Allegro® Design Entry HDL Reuse Tutorial

Product Version 23.1
September 2023

Contents

Module 1 - Introduction to the Tutorial

This module consists of the following lessons:

Completion Time	30 minutes

Lesson 1-1: Objective of the Tutorial

Objective	In this lesson, you will learn how to use the Allegro Design Entry HDL Reuse Tutorial to create reusable design blocks faster.

Overview

The Allegro Design Entry HDL Reuse tutorial explains the process of creating reusable blocks and using them in different designs. Reusable blocks are existing logical blocks that are associated with at least one placed-and-routed physical module. You can place these blocks within larger designs just as you place components from libraries.

The advantages of using reusable blocks in your design include faster turnaround time and improved quality. The turnaround time decreases because you initiate a reusable block in any design without having to re-create it. The quality is enhanced because you can validate a reusable block at the time of its creation and by the time you use it in your design, design validation issues are resolved.

Tutorial Contents

You can use the Allegro Design Entry HDL Reuse tutorial to perform the steps needed to create a reusable block. The tutorial focuses on how to do the following:

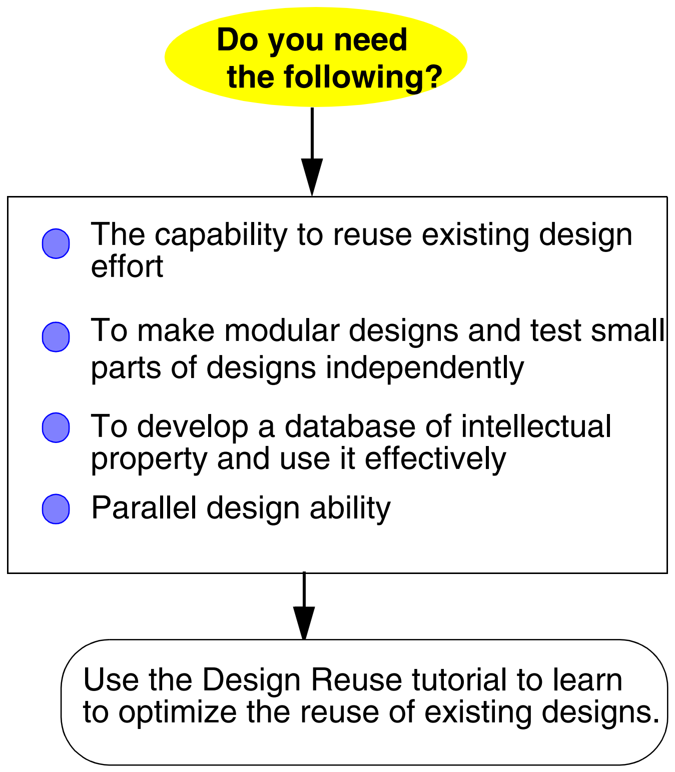
- Create reusable designs (called subdesigns) in Design Entry HDL
- Create reusable modules corresponding to logical subdesigns in Allegro PCB Editor
- Use subdesigns and modules in larger, more complex designs and layouts
- Control design reuse properties in Design Entry HDL and PCB Editor for effective design reuse

Audience

Whether you are a schematic designer or a board designer, you will find that design reuse is an effective way to create complex designs that include independent parts appearing multiple times in the same design or in different designs.

Design reuse involves the use of standalone reusable blocks to create larger, more complex designs.

The Allegro Design Entry HDL Reuse tutorial introduces you to the design reuse flow and enables you to complete all tasks for reusing your designs. You will control design reuse properties to effectively use subdesigns and modules in new designs.


Do you need the following?
The capability to reuse existing design effort
To make modular designs and test small parts of designs independently
To develop a database of intellectual property and use it effectively
Parallel design ability
Use the Design Reuse tutorial to learn to optimize the reuse of existing designs.

Prerequisites

To implement design reuse, you should know how to use the following tools:

- Design Entry HDL (the schematic editor)
- Project Manager (the project creation tool)
- PCB Editor (the board layout tool)

It is assumed that you have worked with the above-mentioned tools, and you can complete basic tasks using these tools. The Allegro Design Entry HDL Reuse tutorial will not explain how to create a schematic, design a layout, or route a design.

Time Required

8 hours for practicing all the exercises in this tutorial

Summary

You can use the Allegro Design Entry HDL Reuse tutorial to learn how to create logical and physical design blocks and reuse them across designs.

What's Next

Go to Lesson 1-2 to learn the benefits of reusing existing PCB designs.

Lesson 1-2: Advantages of Design Reuse

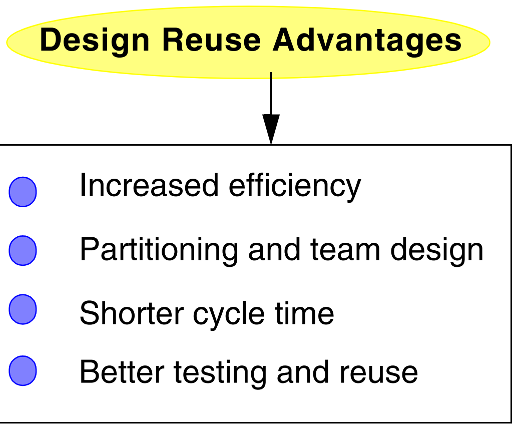
Objective

In this lesson, you will learn the advantages of implementing design reuse.

Overview

Design reuse, as the name implies, is the process of reusing standalone designs in larger designs, and larger designs in increasingly complex hierarchical designs.

Design reuse helps in the following:



- **Simplifying the design process**—You can break a complex design into reusable blocks and test each reusable block independently. This increases quality.
- **Partitioning and team design**—You can use the design reuse methodology to partition large designs into smaller subcircuits that can be logically defined, placed, and routed in parallel, and then brought back into the final board as individual modules. This methodology is useful if you are creating a design in large teams. Different team members can create blocks of designs and these blocks can be reused across the team.
- **Reuse designs across other designs**—Each design you make is a potential input to future development. Rarely, if ever, will you create any design from scratch. When creating any design, you can reuse existing designs or portions of it.

To reuse an existing design, you can make a subdesign in Design Entry HDL and a module in PCB Editor. A module is a placed-and-routed PCB subcircuit. The prime advantage of design reuse is that you need to create the layout of a module only once and you can reuse the module any number of times in different designs.

For example, you will find design reuse particularly useful when creating telecommunication ports, where the same port needs to be used 32, 64, or 128 times in the design.

Important

You may already be using parts of your designs in other designs. This tutorial promotes the synchronization of logical and physical design reuse to derive the benefits discussed above. The flow recommended in the tutorial is intended to serve as an extension of your existing design reuse efforts where you gain considerable additional benefit compared to the existing flow that you might be using in your organization.

Note: To understand the differences between a hierarchical design, which also employs logical design reuse in a very limited sense, and logical and physical design reuse as covered in this tutorial, see the answer to the FAQ What is the difference between creating a hierarchical design and implementing design reuse? on page 82.

Summary

You have learned the advantages of implementing design reuse. You hit the market faster, and develop an intellectual property base. Using reusable blocks, you can implement team design methodology and simplify the design creation process.

What's Next

Go to Lesson 1-3 to learn different steps in implementing design reuse for PCB designs.

Lesson 1-3: Design Reuse Flow

Objective

In this lesson, you will learn the sequence of steps in creating reusable PCB design blocks.

Overview

The design reuse flow involves the entire front to back flow. At a high level, the design reuse flow is based on two sets of broad steps as detailed in the Design Reuse Flow figure on page 14.

Design Reuse Flow

1. **Create a reusable block**—This step involves creating a module in PCB Editor for a logical design in Design Entry HDL.

   You start by defining project settings in Project Manager and creating a design in Design Entry HDL. Next, you package the design and use the packaged files to create a layout. You save this layout as a module and build a logical reuse symbol.

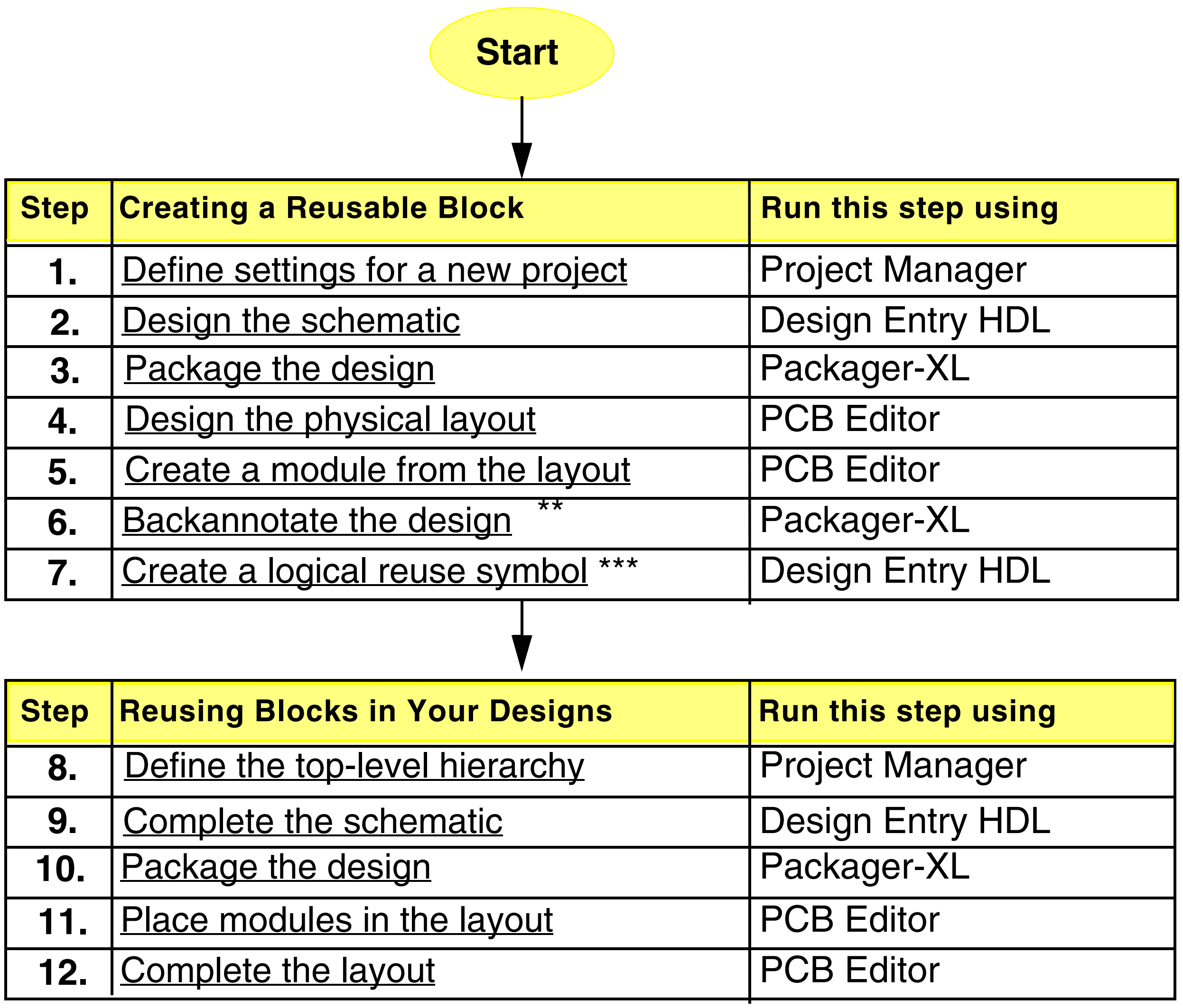
   Note: For more information about these steps, see Module 3 - Creating Reuse Symbols on page 31.

2. **Reuse blocks in your design—**Reuse the logical reuse symbol in another design and create a layout that uses the module you have created.

For more information about these steps, see Module 4 - Reusing the Design on page 51.

You will notice that the design reuse flow includes steps in Project Manager, Design Entry HDL, and PCB Editor. The Design Reuse Flow figure on page 14 includes information about which tool to use to complete each step.

Figure 1-1 Design Reuse Flow



Step	Creating a Reusable Block	Run this step using
1.	Define settings for a new project	Project Manager
2.	Design the schematic	Design Entry HDL
3.	Package the design	Packager-XL
4.	Design the physical layout	PCB Editor
5.	Create a module from the layout	PCB Editor
6.	Backannotate the design **	Packager-XL
7.	Create a logical reuse symbol ***	Design Entry HDL

Step	Reusing Blocks in Your Designs	Run this step using
8.	Define the top-level hierarchy	Project Manager
9.	Complete the schematic	Design Entry HDL
10.	Package the design	Packager-XL
11.	Place modules in the layout	PCB Editor
12.	Complete the layout	PCB Editor

** Backannotating the design is not a mandatory step in the design reuse flow. If you have not made any changes to the layout after the initial packaging of the design, you need not backannotate the changes to the schematic. However, if you make any changes to the property or connectivity information in the layout after the initial packaging of the design, backannotate these changes to the schematic. The changes that require backannotation include connectivity modifications, reference designator changes, and pin/gate swaps.

*** You can create a logical reuse symbol immediately after creating the schematic. However, ensure that the logical reuse symbol is created only after the logic is finalized.

Summary

You have learned the different steps in the Design Reuse flow. Essentially, the flow involves creating a schematic and physical block (complete with layout and routing) that needs to be reused, and then reusing this block in some other design.

What's Next

Go to Lesson 2-1 to learn how to set up environment variables and files for running the tutorial.

2

Module 2 - Getting Started

This module consists of the following lessons:

Completion Time	45 minutes for the written lessons 2 minutes for supporting demonstrations

Lesson 2-1: Installing the Tutorial

Objective	In this lesson, you will learn how to access libraries and symbols, and set the operating system environment variables for running the tutorial.
Overview	To use the Design Reuse tutorial, you need the right tools and you need to access libraries and symbols.

You also need to,

- set operating system environment variables.
- set PCB Editor environment variables.

Right Tools

To set up design reuse, the following tools are required:

- Design Entry HDL
- Project Manager
- PCB Editor

The Allegro PCB Design HDL suite contains these tools.

Accessing Libraries and Symbols ...

To create any design, you need to access libraries, most notably the `standard` library. The examples in this tutorial require you to access a library named `parts_lib`. This library is included in the sample database accompanying this tutorial.

To create any module, you require access to the standard symbols and padstacks. The symbols and padstacks for the examples used in this tutorial are available in a directory named `pcb`.

To copy the libraries and symbols corresponding to the examples used in this tutorial, you need to unzip or untar the sample database accompanying the tutorial. Based on the operating system you are using, perform either of the following steps.

... in Windows

To extract the files to the D: drive, unzip the *des_reuse.zip* file.

Three directories—`site`, `reuse`, and `reuse_archive`–are created. In Lesson 2-2: Understanding the Tutorial Structure, you will explore the structure of these directories.

Setting Environment Variables

When you work with any tool, you need to set its environment variables. Most of the environment variables and the required `PATH` to different executables are set on your system by the installer.

However, there are a few variables that you still need to set. These include `CDS_SITE`, `PATH`, and `HOME`.

Setting CDS_SITE...

Perform the following steps to set the `CDS_SITE` environment variable.

... in Windows

To set the `CDS_SITE` environment variable on a Windows XP computer, follow these steps:

Note: The exact steps for your Windows installation might vary.

1. Right-click the *My Computer* icon on the desktop and select *Properties*.

   The System Properties dialog box appears.

2. Click the *Advanced* tab.
3. Click *Environment Variables*.

   The Environment Variables dialog box appears.

4. Click the *New* button in the System Variables group box.
5. Type `CDS_SITE` in the *Variable* field.
6. Set the `CDS_SITE` to `d:\`*`des_reuse`*`\site` in the *`Value`* field.
7. Click *`OK`*.

   Here, `d` is the drive and *`des_reuse`* is the directory in which you have extracted the zip file.

8. Click *OK* to close the Environment Variables dialog box.
9. Click *OK* to close the System Properties dialog box.

Setting PATH

To use the tools (Design Entry HDL and PCB Editor), ensure that the `PATH` statement in your computer includes the following:

- ❑ `<your_install_dir>/tools/bin`
- ❑ `<your_install_dir>/tools/fet/bin`
- ❑ `<your_install_dir>/tools/pcb/bin`

Setting HOME

You need to set $HOME to the parent directory of the reuse, reuse_archive, and site directories, which is the `des_reuse` directory.

The `des_reuse` directory is created when you unzip or untar the tutorial database.

Note: Besides the environmental variables mentioned above, you need the `MODULEPATH`, `PADPATH`, and `PSMPATH` environment variables to be set for PCB Editor to access the symbols and modules.

You will do this in Lesson 2-4: Setting PCB Editor Environment Variables in the tutorial.

Summary

You learned how to set environmental variables for running Design Entry HDL and PCB Editor in a design reuse environment.

What's Next

Go to Lesson 2-2 to learn how to use the different files and directories in the tutorial database.

Lesson 2-2: Understanding the Tutorial Structure

Objective

In this lesson, you will learn the function of important files and directories in the tutorial database.

Overview

After you uncompress the tutorial zip file, the following folders and files are created.

Figure 2-1 Tutorial Directory Structure

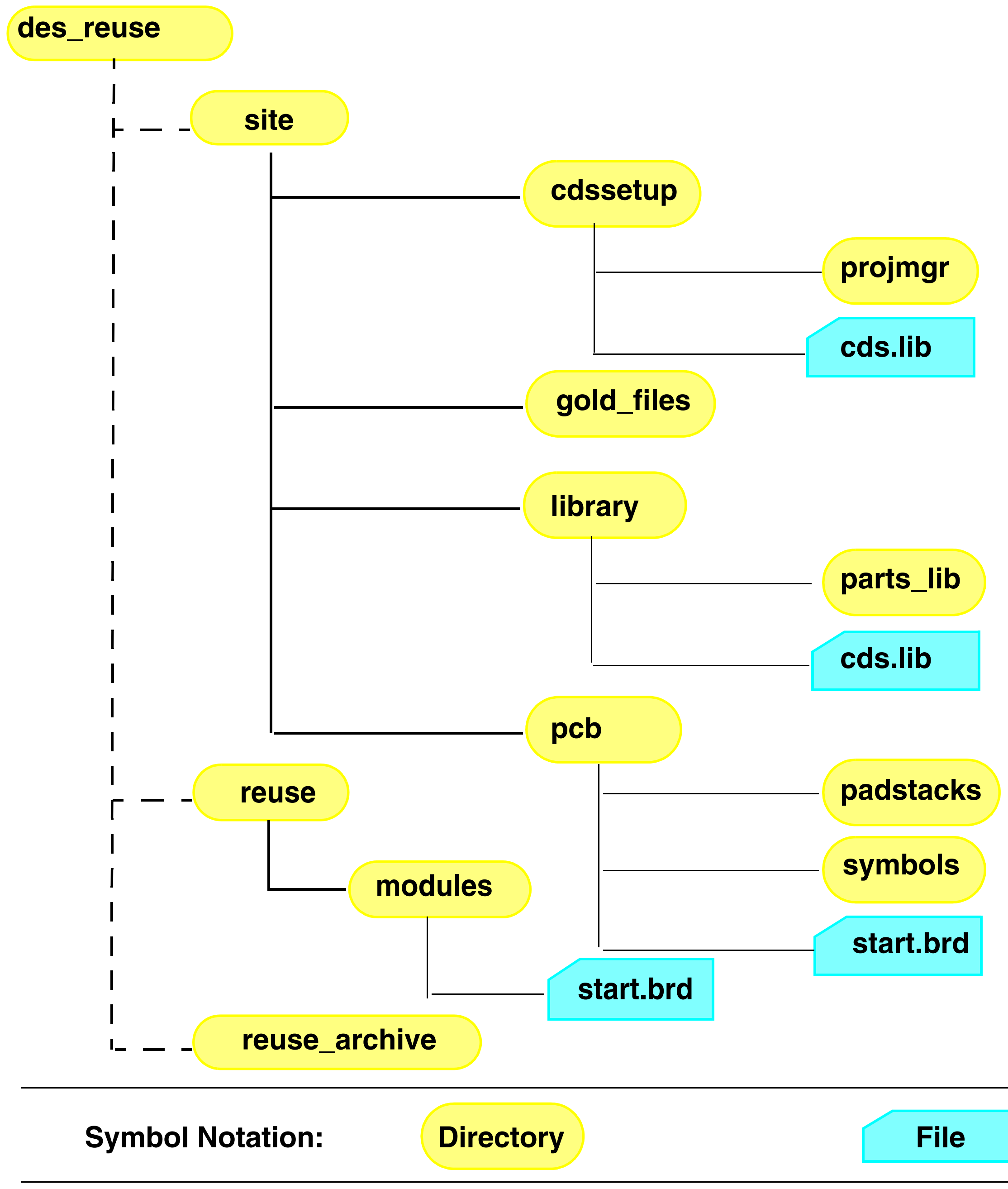
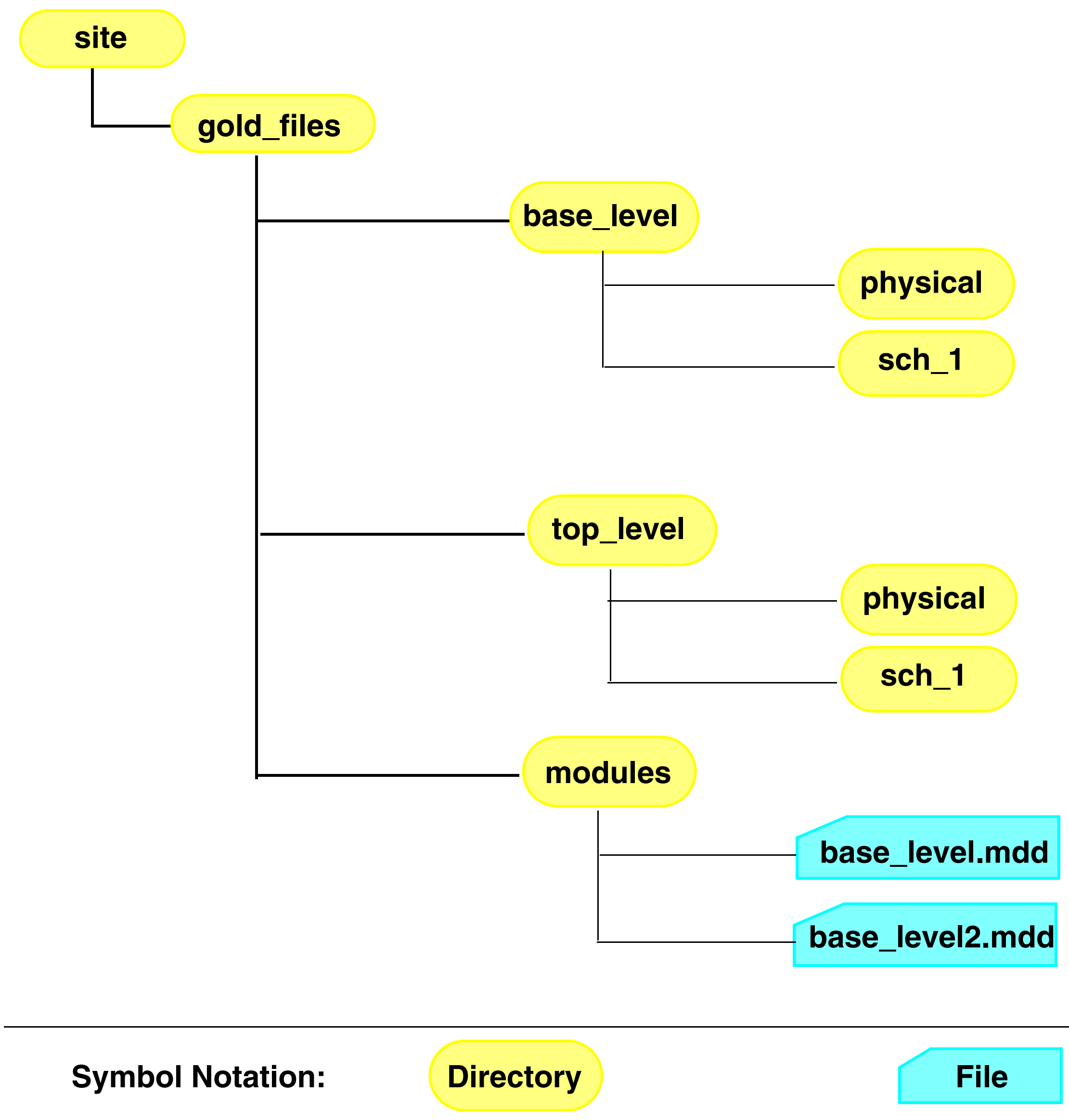


Figure 2-2 Gold Files



Assuming that you have unzipped or untarred the tutorial in the <*des_reuse*> directory, you will see three directories—`site`, `reuse`, and `reuse_archive`.

site directory

The `site` directory (see Figure 2-1 on page 22) consists of four directories—`cdssetup`, `gold_files`, `library`, and `pcb`.

1. **cdssetup**—This directory contains a `projmgr` directory and a file called `cds.lib`. The `projmgr` directory contains the `site.cpm` file, a standard project file that you can use for your site. The `cds.lib` file contains information about the libraries being accessed by your project.
2. **gold_files**—This directory has the output files from each assignment in the tutorial. As the tutorial progresses, you will use the source files to create your own output. The gold files are there in case you wish to skip some assignments and go directly to an advanced assignment.

   All steps required to create these files are detailed in the tutorial. The `gold_files` directory contains three subdirectories—`base_level`, `top_level`, and `modules`. The `base_level` and `top_level` subdirectories each include two subdirectories—`sch_1` and `physical`. The `sch_1` subdirectory contains the `page1.csa` file, which contains the schematic representation of the designs you will create in the tutorial. The `physical` directory contains different board files (pre-placed, placed, and routed) that you will create in the tutorial. The `modules` directory contains two modules that you will create and use in the tutorial.

   You can view the files in the `gold_files` directory. If you want to skip a step such as creating the logical schematic or the board (though it is recommended that you follow all steps), you can copy the files in the `gold_files` directory to the required directory in your design. Details about the files and the directories where they should be located in your design are available in Appendix B, “Sample Designs.”
3. **library—**This directory includes the local library, `parts_lib`, which will be used in this tutorial. This library contains logical symbols. The `library` directory also contains a `cds.lib` file.
4. **pcb—**This directory includes all the symbols and padstacks that you will use in the tutorial. All symbols are stored in the `symbols` subdirectory and all padstacks are stored in the `padstacks` subdirectory. Besides these directories, the `pcb` directory contains the `start.brd` file, which you will use in creating the initial board file.

reuse directory	This directory contains a directory named `modules`, which contains the `start.brd` file. The `modules` directory is the directory where you will store modules that you create in this tutorial. This directory is included in the `MODULEPATH` definition, which allows PCB Editor to read and use these modules. The `start.brd` file is used to create the initial board file.
reuse_archive directory	This directory contains an archived version of the design used in the tutorial. This directory is included for your reference.

Summary

You have learned the function of important files and directories in the tutorial database. Whenever you work with Design Entry HDL or PCB Editor, some key files are generated by the tools, which are stored in standard locations. Knowledge of these files will help you quickly create or modify a design.

What's Next

Go to Lesson 2-3 to learn how to create a project file in Project Manager.

Lesson 2-3: Creating the First Project

Objective

In this lesson, you will learn to create a project file in Project Manager. You will use the project file in all exercises in this tutorial.

Overview

You have learned about the tutorial structure. To create a new design or layout, you need to create a project (`*.cpm`) file. This file stores the names of design libraries, the top-level design name (that is, the root design), tool settings, global settings, and other settings that will be consistently used by the project. You use Project Manager to create a new project file.

Task Overview

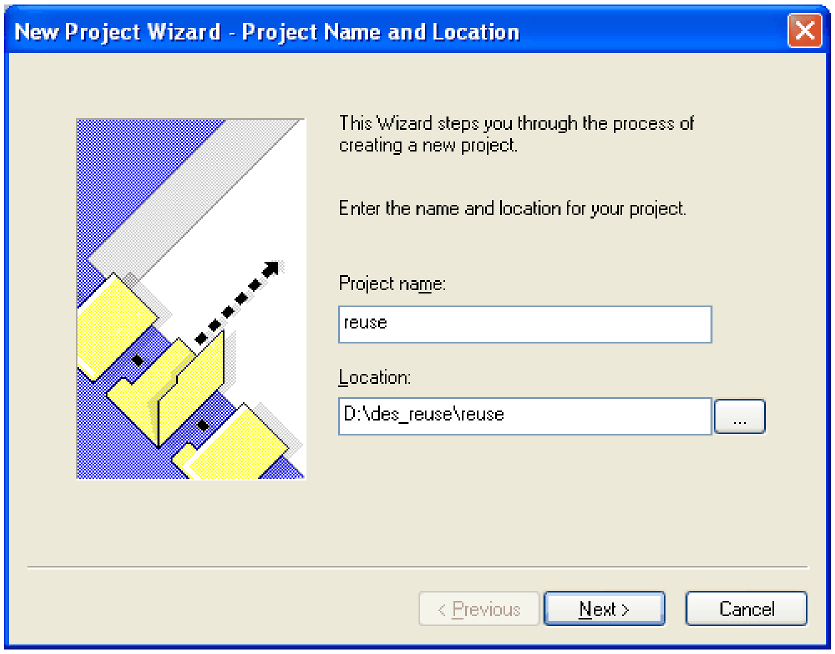
You will create a new project named `reuse` in the `reuse` directory and add the `standard`, `parts_lib`, and `reuse_lib` libraries in the project libraries list. Define the name of the design as `base_level`.

After you have created the project file, you will create a logical schematic and a physical module corresponding to the `base_level` design in Module 3 - Creating Reuse Symbols and reuse the `base_level` design in another design named `top_level` in Module 4 - Reusing the Design.

Procedure

1. Launch Project Manager.
2. Choose *File – New – New Design* to open the New Project Wizard.

   The New Project Wizard starts.



This Wizard steps you through the process of creating a new project.

Enter the name and location for your project.

Project name: reuse

Location: D:\des_reuse\reuse

3. Define `reuse` as the project name, set the location for the project to the `reuse` directory which was created by unzipping or untarring of the tutorial database, and click *Next* to move to the next screen.

   The Project Libraries screen appears. The list of available libraries and the ones used in the project is displayed.

4. To move a library from the *Available Libraries* list to the *Project Libraries* list, select the library, and click the *Add* button. Ensure that the `standard`, `parts_lib`, and `reuse_lib` libraries are listed in the *Project Libraries* list. After moving libraries to the *Project Libraries* list, click *Next* to move to the next screen.

   The Design Name screen appears. In this screen, you choose the library and the cell name that will be used as the top-level drawing for your design. When you create a new project, Project Manager creates a local library for the design. In this case, a library named `reuse_lib` is already selected in the *Library* list. This is the default library created by Project Manager.
5. Specify the design name as `base_level`, and click *Next* to move to the next screen.

   The Summary screen appears detailing your selection. If you want to change any information, click *Previous* to reach the screen where you want to modify information.
6. Click *Finish* to complete defining the settings for your project.
7. Click *OK* in the `New Project creation successful` message box.
8. Click *OK.*

   You have defined the settings for your project. Note that the title bar of Project Manager displays the name of the new project, `reuse.cpm`.

Now try this interactive exercise: Creating a Project.

Summary

You learned how to create a new project in Project Manager. You will use this project for creating schematics and boards.

What's Next

Go to Lesson 2-4 to learn how to set up environment variables in PCB Editor.

Lesson 2-4: Setting PCB Editor Environment Variables

Objective

In this lesson, you will set the `MODULEPATH`, `PADPATH`, and `PSMPATH` environment variables in PCB Editor for successful design reuse.

Overview

The `MODULEPATH`, `PADPATH`, and `PSMPATH` environment variables are set so that PCB Editor can access the symbols and modules used in this tutorial.

Unlike the environmental variables that are populated in the `.cshrc` file, PCB Editor variables are populated in the `pcbenv` directory under the $`HOME` directory.

Task Overview

Set the `MODULEPATH`, `PADPATH`, and `PSMPATH` environment variables as follows:

- Set `MODULEPATH` to:

  `$HOME/reuse/modules`
- Set `PSMPATH` to:

  `$CDS_SITE/pcb/symbols`
- Set `PADPATH` to:

  `$CDS_SITE/pcb/padstacks`

Note: Please note here that you need to replace $CDS_SITE with the value of CDS_SITE variable on your machine.

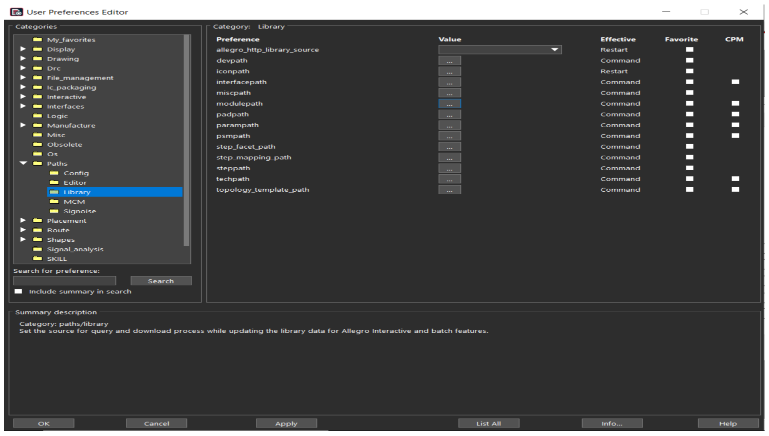
Procedure

1. Click the *Setup* icon in Project Manager.

   The Project Setup dialog box appears.
2. Click the *Tools* tab.

3. Click the *Setup* button next to the *PCB Editor* icon.

   The User Preferences Editor appears. You can now make changes in the design path.



Category: Library

Preference	Value	Effective	Favorite	CPM
allegro_http_library_source		Restart	☐	
devpath	...	Command	☐	
iconpath	...	Restart	☐	
interfacepath	...	Command	☐	☐
miscpath	...	Command	☐	
modulepath	...	Command	☐	☐
padpath	...	Command	☐	☐
parampath	...	Command	☐	☐
psmpath	...	Command	☐	☐
step_facet_path	...	Command	☐	
step_mapping_path	...	Command	☐	
steppath	...	Command	☐	
techpath	...	Command	☐	☐
topology_template_path	...	Command	☐	☐

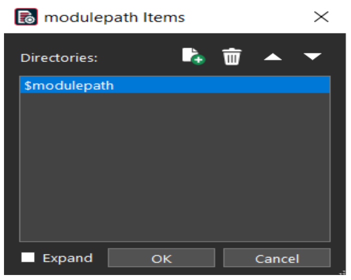
Summary description
Category: paths/library
Set the source for query and download process while updating the library data for Allegro Interactive and batch features.

Tip

You can also access the User Preferences Editor by choosing *Setup – User Preferences* in PCB Editor.

1. Choose *Paths* in the *Categories* box.
2. Choose *Library.*
3. Click the *Value* button corresponding to *modulepath* to specify a value.
4. Click the Add New button.

   A browse button and a blinking pointer appear.



$modulepath

5. Type `$HOME/reuse/modules` and press Enter.
6. Use the *Move Up* (🠉) arrow to move the path you defined to the top of the list. In other words, the path you defined must be the first path in the list.
7. Click *OK*.

   `$HOME` points to your home directory where you have created the `reuse` directory, which contains the design being used in the tutorial.
8. Repeat steps 5 through 7 to specify the value for the `PADPATH` variable as `$CDS_SITE/pcb/padstacks` and the `PSMPATH` variable as `$CDS_SITE/pcb/symbols`.

   Ensure that the path pointed by the `PADPATH` and `PSMPATH` variables must be the first statement in the `PADPATH` and `PSMPATH` declaration.
9. Click *OK* to close the User Preferences Editor.
10. Click *OK* to close Project Setup.

Summary

You have learned how to set the `MODULEPATH`, `PADPATH`, and `PSMPATH` environment variables in PCB Editor. PCB Editor uses these variables to place modules in board files.

What's Next

Go to Lesson 3-1 to learn how to define the schematic block of the design that will be used in other designs.

3

Module 3 - Creating Reuse Symbols

This module consists of the following lessons:

Completion Time

90 minutes for the written lessons

Overview

To reuse a design in another design, first create the design and then store it as a reusable logical and physical symbol. For this, you need to complete the steps displayed in the Creating a Reusable Logical Block figure on page 32.

You have already completed the first step, which involved defining the settings for the new project, in Module 2 - Getting Started. You will complete the remaining steps in this chapter.

Figure 3-1 Creating a Reusable Logical Block

Step	Creating a Reusable Block	Run this step using
1.	Define the settings for the new project	Project Manager
2.	Design the schematic block	Design Entry HDL
3.	Package the design	Packager-XL
4.	Design the physical layout	PCB Editor
5.	Create a module from the layout	PCB Editor
6.	Backannotate the design	Packager-XL
7.	Create a logical reuse symbol	Design Entry HDL

Note: Backannotating the design is an optional step and is required only if you have made any changes to the layout after the initial packaging of the design.

Lesson 3-1: Defining the Schematic Block

Objective

In this lesson, you will learn how to define a schematic block. You will later package this block and use it to create a board, which can be reused in other designs.

Overview

To reuse any design, you need to first create the schematic block corresponding to it. This schematic block is later packaged and used to create both physical and logical symbols. You can later reuse these symbols in other designs.

Task Overview

Create the `base_level` design based on the BASE_LEVEL Logical Design on page 102.

Procedure

1. Click *Design Entry* in Project Manager.

   The Design Entry HDL schematic editor opens. The title bar of Design Entry HDL displays the drawing name `BASE_LEVEL.SCH.1.1.`

   In the drawing name, `BASE_LEVEL` is the cell name, `SCH` is the view name, and 1.1 represent the version and page number.

   A default page border is automatically added for the new page.

   You will now use Design Entry HDL to capture the circuit in the BASE_LEVEL Logical Design figure on page 102.

   Note: If you choose not to create the `base_level` design, you can copy the golden schematic data to your work area. Instructions about copying the data to your work area are provided with the figure.

2. Choose *Component – Add* to display the Add Component dialog box.

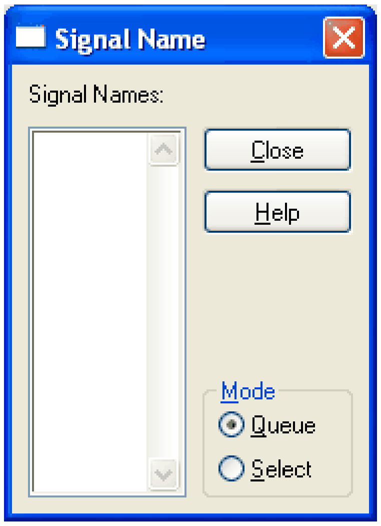
   The Part Information Manager dialog box appears.

.

3. Choose the `F74`, `F08`, and `F04` components from the `parts_lib` library, and the `INPORT` and `OUTPORT` components from the `standard` library. Place all these components on the schematic as displayed in the BASE_LEVEL Logical Design figure on page 102.
4. Choose *`Wire – Draw`* and draw wires between all the components.
5. Choose *`Wire – Signal Name`* to display the Signal Name dialog box.

   The Signal Name dialog box appears.

.



6. Type `CLK_IN`, `START`, `FINISH`, `PAUSE`, `RESET1`, `RESET2`, and `RESET3` to specify signal names, and attach these signals to the middle of the wires as displayed in the BASE_LEVEL Logical Design figure on page 102.
7. Choose *File – Save* to save the design, and then *File – Exit* to exit Design Entry HDL.

Summary

You created a basic schematic block. You can now package the design to create netlists that PCB Editor can import for creating a board file.

Now try this interactive exercise: Creating a Schematic Block.

What's Next

Go to Lesson 3-2 to learn how to package a design that can be reused in other designs.

Lesson 3-2: Packaging the Design

Objective

In this lesson, you will learn how to package a design by using the Export Physical tool.

Overview

After you create a design, you package it using Packager-XL. Packager-XL uses the `chips` view to map the parts in the schematic to physical packages used in the PCB layout. For the purpose of design reuse, you need to package the design as a subdesign, which contains reuse properties that allow it to be reused in other designs. The `GEN_SUBDESIGN` directive is used to create a new subdesign.

Different design reuse operations require effective use of the `GEN_SUBDESIGN`, `FORCE_SUBDESIGN`, and `USE_SUBDESIGN` directives. You will learn when to use these directives in the tutorial. For a quick summary of the three directives, see the answer to the FAQ In which situations should I use the GEN_SUBDESIGN, FORCE_SUBDESIGN, and USE_SUBDESIGN directives? on page 87.

Task Overview

Use Export Physical to package the `base_level` design as a subdesign state file. Ensure Packager-XL retains the packaging assignments made in the previous runs while packaging the design. Use `start.brd` as the input board file and specify `base_level.brd` as the output board file.

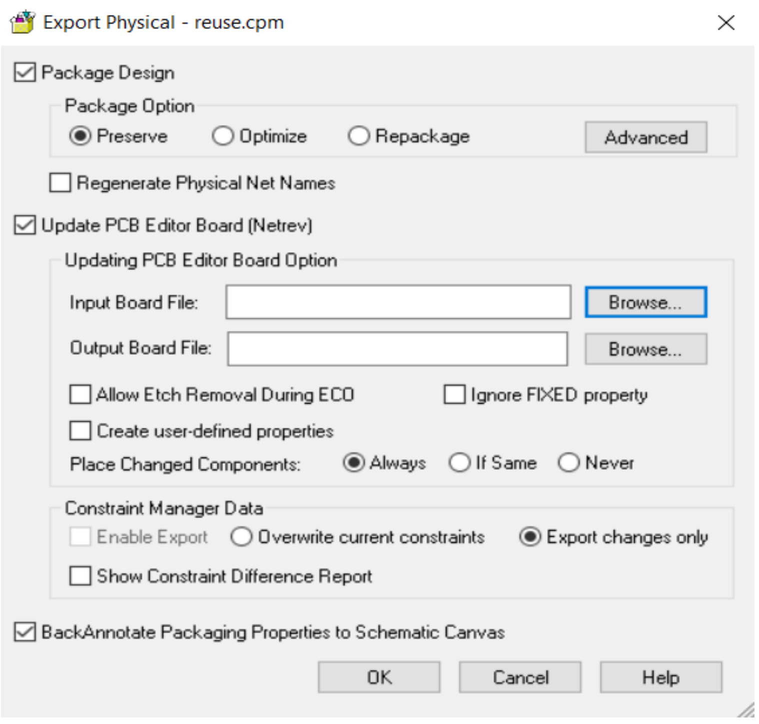
Procedure

1. Launch Export Physical by choosing *Tools – Design Sync – Export Physical* in Project Manager.

   The Export Physical dialog box appears. The name of the design, `reuse.cpm`, is displayed on the title bar. You can use Export Physical to:

   - ❑ Set the packaging options
   - ❑ Package the design
   - ❑ Update the PCB Editor board by running Netrev

.



Export Physical - reuse.cpm

[✓] Package Design
Package Option: [● Preserve] ○ Optimize ○ Repackage — Advanced
[ ] Regenerate Physical Net Names
[✓] Update PCB Editor Board (Netrev)
Updating PCB Editor Board Option
Input Board File: — Browse...
Output Board File: — Browse...
[ ] Allow Etch Removal During ECO   [ ] Ignore FIXED property
[ ] Create user-defined properties
Place Changed Components: [● Always] ○ If Same ○ Never
Constraint Manager Data
[ ] Enable Export ○ Overwrite current constraints [● Export changes only]
[ ] Show Constraint Difference Report
[✓] BackAnnotate Packaging Properties to Schematic Canvas
OK Cancel Help

2. Select the *Package Design* check box to enable packaging.

   You will now specify that the `base_level` design will be used as a subdesign. For this, you must make changes in the Packager Setup - Subdesign tab.

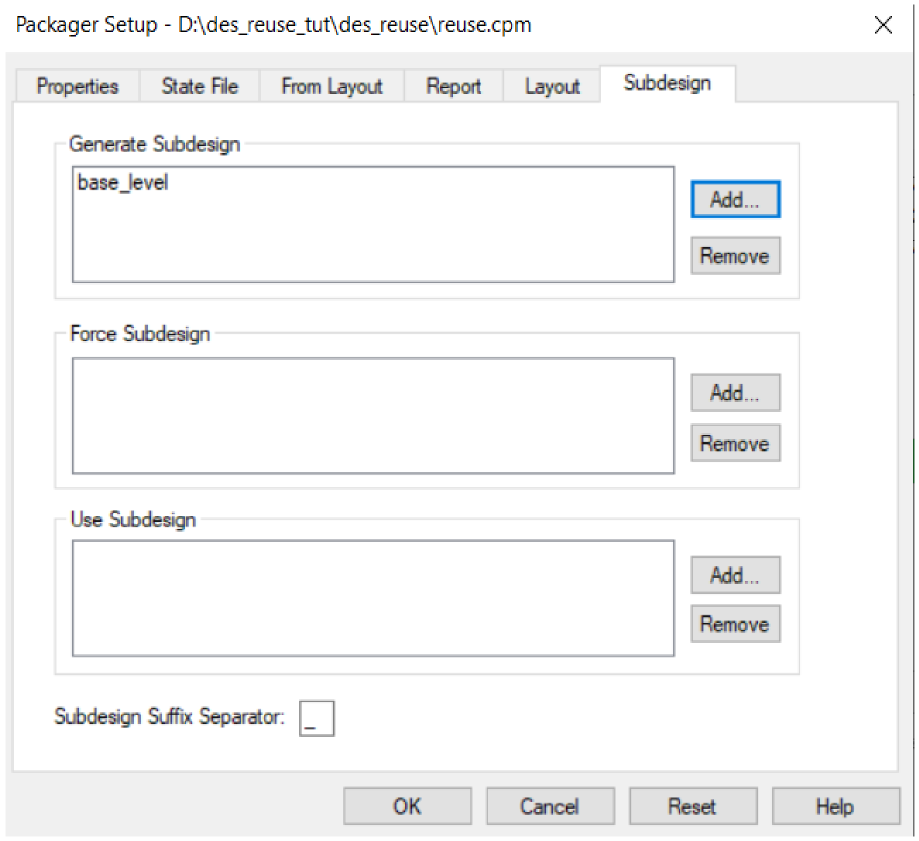
3. Click *Advanced* in the *Export Physical* dialog box.

   The *Packager Setup* dialog box appears.

4. Click the *Subdesign* tab.

The Packager Setup - Subdesign tab appears

.



Generate Subdesign: base_level

Force Subdesign:

Use Subdesign:

Subdesign Suffix Separator: _

5. Click the *Add* button in the *Generate Subdesign* group box.

   The Add Subdesign dialog box appears. You specify the name of the new subdesign here.

6. Type `base_level` in the *Design name* field. Accept the default name if already typed.

7. Click *OK.*

   The name of the subdesign appears in the *Generate Subdesign* group box.

8. Click *OK* to close the *Packager Setup* dialog box.

   You have modified the packaging settings.

   Note: For more information about subdesigns, see the Cadence document *Packager-XL Reference*.

9. Define the following packaging options in the Export Physical dialog box.

   a. Select the *Preserve* option button to specify packaging in the `preserve` mode.

      Note: If you have created modules using the existing packaging assignments, selecting `preserve` as the packaging option retains the existing packaging assignments, which prevents rework in PCB Editor.

   b. Select the *Update PCB Editor Board (Netrev)* check box to specify that Netrev should update the PCB Editor board.

   c. Click *Browse* adjacent to the *Input Board File* field, browse to the `modules` directory (`des_reuse\reuse\modules`), and select the `start.brd` file.

   d. Type `base_level.brd` in the *Output Board File* field.

The Export Physical dialog box includes options to manage ECO changes and electrical constraints. You can use these options as you would use them in the normal packaging flow.

There is an option for backannotating the schematic. The normal design reuse flow does not require backannotation to be a mandatory step. However, if you make changes to the board after the last packaging of the design, you can use the *Backannotate to Schematic Canvas* check box to backannotate the schematic.

10. Start packaging by clicking *OK*.

    The Progress window appears. The following activities are listed.

    a. **Packaging the design**—Packager-XL creates the logical-to-physical assignments. Packager-XL also assigns a physical reference designator to each component by using the packaging properties (`LOCATION`, `SEC`, and `PN`).

    b. **Updating PCB Editor board**—Netrev is launched to read the packaged data, and create or update the PCB Editor board.

A box appears with the message that export has successfully completed.

c. If you want to see results of the Packager-XL run, click *Yes*. Otherwise, click *No.*

Tip It is good practice to check the `pxl.log` and `netrev.lst` files to see if Packager-XL or Netrev has generated any errors or warnings. You may then need to correct your design or layout to fix any errors.

Summary You have learned to use the `GEN_SUBDESIGN` subdirective. The use of this directive while packaging a design ensures that a state file corresponding to the subdesign is created. This file is used in implementing design reuse.

What's Next Go to Lesson 3-3 to learn how to design the layout for a design.

Lesson 3-3: Designing the Physical Layout

Objective In this lesson, you will learn to design the layout for a design using the netlists generated by packager-XL.

Overview You have packaged your design and exported the packaged data to the PCB Editor board. Next, you need to complete the physical layout of the design and route it. For this, PCB Editor will be used.

Task Overview

Use PCB Editor to create a board based on the BASE_LEVEL Physical Design figure on page 104. Save the board file by the name `placed.brd`. Next, route the board and save it with the name `routed.brd`.

Procedure

1. Launch PCB Editor by clicking *`Layout`* in Project Manager.

   You might be prompted to select an option from the Product Choices dialog box. Once you select an appropriate product from the list of choices, PCB Editor is launched displaying the `base_level.brd` file. This file uses the same template as the `start.brd` file. You will now use PCB Editor to place the circuit as shown in the BASE_LEVEL Physical Design figure on page 104.

2. Choose *Place – Manually* to open the Placement dialog box..
3. Click + to the left of the *Components by refdes* list to expand it if the list is not already expanded.

   Components in the design appear in the list.

4. Select the `U1`, `U2`, `U3`, and `U4` components by clicking the check box next to them.

   You have selected the components. You can now move the pointer to a place in the canvas where you want the components to be placed. As you move the pointer, you will note that a component appears at the end of the pointer. To place a component, select the location on the canvas and click there.

5. Complete the circuit displayed in the BASE_LEVEL Physical Design figure on page 104.

   Do not close the Placement dialog box until you have placed all the components.

6. Click *OK* to close the Placement dialog box.

7. Choose *File – Save As* and save the design as `placed.brd` in the `/reuse/worklib/base_level/physical` directory.
8. Manually route the board as shown in the BASE_LEVEL Routed Design figure on page 105.

   Note: You can create modules that do not have any etch. A module in PCB Editor can be fully routed, partially routed, or not routed at all. When you create a module using a routed design, you obtain the maximum benefit of design reuse. However, even if you do not route the board, you can reuse it in other layouts.
9. Choose *File – Save As* and save the board as `routed.brd`.

Now try this interactive exercise: Creating a Layout.

Summary

You have learned how to place components on the board and route them.

What's Next

Go to Lesson 3-4 to learn how to create a module for a physical design.

Lesson 3-4: Creating a Module from the Layout

Objective

In this lesson, you will learn to set the `MODULEPATH`, `PADPATH`, and `PSMPATH` environment variables in PCB Editor for successful design reuse.

Overview

A module is a board that contains special reuse properties, allowing it to be reused in other modules or designs.

Use PCB Editor to create a module for the `base_level` board file, and name the module `base_level.mdd`. Store the module in the directory defined by the environment variable `MODULEPATH`.

Procedure

1. Choose *Tools – Create Module* in PCB Editor to start the process of creating a module.

   PCB Editor responds with the following message in the Console window:

   ```
   Select items for the module.
   ```

   Note: You can run the `create module` command at the Console window to start the process of creating a module.

2. Select all the items for the module.

   OR

   Specify the object selection criteria. The objects you choose depend on what you want to include in the module. Use *Find Filter* to remove the objects to be excluded from the module.

   For the current module, choose all objects *except* groups. To choose available objects, select all check boxes except *Groups* and grayed-out check boxes in the *Find* tab.

The *Find Filter* displays when you select the *Find* tab on the Control Panel. You use the *Find Filter* to find design elements by directly selecting objects in the design or by using the Find By Name/Property dialog box.

The elements in the *Find Filter* that are available for the active command are in bold text and have their check boxes selected. Depending on the command that is active, the elements available for selection change. You can select or deselect any elements by clicking the check box on or off, or you can select or deselect all the elements with the *All On/All Off* buttons.

3. Right-click anywhere in the PCB Editor design window, and choose *Selection Set – Temp Group*.

   To choose all elements within a particular area, click and drag the mouse to draw a rectangle around the elements you want to enclose. For this tutorial, drag a rectangle around all four components (U1, U2, U3, and U4) ensuring that all etches are also included within the rectangle area.

   Tip

   To individually select components and etches, click them. To delete any item from the *Temp Group*, press the `Ctrl` key and click the item to delete. To delete an encapsulated item, keep the `Ctrl` key pressed and click and drag the encapsulated item.

4. Right-click anywhere in the Design Window, and choose *Complete*.

   Note all selected components and edges change to red.

   PCB Editor responds with the following message in the Console Window:

   ```
   Pick Origin
   ```

5. Select the Origin by clicking near the middle of the four components.

   The Save As dialog box appears. Define the location where you want to place the module, and the name of the module.

6. Navigate to the `modules` subdirectory under the `reuse` directory in the *Save in* field. This is the location where the module will be saved.

7. Type `base_level.mdd` in the *`File name`* field to name the module, and click *Save*.

   You have created a module corresponding to the `routed.brd` file.

   It is a good practice to save the module with the same name as the logical design you created in Design Entry HDL. If you do not want to follow this recommendation, you can specify the `REUSE_MODULE` property on the logical symbol as the name of the module. See REUSE_MODULE on page 73 for details.

   Avoid using large module names. Shorten it to the minimum. This will ensure that you do not encounter any errors in unnamed net names while placing the module in another board file.

8. Choose *File – Exit* to close PCB Editor.

   If prompted to save `routed.brd`, click *Yes*.

Now try this interactive exercise: Creating a Module.

Summary

You have learned how to create a module for a board. You can include this module in other board files and thereby reuse existing designs.

What's Next

Go to Lesson 3-5 to learn how to backannotate the changes made in the board back to the schematic.

Lesson 3-5: Backannotating the Design

Objective

In this lesson, you will learn how to backannotate a design.

Overview

Backannotating involves running Packager-XL in the Feedback mode, where all changes in the board after the last packaging run are fed back to the schematic.

Important

Backannotating the design is not a mandatory step in the design reuse flow. If you have made property or connectivity changes to the layout after the initial packaging of the design, you can backannotate your design.

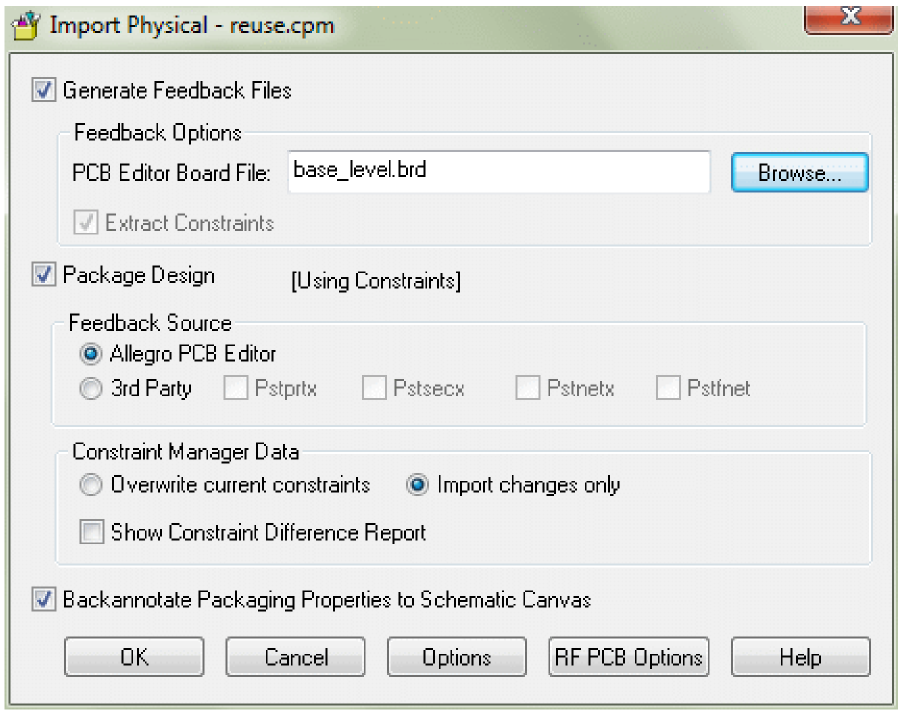
In the current case, the schematic and the board are in sync and, therefore, explicit backannotation is not required. However, if you are not sure whether the schematic and the board are in sync, you can backannotate the design. This lesson is for your practice.

Procedure

1. Launch Import Physical by choosing *Tools –Design Sync – Import Physical* in Project Manager.

   The Import Physical dialog box appears.

.


Import Physical - reuse.cpm

[✓] Generate Feedback Files

Feedback Options

PCB Editor Board File: base_level.brd

[✓] Extract Constraints

[✓] Package Design [Using Constraints]

Feedback Source

(●) Allegro PCB Editor

( ) 3rd Party [ ] Pstprtx [ ] Pstsecx [ ] Pstnetx [ ] Pstfnet

Constraint Manager Data

( ) Overwrite current constraints (●) Import changes only

[ ] Show Constraint Difference Report

[✓] Backannotate Packaging Properties to Schematic Canvas

You can use Import Physical to do the following:

- ❑ Generate the feedback files from a PCB Editor board.
- ❑ Package the design in the Feedback mode where the feedback files generated from PCB Editor are used to update the schematic.

2. Specify the packaging options to use the `routed.brd` file for generating the feedback files.

a. Select the *Generate Feedback Files* check box.

b. Choose `routed.brd` in the *PCB Editor Board File* field to specify the board file*.*

c. Select the *Package Design* check box.

d. Select the *Allegro PCB Editor* option button to indicate the feedback source.

Note that there is an option for backannotating the schematic. The normal design reuse flow does not require backannotation as a mandatory step. If you make changes to the board after the last packaging of the design, you can use the *Backannotate to Schematic Canvas* check box to backannotate the schematic. However, in the current case, do not select this check box.

3. Click *OK* to start packaging the design in the Feedback mode.

The Progress window appears. Note that the following tasks are listed:

- ❑ **Generating feedback files**—In this step, Import Physical runs the PCB Editor extract program and generates feedback files using the Genfeedformat tool.
- ❑ **Feedback the design**—In this step, Import Physical runs Packager-XL in the Feedback mode and packages the physical design.

A message box appears that the import has been successfully completed.

4. Click *Yes* if you want to see results of the Packager-XL run. Otherwise, click *No*.

Now try this interactive exercise: Backannotating the Design.

Summary

You learned how to backannotate a design using Import Physical. You should backannotate a design if you have made changes in properties or connectivity in the board after it has been packaged. Backannotation ensures that the schematic and the board are in sync.

What's Next

Go to Lesson 3-6 to learn how to create a logical reuse symbol from a schematic.

Lesson 3-6: Creating the Logical Reuse Symbol

Objective

In this lesson, you will learn how to create a logical reuse symbol from a schematic using the symbol generator in Design Entry HDL.

Overview

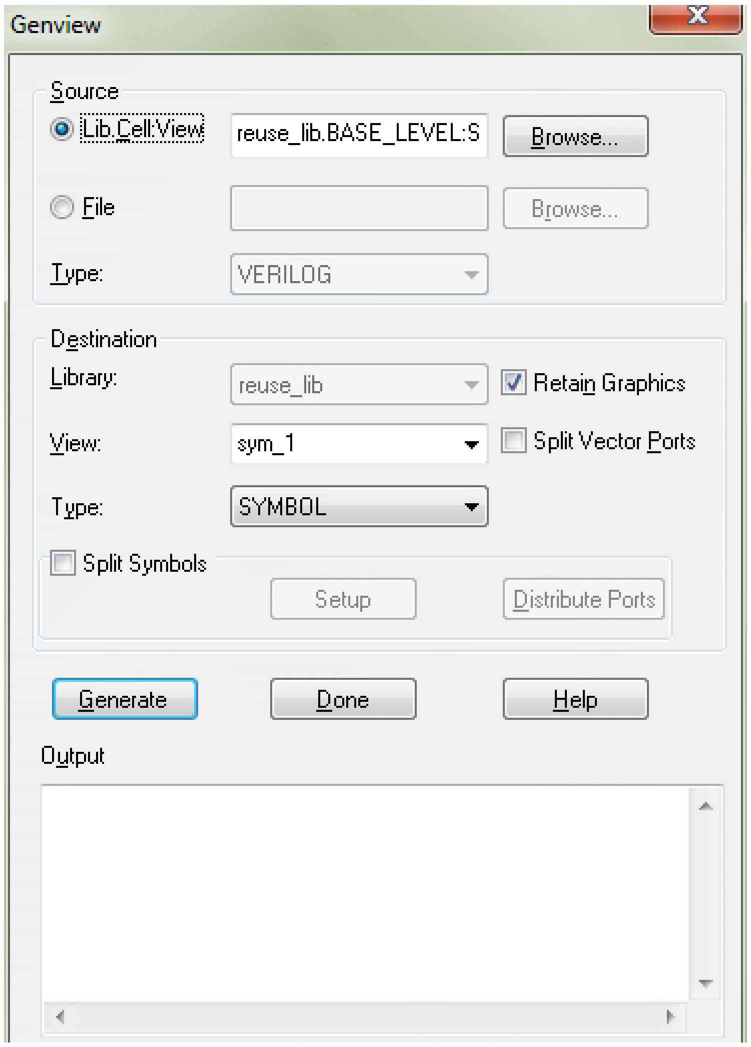
Use the symbol generator in Design Entry HDL to create a logical reuse symbol. Specify the `reuse_lib` library, the `base_level` cell, and the `sch_1` view as the source for the symbol. Store the new symbol in the `reuse_lib` library, the `sym_1` view, and the `SYMBOL` type.

Procedure

1. Launch Design Entry HDL.
2. Choose *Tools – Generate View* to generate a symbol for the existing design.

   The Genview dialog box appears

.



Genview

Source
- (●) Lib.Cell:View: reuse_lib.BASE_LEVEL:S
- ( ) File:
- Type: VERILOG

Destination
- Library: reuse_lib
- View: sym_1
- Type: SYMBOL
- [✓] Retain Graphics
- [ ] Split Vector Ports
- [ ] Split Symbols

3. Set the Genview view options.

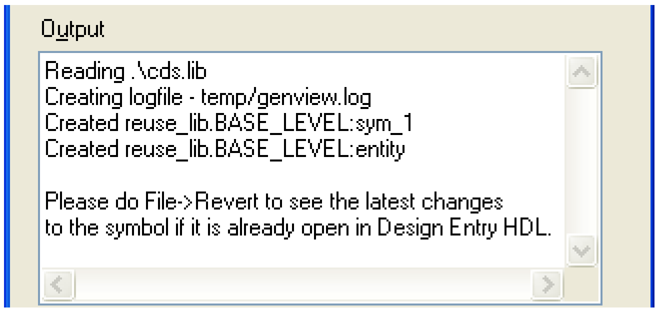
   Most of the following options are automatically selected.

4. Ensure that the *Lib.Cell:View* option button is selected.
5. Ensure that the source is `reuse_lib.BASE_LEVEL:SCH_1`. If it is not set by default, use the Browse button to set this value.

Note: There is a period (.) after the library name (`reuse_lib`) and a colon after the cell name (`BASE_LEVEL`).

6. Check that the destination library is `reuse_lib`.
7. Ensure that `sym_1` is the destination view.
8. Set `SYMBOL` as the destination type.
9. Click *Generate* to start the process of creating the symbol.

   After generating the symbol, Genview displays the following:



```
Reading .\cds.lib
Creating logfile - temp/genview.log
Created reuse_lib.BASE_LEVEL:sym_1
Created reuse_lib.BASE_LEVEL:entity

Please do File->Revert to see the latest changes
to the symbol if it is already open in Design Entry HDL.
```

   Genview also displays a message that the Genview operation is completed.

10. Click *OK* to close the Genview message box.
11. Choose *Done* to close the Genview dialog box.
12. Choose *File – Exit* to exit Design Entry HDL.

You have created a symbol for the `BASE_LEVEL` design. If you browse to the `sym_1` subdirectory in the `base_level` directory, you will find two new files, `master.tag` and `symbol.css`, which contain symbol information.

Now try this interactive exercise: Creating a Logical Symbol.

Summary

You learned to create a logical reuse symbol for a schematic. A logical reuse symbol is equivalent to creating a new component. You can use the logical reuse symbol in other schematics and thereby implement design reuse.

What's Next

Go to *Lesson 4-1* to learn how to change the design name for a project in Project Manager.

4

Module 4 - Reusing the Design

This module consists of the following lessons:

Completion Time	90 minutes for the written lessons

Overview

To reuse a design in another design, you have to create the design and store it as a reusable logical and physical symbol. In Module 3 - Creating Reuse Symbols on page 31, you learned to create a reusable logical symbol and module.

After creating a reusable logical symbol and module, you use them in a top-level design using the process covered in the Reusing Blocks in Another Design figure on page 52. In the current chapter, you will learn to use the reusable symbol in another design. You will also learn to find design differences between the schematic and the board.

Figure 4-1 Reusing Blocks in Another Design

Step	Reusing Blocks in Your Designs	Perform step in:
1.	Define the top-level hierarchy	Project Manager
2.	Complete the schematic	Design Entry HDL
3.	Package the design	Packager-XL
4.	Place modules in layout	PCB Editor
5.	Complete the layout	PCB Editor

Lesson 4-1: Defining the Top-Level Hierarchy

Objective

In this lesson, you will learn how to change the design name for a project.

Overview

While implementing design reuse, you often work with different designs. For example, you will have the design that you intend to reuse and you will have designs in which you intend another design. While you might have multiple designs in a project, you can work with only one design at one time. To work with different designs, you switch them by changing the design name in Project Manager.

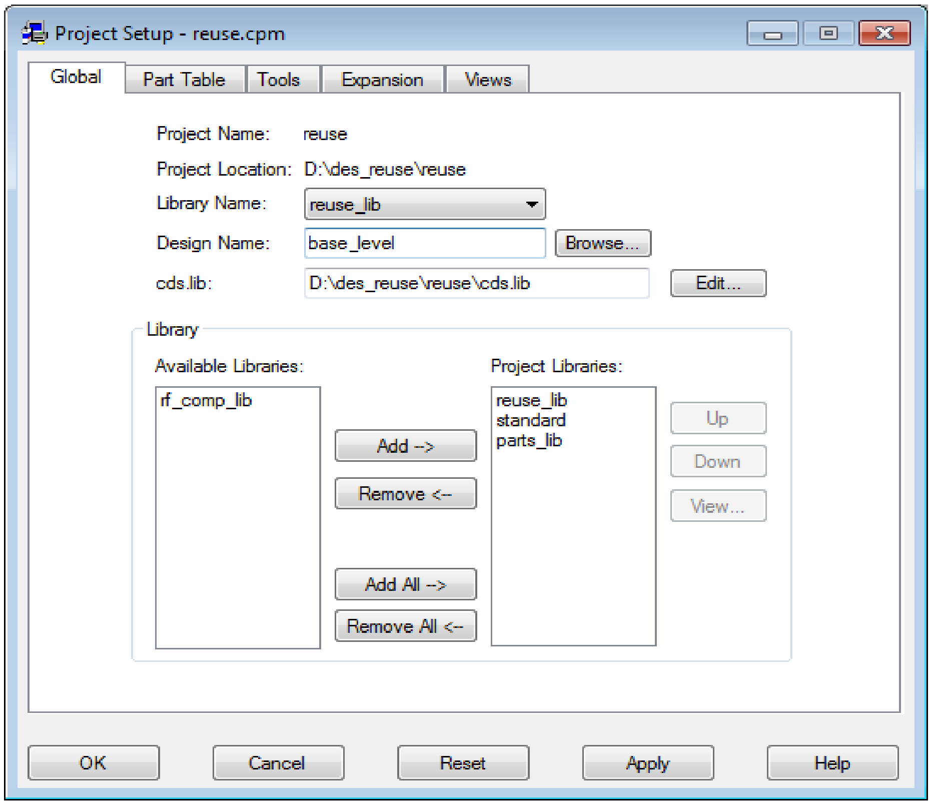
Task Overview

Specify the name of the new design as `top_level` in Project Manager.

Procedure

1. Click *Setup* in Project Manager.

   The Project Setup dialog box appears with the Global tab selected. Note that the name of the design is `base_level`. You will change this name to `top_level`.



Project Name: reuse
Project Location: D:\des_reuse\reuse
Library Name: reuse_lib
Design Name: base_level
cds.lib: D:\des_reuse\reuse\cds.lib

Library
Available Libraries: rf_comp_lib
Project Libraries: reuse_lib, standard, parts_lib

2. Type `top_level` in the *Design Name* field.
3. Click *OK.*

You will notice that a new subdirectory named `top_level` is created in the `worklib` directory. The design name in the project file (`reuse.cpm`) is also changed to `top_level`.

Now try this interactive exercise: Changing the Design Name.

Summary

You changed the design name for a project. Changing design names helps you work with multiple designs in a project. You can implement a hierarchical design or team design using multiple designs in a project.

What's Next

Go to Lesson 4-2 to learn how to create a schematic for the top-level block, where you will import logical reuse symbols.

Lesson 4-2:Defining the Schematic Block

Objective

In this lesson, you will learn how to define a schematic block for the top-level design. A top-level design in a hierarchical environment often references other designs, which have already been created and saved as logical symbols.

Overview

You have defined the design name for the top-level schematic. You can now define the schematic that will use the `base_level` schematic as a subdesign.

Create the `top_level` design based on the TOP_LEVEL Logical Design figure on page 106.

Procedure

1. Click *Design Entry* in Project Manager.

   The Design Entry HDL schematic editor opens. Note that the title bar of Design Entry HDL displays the drawing name `TOP_LEVEL.SCH.1.1`.

2. Choose *Component – Add* to display the Part Information Manager.
3. Select `reuse_lib` in the *Library* list.
4. Select `BASE_LEVEL` as the cell name.

   The `BASE_LEVEL` component is attached to the pointer. You can now place it on the schematic.
5. Place two `BASE_LEVEL` symbols on the schematic and complete the design as displayed in the TOP_LEVEL Logical Design figure on page 106.
6. Choose *File – Save* to save the design.

   The `sch_1` view is created in the `top_level` cell, which is in the `worklib` library.
7. Choose *File – Exit* to exit Design Entry HDL.

Summary

You created the top-level schematic block and reused the symbols that already existed in it.

What's Next

Go to Lesson 4-3 to learn how to package a design that contains reused subdesigns.

Lesson 4-3: Packaging the Design

Objective

In this lesson, you will learn to package a design that contains subdesigns.

Overview

You have created the design for the top-level schematic. You must now package the design so that Packager-XL can assign reference designators to the components in the `base_level` block, which is being reused.

Task Overview

Package the `top_level` design. Ensure that packaging in the subdesign state file is applied to each instance of the subdesign. Update the board using the `start.brd` file as the input board file and specify that the output board file will be named as `top_level.brd`.

Procedure

1. Launch Export Physical by choosing *Tools –Design Sync – Export Physical* in Project Manager.
2. Select the *Package Design* check box to enable packaging.

   Specify that the `base_level` design will be used as a subdesign and that packaging in the subdesign state file will be applied to each instance of the subdesign. For this, make changes in the Packager Setup - Subdesign tab.

   a. Click *Advanced* in the Export Physical dialog box.

      The Packager Setup dialog box appears.

   b. Click the *Subdesign* tab.

   c. Click the *Add* button in the *Force Subdesign* box.

      The Add Subdesign dialog box appears.

   d. Type `base_level` in the *Design name* field and click *OK.*

      The name of the subdesign is filled in the *Force Subdesign* box. This ensures that the packaging in the `base_level` subdesign state file will be applied to all instances of `base_level` in the `top_level` design.

   e. (Optional) You can remove `base_level` from the *Generate Subdesign* box. If you retain `base_level` in the *Generate Subdesign* box, a subdesign state file for the `base_level` will be produced.

   Note: If you retain any design in the *Generate Subdesign* box that is not the current root-level design, Packager-XL generates a warning message specifying that an invalid module name is specified in the `GEN_SUBDESIGN` directive.

   If you want to reuse the `top_level` design in some other designs, add `top_level` to the *Generate Subdesign* list.

   For more information about the `GEN_SUBDESIGN`, `FORCE_SUBDESIGN`, and `USE_SUBDESIGN` directives, see the answer to the FAQ In which situations should I use the GEN_SUBDESIGN, FORCE_SUBDESIGN, and USE_SUBDESIGN directives? on page 87

3. Define the packaging options in the *Export Physical* dialog box.

   a. Select the *Preserve* option button to specify packaging in the `preserve` mode.

   b. Select the *Update PCB Editor Board (Netrev)* check box to specify that Netrev must update the PCB Editor board.

   c. Click the *Browse* button adjacent to the *Input Board File* field, browse to the `modules` directory, and choose the `start.brd` file.

   d. Type `top_level.brd` in the *Output Board File* field.

      If the `PADPATH`, `PSMPATH`, and `MODULEPATH` environment variables are not set properly, packaging may not complete properly. To set environment variables, see Lesson 2-4: Setting PCB Editor Environment Variables on page 28.

4. Start packaging by clicking *OK*.

   Packager-XL completes packaging.

5. A message box appears that Export has successfully completed. If you want to see results of the Packager-XL run, click *Yes*. Otherwise, click *No*.

Summary

You have learned to use the `USE_SUBDESIGN` subdirective. The use of this directive ensures that the subdesign specified in the directive is packaged and reused in the design.

What's Next

Go to Lesson 4-4 to learn how to create the physical layout for the top-level design that uses existing modules.

Lesson 4-4: Designing the Physical Layout

Objective

In this lesson, you will learn to create the physical layout for the top-level design that has modules placed in it.

Overview

Use PCB Editor to create a board based on the TOP_LEVEL Physical Design figure on page 108. Save the board file with the name `placed.brd`. Next, route the board and save it with the name `routed.brd`.

Procedure

1. Launch PCB Editor by clicking the *`Layout`* button in Project Manager.

   PCB Editor opens displaying the `top_level.brd` file. This file uses the same template as the `start.brd` file.

2. Use PCB Editor to place the circuit in the TOP_LEVEL Physical Design figure on page 108.

   a. Choose *Place – Manually* to open the Placement dialog box.

b. Click the + sign to the left of the *Components by refdes* list to expand it.

A list of components, which includes `JT1` and `JT2,` is displayed.

c. Choose `JT1 and JT2` by clicking the check boxes next to them.

d. You have selected the components. Choose the `BASE_LEVEL` modules now.

Note: If you get an error message that the symbol for the selected component or module is not available, then the environment variables are not properly set. Ensure that the `PADPATH`, `PSMPATH`, and `MODULEPATH` environment variables are set as described in Lesson 2-4: Setting PCB Editor Environment Variables on page 28.

e. Click the + sign to the left of the *Module instances* list to expand it.

Two modules `BASE_LEVEL/BASE_LEVEL_1` and `BASE_LEVEL/BASE_LEVEL_2` appear.

Note: The text before the / character (`BASE_LEVEL`) represents the module name, and the text after the / character (`BASE_LEVEL_1` and `BASE_LEVEL_2`) represents the instance name. You can specify an instance name in the schematic. However, if you do not specify the instance name, Packager-XL automatically assigns the instance name as the module name and appends it by a number incrementing from `1`. For more information about design reuse properties, see Lesson 5-1 Understanding Design Reuse Properties on page 72.

f. Complete the circuit.

g. Click the + sign to the left of the *Components by refdes* list to expand it.

Four components with refdes `U1`, `U2`, `U3`, and `U4` appear in the list.

h. Place components on the canvas.

As you place the components on the canvas, the check boxes corresponding to them are automatically cleared in the Placement dialog box.

3. Choose *File – Save As* and save the design as `placed.brd`.

You have created the layout for a design that reuses modules that you have already created. These modules offer you the maximum benefit of design reuse as you get the synchronization of the logical and physical design. If you want, you can route the board.

4. Manually route the board as shown in the TOP_LEVEL Routed Design figure on page 109.

Note: If you choose not to route the board, you can copy the golden board data to your work area.

5. Choose *File – Save As* and save the board as `routed.brd`.

You can import or export subdesigns from within PCB Editor. For more information about importing and exporting subdesigns, see the answer to the FAQ How can I import or export subdrawings from within PCB Editor? on page 92.

Summary

You have learned how to place components and modules on a board and route them.

What's Next

Go to Lesson 4-5 to learn how to create differences in the board and the schematic.

Lesson 4-5: Creating Design Differences

Objective

In this lesson, you will learn to create differences in the board and the schematic.

Overview

You have created the layout for the `top_level` design and stored it as `placed.brd`. You also have routed `placed.brd` and saved the resulting board as `routed.brd`. If you backannotate this board, it will not show any differences.

However, in real life, you often can make changes in the board. For example, you add shunt terminators or make refdes (reference designator) changes, which cause differences between the board and the schematic. You can detect design differences using the Design Synchronization tool. In this procedure, you will create design differences and in the next procedure, Lesson 4-6: Resolving Design Differences on page 63, you will resolve those differences.

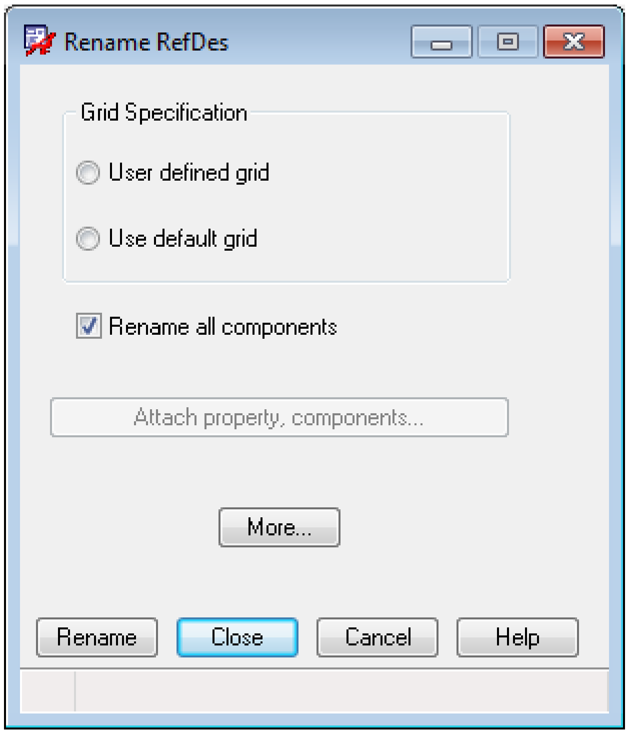
Task Overview

Use PCB Editor to rename reference designators in the `routed.brd` board file, and save the new board file as `renamed.brd.`

Procedure

1. Choose *Logic – Auto Rename Refdes – Rename*.

The Rename RefDes dialog box appears.



[✓] Rename all components

Rename all components is selected by default. Because you will rename all components, leave the default selection unchanged.

2. Click the *Rename* button to rename all components.

   Note that the following message appears in the Console Window:

   ```
   Auto Rename of Reference Designators IN PROGRESS
   ```

   Wait until PCB Editor completes renaming reference designators. When all reference designators are renamed, PCB Editor displays the following message:

   ```
   Auto rename of Refdes COMPLETE. 10 components renamed.
   ```

3. Click *Close* in the Rename RefDes dialog box.

   The reference designators for all components are changed.

4. Choose *File – Save As* and save the board as `renamed.brd`.

You can check the reference designators in Design Entry HDL. While the reference designators have changed in PCB Editor, they have not changed in Design Entry HDL.

Summary

You have learned how to create design differences between the board file and the schematic.

What's Next

Go to Lesson 4-6 to learn how to resolve differences between the board file and the schematic.

Lesson 4-6: Resolving Design Differences

Objective

In this lesson, you will learn to synchronize the board file and the schematic.

Overview

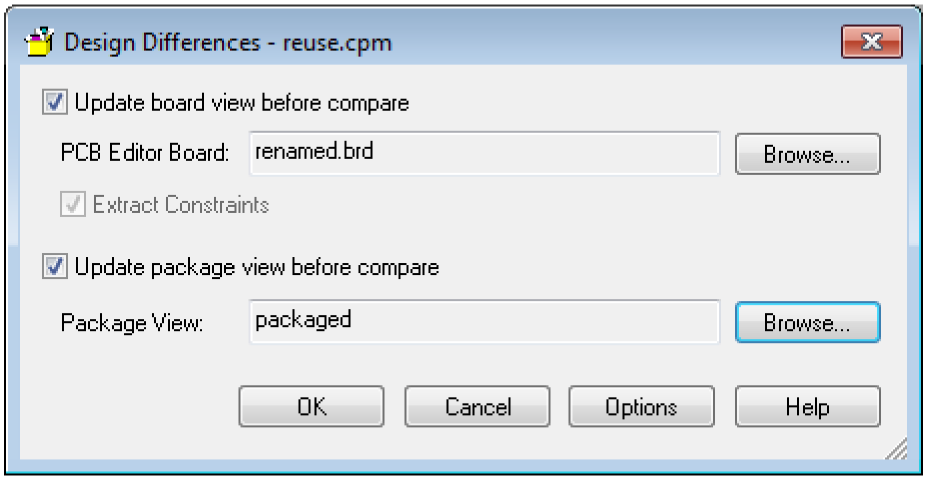
Use Design Differences to synchronize the schematic and the board file.

Procedure

1. Launch Design Differences by choosing *Tools – Design Sync – Design Differences* in Project Manager.

   The Design Differences dialog box appears. You can now update the schematic or the board, or find differences between them.

.


Design Differences - reuse.cpm

☑ Update board view before compare

PCB Editor Board: renamed.brd

☑ Extract Constraints

☑ Update package view before compare

Package View: packaged

Set Design Differences to find differences between the schematic and the board.

2. Select the *Update board view before compare* check box.
3. In the PCB Editor Board field, click the *Browse* button and navigate to `/des_reuse/reuse/worklib/top_level/physical`. Choose `renamed.brd` and click `OK`.
4. Deselect the *Update package view before compare* check box.

   Since the files in the *packaged* view were updated in the Lesson 4-3: Packaging the Design on page 55, you need not select the *Update package view before compare* check box. However, if you need to update the *packaged* view of the schematic design before comparing the schematic and the layout, select the *Update package view before compare* check box.
5. Click *OK* to start Design Differences.

   A Progress window appears. First, Design Differences imports the design from PCB Editor. Next, it generates the design differences. Finally, the Design Differences window appears displaying the Message Log and RefDes Difference windows. The RefDes Difference window lists all components that have a different `LOCATION` property in the schematic and the board.

   Tip You can choose a difference in Design Differences and find the corresponding component in Design Entry HDL and PCB Editor.
6. Ensure that Design Entry HDL is running. If Design Entry HDL is not running, click *Design Entry* in Project Manager.

   Design Entry HDL opens the `TOP_LEVEL` design.

Note that a host of new properties appear on the design. You can now highlight a difference in Design Differences and the corresponding component will be selected in both Design Entry HDL and PCB Editor.

7. Double-click the first row in the RefDes Difference window in Design Differences to highlight the corresponding components in Design Entry HDL and PCB Editor.

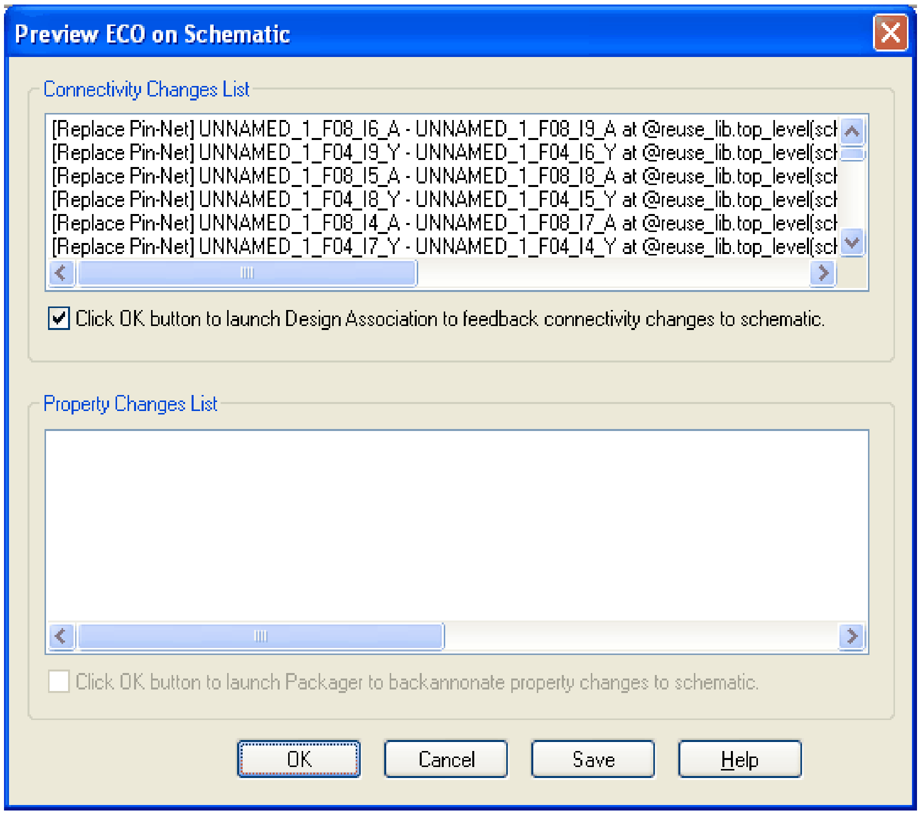
   The component corresponding to the highlighted difference appears in red in Design Entry HDL. The same component is also highlighted in PCB Editor and a message stating the same also appears in the Console Window.

   You have seen the design differences between the schematic and the board. Now, you can resolve these design differences.

8. Choose *Sync – Update Design Entry Schematic* to update the schematic with the latest information contained in the board.

   The Preview ECO on Schematic dialog box appears.

.



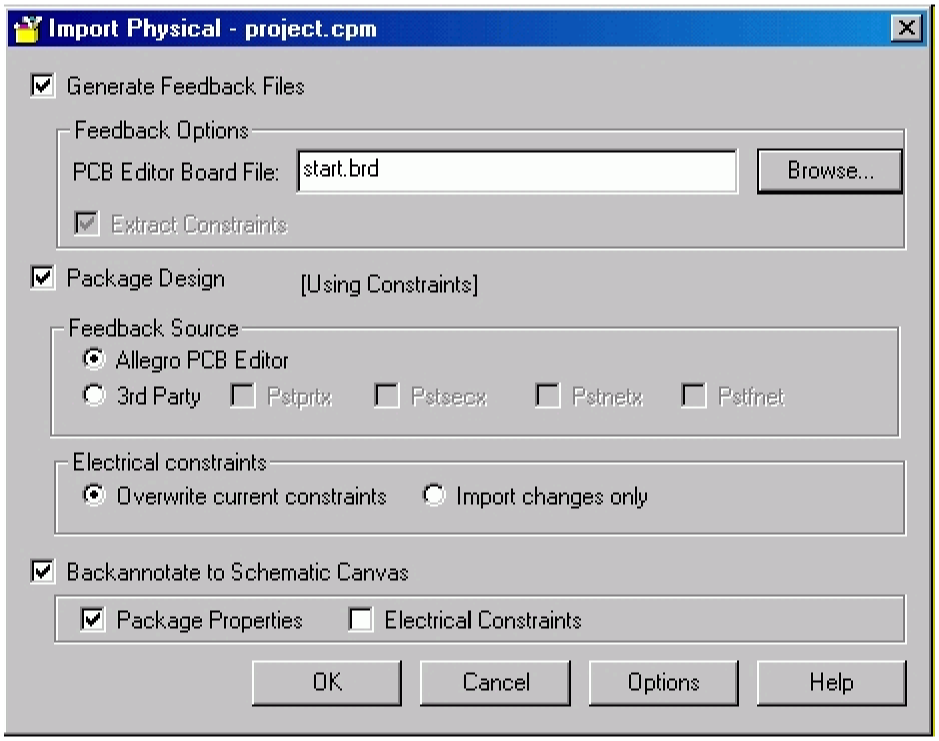
```
[Replace Pin-Net] UNNAMED_1_F08_I6_A - UNNAMED_1_F08_I9_A at @reuse_lib.top_level(sch
[Replace Pin-Net] UNNAMED_1_F04_I9_Y - UNNAMED_1_F04_I6_Y at @reuse_lib.top_level(sch
[Replace Pin-Net] UNNAMED_1_F08_I5_A - UNNAMED_1_F08_I8_A at @reuse_lib.top_level(sch
[Replace Pin-Net] UNNAMED_1_F04_I8_Y - UNNAMED_1_F04_I5_Y at @reuse_lib.top_level(sch
[Replace Pin-Net] UNNAMED_1_F08_I4_A - UNNAMED_1_F08_I7_A at @reuse_lib.top_level(sch
[Replace Pin-Net] UNNAMED_1_F04_I7_Y - UNNAMED_1_F04_I4_Y at @reuse_lib.top_level(sch
```

[✓] Click OK button to launch Design Association to feedback connectivity changes to schematic.

[ ] Click OK button to launch Packager to backannonate property changes to schematic.

9. Click *OK* to update the differences.

The Message log in the Design Differences window is updated and the Import Physical dialog box is displayed.



Import Physical - project.cpm

[✓] Generate Feedback Files

Feedback Options

PCB Editor Board File: start.brd

[✓] Extract Constraints

[✓] Package Design [Using Constraints]

Feedback Source

(●) Allegro PCB Editor

( ) 3rd Party [ ] Pstprtx [ ] Pstsecx [ ] Pstnetx [ ] Pstfnet

Electrical constraints

(●) Overwrite current constraints ( ) Import changes only

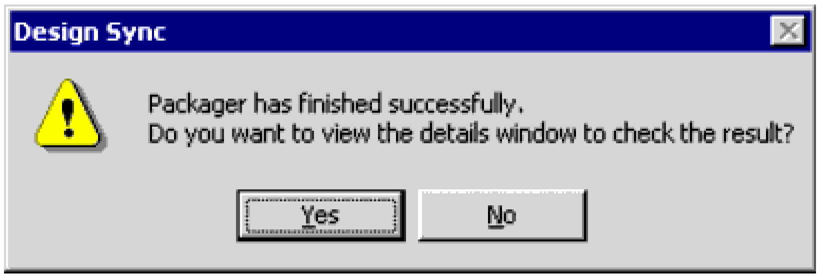
[✓] Backannotate to Schematic Canvas

[✓] Package Properties [ ] Electrical Constraints

Note that you are not allowed to generate the feedback files. There is an option to backannotate the schematic. Do not select this option as then you would not be able to see the differences between the schematic and the board.

10. Click *OK*.

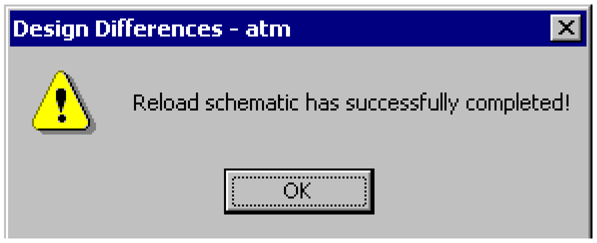
A Progress Window appears stating that design is netlisted and being fed back. Finally a message box appears asking whether you want to see Packager results.



Packager has finished successfully.
Do you want to view the details window to check the result?

11. Click *No*.

The control is passed back to Design Differences, which displays a message that schematic has successfully loaded.



Reload schematic has successfully completed!

12. Click *OK*.

Design Differences compares the board and the schematic information. It has detected no differences between the schematic and the `renamed.brd` file.

13. Click *OK* to close the message box.

14. You can run the *File – Update Differences* command in Design Differences to confirm that differences do not exist between the `renamed.brd` board file and the schematic.

15. Click *File – Exit* to exit from Design Differences.

Summary

You learned to synchronize the differences between a board file and the schematic. Design synchronization ensures that changes made by (Engineering Change Orders) ECOs late in the design cycle are reflected back in the schematic.

What's Next

Go to Lesson 5-1 to learn how to use design reuse properties for implementing design reuse.

5

Module 5 - Using Design Reuse Properties

This module is divided into the following lessons:

Completion Time	1 hour for the written lessons

Overview

The following two properties control the behavior of reused modules:

- `REUSE_INSTANCE`
- `REUSE_MODULE`

Understanding the function of the above properties will help you gain better control over design reuse. In this module, you would learn the function of these properties and learn to apply them in different designs.

Lesson 5-1 Understanding Design Reuse Properties

Objective

In this lesson, you will learn the function of different design reuse properties.

REUSE_ INSTANCE

The `REUSE_INSTANCE` property is assigned while using modules. If you do not assign the `REUSE_INSTANCE` property on the reuse block when instantiating it in Design Entry HDL, Packager-XL uses <*reuse_block_name*>_<*subdesign_suffix*> to generate a unique value for `REUSE_INSTANCE`. PCB Editor uses the `REUSE_INSTANCE` property to differentiate between multiple instances of a reuse module.

You can assign the `SUBDESIGN_SUFFIX` property on a reused block in Design Entry HDL to specify the suffix to be added to all reference designators in a subdesign module. If you have not specified the `SUBDESIGN_SUFFIX` property on the reused block in Design Entry HDL, Packager-XL will generate a unique number for the subdesign and use it as the `SUBDESIGN_SUFFIX` property. This default number starts from 1 and increments by 1 for subsequent blocks.

Note: For more information about controlling reference designators, see the answer to the FAQ How do I control reference designators in subdesigns? on page 84.

Unlike the other schematic properties, the `REUSE_INSTANCE` property defined on the uppermost block wins in the case of nested blocks.

REUSE_MODULE

By default, Packager-XL uses the REUSE_NAME property to name modules. You can assign the REUSE_MODULE property to assign a custom name to a module. You need the REUSE_MODULE property when you are creating multiple modules for the same design with each module having a different layout. In such cases, you can use the REUSE_MODULE property to assign different names to different instances of modules. When you are placing modules on a board, you can use the REUSE_MODULE property to choose the appropriate module in the board.

The order of precedence for determining the .mdd filename that PCB Editor looks for is:

- REUSE_MODULE
- REUSE_NAME (default)

Note: The REUSE_NAME property is always assigned by Packager-XL and is always equal to the logical design name in Design Entry HDL. You cannot change this property. If you want to assign a different module name for the .mdd file than the logical design name in Design Entry HDL, use the REUSE_MODULE property.

Note: For more information about assigning the REUSE_MODULE property, see Lesson 5-2:Using an Alternate Physical Module on page 74.

Summary

You learned the functions of three design reuse properties. You learned:

- If you have multiple modules with the same definition, you can use REUSE_INSTANCE to distinguish individual modules.
- You can use the REUSE_MODULE property to store a user-defined name for a module. When PCB Editor finds this property on the components in a reuse module, it loads the .mdd file corresponding to REUSE_MODULE.

What's Next

Go to Lesson 5-2 to learn how to create and specify an alternate physical module.

Lesson 5-2:Using an Alternate Physical Module

Objective

In this lesson, you will learn to use an alternate physical module in a board.

Overview

In Module 4 - Reusing the Design on page 51, you learned to reuse a logical symbol and a physical module in another design. You did not change any design reuse properties.

However, there might be cases where you require to change these properties. For example, you can have different versions of a physical module, and you require to use one or more instances of these different versions in another board.

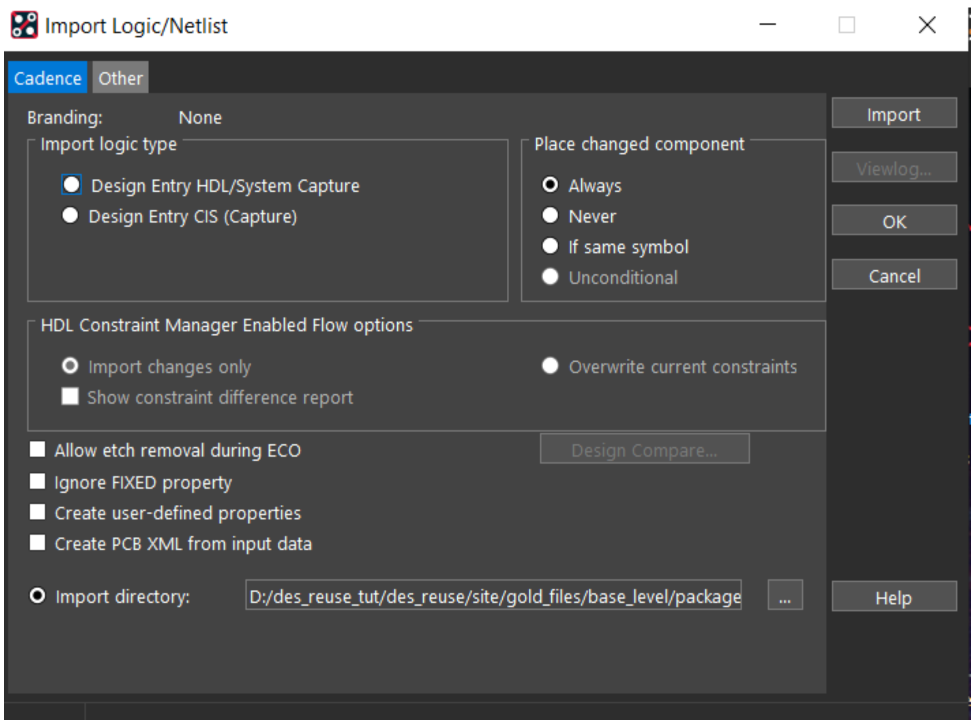
Task Overview

Create a new physical module named `base_level2.mdd` in PCB Editor using the `base_level` design as the top-level design and `start.brd` as the initial board file. Next, launch Design Entry HDL and assign the `REUSE_MODULE = base_level2` property to one of the instances. Package the design and verify that you are using both the `base_level` and the `base_level2` modules.

Procedure

1. Exit from Design Entry HDL and PCB Editor by choosing *File – Exit* in the respective tools.
2. Click *Setup* in Project Manager.
3. Enter `base_level` in the *Design Name* field.
4. Click *OK* to close the Project Setup dialog box.
5. Click *`Layout`* in Project Manager to launch PCB Editor.
6. Choose *File – Open* and open the `start.brd` board file.
7. Choose *File – Import – Logic* to import the `base_level` schematic.

   The Import Logic dialog box appears.


Import Logic/Netlist

Cadence | Other

Branding: None

Import logic type
- [✓ Design Entry HDL/System Capture]
- Design Entry CIS (Capture)

Place changed component
- [✓ Always]
- Never
- If same symbol
- Unconditional

HDL Constraint Manager Enabled Flow options
- [✓ Import changes only]
- Show constraint difference report
- Overwrite current constraints

- Allow etch removal during ECO
- Ignore FIXED property
- Create user-defined properties
- Create PCB XML from input data

[✓ Import directory:] D:/des_reuse_tut/des_reuse/site/gold_files/base_level/package

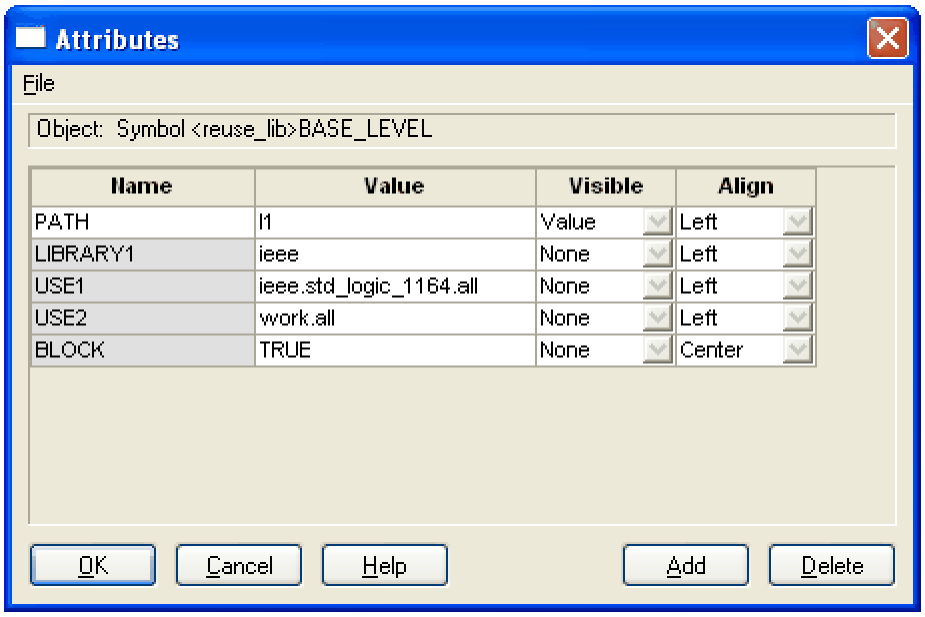
8. Ensure that *Design entry HDL* is selected as the import logic type.
9. In the *Import directory* field, specify the .../`base_level/packaged` directory.
10. Click *Import Cadence* to import the logic.
11. Using the steps mentioned in the Lesson 3-3: Designing the Physical Layout task, create a board as displayed in the Alternate BASE_LEVEL Physical Design figure on page 110.
12. Using the steps mentioned in the Lesson 3-4: Creating a Module from the Layout task, create a module for the board file you created in the last step.
13. Save the module as `base_level2.mdd` in the `modules` subdirectory in the `reuse` directory.
14. Save the board as `base_level2.brd`.
15. Click *File – Exit* to exit from PCB Editor.
16. Click *Setup* in Project Manager.
17. Enter `top_level` in the *Design Name* field, and click *OK* to close the Packager Setup dialog box.

    You have changed the design name to `top_level`. You can now open Design Entry HDL and add design reuse-specific properties to one instance of the `BASE_LEVEL` block.
18. Click *Design Entry* in Project Manager.

    You will notice two instances of the `BASE_LEVEL` block. Now, add the `REUSE_MODULE = base_level2` property to the first block from the top.

    Note: The name `base_level2` must map to the module you have created above in step 13.
19. Click the *Text Attributes* tool button to display text attributes.
20. Click the top `BASE_LEVEL` block.

The Attributes dialog box appears.



Object: Symbol <reuse_lib>BASE_LEVEL

Name	Value	Visible	Align
PATH	I1	Value	Left
LIBRARY1	ieee	None	Left
USE1	ieee.std_logic_1164.all	None	Left
USE2	work.all	None	Left
BLOCK	TRUE	None	Center

21. Click *Add*, and add a new property by typing `REUSE_INSTANCE` in the *Name* field and `HIGH` in the *Value* field.

You have added the `REUSE_MODULE = BASE_LEVEL2` and `REUSE_INSTANCE = HIGH` property to the first instance of the `base_level` block in the `top_level` design. These properties will help you select an appropriate module and instance for layout in the board.

22. Click *OK* to accept the new property and close the Attributes dialog box.

23. Choose *File – Save* to save the schematic.

24. Choose *File – Exit* to exit from Design Entry HDL.

25. Package the design in the `preserve` mode. Update the PCB Editor board using `start.brd` as the input board file, and `top_level_new.brd` as the output board file.

  a. Choose *Design Sync – Export Physical* in Project Manager to launch Export Physical.

  b. Select the *Package Design* check box to enable packaging.

  c. Select the *Preserve* option button to specify packaging in the `preserve` mode.

  d. Select the *Update PCB Editor Board (Netrev)* check box to make Netrev update the PCB Editor board.

  e. Click the *Browse* adjacent to the *Input Board File* field, and choose the `start.brd` file.

  f. Type `top_level_new.brd` in the *Output Board File* field.

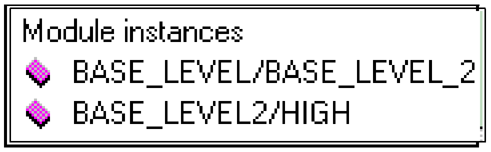
  g. Click *OK* to start packaging.

  h. Click *No* to close the confirmation box.

26. Open PCB Editor.

27. Using the steps mentioned in Lesson 3-3: Designing the Physical Layout, place the two modules and the JT1 and JT2 components.

You will notice that the Placement dialog box displays the new instance name (`HIGH`) and a new module name (`base_level2`) for one of the modules.


Module instances
BASE_LEVEL/BASE_LEVEL_2
BASE_LEVEL2/HIGH

When you place both the displayed modules on the board, both instances of modules (`base_level.mdd` and `base_level2.mdd`) will be placed in the layout.

28. Choose *File – Save As* and save the board as `mul_inst.brd`.

Now try this interactive exercise: Adding Reuse Properties.

Summary

You learned to place two different modules corresponding to the same logical design in a board. These modules have differences in layout and routing. The modules are differentiated by reuse properties.

What's Next

Congratulations! You have completed all exercises in the Design Reuse Tutorial. You can now create designs and reuse them in other designs.

Appendix A, "Frequently Asked Questions," lists answers to different questions asked by frequent design reuse practitioners. You can refer to these FAQs to better understand the different aspects of implementing design reuse for the Cadence PCB products.

A

Frequently Asked Questions

This FAQ has answers to the following questions:

- Is it possible to import swapped pin information or sizeable components that have the “HAS_FIXED_SIZE” property back to design entry HDL? I have swapped pins in Allegro, but on importing them back to DEHDL they are not visible.
- What is the difference between creating a hierarchical design and implementing design reuse?
- Can I have a subdesign as a standalone design?
- How do I control reference designators in subdesigns?
- In which situations should I use the GEN_SUBDESIGN, FORCE_SUBDESIGN, and USE_SUBDESIGN directives?
- What happens if I do not specify the subdesign name in the FORCE_SUBDESIGN or USE_SUBDESIGN directive while packaging the top-level design?
- Can I reuse a design that includes instances of other designs?
- How do I add properties in the top-level design and force them to appear in one or multiple sub-level design blocks?
- Can I edit properties in a reusable block while using it in my design?
- Can I have nested reuse modules? What precautions must I follow to manage nested reuse modules?
- How can I use modules in PCB Editor that are not associated with a schematic?
- How can I import or export subdrawings from within PCB Editor?
- Can I leverage unused gates in a module?
- If I make changes in PCB Editor to a part or parts from a reused module and I feedback the changes to the Design Entry HDL schematic, do I need to add any reuse-specific directives while packaging?

- If I use a block in a design then do I have to exercise any precaution while naming logical libraries?
- How do I specify a temp location for the library containing a read-only block?

Is it possible to import swapped pin information or sizeable components that have the “HAS_FIXED_SIZE” property back to design entry HDL? I have swapped pins in Allegro, but on importing them back to DEHDL they are not visible.

Yes, it is possible to obtain missing Section information in opf. You can use following directive to get obtain the section information associated with swapped pins while running B2F.

```
START_PKGRXL

…...

IMPORT_HFS_HARDSEC_ON_SWAP_PINS  'ON'

END_PKGRXL
```

What is the difference between creating a hierarchical design and implementing design reuse?

Overview of Hierarchical Design

When you create a software application, theoretically, it is possible that you can use a single file to create the application. However, maintenance will turn costly and painful. You use subroutines and functions to break the complexity of the software application, and save the code in different logically connected files to allow for easier maintenance and troubleshooting. Similar to creating a software application in one large file is the process of creating a flat schematic. You can have your complete design in a single flat schematic. However, you can use a hierarchical design to divide the complexity of a design into logical subsets or blocks. This will allow for easier maintenance and distribution of work within the design team.

In a hierarchical design, you create blocks to embed a logical design within another design. Each block is promoted once in the design but is used multiple times in the schematic.

Advantages of Hierarchical Design

The use of hierarchical design makes the top-level data flow easy to read. Because you do not need to manually add components each time you use a block, the design remains compact. You can replicate these blocks, and debug and simulate them. If you need to make changes to a component in a block, you can update the block and re-initialize each instance of that block.

Limitations of Hierarchical Design and the Need for Design Reuse

You might need to reuse the blocks created for one design in another design. In such cases, if you use hierarchical blocks, you will be able to reuse them only in the schematic. However if you do design reuse, you would be able to use the blocks both in the schematic and on the board, that is, across the PCB front-to-back flow. Further, the design reuse block serves as a ready-made PCB board that does not require any placement and routing.

Unlike blocks in a hierarchical design, the modules in reused designs contain the schematic design, the physical netlist (packaging information), and the physical board placement and routing information. This adds the edge to reusability. After you have saved modules in a library, you can view and reuse them in any future design.

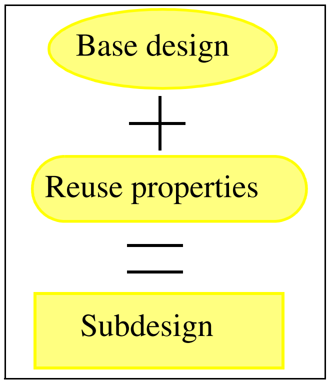
Modules can be updated if the master module is changed (similar to the way in which library components are refreshed). Components within modules can also be updated if those components have changed in the library.

In summary, creating a hierarchical design is a best practice for design creation, which increases productivity and reuses effort for a particular design. Design reuse is, however, an organization-level activity where you reuse your existing intellectual property (IP).

Note: This tutorial covers the procedure for creating a reusable logical and physical module. To reuse these modules effectively, you need an IP management system that manages the metatags required to access the individual modules in different designs. Using such an IP management system is outside the scope of the tutorial.

Can I have a subdesign as a standalone design?

Yes, you can have a subdesign as a standalone design. A subdesign is a complete design in itself. It has the same property and connectivity information as a standalone design and some additional reuse properties that provide the necessary information whenever the subdesign is reused in a larger design.



Therefore, it is possible to have a subdesign as a standalone design.

Note: To create a subdesign as a standalone design, make the subdesign as the top-level design.

How do I control reference designators in subdesigns?

Packager-XL assigns reference designators to components in a subdesign based on information contained in the `PHYS_DES_PREFIX` directive and the `SUBDESIGN_SUFFIX` property. The default reference designator assigned to subdesign instances has the following syntax:

<PHYS_DES_PREFIX><REFDES_INCREMENT><SD_SUFFIX_SEPARATOR><SUBDESIGN_SUFFIX>

where, the value of the `PHYS_DES_PREFIX` property is obtained from the schematic, ptf files, the `chips.prt` file, or the project file. The value of `REFDES_INCREMENT` is automatically assigned by Packager-XL. The default value of the `SD_SUFFIX_SEPARATOR` property is an underscore (_). The default value of the `SUBDESIGN_SUFFIX` property is `1` and for each subsequent component, the value increments by `1`. To change this value, assign a new property to a subdesign instance by using the Text Attributes dialog box in Design Entry HDL. This makes the packaging assignments predictable as you can trace parts on the board to a specific subdesign instance.

The following table explains each individual section of the reference designator and how you can control it.

Section	File where the value is stored	Who controls the assignment	How can you change the value
`PHYS_DES_PREFIX`	schematic, file`chips.prt` file, `ptf` files, or the project file.	Default is U. You can change it. **Note:** The order of precedence is schematic, ptf file, `chips.prt` file, and the project file.	■ Enter a new value in the *Default Ref Des Pattern* field in the Packager Setup - Layout tab. ■ Attach the `PHYS_DES_PREFIX` property to a component. **Note:** If there are multiple instances of a component and each instance is assigned a `PHYS_DES_PREFIX` property, then Packager-XL will use only one value. It will flag an error stating that multiple `PHYS_DES_PREFIX` properties are found on the schematic for the same physical part and it will exit with error status 1. ■ Change the value in the `chips.prt` or ptf file for a part.
`REFDES_INCREMENT`	Automatically assigned by Packager-XL and is stored in the `pxl.state` file.	Packager-XL controls this value.	You cannot change this value.

Section	File where the value is stored	Who controls the assignment	How can you change the value
`SD_SUFFIX_SEPARATOR`	Project file	Default is an underscore (_). You can change it.	■ Enter a new value in the *Subdesign Suffix Separator* field in the Packager Setup - Subdesign tab. ■ Set the `SD_SUFFIX_SEPARATOR` directive using the following syntax: `SD_SUFFIX_SEPARATOR` *character_name* where, *character_*`name` represents the new suffix for renaming reference designators.
`SUBDESIGN_SUFFIX`	Automatically assigned by Packager-XL and is stored in the `pxl.state` file.	Default is 1. You can change it.	Attach the `SUBDESIGN_SUFFIX` property to a component in Design Entry HDL.

Using the SUBDESIGN_SUFFIX property

To assign a suffix of 5 to an instance of the `base_level` subdesign, add the following property to the schematic block:

```
        SUBDESIGN_SUFFIX 5
```

This appends a suffix of `_5` to all reference designators in that schematic block. For example, you may have the following values of reference designators in the subdesign:

`U1_5, U2_5, and U3_5`

In the above values, the characters represent the following:

Character	Represents

U	`PHYS_DES_PREFIX`
_	`SD_SUFFIX_SEPARATOR`
5 (after the underscore)	`SUBDESIGN_SUFFIX`

Caution

`SUBDESIGN_SUFFIX` values must be unique across all subdesign instances in the design, that is, you cannot have `SUBDESIGN_SUFFIX = 1` on more than one subdesign instance even when the subdesigns themselves are different modules. This ensures that all reference designators are unique.

If you add the same `SUBDESIGN_SUFFIX` value on more than one instance, Packager-XL uses the first `SUBDESIGN_SUFFIX` property value it finds and creates a new value for the other instances while ignoring the value that you have assigned. Packager-XL also generates an error (269) stating that the `SUBDESIGN_SUFFIX` value already exists, and therefore it ignores the `SUBDESIGN_SUFFIX` value.

Note: If you do not use the `SUBDESIGN_SUFFIX` property but use the `FORCE_SUBDESIGN` directive, the suffixes assigned by Packager-XL might get reassigned.

To prevent suffixes from being reassigned,

➤ Specify the `USE_SUBDESIGN` directive when packaging new subdesign instances.

Note: If you delete a subdesign instance, the suffix for the deleted instance is not reused.

In which situations should I use the GEN_SUBDESIGN, FORCE_SUBDESIGN, and USE_SUBDESIGN directives?

The `GEN_SUBDESIGN` is used to create a subdesign, while the `FORCE_SUBDESIGN` and `USE_SUBDESIGN` directives specify how to load the subdesign state file.

You use the `GEN_SUBDESIGN` directive to specify the module (hierarchical block) for which you want to generate the subdesign state file. When this subdesign is used in other designs, Packager-XL reads the subdesign state file to obtain information about the design.

Note: You can use the `GEN_SUBDESIGN` directive only when the design name matches the root-level design. Packager-XL generates a subdesign state file only for the root-level design.

If you have generated a subdesign using the `GEN_SUBDESIGN` directive and want to use it in the top-level design, you need to use either the `FORCE_SUBDESIGN` or `USE_SUBDESIGN` directive. The primary difference between the two directives is in how often the data in the subdesign is read at the time of packaging the top-level design that contains the subdesign. If you use the `FORCE_SUBDESIGN` directive, Packager-XL reads the packaging information from the subdesign state file and applies it to every instance of the subdesign in the top-level (root) design. If you specify the `USE_SUBDESIGN` directive, Packager-XL applies the packaging in the subdesign state file to only those instances of the subdesign that have not been previously packaged in the top-level (root) design.

If you are generating the top-level design and including instances of the low-level design (which has its subdesign state file) in it for the first time, `FORCE_SUBDESIGN` and `USE_SUBDESIGN` behave similarly.

Note: If you make any changes in the subdesign, use the `FORCE_SUBDESIGN` directive while packaging. If you have not made any changes to the subdesign, use the `USE_SUBDESIGN` directive while packaging. The `USE_SUBDESIGN` directive allows you to start with the packaging in the subdesign state file and later customize it for new subdesign instances.

Note: If you do not use the `SUBDESIGN_SUFFIX` property but use the `FORCE_SUBDESIGN` directive, the suffixes assigned by Packager-XL might get reassigned. To prevent suffixes from being reassigned, specify the `USE_SUBDESIGN` directive when packaging new subdesign instances. For more information about how `SUBDESIGN_SUFFIX` impacts the naming of reference designators, see the answer to FAQ How do I control reference designators in subdesigns? on page 84.

What happens if I do not specify the subdesign name in the FORCE_SUBDESIGN or USE_SUBDESIGN directive while packaging the top-level design?

All subdesigns that are included in the top-level design must have the `FORCE_SUBDESIGN` or `USE_SUBDESIGN` directive set for them. If you do not add the subdesign that you want to be included in the top-level design in the `FORCE_SUBDESIGN` or `USE_SUBDESIGN` directive list, you will not be able to place the physical module in PCB Editor.

You can specify a subdesign, for example `base_level`, in the `FORCE_SUBDESIGN` list by adding the name of the subdesign (`base_level` in the *Force Subdesign* box in the Packager Setup - Subdesign tab.

Note: The procedure covered in the Lesson 4-3: Packaging the Design on page 55 explains how to use the `FORCE_SUBDESIGN` directive.

Important

You might come across the above problem in situations where you have changed the logical representation of the subdesign without updating the physical representation of the subdesign or vice versa after the subdesign has been instantiated in the top-level design.

Can I reuse a design that includes instances of other designs?

You can reuse a design that includes instances of other designs. In such cases, you will end up having a minimum of three levels of hierarchy.

For example, assume you have a design named `MID` that includes two instances of a subdesign named `LOW`. You can now use an instance of the `MID` design in another design named `TOP`. For both `MID` and `TOP` designs, you can have other components besides the `LOW` subdesign instances.

To create the `TOP` design, complete the following steps:

1. Create the `LOW` subdesign.
2. Create a module and a logical reuse symbol for the `LOW` design.
3. Create the `MID` design. Include instances of the `LOW` design in the `MID` design by using the `FORCE_SUBDESIGN` directive.

   Note: For more information about using the `FORCE_SUBDESIGN` and `USE_SUBDESIGN` directives, see the answer to FAQ In which situations should I use the GEN_SUBDESIGN, FORCE_SUBDESIGN, and USE_SUBDESIGN directives? on page 87 .
4. Create a module and a logical reuse symbol for the `MID` design.
5. Create the `TOP` design. Include instances of the `MID` design in the `TOP` design by using the `FORCE_SUBDESIGN` directive.

How do I add properties in the top-level design and force them to appear in one or multiple sub-level design blocks?

By default, if you add properties in the top-level design, those properties might not be in all the sub-level designs. For example, if you have a schematic property `Prop1` in the top-level design and a sub-level design, then the property in the sub-level design will win. If you want to add properties in the top-level design and want to force them in one or more sub-level design blocks, add those properties in the Occurrence Edit mode. Any property added in the Occurrence Edit mode automatically wins over the sub-level designs.

➤ To enter the Occurrence Edit mode, choose *Tools – Occurrence Edit* in Design Entry HDL.

Important

Design reuse methodologies typically call for promoting the reusable block to a read-only reference library. As a result, if you use any of these blocks, you cannot make connectivity changes to these blocks or write property changes to the schematic file. If you want to make property changes, you need to store these changes in the Occurrence Property File (OPF).

Note: For more information about the Occurrence Property file, which stores all property assignments made in the Occurrence Edit mode, see the Cadence document *Packager-XL Reference*.

Can I edit properties in a reusable block while using it in my design?

Yes, you can edit properties in a reusable block while you place them in your design. For example, assume you have two `base_level` blocks in a design named `top-level` and you add a property `Prop1` in the first `base_level` block. If you now package the design and run Netrev, and assuming that `Prop1` is defined as transferable in the Property Flow Setup and the same is defined as a user property in PCB Editor, then `Prop1` will appear in the module corresponding to the first `base_level` block.

Tip

If you include reusable blocks in your design as lower-level blocks, the probability is extremely high that they are read-only. You can edit properties in a lower-level (read-only) block only in the Occurrence Edit mode.

Can I have nested reuse modules? What precautions must I follow to manage nested reuse modules?

You can have nested reuse modules. For example, you can have a three-level nested module. At the highest-level, assume you have a module named `TOP_LEVEL` that uses another module named `MID_LEVEL`, which in turn uses a module named `LOW_LEVEL`. You can even have four-level or five-level nested modules.

In case of a nested module, if you make an ECO at any level, it is important to individually refresh each subsequent higher level module in the design. For example, in the above design, if you make an ECO in the `LOW_LEVEL` module, you must first update the `MID_LEVEL` module to include this change by performing a module refresh of the `LOW_LEVEL` module in the `MID_LEVEL` module. Next, you must update the `TOP_LEVEL` module by performing a module refresh of the `MID_LEVEL` module in it.

Caution

If you are using nested reuse modules, it is not possible to edit the top-level module and refresh a low-level module if the low-level module is included in any mid-level module.

How can I use modules in PCB Editor that are not associated with a schematic?

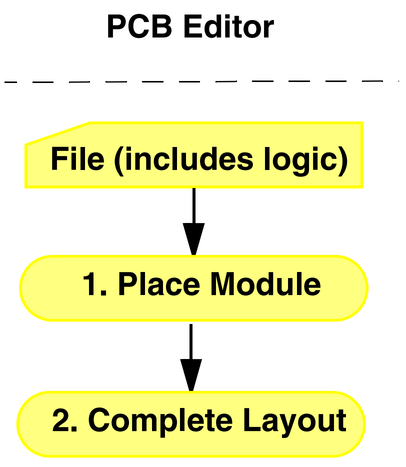
The design reuse flow covered in this tutorial is schematic-driven. You create a schematic and use it first to generate a board and then to create a module for the board. This way you get a reusable module that includes the synchronization of logical and physical design information.

Besides the schematic-driven process of creating modules, PCB Editor also provides the ability to place modules created from existing physical logic and complete the layout. These modules need not be associated with the schematic. You can decide whether you want to route these modules and have them fully routed, partially routed, or not routed at all.

The following diagram explains the module-driven process for using modules across designs.

1. **Place Module**—Choose *Tools – Create Module* (`place manual` command) to place a module definition on the board, thus creating a module instance.
2. **Complete Layout**—Layout the design using the appropriate PCB Editor features.

Note: If your goal is to reuse design only in the physical space then you can gain advantage of using the module-driven process for using modules across designs. However, if you want to create and use modules that offer synchronization of both logical and physical designs, then use the Lesson 1-3: Design Reuse Flow on page 12 for creating such modules.



How can I import or export subdrawings from within PCB Editor?

You can copy and paste elements between designs or symbol drawings by using the *File – Import – Subdrawing* (`clppaste`) and *File – Export – Subdrawing* (`clpcopy`) commands. PCB Editor stores the elements that you copy in a clipboard (.clp) file.

Important

Use of `clpcopy` and `clppaste` will help you quickly reuse physical design in your board. However, if you have Design Entry HDL and want to make reusable blocks that contain synchronization of logical and physical information, use the Lesson 1-1: Objective of the Tutorial on page 7 for creating such blocks.

You can copy and paste elements from

- Design to clipboard to design
- Design to clipboard to symbol drawing
- Symbol drawing to clipboard to design
- Symbol drawing to clipboard to symbol drawing

You can create a library of clipboard files, each containing frequently used or unique elements for future use. For example, if you are creating multilayer ceramic modules, an element library lets you organize your design into a hierarchy of elements. You could create the following hierarchy, storing each element in a separate clipboard file:

- First Level—Store a single rectangle as a pad
- Second Level—Group multiple pads together to form a chip site
- Third Level—Add pin escapes to the chip site to create a cell
- Fourth Level—Group several cells to form a larger cell

You can continue to build this element hierarchy until the entire module design is complete. At any level in the hierarchy, you can use the *Find Filter* (Find by Property) to identify all of the pieces that form that level.

Note: For more information about the restrictions, prerequisites, and recommendations for importing or exporting subdrawings in PCB Editor, see the Cadence document *Allegro PCB and Package User Guide: Preparing the Layout.*

Exporting Subdrawings in PCB Editor

You need to perform the following steps to export subdrawings in PCB Editor.

1. Choose *File – Export – Sub-Drawing* or enter the `clpcopy` command at the Console Window.

   PCB Editor displays the following message:

   ```
   Enter selection point
   ```

   Use the *Find Filter* to help select elements for the clipboard file.

2. In the *Options* tab, enter whether you want to preserve the reference designator information in the clipboard file. To preserve the reference designator information, select the *Preserve Refdes* check box.

   Note: This applies only in board drawings.

3. If you are copying a single element, position the cursor over the element you want to copy and click. If you want to copy a group of elements, perform one of the following two methods:

   ❑ Draw a box around all the elements you want to select.

   ❑ Right-click and select the *Temp Group* option. You can now individually select all elements, and then select *Complete* from the shortcut menu.

   PCB Editor displays the following message:

   ```
   Pick clipboard origin point
   ```

   The origin is a reference point to help you orient the element when you paste it into another design.

4. Move the cursor to the origin point and click.

   The Clipboard Filename dialog box is displayed.

5. Enter the name of the clipboard file and click *Save*.

   If you do not enter a file name, PCB Editor uses the default `standard.clp`. If you enter a filename without an extension, PCB Editor adds the default extension `.clp`.

Importing Subdrawings in PCB Editor

You need to perform the following steps to import subdrawings in PCB Editor.

1. Choose *File – Import – Sub-Drawing* or enter the `clppaste` command at the Console Window to display the Select Subdrawing to Import dialog box.

2. Enter the name of the clipboard file you want to paste and click *OK*.

   Note: You can paste a clipboard file to a specific location by providing a full path name in the File name field. By default, PCB Editor looks in the current working directory if you have not previously used *File – Import – Sub-Drawing*. If the command has been previously used, PCB Editor defaults to the last directory used by the command.

   When you select a clipboard file to import into your drawing:

   ❑ A Clipboard Parameters dialog box is displayed, which includes the *Rotate angle increment* and *Clip_drawing* fields.

   ❑ In the Design window, a rectangle appears, attached to the cross-hair cursor, indicating the boundary of the elements in the clipboard file you are about to paste.

   The cursor is at the origin point you specified when you created the clipboard file, and might be outside the rectangle. When you position the new elements, they must be within the extents of the active design.

3. Fill out the Clipboard Parameters dialog box.

   The fields on the Clipboard Parameters dialog box are as follows:

   Rotate Angle Increment identifies the angle of the elements pasted into the active design.

   Clip_drawing lets you identify a property value for the pasted elements that overrides the `CLIP_n` value. The property value can have as many as 32 characters. The exclamation point (!) and single quotation marks are not allowed.

4. If the clipboard file contains preserved reference designators, and the clipboard file is being loaded into an PCB Editor design, enter whether you want to assign the reference designator information from the clipboard file. In the Options tab, select the *Assign Refdes* check box if you want to assign it.

5. Position the cross-hair cursor where you want the origin point of the pasted elements to be located.

   You must position the new elements within the extents of the active design.

   As you move the cursor to position the elements, you can use the following options from the Design window pop-up menu:

   ❑ **Cancel**—Stops the paste process without pasting anything into the design.

   ❑ **Rotate**—Lets you rotate the paste rectangle by the increments you specified in the Rotate Angle Increment option of the Clipboard Parameters dialog box. During rotation, the Design window pop-up menu includes an *Oops* option that lets you exit rotation mode, but does not exit *File – Import – Sub-Drawing*.

6. When you have positioned the elements, click to paste.

PCB Editor performs a DRC check on the new elements and places them on the board.

Can I leverage unused gates in a module?

By default, Packager-XL makes strict packages for the `REUSE_INSTANCE` property. This allows Packager-XL to leverage unused gates in a module but it cannot leverage unused gates across two reused modules.

How does the Optimize option behave while packaging subdesigns?

Assume you have a design named `top_level`, which contains 2 instances of `base_level` subdesign and other components such as `LS00`. If you package the `top_level` design using the `Optimize` option, then Packager-XL will not optimize packaging between the components in the subdesign block (that is components in `base_level`) and other components (such as `LS00`) in the design. The reference designators in the subdesigns will be retained and any optimizing will only work for components that are not part of subdesigns or for the components that are within the subdesigns but not for components within the subdesign and outside the subdesign.

If I make changes in PCB Editor to a part or parts from a reused module and I feedback the changes to the Design Entry HDL schematic, do I need to add any reuse-specific directives while packaging?

When you make changes to any part in PCB Editor, such as changes in reference designators, and backannotate the changes to the Design Entry HDL schematic, you do not have to use any reuse-specific directive while packaging the design.

`GEN_SUBDESIGN`, `FORCE_SUBDESIGN`, and `USE_SUBDESIGN` all work when you are packaging the design. Their behavior is same in both the Forward mode (when you use Export Physical) and the Feedback mode (when you use Import Physical). Therefore if you have `FORCE_SUBDESIGN` and `USE_SUBDESIGN` set in a design, then Packager-XL will read the subdesign state file as per the directive set. If `GEN_SUBDESIGN` is set, then Packager-XL will generate a subdesign module of the current design. If you use `GEN_SUBDESIGN` in the Feedback mode, Packager-XL will include the changes made in PCB Editor in the generated subdesign. As a result, during the Feedback mode, you need not add any extra reuse directives.These directives would already be set during the Forward mode.

If I use a block in a design then do I have to exercise any precaution while naming logical libraries?

You can use different logical library names for the top-level and lower levels of your design. For example, if there are two users, `A` and `B`, working with reusable blocks, they can do the following:

1. `A` creates a block called `a_block` in a library named `a_lib`.
2. `B` creates another block called `b_block` in a library named `b_lib`.
3. Both `A` and `B` continue completing the design. After some time `A` declares the block is completed.
4. The `a_block` is promoted from library `a_lib` and is moved to a read only area named `common`.
5. B decides to use the block created by A. For this, `B` can now use any logical library name to access the blocks in `common`. For example, `B` can write the following statement in the `cds.lib` file:

   ```
   DEFINE reuse_lib ../reuse/common
   ```

   Instead of `reuse_lib`, you could use any other name including `a_lib` and `b_lib`.

Note:

a. If you are working in the Constraint Manager enabled flow, Constraint Manager will not ignore library names and cause errors if different logical library names are used for the same block. Therefore, you should not rename a lower level block or the root level library name.

b. To handle global modules in lower levels, use the `TMP` directive for handling read-only blocks.

How do I specify a temp location for the library containing a read-only block?

If you use a read-only block in your design, you must use the `TMP` attribute in the `cds.lib` file of your project to specify a temp location for the library containing the read-only block. To use the `TMP` attribute in the `cds.lib` file, do the following:

1. Add the following entry in the `cds.lib` file.

   ```
   ASSIGN <logical name of library containing read-only block> TMP <path to temp
   directory>
   ```

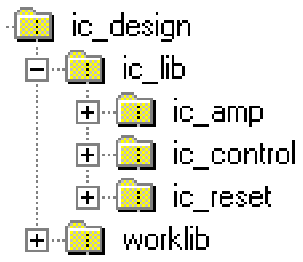
2. Create the `temp` directory.

If you do not do this, Design Entry HDL displays the following error message when you netlist the design for packaging or simulation:

```
NTL_ERROR : Need to analyze the design for cell : <cell name> in library :
<library name>", root->cellName, root->libName

Cell : <cell name> in library : <library name> is marked as read only", root-
>cellName, root->libName

Please provide write permissions in the cell for analysing the design.

You can use the TMP directive in cds.lib file.
```

This error occurs because Design Entry HDL is unable to generate files in the libraries that contain the read-only block when the design is netlisted.

The following figure explains the usage of the `TMP` attribute.



The library `ic_lib` is a read-only library. The cell `ic_reset` is a read-only block used in the design `ic_design`. The library `ic_lib` is defined in the `cds.lib` file as below:

```
DEFINE ic_lib ./ic_lib
```

To use the `TMP` attribute in the `cds.lib` file, do the following:

1. Add the following entry in the `cds.lib` file.

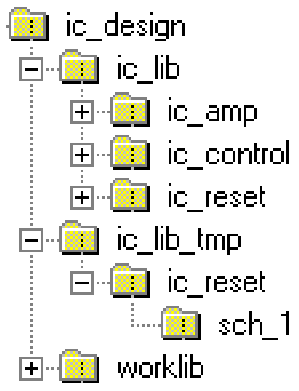
   ```
   ASSIGN ic_lib TMP ./ic_lib_tmp
   ```

   This means that the `temp` directory for the `ic_lib` library is `./ic_design/ic_lib_tmp/`

2. Create the `temp` directory.

   ```
   mkdir ic_design/ic_lib_tmp
   ```

The following figure displays the directory structure after the design is netlisted.



Note that Design Entry HDL has created the following:

- A directory `ic_reset` under `ic_lib_temp`

  This is the temp location for the cell `ic_reset` that contains the read-only block used in the design.

- A directory `sch_1` under `ic_reset`

  This is the `sch_1` view for the cell `ic_reset`. All the files generated by Design Entry HDL when the read-only block `ic_reset` is netlisted are written to the specific views.

What care I should take while implementing design reuse when using Allegro Constraint Manager from Design Entry HDL?

If you are working in a team design environment where one person creates blocks and other team members integrate them in their designs then while using Constraint Manager with Design Entry HDL do the following:

- If you assign constraints to a block and then reuse it in another block, the constraints would not be read by Design Entry HDL-Constraint Manager. It is recommended that while reusing blocks in another design, assign electrical constraints only at the top-level block and not in the blocks being reused. If you need to assign constraints in lower-level blocks assign them as electrical constraint properties in the schematic canvas.

- If you have assigned constraints to lower-level blocks and want to use them then backannotate the design. However, if you have used ECSets or differential pairs, then these will not get updated to Design Entry HDL on backannotation.

How do I make changes in a reused subdesign?

Even after you are convinced that the design is logically and physically complete, ECOs can crop up. The fact remains that designs continue to change. ECOs cropping up at the last minute can include the following:

1. Property changes on the schematic component.
2. Connectivity changes in the schematic design.
3. Layout changes including modification of component location and routing (etch) changes.

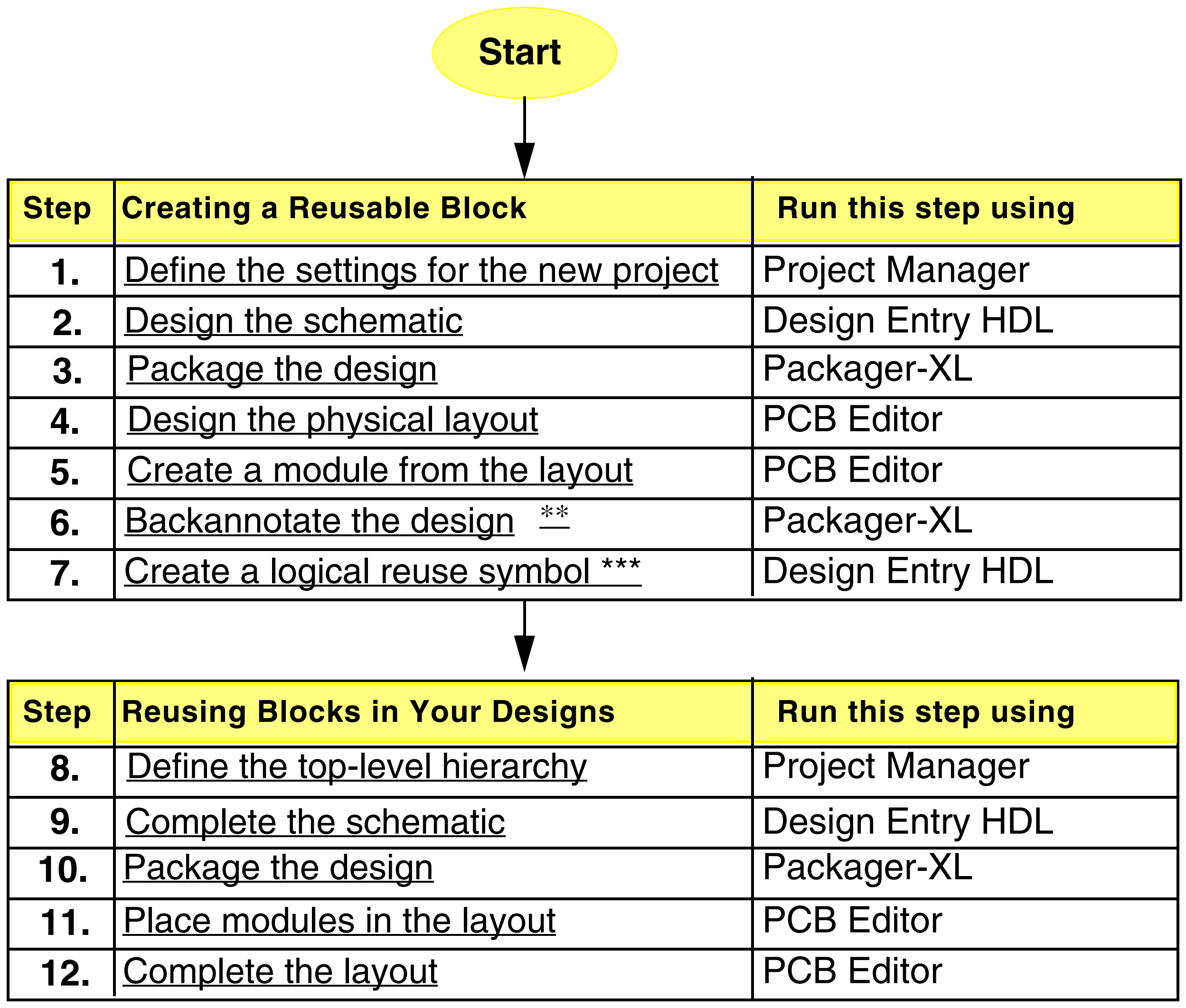
After making the ECOs mentioned above, you need to update all instances of the module where it is being reused. The Design Reuse Flow figure on page 100 depicts the standard design reuse flow containing 12 steps. Based on where you make any changes in the subdesign, you are required to repeat any or all of the subsequent steps.

For example, assume you need to make property and connectivity changes in a subdesign named `base_level`. Note that these changes are in step 2 of the design reuse flow. After making the changes, you must package the design (step 3). However after packaging the design, you are not required to design the layout because you do not have any changes in the layout. Next, you must re-create the module (step 5) to update it with the changes made in the schematic. You can now backannotate the design (step 6), and re-create the logical reuse symbol (step 7). After generating a new reuse symbol, change the design name to the design that contains reusable blocks, that is `top_level` (step 8). Next, reinstate all instances of the `base_level` design (step 9) and package it using the `FORCE_SUBDESIGN` directive (step 10). Finally, reinstate all modules for `base_level` in the `top_level` board (step 11).

If you make changes in the layout of a component, you must complete all steps beginning from step 4 till step 11 in the design reuse flow.

Note: For more information about how to handle ECOs in nested modules, see the answer to the FAQ Can I have nested reuse modules? What precautions must I follow to manage nested reuse modules? on page 90

Figure A-1 Design Reuse Flow



Step	Creating a Reusable Block	Run this step using
1.	Define the settings for the new project	Project Manager
2.	Design the schematic	Design Entry HDL
3.	Package the design	Packager-XL
4.	Design the physical layout	PCB Editor
5.	Create a module from the layout	PCB Editor
6.	Backannotate the design **	Packager-XL
7.	Create a logical reuse symbol ***	Design Entry HDL

Step	Reusing Blocks in Your Designs	Run this step using
8.	Define the top-level hierarchy	Project Manager
9.	Complete the schematic	Design Entry HDL
10.	Package the design	Packager-XL
11.	Place modules in the layout	PCB Editor
12.	Complete the layout	PCB Editor

** Backannotating the design is not a mandatory step in the design reuse flow. If you have not made any changes to the layout after the initial packaging of the design, you are not required to backannotate the changes to the schematic. However, if you make any changes to property or connectivity information in the layout after the initial packaging of the design, backannotate these changes to the schematic. The changes that require backannotation include connectivity modifications, reference designator changes, and pin/gate swaps.

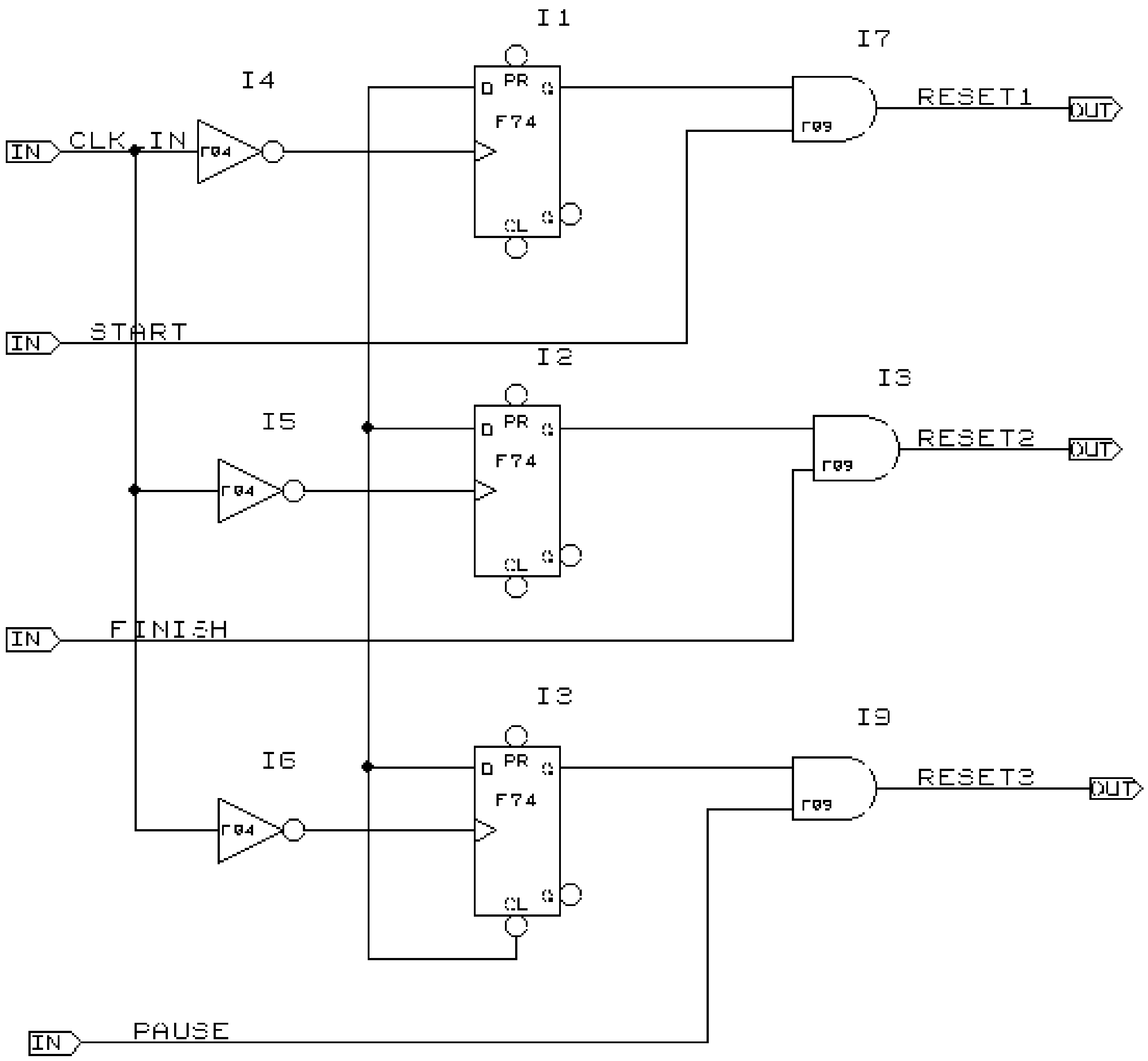
*** You can create a logical reuse symbol immediately after creating the schematic. However, ensure that the logical reuse symbol is created only when all logic is finalized. The interface nets should be assigned.

B

Sample Designs

This appendix includes snapshots of the following designs:

Figure B-1 BASE_LEVEL Logical Design



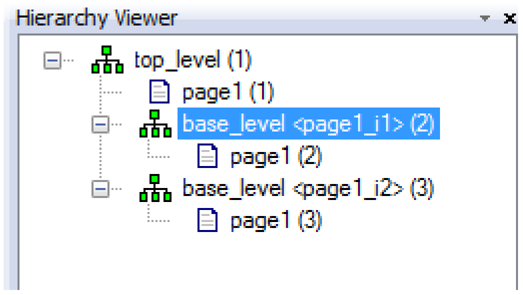
The components used in the `BASE_LEVEL` logical design are `F74`, `F04`, `F08`, `INPORT`, and `OUTPORT`.

Using Gold Files

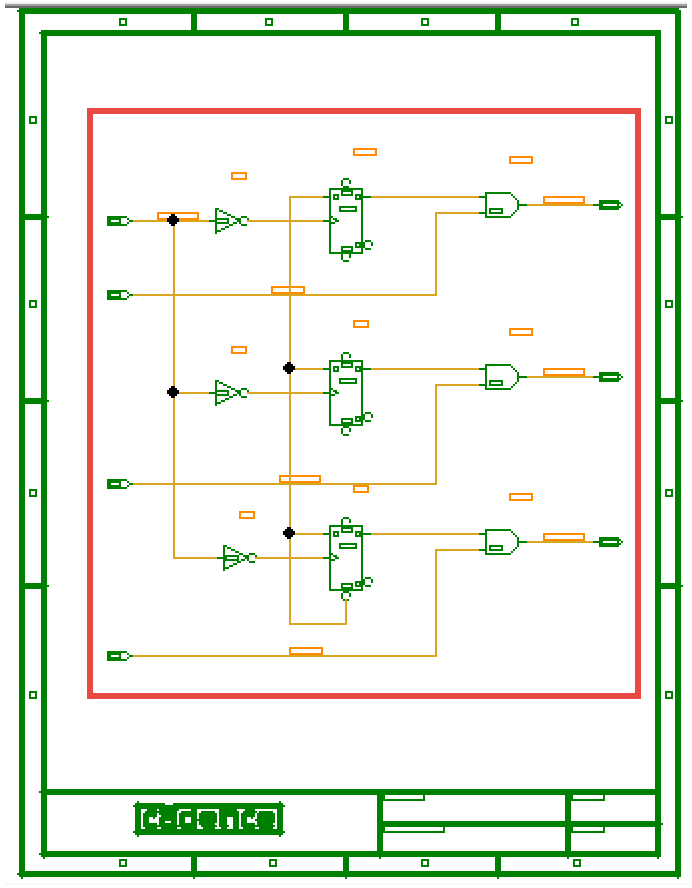
If you choose not to create the schematic, copy the gold schematic data to your work area.First, open another session of DE HDL.

1. Navigate to `.../des_reuse/reuse_archive` and open `reuse.cpm`.

2. Select `base_level <page1_i1> (2)` in Hierarchy Viewer as illustrated in the following image:



3. In the schematic page, select the circuity within the red box by creating a group:

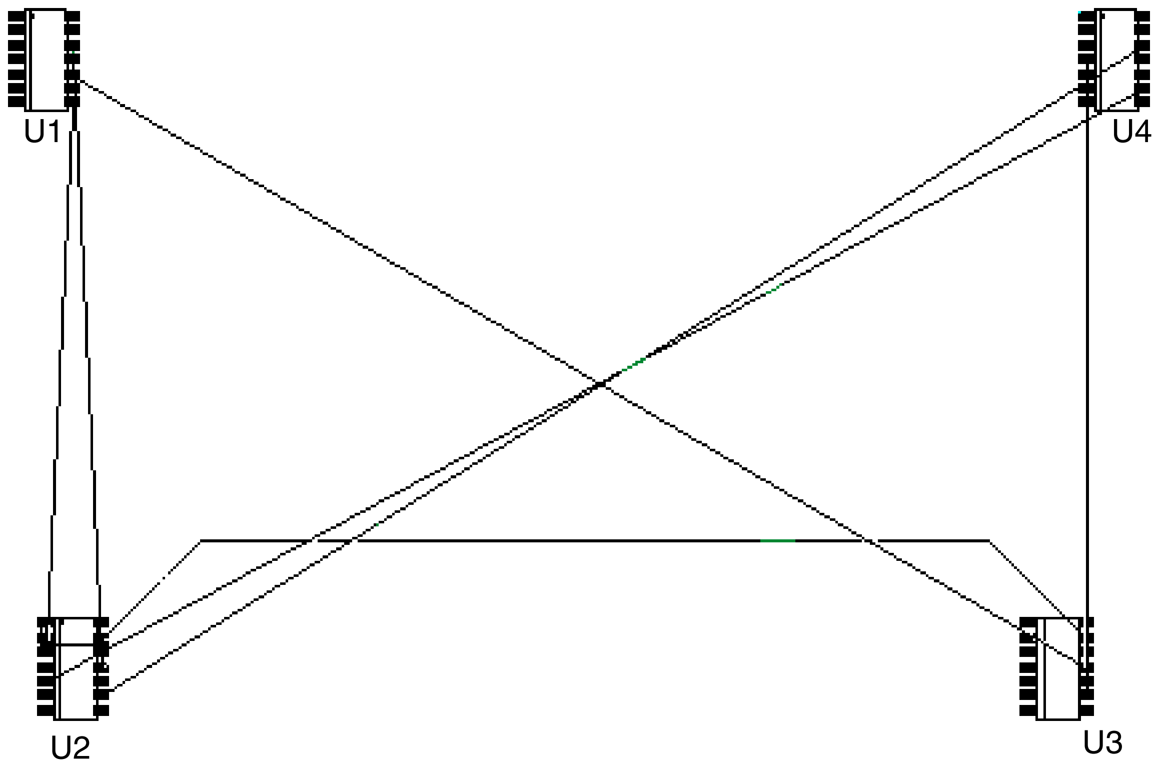


4. Choose *Group – Copy All[A]*.

5. In your work area, paste the circuity within the default page automatically created by Design Entry HDL.

6. Choose *File – Save* to save the design and then *File – Exit* to exit Design Entry HDL.

Figure B-2 BASE_LEVEL Physical Design

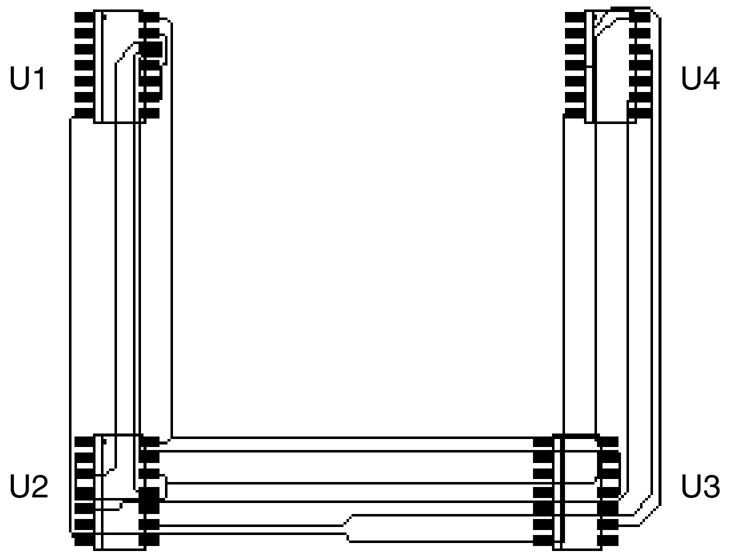


Note: Do not place the four components too far away. This might create placement problems when you generate the module from this design and place it in the `top_level` design.

Using Gold Files

If you choose not to create the board layout, you can copy the gold board data to your work area. For this, copy the `placed.brd` file from the `des_reuse/site/gold_files/base_level/physical` folder to the …`/des_reuse/reuse/worklib/base_level/physical` directory.

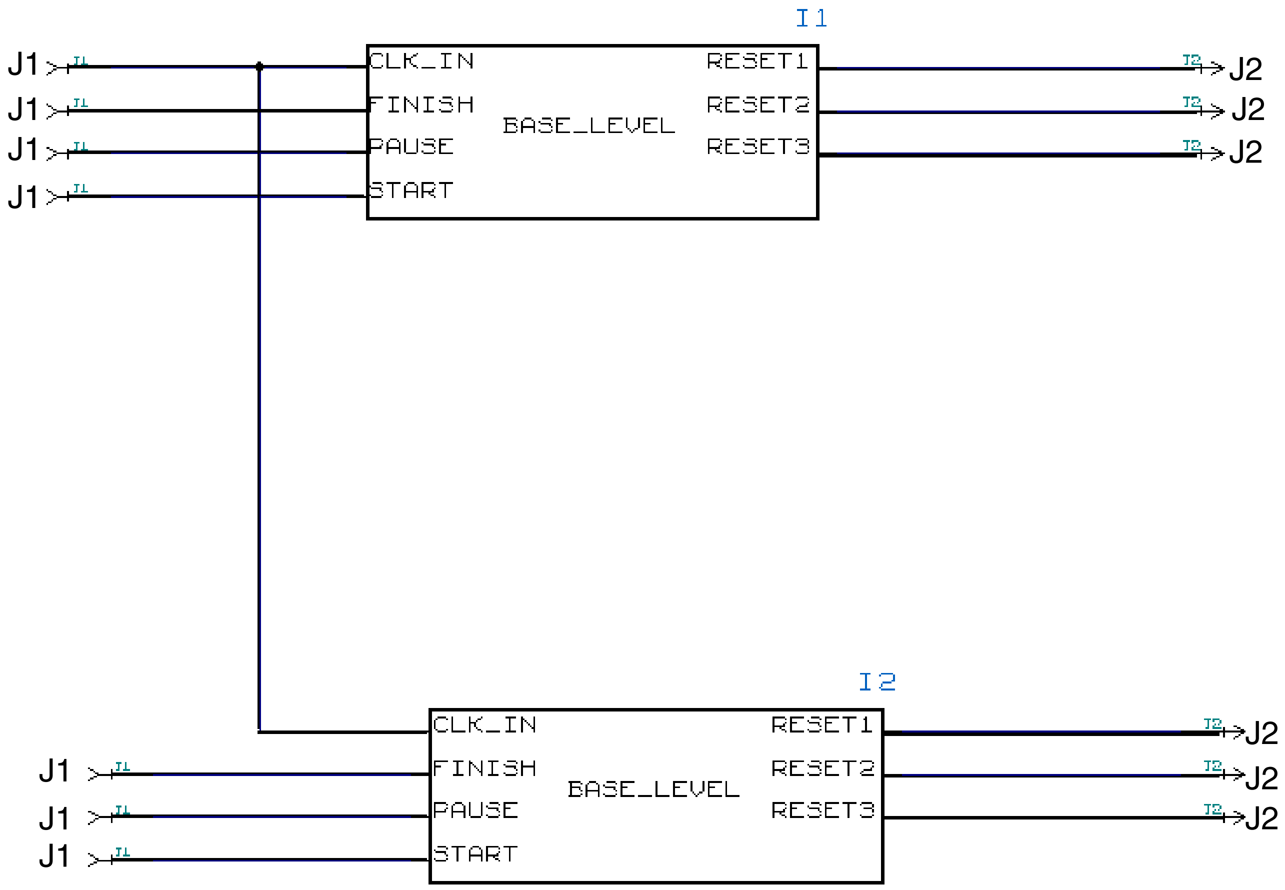
Figure B-3 BASE_LEVEL Routed Design



Using Gold Files

If you choose not to route the board, you can copy the golden board data to your work area. For this, copy the `routed.brd` file from the `des_reuse/site/gold_files/base_level/physical` directory to the ...`des_reuse/reuse/worklib/base_level/physical` directory, open it in PCB Editor, and save it again with the same name.

Figure B-4 TOP_LEVEL Logical Design



The following components are used in the `TOP_LEVEL` logical design:

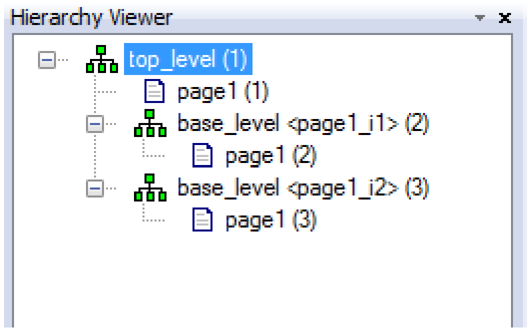
1. `BASE_LEVEL` component (`reuse_lib` library)
2. J1 is the `CONN20` component (`parts_lib` library)
3. J2 is the `CONN30` component (Version 2) (`parts_lib` library)

Using Gold Files

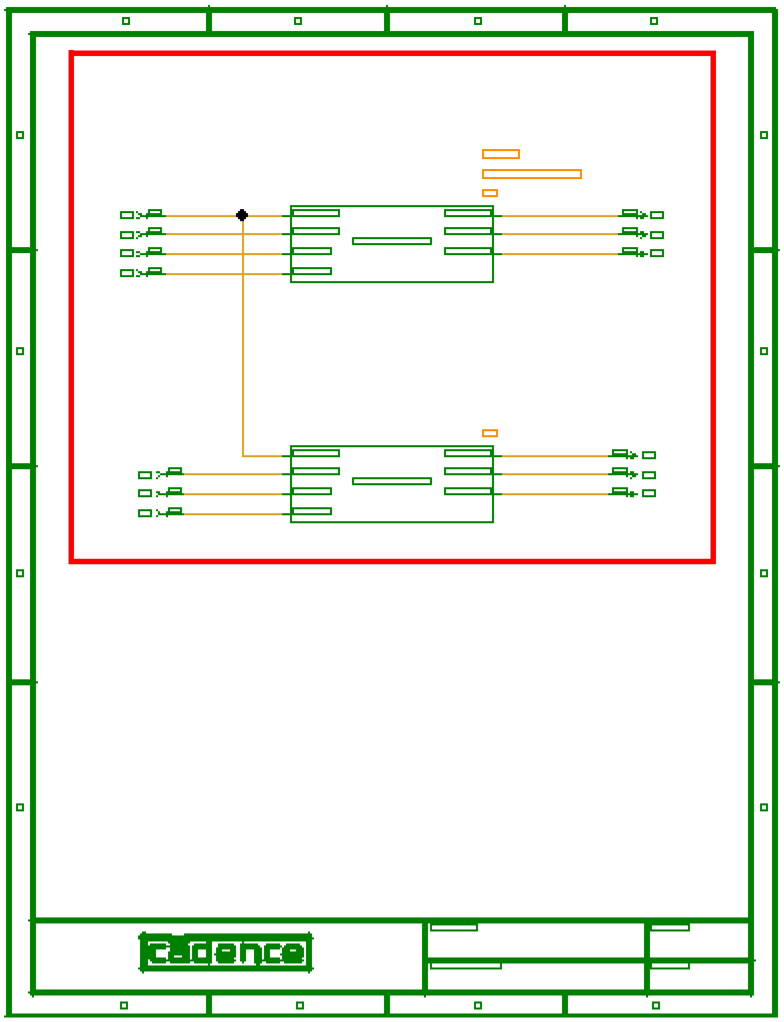
If you choose not to capture the schematic, you can copy the golden schematic data to your work area. For this, do the following:

1. Navigate to `.../des_reuse/reuse_archive` and open `reuse.cpm`.

2. Select `top_level (1)` in Hierarchy Viewer as illustrated in the following image:



3. In the schematic page, select the circuity within the red box by creating a group:

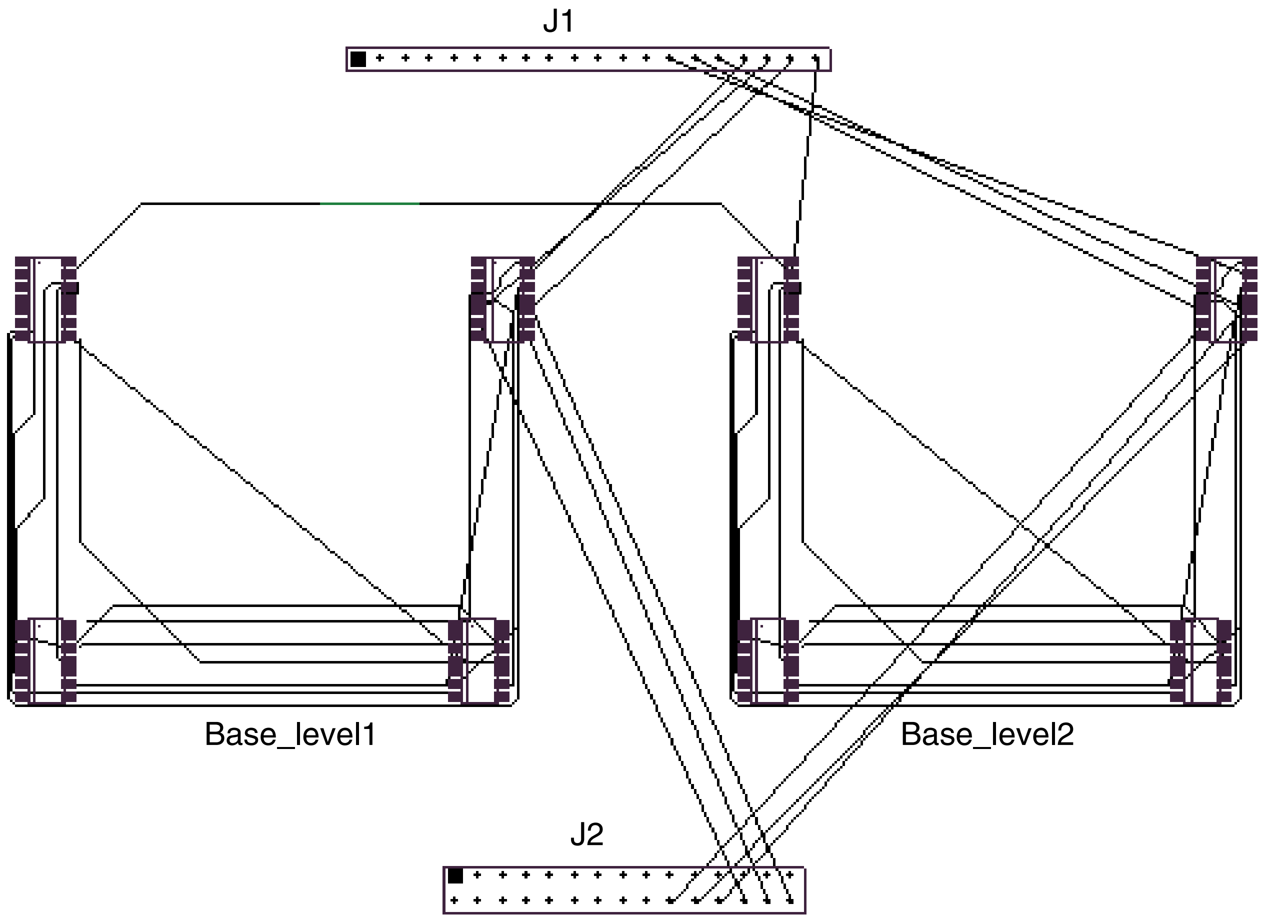


4. Choose *Group – Copy All[A]*.

5. In your work area, paste the circuity within the default page automatically created by Design Entry HDL.

6. Choose *File – Save* to save the design and then *File – Exit* to exit Design Entry HDL.

Figure B-5 TOP_LEVEL Physical Design

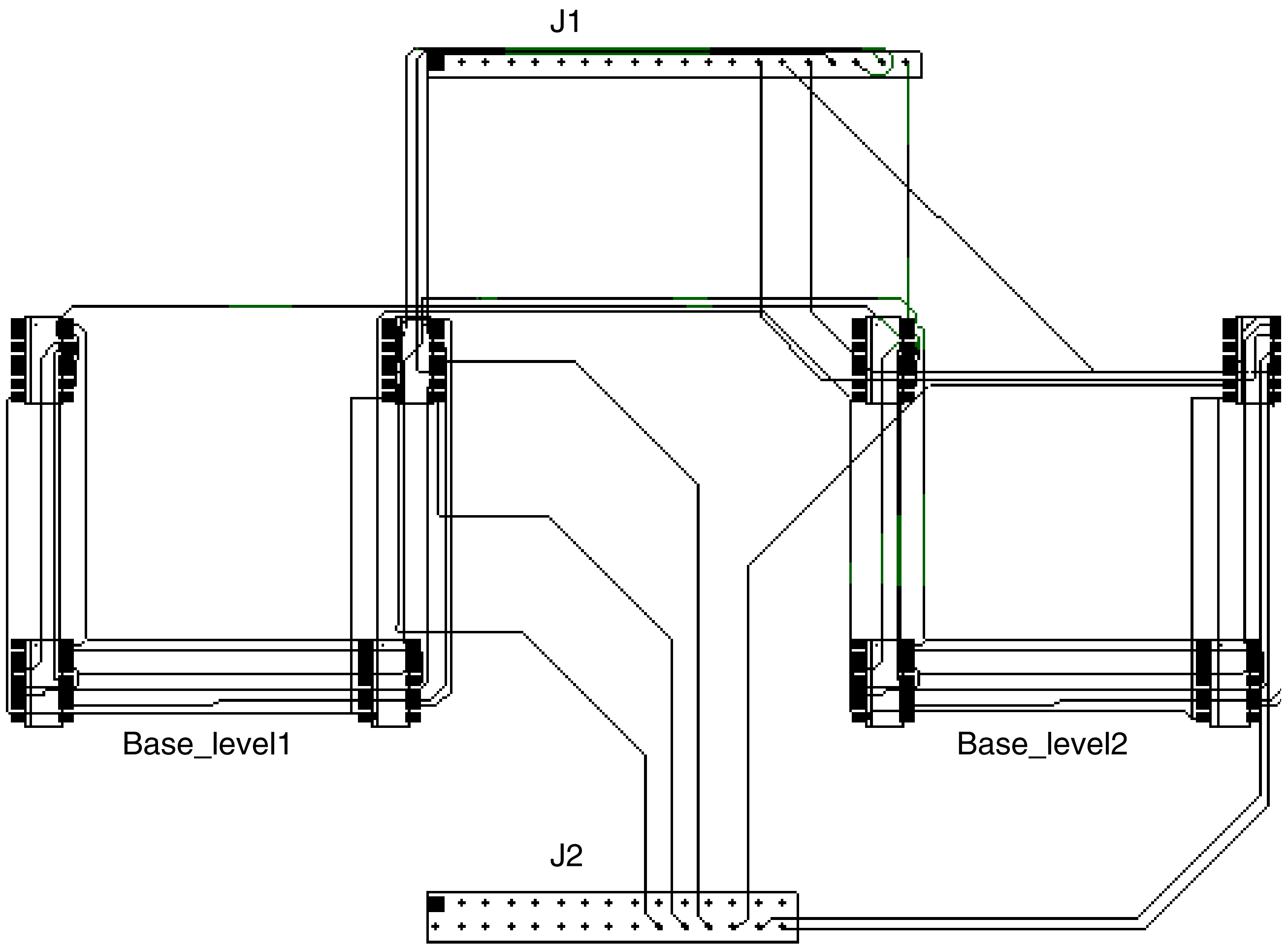


Using Gold Files

If you choose not to create the board layout, you can copy the golden board data to your work area.

For this, in PCB Editor, open the `placed.brd` file from `/reuse/reuse_archive/worklib/top_level/physical` and save it with the same name in `/reuse/worklib/top_level/physical`.

Figure B-6 TOP_LEVEL Routed Design

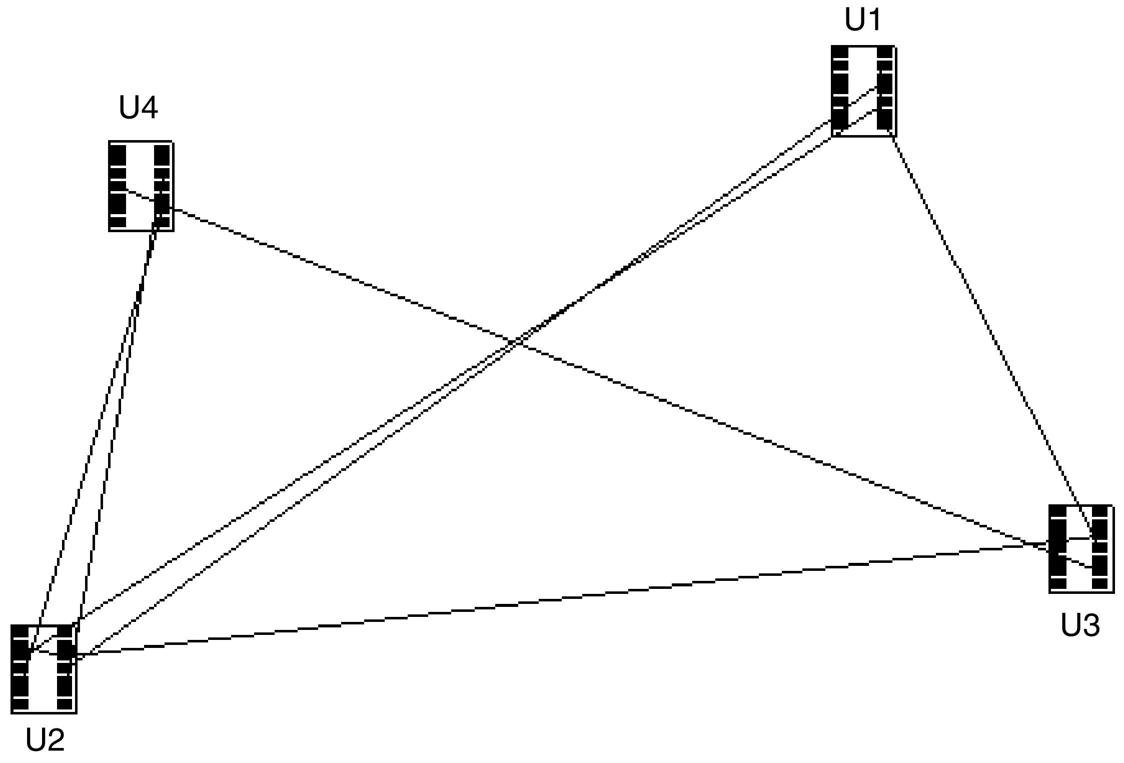


Using Gold Files

If you choose not to route the board, you can copy the golden board data to your work area.

For this, in PCB Editor, open the `routed.brd` file from `/reuse/reuse_archive/worklib/top_level/physical` and save it with the same name in `/reuse/worklib/top_level/physical`.

Figure B-7 Alternate BASE_LEVEL Physical Design



Using Gold Files

If you choose not to create the board layout, you can copy the golden board data to your work area. For this, copy the `base_level2.brd` file from the `/reuse/worklib/base_level/physical` directory, open it in PCB Editor, and save it again with the same name.

C

Frequently Used Terms in This Tutorial

You will find the following terms frequently used in this tutorial:

Design Reuse

At the micro level, design reuse is the process of creating independent standalone designs that can be used within larger, more complex designs. At the macro level, design reuse represents the process of utilizing an IP Management System that stores logical and physical modules with appropriate characterization, which enables the effective reuse of modules.

Design Synchronization

Design synchronization is the process of ensuring that the schematic and board are synchronized, that is, they have the same set of components with the same set of properties and connectivity. If you make changes in the schematic or the board after the initial packaging of the design, the schematic and the board will become out of sync.

Note: For more information about the design synchronization process, see the Cadence document *Design Synchronization and Packaging User Guide*.

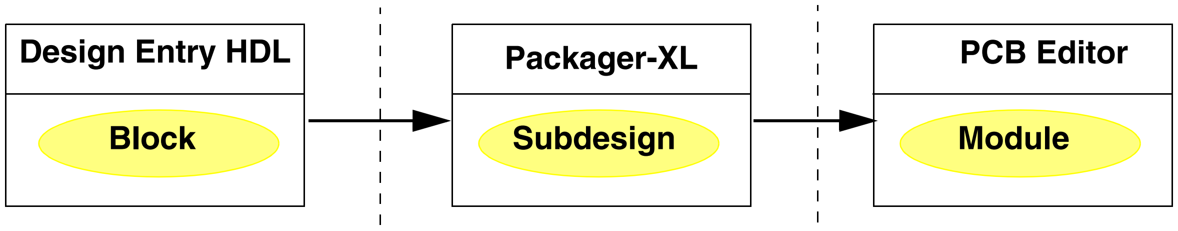
Hierarchical Block

A hierarchical block, also called a block, represents a schematic design for implementing functions that are complete in themselves. This block might be instantiated multiple times in another schematic, and each instance of the block can be assigned a different set of properties.

Note: A block may or may not contain packaging information.

Subdesign

You can use a hierarchical block as it is or use it as a subdesign. A subdesign uses the packaging information of the block for its logic. To create a subdesign, package the design using the `GEN_SUBDESIGN` directive. You will learn to create a subdesign in the Lesson 3-2: Packaging the Design.



Module

A module represents the PCB Editor layout for a particular design, which can be reused within larger, more complex layouts. Modules include a bounding box and a module name. A module has the same information as a subdesign.

Note: Modules created in PCB Editor are stored in files with a `.mdd` extension.

Index