cadence™

Model Integrity User Guide

Product Version 23.1
September 2023

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

Contents

1

Introduction

Welcome to Model Integrity

Model Integrity is a high-speed design editing tool that helps you ensure the integrity of the model data required for high-speed circuit simulations. It allows you to create, manipulate, and validate models quickly in an easy-to-use editing environment. Model Integrity provides a model browser and syntax checker (parser) for models written in IBIS as well as for advanced models written in Cadence's device modeling language, DML. Model Integrity supports the following device model formats:

- Cadence DML
- IBIS 4.1
- Mentor/Quad XTK

Model Integrity supports an HSPICE-to-IBIS conversion module that assists in creating IBIS models from HSPICE simulation runs. Use the output of the HSPICE simulation run, IBIS, and buffer options file to quickly create an IBIS model to identify V-I and V-T tables for typical, maximum/minimum corner cases from the HSPICE run file. Similar functionality exists for translating Spectre buffer model simulations to DML.

Model Integrity Features

Model Integrity assists in reviewing and validating models that you create, including extracting buffer model files from SPICE and creating a complete IBIS file by combining buffer model files and a pinlist file.

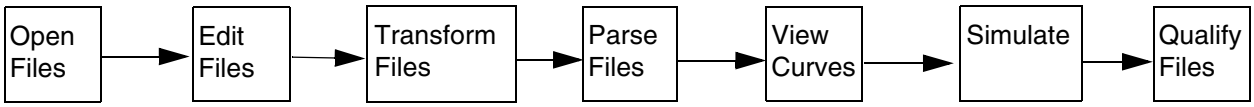
Table 1-1 describes the major features of Model Integrity.

Table 1-1 Feature List

Feature	Description
Multiple Model Files	■ Work with more than one file open; cut and paste amongst files; save files to new names.
Multiple Access Functionality	■ Access Model Integrity features in several ways from right-clicking on items in the Physical or Object view, selecting menu options from the Menu toolbar, or selecting an icon in the toolbar.
Object Viewing	■ View file contents as objects; sort by file or object property. ■ View model tables as both text and graphics. ■ Sort models by file, name, type, or other property.
Model Simulation	■ Parse files using the latest version of *ibischk* for IBIS files and *dmlcheck* for DML files. ■ Translate files using *ibis2signoise*, *quad2signoise*, among other translators provided in Model Integrity. ■ Simulate I/O buffers within IBIS and DML files. ■ Set and check file status and qualification stamps. ■ Wrap HSPICE and Spectre models into DML MacroModels ■ Wrap spice subcircuits into ESPICE models for use in SigXplorer
File Settings Editing	■ Set up options for translation, buffer extraction, and the IBIS Wizard.
Buffer Extraction	■ Extract IBIS buffer models from SPICE output file

The Model Integrity Flow

To ensure models are suitable for simulating, you can use Model Integrity to:



Not all file types can complete the entire flow. For more information about what you can do with different file types in Model Integrity, see Chapter 2, “Working with Files.”

Starting Model Integrity

Supported Unix Platforms

To start Model Integrity running on Cadence supported Unix-based platforms

➤ Type the following command within a Terminal window:

```
modelintegrity
```

Supported Windows Platforms

To start Model Integrity running on supported Windows platforms

1. Choose *Start – Run* from the desktop.
2. Type `modelintegrity` in the *Run* dialog box.
3. Click *OK*.

Allegro PCB SI

➤ Select *Tools – Model Integrity* from the menu bar.

Accessing Help

The Model Integrity information set consists of online books accessible from Cadence Help in both HTML and PDF formats. All documentation can be accessed from *Help – Documentation*.

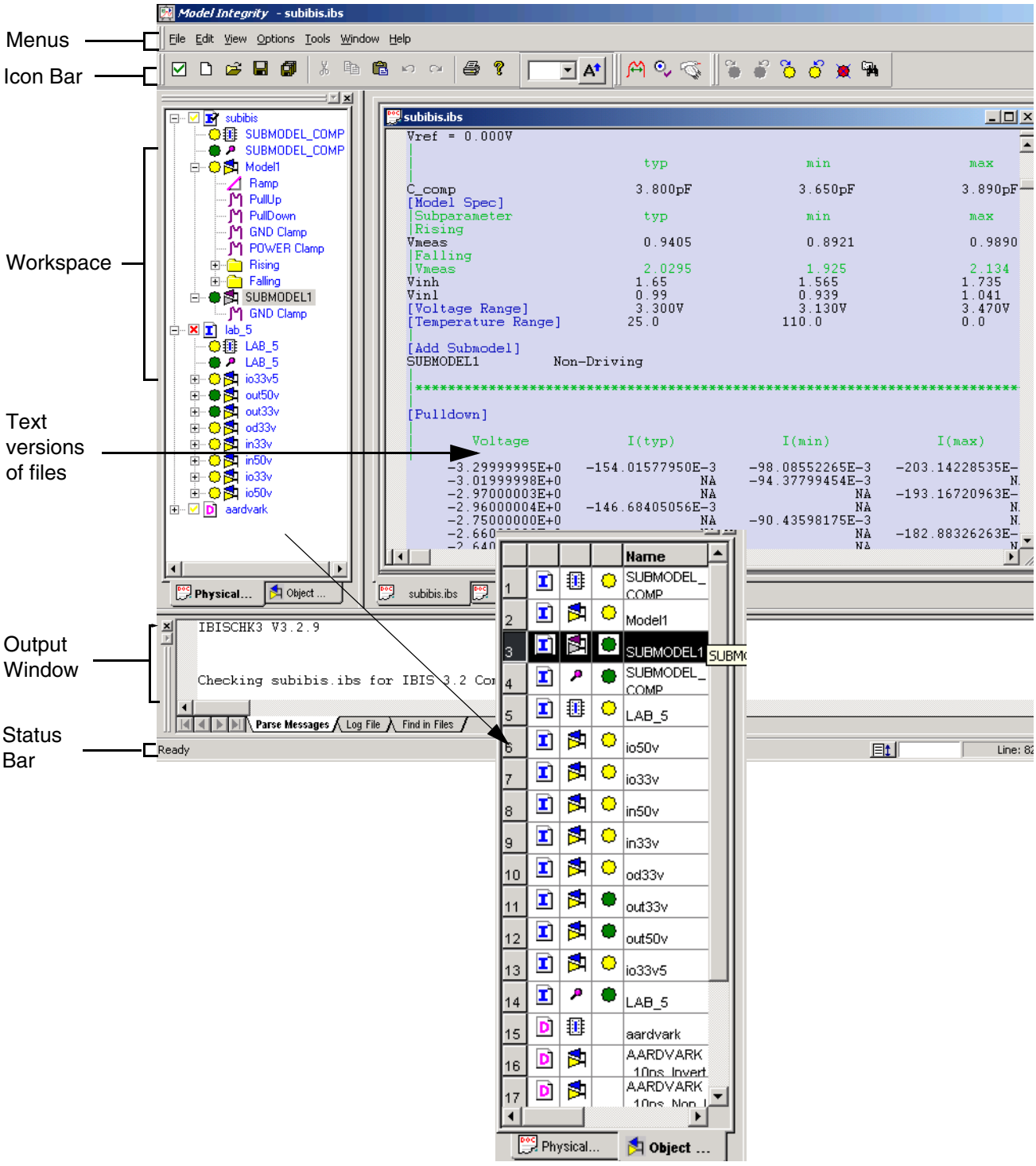
Refer to the	for this information
Model Integrity User Guide	This book is for users who want to know how to use Model Integrity in the high-speed design flow. This book complements the information in the *Model Integrity Command Reference*.
Model Integrity Command Reference	This book contains descriptions and procedures for all commands, organized by menu sequence. If you click Help in the dialog box, or if you highlight the menu command and press F1, the appropriate command description from this book appears. This book complements the information in the *Model Integrity User Guide*.

The Model Integrity User Interface

In Model Integrity, you work within the Physical and Object views, and the Edit and Output windows to manipulate the model file information and objects within the files. Menu options let you create a complete IBIS file, extract objects to a new file, wrap models, and simulate, parse, and create buffer models.

Note: Many Model Integrity menu selections are available only in context to the type of file that is active.

Figure 1-1 The Model Integrity User Interface



The user interface supports both right-button, context-sensitive menu drop-downs as well as a list of file type-specific commands under *Tools* in the menu bar of the main GUI. These menu items list the functionality available for the active file type. Figures 1-2 and 1-3 illustrate two examples of right-button, context-sensitive menu drop-downs; figures 1-4 and 1-5 illustrate examples of the same drop-downs in the Tools category of the GUI menu bar. File-specific functionality is documented in the following chapters.

Figure 1-2 Available Touchstone File Functionality in Context Menu

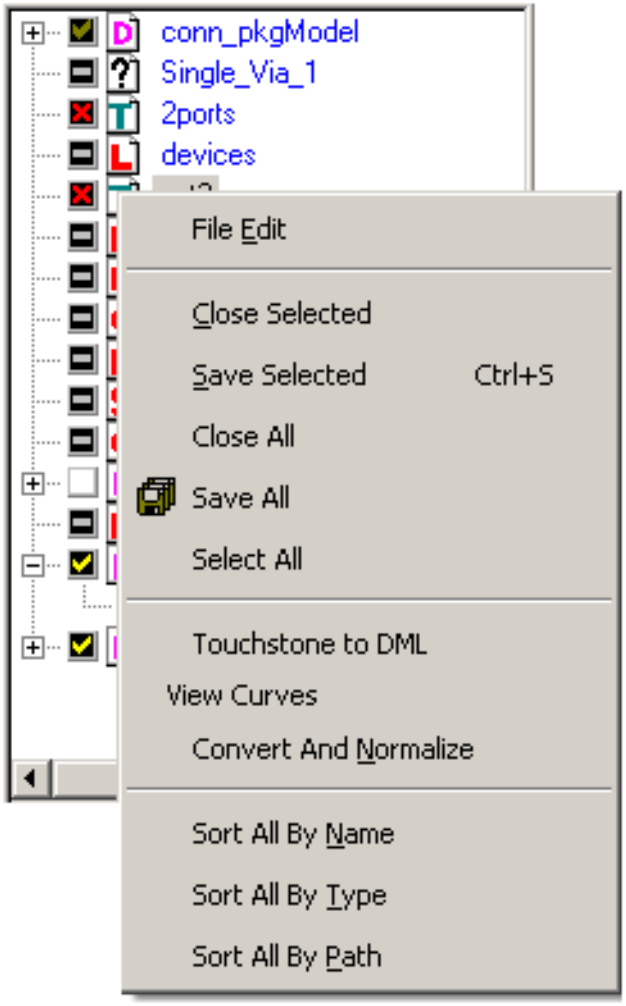


Figure 1-3 Available ESPICE File Functionality in Context Menu

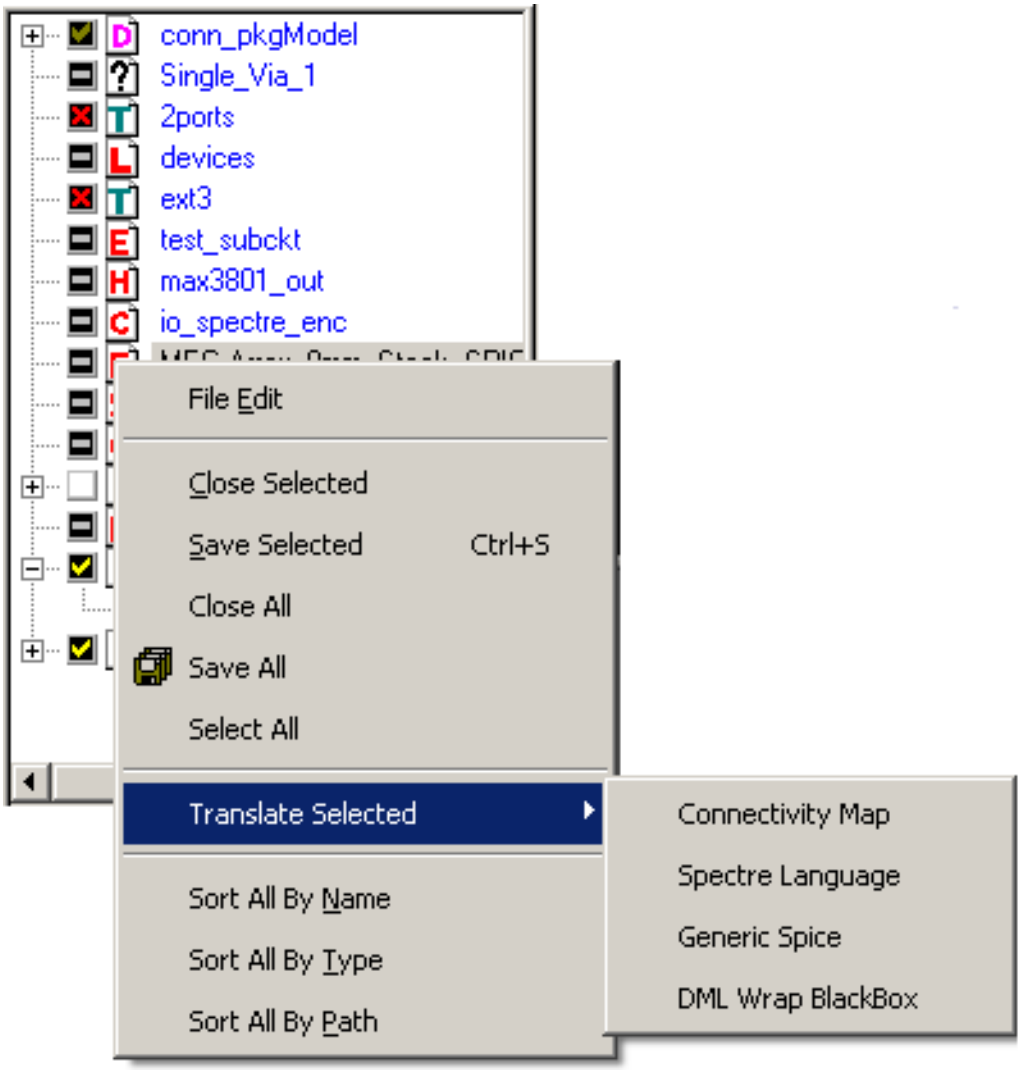


Figure 1-4 Available Touchstone File Functionality in Tools Menu

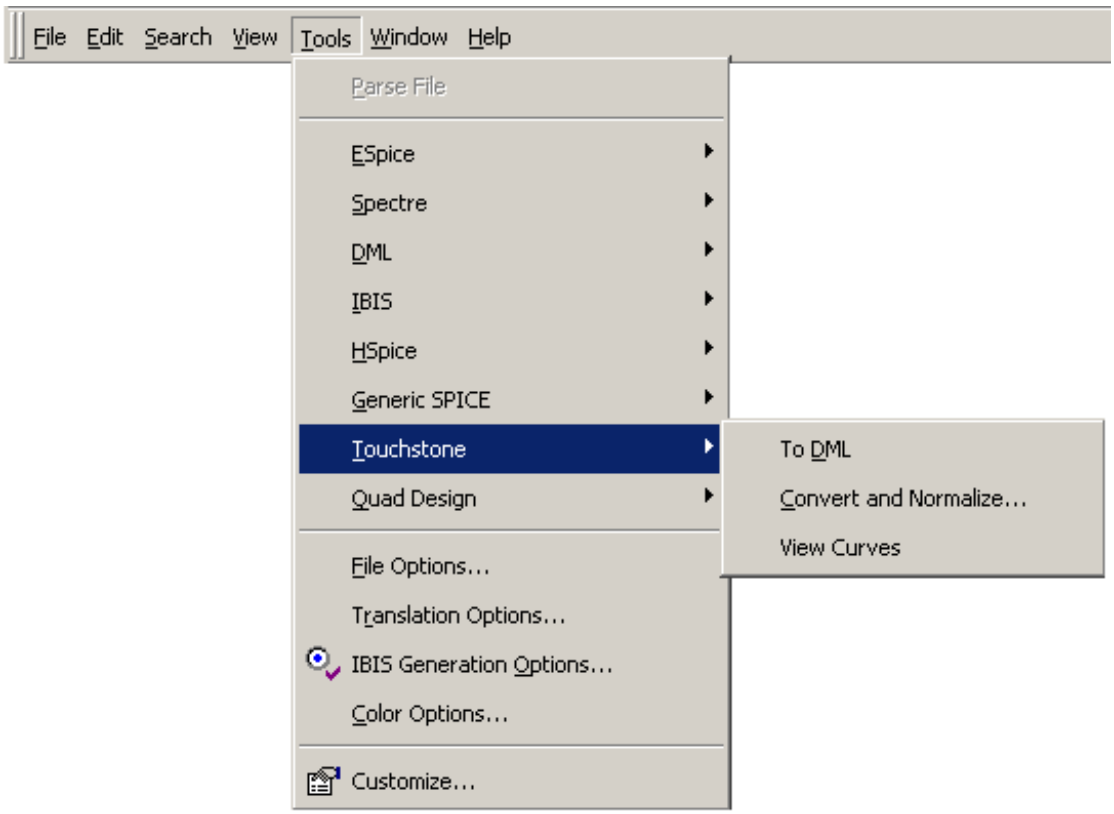
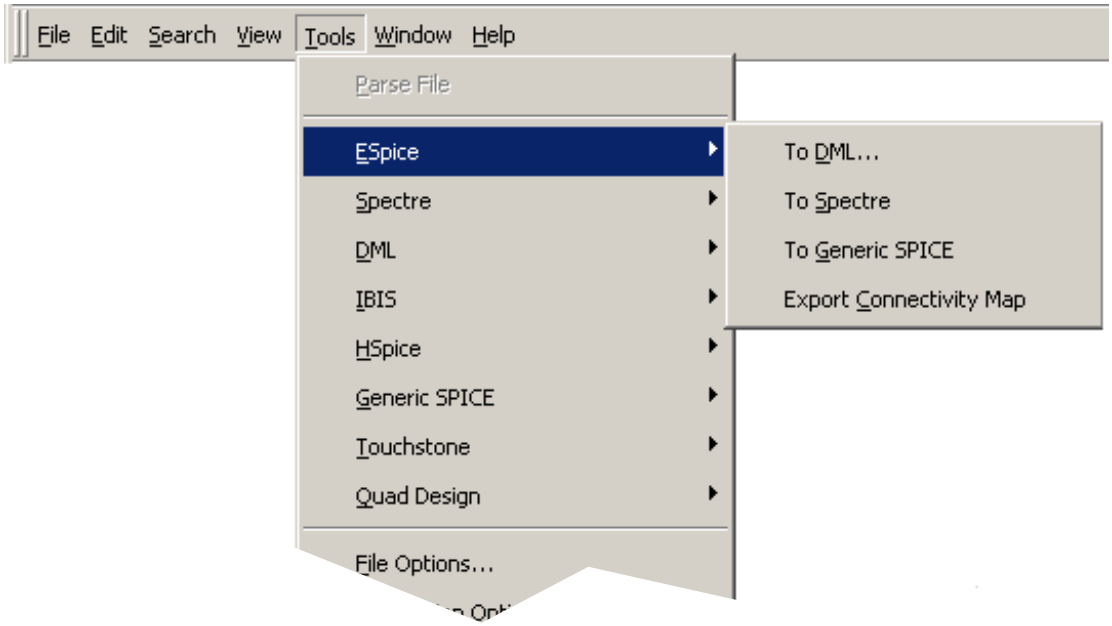


Figure 1-5 Available ESPICE File Functionality in Tools Menu



2

Working with Files

In order to perform operations on files in Model Integrity, the file you want to work with must be open and selected; that is, you may have multiple files open in MI, but you must select one specific file to edit it. If you have not opened any files, the operation selections listed for each file type are grayed out (that is, inactive). To activate a file type operation (for example, *Tools –Touchstone – To DML*), you must have a Touchstone file open *and selected* in Model Integrity. The only operations you *can* perform when no files are open in MI are:

ESPICE – To DML...

Spectre – Wrap Spectre Model in DML...

Spectre – DML Model from Simulation Results...

HSPICE – Wrap Hspice Model in DML...

Generic SPICE – To DML...

When you select any of these commands from the Tools menu, MI opens a dialog box from where you can browse for a Spice input file and a DML output file for the purpose of wrapping the Spice file into a DML-formatted file.

Once you open and select a file, Model Integrity allows you to edit it in a variety of ways. How you can edit the file depends on the type of file it is (as shown in Tables 2-1 and 2-2). The sections following Table 2-1 and Table 2-2 describe available functions for each file type. For detailed information on operations involving file transformation, see Chapter 3, “Transforming Files.”

Table 2-1 File Type/Activation Command Matrix (in Tools Menu)

File Type	Command	File Type to Activate Command
ESPICE	To DML	None
	To Spectre	ESPICE file
	To Generic	ESPICE file
	Export Connectivity Map	ESPICE file
Spectre	Wrap SPICE Model in DML	Spectre file
	DML Model from Simulation Results	None
DML	Mark Qualified	DML file
	Merge DML	DML files
	Make DML Index	DML files
	Rename/Reorder SPICE Pins	DML ESPICE black box file
	View Curve	DML Buffer file
IBIS	To DML	IBIS file
	ICM to Blackbox	ICM file
	ICM to PackageModel	ICM file
	ICM to ConnectorModel	ICM file
	Create Complete IBIS File	IBIS pin file
	View Curves	IBIS buffer file
HSPICE	Wrap HSpice Model in DML	HSPICE file
	To IBIS Buffer File	HSPICE output file
Generic SPICE	Wrap SPICE Model in DML	SPICE file
Touchstone	To DML	Touchstone file
	Convert and Normalize	Touchstone file
	View Curves	Touchstone file
Quad Design	To DML	Quad file

Table 2-2 Model Integrity Support Matrix

File Type	Text Edit	Transform	Parse	View Curves	Simulate	Qualify	Wrap
DML .dml	X		X	X	X	X	
IBIS .ibs .pkg .ebd .buf .pin	X	X	X	X	X		
Touchstone .s?p	X	X	X	X			
Quad .mod	X	X					
HSPICE Input .sp .cir	X	X		X			X
HSPICE Output .lis .sp	X X	X X		X			
ESpice .spc	X	X					X
Generic Spice .ckt	X	X					X
SpectreSpice .csc	X	X					X

The Model Integrity workspace (Physical and Object View) updates when a file extension is changed. Model Integrity re-evaluates the open file type based on the new extensions and

appropriately changes the file icons. Model Integrity removes any objects from the workspace associated with a file previously parse-able. The parsing status of the file changes to reflect whether you can parse the file. If you can parse the file, the file stamp, if current, sets the parsing status. See **Status Stamps** on page 20 for more information.

Note: Undefined file types display a question mark (?) in their respective icon in the Physical View window and do not allow for manipulation beyond standard text editing.

The following items describe in greater detail the type of file manipulation available with file types. Note that not all file types support every operation listed here.

- Parse Selected

  Displays information in the *Parse Messages* tab of the status window containing any syntax errors in the selected file. See Chapter 4, “Parsing to Qualifying Files” for details.

- ICM to BlackBox

  Outputs a `.icm` file in DML Espice BlackBox model format

- ICM to PackageModel
- Outputs a `.icm` file in DML Espice PackageModel format
- ICM to ConnectorModel
- Outputs a `.icm` file in DML Espice ConnectorModel format
- Make DML Index

  Outputs a `.ndx` file containing the names of the subcircuit package devices within the selected `.dml` file

- Merge DML

  Outputs a `MergeDML.dml` file that is a merger of all selected DML files.

- Spectre/HSPICE DML Wrapper
  - BlackBox

    Wraps Spectre or HSPICE model subcircuits into DML format.

  - MacroModel

    Outputs a 7 or 8-terminal Spectre or HSPICE subcircuit wrapped into DML format, supporting I/O Buffer and Differential I/O wrapping (the latter for 8-terminal subcircuits).

If Spice circuit port requirements are not met, the process does not complete and an alert message indicates which requirements were deficient.

Default setting of rising/falling dt is 100ps. If required, you can change these settings in the PCB SI or SigXplorer library.

- Spectre-to-DML Model Translation

  Outputs a Spectre buffer model in DML based on ASCII-formatted files in a specified output directory.

- Translate Selected (from ESpice)

  Translates the selected file into one of the following:

  - Connectivity Map
  - Spectre Language
  - Generic Spice
  - DML Wrap BlackBox

- DML Wrap BlackBox

  Wraps Spice subcircuits into DML format.

- Create Complete IBIS File

  Opens the IBIS File Creation Wizard. (See Creating a Complete IBIS File on page 34 for details.)

- Create IBIS Buffer File

  Invokes the SPICE-to-IBIS functionality by bringing up the Extract to IBIS Buffer File dialog box.

  Note: You must extract the models to an IBIS buffer file before creating a complete IBIS file.

- View Curves

  Opens SigWave and displays the curve points of the subcircuits in the selected file.

- Touchstone to DML

  Translates `.s?p` files to DML format for parsing using the `ts2dml` executable. (See Touchstone to DML on page 26 for details.)

- Convert and Normalize

Lets you edit network parameters and reference impedances in a Touchstone file. (See Converting and Normalizing Touchstone Files on page 32 for details.)

Status Stamps

When you edit, transform, and in other ways manipulate models, status stamps record the operation performed on the file. These stamps appear as text inside the file itself (not as a separate status file so you do not have to remember to move the status file along with the original. It also prevents you from inadvertently destroying a separate status file). The stamp consists of a commented block of text at the bottom of the file and begins and ends with a recognizable tag. Each line of the stamp is preceded with a file-specific comment, so it does not interfere with a file operation.

Note: When you modify a file, you must save it before a stamp appears.

All operations that Model Integrity performs—except SigWave—create stamps. Each of the following operations adds a status stamp to the file:

- Transform (`ibis2signoise, quad2signoise, spc2spc, ts2dml`)
- Parse (`dmlcheck, ibischck3, ts2dml`)
- Simulate (`modelsim`)
- Qualify (DML file only). See Qualifying Files on page 47 for more information.

A stamp appears in both the input and output files associated with an operation if that operation creates an output file.

The following example of a DML file shows a series of status stamps.

```
;******Modified by Cadence Design Systems*****
;[dmlcheck v15-0-40A]
;     EXECUTED_AT=   Mon Jan 27 16:2:0 2003
;     FILE_MOD_TIME= Mon Jan 27 16:02:04 2003
;     RESULT= 0 errors, 2 warnings
;     OPTIONS= -ab

;[modelsim AARDVARK_Nchan input]
;     VERSION= v15-0-40A
;     EXECUTED_AT=   Mon Jan 27 16:1:7 2003
;     FILE_MOD_TIME= Mon Jan 27 16:00:52 2003
;     OUTPUT= E:\tmp\mi.run\AARDVARK_Nchan_Input.sim
;     OPTIONS= RSeries= 50 Tperiod= 10n

;[mkdeviceindex]
;     VERSION= v15-0-40A
;     EXECUTED_AT=   Mon Jan 27 16:0:55 2003
```

```
;     FILE_MOD_TIME= Mon Jan 27 16:00:52 2003
;     OUTPUT= E:\tmp\aardvark.ndx
;***End Cadence Design Systems Modification***

;[Qualified]
          ;     VERSION= v15-0-40A
          ;     TIME=  Mon July 15 17:31:41
```

The stamp for each operation is unique, because the purpose of each operation differs. The following is a description of status stamps:

- OPERATION KEYWORD

  Model Integrity uses the keyword line to determine if a previously existing stamp should be overwritten by the new stamp. If the keywords match, the old stamp is overwritten. Keywords for most stamps are simply the operation name, for example, `ibis2signoise`. This means that one stamp of `ibis2signoise` is maintained in the file and each time `ibis2signoise` runs, this stamp is overwritten. The modelsim keyword contains more information so that one stamp is maintained for each buffer model simulated with each circuit, for example, [`modelsim A1G04_5_INP_5V input`]. The keyword for `ibischk4` and `dmlcheck` includes the version, so a stamp exists for every version used to parse the file. This way, you can compare the results between versions, for example, [`dmlcheck v15-0-40A`].

- VERSION

  Used only for stamps whose keyword does not include the version: `ibis2signoise`, `quad2signoise, spc2spc, mkdeviceindex, ts2dml, mergedml, modelsim`, qualified.

- OUTPUT

  Used only for operations that produce an output:

  `ibis2signoise, quad2signoise, spc2spc, mkdeviceindex, ts2dml, modelsim`

- RESULT

  Used only to record parsing results for `dmlcheck` and `ibischk4`.

- OPTIONS

  Used only when you specify an option in the Translation Options Editor for one of the translators: (`ibis2signoise`, `spc2spc` or `quad2signoise`). Also used to record `ibischk4` and `dmlcheck` options.

Below are some implementation details.

- The stamp for `mergedml` is included in the output file only. Therefore the `mergedml` stamp has an INPUT section but no OUTPUT section, since the file itself is the output file.
- The stamp for `modelsim` is included in the input file only, since the output files are not model type files, and in the case of SigWave, no comment character is available.
- The stamp entry in the file is not recorded in the Model Integrity Edit window undo and redo queue.
- The FILE_MOD_TIME is unaffected by the EXECUTED_AT time.
- If the file is read-only, no stamp is recorded in the file. However, Model Integrity issues a warning, and a note is added to the log file, which is displayed in the Output Window.
- When *Parse On File Open* is turned off in the Model Integrity File Settings dialog box, you use the `dmlcheck` or `ibischk4` stamp to re-establish the file-level parsing status only, if the stamp is more recent than both the file modification date and the Qualification stamp.

3

Transforming Files

Creating New Files

Transforming a file takes different forms, depending on the file type. Model Integrity allows you to perform these different functions:

- Convert a file

  Converting a file is a simple changing of one format to another without any accompanying changes (such as simulation results, etc.) to the data.

- Translate a file

  Translating a file involves performing calculations to data in one format, then converting the results of those calculations to a second format

- Merge a file

  Combine components from different files to create a new file

- Extract objects to a new file

  Extraction involves selecting specific data contained in one format and performing a translation on that data to another format.

- Wrap a file

  Wrapping involves adding specific data from one file to another file by enveloping the content of the data in the origination file around the content of the destination file without changing the format of the origination data.

Table 3-1 shows the type of transforming operations you can do in Model Integrity:

Table 3-1 Transformation Operations

Operation	Translate	Extract	Convert	Wrap	Merge
`spc2spc`	X		X		
`ibis2signoise`	X	X	X		
`quad2signoise`	X	X	X		
`spc2dml`				X	
`ts2dml`	X	X	X	X	
`MergeDML`					X

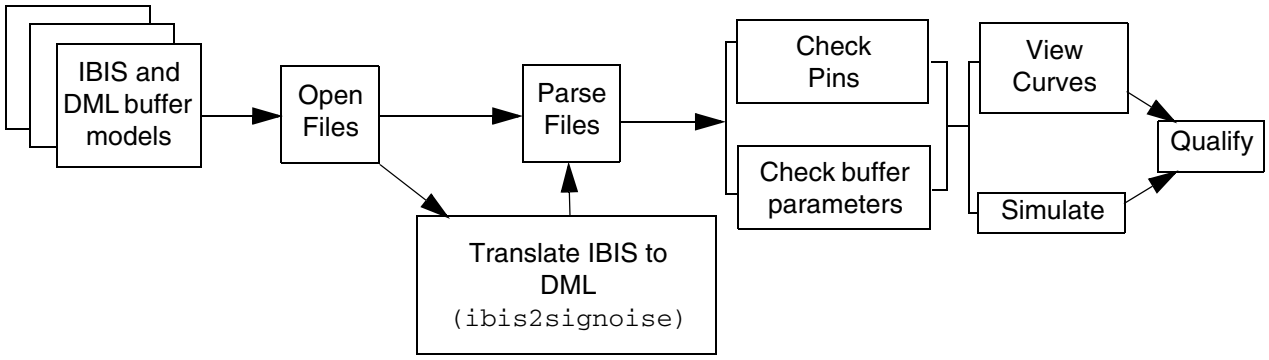
In addition, Model Integrity lets you transform SPICE to IBIS and provides full S-Parameters (scattered parameters) support.

Translating Files

All device models for Allegro SI are in DML format (The Device Modeling Language is a Cadence proprietary modeling language). For more information on DML, see the Allegro SI Device Modeling Language User Guide. You can translate files such as IBIS, Touchstone and SPICE into DML format.

For example, to qualify an IBIS file, you can first translate the file into DML. When you select an IBIS file, the IBIS to DML translation (*ibis2signoise*) menu option is visible. During the translation of an IBIS file into DML format, Model Integrity performs additional checks on the IBIS file formats.

Figure 3-1 Qualifying IBIS and DML buffer models



For more information on the IBIS flow, see IBIS Capabilities on page 42.

The menu options associated with file translation are visible based on the type of file selected in the Physical View or the active file in the Edit Window.

As you transform a model file, Model Integrity creates a new file in the same directory as the original along with an associated log file. The new file has the same root name as the original, but is given an appropriate extension. For example, `myfile.ibis` becomes `myfile.dml`.

IBIS to DML (ibis2signoise)

Model Integrity allows you to perform all available functionality on IBIS files except qualifying. Model Integrity uses `ibis2signoise` to transform IBIS files into DML format. After transformation, you can qualify the file.

QUAD to DML (quad2signoise)

Model Integrity transforms `.mod` files to DML format for parsing and qualifying using the `quad2signoise` executable.

ESpice to SPICE (spc2spc)

Model Integrity converts ESpice (Cadence version of SPICE) to generic SPICE in the SPICE to IBIS flow. Once the ESpice file is translated, you can use the IBIS File Creation Wizard to continue the translation to IBIS and then to DML to qualify your models. See Translating a buffer model from a file on page 28 for information.

SPICE Output to IBIS

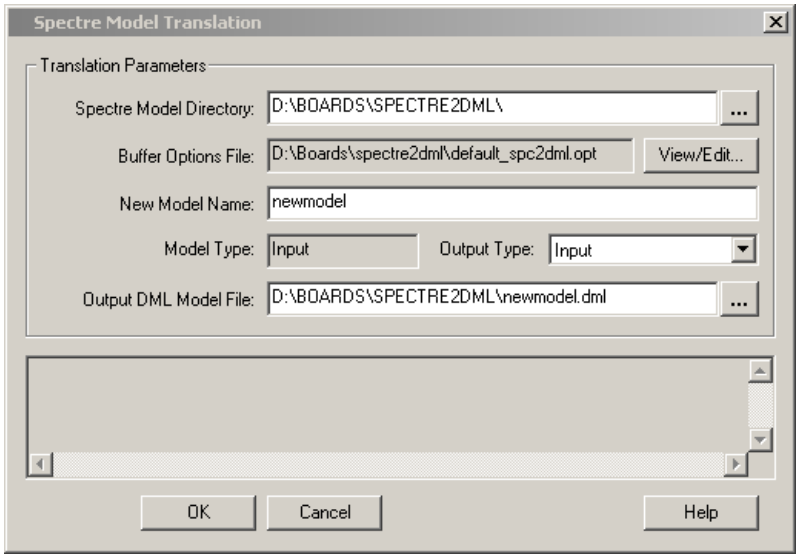
In the Model Integrity flow, you must run the SPICE simulations externally for each buffer. The *.`lis` file for the buffer can open in Model Integrity. Extract the buffer model data from the *.`lis` file and combine the extracted data and the IBIS options to create a new buffer model file.

Creating a DML Model From a Spectre Simulation

Model Integrity provides a complete environment that supports a Spectre-to-DML flow for buffer model generation. This functionality allows you to create highly accurate behavioral models for simulation in SigXplorer and PCB SI. The process parameters are based on

template and option files that contain all necessary model data. These files must be located in specific directories that you reference in the dialog box.

Figure 3-2 Spectre Model Translation Dialog Box



Prerequisites to Running the Translator

1. Edit the Cadence-provided template files, `template_iv.scs` and `template_vt.csc`, as necessary. Templates must reside in the Spectre model directory.

   Note: See Appendix A, "SPICE Templates," for complete details on working with Cadence template files.

2. After you have edited the template files, run them through Spectre from your operating system prompt.

   ```
   spectre <template_name.scs>
   ```

   Two subdirectories are created in your Spectre model directory, `VT_raw` and `IV_raw`.

   - `VT.raw` must contain the results for minimum, maximum, and typical VT data. These files are:

     `VT_max.tran`

     `VT_min.tran`

     `VT_typ.tran`

     `VT_max_info.info`

```
VT_min_info.info

VT_typ_info.info
```

- `IV.raw` must contain the results for minimum, maximum, and typical IV data. These files are:

```
IV_max.tran

IV_min.tran

IV_typ.tran

IV_max_info.info

IV_min_info.info

IV_typ_info.info
```

Translating a Spectre model to DML

1. Choose *Tools — Spectre — DML Model from Simulation Results*.

   The Spectre Model Translation dialog box appears.

2. Configure the process parameters from the controls in the dialog box, as described below.

Spectre Model Directory	Specifies the location of the source Spectre model data; specifically, the VT and IV simulation results for the model. These files must reside in `VT.raw` and `IV.raw` subdirectories in the selected model directory
Buffer Options File	Specifies a fully qualified pathname of the options file containing the required parameters for the buffer generation process. This is a read-only field that defaults to the current active file.
View/Edit	Opens the IBIS Options File Editor that allows you to edit the parameters of the buffer options file or select a different one.
New Model Name	Specifies the name of the new model to be exported to the DML file. The name represents the default model output file name as well as the model name within the file.
Model Type	Displays the model type (Input, I/O, etc.) specified by the data in the selected Spectre model directory field.

Output Type	Specifies the type of model you wish to create. The default selection is based on the input Spectre model, but the drop-down list contains all the possible output options.
Output DML Model File	Specifies the full pathname to the DML model file
Status field	Displays self-explanatory error and warning messages based on your input to the controls in the dialog box.
OK	Executes the Spectre-to-DML process
Cancel	Closes the dialog box without executing the process.

3. When you have set all parameters, click *OK*.

   You are prompted to view the results upon completion. If you elected to view the translated DML file, it opens in Model Integrity. (Based upon size, this process may take a few minutes.)

   Note: An `mi.run` directory containing log and parse files is created in the Spectre model directory.

Touchstone to DML

Model Integrity transforms `.s?p` files to DML format for parsing using the `ts2dml` executable.

Translating a buffer model from a file

1. In Model Integrity, open the *.`lis` file.
2. When the file icon appears in the left column

   right-click over the file name and select *Create IBIS Buffer File* from the pop-up menu
   –or–
   select *Tools – HSpice – To IBIS Buffer File*.
3. Check your IBIS options in the Extract to IBIS Buffer File dialog box, then edit and save any changes.

   Note: The dialog box offers the option of generating separate clamp data tables, in addition to pullup and pulldown curves. For Input, 3-state, and I/O buffer types, the clamp currents should be generated, since clamp current flows even when a buffer is not enabled (driving). Clamp currents are not normally used for Output, Open Sink, or Open Drain buffer types, although they are permitted by the IBIS Specification.

There is also a resistance option for identifying on-die resistors. Resistors smaller than this limit are identified, if they exist, and their currents included in the clamp current tables.

You select your preferred options using the check boxes and text boxes. You can view the extracted data tables as curves at this time. You can change your options and view the updated data tables.

1. When you are satisfied with the extracted tables, save the buffer model to a file. The default name is `<buffer>.buf`, but you can change it. You can change the default extension used in buffer model file creation in the File Settings dialog box (*Tools – File Options*).
2. Use the IBIS File Creation wizard to create an IBIS file before simulating.
3. Simulate the buffer model file. After using SPICE to simulate the buffer, Model Integrity extracts the SPICE data and formats it into a buffer model section for an IBIS file. Save it to a buffer model file (`*.buf or *.mr1`).

Parse the file, unless already done. Examine each data table, preferably by viewing the tables graphically. This identifies any significant non-monotonic data points.

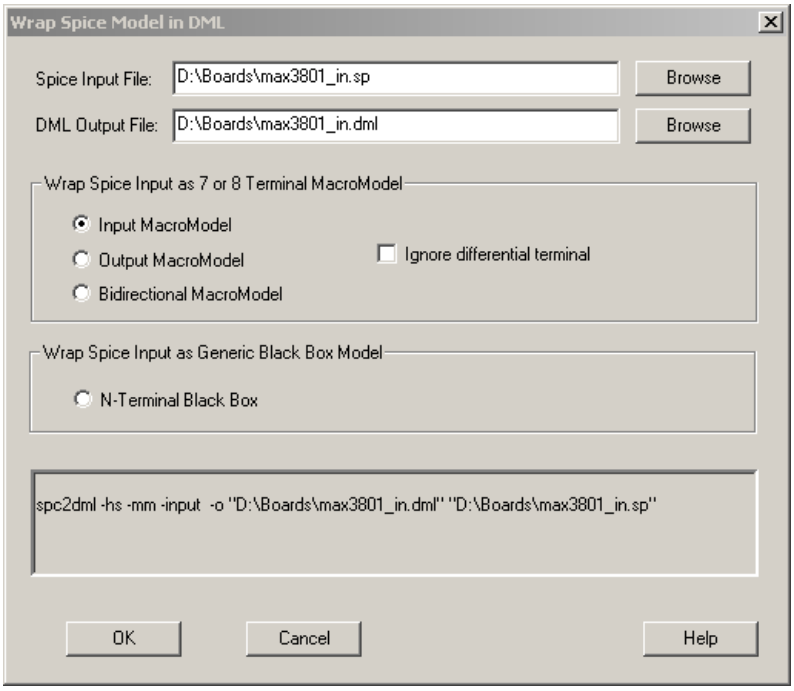
DML File Wrapping

Model Integrity lets you wrap SPICE subcircuits into a DML file directly from the user interface. You can perform this task in two fashions.

From the Wrap Spice Model in DML Dialog Box

With an appropriate SPICE file active, select *Tools – <file_type> – To DML* to display the Wrap Spice Model in DML dialog box.

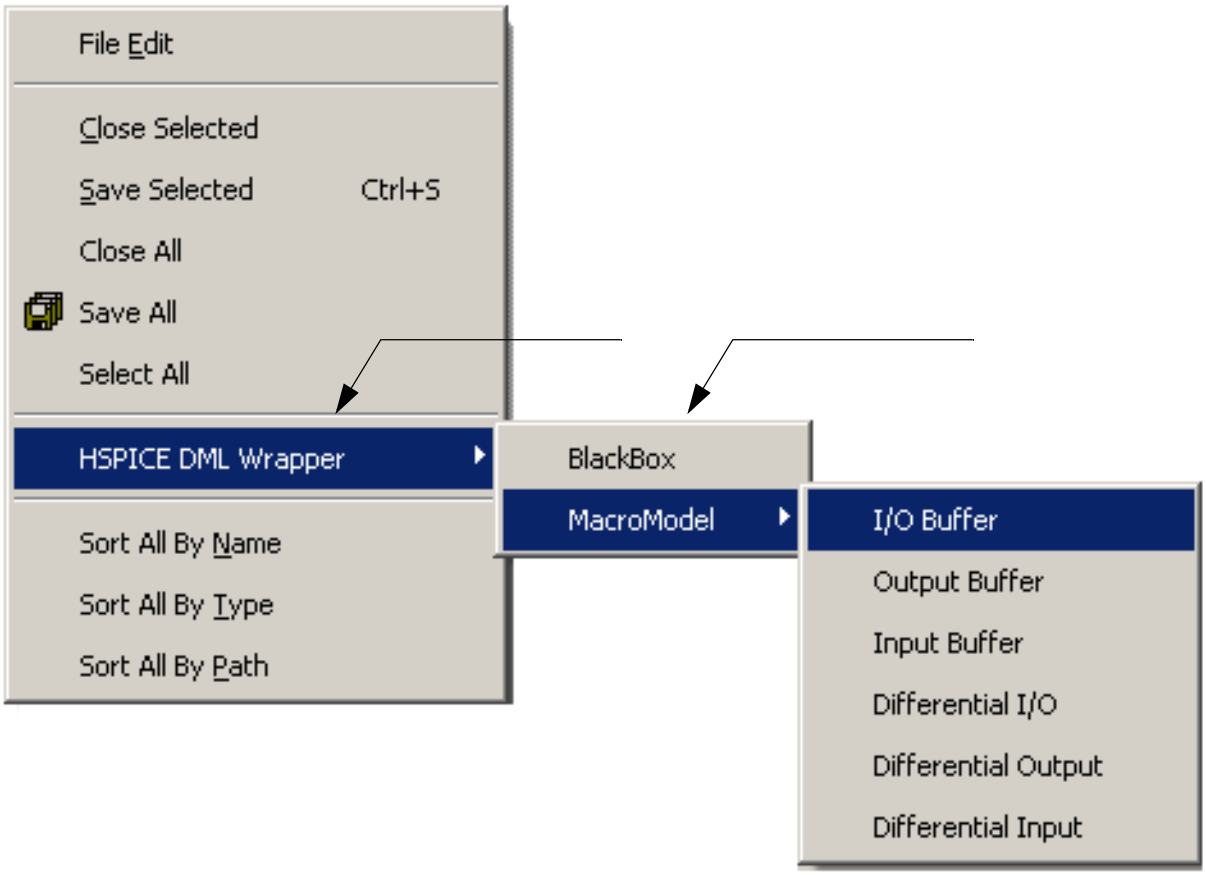
Figure 3-3 Wrap Spice Model in DML Dialog Box



From the File-Specific Context Menu

Right-click on the file icon of a 7 or 8 terminal subcircuit in the Physical View tab of the MI workspace, you can choose the *BlackBox* or *MacroModel* option, as shown in Figure 3-4. (Eight terminal subcircuits provide Differential IO type capability.)

Figure 3-4 HSpice/Spectre DML Wrapper — Context Menu Selection



When you run the *spc2dml* utility in either of the fashions described above, Model Integrity checks that all Spice circuit port requirements are met before completing the wrapping process. If conditions do not allow completion of the process, alert messages are displayed that describe which requirements were not met. As a default, rising/falling delay time is set to 100ps. If necessary to meet requirements, you may have to change the `dt` value within the text file or in your SI/SigXplorer library file.

In addition to MacroModel wrapping, you also can wrap generic Spice subcircuits into ESpice models that you then can use directly in SigXplorer. By either checking the *N- Terminal Black Box* control in the Wrap Spice Model in DML dialog box or right-clicking on the file icon of a generic Spice file in the Physical View tab of the MI workspace, you can choose the *DML Wrap BlackBox* option, as shown in Figures 3-5 and 3-6.

Figure 3-5 Black Box Control in Dialog Box

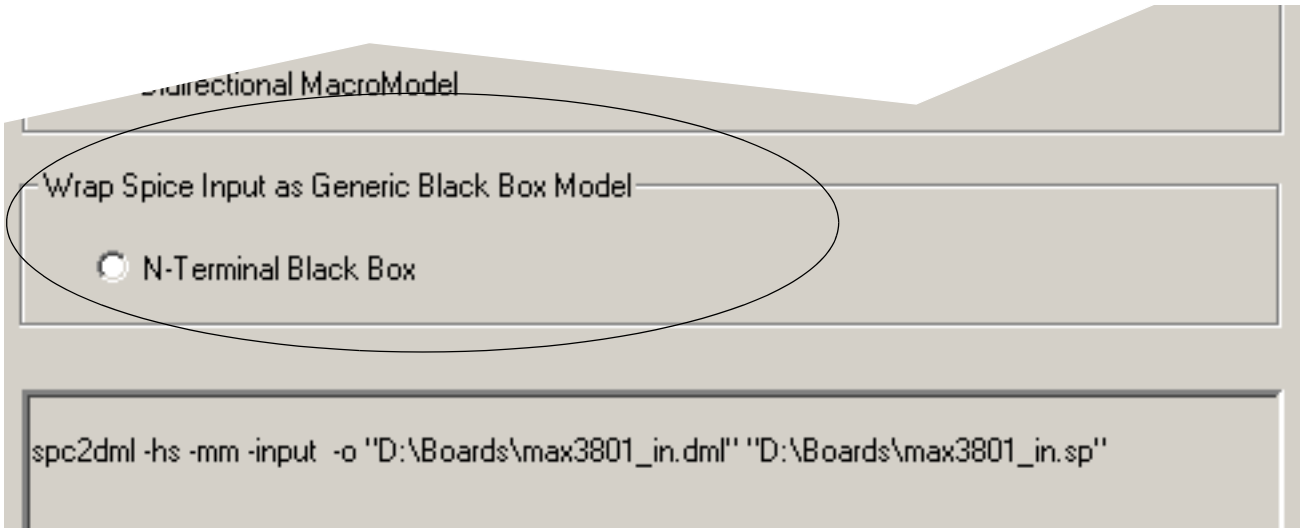
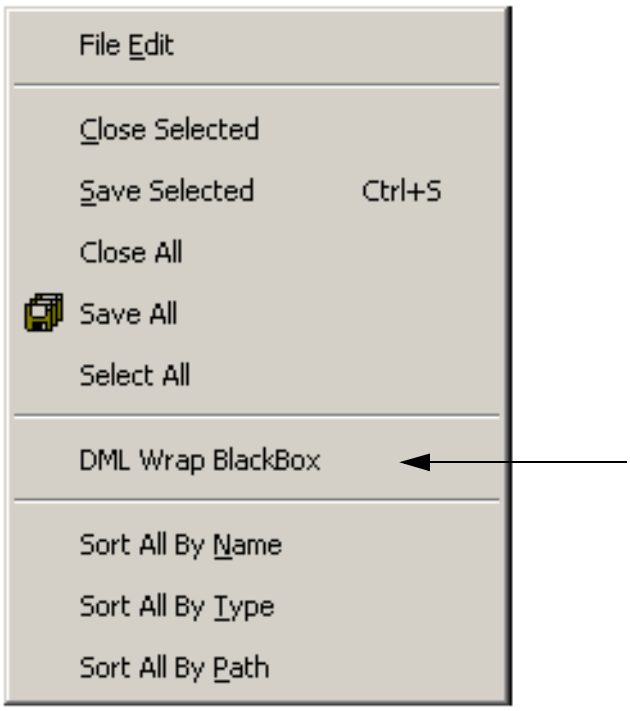


Figure 3-6 DML Wrap BlackBox— Menu Selection



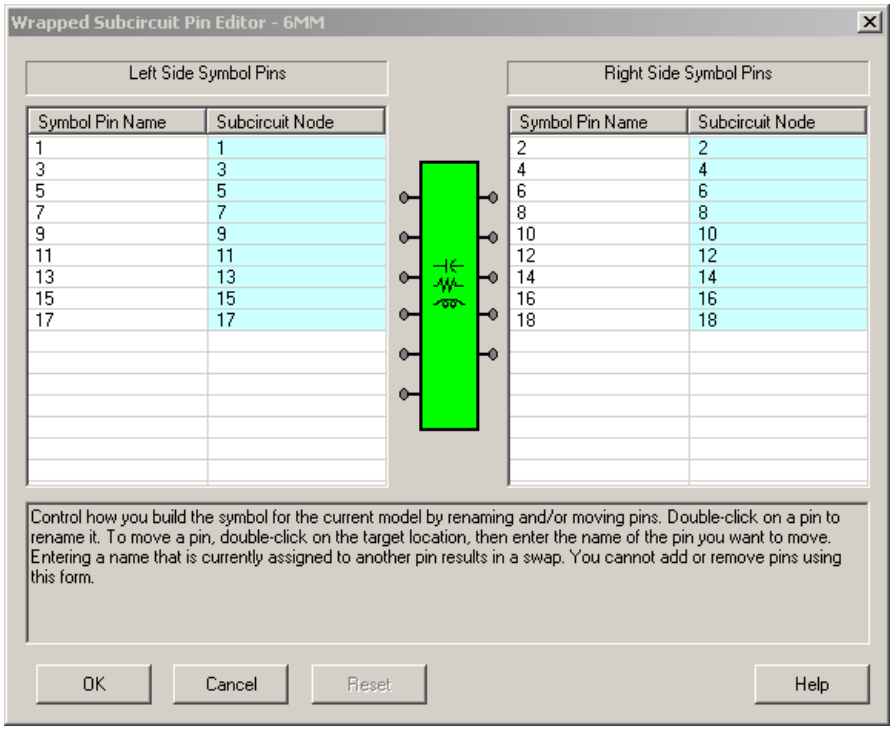
When you select this option, the top-level subcircuit in the Spice file is contained within a new **ckt.dml* file. An appropriate `Pin Connections` section is included in the new file.

In the case of Spice files that contains multiple top-level subcircuits, you are given the option—by way of a pop-up check list—of selecting the top-level subcircuits that you want included in the **ckt.dml* file.

Reordering and Renaming Pins in a DML ESpice Model

Designed primarily for symbols created in SigXplorer, pin editing of wrapped subcircuits lets you rename and/or reorder pins in the symbols of DML ESpice models. When you invoke *Reorder/Rename ESpice Pins*, either from the right-button context menu on a selected DML ESpice packaged device or from the *Tools – DML* submenu, the Wrapped Subcircuit Pin Editor appears.

Figure 3-7 Wrapped Subcircuit Pin Editor



SigXplorer creates symbols with pins that alternate left-to-right and top-to-bottom, distributed as evenly as possible. Pin editing lets you control on which side of the symbol the pins are located; therefore, reordering the pins in the models subcircuit statement.

Procedure

1. Select a DML ESpice symbol in the Model Integrity Physical View tree.

2. Perform either of the following:

   Select *Tools – DML – Reorder/Rename ESpice Pins*
   –or–
   Right-button click on the symbol icon and select *Reorder/Rename ESpice Pins*.

   The Wrapped Subcircuit Pin Editor appears, as shown in Figure 3-7.

Renaming a pin

   a. Select a pin for renaming by either double-clicking on the pin, or by highlighting it and pressing the space bar. You can also use cut-and-paste to rename a pin.

Note: The dialog box does not support drag-and-drop.

   a. Enter a new name for the selected pin.

Note: Unsupported characters (!, @, #, $, etc.) are blocked and duplicate names are not allowed. A pin name containing unsupported characters cannot be pasted into the pin editor. If the pin name you enter is identical to the name of an existing subcircuit node or symbol pin, the pin is moved (reordered) rather than renamed.

   a. At any point in this process, you can click *Reset* or press *Ctrl-Z* to undo any changes you have made.

Reordering a pin

   a. Double-click or press the space bar on your keyboard at the desired location.

   b. Enter the pin name.

      The named pin will be swapped from its original location to the new position.

   c. At any point in this process, you can click *Reset* or press *Ctrl-Z* to undo any changes you have made.

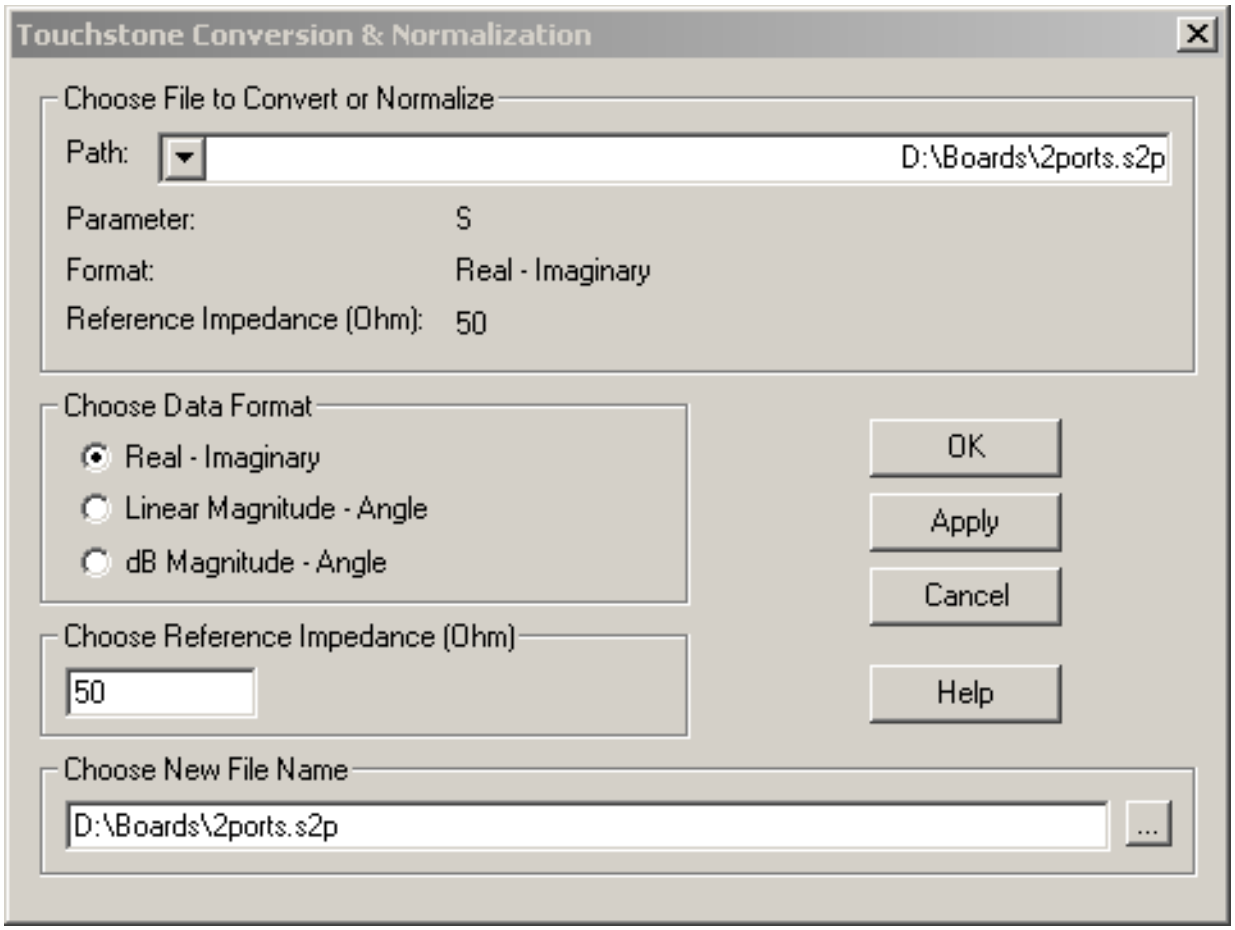
3. When you have completed editing pins, click *OK* to save your changes and close the dialog box.

Converting and Normalizing Touchstone Files

Use the Touchstone Conversion and Normalization dialog box to edit network parameters and reference impedances in a Touchstone file. The dialog is accessed from *Tools – Touchstone*

– *Convert and Normalize* or from the right-button context menu drop-down on an active Touchstone file icon.

Figure 3-8 Conversion and Normalization Dialog Box



Conversion occurs when you want to view the file data in another format by changing the parameter without altering the original data. Normalization occurs when you change the reference impedance but retain the original parameter.

You can perform normalization and conversion simultaneously. Click *Apply* after each change you make to keep the dialog box open. You can select and edit other Touchstone files without closing the dialog box by using the same method. As you create each new file, it opens in Model Integrity.

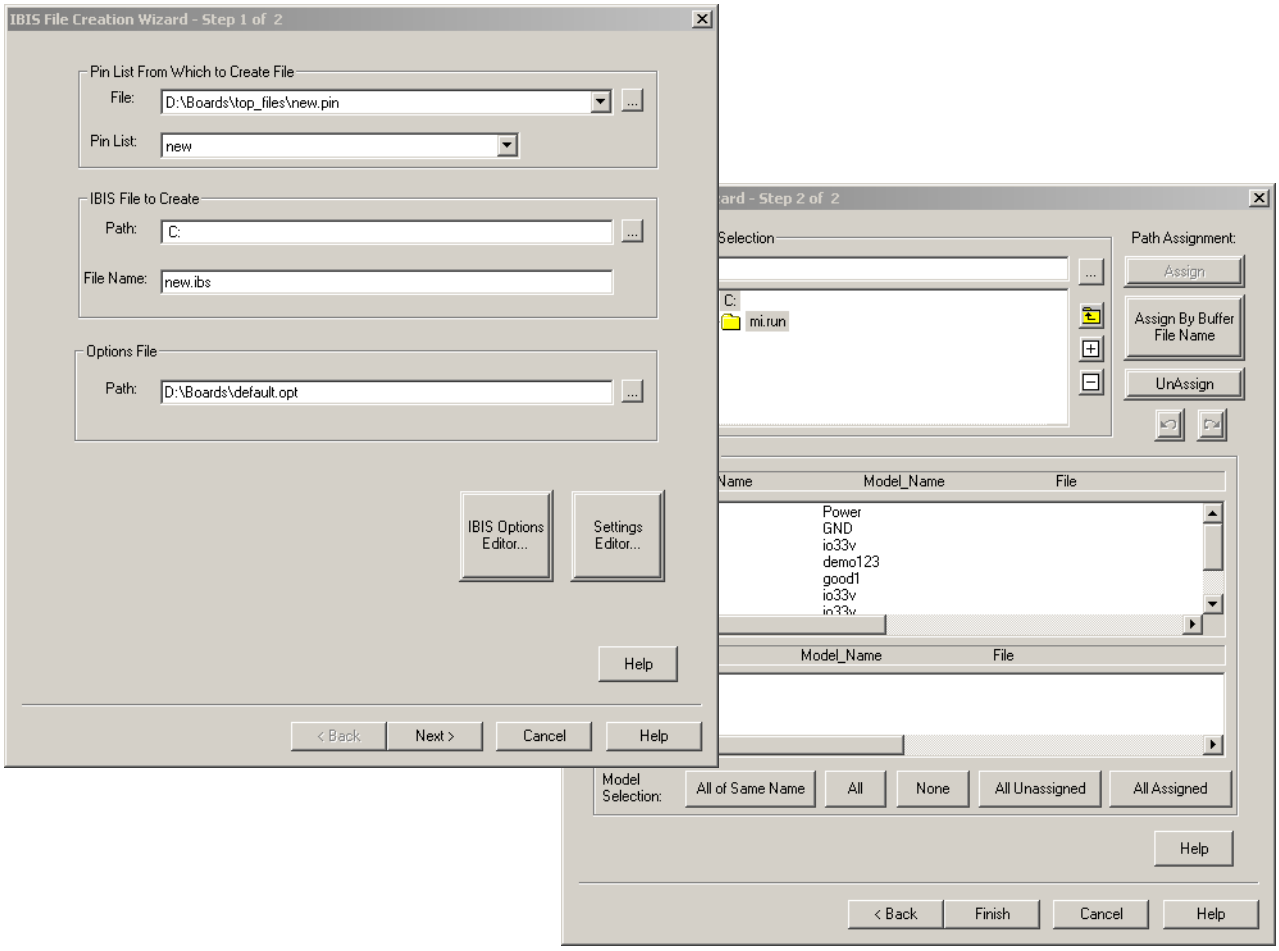
Creating a Complete IBIS File

Model Integrity lets you create IBIS models that are correct by construction. Model Integrity supports all IBIS buffer model types, including Open Source and Open Sink. You can create a new buffer model from an existing one, but it must be of the same type.

You can create a complete IBIS file by combining one pin list file and one or more buffer model files. The pinlist and buffer model files are concatenated. You build the IBIS header section using data from the IBIS Options file as shown in Creating an IBIS model on page 42.

You can use the IBIS file creation wizard (Figure 3-9) to combine objects, combine a buffer model and a pinlist to create a complete IBIS file that you translate to DML for parsing and qualifying the models.

Figure 3-9 IBIS File Creation Wizard



Note: All IBIS buffer models used in the file must already be available, either as separate buffer model files or within a complete IBIS file.

IBIS File Creation Wizard - Step 1 of 2

Options

Pin List From Which to Create File

Identifies the pin list used to create the file.

File — Lists the path of all IBIS files (.ibs) and pin list files (.pin) open in Model Integrity. If the file is not listed, use the browse button to search for new IBIS or pin list files.

Pin List — Lists all pin lists contained in the file specified in the *File* field.

IBIS File to Create

Identifies information about the IBIS file created by the wizard.

Path — Lists the path of the file to create. Use the browser to identify a new file path.

File Name — Displays the file name of the file you want to create.

Note: The created file must have an `.ibs` file extension to be recognized by the IBIS parser.

Options File

Identifies the options file used during file processing.

Path	Lists the path of the selected options file.
IBIS Options Editor	Displays the IBIS Options Editor populated with the current options file defined in the *Options File* area. You can choose a new options file in the editor.
File Settings Editor	Displays the File Settings Editor.

Buttons

Next	Continues to Step 2, once you have entered all the necessary information.
Cancel	Closes the window and aborts the IBIS File Creation Wizard.

IBIS File Creation Wizard - Step 2 of 2

Defines a file path for every model listed in the Pin List area.

Options

Buffer Model/Path Selection

Used to identify folders or IBIS models used for pin list assignment.

Folder	Lists the folder level name for the contents displayed in the tree.

UpFolder	Sets the path listed in the Folder edit box up one directory and populates the tree with the contents of the new path.
Expand All	Expands all files in the tree.
Collapse All	Collapses all files in the tree.

Path Assignment

Used to modify the assignment status of files in the list boxes.

Assign	Enabled when you select a model in the tree. Model Integrity assigns the selected model name and path to every model selected in the Pin List area.
Assign By Buffer File Name	Enabled when you select a folder in the tree. To use this button for additional assignments, a buffer model file in the selected folder must have the same name as the model selected in the Buffer Model Assignment area, and the model in the buffer model file must have the same name as the buffer model file. It is similar to the Auto Assignment function, except that you can select specific models, instead of assigning paths to all models visible in the Buffer Model Assignment area.
UnAssign	Clears the path assignment for every model selected in the Buffer Model Assignment area.
Undo	Resets the model and path for models most recently assigned or unassigned.
Redo	Resets back to the most recently assigned or unassigned models.

Pin List Information

Identifies all files in the pin list identified in Step 1, including the Model Selector files.

Note: The Pin, Signal_Name, and Model Selector areas remain unchanged throughout the file creation process.

Model Selector

Controls which models are selected in the list boxes.

All of Same Name	Selects all models with the same name, as the first selected model, from the list of assignable models.
All	Selects all assignable models in the list boxes.
None	De-selects all models.
All Unassigned	Selects all models that have not been assigned.
All Assigned	Selects all models that have been assigned.

Buttons

Back	Returns to the previous window, Step 1.

Finish	When a file path has been assigned to every model listed in the Pin List, click Finish. A complete IBIS file is created and entered into the Physical and Object views. It is not necessary to assign a file path for Power, GND, and NC.
Cancel	Closes the window and aborts the IBIS File Creation Wizard.

IBIS files contain not only the component object, but the pinlist and buffer model objects as well.

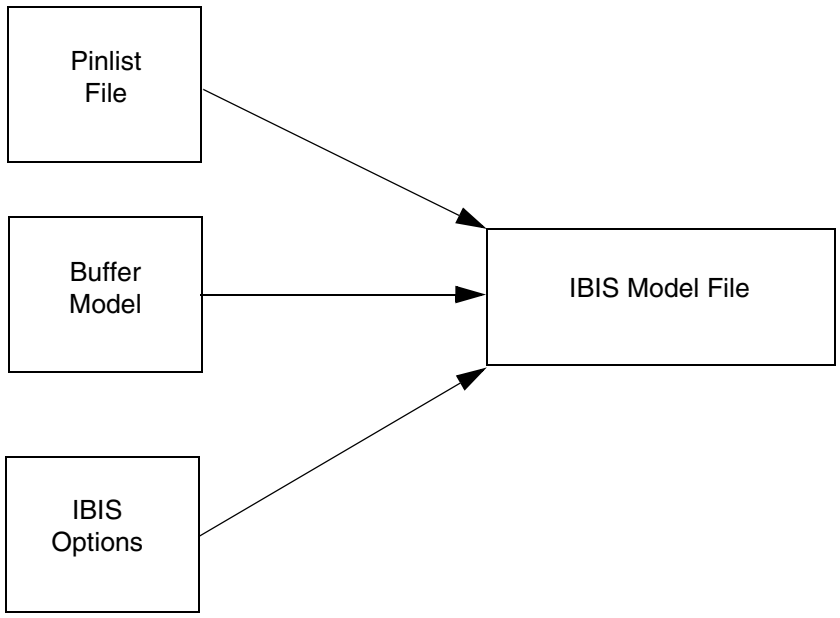
Before creating an IBIS component, you need to have a model file for each buffer used in the component. Additional buffer model files are required for programmable buffers, or buffers used under conditions not covered by the first buffer model. In Model Integrity, buffer models are also located in existing IBIS files.

Once Model Integrity creates the IBIS file, it opens the file in the Edit window. This lets you view the file and add any additional text to the Notes. You can run the IBIS parser on this file by right-clicking over the file name in the Physical view and selecting the parser from the pop-up menu.

Note: Since the file must be saved before parsing, you will be prompted to save the file. If you have deleted any text and do not save the edits, the text deletion will still take place but the tree/spreadsheet will not update, not even to remove the component object. The file parsing status will change to *File Not Parsed*. If you save the file, it will re-parse, and the tree/spreadsheet will update.

You can simulate the buffer model in the IBIS file in *tlsim* (transmission line simulator), if this was not done previously.

Figure 3-10 Creating an IBIS model



IBIS Capabilities

You can perform the following operations on a complete IBIS file:

- Use the navigation buttons to advance to the file line associated with a parser message.
- Translate the IBIS file to DML and view messages in the lower window.
- Parse with the `ibischk4` parser and view parser messages in the Output window.
- View a buffer model's data tables graphically.
- Simulate pre-defined IBIS test circuits, using *tlsim*.
- Extract any pinlist section or buffer model from a complete IBIS file to a new file.

  Note: An IBIS file is built from the buffer model's object, plus the pinlist object for the component.

Setting options

Choose *Tools – File Options* to set and change the preferred file extensions that Model Integrity recognizes.

Note: When you change the preferred extensions in the Model Integrity File Settings dialog box, the type of files that are already open are not updated.

Designating an Options file

You use the IBIS Options File Editor dialog box to set the options for IBIS and SPICE Output files in preparation for parsing and translating them to DML format.

If you do not designate an options file upon invoking Model Integrity, a warning message appears at startup. Model Integrity does provide a default options file, but options must be configured for your specific requirements. Using the IBIS Options dialog box, you set the parse and translation options for your models.

You may have more than one Options file, although only one can be loaded at any time. Options files can also exist for translators such as *ibis2signoise*. You should make sure you have loaded the correct options file and set any design-specific values before translating.

Tip

Multiple *.*opt* files can be saved, allowing you to have a separate *.*opt* file for each I/O interface or technology.

Model Integrity uses IBIS keywords to create complete IBIS files and buffer models. The ANSI/EIA 656-A IBIS specification requires a number of these keywords and subparameters. The Allegro SI simulator, *tlsim*, uses other keywords which are optional in IBIS, to make timing measurements. Use both types of parameters to generate IBIS buffer model files and complete IBIS model files.

Extracting a buffer model from an IBIS file

In Model Integrity, you can use the IBIS pinlist and buffer models from existing IBIS files. This allows you to clone component pinlist and buffer model sections in creating new IBIS files. The pinlist or buffer model sections may be used in place, or they can be extracted to a new file.

4

Parsing to Qualifying Files

You can parse IBIS, Touchstone and DML files to determine syntax errors. These are the only files that support parsing in Model Integrity. (See Translating Files on page 22 for more information.)

Note: Model Integrity uses the `ts2dml` executable for translating and parsing Touchstone files. For IBIS and DML files, place a new `ibischk` version in the hierarchy and overwrite the older one.

When you open a valid model file in Model Integrity, it is automatically parsed using the parser appropriate for the file type.

You can also request to parse an open file at any time while you work with it in Model Integrity. Once parsed, Model Integrity takes the following actions based on the result.

- Warnings and markers display at the bottom of the file's window (within the Edit Window) to mark any syntactical problems encountered in the file. An example for a Touchstone file is:

```
[ts2dml v15-2-28A]
     EXECUTED AT= Tues Jan 6 14:33:50 2005
     FILE_MOD_TIME= Wed Dec 03 10:34:16 2004
     RESULT= 0 errors, 0 warnings
     OPTIONS= -CHK
```

- Warning and error markers display in the Object view to mark any syntactical problems encountered within each model object contained in the file.

  Note: Touchstone files are the exception, as they do not contain any objects.

- The file's parse status displays or updates in the Physical View using a checkbox to the left of the file's icon.

The following table indicates how to interpret the color of the file level parse status mark in the Physical View.

This color	Indicates

Green (check mark))	The file is parsed successfully with no errors or warnings.
Yellow (check mark)	The file is parsed with warnings.
Red (x mark)	The file is parsed unsuccessfully with errors.
No mark (white box)	The file has not been previously parsed by Model Integrity. See Parse on File Open on page 46 for more reasons for no mark.

You can use any version of *ibischk* to check a complete IBIS or DML file. You should examine all warnings and errors in the error log under the Log File tab.

Parsing checks the model against the IBIS parser requirements, checking for syntax problems, missing models, and reporting non-monotonic table lines and mismatches between DC operating points and V-t endpoints. For more information, see the *Allegro PCB SI User Guide*.

Parse on File Open

If there are unparsed files in a multi-file selection, *Parse Selected* appears. A file level parsing status is not always indicative of a parsed file. If *Parse on File Open* is off in the Model Integrity File Settings dialog box, Model Integrity reads in the parsing file status from the qualifying stamp. Also, if *Parse on File Open* is checked, but the *ibischk4* failed, the file will not have a parsing status.

Tip

An object inside a file indicates a parsed file.

Viewing Curves

You can select data curves that are associated with a buffer model within Model Integrity and have them display in SigWave. *Tools – <file_type> – View Curves* or a right-click on a selected buffer model object in either the Physical View or Object View displays a context-sensitive pop-up menu enabling you to select the specific type of data curve you wish to view. Once a curve type is selected, SigWave launches and displays the selected curves.

Note: You use SigWave to overlay curves contained in the original model file and those produced by *ibis2signoise* in the DML file.

In cases where *SigWave* is already running, the currently displayed folders toggle off and a new folder containing the curves that you selected for viewing appear. A `.sim` file occurs every time you choose to display curves.

Tip

You should check curves in an IBIS model before running *ibis2signoise* or check curves in a DML file after running *ibis2signoise* or *dmlcheck*.

Note: Use the `-d` option to prevent changes made by *ibis2signoise*. However, if you use this option, *dmlcheck* will not run automatically.

You can view Touchstone file waveforms in SigWave in Cartesian, Polar Plot or Smith Chart format. If you select *View Cartesian*, SigWave opens with the Touchstone file you selected. SigWave creates a new folder for each file and allows you to select which network connections to view.

Opting to view waveforms in Polar Plot or Smith Chart invokes the View Touchstone Curves dialog box with the file path displaying the selected Touchstone file. From this dialog box, you select the network parameters and graph mode to view the curves.

SigWave uses the Smith Chart executable to display the Touchstone file path and Graph Mode. A second file containing the network parameters is also sent to the Smith Chart executable.

You view data tables graphically to:

- determine whether table data problems reported by the IBIS parser are an artifact of extracting clamp tables or are a potential problem. For example, non-monotonic warnings.
- check changes made by *ibis2signoise*. For example, adding an endpoint to a table to make the slope flat.

Note: Model Integrity catalogs the models into buffer model objects, but does not catalog the submodels. If you want to view submodel curves, translate the IBIS file into DML and view the curves from there.

See Viewing and Editing Waveforms in the *SigWave User Guide* for more information.

Simulating a Buffer Model

Simulation ensures the DML model will work in the Allegro SI environment. Simulation output is presented in a SigWave window. Before translation, you run a simulation for each model. You can also run a simulation on a DML model after running *ibis2signoise*.

You can simulate an unpackaged buffer model within a DML file by running a simulation using *tlsim*. A right-click on the selected buffer model object in either the Physical View or the Object View displays a context-sensitive pop-up menu containing a *Simulate* option that enables you to display a dialog box for modifying the input and output test fixture data that resides in the DML file. (Choosing *Tools – DML – Simulate Buffer* with the buffer model selected effects the same response.) If no such information exists, the dialog box displays the default parameter values.

When you click *Simulate*, tlsim simulates using the test fixture circuits. If the simulation is successful, SigWave launches and displays the waveforms. A simulation log displays in the Log File Window within Model Integrity. For more information on simulating and analyzing the results, see the SigWave User Guide.

You can simulate buffer models using Model Integrity. You can view the data tables graphically. Finally, you can simulate the buffer model with *tlsim* using the pre-defined IBIS test circuit that is included in Model Integrity. For a complete IBIS file, you can also create a complete DML file using the *ibis2signoise* translator, which performs checks on both IBIS syntax and data content.

Verifying simulation results

You verify simulation results to check the model behavior against the I/O characteristics. Verification can also find potential interactions between model, translator, and simulator that could cause problems in the Allegro SI flow. After you have run the simulation, compare transition edge-rate and waveform to the original V-t tables and the data in the IBIS model, as well as the edge rate on the datasheet, using the SigWave plot.

Tip

You can use SigWave to overlay curves contained in the original IBIS file and those produced by *ibis2signoise* in the DML file.

Qualifying Files

You mark a file qualified when you want to indicate that it has been analyzed and is now approved for use in the simulator. Since the simulator only recognizes DML files, file qualification is only offered for these files. Model Integrity checks the file and shows a warning message, if any checks fail. You then have the option to qualify the file despite these warnings.

Note: You must have write access to the file to mark it qualified.

Model Integrity checks the following:

- *dmlcheck* has run and the stamp is more recent than the file modification time.
- *dmlcheck* produced no errors.
- *dmlcheck* was run with the most recent version. For example, if Model Integrity v15-2-40A is running, Model Integrity will warn you if *dmlcheck* was run with any version prior to v15-2-40A.
- *modelsim* has run on all buffer models in the file with at least one circuit and that the stamp is more recent than the file modification time.

Results of qualification include:

- The file is made read-only.
- The file parsing status in the Physical View is set to Qualified, and a stamp is added to the file. This status remains, even if the file is subsequently parsed, and is reset whenever the file is reopened in Model Integrity, even if the *Parse On File Open* option is set in the Settings Dialog.

A

SPICE Templates

Template Files

Cadence provides SPICE templates to assist you in getting your SPICE *.lis files ready for Model Integrity. The templates are in the install directory:

<installation_directory>/share/pcb/modelintegrity/Spice_Templates

Model Integrity requires these files to generate SPICE output files to use to build an IBIS file.

Model Integrity uses position-sensitive data from the *.lis file, and therefore care should be taken in editing the following SPICE templates.

- SPICE templates (single-voltage)
- SPICE templates (dual-voltage)

The *.lis files are in the correct form when you use the provided templates with the SPICE simulator. If you choose to simulate in another SPICE simulator, the SPICE-specific statements in the templates must be replaced with the appropriate commands for your Spice tool. It may also be necessary to run the I-V and V-t simulations separately, in which case the output files (*.lis) must be concatenated.

The appropriate header string must appear in each output data table prior to opening the *.lis file in Model Integrity. These strings (from “temp=” to the beginning of data table values, and the end of data table values to “* job”) are created automatically by SPICE. For measured data or other SPICE simulators, these lines can be cut-and-pasted from the sample *.lis files provided in the templates directory, and the parameter values edited prior to extraction.

Using the SPICE Templates

SPICE templates exist for both single-ended and differential buffers. It is important to use these templates with care. In particular, the names and order of the nodes on the subcircuit (X0) should not be changed, since Model Integrity identifies data table contents by order, and not by column headings.

These templates have been specifically designed for use with SPICE. If you use another simulator, these templates may require some modification.

This may include changing the format for defining parameters and alter sections. It may be necessary to run the I-V and V-t simulations separately.

Because the buffer model extraction process depends on certain features of the buffer's SPICE subcircuit netlist, the following rules should be followed:

- The netlist for the buffer should be a subcircuit (`.subckt`). The templates have been set up to call a buffer subcircuit.
- The buffer subcircuit must include the input, output, output enable logic (if it exists), and ESD structures. It must include capacitors representing on-chip wiring capacitance. If an internal terminating resistor is used on-chip, it must be included in the subcircuit.
- To make sure edge rates from the core into the buffer are typical of the actual chip, it is imperative to include the pre-driver stage in the SPICE subcircuit so that the rise and fall times can be characterized properly.
- Remove components that are not part of the buffer and pre-driver stage, including external terminating resistors, logic portions of the component (such as flip-flops, multiplexers, demultiplexers), package components, and PCB transmission lines and components. You can put these in separate subcircuits, as long as the buffer subcircuit does not call them.
- Remove all external voltage and current sources from the subcircuit file. These are provided by the template. Internal sources should be retained.
- Remove all `.global` statements; all nodes are explicit in the templates.
- Examine all `.OPTIONS` statements that are not required for convergence. The templates contain the preferred options settings.
- Avoid the use of .OPTION SCALE=xxx whenever possible. The use of the SCALE option can make debugging difficult if subcircuits, libraries, or models use different scale factors
- To make it easier to debug any SPICE problems, the hierarchy of the buffer's subcircuit structure should match the physical hierarchy. Each subcircuit in the hierarchy should have a unique name.
- The ESD diodes and/or transistors must not have a zero-Ohm series resistance; typical resistances are on the order of 0.05 to 20 Ohms. The ESD device models should be checked in the model file. If your simulations show an ESD current of 1 MegAmp or more at -Vcc or +2Vcc, the resistance is probably much too low.
- The subcircuit netlist should use a name other than "0" or "gnd" for the reference node. The nodes "0" and "gnd" are treated as the same global node by HSPICE, as well as by

some other simulators. Model Integrity expects explicit (non-global) nodes in buffer subcircuits during extraction.

In the main circuit file (the template being edited):

Note: The default buffer name must be changed to match the name of the subcircuit in the buffer `.subckt` statement. This change must be made for ALL occurrences of the file.

- The subcircuit can be added directly in-line with the main subcircuit file, or it can be added using a `.INClude` statement. If using a `.INClude` statement, the complete path to the included file should be used if the subcircuit file is not in the same directory as the main circuit netlist file (that is, the circuit file that is submitted for simulation).

  Note: SPICE can locate subcircuit files.

  - The circuit file (*.`sp`) and the subcircuit file (*.`inc`) are in the same directory. The subcircuit file name must match the subcircuit name and the file extension must be `.inc`.
  - If the subcircuit file directory includes the `spice.ini` file, the subcircuit file name must match the subcircuit name and the file extension must be `.inc`.
  - An explicit include statement is required if the file name does not match the above requirements. The default path is the same directory.

- All parameters should be set for the buffer. The parameters all appear near the top of the template file.
- The temperature must be set for each simulation corner. This is done by finding each "`.temp`" statement, and setting the temperature to the TYP, MIN, or MAX value specified for the component. Note that this must be done for ALL occurrences. Note that these are junction temperatures, and not case temperatures; for example, this means TYP will be higher than ambient room temperature.
- If only TYP process models are available, the process models can be included in the same file with the subcircuits, or the process file can be included using a `.LIBrary` statement. Guardbanding can be used during extraction to generate estimated min/max (slow/fast) IBIS tables.
- If data is available for all three process corners, use a `.LIBrary` statement to call one process corner within the process library file. This change must be made for ALL of the file. The templates *`_lib.sp` provide examples of process corners. The process corners may be identified by MIN/slow/worst-case, MAX/fast/best-case, or similar terms.
- Make sure that the names of the models used for the transistors and diodes in the subcircuit match the names in the .MODEL statements in process library. If the subcircuit calls a model name and does not find it in the model library, the simulation will not run.

- The STOP time for the transient analysis V-t tables must be long enough to capture the entire transition (V-t waveforms become flat).
- The STOP time for the I-V simulations should not be changed.
- The transient STEP for the V-t tables should be short enough to guarantee there are enough points for accurate IBIS modeling. For example, if the edge time of a buffer is 1 nsec, the transient STEP size should be no larger than 0.01n.
- The order of the nodes on the subcircuit X0 may be changed to match the node order in the subcircuit file, as long as the names match the assigned meaning in the comment statement. If this is done, it must be done for every occurrence in the template.
- The templates should not need to be modified, except for the edits described above. In particular, the names of the nodes on the subcircuit X0 should not be changed, since these are used for setting up signal and power supply sources and SPICE output formatting.

In each template, the subcircuit call to the buffer contains the number of nodes for the given buffer type (input, output, i/o, etc.). In the template, signal sources or power supplies are connected to these nodes. The nodes are as follows.

Node	Buffer Type	Description
1	In	Data/signal from the chip core.
2	out	data/signal pad from the outside world.
3	puref	pullup reference supply (I/O ring Vcc).
4	PCLref	power clamp diode reference supply (normally I/O ring Vcc)
5	pdref	pulldown reference supply (normally gnd for CMOS).
6	GCLref	ground clamp reference supply (normally gnd for CMOS).
7	en	output enable (active low set as default).

The subcircuit name in the template must be replaced with the name of the buffer subcircuit, and the node list appropriate to that buffer. The examples shown here are for an input buffer (3 nodes), and a dual-voltage I/O buffer (6 nodes). Only the node name for X0 should be changed. Note that this must be done for ALL of the X0 statements in the SPICE template.

```
X0 2 3 5 IN_buf          $ <<<------ Change buffer name here
```

X0 1 2 3 4 5 7 IO3buf5 $ <<<------ Change buffer name here

The parameters at the top of the file need to be set to the values for the buffer being simulated. The clamp voltages may be the same as the I/O rail voltages, or they may be different, depending on the buffer design.

The temperature value must be set for all occurrences of ".TEMP". For CMOS, "min" corner uses maximum temperature and "max" corner uses minimum temperature. These are set to 100 and 0 Celsius, respectively, in the templates. The default "typ" temperature is 50 Celsius. The temperature is the on-die "junction" temperature of the transistors (MOSFET or bipolar). Junction temperatures can be calculated from the ambient temperature, the thermal resistance of the packaging, and the average power dissipation:

```
TEMP (Celsius)  =  Thermal_R * P_avg  +  Temp_ambient
```

where:

Thermal_R is package thermal resistance in degrees/Watt.

P_avg is average power dissipation in Watts.

Temp_ambient is ambient temperature in Celsius.

Example of parameter section in the template file:

```
.TEMP   50                      $ Temperature of typical case
*-------------------------------------------------------------------*
.PARAM  PUref_typ  = 3.300V     $ Pullup reference voltage, typ.
.PARAM  PUref_min  = 3.135V     $ Pullup reference voltage, min.
.PARAM  PUref_max  = 3.465V     $ Pullup reference voltage, max.
.PARAM  PCLref_typ = PUref_typ  $ Power clamp reference voltage, typ.
.PARAM  PCLref_min = PUref_min  $ Power clamp reference voltage, min.
.PARAM  PCLref_max = PUref_max  $ Power clamp reference voltage, max.
*-------------------------------------------------------------------*
.PARAM  PDref_typ  = 0.000V     $ Pulldown reference voltage, typ.
.PARAM  PDref_min  = 0.000V     $ Pulldown reference voltage, min.
.PARAM  PDref_max  = 0.000V     $ Pulldown reference voltage, max.
.PARAM  GCLref_typ = 0.000V     $ GND clamp reference voltage, typ.
.PARAM  GCLref_min = 0.000V     $ GND clamp reference voltage, min.
.PARAM  GCLref_max = 0.000V     $ GND clamp reference voltage, max.
*********************************************************************
.PARAM  Vpdref     = PDref_typ  $ Reference voltages for typical case
.PARAM  VGNDclref  = GCLref_typ
.PARAM  Vpuref     = PUref_typ
.PARAM  VPOWclref  = PCLref_typ
*-------------------------------------------------------------------*
.PARAM  Ven        = Vpdref     $ Active-low enable
*.PARAM  Ven        = Vpuref     $ Active-high enable
```

```
*********************************************************************
.PARAM  Vfx_pd_on  = Vpuref
.PARAM  Vfx_pd_off = Vpuref
.PARAM  Vfx_pu_on  = Vpdref
.PARAM  Vfx_pu_off = Vpdref
*
.PARAM  Rfx_pd_on  = 50  $ T-line load impedance (typically 30-125 ohms)
.PARAM  Rfx_pd_off = 50
.PARAM  Rfx_pu_on  = 50
.PARAM  Rfx_pu_off = 50
*
.PARAM  Cfx_pd_on  = 0.0pF  $ Load capacitance (typically 0 pF)
.PARAM  Cfx_pd_off = 0.0pF
.PARAM  Cfx_pu_on  = 0.0pF
.PARAM  Cfx_pu_off = 0.0pF
```

The component X0 is instantiated (contained) in the subcircuit BUFFER. Changes to node order may be made on X0 to match the node order on the subcircuit, as long as both power and reference nodes are explicitly passed.

```
.SUBCKT BUFFER  1      2      3      4      5      6      7
*               in     out    puref  PCLref pdref  GCLref /en
X0 1 2 3 4 5 7 IO3buf5    $ <<<------ Change buffer name here
.ENDS
```

The `.SUBCKT BUFFER` statement should not be changed unless additional nodes are required (such as for differential buffers). In this case, the additional nodes should have sources or loads applied. For example, for a differential buffer, the inverting output node (node 8) should have an opposite current to the non-inverting output node.

The appropriate process file corner (TYP/MIN/MAX) must be called in the main circuit and in each `.ALTer` section. TYP can be called when MIN or MAX process data is not available; guardbanding can be applied during extraction to generate estimated min and max characteristics. The process models or library can be included in the subcircuit file or by using the `.LIBrary` statement.

To protect netlist and process IP, the process models and subcircuit netlist can be placed between the `.PROTECT` and `.UNPROTECT` statements of the main circuit and each `.ALTer` section. This also reduces the size of the SPICE output files, which makes buffer model extraction faster.

Special Cases

There are special cases that are not addressed in the templates provided here. If you are familiar with SPICE, you may want to modify the templates for these cases.

Case 1: Active-high Enable: In the SPICE input netlist, comment out the active-low enable parameter, and uncomment the active-high enable parameter as follows:

```
*.PARAM  Ven        = Vpdref       $ Active-low enable
.PARAM  Ven        = Vpuref       $ Active-high enable
```

Case 2: The subcircuit has more than 7 pins (such as LVDS). The additional subcircuit nodes should follow node 7. For each simulation (alter), the inverting and non-inverting inputs need to be loaded or driven appropriately. This is non-trivial, and great care must be taken to make sure this is done correctly.

Case 3: Using some other SPICE simulator, such as PSpice or Spectre. In this case, you will probably need to break your SPICE template into separate I-V and V-t templates. Each template must be modified to produce the same output file (including table headings) as is produced in SPICE.