

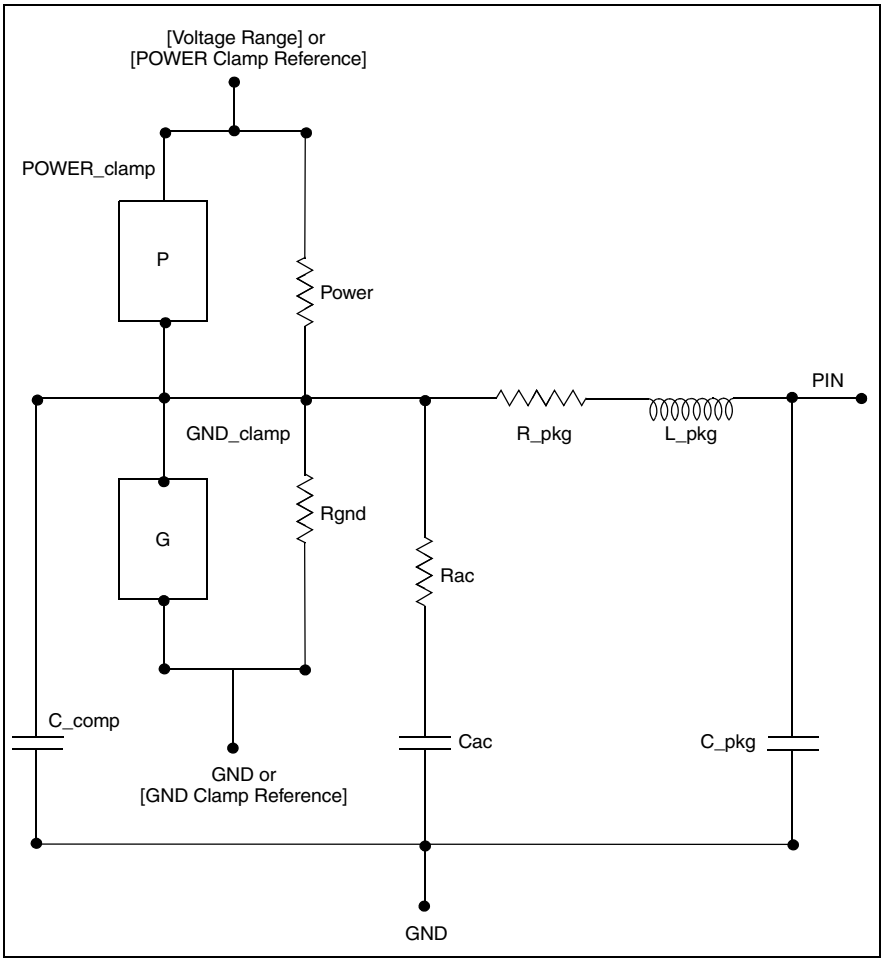


Figure 59 Terminator buffer schematic

If you specify the `power=off` keyword, the simulation does not connect these nodes to voltage sources. You need to connect the nodes to voltage sources directly through an RLC network or transmission line. You can connect node_in to I-, E-, F-, G-, and H-elements.

Note: Rac and Cac are the resistance and capacitance values for an AC terminator.

Series Buffer

The syntax for the series buffer element card is:

```
b_SERIES nd_in nd_out
+ file='filename' model='model_name'
+ [typ={typ|min|max|fast|slow}]
+ [buffer={15|series}]
+ [interpol={1|2}]
+ [nowarn]
+ [all_sm={0|1}]
```

In the preceding syntax, the total number of external nodes is 2. This buffer type is for series models that can be described by the following model keywords:

- [R Series]
- [L Series]
- [Rl Series]
- [C Series]
- [Lc Series]
- [Rc Series]
- [Series Current]
- [Series MOSFET]

Figure 60 on page 204 shows a schematic for the series buffer.

Series Switch Buffer

The syntax for the series switch buffer element card is:

```
b_SER_SW nd_in nd_out
+ file='filename' model='model_name'
+ [typ={typ|min|max|fast|slow}]
+ [buffer={16|series_switch}]
+ [ss_state={on|off}]
+ [interpol={1|2}]
```

```
+ [nowarn]
+ [all_sm={0|1}]
```

In this syntax, the total number of external nodes is 2. This buffer type is for series switch models which can be described by the following model keywords:

- [On]
- [Off]
- [R Series]
- [L Series]
- [RI Series]
- [C Series]
- [Lc Series]
- [Rc Series]
- [Series Current]
- [Series MOSFET]

Voltage Thresholds

Voltages applied to the input and enable nodes are digital signals. They should be either 0 or 1. You can specify input voltage as:

```
V_in nd_in 0 pulse (0 3.3 0 0.5n 0.5n 4n 8n)
```

However, IC circuit simulation currently detects only two thresholds, 20% and 80% of [0,1] swing, that is, 0.2V and 0.8V. If a buffer is non-inverting and in a LOW state, it starts the transition to a HIGH state, if V_in > 0.8V. If the buffer is in HIGH state, it starts the transition to LOW state, if V_in < 0.2V. Specifying input voltage in the range [0, 3.3V] as in the above example does not make LOW -> HIGH transitions better in any way, but can add uncertainty over the 0.5ns time interval when the transition actually occurs.

The schematic in Figure 60 is similar to a series switch buffer, except that the Vgs-Ids table is not available in the OFF state.

The `all_sm` subparameter is optional. It is used to control the method of the Series MOSFET. When all_sm=0, only the first Vgs-Ids table (vds!=0) is used. HSPICE uses the following formula to implement this method:

ids = Ids(vgs, Vds) * vds/Vds

Otherwise, when all_sm=1, all Vgs-Ids tables are used for the Series MOSFET.

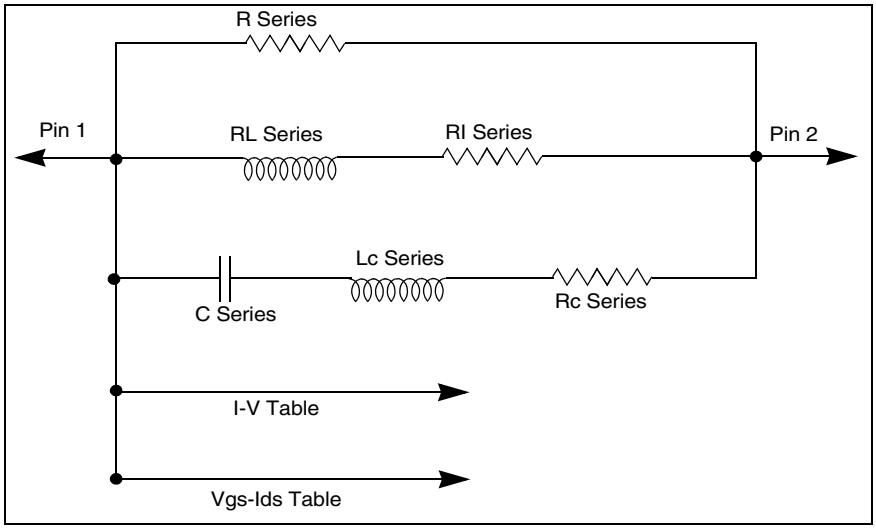


Figure 60 Series buffer schematic

Multilingual Model Support

HSPICE supports buffers of external an model described either in SPICE or in Verilog-A.

The syntax to call SPICE or Verilog-A formatted buffers is:

```
B_SPICE node1 node2 node3 ...
+ file='ibis_filename' model='model_name'
+ [nd_in=node_input]
+ [nd_en=node_enable]
+ [nd_outofin=node_out_of_in]
+ [typ={typ|min|max}] [power={on|off}]
+ [nowarn]
+ [para_begin para1=value1 para2=value2
+ ...
+ para_end]
```

In the preceding syntax,

- The list of nodes must map port names declared in [Ports] of [External Model].
- Specify a [Corner] in [External Model] from which the current simulation extracts data. If `min` or `max` [Corner] is not available, `typ` [Corner] is used. The default is `typ=typ`.

- The file of subcircuit (or module) listed in [Corner] of [External Model] must be manually included in netlist if the subcircuit (or module) is used by the external model.
- Keywords para_begin/para_end are used to assign values to parameters which are declared in [External Model] with Verilog-A language. Using para_begin/para_end for a B-element which is not a [External Model] causes an error.

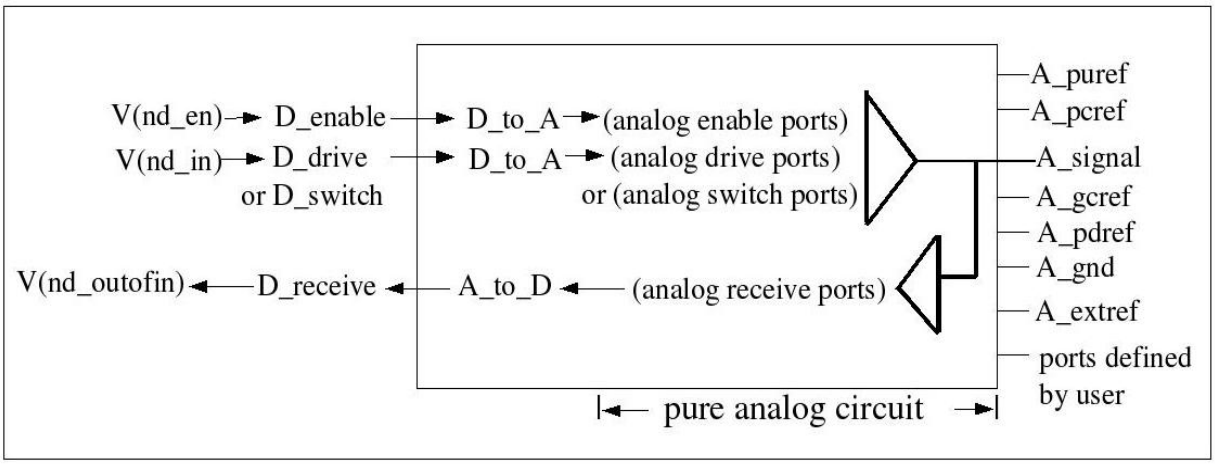


Figure 61 SPICE or Verilog-A Formatted Buffer in HSPICE

The keyword *nd_in* is only used for Output, IO, series switch, and Tri-state buffer, while *nd_en* is only used for IO or Tri-state buffer.

The two keywords specify two nodes: node_input and node_enable. They are similar to nd_in and nd_en in preceding buffers. So voltages at such nodes are controlling the signal that represents the digital signal with values 0 and 1 (see Table 19 on page 187, IBIS Output Buffer, where 1 is the signal triggering buffer to high, and 0 is the signal triggering buffer to low). This digital signal decides the voltage value between analog ports of D_drive (or D_enable) according to [D_to_A] rules in the [External Model] for DC or TRAN analysis.

The format of [D_to_A] specification is:

D_to_A D_drive port1 port2 vlow vhigh trise tfall corner_name

D_to_A D_enable port1 port2 vlow vhigh trise tfall corner_name

To illustrate the procedure, denote the digital signal from node_input (or node_enable) by D_sti.

Then, for DC or initial TRAN analysis:

V(port1,port2)=Vhigh, if D_sti =1;

V(port1,port2)=vlow, if D_sti =0.

For subsequent TRAN analysis:

V(port1,port2) changes from vlow to vhigh linearly within time trise, if D_sti switches from 0 to 1 and V(port1,port2) changes from vhigh to vlow linearly within time tfall, if D_sti switches from 1 to 0.

If polarity is inverting, preceding port1 and port2 should be swapped.

If nd_in or in_en is needed for related type of buffer but it is omitted from the buffer card, then HSPICE still adds the subcircuit or module from [External Model], but without V(port1,port2) added, and it reports a warning.

The keyword nd_outofin is only used for IO or Input buffer.The buffer measures and processes voltages between analog ports in [A_to_D] of [External Model] and sends a response to the nd_out_of_in node.The nd_out_of_in node is connected to the voltage source as shown in Figure 51 on page 184.

The digital signal V_out_of_in assumes values of 0,1 or 0.5— depending on the voltage difference of two ports, vlow, vhigh thresholds in [A_to_D] and Polarity.

The simulation processes V_out_of_in according to the following rules:

IF	THEN
Polarity=Non-Inverting	Initially V_out_of_in is set to 0, if V(port1,port2) < (vlow+vhigh)/2 and to 1 in the opposite case
V(port1,port2)>vhigh	V_out_of_in is set to 1
V(port1,port2)<vlow	V_out_of_in is set to 0
else vlow<=V(port1,port2)<=vhigh	V_out_of_in is set to 0.5
Polarity=Inverting	Initially V_out_of_in is set to 0, if V(port1,port2) > (vlow+vhigh)/2 and to 1 in the opposite case
V(port1,port2)>vhigh	V_out_of_in is set to 0
V(port1,port2)<vlow	V_out_of_in is set to 1

IF	THEN
else vlow<=V(port1,port2)<=vhigh	V_out_of_in is set to 0.5

If an IO buffer is driving, then V_out_of_in is set to 0.5.

The keywords `typ`, `power`, and `nowarn` are similar as in ordinary buffer.

Specifying Required and Optional Common Keywords

This section describes how to specify the most commonly used keywords in HSPICE. Required keywords are file and model.

If optional keywords are not used, the default values are selected. Optional keywords are enclosed in square brackets [ ] in the buffer syntax. Optional keywords can be set with a parameter and use the keyword=parameter_name format. In cases where a keyword has multiple values, [keyword=*val_1*| *val_2* | ... |*val_n*], the default value is listed first.

You cannot sweep the `typ` and `hsp_ver` keywords during analysis.

The following keywords are optional:

- file
- model
- buffer
- typ
- hsp_ver
- power
- interpol
- xv_pu | xv_pd
- ramp_fwf | ramp_rwf
- fwf_tune | rwf_tune
- rwf_pd_dly | fwf_pu_dly
- pd_scal | pu_scal | pc_scal | gc_scal | rwf_scal | fwf_scal
- ss_state

- rm_dly_rwf|rm_dly_fwf|rm_tail_rwf|rm_tail_fwf
- nowarn
- c_com_pu | c_com_pd | c_com_pc | c_com_gc
- detect_oti_mid
- time_control
- .OPTION D_IBIS

file

Syntax

```
file = 'file_name'
```

Required. Identifies the IBIS file. The file_name must be lower case and must specify either the absolute path for the file or the relative path with respect to the directory from which you run the simulator. File path names are restricted to 128 characters.

Example

```
file = '.ibis/at16245.ibs'
file = '/home/oneuser/ibis/models/abc.ibs'
```

model

Syntax

```
model = 'model_name'
```

Required. Identifies the model for a buffer from the IBIS file, specified with the file='...' keyword. The model_name keyword is case-sensitive and must match one of the models from the IBIS file. Model names are restricted to 60 characters.

Example

```
model = 'ABC_1234_out'
model = 'abc_1234_IN'
```

buffer

Syntax

```
buffer = {Buffer_Number | Buffer_Type}
```

In this syntax, buffer_number is an integer within the range $1 \leq N \leq 17$. Each buffer has an assigned number.

Table 21 IBIS Buffer Types and Numbers

Buffer Type	Buffer No.	Number of nodes (nominal or min/max)
INPUT	1	4
OUTPUT	2	4/6
INPUT_OUTPUT	3	6/8
THREE_STATE	4	5/7
OPEN_DRAIN	5	4/6
IO_OPEN_DRAIN	6	6/8
OPEN_SINK	7	4/6
IO_OPEN_SINK	8	6/8
OPEN_SOURCE	9	4/6
IO_OPEN_SOURCE	10	6/8
INPUT_ECL	11	4
OUTPUT_ECL	12	3/5
IO_ECL	13	5/7
THREE_STATE_ECL	14	4/6
SERIES	15	2
SERIES_SWITCH	16	2

Table 21 IBIS Buffer Types and Numbers (Continued)

Buffer Type	Buffer No.	Number of nodes (nominal or min/max)
TERMINATOR	17	3

The value of buffer_number and buffer_type must match the buffer type specified by the model='...' keyword. The buffer= {Buffer_Number | Buffer_Type} keyword provides an extra check for the input netlist. If you omit the keyword, this extra check is not performed.

typ

Syntax

`typ = {typ|min|max|fast|slow}`

If the value of the `typ` buffer parameter is either `typ`, `min`, or `max`, then this value signifies a column in the IBIS file from which the current simulation extracts data. The default is `typ=typ`. If min or max data are not available, `typ` data are used instead.

If the value of the `typ` buffer parameter is `fast` or `slow`, then the simulation uses certain combinations of `min` and `max` data. Table 23 on page 211 specifies the exact type of data used for `fast` and slow `values`.

The `typ` keyword can also be set to a number for a different `typ` data of an IBIS model. To use a parameter for the `typ` keyword, you can only set the parameter to `0`, `-1`, `1`, `2`, or `-2`. Table 22 shows the keywords that can be used:

Table 22 Possible Keywords for the typ Buffer Parameter

Parameter	Number
typ	0
min	-1
max	1
fast	2
slow	-2

Table 23 lists all parameters and data types for all buffers. Specific buffers use relevant data only. No buffer uses all data in the table (for example, only the terminator specifies and uses the Rgnd, Rpower, Rac, Cac parameters).

Table 23 Fast and Slow Data for IBIS Buffers

Parameter/Data	**Fast**	**Slow**
C_comp	min	max
Temp_Range	max	min
Voltage_Range	max	min
Pullup_Ref	max	min
Pulldown_Ref	min	max
POWER_Clamp_Ref	max	min
GND_Clamp_Ref	min	max
Rgnd	max	min
Rpower	max	min
Rac	max	min
Cac	min	max
Pulldown	max	min
Pullup	max	min
GND_ Clamp	max	min
POWER_Clamp	max	min
Ramp	max	min
Rising_waveform	max	min
Falling_waveform	max	min
V_fixture	max	min

hsp_ver

Syntax

```
hsp_ver = hspice_version
```

The default is the current version of the HSPICE simulator. If you prefer the previous version of the IBIS buffer, use the following statement:

```
hsp_ver = <version number>
```

power

Syntax

```
power = {on|off}
```

The default is `power=on`. To connect buffers to power sources that are specified in the IBIS file, use the Voltage Range, Pullup Reference, Pulldown Reference, POWER Clamp Reference, and GND Clamp Reference keywords.

By default, the simulation connects the required voltage sources for such external nodes as Pullup, Pulldown, Power_Clamp, and Ground_Clamp if applicable. Do not connect these nodes to voltage sources. However, you should specify names for these nodes so you can print out the voltage values if required.

If `power=off`, use internal voltage sources are not included in the buffer, and you must add external voltage sources. Use this option if the voltage source is not connected directly to buffer nodes but through a circuit to account for parasitic RLC to simulate power/ground bounce, and so on.

interpol

Syntax

```
interpol = {1|2}
```

Default is `interpol=1` (recommended). The I/V curves and V(t) curves need to be interpolated. The `interpol=1` keyword uses linear interpolation and `interpol=2` uses quadratic bi-spline interpolation.

xv_pu | xv_pd

Syntax

```
xv_pu = nd_state_pu
xv_pd = nd_state_pd
```

The buffers with output function (output, input-output, tristate, and so on) are controlled by one (input) or two (input and enable) controlling signals. To describe the state of a buffer at any moment, use two state variables, St_pu and St_pd, which vary from 0 to 1. For example:

- If the output buffer is in LOW state, then `St_pu=0`, `St_pd=1`.
- If the output buffer transitions from a LOW state to HIGH state, then `St_pu` continuously changes from 0 to 1, while `St_pd` goes from 1 to 0.

The actual time dependence for such a transition is derived from either ramp data or waveforms.

You might want to know exactly how the transition takes place. The `xv_pu=nd_state_pu`, `xv_pd=nd_state_pd` keywords provide such information. Here `nd_state_pu` and `nd_state_pd` are names of additional nodes (which must be unique, and are treated as any other node from the netlist, except for a 16-character limitation). If you include the keywords, then the simulation adds voltage sources (with 1MOhm parallel resistor).

The values of the voltages are St_pu and St_pd. You can print or display them as follows:

```
.PRINT V(nd_state_pu) V(nd_state_pd)]
```

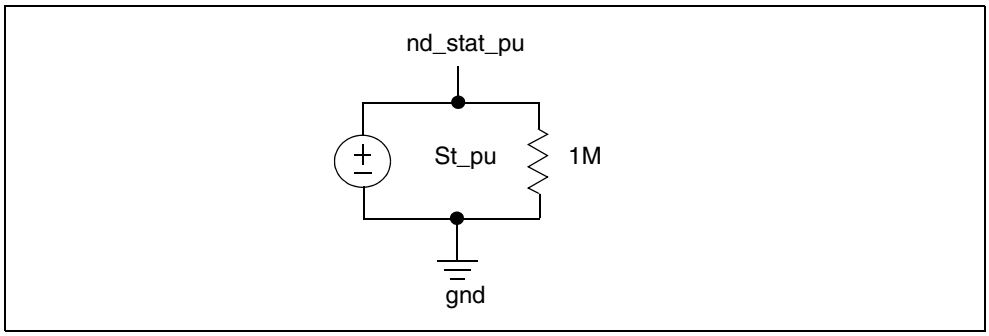


Figure 62 Equivalent Circuit for xv_pu=nd_state_pu Keyword

ramp_fwf | ramp_rwf

Syntax

```
ramp_fwf = {2|1|0}
ramp_rwf = {2|1|0}
```

Default is `ramp_fwf=2` [Falling_Waveform] and `ramp_rwf=2` [Rising_Waveform]. If ramp or waveform data are available, you can use these options to choose which data to use.

The `ramp_fwf` parameter controls falling waveforms/ramp. The `ramp_rwf` parameter controls rising waveforms/ramp.

- Value 0 denotes use ramp data.
- Value 1 denotes use of one waveform:
  - For `ramp_fwf=1`, if more than one falling waveform is available, the simulation uses the first falling waveform for the model.
  - For `ramp_rwf=1`, if more than one rising waveform is available, the simulation uses the first rising waveform for the model.
- Value 2 (default) denotes use of two waveforms:
  - For `ramp_fwf=2`, if more than two falling waveforms are available, HSPICE uses the first two falling waveforms found for the model.
  - For `ramp_rwf=2`, if more than two rising waveforms are available, HSPICE uses the first two rising waveforms found for the model.

If IC circuit simulation cannot perform a specified type of processing (for example, if you specify `ramp_fwf=2`, but only one falling waveform is found), it decrements values of `ramp_fwf` or `ramp_rwf` by one and attempts to process the new values of `ramp_fwf` and/or `ramp_rwf`. In this case, a warning is printed (unless the `nowarn` option is set).

Note: The `ramp_fwf` and `ramp_rwf` parameters are independent, and can have different values.

HSPICE selects two proper VT waveforms in a sampling method according to the initial voltages of VT waveforms and V(out) before a buffer's translation. For example: assume there are 3 rising waveforms whose initial voltage are: 0v, 0.2v, and 0.4v, respectively. Assuming at a current time point, the buffer is in low state (PU off, PD on) with V(out)=0.1v. If input is a rising edge, then the buffer starts translation from low to high. HSPICE selects the two rising

waveforms with initial voltages of 0v and 0.2v. In other words, it selects two waveforms so that $\text{Initial\_VT1} \leq V(\text{out}) \leq \text{Initial\_VT}$.

Invalid Ramp Warning

By default, HSPICE sets the parameters `ramp_rwf` and `ramp_fwf` to 2. HSPICE then looks for two [Rising Waveform] and two [Falling Wave Form] tables in the IBIS file. The invalid ramp warning is issued when you do not have the same number of tables in the IBIS file.

You can avoid this warning by setting the proper values for `ramp_rwf` and `ramp_fwf`. If you do not have any waveform tables then, set both parameters to 0 and the [Ramp] data from the IBIS file is used.

fwf_tune | rwf_tune

Syntax

```
fwf_tune = fwf_tune_value
rwf_tune = rwf_tune_value
```

The `fwf_tune_value` and `rwf_tune_value` keywords are numbers between 0 and 1. For ramp data, the default is fwf_tune=0.1 and rwf_tune=0.1; For single VT waveform, the default is fwf_tune=0.25 and rwf_tune=0.25. The `fwf_tune`, `rwf_tune` parameters specify transition time for circuitry (either pullup or pulldown) that goes from the ON to OFF state. This is specified as a fraction of time (`delta_T`) for a transition for the opposite circuitry (either pulldown or pullup) from OFF to ON state. Here, delta_T means the real transition time. For ramp data, delta_T is dT/0.6; for single VT waveform, delta_T is obtained by subtracting the non switching time of the waveform from the total time of it.

The following two parameters control the algorithm for processing ramp and a single VT waveform.

- Use `fwf_tune` only when `ramp_fwf` is 0 or 1.
- Use `rwf_tune` only when `ramp_rwf` is 0 or 1.

Figure 63 on page 216 shows the effect of these parameters when switching the output buffer from LOW to HIGH.

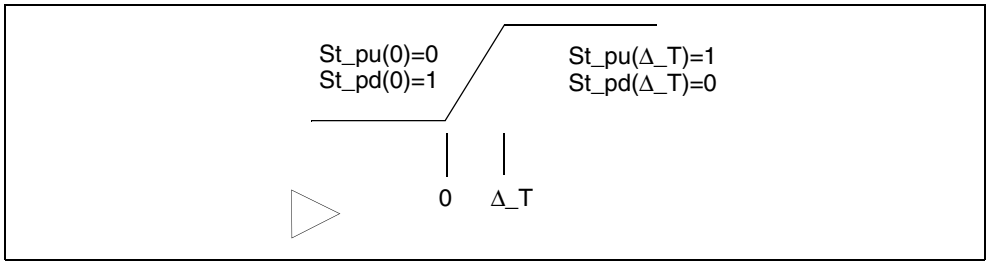


Figure 63 Change in Values of St_pu(t) and St_pd(t) When a Buffer is Switched from LOW to HIGH

Initially, St_pd=1, St_pu=0. Both ramp data and a single rising waveform provide information about the switching process. A time interval, delta_T occurs during the transition from LOW -> HIGH. The difference between the two data types (ramp and a single rising waveform) is that the shape of the waveform for ramp is fixed as a linearly growing function from LOW to HIGH. By contrast, an actual waveform accounts for an arbitrary time dependence.

However, this is not enough information to determine St_pu(t) and St_pd(t) [recall that St_pu(0)=0, St_pd(0)=1, St_pu(delta_T)=1, St_pd(delta_T)=0]. Mathematically, this represents one linear equation with two unknowns that have an infinite number of solutions. To resolve this problem, you must impose additional conditions on St_pu and St_pd.

Synopsys IBIS device models use the following approach.

The circuitry that goes from ON to OFF (for rising waveforms, pulldown circuitry) usually undergoes this transition much faster than the circuitry that goes from OFF to ON (for rising waveforms, pullup circuitry), we specify a fraction of time in units of delta_T, during which the circuitry that goes from ON to OFF undergoes the transition.

Therefore, if `rwf_tune=0.1`, then during 0.1*delta_T, the pulldown circuitry switches from ON to OFF. The transition is a linear function of time. After imposing this additional condition, you can uniquely find the rate of transition for the circuitry that goes from the OFF state to ON state.

This approach is also valid for the `fwf_tune` parameter. If resultant St_pu, St_pd are not reasonable, (e.g., they are far from the span between 0 and 1, then the preceding approach is discarded and the additional condition St_pu+St_pd=1 is used instead. In this situation, rwf_tune and fwf_tune are useless and ignored.

The fwf_tune and rwf_tune parameters are optimization parameters. The significance of these parameters strongly depends on I/V curves for pullup and pulldown circuitries. A change in fwf_tune and rwf_tune can be insignificant or very significant, depending on the I/V curves. Adjust these parameters slightly to evaluate the accuracy of the model.

If you use two waveforms, the corresponding system of equations is completely defined mathematically and the fwf_tune and rwf_tune parameters are not used (ignored if specified). However, if the data in two waveforms are inaccurate or inconsistent with other data, circuit simulation can use a single waveform or ramp data instead of two waveforms (issues a warning). If this occurs, fwf_tune and/or rwf_tune are used even if ramp_fwf=2 and ramp_rwf=2.

If the two-waveform data is inconsistent or inaccurate, the results can be less accurate than ramp or one-waveform results. You can compare that two-waveform results against ramp and one-waveform results.

You can use the xv_pu=nd_state_pu and xv_pd=nd_state_pd keywords to print or view the St_pu(t) and St_pd(t) state evolution functions.

rwf_pd_dly | fwf_pu_dly

Syntax

```
rwf_pd_dly = rwf_pd_dly_value
fwf_pu_dly = fwf_pu_dly_value
```

The rwf_pd_dly and fwf_pu_dly parameters can improve the accuracy of a single VT waveform-based IBIS buffer. They provide internal delay between external stimulus triggering and pulldown circuit turn off for rising edge, or pullup circuit turn off for falling edge.

The rwf_pd_dly_value and fwf_pu_dly_value keywords are numbers between 0 and 1. The default is the ratio of beginning non switching time in the related VT waveform to the total transition time of the VT waveform (i.e., last time point of it).

- Use rwf_pd_dly only when ramp_rwf=1
- Use fwf_pu_dly only when ramp_fwf=1

A rwf_pd_dly parameter is used with a rwf_tune parameter to provide a more real pulldown circuit turn off transition for a rising edge, like that shown in Figure 63 on page 216 for St_pd(t). Similarly, a fwf_pu_dly parameter is used with a fwf_tune parameter to provide a more real pullup circuit turn off

transition for a falling edge. In Figure 64, the internal delay is `rwf_pd_dly`*T, where T is the transition time of the single rising or falling waveform.

Note: *rwf_pd_dly_value* + r*wf_tune* should be less than 1. Otherwise, HSPICE will reset *rwf_tune* to *1-rwf_pd_dly_value*. Similarly, *fwf_pd_dly_value* + *fwf_tune* should be less than 1. Otherwise, HSPICE will reset *fwf_tune* to *1-fwf_pd_dly_value*.

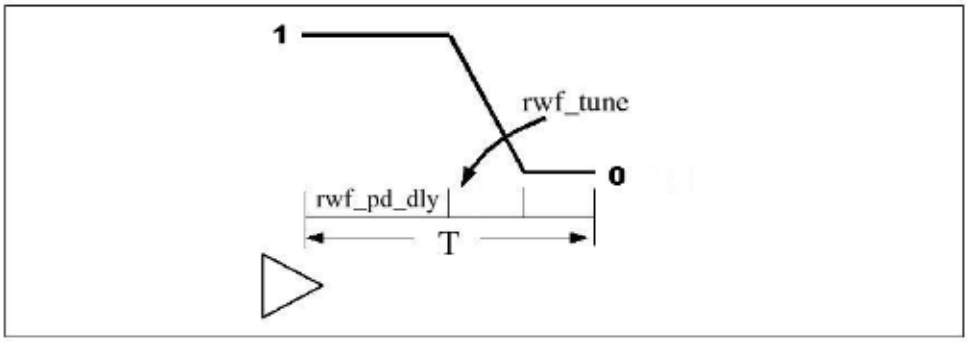


Figure 64 Pulldown Circuit Turn Off Transition and Internal Delay Calculation

pd_scal | pu_scal | pc_scal | gc_scal | rwf_scal | fwf_scal

Syntax

```
pd_scal=pd_scal_value
pu_scal=pu_scal_value
pc_scal=pc_scal_value
gc_scal=gc_scal_value
rwf_scal=rwf_scal_value
fwf_scal=fwf_scal_value
```

All IBIS and I-V curves can be scaled in HSPICE. The `rwf_scal` and `fwf_scal` parameters scale the rising and falling V-T curves. The `pd_scal`, `pu_scal`, `pc_scal`, and `gc_scal` parameters scale the pullup, pulldown, ground and power clamp I-V curves of the buffer. The default for all parameters is `1`. If you set the `rwf_scal` and `fwf_scal` parameters, you affect the rise (for `rwf_scal`) and fall (for `fwf_scal`) times and the delay of buffer. Scaling the I-V curves of the buffer effectively scales the drive strength of the buffer.

ss_state

Syntax

```
ss_state = {on|off}
```

The `ss_state` keyword turns `on` or `off` the state electrical models in the .ibis file. The default is `ss_state=on`, but only for the series switch buffer model data.

rm_dly_rwf|rm_dly_fwf|rm_tail_rwf|rm_tail_fwf

Syntax

```
rm_dly_rwf={default | rdly_time_value}
rm_dly_fwf={default | fdly_time_value}
rm_tail_rwf={default | rtail_time_value}
rm_tail_fwf={default | ftail_time_value}
```

These four keywords are used to remove non-switching time from VT waveforms in IBIS model.

- `rm_dly_rwf` and `rm_dly_fwf` remove initial delays from [Rising Waveform] and [Falling Waveform] respectively.
- `rm_tail_rwf` and `rm_tail_fwf` remove flat tails from [Rising Waveform] and [Falling Waveform] respectively.

For every keyword, if the value is set as a positive number, then that amount of time is removed from all corresponding VT waveforms. For example, if rm_dly_rwf=1ns, then 1ns initial delay is removed from each [Rising Waveform] in the related IBIS model.

For every keyword, if the value is set as 'default', then for each VT waveform, HSPICE computes the amount of no-switching time based on the criteria that the voltage change should be less than 0.2% of full voltage swing, then selects the minimum one as the final value used. For example, if rm_dly_rwf=default, then HSPICE computes the initial delay of each [Rising Waveform] (Figure 65), it then selects the minimum value of such delays as the value of the keyword.

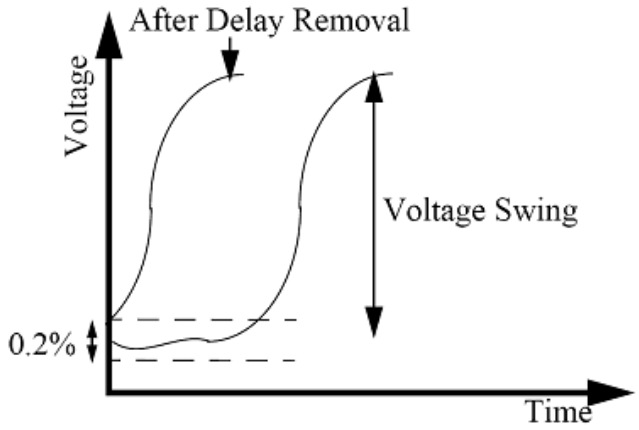


Figure 65 Removal of initial delay of [Rising Waveform] by rm_dly_rwf=default

HSPICE issues a warning if the value of any keyword is more than the default, (more than the value obtained by the 0.2% of full voltage swing criteria.

Note: For a IBIS models with a sub model like the driver schedule, the value of every keyword set in the B-element card is applied on all sub models.

nowarn

Syntax

```
nowarn
```

The `nowarn` keyword suppresses warning messages from the IBIS parser. There is no equal sign “=” and value after the `nowarn` keyword. Do not use `nowarn` as the first keyword after the nodes list. Use at least one keyword followed by “=” and a value between the list of nodes and the `nowarn` keyword.

c_com_pu | c_com_pd | c_com_pc | c_com_gc

Syntax

```
c_com_pu = c_com_pu_value
c_com_pd = c_com_pd_value
c_com_pc = c_com_pc_value
c_com_gc = c_com_gc_value
```

By default (default 1) the C_comp die capacitance connects between node_out (nd_in for input buffer) and ideal ground. To simulate power bounce and ground bounce, split C_comp into several parts. Then connect between node_out (*nd_in* for input buffer) and some (or all) of the node_pu, node_pd, node_pc, and node_gc nodes.

If you specify at least one of the optional parameters (`c_com_pu`, `c_com_pd`, `c_com_pc`, and `c_com_gc`), then the default (1) does not apply, and unspecified parameters have a value of zero (default 2). The `c_com_pu`, `c_com_pd`, `c_com_pc`, and `c_com_gc` values are dimensionless, and denote fractions of C_comp connected between node_out (nd_in for input buffer) and respective nodes (either node_pu, node_pd, node_pc, or node_gc). For example, C_comp*c_com_pu is capacitance connected between node_out and node_pu.

Do not specify negative values for `c_com_pu`, `c_com_pd`, `c_com_pc`, and `c_com_gc`.

It is expected that `c_com_pu` + `c_com_pd` + `c_com_pc` + `c_com_gc` = 1. However, HSPICE-based simulators do not enforce this requirement, and warn you only if the requirement is not satisfied.

In this case, deriving the states assumes that the IBIS files specifies C_comp for the die. The simulation uses different value of C_comp, namely:

```
C_comp*(c_com_pu+c_com_pd+c_com_pc+c_com_gc)
```

Effectively, it means that some additional capacitance connects in parallel to C_comp (possibly negative).

For the output, input-output, and 3-state buffer types, if you do not specify the node_pc and node_gc nodes in the netlist, `c_com_pc` is added to `c_com_pu` and `c_com_gc` is added to `c_com_pd`. After that, `c_com_pc` and `c_com_gc` are not used anymore.

For the open drain, open sink, input-output open drain, and input-output open sink buffer types, if you do not specify the node_pc and node_gc nodes in the netlist, `c_com_pc` if given is ignored, and `c_com_gc` is added to `c_com_pd`. After that, `c_com_gc` is not used anymore.

For the open source, and input-output open source buffer types, if you do not specify the node_pc and node_gc nodes in the netlist, `c_com_gc` if given is ignored, and `c_com_pc` is added to `c_com_pu`. After that, `c_com_pc` is not used anymore.

For the output ECL, input-output ECL, and 3-state ECLbuffer types, if you do not specify the node_pc and node_gc nodes in the netlist, `c_com_pc` and `c_com_gc` are ignored (assign zero values).

For the output ECL, input-output ECL, and 3-state ECLbuffer types, if `c_com_pd` is not zero, it is added to `c_com_pu` (`c_com_pd` is not used after that).

Note that HSPICE supports C_comp_pullup, C_comp_pulldown, C_comp_power_clamp, and C_comp_gnd_clamp in the IBIS model only when C_comp is *not* specified in the same model. In this situation, keywords c_com_* (following B element) is ignored and a warning is issued.

detect_oti_mid

Syntax

```
detect_oti_mid={1|0}
```

The default is `detect_oti_mid=1`. If `detect_oti_mid=1`, `V_out_of_in` is calculated according to the rules in the description of Input_Buffer. Otherwise, if `detect_oti_mid=0`, then the only difference with respect to `detect_oti_mid=1` is that when a buffer is behaving as a receiver, `V_out_of_in` does not take the value 0.5 if `V_in` is between `Vinl` and `Vinh`. Instead, it keeps the same value as at last time point.

This keyword is only valid for Input_Buffer or IO_Buffer.

time_control

Syntax

```
time_control=1|0
```

In earlier HSPICE versions, IBIS simulation was dependent on the HSPICE transient time step control methods. When using RUNLVL, HSPICE can relax the time stepping to improve performance but this may not be ideal for IBIS simulations in certain cases. In order to improve IBIS simulation accuracy, a customized time step control for IBIS buffers is used. The default setting for `time_control` is `1`, meaning that the IBIS time step control is enabled.

.OPTION D_IBIS

The `D_IBIS` option specifies the directory containing the IBIS files. For the syntax and examples refer to .OPTION D_IBIS in the *HSPICE Reference Manual: Commands and Control Options*.

Differential Pins

Differential pins refer to the relationship between buffers. This facilitates creating pseudo-differential buffer by two B-elements. Specify these pins in the “Component Description” section of the IBIS standard. Figure 66 and Figure 67 on page 224, and the examples that follow these figures, show how you can simulate differential pins using the Synopsys implementation of IBIS.

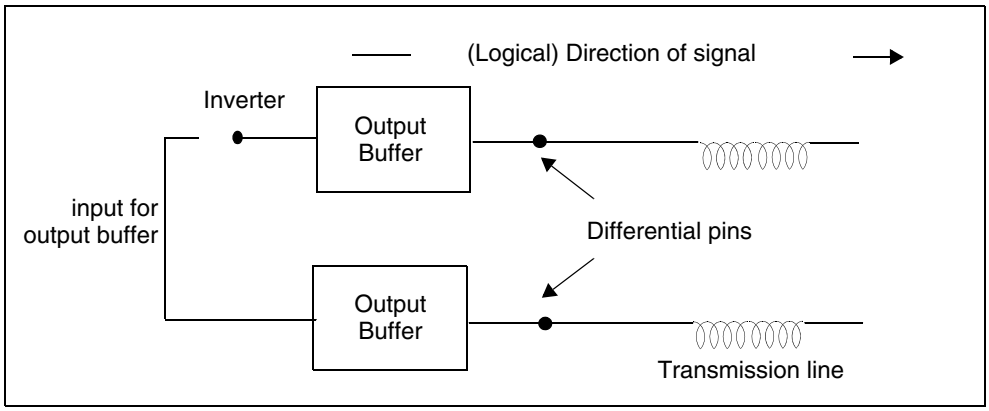


Figure 66 Output Buffers

You must use two separate buffers, each specified in a separate card in the netlist. They are related only through their input, which is differential. To implement the inverter in this situation, you must specify two independent voltage sources that have opposite polarity.

You must specify the out_of_in_1, and out_of_in_2 nodes, even if they are not used. Differential input buffers probe the voltage between nd_in_1 and nd_in_2. A voltage-dependent voltage source processes the voltage.

`V_diff` is a differential voltage parameter from the IBIS file (default is 200 mV). Add a definition of the `V_diff` parameter, the E_diff_out_of_in voltage controlled voltage source, and an R_diff_out_of_in resistor.

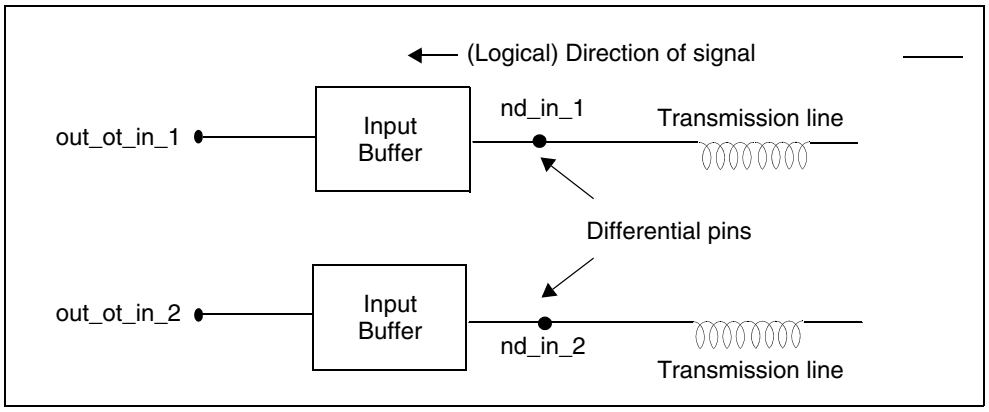


Figure 67 Input Buffers

For example,

```
.PARAM V_diff = 0.2
E_diff_out_of_in diff_out_of_in 0 PWL(1) nd_in_1 nd_in_2
+ '- V_diff' 0 '+ V_diff' 1
R_diff_out_of_in diff_out_of_in 0 1x
```

Use the voltage across `R_diff_out_of_in` as the output of the differential input buffer as shown below:

```
If V(nd_in_1) - V(nd_in_2) < V_diff, V(diff_out_of_in) = 0
if V(nd_in_1) - V(nd_in_2) > V_diff, V(diff_out_of_in) = 1
```

In IBIS4.1, a true differential buffer with SPICE-formatted external model is introduced. You can use it with the rules in sections Multilingual Model Support and SPICE or Verilog-A Formatted B-element Naming Rules.

Buffers in Subcircuits

The following examples demonstrate usage of buffers in subcircuit definition. Also, modification in the circuit description, by changing the power keyword value, is explained.

```
*********************************************
* example 1 * buffers in subcircuit, power=on
*********************************************
v_in1 nd_in1 0 pulse
+ ( 0V 1.0V CLK_Q_PRD DLT_TIME DLT_TIME CLK_H_PRD CLK_PRD )
v_en1 nd_en1 0 1V
v_in2 nd_in2 0 pulse
+ ( 1.1V 0V CLK_Q_PRD DLT_TIME DLT_TIME CLK_H_PRD CLK_PRD )
v_en2 nd_en2 0 1V
x1 nd_out1 nd_in1 nd_en1 nd_outofin1 buffer11
x2 nd_out2 nd_in2 nd_en2 nd_outofin2 buffer11

R_load nd_out1 nd_out2 50

.subckt buffer11 nd_out0 nd_in0 nd_en0 nd_outofin0
b_io_0 nd_pu0 nd_pd0 nd_out nd_in0 nd_en0 nd_outofin0 nd_pc0
+ nd_gc0
+ file = '92lv090b.ibs'
+ model = 'DS92LV090A_DOUT'
+ typ=typ power=on
+ buffer=3
+ interpol=1
xpin nd_out nd_out0 pin22
.ends

.subckt pin22 nd_out nd_out0
R_pin nd_out_c nd_out0 50m
C_pin nd_out_c 0 0.3p
L_pin nd_out nd_out_c 2n
.ends
```

In this example, buffers are connected to power sources implicitly, inside the subcircuit. Subcircuit external terminals does not need to include *nd_pu*, *nd_pd*, *nd_pc*, and *nd_gc*.

```
*********************************************
* example 2* buffers in subcircuit, power=off
*********************************************
v_in1 nd_in1 0 pulse
+ ( 0V 1.0V CLK_Q_PRD DLT_TIME DLT_TIME CLK_H_PRD CLK_PRD )
v_en1 nd_en1 0 1V
v_in2 nd_in2 0 pulse
+ ( 1.1V 0V CLK_Q_PRD DLT_TIME DLT_TIME CLK_H_PRD CLK_PRD )
v_en2 nd_en2 0 1V

x1 nd_power 0 nd_out1 nd_in1 nd_en1 nd_outofin1 nd_power 0
+ buffer11
x2 nd_power 0 nd_out2 nd_in2 nd_en2 nd_outofin2 nd_power 0
+ buffer11

R_load nd_out1 nd_out2 50

.subckt buffer11 nd_pu0 nd_pd0 nd_out0 nd_in0 nd_en0
+ nd_outofin0 nd_pc0 nd_gc0
r_0 nd_pu0 nd_pd0 1.23456789x
b_io_0 nd_pu0 nd_pd0 nd_out nd_in0 nd_en0 nd_outofin0 nd_pc0
+ nd_gc0
+ file = '92lv090b.ibs'
+ model = 'DS92LV090A_DOUT'
+ typ=typ power=off
+ buffer=3
+ interpol=1
xpin nd_out nd_out0 pin22
.ends

.subckt pin22 nd_out nd_out0
R_pin nd_out_c nd_out0 50m
C_pin nd_out_c 0  0.3p
L_pin nd_out nd_out_c 2n
.ends

V_power nd_power 0 3.3V
```

In the example above, only one voltage source (V_power) is used to power all buffers. Specify the power nodes, *nd_pu*, *nd_pd*, *nd_pc*, and *nd_gc*.

Netlist Example with Output Buffer, Transmission Line, and Input Buffer

An example of a netlist that contains an output buffer, transmission line, and input buffer is shown below. A digital signal is supplied to the nd_in node. The output buffer transmits to a network, goes through a transmission line, is received by an input buffer. It is transformed into digital form and available on the out_of_in node.

```
* IBIS Buffer Test
.option post
.tran 0.05n 70n
*
* input source
v1 in  0 pulse ( 0V 1V 1n 1n 1n 9n 20n )
r1 in  in1 50
* tristate enable
v5 enable  0 1V

* transmission line
wline1 n1 0 n2 0 RLGCmodel=pcb N=1 L=0.3

* IBIS buffers
b1 nd_pc nd_gc n2 out_of_in
+ file = 'at16245.ibs'
+ model = 'AT16245_IN'

b4 nd_pu nd_pd n1 in1 enable
+ file = 'at16245.ibs'
+ model = 'AT16245_OUT'
+ ramp_fwf=0 ramp_rwf=0

* load
rload n2 0 50

* RLCG parameters for W-element
.model pcb w modeltype=rlgc n=1
+L0=3.94266e-7
+C0=1.12727e-10
+R0=5.7739
+G0=0
+Rs=0.00141445
+Gd=0

.end
```

Using the IBIS Component Command

The `.IBIS` command creates all buffers, package and interconnection for an entire component in an IBIS file.

For the syntax of the `.IBIS` command, see .IBIS in the *HSPICE Reference Manual: Commands and Control Options*.

The following sections discuss these topics:

- How .IBIS Creates Buffers
- Required Keywords
- Optional Keywords
- Component Calls for SPICE or Verilog-A Formatted Pins
- Component Calls for SPICE or Verilog-A Formatted [External Circuit]
- Buffer Power

How .IBIS Creates Buffers

The `.IBIS` command adds a buffer to the netlist for every pin, according to the `signal_name` and `model_name` defined in the [Pin] keyword in the *ibs* file.

Note: The .IBIS command does not create a buffer if the pin name is a reserved model name, such as `POWER`, `GND`, or `NC`.

`buffer_name = 'cname'_'pin_name'`

- *cname* is defined in the .ibis card in the .sp netlist.
- *pin_name* is defined in the [Pin] keyword in the *ibs* file

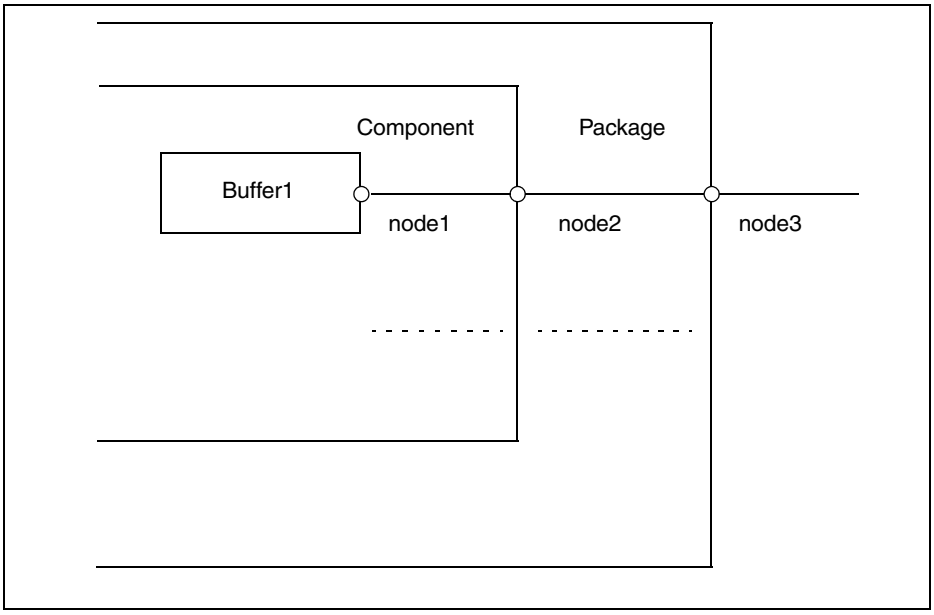


Figure 68 Package and Component Combined

HSPICE connects the new buffers that the `.IBIS` command creates to the following nodes:

- name of buffer 1: ‘cname'_'pin_name’
- name of node 1: 'cname'_'pin_name'_i
- name of node 2: 'cname'_'pin_name'_o
- name of node 3: 'cname'_'pin_name'

Note that:

- 'cname'_'pin_name'_en is the enable node if the buffer has enable node.
- 'cname'_'pin_name'_i is the outofin node if the buffer is an input buffer; for an input/output buffer, the outofin node is 'cname'_'pin_name'_outofin.

Required Keywords

file='*file_name*'

This keyword identifies the IBIS file. The *file_name* parameter must be lower case and must specify either the absolute path for the file or the path relative to the directory from which you run the simulation.

Example

```
file = '.ibis/at16245.ibs'
file = '/home/oneuser/ibis/models/abc.ibs'
```

component='*component_name*'

This keyword identifies the component for an .IBIS command from the IBIS file, specified using the `file='...'` keyword. The *component_name* keyword is case-sensitive, and it must match one of the components from the IBIS file.

Example

```
component = 'procfast'
component = 'Virtex_SSTL_3-I_BG432'
```

Optional Keywords

package

This keyword specifies the type of package to add.

```
package = [0|1|2|3]
```

When package equals

- `0`, then the RLC package is not added into the component.
- `1`, then [Package] (in the *ibs* file) is added.
- `2`, then [Pin] (in the *ibs* file) is added.
- `3` (default), and if [Package Model] is defined, set package with a package model. If the [Package Model] is not defined, set the package with [Pin]. If the package information is not set in [Pin], set the package with [Package] as a default. You can define the [Package Model] in IBIS or PKG files.

Example

```
.ibis p_test
+ file = 'comp.ibs'
+ component = 'cpu_133mhz_ff'
+ hsp_ver = 2002.4 nowarn
+ package = 3
+ pkgfile = 'test.pkg'
```

This card combines the `.pkg` and `.ibis` files. If you use [Package Model] in the `cpu_133mhz_ff` component, HSPICE searches for the pkg model in the `comp.ibs` and `test.pkg` files.

pkgfile='pkg_file_name'

You can define the package model file using the `pkgfile` keyword.

`pkgfile = '`*`pkg_file_name`*`'`

If you cannot find a package model defined within a component in both the PKG and IBIS files, HSPICE issues an error message.

[Model Selector] Support

You can select a model using the `mod_sel` keyword.

```
mod_sel = 'selName_1=modName_1, selName_2=modName
   _2,...,selName_n=modName_n'
```

If `selName` and `modName` cannot be found in a related IBIS file or they cannot be matched, then an error occurs. In addition, the input string should be less than 1024 characters. If a model selector is used for a pin of a component, but `mod_sel` is not set, then the first model under the corresponding [Model Selector] is selected as default.

Other Optional Keywords

The following keywords are the same as for the B-element (I/O buffer). For more information, see Specifying Required and Optional Common Keywords on page 207

- `typ`
- `interpo`
- `ramp_rwf`
- `ramp_fwf`

- rwf_tune
- fwf_tune
- pd_scal
- pu_scal
- pc_scal
- gc_scal
- rwf_scal
- fwf_scal
- nowarn
- hsp_ver
- c_com_pd
- c_com_pu
- c_com_pc
- c_com_gc
- detect_oti_mid

If any keyword above is used, then it will be applied to all buffers created by the .IBIS command.

Component Calls for SPICE or Verilog-A Formatted Pins

The syntax to call a [Component] having SPICE or Verilog-A formatted [Pin] is:

```
.IBIS ibis_command_name
+ file='ibis_filename' component='component_name'
+ [package=3|0|1|2] [pkgfile='pkg_file_name']
+ [nowarn]
+ [mod_sel = 'selName_1=modName_1,
+ selName_2=modName_2,...,selName_n=modName_n']
```

In the preceding syntax,

- The *component_name* specification must involve one or more SPICE formatted [Pin].
- Usage of the other parameters follow .ibis conventions.

Other parameters not shown above have no affect on a SPICE formatted buffer.

If the buffer is a series buffer or a SPICE or Verilog-A formatted true differential buffer in which two pins are involved, then:

```
buffer_name='cname'_'pin1_name1'_'pin2_name'
```

For related node names, refer to SPICE or Verilog-A Formatted B-element Naming Rules.

Component Calls for SPICE or Verilog-A Formatted [External Circuit]

[External Circuit] in a component can be divided into two types. One is used to describe the buffer and is similar to [External Model]; the other is used to describe interconnection inside a component. For the former, [Pin Mapping] can not be used in order to avoid ambiguous connections. For the latter, a die node declared in Port_map of [Circuit Call] can be probed in a netlist by the naming rule:

'cname'_'die_node_name'

Here, cname is defined in the .ibis card in the *.sp* netlist. In addition, die pad must not occur in Port_map since related connection syntax is still not available in the latest IBIS spec. If a buffer is described by [External Circuit], then it is set as power off, so voltage sources are not be created automatically to connect to external terminals of Pullup, Pulldown, Powerclamp and Groundclamp.

Buffer Power

The buffer component creates buffers that are always connected to the power sources that are specified in the IBIS file. You can specify power sources two different ways, as described in the following sections:

- Buffer Power ON
- Buffer Power OFF

Buffer Power ON

If [Pin Mapping] is not defined in the *ibs* file or hsp_ver <= 2002.2, HSPICE automatically sets the power of buffers to ON. Use the [Voltage Range], [Pullup Reference], [Pulldown Reference], [POWER Clamp Reference], and [GND

Clamp Reference] keywords in the *ibs* file to set the voltage source inside the buffer.

The buffer nodes expect input and output node names in this format:

```
'buffer_name'_<node_name>
 buffer_name = 'cname'_'pin_name'
```

- `cname` is defined in the .ibis card in the .sp netlist.
- `pin_name` is defined in the [Pin] keyword in the *ibs* file
- `<node_name>` is different for different types of buffers as shown in the following list:

INPUT	pc , gc
OUTPUT	pu , pd , pc , gc
INPUT_OUTPUT	pu , pd , en , outofin , pc , gc
THREE_STATE	pu , pd , en , pc , gc
OPEN_DRAIN	pu , pd , pc , gc
IO_OPEN_DRAIN	pu , pd , en , outofin , pc , gc
OPEN_SINK	pu , pd , pc , gc
IO_OPEN_SINK	pu , pd , en , outofin , pc , gc
OPEN_SOURCE	pu , pd , pc , gc
IO_OPEN_SOURCE	pu , pd , en , outofin , pc , gc
INPUT_ECL	pc , gc
OUTPUT_ECL	pu , pc , gc
IO_ECL	pu , en , outofin , pc , gc
THREE_STATE_ECL	pu , en , pc , gc

Note: For more information about nodes for different buffers, see Buffer Types on page 182

Table 24 shows the names of the input and output nodes for the buffers:

Table 24 Input and Output Node Names for Buffers

Buffer Type	Node Names
INPUT and INPUT_ECL buffers	'cname'_'pin_name' (for in node)
	'cname'_'pin_name'_i (for outofin node)
Other types of buffers	‘cname’_’pin_name’_i (for in node)
	‘cname’_’pin_name’ (for out node)

If the buffer has an enable terminal, you must create a node named buffer_name_en to enable the buffer.

You can test the following sample cases using the same method explained.

In this example, you can test the buffers with the B -element.

```
*********
.tran 50p 30n
.option post probe
.subckt twobus_b out1 out2
boutput1 nd_pu1 nd_pd1 out1 nd_in1 nd_pc1 nd_gc1
+ file = 'pinmap.ibs'
+ model = 'out50v'
+ buffer=2 power_on
+ nowarn
boutput2 nd_pu2 nd_pd2 out2 nd_in2 nd_pc2 nd_gc2
+ file = 'pinmap.ibs'
+ model = 'out50v'
+ buffer=2 power=on
+ nowarn
Vin1_b nd_in1 0 0V pulse ( 0V 1V 1n 0.1n 0.1n 7.5n 15n )
Vin2_b nd_in2 0 0V pulse ( 0V 1V 1n 0.1n 0.1n 7.5n 15n )
.ends

Xtwobus_b out1_b out2_b twobus_b

Rout1_b out1_b 0 50
Rout2_b out2_b 0 50

.probe tran
+ out1_of_output_b = v(out1_b)
+ out2_of_output_b = v(out2_b)
+ in_of_output_b = v(Xtwobus_b.nd_in1)
.end
```

In this example, you can test the buffers that the .IBIS command creates.

```
*********
.tran 50p 30n
.option post probe
* add component
.ibis pcomp
+ file = 'pinmap.ibs'
+ component = 'NO_PINMAPPING'
+ nowarn
+ package = 0
Vin1 pcomp_1_i 0 0V pulse ( 0V 1V 1n 0.1n 0.1n 7.5n 15n )
Vin2 pcomp_2_i 0 0V pulse ( 0V 1V 1n 0.1n 0.1n 7.5n 15n )
Rout1_1 pcomp_1 0 50
Rout2_1 pcomp_2 0 50

.probe tran
+ out1_of_nopm = v(pcomp_1)
+ out2_of_nopm = v(pcomp_2)
+ in_of_output = v(pcomp_1_i)
.end
```

The following *ibs* file is:

```
|*******************************************************
| IBIS file pinmap.ibs
|*******************************************************|
[IBIS ver] 3.1
[File name] pinmap.ibs
[File Rev] 1.0
[Date] 5/24/2002
[Source] spice models
[Notes] This ibis file tests the pinmap
results of HSPICE .ibis command. Pinmapping
is considered only if hsp_ver > 2002.2.

|*******************************************************
| Component NO PINMAPPING
|*******************************************************
|
[Component] NO_PINMAPPING
|
[Manufacturer] TEST
|

        [Package]    typ    min   max

R_pkg   0            NA     NA

L_pkg   0            NA     NA
```

```
C_pkg   0            NA     NA

   [Pin]   signal_name   model_nam   R_pin      L_pin      C_pin
                         e
   OUT1    out50v        50.0m       2.01554n   0.33960p
   OUT2    out50v        50.0m       1.83759n   0.31097p
   GND1    GND           50.0m       1.89274n   0.32012p
   VCC1    POWER         50.0m       1.74394n   0.31941p
   GND2    GND           50.0m       1.89274n   0.32012p
   VCC2    POWER         50.0m       1.74394n   0.31941p

.....................
|model information
.....................
[END]
```

Buffer Power OFF

If [Pin Mapping] is defined both in the *ibs* file and hsp_ver > 2002.2, HSPICE automatically uses the [Pin Mapping] keyword in the *ibs* file to load voltage sources to the buffers. HSPICE turns OFF the power of buffers that the component created.

In this case, the nd_pc, nd_pu, nd_gc, and nd_pd nodes connect to the power and ground bus according to the information in [Pin Mapping]. All other nodes are the same as in Buffer Power ON on page 233

The following two .sp files are equivalent. The first file uses the B-element to describe buffers:

```
* test for IO,OUTPUT,3-STATE,INPUT BUFFER in component w/pin
mapping
.option post probe
.tran 50p 30n
.param vhi = 2.5

.subckt io_o_3st_i_b in st gnd1 vcc1 io output
bin vcc1 gnd1 in nd_out1
+ file = 'pinmap.ibs'
+ model = 'DS92LV090A_IN1'
+ buffer=1 power=off
+ nowarn
b3st vcc1 gnd1 st nd_in1 nd_en1 vcc1 gnd1
+ file = 'pinmap.ibs'
+ model = 'DS92LV090A_RO'
+ buffer=4 power=off
+ nowarn
bio vcc1 gnd1 io nd_in2 nd_en2 nd_ofi vcc1 gnd1
+ file = 'pinmap.ibs'
+ model = 'DS92LV090A_DOUT'
+ buffer=3 power=off
+ nowarn
boutput vcc1 gnd1 output nd_in3 vcc1 gnd1
+ file = 'pinmap.ibs'
+ model = 'out50v'
+ buffer=2 power=off
+ nowarn
Vin1_b nd_in1 0 0V pulse ( 0V 1V 1n 0.1n 0.1n 7.5n 15n )
Ven1_b nd_en1 0 0V pulse ( 0V 1V 1n 0.1n 0.1n 7.5n 15n )
Vin2_b nd_in2 0 0V pulse ( 0V 1V 1n 0.1n 0.1n 7.5n 15n )
Ven2_b nd_en2 0 0V pulse ( 0V 1V 1n 0.1n 0.1n 7.5n 15n )
Vin3_b nd_in3 0 0V pulse ( 0V 1V 1n 0.1n 0.1n 7.5n 15n )
Rout_b nd_out1 0 50
.ends

Xio_o_3st_i_b in_b st_b gnd1_b vcc1_b io_b output_b io_o_3st_i_b
Vvcc1_b vcc1_b 0 3.3v
Vgnd1_b gnd1_b 0 0v
Vin_b in_b 0 0V PULSE (0 vhi 1ns 0.5ns 0.5ns 5ns 10ns)
Rout1_b st_b 0 50
Rout2_b io_b 0 50
Rout3_b output_b 0 50
.probe tran
+ out_of_in_b = v(Xio_o_3st_i_b.nd_out1)
+ out_of_3st_b = v(st_b)
+ out_of_io_b = v(io_b)
+ out_of_output_b = v(output_b)
+ in_of_output_b = v(Xio_o_3st_i_b.nd_in3)
```

```
.end
```

The second file uses a the .IBIS command to create buffers:

```
* test with component
.option post post_version=9601 probe
.tran 50p 30n
.param vhi = 2.5
.ibis pcomp
+ file = 'pinmap.ibs'
+ component = 'IO_OUTPUT_3ST_INPUT'
+ nowarn
+ package = 0

Vin1 pcomp_3_i 0 0V pulse ( 0V 1V 1n 0.1n 0.1n 7.5n 15n )
Ven1 pcomp_3_en 0 0V pulse ( 0V 1V 1n 0.1n 0.1n 7.5n 15n )
Vin2 pcomp_6_i 0 0V pulse ( 0V 1V 1n 0.1n 0.1n 7.5n 15n )
Ven2 pcomp_6_en 0 0V pulse ( 0V 1V 1n 0.1n 0.1n 7.5n 15n )
Vin3 pcomp_7_i 0 0V pulse ( 0V 1V 1n 0.1n 0.1n 7.5n 15n )
Rout pcomp_2_i 0 50

Vvcc1 pcomp_5 0 3.3v
Vgnd1 pcomp_4 0 0v
Vin pcomp_2 0 0V PULSE (0 vhi 1ns 0.5ns 0.5ns 5ns 10ns)
Rout1 pcomp_3 0 50
Rout2 pcomp_6 0 50
Rout3 pcomp_7 0 50

.probe tran
+ out_of_in = v(pcomp_in_2)
+ out_of_3st = v(pcomp_3)
+ out_of_io = v(pcomp_6)
+ out_of_output = v(pcomp_7)
+ in_of_output = v(pcomp_7_i)
.end
```

The *.ibs* file is:

```
|******************************************************
| IBIS file pinmap.ibs
|******************************************************|
[IBIS ver] 3.1
[File name] pinmap.ibs
[File Rev] 1.0
[Date] 5/24/2002
[Source] spice models
[Notes] This ibis file tests the pinmap
results of HSPICE .ibis command. Pinmapping
is considered only if hsp_ver > 2002.2.
|******************************************************
| Component IO,OUTPUT,3-STATE,INPUT BUFFER
|******************************************************


[Component]        IO_OUTPUT_3ST_INPUT[Manufacturer]

TEST


        [Package]    typ   min   max

R_pkg   0            NA    NA

L_pkg   0            NA    NA

C_pkg   0            NA    NA

|
|******************************************************
|

  [Pin]   signal_name      model_name  R_pin     L_pin     C_pi
                                                         n
  NC      NC               50.0m       2.11433n  0.27505p
  IN      DS92LV090A_IN1   50.0m       2.01554n  0.33960p
  ST      DS92LV090A_RO    50.0m       1.94357n  0.32802p
  GND1    GND              50.0m       1.89274n  0.32012p
  VCC1    POWER            50.0m       1.74394n  0.31941p
  IO      DS92LV090A_DOUT  50.0m       1.83759n  0.31097p
  OUTPUT  out50v           50.0m       1.83759n  0.31097p
```

```
|**************************************************************
   |

  [Pin       pulldown_r   pullup_r   gnd_clamp_r   power_clamp_r
  Mapping]   ef           ef         ef            ef

  NC         NC           NC         NC

  NC         NC           GND1       VCC1

  GND1       VCC1         GND1       NC

  NC         VCC1         NC         NC

  GND1       VCC1         GND1       VCC1

  GND1       VCC1         GND1       VCC1


.....................
|model information
.....................
[END]
```

Using IBIS Package Modeling

In addition to `.IBIS`, HSPICE provides the `.PKG` command to parse package model definition in either *.ibs* or *.pkg* files.

The `.PKG` command automatically creates a series of W-elements or discrete R, L and C components. (For the complete description of this command, see the *HSPICE Reference Manual: Commands and Control Options* .PKG.)

Syntax

```
.PKG pkgname
+ pkgfile='pkgfilename'
+ model='pkgmodelname'
```

Examples

Example 1

```
.pkg p_test
+ pkgfile='processor_clk_ff.ibs'
+ model='FCPGA_FF_PKG'
```

Example 2—This example shows how in the second line, pin1 is referenced as the element name changes:

```
p_test_pin1_dia and p_test_pin1
w_p_test_pin1_? ? or r_p_test_pin1_? ? ...
```

Accessing Nets inside a Package Model

You can access the nets (internal pins) of the package model defined in the IBIS file. The node name is `compName_pinName`. Where `compName` is the component name defined in the `.IBIS` command and `pinName` is the Pin name given by the [Pin] definition in the package model. For example

```
.ibis buff
+ file = 'buff1.ibs'
+ component = 'Comp12'
```

The IBIS file contains the following package information for the pins that you want to access.

```
[Package Model] pkg_model
[Pin]  signal_name       model_name        R_pin    L_pin    C_pin
vdd    vdd                   POWER
gnd    gnd                   GND
out    out                   driver
out2   out2                  driver
```

Using `.PROBE` statements, you can probe the nets as follows:

```
.probe v(buff_vdd) $ pin vdd defined in the package model
+ v(buff_gnd)      $ pin gnd defined in the package model
+ v(buff_out)      $ pin out defined in the package model
+ v(buff_out2)     $ pin out2 defined in the package model
```

Note: If [pin mapping] is not defined in the IBIS file, then there is no connection to the pins of *vdd* and *gnd* and the nodes of *buff_vdd* and *buff_gnd* are not created. HSPICE creates voltage sources automatically for pins of *out* and *out2*.

If [pin mapping] is defined in the IBIS file, then HSPICE connects PU, PD, PC, and GC of *out* and *out2* to pins of vdd and gnd (in fact, to nodes of *buff_vdd_o* and *buff_gnd_o*, due to package) according to [pin mapping].

Using IBIS Board-Level Components

A board-level component is used to describe a printed circuit board (PCB) or substrate that can contain components or even other boards and can connect to another board through a set of user visible pins. The electrical connectivity of such a board level component is referred to as an “Electrical Board Description” (EBD). The `.EBD` command provides connectivity descriptions of chip to chip, or multiple-chips on a single package.

The `.EBD` command is associated with `.IBIS` command, because the EBD file contains the reference designator of the component. HSPICE search paths between the component pins and the EBD pins are displayed in the [Path Description] of the EBD file. The component pins are designated with `keyword:component` in the `.EBD` command, which automatically create Lumped RLCs and distributed RLCs (W-elements) to express those paths. HSPICE then simulates the newly-created circuit.

In the following netlist, you can see the important elements and internal nodes that are created in the naming rules of the `.EBD` command.

```
*************************************************************
*    Analysis And Options
*************************************************************

.op
.tran 10p 30n

.option probe
*************************************************************
*    Stimulus
*************************************************************

Vin cmp1_5_i 0 0V pulse ( 0V 3V 2n 0.1n 0.1n 7.5n 15n )

Vpu cmp1_1 0 2.5v
vpc cmp1_2 0 2.5v
vpd cmp1_3 0 0v
vgc cmp1_4 0 0

*************************************************************
*    Rload
*************************************************************

Rd1 ebd1_a3 0 50
Rd2 ebd1_a3 0 50

*************************************************************
*    Define Component
*************************************************************
.ibis cmp1
+ file = 'pinmap.ibs'
+ component = 'Component1'
+ package = 0

*************************************************************
*   Define EBD
*************************************************************

.ebd ebd1
+ file = 'pinmap.ebd'
+ model = 'Board1'
+ component = 'cmp1:u21'

*************************************************************
*  Output
*************************************************************

.probe tran
```

```
+ cmp1_5_out = v(cmp1_5)       $ buf_5(Output): cmp1_5
+ cmp1_5_in  = v(cmp1_5_i)    $ no package , so the out node is
cmp1_5,
                               $ not cmp1_5_o

+ ebd1_a3    = v(ebd1_a3)
+ ebd1_a4    = v(ebd1_a4)

.end
```

The following sections discuss these topics:

- .EDB and .IBIS Command Syntax
- Circuit Topology Created by the .EBD and .IBIS Commands
- B-element Naming Rules
- Circuit Topology Created with SPICE or Verilog-A Formatted Pins
- IBIS Board-Level Component Examples

.EDB and .IBIS Command Syntax

Use the following syntax when using IBIS board-level components:

```
.EBD 'ebd_name'
+ file = 'ebd_file_name' $ case-sensitive
+ model = 'ebd_model_name' $ case-sensitive
+ component = 'ibis_name:ref_des' $ case-insensitive
+ [component = 'ibis_name:ref_des'[...]]

.IBIS 'ibis_name'
+ file = 'ibis_file_name' $ case-sensitive
+ component = 'component_name' $ case-sensitive
+ [package = 0|1|2|3]
+ [other_keyword = value [...]]
```

Note: Parameters surrounded by arrow brackets (< >) are not required.

Table 25 .EBD/.IBIS Required Argument Descriptions

Parameter	Description
ebd_name	Specifies the name of the .EBD command. This argument is case-insensitive.

Table 25 .EBD/.IBIS Required Argument Descriptions (Continued)

Parameter	Description
ibis_name	Specifies the name of associated .IBIS command. This argument is case-insensitive.
ebd_file_name	Identifies the EBD file. You must specify either the absolute path for the file or the path relative to the directory from where you run the simulation. This argument is case-sensitive. For example:`file = './ebd/test.ebd'` `file = '/home/usr/ebd/test.ebd'`
ebd_model_name	Identifies the Board-Level Component. Visible in the [Begin Board Description] keyword in the EBD file. This argument is case-sensitive.
ibis_name:ref_des	Maps an IBIS buffer component to a reference designator. ▪ ibis_name: Name of the associated .IBIS command to identify IBIS buffer component. ▪ ref_des: Reference designator. Visible in the Node sub-parameter of the [Path Description] keyword in the EBD file. For example:`component = 'cmp2:u22'`

Circuit Topology Created by the .EBD and .IBIS Commands

The circuit topology in Figure 69 is created by the `.EBD` and `.IBIS` commands:

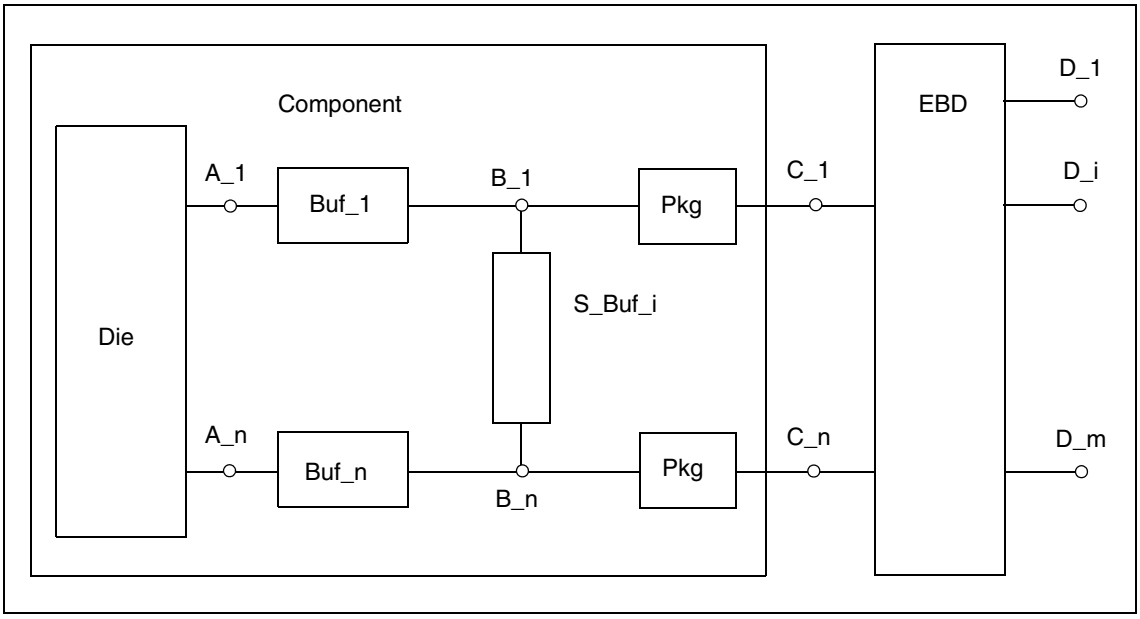


Figure 69 Circuit Topology Created by the .EBD and .IBIS Commands

B-element Naming Rules

The following rules in Table 26 apply when naming B-elements in IBIS board-level components:

Table 26 B -element Naming Rules

Buffer	Name	Note
Buf_#	‘ibis_name’_’ibis_pin_name’	If model_name shows NC/POWER/GND in the column of the [Pin] keyword in the IBIS file, no Buf_# buffer exists.
S_Buf_#	‘ibis_name’_’pin1_name’_’pin2_name’	If the [Series Pin Mapping] keyword is not in the IBIS file, no S_Buf_# series/series switch buffer exists.

Table 27 Node Naming Rules

Node	Name	Corresponding Buffer Node
A_#	‘ibis_name’_’ibis_pin_name’_i	Terminator buffer: No A_# nodes Input/Input_ecl buffer: nd_out_of_in Other buffers: nd_in If no Buf_#, no A_# nodes
B_#	‘ibis_name’_’ibis_pin_name’_o	Input/input_ecl buffer: nd_in Other buffers: nd_out Note: If no package (i.e. the package keyword in the .IBIS command is set to 0 or no relative sub- parameter is given in [Pin]/[Package] in IBIS file), then no B_# nodes. The corresponding nodes are named as C_# nodes.
C_#	‘ibis_name’_’ibis_pin_name’	
D_#	‘ebd_name’_’ebd_pin_name’	
Enable	‘ibis_name’_’ibis_pin_name’_en	Buffer that has the nd_en Enable node. Other buffers: No

Table 27 Node Naming Rules

Node	Name	Corresponding Buffer Node
Out_of_in	'ibis_name'_'ibis_pin_name'_outofin	Buffer that has out_of_in Node except. Input/Input_ECL Buffer
Pulldown of Buf_#	'ibis_name'_'ibis_pin_name'_pd or 'ibis_name'_'ibis_pin_pd_name_o' (if pin mapping is defined in the IBIS file)	Buffer that has the Pulldown nd_pd node. Other buffers: No
Pullup of Buf_#	'ibis_name'_'ibis_pin_name'_pu or 'ibis_name'_'ibis_pin_pu_name_o' (if pin mapping is defined in the IBIS file)	Buffer that has the Pullup nd_pu node. Other buffers: No
GND Clamp of Buf_#	'ibis_name'_'ibis_pin_name'_pd or 'ibis_name'_'ibis_pin_gc_name_o' (if pin mapping is defined in the IBIS file)	Buffer that has the GND Clamp nd_gc node. Other buffers: No
POWER Clamp of Buf_#	'ibis_name'_'ibis_pin_name'_pc or 'ibis_name'_'ibis_pin_pc_name_o' (if pin mapping is defined in the IBIS file)	Buffer that has POWER Clamp nd_pc node. Other buffers: No

In the following node names, the first matched pin name in [Pin] consists of ground/power connections given in [Pin Mapping] in the **.ibs* file.

- 'ibis_name': Name of `.IBIS` command.
- 'ebd_name': Name of `.EBD` command.
- 'ibis_pin_name': Pin Name, the first column in Keyword: [Pin] in IBIS file.
- 'ebd_pin_name': Pin Name, the pin name given in [Path Description] in EBD file.
- 'pin1_name': One of two pins joined by a series model, the first column in Keyword: [Series Pin Mapping] in IBIS file.
- 'Pin2_name': The other of two pins joined by a series model, the second column in Keyword: [Series Pin Mapping] in IBIS file.
- 'ibis_pin_pd_name'
- 'ibis_pin_pu_name'
- 'ibis_pin_gc_name'
- 'ibis_pin_pc_name': The first matched pin name in [Pin] which are ground/power connections given in [Pin Mapping] in IBIS file.

Circuit Topology Created with SPICE or Verilog-A Formatted Pins

The circuit topology in Figure 70 is created by the .EBD and .IBIS commands where the IBIS component model contains SPICE or Verilog-A formatted pins:

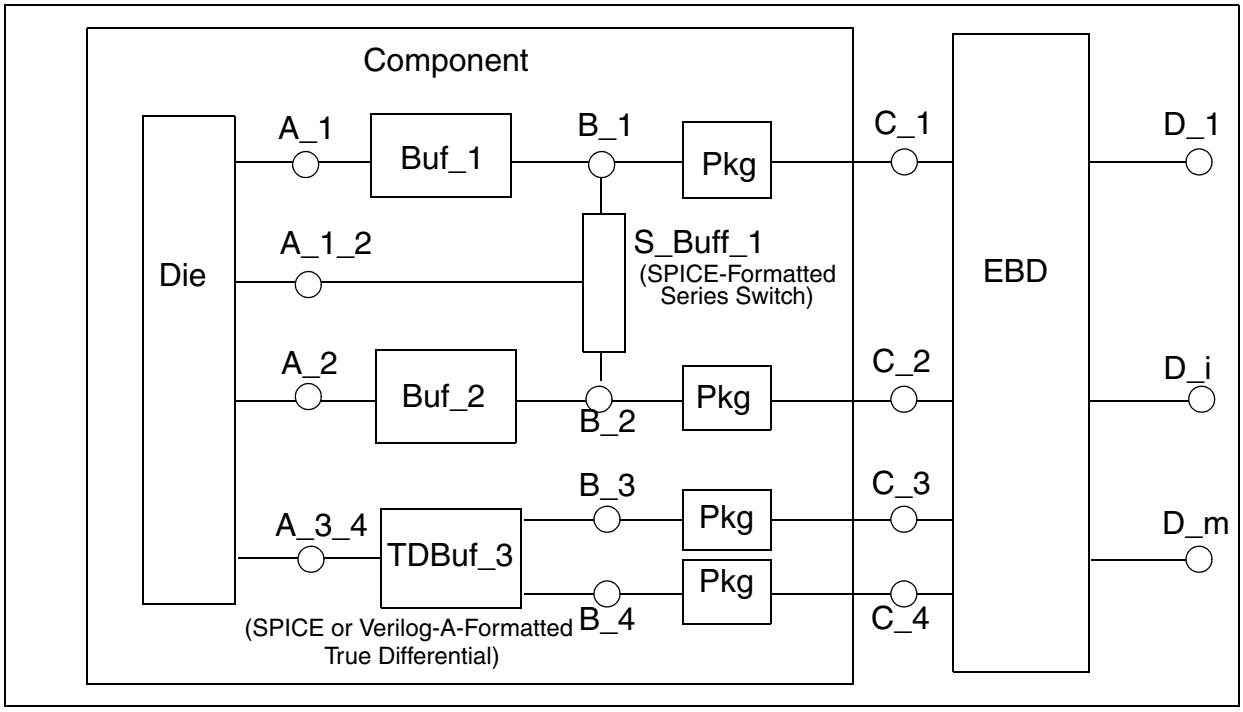


Figure 70 Circuit topology with SPICE or Verilog-A Formatted pins

SPICE or Verilog-A Formatted B-element Naming Rules

The following rules (Table 28) apply when naming SPICE or Verilog-A formatted B-elements in IBIS board level components

Table 28 SPICE or Verilog-A Formatted B-element Naming Rules

Buffer	Name	Comment
Buf_#	‘ibis_name’_’ibis_pin_name’	If model_name shows NC/POWER/GND in the column of the [Pin] keyword in the IBIS file, no Buf_# buffer exists.
TDBuf_#	'ibis_name'_'pin1_name'_'pin2_name’	If [Diff Pin] keyword is not in the IBIS file, or it is not a true differential buffer, no TDBuf_# buffer exists.
S_Buf_#	'ibis_name'_'pin1_name'_'pin2_name'	If the [Series Pin Mapping] keyword is not in the IBIS file, no S_Buf_# series/series switch buffer exists.

The following rules (Table 29) apply when naming SPICE or Verilog-A formatted nodes in IBIS board level components:

Table 29 Node naming rules—SPICE/Verilog-A formatted buffers in component

Node	Name	Corresponding buffer node/port
A_#	‘ibis_name’_’ibis_pin_name’_i	Terminator buffer: No A_# nodes Input/Input_ecl buffer: nd_out_of_inOther buffers: nd_in
B_#	ibis_name'_'ibis_pin_name'_o	Input/input_ecl buffer: nd_in Other buffers: nd_out Note: If no package (i.e. the package keyword in the .IBIS command is set to 0 or no relative sub-parameter is given in [Pin] / [Package] in IBIS file), then no B_# nodes. The corresponding nodes are named as C_# nodes.
C_#	'ibis_name'_'ibis_pin_name'	
D_#	'ebd_name'_'ebd_pin_name'	
A_#_#	'ibis_name'_'pin1_name'_'pin2_name'_i	Input_diff buffer: nd_out_of_in Other buffers: nd_in
enable	'ibis_name'_'ibis_pin_name'_en or 'ibis_name'_'pin1_name'_'pin2_name'_en (if buffer is true differential buffer)	Buffer that has an enable node (nd_en)
Out_of_in	'ibis_name'_'ibis_pin_name'_outofin or 'ibis_name'_'pin1_name'_'pin2_name'_outofin (if buffer is true differential buffer)	Buffer that has out_of_in node except Input, Input_ECL and, Input_diff buffers

Table 29 Node naming rules—SPICE/Verilog-A formatted buffers in component

Node	Name	Corresponding buffer node/port
Pulldown of Buf_#	'ibis_name'_'ibis_pin_name'_pd or 'Ibis_name'_'ibis_pin_pd_name'_o' (if pin mapping is defined in the IBIS file)	A_pdref
Pullup of Buf_#	'ibis_name'_'ibis_pin_name'_pu or 'ibis_name'_'ibis_pin_pu_name'_o (if pin mapping is defined in the IBIS file)	A_puref
GND Clamp of Buf_#	'ibis_name'_'ibis_pin_name'_gc or 'ibis_name'_'ibis_pin_gc_name'_o (if pin mapping is defined in the IBIS file)	A_gcref
POWER Clamp of Buf_#	clamp'ibis_name'_'ibis_pin_name'_pc or 'ibis_name'_'ibis_pin_pc_name'_o (if pin mapping is defined in the IBIS file)	A_pcref
External Reference of Buf_#	reference 'ibis_name'_'ibis_pin_name'_ext or 'ibis_name'_'ibis_pin_ext_name'_o (if pin mapping is defined in the IBIS file)	A_extref
Global reference	0	A_gnd
Non-inverting node of S_Buf_#	'ibis_name'_'pin1_name'_o	A_pos
Inverting node of S_Buf_#	'ibis_name'_'pin2_name'_o	A_neg
Non-inverting node of TDBuf_#	'ibis_name'_'pin1_name'_o	A_signal_pos

Table 29 Node naming rules—SPICE/Verilog-A formatted buffers in component

Node	Name	Corresponding buffer node/port
Inverting node of TDBuf_#	'ibis_name'_'pin2_name'_o	A_signal_neg
other nodes of Buf_#	'ibis_name'_'ibis_pin_name'_'port_name'	other ports defined by user
other nodes of TDBuf_#	'ibis_name'_'pin1_name'_'pin2_name'_ 'port_name'	other ports defined by user

The following names are the same as in Table 27 on page 249 except that for TDBuf_#, 'pin1_name','pin2_name' are two pins joined by a true differential buffer model. They are the first and second columns in the keyword: [Diff Pin] in the IBIS file:

- 'ibis_name'
- 'ibis_pin_name'
- 'pin1_name'
- 'pin2_name'
- 'ibis_pin_pd_name'
- 'ibis_pin_pu_name'
- 'ibis_pin_gc_name'
- 'ibis_pin_pc_name'

IBIS Board-Level Component Examples

The following examples show how you can use the `.IBIS` and `.EBD` commands for board-level components.

Example 1: .IBIS Command File

The complete netlist, IBIS model, package and *.ebd* files for this example can be found in $installdir/demo/hspice/ibis/ebd.

The IBIS model for this example is the file *4mx32.ibs*. To use the .IBIS command, it is necessary to find the [Component] keyword in the *.ibs* file as in the following part of the *.ibs* file:

```
|
[Component]       K4D263238A
|
[Manufacturer]    TEST
[Package]
| variable        typ                 min                 max
R_pkg             0.17                77.00m              0.77
L_pkg             2.84nH              0.56nH              6.04nH
C_pkg             3.20pF              0.55pF              5.20pF
|
[Pin]  signal_name      model_name       R_pin    L_pin    C_pin
B3     DM0              CTRL_PIN          0.71     5.75nH   5.20pF
B5     DQ3              DQ_PIN            0.12     0.86nH   0.57pF
B6     DQ2              DQ_PIN           83.00m    0.64nH   0.92pF
B7     DQ0              DQ_PIN           88.00m    0.56nH   0.53pF
C6     DQ1              DQ_PIN            0.11     0.78nH   0.56pF
M11    CK               CLK_PIN           0.36     1.64nH   2.52pF
```

In the example above, the component name is K4D263238A. In addition to the [Component] keyword, this portion of the *.ibs* file also shows the pin list. This information becomes important later.

Note: There are three models relating to this component: CTRL_PIN, DQ_PIN, and CLK_PIN. If ramp_rwf and ramp_fwf keywords are set, then all models take the keyword values. Of course, such keywords may be nonsense for some model types.

This component only uses [Ramp] to define the V-t characteristics of the buffers so; the ramp_rwf and ramp_fwf keywords can be set to 0 for this example. There is package information for this component that is contained in the package file *4mx32.pkg*. The package information can be specified in the .IBIS command. The .IBIS command requires an instance name of the component to be specified. For this case, the name bc1 is used. The ibis component definition can then be written as:

```
* ibis component
.ibis bc1 component='K4D263238A'
+ file='4mx32.ibs'
+ ramp_rwf=0 ramp_fwf=0
+ package=3
+ pkgfile='4mx32.pkg'
```

Note: The component keyword is case sensitive and the specified component name must match the case of [Component] in the *.ibs* file or HSPICE prints an error message that the component is not defined.

Example 1: .EBD Command File

To use the .EBD command, it is necessary to find the name of the board as given by the [Begin Board Description] keyword in the *.ebd* file. The name of the board is used for the model name in the .EBD command. In this example, the name of the board description is Board1. The reference designators of the components on the board are given by the [Reference Designator Map] keyword and are used by the component keyword of the .EBD command. In this example, there is only one reference designator, U1. Part of the *.ebd* file:

```
[Begin Board Description]  Board1
[Manufacturer]             Test
|
[Number Of Pins] 2
|
[Pin List]  signal_name
 B7          DQ0
 C6          DQ1
[Path Description] Trace1
Pin B7
Len = 0.5 L=8.35n C=3.34p R=0.01 /
Len = 0.1 L=2.1n  C=1.15p R=0.01 /
Node u1.B7
[Path Description] Trace2
Pin C6
Len = 0.5 L=8.35n C=3.34p R=0.01 /
Len = 0.1 L=2.1n C=1.15p R=0.01 /
Node u1.C6
[Reference Designator Map]
| Ref Des  File name       Component name
u1        4mx32.ibs          K4D263238A
[End Board Description]
```

The .EBD command requires an instance name of the board to be specified. For this case, the name sb1 is used. The board definition can be written as:

```
* board description
.ebd sb1
+ file='4mx32.ebd'
+ model='board1'
+ component=bc1:u1
```

Now that both the ibis component and board description are defined, the input, enables, and terminations for the buffers can be defined. The node naming convention follows the node naming rules defined in Table 27 on page 249. The complete netlist can be written as:

```
* EBD Usage Example
* ibis component
.ibis bc1 component='K4D263238A'
+ file='4mx32.ibs'
+ ramp_rwf=0 ramp_fwf=0
+ pkgfile='4mx32.pkg'
+ package=3

* board description
.ebd sb1
+ file='4mx32.ebd'
+ model='board1'
+ component=bc1:u1

* enables
v3 bc1_b7_en 0 1
v4 bc1_c6_en  0 1
* data
vdq0 bc1_b7_i 0 pulse(0 1 1n 0.1n 0.1n 5n 20n)
vdq1 bc1_c6_i 0 pulse(0 1 1n 0.1n 0.1n 5n 15n)

* terminations
r1 sb1_b7 0 50
r2 sb1_c6 0 50
```

Example 2: .IBIS Command File

```
** HSPICE Netlist **
.ibis cmp1
+ file = 'test1.ibs'
+ component = 'Component1'
+ package = 1

.ebd ebd1
+ file = 'test.ebd'
+ model = 'Board1'
+ component = 'cmp1:u21'

** IBIS file: test1.ibs **
[Component] Component1
| ... ...
[Pin] signal_name model_name R_pin L_pin C_pin
|
1     RAS0       Buffer1   200.0m 5.0nH 2.0pF
2     gnd        GND       209.0m NA    2.5pF
3     pwr        POWER     NA     6.3nH NA
| ... ...
[Pin Mapping] pulldown_ref pullup_ref gnd_clamp_ref
power_clamp_ref ext_ref
|
1             GNDBUS1      PWRBUS1 |
2             GNDBUS1      NC      |
3             NC           PWRBUS1
| ... ...
[Series Pin Mapping] pin_2 model_name function_table_group
2                  3     CBTSeries   1 | Four independent groups
```

Example 2: .EBD Command File

Here is an example of an EBD file:

```
[Begin Board Description] Board1
|... ...
[Pin List] signal_name
A1 GND
A2 data1
A3 data2
|... ...
[Path Description] CAS_2
Pin A1
Len = 0.5 L=8.35n C=3.34p R=0.01 /
Node u21.1
Len = 0.5 L=8.35n C=3.34p R=0.01 /
Node u21.2
Len = 0.5 L=8.35n C=3.34p R=0.01 /| this section is
                                 | discarded because
                                 | the Node u23.3
                                 | reference
                                 | designator:u23 is not
                                 | designated to IBIS
                                 | component in the .ebd
                                 | ebd1 command.
```

The nodes listed in Table 30 are used with B-elements in the example above.

Table 30 B -element Nodes

Buffer/Node	Name
A_1	cmp1_1_i
B_1	cmp1_1_o
B_2	cmp1_2_o
B_3	cmp1_3_o
C_1	cmp1_1
C_2	cmp1_2
C_3	cmp1_3
D_1	ebd1_a1
D_2	ebd1_a2
D_3	ebd1_a3

Table 30 B -element Nodes (Continued)

Buffer/Node	Name
Pulldown of Buf_1	cmp1_2_o
Pullup of Buf_1	cmp1_3_o
GND Clamp of Buf_1	cmp1_1_gc
POWER Clamp of Buf_1	cmp1_1_pc
Buf_1	cmp1_1
S_Buf_1	cmp1_2_3

Using IBIS Interconnect Modeling (ICM)

ICM is a standard for behavioral descriptions of interconnect electrical characteristics. The text describes a consistent format that can be parsed by software, allowing simulation tool and interconnect vendors to create models compatible with their own products.

The HSPICE `.ICM` command automatically creates port names that reference the pin name of an ICM model and generates a series of element nodes on the pin.

Syntax

```
.ICM icmname
+ file='icmfilename'
+ model='icmmodelname'
```

Example

```
.ICM icm1
+ file='test1.icm'
+ model='FourLineModel1'
```

For the complete description of the command, See the *HSPICE Reference Manual: Commands and Control Options,* .ICM.

5

Ideal and Lumped Transmission Line Models

Describes ideal transmission lines using the T-element and lossy transmission lines using the U-element. The U-model computes transmission line electrical properties from common wire and interconnect cross-sectional geometries.

HSPICE ships hundreds of examples for your use. Find paths to U-element demo files under Transmission Lines Examples and Signal Integrity Examples.

Some applications do not require the sophisticated models used by the HSPICE W-element or S-element in order to well predict transmission line effects. In such cases, the less complex T-element and U-element transmission line models can provide adequate solutions.

The T-element is useful for modeling ideal transmission lines in terms of their impedance and delay. The U-element is useful for modeling lossy non-ideal transmission lines in terms of lumped-element equivalent circuits. The T- and U-elements can model effects such as:

- Time delay
- Phase shift
- Coupling and Crosstalk
- Distortion

For information regarding the physics of modeling transmission lines with ideal and lumped equivalents, see Appendix A, Transmission Line Theory.

These topics are discussed below:

- Selecting Ideal or Lossy Transmission Line Elements
- Ideal T-element Transmission Lines
- The Lossy U-element Transmission Line

- The U Model for Transmission Lines
- U-element Examples, Models, and Applications

Selecting Ideal or Lossy Transmission Line Elements

An electrical model that simulates the behavior of an interconnect or transmission line must consider all of the following:

- Physical nature or electrical properties of the interconnect
- Bandwidth or risetime of signals of interest
- Interconnect's actual time delay or equivalent electrical length
- Complexity and accuracy of the model

When accuracy is the utmost priority, and all high-frequency effects must be taken into account, use the W-element or S-element with models based on measurements or EM analyses. When basic models are adequate, you can choose from the following circuit models for interconnects:

- No model at all. Use a common node to connect two elements.
- Lumped models with R, L, and C-elements as described in the *HSPICE User Guide: Simulation and Analysis*. These include a series resistor (R), a shunt capacitor (C), a series inductor and resistor (RL), or a series resistor and a shunt capacitor (RC).
- Transmission line models such as an ideal transmission line (T-element) or a lossy transmission line (U-element)

The following tables and charts explain the factors involved when determining the best choice between RLC-elements, the T-element, or U-element transmission line models.

The following sections discuss these topics:

- Source Properties
- Interconnect Properties
- Selection of Ideal or Lossy Transmission Line Elements
- Transmission Lines: Example

Source Properties

t_{rise} = source risetime

R_{source} = source output impedance

Interconnect Properties

Z0 = characteristic impedance

TD = time delay of the interconnection

or

R = equivalent series resistance

C = equivalent shunt capacitor

L = equivalent series inductance

Figure 71 shows how to select a model based on source and interconnect properties.

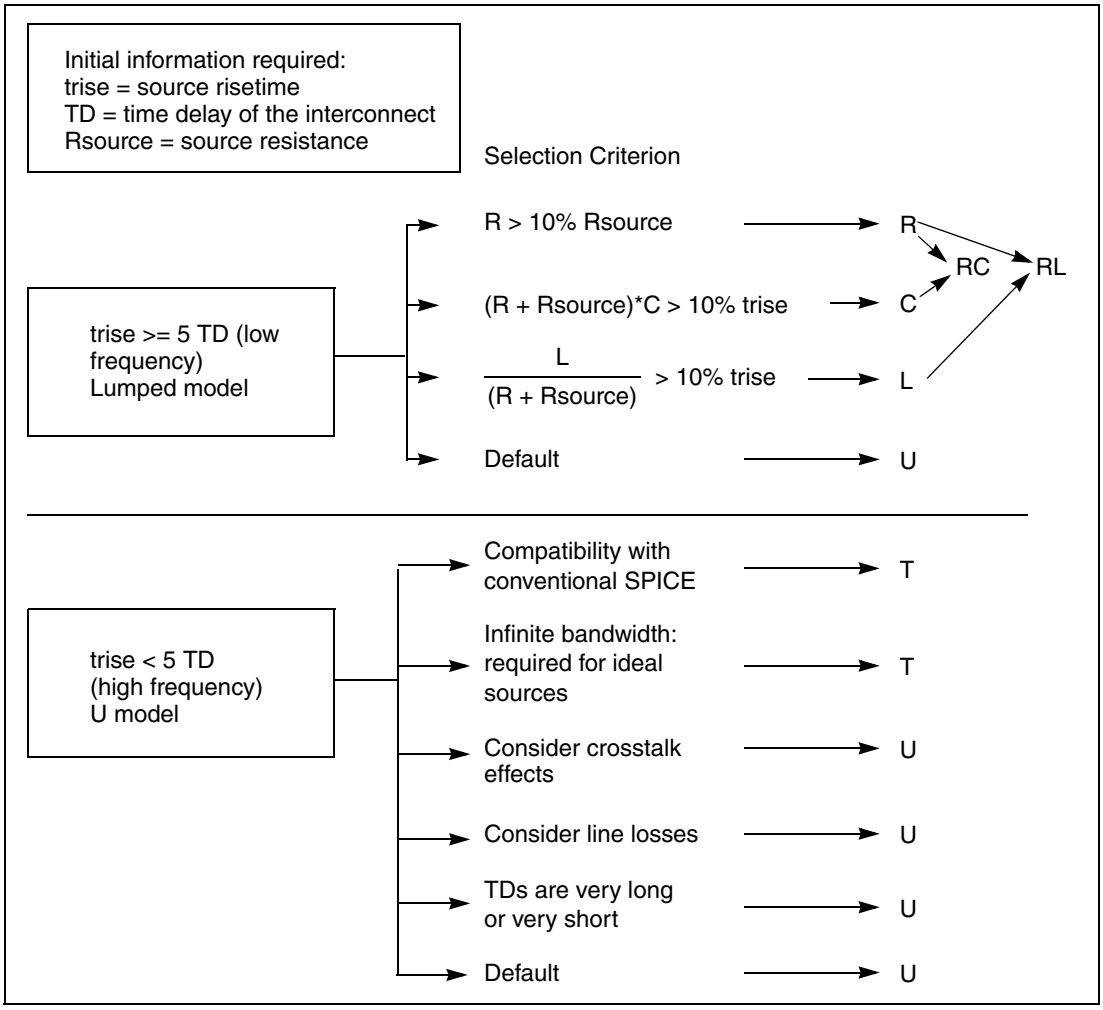


Figure 71 Wire Model Selection Chart

Use the U model with either the ideal T-element or the lossy U-element. You can also use the T-element alone without the U model. HSPICE offers both a flexible definition of the conventional SPICE T-element and an accurate U-element lossy simulation.

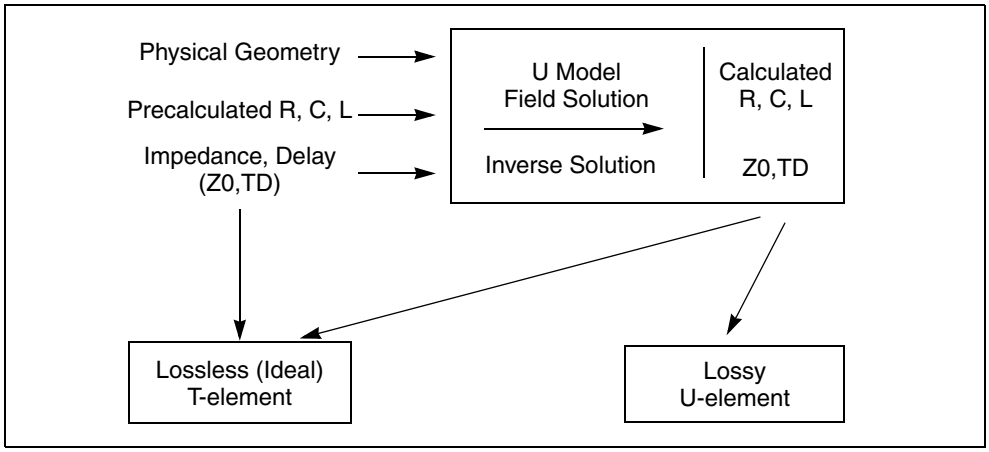


Figure 72 U Model, T-element and U-element Relationship

The T and U-elements do not support the [M=*val*] multiplier function. If a U or T element is used in a subcircuit and an instance of the subcircuit has a multiplier applied, that multiplier is not applied to the T or U-element.

A warning message similar to the following is issued in both the status file (.st0) and the output file (*.lis*) if the smallest transmission line delay is less than TSTOP/10e6:

```
**warning**: the smallest T-line delay (TD) = 0.245E-14 is too small
Please check TD, L and SCALE specification
```

This feature is an aid to finding errors that cause excessively long simulations.

Note: All transmission lines have a ground reference for the signal conductors. In this manual, the ground reference is called the reference plane so that it is not confused with SPICE ground. The reference plane is the shield or the ground plane of the transmission line element. The reference plane nodes may or may not be connected to SPICE ground.

Selection of Ideal or Lossy Transmission Line Elements

The ideal and lossy transmission line models each have particular advantages, and they may be used in a complementary manner. Both model types are fully

functional in AC analysis and transient analysis. Some of the comparative advantages and uses of each type of model are listed in Table 31.

Table 31 Ideal versus Lossy Transmission Line

Ideal Transmission Line	Lossy Transmission Line
lossless	includes loss effects
used with voltage sources	used with buffer drivers
no limit on input risetime	prefiltering necessary for fast rise
less CPU time for long delays	less CPU time for short delays
differential mode only	supports common mode simulation
no ground bounce	includes reference plane reactance
single conductor	up to five signal conductors allowed
AC and transient analysis	AC and transient analysis

The ideal line is modeled as a voltage source and a resistor. The lossy line is modeled as a multiple lumped filter section as shown in Figure 73.

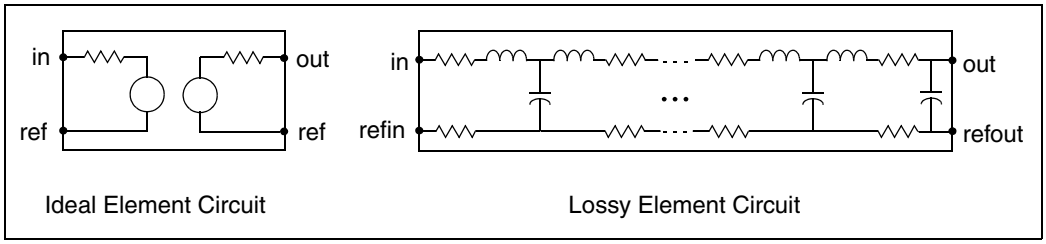


Figure 73 Ideal versus Lossy Transmission Line Model

Because the ideal element represents the complex impedance as a resistor, the transmission line impedance is constant, even at DC values. On the other hand, you must prefilter the lossy element if ideal piecewise linear voltage sources are used to drive the line.

Transmission Lines: Example

The following file fragment is an example of how both T-elements and U-elements can be referred to a single U model as indicated in Figure 72 on page 267. The file specifies a 200 mm printed circuit wire implemented as both a U-element and a T-element. The two implementations share a U model that is a geometric description (`ELEV=1`) of a planar structure (`PLEV=1`).

```
T1 in gnd t_out gnd micro1 L=200m
U1 in gnd u_out gnd micro1 L=200m
.model micro1 U LEVEL=3 PLEV=1 ELEV=1 wd=2m ht=2m th=0.25m
+ KD=5
```

The next section provides details of element and model syntax.

Table 32 Transmission Line Parameters

Parameter	Description
T1, U1	Element names
micro1	The model name
in, gnd, t_out, and u_out	Nodes
L	The length of the signal conductor
wd, ht, th	Dimensions of the signal conductor and dielectric
KD	The relative dielectric constant

Ideal T-element Transmission Lines

The following sections discuss these topics:

- Syntax

Syntax

General form:

`Txxx in refin out refout Z0=val TD=val [L=val]`

```
+ [IC=v1,i1,v2,i2]

Txxx in refin out refout Z0=val F=val [NL=val]
+ [IC=v1,i1,v2,i2]
```

U Model form:

```
Txxx in refin out refout mname L=val
```

Parameter	Description
Txxx	Lossless transmission line element name. Must begin with T, followed by up to 1023 alphanumeric characters.
in	Signal input node.
refin	Ground reference for the input signal.
out	Signal output node.
refout	Ground reference for the output signal.
Z0	Characteristic impedance of the transmission line (Ohms).
TD	Propagation time delay of the transmission line (in seconds). If physical length (L) is specified, then units for TD are considered in seconds per meter.
L	Physical length of the transmission line, in units of meters. Default=1.
IC=v1,i1,v2,i2	Initial conditions of the transmission line. Specify the voltage on the input port (v1), current into the input port (i1), voltage on the output port (v2), and the current into the output port (i2).
F	Frequency at which the transmission line has the electrical length specified in NL.
NL	Normalized electrical length of the transmission line (at the frequency specified in the F parameter), in units of wavelengths per line length. Default=0.25, which is a quarter-wavelength.
mname	U-model reference name. A lossy transmission line model, representing the characteristics of the lossless transmission line.

Only one input and output port is allowed.

Example 1

The T1 transmission line connects the in node to the out node:

```
T1 in gnd out gnd Z0=50 TD=5n L=5
```

- Both signal references are grounded.
- Impedance is 50 ohms.
- The transmission delay is 5 nanoseconds per meter.
- The transmission line is 5 meters long.

Example 2

The Tcable transmission line connects the in1 node to the out1 node:

```
Tcable in1 gnd out1 gnd Z0=100 F=100MEG NL=1
```

- Both signal references are grounded.
- Impedance is 100 ohms.
- The normalized electrical length is 1 wavelength at 100 MHz.

Example 3

The Tnet1 transmission line connects the driver node to the output node:

```
Tnet1 driver gnd output gnd Umodel1 L=1m
```

- Both signal references are grounded.
- Umodel1 references the U-model.
- The transmission line is 1 millimeter long.

Lossless Voltage and Current Propagation

For the ideal transmission line, voltage and current propagate without loss along the length of the line (±x direction) with spatial and time-dependence given according to the following equation:

$$v(x,\ t)\ =\ Re[Ae^{j(\omega t - \beta x)} + Be^{j(\omega t + \beta x)}]$$

$$v(x,\ t)\ =\ Re\left[\frac{A}{Z_0}e^{j(\omega t - \beta x)} - \frac{B}{Z_0}e^{j(\omega t + \beta x)}\right]$$

The A represents the incident voltage, B represents the reflected voltage, Z_0 is the characteristic impedance, and β is the propagation constant. The latter are related to the transmission line inductance (L) and capacitance (C) by the following equation:

$$Z_0 = \sqrt{\frac{L}{C}}$$

$$\beta = \omega\sqrt{LC}$$

The L and C terms are in per-unit-length units (Henries/meter, Farads/meter). The following equation gives the phase velocity:

$$\upsilon_\rho = \frac{\omega}{\beta} = \frac{1}{\sqrt{LC}}$$

At the end of the transmission line ($x = l$), the propagation term βl becomes the following equation:

$$\beta l = \omega\sqrt{LC} \cdot l = \omega\frac{l}{v_p}$$

This is equivalent to an ideal delay with the following value:

$$T = \frac{l}{V_P} = \sqrt{LC} \cdot l$$

Where,

T: absolute time delay (sec)

l: physical length (L) (meters)

V_P: phase velocity (meters/sec)

Using standard distance=velocity*time relationships, the HSPICE T-element parameter values are related to these terms according to:

$$V_P = f \cdot \lambda = \frac{1}{t_d}$$

Where,

f: frequency

λ: wavelength

t_d: relative time delay (TD) (sec/meter)

$$T = \frac{l}{V_p} = t_d \cdot l = \frac{l}{f \cdot \lambda} = \frac{l/\lambda}{f} = \sqrt{LC} \cdot l$$

Where,

l: physical length (L) (meters)

$l/\ \lambda$: normalized length (NL)

f: frequency at NL (F) (Hz)

$$T\ =\ TD\cdot\ \ L=\frac{NL}{L}=\sqrt{LC}\cdot\ \ L$$

The T-element is therefore an ideal behavioral model given in terms of Zo and delay, yet you can relate it back to inductance, capacitance, velocity, and/or length quantities as needed. HSPICE allows you to specify the T-element transmission line in four different ways:

- Z_0, TD
- Z_0, TD, L
- Z_0, NL, F
- L, with $\sqrt{\frac{L}{C}}$ and $\sqrt{LC}$ values taken from a U model.

Ideal Transmission Line Model

The ideal transmission model is based on a characteristic impedance (Z_0), and propagation delay (TD). The Z_0 and TD values may be obtained from a U model.

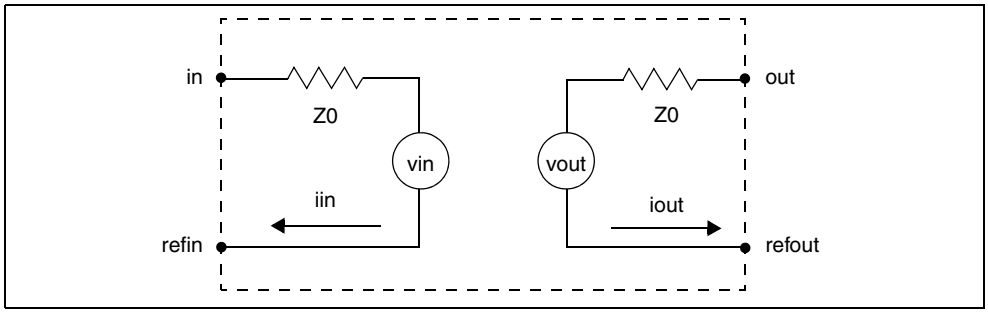


Figure 74 Ideal Element Circuit

The input and output of an ideal transmission line have these relationships:

$$Vin\big|_t\ =\ V(out-refout)\big|_{t-TD}+(iout\times Z_0)\big|_{t-TD}$$

$$Vout|_t = V(in - refin)|_{t-TD} + (iin \times Z_0)|_{t-TD}$$

The signal (propagation) delays for ideal transmission lines are specified as:

- TD_{eff}=TD (if only TD is given), or
- TD_{eff}=TD*L if TD and L are given, or
- TD_{eff}=NL/F if NL and F are given, or
- TD_{eff}=TD if you use a U model.

The ideal transmission line only delays the difference between the signal and the reference. Note that although you can specify length as an input parameter, it is used only to compute the effective propagation delay. Some applications, such as a differential output driving twisted pair cable, require both differential and common mode propagation. Use a W-element or U-element, if you need to model both propagation modes and coupling. If not, you can use two T-elements (see Figure 75) to model uncoupled differential signals. In this figure, the two lines are completely uncoupled but the delay and impedance values are correctly modeled.

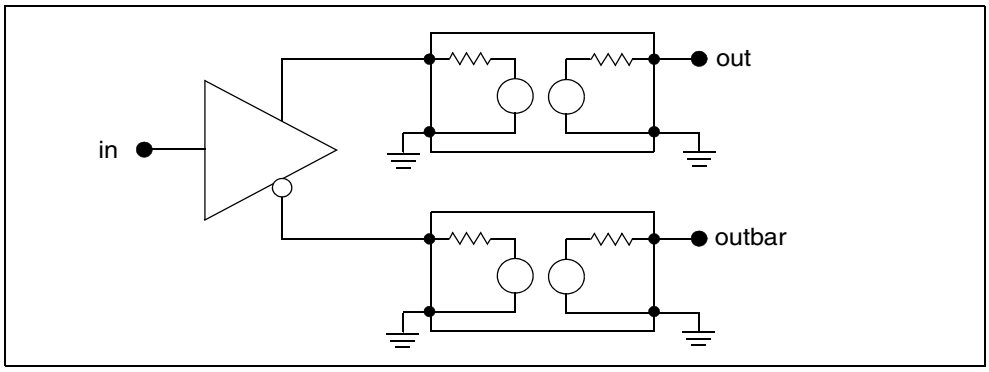


Figure 75 Use of Two T-elements for Differential Signals

You cannot implement coupled lines with the T-element so use W-elements or U-elements for applications requiring two or three coupled conductors.

Synopsys circuit simulation uses a transient timestep that does not exceed half the minimum line delay. Very short transmission lines (relative to the analysis time step) cause long simulation times. You can replace very short lines with a single R, L, or C-element (see Figure 71 on page 266).

The Lossy U-element Transmission Line

The following sections discuss these topics:

- Syntax

Syntax

```
Uxxx in1 [in2 ...in5] refin out1 [out2 ...out5]
+ refout mname L=val [LUMPS=val]
```

Parameter	Description
Uxxx	Lossy (U-element) transmission line element name. Must begin with U, followed by up to 1023 alphanumeric characters.
inx	Signal input node for the x^{th} transmission line (in1 is required).
refin	Ground reference for the input signal.
outx	Signal output node for the x^{th} transmission line (each input port must have a corresponding output port).
refout	Ground reference for the output signal.
mname	Model reference name for the U-model lossy transmission-line.
L	Physical length of the transmission line, in units of meters.
LUMPS	Number of lumped-parameter sections used to simulate the element.

In this syntax, the number of ports on a single transmission line is limited to five in and five out. One input and output port, the ground references, a model reference, and a length are all required.

The U-element models single and coupled lossy transmission lines for various planar, coaxial, and twinlead structures. When a U-element is included in your netlist, simulation creates an internal network of R, L, C, and G-elements to represent up to five lines and their coupling capacitances and inductances.

You can specify interconnect properties in three ways:

- Specify the R, L, C, and G (conductance) parameters in a matrix form (ELEV=2).
- Provide common electrical parameters, such as characteristic impedance and attenuation factors (ELEV=3).
- Specify the geometry and the material properties of the interconnect (ELEV=1).

This section initially describes how to use the third method.

The U Model is optimized for typical geometries used in ICs, MCMs, and PCBs. The model's closed form expressions are optimized via measurements and comparisons with several different electromagnetic field solvers.

The U-element geometric model handles one to five uniformly spaced transmission lines, all at the same height. Also, the transmission lines can be on top of a dielectric (microstrip), buried in a sea of dielectric (buried), have reference planes above and below them (stripline), or have a single reference plane and dielectric above and below the line (overlay). Thickness, conductor resistivity, and dielectric conductivity allow for calculating loss as well.

The U-element statement contains the element name, the connecting nodes, the U model reference name, the length of the transmission line, and, optionally, the number of lumps in the element. You can create two kinds of lossy lines: lines with a reference plane inductance (LRR, controlled by the model parameter LLEV) and lines without a reference plane inductance. Wires on integrated circuits and printed circuit boards require reference plane inductance.

The reference ground inductance and the reference plane capacitance to SPICE ground are set by the HGP, CMULT, and optionally, the CEXT parameters.

The U Model for Transmission Lines

Syntax

The following shows the netlist syntax for the U model.

```
.MODEL mname U LEVEL=3 ELEV=val PLEV=val <DLEV=val>
+ <LLEV=val> + <Pname=val> ...
```

Parameter	Description
LEVEL=3	Selects the lossy transmission line model
ELEV=val	Selects the electrical specification format including the geometric model val=1
PLEV=val	Selects the transmission line type
DLEV=val	Selects the dielectric and ground reference configuration
LLEV=val	Selects the use of reference plane inductance and capacitance to HSPICE ground.
Pname=val	Specifies a physical parameter, such as NL or WD (see Table 33 on page 280) or a loss parameter, such as RHO or NLAY (see Table 34 on page 282).

Table 33 on page 280 and Table 34 on page 282 list the model parameters.

In the above figure, a) sea, `DLEV=0`; b) microstrip, `DLEV=1`; c) stripline, `DLEV=2`; and d) overlay, `DLEV=3`.

The following sections discuss these topics:

- Selecting U Models
- Lossy U Model Parameters for Planar Geometries
- Lossy U Model Parameters for Geometric Coax (PLEV=2, ELEV=1)
- Lossy U Model Parameters Geometric Twinlead (PLEV=3, ELEV=1)

Selecting U Models

The U model allows three different description formats: geometric/physical, precomputed, and electrical. It provides equally natural description of vendor parts, physically described shapes, and parametric input from field solvers. The description format is specified by the required model parameter `ELEV` as follows:

- `ELEV=1` – geometric/physical description such as width, height, and resistivity of conductors. It is used by board designers dealing with physical design rules.
- `ELEV=2` – precomputed parameters. These are available with some commercial packaging, or as a result of running a field solver on a physical description of commercial packaging.
- `ELEV=3` – electrical parameters such as delay and impedance, available with purchased cables. It allows one conductor and ground plane for `PLEV=1`.

The U model explicitly supports transmission lines with several types of geometric structures. The geometric structure type is indicated by the `PLEV` model parameter as follows:

- `PLEV=1` – Selects planar structures, such as microstrip and stripline, (the usual conductor shapes on integrated circuits and printed-circuit boards).
- `PLEV=2` – Selects coax, which frequently is used to connect separated instruments.
- `PLEV=3` – Selects twinlead, which is used to connect instruments and to suppress common mode noise coupling.

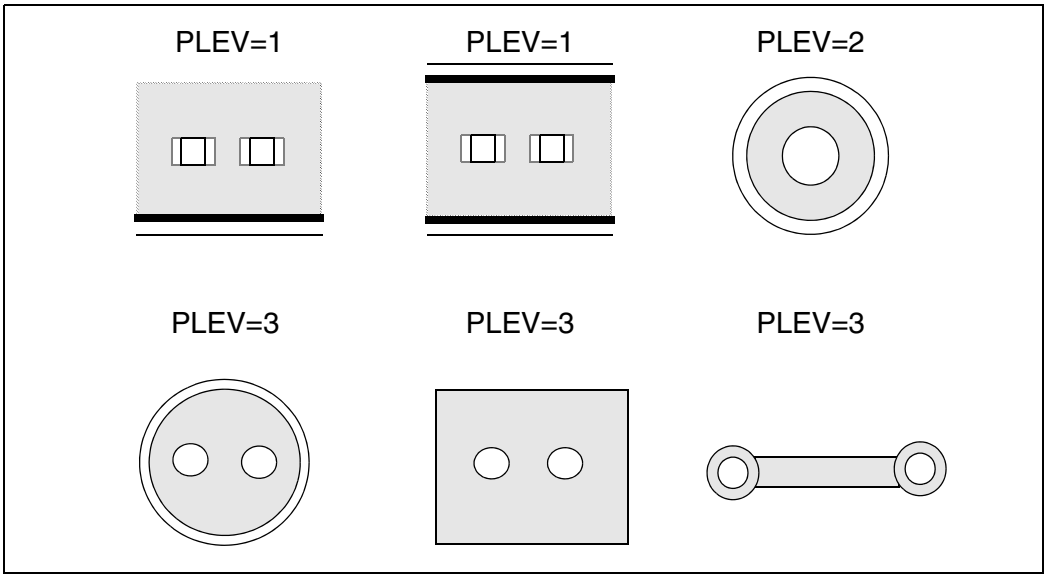


Figure 76 U Model Geometric Structure (PLEV) Options

Figure 77 shows the four dielectric (DLEV) configurations available for the geometric U model. You use the DLEV switch to specify one of these configurations. The geometric U model uses ELEV=1, and the configurations shown are for PLEV=1 planar geometries.

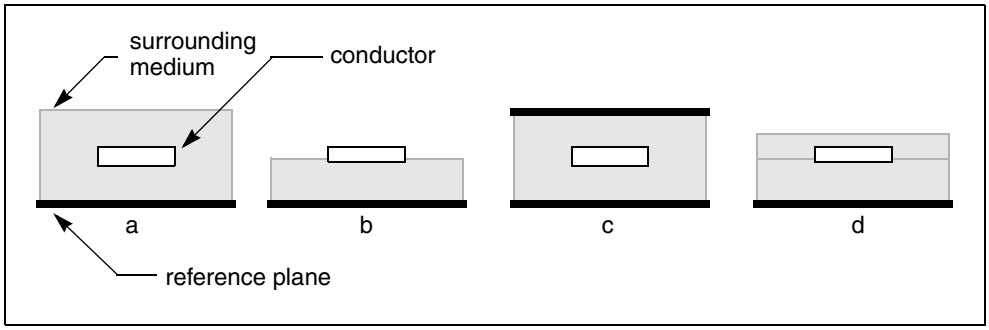


Figure 77 Dielectric and Reference Plane (DLEV) Configuration Options

In the above figure, a) sea, DLEV=0; b) microstrip, DLEV=1; c) stripline, DLEV=2; and d) overlay, DLEV=3.

Lossy U Model Parameters for Planar Geometries

The following sections discuss these topics:

- Common Planar Model Parameters
- Physical Parameters
- Loss Parameters
- Geometric Parameter Recommended Ranges
- Reference Planes and HSPICE Ground with LLEV
- Estimating the Skin Effect Frequency
- Number of Lumped-Parameter Sections
- Ringing
- Geometric Parameters (ELEV=1)

Common Planar Model Parameters

The parameters for U models are listed in Table 33.

Table 33 U-element Physical Parameters

Parameter	Units	Default	Description
LEVEL		required	(=3) required for lossy transmission lines model
ELEV		required	Electrical model (=1 for geometry)
DLEV			Dielectric model (=0 for sea, =1 for microstrip, =2 stripline, =3 overlay; default is 1)
PLEV		required	Transmission line physical model (=1 for planar)
LLEV			Omit or include the reference plane inductance (=0 to omit, =1 to include; default is 0)
NL			Number of conductors (from 1 to 5)
WD	m		Width of each conductor
HT	m		Height of all conductors
TH	m		Thickness of all conductors
THB	m		Reference plane thickness
TS	m		Distance between reference planes for stripline (default for DLEV=2 is 2 HT + TH. TS is not used when DLEV=0 or 1)
SP	m		Spacing between conductors (required if NL > 1)
KD			Dielectric constant
XW	m		Perturbation of conductor width added (default is 0)
CEXT	F/m		External capacitance between reference plane and ground. Only used when LLEV=1, this overrides the computed characteristic.
CMULT		1	Dielectric constant of material between reference plane and ground (default is 1 – only used when LLEV=1)

Table 33 U-element Physical Parameters (Continued)

Parameter	Units	Default	Description
HGP	m		Height of the reference plane above HSPICE ground. Used for computing reference plane inductance and capacitance to ground (default is 1.5*HT – HGP is only used when LLEV=1).
CORKD			Perturbation multiplier for dielectric (default is 1)
WLUMP		20	Number of lumps per wavelength for error control
MAXL		20	Maximum number of lumps per element

The U model has two parametric adjustments: `XW` and `CORKD`. `XW` adds to the width of each conductor, but does not change the conductor pitch (spacing plus width). It is useful for examining the effects of conductor etching. `CORKD` is a multiplier for the dielectric value. Some board materials vary more than others. It provides an easy way to test tolerance to dielectric variations.

Physical Parameters

The dimensions for one and two-conductor planar transmission lines are shown in Figure 78.

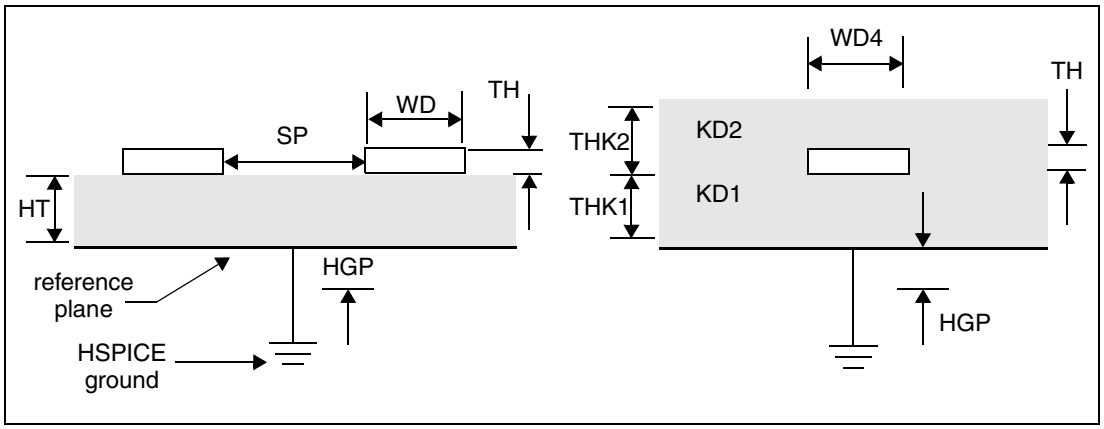


Figure 78 U -element Conductor Dimensions

Loss Parameters

Table 34 lists the loss parameters for the U model.

Table 34 U -element Loss Parameters

Parameter	Units	Description
RHO	ohm· m	Conductor resistivity (default is rho of copper, 17E-9 ohm· m)
RHOB	ohm· m	Reference plane resistivity (default value is for copper)
NLAY		Number of layers for conductor resistance computation (=1 for DC resistance or core resistance, =2 for core and skin resistance at skin effect frequency)
SIG	mho/m	Dielectric conductivity

Losses have a large impact on circuit performance, especially as clock frequencies increase. `RHO`, `RHOB`, `SIG`, and `NLAY` are parameters associated with losses. Time domain simulators, such as SPICE, cannot directly handle losses that vary with frequency. Both the resistive skin effect loss and the effects of dielectric loss create loss variations with frequency. `NLAY` is a switch that turns on skin effect calculations in circuit simulation and analysis. The skin effect resistance is proportional to the conductor and backplane resistivities, `RHO` and `RHOB`.

Dielectric conductivity is included using `SIG`. The U model computes skin effect resistance at a single frequency and uses that resistance as a constant. The dielectric `SIG` computes a fixed conductance matrix, which is constant for all frequencies. To closely approximate losses, compute resistances and conductances at the frequency of maximum power dissipation. In AC analysis, resistance increases as the square root of frequency above the skin-effect frequency, and resistance is constant below the skin effect frequency.

Geometric Parameter Recommended Ranges

The U-element analytic equations compute quickly, but have a limited range of validity. The U-element equations were optimized for typical IC, MCM, and PCB

applications. Table 35 lists the recommended minimum and maximum values for U-element parameter variables.

Table 35 Recommended Ranges

Parameter	Minimum	Maximum
NL	1	5
KD	1	24
WD/HT	0.08	5
TH/HT	0	1
TH/WD	0	1
SP/HT	0.15	7.5
SP/WD	1	5

The U-element equations lose their accuracy when you use values outside the recommended ranges. Because the single-line formula is optimized for single lines, you can see a difference between the parameters of single lines and two coupled lines at a very wide separation. The absolute error for a single line parameter is less than 5% when used within the recommended range. The main line error for coupled lines is less than 15%. Coupling errors can be as high as 30% in cases of very small coupling. Since the largest errors occur at small coupling values, actual waveform errors are kept small.

Reference Planes and HSPICE Ground with LLEV

The schematic for a single lump of the U model with `LLEV=0`, is shown in Figure 79. If `LLEV=1`, the schematic includes inductance in the reference path as well as capacitance to HSPICE ground.

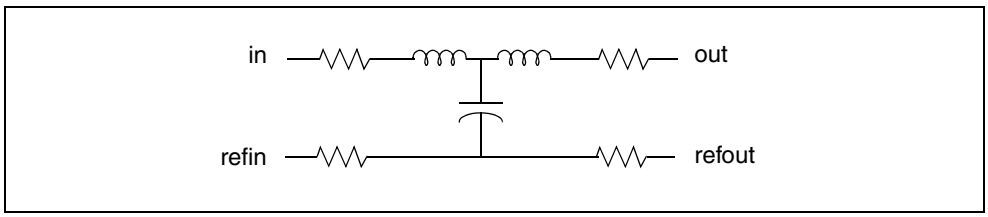


Figure 79 Lossy Line with Reference Plane when LLEV=0

Figure 80 shows a single lump of a U model for a single line with reference plane inductance. If `LLEV=1`, the reference plane inductance is computed, and capacitance from the reference plane to HSPICE ground is included in the model. The reference plane is the ground plane of the conductors in the U model.

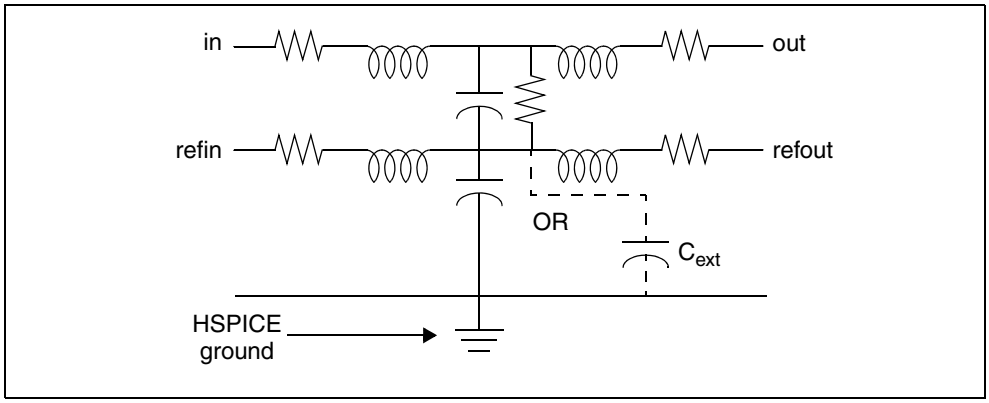


Figure 80 Schematic of a U-element Lump when LLEV=1

The model reference plane is not necessarily the same as HSPICE ground. For example, a printed circuit board with transmission lines can have a separate reference plane above a chassis. Simulation with the U model uses either HGP, the distance between the reference plane and HSPICE ground, or C_{ext} to compute the parameters for the ground-to-reference transmission line.

If you use `HGP`, the capacitance per meter of the ground-to-reference line is computed based on a planar line of width (NL+2)(WD+SP) and the height of `HGP` above the SPICE ground. `CMULT` is the dielectric constant of the ground-to-reference transmission line. If you specify C_{ext}, then C_{ext} is the capacitance per meter for the ground-to-reference line. The inductance of the ground-to-

reference line is computed from the capacitance per meter and an assumed propagation at the speed of light.

Estimating the Skin Effect Frequency

Most of the power in a transmission line is dissipated at the clock frequency. As a first choice, simulation estimates the maximum dissipation frequency, or skin effect frequency, from the risetime parameter. The risetime parameter is set with the `.OPTION` statement (for example, `.OPTION RISETIME=0.1ns`).

Some designers use 0.35/*trise* to estimate the skin effect frequency. This estimate is good for the bandwidth occupied by a transient, but not for the clock frequency, at which most of the energy is transferred. In fact, a frequency of 0.35/trise is far too high and results in excessive loss for almost all applications. Simulation computes the skin effect frequency from 1/(15*trise). If you use precomputed model parameters (`ELEV=2`), compute the resistance matrix at the skin effect frequency.

When the risetime parameter is not given, simulation uses other parameters to compute the skin effect frequency. The circuit simulator examines the `.TRAN` statement for tstep and delmax and examines the source statement for trise. If you set any one of the parameters tstep, delmax, and trise, simulation uses the maximum of these parameters as the effective risetime.In AC analysis, the skin effect is evaluated at the frequency of each small-signal analysis. Below the computed skin effect frequency (`ELEV=1`) or `FR1` (`ELEV=3`), the AC resistance is constant. Above the skin effect frequency, the resistance increases as the square root of frequency.

Number of Lumped-Parameter Sections

The number of sections (lumps) in a transmission line model also affects the transmission line response. Simulation computes the default number of lumps from the line delay and the signal risetime. There should be enough lumps in the transmission line model to ensure that each lump represents a length of line that is a small fraction of a wavelength at the highest frequency used. It is easy to compute the number of lumps from the line delay and the signal risetime by using an estimate of 0.35/trise as the highest frequency.

For the default number of lumps, simulation uses the smaller of 20 or 1+(20*TDeff/trise), where TDeff is the line delay. In most transient analysis cases by using more than 20 lumps gives a negligible bandwidth improvement at the cost of increased simulation time. In AC simulations over many decades of frequency with lines over one meter long, more than 20 lumps may be needed for accurate simulation.

Ringing

Sometimes a transmission line simulation shows ringing in the waveforms as in Figure 97 on page 308. If the ringing is not verifiable by measurement, it might be due to an incorrect number of lumps in the transmission line models or due to the simulator integration method. Increasing the number of lumps in the model or changing the integration method to Gear reduces the amount of ringing due to simulation errors. The default integration method is `TRAP` (trapezoidal), but you can change it to Gear with the statement `.OPTION METHOD=GEAR`.

See Oscillations Due to Simulation Errors on page 326 for more information about the number of lumps and ringing.

The next section covers parameters for geometric lines. It discusses coaxial and twinlead transmission lines, and the previously-described planar type.

Geometric Parameters (ELEV=1)

Geometric parameters provide a description of a transmission line in terms of the geometry of its construction and the physical constants of each layer or other geometric shape involved.

PLEV=1, ELEV=1 Geometric Planar Conductors

Planar conductors are used to model printed circuit boards, packages, and integrated circuits. The geometric planar transmission line is restricted to:

- One conductor height (`HT` or `HT1`)
- One conductor width (`WD` or `WD1`)
- One conductor thickness (`TH` or `TH1`)
- One conductor spacing (`SP` or `SP12`)
- One dielectric conductivity (`SIG` or `SIG1`)
- One or two relative dielectric constants (`KD` or `KD1`, and `KD2` only if `DLEV=3`)

Common planar conductors include:

- `DLEV=0` – microstrip sea of dielectric. This planar conductor has a single reference plane and a common dielectric surrounding conductor (see Figure 81 on page 287).
- `DLEV=1` – microstrip dual dielectric. This planar conductor has a single reference plane and two dielectric layers (see Figure 83 on page 288).

- `DLEV=2` – stripline. This planar conductor has an upper and lower reference plane (see Figure 82 on page 288). Both symmetric and asymmetric spacing are available.
- `DLEV=3` – overlay dielectric. This planar conductor has a single reference plane and an overlay of dielectric material covering the conductor (see Figure 84 on page 289).

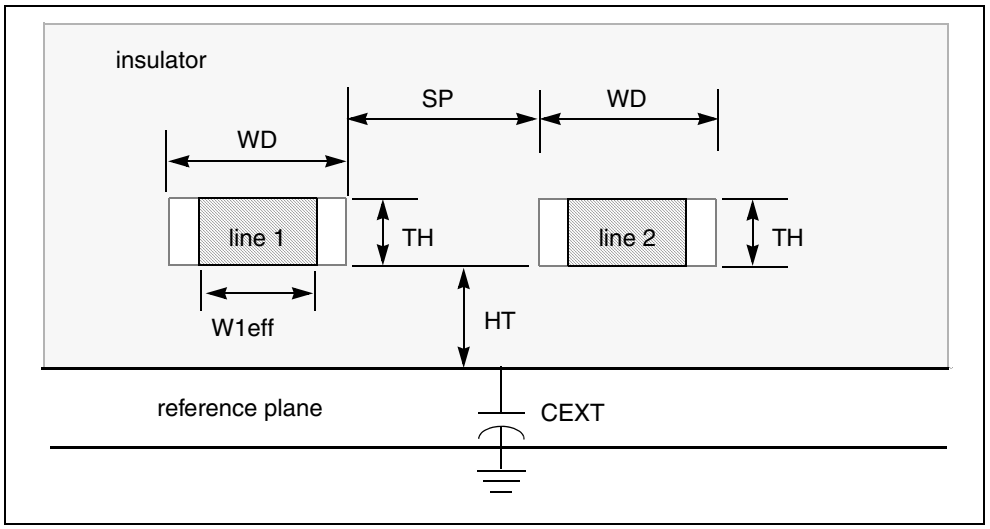


Figure 81 Planar Transmission Line, DLEV=0, Sea of Dielectric

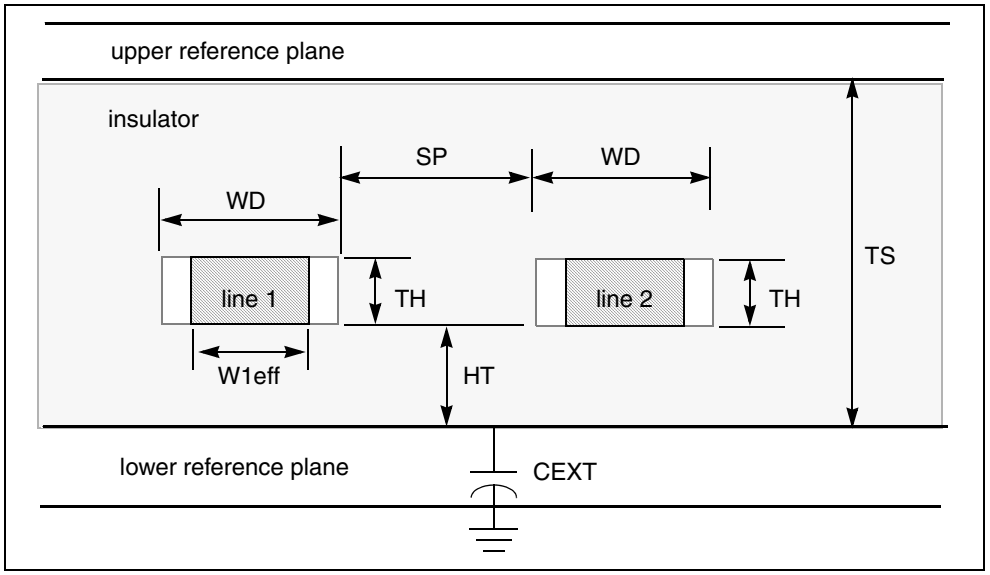


Figure 82 Planar Transmission Line, DLEV=2, Stripline

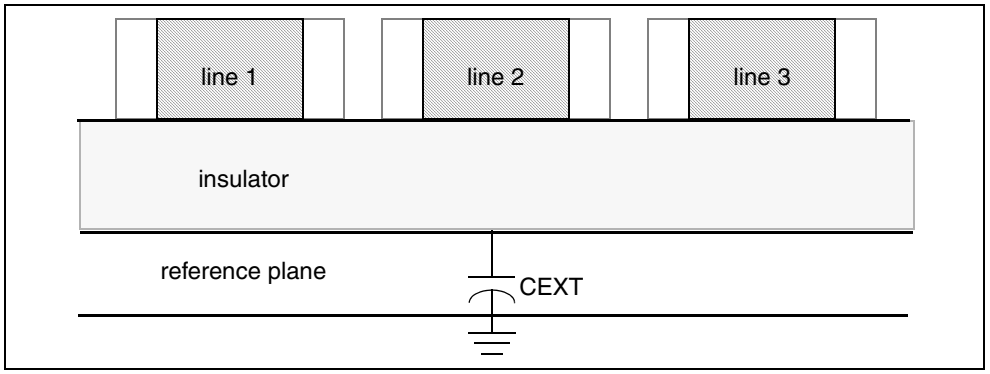


Figure 83 Planar Transmission Line, DLEV=1, Microstrip

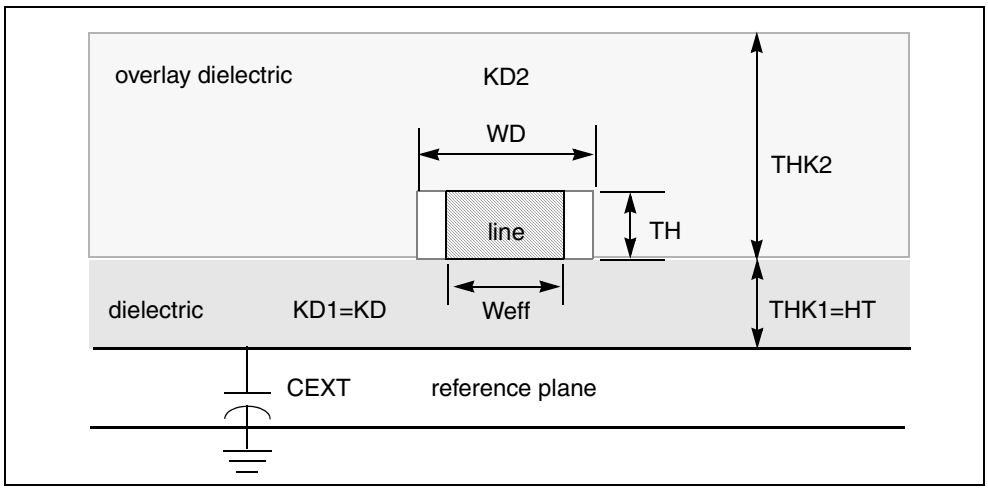


Figure 84 Planar Transmission Line, DLEV=3, Overlay Dielectric

Table 36 Geometric Parameters

Name (Alias)	Units	Default	Description
DLEV	—	1.0	0: microstrip sea of dielectric 1: microstrip layered dielectric 2: stripline
NL	—	1	Number of conductors
NLAY	—	1.0	Layer algorithm: 1: DC cross section only 2: skindepth cross section on surface plus DC core
HT(HT1)	m	req	Conductor height
WD(WD1)	m	req	Conductor width
TH(TH1)	m	req	Conductor thickness
THK1	m	HT	Dielectric thickness for DLEV=3
THK2	m	0.0	Overlay dielectric thickness for DLEV=3 0≤THK2<3· HT (see note after this table)

Table 36 Geometric Parameters (Continued)

Name (Alias)	Units	Default	Description
THB	m	calc	Reference conductor thickness
SP(SP12)	m	req	Spacing: line 1 to line 2 required for nl > 1
XW	m	0.0	Difference between drawn and realized width
TS	m	calc	Height from bottom reference plane to top reference plane TS=TH+2· HT (DLEV=2, stripline only)
HGP	m	HT	Height of reference plane above SPICE ground – LLEV=1
CMULT	—	1.0	CPR multiplier for dielectric constant of material between shield and SPICE ground if LLEV=1 and CEXT is not present
CEXT	F/m	und	External capacitance from reference plane to circuit ground point – used only to override HGP and CMULT computation
RHO	ohm· m	17E-9	Resistivity of conductor material – defaults to value for copper
RHOB	ohm· m	rho	Resistivity of reference plane material
SIG1(SIG)	mho/m	0.0	Conductivity of dielectric
KD1(KD)	—	4.0	Relative dielectric constant of dielectric
KD2		KD	Relative dielectric constant of overlay dielectric for DLEV=3 1 < KD1 < 4· KD
CORKD	—	1.0	Correction multiplier for KD

Note: If `THK2` is greater than three times `HT`, simulation accuracy decreases. A warning message is issued to indicate this. A reference plane is a ground plane, but it is not necessarily at SPICE ground potential.

Lossy U Model Parameters for Geometric Coax (PLEV=2, ELEV=1)

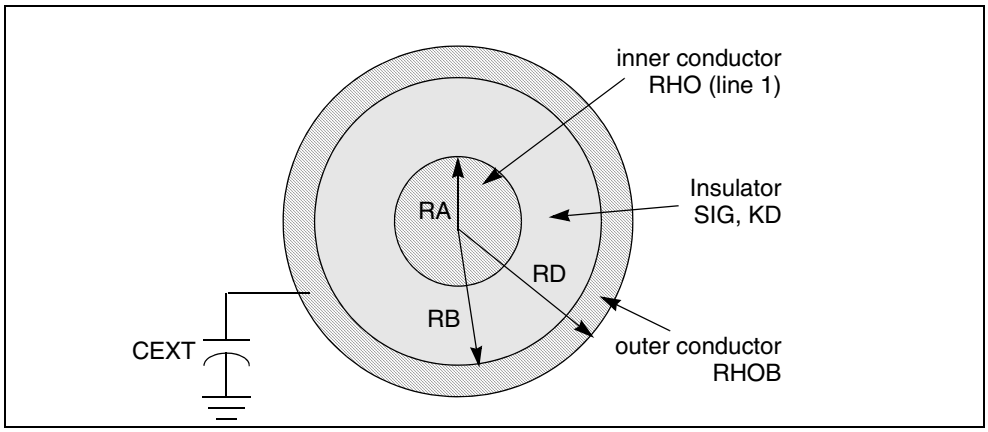


Figure 85 Geometric Coaxial Cable

Table 37 Lossy U Model Parameters

Name (Alias)	Units	Default	Description
RA	m	req	Outer radius of inner conductor
RB	m	req	Inner radius of outer conductor (shield)
RD	m	ra+rb	Outer radius of outer conductor (shield)
HGP	m	RD	Distance from shield to SPICE ground
RHO	ohm· m	17E-9	Resistivity of conductor material – defaults to value for copper
RHOB	ohm· m	rho	Resistivity of shield material
SIG	mho/m	0.0	Conductivity of dielectric
KD	—	4.0	Relative dielectric constant of dielectric
CMULT	—	1.0	Multiplier (used in defining CPR) for dielectric constant of material between shield and SPICE ground when LLEV=1 and CEXT is not present

Table 37 Lossy U Model Parameters (Continued)

Name (Alias)	Units	Default	Description
CEXT	F/m	und.	External capacitance from shield to SPICE ground – used only to override HGP and CMULT computation
SHTHK	m	2.54E-4	Coaxial shield conductor thickness

Lossy U Model Parameters Geometric Twinlead (PLEV=3, ELEV=1)

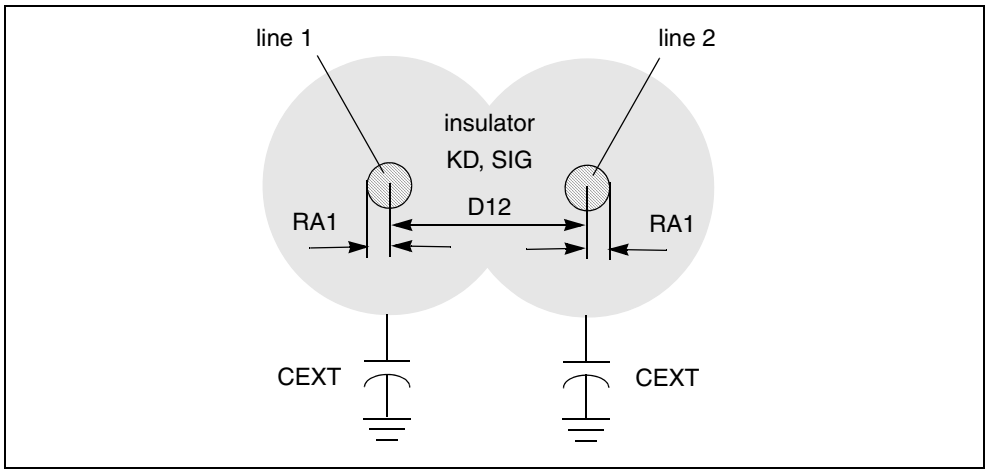


Figure 86 Geometric Embedded Twinlead, DLEV=0, Sea of Dielectric

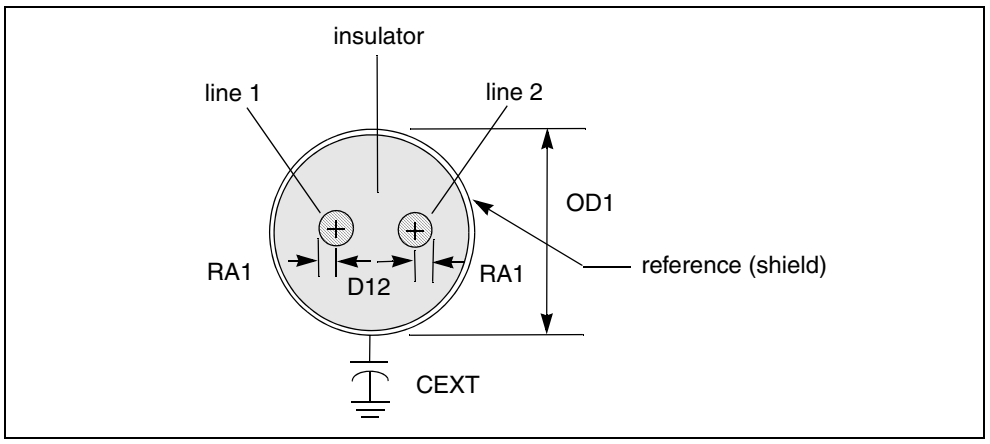


Figure 87 Geometric Twinlead, DLEV=1 with Insulating Spacer

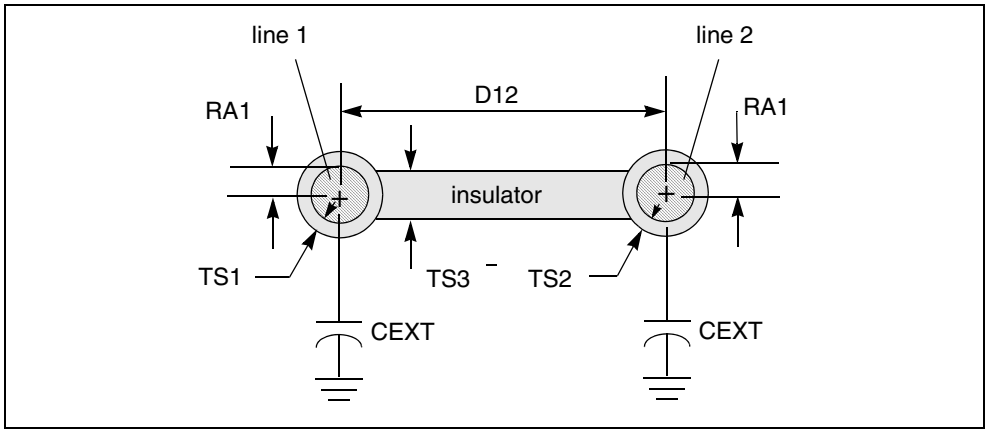


Figure 88 Geometric Twinlead, DLEV=2, Shielded

Table 38 Geometric Twinlead Parameters (ELEV=1)

Name (Alias)	Units	Default	Description
DLEV	—	0.0	0: embedded twinlead 1: spacer twinlead 2: shielded twinlead
RA1	m	req.	Outer radius of each conductor
D12	m	req.	Distance between the conductor centers

Table 38 Geometric Twinlead Parameters (ELEV=1) (Continued)

Name (Alias)	Units	Default	Description
RHO	ohm· m	17E-9	Resistivity of first conductor material – defaults to value for copper
KD		4.0	Relative dielectric constant of dielectric
SIG	mho/m	0.0	Conductivity of dielectric
HGP	m	d12	Distance to reference plane
CMULT	—	1.0	Multiplier {used in defining CPR} for dielectric constant of material between reference plane and SPICE ground when LLEV=1 and CEXT is not present.
CEXT	F/m	undef.	External capacitance from reference plane to SPICE ground point (overrides LRR when present)
TS1	m	req.	Insulation thickness on first conductor
TS2	m	TS1	Insulation thickness on second conductor
TS3	m	TS1	Insulation thickness of spacer between conductor
The following parameters apply to shielded twinlead:			
RHOB	ohm· m	rho	Resistivity of shield material (if present)
OD1	m	req.	Maximum outer dimension of shield
SHTHK	m	2.54E-4	Twinlead shield conductor thickness

The following sections discuss these topics:

- Precomputed Model Parameters (ELEV=2)
- Conductor Width Relative to Reference Plane Width
- Alternative Multi-conductor Capacitance/Conductance Definitions
- Measured Parameters (ELEV=3)

Precomputed Model Parameters (ELEV=2)

Precomputed parameters allow the specification of up to five signal conductors and a reference conductor. These parameters may be extracted from a field solver, laboratory experiments, or packaging specifications supplied by vendors. The parameters supplied include:

- Capacitance/length. Each conductor has a capacitance to all other conductors.
- Conductance/length.Each conductor has a conductance to all other conductors due to dielectric leakage.
- Inductance/length. Each conductor has a self inductance and mutual inductances to all other conductors in the transmission line.
- Resistance/length. Each conductor has two resistances, high frequency resistance due to skin effect and bent wires and DC core resistance.

Figure 89 on page 295 identifies the precomputed components for a three-conductor line with a reference plane. The names for the resistance, capacitance, and conductance components for up to five lines are shown in Figure 90 on page 296.

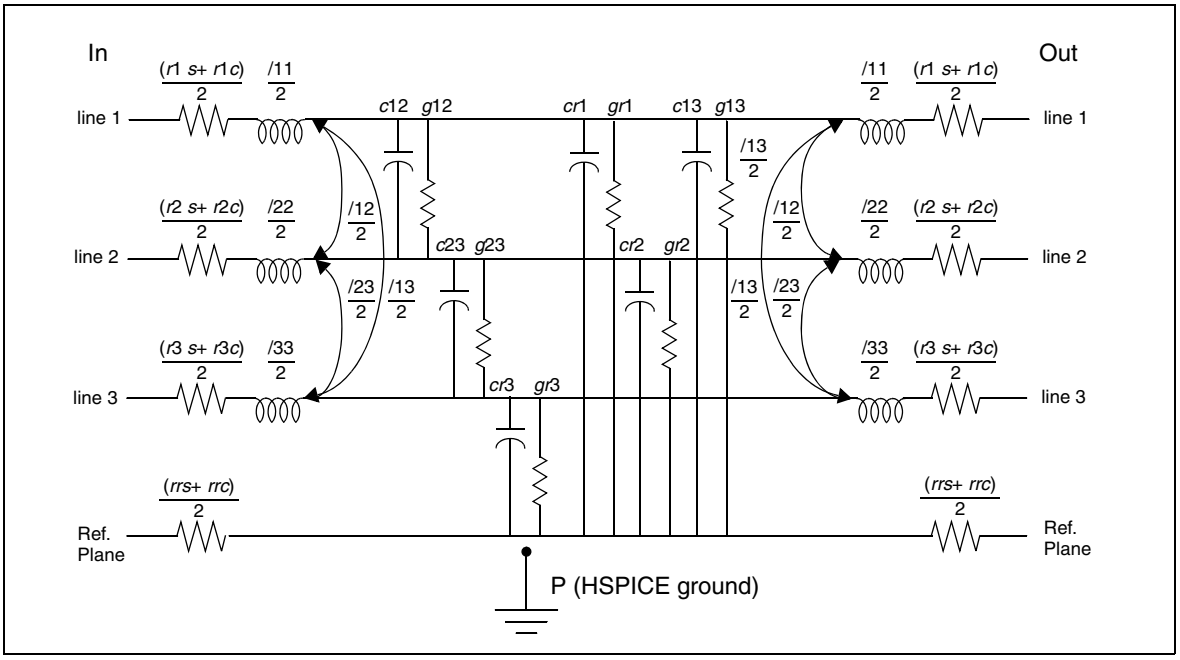


Figure 89 Precomputed Components for 3 Conductors and a Reference Plane

	Ref. plane	line 1	line 2	line 3	line 4	line 5	
HSPICE ground	CPR GPR	CP1 GP1	CP2 GP2	CP3 GP3	CP4 GP4	CP5 GP5	
Ref. plane	RRR LRR	CR1 GR1 LR1	CR2 GR2 LR2	CR3 GR3 LR3	CR4 GR4 LR4	CR5 GR5 LR5	
line 1		R11 L11	C12 G12 L12	C13 G13 L13	C14 G14 L14	C15 G15 L15	LLEV=1 parameter only
line 2			R22 L22	C23 G23 L23	C24 G24 L24	C25 G25 L25	
line 3				R33 L33	C34 G34 L34	C35 G35 L35	
line 4					R44 L44	C45 G45 L45	
line 5						R55 L55	

Figure 90 ELEV=2 Model Keywords for Conductor PLEV=1

All precomputed parameters default to zero except `CEXT`, which is not used unless it is defined. The units are standard MKS in every case, namely:

Table 39 Precomputed Model Parameters

Parameter	**Units**
capacitance	F/m
inductance	H/m
conductance	mho/m
resistance	ohm/m

The following are four additional parameters, LLEV (which defaults to 0), CEXT, and GPR:

- LLEV=0 – reference plane conductor is resistive only (default).
- LLEV=1 – reference plane inductance is included as well as common mode inductance and capacitance to SPICE ground for all conductors.
- CEXT – external capacitance from the reference plane to SPICE ground. When CEXT is specified, it overrides CPR.
- GPR – conductance to circuit ground; is zero except for immersion in a conductive medium.

Conductor Width Relative to Reference Plane Width

For the precomputed lossy U model (ELEV=2), the conductor width must be smaller than the reference plane width, which makes the conductor inductance smaller than the reference plane inductance. If the reference plane inductance is greater than the conductor inductance, simulation reports an error.

Alternative Multi-conductor Capacitance/Conductance Definitions

Three different definitions of capacitances and conductances between multiple conductors are currently used. In this manual, relationships are written explicitly only for various capacitance formulations, but they apply equally well to corresponding conductance quantities, which are electrically in parallel with the capacitances. The symbols used in this section, and where you are likely to encounter these usages, are:

- CXY: branch capacitances, input, and circuit models.
- Cjk: Maxwell matrices for capacitance, multiple capacitor stamp for MNA (modified nodal admittance) matrix, which is a SPICE and HSPICE internal. Also the output of some field solvers.
- CX: capacitance with all conductors except X grounded. The output of some test equipment.
- GXY, Gjk, GX: conductances corresponding to above capacitances

The following example uses a multiple conductor capacitance model, a typical U model transmission line. The U -element supports up to five signal conductors plus a reference plane, but the three conductor case, Figure 91 on page 298, demonstrates the three definitions of capacitance. The branch capacitances are specified in HSPICE notation.

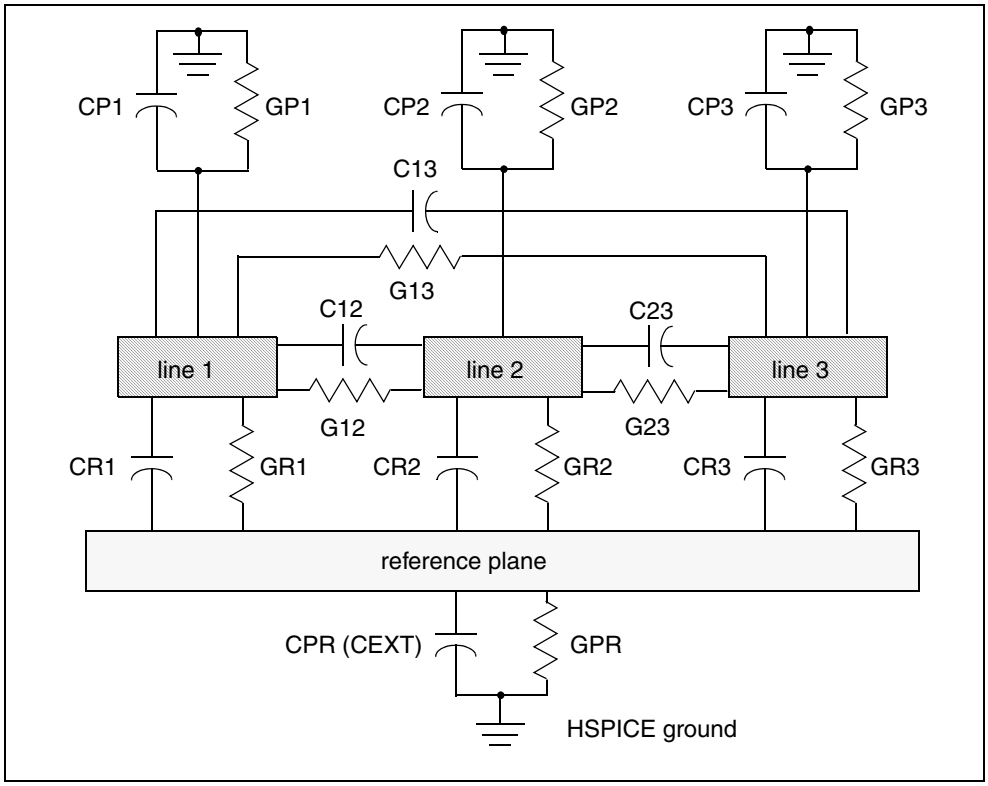


Figure 91 Single-Lump Circuit Capacitance

The branch and Maxwell matrices are completely derivable from each other. The “O.C.G.” (“other conductors grounded”) matrix is derivable from either the Maxwell matrix or the branch matrix. Thus:

$$CX = \sum_{X \neq Y} CXY$$

Equation 68

`Cjk` = `CX` on diagonal

= `-CXY` off diagonal

The matrices for the example given above provide the following “O.C.G.” capacitances:

```
C1 = CR1 + C12 + C13
C2 = CR2 + C12 + C23
C3 = CR3 + C13 + C23
CR = CR1 + CR2 + CR3 + CPR
```

Also, the Maxwell matrix is specified as:

Equation 69

$$Cjk = \begin{bmatrix} CR & -CR1 & -CR2 & -CR3 \\ -CR1 & C1 & -C12 & -C13 \\ -CR2 & -C12 & C2 & -C23 \\ -CR3 & -C13 & -C23 & C3 \end{bmatrix}$$

The branch capacitances also may be obtained from the Maxwell matrices. The off-diagonal terms are the negative of the corresponding Maxwell matrix component. The branch matrix terms for capacitance to circuit ground are the sum of all the terms in the full column of the maxwell matrix with signs intact:

```
CPR = sum (Cjk), j=R, k=R:3
CP1 = sum (C1k), j=1, k=R:3
CP2 = sum (C2k), j=2, k=R:3
CP3 = sum (C3k), j=3, k=R:3
```

CP1, ... CP5 are not computed internally with the geometric (`ELEV=1`) option, although CPR is. This, and the internally computed inductances, are consistent with an implicit assumption that the signal conductors are completely shielded by the reference plane conductor. This is true, to a high degree of accuracy for stripline, coaxial cable, and shielded twinlead, and to a fair degree for MICROSTRIP. If accurate values of CP1 and so forth are available from a field solver, they can be used with `ELEV=2` type input.

If the currents from each of the other conductors can be measured separately, then all of the terms in the Maxwell matrix may be obtained by laboratory experiment. By setting all voltages except that on the first signal conductor equal to 0, for instance, you can obtain all of the Maxwell matrix terms in column 1.

Equation 70

$$\begin{bmatrix} IR \\ I1 \\ I2 \\ I3 \end{bmatrix} = \begin{bmatrix} CR & -CR1 & -CR2 & -CR3 \\ -CR1 & C1 & -C12 & -C13 \\ -CR2 & -C12 & C2 & -C23 \\ -CR3 & -C13 & -C23 & C3 \end{bmatrix} \cdot jw \cdot \begin{bmatrix} 0.0 \\ 1.0 \\ 0.0 \\ 0.0 \end{bmatrix} = jw \cdot \begin{bmatrix} -CR1 \\ C1 \\ -C12 \\ -C13 \end{bmatrix}$$

The advantage of using branch capacitances for input derives is that only one side of the off-diagonal matrix terms are input. This makes the input less tedious and provides fewer opportunities for error.

Measured Parameters (ELEV=3)

When measured parameters are specified in the input, the program calculates the resistance, capacitance, and inductance parameters using TEM transmission line theory with the `LLEV=0` option. If redundant measured parameters are given, the program recognizes the situation, and discards those which are usually presumed to be less accurate. For twinlead models, `PLEV=3`, the common mode capacitance is one thousandth of that for differential-mode, which allows a reference plane to be used.

The `ELEV=3` model is limited to one conductor and reference plane for `PLEV=1`.

Table 40 Measured Parameters

Name (Alias)	Units	Default	Description
PLEV			1: planar 2: coax 3: twinlead
ZK	ohm	calc	Characteristic impedance
VREL	—	calc	Relative velocity of propagation (delen / (delay · clight))
DELAY	sec	calc	Delay for length delen
CAPL		1.0	Linear capacitance in length clen
AT1		1.0	Attenuation factor in length atlen. Use dB scale factor when specifying attenuation in dB.
DELEN	m	1.0	Unit of length for delay (for example, ft.)
CLEN	m	1.0	Unit of length for capacitance
ATLEN	m	1.0	Unit of length for attenuation
FR1	Hz	req.	Frequency at which AT1 is valid. Resistance is constant below FR1, and increases as $\sqrt{}$(frequency) above FR1.

Parameter Combinations

You can use several combinations of measured parameters to compute the L and C values used internally. The full parameter set is redundant. If you input a

redundant parameter set, the program discards those that are presumed to be less accurate. Table 42 on page 301 shows how each of seven possible parameter combinations are reduced, if need be, to a unique set and then used to compute C and L.

Three different delays are used in discussing transmission lines:

Table 41 Delays Used with Transmission Lines

Delay	Description
DELAY	U Model input parameter that is the delay required to propagate a distance “dlen”
TD	T-element input parameter signifying the delay required to propagate one meter
TDeff	Internal variable, which is the delay required to propagate the length of the transmission line T -element or U -element.

Table 42 Lossless Parameter Combinations

Input Parameters	Basis of Computation
ZK, DELAY, DELEN, CAPL, CLEN	Redundant. Discard CAPL and CLEN.
ZK, VREL, CAPL, CLEN	Redundant. Discard CAPL and CLEN.
ZK, DELAY, DELEN	VREL=DELEN/(DELAY · CLIGHT)
ZK and VREL	C=1/(ZK · VREL · CLIGHT) L=ZK/(VREL · CLIGHT)
ZK, CAPL, CLEN	C=CAPL/CLEN L=C · ZK^2
CAPL, CLEN, DELAY, DELEN	VREL=DELEN/(DELAY · CLIGHT)
CAPL, CLEN, VREL	LC=CAPL/CLEN LL=1/(C · $VREL^2$ · $CLIGHT^2$)

Loss Factor Input

You can specify the attenuation per unit length as either an attenuation factor or a decibel attenuation. Because the data might be available either as input/output or output/input, decibels greater than 0, or factors greater than 1 are assumed to be input/output. The following example shows the four ways that one may specify that an input of 1.0 is attenuated to an output of 0.758.

Table 43 Input Attenuation Variations

AT1 Input	Computation of attenuation factor and linear resistance
AT1 = –2.4dB	v(out)/v(in) = 0.758 = $10^{(+AT1/20)}$ (for dB < 0)
AT1 = +2.4dB	v(out)/v(in) = 0.758 = $10^{(-AT1/20)}$ (for dB > 0)
AT1 = 1.318	v(out)/v(in) = 0.758 = 1/AT1 (for ATl < 1)
AT1 = 0.758	v(out)/v(in) = 0.758 = AT1 (for ATl > 1)

The attenuation factor is used to compute the exponential loss parameter and linear resistance.

Equation 71
$$\alpha = \frac{ln((v(in))/(v(out)))}{ATlin}$$

Equation 72
$$LR = 2 \cdot \alpha \cdot \sqrt{(LL)/(LC)}$$

U-element Examples, Models, and Applications

The following examples show the results of simulating a stripline geometry using the U Model in a PCB scale application and in an IC scale application.

The following sections discuss these topics:

- Three Coupled Lines, Stripline Configuration
- Three Coupled Lines, Sea of Dielectric Configuration
- Simulation Output
- IcWire Output Section

- Capacitance and Inductance Matrices
- Five Coupled Lines, Stripline Configuration
- U Model Applications
- Solving Ringing Problems with U-elements

Three Coupled Lines, Stripline Configuration

Figure 92 on page 303 shows three coupled lines in a stripline configuration on an FR4 printed circuit board. A simple circuit using three coupled striplines is shown in Figure 93 on page 303.

This example is based on demonstration netlist stripline.sp, which is available in directory $<installdir>/demo/hspice/tline:

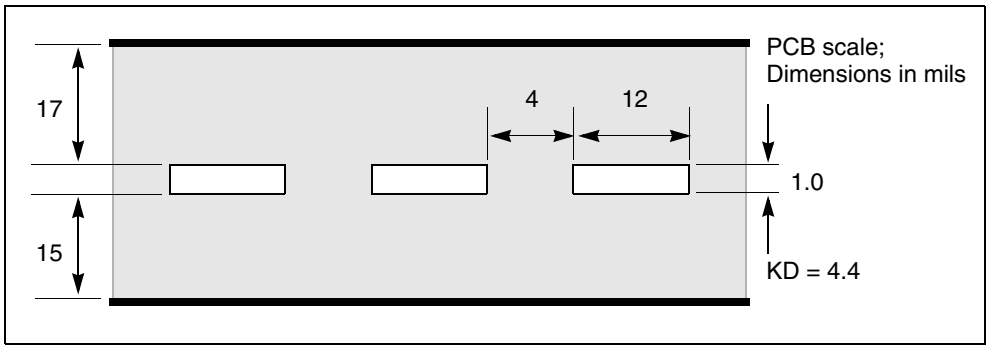


Figure 92 Three Coupled Striplines (PCB Scale)

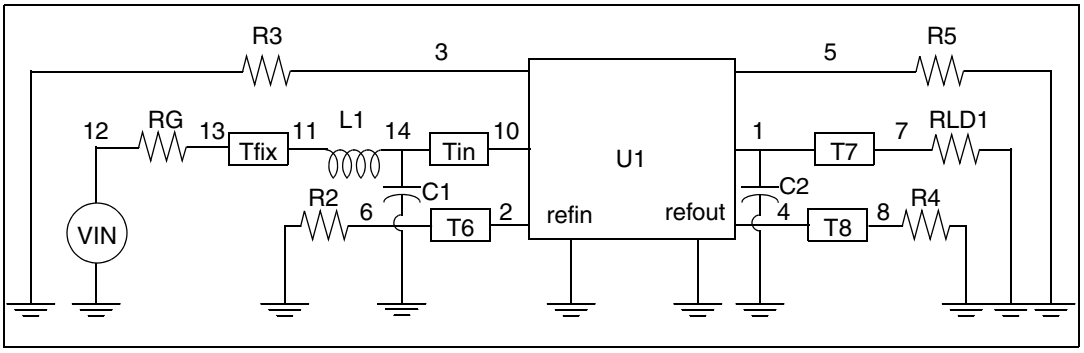


Figure 93 Schematic Using the Three Coupled Striplines U Model

```
* Stripline circuit
.Tran 50ps 7.5ns
.OPTION Post NoMod Accurate Probe Method=Gear
VIN 12 0 PWL 0 0v 250ps 0v 350ps 2v
L1 14 11 2.5n
C1 14 0 2p
Tin 14 0 10 0 ZO=50 TD=0.17ns
Tfix 13 0 11 0 ZO=45 TD=500ps
RG 12 13 50
RLD1 7 0 50
C2 1 0 2p
U1 3 10 2 0 5 1 4 0 USTRIP L=0.178
T6 2 0 6 0 ZO=50 TD=0.17ns
T7 1 0 7 0 ZO=50 TD=0.17ns
T8 4 0 8 0 ZO=50 TD=0.17ns
R2 6 0 50
R3 3 0 50
R4 8 0 50
R5 5 0 50
.Model USTRIP U LEVEL=3 PLev=1 Elev=1 Dlev=2 Nl=3 Ht=381u
+ Wd=305u Th=25u Sp=102u Ts=838u Kd=4.7
.Probe v(13) v(7) v(8) v(6)
.End
```

Figure 94, Figure 95, and Figure 96 show the main line and crosstalk responses. The rise time and delay of the waveform are sensitive to the skin effect frequency, since losses reduce the slope of the signal rise. The main line response shows some differences between simulation and measurement. The rise time differences are due to layout parasitics and the fixed resistance model of skin effect. The differences between measured and simulated delays are due to errors in the estimation of dielectric constant and the probe position.

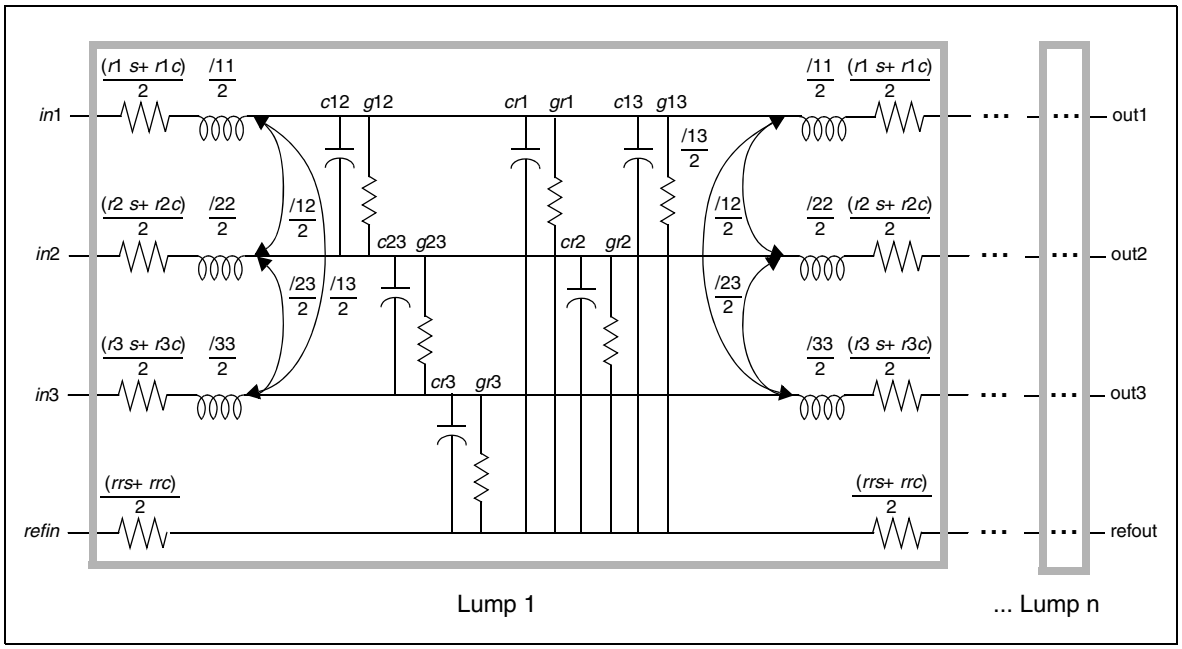


Figure 94 Measured versus Computed Through-Line Response

The gradual rise in response between 3 ns and 4 ns is due to skin effect. During this period, the electric field driving the current penetrates farther into the conductor so that the current flow increases slightly and gradually. This affects the measured response as shown for the period between 3 ns and 4 ns.

Figure 95 shows the backward crosstalk response. The amplitude and delay of this backward crosstalk are very close to the measured values. The risetime differences are due to approximating the skin effect with a fixed resistor, while the peak level difference is due to errors in the LC matrix solution for the coupled lines.

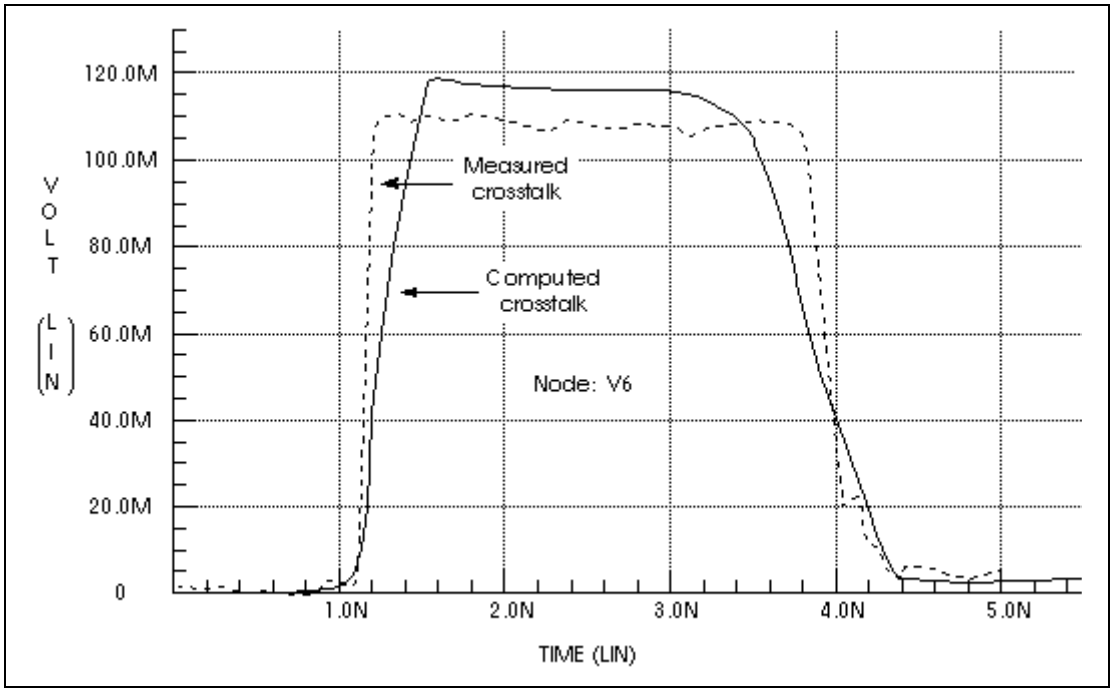


Figure 95 Measured Versus Computed Backward Crosstalk Response

Figure 96 on page 307 shows the forward crosstalk response. This forward crosstalk shows almost complete signal cancellation in both measurement and simulation. The forward crosstalk levels are about one tenth the backward crosstalk levels. The onset of ringing of the forward crosstalk has reasonable agreement between simulation and measurement. However, the trailing edge of the measured and simulated responses differ. The measured response trails off to zero after about 3 ns, while the simulated response does not trail down to zero until 6 ns. Errors in simulation at this voltage level can easily be due to board layout parasitics that have not been included in the simulation.

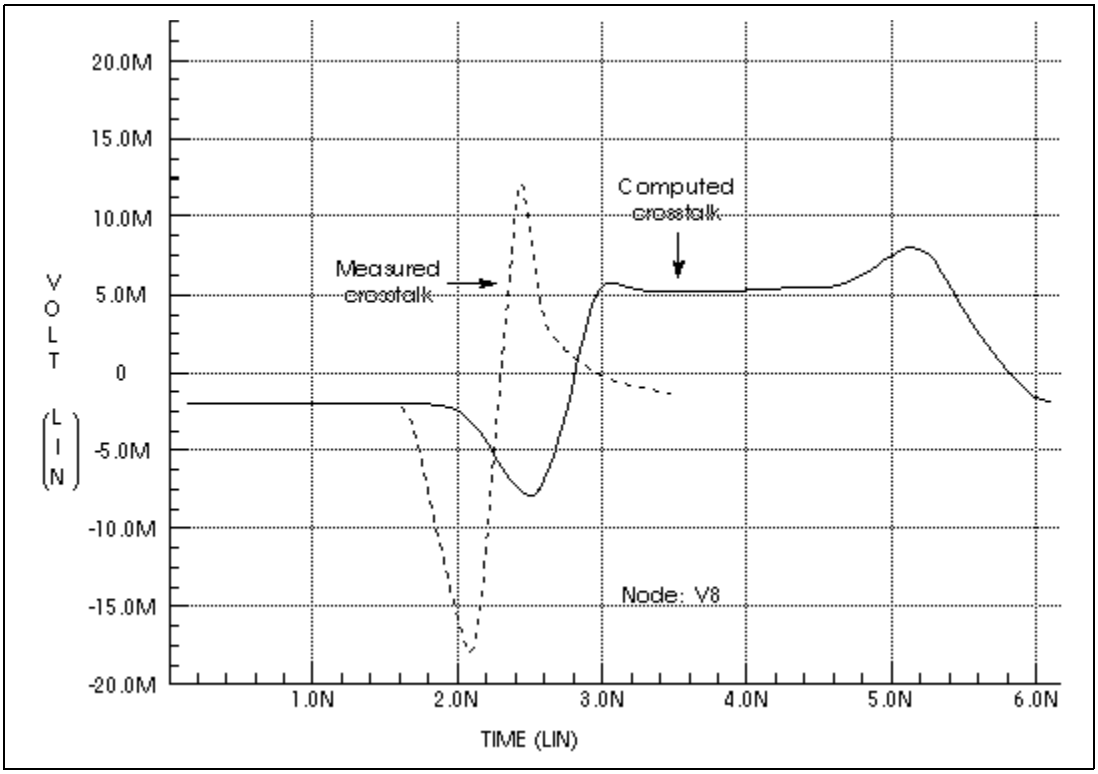


Figure 96 Measured Versus Computed Forward Crosstalk Response

Simulation methods can have a significant effect on the predicted waveforms. Figure 97 on page 308 shows the main line response at Node 7 of Figure 93 on page 303 as the integration method and the number of lumped elements change. With the recommended number of lumps, 20, the Trapezoidal integration method shows a fast risetime with ringing.

The Gear integration method shows a fast risetime and a well damped response. When the number of lumped elements changes to 3, both Trapezoidal and Gear methods show a slow risetime with ringing. In this situation, the Gear method with 20 lumps gives the more accurate simulation.

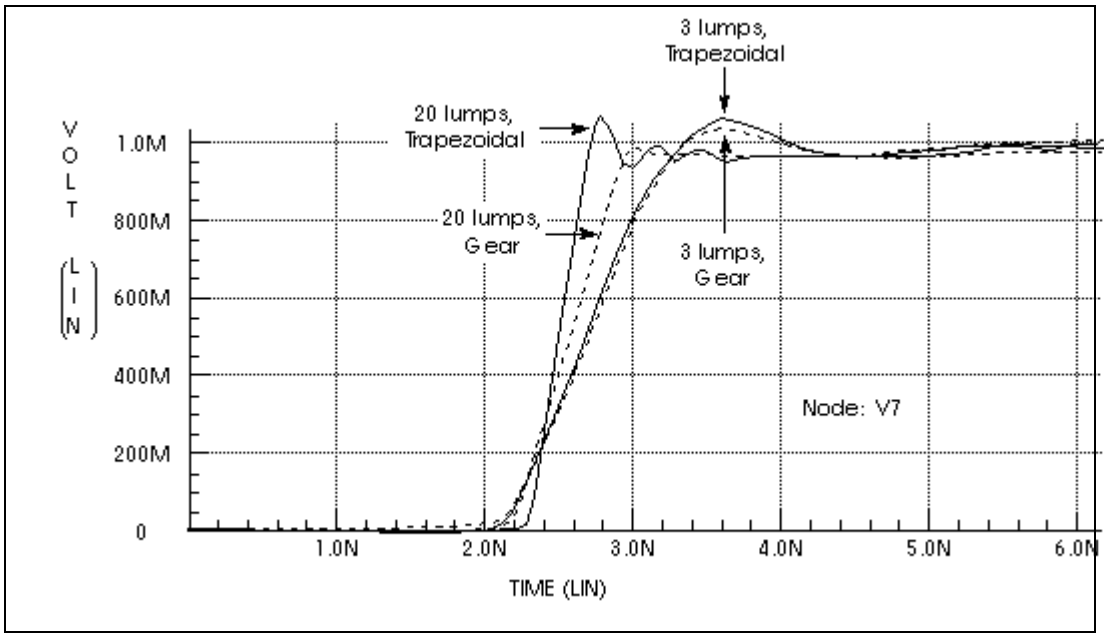


Figure 97 Computed Responses for 20 Lumps and 3 Lumps, Gear and Trapezoidal Integration Methods

Three Coupled Lines, Sea of Dielectric Configuration

This example shows the U -element analytic equations for a typical integrated circuit transmission line application. Three 200µm-long aluminum wires in a silicon dioxide dielectric are simulated to examine the through-line and coupled line response.

The U model uses the transmission line geometric parameters to generate a multisection lumped-parameter transmission line model. Use a single U -element statement to create an internal network of three 20-lump circuits.

Figure 98 shows the IC-scale coupled line geometry.

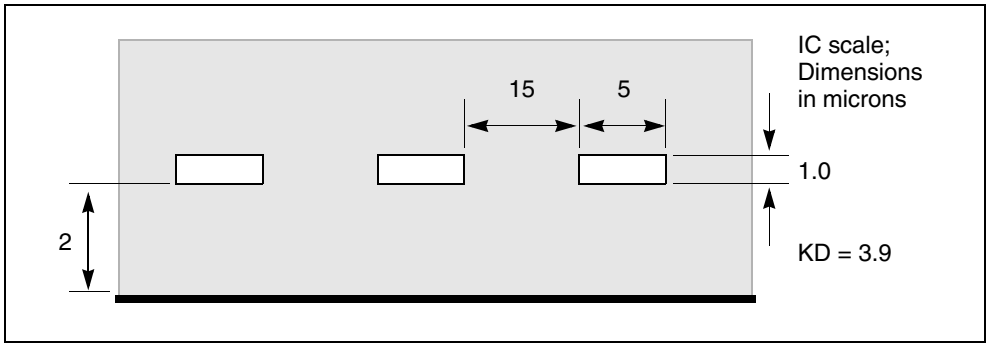


Figure 98 Three Coupled Lines with One Reference Plane in a Sea of Dielectric (IC Scale)

Figure 99 shows one lump of the lumped-parameter schematic for the three-conductor stripline configuration of Figure 98. This internal circuitry represents one U-element instantiation. Internal elements are described in Simulation Output on page 313

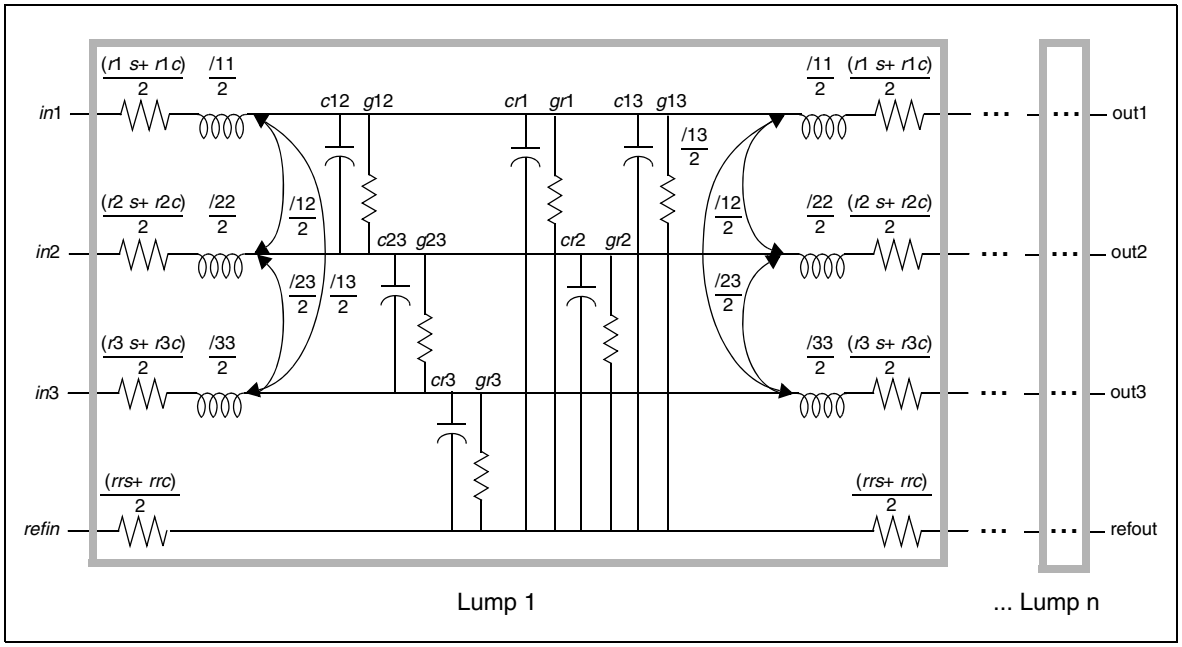


Figure 99 Schematic for Three Coupled Lines with One Reference Plane

Figure 100 on page 310 shows a schematic using the U-element of Figure 99. In this simple circuit, a pulse drives a three-conductor transmission line source terminated by 50Ω resistors and loaded by 1pF capacitors.

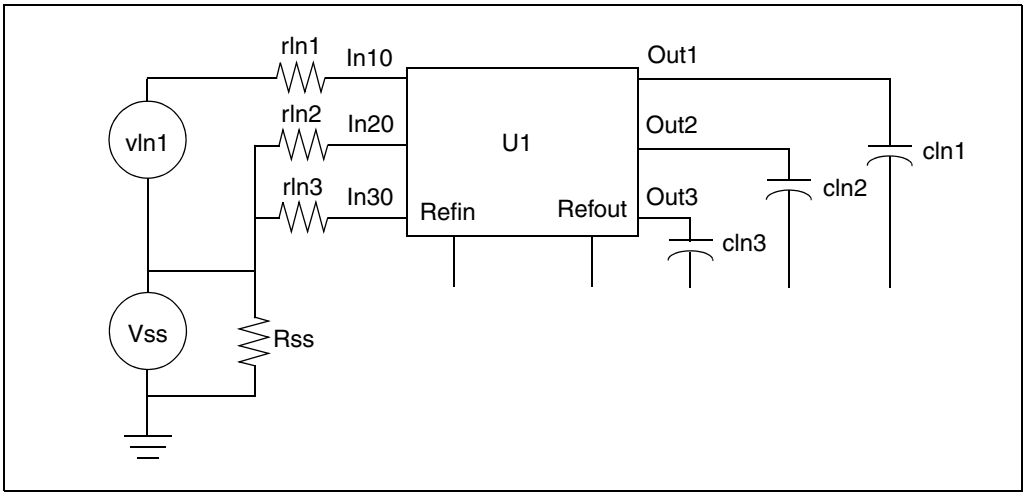


Figure 100 Schematic Using the Three Coupled Lines U Model

The input file for the U-element solution is shown below.

This example is based on demonstration netlist uele.sp, which is available in directory $*installdir*/demo/hspice/tline:

```
*The input file for the U Element solution
.Tran 0.1ns 20ns
.OPTION Post Accurate NoMod Brief Probe
Vss Vss 0 0v
Rss Vss 0 1x
vIn1 In1 Vss Pwl 0ns 0v 11ns 0v 12ns 5v 15ns 5v 16ns 0v
rIn1 In1 In10 50
rIn2 Vss In20 50
rIN3 Vss In30 50
u1 In10 In20 In30 Vss Out1 Out2 Out3 Vss IcWire L=200um
cIn1 Out1 Vss 1pF
cIn2 Out2 Vss 1pF
cIn3 Out3 Vss 1pF
.Probe v(Out1) v(Out2) v(Out3)
.Model IcWire U LEVEL=3 Dlev=0 Nl=3 Nlay=2 Plev=1 Elev=1
+ Llev=0 Ht=2u Wd=5u Sp=15u Th=1u Rho=2.8e-8 Kd=3.9
.End
```

The U-element uses the conductor geometry to create length-independent RLC matrices for a set of transmission lines. You can then input any length, and simulation computes the number of circuit lumps that are required.

Figure 101, Figure 102 and Figure 103 show the through and coupled responses, computed using U-element equations.

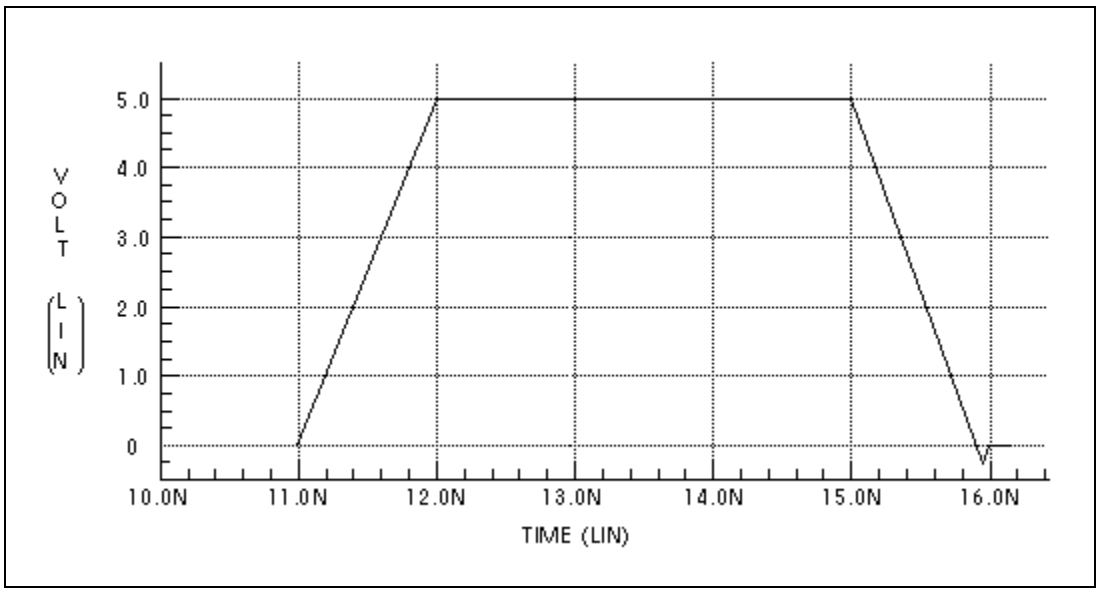


Figure 101 Computed Through-Line Response

Figure 102 shows the nearest coupled line response. This response occurs only during signal transitions.

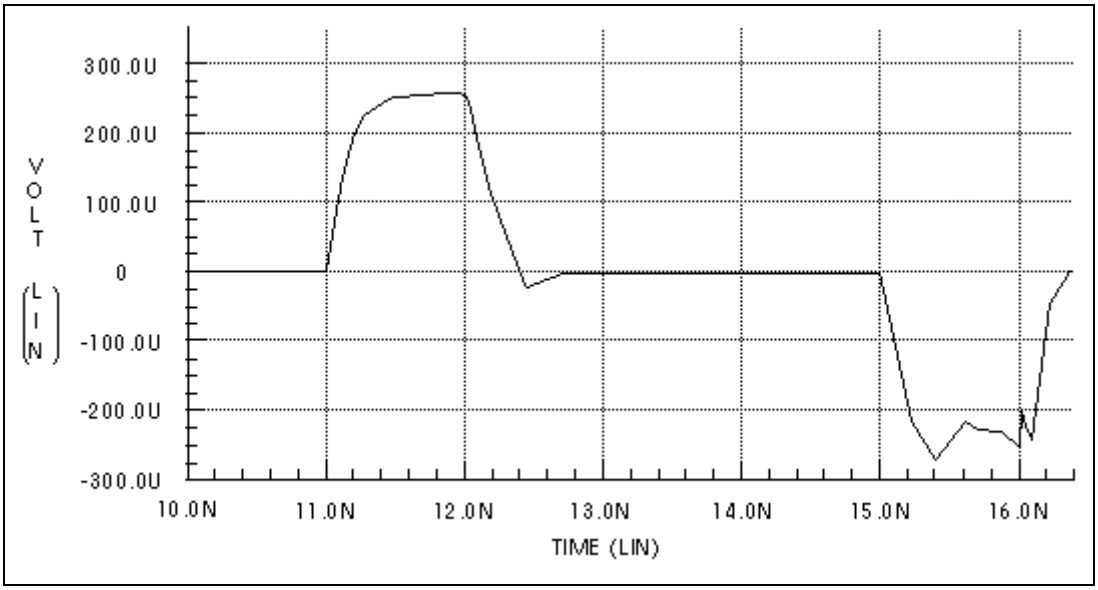


Figure 102 Computed Nearest Coupled-Line Response

Figure 103 on page 312 shows the third coupled line response. The predicted response is about 1/100000 of the main line response.

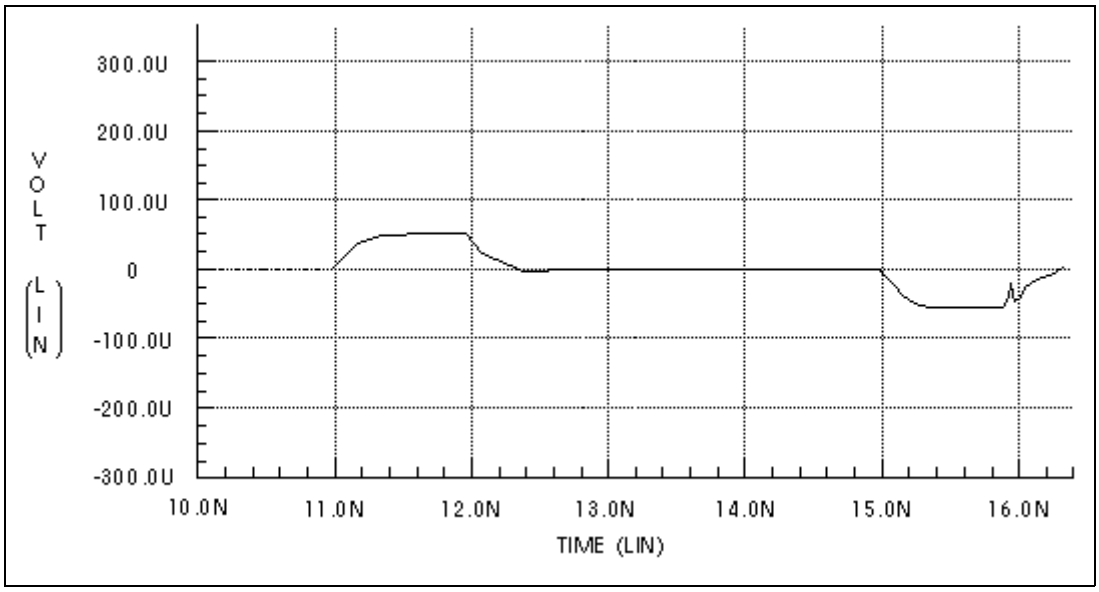


Figure 103 Computed Furthest Coupled-Line Response

By default, simulation prints the model values, including the LCRG matrices for the U-element. All of the printed `LCRG` parameters are identified in the following section.

Simulation Output

The listing below is part of the output from a simulation using the HSPICE input deck for the IcWire U model. Descriptions of the parameters specific to U - elements follows the listing. (Parameters not listed in this section are described in Table 33 on page 280 and Table 34 on page 282.)

IcWire Output Section

```
*** model name:    0:icwire           ****
names   values   units   names   values  units   names   values   units
--- u-model control parameters ---
maxl=   20.00    #lumps  wlump=  20.00           elev=   1.00
plev=   1.00             llev=   0.              nlay=   2.00     #layrs
nl=     3.00     #lines  nb=     1.00    #refpl
--- begin type specific parameters ---
dlev=   0.       #       kd=     3.90            corkd=  1.00
sig=    0.       mho/m   ho=     28.00   n ohm*m rhob=   28.00    n ohm*m
th=     1.00u    meter   thb=    1.42u   meter   skin=   10.31u   meter
skinb=  10.31u   meter   wd2     5.00u   meter   ht2     2.00u    meter
th2     1.00u    meter   sp1     15.00u  meter   wd3     5.00u    meter
ht3     2.00u    meter   th3     1.00u   meter   sp2     15.00u   meter
cr1=    170.28p  f/m     gr1=    0.      mho/m   l11=    246.48n  h/m
cr2=    168.25p  f/m     gr2=    0.      ee      c12=    5.02p    f/m
g12=    0.       mho/m   l12=    6.97n   h/m     l22=    243.44n  h/m
cr3=    170.28p  f/m     gr3=    0.      mho/m   c13=    649.53f  f/m
g13=    0.       mho/m   l13=    1.11n   h/m     c23=    5.02p    f/m
g23=    0.       mho/m   l23=    6.97n   h/m     l33=    246.48n  h/m
--- two layer (skin and core) parameters ---
rrs=    1.12k    ohm/m   rrc=    0.      ohm/m   r1s=    5.60k    ohm/m
r1c=    0.       ohm/m   r2s=    5.60k   ohm/m   r2c=    0.       ohm/m
r3s=    5.60k    ohm/m   r3c=    0.      ohm/m
```

Table 44 U-element Parameters

Parameter	Description
cij	Coupling capacitance from conductor i to conductor j (positive)
crj	Self-capacitance/m of conductor j to the reference plane
cpr	Capacitance of the reference plane to the HSPICE ground
gij	Conductance/m from conductor i to conductor j (zero if sig=0)
grj	Conductance/m from conductor j to the reference plane (0 if sig=0)
gpr	Conductance from the reference plane to the HSPICE ground, always=0

Table 44 U-element Parameters (Continued)

Parameter	Description
hti	Conductor i height above reference plane (only ht is input; all heights same)
lri	Inductance/m from conductor i to the reference plane
lrr	Inductance/m of the reference plane
lij	Inductance/m from conductor i to conductor j
ljj	Self-inductance/m of conductor j
rrc	Reference plane core resistance/m (if NLAY = 2, zero of skin depth > 90% of thb)
rrr	Resistance/m of the reference plane (if NLAY =1)
rrs	Skin resistance/m of the reference plane (if NLAY = 2)
ris	Skin resistance/m of conductor i (if NLAY = 2)
ric	Conductor i core resistance/m (if NLAY = 2, zero if skin depth > 50% of th)
rjj	Resistance/m of conductor j (if NLAY = 1)
skin	Skin depth
skinb	Skin depth of the reference plane
spi	Spacing between conductor i and conductor i +1 (only sp is input; same spacings)
thi	Thickness of conductor i (only th is input; all thicknesses are the same)
wdi	Width of conductor i (only wd is input–all widths are the same)

The total conductor resistance is indicated by rjj when `NLAY=1`, or by ris + ric when `NLAY=2`.

As shown in the next section, some difference between HSPICE and field solver results is to be expected. Within the range of validity shown in Table 35 on page 283 for the U model, simulation comes very close to field solver accuracy. In fact, discrepancies between results from different field solvers can be as large as their discrepancies with simulation. The next section compares some Synopsys physical circuit models to models derived using field solvers.

Capacitance and Inductance Matrices

Simulation places capacitance and inductance values for U -elements in matrix form, for example:

$$\begin{bmatrix} Cii & \dots & Cji \\ . & & \\ . & & \\ . & & \\ Cij & & Cjj \end{bmatrix}$$

Equation 73

Table 45 shows the capacitance and inductance matrices for the three-line, buried microstrip IC-scale example shown in Figure 98 on page 309.

Table 45 Capacitance and Inductance Matrices for the Three-Line, IC-Scale Interconnect System

Capacitance (pF/m)	176 -5.02 -0.65 -5.02 178 -5.02 -0.65 -5.02 176
Inductance (nH/m)	246 6.97 1.11 6.97 243 6.97 1.11 6.97 246

The capacitance matrices in Table 45 are based on the admittance matrix of the capacitances between the conductors. The negative values in the capacitance matrix are due to the sign convention for admittance matrices. The inductance matrices are based on the impedance matrices of the self and mutual inductance of the conductors. Each matrix value is per meter of conductor length. The actual lumped values use a conductor length equal to the total line length divided by the number of lumps.

The above capacitance matrix can be related directly to the output of the IC-scale example. Simulation uses the branch capacitance matrix for internal calculations. For the three-conductors in this example, Figure 104 on page 317 shows the equivalent capacitances in terms of simulation device model parameters.

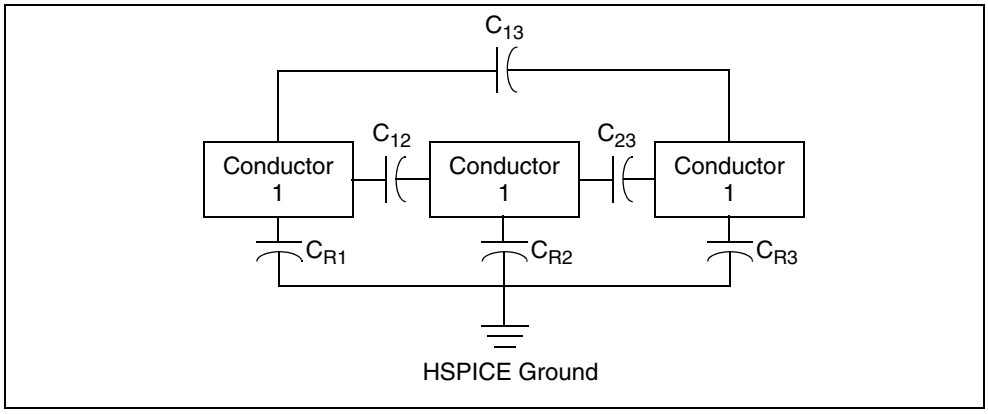


Figure 104 Conductor Capacitances for IC-scale Example

The capacitances of Figure 104 are those shown in Figure 99 on page 309. The HSPICE nodal capacitance matrix of Table 45 is shown below by using the capacitance terms that are listed in the HSPICE output.

$$\begin{bmatrix} C_{R1} + C_{12} + C_{13} & -C_{12} & -C_{13} \\ -C_{12} & C_{R2} + C_{12} + C_{23} & -C_{23} \\ -C_{13} & -C_{23} & C_{R3} + C_{13} + C_{23} \end{bmatrix}$$

Off-diagonal terms are the negative of coupling capacitances (to conform to the sign convention).

The diagonal terms require some computation, for example:

$$C_{11} = C_{R1} + C_{12} + C_{13}$$
$$= 170.28 + 5.02 + 0.65$$
$$= 175.95, \text{ or } 176 \text{ pF/m}$$

Note that the matrix values on the diagonal in Table 45 on page 316 are large: they indicate self-capacitance and inductance. The diagonal values show close agreement among the various solution methods. As the coupling values become small compared to the diagonal values, the various solution methods give very different results. Third-line coupling capacitances of 0.65 pF/m, 1.2 pF/m, and 0.88 pF/m are shown in Table 45 on page 316. Although the differences between these coupling capacitances seem large, they represent a negligible difference in waveforms because they account for only a very small

amount of voltage coupling. Table 45 represents very small coupling because the line spacing is large (about seven substrate heights).

Table 46 shows the parameters for the stripline in Table 105.

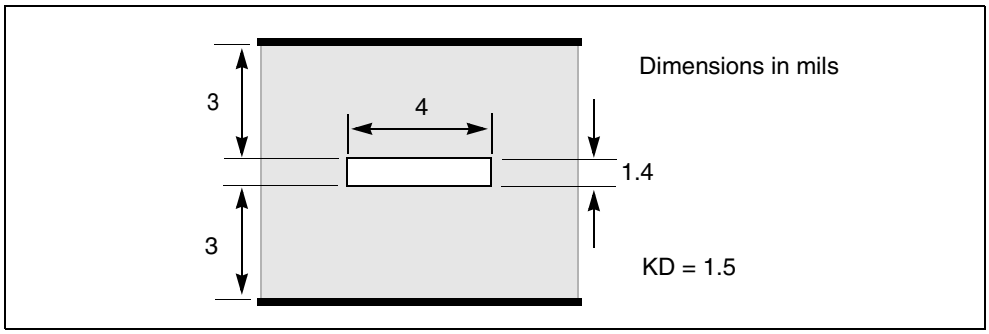


Figure 105 Stripline Geometry Used in MCM Technology

Table 46 Capacitance and Inductance for the Single Line MCM-Scale Stripline

Capacitance (pF/m)	164.4
Inductance (nH/m)	236.5

Five Coupled Lines, Stripline Configuration

This example shows a five-line interconnect system in a PCB technology. Table 47 on page 319 shows the matrix parameters for the line configuration of Figure 106.

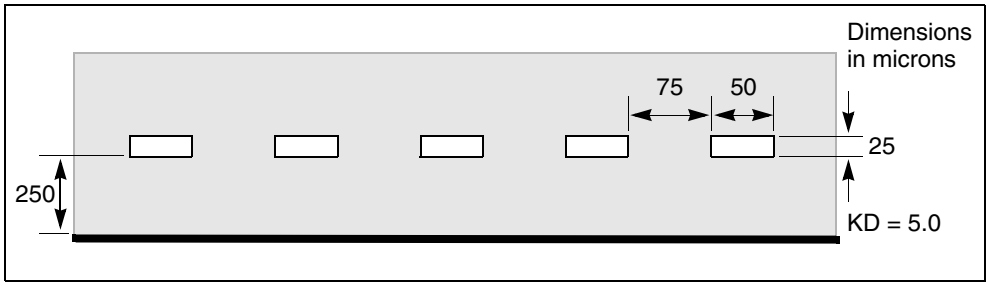


Figure 106 Five Coupled Lines on a PCB

Table 47 Capacitance/Inductance for 5-Line PCB-Scale Interconnect System

Capacitance (pF/m)	59 -19 -2.5 -0.8 -0.4 -19 69 -18 -2.2 -0.8 -2.5 -18 69 -18 -2.5 -0.8 -2.2 -18 69 -19 -0.4 -0.8 -2.5 -19 57
Inductance (nH/m)	676 309 179 116 81 309 641 297 174 116 179 297 637 297 179 116 174 297 641 309 81 116 179 309 676

U Model Applications

This section gives examples of use, and explains some of the aspects of ringing (impulse-initiated oscillation) in real and simulated transmission line circuits.

The following sections discuss these topics:

- Data Entry Examples
- Printed Circuit Board Models
- Via Modeling for PCBs in HSPICE
- Coax Models
- Twinlead Models
- Two Coupled Microstrips

Data Entry Examples

Coax Geometry Entry (`ELEV=1`, `PLEV=2`) with ground reference (`LLEV=1`) and skin effect (`NLAY=2`):

```
uc in1 3 out1 4 wire2 l=1
.model wire2 u LEVEL=3 nlay=2 plev=2 elev=1 Llev=1
+ ra=1m rb=7.22m hgp=20m rho=1.7e-8 kd=2.5
```

Matrix Entry (`ELEV=2`):

```
u1 In1 In2 In3 Vss Out1 Out2 Out3 Vss Wire3 L=0.01
.model Wire3 U LEVEL=3 NL=3 Elev=2 Llev=0
+ rrr=1.12k r11=5.6k r22=5.6k r33=5.6k c13=0.879pF
+ cr1=176.4pF cr2=172.6pF cr3=176.4pF c12=4.7pF c23=4.7pF
+ L11=237nH L22=237nH L33=237nH L12=5.52nH L23=5.52nH
+ L13=1.34nH
```

Coax Measured Data Entry (`ELEV=3`, `PLEV=2`):

```
u10 1 0 2 0 rg58 l=12
.model rg58 u LEVEL=3 plev=2 elev=3
+ zk=50 capl=30.8p clen=1ft vrel=0.66
+ fr1=100meg at1=5.3db atlen=100ft
```

Printed Circuit Board Models

Figure 107 on page 320 illustrates a small cross section of a six-layer printed circuit board. The top and bottom signal layers require a microstrip U model (`DLEV=1`), while the middle signal layers use a stripline U model (`DLEV=2`).

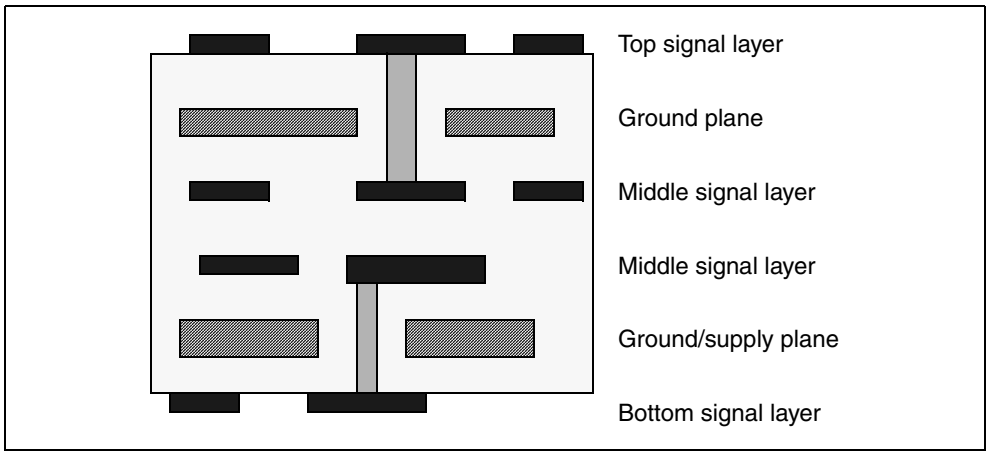


Figure 107 Six-Layer Printed Circuit Board

Important aspects of such a circuit board are:

- Trace impedance is difficult to control because of etch variation
- 6 mil effective trace widths
- 8 mil drawn widths
- 10 mil insulator thickness
- 1 ounce copper 1.3 mil thick

- Microstrip model TOP used for top and bottom
- Stripline model MID used for middle signal layers

Example:

Top and bottom layer model:

```
.MODEL TOP U LEVEL=3 ELEV=1 PLEV=1 TH=1.3mil HT=10mil KD=4.5
+ DLEV=1 WD=8mil XW=-2mil
```

Middle layer model:

```
.MODEL MID U LEVEL=3 ELEV=1 PLEV=1 TH=1.3mil HT=10mil KD=4.5
+ DLEV=2 WD=8mil XW=-2mil TS=32mil
```

Via Modeling for PCBs in HSPICE

You can do modeling of vias for PCBs in pre-layout signal integrity phase of HSPICE simulations. While A 3D field solver is required to accurately model vias, you can use the built-in HSPICE 2D field solver to approximate a via model using the following method:

1. Create a SPICE netlist for a via. The via can be modeled as a lumped pi-model if the via delay is less than 0.1 of the signal edge.

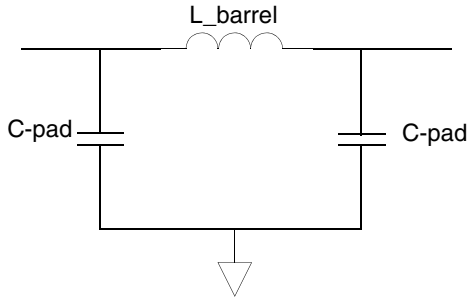


Figure 108 Where: L_barrel is the inductance of the barrel of the via and C_pad is the capacitance of the pad.

2. Use the .LIN command to extract an S-parameter model file in Touchstone format.
3. Use the S-element to use the model file.

Other Considerations

If you are working with a PCB via, note that it can also be modeled as a Pi Lumped Element Model. The pi-model circuit approximates the behavior of short transmission lines (lumped RLC circuits). But the exact values for the

RLC parameters of a via depend on the size and geometry of the via. To determine the exact values, you need to use a 3D field solver, such as the Synopsys Raphael product to extract exact parameter values.

Equivalent RLCs for GHz boards require a 3D EM analysis for accurate modeling, such as Raphael provides to include the magnetic flux in the conductor at DC or for modeling edge effects accurately. If Raphael can produce the RLGC table model, you can simulate with the model in HSPICE and then compare the result with field solvers to evaluate the difference.

Note that modeling IC vias is more complex as you need to consider stress effects (since process-induced mechanical stress can cause electromigration and formation of voids or cracks in vias or metal lines), thermal effects (since vias have much higher thermal conductivity than the dielectric materials (ILD), and modeling via plug resistance, etc.

Coax Models

The following examples are for standard coax. These are obtained from commonly available tables. (The parameter fr1 is the frequency at which a specific amount of attenuation, at1, occurs for a specified length of coax, atlen.) Synopsys simulators accept dB (decibel) and ft. (foot) units.

Example:

```
.model rg9/u    u          LEVEL=3     plev=2       elev=3
+              Zk=51       vrel=.66
+             fr1=100meg at1=2.1db   atlen=100ft
*
.model rg9b/u   u          LEVEL=3     plev=2       elev=3
+              Zk=50       vrel=.66
+             fr1=100meg at1=2.1db   atlen=100ft
*
.model rg11/u   u          LEVEL=3     plev=2       elev=3
+              Zk=75       vrel=.78
+             fr1=100meg at1=1.5db   atlen=100ft
*
.model rg11a/u u          LEVEL=3     plev=2       elev=3
+              Zk=75       vrel=.66
+             fr1=100meg at1=1.9db   atlen=100ft
*
.model rg54a/u u          LEVEL=3     plev=2       elev=3
+              Zk=58       vrel=.66
+             fr1=100meg at1=3.1db   atlen=100ft
*
.model rg15/u   u          LEVEL=3     plev=2       elev=3
+              Zk=53.5     vrel=.66
+             fr1=100meg at1=4.1db   atlen=100ft
*
.model rg53/u   u          LEVEL=3     plev=2       elev=3
+              Zk=53.5     vrel=.66
+             fr1=100meg at1=4.1db   atlen=100ft
*
.model rg58a/u u          LEVEL=3     plev=2       elev=3
+              Zk=50       vrel=.66
+             fr1=100meg at1=5.3db   atlen=100ft
*
.model rg58c/u u          LEVEL=3     plev=2       elev=3
+              Zk=50       vrel=.66
+             fr1=100meg at1=5.3db   atlen=100ft
*
.model rg59b/u u          LEVEL=3     plev=2       elev=3
+              Zk=75       vrel=.66
+             fr1=100meg at1=3.75db   atlen=100ft
*
.model rg62/u   u          LEVEL=3     plev=2       elev=3
+              Zk=93       vrel=.84
+             fr1=100meg at1=3.1db   atlen=100ft
*
.model rg62b/u u          LEVEL=3     plev=2       elev=3
```

Twinlead Models

Example:

```
.model tw/sh    u          LEVEL=3     plev=3       elev=3
*               Shielded TV type twinlead
+               Zk=300     vrel=.698
+              fr1=57meg   at1=1.7db   atlen=100ft
*
.model tw/un    u          LEVEL=3     plev=3       elev=3
*               Unshielded TV type twinlead
+               Zk=300     vrel=.733
+              fr1=100meg at1=1.4db   atlen=100ft
```

Two Coupled Microstrips

Figure 109 on page 325 shows two metal lines formed of the first aluminum layer of a modern CMOS process. The microstrip model assumes that the metal strips sit on top of a dielectric layer that covers the reference plane.

This example is based on demonstration netlist strip2.sp, which is available in directory $<installdir>/demo/hspice/tline:

```
* file strip2.sp two microstrips coupled together
*.... the tests following use geometric/physical model
.option post acct list
* ...signal source
.tran 1ps 500ps
.param rx=56
* excitation volatge + prefilter
v1 np1 0 pwl 0.0s 0v 50ps 0v 60ps 1v
xri1 np1 ni1 rcfilt rflt=rx tdflt=1ps
* ...circuit definition
ue1 ni1 ni2 0 no1 no2 noref u1 l=9.0m
ri2 ni2 0 rx
ro1 no1 noref rx
ro2 no2 noref rx
* ...model definition
.model u1 u level=3 plev=1 elev=1 nl=2
+ kd=3.5 xw=0.1u rho=17e-9 rhob=20e-9
+ wd=2.0u ht=2.0u th=0.8u sp=2.0u
+ llev=1 maxl=50
*
.end
*
```

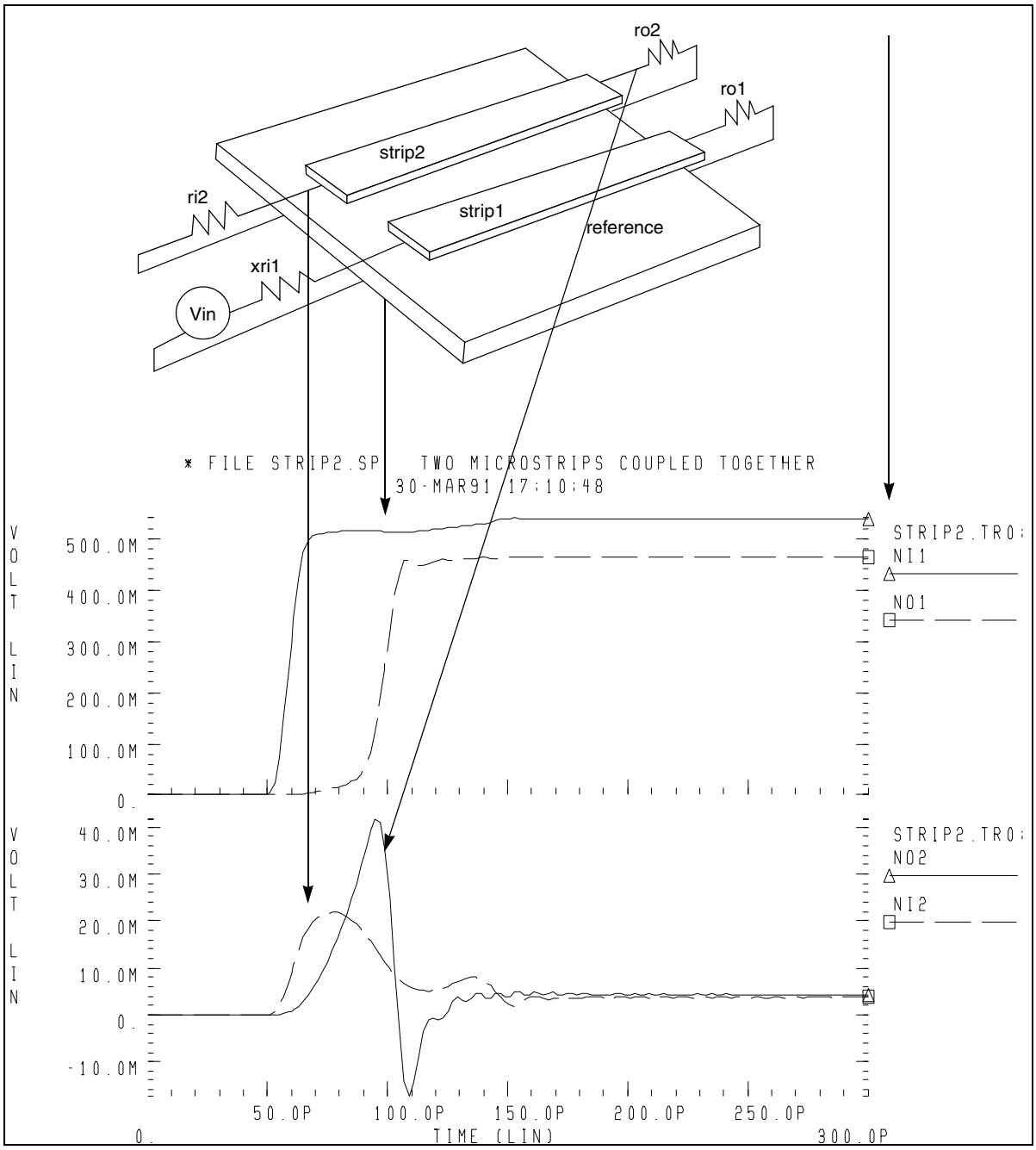


Figure 109 Two Coupled Microstrips Geometrically Defined as LSI Metallization

Solving Ringing Problems with U-elements

Ringing oscillations at sharp signal edges may be produced by:

- Oscillations due to the simulator
- Oscillations due to lossy approximation of a transmission line (U-element)
- Signal reflections due to impedance mismatch

The primary reason for using a circuit simulator to measure high speed transmission line effects is to calculate how much transient noise the system contains and to determine how to reduce it to acceptable values.

The following sections discuss these topics:

- Oscillations Due to Simulation Errors
- Timestep Control Error
- Incorrect Number of Element Lumps
- Default Computation
- Using a Multi-Stage RC Filter to Prevent Ringing
- Signal Reflections Due to Impedance Mismatch

Oscillations Due to Simulation Errors

The system noise results from the signal reflections in the circuit. It may be masked by noises from the simulator. Simulator noise must be eliminated in order to obtain reliable system noise estimates. The following sections describes ways to solve problems with simulator noise.

Timestep Control Error

The default method of integrating inductors and capacitors is trapezoidal integration. While this method gives excellent results for most simulations, it can lead to what is called trapezoidal ringing. This is numerical oscillations that look like circuit oscillations, but are actually timestep control failures. In particular, trapezoidal ringing can be caused by any discontinuous derivatives in the nonlinear capacitance models, or from the exponential charge expressions for diodes, BJTs, and JFETs.

Set the `.OPTION METHOD=GEAR` to change the integration method from trapezoidal. The GEAR method does not ring and, although it typically gives a slightly less accurate result, is still acceptable for transient noise analysis.

Incorrect Number of Element Lumps

Use the correct number of lumps in a lossy transmission line element. Too few lumps results in false ringing or inaccurate signal transmission, while too many lumps leads to an inordinately long simulation run.

Sometimes, as in verification tests, it is necessary to be able to specify the number of lumps in a transmission line element directly. The number of lumps in a lossy transmission line element can be directly specified, defaulted to an accuracy and limit based computation, or computed with altered accuracy and limit and risetime parameters.

Default Computation

In the default computation, `LUMPS=1` until a threshold of total delay versus risetime is reached:

```
TDeff = RISETIME/20
```

- `TDeff` = total end-to-end delay in the transmission line element
- `RISETIME` = the duration of the shortest signal ramp as given in the statement:

  ```
  .OPTION RISETIME = value
  ```

At the threshold, two lumps are used. Above the threshold, the number of lumps is determined by:

```
number of lumps = minimum of 20 or [1+(TDeff/RISETIME)*20]
```

The upper limit of 20 is applied to enhance simulation speed.

If the standard accuracy-based computation does not provide enough lumps, or if it computes too many lumps for simulation efficiency, you can use one of several methods to change the number of lumps on one or more elements:

- Specify `LUMPS=value` in the element statement.
- Specify `.MODEL MAXL=value` and `.MODEL WLUMP=value`.
- Specify a different `RISETIME=value` in an `.OPTION` statement.
- Specify `LUMPS=value` (direct specification)

Direct specification overrides the model and limit based computation, applying only to the element specified in the element statement as in the following example:

```
U35 n1 gnd n2 oref    model lumps=31 L=5m
```

The preceding example specifies 31 lumps for an element of length 5 mm.

Specify MAXL and WLUMP (altered accuracy and limit parameters). You can alter the default computation for all the elements that refer to a particular model by specifying the MAXL and WLUMP model parameters (which would otherwise default to 20). In the nondefault case the number of lumps, the threshold, and the upper limit all would be changed:

```
lumps = min{MAXL, [1+(TDeff/RISETIME)*WLUMP]}
Threshold: TDeff = RISETIME/WLUMP
Upper lim: MAXL
```

Specify a different RISETIME parameter in the .OPTION statement. You can change the threshold and number of lumps computed for all elements of all models, reduce or increase the RISETIME analysis parameter. Note that care is required if RISETIME is decreased, because the number of lumps may be limited by MAXL in some cases where it was not previously limited.

Using a Multi-Stage RC Filter to Prevent Ringing

Artificial sources such as pulse and piecewise linear sources often are used to simulate the action of real output buffer drivers. Since real buffers have a finite cutoff frequency, a multistage filter can be used to give the ideal voltage source reasonable impedance and bandwidth.

You can place a multistage RC filter, shown below, between the artificial source and any U -element to reduce the unrealistic source bandwidth and, consequently, the unrealistic ringing. To provide as much realism as possible, the interposed RC filter and the PWL (piecewise linear) source must be designed together to meet the following criteria:

- Reduce the ringing to acceptable levels
- Preserve the realistic bandwidth of the source signal
- Provide a driver with any chosen impedance
- Provide accurately timed transient signals

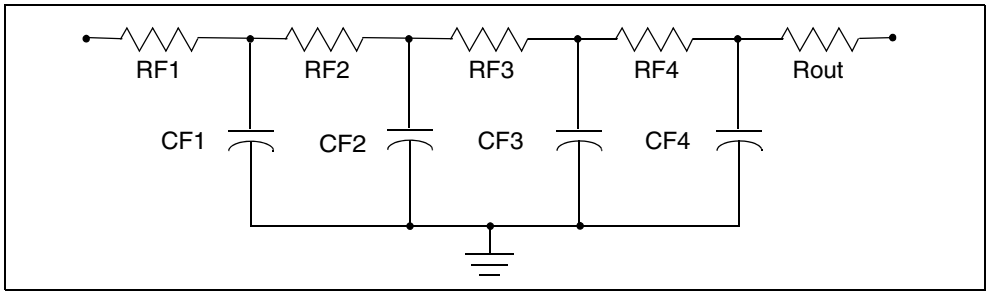


Figure 110 Circuit Diagram of an RC Filter

Example:

```
.MACRO RCFNEW in out gnd_ref RFLT=50 TDFLT=100n
*
*     Begin RCFILT.inc (RC filter) to smooth and match pulse
*     sources to transmission lines. User specifies impedance
*     (RFLT) and smoothing interval (TDFLT). TDFLT is usually
*     specified at about .1*risetime of pulse source.
*
*    cutoff freq, total time delay; and frequency dependent
*     impedance and signal voltage at "out" node
*
* smoothing period = TDFLT.....(equivalent boxcar filter)
* delay= TDFLT
* fc=2/(pi*TDFLT).............(cutoff frequency)
* Zo~ RFLT*(.9 + .1/sqrt( 1 + (f/fc)^2 )
*     V(out)/V(in)= [1 / sqrt( 1 + (f/fc)^2 )]^4
*
.PARAM TD1S='TDFLT/4.0'
RF1 in n1 '.00009*RFLT'
RF2 n1 n2 '.0009*RFLT'
RF3 n2 n3 '.009*RFLT'
RF4 n3 n4 '.09*RFLT'
Rout n4 out '.90*RFLT'
*
.PARAM CTD='TD1S/(.9*RFLT)'
CF1 n1 0 '10000*CTD'
CF2 n2 0 '1000*CTD'
CF3 n3 0 '100*CTD'
CF4 n4 0 '10*CTD'
*
.EOM
```

From the comments embedded in the macro, the output impedance varies from `RFLT` in the DC limit to 3% less at FC and 10% less in the high frequency limit.

$$(RFLT|_{DC} = 0.99999 \cdot \ RFLT)\backslash$$

$$RFLT|_{FC} = 0.97 \cdot \ RFLT$$

Therefore, setting `RFLT` to the desired driver impedance gives a reasonably good model for the corrected driver impedance. `TDFLT` is generally set to 40% of the voltage source risetime.

```
* excitation voltage + prefilter
V1 np1 0 PWL 0.0s 0v 50ps 0v 60ps 1v
xrI1 NP1 NI1 RCFNEW rflt=rx tdflt=4ps
```

Note: RCFNEW is an automatic include file named *$installdir*/parts/behave/rcfilt.inc, where *$installdir* is the installation directory.

Figure 111 on page 330 shows the input and output voltages for the filter.

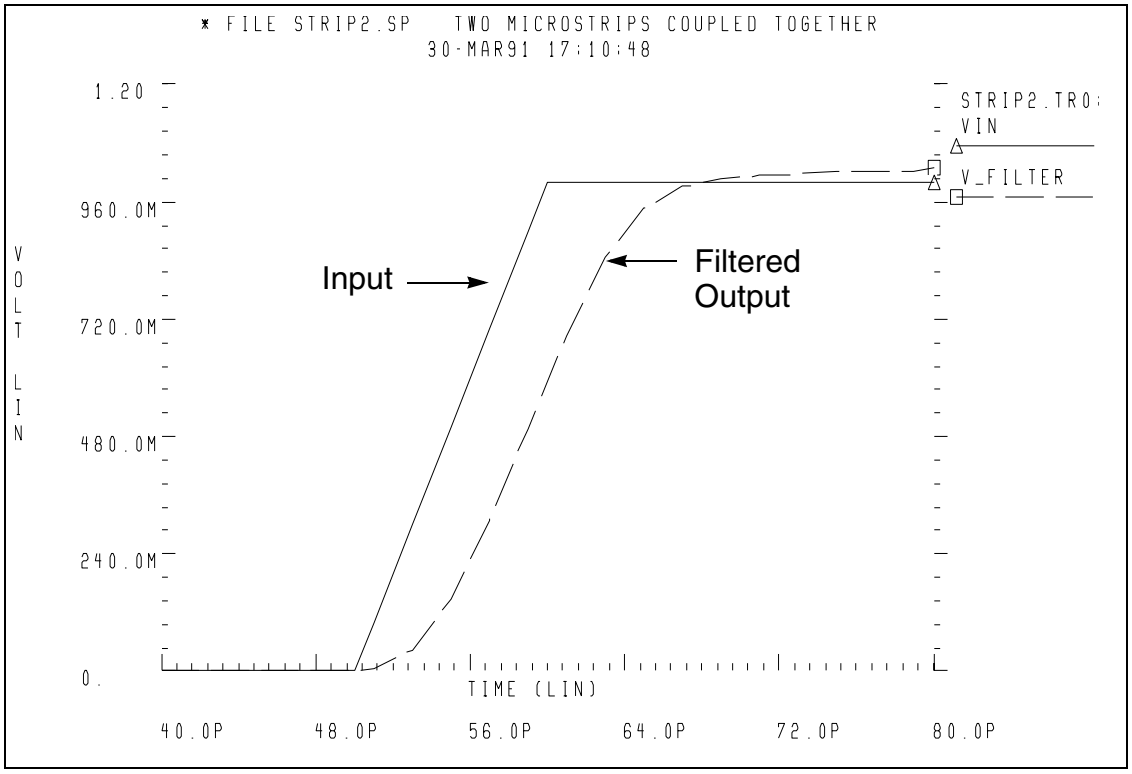


Figure 111 Multistage RC Filter Input and Output

Signal Reflections Due to Impedance Mismatch

The effect of impedance mismatch is demonstrated in the following example. This circuit has a 75 ohm driver, driving 3 inches of 8 mil wide PCB (middle layer), then driving 3 inches of 16 mil wide PCB.

Figure 112, and Figure 113 on page 332, and Figure 114 on page 333, show operational characteristics of such a circuit. The first steady value of impedance is 75 ohms, which is the impedance of the first transmission line section. The input impedance falls to 56 ohms after about 2.5ns when the negative reflection from the nx1 node reaches nil. This TDR displays one idiosyncrasy of the U-element. The high initial value of Zin (TDR) is due to the fact that the input element of the U-element is inductive. The initial TDR spike can be reduced in amplitude and duration by simply using a U-element with a larger number of lumped elements.

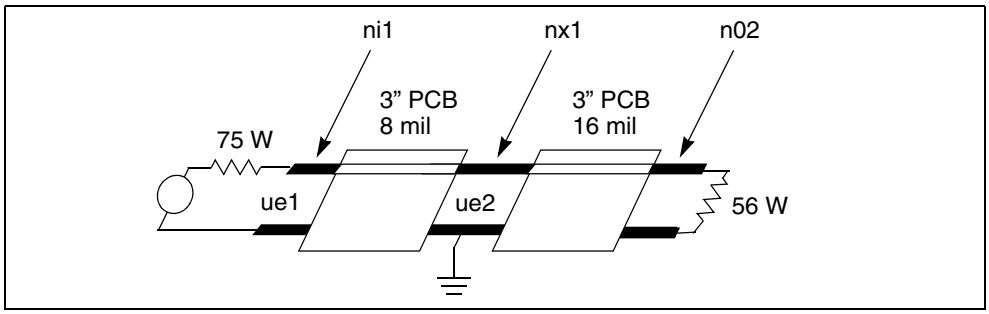


Figure 112 Mathematical Transmission Line Structure

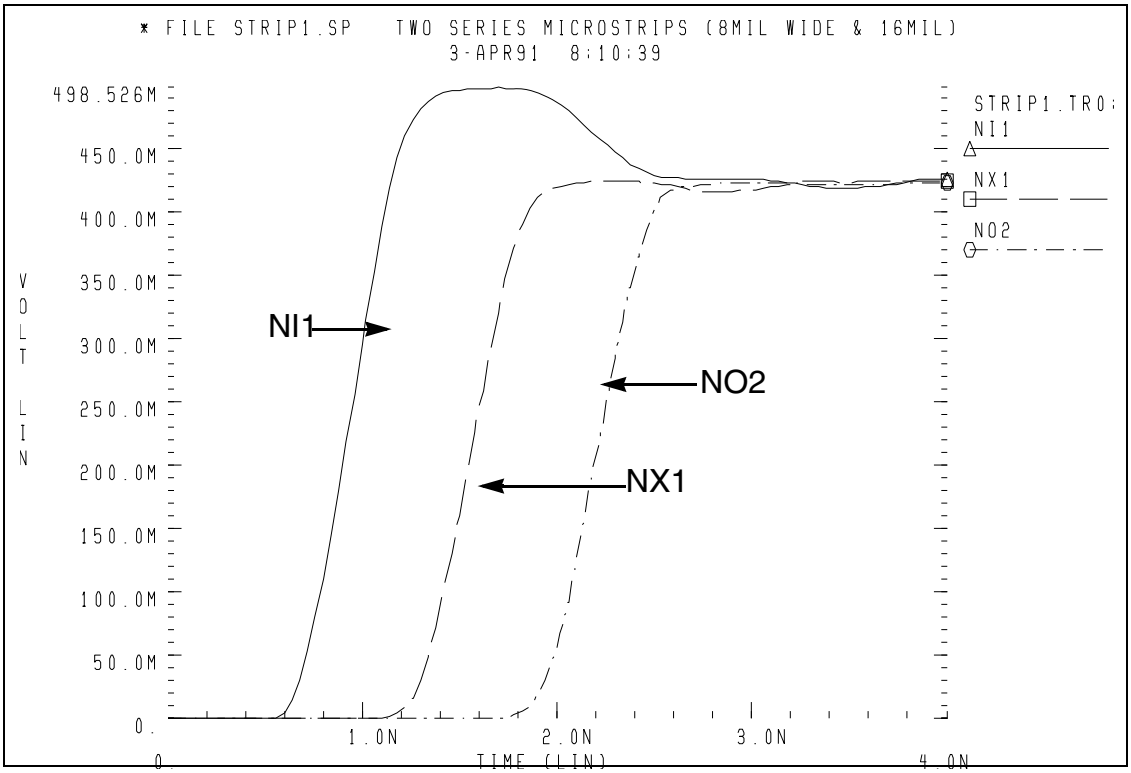


Figure 113 Waveforms in Mismatched Transmission Line Structure

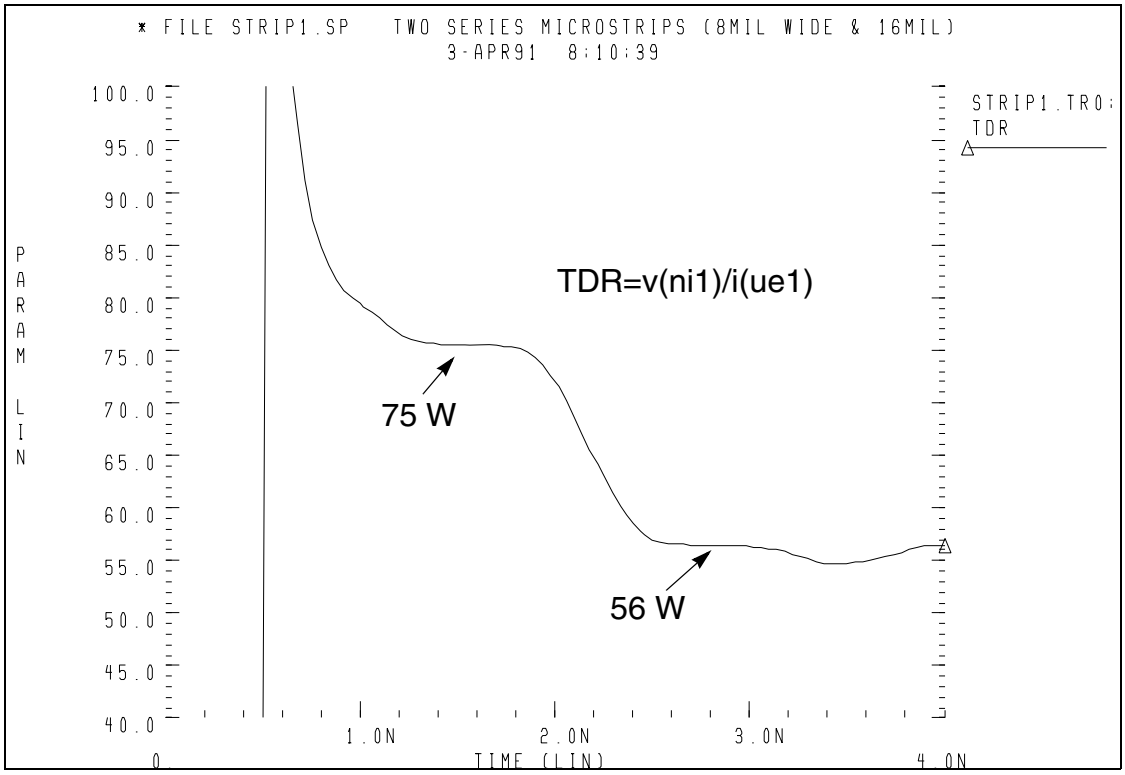


Figure 114 Impedance from TDR at Input

This example is based on demonstration netlist strip1.sp, which is available in directory *$installdir*/demo/hspice/tline:

```
* file strip1.sp two series microstrips (8mil wide & 16mil)
* drive 3 inches of 8mil middle layer pcb, series connected to
* 3 inches of 16mil wide middle layer pcb, 500ns risetime driver
*.... the tests following use geometric/physical model
.option post acct list
* ...signal source
.tran 20ps 4ns
.probe tdr=par('v(ni1)/i(ue1)')
.param r8mil=75 r16mil=56
* excitation volatge + prefilter
v1 np1 0 pwl 0.0s 0v 500ps 0v 1n 1v
xri1 np1 ni1 rcfilt rflt=r8mil tdflt=50ps
* ...circuit definition
ue1 ni1 0 nx1 0 u1 l=3000mil
ue2 nx1 0 no2 0 u2 l=3000mil
ro2 no2 0 r16mil
* ...model definition -- 8mil middle metal layer of a copper pcb
.model u1 u level=3 plev=1 elev=1 nl=1
+ th=1.3mil ht=10mil ts=32mil kd=4.5 dlev=0
+ wd=8mil xw=-2mil
* ...model definition -- 16mil middle metal layer of a copper pcb
.model u2 u level=3 plev=1 elev=1 nl=1
+ th=1.3mil ht=10mil ts=32mil kd=4.5 dlev=0
+ wd=16mil xw=-2mil
*
.end
```

A

Transmission Line Theory

This appendix relates distributed RLGC values of a transmission line to its characteristic impedance, transmission velocity, and loss; uses the concepts of self and mutual inductance to explain crosstalk; describes rules of thumb for various types of clock pulses; discusses the sources of transmission line attenuation.

The following topics are covered in this appendix.

- Lossless Transmission Line Model
- Lossy Transmission Line Model
- Impedance
- Inductance
- Crosstalk in Transmission Lines
- Risetime, Bandwidth, and Clock FrequencyDefinitions of Transmission Line Terms
- Relationships and Rules of Thumb
- Attenuation in Transmission Lines

Lossless Transmission Line Model

As a signal propagates down a pair of conductors, each section acts electrically as a small lumped element circuit. In its simplest form, called the lossless model, the equivalent circuit of a transmission line has just inductance and capacitance. These elements are distributed uniformly down the length of the line as shown in Figure 115.

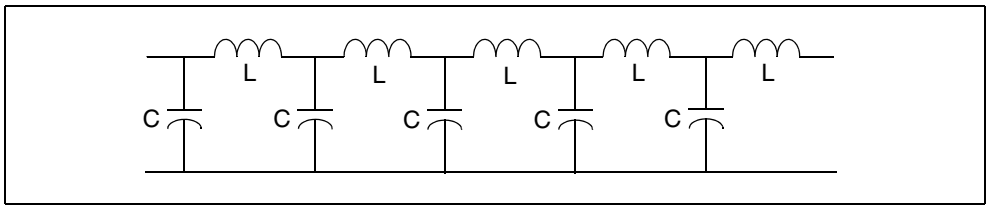


Figure 115 Equivalent Circuit Model of a Lossless Transmission Line

From this electrical circuit model, the two important terms that characterize a transmission line can be derived: the velocity of a signal (v) and the characteristic impedance (Z_0).

$$v = \frac{1}{\sqrt{L_L C_L}} \quad \text{and} \quad Z_0 = \sqrt{\frac{L_L}{C_L}}$$

- L_L = inductance per unit length
- C_L = capacitance per unit length

This is the basis for the T-element. It accounts for a characteristic impedance (Z_0) and a time delay (TD). The time delay depends on the propagation velocity (v) and distance (d) between the two ends of the transmission line: $TD = \frac{d}{v}$

Lossy Transmission Line Model

When loss is significant, the effects of a per unit length series resistance (R) and a dielectric conductance (G) should be included. Figure 116 shows the equivalent circuit model of a lossy transmission line with distributed “lumps” of R, L, and C -elements.

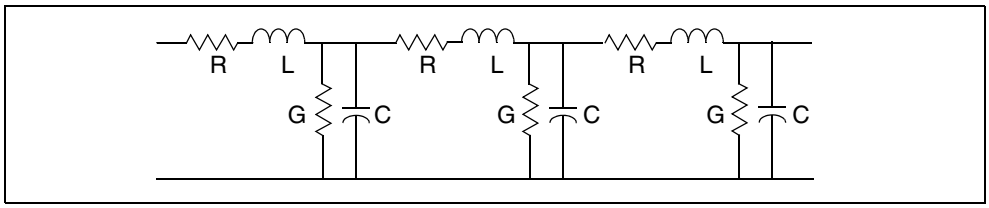


Figure 116 Equivalent Circuit Model of a Lossy Transmission Line

The U-element can be used to provide an equivalent circuit model for a lossy transmission line. In a transient simulation, the U-element automatically accounts for frequency-dependent characteristic impedance, dispersion (frequency dependence in the velocity), and attenuation by forming a lumped element equivalent for the lossy transmission line.

Common types of transmission line cross sections include microstrip, stripline, coax, wire over ground, and twisted pair. There is a direct relationship between cross section, velocity of propagation, and characteristic impedance.

In a balanced transmission line, the two conductors have similar properties and are electrically indistinguishable. For example, each wire of a twisted pair has the same voltage drop per length down the line The circuit model for each wire has the same resistance capacitance and inductance per length.

This is not the case with a microstrip line or a coaxial cable. In those structures, the signal conductor has a larger voltage drop per length than the other conductor. The wide reference plane in a microstrip or the larger diameter shield in a coax have lower resistance per length and lower inductance per length than the signal line. The equivalent circuit model for unbalanced lines typically assumes the resistance and inductance per length of the ground path is zero and all the voltage drop per length is on the signal conductor. Even though the inductance of the reference plane is small, it can play a significant role when there are large transient currents.

Impedance

The impedance of a device (Z) is defined as the instantaneous ratio of the voltage across the device (V) to the current through it: $Z = \frac{V}{I}$

Impedance of Simple Lumped Elements

The impedance of a device can be thought of as the quality of the device that causes it to transform a current through it into a voltage across it: $V = ZI$

The admittance (Y) is less often used to characterize a device. It is the inverse of the impedance: $Y = \frac{1}{Z} = \frac{I}{V}$

There are three ideal circuit elements used to describe passive components: a resistor, a capacitor, and an inductor. They are defined by how they interact with voltage across them and current though them:

Resistor with resistance (R): $V = IR$

Capacitor with capacitance (C): $I = C\frac{dV}{dt}$

Inductor with inductance (L): $V = L\frac{dI}{dt}$

When the voltage or current signals are time dependent, the impedance of a capacitive or inductive element is a very complicated function of time. You can simulate it, but it is difficult to build an intuitive model.

The impedance of a capacitor rotates the phase of the current 90° in the negative or direction to generate the voltage across the capacitor. The impedance of an inductor rotates the current 90° in the positive direction to generate the voltage across the inductor. For a resistor, the current and voltage have the same phase.

In the frequency domain when all signals are sine waves in the time domain, the impedance of a capacitor and an inductor is frequency dependent, decreasing with frequency for a capacitor and increasing with frequency for an inductor. The impedance of a resistor is constant with frequency.

In the real world of finite dimensions and engineered materials, ideal circuit elements have associated parasitics, which cause them to behave in complex ways that are very apparent at high frequencies.

Characteristic Impedance

A controlled impedance transmission line is a pair of conductors that have a uniform cross section and uniform distribution of dielectric materials down their

length. A short segment, Δx of the transmission line has a small capacitance associated with it, ΔC, which is the capacitance per length, C_L, times the Δx:

$$\Delta C = C_L \Delta x$$

When a voltage signal is introduced at one end, the voltage between the conductors induces an electric that propagates the length of the line at the speed of light in the dielectric. As the voltage signal moves down the line, each new section of line charges up. The new section of line, Δx, is charged up in a time Δt:

$$\Delta t = \frac{\Delta x}{v}$$

If the voltage (V) moves down the line at a constant speed, and the capacitance per length is uniform through the line, then the constant voltage applied to the front end draws a constant charging current (I):

Equation 74
$$I = \frac{\Delta Q}{\Delta t} = \frac{\Delta C V}{\Delta t} = \frac{C_L \Delta x V}{\Delta t} = C_L v V$$

This constant voltage with constant current has the behavior of a constant impedance (Z):

Equation 75
$$Z = \frac{V}{I} = \frac{V}{C_L v V} = \frac{1}{C_L v}$$

The impedance is determined by the speed of the signal and the capacitance per length of the pair of conductors, both intrinsic properties of the line. This intrinsic impedance is termed the characteristic impedance of the line (Z_0).

If a measurement is made at one end of the line in a short time compared to the round trip time delay, the line behaves as a resistor with a resistance equal to the characteristic impedance of the line. Transmission line effects are only important when rise times are comparable or shorter than the round trip time delay.

For example, if the rise time of a device is 1 ns, and it drives an interconnect trace in FR4 which is longer than three inches, the load on the device during the risetime is purely resistive. For CMOS devices, which are used to drive high resistance loads, the typical 50 ohm resistance they see initially can significantly distort the waveform from what is expected.

It is only during the initial surge of the voltage that a transmission line behaves as a constant impedance with a value equal to its characteristic impedance. For this reason the characteristic impedance of a line is also called the surge impedance. The surge time during which the impedance is constant is the round trip time of flight or twice the time delay. Reflections from the far end complicate the electrical behavior of the line after the surge time.

The instantaneous impedance measured at the front end of a transmission line is a complicated function of time. It depends on the nature of the terminations at the far end. When the line is shunted to ground with a resistor of value equal to the characteristic impedance of the line, there is no reflection back, and the front end of the line behaves as a resistive load. When the termination at the far end is open, the impedance at the front end starts out at the characteristic impedance and eventually, after multiple reflections, approaches an infinite impedance. During some periods the instantaneous impedance may be zero. These transient effects are fully simulated with T -elements and U -elements.

Inductance

This section gives an operational definition of inductance and explains the differences between mutual and subtle inductance.

These topics are discussed in the following sections:

- Mutual Inductance and Self Inductance
- Operational Definition of Inductance
- Mutual Inductance
- Self Inductance
- Reference Plane Return Paths

Mutual Inductance and Self Inductance

The most confusing, subtle, and important parameter in high-speed packaging and interconnect design is inductance. It plays a key role in the origin of simultaneous switching noise, also called common ground inductance, and a key role in crosstalk between transmission line structures.

Operational Definition of Inductance

Consider an inductor to be any section of circuit element which carries current: an interconnect trace, a ground plane, a TAB lead frame, a lead in a DIP package, the lead of a resistor or a pin in a connector. An inductor does not have to be a closed circuit path, but can be a small section of a circuit path.

A changing current passing through an inductor generates a voltage drop. The magnitude of the voltage drop (ΔV) for an inductance (L) and change in current (dI/dt) is:

Equation 76 $$\Delta V = L\frac{dI}{dt}$$

This definition can always be used to evaluate the inductance of a section of a circuit. For example, with two long parallel wires, each of radius (r) and a center-to-center separation s, you can measure the voltage drop per length for one of the wires when a changing current dI/dt flows through one wire and back through the other. The induced voltage per length on one of the wires is:

$$V_L = \frac{\mu_0}{2\pi} Ln\left(\frac{s}{r} - \frac{r}{s}\right)\frac{dI}{dt}$$ [V in mV/inch, I in mA, t in ns]

From this expression, the effective inductance per length of a wire is:

$$L_L = \frac{\mu_0}{2\pi} Ln\left(\frac{s}{r} - \frac{r}{s}\right) \approx 5Ln\left(\frac{s}{r}\right)(s \gg r)$$ [nH/inch]

Mutual Inductance

A second effect also is important: the induced voltage from currents that are adjacent to, but not in, the same circuit path. This is caused by the mutual inductance between two current elements. A section of conductor in a circuit, labeled 1, may have an induced voltage generated across it because of currents not in circuit 1, but from circuits 2, 3, and 4.

The voltage generated across the section of circuit 1, V_1, is given in its general form by:

Equation 77 $$V_1 = L_{11}\frac{dI_1}{dt} + L_{12}\frac{dI_{I2}}{dt} + L_{13}\frac{dI_3}{dt} + L_{14}\frac{dI_4}{dt}$$

The notation for mutual inductance (L_{ab}) is related to the induced voltage on circuit element a, from the current element, b. In some texts, the symbol used is M, rather than L. The special case of the induced voltage on a circuit element from its own current (L_{aa}) is called self inductance.

Mutual inductance relates to the magnitude of induced voltage from an adjacent current. The magnitude of this voltage depends on the flux linkages between the two circuit elements.

Self Inductance

The self inductance of an isolated single trace is a well-defined, absolute mathematical quantity, but not a measurable physical quantity. There is always a return current path somewhere, and the mutual inductance from this return current path induces a voltage on the circuit element that subtracts from the self inductance. Self inductance can never be measured or isolated, independent of a mutual inductance of a return current path.

In the example above of two long parallel wires, the measured inductance per length (L_L) of one wire is neither the self inductance nor the mutual inductance of the wire. It is a combination of these two terms. If the universe contained just the two wires, the measured voltage drop per length would be:

Equation 78
$$V_L = L_{11}\frac{dI_1}{dt} - L_{12}\frac{dI_1}{dt} = (L_{11} - L_{12})\frac{dI_1}{dt} = L_L\frac{dI_1}{dt}$$

The minus sign reflects the opposite directions of the currents I_1 and I_2. Operationally, when the inductance per length of one wire is measured, what is really measured is the difference between its self inductance and the mutual inductance of the return path. Because of this effect, it is clear that the nature of the return path greatly influences the measured inductance of a circuit element.

Reference Plane Return Paths

The capacitance per length (C_L) of any planar transmission line is:

$$C_L = \frac{85}{Z_0}\sqrt{\varepsilon_r} \text{ [pF/inch]}$$

The inductance per length of the signal line (L_L) is:

$$L_L = 0.085 Z_0 \sqrt{\varepsilon_r} \text{ [nH/inch]}$$

This is the self inductance of the signal line, minus the mutual inductance of the return current in the reference plane.

For example, the inductance per length of a transmission line with characteristic impedance of 50 ohms in an FR4 printed circuit board is 9.6 nH/inch. The capacitance per length is 3.8 pF/inch.

In the equivalent circuit of a lossless transmission line, the series inductance per length is 9.6 nH/inch, and the shunt capacitance to ground is 3.8 pF/inch.

In the notation of the U -element for an `ELEV=2` (RCLK equivalent model) and a `PLEV=1` (microstrip cross section), the parameters to model this lossless transmission line are

```
C11 = 3.8 pF/inch, L11 = 9.6 nH/inch
```

The `LLEV=0` parameter simplifies the inductance problem by automatically calculating the inductance of the line as the difference between the self inductance of the line and the mutual inductance of the return signal path.

In many texts, the term L11 is generically used as the self inductance. `LLEV=1` assumes a circuit ground point, separate from the reference plane of the transmission line. Thus the `LLEV=1` option includes an approximation to the self inductance of both the signal conductor and the reference plane, while `LLEV=0` assumes a reference plane return current.

Crosstalk in Transmission Lines

When there are adjacent transmission lines, for instance line 2 and line 3, the coupling capacitance and inductance between them and the quiet line, line 1, lead to crosstalk.

In the notation of the Synopsys transmission line models, the voltage per length on transmission line 1, V_1, including the mutual inductance to lines 2 and 3 is:

Equation 79

$$V_1 = L11\frac{dI_1}{dt} + L12\frac{dI_2}{dt} + L13\frac{dI_3}{dt}$$

`LLEV=0` simplifies the inductance analysis by automatically including the effects of the return current path. The first inductance term (L11) is the

inductance per length of the transmission line (1) including the self inductance of the line and the mutual inductance of the return ground path.

The second term, the coupling inductance of the second transmission line (L12), includes the mutual inductance of the second signal line and the mutual inductance of the return current path of the second line. Because these two currents are in opposite directions, the mutual inductance of the pair is much less than the mutual inductance of just the second signal trace alone.

The third term (LL13) includes the mutual inductance of the signal path in the third transmission line and the mutual inductance of its return path through the reference plane.

The coupling inductances (L12 and L13) include the mutual inductances of the adjacent signal lines and their associated return paths. They are more than the mutual inductance of the adjacent traces. This is equivalent to the operational inductances that you can measure with a voltmeter and a dI/dt source.

Risetime, Bandwidth, and Clock Frequency

In the time domain, a clock waveform can be described in terms of its period (T_{clock}), its frequency (F_{clock}), and a risetime (τ_{edge}). Figure 117 illustrates these features.

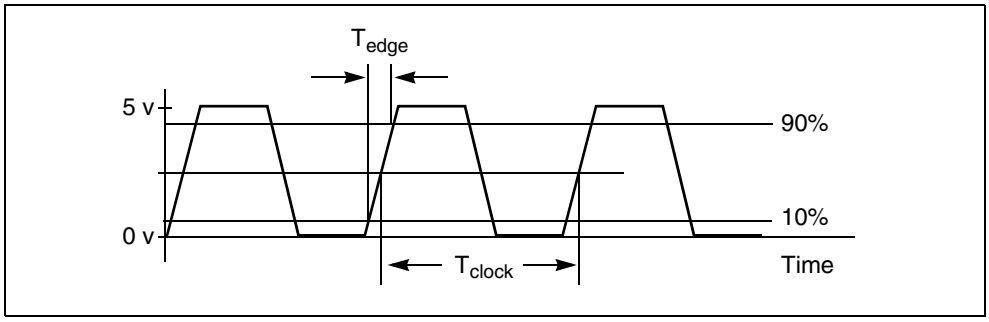


Figure 117 Clock Waveform

The risetime is typically the time between the 10% to 90% points.

To describe this waveform in terms of sine wave components, the highest sine wave frequency required (the bandwidth, BW) depends on the risetime. As the bandwidth increases and higher sine wave frequency components are

introduced, the risetime of the reconstructed waveform decreases. The bandwidth of a waveform is determined by the fastest risetime it contains. The risetime and bandwidth are related by:

Equation 80 $$BW = \frac{0.35}{\tau_{edge}} \text{ or } \tau_{edge} = \frac{0.35}{BW}$$

The risetime of a clock waveform and the clock period are only indirectly related. The risetime of a system is determined by the output driver response and the characteristics of the packaging and interconnect. In general, the risetime is made as long as possible without degrading the clock period.

Without specific information about a system, it is difficult to know the precise risetime, given just the clock frequency or period. In a fast system such as an oscillator with only one gate, the period might be two times the risetime:

Equation 81 $$T_{period} \approx 2\tau_{edge}$$

For a complex system such as a microprocessor board, the period might be 15 times as long as the risetime:

Equation 82 $$T_{period} \approx 15\tau_{edge}$$

In each case, the bandwidth is always related to the risetime by the first expression in this section, and the clock frequency and clock period are always related by:

Equation 83 $$F_{clock} = \frac{1}{T_{clock}}$$

The microprocessor example reports the worst case of the shortest risetime for a given clock period. Combining these expressions shows the relationship between clock frequency and bandwidth:

Equation 84 $$F_{clock} = \frac{1}{T_{clock}} = \frac{1}{15\tau_{edge}} = \frac{BW}{15 \cdot 0.35} \approx \frac{1}{5}BW \text{ or } BW \approx 5F_{clock}$$

In general, the highest sine wave frequency component contained in a clock waveform is five times the clock frequency. The important assumption is that there are about 15 risetimes in one period. If the risetime is actually faster than this assumption, the bandwidth is higher. To provide a safety margin, a package or interconnect is characterized or simulated at a bandwidth of about 10 to 20

times the clock frequency, which corresponds to roughly two to four times the bandwidth of the signal.

Definitions of Transmission Line Terms

The following table gives the definitions of transmission line terms.

Table 48 Transmission Line Terms

Term	Description
F_{clock}	Clock frequency in units of frequency such as Hz
T_{clock}	Clock period in units of time such as secs or ns
t_{edge}	Rise or fall time in units of time such as sec or ns
BW	Bandwidth in units of frequency such as Hz or MHz
F	A repetitive frequency in units of Hz or MHz
f	A sine wave frequency in units of Hz or MHz
ω	An angular frequency in units of radians/sec
t	Time or a conductor thickness, units of sec or length
V(t)	Instantaneous voltage in units of volts
I(t)	Instantaneous current in units of amps or mA
Z(t)	Instantaneous impedance in units of ohms
$Z(\omega)$	Frequency domain, complex impedance in units of ohms
C	Capacitance in units of Farads or microFarads
R	Resistance in units of ohms
L	Inductance in units of Henrys or nanoHenries
v	Speed of light in the medium in units of length/time

Table 48 Transmission Line Terms (Continued)

Term	Description
d	A length in units such as inches
TD	Time delay in units of time such as sec or nsec
CL	Capacitance per length in units such as pF/inch
LL	Inductance per length in units such as nH/inch
RL	Resistance per length in units such as ohms/inch
ε_r	Relative dielectric constant, $\varepsilon/\varepsilon_0$, dimensionless
r	Reflection coefficient, dimensionless
DZ	A small change in characteristic impedance
GL	Conductance per length units of Mhos (Siemens)/length
tan(δ)	Dissipation factor of a material, dimensionless
d	Skin depth of a conductor, units of length such as meter
ρ	Resistivity of a conductor, units of ohm-length
μ_0	Permeability of free space = 4 p x 10-7 Henry/meter
α	Attenuation per length, units of dB/len or nepers/len
w	A conductor width
n	Number of squares in a planar conductor, dimensionless
Rsq	Sheet resistance of a planar conductor, units: ohms/sq
ε_0	Permittivity of free space=0.225 pF/inch=0.0885 pF/cm
ε_{eff}	Effective dielectric constant due to mixed dielectrics
h	A dielectric thickness in units of length such as mils

Table 48 Transmission Line Terms (Continued)

Term	Description
R	Resistance in ohms

Relationships and Rules of Thumb

This section explains the rules and properties associated with transmission lines.

Topics:

- Time and Frequency Relationships
- Transmission Line Effects
- Intrinsic Properties
- Reflections
- Loss and Attenuation
- Physical Design Quantities

Time and Frequency Relationships

Equation 85 $$F_{clock} = \frac{1}{T_{clock}}$$

Equation 86 $$BW = \frac{0.35}{\tau_{edge}} \text{ or } \tau_{edge} = \frac{0.35}{BW}$$

Equation 87 $$T_{period} \approx 15\tau_{edge}$$

Equation 88 $$F_{clock} \approx \frac{1}{5}BW \text{ or } BW \approx 5F_{clock}$$

Transmission Line Effects

Transmission line analysis recommended for:

$$BW > \frac{v}{10d} = \frac{1}{10 \cdot TD}$$

$$\tau_{edge} < \frac{5d}{v} = 5 \cdot TD$$

On FR4 material:

$BW > \frac{600}{d}$ [BW in MHz, d in inches]

$F_{clock} > \frac{120}{d}$ [Fclock in MHz, d in inches]

$\tau_{edge} < \frac{d}{7.5}$ t_{edge} [in ns, d in inches]

Intrinsic Properties

$Z_0 = \frac{1}{vC_L} = \sqrt{\frac{L_L}{C_L}}$ [CL is in pF/inch

$$v = \frac{1}{\sqrt{L_L C_L}}$$

$$C_L = \frac{1}{vZ_0} = \frac{\sqrt{\varepsilon_r}}{cZ_0} = \frac{85}{Z_0}\sqrt{\varepsilon_r}$$ [pF/inch]

$L_L = \frac{1}{C_L v^2} = \frac{7.3\varepsilon_r}{C_L}$ [LL in nH/inch, CL in pF/inch]

Typical polymers:

$TD = \frac{d}{6}$ [TD in ns, d in inches]

$C_L = \frac{170}{Z_0}$ [pF/inch]

$L_L = 0.172 Z_0$ [nH/inch]

50 ohm lines:

$C_L = 1.7\sqrt{\varepsilon_r}$ [pF/inch = 3.4 pF/inch (typical polymer)]

$L_L = 4.3\sqrt{\varepsilon_r}$ [nH/inch = 8.6 nH/inch (typical polymer)]

Transmission line of length d:

Equation 89 $C = C_L d = \frac{TD}{Z_0}$

Equation 90 $L = L_L d = Z_0 TD$

Reflections

Reflection coefficient from Z_1 to Z_2:

Equation 91 $r = \frac{Z_2 - Z_1}{Z_2 + Z_1}$

Reflection from a ΔZ of short length with time delay, TD, $\tau_{edge} >$ TD:

Equation 92 $r = \frac{\Delta Z}{2Z_0}\left(\frac{TD}{\tau_{edge}}\right)$

Reflection from a series lumped L surrounded by a transmission line:

Equation 93 $r = \frac{L}{2Z_0\tau_{edge}}$

Reflection from a lumped C load to ground, on a transmission line:

Equation 94 $r = \frac{CZ_0}{2\tau_{edge}}$

Loss and Attenuation

Skin depth of a conductor:

Equation 95 $$\delta = \sqrt{\frac{\rho}{\pi\mu_0 f}} = 500 \cdot \sqrt{\frac{\rho}{f}}$$

δ in meters, ρ in ohm.meter, f in Hz (in copper), at f = 1e+9, and ρ =1.7e-8 ohm.meter, δ = 2.0e-6.

Low loss approximation for attenuation per length:

$$\alpha = \frac{1}{2}\left(\frac{R_L}{Z_0} + G_L Z_0\right) \quad \text{[nepers/length]}$$

$$\alpha = 4.34\left(\frac{R_L}{Z_0} + G_L Z_0\right) \quad \text{[dB/length]}$$

Attenuation per length due to just dielectric loss:

$$\alpha = 2.3 f \tan(\delta)\sqrt{\varepsilon_r} \quad \text{[dB/inch, f in GHz]}$$

$$\alpha \approx 0.05 \quad \text{[dB/inch for FR4 at 1 GHz]}$$

Attenuation per length due to metal, t < δ:

$$\alpha = 43.4\frac{\rho}{twZ_0} \quad \text{[dB/inch with } \rho \text{ in ohm/meter, t in microns, w in mils]}$$

For 1 ounce copper microstrip, 5 mils wide, 50 ohm:

$$\alpha = 0.01 \quad \text{[dB/inch]}$$

Attenuation per length due to metal, 50 ohm line, skin depth limited, t > δ:

$$\alpha = 0.55\frac{\sqrt{\rho f}}{w} \quad \text{[dB/inch with } \rho \text{ in ohm.meter, w in mils, f in GHz]}$$

For 1 ounce copper microstrip, 5 mils wide, 50 ohm, at 1 GHz:

$$\alpha = 0.15 \quad \text{[dB/inch]}$$

Physical Design Quantities

For a planar interconnect:

Sheet resistance:

$$R_{sq} = \frac{\rho}{t} \quad \text{[ohm/sq]}$$

Number of squares:

$$n = \frac{d}{w}$$

Resistance:

$$R_L = n \cdot R_{sq} \quad \text{[ohm/inch, Rsq in ohm/sq]}$$

Resistance per length:

$$R_L = 10\frac{\rho}{tw} \quad \text{[ohm/inch, } \rho \text{ in ohm} \cdot \text{ meter, t in microns, w in mils]}$$

Parallel plate, no fringe fields:

$$C_L = 0.225\varepsilon_r\left(\frac{w}{h}\right) \quad \text{[pF/inch]}$$

Microstrip capacitance per length good to ~20%:

$$C_L = 0.45\varepsilon_r\left(\frac{w}{h}\right) \quad \text{[pF/inch]}$$

Microstrip capacitance per length, good to ~5%:

$$C_L = \frac{1.41\varepsilon_{eff}}{ln\left(\frac{8h}{w} + \frac{w}{4h}\right)}$$

$$\varepsilon_{eff} = \left(\frac{\varepsilon_r + 1}{2}\right) + \left(\frac{\varepsilon_r - 1}{2}\right)\left(1 + \frac{10h}{w}\right)^{-\frac{1}{2}}$$

Stripline capacitance per length good to ~20%:

$$C_L = 0.675\varepsilon_r\left(\frac{w}{h}\right) \quad \text{[pF/inch]}$$

Stripline capacitance per length, good to ~5%:

$$C_L = \frac{0.9\varepsilon_r}{ln\left(1+\frac{2h}{w}\right)} \quad \text{[pF/inch]}$$

Inductance per length of one wire in a pair of two parallel wires (r = radius, s = center to center spacing, s » r):

$$L_L = 5Ln\left(\frac{s}{r}\right) \quad \text{[nH/inch]}$$

Inductance per length for a circular loop:

$$L_L = 26 \quad \text{[nH/inch of perimeter]}$$

Inductance per length of controlled impedance line when the return line is a reference plane:

$$L_L = 0.086Z_0\sqrt{\varepsilon_r} = \frac{7.3\varepsilon_r}{C_L} \quad \text{[nH/inch with } C_L \text{ in pF/inch]}$$

Conductance per length:

$$G_L = \omega\tan(\delta)C_L$$

Attenuation in Transmission Lines

The T-element, common to most Berkeley-compatible SPICE tools, uses the lossless model for a transmission line. This model adequately simulates the dominant effects related to transmission behavior: the initial driver loading due to a resistive impedance, reflections from characteristic impedance changes, reflections introduced by stubs and branches, a time delay for the propagation of the signal from one end to the other and the reflections from a variety of linear and nonlinear termination schemes.

In systems with risetimes are on the order of 1 ns, transmission line effects dominate interconnect performance.

In some high-speed applications, the series resistance seriously effects signal strength and should be taken into account for a realistic simulation.

The first-order contribution from series resistance is an attenuation of the waveform. This attenuation decreases the amplitude and the bandwidth of the propagating signal. As a positive result, reflection noise decreases so that a lossless simulation is a worst case. As a negative result, the effective propagation delay is longer because the risetimes are longer. A lossless simulation shows a shorter interconnect related delay than a lossy simulation.

The second-order effects introduced by series resistance are a frequency dependence to the characteristic impedance and a frequency dependence to the speed of propagation, often called dispersion. Both the first order and second order effects of series resistance generic to lossy transmission lines are simulated using the U model.

The following toics are discussed in the nest sections:

- Physical Basis of Loss
- Skin Depth
- Dielectric Loss

Physical Basis of Loss

The origin of loss is the series resistance of the conductors and the dielectric loss of the insulation. Conductor resistance is considered in two parts, the DC resistance and the resistance when skin depth plays a role. The dielectric loss of the insulation, at low frequency, is described by the material conductivity (σ) (SIG in the Synopsys U model), and at high frequency, by the dissipation factor, tan(δ). These material effects contribute a shunt conductance to ground (G).

In a planar interconnect such as a microstrip or stripline, the resistance per length of the conductor R_L is:

Equation 96
$$R_L = \frac{\rho}{wt}$$

ρ = bulk resistivity of the conductor

w = the line width of the conductor

t = thickness of the conductor

Example 1

FR4, 5 mil wide line, half ounce copper:

R_L = 0.24 ohm/inch

Example 2

Cofired ceramic, 4 mil wide, 0.75 mils thick Tungsten:

```
R_L = 2.7 ohm/inch
```

Example 3

Thin film copper, 1 mil wide, 5 um thick:

```
R_L = 3.6 ohm/inch
```

Skin Depth

At high frequency, the component of the electric field along the conductor, which drives the current flow, does not fully penetrate the conductor depth. Rather, its amplitude falls off exponentially. This exponential decay length is the *skin depth* (δ). When the signal is a sine wave, the skin depth depends on the conductor's resistivity (ρ), and the sine wave frequency of the current (f):

$$\delta = \sqrt{\frac{\rho}{\pi \mu_0 f}} = 500 \cdot \sqrt{\frac{\rho}{f}} \quad \delta \text{ [in meters, r in ohmum, f in Hz]}$$

A real signal has most of its energy at a frequency of $1/t_{period}$, where t_{period} is the average period. Because most of the loss occurs at this frequency as a first approximation >> should be used to compute the skin depth. In practice as mentioned previously, approximating tperiod by $15(\tau_{edge})$ works well. With a 1 ns rise time, the skin depth of copper is 8 microns, assuming $15(\tau_{edge})$ is used for $1/f_{skin}$. For half ounce copper, where the physical thickness is 15 microns, the skin depth thickness should be used in place of the conductor thickness to estimate the high frequency effects.

For thin film substrates with a physical thickness of the order of 5 microns or less, the effects of skin depth can, to the first order, be ignored. In cofired ceramic substrates, the skin depth for 1 ns edges with tungsten paste conductors is 27.6 microns. This is also comparable to the 19 micron physical thickness, and to the first order, the effects of skin depth can be ignored. At shorter rise times than 1 ns, skin depth plays an increasingly significant role.

Dielectric Loss

Two separate physical mechanisms contribute to conductivity in dielectrics, which results in loss: DC conduction and high frequency dipole relaxation. As illustrated in the following section, the effects from dielectric loss are in general negligible. For most practical applications, the dielectric loss from the DC conductivity and the high frequency dissipation factor can be ignored.

To be cautious, estimate the magnitude of the conductance of the dielectric and verify that, for a particular situation, it is not a significant issue. Exercise care in using these material effects in general application.

The bulk conductivity of insulators used in interconnects (σ), typically specified as between 10^{-12} and 10^{-16} siemens/cm, is often an upper limit, rather than a true value. It is also very temperature and humidity sensitive. The shunt conductance per length (G_L) depends on the geometrical features of the conductors in the same way as the capacitance per length (C_L). It can be written as:

Equation 97
$$G_L = \sigma\frac{C_L}{\varepsilon_0\varepsilon_r}$$

At high frequencies, typically over 1 MHz, dipole relaxations begin to dominate the conduction current and cause it to be frequency dependent. This effect is described by the dissipation factor of a material, which ranges from 0.03 for epoxies down to 0.003 for polyimides and less than 0.0005 for ceramics and Teflon. The effective conductivity of a dielectric material at high frequency is:

Equation 98
$$\sigma = 2\pi f\varepsilon_0\varepsilon_r\tan(\delta)$$

The shunt conductance per length of an interconnect when dipole relaxation dominates, is:

Equation 99
$$G_L = 2\pi f\tan(\delta)C_L$$

As a worst case, the frequency corresponding to the bandwidth of the signal can be used to estimate the high frequency conductivity of the material.

Lossy Transmission Line Model

In the lossless transmission line model, only the distributed capacitance (C) and inductance (L) of the interconnect is considered:

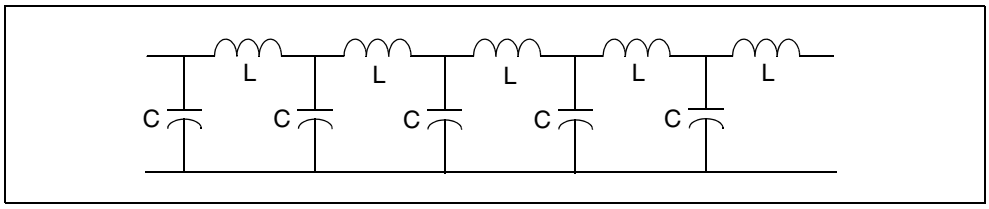


Figure 118 Lossless Transmission Line Model

In the lossy transmission line model, the series resistance and dielectric conductance are introduced into the equivalent circuit model:

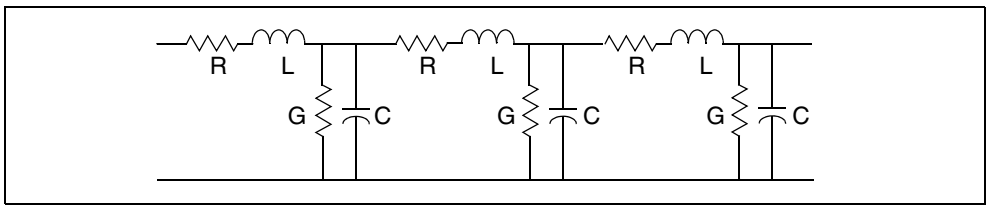


Figure 119 Lossy Transmission Line Model

These four circuit elements, normalized per unit length, can be used to describe all the high frequency properties of a transmission line. When the equivalent circuit equation is solved in the frequency domain, the characteristic impedance is modified to:

Equation 100 $$Z_0 = \sqrt{\frac{R_L + j\omega L_L}{G_L + j\omega C_L}}$$

and the propagation phase term, γ, is:

Equation 101 $$\Upsilon = \alpha + j\beta = \sqrt{(R_L + j\omega L_L)(G_L + j\omega C_L)}$$

In the propagation phase term, β is related to the phase velocity by:

Equation 102 $v = \frac{\omega}{\beta}$

To first order when $R_L << \omega L_L$ and $G_L << \omega C_L$, the characteristic impedance and phase velocity (v) are unchanged from their lossless values. However, a new term, the attenuation per length (α) is introduced.

The attenuation per length is approximately:

$$\alpha = \frac{1}{2}\left(\frac{R_L}{Z_0} + G_L Z_0\right) \quad \text{[nepers/length]}$$

$$\alpha = 4.34\left(\frac{R_L}{Z_0} + G_L Z_0\right) \quad \text{[dB/length]}$$

The total attenuation (αd) determines the fraction of the signal amplitude that remains after propagating the distance (d). When α has the units of dB/length, the fraction of signal remaining is:

Equation 103 $10^{\left(\frac{-\alpha d}{20}\right)}$

A 2 dB attenuation in a signal corresponds to a final amplitude of 80% of the original and 6 dB attenuation corresponds to a final amplitude of 50% of the original. Attenuation on the order of 6 dB significantly changes the signal integrity.

These topics are discused in the following sections:

- Attenuation Due to Conductor Resistance
- Attenuation Due to the Dielectric
- Integrating Attenuation Effects

Attenuation Due to Conductor Resistance

In the typical case of a 50 ohm transmission line, the attenuation per length due to just the series resistance is

$$\alpha = 0.09 R_L \quad \text{[dB/length]}$$

When the resistance per length is of the order of 0.2 ohm/inch or less as is the case in typical printed circuit boards, the attenuation per length is about 0.02 dB/inch. Typical interconnect lengths of 10 inches yields only 0.2 dB, which

would leave about 98% of the signal remaining. Using the lossless T-element to approximate most applications provides a good approximation.

However, in fine line substrates, such as in the previous section, the resistance per length can be on the order of 2 ohms/inch. In such a case, the attenuation is on the order of 0.2 dB/inch. So a 10 inch interconnect line then has an attenuation on the order of 2 dB, which would leave only about 80% of the signal. This is large enough that its effects should be included in a simulation.

Attenuation Due to the Dielectric

When the dielectric completely surrounds the conductors, the attenuation due to just the conductance per length of the dielectric is:

$$\alpha_{dielectric} = 2.3f\tan(\delta)\sqrt{\varepsilon_{eff}} \quad \text{[dB/inch, f in GHz]}$$

The worst case and highest attenuation per length is exhibited by FR4 boards with tan(δ) of the order of 0.02 and a dielectric constant of 5. The attenuation at 1 GHz is about 0.1 dB/inch. For an interconnect 10 inches long, this is 1 dB of attenuation, which would leave about 90% of the signal remaining, comparable to the attenuation offered by a conductor with 1 ohm/inch resistance.

If the resistance per length is larger than 1 ohm/inch— for example in cofired ceramic and thin film substrates, and the dissipation factor is less than 0.005, the attenuation from the conductor losses can be on the order of 10 times greater than dielectric loss. In these applications, the dielectric losses can be ignored.

Integrating Attenuation Effects

All of the first-order effects of attenuation are automatically simulated with the U-element.

With `ELEV=1`, the inputs can be the cross sectional geometry and the material properties of the conductor, bulk resistivity (RHO), the relative dielectric constant of the insulation (KD), and the conductivity of the dielectric (SIG). From these features, the equivalent capacitance per length, inductance per length, series resistance per length, and conductance per length are calculated during simulation.

With `ELEV=2`, you can use estimates, measurements or third-party modeling tools to directly input the equivalent capacitance per length, inductance per length, series resistance per length, and conductance per length.

Simulation automatically generates a model for the specified net, composed of a series of lumped elements that resembles the model for a lossy transmission line. The parameter `WLUMPS` controls the number of lumped elements included per wavelength, based on the estimated rise time of signals in the simulation.

The attenuation effects previously described are a natural consequence of this model. The U-element allows realistic simulations of lossy transmission lines in both the AC and the transient domain.

References

[1] *Handbook of Electronics Calculations for Engineers and Technicians*, McGraw Hill, pages 18-29, 18-23.

B

Time Domain Reflectometry (TDR)

This appendix discusses using digitized TDR in conjunction with HSPICE to select design components.

Optimizing Time Domain Reflectometry (TDR) Packaging

Time domain reflectometry (TDR) is the closest measurement to actual digital component functions. It provides a transient display of the impedance versus time for pulse behavior.

These topics are covered in the following sections:

- Using TDR in Simulation
- TDR Optimization Procedure
- Performing a TDR Simulation in HSPICE

Using TDR in Simulation

When you use a digitized TDR (Time Domain Reflectometry) file, you can use the HSPICE or HSPICE RF optimizer to automatically select design components. To extract critical points from digitized TDR files, use the `.MEASURE` statement, and use the results as electrical specifications for optimization. This process eliminates recurring design cycles to find component values that curve-fit the TDR files.

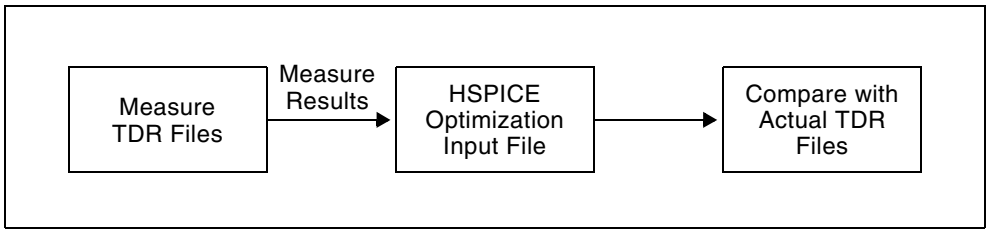


Figure 120 Optimization Process

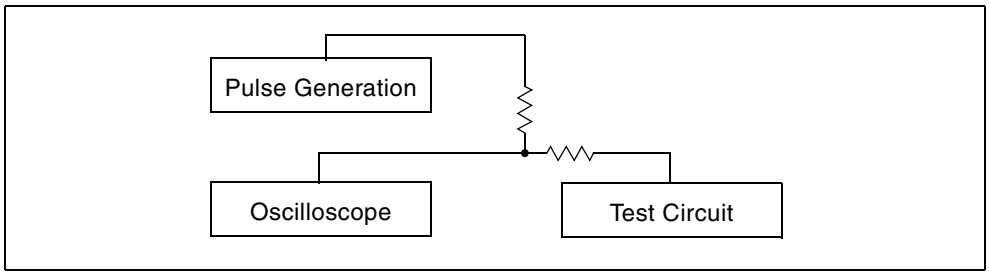


Figure 121 General Method for TDR Optimization

Use the following method for realistic high-speed testing of packaging:

- Test fixtures closely emulate a high-speed system environment.

A HSPICE device model uses ideal transmission lines and discrete components for measurements.

The tested circuit contains the following components:

- Signal generator.
- Coax connecting the signal generator to ETF (engineering test fixture) board.
- ETF board.
- Package pins.

Package body.

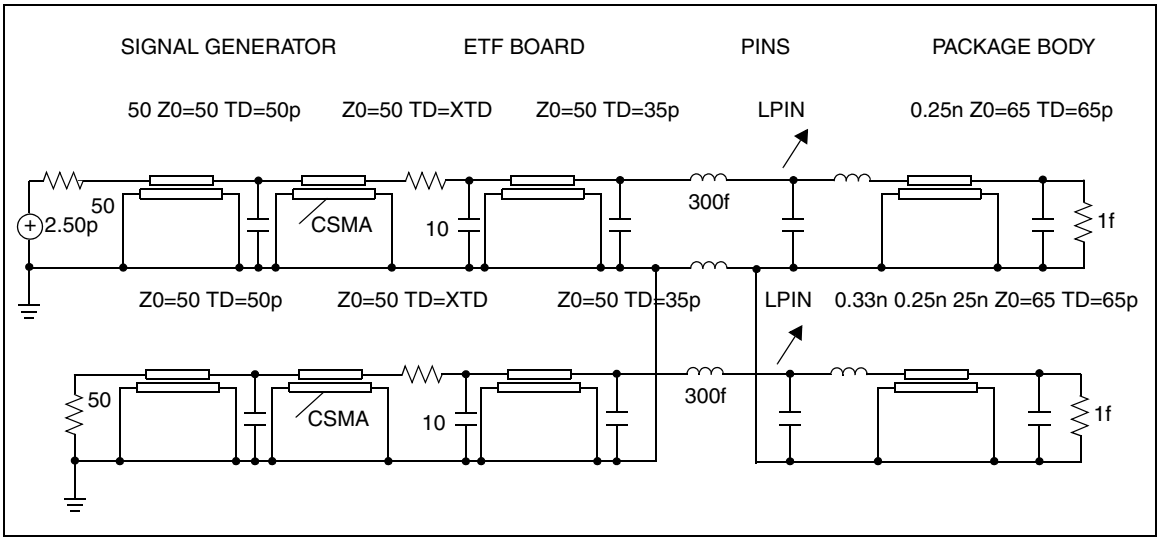


Figure 122 SPICE Model for Package-Plus-Test Fixture Optimized Parameters: XTD, CSMA, LPIN, and LPK

The package tests use a digital sampling oscilloscope to perform traditional time-domain measurements. Use these tests to observe the reflected and transmitted signals. These signals are derived from the built-in high-speed pulse generator and translated output signals into digitized time-domain reflectometer files (voltage versus time).

Use a fully-developed SPICE model to simulate the package-plus-test fixture, then compare the simulated and measured reflected/transmitted signals.

The next section shows the input netlist file for this experiment. Figure 123 through Figure 126 show the output plots.

TDR Optimization Procedure

The sample netlist for this experiment is located in the following directory: $*installdir*/demo/hspice/si/ipopt.sp

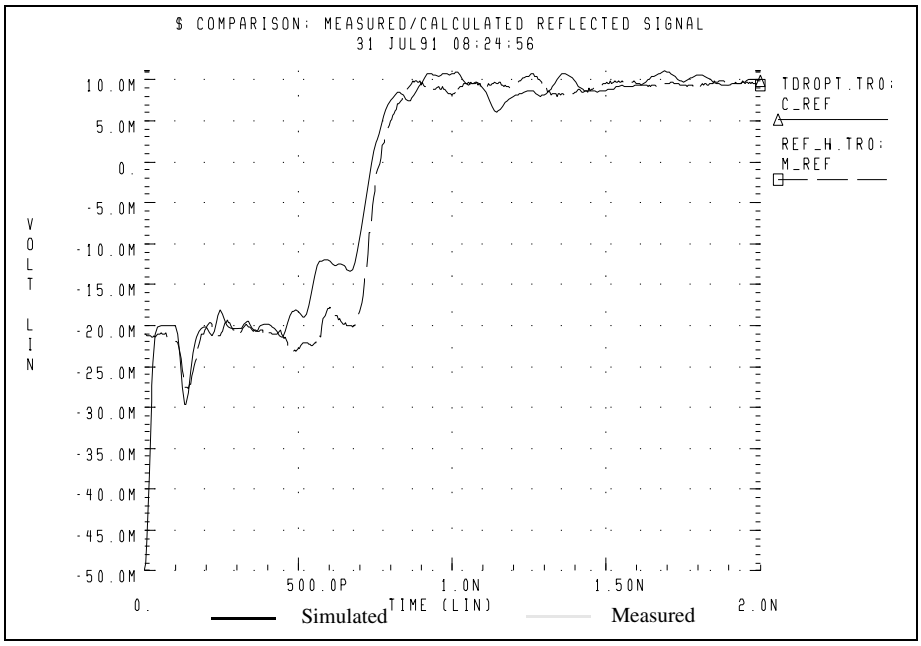


Figure 123 Reflected Signals Before Optimization

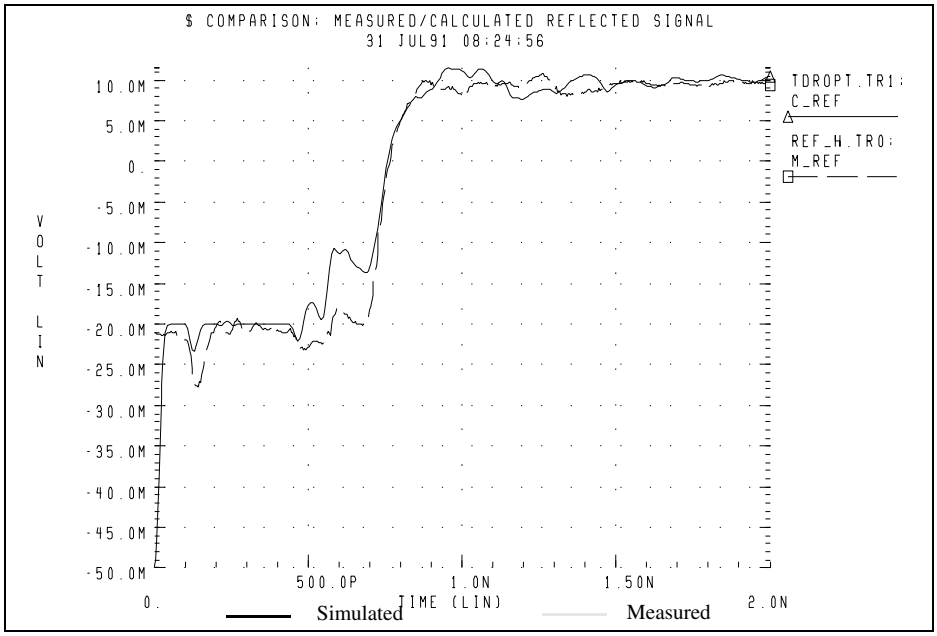


Figure 124 Reflected Signals After Optimization

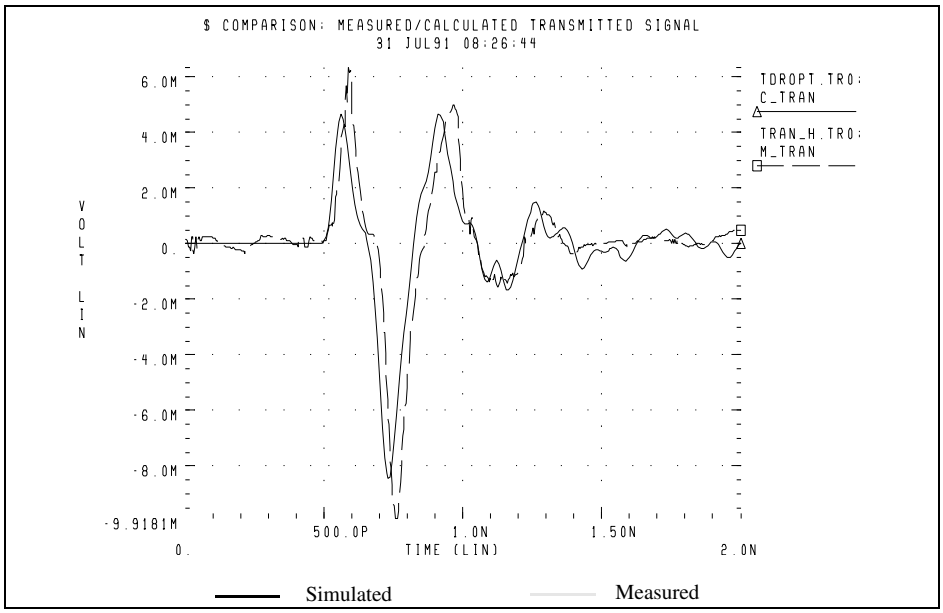


Figure 125 Transmitted Signals Before Optimization

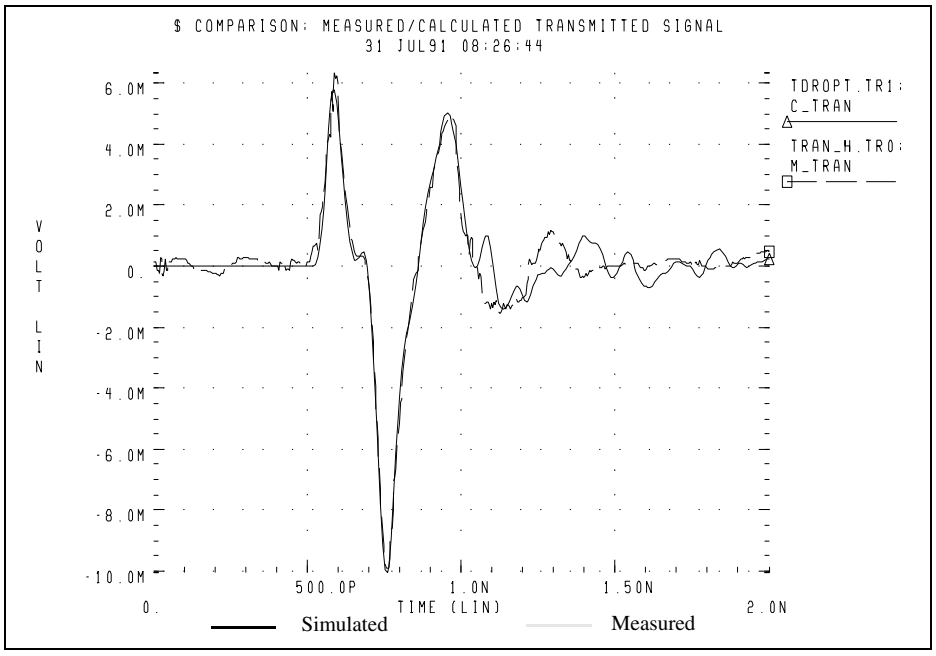


Figure 126 Transmitted Signals after Optimization

Performing a TDR Simulation in HSPICE

When performing a signal integrity analysis, a bit stream is often the first stimulus used to validate the system. If poor results are obtained, it can be useful to first perform a TDR simulation on either the system or the individual models to find unexpected impedance discontinuities or mismatches. A TDR simulation measures the reflections that result from an ideal step edge traveling through a transmission medium.

The following is an example HSPICE testbench to demonstrate TDR simulations with a variety of classic discontinuities. The included testbench is completely functional and uses single-ended lossless transmission lines and lumped elements to demonstrate the analysis but could be easily adapted to use more complex W-element or S-parameter models. A differential equivalent example netlist is supplied in the last section.

Application Example: TDR Simulation

A time domain reflectometry (TDR) simulation measures the reflections that result from an ideal step edge traveling through a transmission medium. This allows you to analyze impedance discontinuities and mismatches. The medium being studied is often an interconnect topology, and could be a combination of circuit board traces, cables, connectors and even the wire and bonding between an IC and its package. The following uses an example HSPICE testbench to demonstrate TDR simulations with a variety of classic discontinuities.

HSPICE provides a number of methods to model transmission lines and media. These include:

- Lumped models with R-, L-, and C-elements connected by lossless (T-element) transmission lines
- S-parameter data (S-element), either from instrument measurements or extracted during previous HSPICE simulations
- The frequency-dependant lossy W-element. The W-element has several different modeling options:
  - The HSPICE field-solver based on materials and geometries
  - RLGC per unit length matrices
  - S-parameter data called by the W-element SMODEL parameter
  - A network of behavioral sources

An interconnect simulation often contains a combination of these methods. Singularly or collectively you can refer to them as the Device Under Test (DUT). It is common for the DUT model, or portions of it to be provided by the manufacturer (as in the case of a connector) or be output by a signal integrity analysis tool which creates a model by analyzing the packaging or layout data.

The following sections discuss these topics:

- Basic TDR Impedance Analysis
- Testbench Netlist Overview
- Input Output Ports
- Selecting the Risetime of the Incident Wave
- Simulating the Example DUTs
- Differential TDR Example Netlist
- Reference

Basic TDR Impedance Analysis

The resulting waveform of the TDR simulation is the combination of the incident wave and reflections that occur when the step edge encounters impedance variations.

For this example and netlist, the following names and equations are used:

- Vincident: An ideal copy of the step edge source used as a reference
- Vmeasured: Voltage at the input to the DUT which is the combination of incident and reflected waves
- Vreflect: The reflected portion only which is obtained by subtracting the incident portion from the measured voltage
- Impedance (Zo): Calculated from the above parameters:

$$\frac{Z_{ref} \cdot \ (V_{incident} + V_{reflect})}{(V_{incident} - V_{reflect})}$$

Using this calculation, you can graph the response of the transmission line in units of characteristic impedance which is often more useful than plotting the voltage of the TDR measured wave. In at least one case, however, the open circuit termination, it is more useful to plot the measured voltage. Note that due to the voltage divider effect, the measured voltage is 1/2 of the incident wave. The following examples demonstrate classic impedance mismatches and discontinuities.

Example 1: Short Circuit Termination

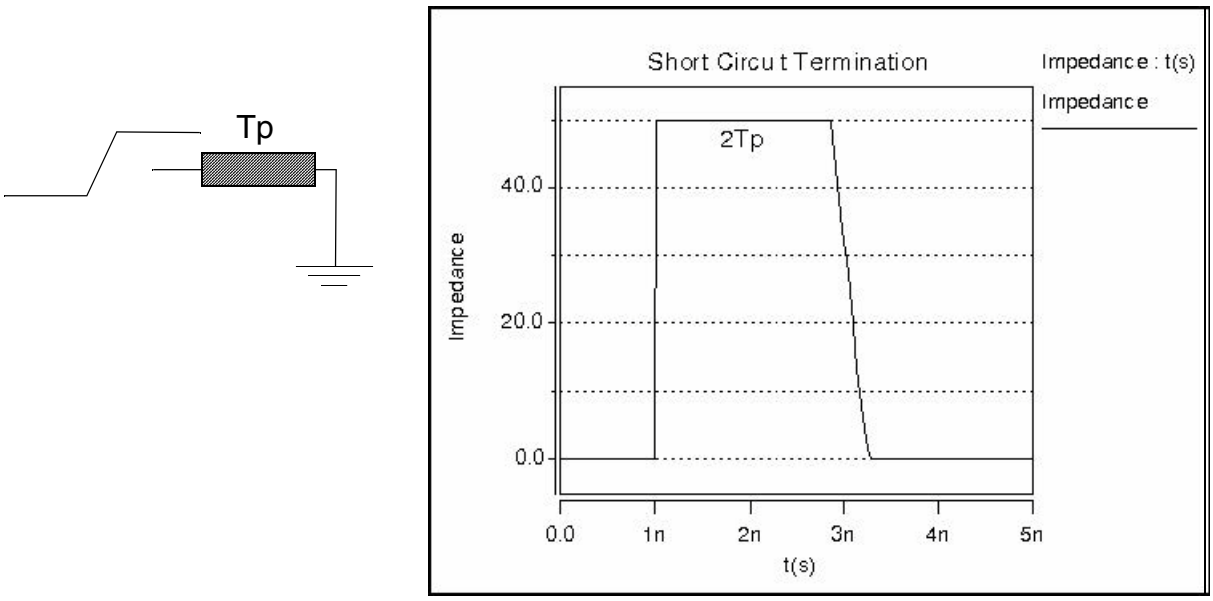


Figure 127 Short Circuit Termination

Example 2: Open Circuit Termination

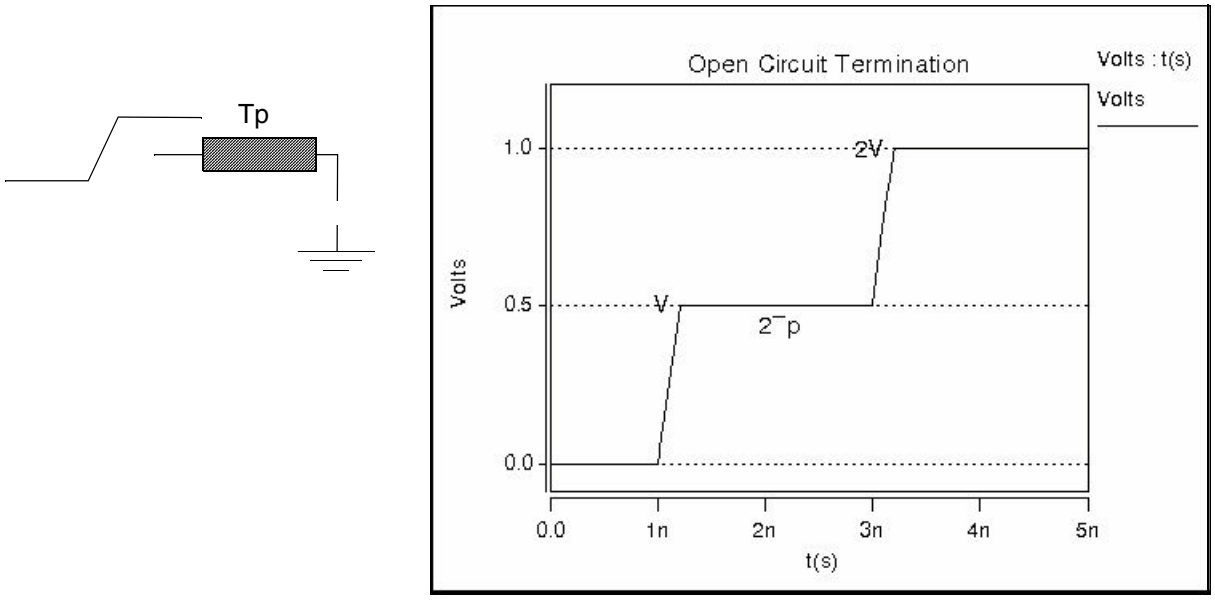


Figure 128 Open Circuit Termination

Example 3: Mismatched Load Termination

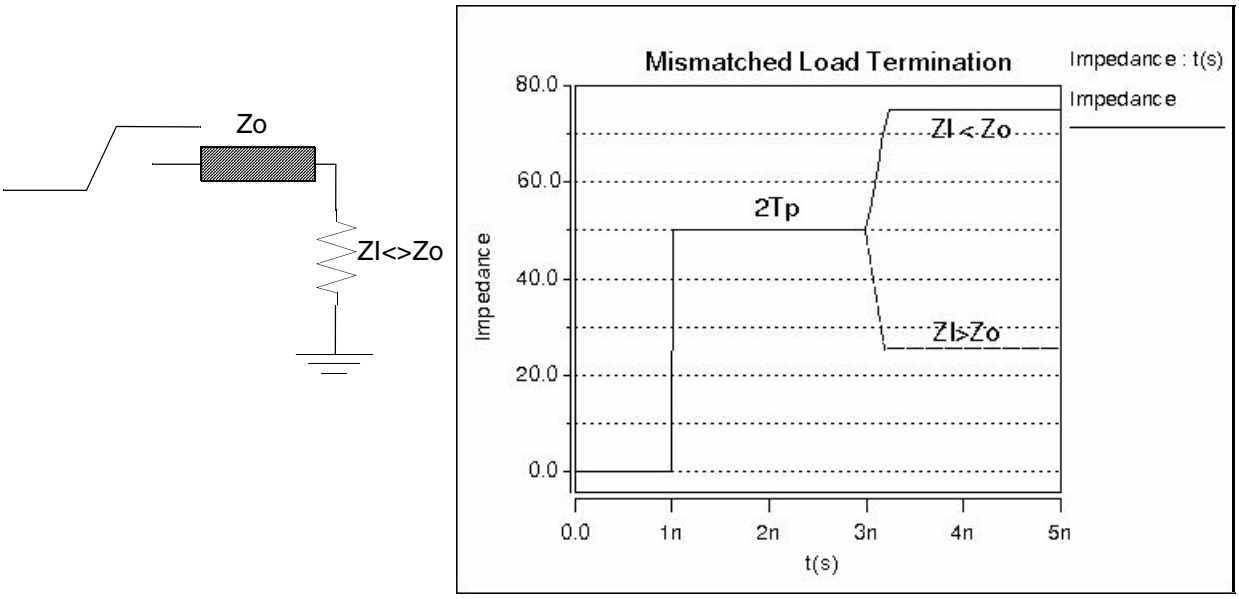


Figure 129 Mismatched Load Termination

Example 4: Shunt Capacitance Discontinuity

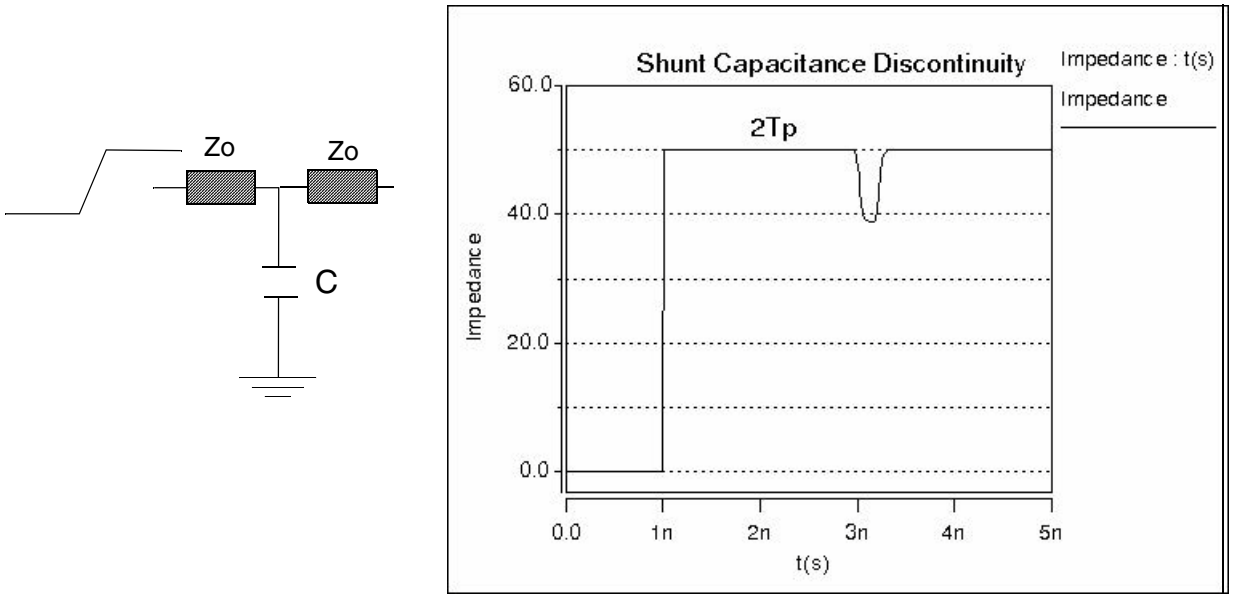


Figure 130 Shunt Capacitance Discontinuity

Example 5: Series Inductance Discontinuity

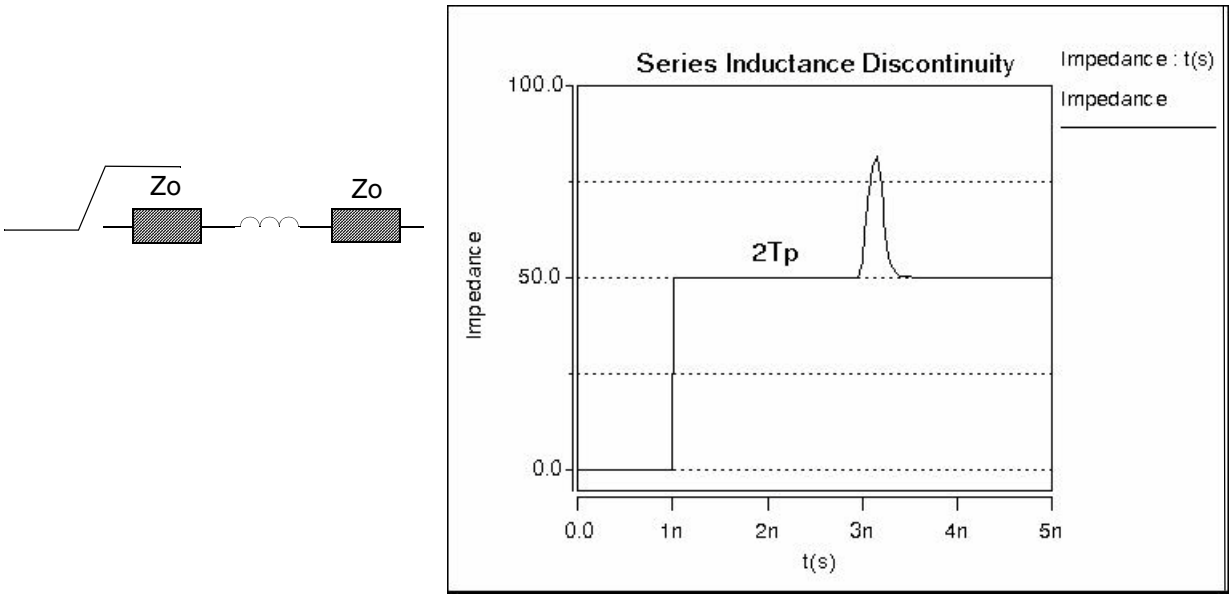


Figure 131 Series Inductance Discontinuity

Testbench Netlist Overview

In creating an HSPICE TDR test bench, this example uses the “P” (port) element for both the input sources and output terminations because it can:

- Represent either single-ended or differential signals, and
- Provide both a source voltage and impedance value in a single construct

The first example simulates a single-ended TDR example. Additional syntax for a differential testbench and DUT is provided below in Differential TDR Example Netlist.

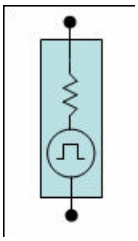
The testbench is composed of four main sections:

1. The port representing the source voltage and impedance for the incident wave
2. The Device Under Test (DUT)
3. An output termination port
4. A reference port and matching termination port

The function of the reference port is to create an ideal version of the incident pulse which is isolated from the DUT and is used to calculate the reflected portion of the TDR measurement.

Input Output Ports

The P element is the basis for the input and output ports. As described above, it can be visualized as an impedance and a voltage source in series:



```
P1 input1 0 z0=50 port=1 pulse (0 1.5 100p 40p 40p)
```

Figure 132 HSPICE Port element

In the example shown in Figure 132, a port named “P1” connects between “input1” and ground and has a characteristic impedance of 50 ohms. It supplies a single 40ps rising edge to input1 after a 100ps delay. If you omit the voltage source argument, the port serves as a simple termination. Each port must have a unique sequential numerical designation in the “port=” argument. The P-element is also used in the .LIN analysis to extract S-parameters.

For more information on port elements see Port Element in the *HSPICE User Guide: Simulation and Analysis*.

Selecting the Risetime of the Incident Wave

One way to determine the risetime of the incident wave is using the knee frequency. To guarantee reliable operation of a digital system, you should develop and verify the circuit design for frequencies below the knee frequency. It can be shown that most energy in digital pulses concentrates below the knee frequency and that the behavior of a circuit at the knee frequency determines its ability to process a step edge. The knee frequency for any digital signal is related to the rise and fall time of its digital edges rather than its clock rate.

The risetime can be calculated based the desired knee frequency and depends on whether it is based on a 10-90% or 20-80% measurement.

- For a 10-90% measurement, Trise = .5/Fknee
- For a 20-80% measurement, Trise = .35/Fknee

For example, the risetime measured at 20-80% needed for a 10GHz knee frequency:

$$Trise = \frac{.35}{10e9Hz} = 35ps$$

In an HSPICE trapezoidal pulse source function, the risetime is specified from 0-100%.

So for a 20-80% risetime, multiply the desired risetime by 1.67 to obtain the “tr” parameter and for a 10-90% measured risetime, multiply the desired risetime by 1.25. For example:

$$(35ps \text{ measured at } 20\text{-}80\%) \cdot 1.67 = \text{tr parameter} = 58.5ps$$

Simulating the Example DUTs

The sample netlist includes three separate example DUTs:

1. An inductive discontinuity
2. A capacitive shunt discontinuity
3. A transmission line followed by a shunt capacitance discontinuity, an impedance mismatch in the second transmission line and a series inductance

The following example shows a combination of behaviors. The netlist for simulation is shown below Figure 133. The results are shown in Figure 134 on page 377.

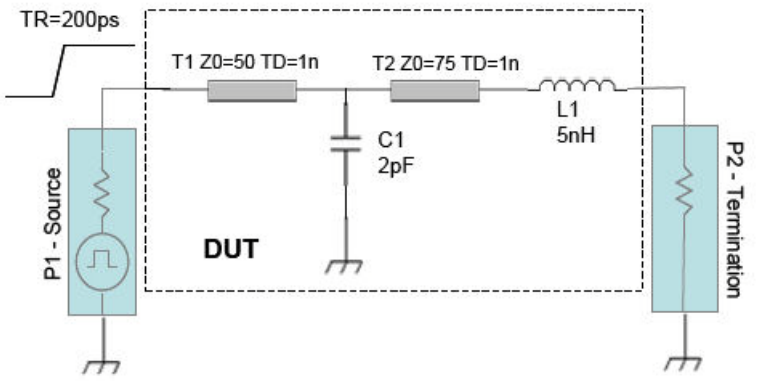


Figure 133 Testbench and DUT

The example test bench below, “HSPICE TDR Netlist with DUT3 Uncommented”, is completely functional and uses simple lossless transmission lines and lumped elements to demonstrate the analysis, but it could just as easily use more complex W-element or S-parameter models. A differential example follows at the end of this section, and contains appropriate modifications to the ports, DUTs and equations.

Example HSPICE TDR Netlist with DUT3 Uncommented

```
** Single-ended TDR example **
*
.opt post probe runlvl=5
.tran 0.1n 20n
*
.param zref=50.0 vlo=0 vhi=1 td=1n tr=.2n tf=.2n pw=50n per=200n
*
* P1 is the input port for the incident wave. It provides
* both an impedance and a pulse source
*
Psource dut_in 0 zo=zref port=1 pulse(vlo vhi td tr tf)
*
* DUT1 - An inductive discountinuity
*T1 dut_in 0 node1 0 Z0=50 td=1n
*L1 node1 node2 5e-9
*T2 node2 0 dut_out 0 Z0=50 td=1n
*
* DUT2 - A capacitive shunt discountinuity
*T1 dut_in 0 node1 0 Z0=50 td=1n
*C1 node1 0 1e-12
*T2 node1 0 dut_out 0 Z0=50 td=1n
*
* DUT3 - Series inductance - shunt capacitance
* with impedance mismatch in the second transmission
* line
T1 dut_in 0 node1 0 Z0=50 td=1n
C1 node1 0 2e-12
T2 node1 0 node2 0 Z0=75 td=1n
L1 node2 dut_out 5e-9
*
* Pterm is the output port of the DUT and provides termination
*
Pterm dut_out 0 zo=50 port=2
*
* Pref and Prefterm are a reference source and termination to use
* as the ideal incident wave in the reflection calculation
*
Pref vref 0 zo=zref port=3 pulse(vlo vhi td tr tf)
Prefterm vref 0 zo=zref port=4
*
* Calculations of the reflected wave and impedance
*
.probe tran vincident = par('v(vref)')
.probe tran vmeasured = par('v(dut_in)')
.probe tran vreflect = par('vmeasured-vincident')
.probe tran Zo = par('zref*(vincident+vreflect)/(vincident-
vreflect)')
```

```
*
.end
```

Procedural Notes:

1. The default `RUNLVL` should be increased from `3` to `5` to observe the subtleties in the discontinuities of the waveforms.
2. Alternately, you can specify:
   `.option runlvl=0 accurate delmax=10p`
   Since the pulse is a single event (non-repeating), you can omit the pw (pulse width) and `per` (period) arguments, although a warning message results. Since `delmax` forces the maximum time step length, it may sacrifice the efficiency of the dynamic time step control. Therefore, unless you have a specific need for dense time points, normally it is recommended that you do not specify `delmax`.
3. Uncomment only one DUT at a time to observe the example discontinuities and impedance mismatches.

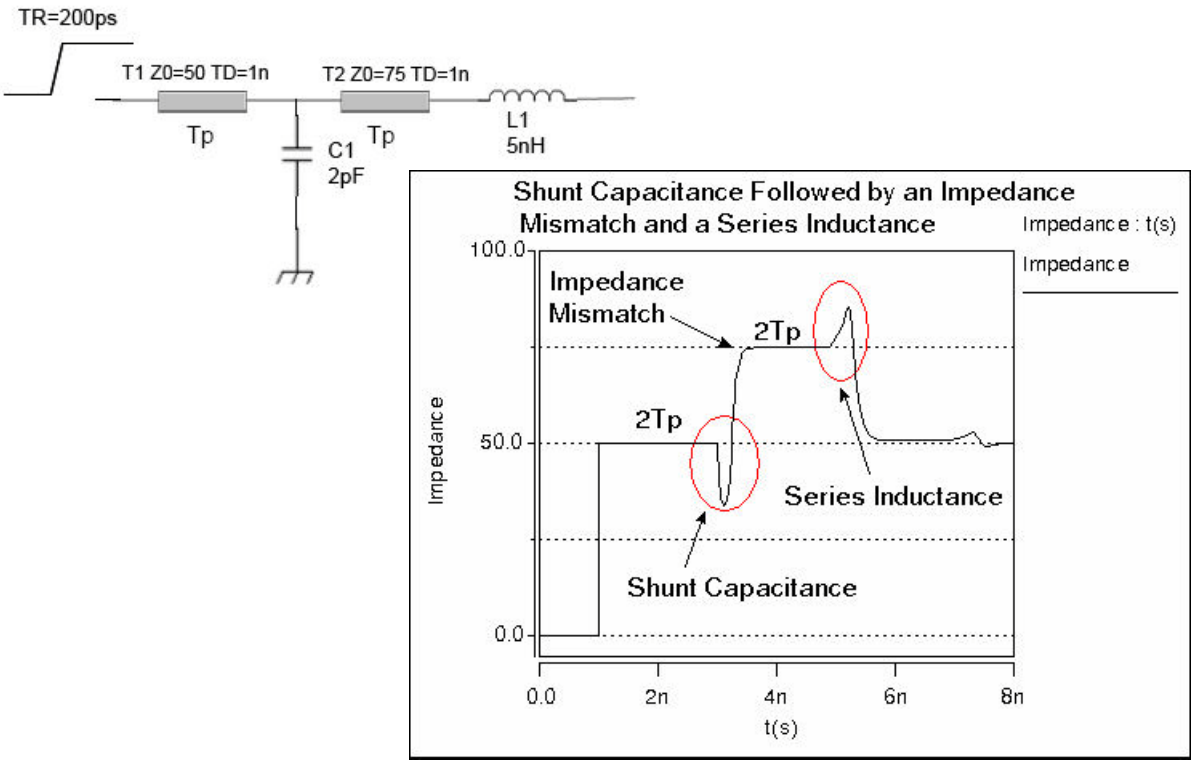


Figure 134 Result of the DUT3 Analysis

Differential TDR Example Netlist

```
** Differential TDR example **
*
.opt post probe runlvl=5
.tran 0.1n 20n
*
.param zref=50.0 vlo=0 vhi=1 td=1n tr=.2n tf=.2n
*
* P1 is the differential input port for the incident wave. It
provides
* both an impedance and a pulse source
*
Psource dut_inp dut_inn 0 zo=zref port=1 pulse(vlo vhi td tr tf)
*
* DUT1 - An inductive discountinuity
* T1P dut_inp 0 node1p 0 Z0=50 td=1n
* T1N dut_inn 0 node1n 0 Z0=50 td=1n
* L1 node1p node2p 5e-9
* L2 node1n node2n 5e-9
* T2P node2p 0 dut_outp 0 Z0=50 td=1n
* T2N node2n 0 dut_outn 0 Z0=50 td=1n
*
* DUT2 - A capacitive shunt discountinuity
* T1P dut_inp 0 node1p 0 Z0=50 td=1n
* T1N dut_inn 0 node1n 0 Z0=50 td=1n
* C1 node1p 0 1e-12
* C2 node1n 0 1e-12
* T2P node1p 0 dut_outp 0 Z0=50 td=1n
* T2N node1n 0 dut_outn 0 Z0=50 td=1n
*
* DUT3 - A shunt capacitance followed by an impedance mismatch
* and a series inductance
T1P dut_inp 0 node1p 0 Z0=50 td=1n
T1N dut_inn 0 node1n 0 Z0=50 td=1n
C1 node1p 0 2e-12
C2 node1n 0 2e-12
T2P node1p 0 node2p 0 Z0=75 td=1n
T2N node1n 0 node2n 0 Z0=75 td=1n
L1 node2p dut_outp 5e-9
L2 node2n dut_outn 5e-9
*
* Pterm is the dut_output port of the DUT and provides termination
*
Pterm dut_outp dut_outn 0 zo=50 port=2
*
* Pref and Prefterm are a reference source and termination to use
as the
* ideal incident wave in the reflection calculation
```

```
*
Pref refp refn 0 zo=zref port=3 pulse(vlo vhi td tr tf)
Prefterm refp refn 0 zo=zref port=4
*
* Calculations of the reflected wave and impedance
*
.probe tran vincident = par('v(refp)-v(refn)')
.probe tran vmeasured = par('v(dut_inp)-v(dut_inn)')
.probe tran vreflect = par('vmeasured-vincident')
.probe tran Zdiff = par('2*zref*(vincident+vreflect)/
(vincidentvreflect)')
*
.end
```

Reference

TDR Impedance Measurements: A Foundation for Signal Integrity Copyright © 2001, Tektronix, Inc.

Index

A

AC resistance transmission lines 285
accuracy, W element tabular model 136
analog, circuit simulation of a digital system 3
analysis, Monte Carlo 165
AT1 model parameter 300
ATLEN model parameter 300

B

bandwidth 344
branch matrix 298
buffer 59, 182, 233
  differential pins 223
  ECL
    input 195
    input/output 199
    output 196
    tristate 197
  IBIS 177
  input 183, 184
    example 227
  input ECL 195
  input/output 193
    open drain 195
    open sink 195
    open source 195
    syntax 191
  input-output ECL 199
  open
    drain 194
    sink 194
    source 194
  output 185, 186
    example 227
  output ECL 196
  series 202
  series switch 202
  terminator 200
  tristate 188, 189
  tristate ECL 197
buffer, IBIS keyword 209
buffers
  creating 228
  IBIS 233

C

c_com_pu | c_com_pd | c_com_pc | c_com_gc, IBIS keyword 220
capacitance
  circuit 298
  matrixes 316
  multiconductor 297
CAPL model parameter 300
CEXT model parameter 280, 290, 292, 294
characteristic impedance 338
circuits
  Signetics drivers 22
  Xilinx FPGAs 26
CLEN model parameter 300
clock frequency 344
CMULT model parameter 280, 290, 291, 294
coax models 322
coax models*See also* transmission lines
component keyword 230
components
  IBIS 228
  IBIS buffers 233
COMPUTE_TABLE keywyord 152
conductance multiconductor 297
convolution integral 60
CORKD model parameter 281, 290
coupled line noise simulation example 31
creating buffers 228
crosstalk 343

D

D_IBIS option 223
D12 model parameter 293
data smoothing, S- and W-element 82
data smoothing, S-model 82
decoupling methods 4

DELAY model parameter 300
delays
  causes 3
  problems and solutions 3
  transmission lines 301
DELEN model parameter 300
design
  high speed, problems and solutions 3
  partitioning 5
detect_oti_mid, IBIS keyword 222
differential, common mode waves 40
DLEV model parameter 277, 280, 286, 289, 293

E

element
  transmission line 269
element parameters
  U Elements 314–315
element parameters, transmission lines
  T-element 270
  U-element 275
ELEV model parameter 277, 280
errors, timestep control error in transmission lines 326
example
  input buffer 227
  output buffer 227
  transmission line 227, 269

F

field programmable gate arrays 26
file, IBIS keyword 208
files, keyword 230
FPGA's 26
FR1 model parameter 300
frequency table model 73
fwf_tune | rwf_tune, IBIS keyword 215

G

golden parser (IBIS) 178
ground
  bounce 28
  plane, transmission lines 267

H

HGP model parameter 281, 290, 291, 294
hsp_ver, IBIS keyword 212
HSPICE
  ground for transmission lines 284
  integration method 286
HT model parameter 280, 289
HT1 model parameter 289

I

IBIS 233
  buffers 228
  component 228
  differential pins 223
  keywords
    optional 207
  models 178
  optional keywords 230
  required keywords 230
  scale parameters 218
  terminology 182
  V-T and I-V curves 218
IBIS buffers 59
IBIS keywords
  buffer 209
  c_com_pu | c_com_pd | c_com_pc | c_com_gc 220
  detect_oti_mid 222
  file 208
  fwf_tune | rwf_tune 215
  hsp_ver 212
  interpol 212
  model 208
  nowarn 220
  pd_scal | pu_scal | pc_scal | gc_scal | rwf_scal | fwf_scal 218
  power 212
  ramp_fwf | ramp_rwf 214
  rm_dly_rwf|rm_dly_fwf|rm_tail_rwf|rm_tail_fwf 219
  rwf_pd_dly | fwf_pu_dly 217
  ss_state 219
  typ 210
  xv_pu | xv_pd 213
ideal and lossy models 264
ideal T-element transmission lines 269
ideal transmission line 268

impedance 337
  versus time 6, 363
inductance 340
  matrixes 316
input buffer 183, 184
  ECL 195
  example 227
input/output
  drivers 22
  ECL buffer 199
  open
    drain buffer 195
    sink buffer 195
    source buffers 195
input-output ECL buffer 199
integration
  algorithms 307, 326
  method 286
interconnect
  analyzing 2
interpol, IBIS keyword 212

J

junction diodes, line-to-line 113

K

KD model parameter 280, 290, 291, 294
KD1 model parameter 290
KD2 model parameter 290
keywords
  component 230
  file 230
  IBIS optional 207
  optional 230
  package 230
  pkgfile 231
  required 230

L

LEVEL model parameter 277, 280
LLEV model parameter 277, 280
lossless
  parameter combinations 301
  transmission lines 268
    model 335
lossy
  transmission lines 268, 275
    model 336
  U model 276
lumped parameters 285

M

matrix
  rational function 63
matrix properties 102
MAXD model parameter 281
Maxwell matrix 299
MEASURE statement, using TDR files with 364
microstrip transmission line 286
mixed-mode S-parameter 37
mixed-mode S-parameters 42
model parameters
  transmission lines
    loss parameters 282
    measured parameters 300
    planar models 280
    precomputed parameters 295
  transmission lines, geometric coax models 291
  U models 280
    basic ELEV parameters 300
    common planar parameters 290
    ELEV = 1 parameters 294
    measured parameters 300
  U models, geometric coax parameters 292
  U models, geometric twinlead parameters 280
MODEL statement, using U model with 276
model, IBIS keyword 208
models
  S-parameter 36
multiconductor capacitance/conductance 297
multiplier function, U and T Elements 267
multiplier parameter, U and T Elements 267
multiport passive and lossy circuits, thermal noise 68
multi-terminal network 35
multi-terminal network, S-parameter 44
mutual inductance 341

N

ND model parameter 280
network
  multi-terminal 35

NL model parameter 289
NLAY model parameter 282, 289
noise
  coupled line 31
  problems and solutions 3
  quota 4
  sources 3
nowarn, IBIS keyword 220

O

O.C.G. matrix 298
OD1 model parameter 294
optimization, TDR 6, 363, 365
optional keywords, IBIS 230
options, D_IBIS 223
oscillation, from simulation errors 326
output
  buffer 185, 186
    example 227
  ECL buffer 196
output ECL buffer 196
overlay dielectric transmission line 287

P

package keyword 230
pd_scal | pu_scal | pc_scal | gc_scal | rwf_scal | fwf_scal, IBIS keywords 218
PEC 151
pkgfile keyword 231
planar conductor parameters 286
PLEV model parameter 277, 278, 280, 300
power, IBIS keyword 212
printed circuit boards 2

R

RA model parameter 291
RA1 model parameter 293
ramp_fwf | ramp_rwf, IBIS keyword 214
rational function
  file format 63
rational function matrix 63
RB model parameter 291
RD model parameter 291
recursive convolution 60
reference
  plane, transmission lines 267
required keywords, IBIS 230
RHO model parameter 282, 290, 291, 294
RHOB model parameter 282, 290, 291, 294
rise time 344
RISETIME option 118
RISETIME option, setting 118
RLGC model definition, W_Element 81
RLGC tabular model, COMPUTE_TABLE keyword 152
rm_dly_rwf|rm_dly_fwf|rm_tail_rwf|rm_tail_fwf, IBIS keywords 219
rwf_pd_dly | fwf_pu_dly, IBIS keyword 217

S

S model
  data smoothing 82
S model syntax 52
scale parameters, IBIS 218
scattering (S) parameters 44
scattering parameter element 44
S-element 44
  data file model examples 65
  recursive convolution 60
  syntax 45
  transmission line 78
self inductance 342
series buffer 202
series switch buffer 202
SHTHK model parameter 292, 294
SIG
  conductivity 354
  model parameter 282, 290, 291, 294
SIG1 model parameter 290
signal integrity 3
Signetics FAST I/O drivers 22
simulation
  circuits
    with Signetics drivers 22
    with Xilinx FPGAs 26
  ground bounce example 28
  signal integrity 2
skin effect frequency 285
small-signal parameter data-table model 73
S-model
  data smoothing 82

SP model 73
SP model parameter 280, 290
SP12 model parameter 290
S-parameter 44
  mixed mode 37
  mixed-mode 42
  S-element 44
s-parameter
  transmission line 80
S-parameter model 36
spatial extent of leading edge 349
ss_state, IBIS keyword 219
stripline transmission line 287
syntax
  S model 52
  SP model 73
syntax, S-element 45
systems, simulating 2

T

T Elements 264
  lossless transmission line model 353
  multiplier function M 267
  transient effects modeling 340
TDR (time domain reflectometry) 6, 363, 364, 365
terminator buffer 200
TH model parameter 280, 289
TH1 model parameter 289
THB model parameter 280, 290
thermal noise 68
THK1 model parameter 289
THK2 model parameter 289
thresholds, voltage 203
time
  domain reflectometry 6
  versus impedance 6, 363
time domain reflectometry 363
timestep
  control
    error 326
  control error 326
  dynamic control 120
  static control 120
transmission line GUI utility 172
transmission lines
  AC resistance 285
  analysis guidelines 349
  attenuation 353
    calculations 351
    effects 359
  bandwidth 344
  branch capacitances 298, 299
  capacitance
    definitions 297
    matrix 316
    to ground 285
  CEXT parameter 297
  characteristic impedance 336, 338
  Cjk symbol 297
  clock frequency 344
  coax
    example 322
    models 291
  common ground inductance 340
  conductance definitions 297
  conductor
    length 297
    resistance 358
    width 297
  coupled
    lines 274
    microstrips example 324
  crosstalk 340, 343
  CX symbol 297
  CXY symbol 297
  delay too small warning 267
  delays 301
  dielectric loss 356
  dispersion 354
  dissipation 356
    factor 354
  dual dielectric 286
  effects 5
  elements 264
    selection 266
  example 227, 269
  geometric parameters 282
  Gjk symbol 297
  GPR parameter 297
  ground plane 267
  GX symbol 297
  GXY symbol 297
  HSPICE
    ground 283
    reference plane 267

ideal 273
  properties 268
impedance 337
  characteristic 338
  lumped elements 338
  mismatch 331
inductance 340
  matrix 316
  mutual 341
  self 342
integration
  algorithms 307
  method 326
interconnect properties 265
LLEV parameter 297
loss
  calculations 351
  factor 302
lossless model 335
lossy model 276, 336, 357
  properties 268
  selector 277
  statement 276
lumped
  parameters 285
  RC model 268
  RLC model 336
matrixes 298, 299, 316–318
measured parameters 300
microstrip 286
model selection 264
models 264
  selection 266
multiple
  conductors 297
  stripline, example 303
mutual inductance 341
number of lumps 311, 327
  computation 327
  default 327
  specification 327
oscillation problems 326
overlay dielectric 287
parameter
  error 283
  physical 281
planar
  conductors 286
  models 280
precomputed parameters 295
  keywords 296
printed circuit boards, example 320
reflection calculations 350
resistive termination 78
ringing 286, 326–330
rise time 344
S Element 78
sea of dielectric 286
  example 309
self inductance 342
SIG conductivity parameter 354
signal reflections 331
simulation 2
skin depth 355
  calculation 351
skin effect frequency 285
source properties 265
stripline 287
switching noise 340
system model 112
theory 335
timestep control errors 326
total conductor resistance 315
transient impedance effects 340
twinlead
  example 324
  models 292, 324
U Element 337
U-element 275
tristate
  buffer 188, 189
  ECL buffer 197
TS model parameter 280, 290
TS1 model parameter 294
TS2 model parameter 294
TS3 model parameter 294
twinlead models 324
twinlead *See* transmission lines
two-port noise parameter data 70
TxLine Tool 172
typ, IBIS keyword 210

U

U Elements 264
  attenuation effects 359
  examples 302

lossy transmission line model 337, 354
multiplier function M 267
physical parameters 280
ringing problems 326
statement 275
transient effects modeling 340
transmission line
loss parameters 282
model 343
U model statement 276
U models
applications 319
conductor
height 280
resistivity 282
spacing 280
thickness 280
width 280
dielectric
conductivity 282
configurations 277, 279
constant 280
electrical specification format 277
ELEV parameter 277, 286
external capacitance 280
model selector 277, 280
.MODEL statement 276
number of
conductors 280
layers 282
lumps 281, 285
parameters 277, 280–302
geometric coax 291
geometric twinlead 292
perturbation
multiplier 281
parameter 280
PLEV parameter 278
reference plane
configuration 277
height 281
resistivity 282
separation distance 280
thickness 280
selection 277
syntax 276
transmission line type selector 277
using the IBIS component 228

V

volltage, current pairs 40
voltage thresholds 203
voltage, current pairs 40
voltage, current waves-nodal waves 38
VREL model parameter 300

W

W element
tabular model accuracy 136
W Elements
RLGC model definition 81
warnings, T-line delay too small 267
WD model parameter 280, 289
WD1 model parameter 289
W-element
TxLine Tool 172
wire 5
model selection
chart 266
WLUMP model parameter 281

X

xv_pu | xv_pd, IBIS keyword 213
XW model parameter 280, 290

Z

ZK model parameter 300