LineSim User’s Guide

HyperLynx

HyperLynx
14715 NE 95th Street, Suite 200
Redmond, WA 98052
support@hyperlynx.com

Table of Contents

Chapter 1: Installation and Licensing

Summary

This chapter describes:

- what kind of PC you need to run LineSim
- how LineSim's licensing works (node-locked and floating)
- how to install LineSim
- how to install the HyperLynx floating-license server application *(for floating-license customers only)*
- additional information about running with a floating license
- how to install LineSim on a remote network computer

System Requirements

Operating Systems

LineSim runs under Windows 95, Windows 98, and Windows NT 4.0 or greater. It does not run under Windows 3.1/3.11 or Windows NT 3.51.

Special Requirement for Installing under Windows NT

IMPORTANT! When you install under Windows NT, you must be logged in as an NT "administrator" (or "supervisor") in order for the hardware-protection key driver to install properly. If you receive an error message when the hardware key's driver attempts to install, it is most likely because you are not logged on with administrator privileges (only administrators are allowed to install drivers).

If this occurs, LineSim's installation program allows you to finish installing the HyperLynx software, then gives you an easy way of installing just the key driver later, after you've logged in as an administrator. See "If the Hardware-Key Driver Fails to Install" below in this chapter for details.

Video

LineSim works with SVGA-or-greater video resolution (i.e., 800x600 or higher). At VGA resolution (640x480), some of LineSim's dialog boxes are too large to fit on the screen. You may still be able to run the program in VGA resolution by moving certain dialog boxes side-to-side as needed.

Memory

LineSim runs on any PC with at least 16 Megabytes of RAM (real, physical memory).

Note: *It is possible that LineSim may not run on a 16-Megabyte machine if most of the memory is used by other applications. If you get "out of memory" errors from LineSim, try closing other open applications, or freeing more memory for Windows to use.*
Another sign that LineSim is low on memory is excessive use of the hard disk. If the hard disk is accessed whenever you perform an operation in LineSim, then Windows is running LineSim out of virtual memory (i.e., off the hard disk rather than out of physical memory), which slows LineSim considerably. Again, free more memory for LineSim to use.

Hard-Disk Space

LineSim itself requires about 40 Megabytes of hard-disk space for installation. Most of this space is used by IC models.

Mouse

LineSim requires a mouse. Portions of its user interface cannot be accessed from the keyboard. Any mouse that runs in Windows works with LineSim.

Parallel Port for Hardware Key

LineSim requires a hardware key to run. If you have a node-locked version of the software, the hardware key plugs into a parallel port on your computer. If you have floating-license version, the key plugs into the computer you use as a HyperLynx license server. (For details on node-locked versus floating licenses, see “About Licensing” below.)

If you have multiple parallel ports, you can plug the hardware key into any of the ports, and LineSim will find it.

Note: *In rare cases under Windows NT, having multiple ports will slow the querying of the key. In these instances, you can tell the key driver to skip ports — see “If Running Windows NT with Multiple Ports and the Key Polls Slowly” below in this chapter.*

LineSim uses the same key that all of HyperLynx’s products use. If you purchase additional HyperLynx products (node-locked or floating), you will be given one or more license codes that enable your key for the additional products.

LineSim hardware key usually works fine in series with a printer, scanner, etc., although you may need to power up the device to get proper key operation. In rare cases, the key will not work with a peripheral device attached, and you will have to move the device to a separate parallel port.

Note: *Generally, the closer LineSim’s key is to the computer, the more reliably it operates. If you have multiple hardware keys on a parallel port and LineSim cannot find its key, try changing the order of the keys and moving LineSim’s closer to the PC.*

About Licensing

How HyperLynx's Licensing Works

LineSim uses a programmable hardware key which allows HyperLynx to license you on a per-product basis. You can use the key on a single computer ("node-locked"), or on a designated "server" computer that grants licenses (when available) over the network to client computers ("floating license").

When You First Receive LineSim

In most cases, when you first receive LineSim, it is *pre-licensed* to run the options you have purchased. In that case, you do not need to enter any license codes: just plug the hardware key into a parallel port of the appropriate computer (your computer or a designated server computer, depending on which type of licensing you purchased), and run.

However, sometimes the product cannot be pre-licensed. If you purchased a HyperLynx product and receive an error message saying you are not licensed to run it, contact HyperLynx or your local reseller to obtain a license code. Usually, the code will be sent by fax or e-mail.

Note: *There is a way of checking without running LineSim which features you are licensed for; see "Reviewing What Features You are Licensed For" below in this chapter for details.*

Node-Locked versus Floating Licensing

HyperLynx supports two types of licensing:

- **Node-locked licensing -** Restricts the HyperLynx applications to running on a single computer (whichever computer has the hardware key attached)
- **Floating (or "networked") licensing -** Allows HyperLynx applications to run from any computer on the network, as long as the applications are not being run on too many other computers; the key remains attached to a designated "server" computer

The primary advantage of floating licensing compared to node-locked is convenience. If you have several users who, at different times, want to run a HyperLynx application, floating licensing allows them to do so without having to transport the hardware key from computer to computer. It is also very easy to add floating licenses if your user demand increases: additional licenses require no extra hardware keys.

Note: *If you have a node-locked copy of a HyperLynx application and want to convert it to a floating license, contact HyperLynx or your local representative.*

Client versus Server Computers (Floating Licenses Only)

In a floating-license scenario, the computer on the network which has the hardware key attached to it and which grants licenses to other computers is called the "server." The other computers on the network which request to run HyperLynx applications are called "clients."

You can have more than one HyperLynx license server on your network. You might want to have several servers, for example, in case one server computer crashes or is removed from the network. (If you order an additional floating license from HyperLynx or your local representative, you're given the choice of activating it on your existing hardware key or on a new key destined for a new server.)

A server can grant licenses to itself, i.e., you can run HyperLynx applications even on a server computer. This might be convenient in a small engineering group where one of the engineers who runs HyperLynx applications also wants his computer to function as the server.

HyperLynx license servers can run on any Windows 95, Windows 98, or Windows NT (version 4.0 or later) computer on the network. To be a HyperLynx server, the designated computer does not have to be a "server" in any other sense; "ordinary" PCs can act as HyperLynx servers.

Installing LineSim (Node-Locked or Floating Client)

This section describes how to install LineSim (and other HyperLynx application programs) for a node-locked computer or floating-license *client* computer.

Note: *A separate section below describes how to install the HyperLynx License Server software, for a floating-license* ***server*** *computer. See "Installing the Floating-License Server."*

The HyperLynx CD-ROM contains a file called "INSTALL.TXT" in the root directory of the CD; this file contains the most-recent installation instructions. The file "README.TXT" contains information about new features in the version of software you're installing.

What You're Installing

HyperLynx Applications

A complete LineSim installation consists of one or both of the following components, depending on what kind of licensing (node-locked or floating) you purchased:

"LineSim/BoardSim / analysis options"	Installs the LineSim base product (plus BoardSim, the EMC option, and the Crosstalk option, too, if you purchased them)
HyperLynx License Server	Installs HyperLynx's license server, which serves floating licenses to "client" computers; only needed if you purchased a floating license

You need to install the HyperLynx License Server application only if you purchase one or more floating licenses (see "Installing the HyperLynx License Server" and "Licensing a Floating-License Client Computer" below for more details).

Hardware-Key Driver (Windows NT Only)

Windows 95 and Windows 98

Under Windows 95 and Windows 98, no driver is required for the hardware key.

Windows NT V4.0 or Greater

The hardware-key driver is required under Windows NT V4.0 and greater. Normally, installation and start-up of the driver is completely automatic; it happens transparently as part of the HyperLynx-application installation.

However, if you are not logged on as an “administrator,” the driver cannot install. In this case, the remainder of the software will finish installing, then you can easily return (after logging in with administrator privileges) to install only the driver — see “If the Hardware-Key Driver Fails to Install” below for details.

Steps for Installing the Software

Windows 95, Windows 98, and Windows NT 4.0 have an “AutoPlay” feature that will automatically start applications when a CD-ROM is inserted into a drive. Most Windows computers have this feature enabled.

However, it is possible that AutoPlay is not enabled on your system for security reasons. AutoPlay will also be disabled if you hold down the shift key when the CD-ROM is inserted.

If Windows AutoPlay is enabled on your computer: ***(normal installation)***

1. Insert the HyperLynx CD-ROM into your CD-ROM drive.
2. After a short period, the HyperLynx Install program will run automatically and open the HyperLynx Install dialog box.
3. Click the Install HyperLynx Software button. This will install LineSim (and any other HyperLynx applications you have purchased). Follow the on-screen instructions until this portion of the installation is complete.

4. If you want to view HyperLynx's manuals online and you do not already have a copy of Adobe's Acrobat viewer (version 3.0 or greater) installed on your computer, click the Install Adobe Acrobat button. Follow the on-screen instructions until this portion of the installation is complete.

5. Optionally, click the View Readme and View License Agreement buttons to display this information.

6. Installation is now complete. Click Exit.

If Windows AutoPlay is disabled on your computer:

1. Insert the HyperLynx CD-ROM into your CD-ROM drive.

2. Using the Windows Explorer, double-click the Setup application (SETUP.EXE) on the CD-ROM and the HyperLynx Install program will run automatically and open the HyperLynx Install dialog box.

3. Continue with step 3 in the "If Windows AutoPlay is enabled" section just above.

If the Hardware-Key Driver Fails to Install (Windows NT Only)

Under Windows NT, only users with "administrator" privileges can install device drivers. Accordingly, if you install the HyperLynx software but are not logged on as an administrator, the portion of the installation that installs the hardware-key driver will fail ("You need System Administrator privileges…"). However, you can easily finish installing the other portions of the software (PCB-layout translators, Acrobat reader, etc.), then re-install just the key driver after you've logged back on with the required privileges.

To install just the hardware-key driver after a failed driver installation:

1. Log on to Windows NT with administrator privileges. (See your network administrator if you don't know how to log on as an administrator.)

2. From the HyperLynx CD-ROM, run the HyperLynx installation program (using AutoPlay or by running SETUP.EXE). The HyperLynx Install dialog box opens.

3. Click the Configure HyperLynx License button. The HyperLynx Licensing dialog box opens.
4. Click the Install Sentinel Device Driver button. This runs just the hardware-key portion of the installation.

If you do not have the HyperLynx CD-ROM immediately accessible, you can run the key-driver installation program directly. From the Windows Explorer, look in the SENTINEL subdirectory under your HyperLynx installation, and double-click on the file NT_KEY.EXE. This executes the key installation directly.

Testing the LineSim Installation

General Test

To test the installation:

1. *If you are running a node-locked copy of LineSim*, make sure the hardware key is plugged into a parallel port on your PC.

   If you are running a floating-license copy, make sure the HyperLynx License Server and hardware key are installed on a PC, and that both your PC and the server PC are running on the network. Also, be sure that the HyperLynx License Server application is running on the server PC.
2. Choose the HyperLynx Simulation Software icon from the HyperLynx group on the Start menu (Start/Programs/HyperLynx).
3. LineSim should open, ready to load a .TLN file (i.e., a schematic file).

If a Node-Locked Hardware Key is Not Working *(Node-Locked Only)*

For a node-locked copy of LineSim, the following are symptoms of LineSim’s hardware key not working correctly:

♦ a “hardware key” or “copy protection” error message

- ♦ the program opens, but most of its menus are not visible, and there is no way to load a schematic file (".TLN" file)

If either of these occur, run the steps in "Testing the Hardware-Protection Key" below.

If the Program's Menus and Buttons are Missing Features, and You Can't Load a Schematic *(Node-Locked or Floating Client)*

If you open LineSim, but get only a limited set of menus and toolbar buttons (e.g., only the Notepad and Help buttons on the toolbar), LineSim is unable to find valid licensing. Specifically, if your copy is node-locked, then this symptom means that the hardware key is unplugged or not functioning correctly. If your copy is floating, this symptom means that LineSim cannot find any HyperLynx license servers.

Testing the Hardware-Protection Key *(Node-Locked or Floating Server)*

If LineSim does not run properly and you suspect that the hardware-protection key may be the cause, you can test the key and its driver.

To test the hardware key:

1. From the HyperLynx CD-ROM, run the HyperLynx installation program (using AutoPlay or by running SETUP.EXE). The HyperLynx Install dialog box opens.

2. Click the Configure HyperLynx License button. The HyperLynx Licensing dialog box opens.

3. Note the status message in the Hardware Protection Key area:
   If the key is functioning properly, the message says "A hardware protection key was found."
   If the key is not functioning, the message says "No hardware protection key was found."

4. To run diagnostics on the key, click the Test Hardware Key button. A program supplied by the key vendor runs; click the Find Keys button to run the vendor tests.

If you do not have the HyperLynx CD-ROM immediately accessible, you can run the hardware-key test program directly. From the Windows Explorer, look in the UTIL subdirectory under your HyperLynx installation, and double-click on the file FIND32.EXE. (This is the vendor-supplied test program.)

If Running Windows NT with Multiple Parallel Ports and the Key Polls Slowly *(Windows NT Only)*

If you are running under Windows NT and your computer has multiple parallel ports, it is possible that the hardware-protection key driver will run slowly, as it polls from port to port looking for the key. If you experience “sluggish” operation, you can tell the driver not to waste time looking at unused ports.

To instruct the hardware key driver not to poll unused ports:

1. From the Windows Explorer, look in the SENTINEL\WIN_NT subdirectory under your HyperLynx installation, and double-click on the file SETUPX86.EXE. The Rainbow Technologies Sentinel dialog box opens (this program is provided by the key vendor).
2. From the Functions menu, choose Configure Sentinel Driver. A dialog box opens.
3. Click the Help button to read about how to disable polling of unused ports.

Reviewing What Features You are Licensed For

If your installation is working, but certain features are not available (for example, an analysis option you thought you were licensed for), you can check for which features you are currently licensed.

To check what features you are licensed for:

1. From the HyperLynx CD-ROM, run the HyperLynx installation program (using AutoPlay or by running SETUP.EXE). The HyperLynx Install dialog box opens.
2. Click the Configure HyperLynx License button. The HyperLynx Licensing dialog box opens.

3. Click the Show Licenses button. The Licensing dialog box opens; it lists all of the features for which you are currently licensed, and their status.

If you do not have the HyperLynx CD-ROM immediately accessible, you can run the license-checking program directly. From the Windows Explorer, look in the UTIL subdirectory under your HyperLynx installation, and double-click on the file SPVIEW.EXE.

If You Receive a “DOS Error” Message

If you attempt to run LineSim under Windows NT and receive the error message “unexpected DOS error: 11” (or a similar message), your version of Windows NT is out-of-date. LineSim requires Windows NT V4.0 or later.

Laptops Must Have Parallel Port Powered Up

If you are running LineSim on a laptop computer, be sure that the parallel port is powered up. Some laptops will default to shutting down the parallel port in order to reduce power consumption. For details on the symptoms that may occur if the port is not powered, see the sections above.

Installing the Floating-License Server

This section describes how to install the HyperLynx Floating-License Server program. This installation is required only if you purchased a floating license, and happens only on the machine which you want to act as the server of HyperLynx licenses. On client computers (the ones that will actually run the software), you only need to install the HyperLynx application programs (not the license server); see “Installing LineSim” above for details.

Steps for Installing the License Server

Windows 95, Windows 98, and Windows NT 4.0 have an “AutoPlay” feature that will automatically start applications when a CD-ROM is inserted into a CD-ROM drive. Most Windows computers have this feature enabled.

However, it is possible that AutoPlay is not enabled on your system for security reasons. AutoPlay will also be disabled if you hold down the shift key when the CD-ROM is inserted.

If Windows AutoPlay is enabled on your computer: ***(normal installation)***

1. Insert the HyperLynx CD-ROM into your CD-ROM drive.
2. After a short period, the HyperLynx Install program will run automatically and open the HyperLynx Install dialog box.
3. Click the Configure HyperLynx License button. The HyperLynx Licensing dialog box opens.
4. In the Floating License area, click the Install Server button. This will install the license server, including installation (for Windows NT; not required for Windows 95 or 98) of the hardware-protection key driver. Follow the on-screen instructions until the installation is complete.

If Windows AutoPlay is disabled on your computer:

1. Insert the HyperLynx CD-ROM into your CD-ROM drive.
2. Using the Windows Explorer, double-click the Setup application (SETUP.EXE) on the CD-ROM and the HyperLynx Install program will run automatically and open the HyperLynx Install dialog box.
3. Continue with step 3 in the “If Windows AutoPlay is enabled” section just above.

Testing the License Server Installation

General Test

To test the installation:

1. Make sure the hardware key is plugged into a parallel port on the server PC.
2. Choose the HyperLynx License Server icon from the HyperLynx License group on the Start menu (Start/Programs/HyperLynx License).

3. The License Server should open in a dialog box.

Testing the Hardware-Protection Key

If the license server fails to run and the error appears to be related to the hardware-protection key, read section “Testing the Hardware-Protection Key” (and following) above to test the key.

Testing the Server with Client Computers

To test that the server (once it is running on the server PC) can serve licenses to client computers, see section “Licensing a Floating-License Client Computer” below.

Licensing a Node-Locked Computer

When you first receive your HyperLynx software, your licensing is pre-configured; you should be able to install the software and begin using it without needing to enter any licensing code. Use the license editor only when you need to enter a new license code you have been faxed from HyperLynx, or to check the status of your current licensing.

On a node-locked computer, the HyperLynx hardware key must be attached to a parallel port. Licensing is for the local computer only; other computers on the network cannot run HyperLynx applications. (If you are interested in a “floating” license which can be shared by multiple computers on the network, contact HyperLynx or your local representative.)

Using the License Editor

To open the licensing editor:

1. From the Options menu, choose Licensing.

The next sections describe information displayed in the licensing editor.

User Name

The first time you run LineSim after entering a new license code, the program queries you for the name of the licensed user. The user name is “burned” into the hardware key; from then on, it “travels” with the key. **This is a one-time-only operation, so take care to enter the user name correctly the first time.**

Note: *The user name appears on all print-outs from LineSim.*

Products and License Status

A list box in the licensing editor shows the current status for all HyperLynx products. Status information includes:

- product name
- kind of license
- expiration date of the option’s license

There are two kinds of licenses:

- **full license:** when the expiration date is passed, the software keeps running; HyperLynx or your local representative will contact you about purchasing another year’s worth of product maintenance
- **trial license:** when the expiration date is passed, the software stops running

Key Serial Number

The licensing editor shows the serial number of your hardware key. If you ever need to know your key number (for example, when you contact HyperLynx for technical support), looking in the license editor is easier than looking physically on the key.

Entering a New License Code

HyperLynx license codes are 12-digit hexadecimal numbers. The codes are case insensitive.

To enter a new license code:

1. In the Licensing dialog box, type the code number into the License Code edit box.

The box only accepts valid, HyperLynx-generated license codes; if you type a random code, LineSim rejects it with an error message. Codes are hardware-key-specific, i.e., if you try to license a key with a code generated for a different key, the program will reject the code and report an error.

Note: *Once a license code has been successfully entered, it is not displayed in the dialog box. The license resides in the hardware key; the code is no longer needed.*

You can re-enter a license code for a key that has already been programmed with the same license; the key will simply be re-programmed.

Licensing a Floating-License Client Computer

With floating licensing, your computer acts as a “client” that requests permission to run HyperLynx applications from a “server” computer on the network. The HyperLynx hardware key is attached to the server computer. If a license is available for the HyperLynx application you request, the server grants you one and you’re able to run; if no licenses are available (because other computers on the network are using them), you have to wait until a license becomes available (when another user is finished using it).

The licensing is administered by the server computer; as a client, you simply try to access a license. In order to do this, your computer must be able to “see” one or more HyperLynx server computers on the network.

Connecting Your Client Computer to HyperLynx Server(s)

To make setting up floating licenses from a client computer as simple as possible, HyperLynx applications are designed to automatically detect and connect to HyperLynx servers on the network. Therefore, you may never need to use the License Servers dialog box. The dialog box gives you a way of seeing which servers are available, and of testing your connections to them, if you ever need to.

Each time a HyperLynx application executes on a client computer, it polls the network for one or more HyperLynx license servers. If a HyperLynx server with an available license is found on the network, the application will continue to run. If no server or no available license is found, the application will run but without any useful features.

The HyperLynx License Servers dialog box lets you configure which computers on your network will be queried when you try to run HyperLynx products from your client machine. The License Servers dialog box shows which HyperLynx Servers are available on the network, and lets you test your connections to those servers.

To open the HyperLynx License Servers dialog box:

1. From the Options menu, choose Licensing.

The next sections describe information displayed in the dialog box.

Note: *If Options/Licensing opens the "Licensing" dialog box instead of the HyperLynx License Servers dialog box, then LineSim has found a node-locked key on your local PC. The HyperLynx License Servers box opens only if no key is present on your PC.*

HyperLynx License Servers List

The HyperLynx License Servers area lists up to four HyperLynx Servers that LineSim has found on the network. If no servers are listed, then none were successfully found. Often, there will only be one server; only if you purchase multiple floating licenses and request more than one key will multiple servers exist.

The server names are filled in automatically by LineSim. Normally, you do not need to type in the names, although you might do so to test your connection to a server that is not being found automatically. Server computers will be queried for licenses in the order in which they are listed. If Server1 cannot be contacted or does not have a license available for checkout, then Server2 will be queried; if Server2 fails, then Server3 is queried; and so forth.

The Lock Server Name check boxes to the right of each server name are normally unchecked, so that automatic detection of servers occurs. If you have trouble with automatic detection or want to by-pass the short time delay associated with it, you can enable these check boxes; see “Manually ‘Locking’ a Server Connection” below for details.

Testing the Connection to a Server

To test the connection to a server:

1. Find the server name in the list of servers. If you are attempting to test the connection to a server that is not listed, type the server’s name in the first available data box (see below for a description of valid server names).

2. Click the Test button next to the server’s name. If the server is found, a dialog box will open with a list of the licenses available from the server. If the server cannot be found, then after a brief pause, a “could not access” error will appear.

A successful test will re-establish the connection to a server whose connection has been “broken.” (Servers which “go down” are automatically marked as such on client computers and will not be queried for license requests until the connection is re-established.)

About Server Names

Each server has a unique name depending on the name by which the server computer is “known” on the network. Usually, the server is found for you and the name filled in automatically.

To determine a server’s exact name:

1. On the server computer, maximize the HyperLynx License Server application. (If the Server application isn’t running on the server, start it.) You may want to ask your network administrator to do this for you.

2. Look at the Server application’s title bar. The server’s name appears after the words “HyperLynx License Server on \\”. The name begins immediately after the ‘\\’ characters.

For example, on a server computer named “DEFIANT”, in the title bar of the HyperLynx License Server you would see “HyperLynx License Server on \\DEFIANT”. The proper server name is “DEFIANT” (not “\\DEFIANT” - no slashes).

If a Server Connection Fails

Test the connection to the server as described above. If the test fails, verify that:

- the server name is correct in your client machine’s HyperLynx License Servers dialog box (see above for how to determine the proper name)

- the HyperLynx License Server application is running on the server computer

- your network is functioning properly; a good test is to verify that the client computer can see and mount shared drives on the server computer

If all of these items are verified but you still can’t get the server connection to work, check with your network administrator that the client and server machine’s networking configuration is properly set up. HyperLynx floating licenses uses “mailslot” network technology, which is a central mechanism in Windows networking.

Manually “Locking” a Server Connection

In rare cases, LineSim may fail to automatically detect a server, even though the server connection works correctly if the server’s name is manually specified. If this occurs, you can manually enter the server's name, then "lock" it so that LineSim always uses that server, and does not rely on automatic detection to find it.

To manually lock a server connection:

1. Type the server's name into one of the License Server boxes.
2. Verify that the manual connection works by clicking the Test button (see "Testing the Connection to a Server" above for details).
3. Click on the Lock Server Name check box next to the Test button.
4. Exit LineSim and restart for the change to take effect.

Now, LineSim will automatically use the locked server, and not rely on automatic polling to find it. Automatic detection will still occur for any other License Server boxes whose Lock check box is not enabled.

You can also use the locking feature to speed up LineSim's licensing. E.g., you may see a brief delay when the program is first started while it polls for servers. To eliminate this, enter at least one server's name and lock it, and also lock the other, empty boxes. This will prevent LineSim from polling at all.

Note that servers are tried in the order listed in the dialog box, from 1 to 4. If there is a server with more features or faster access that you want checked first for available licenses, manually put it first in the list.

Important! *Licenses for features on servers that are manually added and locked will not be available for use until LineSim is exited and restarted.*

If A Server Stops Running

If a HyperLynx server ceases operation (e.g., the server computer goes down or the HyperLynx License Server application is stopped), the client computer automatically disables its connection to that server. (The client remains connected to other HyperLynx servers, if others are present on the network.) A client can re-enable a server connection by either re-starting the HyperLynx application (i.e., closing and re-running LineSim), or by executing a successful "test" operation as described above. If a server's features change (e.g., new licenses codes are entered on it), the new features will only be recognized on a client computer after restarting the HyperLynx application on the client machine.

User Name

When you first run LineSim with a floating license, a dialog box appears asking for your User Name. This name appears on hardcopy printouts. It can be up to 20 characters long.

Thereafter, the User Name appears in the HyperLynx License Servers dialog box. You can change it on a particular client computer at any time.

To change your User Name:

1. From the Options menu, choose Licensing. The HyperLynx License Servers dialog box opens.
2. In the User Name area, type the new name.

Installing to Multiple Client Computers from a Network Hard Disk

If you have a floating-license copy of LineSim and want to install it easily on multiple client computers, you can copy the installation CD-ROM to a shared network hard drive and run installation from the network drive, or “share” the CD-ROM drive and access the CD-ROM directly from remote computers.

Installing on a Remote Network Computer

LineSim is normally installed on the computer that will be running the applications, but it is possible to install it on a remote host system. The following paragraphs describe how.

Summary of steps for installing HyperLynx applications to run on a remote computer:

1. Install the HyperLynx application software on the remote computer

2. Create a local directory for storing a local HyperLynx .INI file
3. Create a shortcut to the remote HyperLynx application
4. Add a command-line option to the shortcut that specifies the local .INI file

Detailed steps:

1. Install LineSim on the remote host computer. Share the installation directory so that remote computers can access it. Remote users will not need "write access" to this directory; in fact, it is wise to share the directory as "read only."

2. On the local client computer, create a HyperLynx working directory, for example "C:\HYPERLYNX". Create a subdirectory beneath this directory for user data, typically "HYPFILES".

3. Create a "shortcut" on the local client computer that points to the executable file "BSW.EXE" on the remote host computer. A simple way to do this is to use the Windows Explorer: right-click on "BSW.EXE" in the shared host directory, then select "Create Shortcut". Since the shared host directory is read-only, Windows will create the shortcut on the local, client desktop.

4. Specify where the client's local version of BSW.INI will be located; this is done with a command line option as follows: edit the shortcut's properties (right-click on the shortcut and select Properties) and add "-INIF:DRIVE:\<DIRECTORY>\BSW.INI" to the Target line.

   For example:
   If the host's shared directory is " \\SERVER1\HYPERLYNX466"
   and the client's working directory is "C:\HYPERLYNX"

   then the target line is:
   \\SERVER1\HYPERLYNX466\BSE.EXE -INIF:C:\HYPERLYNX\BSW.INI

To complete the installation, double-click the client-computer HyperLynx shortcut to run LineSim. The program should open, ready to load a .TLN file. When you exit the program, a BSW.INI file should be written in the client computer's C:\HYPERLYNX directory.

Viewing Manuals Online

This and all HyperLynx manuals are available for viewing online, in the form of .PDF (Acrobat) files. For details on installing the Adobe Acrobat viewer from the HyperLynx CD-ROM, see section “Steps for Installing the Software” above.

For convenience, any of the manuals can be opened for viewing from inside LineSim.

To open a manual .PDF file for online viewing:

1. From the Help menu, choose Manuals. The Adobe Acrobat viewer (if installed on your computer) is launched on a HyperLynx table-of-contents page.
2. In the table of contents, click on the name of the manual which you want to view. The manual file is opened.
3. View the manual using the features in the Acrobat viewer. For help with Acrobat features, refer to its online Help system.

You can also open a manual for viewing by double-clicking on it in the Windows Explorer. The manual files are located in the MANUALS sub-directory under the main LineSim directory.

Chapter 2: Quick Start!

Summary

This chapter is a quick summary of the steps required to enter and simulate a signal-integrity schematic in LineSim. It is intended for:

- ♦ new users who refuse to read manuals
- ♦ experienced users who need a quick reminder

This chapter covers only the main points and the most-common operations. It is *not* a substitute for the detailed chapters in this manual or the online Help.

Steps for Simulating a Signal-Integrity Schematic

Hint: *Components are **activated** in the LineSim schematic by **left**-clicking. Components are **edited** by **right**-clicking.*

Basic steps for simulating a signal-integrity schematic in LineSim:

1. From the File menu, choose New LineSim File.
2. Activate transmission-line segments by left-clicking on dark transmission-line “ghosts.”
3. Right-click on the transmission lines; enter electrical or geometric properties.
4. Activate driver and receivers by left-clicking on IC ghosts.

5. Right-click on the devices; choose IC models.
6. Activate passive components by left-clicking on resistor/capacitor ghosts.
7. Right-click on passive components; enter component values. (For ferrite beads, choose a model.)
8. Place oscilloscope probes by choosing Attach Probes from the Scope/Sim menu.
9. Open the oscilloscope by choosing Run Scope from the Scope/Sim menu; set scope parameters.
10. Click the Start Simulation button to begin simulating.

The following sections give more details.

Creating a New LineSim Schematic

LineSim performs signal-integrity simulations of schematics you enter in the tool. A “signal-integrity schematic” is different than an ordinary PCB or logic schematic, because it includes both electrical *and physical* information, not just electrical.

To create a new schematic:

1. From the File menu, choose New LineSim File.
   OR
   Click the New LineSim Schematic button on the toolbar.

Activating Transmission-Line Segments

LineSim schematics are built from a mixture of several component types:

- transmission-line segments
- IC devices (drivers and receivers)
- passive components (resistors, capacitors, inductors, and ferrite beads)

The LineSim schematic window consists of a large, scrolling grid of components. The schematic is subdivided into “cells”; each cell contains one of each type of component (including a “vertical” and “horizontal” transmission line). A particular component becomes “active” (i.e., becomes part of the schematic) when you left-click on it to activate it.

To activate a transmission line:

1. Point in the schematic to the dark “ghost” of the transmission line. When you point to the line, a red box appears around it.
2. Click with the left mouse button. The transmission line turns white and activates.

Entering Transmission-Line Properties

Before LineSim can simulate a transmission line, you must specify its electrical properties. There are several ways to set transmission-line properties in LineSim:

♦ by entering the electrical properties directly

♦ by entering geometric data for the line’s PCB cross-section type (microstrip, buried microstrip, or stripline)

♦ by tying the line to a stackup layer, and creating the stackup in LineSim’s Stackup Editor

To enter a transmission line’s properties:

1. Point to the transmission line and click with the right mouse button. The Edit Transmission Line dialog box opens.
2. In the Transmission Line Type area, click the radio button for the kind of modeling you want to use.
3. If the type is Simple, enter the line’s electrical properties directly in the area to the right. If the type is other than Simple, enter the properties in the Value-tab area that appeared when you clicked the radio button.

4. Click OK. The transmission line's electrical properties are exported to the schematic. The line's impedance, delay, and length appear inside the line in the schematic.

Activating IC Devices (Drivers and Receivers)

To activate an IC:

1. Point in the schematic to the dark "ghost" of the IC. (A ghosted IC looks like a double-triangle I/O-buffer symbol.) When you point to the IC, a red box appears around it.

2. Click with the left mouse button. The IC turns white and activates.

Choosing a Model for an IC

Before LineSim can simulate an IC, you must choose a model for it.

To choose driver- and receiver-IC models:

1. Point to the IC and click with the right mouse button. The Assign IC Models dialog box opens.

2. Double-click on the appropriate reference designator in the list box, and choose a library and device (and a pin/signal, if an IBIS or .PML model); click OK; set the buffer state to Output or Input; if a driver, check the Vcc/Vss Pins.

The reference designator for an IC is "U(*xy*)" where *x* is the column letter and *y* the row number for the cell of the schematic on which the IC appears.

Note: *LineSim supports three model formats, .MOD and .PML (HyperLynx formats) and IBIS (an industry standard). .MOD and .PML models are in libraries with file extensions .MOD and .PML, and IBIS models are in libraries with extension .IBS. There is a large collection of standard-logic models in library GENERIC.MOD; there are generic technology models (e.g., 3.3-V fast CMOS) in library EASY.MOD*
Signal-integrity simulations require only models for device families, not specific devices, since only output-buffer and input-stage characteristics need be modeled.

Activating Passive Components (Resistors and Capacitors)

LineSim supports the following passive components:

- pull-up resistor
- pull-down resistor
- capacitor
- AC terminator (resistor-capacitor combination)
- series resistor (or line-to-line resistor)
- series capacitor
- series inductor
- series ferrite bead

Any or all of these components can be activated at each schematic cell.

To activate a passive component:

1. Point in the schematic to the dark “ghost” of the component. When you point to the component, a red box appears around it.
2. Click with the left mouse button. The component turns white and activates.

Entering Terminating-Component Values

Before LineSim can simulate a passive component, you must specify its value.

To enter a component’s value:

1. Point to the component and click with the right mouse button. A dialog box opens.
2. In the Resistance, Capacitance, or Inductance edit box, type the component’s value. Click OK.

Note: *For ferrite beads, you load a model rather than setting a simple value. See Chapter 4, section "Choosing Ferrite-Bead Models" for details.*

Attaching Oscilloscope Probes

LineSim lets you place oscilloscope probes at any of the device pins in a schematic.

To attach oscilloscope probes:

1. From the Scope/Sim menu, choose Attach Probes.
2. Choose probe connections for the pins of interest.

Hint: *To probe a location that has no component pin, activate an IC, but do not choose a model for it. An unknown IC (labeled in the schematic "????") has no effect on the circuit.*

Opening the Oscilloscope and Simulating

To run a simulation:

1. From the Scope/Sim menu, choose Run Scope.
   OR
   Click the Open Oscilloscope/Simulator button on the toolbar.
2. Select the Driver Waveform edge and change the scope's scale settings, if necessary. Begin simulating by clicking the Start Simulation button.

Congratulations — you have run your first signal-integrity simulation with LineSim!

Chapter 3: Entering Signal-Integrity Schematics

Summary

This chapter describes:

- "signal-integrity schematics," and how they differ from PCB or logic schematics
- how to enter schematics in LineSim, including how to enter:
- transmission lines
- ICs (drivers and receivers)
- passive components (resistors, capacitors, inductors, and ferrite beads)
- how to zoom and scroll in the schematic editor
- how to save and re-load schematics
- how to print schematics and copy them to the Windows Clipboard

For information on how to *model* components in a schematic (e.g., how to choose an IC model, change a resistor value, or model a transmission line) see Chapters 4 and 6.

What is a “Signal-Integrity Schematic

?”Simulations in LineSim run off of “signal-integrity schematics” that you enter in LineSim’s integrated schematic editor. Because LineSim is intended as a “what-if” analysis tool and is often used *pre-layout*, LineSim does not read data from a PCB-layout tool or any other kind of software.

Note: *If you wish to perform simulations based on routed PCB-layout data, you need* ***BoardSim****, HyperLynx’s* ***post-layout*** *analysis tool. To best address signal-integrity problems throughout the design cycle, HyperLynx recommends both pre-layout analysis with LineSim and post-layout analysis with BoardSim.*

About “Signal-Integrity Schematics

There are several important differences between the signal-integrity schematics you enter in LineSim and traditional (e.g., PCB) schematics: ”

A traditional schematic...	**A signal-integrity schematic...**
...contains only electrical information	...contains both electrical and *physical* information
...shows *which* components are connected, but not *how* they’re connected	...shows which components are connected *and* how they’re connected
...includes only components	...includes both components and *transmission-line* segments
...includes information about how ICs function *logically* (e.g., an OR gate)	...includes information only about IC I/O *buffers*, and no information about how ICs function internally

For example, Figure 3-1 shows a portion of a PCB schematic containing an unpackaged logic gate and a buffer IC; Figure 3-2 shows a possible corresponding signal-integrity schematic.

Figure 3-1: A Portion of a PCB Schematic

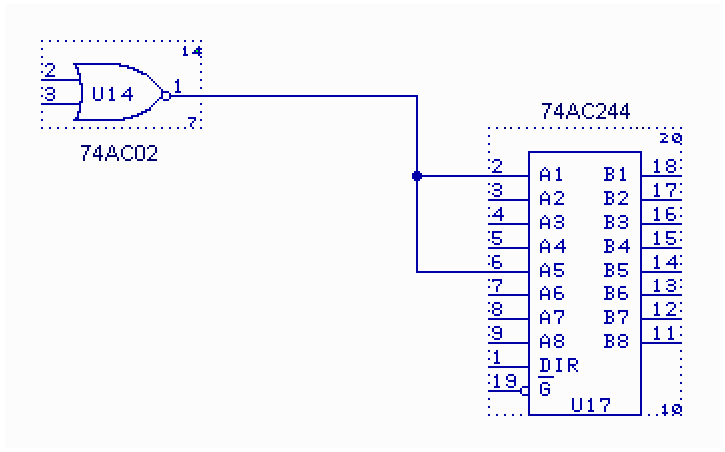
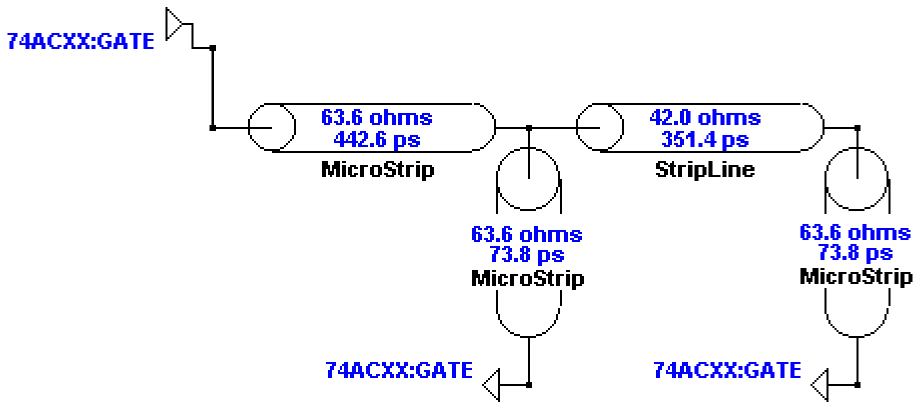


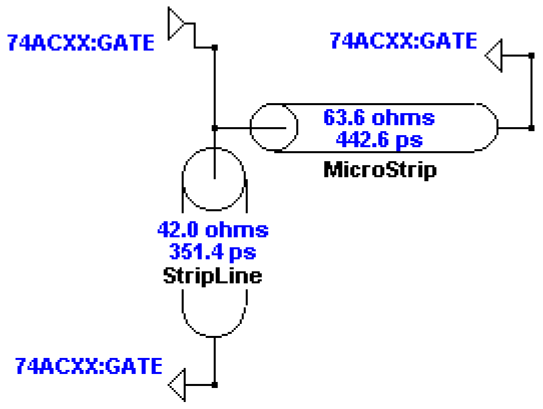
Figure 3-2: A Possible Signal-Integrity Schematic Corresponding to Figure 3-1



Notice in Figures 3-1 and 3-2 that the signal-integrity version of the schematic (3-2) contains information about *how* the components are connected, information that is missing from the PCB schematic (3-1). The connections are expressed as transmission lines, whose basic electrical properties (characteristic impedance and delay) are shown directly on the schematic.

In case you're thinking that the connectivity in Figure 3-2 is *implied* in Figure 3-1, consider Figure 3-3: another perfectly valid way of connecting the components in Figure 3-1, and one which looks considerably different than the wiring in the PCB schematic.

Figure 3-3: An Alternate Signal-Integrity Schematic for Figure 3-1



Transmission Lines and Schematics

A transmission line in a signal-integrity schematic can represent almost any kind of electrical connection, for example:

- ♦ a section of PCB trace
- ♦ a connector
- ♦ a cable

Thus, LineSim can be used to model almost any kind of system that involves interconnected electronic components, for example:

- ♦ a single net on a PCB
- ♦ a series of nets and connectors spanning two or more PCBs

- a collection of nets, connectors, and cables making up a complex signal path across an entire electronic system

The LineSim Schematic Editor

In LineSim, signal-integrity schematics are built from a mixture of several component types:

- transmission-line segments
- IC devices (drivers and receivers)
- passive components (resistors, capacitors, inductors, and ferrite beads)

The LineSim schematic editor consists of a huge, scrolling grid of components. Initially, all of the components are "ghosted," i.e., turned off. To create a schematic, you activate the components you want included in the simulation. You can simulate as soon as you've activated and modeled all of the desired components.

Creating a New Schematic

To create a new schematic:

1. From the File menu, choose New LineSim File.
   OR
   Click the New LineSim Schematic button on the toolbar.

The LineSim schematic editor opens, showing a portion of the component grid. If another LineSim schematic was open when you requested a new one, LineSim asks if you want to save the previous schematic. (See "Saving a Schematic" below in this chapter for details.)

Default Schematic Name

By default, the name of a new schematic is "UNNAMED0.TLN".

Schematic Cells

The schematic is subdivided into “cells.” Each cell contains one of each of the following:

- an IC (can be made either driver or receiver)
- a “vertical” transmission line
- a “horizontal” transmission line
- a pull-up resistor (can be tied to any voltage)
- a pull-down resistor (can be tied to any voltage)
- a capacitor
- a series component (in series with the horizontal transmission line), which can be a resistor, a capacitor, an inductor, or a ferrite bead

The entire available schematic consists of a large matrix of these cells.

Cells are labeled by row and column. Rows are labeled numerically; columns are labeled alphabetically. For example, the cell in the upper-left corner is numbered A0 (column A, row 0). The cell to its immediate right is labeled B0 (column B, row 0), while the cell immediately below is A1 (column A, row 1). This is similar to the cell-labeling scheme used in most spreadsheet programs.

Entering Transmission Lines

Activating a Transmission Line

To activate a transmission line (vertical or horizontal):

1. Point in the schematic to the dark “ghost” of the transmission line. (A transmission line is tube-shaped.) When you point to the line, a red box appears around it.

2. Click with the left mouse button. The transmission line turns light in color and activates.

When a transmission line has been activated in the current schematic for the first time, it has a set of properties — characteristic impedance, delay, physical length, and “model style” — that depend in part on a default PCB stackup that is created with each LineSim new schematic. More specifically:

- the transmission line is modeled with the “stackup” style, meaning that it is assumed to reside on a layer in a certain stackup; LineSim automatically generates a default stackup every time you create a new schematic; you can edit this stackup (see Chapter 7 generally) and control its default properties (see Chapter 7, section “Setting Default Layer Characteristics”); you also detach the transmission line from the stackup and model it in a variety of other ways (see Chapter 6, section “Transmission-Line Modeling Methods”)
- the characteristic impedance and delay depend on the current stackup and the transmission line’s layer position in it; you can move the line to a different layer (see Chapter 6, section “Entering a Stackup-Based Model”)
- the physical length is set to 3 inches; you can change this to any value (see Chapter 6)

These properties are displayed directly on the transmission line, in the schematic.

If you activate a transmission line; model it; de-activate it (see “De-Activating a Transmission Line” below for details); then re-activate it, the transmission line “remembers” the set of properties it had last time it was activated.

Transmission-Line States

In the schematic editor, transmission lines have three possible states:

activated	turned on, so that the line is included in simulations
shorted	short-circuited, so that the line is absent from simulations, and behaves like an electrical short circuit, tying together

	the nodes on either side with 0 ns delay
de-activated	turned off, so that the line is absent from simulations, and behaves like an electrical “open”

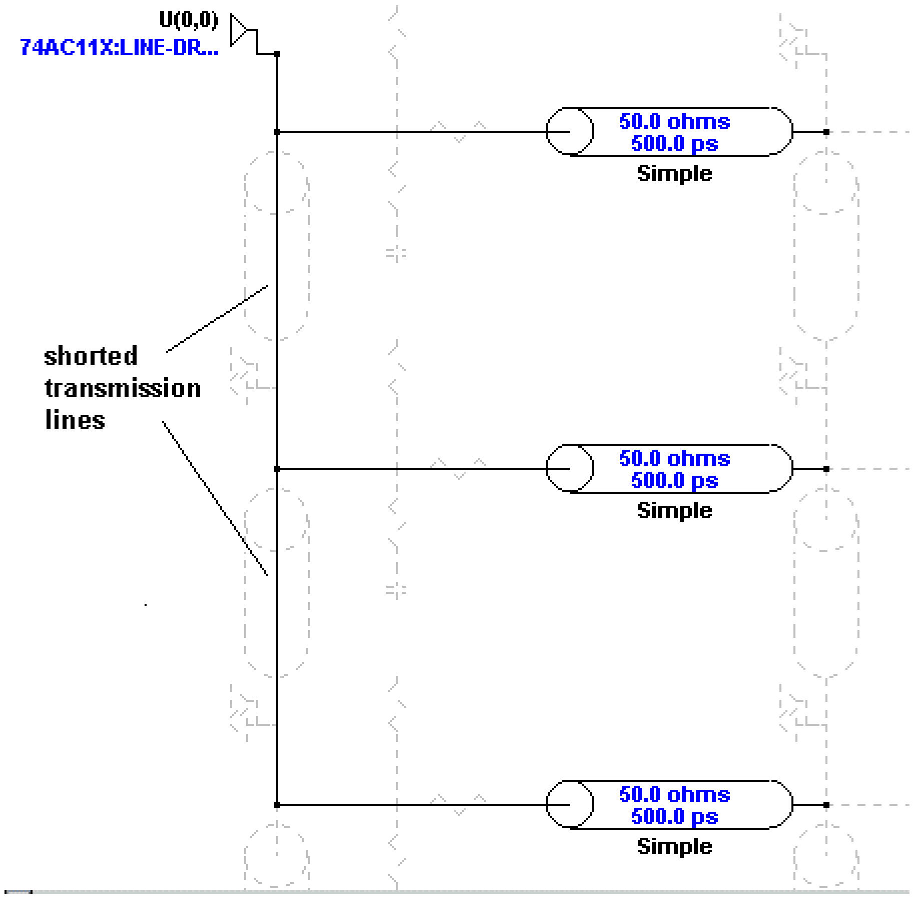
Clicking on a transmission line three times cycles the line through each state and back to the state in which it started.

Shorting a Transmission Line

The “shorted” state is provided as a drawing convenience so that multiple schematic cells can be tied together without an intervening transmission line. Shorted lines are really not transmission lines at all, because they have 0 ns of delay.

For example, suppose you want to drive from an IC into three transmission lines in parallel. Each schematic cell contains only two transmission lines. However, by shorting several lines, you can tie multiple cells together and build the desired circuit; see Figure 3-4.

Figure 3-4: Example of Using Shorted Transmission Lines



To short a transmission line, if the line is currently de-activated (ghosted):

1. Point in the schematic to the dark “ghost” of the transmission line. When you point to the line, a red box appears around it.
2. Click with the left mouse button. The transmission line turns light in color and activates.
3. Click again. The line turns into a short.

To short a transmission line, if the line is currently activated:

1. Point in the schematic to the transmission line.
2. Click with the left mouse button. The line turns into a short.

Note that shorted transmission lines are drawn as “wires” rather than the “tube” shape used to represent an activated line.

De-Activating a Transmission Line

To de-activate a transmission line, if the line is currently activated:

1. Point in the schematic to the transmission line. When you point to the line, a red box appears around it.
2. Click with the left mouse button. The transmission line turns into a short.
3. Click again. The line de-activates.

To de-activate a transmission line, if the line is currently shorted:

1. Point in the schematic to the transmission line.
2. Click with the left mouse button. The line de-activates.

A de-activated transmission line behaves like an open circuit, i.e., an infinite impedance.

Entering ICs (Drivers and Receivers)

Activating an IC

To activate an IC (driver or receiver):

1. Point in the schematic to the dark “ghost” of the IC. (An IC is represented by a double-triangle shape.) When you point to the IC, a red box appears around it.
2. Click with the left mouse button. The IC turns light in color and activates.

By default, when an IC has been activated in the current schematic for the first time, it has no simulation model, and the model name is displayed as “????”. Before simulating, you must choose a model for the IC and set its directionality and state (e.g., output or input). See Chapter 4 for details on IC modeling.

An IC with model name “????” behaves during simulation like an open circuit, i.e., the IC is effectively absent from the schematic.

If you activate an IC; model it; de-activate it (see “De-Activating an IC” below for details); then re-activate it, the IC “remembers” the model it had last time it was activated.

De-Activating an IC

To de-activate an IC:

1. Point in the schematic to the IC.
2. Click with the left mouse button. The IC de-activates.

A de-activated IC behaves like an open circuit, i.e., an infinite impedance.

Entering Passive Components (Resistors, Capacitors, Inductors, and Ferrite Beads)

Types of Passive Components

The LineSim schematic editor supports the following types of passive components:

- ♦ pull-up resistor
- ♦ pull-down resistor
- ♦ capacitor

- series resistor
- series capacitor
- series inductor
- series ferrite bead

These components can be used in any combination. For example, to construct a parallel AC terminator (resistor and capacitor to ground), you can activate both the pull-down resistor and the capacitor (they're in series with each other).

There is no limit to how many passive components you can use in a schematic, or to how they're combined. There is one of each type of component at each schematic cell (see "Schematic Cells" above in this chapter for more details).

Note: *Clamp diodes can also be modeled in LineSim, although not in the same way — with an explicit drawing symbol — as the passive components listed above. For details, see Chapter 4, section "Other Libraries."*

Activating a Passive Component

To activate a passive component:

1. Point in the schematic to the dark "ghost" of the component. When you point to the component, a red box appears around it.

2. *If the component is not a series component,* click with the left mouse button. The component turns light in color and activates.
   OR
   If the component is a series component, a floating menu opens. Move the mouse to the type of component (resistor, capacitor, inductor, or ferrite bead) that you want; then click with the left mouse button. The component appears with the proper electrical symbol.

Resistors, Capacitors, and Inductors

When a resistor, capacitor, or inductor has been activated in the current schematic for the first time, it has a default value which depends on the component type:

pull-up or pull-down resistor	Defaults to 10 kohms
series resistor	Defaults to 0 ohms
capacitor	Defaults to 100 pF if parallel; 0.1 uF if series
inductor	Defaults to 10 nH

Note that the default resistor values are such that the resistors will have little effect on simulation results unless their values are changed (see Chapter 4 for details on changing component values). The component's current value is displayed next to it in the schematic.

If you activate a passive component; changes its value; de-activate it (see "De-Activating a Passive Component" below for details); then re-activate it, the component "remembers" the value it had last time it was activated.

Ferrite Beads

Ferrite beads require a model more complex than that for other, simple passive components (e.g., a resistor). You cannot, for example, adequately model a bead by specifying a single numeric value. LineSim models a bead with a parallel combination of inductance, resistance, and capacitance. The L-R-C model is automatically synthesized from three of the bead's impedance-vs.-frequency points.

By default, when a ferrite bead has been activated in the current schematic for the first time, it has no simulation model, and the model name is displayed as "????" (just like with an IC). Before simulating, you must choose a model for the bead (see Chapter 4 for details on ferrite-bead modeling).

A ferrite bead with model name "????" behaves during simulation like a 0.0-ohm resistor, i.e., the bead is effectively absent from the schematic, except for its lead parasitics. (See Chapter 4, section "IC and Terminating-Component Parasitics" for details on parasitics.)

If you activate a ferrite bead; model it; de-activate it (see "De-Activating a Passive Component" below for details); then re-activate it, the bead "remembers" the model it had last time it was activated.

De-Activating a Passive Component

To de-activate a passive component:

1. Point in the schematic to the component.
2. Click with the left mouse button.
   If the component is not a series component, the component de-activates.
   OR
   If the component is a series component, a floating menu opens. Move the mouse to "None"; then click with the left mouse button. The component de-activates.

A de-activated component behaves as if it's not in the schematic.

Inserting and Deleting Rows and Columns

LineSim's schematic editor, in order to render "large-scale" changes to a schematic easier to make, allows you to insert entire rows or columns in the middle of an existing schematic, and to delete entire rows and columns from a schematic. By an "entire row" is meant all of the cells across an entire row of the schematic, from edge to edge; and similarly for an "entire column."

Inserting Rows and Columns

To insert a row (or column) into an existing schematic:

1. From the Insert menu, choose New Row (or New Column). The cursor changes into a "hand" shape.

2. Position the hand cursor in the schematic such that the heavy red line that follows the cursor is positioned between the two rows (or columns) where you want the inserted row (or column) to appear.

3. Click the left mouse button. The insertion occurs and the cursor changes back to its normal shape.

All of the elements in the portion of the schematic that moves as a result of the insertion are left unchanged, except that they now reside at new locations. IC models, component values, oscilloscope probes, and so forth are unaffected by the insertion. The cell labels on the schematic (see “Schematic Labels” above in this chapter for details), however, remain fixed; the moved elements take on new cell labels.

Deleting Rows and Columns

To delete a row (or column) from an existing schematic:

1. From the Edit menu, choose Delete Row (or Delete Column). The cursor changes into a “hand” shape.

2. Position the hand cursor in the schematic such that the two heavy red lines that follow the cursor are positioned on either side of the row (or column) which you want deleted.

3. Click the left mouse button. A Warning box opens to ask if you’re sure that you want to complete the deletion.

4. Click Yes. The deletion occurs and the cursor changes back to its normal shape.

Be certain to think carefully before clicking Yes to initiate the delete operation, because there is no “undo” command available. If you are working on a large, complex schematic, it is a good idea to save it before performing a complicated delete operation.

All of the elements in the portion of the schematic that moves as a result of the deletion are left unchanged, except that they now reside at new locations. IC models, component values, oscilloscope probes, and so forth in the remaining portions of the schematic are unaffected by the deletion. All elements in the deleted row or column disappear; oscilloscope probes attached to them are automatically detached. The cell labels on the schematic (see “Schematic Cells” above in this chapter for details) remain fixed; the moved elements take on new cell labels.

Zooming and Scrolling

In order to handle schematics of virtually any size and complexity, the LineSim schematic editor includes a full set of zooming and scrolling features. You can zoom out, for example, to more easily work on a large schematic, or you can stay at a further “in” resolution and scroll from section-to-section of a schematic.

Zooming

Zooming In

To “zoom in” on a schematic:

1. From the View menu, choose Zoom In.

The editor zooms in closer to the schematic.

The center point of the new view is the same as the center point of the old view. To set the center point before zooming in, use the scroll bars, or use the “Zoom Area” command (see below for details).

Zooming to an Area Defined by a Box

The Zoom Area command allows you to draw a box around the zoomed-in view you want.

To “zoom in” on an area defined by a box:

1. From the View menu, choose Zoom Area.
   OR
   Click the Zoom into Area button on the toolbar.
   The cursor changes to a “magnifying-glass” shape.

2. Position the mouse cursor where you want one of the corners of the zoom box to be.

3. Click the mouse button and while continuing to hold the button down, “drag out” a box defining the zoom area. The zoom box appears in gray.

4. Release the mouse button.

The schematic editor zooms in to the area defined by the zoom box.

Zooming Out

To “zoom out” on a schematic:

1. From the View menu, choose Zoom Out.

The editor zooms farther out from the schematic.

The center point of the new view is the same as the center point of the old view. To set the center point before zooming out, use the scroll bars.

Zooming to a “Normal” (Default) View

When you first open the schematic editor on a new or saved schematic (see “Saving a Schematic” below in this chapter for details on saving schematics), it chooses a default zoom level. The default level is about right for creating small schematics without having to scroll.

The schematic editor allows you to go in one command from another zoom level to the default or “normal” view.

To zoom to a “normal” (default) view:

1. From the View menu, choose Zoom Normal.

The schematic editor zooms to its default level.

Zooming to the Previous Zoom Level

The schematic editor lets you return from any current zoom level to the previous level. This feature is implemented as a “stack”; as you execute it repeatedly, the editor “walks” back through the previous zoom levels you have used.

To zoom to the previous zoom level:

1. From the View menu, choose Zoom Previous.
   OR
   Click the Return to Previous Zoom button on the toolbar.

The schematic editor zooms to the level at which it was previously.

Quick Way to Access Zoom Menu

There is a quick way to access the View menu, without going to the menu bar.

To access the View menu (and zooming features) without the menu bar:

1. Point to any empty area in the schematic (i.e., point at something other than a component), and click with the right mouse button.

A floating version of the View menu opens at the mouse cursor’s position.

Scrolling

You can scroll in the LineSim schematic editor exactly like in any Windows window, i.e., by using the scroll bars at the right and bottom edges of the editor window. The scroll bars are present regardless of what zoom level the schematic editor is currently at.

Saving and Loading Schematics

LineSim allows you to save the current schematic — actually, the entire session associated with the schematic — so you can stop a session and later return to exactly where you were. You may also want to save schematics to archive various design ideas you’ve had and solutions you’ve found.

.TLN Files

Schematics are saved in an ASCII format into files with extension ".TLN". The schematic file (.TLN) includes all of the data relevant to a schematic and how it was being simulated at the time it was saved, including:

- activated components in the schematic
- IC model selections
- transmission-line models
- terminating-component values
- the position of oscilloscope and spectrum-analyzer probes
- the oscilloscope and spectrum-analyzer settings.

Oscilloscope and spectrum-analyzer waveforms are not saved.

Note: *The .TLN format, unlike BoardSim's .HYP format, is not a published format. Even though they're ASCII files, HyperLynx discourages users from editing .TLN files. The format is cryptic and it is easy to "damage" a file and render it unreadable by LineSim.*

Saving a Schematic

To save a schematic (and the associated session):

1. From the File menu, choose Save As. The Save As dialog box opens.
2. Use the standard Windows methods in the dialog box to change to the directory to which you want to save. By default, the directory is whatever is specified as the .HYP and .TLN File Path, or the last directory in which you opened a schematic (depending on which option you choose; see "Setting the Default Load Directory" below in this chapter for details on setting the default directory).
3. Type a name for the schematic in the File Name box. The ".TLN" extension is optional.

4. Click the Save button. The schematic is saved.

If you do not type the “.TLN” extension, it is automatically appended to the schematic file name. This is true even if you type a different extension, for example “.TMP”; in this case, LineSim will create a file called “<file_name>.TMP.TLN.”

Exiting/Closing without Saving

If you attempt to close the current schematic or exit LineSim without first saving the current schematic, LineSim asks if you want to save. If you click “Yes,” the Save As dialog box opens. If you click “No,” LineSim discards your current schematic.

Saving an Already-Named Schematic

If you have already named your schematic by earlier choosing File / Save As, you can re-save the schematic, to capture any additional changes you’ve made..

To re-save a schematic under its existing name:

1. From the File menu, choose Save.

If you choose File / Save for a new schematic which has not been named, it will save under the default name (UNNAMED0.TLN).

If you are working with a large, complex schematic, it is wise to occasionally save it, in case you computer suffers a power glitch, etc.

Loading a Schematic

To load a saved schematic (and the associated session):

1. From the File menu, choose Open LineSim File.
   OR
   Click the Open LineSim .TLN File button on the toolbar.
   The Open LineSim File dialog box opens.

2. Use the standard Windows methods in the dialog box to change to the directory from which you want to load a saved schematic. The directory is whatever you’ve specified as the default .HYP and .TLN File Path, or the

last directory in which you opened a schematic (depending on which option you choose; see “Setting the Default Load Directory” below in this chapter for details on setting the default directory)

3. Type a name for the schematic in the File Name box. The “.TLN” extension is optional.

4. Click the Open button. The schematic is loaded.

If you do not type the “.TLN” extension, it is automatically appended to the schematic file name. This is true even if you type a different extension, for example “.TMP”; in this case, LineSim will attempt to open a file called “<file_name>.TMP.TLN.”

Missing IC Models

If you load a schematic that includes certain IC models that no longer exist or cannot be found, the schematic will load, but LineSim will issue a warning that certain models were not located.

The missing-model ICs in the schematic will display with name “????” and behave during simulation like a high impedance, i.e., they will effectively be absent from the schematic. You should choose new IC models for them before proceeding.

LineSim looks for all IC models in the Model Library File Path directory. (See “Setting the Default Load Directory” below in this chapter for details on setting the default directory.) You should move all of your models (.MOD, .PML, and .IBS) to whatever directory you have specified for model libraries. (See Chapter 4, section “IC-Model Formats” for details on the model formats.)

If you load a schematic and most or all of its IC models are not found (i.e., a warning is issued and the IC names are “????”), check the Model Library File Path setting — make sure the directory setting and the location of your models agree.

Loading over an Existing Schematic

If you attempt to load a schematic with another schematic currently in the editor, LineSim asks if you want to save the current schematic first. If you click

"Yes" and the current schematic already has a name, the current schematic is saved and then the Open LineSim File dialog box opens.

If you click "Yes" and the current schematic still has the default name "UNNAMED0.TLN," the Save As dialog box opens so you can name the current schematic; after saving the current schematic, the Open LineSim File dialog box opens. (See "Saving a Schematic" above in this chapter for details on naming a schematic file.)

Changing the Schematic Background Color

You can set the schematic editor's background color to any color you choose.

To change the schematic editor's background color:

1. From the Options menu, choose Preferences. The Options dialog box opens.
2. Click on the Color tab.
3. Click on the Edit Color button. (The color "chip" next to the button shows the editor's current background color.)
4. Use the standard Windows Color dialog box to select a new color.
5. Click OK twice.

You can change to any color, including white or black. If you choose white or black, certain objects automatically change to gray, so they're still visible.

Setting the Default Load Directory

LineSim allows you to specify a default directory for the location of your .TLN files. The directory setting is just a convenience: it specifies the default directory that the Open LineSim File dialog box opens on. You can change in the dialog box to any other directory, which means that you can store your .TLN files anywhere you wish, and in multiple directories if you want. You can also instruct LineSim to always open the file dialog box on the directory from which you *last* loaded a .TLN file.

Setting the Default .TLN Files Path

To set the TLN path to a particular directory:

1. From the Options menu, choose Directories.
2. Verify that the Use Directory of Last-Opened File check box is disabled.
3. In the .HYP and .TLN File Path edit box, type the desired directory (with a trailing '\').
   OR
   Click the Browse button; a dialog box opens. Find the directory you want to use and double-click on a .TLN or .HYP file in the directory to set the path. (You can also highlight a .TLN or .HYP file and click Open.)
4. Click OK.

The change takes effect immediately.

To tell LineSim to always use the directory from which you *last* loaded a .TLN file (rather than a particular "fixed" directory):

1. From the Options menu, choose Directories.
2. Click on the Use Directory of Last-Opened File check box, to enable it.
3. Click OK.

To restore the .TLN files directory to its default value:

1. From the Options menu, choose Directories.
2. Verify that the Use Directory of Last-Opened File check box is disabled.
3. In the .HYP and .TLN File Path edit area, click the Default button. The path is reset to its default value.
4. Click OK.

Note: *".HYP" is the format used by HyperLynx's BoardSim. The Set Directories dialog box is shared by both LineSim and BoardSim. If you own both programs and want the Open File dialog box to open on both .TLN and .HYP files, you* *must store the two kinds of files together in the same directory. The default name for the joint directory (see above for how to restore the default)) is HYPFILES, as a subdirectory of the LineSim/BoardSim root directory.*

Printing Schematics

You can print a schematic from LineSim in order to document it.

When LineSim prints a schematic, it effectively draws the smallest rectangle it can around your circuit, then scales the rectangular image to best fit on whatever paper you're printing on.

To print a schematic:

1. From the File menu, choose Print Schematic.
   OR
   Click the Print LineSim Schematic button on the toolbar.

2. In the Print dialog box, check your printer setup. Click OK to begin printing.

LineSim supports color printers; schematics sent to a color printer are output in color.

Setting Up for Printing

You can set up printing-related defaults — e.g., printer choice, paper size, page orientation, etc. — once in LineSim, then have them apply for the remainder of your work session, and for all types of printing (schematics, stackups, oscilloscope results, etc.).

To set up "persistent" printing defaults:

1. From the File menu, choose Print Setup. The Print Setup dialog box opens.

2. Change any parameters you wish, then click OK.

LineSim now remembers the choices you've made, and will continue to use them.

Printing Large Schematics

LineSim prints the entire schematic to a single page, of whatever size is set in the Print Setup dialog box (see “Setting Up for Printing” above for details). If your schematic is large and you’re trying to print to letter or A-sized paper, the objects in the schematic may become very small and therefore hard-to-read.

However, if rather than printing it, you copy the schematic to the Windows Clipboard, then paste the image into another Windows application that supports multi-page printing, you can use the other application get your schematic spread across multiple sheets.

Microsoft Excel has good support for multi-page printing; other Windows spreadsheet programs do as well. Excel (and others) also support plotters. Consider using LineSim’s copy-to-clipboard feature — detailed in “Copying Schematics to the Clipboard” below — in conjunction with one of these applications if you’re trying to print large schematics.

Copying Schematics to the Clipboard

You can copy a schematic to the Windows Clipboard in order to paste it into other Windows applications. The image sent to the Clipboard is formatted, and includes information such as the name of the .TLN file, the date and time, and so forth. You can capture an image of the entire schematic or a sub-portion of it.

To copy a schematic to the Windows Clipboard:

1. From the Edit menu, choose Copy Schematic to Clipboard. The Copy Schematic to Clipboard dialog box opens.

2. The Cells Currently Used in This Schematic area shows the maximum cell coordinates used in the current schematic. By default, the Upper Left/Lower Right Corner Cell combo boxes are set to the maximum coordinates in use, so that the clipboard image will exactly fit the current drawing. However, if you want to capture only a sub-portion of the current schematic, pull down the combo boxes and change the corner values for the image as needed.

3. In the Text and Graphics area, choose colored or black-and-white output.

4. In the Features area, enable/disable the generation of a formatted border and/or user-name and timestamp data in the output.

5. Click OK. The image, cropped and formatted as you specified, is sent to the Windows Clipboard.

LineSim writes to the Clipboard in Windows Enhanced Metafile format. The size of the image may vary depending on which application you paste it into; resize as needed (metafiles are vectored and can be sized without damaging image quality).

Chapter 4: Modeling ICs and Passive Components

Summary

This chapter describes how to model ICs and passive components (resistors, capacitors, and inductors). Specifically, it discusses:

- what “modeling” ICs and passive components means
- what is in the Assign Models dialog box
- how to choose, copy, change, and remove IC models
- how to edit terminating-component values
- how to choose ferrite-bead models
- modeling parasitic values (packages and leads)
- how to update IC models over the Internet

For details on how to model transmission lines, see Chapter 6.

What Does “Modeling” ICs and Passive Components Mean

Before you can simulate a schematic you have drawn, you must choose models and edit values for the components on the net. What steps are required depends on each component’s type: ?

for an IC...	...choose a model
for an R...	...check value; modify if needed
for a C...	...check value; modify if needed
for an L...	...check value; modify if needed
for a ferrite bead...	...choose a model
for a transmission line...	...choose a model (see Chapter 6)

This chapter discusses how to model ICs and passive components (including ferrite beads); for details on modeling transmission lines, see Chapter 6.

IC Models

Before simulating a schematic, you must choose models for the IC components in the schematic. “Choosing a model” for an IC means selecting a model from one of the libraries supplied with LineSim; or modifying a supplied model, saving it to one of your own libraries, and selecting it.

For details on choosing IC models, see “Choosing IC Models” below in this chapter. For details on how ICs behave by default *before* you model them, see Chapter 3, section “Activating an IC.”

Passive Components (Resistors, Capacitors, and Inductors)

For passive components other than ferrite beads (resistors, capacitors, and inductors), LineSim has a model built-in, but it must know the *value* of the component. (A default value is supplied when you first activate the component.)

For details on the different types of passive components (e.g., series resistors, capacitors, and so forth) see Chapter 3, section “Types of Passive Components.” For details on the default values for passive components, see Chapter 3, section “Activating a Passive Component.”

Choosing IC Models

Assign IC Models Dialog Box

To choose a model for an IC:

1. Point in the schematic to the IC. When you point to it, a red box appears.
2. Click with the **right** mouse button. The Assign IC Models dialog box opens.

The Assign IC Models dialog box allows you to control modeling *for all the ICs* in a schematic. This means you can right-click on one IC to edit not only its model, but also the models for all of the ICs in your schematic.

The following sections describe the elements in the Assign IC Models dialog box and how to use them.

The Pins List

The Pins list box displays all of the IC pins in the current schematic. Since in LineSim, each IC effectively has a single pin, the list box displays pins by showing the reference designator for the corresponding IC.

IC Reference Designators

An IC's reference designator is created automatically from the IC's schematic-cell label (see Chapter 3, section "Schematic Cells" for details on cells.)

For example, the IC in cell A1 has reference designator "U(A1)". In the Assign IC Models dialog box, its pin is also displayed as "U(A1)".

Pin Icons

Each IC pin is marked in the Pins list box with an icon that shows the status of its model. The icons allow you to easily see which ICs have models. You can also easily determine which IC pin in the schematic are drivers ("outputs") and which are receivers ("inputs").

Red Question Mark

The red question-mark icon means that the IC is missing modeling information. Before simulating, you must remove at least some of the red question marks by choosing IC models. See "How to Choose an IC Model" below in this chapter for details. ICs which have the red question-mark icon in the Pins list box are named "????" in the schematic editor.

You are not required to remove all red question marks before you can simulate, though typically you would. LineSim requires only a driver-IC model before simulating.

The following matrix summarizes some possibilities:

for a complete simulation	remove all red question marks (i.e., supply all IC models and check all component values)
for a quick simulation	supply a driver-IC model; check all passive component values; ignore receivers
minimum requirement	supply a driver-IC model

Green Driver Icon

When you select an IC model that has an output-only buffer direction; or when you select an IC model and set its buffer state to Output, Output Inverted, Stuck High, or Stuck Low, the IC's icon changes from a red question mark to a green driver shape; the driver shape points to the right (as opposed to a receiver shape, which points to the left). (See "Setting IC Buffer Direction/State" below in this chapter for details on buffer direction and state.)

Green Receiver Icon

When you select a new IC model; or when you change a bi-directional IC model's buffer state to Input, the IC's icon shape changes to a green receiver shape; the receiver shape points to the left (as opposed to a driver shape, which points to the right). (See "Setting IC Buffer Direction/State" below in this chapter for details on buffer direction and state.)

The Models Area

The area to the right of the Pins list box is called the "models area." When you highlight an IC pin in the Pins list box, the models area changes to show you the current status of the IC's model.

Other Sections of the Assign Models Dialog Box

The Assign Models dialog box contains several other areas (e.g., the Buffer Settings area, Vcc/Vss Pin combo boxes, and Model to Paste area), but these are discussed in detail in other sections below in this chapter.

How ICs are Modeled for Signal-Integrity Simulation

A major difference between modeling for digital simulation and modeling for signal-integrity simulation is that for signal-integrity modeling, the simulator does not need to model the logical function of the IC. Only the characteristics of the driver's output stage or the receiver's input stage matter. This means that LineSim only needs models for every unique buffer type, not for every unique IC. For standard-logic devices especially, this simplifies a signal-integrity tool's modeling libraries.

For example, a 74AC04, a 74AC74, and a 74AC161 all have the same output stage, so all three behave the same in a signal-integrity simulation and can be described by a single model. But a 74AC240 has a different, higher-current output stage, so it requires a different model. Likewise, many of the I/Os on Intel's Pentium microprocessor share the same device model, regardless of their logical functions.

Note: *On the other hand, the models in a signal-integrity library must be detailed analog characterizations, not merely logical models.*

The key parameters for a signal-integrity driver model are rise/fall time, "on" impedance or V-I characteristic, the nature of the impedance change from "off" to "on" (and vice versa), and the output capacitance.

For a receiver model, the key characteristics are the nature of the clamp diodes, input resistance, and input capacitance.

IC-Model Formats

LineSim supports three kinds of device models:

.MOD format	a HyperLynx proprietary format, based on "databook" parameters; many of LineSim's standard-logic models are in .MOD format
.PML format	an extension to the .MOD databook format that adds to .MOD component pin-outs and package parasitics
IBIS format	an industry-standard format, supported by a variety of simulation and IC vendors; IC vendors can create detailed, accurate IBIS models without giving away proprietary information; LineSim customers can run IBIS models obtained directly from IC vendors

The following table compares .MOD and IBIS models. For details on each format, see the following sections.

Characteristic	.MOD Format	.PML Format	IBIS Format
ASCII?	✓	✓	✓
keyword-based?		✓	✓
edit in LineSim?	✓	✓	✓
edit with text editor?		✓	✓
packages modeled?		✓	✓
includes min/max?	✓	✓	✓
signals/pins listed in model?		✓	✓
model includes both driver and receiver?	✓		
easy for users to create?	✓		
supported by other simulators?			✓
contains detailed vendor specs?			✓
available on HyperLynx Web site?	✓	✓	✓

The .MOD “Databook” Format

.MOD models are described in ASCII libraries with extension “.MOD.” The .MOD (or “databook”) format is HyperLynx-proprietary and has been supported since 1989. Many of LineSim’s standard-logic models are in a library called GENERIC.MOD. HyperLynx adds additional .MOD libraries as necessary to support newer ICs.

LineSim has a convenient dialog box for modifying .MOD models. .MOD models are based on parameters commonly found in IC databooks, to make the models as easy as possible to create and support. (See Chapter 5 for details on editing models.) You might edit a .MOD model, for example, as a starting point for a new model. Modified .MOD models can be saved to a user library.

Although .MOD files are ASCII, they are not keyword-based and the .MOD ASCII format is not published. As a result, HyperLynx recommends against modifying them with a text editor; use the .MOD model editor in LineSim instead.

Also, .MOD files can model min/max device characteristics as well as typical, but do so through a pair "global" scaling factors that modify up/down the parameters in all .MOD models in a simulation, to give a best-case/worst-case effect. (This contrasts with IBIS models, which contain detailed min/max data on a per-device basis.)

Note: *.MOD files themselves model the silicon portion of an IC; you can use the parasitic modeling capability in LineSim to separately model an IC's package. You can also convert a .MOD model to .PML and include package characteristics.*

.MOD models usually represent an entire family of ICs. For example, the 74ACXX:GATE model in GENERIC.MOD represents the output of any non-line-driver 74AC IC. .MOD models do not contain lists of the specific devices or signals they represent.

.MOD models combine drivers and receivers into a single model. The 74ACXX:GATE model has a driver and receiver; the receiver represents input pins on any 74AC non-line-driver device. For certain devices which are input- or output-only (e.g., a clamp diode), the other "side" of the model can be set to all "off" characteristics and ignored (e.g., for a clamp diode, the output side could be set to all "open" and never used).

The .PML "Package Model Library" Format

The .PML format is an extension to the .MOD databook format that adds component pin-out and package-parasitic information to .MOD models. The

.PML — “Package Model Library” — format was added to give .MOD models a more-equal footing with IBIS models.

The .PML format works by adding a new library-file type (with extension .PML) that defines components in an IBIS-like syntax. .PML component definitions attach specific .MOD models to each pin on an IC, and add directionality (input, output, bi-directional, etc.) to each pin. .PML also defines a component’s package, and supplies parasitic R, L, and C values for each pin.

.PML files do not themselves contain .MOD models. Rather, .PML files define component pin-outs and point to the models in a .MOD file that actually define the pin’s analog behavior. Thus, the .PML file is really just an extension of the .MOD format. A .PML model requires two files: the .PML file and the .MOD file to which it points.

The IBIS Format

IBIS — “I/O Buffer Information Specification” — is an industry standard for signal-integrity IC modeling. Created originally by Intel, IBIS has been rapidly endorsed by all of the major CAE vendors and IC manufacturers.

Note: *HyperLynx is proud that its DOS version of LineSim Pro was the first program in the industry to be declared “IBIS Certified.” HyperLynx was a founding member of the IBIS Open Forum, the industry committee that defines and maintains the IBIS standard.*

IBIS is significant because it allows semiconductor vendors to provide accurate models without giving away the proprietary details of their silicon. This removes a key barrier that previously made models difficult to get, or completely unavailable.

IBIS models are described in ASCII libraries with extension “.IBS.” LineSim ships with a collection of IBIS libraries; new libraries are available to customers on HyperLynx’s Web site (see “Updating Models over the Internet” below in this chapter for details.)

IBIS models can optionally include detailed package models (pin R,L,C) and min/max data. LineSim automatically simulates packages if the data is present; you can specify whether to use min, typ, or max data during simulation. (See Chapter 9, section “Setting IC Operating Parameters” for details.)

IBIS models are arranged into components. Each IBIS library may contain multiple components; each component has a list of signals; each signal has an associated model. Each signal may be input, output, or I/O.

IBIS Specification

The IBIS format is public; the IBIS specification is available in Appendix A of this manual and in LineSim’s Help.

Obtaining Models

Semiconductor Vendors

The IBIS format is specifically intended to allow IC vendors to supply signal-integrity models directly to their customers. You are strongly encouraged to request IBIS models from your IC vendors. As paying customers of the IC vendors, you have greater leverage in demanding models than do EDA suppliers.

If you speak with an IC vendor who needs information about or software tools for IBIS development, please have them contact HyperLynx.

From HyperLynx’s Web Site

As they become available, HyperLynx posts new models on the HyperLynx Web site. In fact, if your computer is connected to the Internet, you can update your IC models at any time from inside LineSim. See “Updating Models over the Internet” below in this chapter for details.

The HyperLynx World Wide Web site also provides extensive links to other IBIS modeling sites. See Chapter 12 for details on accessing the Web site.

How to Choose an IC Model

To choose an IC model:

1. In the Assign IC Models dialog box, in the Pins list box, highlight the IC pin for which you want to choose a model.

2. Click the Select button.
   OR
   Double-click on the pin in the Pins list box.

   The Select IC Model dialog box opens.

3. In the Library list box, highlight the library from which you want choose a model. If desired, use the radio buttons to the left of the Library list box to limit the libraries displayed to one type (IBIS, .PML, or .MOD); you can also click on the EASY.MOD or GENERIC.MOD buttons to jump to those specific libraries. When you highlight a library, the Device and Pin/Signal list boxes update to display the contents of the selected library.

   Note: *The Pin/Signal list box is grayed out for .MOD libraries. Only IBIS and .PML libraries contain lists of component signals/pins.*

4. Highlight the device for which you want a model. If the library is .IBS or .PML, the Pin/Signal list box updates to display the contents of the highlighted device.

5. *If the library is .IBS or .PML,* highlight the pin or signal name for which you are choosing a model. If you want to change from choosing by pin name to choosing by signal name (or vice versa), first click the appropriate radio button in the Select By area. The directionality/state of the highlighted pin/signal is shown graphically in the I/O Type area. If the model is differential (applies to .IBS only), the picture will clearly show two drivers or receivers. The I/O Type area also displays any threshold information present in the model.

6. Click OK.
   OR
   If the library is .MOD, double-click on the device name;
   if the library is .IBS or .PML, double-click on the pin/signal name.

   The Select IC Model dialog box closes, and the model is chosen.

7. Back in the Assign Models dialog box, in the Buffer Settings area, click the appropriate radio button to set the model's driving direction and state. The Settings area displays radio buttons for only those direction/state

combinations that are valid for the selected model (e.g., for an output-only model, there is no "Input" button displayed).

8. *For driver (i.e., output) models only*, look at the Vcc Pin and Vss Pin combo boxes. These allow you to power the driver from any two power-supply nets, or from the driver model's "typical" value (i.e., the voltage recommended internally in the model). The "typical" choice appears at the top of each list.

9. To finish choosing IC models, click Close. Or, to choose a model for another IC pin, repeat steps 1 - 8.

When you close the Assign IC Models dialog box and return to the schematic editor, the IC pins for which you have chosen models are displayed with the models' names in blue.

For more details about the operations described in these steps, see the following sections.

Choosing a Library

The first step in choosing an IC model is choosing a library. You can always use any of .MOD, .PML, or IBIS models; you can mix the model formats freely in a schematic.

LineSim ships with several groups of libraries:

Library Name	**Description**
GENERIC.MOD	Contains HyperLynx's models for a large number of standard-logic families (all 74XX, ECL, and certain other, miscellaneous models)
other .MOD libraries	Contain vendor- or family-specific models, usually created by HyperLynx. An examples is IDT.MOD (IDT FCT devices).
.IBS libraries	Contain family- or component-specific models created by the IC vendors or in a few cases, by HyperLynx. Examples are 82430.IBS (created by Intel), and

Library Name	**Description**
	PLSI2128.IBS (created by HyperLynx).
EASY.MOD	Contains completely generic models that are based on technology types (CMOS or bipolar) and approximate switching speeds (e.g., fast, medium, slow). Use this library when you can't find an exact model for a device and don't have time to create or search for one.

The GENERIC.MOD Library

Most of LineSim's standard-logic models are contained in the library GENERIC.MOD. There are exceptions; some families have their own, separate .MOD libraries.

Note: *LineSim also ships with an obsolete library called LINESIM.MOD. In the old DOS version of LineSim, LINESIM.MOD was the name of the standard-logic library; it is provided now under the same name for backwards compatibility. LineSim does not prevent you from editing or deleting models from LINESIM.MOD, but you should avoid doing so if you have old .TLN files that reference the library.*

Use GENERIC.MOD for all new schematics.

Model Names in GENERIC.MOD

In GENERIC.MOD, the format for most models' names is

family : type

Examples of device families are:

74ACxx

74FCTxx

74BCTxx

MACH

74Fxx

DRAM

The device types and their meanings are

Device Type	**Meaning**
GATE	gates, with “normal” current drive and capacitance; e.g., 74AC00, 74F74
LIN-DRV or BUS-DRV	line drivers, with enhanced current drive and extra capacitance; e.g., 74F244, 74AC245
OPEN-COL	open-collector; e.g., 7406
OPEN-DRN	open-drain
PLD	any PLD or FPGA
FST/SLW	fast or slow edge of a device with programmable slew rate

The naming format is used especially to distinguish between the normal-output devices and the line drivers in standard-logic families. It also identifies models with special driver output stages, like open collector or fast/slow versions of devices with programmable output slew rate.

Other models — especially those for more-specialized devices — use only the common device name, without specifically identifying the model “type.” This is true, for example, of bus-driver families which come only with line-driver outputs.

Modeling Bi-directional Devices

This section describes how to choose models for bi-directional devices when you are using a .MOD library, especially GENERIC.MOD.

When you are modeling a driver with a high-current output, e.g., an 74xx244 or 74xx245 (where xx is any technology type, like “HC” or “F”), use the line-driver version of the model.

When you are modeling a receiver, if the device is bi-directional (i.e., a transceiver) and you are modeling the case where the driver is tristated and the device is receiving, use the line-driver version. (It correctly models the extra capacitance of the tristated driver.)

But if the device is *not* bi-directional and you are modeling one of the input pins, use the “gate” model, *not* the line-driver. The input in this case is no different than a standard gate’s input (no extra capacitance).

The following table summarizes the correct choices for the 74xx24x series:

Device	Correct Driver Model	Correct Receiver Model
74xx240	LIN-DRV	GATE
74xx241	LIN-DRV	GATE
74xx242	LIN-DRV	LIN-DRV
74xx243	LIN-DRV	LIN-DRV
74xx244	LIN-DRV	GATE
74xx245	LIN-DRV	LIN-DRV

Hint: *GENERIC.MOD contains two models — 74HCXX:GATE and 74HCTXX:GATE — that each have two versions. The “-1” version has low-resistance clamp diodes, and the “-2” version has high-resistance. These families come in two versions, depending on the manufacturer. The low-resistance version clamps overshoot and undershoot more effectively. If you’re not sure which version you’re using, check with the manufacturer.*

The EASY.MOD Library

Sometimes you are in a hurry to simulate and don't have an exact model for a device. In these cases, when there's no time to find the appropriate model (e.g., from the vendor) or create one yourself, use the EASY.MOD library to get an approximate model that will get you simulating, and — in most cases — be sufficiently accurate to give good analysis results.

EASY.MOD is based on the fact that the two most-important parameters in a driver-IC model are the basic technology type (CMOS or bipolar?) and the approximate switching time of the output buffer (fast? slow? medium?). If you know those two basic facts about an IC (usually easy to glean from a data sheet), you can choose a model from EASY.MOD. While the model may not be exactly perfect, it will usually be sufficient to proceed with simulation.

To choose a model from EASY.MOD:

1. In the Select IC Model dialog box, in the Library list box, highlight EASY.MOD.
   OR
   Click the EASY.MOD button.

2. Determine the basic technology type of the IC you're modeling. If it is a CMOS (or similar, e.g., NMOS) device, look in the Device list box at the model names beginning with "CMOS". If it is bipolar, look at the models beginning with "TTL".

3. If the device is CMOS, determine if it runs from a 5.0-V or 3.3-V power supply. Narrow the models to those with the appropriate power supply.

4. Determine approximately how fast the IC switches. The data sheet for the device should give at least some feeling for this. If you don't know the answer to this question, make a guess based on the IC's "age." If it is a brand-new device, for example, it likely switches fairly fast. If an older device, it may switch at a medium or even slow rate. If a specialty device intended for very-high-speed applications (e.g., a clock driver or a specialized bus technology), likely it switches "ultra-fast." If you have absolutely no idea about switching speed, choose the "FAST" model.

5. Based on these decisions, narrow the model to a single choice. In the Device list box, double-click on the appropriate choice.

Cannot Save into EASY.MOD

You can edit a model in EASY.MOD, but you cannot save the edited model back into EASY.MOD; you must save to a different library. The model editor will not write into EASY.MOD. (See Chapter 5, section “Editing .MOD IC Models” for details on editing .MOD models.)

Note: *If you modify a HyperLynx-supplied library, you should rename it first. If you do not change the library's name, your version of the library will be overwritten next time you receive updated HyperLynx software.*

Other Libraries

One other useful library in LineSim is DIODES.MOD, which contains models for several generic types of clamp diode. If you are using external clamp diodes for termination, for example, you might try one of the models in DIODES.MOD. (Or start with a model in DIODES.MOD and modify it to match the diode you're using.)

When you use a model in DIODES.MOD, be sure to set the model's buffer state to Input rather than Output. (Diodes are passive devices; they don't “drive.” The output sides of the models in DIODES.MOD are completely “open.”)

MOD Libraries

Choosing a Device

.If you have chosen a .MOD library in the Select IC Model dialog box, the next step is to choose a device. In some cases (particularly if the library is GENERIC.MOD), you will actually choose a device family rather than a specific device. (See “Model Names in GENERIC.MOD” above in this chapter for details.)

For a .MOD library, the Pin/Signal list box is grayed out, because .MOD libraries do not contain lists of component pins/signals. For the same reason, the Select By radio buttons are grayed out, too. Also, because .MOD models always model both driver (i.e., output) and receiver (input) functionalities, the picture in the I/O Type area shows both directions. If the model contains threshold information, the I/O Type area shows it, too.

Since the library and device are the only choices you can make for a .MOD model, you can double-click on the device name to close the dialog box and make your choice.

Model Information

The Information on Selected Device area displays the current library and device choices. The Signal and Pin choices are grayed out, since they do not apply for .MOD libraries. For .MOD libraries, there is no “source” information; the Notes field does not carry model-specific information.

Searching for .MOD Models

For details on searching for an appropriate .MOD model, see “How to Search for an IC Model” below.

IBIS Libraries

Choosing a Device

If you have chosen a .IBS library in the Select IC Model dialog box, the next step is to choose a device. IBIS libraries show explicit IC component names in the Device list box. (There can be one or multiple ICs in a single IBIS library.)

When you highlight a device, the Pin/Signal list box updates to display the contents of the highlighted device. The list-box title changes depending on whether you choose to sort pin/signal names by pin name or by signal name.

To change how pin/signal names are sorted:

1. In the Select By area, click the appropriate radio button.

Choosing a Pin or Signal

The last step is to choose a device pin/signal.

You can double-click on the pin/signal name to close the dialog box and make your choice. There is sometimes a short pause after you double-click (or click OK), while the model data is loaded from the IBIS file into memory.

The Model Information Area

The Information on Selected Device area (near the IC icon) displays the following information for an IBIS library:

- ♦ library name
- ♦ device name
- ♦ signal name
- ♦ pin name
- ♦ informational notes

All information is for the currently highlighted pin/signal.

If you are sorting By Pin, the information area is a way to see what signal name corresponds to the pin name you have highlighted (and vice versa).

The Notes Field

The Notes field contains information supplied by the creator of the model, e.g., who created the model and when, how the model is copyrighted, revision history, limitations or recommendations on the model’s use, etc. To view all of the information, pull the combo box down and scroll if necessary.

Searching for IBIS Models

For details on searching for an appropriate IBIS model, see “How to Search for an IC Model” below.

Previewing Model Directionality/Type

The I/O Type area shows graphically the directionality/type of any pin’s or signal’s model (e.g., output-only, input-only, bi-directional, 3-state output, etc.). The picture changes as you highlight various pins/signals in the Pin/Signal list box.

This feature allows you to see directionality/type without having to actually select the pin/signal and return to the Assign IC Models dialog box.

Note: *The I/O Type area also displays input-receiver and output-driver switching thresholds for the highlighted pin's or signal's model. These values are currently unused in LineSim, but are used by BoardSim to calculate timing delays. For more information on these thresholds, see the BoardSim User's Guide.*

PML Libraries

Choosing a Device

If you have chosen a .PML library in the Select IC Models dialog box, the next step is to choose a device. .PML libraries show explicit IC names in the Device list box. (There are typically many ICs in a single .PML library, although there may be only one.) .

When you highlight a device, the Pin/Signal list box updates to display the contents of the highlighted device. The list-box title changes depending on whether you choose to sort pin/signal names by pin name or by signal name.

To change how pin/signal names are sorted:

1. In the Select By area, click the appropriate radio button.

Choosing a Pin or Signal

The last step is to choose a device pin/signal.

You can double-click on the pin/signal name to close the dialog box and make your choice. There is sometimes a short pause after you double-click (or click OK), while the model data is loaded from the .PML file into memory.

The Model Information Area

The Information on Selected Device area displays the following information for a .PML library:

- library name
- device name
- signal name

- pin name
- informational notes

All information is for the currently highlighted choice.

If you are sorting By Pin, the information area is a way to see what signal name corresponds to the pin name you have highlighted (and vice versa).

The Notes Field

The Notes field contains information supplied by the creator of the model, e.g., who created the model and when, how the model is copyrighted, revision history, limitations or recommendations on the model's use, etc. To view all of the information, pull the combo box down and scroll if necessary.

Previewing Model Directionality/Type

The I/O Type area shows graphically the directionality/type of any pin’s or signal’s model (e.g., output-only, input-only, bi-directional, 3-state output, etc.). The picture changes as you highlight various pins/signals in the Pin/Signal dialog box.

This feature allows you to see directionality/type without having to actually select the pin/signal and return to the Assign IC Models dialog box.

If a model contains threshold information, the I/O Type area shows it, too.

Check Mark Status

After you have chosen the model, you return to the Assign IC Models dialog box. The model’s default direction and state depend on several factors:

- If the model is output-only (can’t be an input), the default state is “Output”
- If the model is input-only (can’t be an output), the default state is “Input”
- If the model can be an input or an output, and there was previously a model specified for the pin, then the default state matches the direction/state set for the previous model

- ♦ If the model can be an input or an output, and there was NOT previously a model specified for the pin (i.e., red question mark), then the default state is “Input”

For more details on setting buffer direction/state, see “Setting IC Buffer Direction/State” below.

Setting IC Buffer Direction/State

To change an IC model’s buffer direction/state:

1. In the Buffer Settings area, click the appropriate radio button.

Only the buttons for the buffer states that are valid for the model are displayed. See “Possible Buffer States” below for more details on possible states.

If you change a model from state Input to some type of Output, or from some type of Output to Input, the icon in the Pins list box changes direction. The icon for an Input points to the left; the icon for an Output points to the right.

There is also a picture in the Buffer Settings area that shows graphically the direction and type of the currently chosen model.

Note: *.MOD models always include both driver and receiver data, so if you choose a .MOD model, both the Input and various Output radio buttons are available. IBIS and .PML models can be unidirectional or bi-directional, so sometimes only a few buffer states are available (e.g., if a receiver pin, only Input).*

Possible Buffer States

IC models can have nearly any mixture of the following buffer states:

- ♦ **Output** — Indicates a driving or output state; the sense of the output is “true,” meaning that in the oscilloscope, the model will follow the sense of the Driver Waveform setting (i.e., will fall on a falling edge and rise on a rising edge). If the oscilloscope is set to waveform type “Oscillator,” the model will start with a rising edge.

- **Output Inverted —** Indicates a driving or output state, but with inverted sense, meaning that in the oscilloscope, the model will invert the sense of the Driver Waveform setting (i.e., will rise on a falling edge and fall on a rising edge). If the oscilloscope is set to waveform type “Oscillator,” the model will start with a falling edge. This setting is most useful for differential-driver pairs, where one pin’s model must be inverted relative to the other.

- **Output Hi-Z —** Indicates a passive, high-impedance state in which a driver or output model has “turned off.” This selection is available exclusively for output-only models that have “tristate” capability; the high-impedance state of an I/O or bi-directional model is selected with state “Input.”

- **Input —** Indicates a receiving or input state. In this state, an I/O or bi-directional model’s driver is turned off, and the pin is receiving. For input-only pins, this is the only available state.

- **Stuck High —** Indicates a driving or output state, but one in which the model stays high and never switches. In the oscilloscope, the model will stay high regardless of the Driver Waveform setting. This setting is useful for wired-OR or wired-AND buses for which you want to investigate the effect of one driver switching while other drivers remain static. It is also extremely important for crosstalk simulations, in which the driver on “victim” net is usually simulated in a static, “stuck” state (see the Crosstalk User’s Guide for details).

- **Stuck Low —** Indicates a driving or output state, but one in which the model stays low and never switches. In the oscilloscope, the model will stay low regardless of the Driver Waveform setting. This setting is useful for wired-OR or wired-AND buses for which you want to investigate the effect of one driver switching while other drivers remain static. It is also extremely important for crosstalk simulations, in which the driver on “victim” net is usually simulated in a static, “stuck” state (see the Crosstalk User’s Guide for details).

Which buffer states are available for any particular model is determined by the model’s internal “type.” In practice, no model can simultaneously have all of the buffer types described above. (For example, types “Input” and “Output Hi-Z” are mutually exclusive.)

Input versus Output Hi-Z

Two buffer states — Input and Output Hi-Z — are similar to each other, but not quite the same. Both generally refer to a state in which an IC pin is high-impedance and not actively driving the net to which it is connected. The difference is that Input is available on either “pure” input pins (pins that can never drive) or on I/O or bi-directional pins that can act as inputs when the output or driving circuitry is shut off; whereas Output Hi-Z is available on pins that can only drive or be turned “off,” but have no receiver-input stage.

If you have an IC pin for which the data sheet refers to the “high-impedance” state, but when you model it in LineSim there is no Output Hi-Z selection available, choose buffer state Input instead. This means that the pin is actually I/O; when the data sheet refers to “high-Z” it means that the driving circuitry is disabled and the pin is in a high-impedance receiving (i.e., “input”) state.

Threshold Voltages

The Buffer Settings area also displays the input-receiver and/or output-driver switching thresholds present in the IC model for the currently highlighted pin. Only the thresholds relevant to current buffer-state choice are displayed, e.g., if for a bi-directional pin you set the buffer state to “Input,” the input thresholds Vih and Vil are shown; if you change the state to “Output,” the output-switching threshold “Vmeasure” is shown.

Setting the Vcc or Vss Pin

For every IC pin with its buffer direction/state set to some type of Output, LineSim allows you to choose a Vcc and a Vss pin, which in turn allows you to power the driver IC from any two power-supply nets. (You can also power the driver from the “typical” power-supply voltage contained in the model itself; see ”Typical’ Power Supplies” below for details.) The models for the driver and all the receivers on the net then run off the voltages to which the Vcc and Vss pins are connected.

LineSim has a fixed set of “built-in” power-supply nets. You can use any of them to power your ICs. You can also change each power-supply net’s voltage. For details on managing power-supply nets, see Chapter 8.

“Typical” Power Supplies

You can run either or both of a driver IC’s supply pins from the “typical” power-supply voltages specified in the driver’s model, rather than connecting explicitly to one of LineSim’s built-in power-supply nets. This is useful primarily because it frees you from having to even worry about what the supply voltage(s) need to be. See “If Both Vcc and Vss are Set to ‘Typical’” below in this chapter for more details on typical power supplies.

Note: *The typical values are built-in to every IC model (whether .MOD, .PML, or .IBS) and specify the voltages at which the IC most commonly runs*

With Multiple Drivers

If multiple driver ICs with different default power-supply values are present in the simulation, LineSim will attempt to connect each to a unique and correct power-supply value. However, the automatic connections may not always be correct, so it is wise to check which supplies each driver IC is running from before simulating. If you run a simulation and see unexpected initial or final voltages, check the driver-IC supply-pin settings.

How LineSim Defaults Vcc and Vss Pins

If you activate a new driver IC in a schematic and choose an IC model, or change an already-activated IC’s model, then the IC is automatically connected to power-supply nets Vcc and Vss. If the present voltages of Vcc and Vss do not match the typical power-supply values in the driver’s model, then you are queried about whether you want the values of the Vcc and Vss power supplies to be changed to match the model’s typical values.

If you load a .TLN file with a driver in it, the same steps occur as in the preceding paragraph, except that the voltage change on Vcc and Vss is automatic, with no querying.

Changing a Vcc or Vss Pin

To change a Vcc or Vss pin:

1. In the Assign IC Models dialog box, in the Pins list box, highlight a driver IC's pin.
2. In the models area, pull down the Vcc Pin or Vss Pin combo box.
3. Choose a pin that connects to the desired power-supply net. (Note that the model's "typical" supply value is available at the top of the combo box's list.) If no power-supply net with the proper voltage is available, choose a supply, then use the power-supply editor to change the voltage on the net to which you attached. See Chapter 8, section "Editing Power-Supply Nets" for details on changing power-supply voltages.

If Both Vcc and Vss are Set to 'Typical'

If both the Vcc Pin and Vss Pin values are set to "use model's 'typical' value", one of the values automatically becomes 0.0V. If the typical supply voltage is positive, then Vss=0.0V; if the typical supply voltage is negative, Vcc=0.0V.

Vcc and Vss Errors

If you set the Vcc and Vss Pin values — or the voltages for the corresponding power-supply nets — such that Vcc is more negative than Vss, or the difference between Vcc and Vss is less than 0.5 V, LineSim will give an error when you attempt to simulate. Correct the values before proceeding.

Other Parameters and Power Supply

If you change a Vcc or Vss voltage to a value other than the typical value specified in a driver's model, LineSim does not automatically correct the other parameters in the model to reflect the new supply voltage.

For example, any of LineSim's .MOD-format 5-V CMOS models can be made to run at VCC=3.3V, but the model's slew rates, "on" impedances, etc. may no longer be correct; compensating them is up to the user. (On the other hand, the models created specifically for 3.3V are correct at 3.3V, but may need compensation to run at, say, 2.5V.)

Reminder About Power Supplies for Receiver ICs

For every IC pin with its buffer direction set to some type of Output, LineSim allows you to choose a Vcc and Vss pin. The simulation models for the driver and all the receivers on the net then run off the voltages to which the Vcc and Vss pins are connected. In the current version of LineSim, you cannot set Vcc or Vss pins for receiver ICs; receivers must run off the same supply voltage as the drivers on a net.

How to Search for an IC Model

If you are in the Select IC Model dialog box, but don't know in which library to find an appropriate device model, LineSim includes a model-finding feature that may help. The model finder accesses in a spreadsheet a database listing of all models available in HyperLynx's shipping software; you can use this database to search for component names, look for all of the IC models from a particular vendor, and so forth.

To open the model-finding database:

1. From inside the Select IC Model dialog box, click the Find Model button. After a brief pause, the IC Model Finder dialog box opens. (The pause occurs because the database of available IC models is very large, and takes some time to load.)

2. If desired, adjust the size of the finder's spreadsheet window to be larger, by grabbing it with the mouse on an edge or in a corner and dragging it to the desired size.

The model-finder spreadsheet lists all of the component models available when HyperLynx last compiled its libraries of device models (thousands of entries). You can search this list for particular components or device names, and sort on various criteria to group together, e.g., all ICs from a particular vendor, all recent models, etc.

To search for a text string in the model-finder spreadsheet:

1. In the Search String box, type the text for which you want to search.

2. Click the Search for String button. After a pause, all of the items matching the string (in any data column) are check marked in the far left-hand column and brought to the top of the spreadsheet.

To sort the spreadsheet based on a column's data:

1. Click the header button (at the top of the column). After a pause, all of the items in the spreadsheet are sorted in ascending order of the column's data.
2. To sort in the opposite order, click the column's header button again.

For example, to bring to the top of the spreadsheet the most-recent models in the database, click once on the File Date column's header button; after a pause, the list is sorted from oldest to newest model (ascending order). Click again, and the list changes to newest-to-oldest order (descending order).

Selecting a Model Directly from the Model Finder

If you find in the model-finder spreadsheet a model that you want to use in the Select IC Model dialog box, you can select it directly from the spreadsheet.

To choose a model directly from the spreadsheet:

1. Click once on the model's line in the spreadsheet, to highlight it.
2. Click OK. The model-finder dialog box closes and the model is automatically selected in the Select IC Model dialog box.

Updating the Model Finder's Database

The model finder stores its database in a comma-separated-values file called HyperLynxIcModels.CSV; this file resides in the LIBS subdirectory along with all of your IC models (.IBS, .MOD, and .PML).

Occasionally, you may want to update the .CSV database file, for example to add to it models that you have obtained or created on your own. LineSim includes a built-in utility for generating an updated database.

To generate an updated model-finder database, which includes all of the models currently in your LIBS subdirectory:

1. From the File menu, choose Generate Model Finder Index.

2. A DOS or command-prompt window opens, and the HyperLynx utility “MAKE_CSV” runs. When it finishes, the DOS box closes and the file HyperLynxIcModels.CSV has been updated.

The database file now includes information about every model currently in your LIBS subdirectory. Any models that you have deleted from the directory are no longer in the database. However, your previous database file has been saved as HyperLynxIcModels.BAK, in case you need to restore it.

Copying IC Models

Once you have interactively chosen a model for one IC pin, it may be convenient to quickly “copy” the same model to other pins. This is particularly true of receiver models, and of .MOD models generally since they are not pin-specific.

For example, suppose a net has one driver (i.e., “output”) and ten receivers, and all the receivers are 74ACxxx inputs. Once you have chosen a model for one of the receivers, it is convenient to paste it immediately to the other receivers on the net.

Copying an Existing Model

To make a copy of a model:

1. In the Assign IC Models dialog box, in the Pins list box, highlight an IC pin that has been assigned the model you want to copy.
2. In the Model to Paste area, click the Copy button.

The information in the Model to Paste area updates to display the copied model. Also, the Paste and Paste All buttons become available (ungrayed).

The Model-to-Paste Information Area

The Model-to-Paste area shows the following information for the copied model:

♦ library name

♦ device name

- signal name
- pin name

All information is for the last-copied IC model.

If you highlight a different component pin in the Pins list box, the information in the Model-to-Paste area remains the same. The Model-to-Paste area is a storage buffer: it always remembers the model you last copied.

Pasting to Another IC Pin

To paste the last-copied model to another IC pin:

1. Be sure that the model you want to paste is displayed in the Model to Paste area. (See “Copying an Existing Model” above for details on copying a model.)
2. In the Assign IC Models dialog box, in the Pins list box, highlight the IC pin to which you want to copy the model.
3. Click the Paste button.

After a brief pause (while the model data is loaded into memory):

- the Pins list box updates with the new model assignment
- the model-information area updates; the fields display the new model’s name and other data

A pasted model defaults to the buffer direction/state of the model that was copied. See “Setting IC Buffer Direction/State” above in this chapter for details on changing the direction.

Pasting to All Other IC Pins

To paste the last-copied model to all other IC pins:

1. Be sure that the model you want to paste is displayed in the Model to Paste area. (See “Copying an Existing Model” above for details on copying a model.)

2. Click the Paste All button.

When the operation is complete, the Pins list box updates with the new model assignments.

A pasted model defaults to the buffer direction/state of the model that was copied. See “Setting IC Buffer Direction/State” above in this chapter for details on changing the direction.

Quickly Creating Multiple Receivers and One Driver of the Same Type

Sometimes it is convenient to make all of the ICs in a schematic use the same model, with one of the pins set as a driver (i.e., an “output”), and the rest as receivers (“inputs”). The Paste All command can be used to do this efficiently, as follows:

To set all pins in the schematic to the same model, with one pin as a driver:

1. Select the desired model for one pin in the schematic. In the Buffer Settings area, set the pin’s direction/state to Input.

2. With the same pin still highlighted in the Pins list box, click the Copy button.

3. Then click the Paste All button. All of the other ICs in the schematic are set to the same model, with every pin’s direction/state set to Input.

4. Now highlight the pin which you wish to be the driver. In the Settings area, click the Output radio button.

Note that if you set the initial pin to state Output before copying and pasting, all of the other pins will also be set during the Paste operation to state Output. Since you actually want them to be Inputs, be sure to perform the copy operation with the pin set to Input, and change it later (after the Paste) to Output.

Changing IC Models

To change a pin’s IC model from one choice to another, simply re-choose the pin’s model. Follow the steps in “How to Choose an IC Model” above in this chapter.

How to Choose Models for a Differential Driver or Receiver

The steps for choosing models for a differential-IC driver or receiver are the same as those for choosing a model for “regular” non-differential IC, except that in the differential case you must choose two models (one for each half of the differential pair) and invert one of the pin’s buffer states. For details on choosing models generally, see “How to Choose an IC Model” above in this chapter.

To choose IC models for a pair of differential pins:

1. In the Assign IC Models dialog box, in the Pins list, highlight a pin on the differential pair.
2. Choose a model for the pin (as described above in “How to Choose an IC Model”).
3. Then highlight the second pin in the differential pair, and choose its model.
4. Of the two pins in the differential pair, in the Pins list box, highlight the one that is the ‘+’ or “positive” pin, and set its buffer direction/state to Output.
5. Highlight the ‘-‘ or “negative” pin, and set its direction/state to Output Inverted.

You can use any type of IC model — IBIS, .MOD, or .PML — in a differential simulation.

Removing IC Models

Occasionally, you may want to remove a previously selected IC model from a component pin, so that the pin has no model. This is equivalent to ”lifting” an IC pin from your board. There are two ways to do this:

To remove an IC model from a pin (method #1):

1. In the Assign IC Models dialog box, highlight in the Pins list box the pin whose model you want to remove.
2. Click the Remove button.

The model previously assigned to the pin is removed, and the green driver or receiver icon changes back to a red question mark.

To remove an IC model from a pin (method #2):

1. In the schematic editor, point to the IC whose model you want to remove.
2. Click with the left mouse button. The IC de-activates, and disappears from the schematic.

Hint: *Either method gives you a way to experiment to see how simulation results are affected by the removal of one or more ICs.*
To add the model back in, if removed with method #1, right-click on the IC and choose the model again in the Assign IC Models dialog box. If removed with method #2, left-click on the IC's “ghost” in the schematic; the re-activates and its previous model is automatically “remembered.”

Editing Passive Components (Resistors, Capacitors, and Inductors)

For passive components other than ferrite beads (resistors, capacitors, and inductors), there is no model to choose, but you can edit the component's value. In fact, you will almost always need to edit a newly activated component's value, since a component's default value is rarely what you want.

Changing a Resistor, Capacitor, or Inductor Value

To change a resistor, capacitor, or inductor value:

1. Point in the schematic to the resistor, capacitor, or inductor. When you point to the component, a red box appears around it.
2. Click with the **right** mouse button. The Edit Resistor Values, Edit Capacitor Values, or Edit Inductor Values dialog box opens.
3. In the Resistance, Capacitance, or Inductance edit box, type the component's value. Values can be entered as a "simple" number (e.g., "1000" or ".01") or in scientific notation (e.g., "1e3" or "1e-2"). Click OK.

The value is measured in ohms for resistors, in picoFarads (pF) for capacitors, and in nanoHenries (nH).

When you close the editing dialog box and return to the schematic editor, the resistor's, capacitor's, or inductor's value (labeled in blue) has been updated.

Choosing Ferrite-Bead Models

Ferrite beads require a model more complex than that for other, simple passive components (e.g., a resistor). Other passive components are modeled with a single value (e.g., "100 ohms" or "33 pF"); ferrite beads are modeled more like an IC, with a detailed model contained in a library. "Choosing a model" for a ferrite bead means selecting a model from one of the libraries supplied with LineSim.

LineSim ships with a library of representative ferrite beads from several leading manufacturers. You can also create your own ferrite-bead models. For details on how LineSim models beads, and how to create your own bead models, see Chapter 5, section "Creating Your Own Ferrite-Bead Models."

How to Choose a Ferrite-Bead Model

To choose a ferrite-bead model:

1. Point in the schematic to the ferrite bead. When you point to the component, a red box appears around it.
2. Click with the **right** mouse button. The Select Ferrite Bead Model dialog box opens.
3. In the Vendor list box, highlight the vendor for whom you want choose a model. The Part Number list box updates to display a list of beads from the highlighted vendor.
4. Highlight the part number matching the bead for which you want a model. The data in the Model Values area and the impedance-vs.-frequency graph update to display the modeling parameters for the highlighted bead.
5. Click OK.
   OR
   Double-click on the part number.

   The Select Ferrite Bead Models dialog box closes, and the model is chosen.

For more details about the operations described in these steps, see the following sections.

About the Ferrite-Bead Models Supplied by HyperLynx

All of the models listed in the Select Ferrite Bead Model dialog box (unless you have created additional models of your own) are contained in a HyperLynx-supplied library called “BSW.FBD.” This library contains a representative sampling of beads from several of the leading manufacturers. Usually, even if the bead you’re using is not contained in the library, you can find a close match to it in BSW.FBD.

If you want to create a bead model of your own, you can do so by creating a file called “USER.FBD.” For details on how to create your own bead models, and how LineSim models ferrite beads generally, see Chapter 5, section “Creating Your Own Ferrite-Bead Models.”

The Model Values Area

For the currently highlighted vendor and part number, the Model Values area shows an equivalent L-R-C model for the bead. These values are automatically synthesized from three of the bead's impedance-vs.-frequency points.

The Impedance-vs.-Frequency Graph

Above the Model Values area and Vendor and Part Number list boxes is a graph showing the impedance of the currently highlighted bead versus frequency. It is this curve that determines how a given bead responds on your board when terminating a signal.

If you are trying to find a model for a bead which is not specifically listed in BSW.FBD (see "About the Ferrite-Bead Models Supplied by HyperLynx" above for details on the BSW.FBD library), look through the HyperLynx-supplied models for the one whose impedance curve best matches the curve for the bead you're using. Key parameters are the peak impedance and the overall shape of the curve (is it fairly flat, or sharply peaked?).

Ferrite-bead data sheets normally show impedance-vs.-frequency curves. If your data sheets don't include curves, contact your bead vendor and demand more information.

Ferrite beads are often described in terms of their impedance at a nominal frequency, usually 100 MHz. However, simply because two vendors' beads are both called "120-ohm" (both have approximately 120 ohms' impedance at 100 MHz) does not mean they behave the same in-circuit. Look at their complete impedance curves to determine how similar they actually are.

Simulating Before a Ferrite-Bead Model is Chosen

If you run a simulation before a model is chosen for a ferrite bead (when it still displays with a red question mark in the Pins list box), the ferrite bead is modeled as a 0.0-ohm resistor. This is equivalent to there being no bead present, or the bead being shorted out, except that the "resistor's" parasitic leads are present.

Changing Ferrite-Bead Models

To change a ferrite bead's model from one choice to another, simply re-choose the bead's model. Follow the steps in "How to Choose a Ferrite-Bead Model" above in this chapter.

Updating Models over the Internet

On computers that are connected to the Internet, LineSim has the ability to automatically connect to HyperLynx's Internet site and update the IC models in your HyperLynx LIBS directory. This feature — called "Instant Online Models" — gives you an easy way to ensure that you always have HyperLynx's latest IC models.

Requirements for Accessing Online Models

In order to access HyperLynx's online models, your PC must have access to the Internet. This means that your machine must either have:

- a permanent Internet connection, ***OR***
- dial-up Internet access, and you must dial in before attempting to access the models

Retrieving Models Online

To access the latest IC models from HyperLynx's Internet site:

1. Verify that your computer is presently connected to the Internet (dial first, if your access requires dial-up).
2. Verify that LineSim is pointing to your model subdirectory: from the Options menu, choose Directories. Verify that the Model Library File Path is set correctly.
3. Then, from the File menu, choose Download IC Models from HyperLynx.
4. The Accessing HyperLynx WWW Site dialog box opens; there may be a pause while a connection to HyperLynx's site is opened and verified. If the

site cannot be accessed (e.g., your computer's Internet connection is "broken"), then after a brief time-out period, an error message will appear.

5. Once the connection to HyperLynx's site is established, the Download an Update dialog box opens. In the Select a Compressed File to Download list box are shown several compressed files that each contain the latest models of a particular kind. For example, the "ibis" file contains all of the latest IBIS models available from HyperLynx. For a description of what each file contains, see the File Descriptions area in the dialog box.

6. Choose a file to download by clicking once on the file in the list box to highlight it.

7. Click the Download button. After a brief pause, the Downloading File dialog box opens to show you the status of the download. When downloading is complete, the status dialog closes and a check mark appears beside the downloaded file; the compressed model file has been placed in your LIBS subdirectory (see step 2 above), uncompressed into multiple models, and deleted. The names of the model files appear near the top of the dialog box as they are uncompressed.

8. Repeat for additional model files.

9. When you are finished updating, click the Close button. Your IC models have been automatically updated.

If You Can't Download Models

If you try to download models but the operation never successfully completes (you get time-out or error messages), the feature is probably failing due to permission problems with a firewall. Unfortunately, there is little you can do to fix this, except trying from another site that is outside the firewall.

Retrieving Models Directly from Manufacturer Web and FTP Sites

Many IC manufacturers now make IBIS IC models available directly from their World Wide Web or FTP sites. While HyperLynx attempts to gather up and make available to customers as many of these models as possible, some manufacturers do not allow their models to be shipped with third-party products. Also, new models are appearing constantly.

Accordingly, you are strongly encouraged to browse manufacturer sites yourself to see what additional IBIS models are available to you. For details on an easy way to do this, see Chapter 12, section "Using the HyperLynx IBIS-Model Web Page."

IC and Terminating-Component Parasitics

LineSim models not only ICs and passive components themselves, but also the packages in which they're housed and the leads that attach them to a PCB. The electrical characteristics of packages and leads are called "parasitics."

For example, in reality, a resistor is not *purely* a resistor; because of its leads and the capacitance between its body and ground, it also has small amounts of inductance, capacitance, and extra resistance. Similarly, an IC's package adds to the silicon itself parasitic L, C, and R.

Default Parasitics

By default, LineSim adds a small amount of parasitic L, C, and R to every IC and passive component. These default values are appropriate for modeling modern, physically small surface-mount packages.

Hint: *If your systems are mostly surface-mount, LineSim's default parasitics are probably sufficiently accurate that you'll never need to change them.*

What are Parasitics?

For ICs

For an IC, the parasitic L, C, and R model the following:

Parasitic	**Description**

Parasitic	**Description**
inductance	the series inductance in the IC pin's package lead
capacitance	the capacitance from the IC pin's lead to the nearest AC-ground plane
resistance	the series resistance in the IC pin's package lead

Parasitics and .MOD Models

.MOD models do not include information about an IC's package. (See "IC-Model Formats" above in this chapter for details on the .MOD format. IBIS and .PML models *do* often include package information.)

However, you can use parasitics to add package information to .MOD-based ICs in your schematic. See "Changing Parasitic Values" below for details on how to change an IC's parasitic values. Note that these values reside at specific locations in a particular schematic; you cannot save the parasitic values into a .MOD model itself.

You can also use .PML models, which extend the .MOD format to include component-specific pin-outs and package parasitics. For details on the .PML format, see "IC-Model Formats" above in this chapter.

Parasitics and IBIS/.PML Models

IBIS and .PML models may optionally contain package parasitics in the model itself. (Most IBIS models have at least a component-wide package model; almost all .PML models include package data. See "IC-Model Formats" above in this chapter for details on the .IBS and .PML formats.) **For IBIS and .PML models, LineSim uses the package parasitics contained in the model.**

For Passive Components

For a passive component (resistor, capacitor, inductor, or ferrite bead), the parasitic L, C, and R model the following:

Parasitic	**Description**

inductance	the series inductance in the component's package lead
capacitance	the body capacitance from the component itself to the nearest AC-ground plane
resistance	the series resistance in the component's package lead

Changing Parasitic Values

LineSim allows you to change a .MOD IC's or passive component's default parasitic values. You might want to do this, for example, to more-exactly model a component's package.

Note: *Parasitics are not supported for IBIS or .PML models. IBIS and .PML models contain package information internally.*

For ICs

To change an IC's parasitic pin values:

1. Point in the schematic to the IC. When you point to it, a red box appears.
2. Click with the **right** mouse button. The Assign IC Models dialog box opens.
3. In the Pins list box, highlight the IC pin for which you want to change the parasitic values.
4. At the top of the dialog box, click the Pin Parasitics tab.
5. In the edit boxes, type the new parasitic values.
6. Click Close.
   OR
   To change other IC pins' parasitics, click the IC Model tab; in the Pins list box, highlight another IC pin; and repeat steps 4 and 5.

For Passive Components

To change a passive component's parasitic values:

1. Point in the schematic to the component. When you point to it, a red box appears.
2. Click with the **right** mouse button. The appropriate dialog box opens.
3. At the top of the dialog box, click the Parasitics tab.
4. In the edit boxes, type the new parasitic values. Click Close.

Saving Parasitic Values

Any changes you make to components' parasitic values are saved in the .TLN file when you save your schematic. See Chapter 3, section "Saving a Schematic" for details on saving schematics.

New IC Model Overwrites IC Parasitics

For an IC component, the parasitic values are overwritten when you choose a new model; the new values are the default parasitic values. Therefore, you should choose IC models first, and after you're fairly certain that you have in place the models you want, *then* change IC parasitics.

Note: *This applies only to IC parasitics. For passive components, once you've changed parasitic values, they remain intact.*

Recommendations on When to Change Parasitics

In most cases, the default parasitics are reasonable values for modern, small surface-mount packages.

However, if you are using predominantly through-hole technology or devices with older-style, physically larger packages *and* using drivers with fast edge rates (e.g., 2 ns or faster), you may want to model your packages more exactly by changing the parasitics. This is especially true for:

- fast drivers in old, large packages (e.g., DIPs)

- through-hole passive components (e.g., axial-leaded resistors and capacitors)

You can also change parasitic values if you want extremely high-accuracy simulation with surface-mount components. You will generally find yourself *increasing* LineSim's parasitic values, even for most surface-mount packages.

See the note above in "New IC Model Overwrites IC Parasitics" about when in the process of entering a schematic it's best to change IC parasitics.

Cannot Set Parasitics to Zero

You cannot set a component's parasitic values to zero (although you *can* make them insignificantly small). If you try to set a parasitic value to zero, LineSim will issue a warning and set the value to a non-zero — though very small — number.

Hint: *HyperLynx recommends against setting parasitics to extremely small values, because doing so slows the simulator, sometimes significantly. Remember that real components always have some measurable parasitic L, C, and R.*

Chapter 5: Editing IC and Ferrite-Bead Models

Summary

This chapter describes:

- ♦ how to edit a .MOD model
- ♦ how to edit an IBIS model
- ♦ where model libraries are stored, and how to point to them
- ♦ how to create your own ferrite-bead models

Editing .MOD IC Models

IC models created in the .MOD format can be edited directly inside LineSim, with the .MOD model editor. If you are unfamiliar with the .MOD format or how it compares to the IBIS modeling format (another type of IC model supported by LineSim), see Chapter 4, section “IC-Model Formats.” (A third supported format, .PML, is an extension of .MOD that adds component pin-out and package characteristics.)

The .MOD Format and Editor

The .MOD format is an ASCII format first created by HyperLynx in 1989 for its LineSim Pro product, and supported now in the Windows versions of LineSim and BoardSim. Though .MOD models are saved into ASCII libraries, the format is not keyword-based and therefore should not be edited directly with a text editor.

Instead, LineSim includes a .MOD model editor that allows you to edit models in a Windows dialog box. The editor allows you to modify existing models and create new ones; new models can be saved into user-defined libraries. .MOD models are based on parameters commonly found in IC databooks, to make the models as easy as possible to create and support.

.MOD Model Parameters

This section describes the parameters that together make up a .MOD model. All of the parameters can be edited by the user.

Output Vs. Input

Every .MOD model contains *both* output driver and input receiver information. This allows a single .MOD model to describe a typical output and input for an IC or family of ICs.

Note: *For ICs with multiple output buffer or input buffer types, multiple .MOD models can be saved into one IC- or family-specific library. If you need to model only an output or only an input, the other half of the model (e.g., the input if you are modeling the output) can be ignored.*

Output Driver Parameters

This section describes the parameters that make up the output-driver half of a .MOD model. See “Editing a .MOD Model” below in this chapter for details on how to change these parameters in the .MOD model editor.

Transistor Type

Transistor type describes the basic technology type of an output-stage transistor (e.g., Schottky-clamped bipolar or CMOS FET). You can customize a transistor’s model further with other parameters like “on” resistance and slew time.

The following table lists the valid transistor types:

Type	**Description**
CMOS	a CMOS FET
silicon	a fully saturating bipolar transistor
Schottky	a Schottky-clamped bipolar transistor
ECL	an ECL emitter-follower; set both the high and low stages to this for an ECL device, even though there is really only one stage (the emitter)
open	nothing, i.e., no transistor at all
ramp	a special construct for simplifying a driver model; a resistance only, which switches between stages infinitely fast

Transistor "On" Resistance

Transistor "on" resistance describes the effective, fully-on impedance of the upper- or lower-stage transistor.

Hint: *This is the slope of the DC output-buffer V-I curve in the databook. It is NOT the resistance implied by the guaranteed worst-case DC currents, i.e., Ioh and Iol; these values are usually unrelated to the driver's dynamic switching characteristics and yield much too large a resistance.*

Slew Time

Slew time specifies the **10%-90%** switching time of the upper- or lower-stage transistor.

Offset Voltage

Offset voltage describes the effective offset for the upper- or lower-stage transistors from the rail voltage. It models the internal biasing of the driver.

Special Note about ECL

Even though ECL output buffers are actually referenced only to Vcc, the Low Offset Voltage is still interpreted in the .MOD editor as being from the low rail, in this case Vee. So, for example, to make an ECL driver switch to –1.55V when its Vee = -4.5, set the low-side offset voltage to +2.95 V.

Clamp-Diode Type

Clamp-diode type specifies the technology type of the upper- or lower-stage output clamp diode. You can customize a clamp diode's model further with the "on" resistance parameter.

The following table lists the valid diode types:

silicon	a silicon clamp diode
Schottky	a Schottky clamp diode

Clamp-Diode "On" Resistance

Clamp-diode "on" resistance describes any resistance effectively in series with the upper- or lower-stage clamp diode.

Hint: *This is the slope of the clamp-diode DC V-I curve in the databook. Sometimes, this data is found in a special section of the databook that covers ESD issues.*

Capacitance

The capacitance specifies the total output capacitance of the driver, including transistors and clamp diodes.

Default Power Supply

Default power supply specifies the default non-ground supply voltage off of which the driver is run. LineSim's built-in Vcc and Vss power-supply nets are set to the default-power-supply voltages when you load a driver IC model. See Chapter 4, section "Setting the Vcc or Vss Pin" for details.

Measurement Thresholds and Loads

These parameters are used in BoardSim's Board Wizard to calculate delay and related data. They do not affect a driver model's waveform, only how those waveforms are *measured* by the Board Wizard. *They are unused in LineSim, but described here for completeness.* The following table gives more detail.

Parameter	**Explanation**
Vmeasure	Voltage at which the driver is considered "switched," i.e., when it transitions past this voltage, it has switched
Rload	The pull-up/down resistor used in the IC manufacturer's standard driver-output test load
Vload	The pull-up/down voltage used in the IC manufacturer's standard driver-output test load
Cload	The capacitance used in the IC manufacturer's standard driver-output test load

A model's Vmeasure value is displayed in both the Select IC Model dialog box (in the I/O Type area) and the Assign IC Models dialog box (in the Buffer Settings) area.

Note: *Most devices use standard values for these parameters (Vmeasure=1.5V, Rload=1000ohms, Vload=0V, Cload=50pF). They do not normally need to be changed from these values unless you are modeling ECL or a newer, low-voltage driver family like LVDS, GTL, etc.*
Vmeasure can also sometimes be calculated as:
(high input threshold + low input threshold) / 2.

Input Receiver Parameters

This section describes the parameters that make up the input-receiver half of a .MOD model. See "Editing a .MOD Model" below in this chapter for details on how to change these parameters in the .MOD model editor.

Input Resistance

Input resistance describes the effective resistance of the receiver's biased input stage.

Hint: *Generally, you can neglect input resistance for signal-integrity simulation, since it is normally a large value. The combination of input resistance and offset voltage should result in the input current specified in the databook.*
For CMOS, the input resistance is typically 1 Mohm or more; for signal-integrity simulation, 1 Mohm is sufficient.

Offset Voltage

Offset voltage describes the equivalent voltage of the receiver's input-stage biasing.

Hint: *The offset voltage is the open-circuit voltage of the input pin. For CMOS, this is typically Vcc/2.*

Clamp-Diode Type

Clamp-diode type specifies the technology type of the upper- or lower-stage input clamp diode. You can customize a clamp diode's model further with the "on" resistance parameter.

The following table lists the valid diode types:

silicon	a silicon clamp diode
Schottky	a Schottky clamp diode

Clamp-Diode "On" Resistance

Clamp-diode "on" resistance describes any resistance effectively in series with the upper- or lower-stage clamp diode.

Hint: *This is the slope of the clamp-diode DC V-I curve in the databook. Sometimes, this data is found in a special section of the databook that covers ESD issues.*

Capacitance

The capacitance specifies the total input capacitance of the driver, including transistors and clamp diodes.

Measurement Thresholds and Loads

These parameters are used by BoardSim's Board Wizard to calculate delay and related data. They affect only how receiver-input signals are *measured* by the Board Wizard. *They are unused in LineSim, but described here for completeness.* The following table gives more detail.

Parameter	**Explanation**
Vih or Vih+	Receiver's primary high-going threshold; receiver is guaranteed to have recognized as a '1' any rising-edge signal that crosses this value
Vih-	Receiver's secondary high-going threshold; for devices with hysteresis, the high-going threshold backs down to this value after Vih+ is crossed; if no hysteresis, disable Schmitt Trigger check box, or set this value = Vih+
Vil+	Receiver's secondary low-going threshold; for devices with hysteresis, the low-going threshold backs up to this value after Vil- is crossed; if no hysteresis, disable Schmitt Trigger check box, or set this value = Vil-
Vil or Vil-	Receiver's primary low-going threshold; receiver is guaranteed to have recognized as a '0' any falling-edge signal that crosses this value

As indicated in the table, only two of the threshold values (Vih+ and Vil-) are needed for most devices; the other two (Vih- and Vil+) are important only for receiver inputs that exhibit hysteresis. If a device has no hysteresis, you can hide the "secondary" thresholds from view, or set them equal to the "primary" values.

To hide the "secondary" thresholds for devices that do not have hysteresis (i.e., whose inputs are not "Schmitts"):

1. In the Edit .MOD Model dialog box, in the Measurement Thresholds and Loads area, click on the Schmitt Trigger check box to disable it.

A model's input-threshold values are displayed in both the Select IC Model dialog box (in the I/O Type area) and the Assign IC Models dialog box (in the Buffer Settings) area.

Note: *Most devices use standard values for these parameters (Vih+ = Vih- = 2.0V, Vil+ = Vil- = 0.8V). They do not normally need to be changed from these values unless you are modeling ECL or a newer, low-voltage driver family like LVDS, GTL, etc.*

Editing a .MOD Model

To edit a .MOD model:

1. From the Edit menu, choose Databook IC Models.

The .MOD model editor opens.

Choosing a Model to Edit

The first step in editing a .MOD model is to choose the library and model.

To choose the model to be edited:

1. In the Library and Model area, pull down the Model Library combo box, and choose the library that the model you want to edit is in.
2. Pull down the Device Model combo box, and choose the model.

The libraries displayed in the combo box are the .MOD files in the directory pointed to by the Model Library File Path directory setting. Only libraries in this directory can be accessed. See "Setting the Models Directory" below in this chapter for details on setting the directory.

Choosing Driver or Receiver

Once the model is chosen, you must choose whether to edit its output-driver or input-receiver half.

To choose the output or input half of the model:

1. In the Library and Model area, click the Output or Input radio button.

If you change the radio-button setting, the contents of the .MOD-model-editor dialog box change to show different parameters.

Editing the Model

To edit the model:

1. Change the transistor types, diode types, and default power-supply voltage, if needed, by pulling down the combo boxes and choosing new values.
2. Change other parameters by typing new values into the edit boxes.

Editing both Driver and Receiver

You can edit both halves of a model (output and input) by editing one half; clicking the opposite radio button in the Library and Model area; then editing the other half.

Creating a New Model

To create a new model, start with an existing one. If you can find a model in GENERIC.MOD (or one of the other libraries that ships with LineSim) that is similar to the model you want to create, choose that model and then edit it.

If you cannot think of a similar model in the supplied libraries, choose any model as a starting point and completely change its parameters, if necessary.

Once you have created the new model, save it; see “Saving a .MOD Model” below in this chapter for details.

Hint: *GENERIC.MOD or EASY.MOD almost always contain a model which is a good starting point for another, new model. Choose the closest match to the IC you’re modeling, and change some parameters.*

As examples of starting with GENERIC.MOD, if the IC is CMOS and has an output slew time faster than 2 ns, start with a 74AC line driver. If it's CMOS with a slower slew rate, start with 74HC. If it's a bipolar IC, start with a 74AS line driver. If it's an ECL IC, start with 100K ECL.

Setting the Default Vcc or Vee Voltage

LineSim's built-in Vcc and Vss power-supply nets are set to the default-power-supply voltages when you load a driver IC model. See Chapter 4, section "Setting the Vcc or Vss Pin" for details.

LineSim gives you a number of choices for default Vcc/Vee voltage of a .MOD model. (You must use one of the pre-supplied choices; you cannot type your own value.) The second voltage in the pair is always 0.0V. For example, if you choose 3.0V, VCC=3.0V and VSS=0.0V; if you choose -5.2V, VCC=0.0V and VEE=-5.2V.

5.0V/0.0V are the default values for all non-ECL models; 0.0V/-4.5V and 0.0V/-5.2V are the defaults for the appropriate ECL models.

Saving a .MOD Model

After you have edited a model, you must save it into a library.

If you are changing an existing model, save it back into the same library and under the same model name as when you chose it. If you are creating a new model, save it into a different library and/or under a different model name.

Saving to the Same Library and Model Name

To save a model to the same library and model name:

1. In the .MOD-model-editor dialog box, click the Save button.

After a brief pause, the library file is updated.

Saving to a Different Library or Model Name

To save a model to a different library and/or model name:

1. In the .MOD-model-editor dialog box, click the Save As button. The Save .MOD Model As dialog box opens.
2. *If you are saving to a different library,* in the Library Name list box, choose or type the library.
3. *If you are saving to a different model name,* in the Model Name box, type the model name.
4. Click OK.

After a brief pause, the library file is updated.

Cannot Save into GENERIC.MOD or EASY.MOD

You can edit a model in the LineSim-supplied libraries GENERIC.MOD and EASY.MOD, but you cannot save the edited model back into either library; you must save to a different library. The model editor will not write into GENERIC.MOD or EASY.MOD.

Note: *It often makes sense to edit a model in GENERIC.MOD or EASY.MOD as a starting point for creating a new model. Once you have loaded and modified the model, save it into an existing library of your own, or into a completely new library.*

Note: *If you modify a HyperLynx-supplied library, you should rename it first. If you do not change the library's name, your version of the library will be overwritten next time you receive updated HyperLynx software.*

Creating a New Library

To create a new library:

1. Follow the steps in “Saving to a Different Library or Model Name,” except, for the library name, type a completely new name.

When you type the library's name, the extension .MOD is optional; if you omit it, it is automatically provided.

Deleting a .MOD Model

You can delete models out of .MOD libraries. Before deleting a model, be certain that you never want to use it again: it will be removed from the .MOD file and lost.

Note: *If a deleted model is recorded in a .TLN file, the pin that calls out the deleted model will not have a model when the schematic is loaded. LineSim will issue a warning, and the missing model's pin will be marked as "????". You must then choose a different model for the pin.*

To delete a .MOD model:

1. From the Edit menu, choose Databook IC Models.
2. In the .MOD-model-editor dialog box, pull down the Model Library combo box and choose the library that the model you want to delete is in.
3. Pull down the Device Model combo box, and choose the model.
4. Click the Delete button.
5. LineSim asks if you are sure you want to delete the model; click OK if so.

Cannot Delete from GENERIC.MOD or EASY.MOD

You cannot delete a model in the LineSim-supplied libraries GENERIC.MOD or EASY.MOD. If you try to, LineSim gives an error message.

Note: *You can delete models from other .MOD libraries (libraries other than GENERIC.MOD and EASY.MOD) supplied with LineSim. If you accidentally delete a model from a HyperLynx-supplied library, you can download the library from the HyperLynx Internet site — or re-install LineSim. See "Updating Models over the Internet" below in this chapter for details.*

Editing IBIS IC Models

IC models created in the IBIS format can be edited directly inside LineSim, using the Visual IBIS Editor. If you are unfamiliar with the IBIS format or how it compares to the .MOD modeling format (another type of IC model supported by LineSim), see Chapter 4, section “IC-Model Formats.” (A third supported format, .PML, is an extension of .MOD that adds component pin-out and package characteristics.)

For details on using the Visual IBIS Editor, see “The Visual IBIS Editor” below. A shareware copy of the editor can be downloaded free of charge from HyperLynx’s World Wide Web site; see Chapter 12 for details on the Web site.

Note: *Because most IBIS models come directly from the semiconductor manufacturer with guaranteed data, they are not typically edited by users. You may want to create your own IBIS libraries, however, to model custom devices, ASICs, etc. But before creating an IBIS model, consider whether the .MOD modeling format would meet your needs: it is simpler and easier to create models with. See Chapter 4, section “IC-Model Formats” for a comparison of the .MOD and IBIS formats.*

The Visual IBIS Editor

The Visual IBIS Editor is a program — available separately but also shipped with LineSim and launchable from it — for creating, editing, verifying, and maintaining IBIS (I/O Buffer Information Specification) device models. The Editor includes a number of useful features for developers of IBIS models, including:

- a large-file Windows text editor
- an integrated IBIS-syntax-check utility
- a graphical viewer for looking at IBIS V-I tables and waveform tables
- an IBIS-file template generator

- a test generator that automatically creates a test schematic for any IBIS-model pin
- a complete, self-contained online Help system
- an online IBIS specification

HyperLynx developed the Editor to encourage the development of IBIS device models. The Editor's "smart" features make creating and verifying IBIS models much simpler than with IBIS-ignorant, standalone tools.

Opening the Editor

In LineSim, you run the Editor by launching it from LineSim's menu bar, then operating it from its own, standalone user interface.

To open the Visual IBIS Editor:

1. From the Edit menu, choose IBIS IC Models.

The Editor opens in its own window, ready to load or create an IBIS file.

In all of the sections below, the menus and toolbar buttons referred to are Visual IBIS Editor menus and buttons, not LineSim's.

Editing an IBIS File

To open an existing IBIS file in the Editor:

1. From the File menu, choose Open.
   OR
   Click the Open button on the toolbar.
2. Choose the file you want to open.
3. Click OK.

The IBIS file appears in the Editor.

To open a file in read-only mode:

1. After opening the file, choose Read Only from the Options menu.

In read-only mode, any changes you attempt to make to the file in the Editor will be ignored.

There are two windows in the Editor:

- a main, upper window for viewing and editing IBIS files
- a secondary, lower window for seeing the results of running the IBIS syntax checker

The Editor works like most other Windows text editors. It includes support for standard Windows features like cut, copy, paste, and so forth; see the following sections for details.

The Editor handles files of arbitrary size. There are no artificial limits (like 32K or 64K) as with some Windows text editors. The Editor is ASCII only; it will not insert binary characters into a file.

Cutting, Copying, Pasting, and Deleting Text

The Visual IBIS Editor supports cutting, copying, pasting, and deleting of text from the menus, from the toolbar, or using the standard Windows accelerator keys.

To cut text:

1. Highlight the text you want to cut.
2. From the Edit menu, choose Cut.
   OR
   Type Ctrl-X.
   OR
   Click the Cut button on the toolbar.

The text disappears.

To copy text:

1. Highlight the text you want copy.

2. From the Edit menu, choose Copy.
   OR
   Type Ctrl-C.
   OR
   Click the Copy button on the toolbar.

The text is stored on the Clipboard.

To paste text:

1. Position the cursor where you want the text to be pasted.

2. From the Edit menu, choose Paste.
   OR
   Type Ctrl-V.
   OR
   Click the Paste button on the toolbar.

The pasted text is inserted at the cursor location.

To delete text:

1. Highlight the text you want to delete.

2. Choose Delete from the Edit menu, or press the delete key.
   The text disappears.

You can also delete text by clicking in the text to position the cursor, then typing with the Backspace key or Delete key.

Undoing an Action

The Visual IBIS Editor provides a single-level “undo” feature.

To undo the previous editing action:

1. From the Edit menu, choose Undo.
   OR
   Type Ctrl-Z.

The last editing action you performed is undone.

Converting Tabs to Spaces

The Visual IBIS Editor has a feature which automatically converts tab characters to space characters. The IBIS specification allows tabs, but recommends against using them because different tools expand them in different ways.

To convert tab characters to space characters:

1. From the Edit menu, choose Convert Tabs to Spaces.

All of the tabs in the file are replaced with spaces.

Going to a Line Number

To go to a line number:

1. From the Search menu, choose Go To Line.
   OR
   Click the Go To button on the toolbar.
   A dialog box opens.

2. Type the line number to which you want to go.

3. Click OK.

The editor jumps to the specified line number, with the matching line appearing at the *top* of the window.

Finding Text

To find text:

1. From the Search menu, choose Find.
   OR
   Click the Find button on the toolbar.
   A dialog box opens.

2. Type text you want to find.

3. Click OK.

The editor jumps to the first occurrence of the specified text (or gives an error that no matching text could be found). The line with the matching text appears at the *top* of the window.

Searching always occurs from the top of the file, regardless of where the cursor is positioned when the search is requested.

To find the next occurrence of the same text:

1. From the Search menu, choose Find Next.
   OR
   Click the Next button on the toolbar.

The editor jumps to the next occurrence of the text.

Viewing an IBIS V-I or Waveform Table

One of the most-valuable features in the Visual IBIS Editor is the ability to graphically view V-I or waveform table data. Viewing table data graphically makes it much easier to find errors in the data, e.g., a mistyped number or bad sign.

Viewing a table is a two-step process:

♦ first, you choose a signal or pin in the IBIS file whose model’s tables you want to view

♦ then, you choose exactly which table you want to see, and you view it

To choose a signal or pin:

1. From the IBIS menu, choose Select Signal or Pin.
   OR
   Click the Select button on the toolbar.
   The Select Model dialog box opens.

2. In the Devices list box, click once to highlight the device that the signal or pin is on.

3. In the Signal list box, highlight the signal you want. If you prefer to choose by pin name, first click the Pin radio button in the Select By area, then highlight the pin.

4. Click OK.

To choose and view a table:

1. From the IBIS menu, choose View Data for Selected Pin.
   OR
   Click the Views button on the toolbar.
   The viewing dialog box opens.

2. Click on the tab for the table you want to view.

The graphical display shows the table's minimum, typical, and maximum curves, if available, each in a different color. The display scales itself automatically to best fit the table's data.

Viewing Rising/Falling Waveform Tables

When you click the Rising Waveform or Falling Waveform tab, to view a model's V-t table(s), an extra combo box labeled "Conditions" appears. This allows you to choose amongst multiple waveform tables for viewing, if there is more than one table in the model.

Validating an IBIS File's Syntax

Another valuable feature in the Visual IBIS Editor is the ability to check an IBIS file's syntax without ever leaving the Editor. This is accomplished by running the official EIA-656 (IBIS) validation-checking program on the file that is currently being edited. You can run the check periodically as you create or edit a model to ensure that you haven't introduced errors.

To run the syntax validation check:

1. From the IBIS menu, choose Run IBIS Validation Check.
   OR
   Click the "IBIS" button on the toolbar (looks like a check mark).
   The validation-checking program is automatically launched.

The lower window in the Editor displays messages from the validation program. If a large number of errors appears, you can move through them with the scroll bar on the right edge of the window.

The validation checker automatically determines (from the [IBIS Ver] record in the file) whether to run a V1.x or V2.x, or V3.x syntax check.

If the file is error-free, the number of warnings and errors is reported as "0."

Creating a New IBIS Model

The Visual IBIS Editor gives you two kinds of assistance with creation of a new IBIS model. One is a template generator that creates a skeleton — complete with all of the required keywords — for an IBIS V1.1 or V2.1 model. The second is a more-powerful feature called the "Easy IBIS File Creation Wizard" that actually "interviews" you about the model's data, then automatically generates the corresponding model file for you.

For details about the template generator, see "Creating a Template for a New IBIS File" below. For details about the Easy IBIS Wizard, see "Running the Easy IBIS File Creation Wizard."

Creating a Template for a New IBIS File

The IBIS template generator automatically creates a template file that contains a "skeleton" IBIS model, in either V1.1 or V2.1 IBIS format. The template gives you a head start on creating model, because it reminds you of required IBIS keywords, syntax, and so forth. The data in tables, etc. is "dummy" and should be replaced manually with the correct information.

To run the IBIS File Creation Wizard:

1. From the IBIS menu, choose Run IBIS Template Creation Wizard. A dialog box opens.

2. Use the radio buttons to specify the IBIS file version you want (V1.1 or V2.1).

3. Click the Next button, then Finish.

4. In the dialog box, type the name of IBIS file for which you want to create a template. Remember that IBIS files names are limited to DOS "8.3" format.

The template generator creates the template file and opens it in the Editor.

Running the Easy IBIS File Creation Wizard

The Easy IBIS Wizard is a powerful model-generation utility that interviews you about the characteristics of the component you want to model, then, based on the information you input, automatically generates a complete IBIS model, syntactically correct and ready-to-simulate.

To begin running the Easy IBIS Wizard:

1. With the Visual IBIS Editor open, from the IBIS menu, choose Run Easy IBIS File Creation Wizard.
   OR
   Click the "Easy IBIS" button on the toolbar.

The Easy IBIS Wizard opens, ready for you begin specifying the IBIS file you want created.

Entering Data in the Easy IBIS Wizard

About Data Entry

Entering Text Strings

The IBIS syntax specifies definite limits to the length of various user-created text strings. In the Easy IBIS Wizard, these length limits are enforced; if you try to type in a string that is too long, the Wizard will refuse to enter the extra characters and will "beep." This cues you to shorten the text string.

Entering Numerical Data

The IBIS syntax allows most numerical values without limits, but IBIS simulators will sometimes have trouble with absurdly large or small values. In the Easy IBIS Wizard, if you enter any such values on a page and click "Next," the Wizard will warn you that one or more values are unreasonable, and force you to re-enter them. This prevents you from generating a model that may not simulate properly in some simulators.

Entering the IC Component Name

The first page of the Easy IBIS Wizard prompts you for a component name for the model you're about to create. Every IBIS file can contain multiple buffer models that are tied together through a pin out to make up a "component"; a component models a complete IC. On its first page, the Wizard asks you for the name of the component, i.e., the IC name. This should generally be a fairly specific name; typically it might include manufacturer, device, and package information.

Entering Header Information

To advance to the Header Info page in the Wizard:

1. From the Wizard's first page, click Next.

At the top of an IBIS file, model developers are encouraged to include a file-creation date, file-revision number, and copyright. These help both the developer and users of a model track multiple revisions of the file. The copyright notice protects the model legally. The data in these fields is free form, although the revision number is normally a number.

Entering the Data Source

To advance to the Data Source page in the Wizard:

1. From the Wizard's Header Info page, click Next.

Near the top of an IBIS file, model developers are encouraged to specify — in some detail, if needed — the source of the model's data. In particular, users are interested in knowing whether the model was developed from measured or simulated data, and any other information pertinent to how the model was created. The data in this field is free-form, and is often multi-line. Use as much detail as is needed.

The Wizard gives a suggested starting phraseology, but its use is not required.

When the Easy IBIS Wizard writes out the data-source section, it will wrap it as needed to prevent from exceeding the 80-character line-length restriction imposed by the IBIS specification. This means that the text may not appear exactly as you enter it on this page of the Wizard.

Entering Notes

To advance to the User Notes page in the Wizard:

1. From the Wizard's Data Source page, click Next.

The [NOTES] section of an IBIS file can be used to add for the user any amount of clarifying detail about the model that is not already captured in the preceding fields. Typical information included here might be caveats about data missing from the model; descriptions of when the model is most accurate, and when it is not; and so forth. The data in this field is free-form, and is often multi-line. Use as much detail as is needed.

When the Easy IBIS Wizard creates the notes section, it will wrap text as needed to prevent it from exceeding the 80-character line-length restriction imposed by the IBIS specification. This means that the text may not appear exactly as you enter it on this page of the Wizard.

Entering a Disclaimer

To advance to the Disclaimer page in the Wizard:

1. From the Wizard's User Notes page, click Next.

The [DISCLAIMER] section of an IBIS file is primarily for use by semiconductor manufacturers. It is typically used to disclaim legal responsibility for the model's accuracy, suitability, and so forth. The data in this field is free-form, and is often multi-line. Use as much detail as is needed.

The Wizard suggests typical disclaimer phraseology, but its use is not required.

When the Easy IBIS Wizard writes out the disclaimer section, it will wrap it as needed to prevent from exceeding the 80-character line-length restriction imposed by the IBIS specification. This means that the text may not appear exactly as you enter it on this page of the Wizard.

Entering Additional Header Information

To advance to the Additional Header Info page in the Wizard:

1. From the Wizard's Disclaimer page, click Next.

The additional header information allows you to enter the name of the IC's manufacturer, and information about whom to contact regarding the model. The manufacturer name is required by the IBIS specification; the remaining fields are optional (although encouraged, since they give the user of the model someone to query if technical questions arise). Any reasonable entry can be made in the manufacturer field; if you are creating a model but do not work for the company that actually manufactures the silicon, enter your own company name.

When the IBIS file is generated, the contact information, if present, is preceded by the phrase "If you have comments concerning this file, please address them to:".

Entering Pin Count and Package Parasitics

To advance to the Part Pin Count page in the Wizard:

1. From the Wizard's Additional Header Info page, click Next.

Every IBIS file must contain a pin out table that lists the pins on the IC component being modeled, and connects each pin to a buffer model in the file (or specifies that a pin is a power pin or not connected). Every file must also contain a table defining the default package parasitics — Rpkg, Lpkg, and Cpkg — to be assumed for pins for which no pin-specific parasitic data is later specified. The pin out table must contain at least one pin (although in a complete model, it would contain an entry for every pin on the IC). The package parasitic data can be all 0.0, although doing so will omit package effects from consideration when the model is simulated.

Predefined versus User-Defined Packages

This page of the Easy IBIS Wizard allows you to specify either your own user-defined package, or start with a predefined one. You specify the number of pins on the package, or use a predefined package, whose definition includes not only a fixed number of pins, but also parasitic R/L/C data for the package.

The parasitics included in a predefined package are typical values for a package of that style (e.g., larger values for DIPs, smaller for SMD packages, etc.). The advantage to using a predefined package is that it comes "for free" with a complete list of pins and parasitics, meaning you have less data to enter than with a user-defined package. However, there may not be a predefined package that matches the IC you're trying to model, so you may need to create your own.

Using a Predefined Package

To select a predefined package:

1. Click the Predefined radio button. The list of predefined packages becomes available.

2. Scroll through the list of predefined packages and highlight the one you want to use.

Creating a User-Defined Package

To create your own user-defined package:

1. Click the User-Defined Package radio button. Several data boxes in the lower part of the Wizard page become available.

2. In the Number of Pins box, type the number of pins on the package you're defining.

3. In the Default Pin Parasitics boxes, enter the default values of R, L, and C to be used for modeling the bond wires and pins of the package. These values will be used during simulation for any pins that do not have pin-

specific R/L/C values (specified later in the Wizard). The values can be 0.0 if you do not wish to simulate package parasitics (provides a less-accurate simulation, but is acceptable).

Entering Pin Data

To advance to the Pin Data page in the Wizard:

1. From the Wizard’s Part Pin Count page, click Next.

The table in the IBIS file that lists the IC component's pins also includes, for each pin, an associated signal name and , optionally, a set of pin-specific package parasitics (Rpkg, Lpkg, and Cpkg). The package data is optional on a per-pin basis. If no pin-specific parasitics are supplied, then all pins will be simulated using the default package parasitics specified on an earlier Wizard page (see “Entering Pin Count and Package Parasitics” above). If pin-specific parasitics are supplied for some pins and not others, then the pins with parasitics will be simulated using the pin-specific values, and the pins without will use the default values.

Adding Signal Names

Every pin in an IBIS file is required to have an associated signal name. The Easy IBIS Wizard creates default signal names for every pin, but you should modify the names to match the signals on the IC you're modeling.

To add signal names to pins (by modifying the default name):

1. In the Pin list box, click once to highlight the pin whose signal name you want to modify.

2. In the Signal Name box, type the new pin name.

3. Repeat as needed for other pins.

Modifying Pin Names

If the IC you're modeling has a package with alphanumeric pin names (or if the default pin numbers are wrong in some way), you can modify the default names as needed.

To modify the default pin names:

1. In the Pin list box, click once to highlight the pin whose pin name you want to modify.
2. In the Pin name list box, type the new name of the pin. The pin's name changes in the Pin list box.
3. Repeat as needed for other pins.

Specifying Pin-Specific Package Parasitics

For any or all pins in the Pin list box, you can specify pin-specific package characteristics (R/L/C). If you are satisfied with the default package values entered on an earlier Wizard page, there is no need to supply pin-specific values. However, especially for some packages that have significantly different R/L/C values depending on pin position, you may want pin-specific data to increase simulation accuracy.

To specify pin-specific package parasitics for a pin:

1. In the Pin list box, click once to highlight the pin for which you want to specify parasitics.
2. In the Pin Parasitic Values area, click on the Customize check box. The R, L, and C data boxes become available.
3. Type the values of R, L, and C you want for the pin.
4. Repeat the steps above as needed for other pins.

You can mix pins that have and do not have pin-specific parasitics. As you highlight various pins in the Pin list box, the ones that have pin-specific values are evident because the Customize check box below enables and the R, L, and C data boxes become active.

Entering Min/Max Scaling

To advance to the Min and Max Scale Factors page in the Wizard:

1. From the Wizard’s Pin Data page, click Next.

Throughout an IBIS file, data can be supplied either with a typical value only, or with typical, minimum, and maximum values. In the Easy IBIS Wizard, to keep entry of a model simple and yet still provide for minimum and maximum data, the values you enter are used as "typical," and then two scaling factors are used to generate min/max data automatically from the typical values. This is a reasonable approach, since ICs are subject to process variations that cause various circuit parameters to scale up and down.

Using this approach, if you have more exact min/max data and want to include it in your model, you can generate the IBIS file, then manually edit it to replace the min/max values created by the Wizard.

Most minimum and maximum values written by the Wizard are created with the scaling factors. An exception is the power-supply min/max values, which are automatically set to +/-10% of the typical value you enter.

To specify the minimum and maximum data scaling factors:

1. In the Min Scale Factor box, type the value to be used to create minimum data.

2. In the Max Scale Factor box, type the value to be used to create maximum data.

Creating Buffer Models

To advance to the Buffer Models page in the Wizard:

1. From the Wizard’s Min and Max Scale Factors page, click Next.

The "core" element in an IBIS file is one or more buffer models. It is these models that actually produce the analog waveforms that signal-integrity simulators use to analyze transmission-line effects and other high-speed phenomena. The pin out table in an IBIS file connects pins/signals with underlying buffer models. In the Easy IBIS Wizard, once you have defined a pin out, you must next create models for the various buffer types present on your IC.

Almost any real IC has at least two buffer types, a basic output or I/O buffer, and an input buffer. Large, complex devices may have many different buffer types, for example, a very strong output buffer on clock-signal outputs, a medium-strength buffer used for address and data lines, and a relatively weak buffer used for non-critical signals. There might also be multiple input-buffer types, with different input capacitances or clamp-diode strengths, for example.

How the Wizard Models Buffers

In the IBIS specification, buffers are described with V-I tables. In the Easy IBIS Wizard, to simplify the data-entry process, these tables are generated from single-valued impedances plus saturation currents. The Wizard uses this data along with internal knowledge about transistor V-I curves for various technologies (like CMOS or TTL) to generate detailed V-I curves from the simplified impedance and saturation data.

In addition, output buffers have slew-rate data, which in the IBIS specification can either be input as a simple slew rate into a given resistive load, or in a more-complex V-t table. The Wizard uses the slew-rate/load method.

Pre-defined versus User-Created Buffer Models

The Easy IBIS Wizard allows you to either create your own custom buffers, or if you're in a hurry or don't know the exact characteristics of the buffer you're trying to create, base your model on a generic, technology-based buffer. Creating a custom buffer requires you to know the approximate driving impedance of the buffer, slew rate, and so forth. Basing a model on a pre-defined buffer requires only that you know the technology type of the model (CMOS or TTL?) and the approximate slew rate (fast, medium, slow?). When you base a model on a pre-defined model, you can either use the pre-defined model "as is," or use it as a starting point but modify some of its characteristics.

Creating a New Buffer Model

To create a new buffer model:

1. Click the New Buffer button.
2. Follow the steps in the section below, "Specifying a New Buffer's Characteristics."

Creating a Buffer Model Based on a Pre-defined Model

To base a buffer model on a pre-defined model:

1. In the list box that shows the currently defined buffer models, click once to highlight the model you want to use as the base for your new model.
2. Click the Copy Buffer button. A copy of the model is created, at the bottom of the list box.
3. With the copied model still highlighted, click the Edit Buffer button.
4. Follow the steps in the section below, "Specifying a New Buffer's Characteristics."

Specifying a New Buffer's Characteristics

Setting Buffer Name, Technology, Type, and Operating Voltage

To advance to the Buffer Model – Operating Voltage page in the Wizard:

1. From the Wizard’s Buffer Models page, click new Buffer or Edit Buffer (see “Creating Buffer Models” above for details).

To specify the buffer name, technology, type, Vcc voltage, and capacitance:

1. In the Buffer Name box, type a name for your new buffer.
2. In the Technology area, click on the radio button for either CMOS or TTL (i.e., bipolar), depending on the technology type of the IC you're modeling.
3. In the Buffer Type area, click on the radio button for the buffer type — input/ output/3-state — that you're modeling. "Open sink" is equivalent to "open drain" or "open collector."
4. In the Operating Voltage edit box, type the typical Vcc value for the IC.
5. In the Die Capacitance box, type the typical die capacitance for this buffer. "Die capacitance" means I/O capacitance (almost always given in the IC data sheet) minus the package capacitance. (2-8 pF is typical.)

Specifying Clamp Diodes

To advance to the Buffer Model – Clamp Diodes page in the Wizard:

1. From the Wizard's Operating Voltage page, click Next.

To specify the data for clamp diodes:

1. In the High Rail Clamp Diode area, pull down the combo box and choose a type for the high-side clamp diode:

   If the buffer has no high-side clamp diode, choose "None".

   If there is a clamp diode, choose between silicon and Schottky diodes (check the IC data sheet for which is correct), and choose an approximate "clamping strength" (strong, typical, weak).

   If you know little about the diodes, choose type "Silicon Typical."

   If you know in detail about the diode's characteristics, choose type "User Defined" (silicon or Schottky, as appropriate), then go to step 2.

   If you did not choose "User Defined", skip to step 3.
2. *If you chose a "User Defined" diode type in step 1*, in the On Impedance edit box, type the value of the clamp diode's effective on resistance.
3. Repeat step 1, but for the Low Rail Clamp Diode.

Specifying Pull-Up and Pull-Down Buffers

Important: *Pull-up data applies only if buffer is type Output, I/O, or 3-State; does not apply if Input or Open Sink.*

To advance to the Buffer Model – Output High Parameters page in the Wizard:

1. From the Wizard's Clamp Diodes page, click Next.

To specify the high-side driver-transistor parameters:

1. In the Rload data box, type the value of the resistance into which the driver's rising-edge switching characteristics are specified (check the data sheet). If you do not know the value, type "1000".
2. In the Slew Time data box, type the amount of time it takes the driver to slew from 20% to 80% of the final DC values (in ns, rising edge).
3. In the Slew Voltage data box, type the voltage difference between the 20% and 80% final DC values (in ns, rising edge).
4. In the Open Circuit Voltage box, type the voltage at which the driver "sits" when it is unloaded (no external load, when switched high). For standard CMOS outputs, this generally equals Vcc; for bipolar, NMOS, and "specialty" outputs, it is different than Vcc.
5. In the Saturation Current box, type the approximate maximum or saturation current of the transistor stage (in Amps). Every output design differs, but all technologies limit at some reasonable value.
6. In the Output Impedance box, type the approximate driving impedance of the buffer. If you do not know this value, read section "Determining Output Impedance" below.

To advance to the Buffer Model – Output Low Parameters page in the Wizard:

1. From the Wizard’s Output High Parameters page, click Next.

Important: *Pull-down data does not apply if buffer type is Input or Open Source.*

To specify the low-side driver-transistor parameters:

1. Repeat steps 1-6 in the preceding section, except enter the data for the low-side transistor stage.

Specifying Input Thresholds

To advance to the Buffer Model – Input Levels page in the Wizard:

1. From the Wizard’s Output Low Parameters page, click Next.

Important: *Applies only if buffer is type Input or I/O; does not apply if Output, 3-State, or Open Sink.*

To specify the input thresholds:

1. In the Logic High Threshold data box, type the value of the worst-case high-going threshold. For most devices, this is 2.0V; it may be higher for certain older CMOS families, like HC. (It may also be different for newer, very-low-Vcc devices.)

2. In the Logic Low Threshold data box, type the value of the worst-case low-going threshold. For most devices, this is 0.8V; it may be different for certain older CMOS families, like HC. (It may also be different for newer, very-low-Vcc devices.)

Specifying Output Polarity and Load Circuit

To advance to the Buffer Model – Input Levels page in the Wizard:

1. From the Wizard's Input Levels page, click Next.

Important: *Applies only if buffer is type Output, I/O, 3-State; does not apply if Input.*

To specify output polarity and manufacturer load circuit:

1. In the Output Polarity area, click on the radio button for the buffer's output polarity (true or inverted). For most devices, this value is "true."

2. In the Vmeasure box, type the voltage at which, for timing measurements (like propagation delay), the manufacturer of the IC considers the output buffer "switched." For most devices, this value is 1.5V. (It may also be different for newer, very-low-Vcc devices.)

3. In the Rref, Vref, and Cref boxes, type the circuit values that describe the manufacturer's standard test loads for timing measurements. This is almost always specified in the data sheet; if not, use the default values of 1000 ohms, 0V, and 50 pF.

Completing a Buffer Model

When you have finished entering all the parameters described in the preceding sections, then finish specifying the buffer model:

1. Click the Finish button. The buffer model is now complete, and ready to be assigned to specific pins on the component IC.

2. Assign the buffer to specific pins on the IC component.

Mapping Buffer Models to Pins

After creating all desired buffer models, then to advance to the Map Buffer Definitions to Pins page in the Wizard:

1. Click Next.

Once you've created the buffer models for all of the pins on the IC component you're modeling (or decided which pre-defined models you can use for various pins), you can map the models to pins. There is no requirement to map every pin to a model, but since pins by default are "no connects," any pins that you do not map will not be available for analysis in an IBIS simulator.

Mapping Buffer Models to IC Pins

To map buffer models to IC pins:

1. In the Buffer Models list box in the upper right corner of the page, click once to highlight the model which you wish to attach to one or more pins. User-created models are listed at the bottom of the list box, below all of the pre-defined models. If you plan to use a pre-defined model for one or more pins, choose it based on the criteria described in section "Choosing a Pre-defined Buffer Model" below.

2. In the list box on left, highlight one or more pins to which you want to attach the highlighted buffer model. You can use standard Windows multiple-selection mouse actions to choose pins (e.g., shift-click to select a group pins, ctrl-click to select isolated multiple pins).

3. Map the highlighted pins to the highlighted buffer model by clicking the right-arrow button. The mapped pins move from the list box on the left to the Pins Attached list box on the right.

4. Repeat steps 1-3 for additional buffer models, as needed.

5. Continue mapping buffers to pins until the entire list of IC pins on the left is exhausted (or until you've mapped every pin you care about).

Removing Mapping on a Pin

To remove a pin that has been mapped (so that it is eligible for a new mapping):

1. In the Pins Attached list box on the right, highlight one or more pins that from which you want to remove mapping(s).

2. Click the left-arrow button.

3. The unmapped pins move from the list box on the right to the list box on the left.

Notice that you can filter the list box on the right in two different ways: to show all of the pins not attached to the currently highlighted model, or to show all of the pins that are not currently attached to ANY model.

Choosing a Pre-defined Buffer Model

The pre-defined buffer models, any of which you are free to use to model pins on ICs for which you're creating IBIS files, are classified by several categories:

- CMOS or TTL?
- 5-V power supply or 3.3-V power supply?
- switching speed: slow, medium, fast, or ultra-fast?
- directionality: I/O (i.e., bi-directional), output-only, or input-only?

If you can categorize the IC buffer you're trying to model by these criteria, then you can choose the proper pre-defined model for the buffer.

The switching speeds in the pre-defined buffer models (fast, slow, etc.) equate to approximately the switching times shown below:

CMOS, 3.3V, ULTRA-FAST = 0.3 ns	CMOS, 5V, MEDIUM = 6 ns
CMOS, 3.3V, FAST = 1 ns	CMOS, 5V, SLOW = 15 ns
CMOS, 3.3V, MEDIUM = 3 ns	TTL, 5V, FAST = 3 ns rising / 2 ns falling
CMOS, 5V, ULTRA-FAST = 0.3 ns	TTL, 5V, MEDIUM = 6 ns rising / 4 ns falling
CMOS, 5V, FAST = 2.5 ns	

Mapping Power and Ground Pins

For power and ground pins, the IBIS specification uses special, reserved keywords. When you map the pins on your IBIS component, map power pins to pre-defined buffer model "POWER", and ground pins to model "GND".

Single-Pin versus Multi-Pin Models

Although you would normally create an IBIS model that has as many pins as the actual IC you're trying to model, there are occasions when you might create a single-pin model. One example would be to test a single custom ASIC buffer that is of interest: you are focused on one buffer type and don't know (or care) about how it will be pinned out in the eventual silicon.

There is a slight behavior difference between single-pin and multi-pin IBIS models. When you map pins to buffer models, if the IBIS component is multi-pin, you are required to map at least one pin to a power-supply buffer. However, if the component is single-pin, then the Wizard assumes you are not modeling a real pin out and relaxes this restriction.

About Unmapped Models

When the Wizard generates the IBIS file, any models that are not mapped to any pins are not written into the resulting IBIS file.

Determining Output Impedance

Driver output impedances are not always specified in the IC datasheet. Lacking a manufacturer's specification for impedance, there are several ways to obtain it, described in the following sections.

Measure from Published V-I Curves

Many manufacturers are now publishing (or have readily available) V-I curves for their output-buffer stages (high and low). If you can obtain such curves, output impedance is easy to calculate.

To measure impedance from a V-I curve:

1. Draw a straight line along the linear portion of the V-I curve, before the current "flattens out" or saturates.

2. Take two points on the line. Measure delta(v) and delta(i) between the points.

3. Then calculate the output impedance as Zout = delta(v) / delta(i)

Extract from a SPICE Model

HyperLynx has application note describing how to extract output-impedance from a SPICE buffer model. Contact HyperLynx or your local reseller to request the application note.

Generating the IBIS File

When you have successfully defined your IC component, created buffer models for all of its pins, and mapped the buffer models to the pins, you are ready to generate the IBIS file representing the IC. The power of the Easy IBIS Editor is that it generates the file automatically for you, with guaranteed-correct syntax.

To generate the IBIS file representing the IC about which you were just interviewed:

1. On the Map Buffer Definitions to Pins page, click the Finish button.

2. The Wizard queries you to name the IBIS file. By default, the name is <component_name>.IBS, where <component_name> is the name you

entered for the IC component on the very first page of the Wizard. However, IBIS file names are restricted to eight characters in length, so if needed, modify the proposed file name so that it is eight characters long or shorter.

3. Click the Save button.

The Wizard automatically generates the IBIS file, then opens it in the Visual IBIS Editor for viewing and editing.

What to Do with the New Model You’ve Just Created

At this point, there are several things you can do with your new IBIS model:

- ♦ you can make modifications to the model, using the text-editing features in the Visual IBIS Editor
- ♦ you can syntax-check the model (although it's extremely unlikely that the Easy IBIS Wizard would generate a syntactically flawed model); for details on running the IBIS syntax checker, see “Validating an IBIS File’s Syntax” above
- ♦ you can view its contents graphically, using the viewing features in the Visual IBIS Editor; see “Viewing an IBIS V-I or Waveform Table” above for details
- ♦ you can test the model, using the LineSim simulator; see "Testing an IBIS File” below for details

Loading Existing Models into the Easy IBIS Wizard

If you have previously created a model with the Easy IBIS Wizard, and want to edit or view it again in the Wizard, you can do so by first re-loading the model's IBIS file into the Visual IBIS Editor.

To re-load into the Wizard a model previously created by the Wizard:

1. In the Visual IBIS Editor, from the File menu, choose Open.
   OR
   Click the Open File button on the toolbar.

Then choose the IBIS file that was generated previously by the Wizard. The Editor opens on the IBIS file.

2. Then, from the Editor's IBIS menu, choose Run Easy IBIS File Creation Wizard.
   OR
   Click the “Easy IBIS” button on the toolbar.

The Wizard opens, and the data previously entered into it is restored for modification or viewing.

The Wizard "remembers" the data that was previously entered and subsequently used to generate the IBIS file, because the Wizard saves a binary file with all of the relevant data. The file is called <IBIS_file_name>.HDS, where <IBIS_file_name> is the name of the IBIS file that the Wizard previously generated, and "HDS" means "HyperLynx Development System."

The HDS file contains data even for buffer models that were not originally assigned to pins, and therefore not written to the original IBIS file.

If you move the IBIS file created by the Wizard to another directory and might want to run the Wizard on it again, be sure to move the .HDS file, also.

Limitations to the Easy IBIS Wizard

The following paragraphs describe some limitations to the Easy IBIS Wizard:

1. There is no direct support for ECL or pseudo-ECL models. If you need to generate such a model, you can start with a standard technology type (like CMOS), then modify the resulting IBIS file as needed to make it work for your ECL buffer. (See the IBIS specification for details on the changes needed.) The CMOS model will be incorrect in some important respects (like table reference voltages and model type), but will at least give you a starting "skeleton" to work with. A second option would be to find an existing ECL IBIS model and modify it (i.e., not use the Easy IBIS Wizard at all).

2. If you own a floating-license (i.e., network-licensed) version of LineSim, and have it open and running when you request that the Easy IBIS Wizard generate and launch a test schematic, the launch will fail with a message

saying "can only run one copy of LineSim." To launch the test schematic successfully, you must first close the open copy of LineSim. This limitation does not exist with a node-locked copy of LineSim, i.e., you can have LineSim running, then launch a test schematic which will successfully open a new copy of LineSim.

Testing an IBIS File

To help you test an IBIS model (whether vendor-supplied or one you just created with the Easy IBIS Wizard — see above for details), the Visual IBIS Editor has a feature which creates a special LineSim schematic, and launches LineSim on that schematic, ready for testing. You must first select, in the Visual IBIS Editor, a particular pin/signal whose buffer model you want tested.

To generate a test schematic for a buffer in an IBIS file, and open it in LineSim:

1. With the IBIS file loaded in the Visual IBIS Editor, from the IBIS menu, choose Select Signal or Pin.
   OR
   Click the Select button on the toolbar.

   The Select Model dialog box opens.

2. Click to highlight the signal whose buffer model you want to test. Or if you prefer to choose by pin name, in the Select By area, first click the Pin radio button, then choose the pin whose model you want.

3. Then, from the IBIS menu, choose Check Model with Simulator.
   OR
   Click the Test button on the toolbar.

LineSim is launched and opened on an automatically generated schematic that invokes the buffer model for the signal/pin you chose. The buffer is tied to a standard test load (50-ohm resistor to ground). You can begin by testing with the schematic "as is" and then modify and enhance it as needed.

About the Schematic Generated for LineSim

The schematic generated for LineSim is placed in the current location specified by LineSim for storing .TLN files (i.e., LineSim schematic files; location is set in the LineSim user interface; from the LineSim Options menu, choose Directories). This storage location contrasts with the IBIS file generated by the Wizard, which is stored in LineSim's default library directory (where all other model files would be residing; again, location is set in the LineSim user interface).

If the Wizard finds an already-existing schematic with the name of the schematic it was about to create, the new schematic will not be created and the old copy will be loaded instead.

More-Advanced Testing

LineSim also ships with a schematic (located in the HYPFILES sub-directory) that has a set of recommended tests for an IBIS model: “IbisTest.tln”. This schematic includes four loads which any IBIS model should switch into with sensible results, provided the model is “good.” If you develop a model and it does not behave acceptably into each of these loads, then the model is probably “bad.”

“IbisTest.tln” assumes that the IBIS model represents a normal push-pull driver that does not need any special external load in order to switch properly. For buffers that do not fit this description, you may need to modify the schematic. (For example, an open-drain driver won't switch unless pulled up by an appropriate resistor. Similarly, ECL drivers need a pull-down to Vtt.) For differential drivers, an alternate schematic (“IbisDiff.tln”) is supplied as a starting point.

Printing a File

To print the file that is open in the Visual IBIS Editor:

1. From the File menu, choose Print.
   OR
   Click the Print button on the toolbar.
   A dialog box opens.

2. Change options, if needed, in the dialog box.

3. Click OK.

Printing occurs to whichever printer Windows is currently connected.

Saving/Closing Files and Exiting

The Visual IBIS Editor allows you to save a file as long as the editor is not running in read-only mode. (For details on read-only mode, see “Editing an IBIS File” above in this chapter.)

To *save* the file that is open in the editor:

1. From the File menu, choose Save.
   OR
   Click the Save button on the toolbar.

To save the file under a new name:

1. From the File menu, choose Save As.
2. Type the new file name, then click Save.

To *close* the file that is open in the editor:

1. From the File menu, choose Close.
   OR
   Click the Close button on the toolbar.

If you attempt to close a file before saving changes that were made to it, the Editor prompts you to save first.

To exit the Editor:

1. From the File menu, choose Exit.
   OR
   Click the Exit button on the toolbar.

Help with the IBIS Standard

A number of resources are available that describe the IBIS standard and how to create models with it:

- in the Visual IBIS Editor, from the Help menu, choose Help IBIS (or click the IBIS button on the toolbar); a Help window opens with the contents of the V2.1 IBIS specification

in this manual:

- Appendix A, "IBIS V2.1 Specification"; the complete IBIS V2.1 specification; V2.1 offers a rich set of modeling features, some intended specifically to help semiconductor vendors in detailed model creation
- Application Note: "Creating IBIS Models"; a guide to creating IBIS models written by HyperLynx and intended for non-semiconductor-vendor users

Setting the Models Directory

LineSim allows you to specify the directory in which your model libraries are stored. LineSim must know this location; otherwise, it cannot load IC models.

By default, LineSim installs its libraries in a subdirectory of the LineSim root directory called LIBS. For simplicity, HyperLynx recommends using the LIBS directory for your IC models. However, if you want to move your models elsewhere, you can change the models directory to point to the new location.

Note that all of your IC libraries must be in the directory you designate as the models path. Libraries not in this directory are "invisible" to LineSim.

To set the model-libraries directory:

1. From the Options menu, choose Directories.
2. In the Model Library File Path edit box, type the desired directory (with a trailing '\').
   OR
   Click the Browse button; a dialog box opens. Find the directory where your IC models are stored and double-click on a model file to set the path. (You can also highlight a file and click Open.)
3. Click OK.

The change takes effect immediately.

LineSim looks in this same directory to find connector models (.SLM files). See Chapter 6, section “Modeling a Transmission Line as a Connector” for details on connector modeling.

To restore the model-libraries directory to its default value:

1. From the Options menu, choose Directories.
2. In the Model Library File Path area, click Default. The path is reset to its default value.
3. Click OK.

LineSim Hint: How to Create a Custom IC Model

Sooner or later, you will need an IC model which neither HyperLynx nor the silicon vendor can immediately supply you. This might be for an ASIC, an obsolete IC, a brand-new IC, etc. Fortunately, it is not difficult to create IC models in LineSim.

The first thing to decide is whether to model the IC with the .MOD or IBIS format. Read Chapter 4, section “IC-Model Formats” for a detailed comparison of the formats.

It is generally easier to create a model with the databook-style .MOD format, because:

- ♦ LineSim includes a dialog-box editor for .MOD models, which creates model libraries and files for you
- ♦ less device data is required to create a .MOD model than to create an IBIS model
- ♦ there is almost always an existing, similar model to use as a starting point for a .MOD model

On the other hand:

- IBIS is the new, emerging signal-integrity modeling standard; creating a model is a good way of becoming familiar with IBIS
- an IBIS model is fairly easy to create using HyperLynx's Easy IBIS Wizard (see section "Running the Easy IBIS File Creation Wizard" above for details)
- IBIS models are portable to other simulators

In order to create a good .MOD model of a driver IC, you must have the following device data:

- the transistor technology (bipolar, Schottky bipolar, CMOS FET, etc.)
- the effective "on" resistance of the upper- and lower-stage output transistors
- the slew time of the low-to-high and high-to-low switching transitions

There are other parameters in the model, too, but the remainder are less critical; you can more safely approximate them, if needed.

In order to create a good IBIS model of a driver IC, you must have:

- at least an approximation of the upper- and lower-stage V-I output curves(although the Easy IBIS Wizard will create a curve for you if you know only the driving impedance, i.e., "on" resistance)
- the slew time of the low-to-high and high-to-low switching transitions

Thus the primary difference between the data requirements for a .MOD model and an IBIS model of a driver IC is the level of detail with which you need to know the driver's V-I output characteristics. A .MOD model runs surprisingly well knowing only the transistor's basic technology (is it bipolar?, CMOS?, etc.) and the effective "on" resistance of the output stage; for a good IBIS model, you need to know at least a few points on the V-I curve, or use the Easy IBIS Wizard, which will create a curve for you from an "on" resistance..

Note: *If you want to create an IBIS model but know only one V-I data point on an output stage's curve, it is critical to know where on the curve that point is. If the point is taken near the "knee" of the curve, such that the driver current saturates beyond the point, then you can safely enter a two-point table with entries 0,0 and V1,I1 in your IBIS table, or better yet, use the implied resistance in the Easy IBIS Wizard. But if the point is taken at the beginning or in the middle of the curve, such that the driver's current keeps increasing beyond the point, the two-entry table is erroneous.*

Why? Because if a voltage is above or below the two points, LineSim holds a driver's current at the previous value in the table in an attempt to model the driver's saturation. If the largest current in your table is significantly lower than the driver's actual maximum current, your model will be in error.

Example: Modeling an ASIC Driver

Suppose you need to model an output buffer on a CMOS ASIC. As far as you can tell, the output buffer looks similar to an AC-standard-logic-family driver, but may have a different slew rate, capacitance, and so forth. This example shows how you can create your own buffer model using the .MOD format.

First, collect whatever data you can about the output driver; this may require going back to the silicon vendor and requesting extra information. The most-critical data is for output V-I characteristics and slew times. (Many vendors are starting to publish V-I curves in their data sheets.)

To begin creating the model:

1. With no schematic created or loaded in LineSim, from the Edit menu choose Databook IC Models.
2. In the Library and Model area, choose Model Library GENERIC.MOD. A model from this library often makes a good starting point for a new model.
3. Choose Device Model 74ACXX:GATE, since the ASIC output is somewhat similar to the 74AC standard-logic family's.
4. Click the Save As button; in Save .MOD Model As dialog box, type Library Name "ASIC.MOD" and Model Name "OUTPUT".

5. Click OK. This saves the 74AC model into a separate model and library (ASIC.MOD), which can now be edited to create the ASIC model.

To edit the model so it matches the ASIC output:

1. In the Output Drivers area of the Edit .MOD Model dialog box, leave the Type parameters set to CMOS.

2. Calculate an effective "on" resistance for the ASIC buffer: find the "knee" point in the upper stage's curve (the point where the current starts to "roll off" or saturate) and the zero-current point; calculate R_on =delta_V/delta_I; enter this in the high stage's Resistance box; *then repeat for the lower stage.* If you do not have this data, you can measure it using a sample IC, a resistor, and a variable voltage supply.
   If you do not have time for even a simple measurement, then guess! Your models do not need to exact to get you a simulation that is at least "in the ballpark." Probably, a CMOS ASIC output is not much different than a 74AC output if the two are fabricated in similar geometries. But be conservative and make the ASIC run "hotter," say, 5 ohms.

3. Enter the Slew Time for the upper and lower stages. If you do not have this data, you can measure it with an oscilloscope. (But be sure that you use a high-bandwidth scope, preferably 500-MHz or above. Otherwise, you may measure only the scope's response, not the true slew time.)
   If you do not have time for a measurement, then guess! Again, the ASIC output probably is not much different than a 74AC output if the two are fabricated in similar geometries. But be conservative and make the ASIC run "hotter," say, 1.0 ns instead of 2.0.

4. Leave the Offset Voltage and Clamp Diode data the same as in the 74AC model. You could measure the effective diode resistance with a resistor and variable power supply, but *driver* clamp diodes are usually insignificant in signal-integrity simulations because the driver itself is such a low impedance.

5. If you know it, enter the driver output Capacitance. If you do not know it, leave the data the same as in the 74AC model. (The difference between, say, 5 pF and 7 pF is not terribly significant.)

To save the model:

1. In the Edit .MOD Model dialog box, click the Save button. You now have a custom model OUTPUT for your ASIC buffer, in a library called ASIC.MOD.

If You Decide to Create an IBIS Model Instead

If you decide to create your own model in IBIS format, use HyperLynx's Easy IBIS Wizard. This "smart" tool, which "interviews" you about your buffer's characteristics and then automatically generates syntactically correct, ready-to-simulate IBIS file from your data, makes creating IBIS files relatively easy, even for novices. See section "Running the Easy IBIS File Creation Wizard" above for complete details.

Creating Your Own Ferrite-Bead Models

A ferrite bead, even though a passive component, requires a complex model that cannot be summed up in simple numeric value (unlike a resistor or capacitor value). Accordingly, LineSim includes a library of ferrite-bead models, much like it includes libraries of IC models. (For details on ferrite-bead libraries and how to choose bead models, see Chapter 4, section "Choosing Ferrite-Bead Models."

Furthermore, just like LineSim allows you to create your own custom IC models and save them into libraries, LineSim also allows you to create custom ferrite-bead models, and save them into your own library.

How Ferrite Beads are Modeled

LineSim models a ferrite bead with an L-R-C model. (This is the same method as used by several of the more-sophisticated SPICE packages.) However, LineSim does not require the models' creator to know the values of L, R, and C; these are complex and would probably never be known even to the vendor of a particular ferrite bead.

Instead, LineSim *synthesizes* an equivalent ferrite-bead model from four pieces of data that can be read from any basic ferrite-bead data sheet:

- the bead’s DC resistance (including package resistance), and
- three points of impedance versus frequency

Even summary data sheets on a ferrite bead almost always give these values: a DC resistance and a graph of impedance versus frequency. The three Z-vs.-f data points can be read from the graph.

Library File for User-Defined Bead Models: USER.FBD

LineSim ships with a library, BSW.FBD, that contains a representative sampling of ferrite-bead models from several leading manufacturers. Usually, even if the exact bead you want to simulate is not modeled in BSW.FBD, you can find a close substitute among the shipping bead models. (For details on how to view the contents of BSW.FBD, see Chapter 4, section “Choosing Ferrite-Bead Models.”)

Still, if you are using a bead which is not in the LineSim-supplied library and for which you want an exact model, you can create a custom model and store it in a library called “USER.FBD.” When you interactively model a ferrite bead in LineSim, the program reads the models in BSW.FBD, and then, if USER.FBD exists, reads its models, too. In the Select Ferrite Bead Model dialog box, the models from USER.FBD are promoted to the beginning of the Vendor list box, so that you see your custom models first. (For details on choosing ferrite-bead models, see Chapter 4, section “Choosing Ferrite-Bead Models.”)

Ferrite-bead (.FBD) library files must be stored in the root LineSim directory. This differs from IC-model libraries (which are stored in the LIBS sub-directory under the LineSim root directory).

Syntax for Ferrite-Bead Models

USER.FBD must be written in LineSim’s .FBD-file format, which is described in Appendix C. The BSW.FBD file contains a header which succinctly describes the format. If you choose to look at BSW.FBD for a format definition (or even to use it as a starting point for your own USER.FBD file), make a copy of the file first and edit the copy; be careful not to edit or otherwise damage BSW.FBD itself.

How to Create a USER.FBD Library

USER.FBD must be ASCII-only; create it in a text editor (like the HyperLynx File Editor), not an editor that introduces non-ASCII formatting characters into the file. The file must be located in LineSim's root directory (i.e., the directory that BSW.EXE is installed in.)

You might want to copy a portion of BSW.FBD to USER.FBD to give yourself a "head start" on creating the new library. Then you can modify existing bead models to create your own. Be careful not to leave any bead-model names in USER.FBD that already exist in BSW.FBD, where "names" means combinations of vendor and part-number names.

For an example of how, in detail, to model a particular ferrite bead, see "How to Create a Custom Ferrite-Bead Model" below.

LineSim Hint: How to Create a Custom Ferrite-Bead Model

LineSim ships with a library of ferrite-bead models (BSW.FBD). However, eventually you may want to model a bead not contained (or with no close equivalent) in LineSim's library. Fortunately, it is easy to add your own ferrite-bead models to the user-defined bead library, USER.FBD.

When you first install LineSim, there is no file USER.FBD. You create it the first time you need to add your own bead model.

The format for .FBD files is described in Appendix C. Before you attempt to create your own definition, you should read the specification thoroughly.

USER.FBD Example: Defining a Bead Model

Suppose you need to model a bead called "Matic19" from a vendor named "Bead-O-Matic." Assume there is no model for this bead in BSW.FBD.

To create USER.FBD:

1. In a text editor (like the HyperLynx File Editor), begin editing a new file. Be sure you use a text editor, not a word processor that inserts non-ASCII formatting characters into the file.

2. At the top of the file, place these two lines:

   ```
   {FBD}
   {VERSION=1.0}
   ```

To enter the bead's definition:

1. Immediately following the two header lines, add the definition of the new bead:

   ```
   ********************* My Ferrite Bead Models *********************
   {MANUFACTURER=Bead-O-Matic}
   {BEAD=Matic19
    (R_DC=0.035)
    (PT1=6.0MHZ,   4.0)
    (PT2=100.0MHZ,19.0)
    (PT3=500.0MHZ,27.0)
   }
   ```

   See "Where the Bead Came From" below for a description of how the bead-model data was determined.

2. End the file with the line:

   ```
   {END}
   ```

To save USER.FBD:

1. Save the file as USER.FBD, into LineSim's root directory. (For example, if LineSim is installed in C:\BSW, save the file as C:\BSW\USER.FBD).

The new bead model will be available in LineSim as soon as you load a schematic or create a new one. (BSW.FBD and USER.FBD are read every time a schematic is loaded.)

Where the Bead Data Came From

In the example model above, the DC resistance value and three impedance-versus-frequency points are all taken directly from the bead's data sheet. (It is standard practice that ferrite-bead data sheets include
Z-vs.-f graphs.)

The only "trick" to creating a model is to know which three points to take from the impedance graph. The .FBD-format specification in Appendix C lists detailed rules for choosing the three points. In this case, the frequency points were at:

- about 10% of the nominal frequency (100 MHz)
- the nominal frequency (since the resonant frequency was not available — off the graph)
- the highest frequency on the graph

Example USER.FBD File

Below is a complete sample USER.FBD file, for two imaginary ferrite beads:

```
{FBD}
{VERSION=1.0}

********************** My Ferrite Bead Models **********************
{MANUFACTURER=Bead-O-Matic}
{BEAD=Matic19
 (R_DC=0.035)
 (PT1=6.0MHZ,   4.0)
 (PT2=100.0MHZ,19.0)
 (PT3=500.0MHZ,27.0)
}

*****************************************************************
{MANUFACTURER=Bead-O-Rama}
{BEAD=Bead120
 (R_DC=0.42)
 (PT1=2.0MHZ,   3.0)
 (PT2=100.0MHZ,120.0)
 (PT3=300.0MHZ,200.0)
}
{END}
```

Chapter 6: Modeling Transmission Lines

Summary

This chapter describes how to model transmission lines. Specifically, it discusses:

- ♦ what “modeling” a transmission line means
- ♦ how to model a transmission line electrically
- ♦ how to model a transmission line geometrically using line-by-line PCB cross sections
- ♦ how to model a transmission line geometrically using a complete stackup
- ♦ how to model connectors, cables, and wires
- ♦ how to copy transmission-line models
- ♦ how to label a transmission line with a comment
- ♦ how to set the units used to measure geometric parameters (lengths and copper weights)

For details on how to model ICs and passive components, see Chapter 4.

What Does “Modeling” Transmission Lines Mean?

Before you can simulate a schematic you have entered, you must choose models and edit values for the components on the net. What steps are required depends on each component’s type:

for an IC...	...choose model (see Chapter 4)
for an R...	...check value; modify if needed (see Chapter 4)
for a C...	...check value; modify if needed (see Chapter 4)
for an L...	...check value; modify if needed (see Chapter 4)
for a ferrite bead...	...choose model (see Chapter 4)
for a transmission line...	...choose model

This chapter discusses how to model transmission lines; for details on modeling ICs and passive components, see Chapter 4.

Transmission-Line Models

Before simulating a schematic, you must create electrical models for the transmission lines in the schematic. The transmission lines in your schematic can represent any kind of electrical connection (PCB trace, connector, cable, and so forth; see Chapter 3, “Transmission Lines and Schematics” for details); the job of modeling the transmission lines means creating an electrical representation for each connection.

Characteristic Impedance and Propagation Velocity

Every transmission line is defined electrically by two key properties:

- characteristic impedance (Z0)
- propagation velocity

Together, these parameters determine how signals interact with and propagate along a transmission line.

Characteristic impedance (or “Z0”) is a property unique to the distributed nature of transmission lines. Because transmission lines consist of a continuous mixture of capacitance and inductance, they “look” instantaneously to a transmitted signal like a resistance.

Transmission-line impedance affects such behavior as signal reflection and step size (the percentage of a switching signal’s swing that enters a transmission line).

Propagation velocity specifies how quickly a signal travels along a transmission line.

Propagation velocity determines, for example, whether or not a signal traveling on a PCB trace is likely to exhibit transmission-line effects. If the total delay time down a PCB trace is short compared to how fast the driving IC switches, the trace will not behave much like transmission line. If the delay time is long, the transmission-line effects become significant.

Transmission-Line Modeling Methods

LineSim gives you several options for modeling a transmission line. You can:

- model the line electrically, by directly entering its characteristic impedance and propagation delay
- model the line geometrically, by tying it individually to a particular kind of PCB cross-section geometry

- model the line geometrically, by tying it to a PCB stackup you've created for the whole schematic
- model the line as a connector, cable, or wire over a ground plane

If you choose to model the line geometrically with a cross section, you can choose from these cross-section types:

- microstrip
- buried microstrip
- stripline

For details on modeling with each method, see the sections below.

Note: *The explicit modeling of transmission lines is one key way in which using LineSim differs from using BoardSim. In BoardSim, transmission-line modeling is automatic, because the transmission-line properties are determined completely by the routing on your PCB + the PCB's stackup.*
LineSim is a pre-route / what-if tool; it makes no assumptions about what your routing will look like. Also, LineSim is not limited to modeling PCB traces; you can use it to model multiple PCBs, connectors, cables, and entire electronic systems.

Choosing a Modeling Method

Direct electrical modeling of a transmission line is the fastest modeling method, if you know the electrical properties to use.

Electrical Modeling ("Simple Model")

For example, if you're running hypothetical experiments with the simulator (e.g., "How long can this line be before we're in trouble?" or "I wonder how low-impedance a line this IC can drive?"), direct electrical modeling is quite convenient. The electrical modeling method may also make sense if you're building controlled-impedance PCBs, for example, if you *know* that every trace on your board is going to be 50 ohms.

LineSim refers to an electrical model as a “simple” model.

Geometric Modeling

On the other hand, in many instances you know what a connection looks like *geometrically* (e.g., how long, built in what type of PCB cross section, and so forth), but not what that geometry implies *electrically*. In these cases, LineSim can convert your geometric parameters to electrical ones automatically.

Stackup Modeling

If your schematic represents a circuit that exists entirely on one PCB, or several PCBs that can all have the same stackup, model your PCB transmission lines with the “stackup” method. This allows you to place each transmission line on a layer in the stackup, then to modify all of the stackup-style lines simultaneously by making global changes in LineSim’s stackup editor. (See Chapter 7 for details on the stackup editor.)

Hint: *If you have a multi-PCB schematic, and the PCBs have different stackups, you can only model one of the stackups with the stackup method. Choose the board with the most transmission lines as the one to be stackup-modeled; the lines on other boards must be modeled with other methods, e.g., individual cross sections.*

Line-by-Line Cross-Section Modeling

If you do not want to tie certain transmission lines in your schematic to a PCB stackup, you can model them individually with an explicit cross-section method. LineSim has individual modeling windows for every type of cross section possible on a multi-layer PCB.

Modeling a Transmission Line Electrically (“Simple Model”)

When you model a transmission line by entering its electrical properties directly, you can specify the line’s:

- characteristic impedance (in ohms)
- propagation delay (in nanoseconds)
- DC resistance (in ohms)

To model a transmission line electrically:

1. Point in the schematic to the transmission line. When you point to the line, a red box appears around it.
2. Click with the **right** mouse button. The Edit Transmission Line dialog box opens.
3. Verify that the Transmission Line Type is set to Simple. If not, click the Simple radio button.
4. In the Transmission Line Properties area, type the line's electrical properties. Click OK.

As you enter the transmission line's electrical properties, LineSim calculates and displays the line's inductance and capacitance. (You cannot enter L and C; they are display-only properties.)

When you close the dialog box, the transmission line's electrical properties are exported to the line in the schematic. The values display in blue on the line.

Modeling a Transmission Line Geometrically with a Cross Section

When you model a transmission line by specifying an individual cross section for the line, you choose between the cross-section types possible on a multi-layer PCB.

PCB Cross Sections

On a multi-layer PCB with at least one ground plane, there are three possible cross section types:

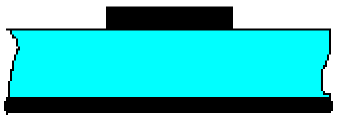
- microstrip
- buried microstrip
- stripline

The following sections describe each cross section.

Microstrip

A microstrip PCB trace is an outer-layer trace, bound on one side by air and on the other by dielectric. See Figure 5-1.

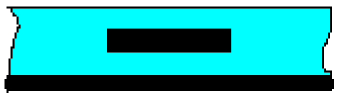
Figure 5-1: A microstrip cross section.



Buried Microstrip

A buried microstrip is an inner-layer trace, but with an AC-ground plane to only one side. For example, on a six-layer board with planes at layers 3 and 4, layers 2 and 5 are buried microstrips. See Figure 5-2.

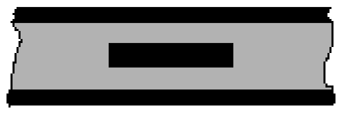
Figure 5-2: A buried microstrip cross section.



Stripline

A stripline is an inner layer trace, with an AC-ground plane to both sides. For example, on a six-layer board with planes at layers 2 and 5, layers 3 and 4 are striplines. See Figure 5-3.

Figure 5-3: A stripline cross section.



Entering a Cross-Section Model

To model a transmission line as a cross section:

1. Point in the schematic to the transmission line. When you point to the line, a red box appears around it.
2. Click with the **right** mouse button. The Edit Transmission Line dialog box opens.
3. In the Transmission Line Type area, click the appropriate cross-section radio button (Microstrip, Buried Microstrip, or Stripline). The dialog-box tab changes automatically to the Values setting.
4. In the edit boxes, type the transmission line's geometric properties (see "Geometric Properties" below for details). Click OK.

As you enter the transmission line's geometric properties, LineSim calculates and displays the line's corresponding electrical properties (in the Electrical Properties area). See "Electrical Properties" below in this chapter for a description of what each property means.

Note: *You cannot enter a cross section's electrical properties; they are calculated only. If you want to enter properties electrically, use the "Simple" model instead (see "Modeling a Transmission Line Electrically" above in this chapter for details).*

When you close the dialog box, the transmission line's electrical properties are exported to the line in the schematic. The values display in blue on the line.

Geometric Properties

The geometric properties that can be entered in a cross-section dialog box vary somewhat from section type to section type. Some properties apply to all cross sections; others do not. For example, Plating Thickness is available only for microstrips.

The following table lists all of the possible geometric parameters, and explains what they mean.

Geometric Parameter	**Description**
Length	The length of a trace, cable, or wire, used to calculate line delay and total L, C, and R.
Plating Thickness	The amount of copper plating used over an outer-layer trace. 1 ounce is a common value.
Conductor Thickness	The amount of base copper used to make a trace. 0.5 ounce is a common value.
Width	The total cross-sectional width of a trace, resulting from base and plated copper. Can't be determined directly from the copper weights; depends on the etching process used.
Radius	The radius of a circular wire.
Dielectric Height	The height (or thickness) of the dielectric between a trace or wire and AC-ground (or "reference") plane.
Dielectric Constant	The constant describing the dielectric properties of the circuit-board or other insulating material. See Appendix D.

Electrical Properties

The following electrical properties are calculated automatically by LineSim, based on the geometric properties you enter for a cross section:

Electrical Property	**Description**
Impedance (Z0)	The characteristic impedance of a trace, cable, or wire.
Delay	The propagation delay from one end of the trace, cable, or wire to the other.
Inductance (L)	The total inductance of a trace, cable, or wire.
Capacitance (C)	The total capacitance of a trace, cable, or wire.
Resistance (R)	The total DC resistance of a trace or wire, assuming copper and 20 degrees Celsius.

Units for Length and Thickness

The units used in the cross-section dialog boxes to measure length and thickness can be changed to select between English and metric, and for thicknesses between weight and length. For details on how to change the units, see “Setting Measurement Units” below in this chapter.

Allowable Geometric Values

LineSim allows a wide range of values for cross-section geometric properties. Still, for some extreme values, an error window may open specifying that the value is too large or small.

Modeling a Transmission Line Geometrically with a Stackup

An alternative to modeling transmission lines one-by-one with individual cross sections is to tie the lines to a global stackup. This method has the advantage of letting you change the stackup and having all of the stackup-style transmission lines change properties automatically with the stackup. See "Choosing a Modeling Method" above in this chapter for more details on how to choose a transmission-line modeling method.

Creating a Stackup

In LineSim, stackups are created and edited using the stackup editor. For details on the stackup editor, see Chapter 7.

The following sections assume that you've created a stackup using the editor.

Entering a Stackup-Based Model

To model a transmission line based on a stackup:

1. Verify that the stackup on which you want to base your modeling has already been created in the stackup editor. (For details on creating stackups, see Chapter 7, section "Creating and Editing a Stackup.")

2. Point in the schematic to the transmission line. When you point to the line, a red box appears around it.

3. Click with the **right** mouse button. The Edit Transmission Line dialog box opens.

4. In the Transmission Line Type area, click the Stackup radio button. (If you own LineSim Crosstalk, be sure to click the Stackup radio button in the Uncoupled list, not the one under the Coupled list — unless you're actually trying to define a coupled line.) The dialog-box tab changes automatically to the Values setting.

5. In the Trace Properties area, pull down the Layer combo box, and click on the stackup layer to which you want to assign the transmission line.

6. In the edit boxes, type the transmission line’s length and width. Click OK.

As you enter the transmission line’s stackup layer and geometric properties, LineSim calculates and displays the line’s corresponding electrical properties (in the Electrical Properties area). See “Electrical Properties” above in this chapter for a description of what each property means.

Note: *You cannot enter a stackup-based transmission line’s electrical properties; they are calculated only. If you want to enter properties electrically, use the “Simple” model instead (see “Modeling a Transmission Line Electrically” above in this chapter for details).*

When you close the dialog box, the transmission line’s electrical properties are exported to the line in the schematic. The values display in blue on the line.

Effects of a Stackup-Based Model

When you assign a transmission line to a stackup layer, you are assigning it to a layer with a certain *name*, not to an absolute layer number.

For example, suppose you have a four-layer stackup including outer layers TOP (layer 1) and BOTTOM (layer 4). Then suppose you assign a transmission line to layer TOP, i.e., to layer (as it would appear in the layer combo-box list):

1, Signal, TOP

Then suppose you use the stackup editor to drag layer TOP to the bottom of the stackup, and layer BOTTOM to the top. The transmission line now resides on the bottom of the board, on the layer TOP that has been moved to the bottom. Sure enough, if you right-click on the line to open the Edit Transmission Line dialog box and click on the Values tab, the layer combo box shows:

4, Signal, TOP

The ability to assign transmission lines to layer *names* is very powerful: it allows you to make changes to *all* of the stackup-based lines in your schematic simply by editing the global stackup. This makes it very easy to experiment with the board-wide effects of stackup changes.

Note that you can only assign a transmission line to a layer that exists in the current stackup. You cannot create layers in the Edit Transmission Line dialog box; the stackup can only be edited in the stackup editor (see Chapter 7 for details.)

Modeling a Transmission Line as a Connector

A good first-order model for a connector is a transmission line with a characteristic impedance and propagation delay that match the effective impedance and delay of the connector.

The .SLM Connector-Modeling Format

LineSim can read connector-description files in an ASCII format called ".SLM" ("single-line model"). This format, based on SPICE syntax, has been defined by HyperLynx (see Appendix B), and is fully compatible with the format in which AMP, Inc. supplies single-line models for many of their connectors.

You can use LineSim to model connectors by reading a model out of a .SLM file, and applying it to a transmission line in your schematic.

Location of .SLM Files

For LineSim to find .SLM files and give you access to them in the user interface, the files must be placed in the models directory, along with your IC models. For details on specifying the location of the models directory, see Chapter 5, section "Setting the Models Directory."

Contents of .SLM File

Each .SLM library file represents a single connector. Multiple models in the file then represent different portions of the connector, for example, different rows, each of which may have a different impedance. The models generally represent mated connector pairs, i.e., both sides of the connector plugged together.

For a description and example of the .SLM format, see Appendix B.

Loading a Connector Model

To load a transmission line with a single-line connector model:

1. Point in the schematic to the transmission line. When you point to the line, a red box appears around it.
2. Click with the **right** mouse button. The Edit Transmission Line dialog box opens.
3. In the Transmission Line Type area, click the Connector radio button. The dialog-box tab changes automatically to the Connectors setting.
4. In the Connector list box, click on the name of the .SLM connector you want to model.
5. In the Pin Model list box, click on the name of the pin or pin type you want. (Some models have only one pin type.) Then click OK.

When you close the dialog box, the connector's electrical properties are exported to the line in the schematic. The values display in blue on the line.

The Connectors Tab

As you choose different libraries and models, information about the currently selected model is read and displayed. Specifically shown are:

- in the Connector Information area:
  - from the currently selected connector, the copyright notice
  - from the currently selected connector, the connector's part and catalog number
  - for the currently selected pin model, the characteristic impedance, propagation delay, and DC resistance
- in the File Viewer list box, for the currently selected model, the library-file source text that describes the model

About Grounding in a Connector

Connector models very often assume that the connector is used with a particular grounding scheme, since the location of ground pins in the connector is essentially what determines others pins' impedances and delays. Be sure to read information in the source text of any particular model related to grounding schemes, to avoid using a model that does not represent your application of the connector.

Modeling a Connector without a .SLM File

If you need to model a connector for which you do not have a .SLM model, you must supply the electrical parameters (characteristic impedance, propagation delay, and DC resistance) yourself. (You could get them, for example, from a data sheet or a connector-vendor's application engineer.)

Once you know the electrical parameters, you can either:

♦ model the connector using a "simple" transmission-line model (enter the electrical data directly; see "Modeling a Transmission Line Electrically" above in this chapter for details)

OR

♦ create a .SLM file to represent the connector

If you choose to create a .SLM file for the connector, see Appendix B for details on the .SLM format.

Modeling a Transmission Line as a Cable or Wire

LineSim includes assistance for modeling transmission lines that represent cables and for wires near an AC-ground plane.

Modeling Cables

LineSim has a built-in list of electrical parameters for certain industry-standard cable types, mostly coaxial cables.

To model a transmission line with a built-in cable type:

1. Point in the schematic to the transmission line. When you point to the line, a red box appears around it.

2. Click with the **right** mouse button. The Edit Transmission Line dialog box opens.

3. In the Transmission Line Type area, click the Cable radio button. The dialog-box tab changes automatically to the Cables setting.

4. Click the radio button for the desired cable type.

5. In the Cable Length edit box, type the transmission line's length. Click OK.

As you choose between cable types and vary the transmission line's length, LineSim calculates and displays the line's electrical properties. (You cannot enter the electrical properties; they are display-only.)

When you close the dialog box, the transmission line's electrical properties are exported to the line in the schematic. The values display in blue on the line.

Modeling Cables that are Not Built-In

If you need to model a cable that is not in LineSim's list, use LineSim's "Simple" electrical model.

To create an electrical model for a cable, you must know the cable's:

- ♦ inductance per unit length, *L*
- ♦ capacitance per unit length, *C*
- ♦ total length, *l*

Inductance and capacitance per unit length are parameters that are available from the cable manufacturer, and usually printed in the cable's data sheet.

Note: *For multi-conductor cables, the parameters typically depend on the grounding configuration used in the cable, e.g., whether you place ground returns on every other wire, every third wire, and so forth. Be sure to account for the grounding scheme you're actually using.*

Then, calculate the cable's characteristic impedance (Z0) and propagation delay from these formulas:

$$Z0 = \sqrt{\frac{L}{C}}$$

$$delay = l \bullet \sqrt{LC}$$

Finally, follow the steps described in “Modeling a Transmission Line Electrically” (above in this chapter) to create the electrical model and export it to the schematic.

Modeling a Wire Over Ground

LineSim can model a wire over an AC-ground plane, provided that the wire has a circular cross section.

To model a wire-over-ground transmission line:

1. Point in the schematic to the transmission line. When you point to the line, a red box appears around it.
2. Click with the **right** mouse button. The Edit Transmission Line dialog box opens.
3. In the Transmission Line Type area, click the Wire Over Ground radio button. The dialog-box tab changes automatically to the Values setting.
4. In the edit boxes, type the transmission line's geometric properties (see “Geometric Properties” above in this chapter for details). Click OK.

As you enter the transmission line's geometric properties, LineSim calculates and displays the line's corresponding electrical properties (in the Electrical Properties area). See "Electrical Properties" above in this chapter for a description of what each property means. (You cannot enter the electrical properties; they are display-only.)

When you close the dialog box, the transmission line's electrical properties are exported to the line in the schematic. The values display in blue on the line.

Copying Transmission Lines

Once you have chosen a model for one transmission line, it may be convenient to quickly "copy" the same model to other lines. This is particularly true of bus-oriented schematics.

For example, suppose you're entering a schematic that describes a bus with ten equally spaced taps. Once you've modeled the transmission line between the first and second taps, it is convenient to paste the same model to the other inter-tap lines on the net.

To make a copy of a transmission-line model:

1. Point in the schematic to the transmission line. When you point to the line, a red box appears around it.

2. Click with the **right** mouse button. The Edit Transmission Line dialog box opens.

3. Click the Copy button. The Transmission Line to Paste area fills with the line's properties, including its comment. Click OK; the dialog box closes.

The Transmission Line to Paste area is a storage buffer: it always remembers the model you last copied, even after the Edit Transmission Line dialog box closes.

Pasting to Another Transmission Line

To paste the last-copied transmission-line model to another line:

1. Back in the schematic editor, point to the line to which you want to paste the copied model. Click with the right mouse button. The Edit Transmission Line dialog box opens.

2. Click the Paste button, then OK.
   OR
   Press <return> on the keyboard.

The dialog box closes, and the copied model's parameters are exported to the pasted line.

Labeling a Transmission Line (Transmission-Line "Comment")

Sometimes it is convenient to label the transmission lines in a schematic in order to better document the circuit you've entered. LineSim refers to a label on a transmission line as a "comment."

To attach a comment label to transmission line:

1. Point in the schematic to the transmission line. Click with the **right** mouse button. The Edit Transmission Line dialog box opens.

2. In the Transmission Line Properties area, in the Comment edit box, type the desired comment.

3. Click OK.

The comment is displayed in the schematic editor, in white near the transmission line.

Setting Measurement Units

What the Measurement Units Do

LineSim allows you to choose in which units you want to view lengths and metal thicknesses in the Edit Transmission Line modeling windows.

For lengths, you choose a measurement system:

- English
- metric

For metal thicknesses (base copper thickness and plating thickness), you choose between:

- length
- weight

The default settings are “English” and “weight,” e.g., lengths in inches and base copper in ounces. International users might prefer “metric” and “length,” e.g., X,Y positions in centimeters and base copper in microns.

Choosing the Measurement System and Metal-Thickness Units

To choose the measurement system:

1. From the Options menu, choose Units.
2. In the Measurement Units area, click the appropriate radio button.
3. Click OK.

To choose the metal-thickness units:

1. From the Options menu, choose Units.

2. In the Metal Thickness Units area, click the appropriate radio button.
3. Click OK.

The changes takes effect immediately.

Chapter 7: Editing Stackups

Summary

This chapter describes:

- review of how stackups are used in LineSim (see Chapter 6, section “Transmission-Line Modeling Methods” for more details)
- what elements make up a stackup
- how to create and edit a stackup
- how LineSim flags stackup errors
- how to calculate characteristic impedances
- how to document a stackup
- some restrictions LineSim places on stackups

How Stackups are Used in LineSim

LineSim uses a stackup only to accomplish one method of transmission-line modeling: the “stackup” model type (see Chapter 6, section “Modeling a Transmission Line Geometrically with a Stackup” for details.) If you never model transmission lines with this method, you never need to worry about stackups in LineSim.

On the other hand, the stackup method of modeling transmission lines is powerful, since it allows you to control the electrical properties of many transmission lines simultaneously by editing a single, global stackup.

For details on how to model transmission lines with the stackup method, see Chapter 6. This chapter describes how to create and edit the stackup itself.

Elements of a Stackup

“Stackup” refers to the how the metal and dielectric layers in a board are ordered and constructed.

There are three types of layers in a board stackup:

signal	a layer that carries signal traces
plane	a layer of solid metal, tied to a DC voltage
dielectric	a non-conducting layer separating two metal layers

The following sections describe in greater detail the geometric and material parameters of signal, plane, and dielectric layers, and of a complete stackup.

Plane Layers

A plane layer is a solid metal layer that is tied to a DC voltage, e.g., VCC or ground. Plane layers function electrically as AC grounds.

LineSim Assumptions about Plane Layers

The current version of LineSim assumes that plane layers allow for effective ground-return currents for the traces on a board. “Effective” means that the plane layer provides a low-inductance, close-to-the-trace return path.

From a practical standpoint, this means LineSim assumes that:

- ♦ plane layers are solid, not hatched or otherwise seriously “broken”
- ♦ plane layers are complete, not partial or mixed significantly with signal traces

If a plane layer in a stackup *is* seriously “compromised” in one of these ways, some of LineSim’s impedance calculations may be inaccurate. If you need to model transmission lines that interact with a compromised ground plane, do not model the lines with the stackup method (see Chapter 6 for descriptions of alternate modeling methods).

Note that the above restrictions do NOT mean that LineSim cannot simulate boards with planes that are not completely perfect. For example, consider a board with a split 5-V/3.3-V power plane. LineSim can handle perfectly well any traces which are isolated to one side or the other of the split (i.e., the 5-V side or 3.3-V side), because the return currents for such traces are never interrupted by the power-plane gap. Traces which cross the split, on the other hand, may be problematic, although even they may simulate with sufficient accuracy if enough bypass capacitors are available in the vicinity of the trace’s crossing of the gap to keep the trace’s return current from deviating too wildly from the signal current.

Signal Layers

A signal layer is a metal layer that contains signal traces. Traces may move between signal layers through vias.

Signal layers may be of any one of three cross-section types:

- microstrip
- buried microstrip
- stripline

LineSim automatically accounts for each signal layer’s cross-section type when it performs impedance calculations. See Chapter 6, section “PCB Cross Sections” for a detailed discussion of PCB cross sections.

Dielectric Layers

A dielectric layer is a non-conducting layer that separates two metal layers. Dielectric layers can be made from a variety of materials, though fiberglass is the most common in PCBs.

Dielectric Constants

Associated with every dielectric material is a property called “relative permittivity,” or “dielectric constant.” Dielectric constant measures how effective a material is in establishing a capacitance.

For a table of dielectric constants for common board dielectrics, see Appendix D.

Why Stackups Matter

Characteristic Impedance and Propagation Velocity

Nearly every detail of a board’s stackup affects two key characteristics of the trace segments on a board:

- ♦ characteristic impedance (Z0)
- ♦ propagation velocity

For a detailed discussion of these properties, see Chapter 6, section “Characteristic Impedance and Propagation Velocity.”

Stackup Parameters that Affect Impedance and Velocity

The following stackup parameters all affect the characteristic impedances and propagation velocities of the trace segments on a board:

- ♦ layer order
- ♦ trace thickness
- ♦ trace width
- ♦ dielectric thickness
- ♦ dielectric constant

LineSim lets you edit all of these parameters in its stackup editor, except for trace width, which is set line-by-line when you model each transmission line. (See Chapter 6, section “Entering a Stackup-Based Model” for details on entering a line’s model.)

Creating and Editing a Stackup

If you plan to model any of the transmission lines in your schematic with the stackup method, you need to first create a stackup for your schematic. Stackups are created in LineSim’s stackup editor.

The Default Stackup

When you open a new schematic in LineSim (see Chapter 3, section “Creating a New Schematic” for details), the schematic comes with a default stackup. This default stackup:

has six layers:

- two outer signal layers, “TOP” and “BOTTOM”
- two inner plane layers, “VCC” and “GND”
- two inner signal layers, “InnerSignal1” and “InnerSignal2”

The other parameters of the stackup — dielectric constant, thickness of dielectrics, and metal-layer thicknesses — are based on default-layer settings you make. See section “Setting Default Layer Characteristics” below for details on specifying the default characteristics of new stackup layers.

If the settings you supply for default layers are such that the default stackup would be thinner than 62.5 mils (still often used as a standard thickness for PCBs), the stackup is automatically thickened by increasing the thickness of the innermost dielectric layer. You may want to change this; see section “Editing Dielectric Layers” below for details on how.

Since the six-layer stackup is only a guess at what you’ll actually be using for your boards, normally you’ll use LineSim’s stackup editor to modify the default stackup to match the one you actually want to model. This may include adding or deleting layers, changing dielectric constants, changing thicknesses, and so forth.

Opening the Stackup Editor

To open the stackup editor:

1. From the Edit menu, choose Stackup.
   OR
   Click the Edit PCB Stackup button on the toolbar.

Changing Layer Order

To change the order of a stackup's layers:

1. In the stackup editor, in the graphical area that displays the stackup, position the mouse cursor over the layer you want to move.

2. Click the mouse button and continue holding it down, like you are beginning a drag-and-drop operation. Notice how the cursor changes to show that you have "grabbed" the layer.

3. Drag the layer up or down until the layer cursor's arrow is positioned at the place where you want to move the layer.

4. Release the mouse button. The layer moves to its new position.

Adding Layers

Adding Signal Layers

To add a signal layer:

1. In the stackup editor, in the graphical area that displays the stackup, click once on the layer that you want to add a signal layer above or below. Notice that a highlight box appears around the clicked-on layer.

2. Click the Signal radio button in the area with the Add Layer buttons.

3. To add a signal layer above the highlighted layer, click the Add Layer ^ button. To add a signal layer below the highlighted layer, click the Add Layer v button.

If you highlighted a metal layer, so that the new signal layer would have been shorted to it, a new dielectric layer is automatically added between the metal layer and the new signal layer. (The new dielectric layer is given a default thickness and dielectric constant.)

The new signal layer is given a default thickness; you can change this default value (see section “Setting Default Layer Characteristics” below for details). To edit the new layer, see “Editing Plane or Signal Layers” below in this chapter.

Adding Plane Layers

To add a plane layer:

1. In the stackup editor, in the graphical area that displays the stackup, click once on the layer that you want to add a plane above or below. Notice that a highlight box appears around the clicked-on layer.

2. Click the Plane radio button in the area with the Add Layer buttons.

3. To add a plane above the highlighted layer, click the Add Layer ^ button. To add a plane below the highlighted layer, click the Add Layer v button.

If you highlighted a metal layer, so that the new plane would have been shorted to it, a new dielectric layer is automatically added between the metal layer and the new plane. (The new dielectric layer is given a default thickness and dielectric constant.)

The new plane layer is given a default thickness; you can change this default value (see section “Setting Default Layer Characteristics” below for details). To edit the new layer, see “Editing Plane or Signal Layers” below in this chapter.

Adding Dielectric Layers

To add a dielectric layer:

1. In the stackup editor, in the graphical area that displays the stackup, click once on the layer that you want to add a dielectric above or below. Notice that a highlight box appears around the clicked-on layer.

2. Click the Dielectric radio button in the area with the Add Layer buttons.

3. To add a dielectric above the highlighted layer, click the Add Layer ^ button. To add a dielectric below the highlighted layer, click the Add Layer v button.

The new dielectric layer is given a default thickness and dielectric constant; you can change these default values (see section “Setting Default Layer Characteristics” below for details). To edit the new layer, see “Editing Dielectric Layers” below in this chapter.

Setting Default Layer Characteristics

As you edit a stackup and add layers to it (see the preceding sections for details), LineSim applies default values to the new layers (e.g., for thickness or dielectric constant). You can edit these values after the layer is added, but sometimes it is more convenient to define what default values you want LineSim to use for new layers — doing so may save your having to edit multiple, individual layers.

To set default stackup-layer values:

1. From the Options menu, choose Preferences.

2. Click the Default Stackup tab.

3. In the edit boxes, type the desired default values.

4. Click OK.

Now, when you create a new layer in the stackup editor, it will have the characteristics you just specified.

These default values also apply to all layers in the default stackup that LineSim creates when you ask it to create a new schematic. For details, see section “The Default Stackup” above.

Deleting Layers

Deleting Layers

To delete a layer:

1. In the stackup editor, in the graphical area that displays the stackup, click once on the layer that you want to delete. Notice that a highlight box appears around the clicked-on layer.

2. Click the Delete Selected Layer button.

Cannot Delete Signal Layers with Transmission Lines

The stackup editor does not allow you to delete signal layers that have transmission lines on them. Deleting such a layer would invalidate your schematic for simulation.

You can delete a signal layer if you first move the transmission lines on that layer to other layers, or if you remove the transmission lines on that layer from the schematic.

Editing Layers

Editing Dielectric Layers

Dielectric-Layer Parameters

A dielectric layer has the following parameters, all of which can be edited:

Parameter	**Required/Optional**
thickness	required
dielectric constant	required

Parameter	**Required/Optional**
material name	optional

Thickness and Dielectric Constant

For LineSim to simulate, a non-zero thickness and a non-zero dielectric constant are required. The thickness can be displayed in either English or metric units; see Chapter 6, section “Setting Measurement Units” for details. For a table of dielectric constants for common board dielectrics, see Appendix D.

The thickness both displays in the graphical stackup area and is printed with the stackup. The dielectric constant is not displayed, but is printed.

Material Name

For a dielectric layer, you can enter a name for the dielectric’s material. The material name is only for reference by a user running the stackup editor; it is not required and has no effect on simulation.

To edit a dielectric layer:

1. In the stackup editor, in the graphical area that displays the stackup, click once on the layer that you want to edit; then click the Edit Selected Layer button.
   OR
   Double-click on the layer that you want to edit.

2. To change thickness or dielectric constant, type the new data in the appropriate edit boxes. To change material name, click the Advanced button, then type the data.

3. Click OK (twice if you edited the material name).

Editing Plane or Signal Layers

Plane- and Signal-Layer Parameters

A plane or signal layer has the following parameters, all of which can be edited except for (in some cases) layer name:

Parameter	**Required/Optional**
base thickness	required
plating thickness	required (but can be 0.0)
layer name	required
bulk resistivity	required; *advanced parameter that you normally do not need to change*
temperature coefficient	required; *advanced parameter that you normally do not need to change*
passivation type	required; outer layers only; *advanced parameter that you normally do not need to change*
material name	optional

Base and Plating Thickness

For LineSim to simulate, a non-zero base thickness is required for every signal or plane layer; the plating value can be any value, including zero. Base thickness refers to the thickness of the base metal; plating adds to the base thickness. Plating thickness is normally 0.0 for inner layers. Both parameters can be displayed in either English or metric units, and in length or weight units; see Chapter 6, section “Setting Measurement Units” for details.

The combination of base + plating thickness displays in the graphical stackup area, and is printed with the stackup.

Layer Name

For a signal or plane layer, you can enter a layer name only *before* the layer has had a transmission line in the schematic placed on it with the stackup modeling method (for details on stackup modeling of transmission lines, see Chapter 6, section “Modeling a Transmission Line Geometrically with a Stackup.”) Once a transmission line somewhere in the schematic has been assigned to a layer, the layer’s name can no longer be edited.

This is true even if the assigned transmission line has subsequently been assigned to another layer or modeling method, or removed from the schematic. Therefore, you should name all of your layers before assigning any transmission lines to them.

Note: *When creating stackups, avoid using layer names that contain any of the following characters:*

( ) { }

These characters are used as delimiters in the stackup section of LineSim’s .TLN file. Names containing the delimiter characters may cause a .TLN file to be read incorrectly, after you save it and then try to re-load it.

Bulk Resistivity

Every signal and plane layer is required to have a bulk resistivity for the layer’s metal material. The resistivity is used when LineSim calculates DC resistances for trace segments on the layer.

Bulk resistivity is considered by LineSim to be an “advanced” parameter, i.e., one which you normally do not need to change. Signal and plane layers automatically default to the bulk resistivity of copper.

For more details on bulk resistivity and how it is used, see “Calculating DC Resistance” below.

Temperature Coefficient

Every signal and plane layer is required to have a resistivity temperature coefficient for the layer’s metal material. The temperature coefficient is used in conjunction with the layer’s resistivity when LineSim calculates DC resistances for trace segments on the layer.

Temperature coefficient is considered by LineSim to be an “advanced” parameter, i.e., one which you normally do not need to change. Signal and plane layers automatically default to the temperature coefficient of copper.

For more details on the temperature coefficient and how it is used, see “Calculating DC Resistance” below.

Passivation Type

Every *outer* signal and plane layer is required to have a passivation type. The passivation type tells LineSim how trace widths vary depending on the method used to passivate your board’s outer layers. Inner layers are not passivated and therefore have no passivation-type selection. Also, the passivation type is only used by LineSim when fabrication compensation is enabled for impedance calculations; for more details, see “Fabrication Compensation” below in this chapter.

Passivation type is considered by LineSim to be an “advanced” parameter, i.e., one which you normally do not need to change.

For more details on passivation type and how it is used, see “Fabrication Compensation” below.

Material Name

For a signal or plane layer, you can enter a name for the metal material. The material name is only for reference by a user running the stackup editor; it is not required and has no effect on simulation.

To edit a signal or plane layer:

1. In the stackup editor, in the graphical area that displays the stackup, click once on the layer that you want to edit; then click the Edit Selected Layer button.
   OR
   Double-click on the layer that you want to edit.

2. To change base thickness, plating thickness, or layer name, type the new data in the appropriate edit boxes. To change bulk resistivity, temperature coefficient, passivation type, or material name, click the Advanced button, then type the data (or choose the passivation type).

3. Click OK (twice if you edited an Advanced property).

Changing a Layer from One Type to Another

The stackup editor allows you to change a layer from one type to another, although this is not typically an operation you need to perform in LineSim.

To change a layer from one type to another:

1. In the stackup editor, in the graphical area that displays the stackup, click once on the layer whose type you want to change; then click the Edit Selected Layer button.
   OR
   Double-click on the layer.

2. In the Set Layer Type area, click the radio button for the new type you want the layer to be.

3. Click OK.

Note: *The only layer-type change that LineSim does not allow is from signal to dielectric, for signal layers that have had transmission lines assigned to them. This change would invalidate your schematic for simulation.*

About “Field Solver” Messages

When you finish editing a stackup (i.e., make changes and then close the stackup-editor dialog box), LineSim briefly calls its field solver to characterize certain aspects of the new stackup. Often, you will see a progress dialog box labeled “HyperLynx” and “Running field solver” while this analysis is running.

The field solver is called regardless of whether or not you are licensed for LineSim’s Crosstalk option. If you are not licensed for Crosstalk analysis, then this is the only time the field solver runs; it is not available during simulation or any other kind of analysis unless you own the Crosstalk option.

Stackup-Error Reporting

Current Errors

If there are currently any errors in the stackup, or if an editing change you make to the stackup causes an error, the stackup editor reports the error immediately. An “error” is any situation that causes the stackup to be electrically invalid, e.g., two signal layers shorted together with no intervening dielectric.

Errors appear in the status line at the top of the stackup-editor dialog box, above the graphical area that displays the stackup. Errors are reported in a red font. If there are no errors, i.e., the stackup is electrically valid, the status line changes to a black font and reports “no errors.”

If there are multiple errors simultaneously, the status line reports them one-at-a-time. Continue fixing the indicated errors until the status line reports no errors.

Always watch the status line as you edit a stackup. The status messages tell you immediately if an editing change has made the stackup invalid.

The Stackup Wizard

The feature in LineSim that reports errors as you edit a stackup is called the "Stackup Wizard.” In addition to reporting errors, the Stackup Wizard can also attempt to automatically fix a “broken” stackup, e.g., by adding dielectrics where they’re missing, and so forth. Normally, though, you would fix errors manually in the stackup editor, since you’re creating the stackup anyhow.

To force the Stackup Wizard to run:

1. From the Wizards menu, choose Stackup Wizard.

The Stackup Wizard opens and runs on the current stackup. If it finds any errors and/or makes any changes to the stackup, they are reported in the Wizard window.

You can also check the results from the last time the Wizard ran by clicking the View Stackup Wizard button in the stackup editor.

Measurement Units

The units used in the stackup editor to measure thicknesses and copper weights can be changed between English and metric (for the measurement system) and length and weight (for copper thickness).

Normally, you use the Preferences choice from the Options menu to change units. However, if you already have the stackup editor open, there is a quicker method than exiting the editor and using the menu.

To change measurement units from inside the stackup editor:

1. With the stackup editor open, click the Measurement Units button. The Units dialog box opens.
2. Make units selections as described in Chapter 6, section “Setting Measurement Units.”

Total Board Thickness

The stackup editor displays the total thickness of your stackup (look below the graphical stackup area). As you edit the stackup, the total thickness changes.

Note: *Total thickness may affect the manufacturability of your board. For example, 62 mils is a standard thickness for many fabricators. Thick boards often increase drilling cost because fewer panels at-a-time can be drilled. On the other hand, an extra-thick board is often used for backplane applications because of the improved rigidity.*

Viewing Characteristic Impedances

When you model a transmission line with the stackup method (see Chapter 6, section “Entering a Stackup-Based Model” for details), LineSim automatically calculates the line’s characteristic impedance and exports it to the schematic. The impedance is displayed in the schematic editor, in blue on the transmission line itself.

You can also view trace impedances in the stackup editor. From the Edit menu, choose Stackup; then follow the steps in “Test Trace Width” below.

Test Trace Width

If you have an electrically valid stackup (no errors reported), the stackup editor shows you the characteristic impedance of the trace segments on each signal layer. The impedances appear next to each signal layer in the graphical area that displays the stackup. This assists you in creating or editing a stackup.

However, impedance depends on the width of the segments, so for each signal layer for which you wish to view impedances, you must specify a “test” trace width.

To edit a layer’s test trace width:

1. In the stackup editor, in the graphical area that displays the stackup, click once on the layer whose width you want to set; then click the Edit Selected Layer button.
   OR
   Double-click on the layer that you want to edit.

2. Type a new value in the Test Trace Width edit box. Click OK.

The layer’s characteristic-impedance value in the graphical area updates immediately. If you print the stackup or copy it to the Windows Clipboard, each layer’s characteristic impedance and test-trace width display next to the layer.

If you have multiple trace widths on a single layer, you must enter each trace width separately to see all of the characteristic impedances.

It is important to realize that the Test Trace Width parameter does NOT affect the impedances calculated for any of the transmission lines in your schematic during simulation or any other kind of analysis. The impedances used for simulation are based on the widths of the individual lines in the schematic. *The Test Trace Width only affects the impedance values displayed in the stackup editor’s graphical area.* You can change the Test Trace Width to see what impedances various-width lines have on each layer in your stackup.

Fabrication Compensation

An advanced feature supported by LineSim's stackup-based impedance calculator is the ability to explicitly consider the impedance effects of how your board is fabricated. Specifically, LineSim can compensate for:

- the effects of etched trace widths versus ideal trace widths
- the effects of metal added to outer layers to passivate the traces

HyperLynx recommends that you ***disable*** *the fabrication compensation features unless you know for certain that your PCB fabricator is* ***not*** *taking these factors into consideration.*

Hint: *Most good board houses automatically compensate for trace-etching effects by changing trace widths and dielectric thicknesses. If yours doesn't, a better solution than enabling this compensation in LineSim may be to change board vendors!*

The default setting for fabrication compensation is disabled.

The following sections describe etch and passivation compensation, to help you decide whether you need these features enabled.

What is Trace-Etch Compensation?

When a board layer is etched, the resulting traces actually end up with a trapezoidal cross section. The top of the trace spends the most time in the etching solution and the bottom the least, so the top of the cross section is narrower than the bottom.

Thus, if you tell a board fabricator, "I want an 8-mil-wide trace," the fabricator will secretly say, "OK, but I'll need to start with a somewhat wider trace, knowing that it will etch to an effective 8-mils width." This compensation is normally supplied transparently by the fabricator so that you don't have to worry about the details of how trace etching occurs.

Only if your fabricator is not supplying this compensation for you should you enable LineSim's compensation. If you tell LineSim to compensate for you, the program will reduce the trace widths you specify by an appropriate amount to account for the effects of etching, i.e., LineSim will assume that your trace widths are pre-etch, not post-etch.

Hint: *If fabrication compensation is disabled, enter the final, manufactured widths of your traces. If fabrication compensation is enabled, enter the pre-fabrication widths of your traces.*

What is Passivation Compensation?

Another fabrication process that can affect trace impedance is “passivation.” If copper is exposed for prolonged periods to air, it will oxidize, which is not acceptable on PCBs. To prevent oxidation, the traces on the outer layers of a board are covered with another material, which shields the copper from exposure to air. If this passivation material is conductive, it will increase the traces’ effective widths, affecting their impedances.

Passivation is another effect that a good board house will compensate automatically for, particularly if they are building controlled-impedance boards.

Enabling Fabrication Compensation

Enabling fabrication compensation gives you access to both trace-etch and passivation compensation. For details on controlling each type of compensation individually, see the sections below.

To enable fabrication compensation:

1. From the Options menu, choose Preferences. The Options dialog box opens.
2. In the Circuit Board Fabrication Compensation area (on the “General” tab), click on the Enable Compensation check box. Click OK.

Alternate method for enabling fabrication compensation, when the stackup editor is open:

1. From inside the stackup editor, in the PCB Fabrication Compensation area, click on the Enable Compensation check box.

Only Affects Stackup-Based Transmission Lines

When fabrication compensation is enabled, impedance calculations for all stackup-based transmission-line models are altered (see Chapter 6, section “Modeling a Transmission Line Geometrically with a Stackup” for details on modeling). Impedances for transmission lines modeled with other methods (line-by-line cross sections, cables, etc.) are *not* affected.

Details of Trace-Etch Compensation

LineSim’s trace-etch compensation works by reducing the trace widths you specify for stackup-modeled transmission lines. The amount of the reduction is based on a percentage of a trace’s “clad,” or base, thickness. LineSim uses a different default percentage for outer versus inner layers, and allows you to change the percentages, if needed, to match a particular fabricator’s manufacturing process.

To change the trace-etch-reduction percentages:

1. From the Options menu, choose Preferences. The Options dialog box opens.
2. In the Circuit Board Fabrication Compensation area (on the “General” tab), type new values in the Decrease Width By edit boxes. Click OK.

Details of Passivation Compensation

LineSim’s passivation compensation works on a layer-by-layer basis, and is available only for the outer layers in the current stackup. If you edit a stackup and, for example, move an outer layer to an inner position, passivation compensation no longer applies to that layer.

Note: *Passivation is only performed for a board’s outer layers. The inner layers are automatically passivated by being “buried” in the board.*

Different passivation techniques affect trace impedances in different ways. The following sections describe how to specify a passivation type and briefly describe what each type means.

To specify a passivation type for an outer layer:

1. From the Edit menu, choose Stackup.
   OR
   Click the Edit PCB Stackup button on the toolbar. The stackup editor opens.
2. In the graphical area that displays the stackup, click once on the outer layer for which you want to specify passivation; then click the Edit Selected Layer button.
   OR
   Double-click on the outer layer that you want to edit.
3. Click the Advanced button.
4. Pull down the Copper Passivation combo box, and choose the passivation type that you want. (Remember, the passivation can only be set of fabrication compensation is enabled.)
5. Click OK twice.

Solder Mask Over Bare Copper (SMOBC)

In SMOBC passivation, the board is passivated with solder mask. SMOBC passivation does not affect trace thickness, because the passivation material is non-conductive. Therefore, choosing this style is the same as choosing “None.”

Hot-Air-Leveled Solder

In hot-air-leveled-solder passivation, the board is passivated by being passed through a solder bath. The solder is then leveled (flattened) with a stream of hot air.

Gold/Nickel and Flash Gold/Nickel

Both of these techniques passivate by plating outer-layer traces with a layer of nickel, then a layer of gold. The flash technique differs in that the gold layer is thinner, to save cost.

-Nickel

Tin The board is passivated with tin-nickel plating.

None

This selection says explicitly not to apply any passivation compensation.

Calculating DC Resistance

When you simulate a schematic, LineSim automatically calculates the DC resistance of every transmission line modeled geometrically as a cross-section type, or modeled with a stackup. To calculate DC resistance, LineSim uses the following parameters:

- ♦ trace width, thickness, and length
- ♦ bulk resistivity of trace's metal
- ♦ resistivity temperature coefficient of trace's metal
- ♦ simulation temperature

Trace width, thickness, and length are based on the data you enter when specifying the transmission line. The remaining parameters (resistivity, temperature coefficient, and temperature) you can adjust as described below.

Bulk Resistivity and Temperature Coefficient

In LineSim, each metal layer's bulk resistivity and temperature coefficient default to the values for copper (resistivity = 1.724e-8 ohms/meter, temperature coefficient = 3.93e-3). This means that unless you are using a metal other than copper (e.g., aluminum, for an MCM application), you do not need to change the resistivity or temperature-coefficient parameters.

To change a metal layer's bulk resistivity and temperature coefficient:

1. In the stackup editor, in the graphical area that displays the stackup, click once on the metal layer whose parameters you want to change; then click

the Edit Selected Layer button.
OR
Double-click on the metal layer that you want to edit.

2. Click the Advanced button.

3. Type the new values in the Bulk Resistivity and Temperature Coefficient edit boxes.

4. Click OK twice.

How Resistivity, Temperature Coefficient, and Temperature are Used

The DC resistance of a piece of metal conductor varies with temperature. When LineSim calculates the DC resistance of a trace, it uses the following equation to include the effects of temperature:

$$R = R_b \bullet (1 + T_c \bullet T)$$

where R_b is the bulk resistivity of the trace's metal; T_c the temperature coefficient; and T the temperature at which the simulation is being run.

Simulation Temperature

The temperature for a LineSim simulation defaults to 20 degrees C.

To change the simulation temperature:

1. From the Options menu, choose Preferences.

2. In the Analysis Options area (on the "General" tab), in the Board Temperature edit box, type the new temperature. Click OK.

The board temperature affects only DC resistance calculations. It does not, for example, also affect IC-model parameters. Since DC resistance does not play a large role in most LineSim simulations, you can generally leave the temperature at its default value without sacrificing any significant simulation accuracy. You *should only change the temperature if you know for some reason that it will have a significant effect on your simulation.*

Note: *Board temperature affects not only the simulator (oscilloscope), but also the Terminator Wizard. For this reason, LineSim refers to this feature as an "analysis" (not just "simulation") option.*

Printing Stackups

You can print a stackup from LineSim in order to document it. This is a good way to document a stackup you've designed in LineSim, and now want to apply to an actual PCB.

To print a stackup:

1. From the Edit menu, choose Stackup.
   OR
   Click the Edit PCB Stackup button on the toolbar.

2. In the stackup editor, click the Print button.

3. Check your printer setup. Portrait orientation is best for large stackups. Click OK to begin printing.

LineSim supports color printers; stackups sent to a color printer are output in color.

Setting Up for Printing

You can set up printing-related defaults — e.g., printer choice, paper size, page orientation, etc. — once in LineSim, then have them apply for the remainder of your work session, and for all types of printing (schematics, stackups, oscilloscope results, etc.).

To set up "persistent" printing defaults:

1. From the File menu, choose Print Setup. The Print Setup dialog box opens.

2. Change any parameters you wish, then click OK.

LineSim now remembers the choices you've made, and will continue to use them.

Copying a Stackup to the Clipboard

You can copy your stackup to the Windows Clipboard in order to paste it into other Windows applications. The image sent to the Clipboard is formatted, and includes information such as the name of the .TLN file, the total board thickness, and so forth.

To copy stackups to the Windows Clipboard:

1. From the Edit menu, choose Stackup.
   OR
   Click the Edit PCB Stackup button on the toolbar.

2. In the stackup editor, click the Copy to Clip button.

LineSim writes to the Clipboard in Windows Enhanced Metafile format. The size of the image may vary depending on which application you paste it into; resize as needed (metafiles are vectored and can be sized without damaging image quality).

Stackup Limitations

Multiple Dielectrics between Two Layers

LineSim's stackup editor allows you to add multiple dielectric layers between metal layers. There is no problem if you set both layers to the same dielectric constant.

You can also set the dielectric layers to different dielectric constants. However, if you do, the accuracy with which LineSim calculates characteristic impedances for trace segments will be diminished.

Note: *The air-to-other-dielectric boundary for microstrip lines* ***is*** *handled with normal accuracy. The preceding caveat applies only to internal dielectric-to-*

dielectric boundaries. It also applies only to single, uncoupled traces segments; for coupled segments, if you own the Crosstalk option, LineSim uses its field solver to calculate impedances, and the field solver handles mismatched dielectric constants without any loss of accuracy.

Must Have at Least One Power Plane

LineSim places several restrictions on plane layers in a stackup, among them that there must be at least one plane layer per board. This means that you cannot model double-sided boards (boards with only two signals layers, and no planes).

Chapter 8: Setting Power-Supply Voltages

Summary

This chapter describes:

- which power-supply nets LineSim provides
- how to attach components to power-supply nets
- how to change power-supply-net voltages

Built-In Power-Supply Nets

LineSim gives you access to a list of power-supply nets that you can use when entering schematics. These nets are divided into two groups:

- nets for powering ICs
- nets to which to tie passive components (e.g., a pull-up resistor)

These nets are “built-in” to LineSim, i.e., they are always available for use, automatically. You can attach components to these nets, and change each net’s voltage, if desired.

Nets for Powering ICs

LineSim provides the following nets for powering ICs:

- Vcc
- Vss
- Vt1 - Vt6

Each IC has two power-supply pins; each pin can be tied to any of the IC power-supply voltages listed above (see “Attaching an IC to Power-Supply Nets” below for details).

Nets for Tying Passive Components To

LineSim provides the following nets to which you can tie parallel (i.e., shunt) passive components:

- VpullUp
- VpullDn

Attaching Components to Power-Supply Nets

Attaching an IC to Power-Supply Nets

ICs are attached to power-supply nets in the Assign IC Models dialog box, as part of the process of choosing a model for the IC. You can specify power-supply nets only for driver ICs; all receivers in the schematic then run off of the driver’s supplies.

To attach an IC to its power-supply nets:

See the detailed instructions in Chapter 4, “Setting the Vcc or Vss Pin.”

Attaching Passive Components to Power-Supply Nets

Parallel (i.e., shunt) passive components in the schematic editor are automatically tied to a power-supply net. Specifically:

- a pull-up resistor ties to VpullUp

- a pull-down resistor ties to VpullDn
- a capacitor ties to VpullDn

If both the pull-down resistor and capacitor are activated, the capacitor ties to VpullDn.

You can only tie parallel (i.e., shunt) passive components to the VpullUp and VpullDn voltages.

Changing Power-Supply-Net Voltages

LineSim contains a power-supply editor which allows you to change the voltages on any of the built-in power-supply nets. You can change, for example, Vcc from 5.0V to 3.3V to see the effects of running the ICs on a schematic with a reduced supply voltage.

Opening the Power-Supply Editor:

To open the power-supply editor:

1. From the Edit menu, choose Power Supplies.

Changing a Power-Supply Voltage

To change a power-supply voltage:

1. In the power-supply editor, in the Power Supply Nets list box, highlight the net whose voltage you want to change.
2. Type the new voltage into the New Voltage box.
3. Click OK.

Power-supply voltages can be positive or negative. Changes to supply voltages take effect immediately (i.e., as soon as you simulate).

The top half of the power-supply editor dialog box is disabled in LineSim; it applies only to BoardSim, where supply nets are not built-in.

Chapter 9: Running Simulations

Summary

This chapter describes:

- what steps you must complete before you can simulate
- how to choose oscilloscope probes for viewing your simulation results
- how to set up the oscilloscope
- how to enter an oscilloscope “comment”
- how to run simulations
- how to make time and voltage measurements
- how to print simulation results
- how to copy simulation results to the Windows Clipboard
- how to export the oscilloscope’s data to another program, like Microsoft Excel

Requirements Before Simulating

You can open LineSim’s oscilloscope as soon as you’ve entered a schematic, but before it will actually run a simulation, you must:

- have an electrically valid stackup
- choose a driver IC for the net

If you try to simulate before having a valid stackup or selecting a driver IC, LineSim will give an error. (For details on these topics, see Chapter 7, section “Creating and Editing a Stackup” and Chapter 4, section “Choosing IC Models.”)

Note: *The requirement to have a valid stackup applies even if you have no transmission lines in the schematic which are modeled with the stackup method. LineSim always starts a new schematic with a valid default stackup, so if you have no stackup-based transmission lines and you don’t edit the default stackup, you should never encounter this problem.*

See “Setting Up the Oscilloscope” below in this chapter for details on opening the oscilloscope.

Choosing Scope Probes

LineSim’s oscilloscope probes work like real oscilloscope probes: you place them at various points on the schematic nets you are simulating to see the voltage waveforms at those points. Each probe can be either a “normal” single-ended probe, or a differential probe.

Where Probes Can be Placed

You can place a scope probe on any component pin on the net you are simulating. LineSim supports a maximum of six probes.

Attaching a Probe to a Pin

Probe Types: Single-Ended or Differential

Each oscilloscope probe can be either a “normal” singled-ended probe or a differential probe. Most of the time, you’ll use single-ended probes. If you are using differential-signal technology (e.g., differential PECL, LVDS, etc.), you’ll probably use differential probes. (You can also look at differential signals single-ended.)

Single-ended probes attach to and display one signal at-a-time. Differential probes attach to two signals, and display the difference between the signals attached to the '+' and '-' sides of the probe.

Attaching Single-Ended Probes

To attach a single-ended scope probe to a pin:

1. From the Scope/Sim menu, choose Attach Probes.
2. In the Pins list box, highlight the pin to which you want attach a probe. Then, in the Probe Channels area, click on the channel button (e.g., Ch1) for the probe that you want to attach.
   OR
   Double-click on the pin to which you want attach a probe; this attaches the next available probe.
3. Click OK.

For a description of how pins are named, see "Pin Names" below.

Each probe channel has a distinct color; the voltage for each probe is displayed in the oscilloscope with the channel's color.

In the Attach Oscilloscope Probes dialog box, the double-clicking method of attaching probes is the fastest, provided you do not care to which color each pin is assigned. If you do care (e.g., you prefer to always have the driver IC's voltage in red), use the highlight-and-click-channel-button method. There is a way to assign all six probes with one button click; see the following section for details.

Attaching Differential Probes

To attach a differential scope probe to a pin:

1. From the Scope/Sim menu, choose Attach Probes.
2. In the Probe Channels area, next to the channel you want to be differential, click on the Differential check box. The Channel button changes into a pair of channel buttons, one labeled '+' and one labeled '-'.

3. In the Pins list box, highlight the pin to which you want attach the '+' side of the differential probe. Then, in the Probe Channels area, click on the '+' button (e.g., Ch1+) for the probe that you just converted to differential.
   OR
   Double-click on the pin which you want attach to the '+' channel (this attaches the next available probe channel).

4. Repeat step 3 for the '-' channel button (e.g., Ch1-).

5. Click OK.

Pin Names

Passive Components

Pins on passive components have names of the form:

<component_type>(XY).<pin_number>

where <component_type> is:

- RP for a pull-up resistor
- RD for a pull-down resistor
- C for a capacitor
- RS for a series resistor
- L for a inductor
- BD for a ferrite bead

(XY) is the component's cell label in the schematic,

and <pin_number> is "1" or "2".

Pin Numbers

Terminating-component pin numbers are assigned as follows:

for pull-up and pull-down resistors, and capacitors:

- ♦ pin 1 is the side farthest from the power-supply net
- ♦ pin 2 is the side attached to the power-supply, or nearest to it

for series components:

- ♦ pin 1 is the left side
- ♦ pin 2 is the right side

ICs

Pins on ICs have names of the form:

U(XY)

where (XY) is the IC's cell label in the schematic. Unlike with passive components, there is no explicit pin number (i.e., "1" or "2").

Attaching All Probes Automatically

To quickly attach all six probes:

1. In the Attach Oscilloscope Probes dialog box, click the Attach All button.
2. Click OK.

This method assigns the six probes to the first six pin locations on the currently selected net. Priority is given to ICs, i.e., if there are six or more ICs in the schematic, plus some passive components, the probes will all attach to ICs.

Probing Where There is No Component

If you want to probe at a position in your schematic where there is no component (e.g., between two transmission lines), use the following "trick."

To probe at a cell where there is no component:

1. Modify the schematic by activating an IC at the cell you want to probe. Assuming that the IC has not been activated previously, so that it has an unknown model (i.e., is labeled “????”), leave the model unspecified. ICs with unknown models have no effect on simulation, i.e., act like open circuits.
   If an IC model *has* been previously specified, right-click on the IC and remove its model. (See Chapter 4, section “Removing IC Models” for details.)

2. Follow the steps described above in “Attaching a Probe to a Pin” to add a probe at the new IC.

Hint: *An alternate method is to activate a 10K pull-up or pull-down resistor at the cell. Such a large resistor will have no effect on your circuit (it’s a very small load), but gives you a component pin at which you can probe. Its package parasitics, though, will be present; for very-high-speed signals, you may want to reduce those to minimum values before simulating.*

If No Probes are Attached

If you do not attach any oscilloscope probes, LineSim will automatically assign single-ended probes to the ICs on the net (and associated nets) being simulated. This guarantees that you will see at least one waveform (usually more) when the simulator runs.

If one or more probes have their differential check boxes enabled, but no probes are attached to signals when you begin simulating, the auto-attach feature will turn the differential probes back into single-ended probes, then attach to IC pins.

How Probes Display in the Schematic

When you exit the Attach Oscilloscope Probes dialog box, the schematic editor updates to show the probe locations. Probes are displayed in the editor as colored arrows labeled with the pin to which they are attached.

For a differential probe, two arrows of the same color are displayed, one marked with a black '+' and one marked with a black '-'.

Detaching Probes

Detaching All Probes Simultaneously

To detach all of the oscilloscope probes simultaneously:

1. In the Attach Oscilloscope Probes dialog box, click the Detach All button.
2. Click OK.

This returns all of the oscilloscope probes to being unattached.

Detaching Probes One-at-a-Time

To detach a single oscilloscope probe:

1. In the Attach Oscilloscope Probes dialog box, in the Pins list box, highlight the pin whose probe you want to detach.
2. Click on the probe's channel button (ChX).

This detaches the probe and returns the channel to being unassigned. If you click again, the probe is re-attached.

Changing an Oscilloscope Probe

Changing a Probe to Another Signal

To change an assigned oscilloscope probe to another signal:

1. In the Attach Oscilloscope Probes dialog box, in the Pins list box, highlight the pin to which you want to change the probe.
2. In the Probe Channels area, click on the channel button for the probe that you want to change.

The new pin overwrites the old assignment.

Changing a Probe from Differential to Single-Ended

To change a probe from being differential back to single-ended:

1. In the Attach Oscilloscope Probes dialog box, in the Probe Channels area, click on the differential probe's Differential check box to disable it.

The '-' channel button disappears. The signal formerly attached to the differential probe's '+' channel is now attached to the newly single-ended probe.

Setting Up the Oscilloscope

To open the oscilloscope:

1. From the Scope/Sim menu, choose Run Scope.
   OR
   Click the Open Oscilloscope/Simulator button on the toolbar.

The Digital Oscilloscope dialog box opens.

Before you simulate, there are several oscilloscope parameters to set up. The following sections describe them. All settings are made in the Digital Oscilloscope dialog box.

Choosing the Driver Waveform

Edge versus Oscillator Stimulus

LineSim allows you to simulate with either a single edge (rising or falling) or a repetitive oscillator waveform.

The single edge is often a better choice when you're trying to isolate transmission-line effects, since you can study how a transition settles out without the possibly confusing effects of additional transitions. The oscillator waveform is better for studying the standing-wave effects of repetitive stimulus.

To choose between edge or oscillator stimulus:

1. Click the Edge or Osc radio button in the Driver Waveform area.

Choosing the Edge Direction

If you choose edge stimulus, LineSim lets you display either the rising or the falling edge of the driver's switching transition.

To choose between a rising or falling edge:

1. Click the Rising Edge or Falling Edge radio button in the Driver Waveform area.

If you want to see both edges simultaneously, set the edge direction one way; run the simulator; set the edge the opposite way; and simulate again.

Specifying the Oscillator Frequency and Duty Cycle

If you choose oscillator stimulus, LineSim lets you specify the frequency and duty cycle of the switching waveform.

To specify the frequency:

1. Type a value (in MegaHertz) into the MHz edit box in the Driver Waveform area.

To specify the duty cycle:

1. Type a percentage value into the Duty edit box in the Driver Waveform area.

The duty cycle value defines the percentage of time that the driver spends high.

Setting the Horizontal Scale

To set the horizontal time scale:

1. In the Horizontal area, click on the right or left arrows to increase or decrease the amount of simulation time that appears on the oscilloscope's screen.

The oscilloscope's time-scale increments are logarithmic (1,2,5,10,...), just like a real oscilloscope's.

If you change the time scale after simulating (with a waveform already on the screen), the waveform disappears and you must re-simulate.

Setting the Vertical Scale

To set the vertical voltage scale:

1. In the Vertical Scale area, click on the right or left arrows to increase or decrease the voltage “magnification.”

The oscilloscope’s voltage-scale increments are logarithmic (1,2,5,10,...), just like a real oscilloscope’s.

If you change the voltage scale after simulating (with a waveform already on the screen), the waveform re-displays with the new setting.

Ground Marker

On the oscilloscope screen, the 0.0-V position (ground) is marked in two ways:

- with a green arrow just to the left of the screen
- with a green dashed line across the screen

You can move the ground position with the Vertical Position control (see the following section).

Setting the Vertical Position

To set the vertical voltage position:

1. In the Vertical Position area, use the slider bar to move the oscilloscope’s vertical position up and down.

When you change the vertical position, the green ground arrow and line move up or down.

If you change the vertical position after simulating (with a waveform already on the screen), the waveform re-displays with the new setting.

Settings Readout

For convenience, the horizontal-scale, vertical-scale, and vertical-position values are summarized and displayed in the oscilloscope display, in white text.

You can disable the settings readout to reduce clutter on the oscilloscope screen.

To disable the oscilloscope settings readout:

1. In the Display area, click on the Show Readout check box to disable it.

The readout disappears from the oscilloscope display. You can re-enable it at any later time.

Setting IC Operating Parameters

You can control from the oscilloscope whether the IC models in a simulation run with best-case, typical, or worst-case operating parameters.

What IC Operating Settings Mean

The IC operating settings are actually combinations of the min and max data in an IBIS model, or scaled versions of a .MOD model (which itself contains only typical data). The combinations are named Slow-Weak, Typical, and Fast-Strong, to be as descriptive as possible.

The following table shows for IBIS models how the combinations are defined:

Parameter	for Fast-Strong	for Slow-Weak
driver current	max	min
slew rate	max	min
clamp-diode current	max	min
component capacitance	min	max
package inductance	min	max

Parameter	for Fast-Strong	for Slow-Weak
package capacitance	min	max
package resistance	min	max

(See Chapter 4, section “The IBIS Format” for details on the IBIS modeling format.)

For .MOD models, all of the models in a simulation are scaled up or down from their typical values by globally defined scaling factors to give Slow-Weak or Fast-Strong operation. (See “Scaling .MOD Models for Best/Worst-Case Operation” below for details on the scaling.) Not all parameters in a .MOD model are scaled; the following table shows for .MOD models how the operating combinations are created:

Parameter	for Fast-Strong	for Slow-Weak
driver slew time	scaled down	scaled up
driver output impedance	scaled down	scaled up
driver/receiver I/O capacitance	scaled down	scaled up

Setting the Operating Parameters

To set the IC operating parameters:

1. Click the appropriate radio button in the IC Modeling area.

If you want to see the results of more than one operating point simultaneously, set the parameters one way; run the simulator; set the parameters another way; and simulate again.

Does Not Affect IBIS Models with Only “Typical” Data

Changing IC operating parameters only affects the IBIS IC models in your circuit that actually contain min/max data. If you change the IC operating parameters but see no change in your simulation waveforms, it is probably because the IBIS model(s) you are using do not have min/max data. The IBIS format allows for min/typ/max data, but only requires typical.

Scaling .MOD Models for Best/Worst-Case Operation

By default, LineSim scales the typical device parameters in all of the .MOD models in a simulation up by a factor of 1.8 to create Slow-Weak operation, and down by 0.6 to create Fast-Strong. However, you can adjust these parameters, if you wish, to increase or decrease the “pessimism” of your simulations.

To change the scaling factors used to create best/worst-case .MOD models during simulation:

1. From the Options menu, choose Preferences. Verify that the General tab is selected.
2. In the .MOD IC Model Best/Worst Case Scale Factors area, type the new scaling in the Min Scale Factor and Max Scale Factor edit boxes.
3. Click OK. The new factors apply immediately, and are saved for future LineSim sessions.

Turning Probes On/Off

You can turn your oscilloscope probes off and on directly in the oscilloscope. Turning a probe “off” means it is still attached in the circuit, but does not display its waveform on the oscilloscope screen.

To turn an oscilloscope probe off:

1. In the oscilloscope, in the Probe Enables area, click on the check box for the channel you want to turn off, to disable it.

If a waveform for the channel you turned off was already on the oscilloscope screen, the waveform disappears.

To turn an oscilloscope probe back on:

1. In the Probe Enables area, click on the check box for the channel you want to turn back on.

Probe Enables

In addition to allowing you to turn probes on and off, the Probe Enables area displays to which pin in the current schematic each probe is attached. If a probe is not attached to a pin, the status says “Open.”

Entering an Oscilloscope Comment

The Digital Oscilloscope dialog box includes an area above the oscilloscope screen labeled “Comment.” The Comment box allows you to enter a description or comment that prints when you print your oscilloscope waveforms. (See “Printing a Simulation” below in this chapter for details on printing.)

To enter a printable comment in the oscilloscope:

1. In the Comment box above the oscilloscope screen, click once with the mouse. A cursor appears in the upper left of the box.

2. Type the description or comment.

Attaching Probes from Inside the Oscilloscope

If you want to modify your probe assignments (e.g., add a probe or change which pins are being probed) and currently have the Digital Oscilloscope dialog box open, you can open the Attach Oscilloscope Probes dialog box without closing the oscilloscope.

To attach probes from inside the oscilloscope:

1. Click on the Probes button. The Attach Oscilloscope Probes dialog box opens.

2. Follow the steps listed in “Attaching a Probe to a Pin” above in this chapter.

Running a Simulation

To run a simulation:

1. In the Digital Oscilloscope dialog box, click the Start Simulation button.

The Simulation Status dialog box opens, and simulation begins.

Pre-Transient Steps

Before anything is displayed on the oscilloscope screen, LineSim:

1. builds a simulation model
2. optimizes the model for faster performance
3. performs a DC simulation

Transient Steps

When the pre-transient steps are completed, LineSim begins its transient simulation. The results of the transient simulation are displayed in the oscilloscope as they are calculated.

LineSim's transient simulations are constrained by the time resolution the program must use for the shortest transmission line in the circuit. LineSim uses intelligent algorithms (the pre-transient “optimizing passes”) to minimize simulation time when possible.

Percent Done and Status Messages

The Simulation Status box displays the percentage of the transient simulation that has completed.

The dialog box also displays messages from LineSim's simulator. The messages tell you what steps are currently being run.

Stopping a Simulation

To stop a simulation before it has completed:

1. In the Simulation Status dialog box, click the Stop button.

Timing and Voltage Measurements

Sometimes it is necessary to make detailed, accurate time, voltage, or slew-rate measurements from the oscilloscope display (e.g., to measure flight times or over/undershoot). The oscilloscope provides a pair of measurement crosshairs for this purpose.

To measure a single voltage and/or time:

1. Position the mouse in the oscilloscope display exactly where you want to make a measurement.
2. Click the left mouse button. A yellow crosshair appears in the display.

Look in the Cursors area (below the display). An accurate voltage and time readout for the crosshair appears.

To measure a second point or measure a delta voltage, delta time, or slew rate:

1. Click again at the second point in the oscilloscope display. A second yellow crosshair appears.

In the Cursors area, a second readout, “delta readout,” and slew-rate readout (“slope”) all appear; the delta readout shows the voltage and time difference between the crosshairs, and the slew rate gives the slope (in V/ns) between the two cursor points.

To turn the crosshairs back off:

1. Click a third time in the oscilloscope display. The crosshairs disappear.

“Live” Cursor Readout

The Cursors area in the oscilloscope also provides a “live” (i.e., constantly updated) display of the position of the mouse cursor whenever it is inside the oscilloscope screen. The position reads in V and ns. Therefore (as an alternative to the method described above), you can also make quick measurements simply by moving the mouse to the point on a waveform at which you want to measure, holding the mouse steady, and looking at the Cursor field.

Printing a Simulation

You can print your simulation results in order to document them.

To print simulation waveforms:

1. In the Digital Oscilloscope dialog box, click the Print button.
2. In the Print dialog box, check your printer setup. Click OK to begin printing.

LineSim supports color printers; simulation results sent to a color printer are output with colored waveforms.

Setting Up for Printing

You can set up printing-related defaults — e.g., printer choice, paper size, page orientation, etc. — once in LineSim, then have them apply for the remainder of your work session, and for all types of printing (schematics, stackups, oscilloscope results, etc.).

To set up “persistent” printing defaults:

1. From the File menu, choose Print Setup. The Print Setup dialog box opens.
2. Change any parameters you wish, then click OK.

LineSim now remembers the choices you’ve made, and will continue to use them.

Copying a Simulation to the Clipboard

You can copy your simulation results to the Windows Clipboard in order to paste them into other Windows applications. The image sent to the Clipboard is formatted, and includes information such as the name of the .TLN file, the oscilloscope settings, and a time and date stamp.

To copy simulation waveforms to the Windows Clipboard:

1. In the Digital Oscilloscope dialog box, click the Copy to Clip button.

LineSim writes to the Clipboard in Windows Enhanced Metafile format. The size of the image may vary depending on which application you paste it into; resize as needed (metafiles are vectored and can be sized without damaging image quality).

Exporting Simulation Data to Another Application (.CSV File)

Occasionally, you may want to export the voltage-versus-time simulation data displayed in the oscilloscope (and the associated currents-versus-time) to another application, like Excel or Mathcad. To facilitate this need, LineSim's oscilloscope can output a comma-separated-values (.CSV) file which includes all of the data displayed in the oscilloscope window for the present simulation. .CSV files can be read directly by Excel (and most other spreadsheet programs), and are easily read by an input routine in mathematical programs like Mathcad.

To write a .CSV file representing the results of a simulation:

1. In LineSim, open the Digital Oscilloscope dialog box and run the simulation whose results you want in a .CSV file.
2. Click the Save As CSV button. The Save Oscilloscope Output dialog box opens.
3. Choose a directory and name for the .CSV file, then click the Save button.

The .CSV file is created. LineSim displays a dialog box reminding you of where and with what name the file was saved.

Format of the .CSV File

The top of the .CSV file records the .HYP-file name, creation date, and other similar data; and also lists to which IC pins the oscilloscope probes were attached when the file was generated. Then, below, arranged in columns, is the time and voltage data for the present oscilloscope simulation, and then in additional columns, the currents corresponding to each voltage.

Important! There is a difference in how voltages and currents are measured in the .CSV file. Voltages are measured *outside* of a device's package (IC or passive component), but currents are measured *inside*. Thus, an oscilloscope probe that appears in the schematic to be outside of a device is really located inside the device's package *for current measurements* (but not voltage).

The .CSV file will open directly in programs like Microsoft Excel (in Excel, use choose File/Open or double-click in the Windows Explorer on the .CSV file). If you are reading the file with a mathematics-package program and do not want the header information at the top (creation date, etc.), you can remove it using any text editor.

The time data in the file is in seconds; voltages are in volts; currents are in amps.

Re-Simulating; Comparing Results

You can simulate a given net or schematic multiple times to see the effects of opposite driver edges, different IC models, IC operating parameters, and so forth. Sometimes, you want to display the waveforms one on top of the other, to make comparisons between the results; other times, you want to display each result individually.

Plotting One Simulation At-A-Time

By default, if you run a series of simulations without closing the Digital Oscilloscope dialog box, the results of each new simulation supersede the previous simulation's results. For example, if you run with the IC operating parameters set to Slow-Weak, you get one set of waveforms. If you then change operating parameters to Fast-Strong and re-simulate, the Slow-Weak waveforms disappear and the Fast-Strong set replaces them.

Displaying the Previous Plot

For purposes of comparison, you can view the results of both the current simulation *and* the previous simulation *simultaneously*.

To display the results of the next-to-the-last simulation:

1. In the Digital Oscilloscope dialog box, in the Display area, click on the Show Previous check box.

As long as the Show Previous box is checked, the oscilloscope will always show results for both the last and the next-to-the-last simulations that were run, even if the oscilloscope is closed and then re-opened.

Saving and Restoring *Any* Plot

You can also save an oscilloscope plot for restoration at *any* time later in your LineSim session. This allows you to make comparisons with any simulation you wish, not just the previous simulation as with the Show Previous feature (see above). The saving occurs to a buffer in memory, not to the hard disk, so you can restore the plot only during the same LineSim session in which you saved it.

To save the results of an oscilloscope plot (to memory):

1. In the Digital Oscilloscope dialog box, click the Copy to Buffer button. The waveform is stored in memory.

To restore a saved plot:

1. In the Digital Oscilloscope dialog box, in the Display area, click on the Show Buffer check box.

You can have both the Show Previous and Show Buffer check boxes enabled, resulting in the waveforms from a saved, the previous, and the current plots all being displayed simultaneously.

Does Not Apply if Oscilloscope Time Scale is Changed

If the oscilloscope's horizontal time scale is changed, any previous simulation results are erased, because there may no longer be sufficient data to display them.

Erasing a Simulation

You can force the results of the current and previous simulations to be erased from the oscilloscope's screen.

To erase the oscilloscope screen:

1. In the Digital Oscilloscope dialog box, click the Erase button.

This causes the current and previous simulations' results to disappear. Results saved in the oscilloscope's buffer (see "Saving and Restoring Any Plot" above for details) are not erased; to make saved waveforms disappear, disable the Show Buffer check box.

Displaying Three Consecutive Simulations

When you are running with IBIS IC models that contain min/typ/max data, you often want to run and compare three consecutive simulations: with the IC operating parameters set to Slow-Weak, then Typical, then Fast-Strong.

To display the results of three consecutive simulations:

1. Click on the Show Previous and Show Buffer check boxes.
2. Run the first simulation, then click the Copy to Buffer button.
3. Run the second simulation.
4. Run the third simulation.

All three sets of waveforms are displayed together.

Simulation Options

In the Options dialog box (choose Options / Preferences), on the Advanced tab, there are several advanced options that affect LineSim's simulation engine. *HyperLynx recommends that LineSim users never change these options.* These are advanced options intended primarily for certain rare situations in the BoardSim product.

Chapter 10: Terminator Wizard

Summary

One of your key weapons in fighting poor signal quality and EMC problems (and sometimes even crosstalk) is *terminators*. Among LineSim's major benefits is helping you to find proper termination schemes for the "problem" nets in your schematics. To this end, LineSim offers a special feature called *the Terminator Wizard*, a "smart" tool which automatically recommends optimal terminating-component values. This chapter describes the Terminator Wizard.

The Terminator Wizard

LineSim includes a "smart" Terminator Wizard to help you find optimal values for the terminating components in your schematics. "Terminating components" include resistors and capacitors, both series and parallel; "optimal values" means resistances and capacitances that give the best waveforms from a signal-integrity and EMC standpoint. The Terminator Wizard works with nets that have single terminators or multiple.

Running the Terminator Wizard

The Terminator Wizard is available to run, any time you choose, on the current schematic. The Wizard works differently depending on whether the schematic net it's running on is terminated or not; and whether there is one or multiple nets in the schematic.

Terminated versus Unterminated Nets

If the net is terminated already, the Wizard bases its analysis on the terminating components present on the net, suggesting, if possible, optimal component values. This feature works on any net with a single termination type (e.g., AC or series) , and with a number of useful topologies involving multiple terminators (see "Running the Terminator Wizard on Nets with Multiple Terminators" below for details).

If the net is not terminated, and the Wizard thinks the net is too long to be unterminated, the Wizard will attempt to suggest a termination strategy — both a type of termination and optimal component value(s).

What is Meant by "Multiple Nets"

Nets in a LineSim schematic are not explicitly named. However, internally, as you draw a schematic, LineSim keeps track of whether there is one or multiple nets in your drawing. There are two conditions that cause multiple nets to exist:

- ♦ you've drawn two or more completely independent circuits; each is completely isolated conductively from the other(s)
- ♦ you've drawn one circuit, but it contains one or more series components that cause the circuit to be divided into multiple nets

Multiple versus Single Nets

If the schematic contains multiple nets, then before presenting its analysis the Wizard opens a dialog box asking which net to analyze. You choose the net by selecting an IC pin that's on the net; see "Running the Wizard" below for details.

If the schematic contains only one net, then the Wizard's analysis is immediate (no need to choose an IC pin).

Running the Wizard

To run the Terminator Wizard:

1. Verify that a driver model has been chosen for the net you want analyzed. (See "Must Have a Driver Model Selected" below in this chapter for details.)

2. Then, from the Wizards menu, choose Terminator Wizard.
   OR
   Click the Open Terminator Wizard button on the toolbar.

3. *Only if there are multiple nets in the schematic* (otherwise, skip to step 4; see “What is Meant by ‘Multiple Nets’” above for a description of how multiple nets occur), the Select Net for Terminator Wizard dialog box opens. In the Select a Device Pin list box, double-click on an IC pin attached to the net you want to analyze. The dialog box closes.

4. The Terminator Wizard dialog box opens.

The Wizard runs automatically, as soon as its dialog box opens, and for the selected net and any nets attached directly to it through series components (or a set of differential pins on a driving IC), displays a list of results in the Terminator Analysis area. The results include information about:

♦ what components were found on the net(s)

♦ the electrical characteristics of the net’s driver IC (driving impedance and slew time)

♦ the physical and electrical characteristics of the net(s) (total length, nominal impedance, and “effective” impedance including IC-loading effects)

If the net has any terminating components, the Wizard also displays:

♦ what kind of termination it found (e.g., series or parallel AC)

♦ recommendations for optimal termination values

♦ information about the physical location of the terminating components

♦ warnings about any component values that seem problematic

♦ warnings about any component locations that seem problematic

If the net does *not* have any terminating components, and the Wizard believes the net is too long to be unterminated, the Wizard displays:

- what kind of termination it recommends
- recommendations for optimal termination values

The component-value recommendations and the recommended “best” termination type (at the bottom of the Terminator Analysis area) are the Wizard’s most-important benefit. See section “Terminator Wizard Results: Optimal Component Values and Recommended Terminators” below for more details. The Wizard’s warnings about improper component values and location are also very useful (see “Terminator Wizard Results: Signal-Integrity Checks/Warnings” below).

At the bottom of the dialog box, in the Messages area, the Wizard displays warning/error messages and hints about improving the net’s signal quality. See the following sections for details on what kinds of messages may appear.

Recognizing Terminator Types

If there is a driver IC present on the currently selected net (see “Must Have a Driver Model Selected” below for details), the Terminator Wizard performs an analysis to determine how the net is terminated, and to find the optimal terminating-component values.

If there are terminating components present on the net, the analysis can succeed only if the Wizard is able to automatically determine from the components what type of termination you are using (e.g., series resistor or AC parallel). To determine the termination type, the Wizard examines:

- what resistor and capacitor components are present
- what nets are connected to each component (e.g., two signal nets or a signal net and an AC ground?)
- where the components are located physically, especially relative to the driver IC
- other topological details of the net’s routing

If the termination type can be identified, the result is displayed in the Terminator Analysis area, in the Termination Type field. If the type cannot be identified, the Termination Type is left as “unknown” (red question mark) and a warning appears in the Messages area; the Wizard then cannot make a recommendation on component values.

Must Have a Driver Model Selected for Complete Analysis

In order for the Terminator Wizard to run a complete analysis and recommend component values, you must have a model selected for the driver IC on the current net (either the single net in the schematic or the one of multiple nets that you chose to simulate). The presence of a driver model is critical because many of the driver’s properties have a profound effect on terminating-component values. Important driver properties include:

- ♦ slew time
- ♦ output impedance
- ♦ physical position on the net

If you run the Wizard without a driver model selected, LineSim gives a warning in the Messages area; even if a termination is present on the net, the Wizard lists the Termination Type as “unknown” (with a red question mark). Some of the statistics about the net are displayed, but no recommendation is made for terminating-component values.

Supported Termination Types and Net Topologies

Important: *Each revision of LineSim adds additional capabilities to the Terminator Wizard, so termination schemes that were not recognized in an earlier version of the program may be in the current one. The following sections describe the Wizard’s features at the time of this writing.*

Types of Terminators and Topologies Recognized by the Wizard

If the Terminator Wizard finds terminating components already on the net being analyzed, it attempts to identify the termination type and determine optimum values for the components. The following table lists the termination types currently recognized by the Terminator Wizard. See the sections following the table for additional details.

Termination/Topology Type	**Comments**
single series R single DC parallel R (pull-up or pull-down) single DC parallel pull-up/pull-down combination single AC parallel R+C	
multiple series Rs, each terminating one branch of a “star” route	“Star” route means a topology in which an IC drives multiple trace branches in parallel, with the branch point close to the driver
multiple parallel terminators, of any mixture of types	“Any mixture of types” means any mix of DC parallel R, DC parallel pull-up/pull-down combo, and AC parallel R+C
single series R + multiple parallel terminators of the same type	The Wizard cannot recognize series R + multiple parallel terminators of *differing* types (e.g., one pull-up R + one AC terminator)
differential trace-to-trace R, if the two traces are driven by an IBIS differential IC model	If the traces in a differential pair are driven by .MOD, .PML, or an IBIS non-differential model, then the Wizard cannot recognize the termination; also, values are recommended in LineSim Crosstalk only — see “Differential Line-to-Line Termination” below

Differential Line-to-Line Termination (LineSim Crosstalk Only)

A line-to-line differential resistor is recognized by the Terminator Wizard. Generally, the Wizard does not support nets (or groups of nets) with multiple drivers present. However, since differential pairs require two drivers for proper circuit operation, an exception is made for them. But unless the IC model used is IBIS and specifically identifies the driving pins as differentially paired, the Wizard has no way of knowing that a differential situation is present, and so won't recognize the termination.

Further, in order to predict an optimal value for such a terminator, the Wizard needs access to LineSim Crosstalk's field solver. Accordingly, recommendations for differential-terminator values are available only if you are licensed for LineSim Crosstalk.

Note: *.MOD and .PML models do not support the concept of differential pin pairs. IBIS models do; the Terminator Wizard requires their use in order to identify differential pairs.*

Some Combinations of Multiple Terminators Not Supported

The Wizard also does not support some complex termination schemes based on multiple terminators. See the table above; any combinations not specifically described in the table are probably not supported.

In situations where the Wizard cannot recognize a complex termination type, use interactive simulation instead of the Wizard to choose optimal component values.

Multiple Drivers Not Supported, Except for Differential IBIS Models

Except for the case of a trace pair driven by an IBIS differential IC model, the Wizard will also not analyze any net that has more than one driver actively selected. In order to analyze such a net (provided it is not differential), change all but one of the drivers into a receiver (or remove the other driver models entirely).

Ferrite Beads Not Supported

The Wizard does not support ferrite-bead terminators. Use interactive simulation to find an optimal ferrite bead; see Chapter 4, section “Choosing Ferrite-Bead Models” for details.

How the Wizard Recognizes “Branched” Topologies

The Wizard supports termination schemes in which a “star-routed” net has each of its branches terminated by a separate series resistor. However, in order to recognize such a topology and make useful component-value recommendations for it, the Wizard must be able to automatically judge whether a given net is actually routed as a valid star.

To make a topological judgment about star routing, the Wizard uses a path-tracing algorithm. If a net has multiple series resistors, it is considered to be a valid star route only if one end of each resistor traces back only to the driver IC, and the other end traces only to receiver ICs.

Terminator Wizard Results: Optimal Component Values and Recommended Terminators

For a terminated net, the optimal component values are often not trivial to find, particularly since the loading effect of IC capacitances effectively alters a net’s characteristic impedance. The Terminator Wizard accounts for all of the IC models currently on the net and its associated nets, factoring the models’ capacitances into an effective characteristic-impedance calculation.

Effective Z0 Value

One of the net-statistic values that the Wizard displays is Effective Z0. The “effective Z0” is a figure that attempts to show by how much the selected net’s actual characteristic impedance is effectively lowered by the presence of IC capacitance along the net and associated nets. This value can be used as a guide when choosing termination resistances, since for nets that are significantly loaded by IC capacitance, the proper termination value is almost always lower than suggested by the net’s actual Z0.

Results for Nets With Single Terminators

For nets with a single terminator already in place, the Wizard attempts to identify the termination type; if the identification succeeds, then the Wizard calculates an optimal value for the terminating component(s).

Recommended Terminating-Component Values

The Terminator Wizard displays its recommended resistor and/or capacitor values at the bottom of the Terminator Analysis area. If the currently selected net uses series or DC parallel termination, only one or more resistor values are recommended; if it uses AC parallel termination, resistor *and* capacitor values are suggested.

If there are multiple resistors or capacitors on a net, then the Wizard's recommended values are identified per-component in the following manner:

```
<component_type> <reference_designator.pin>  suggested:
<value>
```

where <component_type> is the type of component ("resistor" or "capacitor"); <reference_designator.pin> specifies the component's reference designator and a pin on the component; and <value> is the recommended value.

If you make changes to the net being analyzed — for example, change any of its IC models or alter the schematic's stackup — re-run the Wizard to see how the recommended termination values may have changed in response. The series-resistor value, for example, is strongly dependent on your current choice of driver IC.

Applying Recommended Termination Values

If the Terminator Wizard recommends resistor and/or capacitor values for a terminated net, you can easily apply the recommended values to the components in the schematic, then re-simulate to see the resulting waveforms.

To apply component values recommended by the Terminator Wizard to the components in the schematic:

1. Run the Terminator Wizard (see "Running the Wizard" above in this chapter for details). Verify that the Wizard is able to identify the termination type, and recommends component values.

2. Click the Apply Values button.

The recommended component values are exported to the components in the schematic.

To re-simulate using the recommended values:

1. Close the Terminator Wizard by clicking OK.
2. Open the oscilloscope, and re-simulate.

Results for Nets with Multiple Terminators

If a net has multiple terminators, the Wizard behaves similarly to how it does when only a single terminator is present (see section “Results for Nets With Single Terminators” above). However, there is now an additional challenge: the Wizard must also determine which of the terminating strategies it thinks is best, and give you a way to get recommended values for whichever type *you* prefer to use.

Choosing Between Multiple Terminators

When you run the Wizard on a net with multiple terminators (see “Running the Wizard” for details on running), the Wizard first examines the net to see if the multiple-component configuration is one that it can identify and “understand.” (See section “Supported Termination Types and Net Topologies” above for a description of what multiple-terminator combinations are supported.)

If the terminating scheme is successfully identified, the Wizard displays the combination of components in the Termination Type field, in the Terminator Analysis area (e.g., “series, AC, pull-up”). If not, the Wizard marks the Termination Type with a red question mark, and cannot proceed with analysis.

Assuming the Type has been correctly identified, the Wizard then displays in the Preferred Choice area (on the right side of the dialog box) a set of radio buttons that normally (with single-terminator or unterminated nets) is not displayed. The buttons offer several choices for which terminator type to recommend values for: “Best” (meaning let the Wizard choose what it thinks is the most-optimal of the terminator types it found on the net), and two or more selections that specify exactly which terminator type to use.

For example, if a net has three terminators in its layout, series, AC parallel, and DC pull-up, the Wizard will:

- ♦ identify the net as having terminator type "Series, AC, pull-up"
- ♦ present radio buttons in the Preferred Choice box for:
  - ♦ Best (= let the Wizard recommend the best termination type to use)
  - ♦ Series Termination (= force analysis of the series terminator)
  - ♦ AC Termination (= force analysis of the AC terminator)
  - ♦ DC Termination (= force analysis of the DC pull-up terminator)

To tell the Terminator Wizard on which type of terminator you want analysis:

1. In the Preferred Choice area, click the appropriate radio button.

After you make your choice, the Wizard immediately shows its recommended value for that termination type. If you choose "Best," the Wizard will make a recommendation for the terminator type it thinks will best suit your net; the Wizard's choice is listed in the Terminator Analysis area as the "Suggested Termination."

If there are multiple nets in your schematic and you select a new net for analysis, or if you close the schematic and then re-load it, the Wizard does not save your choice of preferred terminator type. If you return to the net to analyze it again, you must re-choose your preferred type.

Simulating with a Particular Terminator

To simulate with a particular terminator that you chose (or the Wizard recommended) during analysis:

1. In the Termination Suggestions area, click the Apply Values button.

The recommended termination values are automatically placed in your circuit. You can now close the Wizard and open the oscilloscope or spectrum analyzer to simulate and see an actual waveform.

How "Unused" Termination Components are Treated

When you set the component values of your preferred type of terminator using the Apply Values buttons, the Wizard must also set the values for the “unused” terminating components in such a way that they do not interfere with the simulation of the preferred terminator. This is accomplished as follows:

- unused series resistors are set to 0.0 ohms
- in an unused AC terminator, the resistor value is set to 1 Mohm (the capacitor is unchanged)
- in an unused DC pull-up/down terminator, the resistor value is set to 1 Mohm

Results for Nets With No Terminators

If you run the Terminator Wizard on a net that has no terminating components, the Wizard first runs an analysis to determine if the net’s signal integrity is likely to be acceptable without termination.

If the Wizard believes that the net does not need termination, then in the Terminator Analysis area, the Termination Type is set to “No termination found”; no termination is suggested; and Apply Values button is grayed out.

On the other hand, if the Wizard concludes that the net is too long to be left unterminated, it will attempt to recommend a termination type, and optimal values for the terminator’s components. The algorithms used to determine the optimal terminator type are complex; they take into account the positions of driver and receiver ICs along the net, the topology of the net’s routing (e.g., daisy-chained versus star-routed), comparison of driver versus net impedance, etc. Part of the process of recommending a terminator is to also recommend its position on the net, since location is often just as important as component values.

For nets with complex routing schemes (e.g., complicated, “non-obvious” branching), the Wizard can sometimes not find an optimal termination scheme. (You may not be able to either.) Generally, The Terminator Wizard works best on nets that are single-receiver, or daisy-chained, or cleanly star-routed (“cleanly” meaning “with clearly identifiable branches”).

Applying Recommended Terminators

The Terminator Wizard cannot automatically modify a schematic to implement a terminator recommended for an unterminated net. Instead, you need to add the recommended components yourself, manually.

To try a terminator recommended by the Wizard for an unterminated net:

1. Note the terminator recommended by the Wizard. Then click OK to close the Wizard dialog box.
2. In the schematic, add the component(s) recommended by the Wizard.
3. Re-open the Terminator Wizard.
4. Click the Apply Values button. The recommended value(s) are written into the component(s) you just added.
5. Click OK to close the Wizard. Proceed with simulation.

Using Standard Component Tolerances for Recommended Values

By default, when the Terminator Wizard calculates recommended terminating-component values, it displays them exactly, without regard for the values you could actually purchase and install on a board, i.e., without considering the components' standard values.

However, you can tell the Wizard to use only standard values, and specify values of which tolerance.

To tell the Wizard to recommend standard component values, for a particular standard tolerance:

1. Run the Terminator Wizard (see "Running the Wizard" above in this chapter for details).
2. In the Terminator Suggestions area, pull down the Apply Tolerance combo box and choose the desired tolerance.

The Terminator Analysis area, if it is currently recommending any component values, updates immediately to display the closest values in the specified tolerance's sequence. If you click the Apply Values button, the components are updated using the standard values.

Terminator Wizard Results: Signal-Integrity Checks/Warnings

In addition to recommending component values and types, the Terminator Wizard automatically runs various signal-integrity checks against the net being analyzed. If a violation is found, it is reported in a red font in the Terminator Analysis area, usually in the messages sub-area.

The checks fall into two broad categories: searching for problematic component values (e.g., resistors that are too large or small), and searching for problematic component location (e.g., a series resistor located too far from the driver it terminates).

The following table lists the signal-integrity checks currently run by the Terminator Wizard. The following sections provide additional details.

Type of Check	**Description**
unterminated-net length	for unterminated nets, checks if the net's length is too long to have no termination; suggests a solution, if possible
component value non-optimal	if a terminating component's value is more than 25% different from the value the Wizard thinks is optimal, the Wizard issues a warning; this makes it easy to find components that probably need "fixing"
driver-to-series-resistor length	for series-resistor terminators, checks if the distance from driver to resistor is too long for effective termination
AC-terminator resistor-to-capacitor length	for AC parallel terminators, checks if the distance between resistor and capacitor is too long for effective termination

Type of Check	**Description**
pull-up/pull-down combo resistor-to-resistor length	for DC parallel pull-up/pull-down combo terminators, checks if the distance between the two resistors is too long for effective termination
receiver-IC stub length	for each receiver IC, checks if its stub length (i.e., distance from the “main” trace routing) is too long
resistor placement relative to receiver ICs	for any non-series type of terminating resistance, checks for improper placement relative to receiver ICs on the net; e.g., will flag a DC parallel resistor that is located too far from a receiver IC
driver impedance exceeding net’s impedance	for each driver, issues a warning if the driver’s impedance exceeds the net’s impedance, i.e., if the driver intrinsically over-terminates the net
driver impedance large enough to cause bad DC levels or excessive tolerance variation in the driver itself	for each driver, issues a warning if the driver impedance is large enough to cause any of the following problems: — likelihood of invalid DC levels (when DC termination used) — likelihood of an excessive portion of series termination residing in the driver itself and therefore subject to excessive tolerance variations

If the Terminator Wizard issues a warning based on one or more of its signal-integrity checks, it does not necessarily mean that your termination won’t work. It may help explain, however, why the waveform you see in the oscilloscope is less than perfect, even if you are using the component values recommended by the Terminator Wizard. The Wizard cannot compensate, for example, for improper component placement (e.g., a series resistor located too far from the driver IC) or poor routing topology.

About Driver-IC Impedance

The Terminator Wizard runs several types of signal-integrity checks having to do with the impedance of the driver IC relative to the characteristic impedance of the net being driven. This section describes why.

One problem that can occur if a driver IC has a higher impedance than the driven net is *over-termination*. Over-termination means that the driver IC by itself has more than enough impedance to series terminate the net it's driving. This implies first of all that there's no point in adding more series resistance to the driver externally; and second, that the driver may be delivering too small a step into the net to make series termination even work. In cases such as this, it may be necessary to use a different driver with lower impedance, or to increase the impedance of the driven net.

Note: *To quickly see what a driver's impedance is, open the Terminator Wizard and look in the Terminator Analysis area. Driver impedance is one of the listed statistics.*

Another set of problems may occur when the driver impedance is lower than net's impedance, but still greater than about 20% of the net's Z_0. Now, series termination is possible; however, a significant portion (20% or more) of the terminating impedance is present in the driver IC itself, and this portion impedance has a wide range of values that depends on the IC's manufacturing tolerance. Such inexactness in the overall series-resistance value may make it difficult to reliably terminate the net.

Also, if the net is DC parallel terminated, the relatively large driver impedance will cause the DC levels on the nets to be shifted noticeably away from the normal, unloaded levels. This may cause threshold-crossing problems.

Driver Models vs. Default Slew Rate

When determining if there are signal-integrity violations for a particular net, the Terminator Wizard examines the slew time of the net's driver-IC model. If there is no driver model specified for the net, the Wizard uses a default rise/fall time for its analysis.

Note: *In many cases, lack of a known driver–IC position will prevent detailed analysis. So even though the Wizard can "fall back" on the default slew-time* *value, it may not be able to provide much meaningful information if it doesn't have a specific driver-IC model to look at.*

To set the default driver rise/fall time:

1. From inside the Terminator Wizard dialog box, click the Preferences button. The Options dialog box opens.
2. Click the BoardSim tab.
3. In the Board Wizard Defaults are, in Default Rise/Fall edit box, type the desired value (in nanoseconds).
4. Click OK.

You can also set the default time from outside the Terminator Wizard dialog box: from the Options menu, choose Preferences; then follow the steps above.

No Placement Checks for Differential Terminators

The current version of the Terminator Wizard does not check for proper positioning of a differential terminator, e.g., whether a line-to-line terminator has excessive stub length or is otherwise mis-placed. Use interactive simulation to gauge the effectiveness of a differential terminator's location.

Terminator Wizard Results: Pin-to-Pin Physical Lengths

One additional feature of the Terminator Wizard is a list of the physical lengths between component pins. This data can be useful if you're trying to debug Wizard warnings about improper component location, or generally as a way of seeing exactly how far various component pins are from each other.

The pin-to-pin length distance is displayed at the bottom of the Terminator Analysis area (scroll down to see it, if needed).

For transmission lines in a schematic which have no specified physical length (e.g., the "simple" type, for which you provide only an impedance and delay), LineSim assumes (for reporting purposes) a propagation delay of 0.5*c*, i.e., half the speed of light. Then the physical length is found as 0.5*c**delay.

Chapter 11: The HyperLynx File Editor

Summary

This chapter describes:

- why LineSim includes a file editor
- how to open the HyperLynx File Editor
- how to use the Editor

Why a HyperLynx File Editor?

LineSim includes a Windows file editor — the "HyperLynx File Editor" — which you can use for editing various files associated with LineSim, e.g., USER.FBD (the user-generated ferrite-bead library) or .SLM files (single-line model files, for modeling connectors).

There is no requirement to use the HyperLynx File Editor; you can view and edit files with any Windows editor you choose. However, the HyperLynx File Editor has some advantages over generic editors that are worth considering.

First, the HyperLynx File Editor is a text editor which not limited in the file sizes it can handle. Second, the HyperLynx File Editor is convenient to use when you're running LineSim because you can launch it from inside LineSim.

Note: *The HyperLynx File Editor was created mostly with BoardSim users in mind, because BoardSim creates certain files (e.g., the Board Wizard's report* *file) that are sometimes very large and therefore difficult to view with ordinary editors.*

Opening The HyperLynx File Editor

To open the HyperLynx File Editor:

1. From the File menu, choose File Editor.
   OR
   Click the File Editor button on the toolbar.

The Editor opens, ready to load a file or begin creating a new one.

Editor Functions as a "Child" of LineSim

The HyperLynx File Editor, because it is launched from inside LineSim, is a "child" application of LineSim, not a completely independent Windows application.

This has two implications:

if you iconize the Editor, its icon "belongs" to LineSim, not to Windows; this means, for example, that the Editor's icon does not appear on the Windows tool bar

if you have the Editor open and close LineSim, the Editor is closed, too

Opening a File

When the HyperLynx File Editor first opens, you can begin typing immediately to create a new file. Or:

To open an existing file in the Editor:

1. From the File menu, choose Open.
   OR
   Click the Open button on the toolbar.

2. Change directories, if needed, and highlight the file you want to open.
3. Click Open.

The File opens in the Editor, ready for editing.

Setting Read-Only Mode

You can set the HyperLynx File Editor to read-only mode, so that it functions as a viewer, but not an editor. This prevents you from accidentally editing a file when all you really want to do is look at it.

To enable read-only mode:

1. From the Options menu, choose Read Only. The check mark appears to indicate that the editor is functioning in read-only mode.

Cutting, Copying, Pasting, and Deleting Text

The HyperLynx File Editor supports cutting, copying, pasting, and deleting of text, from the menus, from the toolbar, or using the standard Windows accelerator keys.

To cut text:

1. Highlight the text you want to cut.
2. Choose Cut from the Edit menu, or click the Cut button on the toolbar, or type Ctrl-X.
   The text disappears, but is saved on the Clipboard.

To copy text:

1. Highlight the text you want to copy.
2. Choose Copy from the Edit menu, or click the Copy button on the toolbar, or type Ctrl-C.
   The text is copied to the Clipboard.

To paste text:

1. Position the cursor where you want the text to be pasted.

2. Choose Paste from the Edit menu, or click the Paste button on the toolbar, or type Ctrl-V.
   The pasted text is inserted at the cursor location.

To delete text:

1. Highlight the text you want to delete.

2. Choose Delete from the Edit menu, or press the delete key.
   The text disappears.

You can also delete text by clicking in the text to position the cursor, then typing with the Backspace key or Delete key.

Undoing an Action

The HyperLynx File Editor provides a single-level “undo” feature.

To undo the previous editing action:

1. From the Edit menu, choose Undo.
   OR
   Type Ctrl-Z.

The last editing action you performed is undone.

Going to a Line Number

To go to a line number:

1. From the Search menu, choose Go To Line.
   OR
   Click the Go To button on the toolbar.
   A dialog box opens.

2. Type the line number you want to go to.

3. Click OK.

The editor jumps to the specified line number, with the matching line appearing at the *top* of the window.

Finding Text

To find text:

1. From the Search menu, choose Find.
   OR
   Click the Find button on the toolbar.
   OR
   Type Ctrl-F.
   A dialog box opens.

2. Type text you want to find.

3. Click OK.

The editor jumps to the first occurrence of the specified text (or gives an error that no matching text could be found). The line with the matching text appears at the *top* of the window.

To find the next occurrence of the same text:

1. From the Search menu, choose Find Next.
   OR
   Click the Next button on the toolbar.
   OR
   Press F3.

The editor jumps to the next occurrence of the text (or gives an error that no matching text could be found). The line with the matching text appears at the *top* of the window.

Printing a File

To print the file that is open in the editor:

1. From the File menu, choose Print.
   OR
   Click the Print button on the toolbar.
   A dialog box opens.

2. Change options, if needed, in the dialog box.

3. Click OK.

Saving a File

To save the file that is open in the editor:

1. From the File menu, choose Save.
   OR
   Click the Save button on the toolbar.

To save the file under a new name:

1. From the File menu, choose Save As.

2. Type the new file name, then click Save.

Closing a File

To close the file that is open in the editor:

1. From the File menu, choose Close.
   OR
   Click the Close button on the toolbar.

If you have edited the file without saving it, you are prompted to save before the file closes.

Exiting the Editor

To exit the editor:

1. From the File menu, choose Exit.
   OR
   Click the Exit button on the toolbar.

If you have edited the file without saving it, you are prompted to save before the Editor closes.

Chapter 12: Getting Technical Support and Updating IC Models

Summary

This chapter describes:

- ♦ how to contact HyperLynx for technical support
- ♦ how to send LineSim files for technical investigation
- ♦ how to connect to the HyperLynx Web site
- ♦ how to update IC models automatically over the Internet
- ♦ a HyperLynx Web information feature called “HyperLynx Web News”

How to Contact HyperLynx for Technical Support

HyperLynx automatically offers technical support to all customers for the first 30 days after they purchase a HyperLynx product. Beyond 30 days, support is available to customers who have purchased product maintenance.

U.S. customers can contact HyperLynx in any of the following ways:

- ♦ e-mail: support@hyperlynx.com

- voice: 425-869-2320
- fax: 425-881-1008

International customers should first contact the VAR or reseller from whom they purchased their HyperLynx software. If local support is not available or is inadequate, then international customers should e-mail to support@hyperlynx.com.

All written correspondence to HyperLynx should include the user name, company name, exact version of HyperLynx software being used, and key serial number. For details on determining your version and key number, see the sections below.

Determining the Version of HyperLynx Software

When requesting technical support, knowing the exact version of HyperLynx software with which you are having a problem is often important.

To determine the version of HyperLynx software you're running:

1. With the HyperLynx software running, from the Help menu, choose About.
2. At the top of the About dialog box, note the version number ("Vx.xx") and the build number (e.g., "168").

Determining the Key Serial Number

When requesting technical support, knowing serial number of your hardware key is important. (Even if you are using a floating license, the hardware key on your server PC has a serial number.)

To determine the serial number of your key:

1. With the HyperLynx software running, from the Options menu, choose Licensing.
2. *If you are running node-locked* (i.e., with the key attached to your PC), the number is displayed in the Key Serial Number field toward the bottom of the dialog box.

If you are running with a floating license (meaning that the key is attached to a remote server PC), then in the Licensing dialog box, in the HyperLynx License Servers area, click the Test button next to a server name. A dialog box opens; the server's key serial number is listed at the top of the box.

Sending Schematic Files to HyperLynx

If you experience problems with a specific schematic, it is often critical to send the .TLN (and other associated files) to HyperLynx for investigation. Not all of a design's information is necessarily included in its .TLN file. For example, you may be using IC models that are not included in HyperLynx's standard product.

To send a complete schematic to HyperLynx:

1. Create a ZIP file containing each of the following files:
   — the .TLN schematic file
   — the file BSW.INI (in the main LineSim directory)
   — any IC model files which are not shipped by HyperLynx

2. E-mail the ZIP file to support@hyperlynx.com

HyperLynx World Wide Web Site

HyperLynx maintains a World Wide Web site, which all customers are encouraged to browse. Of particular interest on the Web site are links to large numbers of other sites offering IBIS models.

To browse the HyperLynx Web site:

1. Point your browser to http://www.hyperlynx.com/

To browse the Web site from inside LineSim:

1. From inside LineSim, from the Help menu, choose HyperLynx on the Web; then choose HyperLynx Web Site.

LineSim attempts to automatically launch your Web browser and connect directly to HyperLynx's home page.

Links to Other Sites Offering IBIS Models

Many IC manufacturers now make IBIS IC models available directly from their World Wide Web or FTP sites. While HyperLynx attempts to ship with LineSim as many of these models as possible, some manufacturers do not allow their models to be shipped with third-party products. Also, new models are appearing constantly.

In order to make searching for new IBIS models on manufacturer Web and FTP sites as easy as possible, HyperLynx maintains a Web page with links to all known IBIS sites. This page is updated approximately once a month.

To go directly to the HyperLynx IBIS page, from inside LineSim:

1. From inside LineSim, from the Help menu, choose HyperLynx on the Web; then choose Links to IBIS Models.
   OR
   From the File menu, choose Go to the IBIS Web Page.

To go to the HyperLynx IBIS Web page “manually”:

1. Point your Web browser to http://www.hyperlynx.com/ibis.html

Updating IC Models Over the Internet

LineSim has the ability (if you are properly connected to the Internet) to update your IC models automatically over the Internet from HyperLynx’s Web site. For complete details, see Chapter 4, section “Updating Models over the Internet.”

HyperLynx Web News

As part of its Web site, HyperLynx offers a feature called “HyperLynx Web News” (or “HyperNews”). Web News gives you a way to receive messages from HyperLynx regarding product updates and patches, technical information about high-speed design, and so forth. You can access the feature from inside LineSim, in conjunction with your Web browser.

To access HyperLynx Web News:

1. From inside LineSim, from the File menu, choose HyperNews.
   OR
   From the Help menu, choose HyperLynx on the Web, then HyperNews.

LineSim automatically invokes your Web browser and points it to a location at HyperLynx's site. In addition to reading the information on the News page, you can also move to other topics on the HyperLynx Web site.

Appendix A: IBIS V2.1 Specification

Summary

This appendix contains the specification for the IBIS signal-integrity modeling format, version 2.1. One section of the specification has been removed, for size and readability reasons: the rarely used [Package Model] and [Define Package Model] keywords. To view the complete specification including these keywords, open the Visual IBIS Editor and access its Help system. (See Chapter 5, section “The Visual IBIS Editor” for details.)

See Chapter 4, section “IC-Model Formats” for details on LineSim’s IC-modeling formats.

Detailed Specification

```
|==============================================================================
| I/O Buffer Information Specification (IBIS) Version 2.1 (December 13, 1995)
|
| IBIS is a standard for electronic behavioral specifications of integrated
| circuit input/output analog characteristics.
|==============================================================================
| Statement of Intent:
|
| In order to enable an industry standard method to electronically transport
| IBIS Modeling Data between silicon vendors, simulation software vendors, and
| end customers, this template is proposed.  The intention of this template is
| to specify a consistent format that can be parsed by software, allowing
| simulation vendors to derive models compatible with their own products.
|
| One goal of this template is to represent the current state of IBIS data,
| while allowing a growth path to more complex models / methods (when deemed
| appropriate).  This would be accomplished by a revision of the base
| template, and possibly the addition of new keywords or categories.
|
| Another goal of this template is to ensure that it is simple enough for
| silicon vendors and customers to use and modify, while ensuring that it is
| rigid enough for software simulation vendors to write reliable parsers.
|
| Finally, this template is meant to contain a complete description of the I/O
| elements on an entire component.  Consequently, several models will need to
| be defined in each file, as well as a table that equates the appropriate
| buffer to the correct pin and signal name.
|
| Version 2.0 of this electronic template was finalized by an industry-wide
| group of simulation experts representing various companies and interests.
| "IBIS Open Forum" meetings were held biweekly to accomplish this task.
|
| Commitment to Backward Compatibility.  Version 1.0 is the first valid IBIS
| ASCII file format.  It represents the minimum amount of I/O buffer
| information required to create an accurate IBIS model of common CMOS and
| bipolar I/O structures.  Future revisions of the ASCII file will add items
| considered to be "enhancements" to Version 1.0 to allow accurate modeling
| of new, or other, I/O buffer structures.  Consequently, all future
| revisions will be considered supersets of Version 1.0, allowing backward
| compatibility.  In addition, as modeling platforms develop support for
| revisions of the IBIS ASCII template, all previous revisions of the
| template must also be supported.
|
| Version 1.1 update.  The file "ver1_1.ibs" is conceptually the same as
| the 1.0 version of the IBIS ASCII format (ver1_0.ibs).  However, various
| comments have been added for further clarification.
|
| Version 2.0 update.  The file "ver2_0.ibs" maintains backward compatibility
| with Versions 1.0 and 1.1.  All new keywords and elements added in Version
| 2.0 are optional.  A complete list of changes to the specification is in the
| IBIS Version 2.0 Release Notes document ("ver2_0.rn").
|
| Version 2.1 update.  The file "ver2_1.ibs" contains clarification text
| changes, corrections, and two additional waveform parameters beyond
| Version 2.0.
|
|==============================================================================
|
| General syntax rules and guidelines for ASCII IBIS files:
|
| 1)  The content of the files is case sensitive, except for reserved
|     words and keywords.  File names must be all lower case.
|
| 2)  The following words are reserved words and must not be used for
|     any other purposes in the document:
|        POWER - reserved model name, used with power supply pins,
|        GND   - reserved model name, used with ground pins,
|        NC    - reserved model name, used with no-connect pins,
|        NA    - used where data not available.
|
| 3)  File names used in the file must only have lower case characters to
|     enhance UNIX compatibility and must conform to DOS rules.  (The length of
|     a file name should not exceed eight plus three characters and it must
|     not contain special characters that are illegal in DOS).
|
| 4)  The file must have no more than 80 characters per line.
|
| 5)  Anything following the comment character is ignored and considered a
|     comment on that line.  The default "|" (pipe) character can be changed
|     by the keyword [Comment Char] to any other character. The [Comment Char]
|     keyword can be used throughout the file as desired.
|
| 6)  Keywords must be enclosed in square brackets, [], and must start in
|     column 1 of the line.
|
| 7)  Underscores and spaces are equivalent in keywords.  Spaces are not
|     allowed in subparameter names.
|
| 8)  Valid scaling factors are:
|        T = tera        k = kilo        n = nano
|        G = giga        m = milli       p = pico
|        M = mega        u = micro       f = femto
|     When no scaling factors are specified, the appropriate base units are
|     assumed.  (These are volts, amperes, ohms, farads, henries, and seconds.)
|     The parser looks at only one alphabetic character after a numerical
|     entry, therefore it is enough to use only the prefixes to scale the
|     parameters.  However, for clarity, it is allowed to use full
|     abbreviations for the units, (e.g., pF, nH, mA, mOhm).  In addition,
|     scientific notation IS allowed (e.g., 1.2345e-12).
|
| 9)  The V/I data tables should use enough data points around sharply curved
|     areas of the V/I curves to describe the curvature accurately.  In linear
|     regions there is no need to define unnecessary data points.
|
| 10) The usage of TAB characters is legal, but they should be avoided as far
|     as possible.  This is to eliminate possible complications which might
|     arise in situations when TAB characters are automatically converted to
|     multiple spaces by text editing, file transfering and similar software.
|     In cases like that, lines might become longer than 80 characters, which
|     is illegal in IBIS files.
|
| 11) Currents are considered positive when their direction is into the
|     component.
|
| 12) All temperatures are represented in degrees Celsius.
|
| 13) Important supplemental information is contained in the last section,
|     "NOTES ON DATA DERIVATION METHOD", concerning how data values are
|     derived.
|
|==============================================================================
|     Keyword:  [IBIS Ver]
|    Required:  Yes
| Description:  Specifies the IBIS template version.  This keyword informs
|               electronic parsers of the kinds of data types that are
|                present in the file.
| Usage Rules:  [IBIS Ver] must be the first keyword in any IBIS file.  It is
|               normally on the first line of the file, but can be preceded
|               by comment lines that must begin with a "|".
|------------------------------------------------------------------------------
[IBIS Ver]      2.1                     | Used for template variations
|
|==============================================================================
|     Keyword:  [Comment Char]
|    Required:  No
| Description:  Defines a new comment character to replace the default
|               "|" (pipe) character, if desired.
| Usage Rules:  The new comment character to be defined must be followed by
|               the underscore character and the letters "char".  For example:
|               "|_char" redundantly redefines the comment character to be
|               the pipe character.  The new comment character is in effect
|               only following the [Comment Char] keyword.  The following
|               characters MAY NOT be used:  A B C D E F G H I J K L M N O P
|               Q R S T U V W X Y Z a b c d e f g h i j k l m n o p q r s t u
|               v w x y z 0 1 2 3 4 5 6 7 8 9 [ ] . _ / = + -
| Other Notes:  The [Comment Char] keyword can be used throughout the file, as
|               desired.
|------------------------------------------------------------------------------
[Comment Char]  |_char
|
|==============================================================================
|     Keyword:  [File Name]
|    Required:  Yes
| Description:  Specifies the name of the IBIS file, "filename.ibs".
| Usage Rules:  The file name must comply with normal DOS rules (8 char. max.
|               and no characters that are illegal in DOS).  In addition, it
|               must be all lower case, and use the extension ".ibs".
|------------------------------------------------------------------------------
[File Name]     ver2_1.ibs
|
|==============================================================================
|     Keyword:  [File Rev]
|    Required:  Yes
| Description:  Tracks the revision level of a particular .ibs file.
| Usage Rules:  Revision level is set at the discretion of the engineer
|               defining the file.  The following guidelines are recommended:
|                  0.x     silicon and file in development
|                  1.x     pre-silicon file data from silicon model only
|                  2.x     file correlated to actual silicon measurements
|                  3.x     mature product, no more changes likely
|------------------------------------------------------------------------------
[File Rev]      1.0                     | Used for .ibs file variations
|
|==============================================================================
| Keywords:     [Date] [Source] [Notes] [Disclaimer] [Copyright]
| Required:     No
| Description:  Optionally clarifies the file.
| Usage Rules:  The keyword arguments can contain blanks, and be of
|               any format.  The [Date] keyword argument is limited to a
|               maximum of 40 characters, and the month should be spelled
|               out for clarity.
|
|               Because IBIS model writers may consider the information in
|               these keywords essential to users, and sometimes legally
|               required, design automation tools should make this information
|               available.  Derivative models should include this text
|               verbatim.  Any text following the [Copyright] keyword must be
|               included in any derivative models verbatim.
|------------------------------------------------------------------------------
[Date]          December 13, 1995          | The latest file revision date
|
[Source]        Put originator and the source of information here.  For
                example:
                From silicon level SPICE model at Intel.
                From lab measurement at IEI.
                Compiled from manufacturer's data book at Quad Design, etc.
|
[Notes]         Use this section for any special notes related to the file.
|
[Disclaimer]    This information is for modeling purposes only, and is not
                guaranteed.                     | May vary by component
|
[Copyright]     Copyright 1995, XYZ Corp., All Rights Reserved
|
|==============================================================================
|     Keyword:  [Component]
|    Required:  Yes
| Description:  Marks the beginning of the IBIS description of the integrated
|               circuit named after the keyword.
| Usage Rules:  If the .ibs file contains data for more than one component,
|               each section must begin with a new [Component] keyword.  The
|               length of the Component Name must not exceed 40 characters,
|               and blank characters are allowed.
|
|               NOTE: Blank characters are not recommended due to usability
|               issues.
|------------------------------------------------------------------------------
[Component]     7403398 MC452
|
|==============================================================================
|     Keyword:  [Manufacturer]
|    Required:  Yes
| Description:  Clarifies the component's manufacturer.
| Usage Rules:  The length of the Manufacturer's Name must not exceed 40
|               characters (blank characters are allowed, e.g., Texas
|               Instruments).  In addition, each manufacturer must use a
|               consistent name in all .ibs files.
|------------------------------------------------------------------------------
[Manufacturer]  Intel Corp.
```

```
|
|==============================================================================
|     Keyword:  [Package]
|    Required:  Yes
| Description:  Defines a range of values for the default packaging resistance,
|               inductance, and capacitance of the component pins.
| Sub-Params:   R_pkg, L_pkg, C_pkg
| Usage Rules:  The typical (typ) column must be specified.  If data for the
|               other columns are not available, they must be noted with "NA".
| Other Notes:  If RLC parameters are available for individual pins, they can
|               be listed in columns 4-6 under keyword [Pin].  The values
|               listed in the [Pin] description section override the default
|               values defined here.  Use the [Package Model] keyword for more
|               complex package descriptions.  If defined, the [Package Model]
|               data overrides the values in the [Package] keyword.
|               Regardless, the data listed under the [Package] keyword must
|               still contain valid data.
|------------------------------------------------------------------------------
[Package]
| variable      typ             min             max
R_pkg           250.0m          225.0m          275.0m
L_pkg           15.0nH          12.0nH          18.0nH
C_pkg           18.0pF          15.0pF          20.0pF
|
|==============================================================================
|     Keyword:  [Pin]
|    Required:  Yes
| Description:  Associates the component's I/O models to its various external
|               pins and signal names.
|  Sub-Params:  signal_name, model_name, R_pin, L_pin, C_pin
| Usage Rules:  All pins on a component must be specified.  The first column
|               must contain the pin name.  The second column, signal_name,
|               gives the data book name for the signal on that pin.  The
|               third column, model_name, associates the I/O model for that
|               pin.  Each model_name must have a [Model] keyword below,
|               unless it is a reserved model name (POWER, GND, or NC).
|
|               Each line must contain either three or six columns.  A pin
|               line with three columns only associates the pin's signal and
|               model.  Six columns can be used to override the default
|               package values (specified under [Package]) FOR THAT PIN ONLY.
|               When using six columns, the headers R_pin, L_pin, and C_pin
|               must be listed.  If "NA" is in columns 4 through 6, the
|               default packaging values must be used.
|
|               Column length limits are:
|                  [Pin]         5 characters max
|                  model_name   20 characters max
|                  signal_name  20 characters max
|                  R_pin         9 characters max
|                  L_pin         9 characters max
|                  C_pin         9 characters max
```

```
|------------------------------------------------------------------------------
[Pin]   signal_name     model_name      R_pin   L_pin   C_pin
|
  1     RAS0#           Buffer1         200.0m  5.0nH   2.0pF
  2     RAS1#           Buffer2         209.0m  NA      2.5pF
  3     EN1#            Input1          NA      6.3nH   NA
  4     A0              3-state
  5     D0              I/O1
  6     RD#             Input2          310.0m  3.0nH   2.0pF
  7     WR#             Input2
  8     A1              I/O2
  9     D1              I/O2
 10     GND             GND             297.0m  6.7nH   3.4pF
 11     RDY#            Input2
 12     GND             GND             270.0m  5.3nH   4.0pF
|  .
|  .
|  .
 18     Vcc3            POWER
 19     NC              NC
 20     Vcc5            POWER           226.0m  NA      1.0pF
|
|==============================================================================
|    Keyword:  [Pin Mapping]
|   Required:  No
|Description:  Used to indicate which power and ground buses a given driver,
|              receiver, or terminator is connected to.
| Sub-Params:  pulldown_ref, pullup_ref, gnd_clamp_ref, power_clamp_ref
|Usage Rules:  Each power and ground bus is given a unique name which must
|              not exceed 15 characters.  The first column contains a pin
|              number.  Each pin number must match one of the pin numbers
|              declared previously in the [Pin] section of the IBIS file.
|              The second column, pulldown_ref, designates the ground bus
|              connections for that pin.  Here the term ground bus can
|              also mean another power bus.  The third column pullup_ref
|              designates the power bus connection.  The fourth and fifth
|              columns gnd_clamp_ref and power_clamp_ref contain
|              entries, if needed, to specify different ground bus
|              and power bus connections than those previously specified.
|
|              If the [Pin Mapping] keyword is present, then the bus
|              connections for EVERY pin listed in the [Pin] section must
|              be given.
|
|              Each line must contain either three or five columns.  Use the
|              NC reserved word for entries that are not needed or that follow
|              the conditions below:
|
|              All entries with identical labels are assumed to be connected.
|              Each unique entry label must connect to at least one pin whose
|              model_name is POWER or GND.
|
|               If a pin has no connection, then both the pulldown_ref
|               and pullup_ref subparameters for it will be NC.
|
|               GND and POWER pin entries and buses are designated by
|               entries in either the pulldown_ref or pullup_ref columns.
|               There is no implied association to any column other than
|               through explicit designations in other pins.
|
|               For any other type of pin, the pulldown_ref column contains
|               the power connection for the [Pulldown] table for non-ECL type
|               [Models].  This is also the power connection for the [GND Clamp]
|               table and the [Rgnd] model unless overridden by a specification
|               in the gnd_clamp_ref column.
|
|               Also, the pullup_ref column contains the power connection
|               for the [Pullup] table and, for ECL type models, the [Pulldown]
|               table.  This is also the power connection for the [POWER Clamp]
|               table and the [Rpower] model unless overridden by a
|               specification in the power_clamp_ref column.
|
|               The column length limits are:
|                       [Pin Mapping]     5 characters max
|                       pulldown_ref     15 characters max
|                       pullup_ref       15 characters max
|                       gnd_clamp_ref    15 characters max
|                       power_clamp_ref  15 characters max
|
|               When 5 columns are specified, the headings gnd_clamp_ref and
|               power_clamp_ref must be used.  Otherwise, these headings can
|               be omitted.
|----------------------------------------------------------------------------
[Pin Mapping]  pulldown_ref     pullup_ref   gnd_clamp_ref  power_clamp_ref
|
1              GNDBUS1          PWRBUS1    | Signal pins and their associated
2              GNDBUS2          PWRBUS2    | ground and power connections
3              GNDBUS1          PWRBUS1      GNDCLMP        PWRCLAMP
4              GNDBUS2          PWRBUS2      GNDCLMP        PWRCLAMP
5              GNDBUS2          PWRBUS2      NC             PWRCLAMP
6              GNDBUS2          PWRBUS2      GNDCLMP        NC
                                           | Some possible clamping connections
|  .                                       | are shown above for illustration
|  .                                       | purposes
|  .
11             GNDBUS1          NC         | One set of ground connections.
12             GNDBUS1          NC         | NC indicates no connection to
13             GNDBUS1          NC         | power bus.
|  .
21             GNDBUS2          NC         | Second set of ground connections
22             GNDBUS2          NC
23             GNDBUS2          NC
|  .
31             NC               PWRBUS1    | One set of power connections.
```

```
32              NC               PWRBUS1   | NC indicates no connection to
33              NC               PWRBUS1   | ground bus.
|  .
41              NC               PWRBUS2   | Second set of power connections
42              NC               PWRBUS2
43              NC               PWRBUS2
|  .
51              GNDCLMP          NC        | Additional power connections
52              NC               PWRCLMP   | for clamps
|
|==============================================================================
|    Keyword:  [Diff Pin]
|   Required:  No
|Description:  Associates differential pins, their differential
|              threshold voltages, and differential timing delays.
| Sub-Params:  inv_pin, vdiff, tdelay_typ, tdelay_min, tdelay_max
|Usage Rules:  Enter only differential pin pairs.  The first column,
|              [Diff Pin], contains a non-inverting pin number.  The second
|              column, inv_pin, contains the corresponding inverting pin number
|              for I/O output.  Each pin number must match the pin
|              numbers declared previously in the [Pin] section of the IBIS
|              file.  The third column, vdiff, contains the specified
|              output and differential threshold voltage between pins if
|              the pins are Input or I/O model types.  For output only
|              differential pins, the vdiff entry is 0 V.  The fourth, fifth,
|              and sixth columns, tdelay_typ, tdelay_min, and tdelay_max,
|              contain launch delays of the non-inverting pins relative to
|              the inverting pins.  The values can be of either polarity.
|
|              If a pin is a differential input pin, the differential input
|              threshold (vdiff) overrides and supersedes the need for Vinh and
|              Vinl.
|
|              If vdiff is not defined for a pin that is defined as requiring a
|              Vinh by its [Model] type, vdiff is set to the default value of
|              200 mV.
|
|Other Notes:  The output pin polarity specification in the table overrides
|              the [Model] Polarity specification such that the pin in the
|              [Diff Pin] column is Non-Inverting and the pin in the
|              inv_pin column is Inverting.  This convention enables
|              one [Model] to be used for both pins.
|
|              Column length limits are:
|                 [Diff Pin]     5 characters max
|                 inv_pin        5 characters max
|                 vdiff          9 characters max
|                 tdelay_typ     9 characters max
|                 tdelay_min     9 characters max
|                 tdelay_max     9 characters max
|
|              Each line must contain either four or six columns.  If "NA" is
|              entered in the vdiff, tdelay_typ, or tdelay_min columns, its
|              entry is interpreted as 0 V or 0 ns.  If "NA" appears in
|              the tdelay_max column, its value is interpreted as the
|              tdelay_typ value.  When using six columns, the headers
|              tdelay_min and tdelay_max must be listed.  Entries for the
|              tdelay_min column are based on minimum magnitudes; and
|              tdelay_max column, maximum magnitudes.  One entry of vdiff,
|              regardless of its polarity, is used for difference magnitudes.
|----------------------------------------------------------------------------
[Diff Pin]  inv_pin  vdiff  tdelay_typ tdelay_min tdelay_max
|
 3           4       150mV    -1ns       0ns      -2ns  | Input or I/O pair
 7           8         0V      1ns        NA        NA  | Output* pin pair
 9          10         NA       NA        NA        NA  | Output* pin pair
16          15       200mV     1ns    | Input or I/O pin pair
20          19         0V       NA    | Output* pin pair, tdelay = 0
22          21         NA       NA    | Output*, tdelay = 0
                                      | * Could be Input or I/O with vdiff = 0
|
|===============================================================================
|     Keyword:  [Model]
|    Required:  Yes
| Description:  Used to define a model, and its attributes.
|  Sub-Params:  Model_type, Polarity, Enable, Vinl, Vinh, C_comp, Vmeas, Cref,
|               Rref, Vref
| Usage Rules:  Each model type must begin with the keyword [Model].
|               The model name must match the one that is listed under
|               the [Pin] keyword and must not contain more than 20 characters.
|               A .ibs file must contain enough [Model] keywords to cover all
|               of the model names specified under the [Pin] keyword, except
|               for those model names that use reserved words (POWER, GND and
|               NC).  Model names with reserved words are an exception and they
|               do not have to have a corresponding [Model] keyword.
|
|               Model_type must be one of the following:
|               Input, Output, I/O, 3-state, Open_drain, I/O_open_drain,
|               Open_sink, I/O_open_sink, Open_source, I/O_open_source,
|               Input_ECL, Output_ECL, I/O_ECL, and Terminator.
|
|               Special usage rules apply to the following.  Some definitions
|               are included for clarification:
|
|               Input              These model types must have Vinl and Vinh
|               I/O                defined.  If they are not defined, the
|               I/O_open_drain     parser issues a warning and the default
|               I/O_open_sink      values of Vinl = 0.8 V and Vinh = 2.0 V are
|               I/O_open_source    assumed.
|
|               Input_ECL          These model types must have Vinl and Vinh
|               I/O_ECL            defined.  If they are not defined, the
|                                  parser issues a warning and the default
|                                  values of Vinl = -1.475 V and Vinh =
|                                   -1.165 V are assumed.
|
|                 Terminator        This model type is an input-only device that
|                                   can have analog loading effects on the
|                                   circuit being simulated but has no digital
|                                   logic thresholds.  Examples of Terminators
|                                   are: capacitors, termination diodes, and
|                                   pull-up resistors.
|
|                 Output            This model type indicates that an output
|                                   always sources and/or sinks current and
|                                   cannot be disabled.
|
|                 3-state           This model type indicates that an output
|                                   can be disabled, i.e. put into a high
|                                   impedance state.
|
|                 Open_sink         These model types indicate that the output
|                 Open_drain        has an OPEN side (do not use the [Pullup]
|                                   keyword, or if it must be used, set I = 0 mA
|                                   for all voltages specified) and the output
|                                   SINKS current.  Open_drain model type is
|                                   retained for backward compatibility.
|
|                 Open_source       This model type indicates that the
|                                   output has an OPEN side (do not use the
|                                   [Pulldown] keyword, or if it must be used,
|                                   set I = 0 mA for all voltages specified) and
|                                   the output SOURCES current.
|
|                 Input_ECL         These model types specify that the model
|                 Output_ECL        represents an ECL type logic that follows
|                 I/O_ECL           different conventions for the [Pulldown]
|                                   keyword.
|
|                 The Model_type and C_comp subparameters are required.  The
|                 Polarity, Enable, Vinl, Vinh, Vmeas, Cref, Rref, and Vref sub-
|                 parameters are optional.  C_comp defines the silicon die
|                 capacitance.  This value should not include the capacitance of
|                 the package.  C_comp is allowed to use "NA" for the min and max
|                 values only.  The Polarity subparameter can be defined as
|                 either Non-Inverting or Inverting, and the Enable sub-
|                 parameter can be defined as either Active-High or Active-Low.
|
|                 The Cref and Rref subparameters correspond to the test load
|                 that the manufacturer uses when specifying the propagation
|                 delay and/or output switching time of the device.  The Vmeas
|                 subparameter is the reference voltage level that the
|                 manufacturer uses for the component.  Include Cref, Rref, and
|                 Vmeas information to facilitate board-level timing simulation.
|                 The assumed connections for Cref, Rref, and Vref are shown in
|                 the following diagram:
```

```
|
|                            _________
|                           |         |
|                           |      |\ |            Rref
|                           |Driver| \|------o----/\/\/\----o Vref
|                           |      | /|      |
|                           |      |/ |     === Cref
|                           |_________|      |
|                                            |
|                                           GND
|
| Other Notes:  A complete [Model] description normally contains the following
|               keywords:  [Voltage Range], [Pullup], [Pulldown], [GND Clamp],
|               [POWER Clamp], and [Ramp].  A Terminator model uses one
|               or more of the [Rgnd], [Rpower], [Rac], and [Cac].  However,
|               some models may have only a subset of these keywords.  For
|               example, an input structure normally only needs the
|               [Voltage Range], [GND Clamp], and possibly the [POWER Clamp]
|               keywords.  If one or more of [Rgnd], [Rpower], [Rac], and [Cac]
|               keywords are used, then the Model_type must be Terminator.
|------------------------------------------------------------------------------
| Signals       CLK1, CLK2,...         | Optional signal list, if desired
[Model]         Clockbuffer
Model_type      I/O
Polarity        Non-Inverting
Enable          Active-High
Vinl = 0.8V                             | input logic "low" DC voltage, if any
Vinh = 2.0V                             | input logic "high" DC voltage, if any
Vmeas = 1.5V                |Reference voltage for timing measurements
Cref = 50pF                 |Timing specification test load capacitance value
Rref = 500                  |Timing specification test load resistance value
Vref = 0                    |Timing specification test load voltage
| variable      typ             min             max
C_comp          12.0pF          10.0pF          15.0pF
|
|==============================================================================
|     Keyword:  [Temperature Range]
|    Required:  Yes, if other than the preferred 0, 50, 100 degree Celsius
|               range
| Description:  Defines the temperature range over which the model is
|               to operate.
| Usage Rules:  List the actual die temperatures (not percentages) in the
|               typ, min, max format.  "NA" is allowed for min and max only.
| Other Notes:  The [Temperature Range] keyword also describes the temperature
|               range over which the various V/I curves and ramp rates were
|               derived.
|------------------------------------------------------------------------------
| variable              typ             min             max
[Temperature Range]     27.0            -50             130.0
|
|==============================================================================
|    Keyword:  [Voltage Range]
|   Required:  Yes, if [Pullup Reference], [Pulldown Reference],
|              [POWER Clamp Reference], and [GND Clamp Reference] are not
|              present
|Description:  Defines the power supply voltage tolerance over which the
|              model is intended to operate.  It also specifies the default
|              voltage rail to which the pull-up and [POWER Clamp] V/I data is
|              referenced.
|Usage Rules:  Provide actual voltages (not percentages) in the typ, min,
|              max format.  "NA" is allowed for the min and max values only.
|Other Notes:  If the [Voltage Range] keyword is not present, then all four
|              of these keywords described below must be present: [Pullup
|              Reference], [Pulldown Reference], [POWER Clamp Reference],
|              and [GND Clamp Reference].  If the [Voltage Range] is present,
|              the other keywords are optional and may or may not be used as
|              required.  It is legal (although redundant) for an optional
|              keyword to specify the same voltage as specified by the
|              [Voltage Range] keyword.
|------------------------------------------------------------------------
| variable             typ             min             max
[Voltage Range]        5.0V            4.5V            5.5V
|
|========================================================================
|    Keyword:  [Pullup Reference]
|   Required:  Yes, if the [Voltage Range] keyword is not present.
|Description:  Defines a voltage rail other than that defined by the
|              [Voltage Range] keyword as the reference voltage for the
|              pull-up V/I data.
|Usage Rules:  Provide actual voltages (not percentages) in the typ, min,
|              max format.  "NA" is allowed for the min and max values only.
|Other Notes:  This keyword, if present, also defines the voltage range over
|              which the min and max dV/dt_r values are derived.
|------------------------------------------------------------------------
| variable             typ             min             max
[Pullup Reference]     5.0V            4.5V            5.5V
|
|========================================================================
|    Keyword:  [Pulldown Reference]
|   Required:  Yes, if the [Voltage Range] keyword is not present.
|Description:  Defines a power supply rail other than 0 V as the reference
|              voltage for the pull-down V/I data.  If this keyword is not
|              present, the voltage data points in the pull-down V/I table
|              are referenced to 0 V.
|Usage Rules:  Provide actual voltages (not percentages) in the typ, min,
|              max format.  "NA" is allowed for the min and max values only.
|Other Notes:  This keyword, if present, also defines the voltage range over
|              which the typ, min, and max dV/dt_f values are derived.
|------------------------------------------------------------------------
| variable             typ             min             max
[Pulldown Reference]   0V              0V              0V
|
|========================================================================
|    Keyword:  [POWER Clamp Reference]
|   Required:  Yes, if the [Voltage Range] keyword is not present.
|Description:  Defines a voltage rail other than that defined by the
|              [Voltage Range] keyword as the reference voltage for the
|              [POWER Clamp] V/I data.
|Usage Rules:  Provide actual voltages (not percentages) in the typ, min,
|              max format.  "NA" is allowed for the min and max values only.
|Other Notes:  Refer the "Other Notes" section of the [GND Clamp Reference]
|              keyword.
|-----------------------------------------------------------------------------
| variable              typ             min             max
[POWER Clamp Reference] 5.0V            4.5V            5.5V
|
|==============================================================================
|    Keyword:  [GND Clamp Reference]
|   Required:  Yes, if the [Voltage Range] keyword is not present.
|Description:  Defines a power supply rail other than 0 V as the reference
|              voltage for the [GND Clamp] V/I data.  If this keyword is not
|              present, the voltage data points in the [GND Clamp] V/I table
|              are referenced to 0 V.
|Usage Rules:  Provide actual voltages (not percentages) in the typ, min,
|              max format.  "NA" is allowed for the min and max values only.
|Other Notes:  Power Supplies: It is intended that standard TTL and CMOS
|              devices be specified using only the [Voltage Range] keyword.
|              However, in cases where the output characteristics of a device
|              depend on more than a single supply and ground, or a pull-up,
|              pull-down, or clamp structure is referenced to something other
|              than the default supplies, use the additional 'reference'
|              keywords.
|------------------------------------------------------------------------------
| variable              typ             min             max
[GND Clamp Reference]   0V              0V              0V
|
|==============================================================================
|    Keywords:  [Pulldown], [Pullup], [GND Clamp], [POWER Clamp]
|    Required:  Yes, if they exist in the device
| Description:  The data points under these keywords define the V/I curves of
|               the pull-down and pull-up structures of an output buffer and
|               the V/I curves of the clamping diodes connected to the GND and
|               the POWER pins, respectively.  Currents are considered positive
|               when their direction is into the component.
|
| Usage Rules:  In each of these sections, the first column contains the
|               voltage value, and the three remaining columns hold the
|               typical, minimum, and maximum current values.  The four
|               entries, Voltage, I(typ), I(min), and I(max) must be placed on
|               a single line and must be separated by at least one white
|               space or tab character.
|
|               All four columns are required under these keywords.  However,
|               data is only required in the typical column.  If minimum
|               and/or maximum current values are not available, the reserved
|               word "NA" must be used.  "NA" can be used for currents in the
|                typical column, but numeric values MUST be specified for the
|                first and last voltage points on any V/I curve.  Each V/I
|                curve must have at least 2, but not more than 100, voltage
|                points.
|
| Other Notes:  The V/I curve of the [Pullup] and the [POWER Clamp] structures
|                are 'Vcc relative', meaning that the voltage values are
|                referenced to the Vcc pin.  (Note: Under these keywords, all
|                references to 'Vcc' refer to the voltage rail defined by the
|                [Voltage range], [Pullup Reference], or [POWER Clamp Reference]
|                keywords, as appropriate.)  The voltages in the data tables are
|                derived from the equation:  Vtable = Vcc - Voutput.
|
|                Therefore, for a 5 V component, -5 V in the table actually
|                means 5 V above Vcc, which is +10 V with respect to ground;
|                and 10 V means 10 V below Vcc, which is -5 V with respect to
|                ground.  Vcc-relative data is necessary to model a pull-up
|                structure properly, since the output current of a pull-up
|                structure depends on the voltage between the output and Vcc
|                pins and not the voltage between the output and ground pins.
|                Note that the [GND Clamp] V/I curve can include quiescent
|                input currents, or the currents of a 3-stated output, if so
|                desired.
|
|                When tabulating data for ECL devices, the data in the pull-down
|                table is measured with the output in the 'logic low' state.
|                In other words, the data in the table represents the V/I
|                characteristics of the output when the output is at the most
|                negative of its two logic levels.  Likewise, the data in the
|                pull-up table is measured with the output in the 'logic one'
|                state and represents the V/I characteristics when the output
|                is at the most positive logic level.  Note that in BOTH of
|                these cases, the data is referenced to the Vcc supply voltage,
|                using the equation  Vtable = Vcc - Voutput.
|
|                Monotonicity Requirements:
|                To be monotonic, the V/I table data must meet any one of the
|                following 8 criteria:
|                  1- The CURRENT axis either increases or remains constant as
|                     the voltage axis is increased.
|                  2- The CURRENT axis either increases or remains constant as
|                     the voltage axis is decreased.
|                  3- The CURRENT axis either decreases or remains constant as
|                     the voltage axis is increased.
|                  4- The CURRENT axis either decreases or remains constant as
|                     the voltage axis is decreased.
|
|                  5- The VOLTAGE axis either increases or remains constant as
|                     the current axis is increased.
|                  6- The VOLTAGE axis either increases or remains constant as
|                     the current axis is decreased.
|                  7- The VOLTAGE axis either decreases or remains constant as
|                    the current axis is increased.
|                 8- The VOLTAGE axis either decreases or remains constant as
|                    the current axis is decreased.
|
|                An IBIS syntax checking program shall test for non-monotonic
|                data and provide a maximum of one note per V/I table if
|                non-montonic data is found.  For example:
|                  "NOTE: Line 300, Pulldown V/I table for model DC040403 is
|                   non-monotonic!  Most simulators will filter this data
|                   to remove the non-monotonic data."
|
|                It is also recognized that the data may be monotonic if
|                currents from both the output stage and the clamp diode are
|                added together as most simulators do.  To limit the complexity
|                of the IBIS Version 2.x syntax checking programs, such
|                programs will conduct monotonicity testing only on one
|                V/I table at a time.
|
|                It is assumed that the simulator sums the clamp curves
|                together with the appropriate pull-up or pull-down curve when
|                a buffer is driving high or low, respectively.  From this
|                assumption and the nature of 3-statable buffers, it follows
|                that the data in the clamping curve sections are handled as
|                constantly present curves and the pull-up and pull-down curves
|                are used only when needed in the simulation.
|
|                The clamp curves of an input or I/O buffer can be measured
|                directly with a curve tracer, with the I/O buffer 3-stated.
|                However, sweeping enabled buffers results in curves that are
|                the sum of the clamping curves and the output structures.
|                Based on the assumption outlined above, the pull-up and
|                pull-down curves of an IBIS model must represent the
|                difference of the 3-stated and the enabled buffer's curves.
|                (Note that the resulting difference curve can demonstrate a
|                non-monotonic shape.)  This requirement enables the simulator
|                to sum the curves, without the danger of double counting, and
|                arrive at an accurate model in both the 3-stated and enabled
|                conditions.
|
|                Since in the case of a non 3-statable buffer, this difference
|                curve cannot be generated through lab measurements (because
|                the clamping curves cannot be measured alone), the pull-up and
|                pull-down curves of an IBIS model can contain the sum of the
|                clamping characteristics and the output structure.  In this
|                case, the clamping curves must contain all zeroes, or the
|                keywords must be omitted.
|-----------------------------------------------------------------------------
[Pulldown]
|  Voltage   I(typ)    I(min)    I(max)
|
   -5.0V    -40.0m    -34.0m    -45.0m
   -4.0V    -39.0m    -33.0m    -43.0m
```

```
|     .
|     .
    0.0V      0.0m      0.0m      0.0m
|     .
|     .
    5.0V     40.0m     34.0m     45.0m
   10.0V     45.0m     40.0m     49.0m
|
[Pullup]                               | Note: Vtable = Vcc - Voutput
|
|  Voltage   I(typ)    I(min)    I(max)
|
   -5.0V     32.0m     30.0m     35.0m
   -4.0V     31.0m     29.0m     33.0m
|     .
|     .
    0.0V      0.0m      0.0m      0.0m
|     .
|     .
    5.0V    -32.0m    -30.0m    -35.0m
   10.0V    -38.0m    -35.0m    -40.0m
|
[GND Clamp]
|
|  Voltage   I(typ)    I(min)    I(max)
|
   -5.0V  -3900.0m  -3800.0m  -4000.0m
   -0.7V    -80.0m    -75.0m    -85.0m
   -0.6V    -22.0m    -20.0m    -25.0m
   -0.5V     -2.4m     -2.0m     -2.9m
   -0.4V      0.0m      0.0m      0.0m
    5.0V      0.0m      0.0m      0.0m
|
[POWER Clamp]                          | Note: Vtable = Vcc - Voutput
|
|  Voltage   I(typ)    I(min)    I(max)
|
   -5.0V   4450.0m       NA        NA
   -0.7V     95.0m       NA        NA
   -0.6V     23.0m       NA        NA
   -0.5V      2.4m       NA        NA
   -0.4V      0.0m       NA        NA
    0.0V      0.0m       NA        NA
|
|==============================================================================
|    Keywords:  [Rgnd], [Rpower], [Rac], [Cac]
|    Required:  Yes, if they exist in the device
| Description:  The data for these keywords define the resistance values of
|               Rgnd and Rpower connected to GND and the POWER pins,
|               respectively.
| Usage Rules:  For each of these keywords, the three columns hold the
|               typical, minimum, and maximum resistance values.  The three
|               entries for R(typ), R(min), and R(max), or the three entries
|               for C(typ), C(min), and C(max) must be placed on a single line
|               and must be separated by at least one white space or tab
|               character.  All three columns are required under these
|               keywords.  However, data is only required in the typical
|               column.  If minimum and/or maximum values are not available,
|               the reserved word "NA" must be used indicating the R(typ) or
|               C(typ) value by default.
| Other Notes:  It should be noted that [Rpower] is connected to 'Vcc' and
|               [Rgnd] is connected to 'GND'.  However, [GND Clamp Reference]
|               voltages, if defined, apply to [Rgnd].  [POWER Clamp Reference]
|               voltages, if defined, apply to [Rpower].  Either or both [Rgnd]
|               and [Rpower] may be defined and may coexist with [GND Clamp]
|               and [POWER Clamp] structures.  If the terminator consists
|               of a series R and C (often referred to as either an AC or RC
|               terminator), then both [Rac] and [Cac] are required.  When
|               [Rgnd], [Rpower], or [Rac] and [Cac] are specified, the
|               Model_type must be Terminator.
|
|                |<-------------TERMINATOR Model--------------->|
|
|                    [Voltage Range] or
|                 [POWER Clamp Reference]
|                          o
|                          |
|                POWER_ o---o---o
|                clamp  |       |
|                    |--o--|    \
|                    |     |    /
|                    | VI  |    \ Rpower    [Package] Keyword
|                    |     |    /            Subparameters *
|                    |--o--|    |        |<----------------->|
|                       |       |
|                       |       |                             PIN
|                 o-----o-------o-----o-----/\/\/\--@@@@@@---o--o
|                 |     |GND_   |     |      R_pkg   L_pkg   |
|                 |     |clamp  |     |                      |
|                 |  |--o--|    |     |                      |
|                 |  |     |    \     |                      |
|                 |  | VI  |    /Rgnd |                      |
|                 |  |     |    \     \                      |
|                 |  |--o--|    /     / Rac                  |
|                 |     |       |     \                      |
|                 |     o---o---o     /                      |
|                 |         |         |                      |
|         C_comp ===        o        === Cac          C_pkg ===
|                 |      GND or       |                      |
|                 |  [GND Clamp Ref]  |                      |
|                 |                   |                      |
|                 o-------------------o----------------------o
|                                     |
|                                     o
|                                    GND
|
|        * Note: More advanced package parameters are available
|                within this standard, including more detailed
|                power and ground net descriptions.
|-----------------------------------------------------------------------------
| variable      R(typ)          R(min)          R(max)
[Rgnd]          330ohm          300ohm          360ohm   | Parallel Terminator
[Rpower]        220ohm          200ohm          NA
|
[Rac]           30ohm           NA              NA
|
| variable      C(typ)          C(min)          C(max)   | AC terminator
[Cac]           50pF            NA              NA
|
|=============================================================================
|     Keyword:  [Ramp]
|    Required:  Yes, except for inputs and terminators
| Description:  Defines the rise and fall times of a buffer.  The ramp rate
|                does not include packaging but does include the effects of the
|                C_comp parameter.
|  Sub-Params:  dV/dt_r, dV/dt_f, R_load
| Usage Rules:  The rise and fall time is defined as the time it takes the
|                output to go from 20% to 80% of its final value.
|                The ramp rate is defined as:
|
|                dV          20% to 80% voltage swing
|                -- = ----------------------------------------
|                dt   Time it takes to swing the above voltage
|
|                The ramp rate must be specified as an explicit fraction
|                and must not be reduced.  The [Ramp] values can use "NA" for
|                the min and max values only.  The R_load subparameter is
|                optional if the preferred 50 ohm load is used.  The R_load sub-
|                parameter is required if a non-standard load is used.
|-------------------------------------------------------------------------------
[Ramp]
| variable       typ             min             max
dV/dt_r         2.20/1.06n      1.92/1.28n      2.49/650p
dV/dt_f         2.46/1.21n      2.21/1.54n      2.70/770p
R_load = 300ohms
|
|===============================================================================
|     Keywords:     [Rising Waveform], [Falling Waveform]
|     Required:     No
|     Description:  Describes the shape of the rising and falling edge
|                    waveforms of a driver.
|     Sub-Params:   R_fixture, V_fixture, V_fixture_min, V_fixture_max,
|                    C_fixture, L_fixture, R_dut, L_dut, C_dut
|     Usage Rules:  Each [Rising Waveform] and [Falling Waveform] keyword
|                    introduces a table of time vs. voltage points that
|                    describe the shape of an output waveform.  These
|                    time/voltage points are taken under the conditions
|                    specified in the R/L/C/V_fixture and R/L/C_dut
|                    subparameters.  The table itself consists of
|                    one column of time points, then three columns of
|                    voltage points in the standard typ, min, and max format.
|                    The four entries must be placed on a single line and
|                    must be separated by at least one white space or tab
|                    character.  All four columns are required.  However, data
|                    is only required in the typical column.  If minimum
|                    or maximum data is not available, use the reserved word
|                    "NA".  The first value in the time column need not be '0'.
|                    Time values must increase as one parses down the table.
|                    The waveform table can contain a maximum of 100 data
|                    points.  A maximum of 100 waveform tables are allowed per
|                    model.  Note that for backwards compatibility, the existing
|                    [Ramp] keyword is still required.  The data in the waveform
|                    table is taken with the effects of the C_comp parameter
|                    included.
|
|                   A waveform table must include the entire waveform;
|                   i.e., the first entry (or entries) in a voltage column
|                   must be the DC voltage of the output before switching
|                   and the last entry (or entries) of the column must be
|                   the final DC value of the output after switching.  Each
|                   table must contain at least two entries.  Thus, numerical
|                   values are required for the first and last entries of any
|                   column containing numerical data.
|
|                   A [Model] specification can contain more than one rising
|                   edge or falling edge waveform table.  However, each new
|                   table must begin with the appropriate keyword and sub-
|                   parameter list as shown below.  If more than one rising or
|                   falling edge waveform table is present, then the data in
|                   each of the respective tables must be time correlated.
|                   In other words, the rising (falling) edge data in each
|                   of the rising (falling) edge waveform tables must be
|                   entered with respect to a common reference point on the
|                   input stimulus waveform.
|
|                   The 'fixture' subparameters specify the loading
|                   conditions under which the waveform is taken.  The R_dut,
|                   C_dut, and L_dut subparameters are analogous to the
|                   package parameters R_pkg, C_pkg, and L_pkg and are used
|                   if the waveform includes the effects of pin
|                   inductance/capacitance.  The diagram below shows the
|                   interconnection of these elements.
|
|                      PACKAGE            |   TEST FIXTURE
|       _________                         |
|      |  DUT    |   L_dut   R_dut        | L_fixture  R_fixture
|      |  die    |---@@@@@--/\/\/\--o-----|--@@@@---o---/\/\/\----- V_fixture
|      |_________|                  |     |         |
|                                   |     |         |
|                                   |     |         |
|                            C_dut ===    |        === C_fixture
|                                   |     |         |
|                                   |     |         |
|                                  GND    |        GND
|
|                   Only the R_fixture and V_fixture subparameters are
|                   required, the rest of the subparameters are optional.
|                   If a subparameter is not used, its value defaults to
|                   zero.  The subparameters must appear in the text after
|                   the keyword and before the first row of the waveform
|                   table.
|
|                   V_fixture defines the voltage for typ, min, and max
|                   supply conditions.  However, when the fixture voltage
|                   is related to the power supply voltages, then the
|                   subparameters V_fixture_min and V_fixture_max can
|                   be used to further specify the fixture voltage for
|                   min and max supply voltages.
|------------------------------------------------------------------------------
[Rising Waveform]
R_fixture = 500
V_fixture = 5.0
V_fixture_min = 4.5           |Note, R_fixture is connected to Vcc
V_fixture_max = 5.5           |Specified, but not used in this example
C_fixture = 50p
L_fixture = 2n
C_dut = 7p
R_dut = 1m
L_dut = 1n
|Time     V(typ)     V(min)     V(max)
 0.0ns     0.3        0.5         NA
 0.5ns     0.3        0.5         NA
 1.0ns     0.6        0.7         NA
 1.5ns     0.9        0.9         NA
 2.0ns     1.5        1.3         NA
 2.5ns     2.1        1.7         NA
 3.0ns     3.0        2.7         NA
 3.5ns     3.2        3.0         NA
|
[Falling Waveform]
R_fixture = 50
V_fixture = 0
|Time     V(typ)     V(min)     V(max)
 10.0ns     3.2        3.0         NA
 10.5ns     3.0        2.7         NA
 11.0ns     2.1        1.7         NA
 11.5ns     1.5        1.3         NA
 12.0ns     0.9        0.9         NA
 12.5ns     0.6        0.7         NA
 13.0ns     0.3        0.5         NA
 13.5ns     0.3        0.5         NA
|
|==============================================================================
|     Keyword:  [End]
|    Required:  Yes
| Description:  Defines the end of the .ibs file.
|------------------------------------------------------------------------------
[End]
|
|==============================================================================
|
|                       NOTES ON DATA DERIVATION METHOD
|
| This section explains how data values are derived.  It describes certain
| assumed parameter and table extraction conditions if they are not
| explicitly specified.  It also describes the allocation of data into
| the "typ", "min", and "max" columns under variations of voltage,
| temperature, and process.
|
| The required "typ" column for all data represents typical operating
| conditions.  For most [Model] keyword data, the "min" column describes
| slow, weak performance, and  the "max" column describes the fast, strong
| performance.  It is permissible to use slow, weak devices or models to
| derive the data for the "min" column, and to use fast, strong devices or
| models to derive the data in the "max" columns under the corresponding
| voltage and temperature derating conditions for these columns.  It is also
| permissible to use typical devices or models derated by voltage and
| temperature and optionally apply proprietary "X%" and "Y%" factors
| described later for further derating.  This methodology has the
| nice feature that the data can be derived either from vendor-proprietary
| silicon models, or typical device measurement over temperature/voltage.
|
| The voltage and temperature keywords and optionally the process models
| control the conditions which define the "typ", "min", and "max" column
| entries for all V/I table keywords [Pulldown], [Pullup], [Gnd Clamp],
| and [Power Clamp]; all [Ramp] subparameters dV/dt_r and dV/dt_f; and
| all waveform table keywords and subparameters [Rising Waveform],
| [Falling Waveform], V_fixture, V_fixture_min, and V_fixture_max.
|
| The voltage keywords which control the voltage conditions are [Voltage
| Range], [Pulldown Reference], [Pullup Reference], [Gnd Clamp Reference],
| and [Power Clamp Reference].  The entries in the "min" columns
| contain the smallest magnitude voltages, and the entries in the "max"
| columns contain the largest magnitude voltages.
|
| The optional [Temperature] keyword will contain the temperature
| which causes or amplifies the slow, weak conditions in the "min" column
| and the temperature which causes or amplifies the fast, strong
| conditions in the "max" column.  Therefore, the "min" column for
| [Temperature] will contain the lowest value for bipolar devices (TTL
| and ECL) and the highest value for CMOS devices.  Default values
| described later are assumed if temperature is not specified.
|
| The "min" and "max" columns for all remaining keywords and subparameters
| will contain the smallest and largest magnitude values.  This applies
| to the [Model] subparameter C_comp as well even if the correlation
| to the voltage, temperature, and process variations are known because
| information about such correlation is not available in all cases.
|
| C_comp is considered an independent variable.  This is because C_comp
| includes bonding pad capacitance, which does not necessarily track
| fabrication process variations.  The conservative approach to using IBIS
| data will associate large C_comp values with slow, weak models, and the
| small C_comp values with fast, strong models."
|
| The default temperatures under which all V/I tables are extracted are
| provided below.  The same defaults also are stated for the [Ramp]
| subparameters, but they also apply for the waveform keywords.
|
| The stated voltage ranges for V/I tables cover the most common, single
| supply cases.  When multiple supplies are specified, the voltages shall
| extend similarly to values which handle practical extremes in reflected
| wave simulations.
|
| For the [Ramp] subparameters, the default test load and voltages are
| provided.  However, the test load can be entered directly by the R_load
| subparameter.  The allowable test loads and voltages for the waveform
| keywords are stated by required and optional subparameters; no defaults
| are needed.  Even with waveform keywords, the [Ramp] keyword continues
| to be required so that the IBIS model remains functional in situations
| which do not support waveform processing.
|
| The following discussion lists test details and default conditions.
|
| 1) V/I curves for CMOS devices:
|      typ = typical voltage, typical temp deg C, typical process
|      min = minimum voltage, max temp deg C, typical process, minus "X%"
|      max = maximum voltage, min temp deg C, typical process, plus "X%"
|
|    V/I curves for bipolar devices:
|      typ = typical voltage, typical temp deg C, typical process
|      min = minimum voltage, min temp deg C, typical process, minus "X%"
|      max = maximum voltage, max temp deg C, typical process, plus "X%"
|
|    Nominal, min, and max temperature are specified by the manufacturer
|    of the part.  The default range is 50 deg C nom, 0 deg C min, and
|    100 deg C max temperatures.
|
|    X% should be statistically determined by the silicon vendor based
|    on numerous fab lots, test chips, process controls, etc..  The value
|    of X need not be published in the IBIS file, and may decrease over
|    time as data on the I/O buffers and silicon process increases.
|
|    Temperatures are junction temperatures.
|
| 2) Voltage Ranges:
|    Points for each curve must span the voltages listed below:
|
|       Curve                   Low Voltage             High Voltage
|       -----------             -----------             ------------
|       [Pulldown]              GND - POWER             POWER + POWER
|       [Pullup]                GND - POWER             POWER + POWER
|       [GND Clamp]             GND - POWER             GND + POWER
|       [POWER Clamp]           POWER                   POWER + POWER
|
|    As described in the [Pulldown Reference] keyword section, the V/I curve of
|    the [Pullup] and the [POWER Clamp] structures are 'Vcc relative', using
|    the equation:  Vtable = Vcc - Voutput.
|
|    For example, a device with a 5 V power supply voltage should be
|    characterized between (0 - 5) = -5 V and (5 + 5) = 10 V; and a
|    device with a 3.3 V power supply should be characterized between
|    (0 - 3.3) = -3.3 V and (3.3 + 3.3) = 6.6 V for the pull-down curve.
```

```
|
|    When tabulating output data for ECL type devices, the voltage points
|    must span the range of Vcc to Vcc - 2.2 V.  This range applies to both the
|    pull-up and pull-down tables.  Note that this range applies ONLY when
|    characterizing an ECL output.
|
|    These voltage ranges must be spanned by the IBIS data.  Data derived
|    from lab measurements may not be able to span these ranges as such and
|    so may need to be extrapolated to cover the full range.  This data must
|    not be left for the simulator to provide.
|
| 3) Ramp Rates:
|    The following steps assume that the default load resistance of 50 ohms is
|    used.  There may be devices that will not drive a load of only 50 ohms
|    into any useful level of dynamics.  In these cases, use the manufacturer's
|    suggested (nonreactive) load and add the load subparameter to the [Ramp]
|    specification.
|
|    The ramp rate does not include packaging but does include the effects of
|    the C_comp parameter; it is the intrinsic output stage rise and fall time
|    only.
|
|    The ramp rates (listed in AC characteristics below) should be derived
|    as follows:
|
|    a. If starting with the silicon model, remove all packaging.  If starting
|       with a packaged device, perform the measurements as outlined below then
|       use whatever techniques are appropriate to derive the actual, unloaded
|       rise and fall times.
|
|    b. If: The Model_type is one of the following: Output, I/O, or
|           3-state (not open or ECL types);
|           Then: Attach a 50 ohm resistor to GND to derive the rising edge
|                 ramp.  Attach a 50 ohm resistor to POWER to derive the
|                 falling edge ramp.
|
|       If: The Model_type is Output_ECL or I/O_ECL;
|           Then: Attach a 50 ohm resistor to the termination voltage
|                 (Vterm = VCC - 2 V).  Use this load to derive both the
|                 rising and falling edges.
|
|       If: The Model_type is either an Open_sink type or Open_drain type;
|           Then: Attach either a 50 ohm resistor or the vendor-suggested
|                 termination resistance to either POWER or the vendor-
|                 suggested termination voltage.  Use this load to derive both
|                 the rising and falling edges.
|
|       If: The Model_type is an Open_source type;
|           Then: Attach either a 50 ohm resistor or the vendor-suggested
|                 termination resistance to either GND or the vendor-suggested
|                 termination voltage.  Use this load to derive both the rising
|                 and falling edges.
```

```
|
|    c. Due to the resistor, output swings will not make a full transition as
|       expected.  However the pertinent data can still be collected as
|       follows:
|           1)  determine the 20% to 80% voltages of the 50 ohm swing
|           2)  measure this voltage change as "dV"
|           3)  measure the amount of time required to make this swing "dt"
|
|    d. Post the value as a ratio "dV/dt".  The simulation tool vendor
|       extrapolates this value to span the required voltage swing range in
|       the final model.
|
|    e. Typ, Min, and Max must all be posted, and are derived at the same
|       extremes as the V/I curves, which are:
|
|       Ramp rates for CMOS devices:
|          typ = typical voltage, typical temp deg C, typical process
|          min = minimum voltage, max temp deg C, typical process, minus "Y%"
|          max = maximum voltage,  min temp deg C, typical process, plus  "Y%"
|
|       Ramp rates for bipolar devices:
|          typ = typical voltage, typical temp deg C, typical process
|          min = minimum voltage, min temp deg C, typical process, minus "Y%"
|          max = maximum voltage,  max temp deg C, typical process, plus  "Y%"
|
|       where nominal, min, and max temp are specified by the manufacturer of
|       the part.  The preferred range is 50 deg C nom, 0 deg C min, and
|       100 deg C max temperatures.
|
|       Note that the derate factor, "Y%", may be different than that used
|       for the V/I curve data.  This factor is similar to the X% factor
|       described above.  As in the case of V/I curves, temperatures are
|       junction temperatures.
|
|    f. During the IV measurements, the driving waveform should have a
|       rise/fall time fast enough to avoid thermal feedback.  The specific
|       choice of sweep time is left to the modeling engineer.
|
| It is expected that this data will be created from silicon vendor
| proprietary silicon-level models, and later correlated with actual device
| measurement.
```

Appendix B: .SLM-File Specification

Summary

This appendix contains a combined sample file / specification for HyperLynx's version of the single-line connector model (.SLM) format. The .SLM format can be used to create single-transmission-line models for connectors. This format is fully compatible with the .SLM files provided by Amp, Inc. for their single-line models.

See Chapter 6, section "Modeling a Transmission Line as a Connector" for details on how to use .SLM files in LineSim.

Example File and Specification

```
* EXAMPLE2.SLM  ******************************************
* HyperLynx single-line connector-modeling format
* Files must have a file extension of .SLM
* Files are fully compatible with AMP, Inc. .SLM models
*
* EXAMPLE CONNECTOR MODEL FOR AN ISA BUS CONNECTOR
**************************************************************
* COPYRIGHT 1996, by HyperLynx, Inc.
* Catalog Number: EXAMPLE2
* Part Number:    0001
**************************************************************
* ROW A  Single t-line model
.SUBCKT EXAMPLE2A-L
R1 1 1 0.010
T1 1 1 1 1 Z0=51.0 TD=0.080NS
```

```
.ENDS EXAMPLE2A-L
* ROW B  Single t-line model
.SUBCKT EXAMPLE2B-L
R1 1 1 0.010
T1 1 1 1 1 Z0=48.0 TD=0.090NS
.ENDS EXAMPLE2B-L
* ROW C  Single t-line model
.SUBCKT EXAMPLE2C-L
R1 1 1 0.010
T1 1 1 1 1 Z0=51.0 TD=100PS
.ENDS EXAMPLE2C-L
****************************************************************
*
***************** Format specification ************************
*
* COMMENTS:
*  Lines that begin with * are comment lines.
*
* COPYRIGHT:
*  COPYRIGHT is optional.
*  Consists of a comment line including the text 'COPYRIGHT'.
*  The text is case insensitive.
*  Everything on the line after and including 'copyright'
*  is displayed to the user.
*
* CATALOG NUMBER:
*  CATALOG NUMBER is optional.
*  Consists of a comment line including the text 'catalog n'.
*  Therefore any of the following are allowed: 'Catalog Num',
*  'CATALOG NOM', 'CATALOG NUMBER:', etc.
*  The text is case insensitive.
*  Everything on the line after and including 'catalog'
*  is displayed to the user.
*
* PART NUMBER:
*  PART NUMBER is optional.
*  Consists of a comment line including the text 'part n'.
*  The text is case insensitive.
*  Everything on the line after and including 'part'
*  is displayed to the user.
*
```

```
* .SUBCKT:
*  .SUBCKT is required.
*  More than one .SUBCKT may included.  For example, each
*  pin on the connector could have a .SUBCKT record, or more
*  typically each row in the connector could have a .SUBCKT.
*  The beginning of a connector model is indicated by a line
*  which starts with a '.' in the first column followed
*  immediately by 'SUBCKT'.  The .SUBCKT record must
*  included a name that ends with the character 'L'.  This
*  convention is used to indicate it is a transmission _L_ine
*  model (not a lumped-element model).
*  Any characters after the name are ignored.  For example:
*  '.SUBCKT testL 1 2'  is valid.
*
* R:
*  R is optional.
*  Represents the DC resistance of the connector from 'in' to 'out'.
*  The line must have the character 'R' in the first column.
*  There must be at least 4 parameters on the line separated
*  by spaces.  The 2nd and 3rd parameters are ignored.
*  Examples:
*     'R1 1 1 0.010'  is 10 miliohms
*     'R4 1 2 10k'    is 10000 ohms
*
* T:
*  T is required.
*  Consists of the transmission-line model used to represent the
*  connector.
*  The line must have the character 'T' in the first column.
*  There must be at least 7 parameters on the line separated
*  by spaces.  The 2nd, 3rd, 4th, and 5th parameters are
*  ignored.  The 6th and 7th parameters must begin with either
*  'TD=' or 'Z0='.  Standard units are excepted.
*  The impedance and delay are the two critical parameters.
*  Examples:
*      'T1 1 1 1 1 Z0=50 TD=40PS'   is 50 ohm, 40 picoseconds
*      'T4 1 2 3 4 TD=0.01NS Z0=34' is 34 ohm, 10 picoseconds
*
* .ENDS
*  .ENDS is required.
*  This represents the end of the SUBCKT.
```

Appendix C: .FBD-File Specification

Summary

This appendix contains a brief specification for HyperLynx's ferrite-bead-modeling (.FBD) format. The .FBD format is used to create the files BSW.FBD and USER.FBD, which together define the ferrite-bead models available to LineSim.

See Chapters 4 and 5 for details on ferrite beads in LineSim.

Specification

```
********************** Ferrite Bead Models **********************
* Copyright 1996, HyperLynx, Inc.
*
* Description of format:
* ----------------------
* {MANUFACTURER=<name>}
* - Sets the manufacturer name for all ensuing beads
* - <name> is limited to 12 characters; truncated if longer
* - Applies until superseded by another MANUFACTURER record
* - Must be a MANUFACTURER record before the first BEAD
*
* Example:
* {MANUFACTURER=DALE}
* ---------------------
* {BEAD=<name>
*  (R_DC=<resistance>)
*  (PT1=<freq>, <impedance>)
*  (PT2=<freq>, <impedance>)
*  (PT3=<freq>, <impedance>)
* }
* - BEAD=<name>  names the bead model; <name> is limited to 30
*   characters
* - R_DC=<resistance>  gives the bead's DC resistance
* - PTx=<freq>, <impedance>  gives a frequency/impedance pair
*   for the bead
* - <freq> numbers can be scaled by a trailing 'M' or 'MHZ' to
*   give values in MegaHertz
* - Exactly 3 freq/Z pairs are required, as follows:
*   - The first point should be at (10-20)% of the nominal
*     impedance (nominal impedance is usually at 100 MHz)
*   - The last point should be the highest frequency shown in
*     the impedance vs. frequency graph for the bead
*   - If the resonant point is available, it should be PT2.
*     If the resonant point is not available, the 2nd point
*     should generally be the impedance at the nominal bead
*     impedance indicated by the vendor (usually 100 MHz).
*     If the nominal impedance and frequency are not specified,
*     then the 2nd point should be taken from the mid-point
*     of the Z-vs-freq graph for the bead
*
* WARNING: To provide accurate modeling, the data must following
*          the above rules.
*
* For example:
* {BEAD=ILB-1206-19
*  (R_DC=0.035)
*  (PT1=6.0MHZ,   4.0)
*  (PT2=100.0MHZ,19.0)
*  (PT3=500.0MHZ,27.0)
* }
*****************************************************************
```

Appendix D: Table of Dielectric Constants

Summary

This appendix lists dielectric constants (i.e., relative permittivities) for a number of common printed-circuit-board materials.

Table

Material Name	**Fiber Material**	**Bulk Material**	**Dielectric Constant**
Rogers RO2800			2.9
Rogers RO4003			3.38
Rogers RO4350			3.48
FR-2	paper	phenolic	4.3
FR-4	fiberglass	epoxy	4.8
FR-5	fiberglass	epoxy	4.8
G-2	staple-glass	phenolic	5.1
G-5	fiberglass	melamine	7.3
G-7	fiberglass	silicone	3.9

Material Name	**Fiber Material**	**Bulk Material**	**Dielectric Constant**
G-10	fiberglass	epoxy	4.8
G-11	fiberglass	epoxy	4.8
XXPC	paper	phenolic	4.1
N-1	Nylon	phenolic	3.6
GORE-PTFE	expanded PTFE/glass	epoxy	2.8
Alumina			10.0

Appendix E: Setting Simulation Options

Summary

This appendix describes several advanced simulation options, available in LineSim's user interface, that affect which algorithms LineSim runs during simulation.

HyperLynx recommends against changing these options except under special circumstances. You should contact HyperLynx for technical support before changing the default settings

Advanced Simulation Options

To see the advanced simulation options:

1. From the Options menu, choose Preferences. The Options dialog box opens.
2. Click on the Advanced tab. A dialog box opens, warning that you should not change the option settings.
3. Click Yes to proceed.

BoardSim Options — *Do Not Apply to LineSim*

The options in the BoardSim area apply only to the BoardSim product, and have no effect on LineSim. LineSim-only users should ignore these options completely. For details on them, see the BoardSim User's Guide.

Combine Line Segments…

When this option is enabled, LineSim combines adjoining transmission lines that have the same impedance into a single, longer line. This optimization results in a faster transient simulation.

High-Accuracy Field Solver *(LineSim Crosstalk Only)*

When this option is enabled, the field solver is forced to run with higher-than-normal spatial resolution. The default (non-high-accuracy) mode has been set to give excellent accuracy for nearly any problem on which LineSim would normally run. However, in rare cases (e.g., extremely narrow trace separations), the higher-accuracy mode may be preferable. However, high-accuracy mode runs substantially slower than normal mode, so should be enabled only when necessary.

Use Field-Solver Cache *(LineSim Crosstalk Only)*

LineSim Crosstalk’s field solver normally uses a smart “cache” to prevent having to re-solve cross sections which it has already encountered and analyzed. Disabling this feature turns off the cache, which may degrade the product’s performance. Since there is rarely (if ever) any reason to disable the cache, it is advisable to never disable this option.

Maximum Line-Length Tolerance

This percentage tells LineSim how large an error (as a percentage of the segment’s total delay) it can make in modeling a segment’s propagation delay. This tolerance applies only to very short segments, and even for short segments, is rarely invoked by LineSim.

Max DC Convergence… *and* Min DC Convergence…

The Max DC Convergence Iterations threshold tells LineSim the maximum number of times it can re-simulate to find a driver’s initial DC voltage. Such iterative simulation is only required when there are multiple IBIS (any model) or .MOD/.PML bipolar IC models present on a net. The default value is almost always sufficient to stabilize a multi-driver DC simulation; in very rare cases the number of iterations should be increased to give more-accurate results.

The Min DC Convergence Threshold tells LineSim how tightly each driver's voltage in a multi-driver DC simulation must converge before the DC simulation can safely be considered "successful." The default value should never be changed unless a circuit and combination of drivers has difficulty converging, and might benefit from a "relaxed" convergence criterion.

Note: *The DC convergence parameters are advanced parameters. If you find a circuit which you believe would benefit from a change in the parameters, contact HyperLynx for advice.*

Restoring Default Settings

To restore the default settings for all of the advanced simulation options:

1. From the Options menu, choose Preferences. The Options dialog box opens.
2. Click on the Advanced tab. A dialog box opens, warning that you should not change the option settings. Click Yes to proceed.
3. Click the Restore Defaults button.

App Note: Creating IBIS Models

Summary

This application note discusses how to create an IBIS model. Most or all of the information here can be gleaned from the IBIS specification, contained in Appendix A. However, you may find this discussion easier to read and follow than the specifications alone.

Detailed Note

.IBS models are not difficult to create. They are based on data which can be collected from real devices, or derived from proprietary silicon models. Whatever the data-collection method, the resulting .IBS model will accurately describe the device's behavior without giving away proprietary information about the silicon's design.

Before attempting to create a .IBS model, you should read the IBIS V2.1 specification, contained in Appendix A. The specification is also available in a text file: "IBIS21.TXT" in the main LineSim directory.

Elements of a .IBS Model

The following elements are required in a .IBS model:

- "default" package R, L, and C
- pin/signal list
- model for each unique I/O type

Models include:

- model type (I/O type)
- component capacitance
- power-supply voltage range
- V-I table for pull-up stage, pull-down stage, GND clamp diode, and Vcc clamp diode (whichever apply)
- rising/falling slew rate

The following elements are optional in a model:

- package R, L, and C for individual pins
- model polarity; enable polarity
- input thresholds

The next few sections describe the model elements in detail.

“Default” Package R, L, and C

The “default” package resistance, inductance, and capacitance specify a range of values for modeling the device package. Typical data is required; min and max data are optional. If only typical data is available, the package R, L, and C can be thought of as the “average” data for the device package.

LineSim converts the R, L, and C values into an equivalent transmission line -- a good model for the IC’s bond-out structure and package pins.

LineSim provides the additional flexibility to set the package parameters to 0.0. This indicates that the data was unavailable; LineSim will omit the package-model transmission line and simulate with the silicon data only.

Semiconductor vendors can determine R, L, and C with high-accuracy test equipment or electromagnetic modeling. End users can get package data from the vendor, use data for a similar package, or set R, L, and C to 0.0 and omit package modeling entirely.

Note: *In most situations, package R, L, and C have only a minor effect on simulation. Don't be afraid to omit them if you have no data.*

Example of package R, L, and C:

```
[Package]
        | variable      typ          min          max
        R_pkg           250.0m       225.0m       275.0m
        L_pkg           15.0nH       12.0nH       18.0nH
        C_pkg           18.0pF       15.0pF       20.0pF
```

Note: *IBIS keywords (delimited by square brackets []) must begin in the first column of the actual .IBS file.*

Pin/Signal List

The pin/signal list is a list that associates device signal names and pin numbers with simulation models. There are simulation models for each unique kind of driver or receiver on a device. Typically, though, many signals will share a single model, e.g., all the address lines on a device might have the same driver structure.

See the IBIS specification (Appendix A) for details on length limits for names, etc.

Example of pin/signal list:

```
[Pin]           signal_name    model_name    R_pin   L_pin   C_pin
|
1       RAS0#           Buffer1
2       RAS1#           Buffer2
3       EN1#            Input1
4       A0              3-state
5       D0              I/O1
6       RD#             Input2
```

7	WR#	Input2
8	A1	I/O2
9	D1	I/O2
10	GND	GND

Note: *See "Package R, L, C for Individual Pins" below for details on R_pin, L_pin, and C_pin.*

Model Type

Within a simulation model, the model type specifies whether the signal is a driver, a receiver, or bi-directional.

If bi-directional, LineSim allows the model's Buffer Direction to be specified as either a driver or a receiver.

Example of model type:

```
Model_type      Output
```

Note: *The IBIS specification defines two model types, 3-state and Open_drain, which are not supported by the modeling constructs in V1.1.*

Component Capacitance

The component capacitance specifies a model's silicon die capacitance. It should not include the capacitance of the device package. Typical data is required; min and max data are optional.

LineSim allows the component capacitance to be 0.0, if no data is available.

Example of component capacitance:

```
| variable     typp           min            max
C_comp        12.0pF         10.0pF         15.0pF
```

Note: *Values of 4-5 pF are common for component capacitance and would be a reasonable approximation in the absence of better data.*

Power-Supply Voltage Range

The power-supply voltage range defines the power-supply tolerance over which a model's data is valid. This should be the same range over which the model's V-I curves and slew rates were measured. All data is required, though the min and max values can be the same as typical if only a nominal voltage is supported.

LineSim allows you to set a driver's supply voltage to a variety of common increments, but only inside the range specified.

Example of power-supply voltage range:

```
| variable          typ     min     max
[Voltage range]             5.0V    4.5V    5.5V
```

V-I Tables

The V-I tables define the V-I curves of the pull-up and pull-down structures in an output driver and the clamp diodes (if any) to Vcc and GND. A table can be omitted if the corresponding structure doesn't exist (for example, the pull-up table for an open-drain output, or the Vcc-side clamp diode for a 74F receiver.)

Typical data is required; min and max data are optional. The IBIS specification disallows more than 100 points in a table, although LineSim doesn't bother enforcing this. Linear interpolation is used to find values that lie between points in the table. Currents for voltages outside the table are assumed equal to the last point in the table.

Note: *The last point — that currents for voltages outside the table are assumed to be equal to the last stated current precludes creating a purely resistive driver by using data points 0,0 and some other V1,I1. For voltages greater than V1, LineSim will still use current I1. Be sure you specify V-I points all the way out to where the current begins to saturate.*

LineSim allows changing between best-case, typical, and worst-case signal specs if max, typical, and min currents are all specified. See Chapter 9, section "What IC Operating Settings Mean" for details. LineSim requires at least two points in a V-I table.

Voltages in pull-up and Vcc-clamp-diode tables are relative to Vcc, not ground. See the IBIS specification (Appendix A) for more details.

End users can collect V-I data in the lab by a variety of means. (Don't be afraid to use an apparatus as simple as a multimeter, power supply, and current-limiting resistor.) Since driver pull-up and pull-down structures cannot be completely isolated on a real device, some approximations must be made for the turning-off portion of each stage's curve. Don't bother with many data points in regions where a curve is fairly linear; LineSim will automatically interpolate.

There is some risk of damaging a device while taking clamp-diode data. If you have any information regarding the diode's fully-on "resistance," construct a table that has zero current until the correct turn-on voltage, then is linear with the slope dictated by the "on" resistance.

Example of V-I table (for a driver pull-up):

```
[Pullup]
|
|       Voltage           I(typ)    I(min)    I(max)
|
        -5.0V             32.0m     30.0m     35.0m
        -4.0V             31.0m     29.0m     33.0m
|              .
|              .
         0.0V             0.0m       0.0m     0.0m
|              .
|              .
         5.0V             -32.0m    -30.0m    -35.0m
        10.0V             -38.0m    -35.0m    -40.0m
```

Slew Rates

The slew rates (or "ramp" rates) define the rise and fall times of a driver. Typical data is required; min and max data are optional. LineSim allows changing between best-case, typical, and worst-case signal specs if max, typical, and min slew rates are all specified.

End users can collect slew-rate data in the lab with a high-bandwidth oscilloscope. To remove package-related effects, slew the driver output into an open load. The slew rates are specified in the IBIS file as a convenient ratio of the delta-V and delta-T values measured on the scope. The IBIS specification recommends measuring between the 20% and 80% points of the driver's swing.

Example of slew rates:

```
[Ramp]
| variable     typ              min              max
dV/dt_r                 4.2/1.8n         3.5/2.5n         5.0/1.1n
dV/dt_f                 2.5/1.5n         2.0/2.3n         3.0/0.8n
```

Package R, L, C for Individual Pins

Individual signals can have their own, unique pin R, L, and C values. These appear in the signal/pin list, and can be specified for any or all of the device's signals. If a signal has no individual data, the default package R, L, and C is used.

LineSim allows the flexibility to specify any or all of the values for each signal. E.g., you can specify a signal-specific C, but leave R and L unspecified (as "NA"); the default R and L will be used.

See "'Default' Package R, L, and C" above for details on package modeling.

Example of individual package R, L, and C:

```
[Pin]  signal_name     model_name      R_pin   L_pin    C_pin
|
1      RAS0#           Buffer1         200.0m 5.0nH     2.0pF
2      RAS1#           Buffer2         209.0m NA        2.5pF
```

3	EN1#	Input1	NA	6.3nH	NA
4	A0	3-state			
5	D0	I/O1			
6	RD#	Input2	310.0m	3.0nH	2.0pF

Model and Enable Polarity

The model polarity specifies whether an output signal is inverting or non-inverting. The enable polarity defines whether an enabling signal is active-high or active-low.

Both of these constructs are read by LineSim but ignored. They are present in the IBIS specification for timing-analysis tools.

Input Thresholds

The input thresholds specify the voltages at which an input signal is recognized as a valid 1 or 0. They are presently not used in LineSim, but are used by BoardSim's Board Wizard to calculate timing delays.

App Note: Converting a SPICE Model to HyperLynx Databook Format

Introduction

Occasionally, you may have no model for a driver IC other than a SPICE model. This application note describes a relatively simple way in which you can convert a SPICE model — as long as you can run it in SPICE to extract some basic behavioral information — into a HyperLynx databook-format (i.e., “.MOD”) model.

Note: *If you cannot run the steps described below (because, for example, you do not have a SPICE package capable of compiling and exercising the model), another option is to get the semiconductor vendor who created the model to run the steps for you.*

Another means of converting from SPICE models exists — a university-written SPICE-to-IBIS translator — but HyperLynx believes that for customers, the method described in this application note is considerably easier and more likely to succeed than using the converter. The converter is not a commercial product; is poorly documented; has bugs; and is not trivial to understand or run. The method described here is relatively straightforward.

SPICE Writer Option

If the conversion process described in this application note is not possible or appealing, a different way of interfacing LineSim and SPICE models is by using HyperLynx’s SPICE Writer option. The SPICE Writer converts schematics in LineSim into detailed SPICE netlists. This enables you to use LineSim to automatically model all the details of your schematic — including impedance calculations — and end up with a netlist in SPICE to which you can attach SPICE IC models. In LineSim Crosstalk, the Writer can even generate netlists that include line-to-line coupling (currently supported for Hspice only).

For information on the SPICE Writer option, contact HyperLynx or your local reseller.

Requirements for Converting

In order to implement the method described in this application note, the following must be true:

- ♦ you have access to a version of SPICE that compiles and runs the model you want to convert
- ♦ you can embed the SPICE model in a simple SPICE test circuit that allows the model's output(s) to be switched high and low
- ♦ you can plot the waveform results of such a simulation so you can measure certain features of the waveforms

Access to a SPICE Simulator

Having a SPICE package and having one that compiles a particular SPICE model are sometimes different things. Some semiconductor-vendor-supplied SPICE models are developed to run in a proprietary in-house SPICE package, and may not compile in a commercial package. Others may require particular versions of Avanti's Hspice simulator. If you lack a suitable SPICE package, fax the semiconductor vendor who created the SPICE model this application note and have them generate the data for you.

SPICE Test Circuit

The SPICE test circuit required by this application note is simple: it entails adding only a few extra nodes to the model's SPICE deck. However, in order to understand where in the SPICE deck to add the new nodes, you need to know which existing nodes represent the input and output of the buffer you're modeling. In some cases, you may also need to tie certain other input nodes (e.g., an enable pin) high or low to make the buffer output respond to input stimuli.

If the SPICE model you're trying to convert is complex or poorly documented, request a schematic diagram or list of key nodes from the semiconductor vendor.

How This Method Works

The conversion method described below has a simple goal: find out enough about the behavior of the SPICE model to fill in the parameters in LineSim's databook (.MOD) model editor. If you can fill in the required parameters, then the editor will generate a .MOD model for you and you're ready to simulate.

To open the editor in LineSim, choose Databook IC Models from the Edit menu. (For more information on the editor and creating databook models generally, see Chapter 5.)

Many of the parameters in the editor are relatively easy to fill in: you can find them in a data sheet, or (in a few non-critical cases) even make reasonable guesses. But two important ones are harder to determine and must usually be found experimentally from the SPICE model:

♦ driver resistance (high and low)

♦ driver slew time (high and low)

Note: *If you happen to find resistance or slew time — or both — in a data sheet or by some other non-SPICE means, use them! This application note assumes that you lack both and have to resort to the SPICE model.*
If the data sheet contains I-V curves for the output buffer, you can convert the curves to equivalent resistances by drawing a straight line through the linear part of the curve (after it "turns on" but before it shows any saturation) and measuring the slope $\Delta V/\Delta I$.

Finding Driver Resistance from a SPICE Model

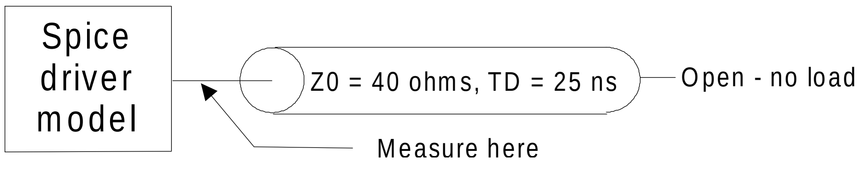
To make a SPICE model "reveal" a driving resistance, you can make use of a simple fact of electromagnetics: **when a device output drives a transmission line, the transmission line *initially* looks exactly like a resistance to ground or Vcc** (depending on whether the driver rises or falls). As soon as a reflection from the end of the transmission line travels back to the device, the line's behavior changes and becomes more complex, of course; ultimately, the line looks like an open circuit (i.e., presents no load to the driver).

But the transmission line's behavior *before any reflections occur* gives a simple and powerful method of finding a driver's resistance via simple voltage division. A transmission line's effective resistance during initial switching is equal to its characteristic impedance (Z_0). Therefore, if you load a driver with a transmission line of known impedance and measure — again, before any reflections occur — how much of the driver's voltage step actually appears in the transmission line, you can trivially solve for the driver's resistance (see below for formulas).

In practice, to isolate a driver's initial switching behavior into a transmission line from the behavior after reflections occur, you can simply use a very long transmission line. In the detailed steps below, for example, a 25-ns line is recommended. Any delay value that is longer than the entire amount of time for which you simulate in SPICE guarantees that your measurements will not be "polluted" by line reflections. Figure 1 shows the recommended transmission line and how it connects to the driver model's output.

Note that driving resistance (and slew time) must be found separately for the high and low stages of a driver, because the stages are often not balanced. (Usually, the low side has lower impedance.)

Figure 1: Connecting the Test Transmission Line to the Driver Model



Finding Driver Slew Time from a SPICE Model

Unlike driver resistance, driver slew time is not difficult to find — no "tricks" are required. Slew times can be measured directly from plots of the driver switching high and low into a transmission-line load. See below for details on making the measurements.

Finding Driver Resistance and Slew Time

To find a driver's high and low resistances and slew times:

First, add a test circuit to the SPICE model:

1. Open the driver's SPICE model in a text editor. Find the SPICE node representing the driver's output. Add a new line to the model that ties the driver output to a simple, lossless transmission line. Set the line's parameters to delay = 25 ns and characteristic impedance = 40 ohms. Leave the far end of the transmission line open.

   For example, if the driver's output node number is 10, add the following line to the SPICE model:

   ```
   TL  10  0  200  0  Z=40 TD=25ns
   ```

   This ties one end of a transmission line to the driver output node (node 10); creates a new, open node at the other end (node 200; use any unused node number); and sets delay and impedance as described above. The "0's" reference the line to SPICE's global ground node.

2. Open the driver's SPICE model in a text editor. Find the SPICE node representing the driver's input. Add a new line to the model that ties the driver input to a ramping voltage generator. Set the generator's ramp time to 1 ns; make it switch from the driver's low power supply to the high supply.

   For example, if the driver's input node number is 20, add the following line to the SPICE model:

   ```
   V  20  0  PWL  0ns 0  1ns  5
   ```

This ties a 0-ohm (ideal) voltage source to the driver input node (node 20), and switches the source in a linear ramp from 0V to 5V in 1 ns.

If the driver is inverting, then compensate by reversing the polarity of the input voltage source.

3. If the driver model requires certain other input nodes to be tied high or low in order for the output to switch, tie them to Vcc or ground as needed.

Then, run a simulation with the driver switching high, and plot the driver's output node:

4. Run a transient SPICE simulation. To ensure that your measurements are not "polluted" by reflections, set the total simulation time to be greater than the driver's switching time, but less than the delay time of the transmission line you added above in step 1.

   For example, if the driver model switches in about 3 ns and you used a 25-ns transmission line as recommended above, run the transient simulation for 10 ns. This is long enough to see the driver switch completely, but not long enough for reflections to return from the line's far end.

5. Plot the simulation voltage versus time at the driver's output node. (This is the voltage at the near end of the transmission line, not the far, open end.) Set up the plot so that the switching fills most of the horizontal scale, so you can make an accurate measurement of the switching time.

Then, from the plot, calculate the driver's high-stage resistance and slew time:

6. Calculate the driving resistance with the following formula:

   $R_{drv}\text{(rise)} = Z_0 (V_{final} / V_{step} - 1)$,

   where Z_0 = characteristic impedance of the transmission line (40 ohms),

V_{final} = final DC voltage swing of the driver,
V_{step} = voltage swing of the initial step into the transmission line
(ignore signs of the voltage values)

The “final DC swing” means how many volts the driver switches edge-to-edge after any transmission line reflections die out. For example, for a 5-V CMOS driver, this would be 5V; for a TTL device, about 3.6V.

The “initial step” swing is the lesser number of volts that the driver swings in the simulation plot. The value is less than the final DC swing because, initially, before reflections, the voltage divides between the resistance of the driver and the resistance (characteristic impedance) of the transmission line. See Figure 2.

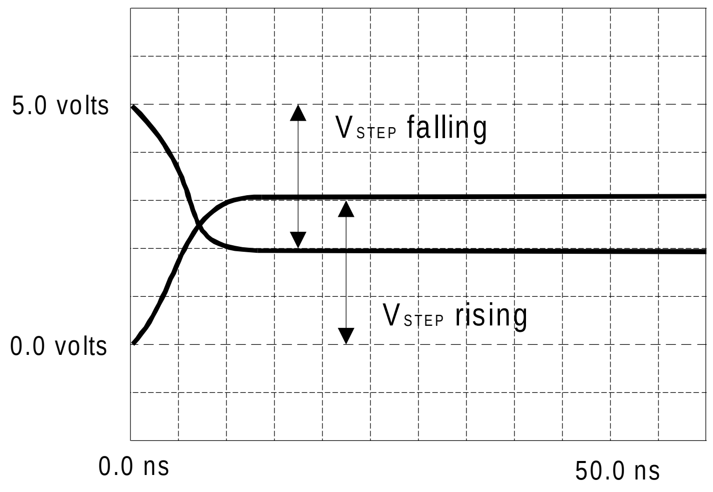
(There’s nothing mysterious about this formula — it’s a simple voltage division, solved for the unknown driving resistance.)

Then, calculate the slew time by measuring the amount of time to go from 10% to 90% of the plotted waveform voltage.

Finally, re-run the simulation with the driver switching low, and use the same methods to find the low-side resistance and slew time:

7. Alter the input voltage source to switch in the opposite direction. Run another simulation and plot the driver-output node (falling edge this time). Use the same formula as above to calculate the driving resistance; measure the 90% to 10% time as the slew time. Use absolute values of all quantities, i.e., ignore negative signs.

Figure 2: How to Measure Vstep



Finding Other Required Model Parameters

Driver resistance and slew time (high and low) are usually the only parameters that must be derived from the SPICE model. The remaining databook model values can generally be found in a data sheet, or in a few non-critical cases, can be guessed at. The following table summarizes how to find the other parameters in the .MOD model editor.

Parameter	**How to Find**
Type (high and low)	Pull down the combo box and choose the appropriate technology type. For a bipolar device, use the data sheet's circuit diagram to determine whether a given stage uses a silicon or Schottky-clamped transistor (unless ECL). Ignore "ramp"; use "open" for the high side of an open collector/drain.
Offset Voltage (high and low)	Set to 0.0V for any device that switches from supply rail to supply rail. For stages that do not switch to a rail, set equal to the amount by which the final voltage is offset from the rail, *minus one diode drop.* E.g., for 5-V TTL, high side

Parameter	**How to Find**
	offset is 1.4V – 0.4V (0.4 for the Schottky-diode drop). For most CMOS devices, use 0.0V, since switching is rail-to-rail.
Clamp Diodes, Type and Resistance (high and low)	Relatively unimportant for driver models. Can do a SPICE DC sweep above/below power-supply rails to find diode “resistance,” but not usually worth the effort. Instead, use standard “good guess” values: Type = Silicon, Resistance = 15 ohms.
Default Power Supply	Set to Vcc for all non-ECL devices; to Vee for ECL devices.
Capacitance	Get from data sheet (C_{out} or “output capacitance”). If not provided (rare), use 5 pF.

Creating the Model

Once you’ve collected all the required parameters, you can actually create the databook (.MOD) model by running LineSim’s model editor; entering the parameters; and saving the model under an appropriate name in user library. For details on this process, see the appropriate section in the User’s Guide or online Help system.

Once the model is saved, you’re ready to simulate.

Index