cādence™

Library Explorer User Guide

Product Version 16.01
December 2007

Contents

Preface

About This Manual

Part Developer and Library Explorer functionality is divided into two levels. The core capabilities are found in all versions of Library Explorer and Part Developer shipped as part of PCB Librarian and the latest packaging configurations of Allegro Design Entry HDL. The extended capabilities are restricted to the PCB Librarian XL license only.

If you use the PCB Librarian XL suite, then in addition to the basic library management features of Library Explorer, you benefit from the revision management system providing built-in control preventing unauthorized changes to your library data. Many other advanced features are provided as part of the PCB Librarian XL and by harnessing the power of PCB Librarian XL, engineering services departments around the world can generate library parts quickly and accurately for use with Cadence Allegro Design Entry HDL schematic and Cadence Allegro layout software.

This guide tells you how to use Library Explorer to manage digital design libraries.

This guide assumes familiarity with a system text editor, HDL language concepts, and the following Cadence tools used to create component symbols and models:

- Part Developer, which lets you create library components
- Part Table Editor, which lets you create part table files
- Allegro Design Entry HDL, which lets you create logic designs by drawing schematics using symbols and functional blocks
- Packager-XL, which lets you prepare your schematic for PCB layout
- Allegro PCB Editor, which lets you create and manage physical layouts

Finding Information in This Manual

Chapter 1, “Getting Started” describes the library management use model for librarians and designers. This chapter also details the steps to launch Library Explorer, Part Developer, and Part Table Editor.

Chapter 2, "Library Concepts," provides a comprehensive description of how the libraries are stored and the different views of a part in a library.

Chapter 3, "Library Explorer Concepts" details the features of Library Explorer. This chapter also describes the files that are required for Library Explorer to function correctly.

Chapter 4, "Creating a New Library" details the tasks and steps that are necessary to create and work with the build areas in Library Explorer.

Chapter 5, "Working in Reference Area" describes the tasks that can be performed in the reference area.

Chapter 6, "Working with Category Files" details the tasks that needs to be done while working with category files.

Chapter 7, "Using Other Tools from Library Explorer" describes the steps that need to be done to launch other tools from Library Explorer.

Appendix A, "Library Explorer Checks," details the verification checks that can be run on a library or a part from within Library Explorer.

Appendix B, "Dialog Box Help," describes the dialog boxes and their fields.

Typographical Conventions

This list describes the syntax conventions used for tools used in the library development and management process. Where applicable, exceptions to these conventions are explicitly indicated.

`literal (LITERAL)`	Nonitalic or (UPPERCASE) words indicate key words that you must enter literally. These keywords represent command (function, routine) or option names.
`argument`	Words in italics indicate user-defined arguments for which you must substitute a value.

|	Vertical bars (OR-bars) separate possible choices for a single argument. They take precedence over any other character. For example, `command` *`argument`* `|` *`argument`*
[ ]	Brackets denote optional arguments. When used with OR-bars, they enclose a list of choices. You can choose one argument from the list.
{ }	Braces are used with OR-bars and enclose a list of choices. You must choose one argument from the list.

...	An ellipsis indicates that you can repeat the previous argument. If they are used with brackets, you can specify zero or more arguments. If they are used without brackets, you must specify at least one argument, but you can specify more.
	`argument``...: specify at least one argument, but more are possible`
	`[`*`argument`*`]...: you can specify zero or more arguments`

,...	A comma followed by an ellipsis indicates that if you specify more than one argument, you must separate those arguments by commas.
`Courier font`	Text in Courier font indicates command-line examples.

Related Documentation

The following manuals give you information about other tools used during the library and part creation and management process:

If you want to know...	**Read**
How to create library parts	Part Developer User Guide
How to create Part Table Files	Part Table Editor User Guide
How to use Allegro Design Entry HDL to enter schematics	Allegro Design Entry HDL User Guide

If you want to know...	Read
More about Allegro Design Entry HDL digital libraries	Allegro Design Entry HDL Libraries Reference
More about how to create and use physical layouts	Allegro documentation
More about properties supported by Cadence PCB design software	PCB Systems Properties Reference

1

Getting Started

Library Management Use Model

Library Explorer is an important tool in the PCB Librarian XL suite that provides you the ability to perform library creation and management tasks. Depending upon whether you are a librarian or a designer, you can use the tools to perform specific tasks.

Tasks Performed by a Librarian

As a librarian, you can do the following.

Use Library Explorer to:

1. Create or open a build area.
2. Import reference libraries or parts you wish to modify into the build area.
3. Create any new libraries or cells.
4. Launch Part Developer to create or edit views.
5. Verify libraries.
6. Export new or modified libraries and parts to the reference library location.
7. Clean up the build library area when finished.

Use Part Developer to:

1. Create or modify symbol, package, simulation, and part table views.
2. Verify the part.

Use Part Table Editor to:

1. Create or modify a part table.
2. Verify a part table.

3. Add new parts to a part table.

Tasks Performed by a Designer

As a designer, you can do the following:

1. Create a project using Project Manager.
2. Specify the project libraries while creating the project.

   Now if you want to either create a new part or modify existing parts in the project libraries, you can launch the Part Developer tool. When you launch the Part Developer tool, it displays the project libraries thus enabling you to modify and create new parts only in them.

Note: Library Explorer will not be available on a project created through Project Manager as library management tasks, such as creating new libraries and categories and copying libraries, should be done only by a librarian.

Starting Library Creation and Management Tools

Starting Library Explorer

From Project Manager

To launch Library Explorer from Project Manager:

➤ Choose *Tools – Library Explorer*.

From the Command Prompt

In UNIX, type the following command:

```
libexp
```

In Windows NT, type the following command at the command prompt:

```
libexp
```

Starting Part Developer

From Project Manager

To launch Part Developer from Project Manager:

➤ Choose *Tools – Part Developer.*

From the Command Prompt

In UNIX, type the following command:

```
pdv
```

In Windows NT, type the following command at the command prompt:

```
pdv
```

Starting Part Table Editor

From the Command Prompt

In UNIX, type the following command:

```
ptf&
```

In NT, type the following command at the command prompt:

```
ptf
```

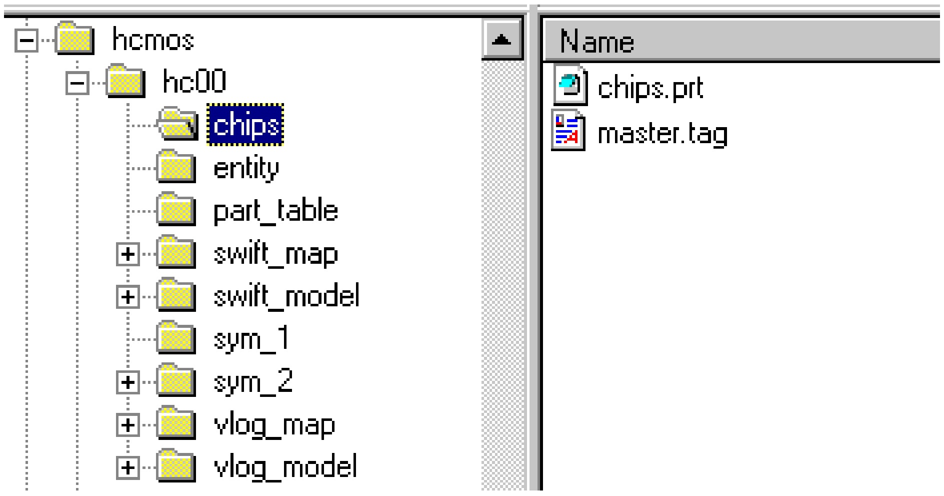
2

Library Concepts

Library Explorer Overview

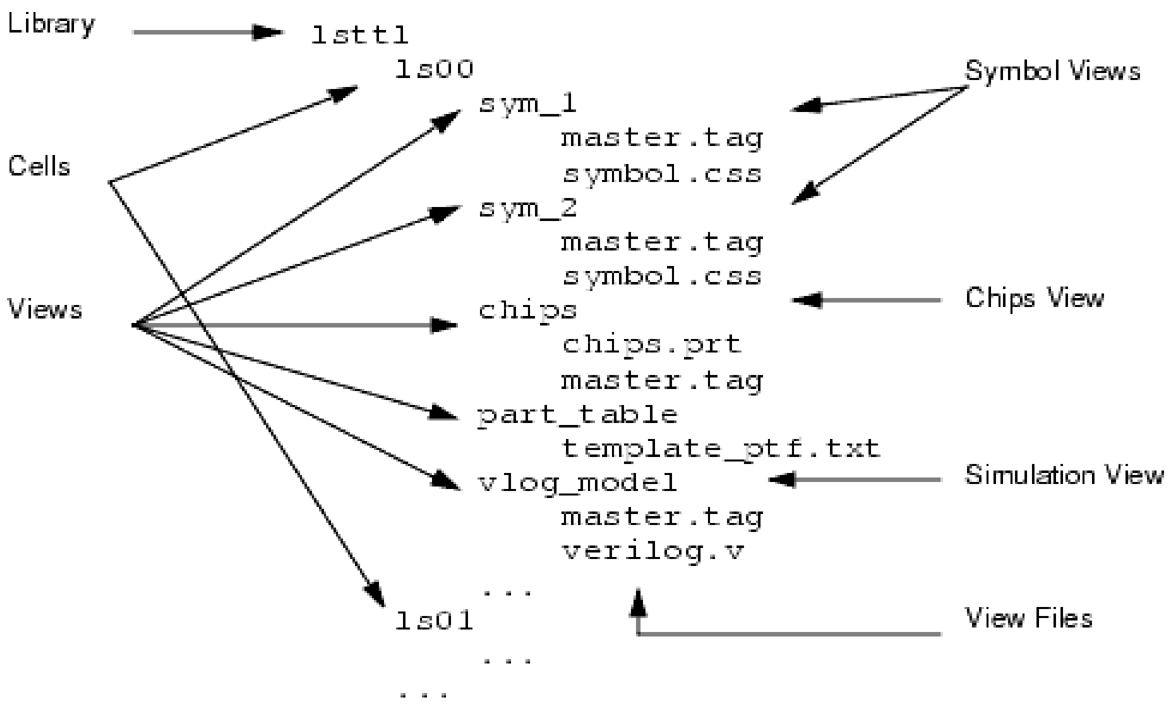
Library Explorer is a tool for managing the part creation and library management process. It manages a build area that is used for creating new components and updating existing components before promoting them to a reference area. The reference area is where all the reference libraries reside. This is the area from where Allegro Design Entry HDL typically picks up the symbols. Components managed by Library Explorer are verified as usable in Design Entry HDL, Rules Checker, and Packager-XL.

Physical Organization of Libraries



The libraries are organized into separate directories, one for each technology. For example, HCMOS parts are in a directory called `hcmos`. Each library contains many subdirectories, one for each of the parts, such as `hc00` and `hc02`. Each part has several views, each of them describing the part in a unique manner. You can see the set of Cadence-supplied libraries in `<your_install_dir>/share/library`.

Lib-Cell-View Architecture



The libraries are based on a library-cell-view architecture. Each part (cell) has several views, each describing the part in a unique way.

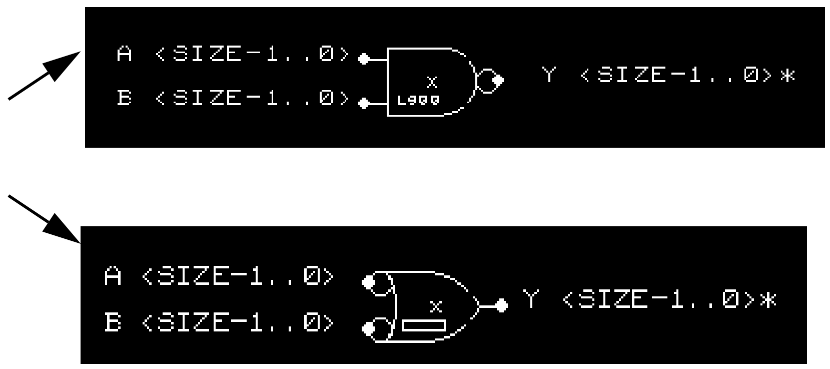
Views

Symbol (sym) View

The symbol view is the logical representation of a part in a Design Entry HDL drawing. Each part can have one or more symbol views that are in effect different versions of the logical representation.

Figure 2-1 Examples of Symbol Views

```
lsttl
     ls00
          sym_1
               master.tag
               symbol.css
          sym_2
               master.tag
               symbol.css
```



You need to create these versions when:

- You need different graphical representations as shown in the above example.
- You need scalable symbols.

  Different versions or symbol views are stored under directories named `sym_1`, `sym_2`, and so on.

Package (chips) View

The package view or the chips view stores the package information, such as pin names, pin numbers, and electrical information, for a part. This view connects the logical view of a component to its physical view.

Pin information, such as pin names, pin types, pin loading and physical pin numbers, is stored in the `chips.prt` file located in the `chips` directory. For more information on the `chips.prt` file, see the Design Entry HDL Libraries Reference.

Figure 2-2 Typical chips.prt file

```
FILE_TYPE=LIBRARY_PARTS;
TIME=’COMPILATION ON THU JAN 10 14:52:02 1991’;
primitive ‘74LS01’,’74LS01_DIP’;
     pin
          ‘B’<0>:
               INPUT_LOAD=’(-0.4,0.02)’;
               PIN_NUMBER=’(12,9,6,3)’;
               PIN_GROUP=’1’;
          ‘A’<0>:
               INPUT_LOAD=’(-0.4,0.02)’;
               PIN_NUMBER=’(11,8,5,2)’;
               PIN_GROUP=’1’;
          ‘-Y’<0>:
               OUTPUT_LOAD=’(8.0,*)’;
               OUTPUT_TYPE=’(OC,AND)’;
               PIN_NUMBER=’(13,10,4,1)’;
     end_pin;
     body
          POWER_PINS=’(VCC:14;GND:7)’;
          FAMILY=’LSTTL’;
          PART_NAME=’74LS01’;
          BODY_NAME=’LS01’;
          MAX_DELAY=’10000’;
          DEFAULT_SIGNAL_MODEL=’SN74LS01N  TI’;
          JEDEC_TYPE=’DIP14_3’;
          CLASS=’IC’;
          TECH=’74LS’;
     end_body;
end_primitive;
END.
```

Entity View

This view contains a Verilog module and a VHDL entity declaration. Both of them describe the list of ports found on the part. This view is automatically created when a part is saved to the disk through Part Developer.

Part Table View

This view has additional properties that are used to customize a part. This view appears as a `part_table` directory and can have multiple files with the `.ptf` extension. This view is used while packaging the part along with the chips view.

Simulation View

When a symbol view is saved to the disk, an entity view is automatically created. In the entity view is a Verilog and VHDL file that contains the names of all the pins on the symbol (known as a module). The simulation view maps the symbol (or module) to a simulation model. The name of the module is mapped to the name of the simulation model. The pin names in the module are mapped to the port names in the simulation model. This file is sometimes called a wrapper because it contains only mapping data. The actual simulation model is stored in an HDL model library. The Cadence-supplied HDL model library is stored in `<`*`your_install_dir`*`>/veriloglib`.

During simulation, the Verilog file in the schematic view is used as the netlist. Each part in this netlist has an entity and a simulation view.

Verilog-XL replaces the parts in the netlist with the simulation models as defined by the wrapper or a map file.

Category (.cat) Files

In addition to the supported views, you can also create a category file (.cat) within each library to organize the parts into functional groups, such as BUFFER, CLOCK-DISTRIBUTION, and so on. The category files are located within each library. This is an optional file.

3

Library Explorer Concepts

The cds.lib and refcds.lib Files

The `cds.lib` and `refcds.lib` files define the logical library names and the physical storage locations for each installed library. The `cds.lib` and `refcds.lib` files determine the libraries that are visible in the build and the reference areas, respectively.

Library Explorer creates both the `cds.lib` file and the `refcds.lib` file in the same hierarchy where a new library project is created.

Build Area

The build area is the work area for private library development needs. The libraries and parts created in the build area are available only to the librarian. After the parts have been verified, these can be exported to the reference area from where other designers can access them. The entries in the build area are controlled through the `cds.lib` file.

Reference Area

It is a good design practice to create a separate area to store the tested and finalized libraries. This area is termed as reference area. The list of libraries in the reference area are stored in the library list file named `refcds.lib`.

The Cadence-supplied standard libraries are added to the reference area during installation. When you finalize a library in the build area, you can export it to the reference area so that other users can also use it. Usually, only the librarian has permissions to export the libraries to the reference area.

You can create, modify, rename, and delete your reference libraries only in your build area where you have the necessary permissions.

To create a new reference library, you must export a build library into the reference area.

To modify, rename, or delete a reference library, you must import it into your build area, make the necessary changes, and export it back to the reference library.

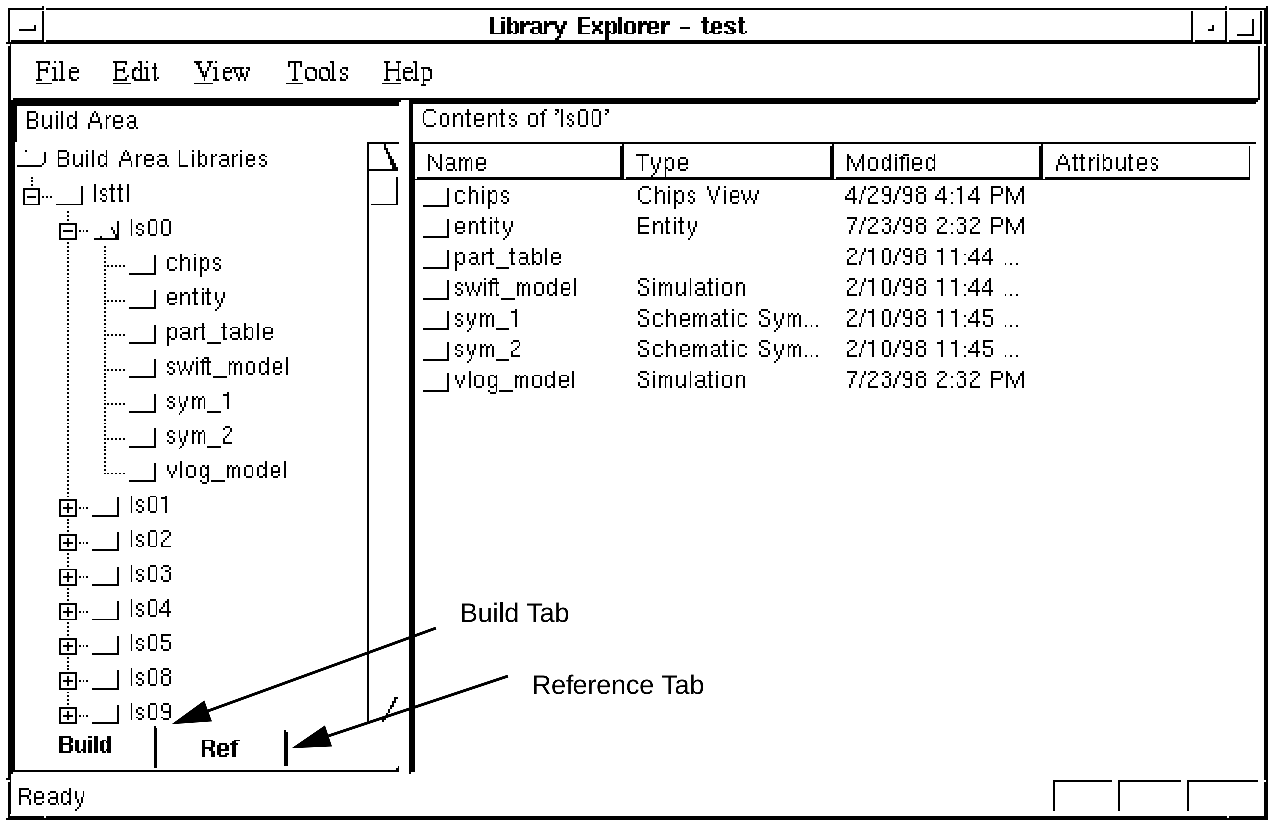
Library Explorer Features

The Library Explorer tool lets you create, view, and maintain part libraries. You can use Library Explorer to:

- Create and maintain build areas.
- Browse reference libraries.
- Import and export files, parts, and libraries to and from the build area.
- Create new libraries and parts.

  You can create new libraries within the build area and new parts under the library. You can then edit the parts using Part Developer.
- Create and maintain library category files.
- Launch Part Developer and other tools to create and edit library parts.
- Select a part or view and launch Part Developer to edit that part. You can also launch Design Entry HDL for editing or viewing symbols, or display a file in the text editor of your choice.
- Run various checks to ensure the validity of the libraries and parts.

Library Explorer Work Environment



Contents of 'ls00'

Name	Type	Modified	Attributes
chips	Chips View	4/29/98 4:14 PM	
entity	Entity	7/23/98 2:32 PM	
part_table		2/10/98 11:44 ...	
swift_model	Simulation	2/10/98 11:44 ...	
sym_1	Schematic Sym...	2/10/98 11:45 ...	
sym_2	Schematic Sym...	2/10/98 11:45 ...	
vlog_model	Simulation	7/23/98 2:32 PM	

Similar to the Window Explorer, the Library Explorer window is divided into two panes. The left pane displays the lib-cell-view structure of the Design Entry HDL libraries. Additionally, there are *Build* and the *Reference* tabs at the bottom of the left pane.

These tabs display the libraries you are browsing. The *Reference* tab displays all libraries listed in your `refcds.lib` file (this file is used only by Library Explorer). The *Build* tab displays the libraries listed in the `cds.lib` file, which is in the build area. These libraries include the project work library and the other local libraries that you have created.

The right pane displays the contents of the libraries or parts selected in the left pane.

As parts are imported, created, or renamed in the build area, the `cds.lib` file and the project library list in the project file are updated. When you choose the *View – Refresh* command in the build area, Library Explorer rereads the `cds.lib` file and updates the build library display.

If you want to add additional reference libraries, you must edit the `refcds.lib` file and add DEFINE statements for the new libraries. When you choose the *View – Refresh* command in the reference view area, Library Explorer rereads `refcds.lib` and updates the reference display. The `refcds.lib` file can be edited by choosing the Reference tab and selecting the *Edit – ModifyLibraryList* option.

4

Working with Build Areas

Creating a New Build Area

The Project Wizard lets you create a new project. Library Explorer treats a new project as a new build area. You can create multiple build areas, with the last build area you created being the active work area.

For example, create a project `triallib` as below.

1. Choose *File – New – Build Area*.

   The New Project Wizard dialog box appears. The wizard guides you through all the steps required to set up a new project.

2. Enter the project file name, `triallib`.

3. Enter the location. For example, enter c:/work.

4. Click *Next*.

   The list of available reference libraries appears. These libraries are the ones that are supplied by Cadence and located in `<your_install_dir>/share/library`. You can add new libraries using the *Add* button, import libraries from a `cds.lib` file using the *Import* button, or remove one or more libraries from the displayed set by using the *Remove* button.

5. If you want to add all available libraries, click *Next.*

   The New Project Wizard - Summary dialog box appears.

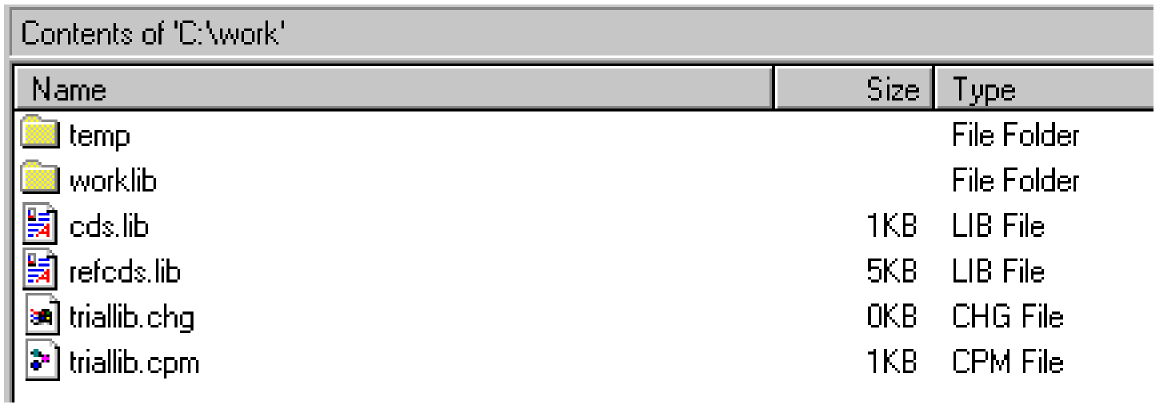
6. Click *Back* if you want to change any values.

7. Click *Finish* to confirm the values you have specified.

   Library Explorer automatically sets up your project.

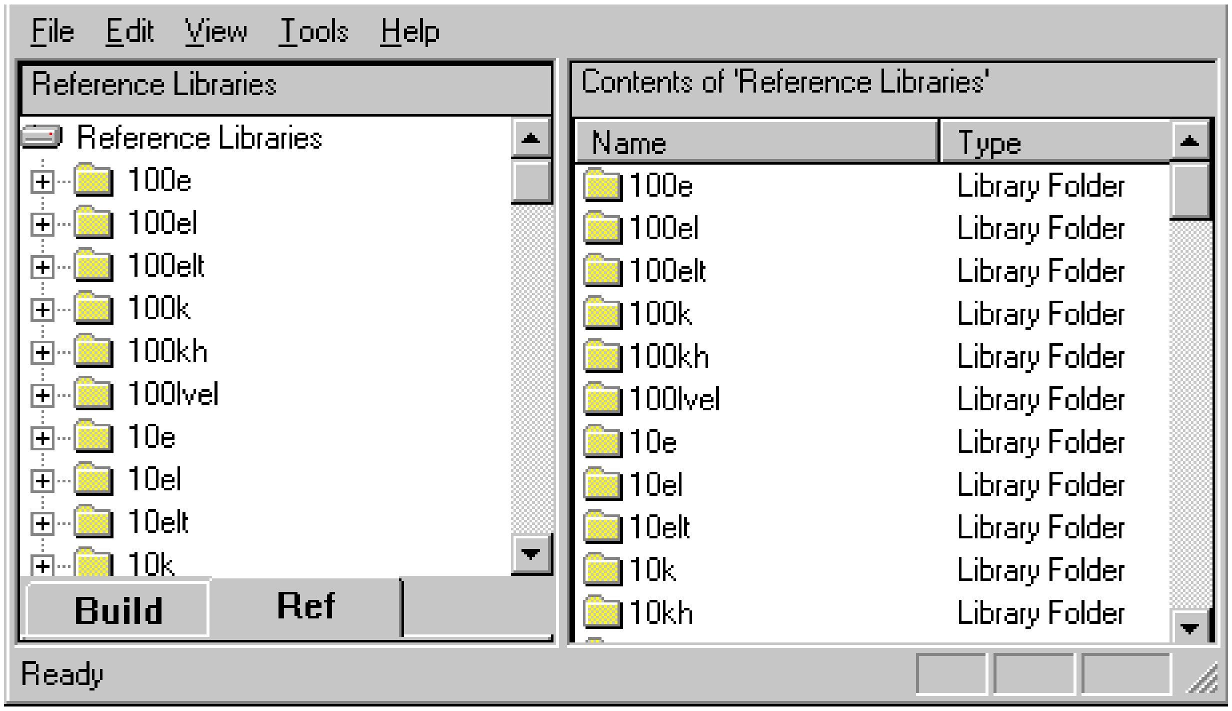
The Project Wizard will create the `<`*`projectname`*`>.cpm` file, the `cds.lib` file, and the `refcds.lib` file in the specified location. It will also create the two folders `temp` and `worklib` in the same hierarchy. The worklib folder is to store the library project and is required for Project Manager to work properly. The temp folder is for all temporary files.

For example, in the `triallib` example, under the `c:/work` folder, you will have the following files and folders:


Contents of 'C:\work'

Name	Size	Type
temp		File Folder
worklib		File Folder
cds.lib	1KB	LIB File
refcds.lib	5KB	LIB File
triallib.chg	0KB	CHG File
triallib.cpm	1KB	CPM File

The triallib project will appear as follows:



Contents of 'Reference Libraries'

Name	Type
100e	Library Folder
100el	Library Folder
100elt	Library Folder
100k	Library Folder
100kh	Library Folder
100lvel	Library Folder
10e	Library Folder
10el	Library Folder
10elt	Library Folder
10k	Library Folder
10kh	Library Folder

Note: Ensure that the *worklib* folder is not deleted from your project area. If this folder is deleted, Library Explorer will display `mkdefcg` errors.

Opening an Existing Build Area

To open an existing build area:

1. Choose *File – Open Build Area.*
2. Select the *.cpm* file for the build area.
3. Click *Open*.

Working in the Build Area

The build area is the area where you do all your library management activities. In the build area, you can:

- Create new libraries and parts
- Import reference libraries, edit them, and export them back to reference areas

Creating a New Library

You can create a new library in the build area. To create a new library:

➤ Choose *File – New – Build Library.*

This will create a library named `new_library` in your build area. You can then rename the library as per you requirements.

Creating New Parts in Build Libraries

You can create new parts in the libraries. Library Explorer launches the Part Developer tool to enable you to create the parts. To create a part:

1. Select the library in which you want to create a new part.
2. Choose *File – New – Part.*
3. Enter the name for the part.
4. Choose *Tools – Part Developer.*

   Part Developer appears.
5. Use Part Developer to create the part.

Importing and Exporting Libraries

Importing Libraries

Library Explorer enables you to modify libraries and parts. However, if you need to modify any library in the reference area, you have to import it to the build area, work on it, and export it back to the reference area.

You need to take care of the following point while importing libraries:

1. When you import a library, the library structure is physically created in the project directory. If the item you are importing already exists in the build area, the *Replace* dialog box appears.

Library Explorer allows complete or partial import of libraries. You can import reference libraries or parts to your build area. When you import an item, Library Explorer maintains the hierarchy in the destination library.

For example, import the `100el` library into the build area as shown below:

1. Choose *File – Import.*

   The Import dialog box appears.

2. Select the libraries and parts you wish to import. For example, select the `100el` library by selecting the check box next to the `100el` entry.

   Note: Libraries selected for import have a check box next to them. All the parts in the library are selected for import. If you do not want to import a part, deselect the check box next to the part. Items partially selected for import have a gray box next to them.

3. Click *OK*.

The *Output* tab appears and displays the status of import for each library.

Exporting Libraries

If you have the necessary rights on the reference area, Library Explorer allows complete or partial export of libraries. You can export libraries and parts from your build area to the reference area. If you export a part, Library Explorer maintains the library hierarchy in the destination library.

To export:

1. Choose *File – Export.*

   The Export dialog box appears.

   Library Explorer selects some items for export in the Export dialog box. You can specify other items for export. You can also deselect the items marked for export.

   The following are selected by default for export:

   ❑ New library and new part.

   ❑ Item that has been imported and modified since the last import.

   ❑ Item being currently edited.

2. Select the library and parts you want to export.
3. Click *OK*.

   The Export Destination dialog box appears.

4. Select the destination library for each of the libraries you wish to export.
5. If you want to export to a new reference library, click *New Reference Library*. This option will create the lib-cell-view hierarchy in the destination that you specify.
6. Specify the name and path for the new reference library.
7. In case you have created a new reference library, select the library name from the *Destination Library* drop-down list.
8. Click *OK* in the Export Destination dialog box.
9. The *Output* tab appears and displays the status of export for each library.
10. Click the *Enter log info* button to enter log file information. The log file is created at the same level as the library parts in the destination reference library. The name of the log file is *`destination_ref_lib_name.`*`log.` For example, if you export a build library `100el` to a new reference library `my100el`, the log file will be named `my100el.log` and saved in the `my100el` library.

Renaming Items in the Build Area

Library Explorer enables you to rename libraries or portions of the libraries in your build area.

1. Choose the library or part that you want to rename.
2. Choose *Edit – Rename* to rename the library or the part.

3. Specify the name and press `Enter`.
4. If you are renaming an imported library, the Confirm Folder Rename dialog box appears. If you want to rename the library in the reference area also, click Yes in the Confirm Folder Rename dialog box. This information is used during the time of export of the libraries into the reference area. If you select No, the reference area library will not get renamed after the export and only the parts will get copied into the reference area.

   For example, import the library *`100el`* into the build area and rename it to *`my100el`*. When the Confirm Folder Rename dialog box displays, click on *Yes*. Then export the library back to the reference area. Notice that after you export it back, the *`100el`* library in the reference area has been renamed to *`my100el.`*

Modifying Parts in a Build Library Area

Library Explorer enables you to modify parts in a build library area. To modify a part:

1. Choose the library in which you want to modify part information.
2. Choose the part you want to modify.
3. Choose *Tools – Part Developer*.

Use Part Developer to modify the selected part. For more information, see the chapter on Part Developer.

Deleting Items in the Build Area

Library Explorer allows you to delete libraries and parts in the build area. To delete libraries and parts:

1. Select the library or part that you want to delete in the build area.
2. Choose *Edit – Delete.*
3. Choose the *Delete from build area* radio button.
4. Click *Yes.*

5

Working in Reference Area

Reference Area Overview

The *Ref* tab in Library Explorer displays the reference libraries. You can create new reference libraries and modify the existing ones. It is a two-step process. First, using Library Explorer, create, modify, rename, and delete libraries in your build area. A build area is the local area where you have the necessary permissions. After you are finished with the build area tasks, export the library to the reference area.

Creating a New Reference Library

You can create a new reference library in the following ways:

- By creating a new reference library while exporting a library
- By renaming an existing build library and exporting it

Specifying a New Reference Library

You can create a new reference library by specifying the selected library as a new reference library while exporting your libraries into the reference area. For example, create a library named `my100el` in the build area and export it as a new reference library.

1. Open a build area.
2. Create or modify library and parts. For example, create the `my100el` library in the build area.
3. Choose *File – Export*.

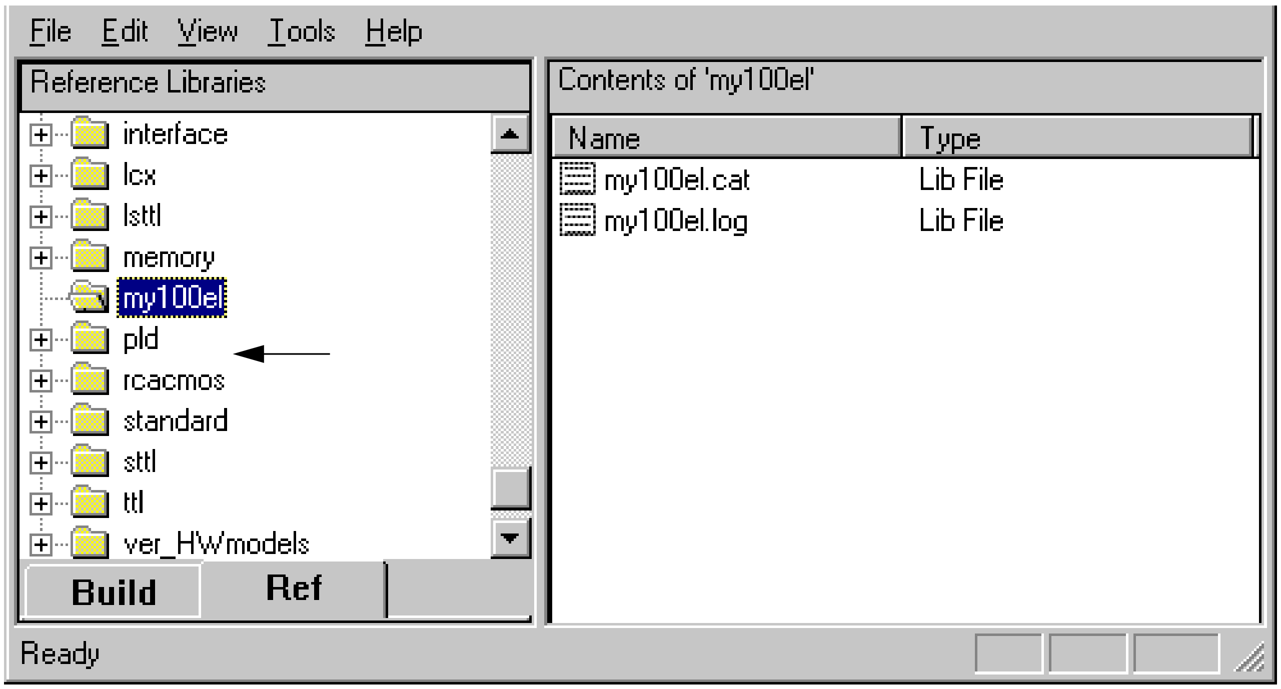
   The Export dialog box appears.
4. Select the library and parts you wish to export. For example, select the `my100el` library.
5. Click *OK*.

The Export Destination dialog box appears.

6. Click *New Reference Library*.
7. Enter a name for the new reference library. For example, enter `my100el`.
8. Click *Browse* to specify the path to the new reference library. For example, browse to select `c:/work/libraries`.
9. Select the new reference library as the destination library for each part. For example, select `my100el` from the destination library drop-down list.
10. Click *OK*.

    This successfully exports the selected library as a new reference library. You can view the new reference library in the reference tab.

    For example, the `my100el` library is displayed in the reference library list.



Renaming an Existing Reference Library

The other way of creating a new reference library is to rename an existing reference library. The steps are:

1. Import the source reference library you want to rename to your Build area.

2. Choose *Edit – Rename*.
3. Export the library back to the reference area.

Creating New Parts in Reference Libraries

Library Explorer enables you to create new parts in reference libraries.

1. Import the reference library to your build area.
2. Select the library in which you want to create a new part.
3. Create the new part.
4. Export the library with the new part back into the reference area.

Adding Reference Libraries

Reference libraries are identified by the contents of the `refcds.lib` file in the project area. You can add a set of reference libraries if required. For example, add the library `mypld` in the `c:/work/libraries` folder into the reference library set.

To add libraries to the reference set:

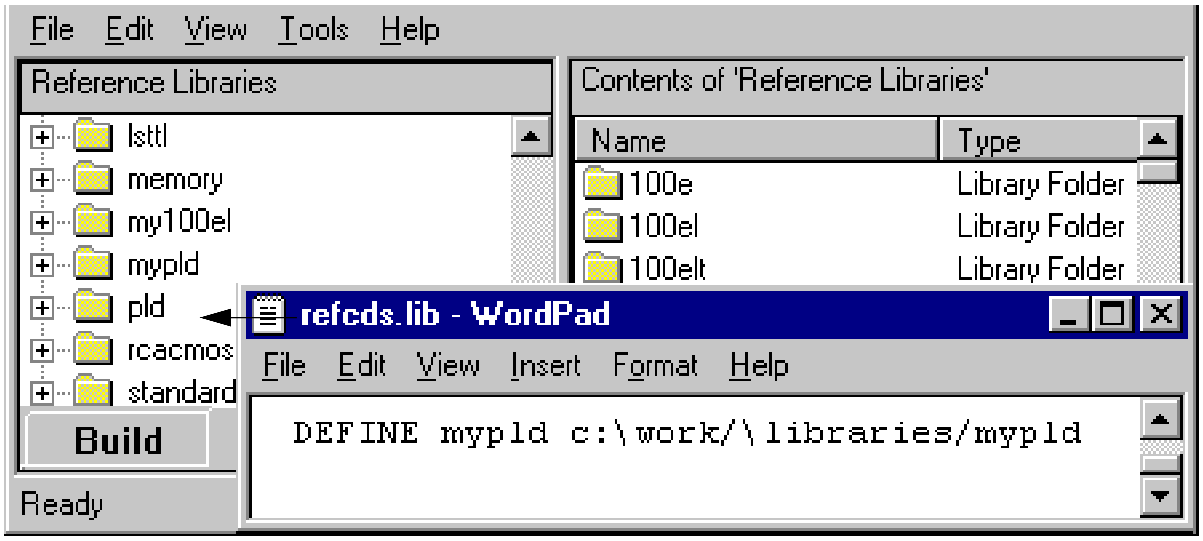
1. Select the *Ref* tab.
2. Choose *Edit – Modify Library List.*

   This opens the `refcds.lib` file in a text editor.
3. Add the appropriate entries in this file and save the file. For example, to add `mypld` in the `c:/work/libraries` hierarchy, enter

   ```
   DEFINE mypld c:/work/libraries/mypld
   ```

   The new entries should appear in the reference area.

For example, the `mypld` library is displayed in the reference library and the corresponding `refcds.lib` entry is displayed.



```
DEFINE mypld c:\work/\libraries/mypld
```

Modifying Parts in a Reference Library

Library Explorer enables you to modify parts in a reference library set. To modify parts in a reference library:

1. Import the reference library to your build area.
2. Modify the part.
3. Export the library with the modified part into the reference area.

Deleting Items in the Reference Area

Assuming that you have the necessary permissions, Library Explorer allows you to delete libraries or portions of the libraries in your reference area. You must import the library to be deleted into the build area, delete it, and then export it back to the reference area to complete the deletion.

1. Import the library or part to be deleted.
2. Select the library or part that you want to delete in the build area.
3. Choose *Edit – Delete*.

   A red cross symbol appears over the deleted folder.

4. Select the *Delete from build area and reference library* option.
5. Click *Yes*.
6. Export the library to the reference library.

   This will delete the library entries from both the `cds.lib` and the `refcds.lib` file and also physically delete the library folder from the storage area.

Renaming Items in the Reference Area

If you have the necessary permissions in the reference area, Library Explorer allows you to rename libraries or portions of the libraries in your reference area. You must import the library to be renamed into the build area, rename it, and then export it back to the reference area.

1. Import the library to be renamed.
2. Choose *Edit – Rename* to rename the build library.

   The Rename dialog box appears.
3. Click *Yes* in the Rename dialog box to rename the reference library.

   Note: If you do not select *Yes*, Library Explorer creates a new reference library when you export.
4. Export the item to rename it in the reference library.

Entering Log File Information

When you export libraries or parts, Library Explorer creates a log file that contains information about the export.

Library Explorer enters some default information in the log file. The information that is entered includes:

- Login ID of the person who performs the export
- Creation status

  Whether the part has been imported or it is a new part. If the part has been imported, the source library and the time of export are logged.
- Verification status

  The time of verification of the part is logged.

- Modification status

  This is relevant only when the part has been marked for renaming or deletion in the reference area.

To enter additional log file information

1. Click the *Enter log Info* button in the Export Destination dialog box.
2. Specify the additional information.
3. Click *OK.*

   The log file gets created in the exported library location at the same level as the parts of the library. The log file has the same name as that of the library file with the exception that it has a `.log` extension.

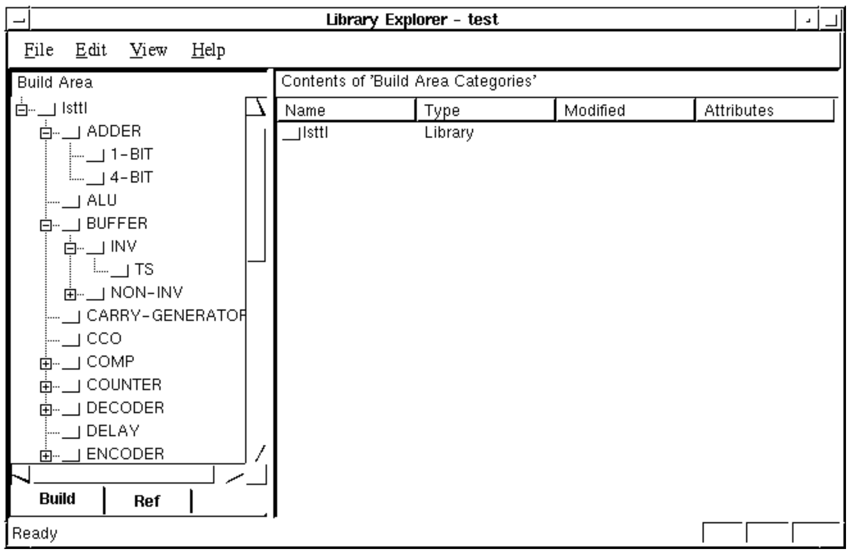
6 Working with Category Files

Category File Overview

You can classify the parts of a library according to some function of the part, such as DECODER, for use with the Component Browser. These subclassifications are called cell categories. Any given cell can be in any number of categories or may not be in any category.

The category information is stored in category files (`.cat` files). Each library has one category file, which is located in the Library directory.

Figure 6-1 Library Explorer in Category View



Contents of 'Build Area Categories'

Name	Type	Modified	Attributes
lsttl	Library		

You can perform these tasks on categories:

- Add new categories.
- Rename categories.
- Delete categories.

- Copy categories.
- Move a category from one library to another library.
- Import categories from another library.
- Export categories to another library.
- Verify categories.

Library Explorer performs the following checks on categories in a selected library:

- Checks for parts that do not fall under categories.
- Checks for those parts that fall under categories but do not exist in the disk.
- Removes stale references from a category file.

Syntax Rules for Working with Category Files

Use the following syntax rules when working with a category file:

- You can arrange categories and parts hierarchically in any order and at any level although typically parts are placed at the end of a category or subcategory.
- You can insert white spaces and lines freely.
- Parts can belong to multiple categories.
- You should contain subcategories in curly braces {}.
- There are no restrictions on category names.
- Use quotes around categories or part names that contain blank spaces, numbers, or non-alphabetic characters.

  For example: 10 MERGE, 5%resistors, or line drivers.
- Names are not case-sensitive.
- Precede comments with a pound sign (#).

Creating a New Category

Library Explorer allows you to create new categories in a library and subcategories within a category. To create a new category:

1. Open a Build area.
2. Choose *View – Categories*.

   Library Explorer displays categories, subcategories, and the associated parts.
3. Select the library or category for which you want to add a new category.
4. Choose *File – New – Category*.

   A tree view item named *New Category* appears.
5. Rename *New Category*.

Deleting a Category

Using Library Explorer, you can delete existing categories. To delete existing categories:

1. Choose the category you want to delete.
2. Choose *Edit – Delete*.

Renaming a Category

Library Explorer provides you the option to rename a category. To rename a category:

1. Select the category you want to rename.
2. Choose *Edit – Rename*. Alternatively, right-click on the selected category and choose *Rename*.
3. Enter the new name and press `Enter`.

Removing Stale References

Library Explorer enables you to remove parts from the category file that do not actually exist on the disk. You have to verify the categories for locating such parts. See Verifying Categories in a Library on page 40 for details.

To remove stale references:

1. Choose *View – Refresh*.
2. Select the library from which you want to remove stale references.

3. Choose *Edit – Remove Stale References*.

Verifying Categories in a Library

Library Explorer performs the following checks on categories in a selected library:

- Checks for any parts that do not fall under categories.
- Checks for those parts that fall under categories but do not exist on the disk.

To verify categories:

1. Select the library in which you want to verify all categories.
2. Choose *View – Refresh*.

- For parts that do not fall under any category, Library Explorer creates a new category named *_Uncategorized*.
- For parts that have been categorized but do not exist in the disk, Library Explorer displays a red check mark to the left of the part name when you view the category.

Importing Categories

A category tree selection (without the parts) can be imported into a library. This feature is useful when you create your own libraries and need to categorize the parts. Then, you can import an existing category file and populate it to create categories of the new library.

To import categories:

1. Select the library in the category view.
2. Choose *File – Import Categories*.
3. Select the file from where you want to import categories.

After you import a category file, you will notice that the categories are not populated. To populate them:

1. Choose *View – Refresh.*

   This will create an *_UnCategorized* folder in the selected library. All the parts of the library will be listed in this folder.

2. Select the *_UnCategorized* folder.

3. Drag and drop the parts into the required categories.
4. Choose *View – Refresh* to ensure that all the parts have been placed into the required categories.

Exporting Categories

Categories in the currently selected library can be saved into a file. This helps you create a template that you can use to categorize other libraries.

To export categories:

1. Select the library in the category view.
2. Choose *File – Export Categories*.

   The Save As dialog box appears.
3. Select the destination location.
4. Specify the filename.
5. Choose *Save*.

7

Using Other Tools from Library Explorer

Launching Tools

To provide a smooth flow in the creation and maintenance of libraries, Library Explorer provides the ability to launch a number of tools.

From Library Explorer, you can:

- Launch Part Developer to create a new part or modify an existing part.
- Launch Allegro Design Entry HDL to view the symbol drawing for a selected part.
- Launch Part Table Editor on a selected part to modify its properties.

Launching Part Developer

You can launch Part Developer:

- to create a new part.
- modify an existing part.

Launching Part Developer to Create a New Part

1. Select a library in the Build area.
2. Choose *Tools – Part Developer*.

The Part Developer tool appears.

Launching Part Developer to Modify a Part

To launch Part Developer on a part:

1. Select the part that you want to modify.

2. Choose *Tools – Part Developer*.

   The Part Developer tool appears.

Note: Part Developer can also be launched by double-clicking on symbol, package, and part table files. If you launch Part Developer from the Reference area, Part Developer opens in read-only mode.

Launching Allegro Design Entry HDL

Design Entry HDL is used to view the symbol drawing of a part.

To launch Design Entry HDL:

1. Select a symbol view of a part.
2. Choose *Tools – Design Entry HDL.*

Launching Part Table Editor

Part Table Editor enables you to create `.ptf` files, which are used to define additional properties on a part.

To launch Part Table Editor:

1. Select a library-level or cell-level PTF file.
2. Choose *Tools – Part Table Editor*.

   The Part Table Editor tool appears. See the chapter on Part Table Editor for more details.

Opening Files using a Text Editor

To open a file in the text editor:

1. Select the file you want to open in a text editor.
2. Choose *Edit – Open*.

   Library Explorer displays the file using the default text editor.
3. Select the checks for *View Verification, Instantiation* and *Packaging, and Verify with Template.* This is a one-time setup for one session.
4. Click *OK*.

After the checks are run, a log file, `cp.msg`, will be created. This log file is stored under the folder `Checkplus` in the project hierarchy. For example, if your project name is `tirallib.cpm` and it is stored at `c:/work`, the `checkplus` folder gets created in `c:/work` hierarchy.

Launching Library-Testing Utilities

After you create or modify libraries, you need to verify them to ensure that they do not have any errors. To run checks on a library or a part:

1. Select the item to be checked.
2. Choose *Tools – Verify*.

   The Verification dialog box appears.
3. Select the type of check you want to run.
4. Click *Options*.

   The Options dialog box appears. Configure the options as required.
5. Click *OK.*

   After the checks are run, a log file, `cp.msg`, will be created. This log file is stored under the folder *Checkplus* in the project hierarchy. For example, if your project name is `triallib.cpm` and it is stored under `c:/work`, then the `checkplus` folder gets created in the `c:/work` hierarchy.

Viewing Physical Properties

Physical properties are defined in the `chips.prt` file. They serve a specific function and are required for the different Cadence tools in the PCB flow to work. You can view the physical properties of a part using Library Explorer. To edit these properties, you have to use Part Developer.

Physical properties are:

- JEDEC_TYPE
- ALT_SYMBOLS

Note: For more information on these properties, see the PCB Systems Properties Reference.

You can also launch Allegro PCB Editor to view a selected footprint. To launch PCB Editor on the footprint, you need to specify the HDL_PSMPATH directive in the project file (.cpm).

You can specify the HDL_PSMPATH directive by clicking on the *Allegro Setup* button in the Physical Properties dialog box.

To view the footprint:

1. Select the part.
2. Choose *View – View Footprint*.

   The Physical Properties dialog box appears, displaying the physical properties.

A

Library Explorer Checks

You can run the following checks:

View Verification

You can run the following checks:

- Symbol origin is centered.
  Checks whether the origin always lies within the symbol and the symbol is at a distance less than the maximum allowed offset from the origin.
- Tristate pins have input and output loads defined.
  Checks the presence of pin properties OUTPUT_LOAD and INPUT_LOAD for every tristate pin. This is denoted by the property OUTPUT_TYPE =TS,TS.
- Mandatory properties present in package file.
  Checks whether the properties named BODY_NAME, PART_NAME, CLASS, and JEDEC TYPE are present in the chips.prt file.
- Consistent symbol name in symbol and package file.
  Checks whether the symbol text is the same as BODY_NAME in the chips.prt file.
- Consistent symbol and package in pin list.
  Checks whether the pins are the same across symbol and package views.

Instantiation and Packaging

Instantiation and Packaging checks include the following:

- Use project ptf files for verification

  If you select this rule, part table files are used in instantiation and packaging.

  If no part table files are specified in the project file, the cell-level ptf is used by default.
- Upto PCB Editor board (netrev)

  If you select this option, the part or library is verified for the complete front-to-back flow.
- Generate pass/fail report

This option is enabled only if you select more than one part or when you select a library that has more than one part.

Note: If you select the *Generate pass/fail* option, each part is verified separately. This is a time-consuming process.

Advanced View Checks

Select this option to launch Rules Checker. You can run your own custom-defined checks using Rules Checker.

VHDL Compilation

Use this option to compile the generated VHDL wrapper. You can use either NCVHDL or CV to compile the wrapper. You can specify the tool to compile the VHDL wrapper in the Enter command in the VHDL compilation dialog box. This dialog box is displayed when you click the Options button in the Verifications dialog box.

Verilog Compilation

Use this option to compile the generated Verilog wrapper. You can use NCVERILOG to compile the wrapper. You can specify the tool to compile the Verilog wrapper in the Enter command for Verilog compilation dialog box. This dialog box is displayed when you click the *Options* button in the Verifications dialog box.

Verify with Templates

Select this option to verify a selected part against a template. The result of the verification is displayed in a report. The verification is done as per the following rules:

Property Checks

The property check is done on all packages for the following:

- All properties listed in the template for a package must exist in each package of the part
- The value of the property in each package must match the value in the template unless the value in template is "?" or blank.

Pin Load Checks

This is done on all pins in all packages as per the following rules:

- If PINUSE="UNSPEC" exists for a pin, all checks are bypassed on that pin
- If a pin type is not determined, it is an error
- If a pin type is determined, its load value is checked against the load value of that pin type in the template. An error is generated if the load values don’t match.
- Error is shown if any of the loads for a pin type is missing

Symbol Checks

All symbols are checked for a given part as per the following rules:

- All symbols must have at least one connection with a line stub or a bubble else an error stating that check cannot proceed is shown.
- All lines are assumed to be vertical or horizontal. Arcs are not supported in this release.
- Each Bubble is interpreted to have two virtual stub lines - horizontal and vertical.
- No two connections can have the same X, Y coordinates else error is shown to the user.
- The location of connections is derived based on the direction of the stubs attached to the connection. Therefore, only connections that have a stub or a bubble on them are checked.
- Stub length is calculated based on the integer average of all stub lengths.

- The outline of the body is derived by searching for the perpendicular line from the end of stub. As the stub size varies, the search is made within the range of the stub size variance. After getting a single outline, the rest are traced as the ones connected the outline end points. The procedure is executed recursively for each detected outline.
- Minimum pin spacing are calculated for all sides (Top, Left, Right, Bottom).
- For pin texts, the property `PIN_TEXT` is used. If it is not found, pin texts are searched for within 1/3 of the average stub length from the end of the stub for each connection. In addition, the location (x,y) of the pin note must not be mis-aligned by more than 1/2 pin spacing.
- Grid is derived by taking the highest common factor of all differences of values of X and Y on respective coordinates. Only the connection (Logic) grid is derived. The symbol grid is not derived even though the template mentions it as Symbol Grid.
- All properties listed in the template for symbols must exist on each symbol in the part.
- The value of the property in each symbol must match the value in the template unless the value in template is "?" or blank.
- The alignment of the property must match the one specified in template.
- The visibility of the property must match the one specified in template.

Pin Checks

Each pin is checked as per the following rules:

- Each pin based on the type as defined in the template must be at the location area in symbol as defined in the template for that type.
- The text size of the pin must match the size specified in the template.
- The *Use Pin Names For Text* is checked only if the template sets it to true.
- The text style for pin text is checked with value in template. If the style is vertical for top and bottom pins and horizontal for pins on left and right then the style is considered to be Automatic. The angles of 90 and 270 are considered equivalent and vertical and 0 and 180 are considered equivalent and horizontal.
- All pins with spacing less than the spacing specified in the template are marked as errors.

Grid Checks

Grid checks are done with the following rules:

- Conversions are done for calculating and matching the grid values for Inches, Metric and Fractional (Fractional is currently not supported in Part Developer). This is then matched with the template value.

Outline Checks

The outline checks are done with the following rules:

- All detected outlines are checked to match the thickness specified in template.

Minimum Size Checks

- The minimum pin spacing values on the left and right is read for each symbol and verified against the minimum pin spacing left and right value stored in the template. An error is generated if the value in the symbol is less than that of the value stored in the template.
- The minimum pin spacing values on the top and bottom is read for each symbol and verified against the minimum pin spacing top and bottom value stored in the template. An error is generated if the value in the symbol is less than that of the value stored in the template.
- The minimum symbol height value is read for each symbol and verified against the minimum symbol height stored in the template. An error is generated if the value in the symbol is less than that of the value stored in the template.
- The minimum symbol height width is read for each symbol and verified against the minimum symbol width stored in the template. An error is generated if the value in the symbol is less than that of the value stored in the template.

The output of the verification is displayed in a dialog box. The output is divided into two sections, Overview and Details.

In the Overview section, the overview of the differences are displayed. In the details section, the differences are detailed.

B
Dialog Box Help

New Project Wizard - Project Name and Location

Enter the name and location of the project on the *Project Name* and *Location* page of the New Project Wizard. This page has two fields, *Project name* and *Location*.

Project name	Enter the name for the project that you are creating. For example My_Project. This field can take any number of characters. However, note that you should not enter special characters or spaces. The special characters are changed to #21 and the spaces are converted to #20. For example, if you enter the project name as My Project, it is converted to My#20Project.
Location	Specify the location of the project.

New Project Wizard - Libraries

This page displays a list of libraries that can be added as reference libraries. The list of libraries is displayed from the Cadence-supplied `cds.lib,` which is stored in the *`$cds_inst_dir`*/`share/library` location.

Note: By default, the displayed libraries are added to the reference library list. To remove any of the displayed libraries from the list of reference libraries, select the library and click *Remove*.

To manage the number of libraries required for reference, use the *Add, Import,* and *Remove* buttons.

Add	Click this button if you want to add more libraries to your list of reference libraries. Clicking *Add* displays a dialog box where you can browse and select the libraries that you want to add.
Import	Click this button if you want to add another library definition file (`cds.lib`) to your reference libraries.
Remove	Click this button to delete the selected libraries from the reference library list.

New Project Wizard - Summary

The *Summary* page displays the details of the project. If you need to change any of the entries, you can go back and incorporate the required changes.

Verification

The *Verification* dialog box enable you to verify the parts on several parameters, such as packaging. For all the checks, click the *Option* button to enable the options.

View Verification	Select the *View Verification* radio button to run the following checks: ■ Symbol origin is centered. Checks whether the origin always lies within the symbol outline and the symbol is at a distance less than the maximum allowed offset from the origin. ■ Tristated pins have input and output loads defined. Checks the presence of pin properties OUTPUT_LOAD and INPUT_LOAD for every tristate pin. The presence of tristated pins is denoted by the property OUTPUT_TYPE =TS,TS. ■ Mandatory properties present in the package file. Checks whether the properties named BODY_NAME, PART_NAME, CLASS, and JEDEC TYPE are present in the `chips.prt` file. ■ Consistent symbol name in symbol and package file. Checks whether the symbol text is the same as BODY_NAME in the `chips.prt` file. ■ Consistent symbol and package in the pin list. Checks whether the pins are the same across symbol and package views.

Instantiation and Packaging	Select the *Instantiation and Packaging* radio button to: ■ Use project ptf files for verification If you select this rule, the part table files are used in instantiation and packaging. If no part table files are specified in the project file, the cell-level ptf is used as default. ■ Upto PCB Editor board (netrev) If you select this option, Part Developer verifies the part or library for the complete front-to-back flow. ■ Generate pass/fail report This option is enabled only if you select more than one part or when you select a library that has more than one part. **Note**: If you select the *Generate pass/fail* option, Part Developer verifies each part separately. This is a time-consuming process.
Advanced View Checks	Select this option to launch Rules Checker. You can run your own custom-defined checks using Rules Checker.
VHDL Compilation (*Available only in PCB Librarian XL*)	Select the *VHDL Compilation* radio button to compile the VHDL wrappers. You can determine the compiler to be used by clicking *Option* and entering the VHDL compiler name, such as NCVHDL.
Verilog Compilation (Available only in PCB Librarian XL)	Select the *Verilog Compilation* radio button to compile the Verilog wrappers. You can determine the compiler to be used by clicking *Option* and entering the VHDL compiler name, such as NCVLOG.
Verify With Template (Available only in PCB Librarian XL)	Select the *Verify With Template* radio button to verify a part against a selected template. You can verify the part against: ■ Property values in a template ■ Pin load values in a template ■ Symbol information in a template

Index