cādence™

Part Developer Tutorial

Product Version 16.2
November 2008

Contents

1 Introduction

Part Developer is used to create parts. Its suite of features include:

- An Integrated Development Environment (IDE)
- Support for all part types
- Ability to create parts from PDFs
- Ability to import data from a variety of data formats, such as:
    - Capture
    - EDA XML
    - Si2 PinPak XML
    - Comma-Separated Value (CSV) file
    - Synopsis PTM model
    - Verilog model
    - VHDL model
- Ability to do engineering change order (ECO) updates from supported data formats
- Ability to export data in a variety of formats, such as:
    - Capture
    - EDA XML
    - Comma-Separated Value (CSV) file
- Creation and maintenance of part log and version information
- Interface Comparator, an easy-to-use tool for correcting part errors
- Powerful graphics-editing capabilities

This tutorial teaches you how to use Part Developer to quickly and effectively create and modify library parts.

Audience Profile

The intended audience profile for this tutorial includes users who maintain and modify digital libraries for design entry.

Pre-Requisites

The tutorial assumes familiarity with the following products:

- Project Manager
- Allegro Design Entry HDL
- PCB Symbol Editor

Important

This tutorial provides a step-by-step instruction on how to use Part Developer. For a detailed explanation of the features, see *Part Developer User Guide*.

How to Use This Tutorial

This tutorial provides a hands-on exercise on creating and modifying library parts. To gain the most from this tutorial, you should try out all the steps as documented in the tutorial. The tutorial is based on data provided through parts and datasheets in various formats, such as PDF, XML, and CSV.

Using the Samples

The tutorial works with the samples that are installed along with the software. The samples are stored in the following location:

`<your_install_dir>/doc/pdv_tut/tutorial_data`

This directory has the following subdirectories that are required for the successful completion of the tutorial.

Directory	Contents
`datasheets`	Contains the datasheets used in the tutorial
`import_files`	Contains the files for use in import procedures
`library_project`	Contains the project files `library_project.cpm` and `dp_proj.cpm` and the libraries `my_lib` and `dp_lib` used in the tutorial

Before starting the tutorial, do the following:

- Copy the samples to a local directory for which you have write permissions

  For the commands specified in the tutorial, you need to replace `your_work_area` with the name of the local directory in which you have copied the samples.

- Unset the `CDS_SITE` environment variable on your computer if it is set

Chapter Overviews

Chapter 2, "Getting Started," gets your started with the tutorial.

Chapter 3, "Creating a Flat Part," provides step-by-step instructions on how to create a flat part.

Chapter 4, "Creating Parts from CSV Files," describes how to create a part from data stored in a CSV file.

Chapter 5, "Creating Split Parts," details the methodology and steps in creating split parts.

Chapter 6, "Creating Parts from PDFs," describes the steps involved in creating parts from PDF datasheets.

Chapter 7, "Creating Asymmetrical Parts," provides step-by-step instructions on how to create asymmetrical parts.

Chapter 8, "Working with Differential Pairs," describes how to create differential pairs in different ways and remove differential pair information when required.

Chapter 9, "Creating Sizeable and HAS_FIXED_SIZE Symbols," details the methodology and steps involved in creating sizeable and HAS_FIXED_SIZE symbols.

Chapter 10, “Modifying Packages,” details the steps involved in modifying packages.

Chapter 11, “Modifying Symbols,” describes the steps involved in modifying symbols.

Chapter 12, “Editing Symbol Graphics,” describes the steps to edit symbol graphics through the Symbol Editor.

Chapter 13, “Importing and Exporting,” covers commonly used import and export procedures.

Chapter 14, “Part Logging and Versioning,” describes the methodology and steps in maintaining part versions and part logs.

Chapter 15, “Interface Comparator,” describes how to use the Interface Comparison feature to validate and correct parts.

2 Getting Started

Objective

To become familiar with different part types and ways to launch Part Developer.

In this chapter, you will learn:

- What a part is
- Types of parts
- Part creation methodology
- How to open an existing library project

Part Definition

Parts usually correspond to physical objects in a PCB, such as gates, chips, and connectors, which come in packages, such as DIP and SOIC. Normally, each of these packages has one or more functions repeated one or more times. These functions are represented graphically through symbols. The symbols are used in Design Entry HDL while the packages are used in Allegro PCB Editor.

In the HDL environment, a part is a collection of one or more of the following views:

- Package
- Symbol
- Simulation (mapfiles and wrappers)
- Part Table File

Note: For detailed explanations about the views of a part, see *Design Entry HDL Libraries Reference*.

Using Part Developer, you can easily create the views of a part.

Part Types

Parts can be classified into three types: symmetrical, asymmetrical, and split.

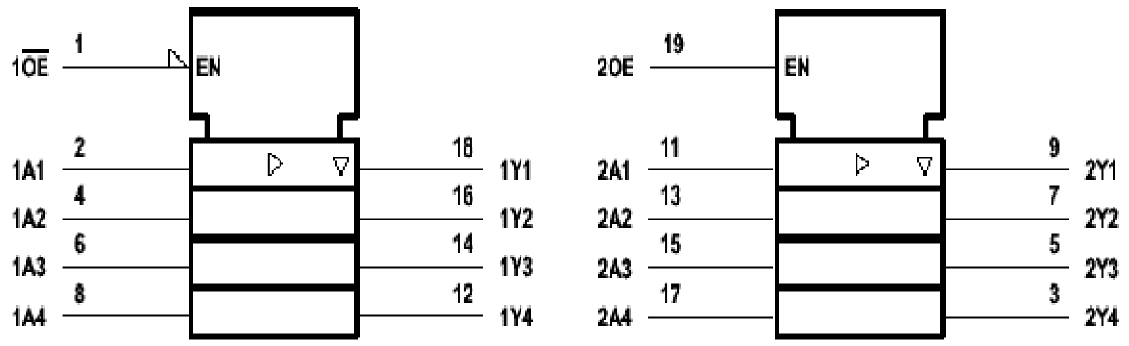
Symmetrical Parts

A part that has only one logical function repeated one or more times in a package is called a symmetrical part. For example, LS00 with four independent 2-input NAND gates is a symmetrical part.

In the context of the `chips.prt` implementation, a symmetrical part has the same logical pin list across all slots. This implies that all logical pins are present in all slots of the part.

Asymmetrical Parts

An asymmetrical part is a part in which multiple functions are present in a package. For example, LS241, an 8-slot part, with two kinds of functionality, is an asymmetrical part. The first four slots in such a package have the pin list A, Y, OE*, VCC, and GND, and the second four slots have the pin list A, Y, OE, VCC, and GND.



In the context of `chips.prt` implementation, an asymmetrical part has different pin lists across the slots.This implies that not all logical pins are present in all the slots of an asymmetrical part. The slots in which a pin is not present are represented by 0 in the `chips.prt` file.

An asymmetrical part has multiple symbols, where each symbol represents one functionality. For example, LS241 has two symbols, each representing a specific function.

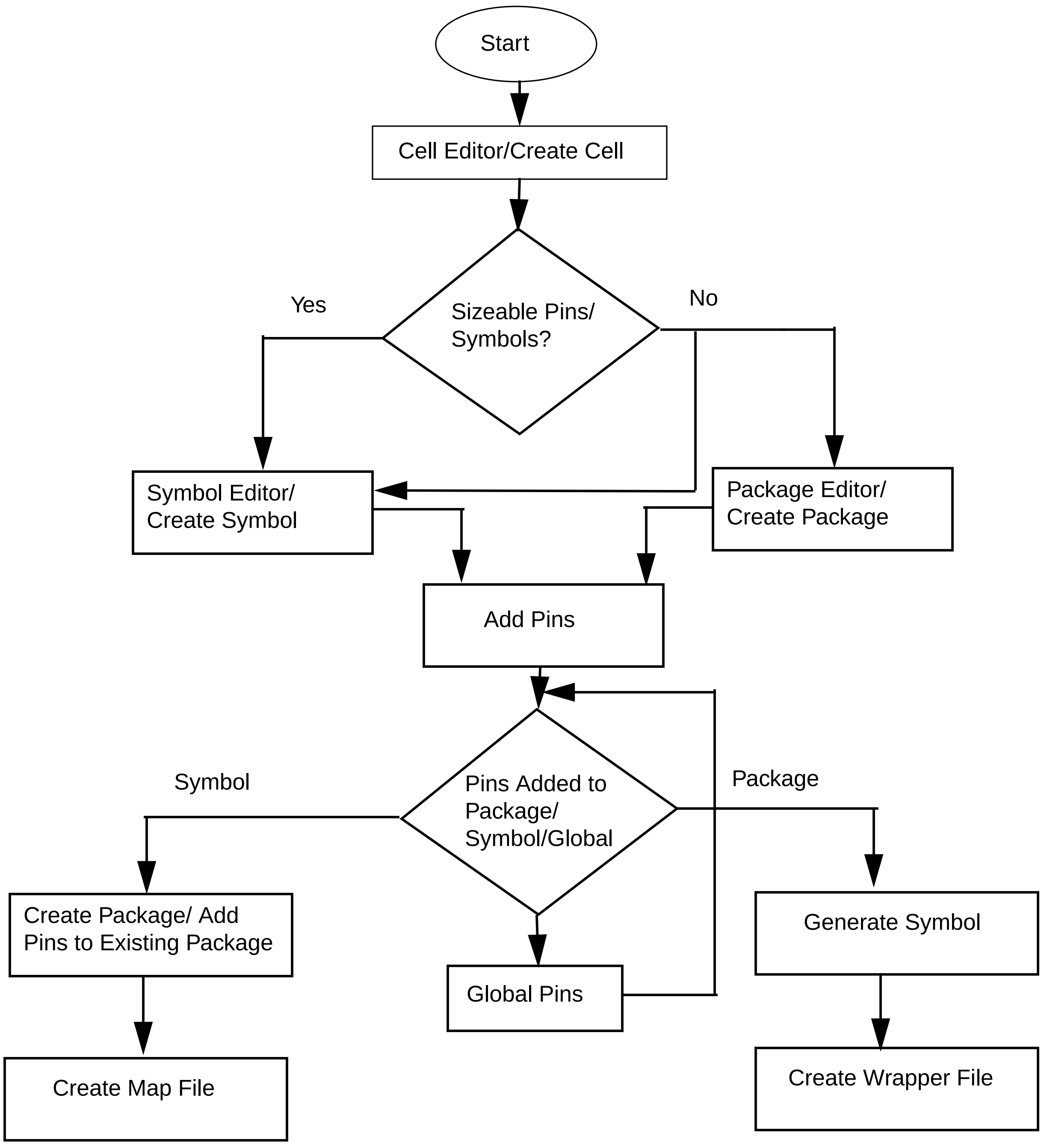
Split Parts

A split part is a special case of an asymmetrical part. This part consists of a package in which logical pins are split across multiple slots. Split parts are useful for creating symbols for large pin-count devices. In a split part, each symbol represents a different function. The difference is that while in an asymmetrical part it is possible that a symbol represents multiple slots, a split part has one symbol representing only one slot.

To create and use split parts, you need to add either the `SPLIT_INST` and `$LOCATION` properties or the `SPLIT_INST_NAME` property on the symbol or the chips. This can be done through *Tools – Setup*. For more information about these properties, see *PCB Systems Properties Reference*.

Part Creation Methodology

The following graphic illustrates the methodology to be followed while creating parts:


Start
Cell Editor/Create Cell
Sizeable Pins/ Symbols?
Yes
No
Symbol Editor/ Create Symbol
Package Editor/ Create Package
Add Pins
Pins Added to Package/ Symbol/Global
Symbol
Package
Create Package/ Add Pins to Existing Package
Global Pins
Generate Symbol
Create Map File
Create Wrapper File

As displayed, the part creation process begins with creating the new cell in a selected library followed by the logical pin entry. Logical pin entry can be done in one of the following ways:

- Using the Add Pin dialog box. You open this dialog box from within the Package or the Symbol Editor. You can add all the pins for a part through this dialog box.
- Add logical pins directly to a package using the *Logical Pins* grid of the Package Editor.
- Add logical pins directly to a symbol using the *Logical Pins* grid of the Symbol Editor.

Important

To make a sizeable symbol, you need to enter sizeable logical pins using the *Add Pin* dialog box accessed through the Symbol Editor. This is required because `SIZE` is a pin-level property applicable on a symbol pin.

After pins are added, you create packages or symbols. Packages can be created using the Package Editor. The Symbol Editor is used to create symbols. The wrappers and mapfiles can be created after symbols and packages are created. The Verilog/VHDL map/wrapper file editors are used to create wrappers and map files.

Starting the Tool

Part Developer can be launched from the following:

- Project Manager
- Command prompt
- Library Explorer

You will use Project Manager to launch Part Developer.

1. Type the following command at the command prompt:

   ```
   projmgr
   ```

   The Cadence Product Choices dialog box appears.

2. Select the Allegro PCB Librarian XL license and click *OK*.

   Note: If you want Project Manager to use the selected license for all future invocations, select the *Use As Default* check box.

   Project Manager opens.

   Note: Project Manager enables you to create both library and design projects. To know more about how to create library and design projects, see *Project Manager User Guide* and *Library Explorer User Guide*.

Next, you will open the library project located at `<`*`your_work_area`*`>/tutorial_data/library_project`.

3. To open an existing library project, click *Open Project*.

   The Open Project dialog box appears.

4. Browse to `<`*`your_work_area`*`>/tutorial_data/library_project`, select the `tutorial_project.cpm` project file, and click *Open*.

   The Allegro PCB Librarian XL page opens in Project Manager.

5. Click *Part Developer* to launch Part Developer on the selected project.

   Part Developer opens.

Summary

In this chapter, you learned about the definition of a part, different part types, and the part creation methodology. You also learned how to open a library project in Project Manager and launch Part Developer in the project.

3

Creating a Flat Part

Objective

To become familiar with the steps in creating a single-slot flat part.

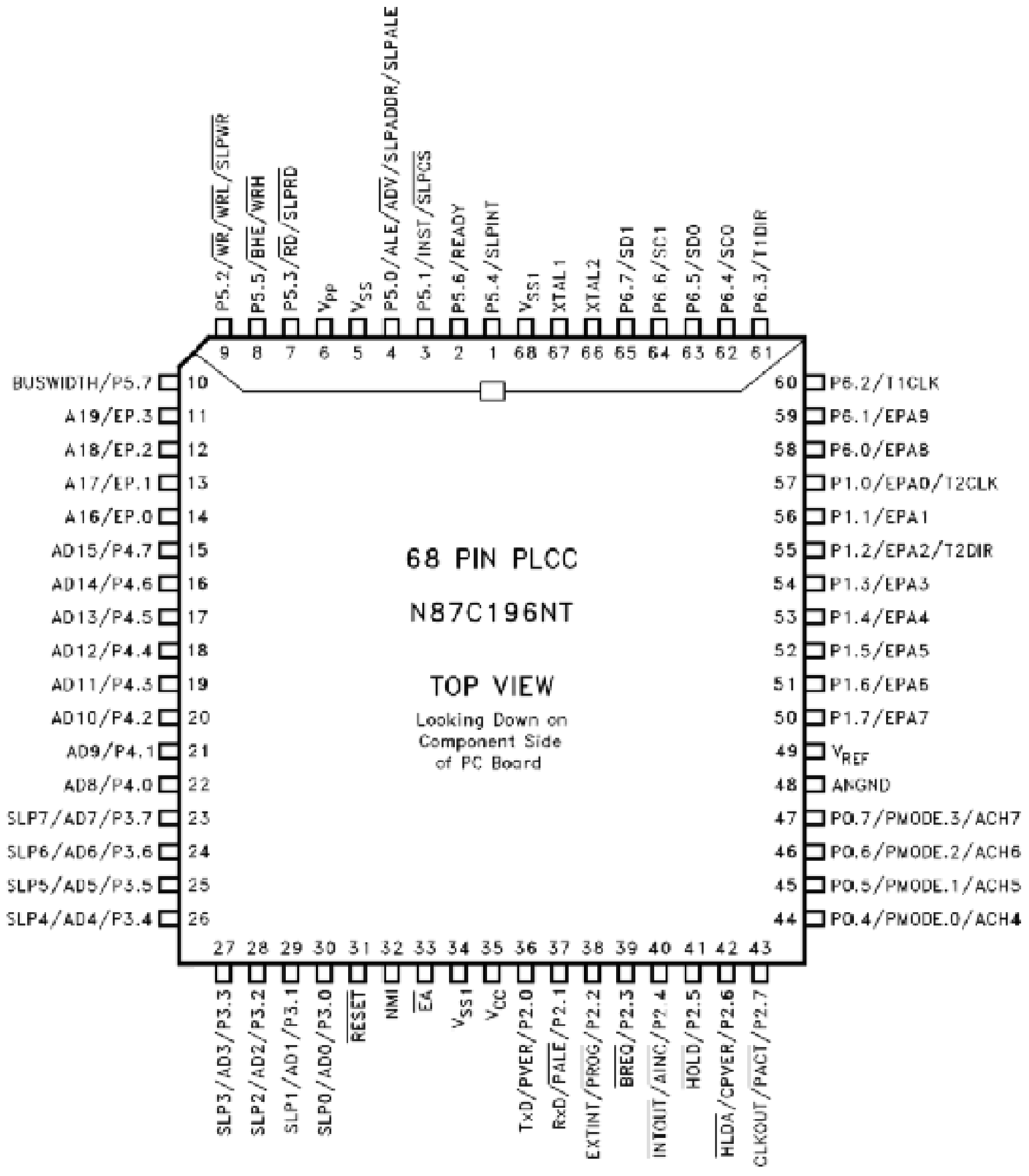
In this chapter, you will learn to:

- Set up Part Developer
- Specify the logical and physical pin information
- Create packages:
  - Understand several techniques in Part Developer for the quick development of parts and the views
  - Access Allegro footprints to specify the `JEDEC_TYPE` property
  - Use filters
  - Extract pin numbers from the Allegro footprint
  - Move global pins from the *Logical Pins* grid to the *Global Pins* grid
  - Verify the part with the selected footprint
- Create symbols

Overview

The n87c196nt part is used to describe the steps in creating a flat part. The datasheet (`n87c196nt.pdf`) is available at `<`*`your_inst_dir`*`>/doc/pdv_tut/tutorial_data/datasheets`. It is a 68-pin part and comes in the PLCC package.

A partial snapshot of the datasheet is displayed below:



Setting Up Part Developer

The first step toward creating a part is to set up Part Developer to provide default values to various fields, such as input and output loads, and user-defined properties to packages and symbols. For more information, see the *Configuring Part Developer* chapter in *Part Developer User Guide*.

Task Overview

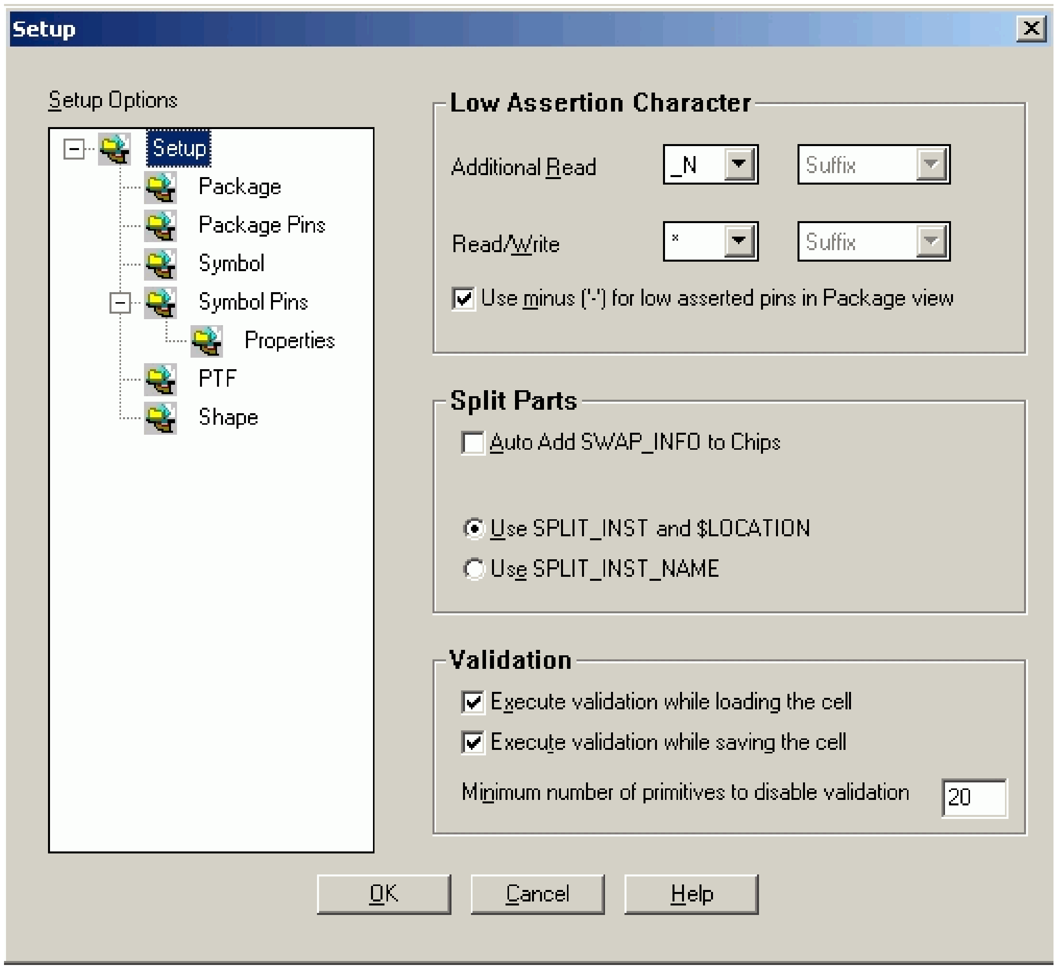
Set up Part Developer to do the following:

- Associate and display a property called `Library_Name` with the value `my_library` for all packages and symbols.
- All the packages should have a property called `PACKAGE_CREATOR` with value as `?`.
- Symbols should display the pin text in 0.08 inches.

Steps

1. Choose *Tools – Setup*.

   The Setup dialog box appears.



Low Assertion Character

Additional Read: _N | Suffix

Read/Write: * | Suffix

[✓] Use minus ('-') for low asserted pins in Package view

Split Parts

[ ] Auto Add SWAP_INFO to Chips

(●) Use SPLIT_INST and $LOCATION

( ) Use SPLIT_INST_NAME

Validation

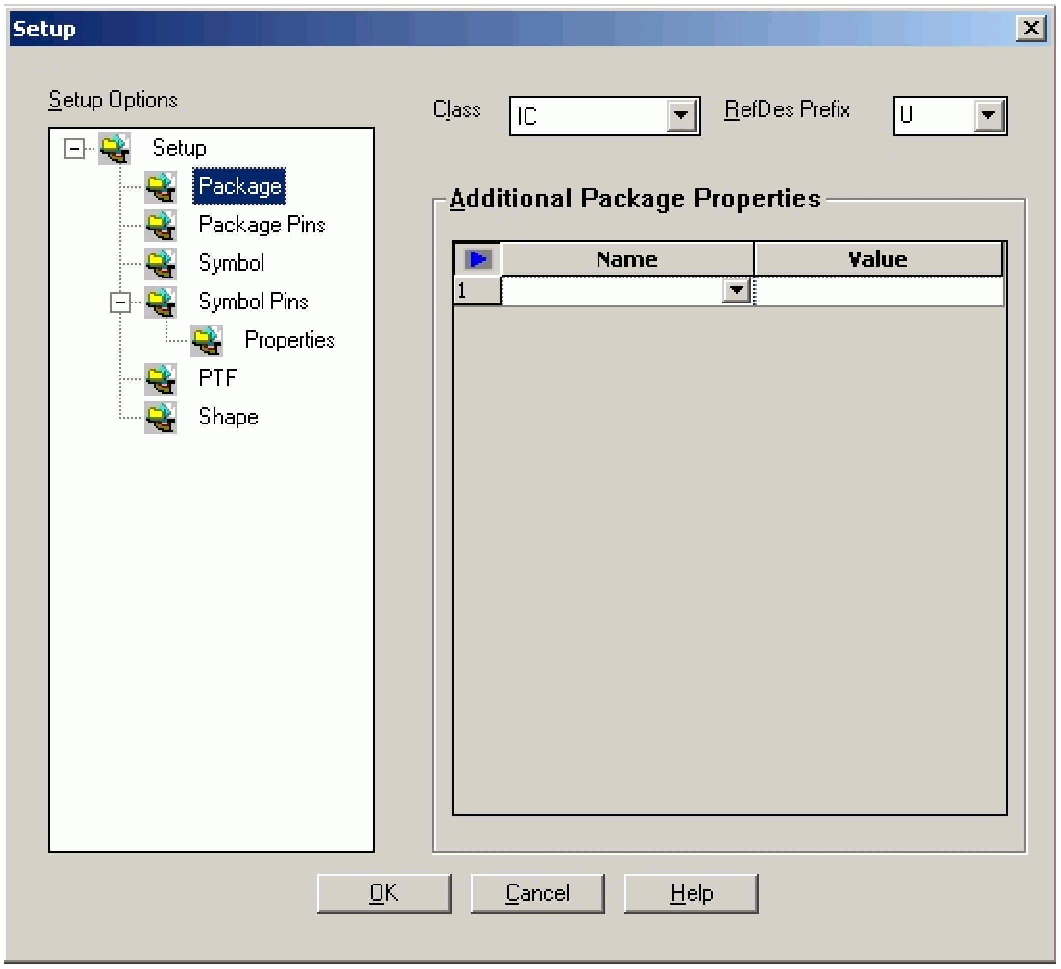
[✓] Execute validation while loading the cell

[✓] Execute validation while saving the cell

Minimum number of primitives to disable validation: 20

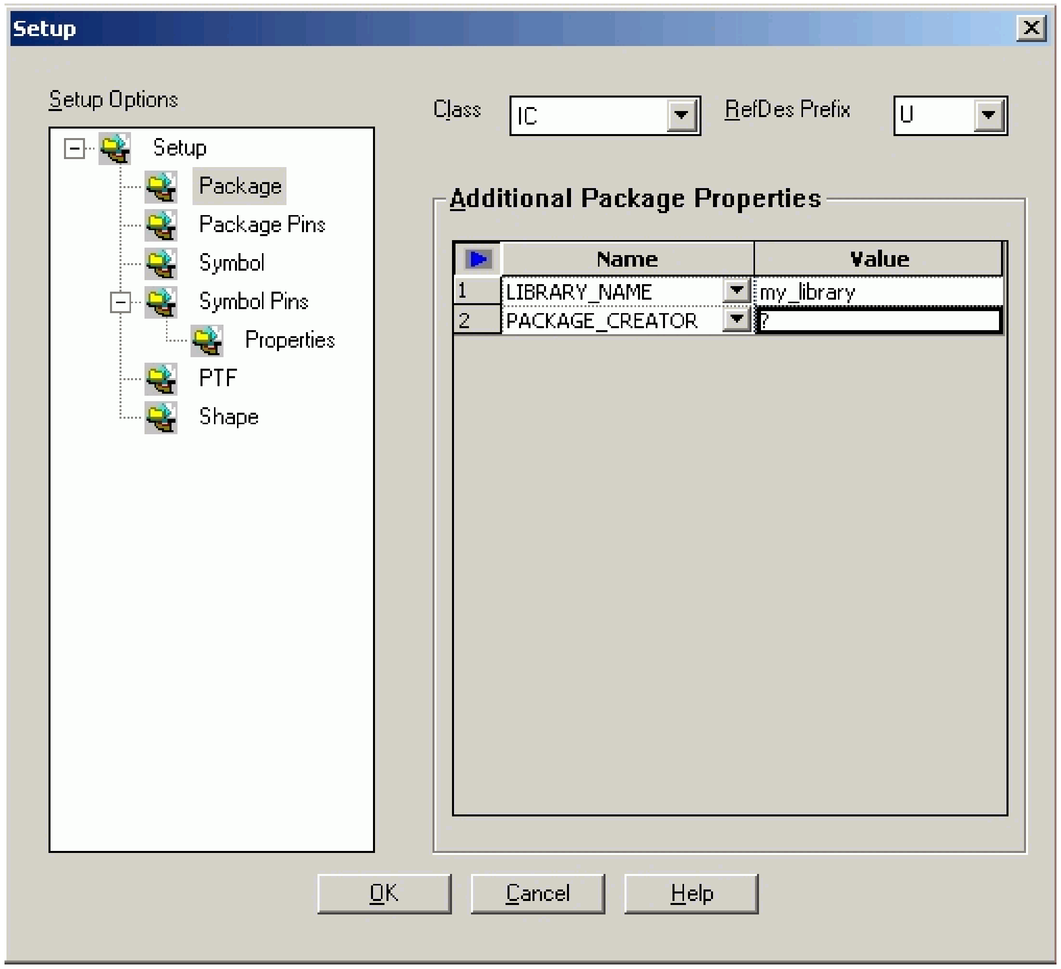
2. Click on the *Package* node in the *Setup Options* tree.

   The *Package* options appear in the Setup dialog box.



3. Type `LIBRARY_NAME` and `my_library` in the *Name* and *Value* columns, respectively.
4. Press Ctrl + I to create a blank row in the *Additional Package Properties* grid.
5. Type `PACKAGE_CREATOR` and ? in the *Name* and *Value* columns, respectively.

The *Package* panel should appear as follows:



Class: IC
RefDes Prefix: U

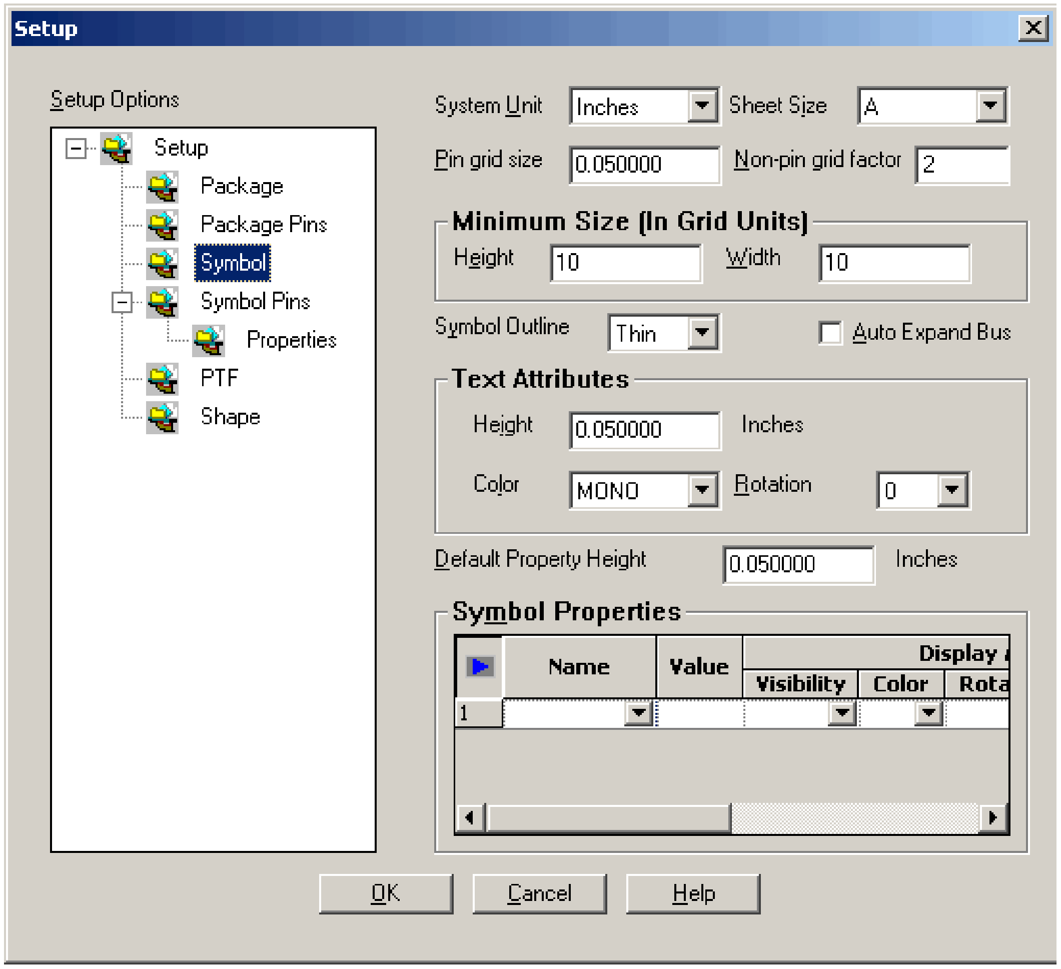
Additional Package Properties

	Name	Value
1	LIBRARY_NAME	my_library
2	PACKAGE_CREATOR	?

Next, you need to add the `LIBRARY_NAME` property to the symbols and configure the symbol pin text to display pin text in 0.08 inches.

6. Click on the *Symbol* node in the *Setup Options* tree.

The *Symbol* panel appears in the Setup dialog box.



System Unit: Inches
Sheet Size: A
Pin grid size: 0.050000
Non-pin grid factor: 2

Minimum Size (In Grid Units)
Height: 10
Width: 10

Symbol Outline: Thin
Auto Expand Bus: ☐

Text Attributes
Height: 0.050000 Inches
Color: MONO
Rotation: 0

Default Property Height: 0.050000 Inches

Symbol Properties

	Name	Value	Display Visibility	Color	Rota
1					

7. Enter `LIBRARY_NAME` and `my_library` in the *Name* and *Value* columns, respectively, in the *Symbol Properties* grid.

   Next, you need to determine the display parameters.

8. Select *Both* in the *Visibility* column to ensure that both the property name and its value are visible in the symbol.

9. Select *Mono* in the *Color* field.

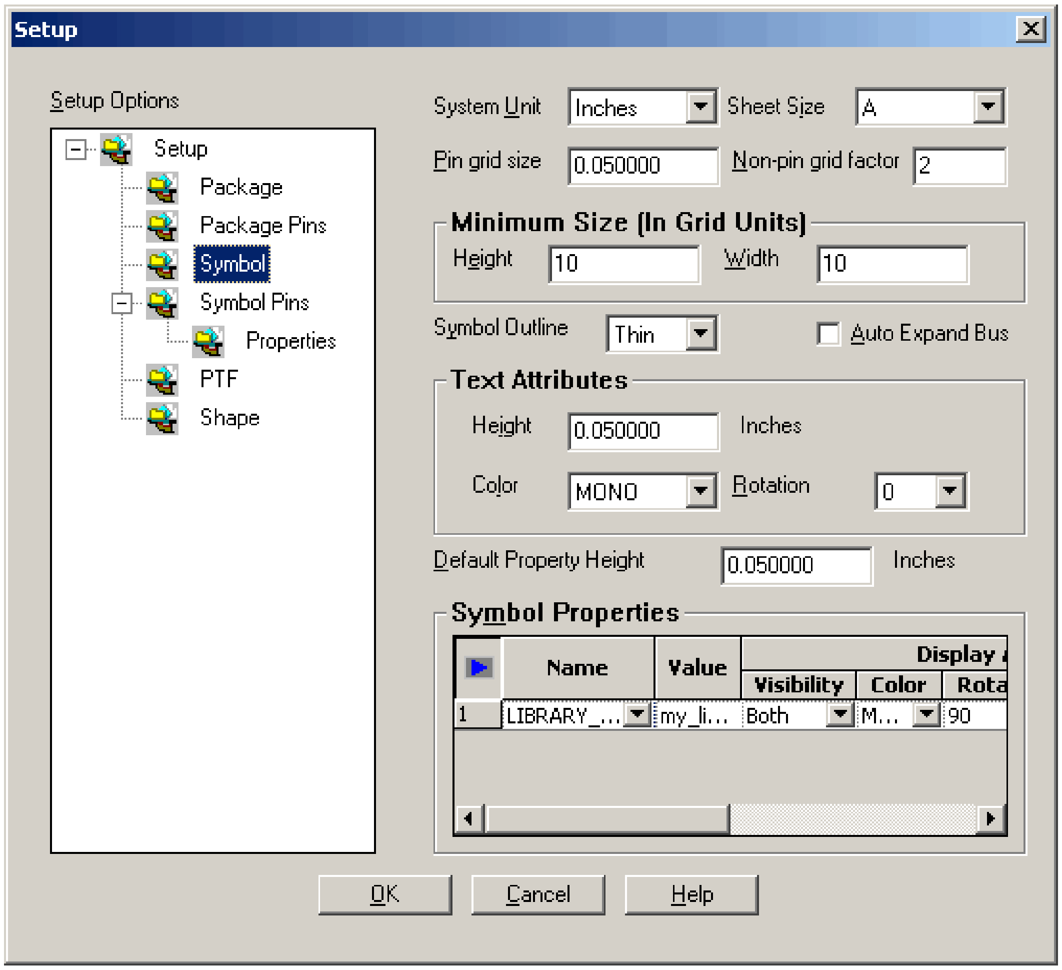
10. Select 90 in the *Rotation* field.

    This ensures that the property is displayed at an angle of 90 degrees to the symbol.

11. Select *Top-Left* in the *Location* field.

    This ensures that the property is displayed in the top-left corner of the symbol.

The filled *Symbol* panel is as follows:



System Unit: Inches
Sheet Size: A
Pin grid size: 0.050000
Non-pin grid factor: 2

Minimum Size (In Grid Units)
Height: 10
Width: 10

Symbol Outline: Thin
Auto Expand Bus: ☐

Text Attributes
Height: 0.050000 Inches
Color: MONO
Rotation: 0

Default Property Height: 0.050000 Inches

Symbol Properties

	Name	Value	Display Visibility	Color	Rota
1	LIBRARY_...	my_li...	Both	M...	90

Next, set up Part Developer so that pin text appears in 0.08 inches.

12. Click on *Symbol Pins* in the *Setup Options* tree.

13. Change the value in the *Pin Text Height* column to `0.0800`.

14. Click *OK*.

Next, you will create the part.

Creating the n87c196nt Part

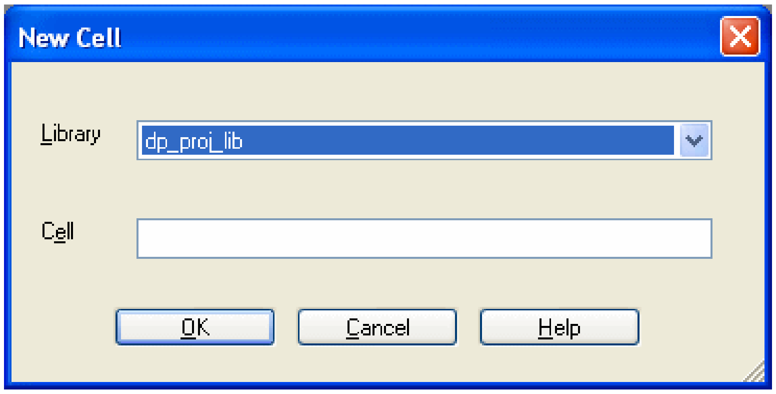
Task Overview

Create the n87c196nt part in the `my_lib` library.

Steps

1. Choose *File – New – Cell*.

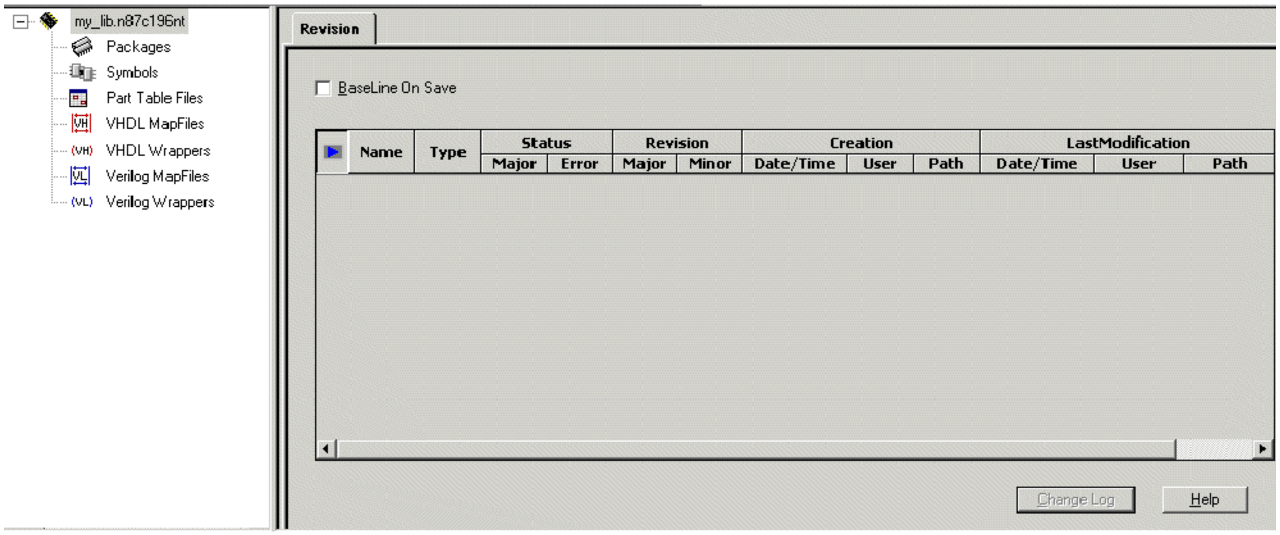
   The New Cell dialog box appears.



Library: dp_proj_lib

2. Select the `my_lib` library.
3. Type `n87c196nt` in the *Cell* field and click *OK*.

The part n87c196nt appears in the Cell Editor.



Next, you enter the logical and physical pins.

First, you create either a package or a symbol. In creating a package first, you have the benefit of specifying the logical and physical pins and the footprint information and doing the logical-to-physical pin mapping in a single step. Creating symbols is useful when you want to enter sizeable pins.

You will create the package first because n87c196nt does not have any sizeable pins.

Creating a Package

Task Overview

Create a package to facilitate pin entry.

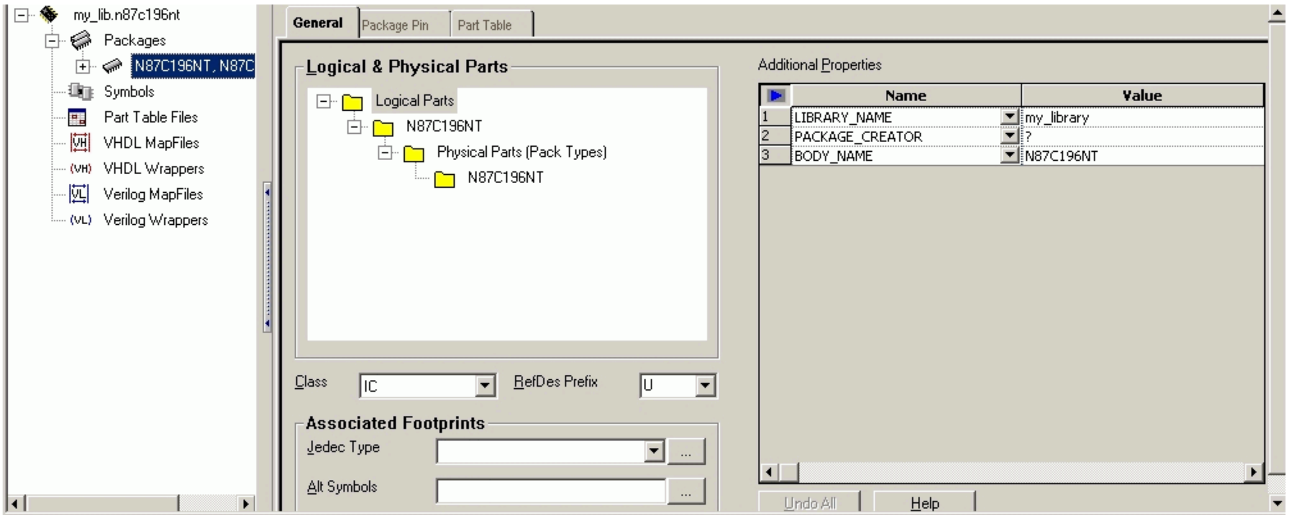
Steps

1. Right-click on the *Packages* entry in the cell tree and select *New*.

   A new package gets created. By default, the package name has the same name as the cell name.

   On the General page, the *Logical & Physical Parts* tree shows the logical and physical parts for a part. A logical part defines the logical pins for a part and is mapped to one or more physical parts. A physical part consists of the logical-to-physical pin mapping and a set of physical properties. Each primitive entry in the *chips* file represents a physical part. The name of a physical part is either the same as the logical part or the logical part name suffixed by a package type. The default physical part has the same name as the logical part. The packages that are valid for the specified *PART_NAME* appear under the *Pack_Type* entry of the tree.

For more information about logical and physical parts, see the chapter *How Packager-XL Selects and Names Parts* in the *Packager-XL Reference*.

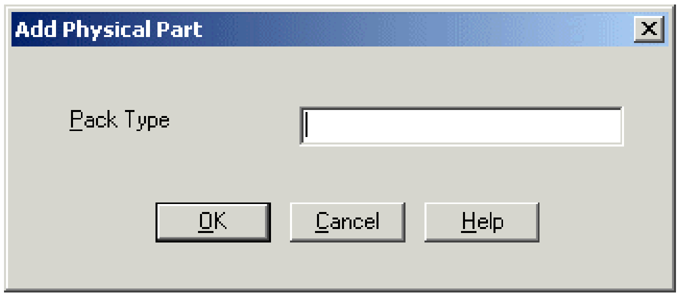


	Name	Value
1	LIBRARY_NAME	my_library
2	PACKAGE_CREATOR	?
3	BODY_NAME	N87C196NT

Notice that the properties that you have specified in *Setup* appear in the *Additional Properties* grid.

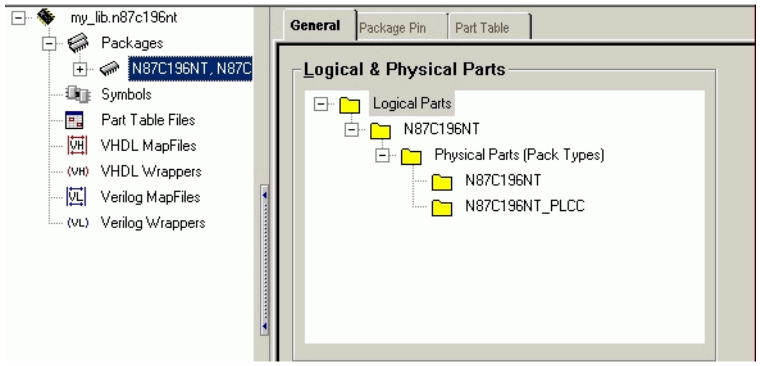
2. Because the PLCC package has the same pin configuration as the default package, right-click on the *Physical Parts (Pack Types)* entry in the *Logical & Physical Parts* tree and select *New*.

   The Add Physical Part dialog box appears.



3. Specify PLCC in the *Pack Type* field and click *OK*.

The new package appears in the *Logical & Physical Parts* tree.



Next, you need to enter the logical pins.

Entering the Logical Pins

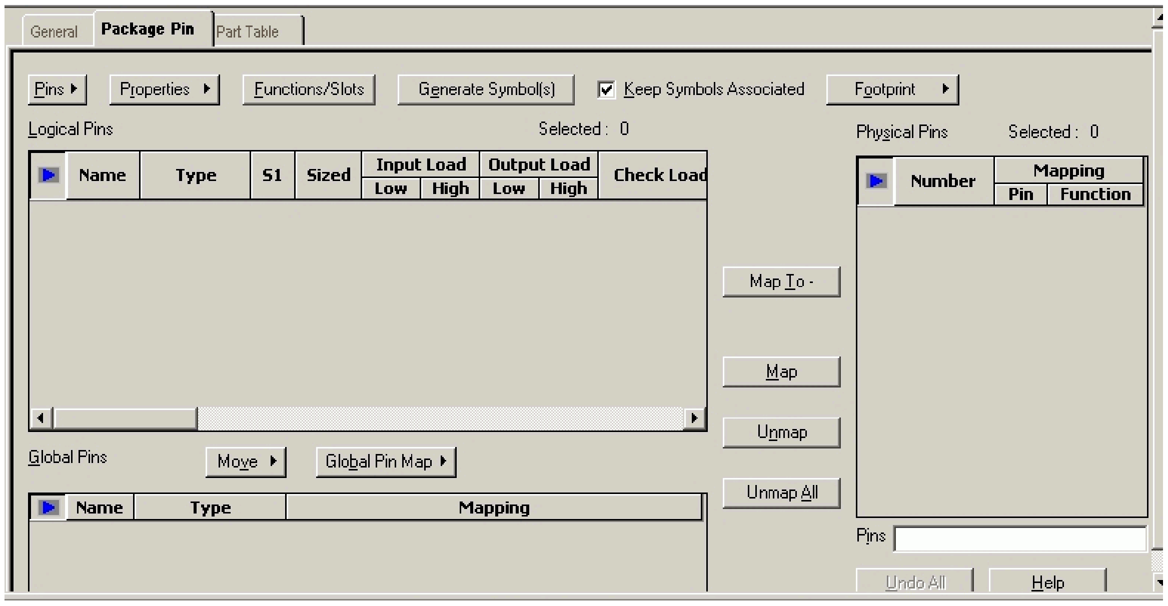
Task Overview

Enter the logical pins as per the datasheet.

Steps

1. Click the *Package Pin* tab.

The Package Pin page appears.



2. To add logical pins, choose *Pins – Add*.

   The Add Pin dialog box appears.

   Enter the pin information as given in the datasheet into Part Developer. Open the datasheet and go to the Pin Descriptions table.

3. The first few pins in the Pin Descriptions table are power pins. To enter a power pin, select *POWER* in the *Type* drop-down list box.

4. Enter `vcc` in the *Prefix* field and click *Add*.

   Similarly, add `Vss`,`Vss1`, `VREF`, `VPP`, and `ANGND` as power pins.

Important

Note that the Pin Descriptions table has multiple entries for the Vss1 power pin. In Part Developer, you need to enter the name only once in the logical pins list. Later, you can map multiple physical pins to a power pin.

5. Next, to enter an input pin, select *INPUT* from the *Type* drop-down list box.

6. Enter `XTAL1` in the *Prefix* field and click *Add*.

7. The next couple of pins in the Pin Descriptions table are output pins. To enter an output pin, select *OUTPUT* from the *Type* drop-down list box.

8. Enter `XTAL2` in the *Prefix* field box and click *Add.*

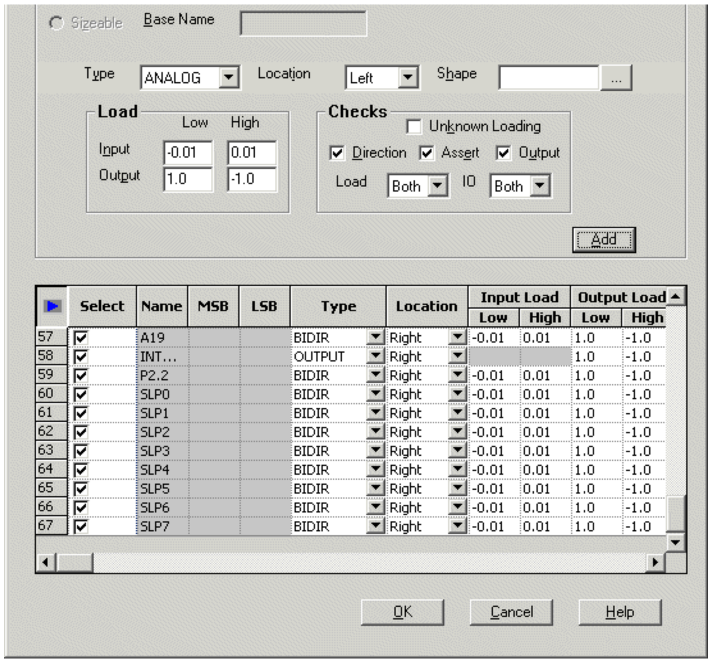
Similarly, add P2.7 as an output pin.

9. RESET, the next pin in the table, is an active low input pin. To add an active low input pin, select *INPUT* from the *Type* drop-down list box.
10. Enter `RESET` in the *Prefix* field.
11. To make a pin active low, enter * in the *Suffix* field and click *Add*.

    Similarly, add all the pins up to P6.3 specified in the Pin Descriptions table of the datasheet.
12. The pins `EPA0-9` are *BIDIR* pins. To add pins such as these, enter `EPA` in the *Prefix* field, `0` in the *From* field, and `9` in the *To* field.
13. Select *BIDIR* in the *Type* drop-down list box and click *Add*.

    Similarly, enter all the pins as specified in the datasheet.

    The filled Add Pin dialog box should appear as follows (displayed partially):



Type: ANALOG Location: Left Shape:

Load

	Low	High
Input	-0.01	0.01
Output	1.0	-1.0

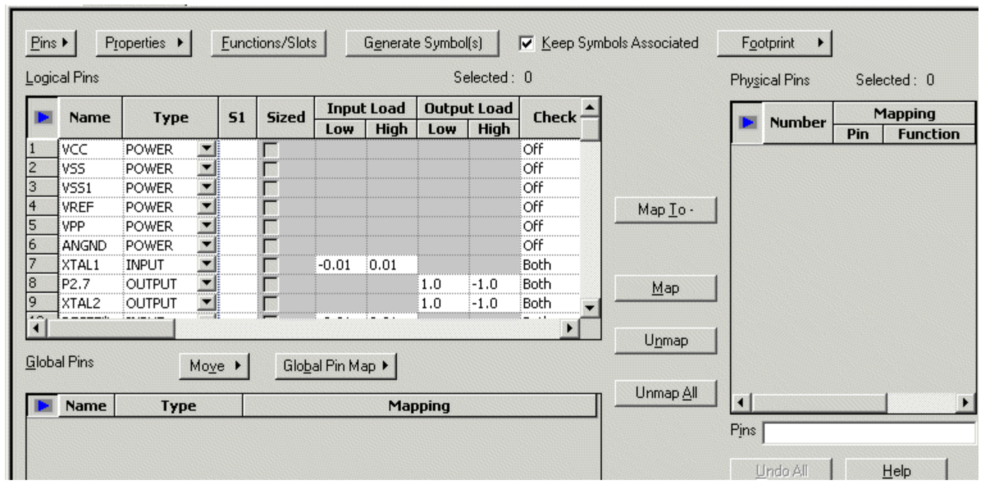
Checks: ☐ Unknown Loading ✓ Direction ✓ Assert ✓ Output
Load: Both IO: Both

	Select	Name	MSB	LSB	Type	Location	Input Load		Output Load	
							Low	High	Low	High
57	✓	A19			BIDIR	Right	-0.01	0.01	1.0	-1.0
58	✓	INT...			OUTPUT	Right			1.0	-1.0
59	✓	P2.2			BIDIR	Right	-0.01	0.01	1.0	-1.0
60	✓	SLP0			BIDIR	Right	-0.01	0.01	1.0	-1.0
61	✓	SLP1			BIDIR	Right	-0.01	0.01	1.0	-1.0
62	✓	SLP2			BIDIR	Right	-0.01	0.01	1.0	-1.0
63	✓	SLP3			BIDIR	Right	-0.01	0.01	1.0	-1.0
64	✓	SLP4			BIDIR	Right	-0.01	0.01	1.0	-1.0
65	✓	SLP5			BIDIR	Right	-0.01	0.01	1.0	-1.0
66	✓	SLP6			BIDIR	Right	-0.01	0.01	1.0	-1.0
67	✓	SLP7			BIDIR	Right	-0.01	0.01	1.0	-1.0

Note that by default, the check box in the *Select* column is selected for all pins. This implies that all the pins will get added to the package.

14. Click *OK*.

The pins appear in the *Logical Pins* grid of the Package Editor.



	Name	Type	S1	Sized	Input Load		Output Load		Check
					Low	High	Low	High	
1	VCC	POWER							Off
2	VSS	POWER							Off
3	VSS1	POWER							Off
4	VREF	POWER							Off
5	VPP	POWER							Off
6	ANGND	POWER							Off
7	XTAL1	INPUT			-0.01	0.01			Both
8	P2.7	OUTPUT					1.0	-1.0	Both
9	XTAL2	OUTPUT					1.0	-1.0	Both

Number	Mapping	
	Pin	Function

Name	Type	Mapping

Next, you need to specify the physical pins. Physical pins can be entered in one of the two ways:

- Manually
- By specifying a footprint and then extracting the pins from the footprint

We will be using the second option to add the physical pins.

Specifying the Footprints

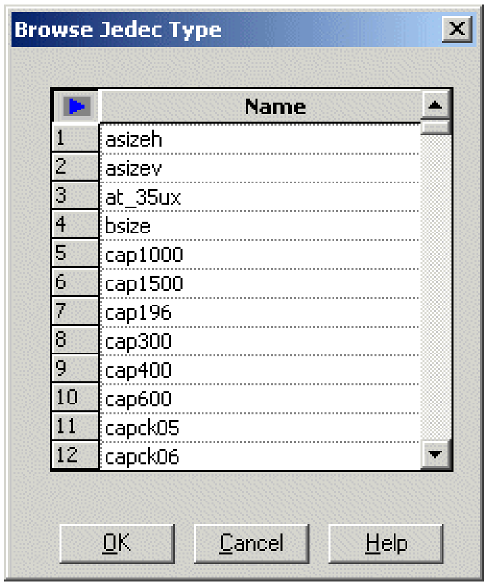
Task Overview

Specify the JEDEC_TYPE and ALT_SYMBOLS properties for the package.

Steps

1. Click on the *General* tab.
2. To specify the JEDEC_TYPE, click on the browse button next to the *Jedec Type* field in the *Associated Footprints* group box.

The Browse Jedec Type dialog box appears.



	Name
1	asizeh
2	asizev
3	at_35ux
4	bsize
5	cap1000
6	cap1500
7	cap196
8	cap300
9	cap400
10	cap600
11	capck05
12	capck06

As you can see, this is a big list and you need to scroll down to get to the required data. To view only a select set of footprints, you can set filters.

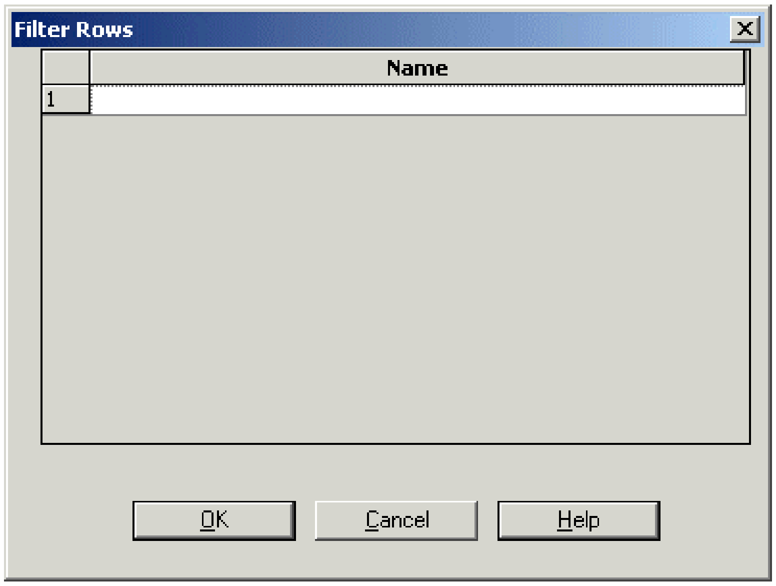
Important

Filters can be set in any of the grids, such as *Logical Pins*, *Physical Pins,* and so on.

Setting a Filter

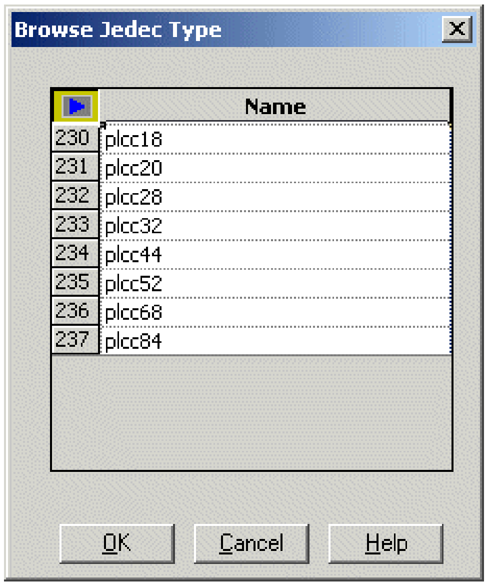
1. Right-click on any one of the rows in the Browse Jedec Type dialog box and select *Filter Rows*.

The Filter Rows dialog box appears.



2. Enter `PLCC*` in the *Name* column and click *OK*.

   The Browse Jedec Type dialog box displays only the PLCC footprints.



	Name
230	plcc18
231	plcc20
232	plcc28
233	plcc32
234	plcc44
235	plcc52
236	plcc68
237	plcc84

3. Select *plcc68* and click *OK*.
4. Similarly, select the Alt_Symbols for the part.

Note: To remove filter, you need to right-click on the filtered grid and select the *Unhide All Rows* options.

Next, you will extract the physical pins from the footprint.

Entering Physical Pins

Task Overview

Extract physical pin information from the footprint.

Steps

1. Click on the *Package Pin* tab.

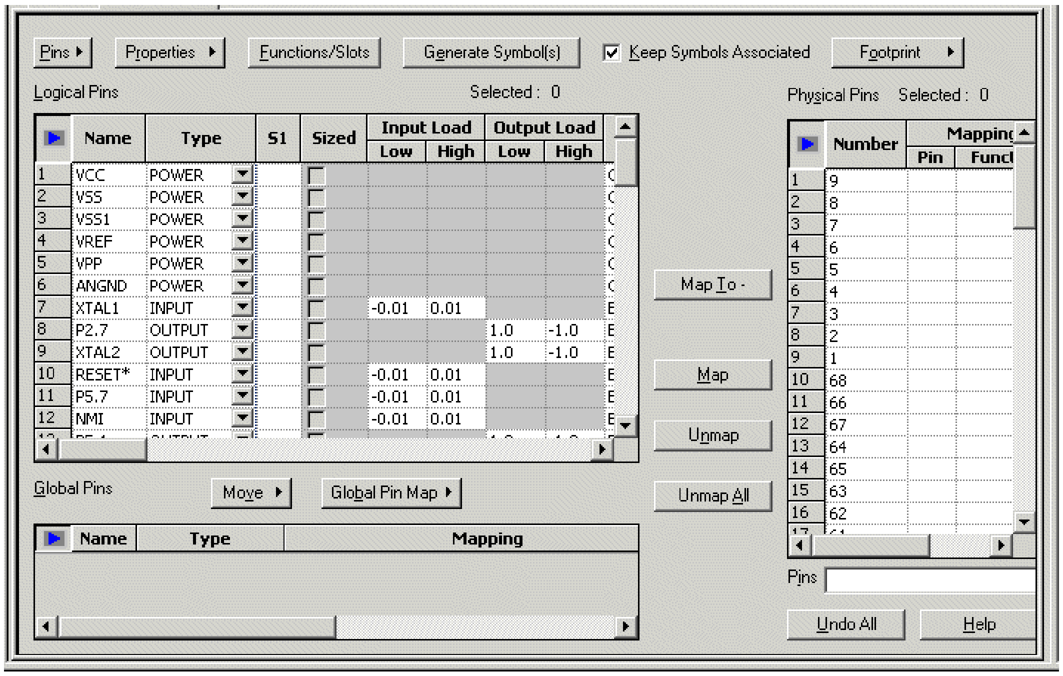
   The Package Pin page appears.

2. Select *Footprint – Extract from Footprint*.

   The Part Developer message box appears. This message box states that all physical pins that exist in the package but are not available in the selected footprint will be deleted.

3. Because there are no existing physical pins, click *Yes*.

The physical pins get extracted from the footprint and appear in the *Physical Pins* grid. Note that the pin numbers are not sorted.



	Name	Type	S1	Sized	Input Load Low	Input Load High	Output Load Low	Output Load High
1	VCC	POWER		☐				
2	VSS	POWER		☐				
3	VSS1	POWER		☐				
4	VREF	POWER		☐				
5	VPP	POWER		☐				
6	ANGND	POWER		☐				
7	XTAL1	INPUT		☐	-0.01	0.01		
8	P2.7	OUTPUT		☐			1.0	-1.0
9	XTAL2	OUTPUT		☐			1.0	-1.0
10	RESET*	INPUT		☐	-0.01	0.01		
11	P5.7	INPUT		☐	-0.01	0.01		
12	NMI	INPUT		☐	-0.01	0.01		

	Name	Type	Mapping

	Number	Mapping Pin	Mapping Funct
1	9		
2	8		
3	7		
4	6		
5	5		
6	4		
7	3		
8	2		
9	1		
10	68		
11	66		
12	67		
13	64		
14	65		
15	63		
16	62		

4. To sort the physical pins, click on the *Number* column heading.

Notice that the power pins are listed in the *Logical Pins* grid. This implies that when a symbol is generated from the package, the power pins will be visible in the symbol. To avoid that, you need to move the pins to the *Global Pins* grid. The pin types that are considered global pins are as follows:

- POWER
- GROUND
- NC

Moving Pins from Logical Pins to Global Pins

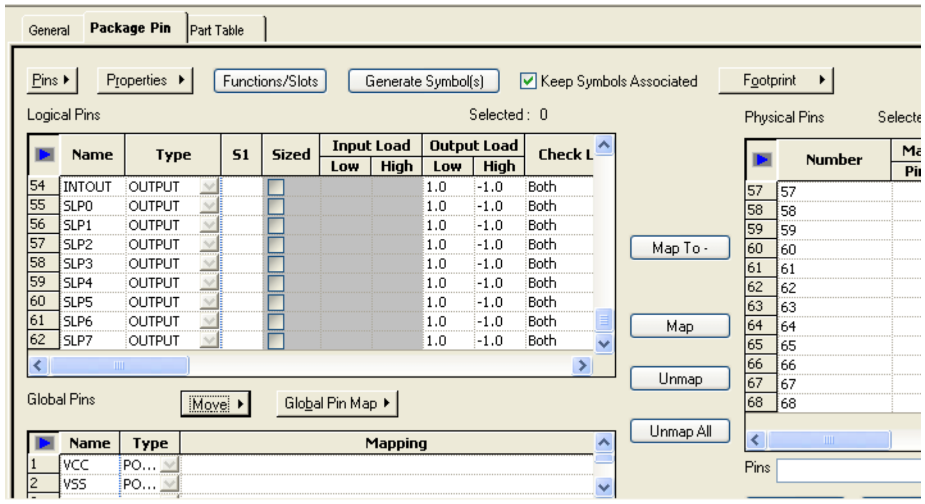
Task Overview

Move the power pins to the *Global Pins* grid.

Steps

1. Click on the *Type* heading to sort the pins by their type.
2. Select all the power pins and select *Move – Logical Pins to Global*.

   The selected power pins move to the *Global Pins* grid.



	Name	Type	S1	Sized	Input Load		Output Load		Check L
					Low	High	Low	High	
54	INTOUT	OUTPUT					1.0	-1.0	Both
55	SLP0	OUTPUT					1.0	-1.0	Both
56	SLP1	OUTPUT					1.0	-1.0	Both
57	SLP2	OUTPUT					1.0	-1.0	Both
58	SLP3	OUTPUT					1.0	-1.0	Both
59	SLP4	OUTPUT					1.0	-1.0	Both
60	SLP5	OUTPUT					1.0	-1.0	Both
61	SLP6	OUTPUT					1.0	-1.0	Both
62	SLP7	OUTPUT					1.0	-1.0	Both

	Number	Ma Pi
57	57	
58	58	
59	59	
60	60	
61	61	
62	62	
63	63	
64	64	
65	65	
66	66	
67	67	
68	68	

	Name	Type	Mapping
1	VCC	PO...	
2	VSS	PO...	

Next, you need to do the logical-to-physical and global-to-physical pin mapping.

Pin Mapping

Task Overview

Do logical-to-physical and global-to-physical pin mappings.

Steps

The general method to map physical pins with logical pins is to select the slots in which the logical pins are present and then select the pins from the *Physical Pins* list, and click *Map*. The mapping is done in the order in which the logical and physical pins were selected.

Because n87c196nt is a flat part, there is only one slot. Therefore, for all the pin mappings will be done for the slot S1.

For global-to-physical pin mapping, you need to select the required global pins and corresponding physical pins, and click *Map*. It is possible to map one global pin to more than one physical pin.

1. As per the datasheet, physical pin 1 is mapped to logical pin P5.4. Therefore, select *1* in the *Physical Pins* list and slot S1 for logical pin *P5.4* in the *Logical Pins* list, and click *Map*.

   The physical pin 1 is mapped to the logical pin P5.4.

   Next, we will map multiple pins in a single step.

2. Select the slots next to pins P5.6, P5.1, and P5.0 and then select physical pins 2,3, and 4 and click *Map*. The pins get mapped in the order in which the slots were selected.

   The next two physical pins are mapped to power pins.

Important

Unlike logical pins, global pin mapping can be done for only one pin at a time. This is because one global pin can be mapped to multiple physical pins.

3. Select *VSS* in the *Global Pins* grid and 5 in the *Physical Pins* grid and click *Map*.

   As you do the pin mappings, the logical and the physical pins grid will keep getting filled up. This can be disconcerting when dealing with a large number of pins. Using an RMB option, you can configure Part Developer to hide the mapped pins.

Hiding Mapped Pins

Task Overview

Hide the mapped pins.

Steps

1. Right-click on the *Logical Pins* grid and select *Hide Mapped Pins*.
2. Right-click on the *Physical Pins* grid and select *Hide Mapped Pins*.

   The mapped pins are hidden from the *Logical* and *Physical Pins* grids. You may do this as many number of times as required.

Important

The first header of a grid provides a visual indication of whether all the rows and columns are displayed in the grid. If all the rows/columns are visible, the filter viewer appears in blue. In case some rows and columns are hidden, the viewer appears in green.

Continuing Pin Mapping

1. Complete the pin mapping by following the methods mentioned above.
2. Save the part.

   Next, you will create the symbol.

Creating Symbols

A symbol represents a unique function group in a package. There are multiple ways to create symbols for a package.

1. Use the *Generate Symbols* pop-up menu option in the cell tree on the package name or the function group.
2. Use the *Generate Symbol(s)* button on the Package Pin page of the Package Editor.

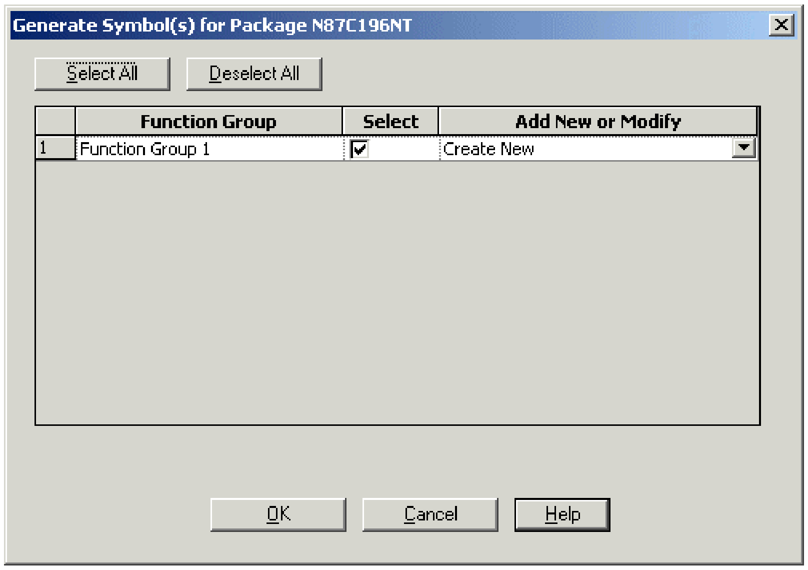
Task Overview

Create the symbol using the *Generate Symbol(s)* button in the *Package Pin* page of the Package Editor.

Steps

1. Click *Generate Symbol(s)*.

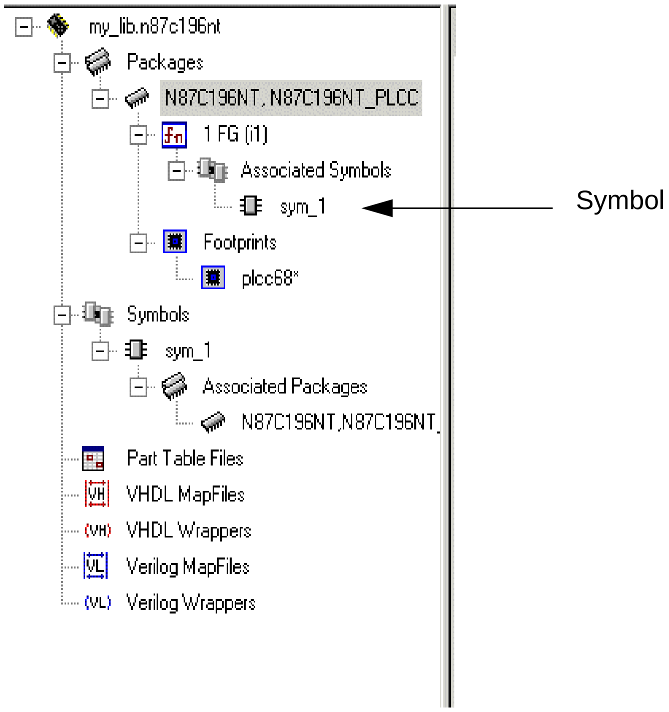
The Generate Symbol(s) for Package N87C196NT dialog box appears.



	Function Group	Select	Add New or Modify
1	Function Group 1	✓	Create New

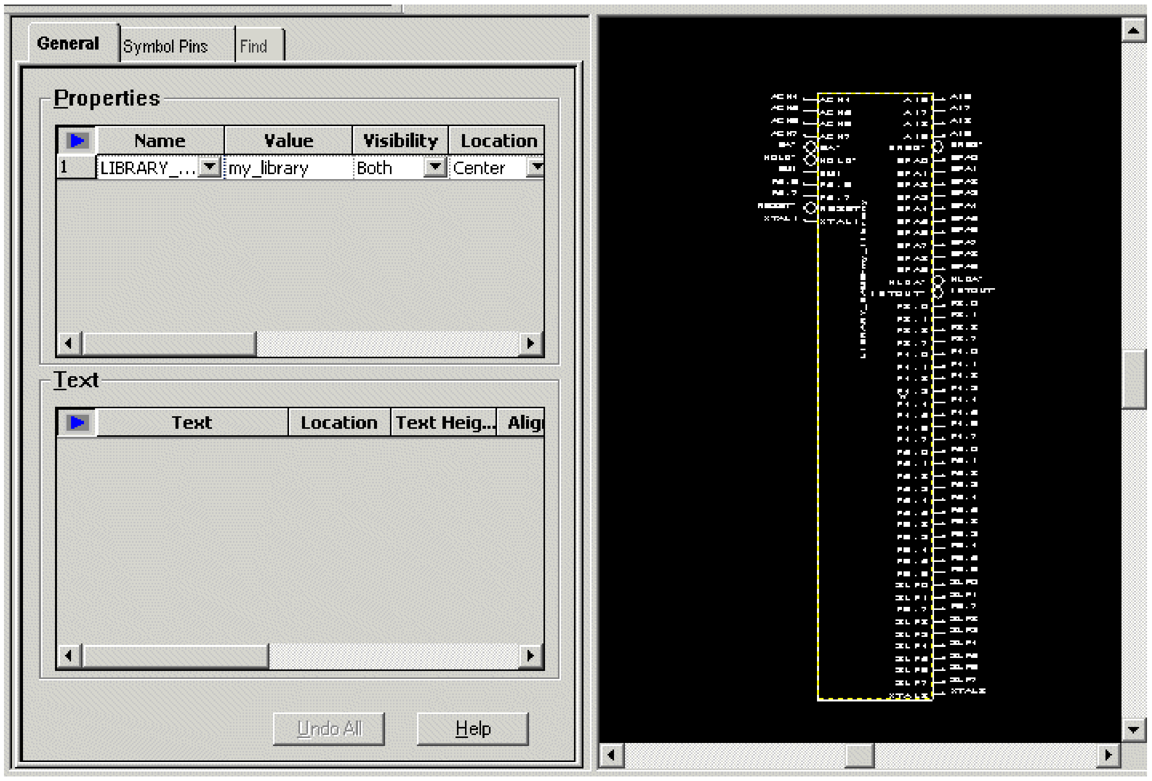
2. Click *OK*.

   The symbol is created and appears in the cell tree.



3. To view the symbol, click on *sym_1.*

The symbol details appear in the Symbol Pins panel and the symbol graphics are shown on the Symbol Editor canvas.



	Name	Value	Visibility	Location
1	LIBRARY_...	my_library	Both	Center

	Text	Location	Text Heig...	Alig

Summary

This completes the task of creating a single-slot flat part.

4

Creating Parts from CSV Files

Objective

To become familiar with the steps in creating parts by importing data from a CSV file.

In this chapter, you will:

- Understand the CSV file format from which data can be imported into Part Developer
- Import data from a CSV file

Overview

Part Developer can import part information stored in a comma-separated value (.csv) file and create packages and symbols from it.

The entries in the CSV file must be in the name-value pair format.

By default, the following header keywords are supported:

- package_name
- assertion_char
- assertion
- jedec_type
- pin_name
- pin_number
- pin_type
- pin_location
- pin_position
- load_setupfile

- symbol
- pin_shape
- diff_pair_pins_pos
- diff_pair_pins_neg

Important

The minimum of headers required are as follows:

- `pin_name` and `pin_number` for flat parts
- `pin_name`, `pin_number`, and `symbol` for multi-section parts

Important

If required, you can change the header keywords according to your specifications. See the *Configuring the Predefined Headers for CSV Import* section in the *Advanced Tasks* chapter in the *Part Developer User Guide*.

The CSV File Format

The package and symbol information is determined in the following way:

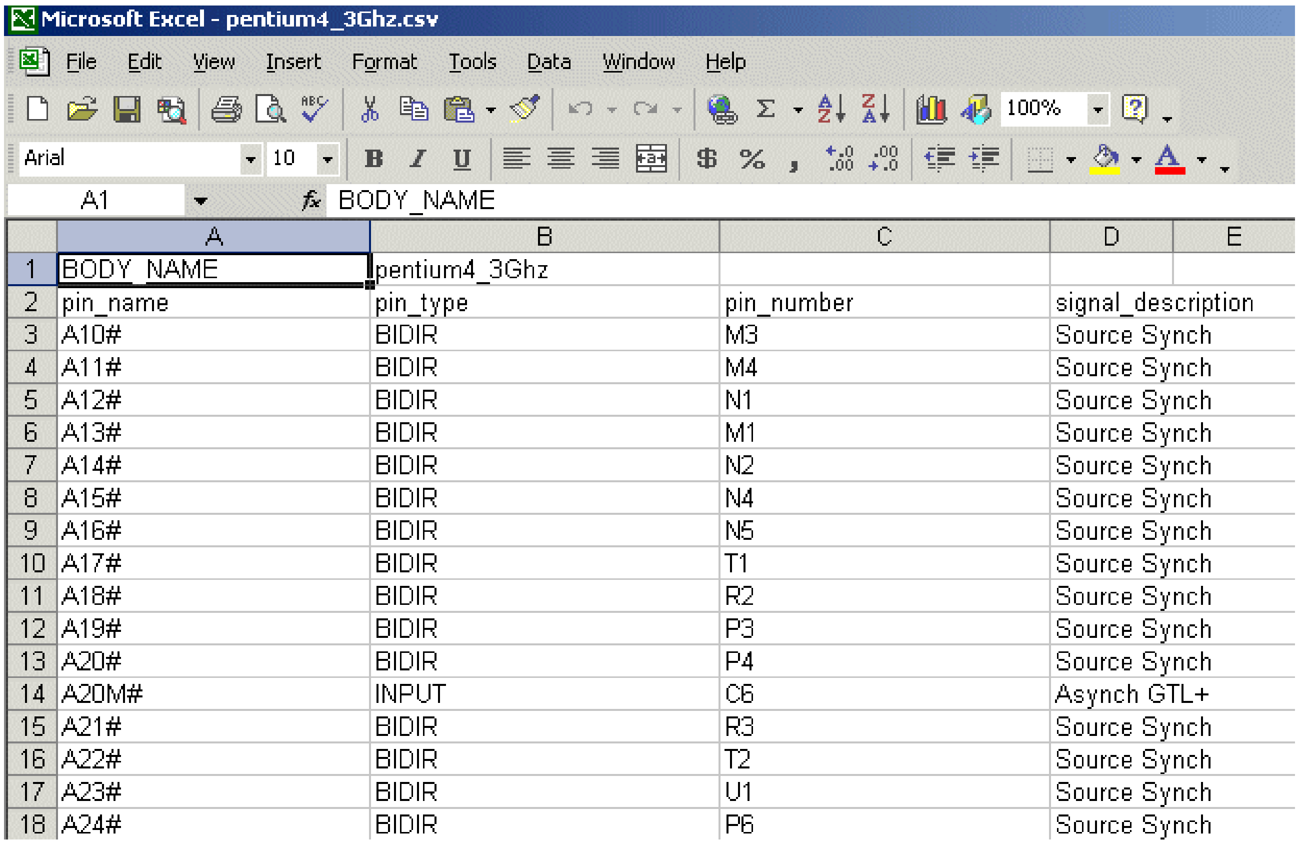
- The `package_name` entry is used to derive the name of the package. If this entry is missing, the cell name is used for the package name. You can create equivalent packages (aliases) by specifying comma-separated values, such as 7400_DIP, 7400_CCC, and so on.
- The `assertion_char` entry is used to determine which pins will be treated as low-asserted. If this entry is present, the values specified in the *Read/Write* and *Additonal Read* fields in *Setup* are ignored.
- The `assertion` entry is used to determine whether a pin is low-asserted or not. The values that will determine the low assertion are specified using the `Import_Csv_LowassertFlag` directive. By default, the values `L` and `Low` are specified.
- The `jedec_type` entry is used to determine the value of the JEDEC_TYPE property.
- The `load_setupfile` entry is used to determine the location of the project file from which to read the setup values. The setup values of the current project will be ignored.
- Entries under the `pin_name` column are used as pin names.

- Entries under the `pin_number` column are used as pin numbers.
- Entries under the `pin_type` column are used as pin types.
- Entries under the `output_load` column are used as the output load values for a pin type.
- Entries under the `pin_location` column are used to determine the pin location for specific pin types.
- Symbols are created only if the symbol entry is present in the header line that describes the pins.
- Entries under the `pin_position` field determines the position of the pin from the origin. This will appear as the value of the *Position* property in the Symbol Editor.
- All name-value pair entries before the `pin_number`, `pin_name`, `pin_type`, and `symbol` headings are imported as additional package properties.

Importing Data from a CSV File

The `pentium4_3Ghz.csv` file in the `<`*`your_inst_dir`*`>/doc/pdv_tut/tutorial_data/datasheets` location will be used in this chapter to demonstrate the steps involved in importing data from a CSV file.

A part of the CSV file is displayed below:


Microsoft Excel - pentium4_3Ghz.csv
File Edit View Insert Format Tools Data Window Help
100%
Arial 10
A1
BODY_NAME

	A	B	C	D	E
1	BODY_NAME	pentium4_3Ghz			
2	pin_name	pin_type	pin_number	signal_description	
3	A10#	BIDIR	M3	Source Synch	
4	A11#	BIDIR	M4	Source Synch	
5	A12#	BIDIR	N1	Source Synch	
6	A13#	BIDIR	M1	Source Synch	
7	A14#	BIDIR	N2	Source Synch	
8	A15#	BIDIR	N4	Source Synch	
9	A16#	BIDIR	N5	Source Synch	
10	A17#	BIDIR	T1	Source Synch	
11	A18#	BIDIR	R2	Source Synch	
12	A19#	BIDIR	P3	Source Synch	
13	A20#	BIDIR	P4	Source Synch	
14	A20M#	INPUT	C6	Asynch GTL+	
15	A21#	BIDIR	R3	Source Synch	
16	A22#	BIDIR	T2	Source Synch	
17	A23#	BIDIR	U1	Source Synch	
18	A24#	BIDIR	P6	Source Synch	

As displayed, the CSV file has the following fields:

- pin_name
- pin_type
- pin_number
- a pin property called signal_description

  Note: The `signal_description` property is not used by any downstream tools. It has been used in the tutorial to demonstrate some of the features of Part Developer.

You will now import the CSV file and create the part.

Task Overview

Import the pentium3_4GHz.csv file into Part Developer. The cell to be created is `pentium3_4GHz` in the `my_lib` folder.

Steps

1. Choose *File – Import and Export*.

   The Import and Export wizard appears.

2. Choose *Import Comma Separated Value (.csv) file* and click *Next*.

   The Select Source page appears.

3. Browse to the `<your_work_area>/tutorial_data/datasheets` location and open the `pentium4_3GHz.csv` file.

4. Click *Next*.

   The Select Destination page appears. The names of the cell and destination library are seeded by default. The name of the cell is the same as that of the CSV file.

5. Select `my_lib` as the destination library.

6. Click *Next*.

   The Preview of Import Data page appears. This page gives you a preview of the part that is being created from the CSV data.

7. Click *Finish*.

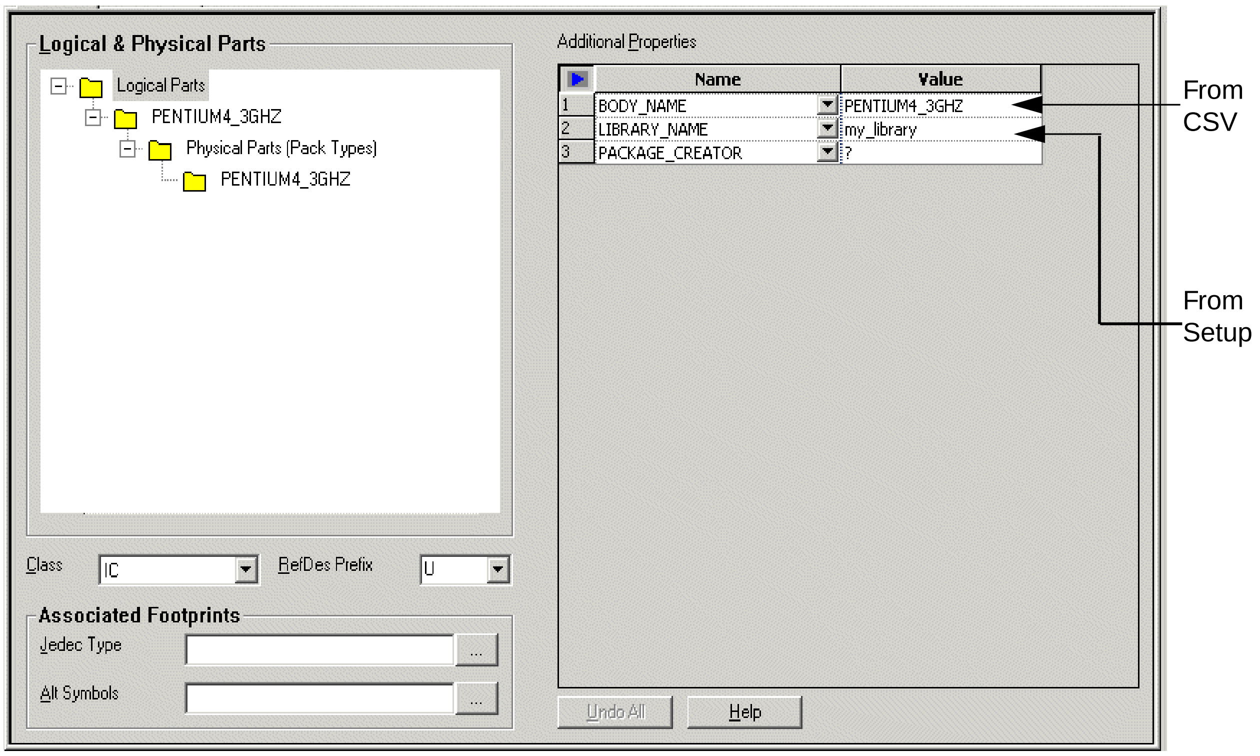
   The part is created and loaded in the Cell Editor.

8. Select the *PENTIUM4_3GHZ* package in the cell tree.

   The Package Editor loads the package.

9. Click on the *General* tab.

The General page appears.

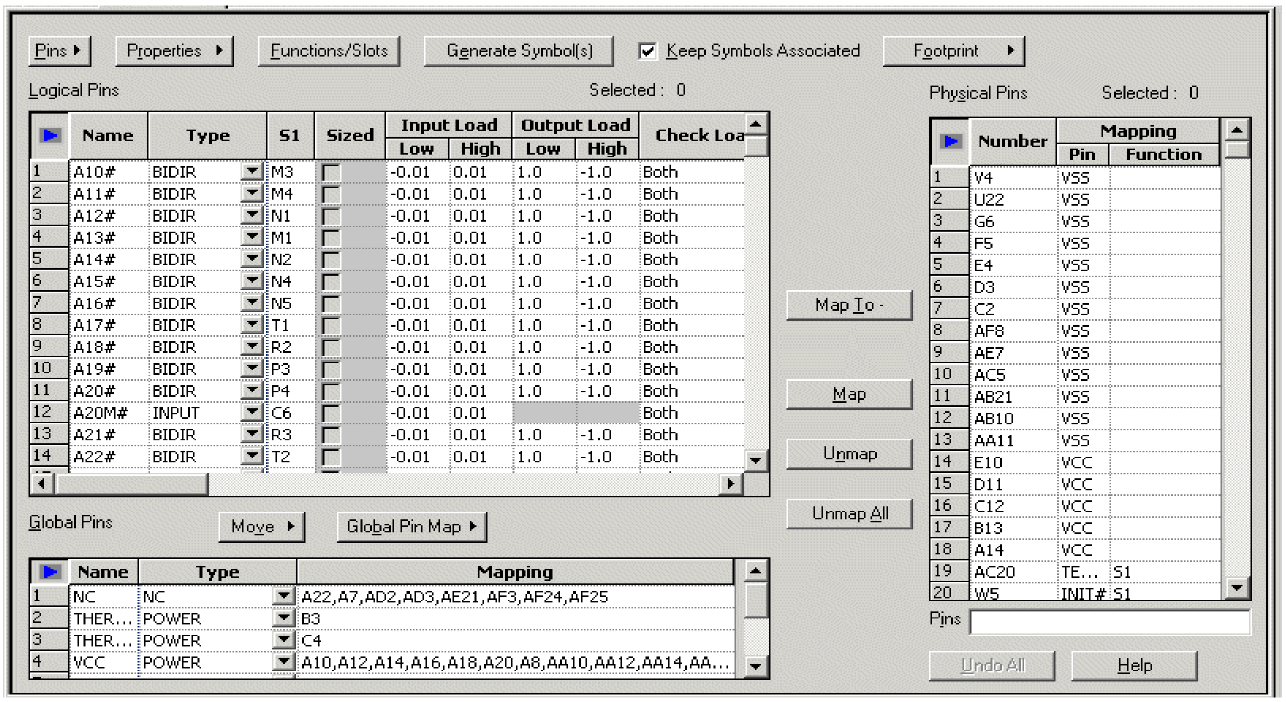


	Name	Value
1	BODY_NAME	PENTIUM4_3GHZ
2	LIBRARY_NAME	my_library
3	PACKAGE_CREATOR	?

Notice that the `BODY_NAME` entry in the CSV file appears as a package property. Also notice that the `LIBRARY_NAME` and `PACKAGE_CREATOR` properties, which were added in Setup in the previous chapter, also get added to the package.

10. Click on the *Package Pin* tab.

The Package Pin page appears.



Logical Pins Selected : 0

	Name	Type	S1	Sized	Input Load		Output Load		Check Loa
					Low	High	Low	High	
1	A10#	BIDIR	M3		-0.01	0.01	1.0	-1.0	Both
2	A11#	BIDIR	M4		-0.01	0.01	1.0	-1.0	Both
3	A12#	BIDIR	N1		-0.01	0.01	1.0	-1.0	Both
4	A13#	BIDIR	M1		-0.01	0.01	1.0	-1.0	Both
5	A14#	BIDIR	N2		-0.01	0.01	1.0	-1.0	Both
6	A15#	BIDIR	N4		-0.01	0.01	1.0	-1.0	Both
7	A16#	BIDIR	N5		-0.01	0.01	1.0	-1.0	Both
8	A17#	BIDIR	T1		-0.01	0.01	1.0	-1.0	Both
9	A18#	BIDIR	R2		-0.01	0.01	1.0	-1.0	Both
10	A19#	BIDIR	P3		-0.01	0.01	1.0	-1.0	Both
11	A20#	BIDIR	P4		-0.01	0.01	1.0	-1.0	Both
12	A20M#	INPUT	C6		-0.01	0.01			Both
13	A21#	BIDIR	R3		-0.01	0.01	1.0	-1.0	Both
14	A22#	BIDIR	T2		-0.01	0.01	1.0	-1.0	Both

Global Pins

	Name	Type	Mapping
1	NC	NC	A22,A7,AD2,AD3,AE21,AF3,AF24,AF25
2	THER...	POWER	B3
3	THER...	POWER	C4
4	VCC	POWER	A10,A12,A14,A16,A18,A20,A8,AA10,AA12,AA14,AA...

Physical Pins Selected : 0

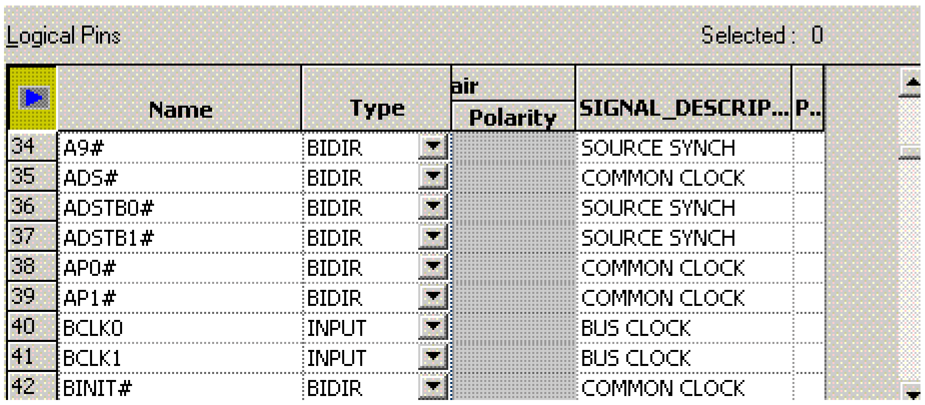
	Number	Mapping	
		Pin	Function
1	V4	VSS	
2	U22	VSS	
3	G6	VSS	
4	F5	VSS	
5	E4	VSS	
6	D3	VSS	
7	C2	VSS	
8	AF8	VSS	
9	AE7	VSS	
10	AC5	VSS	
11	AB21	VSS	
12	AB10	VSS	
13	AA11	VSS	
14	E10	VCC	
15	D11	VCC	
16	C12	VCC	
17	B13	VCC	
18	A14	VCC	
19	AC20	TE...	S1
20	W5	INIT#	S1

Note that the pin names, types, and mappings are added as specified in the CSV file.

Next, you will ensure that the `SIGNAL_DESCRIPTION` pin property has also been added.

11. Right-click anywhere in the *Logical Pins* grid and select *Hide Load Cols*.

The load columns get hidden, and you can see the SIGNAL_DESCRIPTION property.



Logical Pins Selected : 0

	Name	Type	air	SIGNAL_DESCRIP...	P..
			Polarity		
34	A9#	BIDIR		SOURCE SYNCH	
35	ADS#	BIDIR		COMMON CLOCK	
36	ADSTB0#	BIDIR		SOURCE SYNCH	
37	ADSTB1#	BIDIR		SOURCE SYNCH	
38	AP0#	BIDIR		COMMON CLOCK	
39	AP1#	BIDIR		COMMON CLOCK	
40	BCLK0	INPUT		BUS CLOCK	
41	BCLK1	INPUT		BUS CLOCK	
42	BINIT#	BIDIR		COMMON CLOCK	

12. Choose *File – Save* to save the part.

Summary

In this chapter, you learned about importing data from a CSV file to create a part.

5

Creating Split Parts

Objective

To become familiar with the steps in creating split parts.

In this chapter, you will learn:

- About the methodology for creating split parts
- How to create a split part by adding multiple slots
- How to create symbols for each slot

Overview

Parts are typically split for reasons such as:

- To reflect the way part functionality is used in a schematic.
- To display large pin-count parts better.

Methodology for Creating Split Parts

The following methodology should be followed for creating split parts:

1. Decide whether to use the SPLIT_INST and $LOCATION properties or the SPLIT_INST_NAME property for the split part.
2. Create the package with a single slot.
3. Do the logical-to-physical pin mapping for the first slot.
4. Create the necessary slots.
5. Distribute the pins across the slots.

The advantages of the above method are:

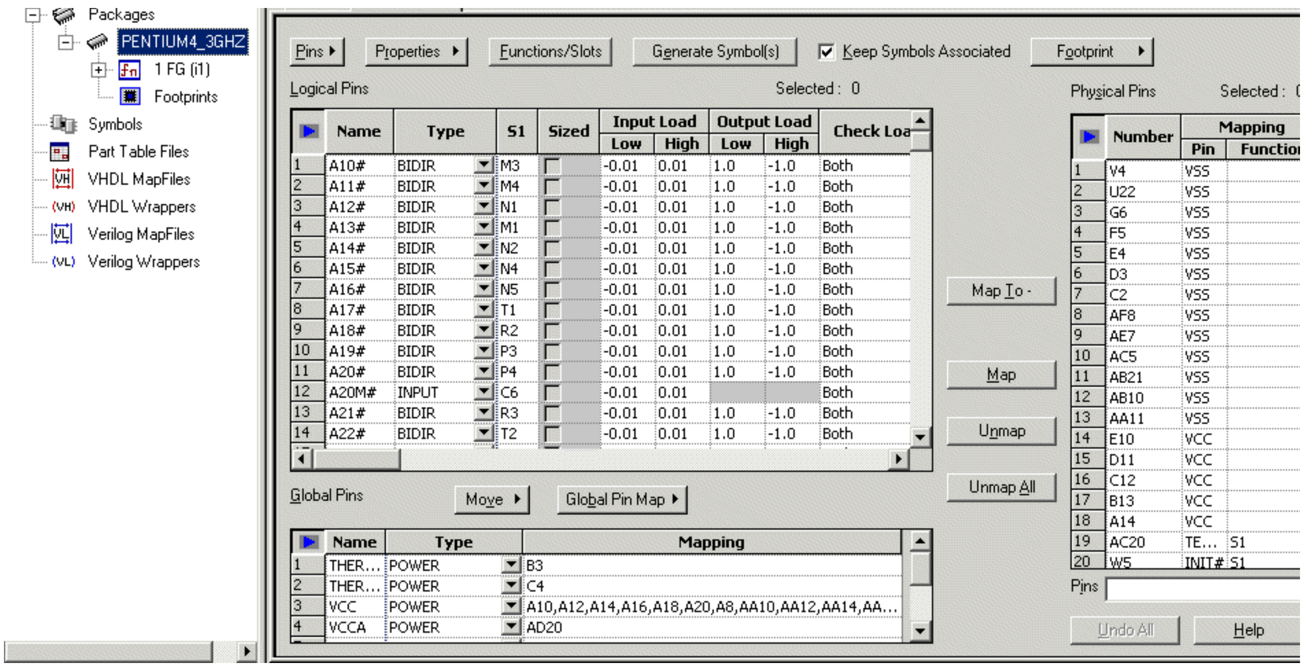
- It is easy to do mappings for a single slot.
- After distributing the pins, the slots that are left unmapped are automatically marked as -. This results in the pin number 0 getting added in `chips.prt` to the slots where the logical pin is not present.

Task Overview

Split the `pentium4_3GHz _for_splitting` part in the `my_lib` library into four parts and create symbols for each part. Each slot should have 50 pins each.

Steps

1. Open the `pentium4_3GHz _for_splitting` part from `my_lib`.
2. Select the *PENTIUM4_3GHZ* package.



Logical Pins Selected : 0

	Name	Type	S1	Sized	Input Load		Output Load		Check Loa
					Low	High	Low	High	
1	A10#	BIDIR	M3	☐	-0.01	0.01	1.0	-1.0	Both
2	A11#	BIDIR	M4	☐	-0.01	0.01	1.0	-1.0	Both
3	A12#	BIDIR	N1	☐	-0.01	0.01	1.0	-1.0	Both
4	A13#	BIDIR	M1	☐	-0.01	0.01	1.0	-1.0	Both
5	A14#	BIDIR	N2	☐	-0.01	0.01	1.0	-1.0	Both
6	A15#	BIDIR	N4	☐	-0.01	0.01	1.0	-1.0	Both
7	A16#	BIDIR	N5	☐	-0.01	0.01	1.0	-1.0	Both
8	A17#	BIDIR	T1	☐	-0.01	0.01	1.0	-1.0	Both
9	A18#	BIDIR	R2	☐	-0.01	0.01	1.0	-1.0	Both
10	A19#	BIDIR	P3	☐	-0.01	0.01	1.0	-1.0	Both
11	A20#	BIDIR	P4	☐	-0.01	0.01	1.0	-1.0	Both
12	A20M#	INPUT	C6	☐	-0.01	0.01			Both
13	A21#	BIDIR	R3	☐	-0.01	0.01	1.0	-1.0	Both
14	A22#	BIDIR	T2	☐	-0.01	0.01	1.0	-1.0	Both

Global Pins

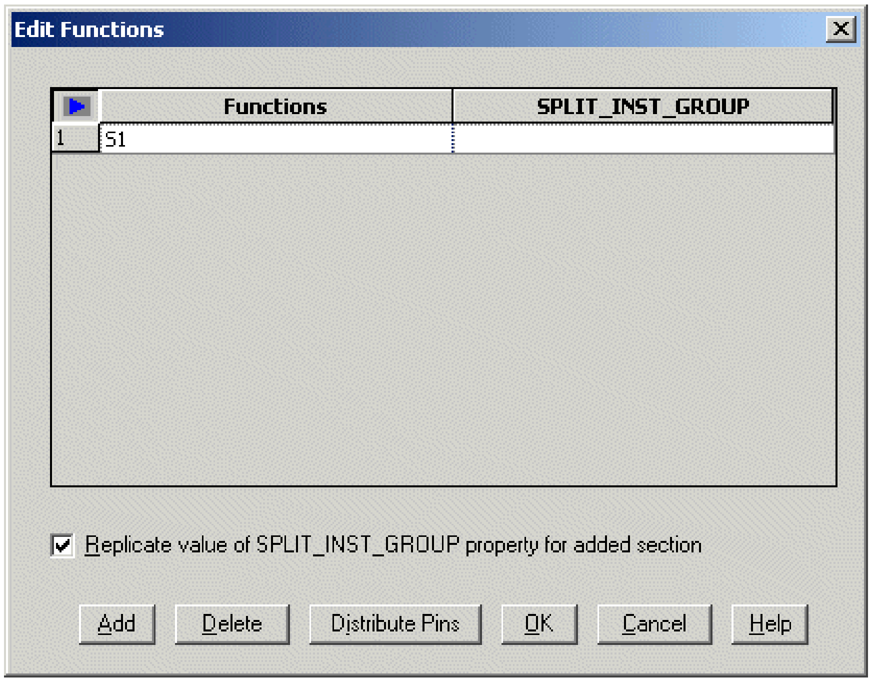
	Name	Type	Mapping
1	THER…	POWER	B3
2	THER…	POWER	C4
3	VCC	POWER	A10,A12,A14,A16,A18,A20,A8,AA10,AA12,AA14,AA…
4	VCCA	POWER	AD20

Physical Pins Selected :

	Number	Mapping	
		Pin	Functio
1	V4	VSS	
2	U22	VSS	
3	G6	VSS	
4	F5	VSS	
5	E4	VSS	
6	D3	VSS	
7	C2	VSS	
8	AF8	VSS	
9	AE7	VSS	
10	AC5	VSS	
11	AB21	VSS	
12	AB10	VSS	
13	AA11	VSS	
14	E10	VCC	
15	D11	VCC	
16	C12	VCC	
17	B13	VCC	
18	A14	VCC	
19	AC20	TE…	S1
20	W5	INIT#	S1

3. Click *Functions/Slots*.

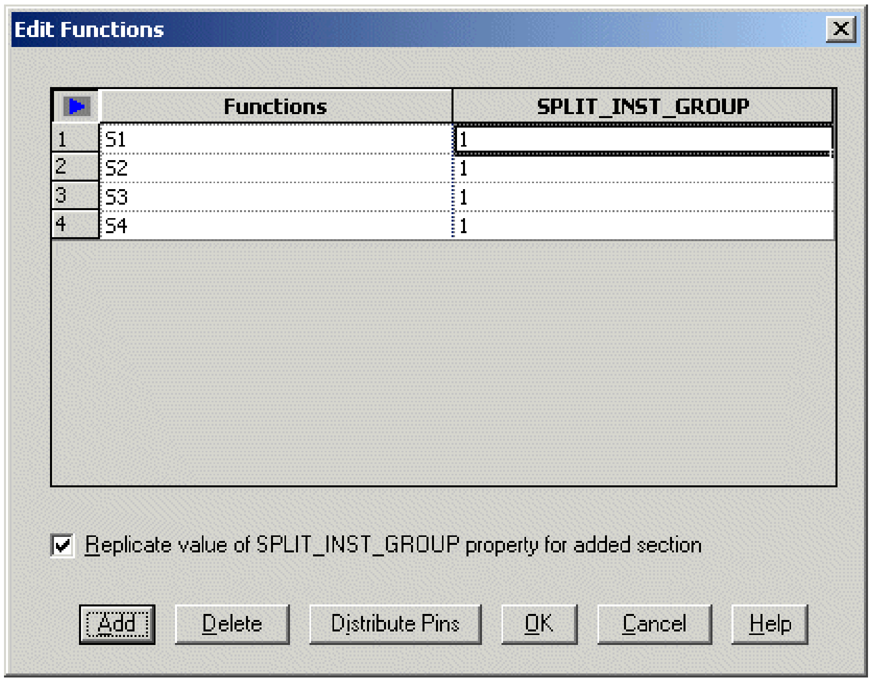
The Edit Functions dialog box appears.



4. Enter 1 in the *SPLIT_INST_GROUP* field. The SPLIT_INST_GROUP property is used for split parts. The value enables Part Developer to determine which slots of a split part combine to form one logical group.
5. To add more slots, click *Add*.

   The Specify the number of slots dialog box appears.
6. To create three more slots, type 3 in the *Slot Count* field and click *OK*.

The four slots are created.



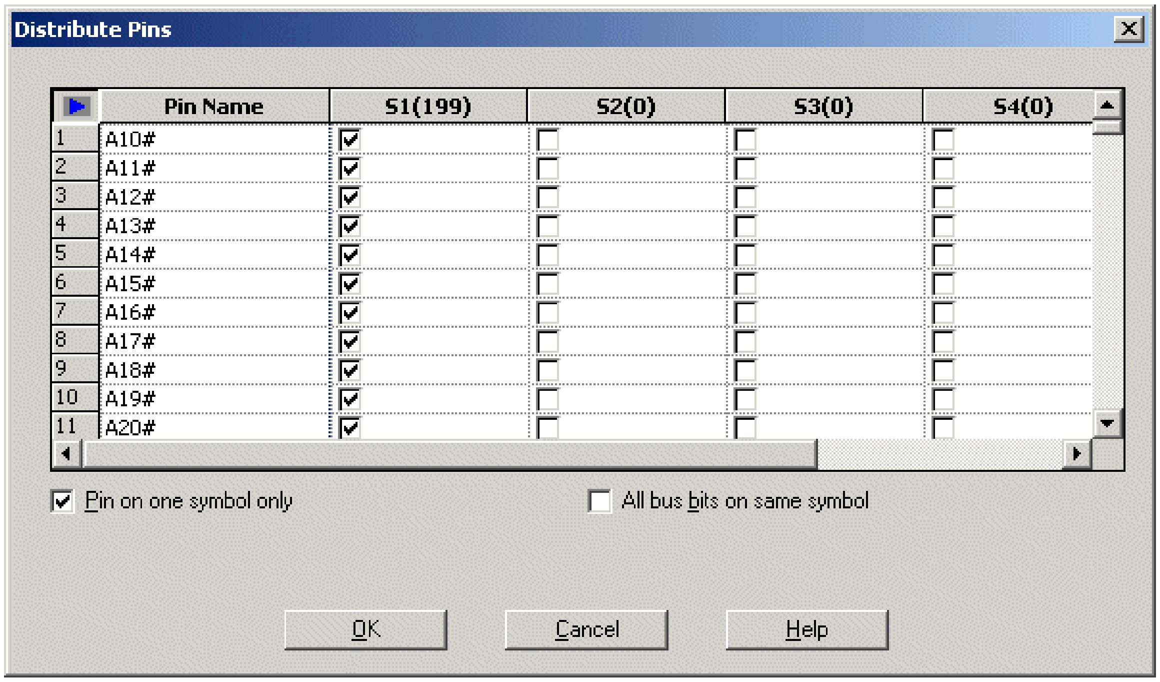
	Functions	SPLIT_INST_GROUP
1	S1	1
2	S2	1
3	S3	1
4	S4	1

☑ Replicate value of SPLIT_INST_GROUP property for added section

Next, you need to distribute the pins across the four slots.

7. Click *Distribute Pins*.

The Distribute Pins dialog box appears.



Distribute Pins

	Pin Name	S1(199)	S2(0)	S3(0)	S4(0)
1	A10#	✓	☐	☐	☐
2	A11#	✓	☐	☐	☐
3	A12#	✓	☐	☐	☐
4	A13#	✓	☐	☐	☐
5	A14#	✓	☐	☐	☐
6	A15#	✓	☐	☐	☐
7	A16#	✓	☐	☐	☐
8	A17#	✓	☐	☐	☐
9	A18#	✓	☐	☐	☐
10	A19#	✓	☐	☐	☐
11	A20#	✓	☐	☐	☐

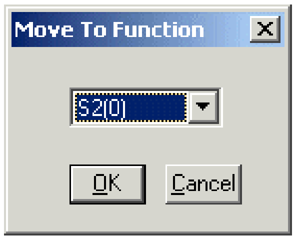
[✓] Pin on one symbol only
[ ] All bus bits on same symbol

By default, all the 199 logical pins are present in the first slot. You need to distribute these pins across the four slots.

8. Select the cells 51-100 under S1.

9. Right-click on the selection and choose *Move To*.

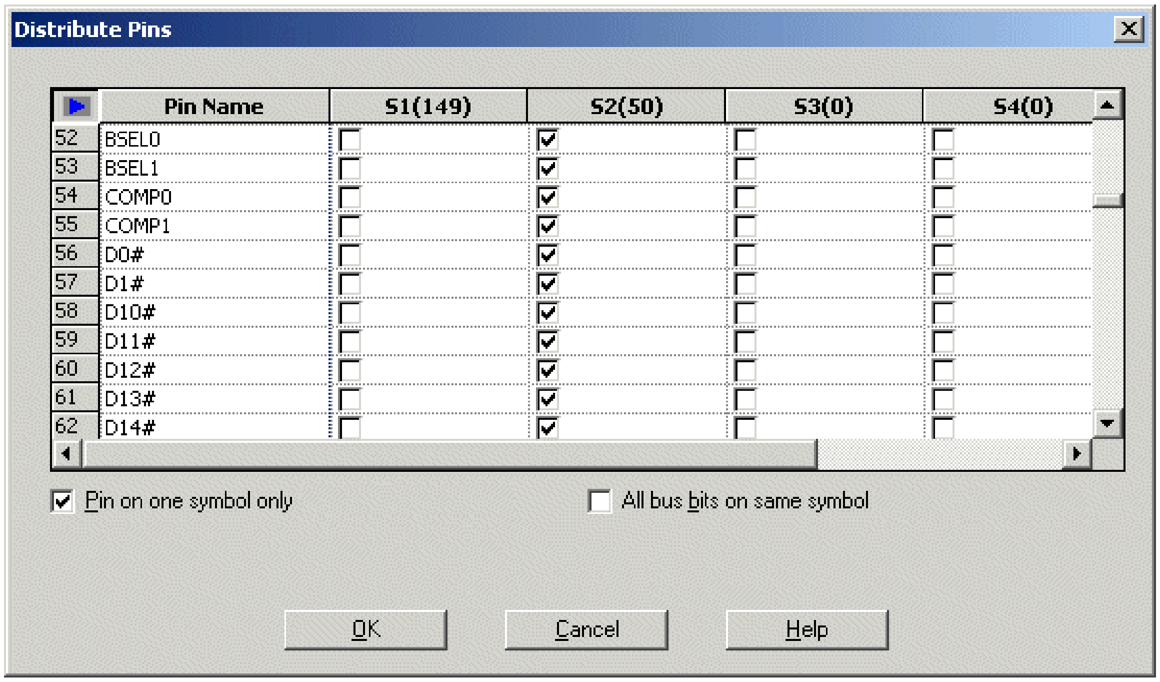
The Move To Function dialog box appears.



This dialog box displays all the slots other than the slot from which it has been called.

10. To move the pins to the second slot, click *OK*.

The selected pins are moved to the second slot.

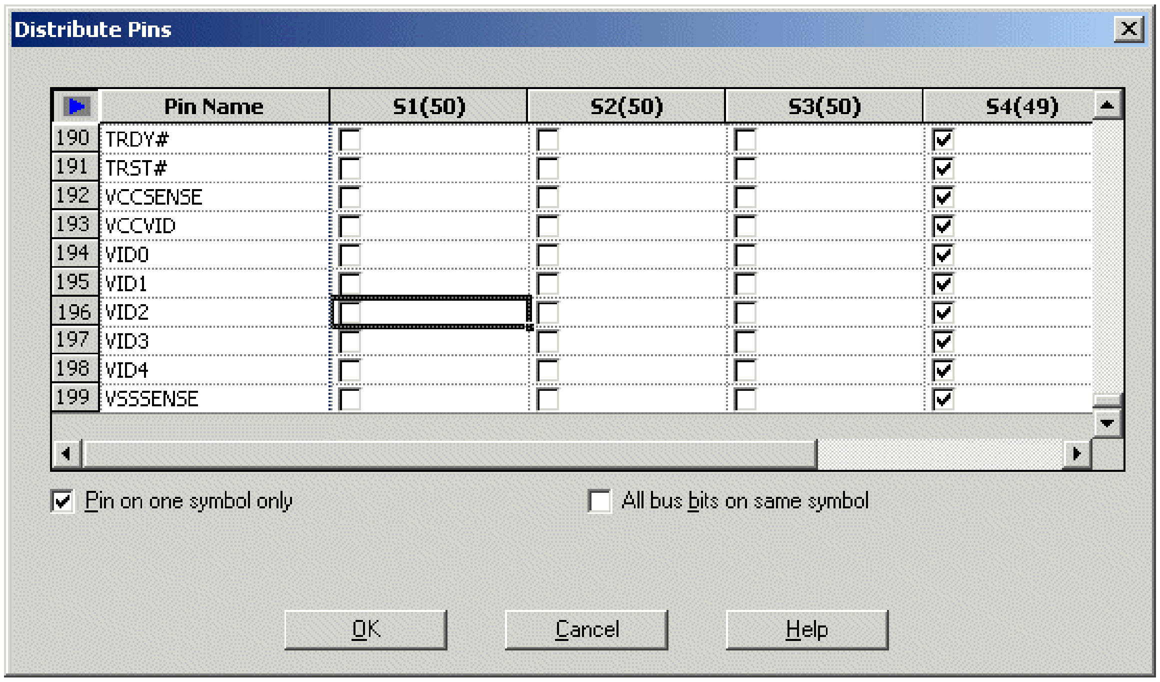


	Pin Name	S1(149)	S2(50)	S3(0)	S4(0)
52	BSEL0	☐	✓	☐	☐
53	BSEL1	☐	✓	☐	☐
54	COMP0	☐	✓	☐	☐
55	COMP1	☐	✓	☐	☐
56	D0#	☐	✓	☐	☐
57	D1#	☐	✓	☐	☐
58	D10#	☐	✓	☐	☐
59	D11#	☐	✓	☐	☐
60	D12#	☐	✓	☐	☐
61	D13#	☐	✓	☐	☐
62	D14#	☐	✓	☐	☐

✓ Pin on one symbol only ☐ All bus bits on same symbol

11. Similarly, move pins 101-150 and pins 151-199 to the third and fourth slots, respectively.

After all the pins are distributed, the Distribute Pins dialog box should appear as follows:

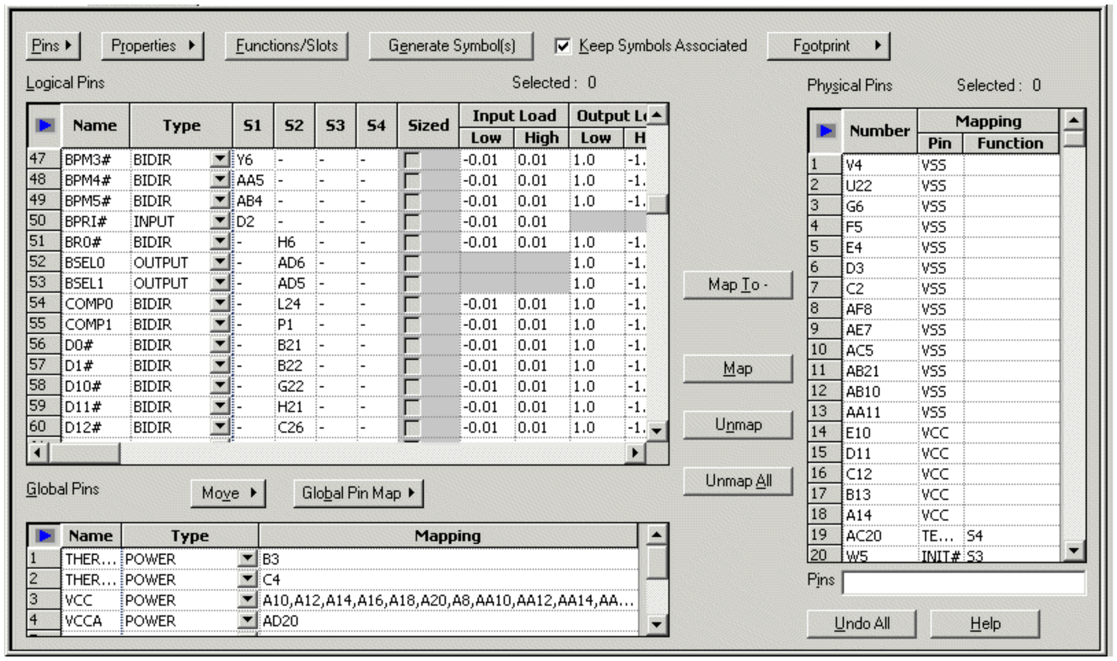


	Pin Name	S1(50)	S2(50)	S3(50)	S4(49)
190	TRDY#	☐	☐	☐	✓
191	TRST#	☐	☐	☐	✓
192	VCCSENSE	☐	☐	☐	✓
193	VCCVID	☐	☐	☐	✓
194	VID0	☐	☐	☐	✓
195	VID1	☐	☐	☐	✓
196	VID2	☐	☐	☐	✓
197	VID3	☐	☐	☐	✓
198	VID4	☐	☐	☐	✓
199	VSSSENSE	☐	☐	☐	✓

✓ Pin on one symbol only ☐ All bus bits on same symbol

12. Click *OK*.

The pins are distributed across the four slots. The slots in which a logical pin is not present is mapped to `-`.



	Name	Type	S1	S2	S3	S4	Sized	Input Load		Output Lo	
								Low	High	Low	H
47	BPM3#	BIDIR	Y6	-	-	-	☐	-0.01	0.01	1.0	-1.
48	BPM4#	BIDIR	AA5	-	-	-	☐	-0.01	0.01	1.0	-1.
49	BPM5#	BIDIR	AB4	-	-	-	☐	-0.01	0.01	1.0	-1.
50	BPRI#	INPUT	D2	-	-	-	☐	-0.01	0.01		
51	BR0#	BIDIR	-	H6	-	-	☐	-0.01	0.01	1.0	-1.
52	BSEL0	OUTPUT	-	AD6	-	-	☐			1.0	-1.
53	BSEL1	OUTPUT	-	AD5	-	-	☐			1.0	-1.
54	COMP0	BIDIR	-	L24	-	-	☐	-0.01	0.01	1.0	-1.
55	COMP1	BIDIR	-	P1	-	-	☐	-0.01	0.01	1.0	-1.
56	D0#	BIDIR	-	B21	-	-	☐	-0.01	0.01	1.0	-1.
57	D1#	BIDIR	-	B22	-	-	☐	-0.01	0.01	1.0	-1.
58	D10#	BIDIR	-	G22	-	-	☐	-0.01	0.01	1.0	-1.
59	D11#	BIDIR	-	H21	-	-	☐	-0.01	0.01	1.0	-1.
60	D12#	BIDIR	-	C26	-	-	☐	-0.01	0.01	1.0	-1.

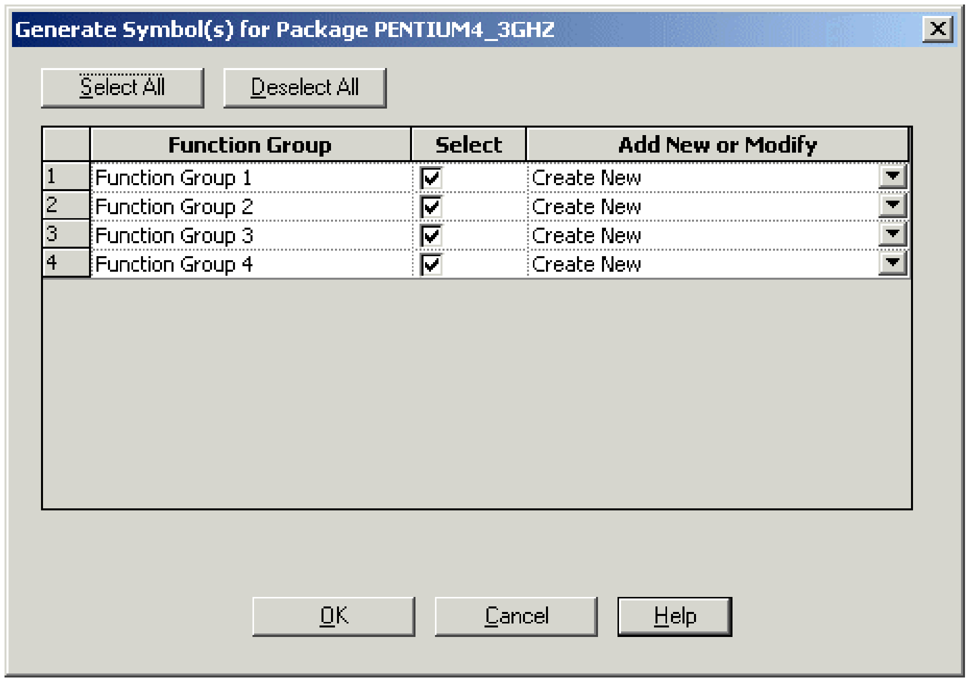
	Name	Type	Mapping
1	THER…	POWER	B3
2	THER…	POWER	C4
3	VCC	POWER	A10,A12,A14,A16,A18,A20,A8,AA10,AA12,AA14,AA…
4	VCCA	POWER	AD20

	Number	Mapping	
		Pin	Function
1	V4	VSS	
2	U22	VSS	
3	G6	VSS	
4	F5	VSS	
5	E4	VSS	
6	D3	VSS	
7	C2	VSS	
8	AF8	VSS	
9	AE7	VSS	
10	AC5	VSS	
11	AB21	VSS	
12	AB10	VSS	
13	AA11	VSS	
14	E10	VCC	
15	D11	VCC	
16	C12	VCC	
17	B13	VCC	
18	A14	VCC	
19	AC20	TE…	S4
20	W5	INIT#	S3

Next, you will create the symbols for the four slots.

13. Click *Generate Symbol(s)*.

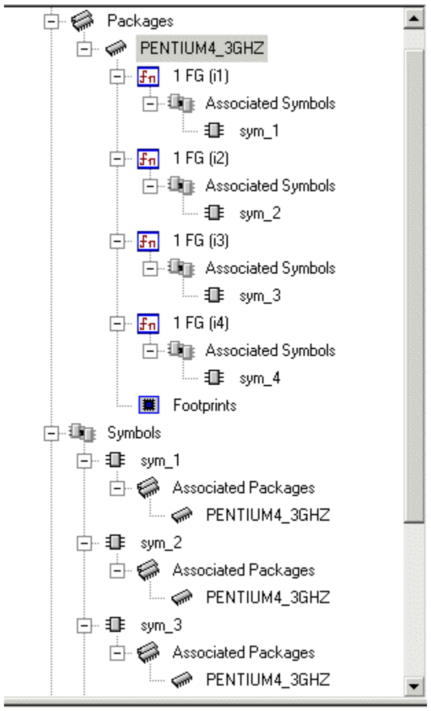
The Generate Symbol(s) for Package PENTIUM4_3GHZ dialog box appears.



	Function Group	Select	Add New or Modify
1	Function Group 1	✓	Create New
2	Function Group 2	✓	Create New
3	Function Group 3	✓	Create New
4	Function Group 4	✓	Create New

14. To create symbols for all functions, click *OK*.

The symbols are created for all the function groups.



15. Choose *File – Save* to save the part.

Summary

In this chapter, you learned how to create split parts.

6

Creating Parts from PDFs

Objective

To become familiar with the steps involved in creating parts from PDFs.

In this chapter, you will learn to:

- Enter pin information directly from a PDF file to the *Logical Pins* grid in the Package Editor.
- Use a spreadsheet to manipulate pin information and then use it to create parts.

Overview

Part Developer helps you create parts from datasheets available in the PDF format. From the datasheet, you can do a text-copy of pin information and paste directly into the *Logical Pins* grid and the *Physical Pins* grid.

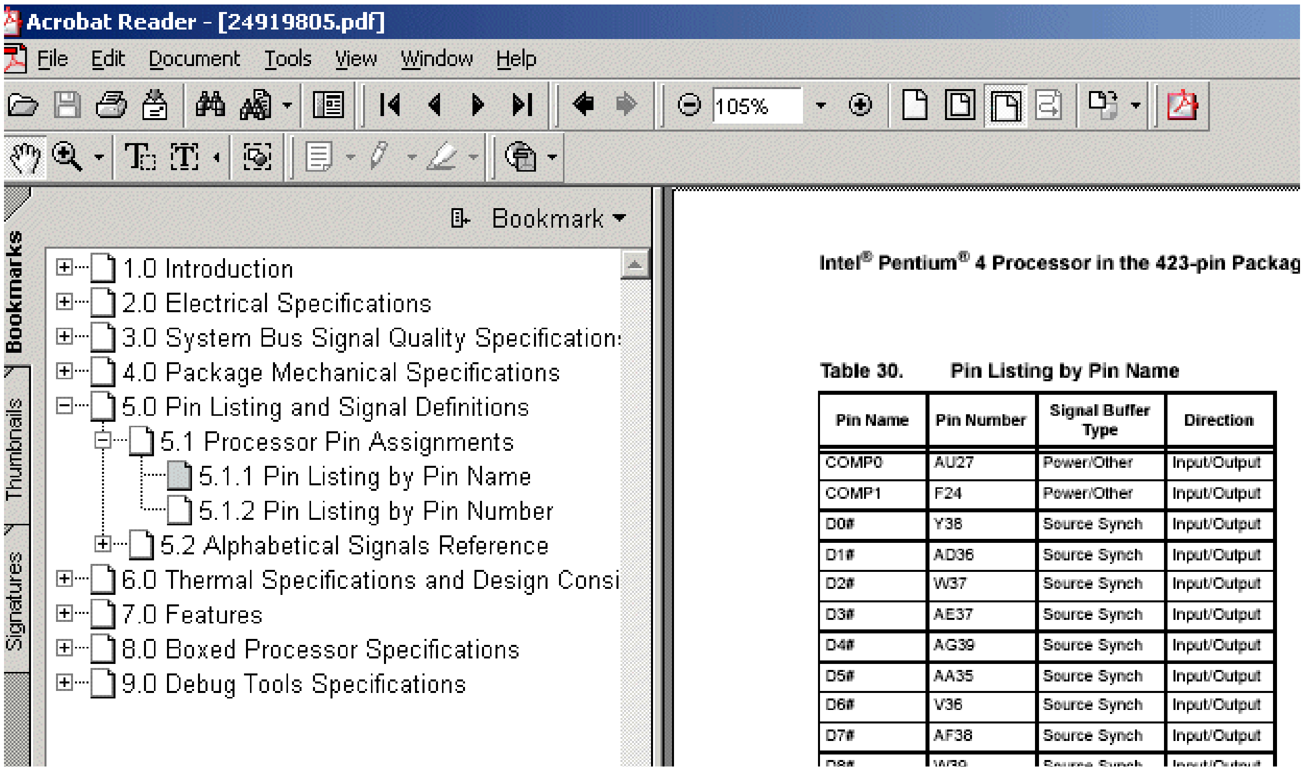
Creating Parts from PDFs

Task Overview

Create the part from the Pentium datasheet (`24919805.pdf`) located in `<`*`your_inst_dir`*`>/doc/pdv_tut/tutorial_data/datasheets`.

Pentium Datasheet

Given below is the relevant portion from the datasheet of the Pentium processor. The pin list is displayed partially.


Acrobat Reader - [24919805.pdf]
File Edit Document Tools View Window Help
105%
Bookmarks
Thumbnails
Signatures
Bookmark

- 1.0 Introduction
- 2.0 Electrical Specifications
- 3.0 System Bus Signal Quality Specification:
- 4.0 Package Mechanical Specifications
- 5.0 Pin Listing and Signal Definitions
  - 5.1 Processor Pin Assignments
    - 5.1.1 Pin Listing by Pin Name
    - 5.1.2 Pin Listing by Pin Number
  - 5.2 Alphabetical Signals Reference
- 6.0 Thermal Specifications and Design Consi
- 7.0 Features
- 8.0 Boxed Processor Specifications
- 9.0 Debug Tools Specifications

Intel® Pentium® 4 Processor in the 423-pin Packag

Table 30. Pin Listing by Pin Name

Pin Name	Pin Number	Signal Buffer Type	Direction
COMP0	AU27	Power/Other	Input/Output
COMP1	F24	Power/Other	Input/Output
D0#	Y38	Source Synch	Input/Output
D1#	AD36	Source Synch	Input/Output
D2#	W37	Source Synch	Input/Output
D3#	AE37	Source Synch	Input/Output
D4#	AG39	Source Synch	Input/Output
D5#	AA35	Source Synch	Input/Output
D6#	V36	Source Synch	Input/Output
D7#	AF38	Source Synch	Input/Output

Steps

There are two ways in which you can enter pin information into Part Developer:

- Directly into Part Developer

  This method requires you to individually copy the *Pin Name* and *Pin Number* columns in Part Developer and then manually update the pin type information.

- Copying into Excel/Star Office and then copying the information into Part Developer

  This method provides the benefit of copying the entire pin name, type, and mapping information into Part Developer in a single step. The pin type information is also updated automatically.

Tip

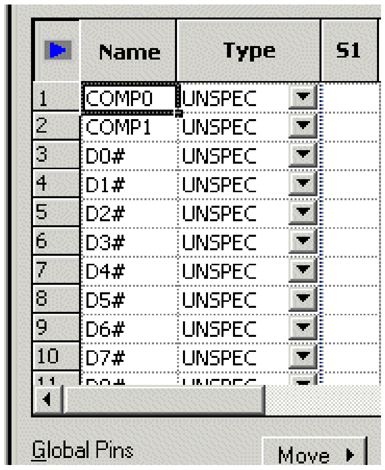
Acrobat Reader 5.1 should be used to read the PDF files. It has a *Column Select* option, which ensures pin names that are copied from the PDF are pasted as individual pins in the pin grid. Copying pin information from earlier versions of

Acrobat Reader results in all the pin names appearing as a single pin name in the *Logical Pins* grid. To fix this, use the *Edit – Paste Special(Grid) – Convert Whitespaces to NewLine* option when pasting data into Part Developer.

Directly into Part Developer

1. Open the datasheet in Acrobat Reader.
2. Select the *Column Select Tool* option.
3. Select the *Pin Name* column and press Ctrl + C.
4. Launch Part Developer and create a new part and package.
5. Go to the *Package Pin* page of the Package Editor.
6. Press Ctrl + I to insert a new row in the *Logical Pins* grid.
7. Select the empty cell under the *Name* column and press Ctrl + V to paste the pin names into the *Logical Pins* grid.

   The pin names appear under the *Name* column.

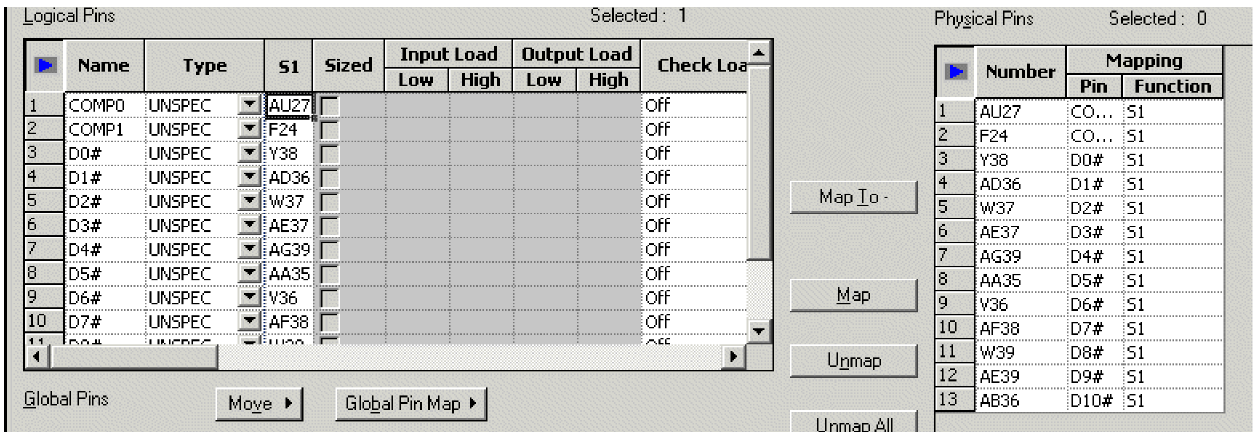


	Name	Type	S1
1	COMP0	UNSPEC	
2	COMP1	UNSPEC	
3	D0#	UNSPEC	
4	D1#	UNSPEC	
5	D2#	UNSPEC	
6	D3#	UNSPEC	
7	D4#	UNSPEC	
8	D5#	UNSPEC	
9	D6#	UNSPEC	
10	D7#	UNSPEC	

   Next, you need to copy the pin numbers that are mapped to the pin names.

8. Select the *Pin Number* column in the datasheet and press Ctrl + C.
9. Select the first cell under the *S1* column and press Ctrl + V.

The physical pin numbers are copied into the *Logical Pins* grid. The *Physical Pins* grid is also updated automatically with the pin-mapping information.



Logical Pins   Selected : 1

	Name	Type	S1	Sized	Input Load		Output Load		Check Loa
					Low	High	Low	High	
1	COMP0	UNSPEC	AU27	☐					Off
2	COMP1	UNSPEC	F24	☐					Off
3	D0#	UNSPEC	Y38	☐					Off
4	D1#	UNSPEC	AD36	☐					Off
5	D2#	UNSPEC	W37	☐					Off
6	D3#	UNSPEC	AE37	☐					Off
7	D4#	UNSPEC	AG39	☐					Off
8	D5#	UNSPEC	AA35	☐					Off
9	D6#	UNSPEC	V36	☐					Off
10	D7#	UNSPEC	AF38	☐					Off

Physical Pins   Selected : 0

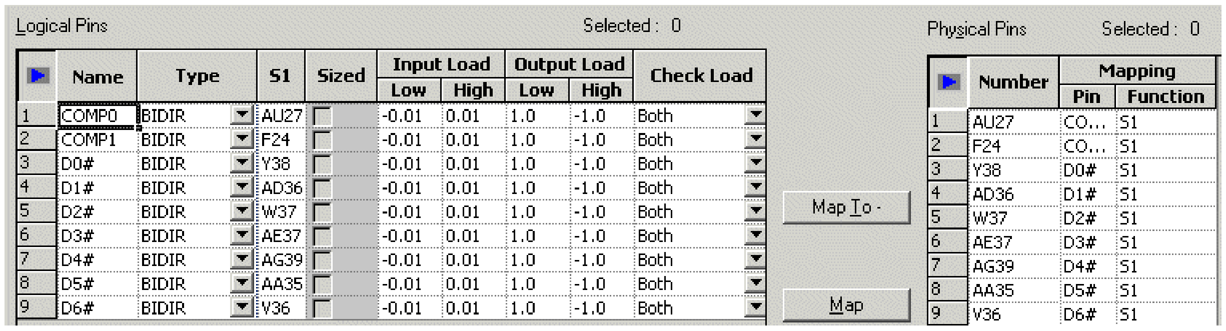
	Number	Mapping	
		Pin	Function
1	AU27	CO...	S1
2	F24	CO...	S1
3	Y38	D0#	S1
4	AD36	D1#	S1
5	W37	D2#	S1
6	AE37	D3#	S1
7	AG39	D4#	S1
8	AA35	D5#	S1
9	V36	D6#	S1
10	AF38	D7#	S1
11	W39	D8#	S1
12	AE39	D9#	S1
13	AB36	D10#	S1

Next, you need to copy the direction information from the PDF into the Type column to determine the pin types. In case the direction is missing in the PDF, you will need to manually determine the pin type in Part Developer.

10. Select the *Direction* column in the datasheet and press Ctrl + C.

11. Select the first cell under the *Type* column and press Ctrl + V.

In the datasheet, the direction of the pins is specified as `Input/Output`. However, on copying, the pin type is changed to `BIDIR`. This automatic translation is handled through the entries in the `propfile.prop` file located at `<`*`your_inst_dir`*`>/share/cdssetup/LMAN`. For more information, see the *Advanced Tasks* chapter in *Part Developer User Guide.*



Logical Pins   Selected : 0

	Name	Type	S1	Sized	Input Load		Output Load		Check Load
					Low	High	Low	High	
1	COMP0	BIDIR	AU27	☐	-0.01	0.01	1.0	-1.0	Both
2	COMP1	BIDIR	F24	☐	-0.01	0.01	1.0	-1.0	Both
3	D0#	BIDIR	Y38	☐	-0.01	0.01	1.0	-1.0	Both
4	D1#	BIDIR	AD36	☐	-0.01	0.01	1.0	-1.0	Both
5	D2#	BIDIR	W37	☐	-0.01	0.01	1.0	-1.0	Both
6	D3#	BIDIR	AE37	☐	-0.01	0.01	1.0	-1.0	Both
7	D4#	BIDIR	AG39	☐	-0.01	0.01	1.0	-1.0	Both
8	D5#	BIDIR	AA35	☐	-0.01	0.01	1.0	-1.0	Both
9	D6#	BIDIR	V36	☐	-0.01	0.01	1.0	-1.0	Both

Physical Pins   Selected : 0

	Number	Mapping	
		Pin	Function
1	AU27	CO...	S1
2	F24	CO...	S1
3	Y38	D0#	S1
4	AD36	D1#	S1
5	W37	D2#	S1
6	AE37	D3#	S1
7	AG39	D4#	S1
8	AA35	D5#	S1
9	V36	D6#	S1

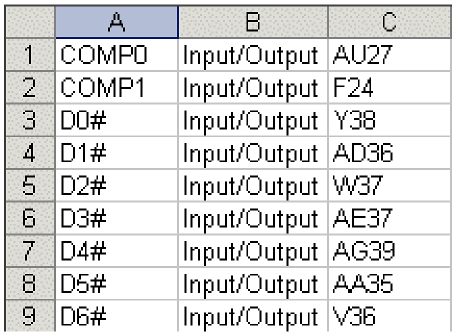
Copying First into Excel/Star Office and then into Part Developer

1. Open the datasheet in Acrobat Reader.

2. Select the *Column Select Tool* option.

3. Select the *Pin Name* column and press Ctrl + C.
4. Open an Excel or Star Office spreadsheet and press Ctrl + V to paste the data in the first column.
5. Next, copy the *Direction* and *Pin Number* columns and paste into the columns adjacent to the pin names column.

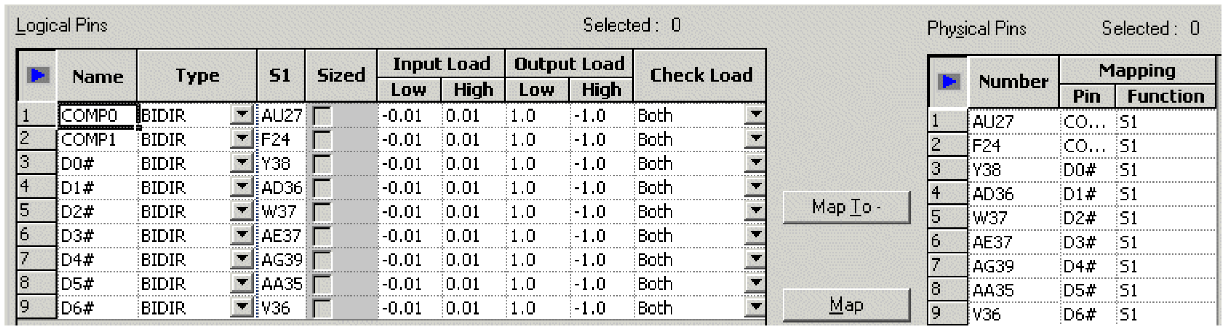
   The filled spreadsheet should appear like the one displayed below:



	A	B	C
1	COMP0	Input/Output	AU27
2	COMP1	Input/Output	F24
3	D0#	Input/Output	Y38
4	D1#	Input/Output	AD36
5	D2#	Input/Output	W37
6	D3#	Input/Output	AE37
7	D4#	Input/Output	AG39
8	D5#	Input/Output	AA35
9	D6#	Input/Output	V36

6. Select the three columns in the spreadsheet and press Ctrl + C.
7. Press Ctrl + I to insert a blank row in the *Logical Pins* grid.
8. Select the empty cell under the *Name* column in *Logical Pins* grid and press Ctrl + V.

   The pin information along with the pin type and the mapping information is copied into the *Logical Pins* grid. The *Physical Pins* grid is also updated automatically. Note that the `Input/Output` pin type is automatically converted to the `BIDIR` type in the *Logical Pins* grid. This translation is controlled through the `propfile.prop` file located at `<`*`your_inst_dir`*`>/share/cdssetup/LMAN`. For more information, see the *Advanced Tasks* chapter in *Part Developer User Guide.*



Logical Pins Selected : 0

	Name	Type	S1	Sized	Input Load		Output Load		Check Load
					Low	High	Low	High	
1	COMP0	BIDIR	AU27	☐	-0.01	0.01	1.0	-1.0	Both
2	COMP1	BIDIR	F24	☐	-0.01	0.01	1.0	-1.0	Both
3	D0#	BIDIR	Y38	☐	-0.01	0.01	1.0	-1.0	Both
4	D1#	BIDIR	AD36	☐	-0.01	0.01	1.0	-1.0	Both
5	D2#	BIDIR	W37	☐	-0.01	0.01	1.0	-1.0	Both
6	D3#	BIDIR	AE37	☐	-0.01	0.01	1.0	-1.0	Both
7	D4#	BIDIR	AG39	☐	-0.01	0.01	1.0	-1.0	Both
8	D5#	BIDIR	AA35	☐	-0.01	0.01	1.0	-1.0	Both
9	D6#	BIDIR	V36	☐	-0.01	0.01	1.0	-1.0	Both

Physical Pins Selected : 0

	Number	Mapping	
		Pin	Function
1	AU27	CO...	S1
2	F24	CO...	S1
3	Y38	D0#	S1
4	AD36	D1#	S1
5	W37	D2#	S1
6	AE37	D3#	S1
7	AG39	D4#	S1
8	AA35	D5#	S1
9	V36	D6#	S1

Caution

When copying data from PDFs, invalid characters in pin names need to be fixed manually. Part Developer will generate errors if an attempt is made to save a part with invalid characters in pin names.

9. Choose *File – Save* to save the part.

Summary

In this chapter, you learned to create parts from PDF datasheets.

7 Creating Asymmetrical Parts

Objective

To become familiar with steps involved in creating asymmetrical parts.

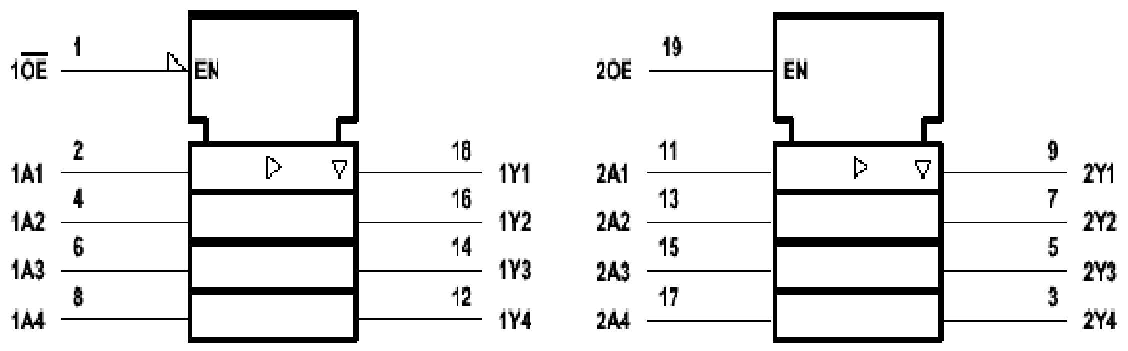
In this chapter, you will learn to:

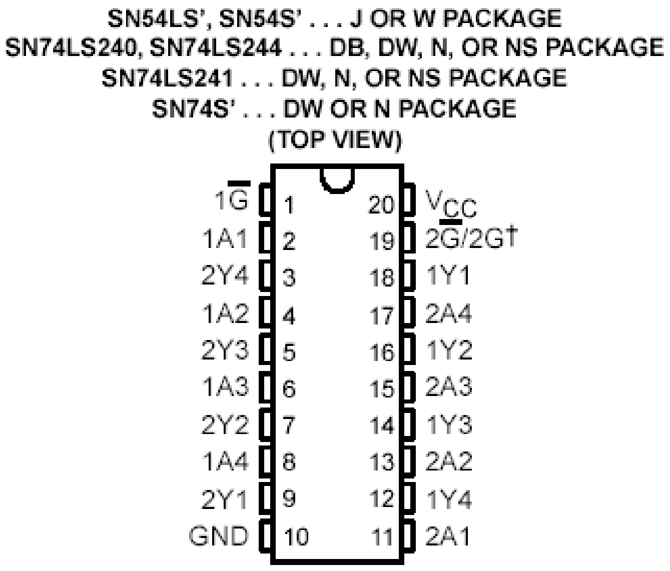
- Use the Package Editor to enter pin information.
- Specify pin information for multiple slots.
- Create symbols for each slot group.

Overview

An asymmetrical part is a part in which multiple functions are present in a package. For example, LS241, an 8-slot part, with two different functions, is an asymmetrical part. This chapter teaches you how to create LS241. By following the steps detailed here, you can create asymmetrical parts.

Understanding the LS241 Part




SN54LS', SN54S' . . . J OR W PACKAGE
SN74LS240, SN74LS244 . . . DB, DW, N, OR NS PACKAGE
SN74LS241 . . . DW, N, OR NS PACKAGE
SN74S' . . . DW OR N PACKAGE
(TOP VIEW)
1G 1
1A1 2
2Y4 3
1A2 4
2Y3 5
1A3 6
2Y2 7
1A4 8
2Y1 9
GND 10
20 VCC
19 2G/2G†
18 1Y1
17 2A4
16 1Y2
15 2A3
14 1Y3
13 2A2
12 1Y4
11 2A1

As displayed, LS241 is an 8-slot part with a low-asserted enable signal (1OE*) and a high-asserted enable signal (2OE). The high-asserted enable signal 2OE is present in four slots and the low-asserted enable signal 1OE* is present in the remaining four. This divides the part into two groups. The first group has 2OE as the enable pin and the second group has 1OE* as the enable pin. Because the functionality of each slot in a group is the same and because of the different assertion signals across the slots, the logical pin lists for the sections or slots are different.

Task Overview

Do the following:

- Create the `LS241` part in the `my_lib` library.
- Create a package.
- Enter the pin information through the Package Editor.
- Create symbols for the different slot groups.

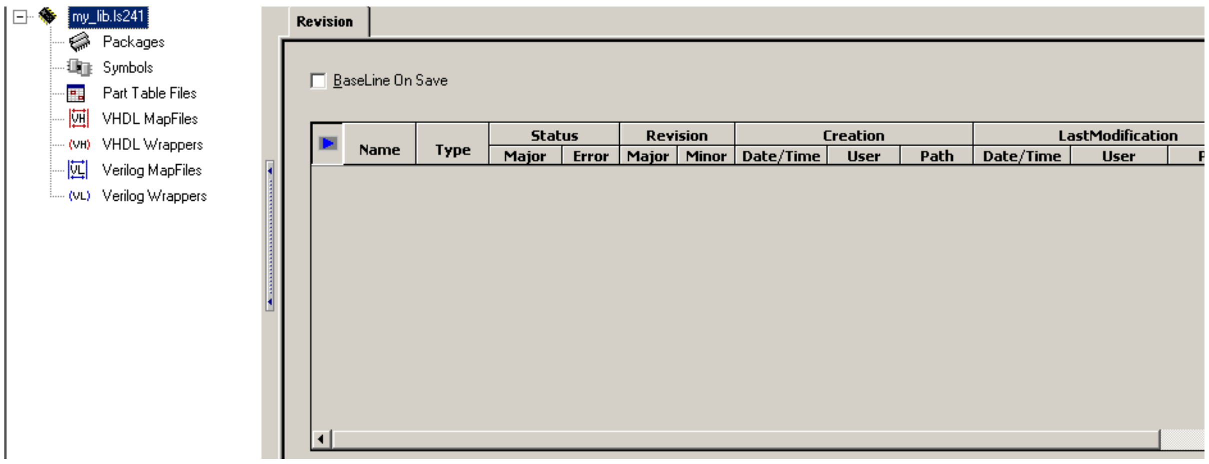
Steps

1. Select *File – New – Cell*.

   The New Cell dialog box appears.

2. Select the *my_lib* library.
3. Enter `ls241` in the *Cell* field and click *OK*.

   The Cell Editor appears with the empty LS241 part.



	Name	Type	Status		Revision		Creation			LastModification	
			Major	Error	Major	Minor	Date/Time	User	Path	Date/Time	User

4. Right-click on the *Packages* entry in the cell tree and select *New.*

   A new package, *LS241*, is created and loaded in the Package Editor.
5. Right-click in the *Logical Pins* grid on the *Package Pin* page and select *Insert Row After*.

   A blank row is created.

   Because LS241 has eight slots, you will need to create eight slots.
6. To create eight slots, click *Functions/Slots*.

   The Edit Functions dialog box appears.

   Since slot S1 already exists, you will need to add seven more slots.
7. Click *Add,* specify 7 in the Specify the number of slots dialog box, and click *OK*.
8. Click *OK to close the* Edit Functions dialog box.

   Next, you will enter the pins.
9. Enter `1OE*` in the *Name* column.
10. Because the pin is of type input, select INPUT from the *Type* drop-down list.

    Now, the pin 1OE* is common across the four slots.
11. Since 1OE* is mapped to physical pin 1, enter `1` in the *S1, S2, S3, and S4* columns.

12. Since 1OE* is not present in the remaining four slots, select slots S5 to S8 and click *Map To -*.

    This maps the selected slots to `-`.

13. To add another row, press Ctrl + I.

    A new row gets added to the Logical Pins grid.

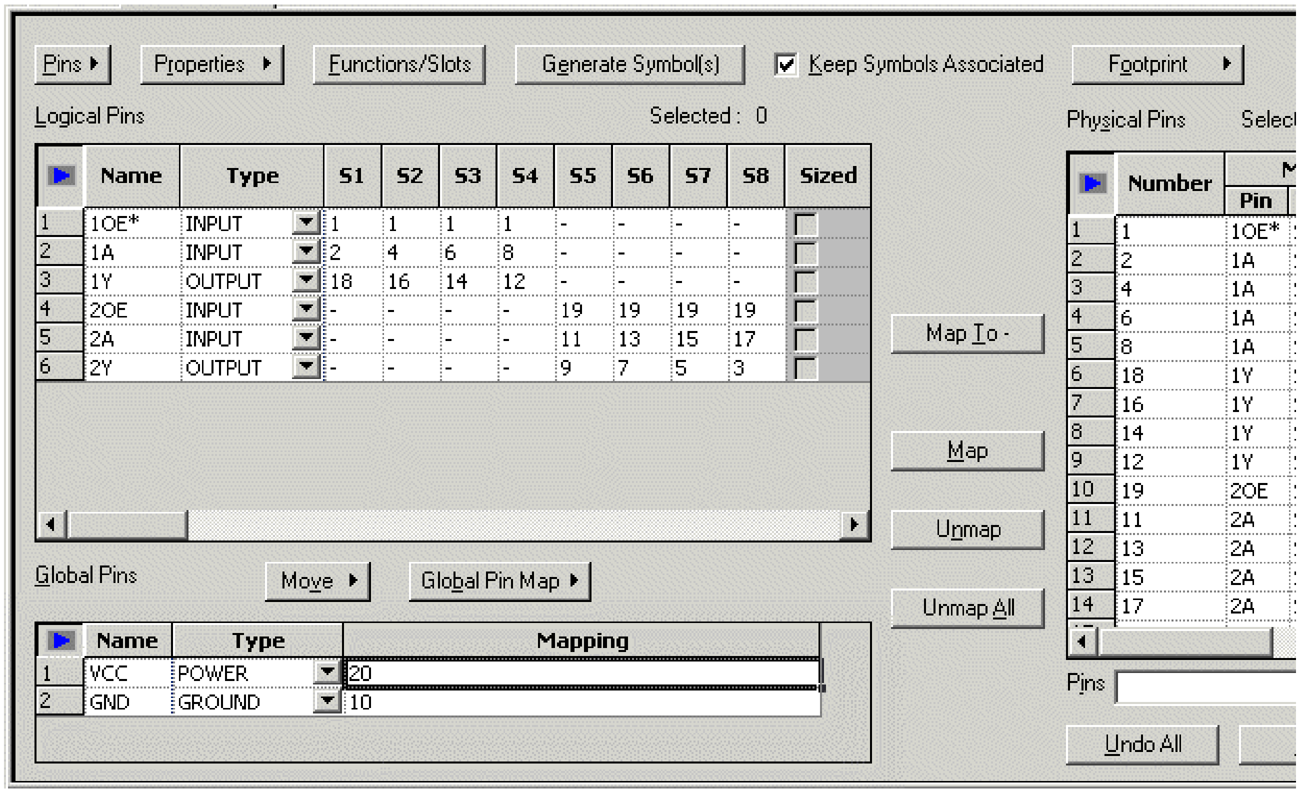
14. Enter `1A` in the *Name* column.
15. Select *INPUT* from the *Type* drop-down list.
16. Since `1A` is present in the first 4 slots and mapped to the physical pins `2`,`4`, `6` and `8`, enter `2`, `4`, `6`, and `8` under S1, S2, S3, and S4, respectively.
17. Select the slots S5 to S8 for pin 1A, and click *Map To -*.
18. To add another row, press Ctrl + I.

    A new row gets added to the Logical Pins grid.

19. Enter `1Y` in the *Name* column.
20. Select *OUTPUT* from the *Type* drop-down list.
21. Since `1Y` is present in the first 4 slots and mapped to physical pins `18`,`16`, `14,` and `12`, enter `18`,`16`, `14,` and `12` under S1, S2, S3, and S4, respectively.
22. Select the slots S5 to S8 for pin 1Y and click *Map To -*.

    Similarly, enter the remaining pins.

After pins are entered, the Package Pin page should appear as follows:



☑ Keep Symbols Associated

Logical Pins   Selected : 0

	Name	Type	S1	S2	S3	S4	S5	S6	S7	S8	Sized
1	1OE*	INPUT	1	1	1	1	-	-	-	-	☐
2	1A	INPUT	2	4	6	8	-	-	-	-	☐
3	1Y	OUTPUT	18	16	14	12	-	-	-	-	☐
4	2OE	INPUT	-	-	-	-	19	19	19	19	☐
5	2A	INPUT	-	-	-	-	11	13	15	17	☐
6	2Y	OUTPUT	-	-	-	-	9	7	5	3	☐

Global Pins

	Name	Type	Mapping
1	VCC	POWER	20
2	GND	GROUND	10

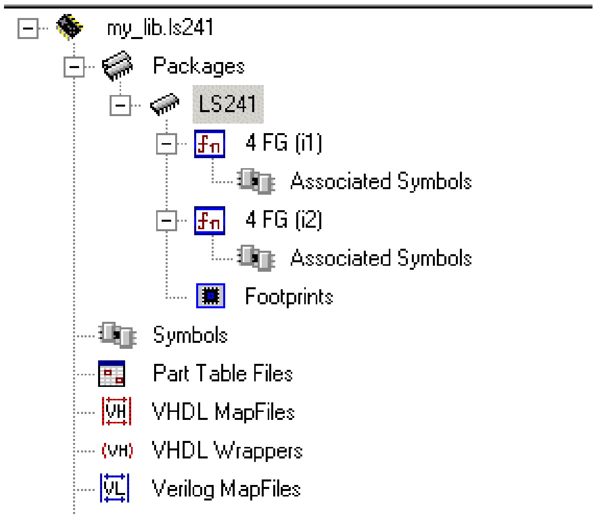
Physical Pins   Selec

	Number	Pin
1	1	1OE*
2	2	1A
3	4	1A
4	6	1A
5	8	1A
6	18	1Y
7	16	1Y
8	14	1Y
9	12	1Y
10	19	2OE
11	11	2A
12	13	2A
13	15	2A
14	17	2A

Pins

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

The *Packages* entry in the cell tree is updated to show the function groups in the package and the number of slots in the function group. In this case, there are two function groups with four slots in each function group.



- my_lib.ls241
  - Packages
    - LS241
      - 4 FG (i1)
        - Associated Symbols
      - 4 FG (i2)
        - Associated Symbols
      - Footprints
  - Symbols
  - Part Table Files
  - VHDL MapFiles
  - VHDL Wrappers
  - Verilog MapFiles

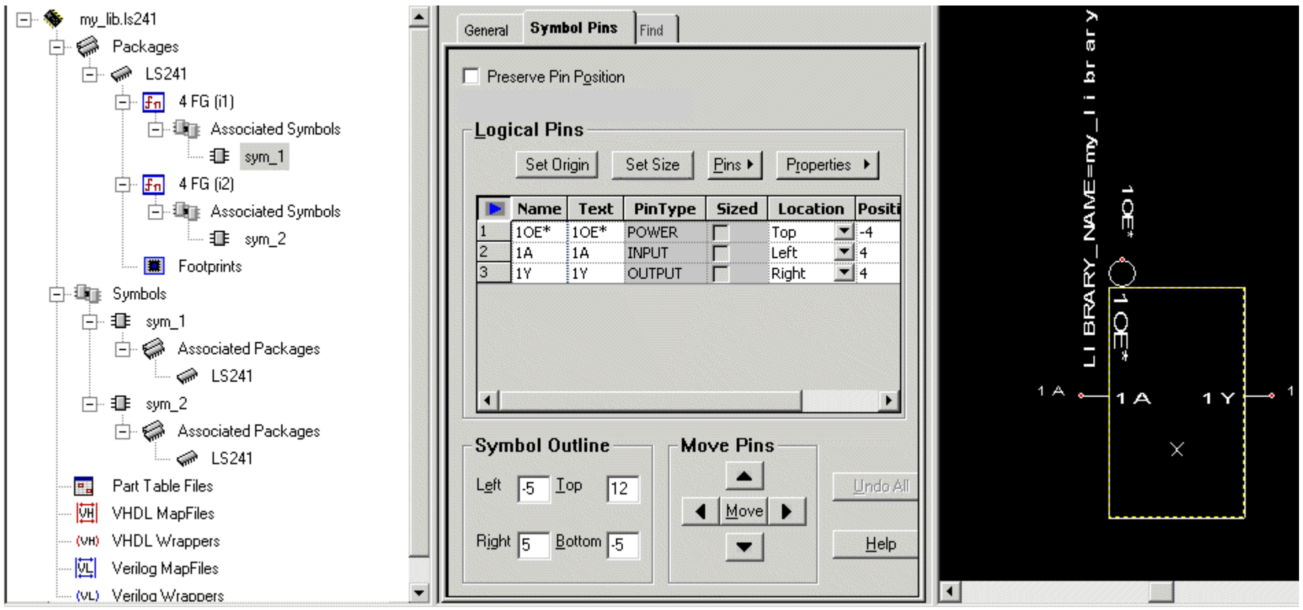
Next, you will create a symbol for each function group.

24. To generate a symbol for the first function group, right-click on 4 FG[i1] and choose *Generate Symbol(s)* from the pop-up menu.

The symbol is generated for the function group.

25. Similarly, generate a symbol for the other function group.

The following graphic shows the first symbol displayed on the Symbol Editor canvas.



- my_lib.ls241
  - Packages
    - LS241
      - 4 FG (i1)
        - Associated Symbols
          - sym_1
      - 4 FG (i2)
        - Associated Symbols
          - sym_2
      - Footprints
  - Symbols
    - sym_1
      - Associated Packages
        - LS241
    - sym_2
      - Associated Packages
        - LS241
  - Part Table Files
  - VHDL MapFiles
  - VHDL Wrappers
  - Verilog MapFiles
  - Verilog Wrappers

Preserve Pin Position: unchecked

Logical Pins

	Name	Text	PinType	Sized	Location	Positi
1	1OE*	1OE*	POWER	☐	Top	-4
2	1A	1A	INPUT	☐	Left	4
3	1Y	1Y	OUTPUT	☐	Right	4

Symbol Outline: Left -5, Top 12, Right 5, Bottom -5

26. Choose *File – Save* to save the part.

Summary

In this chapter, you learned how to create asymmetrical parts.

8

Working with Differential Pairs

Objective

To become familiar with the steps in adding and removing differential pair information in Part Developer.

In this chapter, you will learn to:

- Autocreate differential pairs in all cells of a library
- Autocreate differential pairs through the Package Editor
- Create a differential pair from selected pins
- Remove differential pair properties from a differential pair

Overview

When creating parts in Part Developer, you can capture differential pair information from datasheets. If you have legacy libraries without differential pair information, you can run a batch utility and create differential pairs based on specified differential pair recognition rules. This chapter covers various ways of creating differential pairs and the procedure for removing differential pair information.

To try the various procedures described in this chapter, you will use the project `dp_proj` and the library `dp_proj_lib` in the `library_project` folder at `<`*`your_inst_dir`*`>/doc/pdv_tut/tutorial_data`. Make sure that you have copied the `library_project` folder to `<`*`your_work_area`*`>.`

Points to Remember about Differential Pair Support in Part Developer

- The differential pair property is associated with logical pins.
- The positive and negative pins comprising a differential pair must have the same pin type.

- The differential pair property cannot be associated with `GROUND`, `POWER`, and `NC` pin types.
- The differential pair property is saved in chips only.

Autocreating Differential Pairs in All Cells of a Library

Add differential pair information to all parts in the `dp_lib` library based on the following differential pair recognition rule:

```
DiffPair_Recognition_Rules ’n:SUFFIX,p:SUFFIX;-
     :SUFFIX,+:SUFFIX;_L:SUFFIX,_H:SUFFIX;_LOW:SUFFIX,_HIGH:SUFFIX;_N:SUFFIX,_P:
     SUFFIX’
```

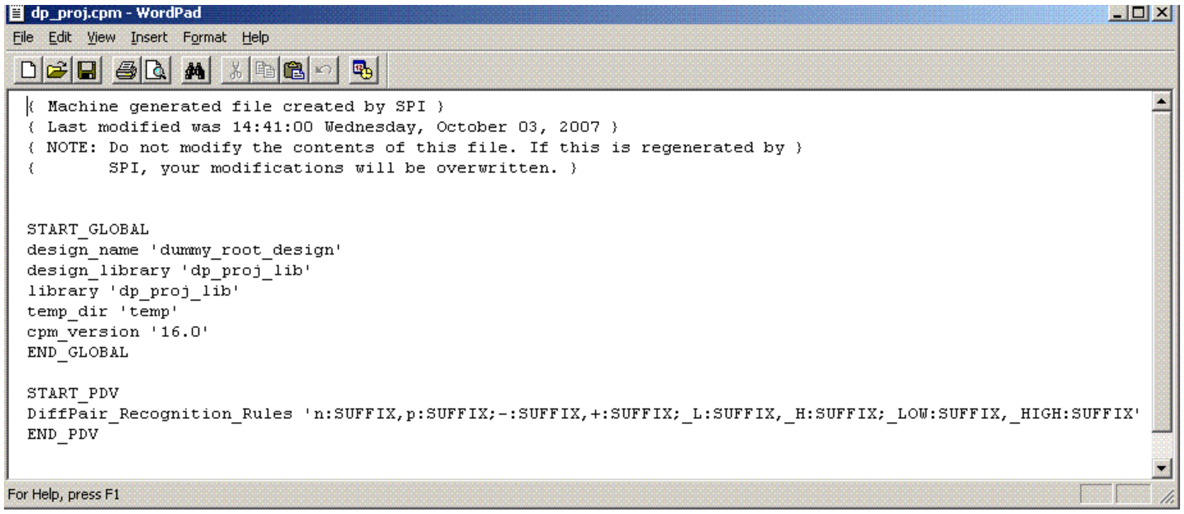
The task involves the following subtasks:

1. Configuring the default differential pair recognition rule in your local project file to add the naming scheme `_N:SUFFIX,_P:SUFFIX`
2. Configuring the low assertion setup in Part Developer to disallow the use of the `_N` suffix
3. Running the `con2con` utility on the `dp_lib` library with the `autocreatediffpair` option

Configuring the Default Differential Pair Recognition Rule for a Project

To configure the default differential pair recognition rule for a project, you need to copy the `DiffPair_Recognition_Rules` definition from the installation CPM file (`cds.cpm`) or the site CPM file (`setup.cpm`) to the project file. For the task at hand, the default differential pair recognition rule has been added from `cds.cpm` to the project file `dp_proj.cpm`.

1. Open the `dp_proj.cpm` project in a text editor.



```
{ Machine generated file created by SPI }
{ Last modified was 14:41:00 Wednesday, October 03, 2007 }
{ NOTE: Do not modify the contents of this file. If this is regenerated by }
{       SPI, your modifications will be overwritten. }


START_GLOBAL
design_name 'dummy_root_design'
design_library 'dp_proj_lib'
library 'dp_proj_lib'
temp_dir 'temp'
cpm_version '16.0'
END_GLOBAL

START_PDV
DiffPair_Recognition_Rules 'n:SUFFIX,p:SUFFIX;-:SUFFIX,+:SUFFIX;_L:SUFFIX,_H:SUFFIX;_LOW:SUFFIX,_HIGH:SUFFIX'
END_PDV
```

Notice that the default `DiffPair_Recognition_Rules` definition does not include the naming scheme `_N:SUFFIX,_P:SUFFIX`.

2. To add the naming scheme, click before the closing `’` character, type the following:

   `;_N:SUFFIX,_P:SUFFIX`

   The differential pair recognition rule is modified to:

```
DiffPair_Recognition_Rules ’n:SUFFIX,p:SUFFIX;-
     :SUFFIX,+:SUFFIX;_L:SUFFIX,_H:SUFFIX;_LOW:SUFFIX,_HIGH:SUFFIX;_N:SUFFIX,_P:
     SUFFIX’
```

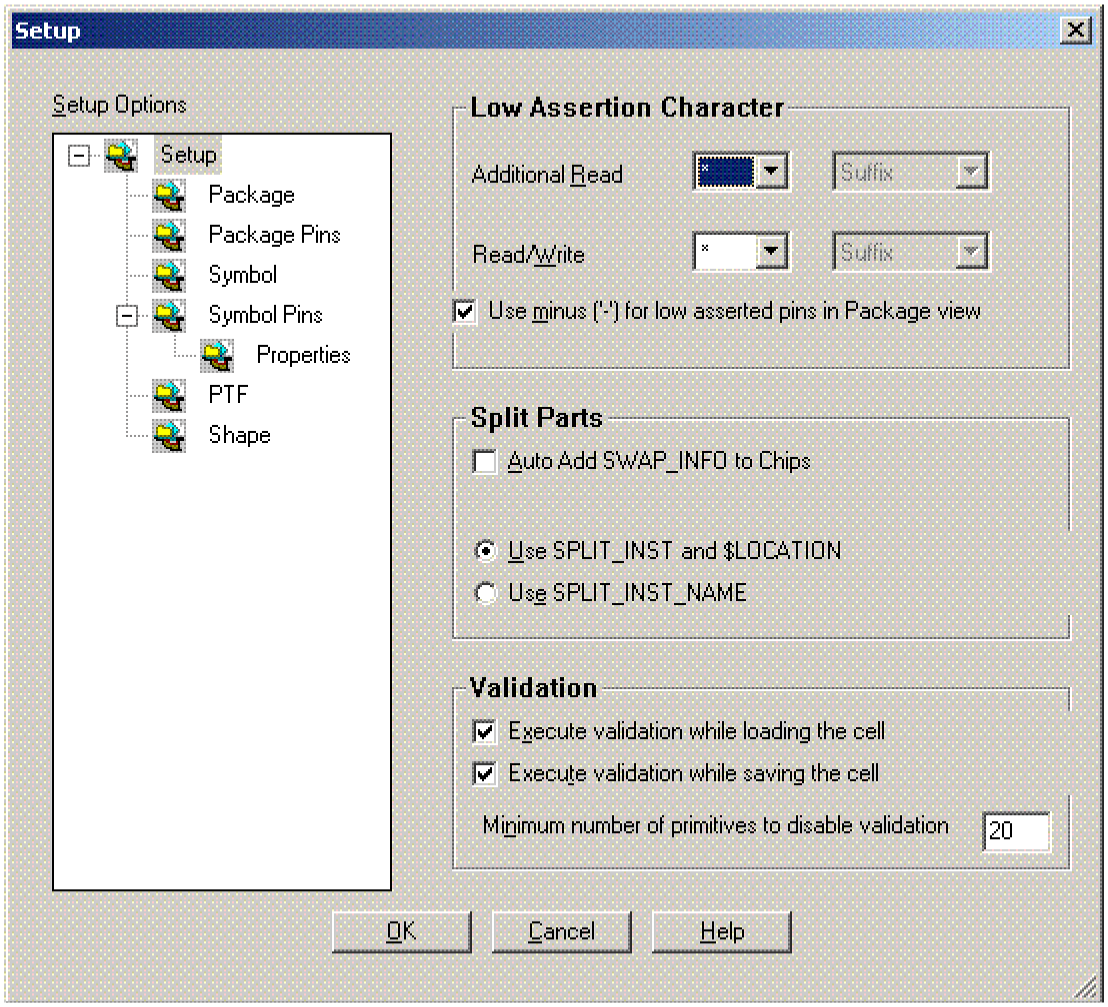
3. Save and close the project file.

Configuring the Low Assertion Setup to Disallow the Use of the _N Suffix

By default, Part Developer considers pin names with the `_N` suffix as low-asserted pins. Therefore, you need to modify the default low assertion setup if you want Part Developer to use the `_N` suffix to identify negative pins of differential pairs.

1. Choose *File – Open Project*.
2. Click the *Browse* button to select the `dp_proj.cpm` project file from the `library_project` folder and click *Open*.
3. Click *OK*.
4. Choose *Tools – Setup*.

5. Click on the down arrow button in the *Additional Read* list box.
6. Select *.



Low Assertion Character
Additional Read: * | Suffix
Read/Write: * | Suffix
[✓] Use minus ('-') for low asserted pins in Package view

Split Parts
[ ] Auto Add SWAP_INFO to Chips
(●) Use SPLIT_INST and $LOCATION
( ) Use SPLIT_INST_NAME

Validation
[✓] Execute validation while loading the cell
[✓] Execute validation while saving the cell
Minimum number of primitives to disable validation: 20

7. Click *OK*.

Running the con2con Utility on a Library with the autocreatediffpair Option

After you have configured the differential pair recognition rule according to your requirements, you can run the `con2con` utility with the `autocreatediffpair` option to add differential pair information to libraries in batch mode. For the task at hand, you will add differential pair information to all parts in the `dp_lib` library.

To run the `con2con` utility on the `dp_lib` library with the `autocreatediffpair` option:

➤ Type the following command at the command prompt:

```
con2con -product pcb_librarian_expert -proj <your_work_area>/library_project/
     dp_proj.cpm -cdslib <your_work_area>/library_project/cds.lib -lib
     dp_proj_lib -autocreatediffpair
```

Important

In the specified `con2con` command, <your_work_area> is the location where you have copied the `library_project` folder from <*your_inst_dir*>/`doc/pdv_tut/tutorial_data` to try the procedures detailed in this tutorial.

Autocreating Differential Pairs through the Package Editor

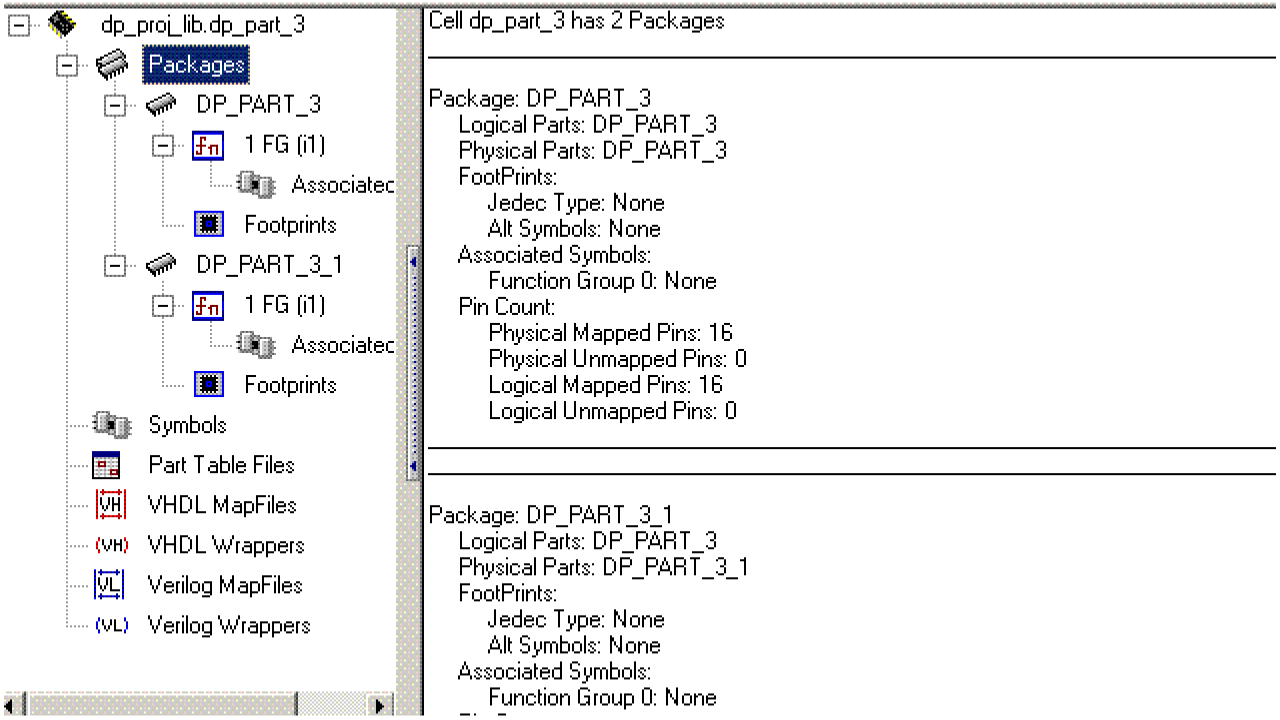
The `dp_proj_lib` library contains parts with differential pair properties. If you add new parts to this library, you can add differential pair properties only to a selected part through the Package Editor.

Create a part, `dp_part_3`, in the `dp_proj_lib` library with the following pin information in packages `DP_PART_3` and `DP_PART_3_1`:

Pin Name	Pin Number	Pin Type
DATA_L<4..0>	1,2,3,4,5	INPUT
DATA_H<4..0>	6,7,8,9,10	INPUT
AS1-	11	ANALOG
AS1+	12	ANALOG
-RD	13	OUTPUT
+RD	14	OUTPUT
VCC	15	POWER
GND	16	GROUND

For information on how to create a part, see the *Creating a Flat Part* chapter.

After the part is created, perform the following steps to automatically create differential pairs in both the packages:

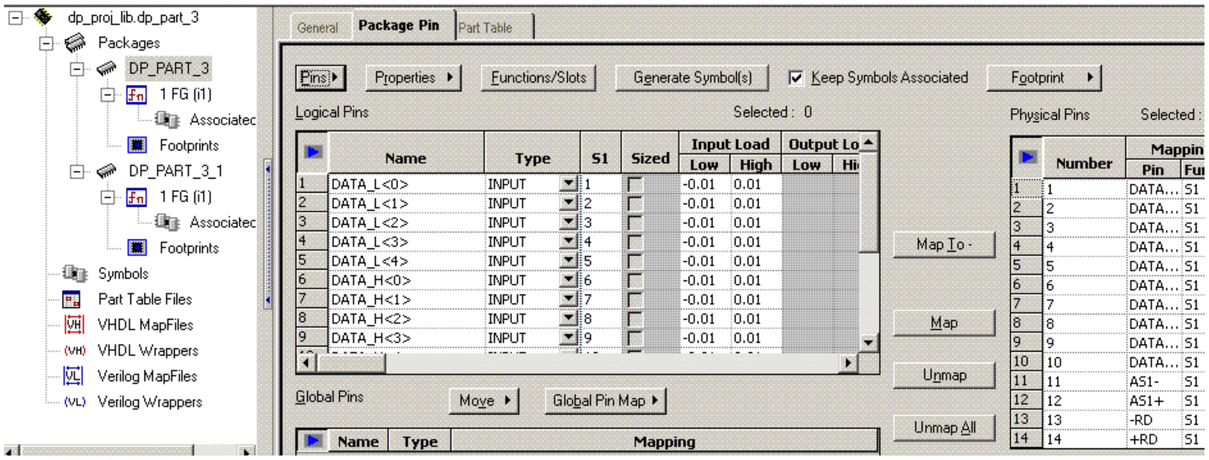


Cell dp_part_3 has 2 Packages

Package: DP_PART_3
Logical Parts: DP_PART_3
Physical Parts: DP_PART_3
FootPrints:
Jedec Type: None
Alt Symbols: None
Associated Symbols:
Function Group 0: None
Pin Count:
Physical Mapped Pins: 16
Physical Unmapped Pins: 0
Logical Mapped Pins: 16
Logical Unmapped Pins: 0

Package: DP_PART_3_1
Logical Parts: DP_PART_3
Physical Parts: DP_PART_3_1
FootPrints:
Jedec Type: None
Alt Symbols: None
Associated Symbols:
Function Group 0: None

1. Select either of the two packages.



Logical Pins Selected : 0

	Name	Type	S1	Sized	Input Load		Output Lo	
					Low	High	Low	Hi
1	DATA_L<0>	INPUT	1		-0.01	0.01		
2	DATA_L<1>	INPUT	2		-0.01	0.01		
3	DATA_L<2>	INPUT	3		-0.01	0.01		
4	DATA_L<3>	INPUT	4		-0.01	0.01		
5	DATA_L<4>	INPUT	5		-0.01	0.01		
6	DATA_H<0>	INPUT	6		-0.01	0.01		
7	DATA_H<1>	INPUT	7		-0.01	0.01		
8	DATA_H<2>	INPUT	8		-0.01	0.01		
9	DATA_H<3>	INPUT	9		-0.01	0.01		

Physical Pins Selected :

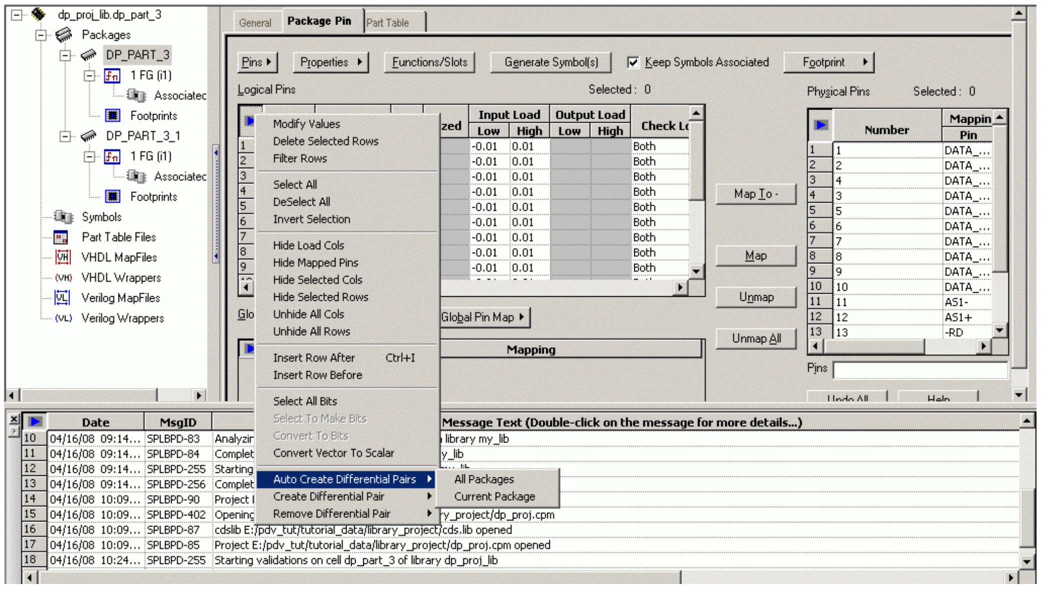
	Number	Mappin	
		Pin	Fu
1	1	DATA...	S1
2	2	DATA...	S1
3	3	DATA...	S1
4	4	DATA...	S1
5	5	DATA...	S1
6	6	DATA...	S1
7	7	DATA...	S1
8	8	DATA...	S1
9	9	DATA...	S1
10	10	DATA...	S1
11	11	AS1-	S1
12	12	AS1+	S1
13	13	-RD	S1
14	14	+RD	S1

Global Pins

	Name	Type	Mapping

2. Right-click in the *Logical Pins* grid.

3. Choose *Auto Create Differential Pairs - All Packages*.



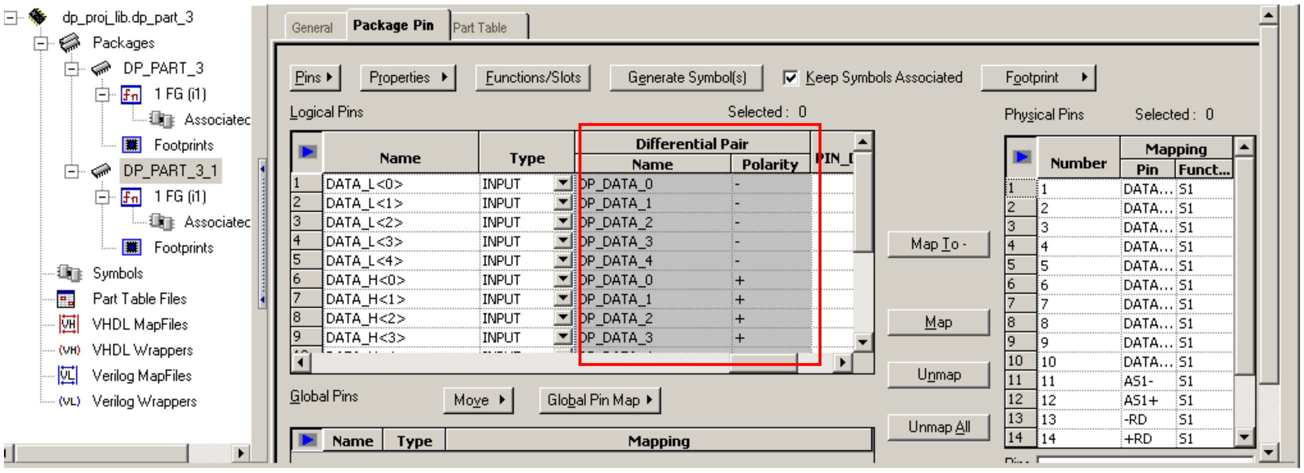
	Input Load Low	Input Load High	Output Load Low	Output Load High	Check Lo
1	-0.01	0.01			Both
2	-0.01	0.01			Both
3	-0.01	0.01			Both
4	-0.01	0.01			Both
5	-0.01	0.01			Both
6	-0.01	0.01			Both
7	-0.01	0.01			Both
8	-0.01	0.01			Both
9	-0.01	0.01			Both

	Number	Mappin Pin
1	1	DATA_...
2	2	DATA_...
3	4	DATA_...
4	3	DATA_...
5	5	DATA_...
6	6	DATA_...
7	7	DATA_...
8	8	DATA_...
9	9	DATA_...
10	10	DATA_...
11	11	AS1-
12	12	AS1+
13	13	-RD

	Date	MsgID	Message Text (Double-click on the message for more details...)
10	04/16/08 09:14...	SPLBPD-83	Analyzir … library my_lib
11	04/16/08 09:14...	SPLBPD-84	Complet … y_lib
12	04/16/08 09:14...	SPLBPD-255	Starting
13	04/16/08 09:14...	SPLBPD-256	Complet
14	04/16/08 10:09...	SPLBPD-90	Project I
15	04/16/08 10:09...	SPLBPD-402	Opening … y_project/dp_proj.cpm
16	04/16/08 10:09...	SPLBPD-87	cdslib E:/pdv_tut/tutorial_data/library_project/cds.lib opened
17	04/16/08 10:09...	SPLBPD-85	Project E:/pdv_tut/tutorial_data/library_project/dp_proj.cpm opened
18	04/16/08 10:24...	SPLBPD-255	Starting validations on cell dp_part_3 of library dp_proj_lib

Choosing *All Packages* ensures that the differential pair properties are added in both packages to all pins that match any of the naming schemes specified in the `DiffPair_Recognition_Rules` definition in the project CPM file. You choose *Current Package* when you want the differential pair properties to be added only in the selected package.

The following graphic shows differential pair information added to the `DP_PART_3_1` package:



	Name	Type	Differential Pair Name	Differential Pair Polarity	PIN_I
1	DATA_L<0>	INPUT	DP_DATA_0	-	
2	DATA_L<1>	INPUT	DP_DATA_1	-	
3	DATA_L<2>	INPUT	DP_DATA_2	-	
4	DATA_L<3>	INPUT	DP_DATA_3	-	
5	DATA_L<4>	INPUT	DP_DATA_4	-	
6	DATA_H<0>	INPUT	DP_DATA_0	+	
7	DATA_H<1>	INPUT	DP_DATA_1	+	
8	DATA_H<2>	INPUT	DP_DATA_2	+	
9	DATA_H<3>	INPUT	DP_DATA_3	+	

	Number	Mapping Pin	Mapping Funct...
1	1	DATA...	S1
2	2	DATA...	S1
3	3	DATA...	S1
4	4	DATA...	S1
5	5	DATA...	S1
6	6	DATA...	S1
7	7	DATA...	S1
8	8	DATA...	S1
9	9	DATA...	S1
10	10	DATA...	S1
11	11	AS1-	S1
12	12	AS1+	S1
13	13	-RD	S1
14	14	+RD	S1

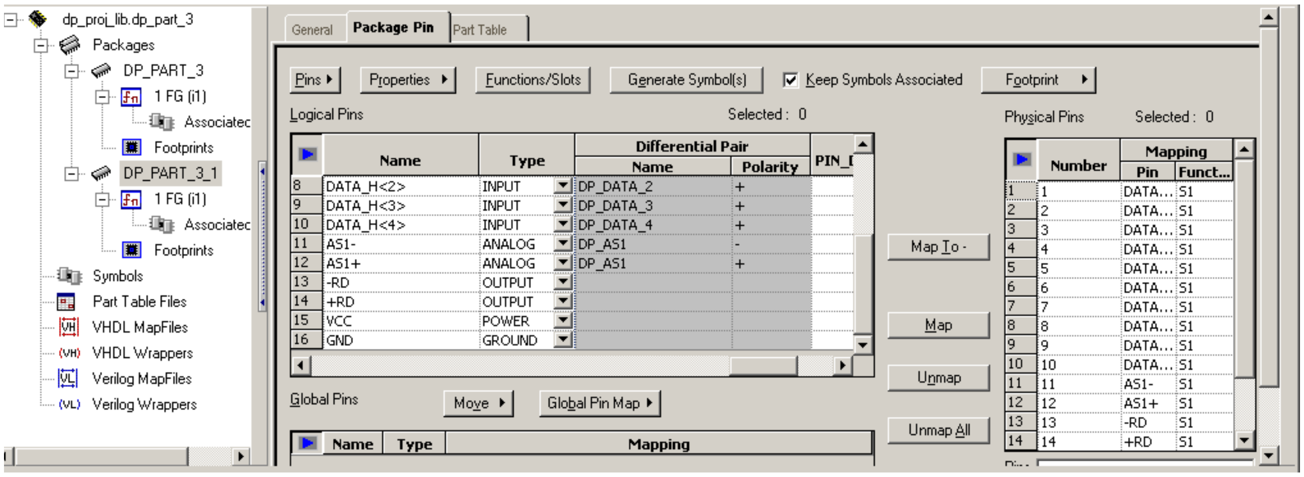
Creating a Differential Pair from Selected Pins

Typically, you select two pins that should constitute a differential pair and choose *Create Differential Pair* from the shortcut (RMB) menu in the following situations:

- You have added new pins to a part that already has differential pair information for existing pins.
- You want to create a differential pair even if the names of the constituent pins do not follow any of the naming schemes specified in the `DiffPair_Recognition_Rules` definition.

For the task at hand, you will create a differential pair of the `+RD` and `-RD pins` only in the `DP_PART_3_1` package.

1. Select the `DP_PART_3_1` package.



Logical Pins Selected : 0

	Name	Type	Differential Pair		PIN_I
			Name	Polarity	
8	DATA_H<2>	INPUT	DP_DATA_2	+	
9	DATA_H<3>	INPUT	DP_DATA_3	+	
10	DATA_H<4>	INPUT	DP_DATA_4	+	
11	AS1-	ANALOG	DP_AS1	-	
12	AS1+	ANALOG	DP_AS1	+	
13	-RD	OUTPUT			
14	+RD	OUTPUT			
15	VCC	POWER			
16	GND	GROUND			

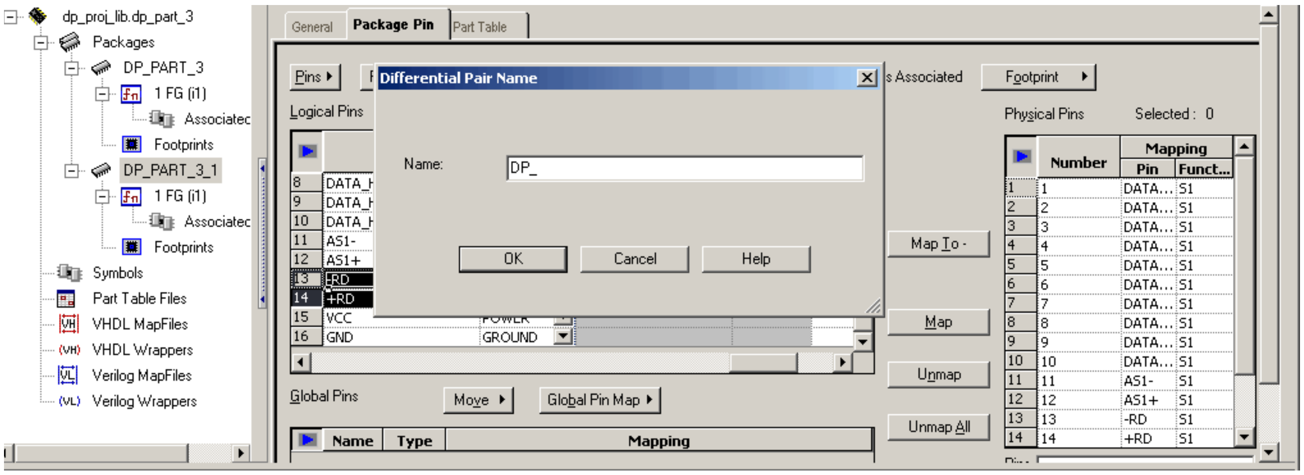
	Name	Type	Mapping

Physical Pins Selected : 0

	Number	Mapping	
		Pin	Funct...
1	1	DATA...	S1
2	2	DATA...	S1
3	3	DATA...	S1
4	4	DATA...	S1
5	5	DATA...	S1
6	6	DATA...	S1
7	7	DATA...	S1
8	8	DATA...	S1
9	9	DATA...	S1
10	10	DATA...	S1
11	11	AS1-	S1
12	12	AS1+	S1
13	13	-RD	S1
14	14	+RD	S1

2. Select the pins `+RD` and `-RD`.
3. Right-click on the selection and choose *Create Differential Pair – Current Package*.

The Differential Pair Name dialog box appears.



Differential Pair Name

Name: DP_

8	DATA_I	
9	DATA_I	
10	DATA_I	
11	AS1-	
12	AS1+	
13	-RD	
14	+RD	
15	VCC	POWER
16	GND	GROUND

Selected : 0

Number	Mapping	
	Pin	Funct...
1	DATA...	S1
2	DATA...	S1
3	DATA...	S1
4	DATA...	S1
5	DATA...	S1
6	DATA...	S1
7	DATA...	S1
8	DATA...	S1
9	DATA...	S1
10	DATA...	S1
11	AS1-	S1
12	AS1+	S1
13	-RD	S1
14	+RD	S1

Name	Type	Mapping

The *Name* field displays a name for the differential pair that Part Developer derives by adding the prefix or suffix specified in the `Default_Diffpair_Value` directive in `cds.cpm` to the basenames of the pin. In the current scenario, the *Name* field displays only the `Default_Diffpair_Value` directive value because the selected pins do not follow any of the rules specified through the `DiffPair_Recognition_Rules` directive.

For more information, see the Naming Differential Pairs section of the Creating Parts chapter of *Part Developer User Guide*.

4. Specify the differential pair name as `DP_RD`.

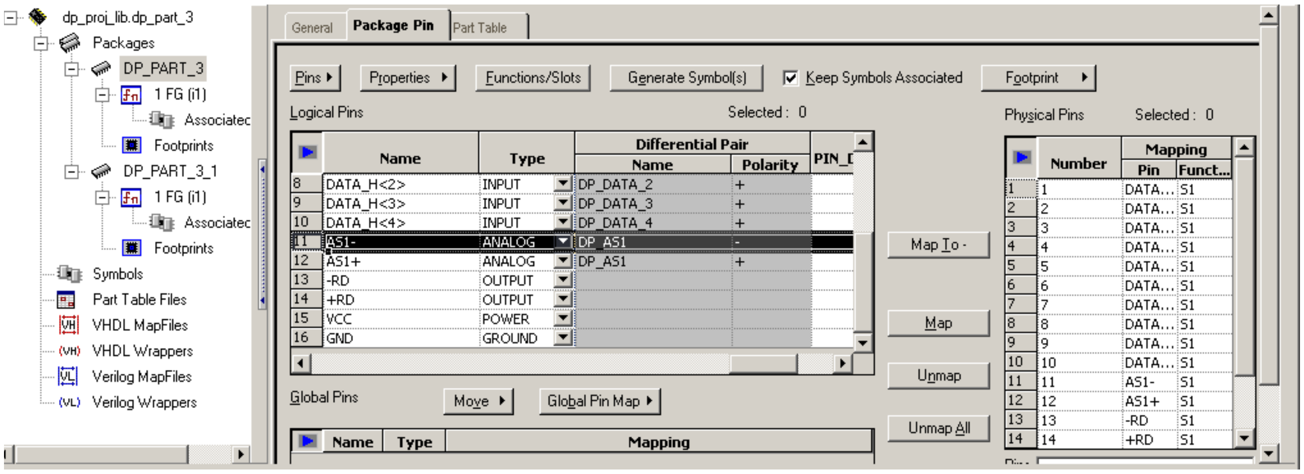
5. Click *OK* to save the modified differential pair name.

The differential pair information is added to pins `+RD` and `-RD.`

Removing Differential Pair Properties from a Differential Pair

Remove differential pair information from the differential pair `AS1` in the `DP_PART_3` package.

1. Select any of the two pins AS1+ and AS1-.



Logical Pins Selected : 0

	Name	Type	Differential Pair Name	Differential Pair Polarity	PIN_I
8	DATA_H<2>	INPUT	DP_DATA_2	+	
9	DATA_H<3>	INPUT	DP_DATA_3	+	
10	DATA_H<4>	INPUT	DP_DATA_4	+	
11	AS1-	ANALOG	DP_AS1	-	
12	AS1+	ANALOG	DP_AS1	+	
13	-RD	OUTPUT			
14	+RD	OUTPUT			
15	VCC	POWER			
16	GND	GROUND			

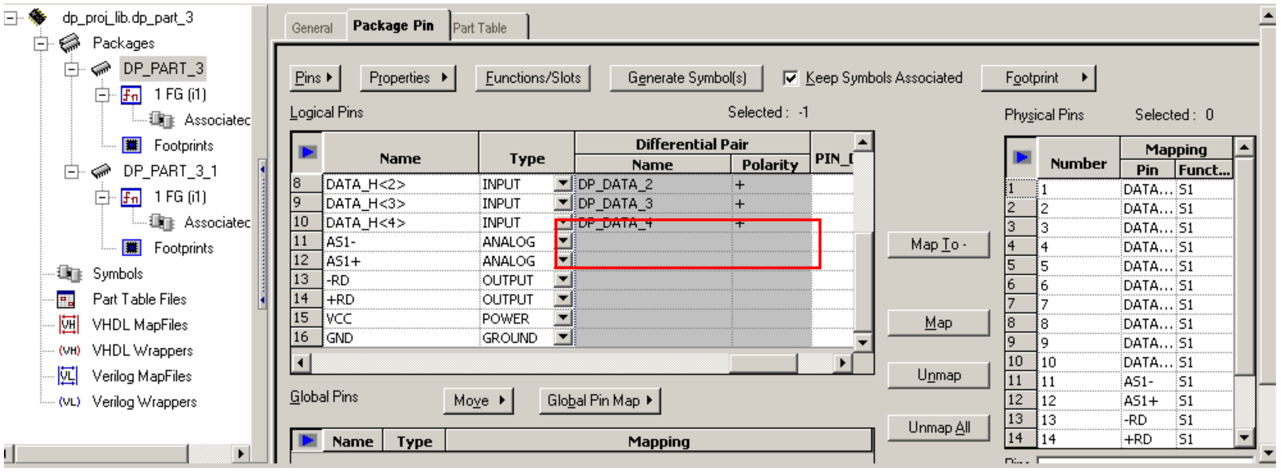
Physical Pins Selected : 0

	Number	Mapping Pin	Mapping Funct...
1	1	DATA...	S1
2	2	DATA...	S1
3	3	DATA...	S1
4	4	DATA...	S1
5	5	DATA...	S1
6	6	DATA...	S1
7	7	DATA...	S1
8	8	DATA...	S1
9	9	DATA...	S1
10	10	DATA...	S1
11	11	AS1-	S1
12	12	AS1+	S1
13	13	-RD	S1
14	14	+RD	S1

Name	Type	Mapping

2. Right-click on the selection and choose *Remove Differential Pair – Current Package*.

   The differential pair information is removed from both pins.



Logical Pins Selected : -1

	Name	Type	Differential Pair Name	Differential Pair Polarity	PIN_I
8	DATA_H<2>	INPUT	DP_DATA_2	+	
9	DATA_H<3>	INPUT	DP_DATA_3	+	
10	DATA_H<4>	INPUT	DP_DATA_4	+	
11	AS1-	ANALOG			
12	AS1+	ANALOG			
13	-RD	OUTPUT			
14	+RD	OUTPUT			
15	VCC	POWER			
16	GND	GROUND			

Physical Pins Selected : 0

	Number	Mapping Pin	Mapping Funct...
1	1	DATA...	S1
2	2	DATA...	S1
3	3	DATA...	S1
4	4	DATA...	S1
5	5	DATA...	S1
6	6	DATA...	S1
7	7	DATA...	S1
8	8	DATA...	S1
9	9	DATA...	S1
10	10	DATA...	S1
11	11	AS1-	S1
12	12	AS1+	S1
13	13	-RD	S1
14	14	+RD	S1

Name	Type	Mapping

Summary

In this chapter, you learned how to create differential pairs according to your requirements and remove differential pair properties from packages.

9

Creating Sizeable and HAS_FIXED_SIZE Symbols

Objective

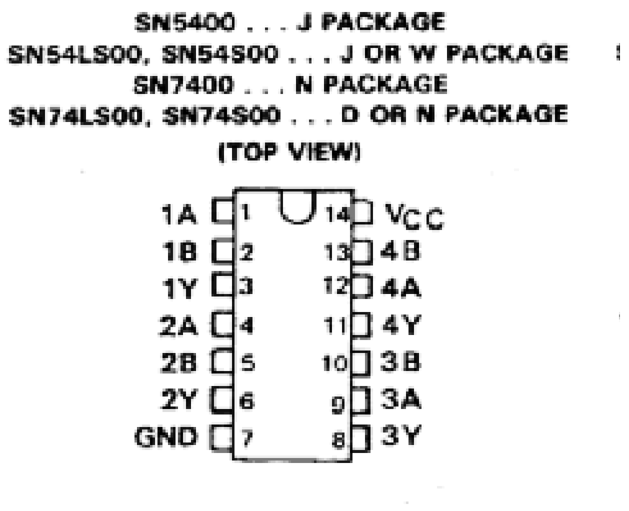
To become familiar with steps involved in creating sizeable and `HAS_FIXED_SIZE` symbols.

In this chapter, you will learn:

- About the methodology in creating sizeable and `HAS_FIXED_SIZE` symbols
- How to create a sizeable symbol
- How to create a `HAS_FIXED_SIZE` symbol

Overview

You will create the `LS00` part to understand the steps involved in creating sizeable parts. The pin information is displayed below:



Methodology

The following needs to be done to create sizeable and `HAS_FIXED_SIZE` symbols:

1. Create a symbol through the Symbol Editor and add the sizeable pins.
2. Create packages.
3. Create the necessary slots.
4. Create a second symbol, *sym_2*, by making a copy of the existing symbol, sym_1.
5. Specify the value of the `SIZE` property on the basis of the number of slots in *sym_2*. This will add the `HAS_FIXED_SIZE` property to *sym_2*.

Caution

Do not specify the value of the SIZE property in the first symbol because it might result in an incorrect generation of the entity view.

Task Overview

Create the LS00 part in the `my_lib` library and create sizeable and `HAS_FIXED_SIZE` symbols.

Steps

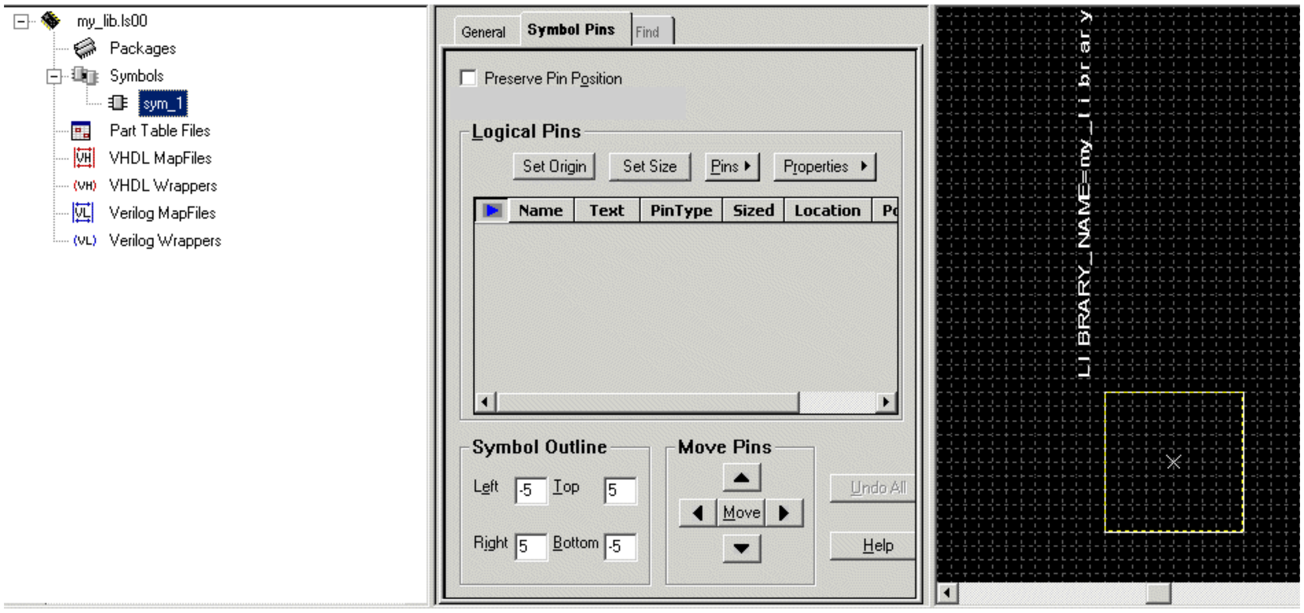
1. Choose *File – New – Cell*.

   The New Cell dialog box appears.
2. Select the *my_lib* library.
3. Enter `ls00` in the *Cell* field and click *OK*.

   The Cell Editor appears with the empty ls00 part.

   To enter sizeable pins, you need to use the Add Pin dialog box accessed through the Symbol Editor.
4. To access the Symbol Editor, right-click on the *Symbols* entry in the cell tree and choose *New*.

A new symbol, *sym_1,* is created for the part.



5. To access the Add Pin dialog box, choose *Pins – Add* from the Symbol Pins page.

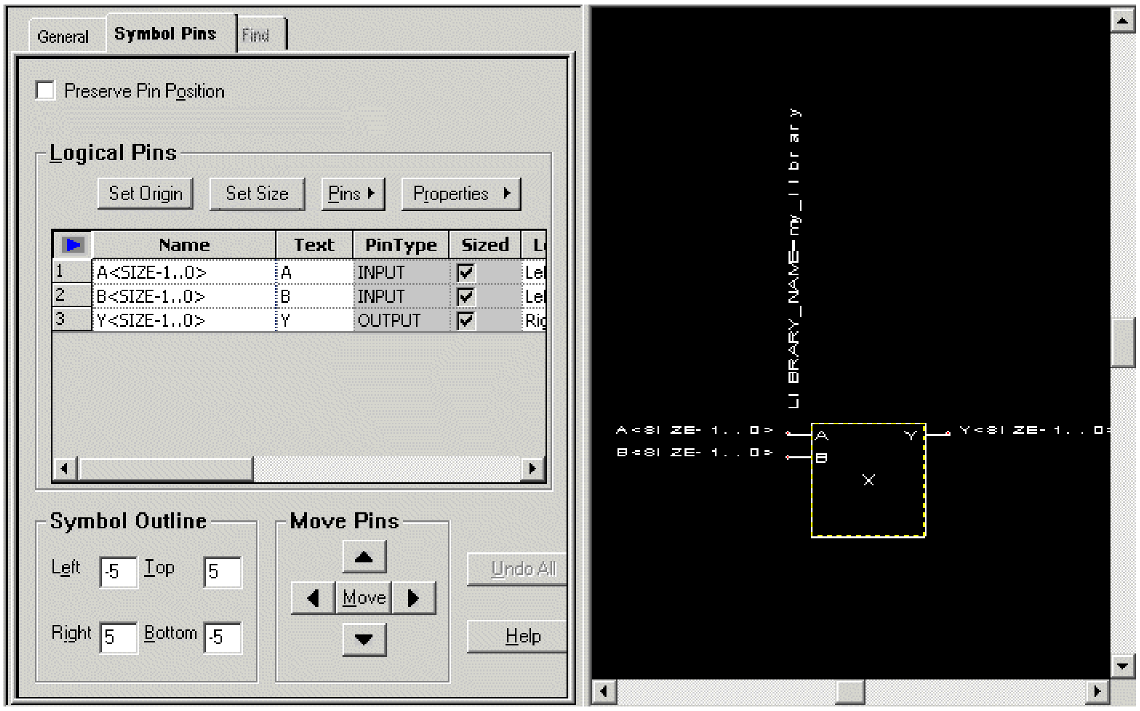
   The Add Pin dialog box appears.

6. Choose the *Sizeable* option.

   As shown in the datasheet, LS00 has two input pins, A and B, and one output pin, Y. These pins are present in four slots.

7. Enter A in the *Base Name* field.
8. Select *INPUT* from the *TYPE* drop-down list and click *Add*.
9. Enter B in the *Base Name* field.
10. Select *INPUT* from the *TYPE* drop-down list and click *Add*.
11. Enter Y in the *Base Name* field.
12. Select *OUTPUT* from the *TYPE* drop-down list and click *Add*.
13. To add the pins to the symbol, click *OK*.

The pins get added to the symbol with the `SIZE` property.



	Name	Text	PinType	Sized	Lo
1	A<SIZE-1..0>	A	INPUT	✓	Lel
2	B<SIZE-1..0>	B	INPUT	✓	Lel
3	Y<SIZE-1..0>	Y	OUTPUT	✓	Rig

Next, you will create a sizeable symbol.

Creating a Sizeable Symbol

1. Right-click on *sym_1* and choose *Generate Package*.

   The package is created for the LS00 part.

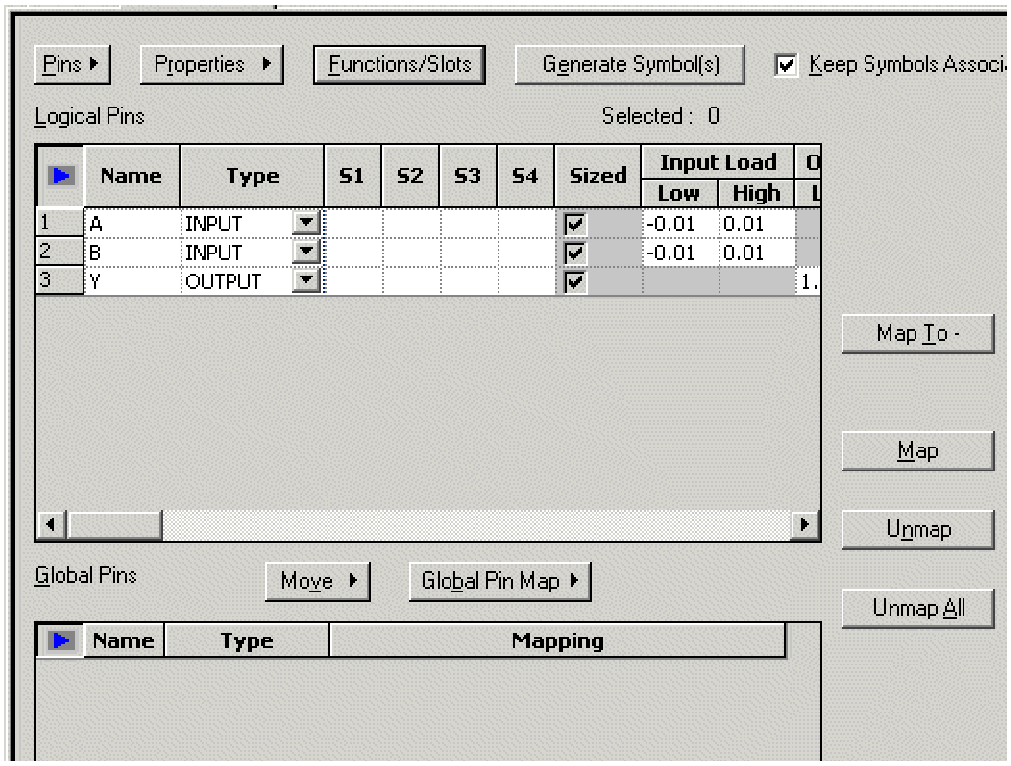
   Since the pins are spread in four slots, you will need to create four slots.

2. To create the four slots, click *Functions/Slots*.

   The Edit Functions dialog box appears.

3. To create 4 slots, click *Add* and specify 3 in the Specify the number of slots dialog box.
4. Click *OK*.
5. Click *OK* to close the Edit Functions dialog box.

The four slots appear in the Package Editor.

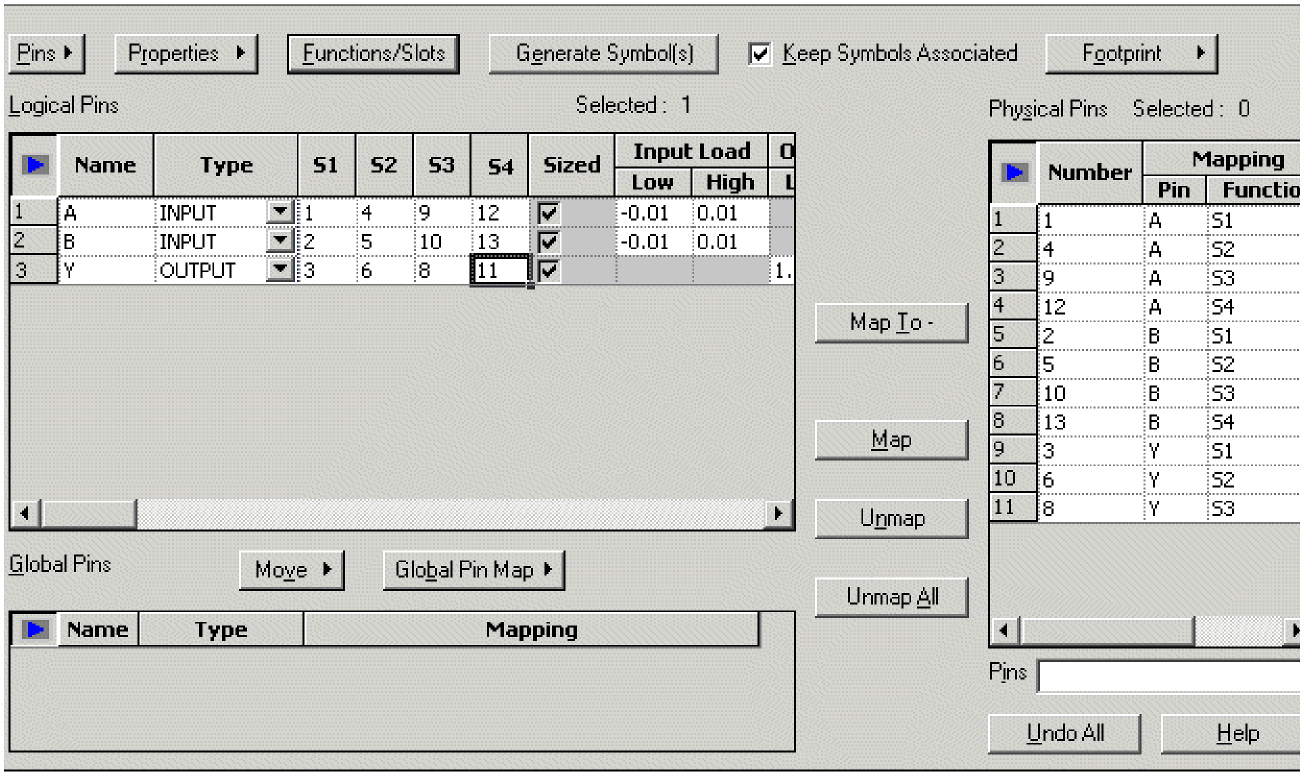


	Name	Type	S1	S2	S3	S4	Sized	Input Load		O
								Low	High	L
1	A	INPUT					✓	-0.01	0.01	
2	B	INPUT					✓	-0.01	0.01	
3	Y	OUTPUT					✓			1.

	Name	Type	Mapping

6. Since pin A is mapped to the physical pins1, 4, 9, and 12, enter 1, 4, 9, and 12 under S1, S2, S3, and S4, respectively.
7. Similarly, map the pins B and Y.

After entering the pins, the Package Pin page should appear as:



Logical Pins Selected : 1

	Name	Type	S1	S2	S3	S4	Sized	Input Load		O
								Low	High	L
1	A	INPUT	1	4	9	12	✓	-0.01	0.01	
2	B	INPUT	2	5	10	13	✓	-0.01	0.01	
3	Y	OUTPUT	3	6	8	11	✓			1.

Physical Pins Selected : 0

	Number	Mapping	
		Pin	Functio
1	1	A	S1
2	4	A	S2
3	9	A	S3
4	12	A	S4
5	2	B	S1
6	5	B	S2
7	10	B	S3
8	13	B	S4
9	3	Y	S1
10	6	Y	S2
11	8	Y	S3

Global Pins

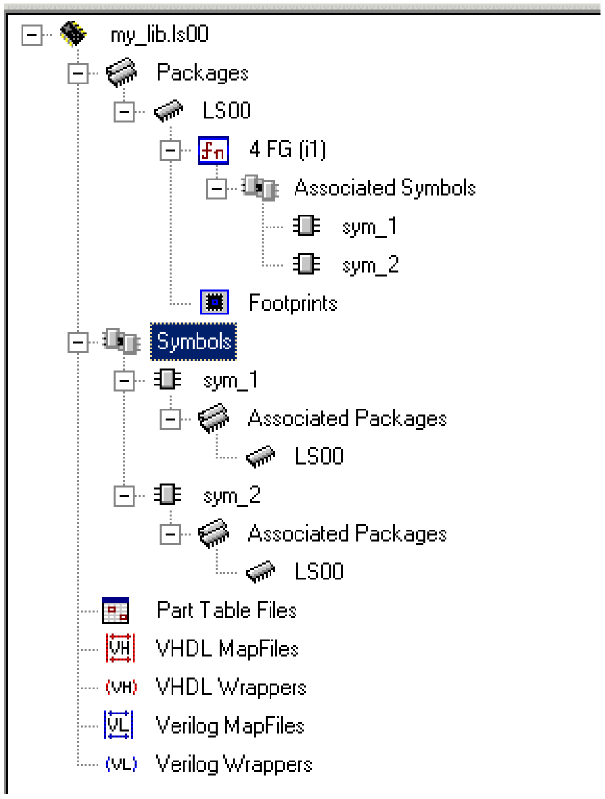
	Name	Type	Mapping

Next, you will create a `HAS_FIXED_SIZE` symbol.

Creating a HAS_FIXED_SIZE Symbol

1. Right-click on the *sym_1* entry under *Symbols* in the cell tree and choose *Copy*.
2. Right-click on the *Symbols* entry in the cell tree and choose *Paste*.

This creates a new symbol, *sym_2,* which is a copy of *sym_1*.

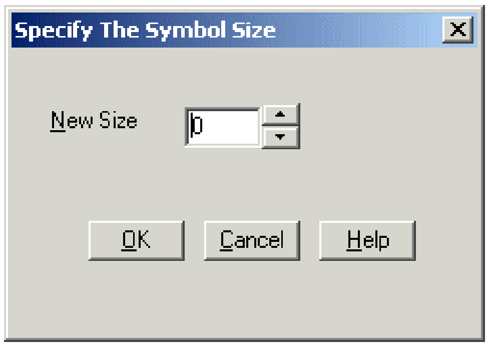


3. Click on *sym_2*.

sym_2 gets loaded in the Symbol Editor.

4. Click *Set Size*.

The Specify The Symbol Size dialog box appears.



New Size: 0

5. Enter 4 as the value of the `SIZE` property in the *New Size* field.

Note: The `SIZE` property value depends on the number of slots in which the pin is present. In this case, the value is 4 because the pin is present in 4 slots.

6. Save the part.

   The `HAS_FIXED_SIZE` property will be automatically added to the `symbol.css` file of *sym_2*.

   Note: The `HAS_FIXED_SIZE` property is not visible from within Symbol Editor.

Caution

Do not change the pin_name <0> pin for HAS_FIXED_SIZE symbols. This might lead to the deletion of all the bits of the pin. For example, renaming A<0> might lead to the deletion of all the bits of pin A.

Summary

In this chapter, you learned how to create sizeable and `HAS_FIXED_SIZE` symbols.

10

Modifying Packages

Objective

To become familiar with the steps involved in modifying packages.

In this chapter, you will learn to:

- Add pins to a package
- Add properties to a package
- Delete pins from a package
- Delete pins from all packages and symbols
- Modify pin types
- Move global pins to logical pins
- Move logical pins to global pins
- Modify the footprint information
- Add Package Pin properties
- Specify pin swappability

Overview

Often, you need to modify a package to meet changed requirements. Using Part Developer, part modifications can be done quickly and effectively. This chapter covers some of the common part-modification tasks. You will use the library part `mypart` in the `my_lib` library to perform the part-modification tasks.

The part has two packages, MYPART and MYPART_1, and two symbols, sym_1 and sym_2. The symbol sym_1 is associated with the package MYPART and sym_2 with the package MYPART_1.

Adding Pins to a Package

Add an input pin, `my_pin`, to the MYPART package.

1. Choose *File – Open – Cell.*

   The Open Cell dialog box appears.

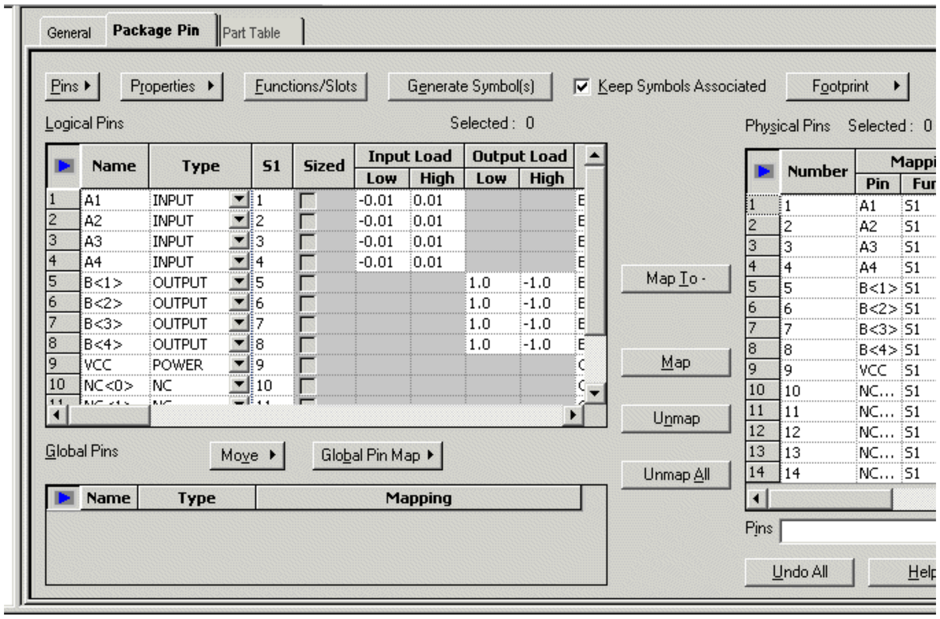
2. Select the *my_lib* library.

3. Select the cell *mypart* and click *OK*.

   The part `mypart` loads in the Cell Editor.

4. To select the MYPART package, click the *MYPART* entry under the *Packages* entry in the cell tree.

   MYPART gets loaded in the Package Editor.



[✓] Keep Symbols Associated

Logical Pins   Selected : 0

	Name	Type	S1	Sized	Input Load		Output Load		
					Low	High	Low	High	
1	A1	INPUT	1	☐	-0.01	0.01			E
2	A2	INPUT	2	☐	-0.01	0.01			E
3	A3	INPUT	3	☐	-0.01	0.01			E
4	A4	INPUT	4	☐	-0.01	0.01			E
5	B<1>	OUTPUT	5	☐			1.0	-1.0	E
6	B<2>	OUTPUT	6	☐			1.0	-1.0	E
7	B<3>	OUTPUT	7	☐			1.0	-1.0	E
8	B<4>	OUTPUT	8	☐			1.0	-1.0	E
9	VCC	POWER	9	☐					C
10	NC<0>	NC	10	☐					C

Physical Pins   Selected : 0

	Number	Mappi	
		Pin	Fur
1	1	A1	S1
2	2	A2	S1
3	3	A3	S1
4	4	A4	S1
5	5	B<1>	S1
6	6	B<2>	S1
7	7	B<3>	S1
8	8	B<4>	S1
9	9	VCC	S1
10	10	NC...	S1
11	11	NC...	S1
12	12	NC...	S1
13	13	NC...	S1
14	14	NC...	S1

Global Pins

	Name	Type	Mapping

5. Select the last row in the *Logical Pins* grid.

6. Press Ctrl + I.

   A blank row appears in the *Logical Pins* grid.

7. Enter `my_pin` in the *Name* column and select *INPUT* from the *Type* drop-down list.

8. Enter the physical pin number 15 under the column S1 and click on the Physical Pins grid.

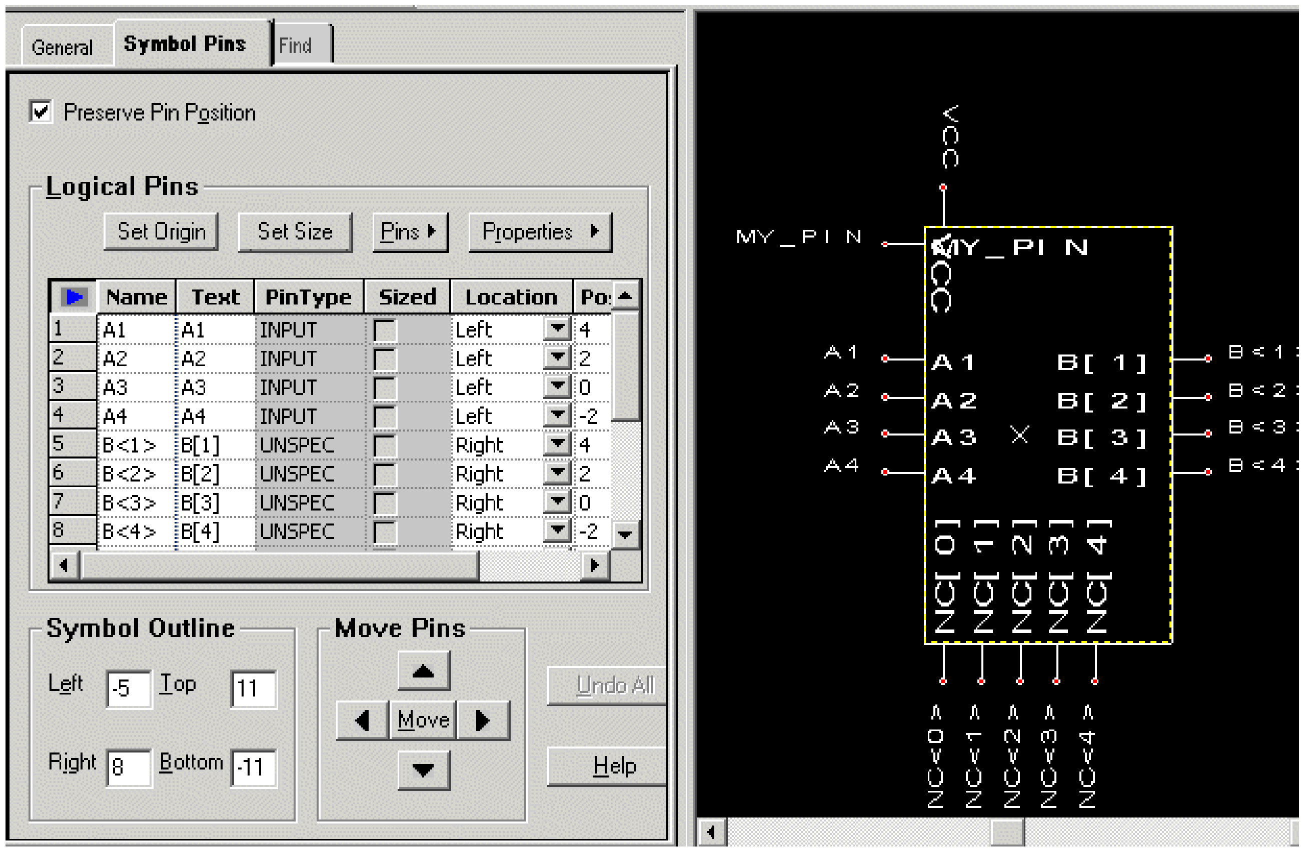
9. Save the part.

Important

The symbol `sym_1` is associated with the package `MYPART`. Therefore, when a pin is added to the package, the symbol pin list will also have to be updated to make sure that the symbol is in sync with the package. If the *Keep Symbols Associated* check box is selected, Part Developer automatically updates the symbol pin list whenever the package pin list is modified.

10. The *Keep Symbols Associated* check box is selected by default. To check that `my_pin` has been added to the symbol as well, select the symbol *sym_1*.

    The symbol gets loaded in the Symbol Viewer. Notice that the pin appears in the top-left corner of the symbol outline.



[✓] Preserve Pin Position

	Name	Text	PinType	Sized	Location	Po:
1	A1	A1	INPUT	☐	Left	4
2	A2	A2	INPUT	☐	Left	2
3	A3	A3	INPUT	☐	Left	0
4	A4	A4	INPUT	☐	Left	-2
5	B<1>	B[1]	UNSPEC	☐	Right	4
6	B<2>	B[2]	UNSPEC	☐	Right	2
7	B<3>	B[3]	UNSPEC	☐	Right	0
8	B<4>	B[4]	UNSPEC	☐	Right	-2

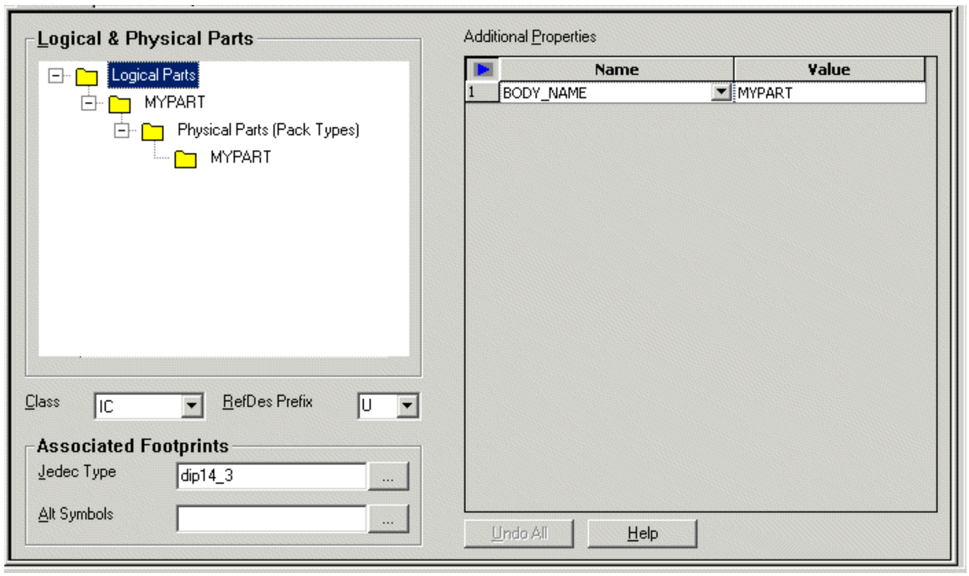
Symbol Outline: Left -5, Top 11, Right 8, Bottom -11

Adding Properties to a Package

Add the property `library_modified_date` with the value `?` to the MYPART package.

1. Select the MYPART package.
2. Click on the *General* tab.

   The General page appears.



Class: IC
RefDes Prefix: U
Jedec Type: dip14_3

	Name	Value
1	BODY_NAME	MYPART

3. Select the `BODY_NAME` property and press `Ctrl` + `I`.

   A blank row gets created in the *Additional Properties* grid.
4. Enter `LIBRARY_MODIFIED_DATE` in the *Name* column and `?` in the *Value* column.
5. Save the part.

Deleting Pins from a Package

Delete the pin `my_pin` from the `MYPART` package.

1. Select the *MYPART* package.
2. Right-click on the row in the *Logical Pins* grid that contains the pin `MY_PIN` and select *Delete Selected Rows*.

   The pin is deleted from the package.

Important

Because the *Keep Symbols Associated* option is selected, the pin will be automatically deleted from sym_1 as well.

3. Save the part.

   On save, a warning is generated stating that pin MY_PIN is not present in any package or symbol. This is because when a pin is deleted from a package or symbol without using the *Global Delete* option or from the Add Pin dialog box, the pin is not deleted from the Part Developer database. It is stored so that it can be added to a package or symbol if required. To delete the pin completely:

   a. Open the Add Pin dialog box.

   b. Right-click on the pin and select *Delete Selected Rows.*

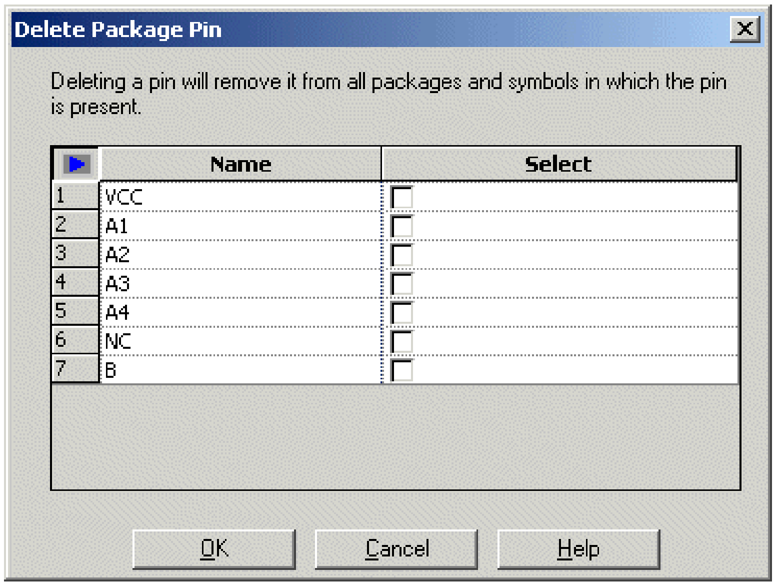
   c. Click *OK*.

Deleting Pins from All Packages and Symbols

Delete the input pin A1 from all the packages and symbols.

1. Select any package.
2. Choose *Pins – Global Delete*.

   The Delete Package Pin dialog box appears.


Delete Package Pin

Deleting a pin will remove it from all packages and symbols in which the pin is present.

	Name	Select
1	VCC	☐
2	A1	☐
3	A2	☐
4	A3	☐
5	A4	☐
6	NC	☐
7	B	☐

3. Select the pin A1 and click *OK*.

   Pin A1 is deleted from all the packages and symbols.

4. Save the part.

Important

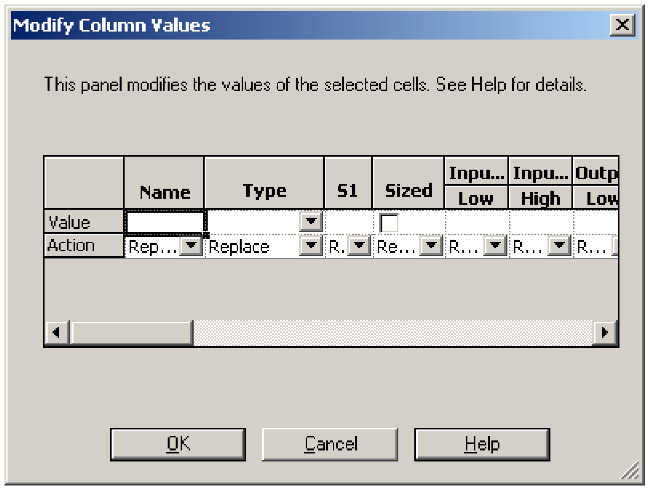
There is no undo action available for pin deletion operations done using the *Delete Package Pin* option. Pins deleted using the *Delete Package Pin* option are deleted from all packages and symbols.

Modifying Pin Types

Modify the input pins in the MYPART package to BIDIR pins.

1. Select the package *MYPART*.
2. Select the input pins A2, A3, and A4 in the *Logical Pins* grid.
3. Right-click on the selection and choose *Modify Values* from the pop-up menu.

   The Modify Values dialog box appears.

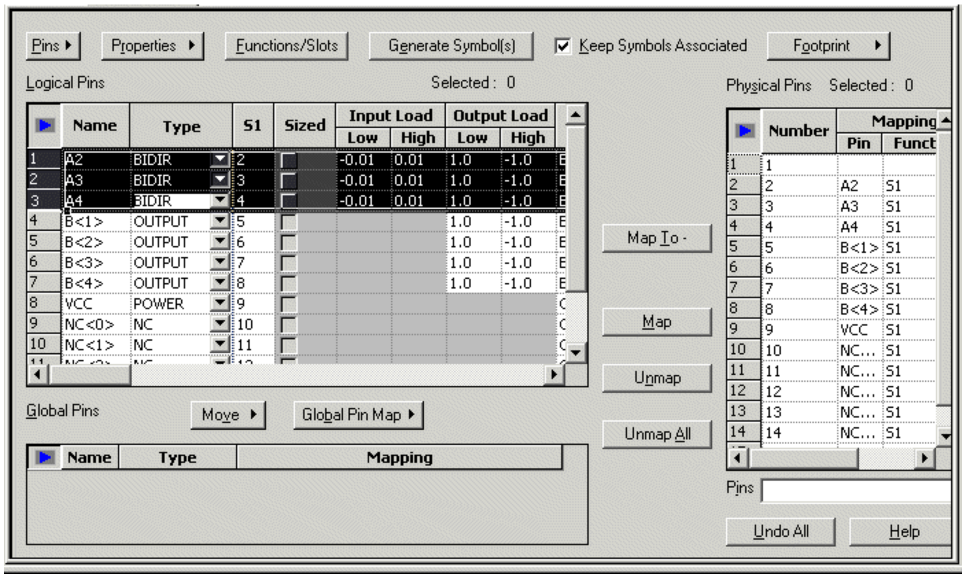


This panel modifies the values of the selected cells. See Help for details.

	Name	Type	S1	Sized	Inpu... Low	Inpu... High	Outp Low
Value				☐			
Action	Rep...	Replace	R.	Re...	R...	R...	R...

4. Select *BIDIR* from the *Type* drop-down list.
5. Click *OK*.

The pin types are modified to BIDIR.



	Name	Type	S1	Sized	Input Load		Output Load		
					Low	High	Low	High	
1	A2	BIDIR	2	☐	-0.01	0.01	1.0	-1.0	E
2	A3	BIDIR	3	☐	-0.01	0.01	1.0	-1.0	E
3	A4	BIDIR	4	☐	-0.01	0.01	1.0	-1.0	E
4	B<1>	OUTPUT	5	☐			1.0	-1.0	E
5	B<2>	OUTPUT	6	☐			1.0	-1.0	E
6	B<3>	OUTPUT	7	☐			1.0	-1.0	E
7	B<4>	OUTPUT	8	☐			1.0	-1.0	E
8	VCC	POWER	9	☐					C
9	NC<0>	NC	10	☐					C
10	NC<1>	NC	11	☐					C

Name	Type	Mapping

	Number	Mapping	
		Pin	Funct
1	1		
2	2	A2	S1
3	3	A3	S1
4	4	A4	S1
5	5	B<1>	S1
6	6	B<2>	S1
7	7	B<3>	S1
8	8	B<4>	S1
9	9	VCC	S1
10	10	NC...	S1
11	11	NC...	S1
12	12	NC...	S1
13	13	NC...	S1
14	14	NC...	S1

6. Save the part.

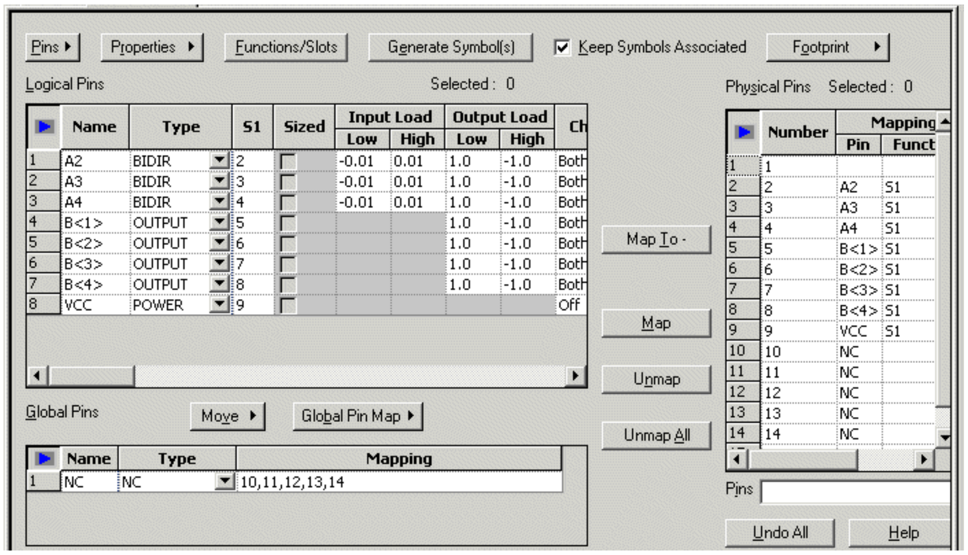
   A warning message is generated stating that a list of specified pins have different pin types in different packages.

Moving Logical Pins to Global Pins

Move the `NC` pins in the MYPART package to the *Global Pins* grid.

1. Select the MYPART package.
2. Select all the `NC` pins in the *Logical Pins* grid.
3. Choose *Move – Logical Pins to Global*.

The NC pins move to the *Global Pins* grid.



[✓] Keep Symbols Associated

Logical Pins Selected : 0

	Name	Type	S1	Sized	Input Load		Output Load		Ch
					Low	High	Low	High	
1	A2	BIDIR	2	☐	-0.01	0.01	1.0	-1.0	Both
2	A3	BIDIR	3	☐	-0.01	0.01	1.0	-1.0	Both
3	A4	BIDIR	4	☐	-0.01	0.01	1.0	-1.0	Both
4	B<1>	OUTPUT	5	☐			1.0	-1.0	Both
5	B<2>	OUTPUT	6	☐			1.0	-1.0	Both
6	B<3>	OUTPUT	7	☐			1.0	-1.0	Both
7	B<4>	OUTPUT	8	☐			1.0	-1.0	Both
8	VCC	POWER	9	☐					Off

Global Pins

	Name	Type	Mapping
1	NC	NC	10,11,12,13,14

Physical Pins Selected : 0

	Number	Mapping	
		Pin	Funct
1	1		
2	2	A2	S1
3	3	A3	S1
4	4	A4	S1
5	5	B<1>	S1
6	6	B<2>	S1
7	7	B<3>	S1
8	8	B<4>	S1
9	9	VCC	S1
10	10	NC	
11	11	NC	
12	12	NC	
13	13	NC	
14	14	NC	

Pins

Important

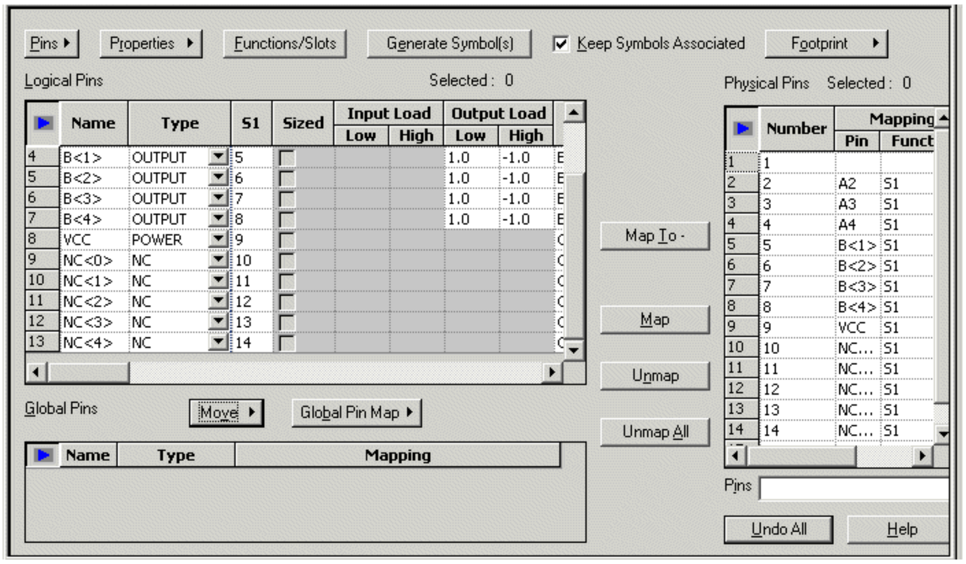
The NC pins, which appeared as bits of vector pins, are collapsed and placed as a single pin in the *Global Pins* grid. The *Mapping* column displays the physical pins that are mapped to the NC pins.

Moving Global Pins to Logical Pins

Move the NC pins in the MYPART package to the *Logical Pins* grid.

1. Select the MYPART package.
2. Select the NC pins in the *Global Pins* grid.
3. Choose *Move – Global Pins to Logical*.

The `NC` pins move to the *Logical Pins* grid.



Logical Pins Selected : 0

	Name	Type	S1	Sized	Input Load		Output Load		
					Low	High	Low	High	
4	B<1>	OUTPUT	5	☐			1.0	-1.0	E
5	B<2>	OUTPUT	6	☐			1.0	-1.0	E
6	B<3>	OUTPUT	7	☐			1.0	-1.0	E
7	B<4>	OUTPUT	8	☐			1.0	-1.0	E
8	VCC	POWER	9	☐					C
9	NC<0>	NC	10	☐					C
10	NC<1>	NC	11	☐					C
11	NC<2>	NC	12	☐					C
12	NC<3>	NC	13	☐					C
13	NC<4>	NC	14	☐					C

Physical Pins Selected : 0

	Number	Mapping	
		Pin	Funct
1	1		
2	2	A2	S1
3	3	A3	S1
4	4	A4	S1
5	5	B<1>	S1
6	6	B<2>	S1
7	7	B<3>	S1
8	8	B<4>	S1
9	9	VCC	S1
10	10	NC...	S1
11	11	NC...	S1
12	12	NC...	S1
13	13	NC...	S1
14	14	NC...	S1

Global Pins

	Name	Type	Mapping

Pins

Important

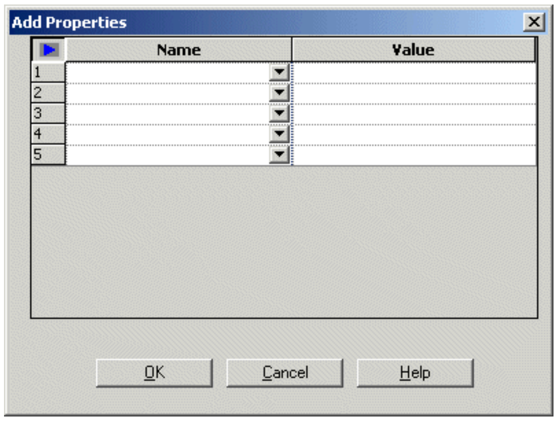
The `NC` pins, which appeared as a single pin in the *Global Pins* grid, appear as bits of a vector pin in the *Logical Pins* grid.

Adding Package Pin Properties

Add the package pin property `my_pin_property` with value `val 1` to the pins A2 and A3 of the MYPART package.

1. Select the package MYPART.
2. Choose *Properties – Add*.

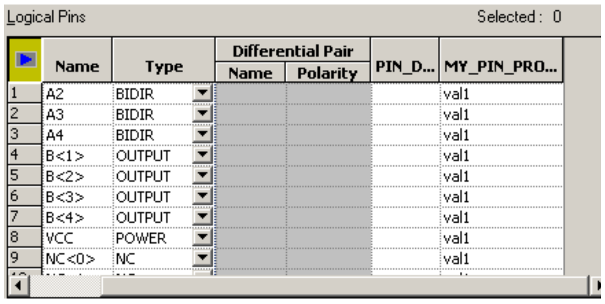
The Add Properties dialog box appears.



3. To specify the properties, enter `my_pin_property` in the *Name* column and `val1` in the *Value* column.
4. Click *OK*.

   By default, the property gets added to all the pins. To remove the property from a specific pin, you need to delete the value of the property for that pin.
5. To hide the unnecessary columns, right-click anywhere in the *Logical Pins* grid and select *Hide Load Cols*.

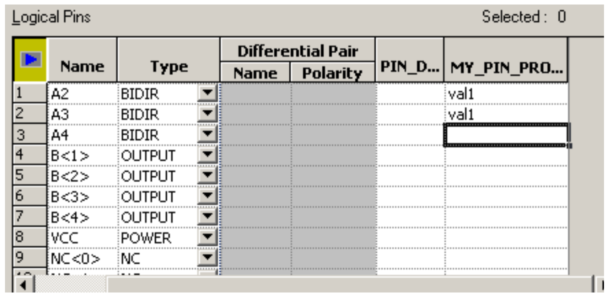
   The columns are hidden.


Logical Pins　　Selected : 0

	Name	Type	Differential Pair		PIN_D...	MY_PIN_PRO...
			Name	Polarity		
1	A2	BIDIR				val1
2	A3	BIDIR				val1
3	A4	BIDIR				val1
4	B<1>	OUTPUT				val1
5	B<2>	OUTPUT				val1
6	B<3>	OUTPUT				val1
7	B<4>	OUTPUT				val1
8	VCC	POWER				val1
9	NC<0>	NC				val1

6. To delete the pin property from pins A4 to NC<4>, delete the property values.

After the property values are deleted, the *Logical Pins* grid should appear as follows:


Logical Pins Selected : 0

	Name	Type	Differential Pair Name	Differential Pair Polarity	PIN_D...	MY_PIN_PRO...
1	A2	BIDIR				val1
2	A3	BIDIR				val1
3	A4	BIDIR				
4	B<1>	OUTPUT				
5	B<2>	OUTPUT				
6	B<3>	OUTPUT				
7	B<4>	OUTPUT				
8	VCC	POWER				
9	NC<0>	NC				

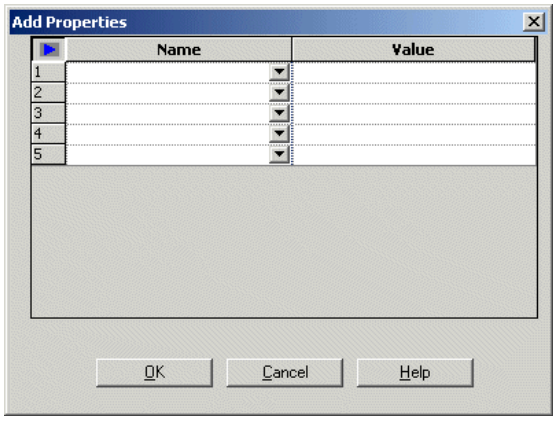
Specifying Pin Swappability

Pin swappability is determined by the `PIN_GROUP` pin property. The pins that have the same values for this property are considered swappable.

Make the pins A2 and A4 of the MYPART package swappable.

1. Select the MYPART package.
2. Choose *Properties – Add.*

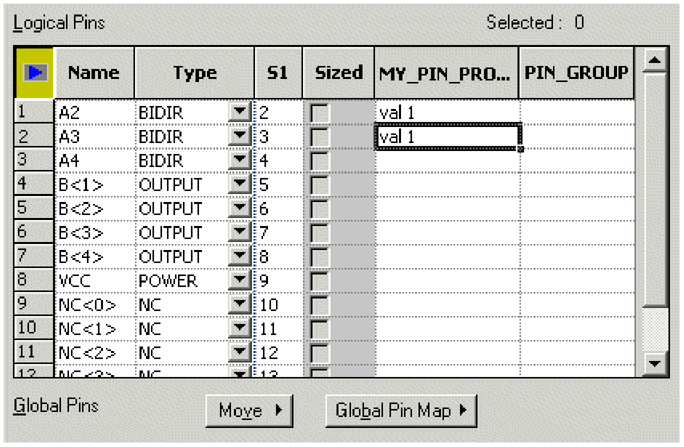
   The Add Properties dialog box appears.



3. Select *PIN_GROUP* from the *Name* drop-down list.
4. Click *OK*.

   By default, the property gets added to all the pins.
5. To hide the unnecessary columns, right-click anywhere in *Logical Pins* grid and select *Hide Load Cols*.

The columns are hidden.



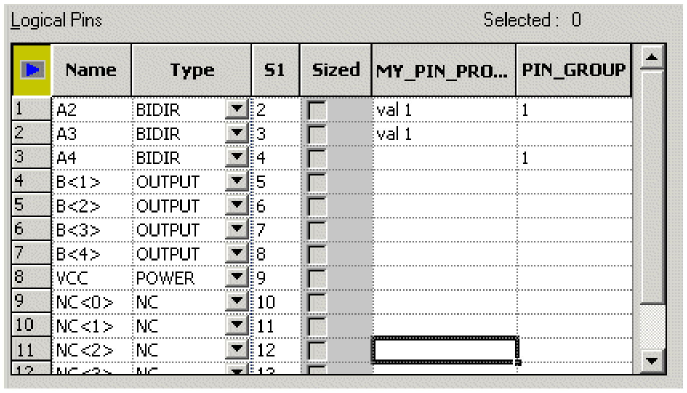
Logical Pins | Selected : 0

	Name	Type	51	Sized	MY_PIN_PRO...	PIN_GROUP
1	A2	BIDIR	2	☐	val 1	
2	A3	BIDIR	3	☐	val 1	
3	A4	BIDIR	4	☐		
4	B<1>	OUTPUT	5	☐		
5	B<2>	OUTPUT	6	☐		
6	B<3>	OUTPUT	7	☐		
7	B<4>	OUTPUT	8	☐		
8	VCC	POWER	9	☐		
9	NC<0>	NC	10	☐		
10	NC<1>	NC	11	☐		
11	NC<2>	NC	12	☐		

Global Pins | Move | Global Pin Map

6. To make pins A2 and A4 swappable, assign the value `1` for the `PIN_GROUP` property for pins A2 and A4.

   The *Logical Pins* grid should appear as follows:



Logical Pins | Selected : 0

	Name	Type	51	Sized	MY_PIN_PRO...	PIN_GROUP
1	A2	BIDIR	2	☐	val 1	1
2	A3	BIDIR	3	☐	val 1	
3	A4	BIDIR	4	☐		1
4	B<1>	OUTPUT	5	☐		
5	B<2>	OUTPUT	6	☐		
6	B<3>	OUTPUT	7	☐		
7	B<4>	OUTPUT	8	☐		
8	VCC	POWER	9	☐		
9	NC<0>	NC	10	☐		
10	NC<1>	NC	11	☐		
11	NC<2>	NC	12	☐		

7. Save the part.

Summary

In this chapter, you learned how to modify packages.

11

Modifying Symbols

Objective

To become familiar with the steps involved in modifying symbols.

In this chapter, you will learn to:

- Add a pin to a symbol.
- Add a property to a symbol.
- Add a symbol pin property.
- Modify a symbol pin property.
- Delete a pin from a symbol.
- Add symbol text.
- Move a symbol pin.
- Modify a symbol outline.
- Expand and collapse vector pins.
- Modify the PIN_TEXT property.

Overview

Often, you may need to modify a symbol to meet changed requirements. Using Part Developer, symbol modifications can be done quickly and effectively. This chapter will cover some of the common part modification tasks. You will use the library part `mypart` in the `my_lib` library to perform part modification tasks.

The part has two packages, `MYPART` and `MYPART_1`, and two symbols, sym_1 and `sym_2`. The symbols `sym_1` and `sym_2` are associated with the packages `MYPART` and `MYPART_1`, respectively.

Adding Pins to a Symbol

Add an input pin, `A1`, to `sym_1`.

1. Select the symbol *sym_1* in the cell tree.

   The symbol is loaded in the Symbol Editor.

2. Choose *Pins – Add*.

   The Add Pin dialog box appears.

3. Ensure that the *Scalar* option is selected and enter `A1` in the *Prefix* field.

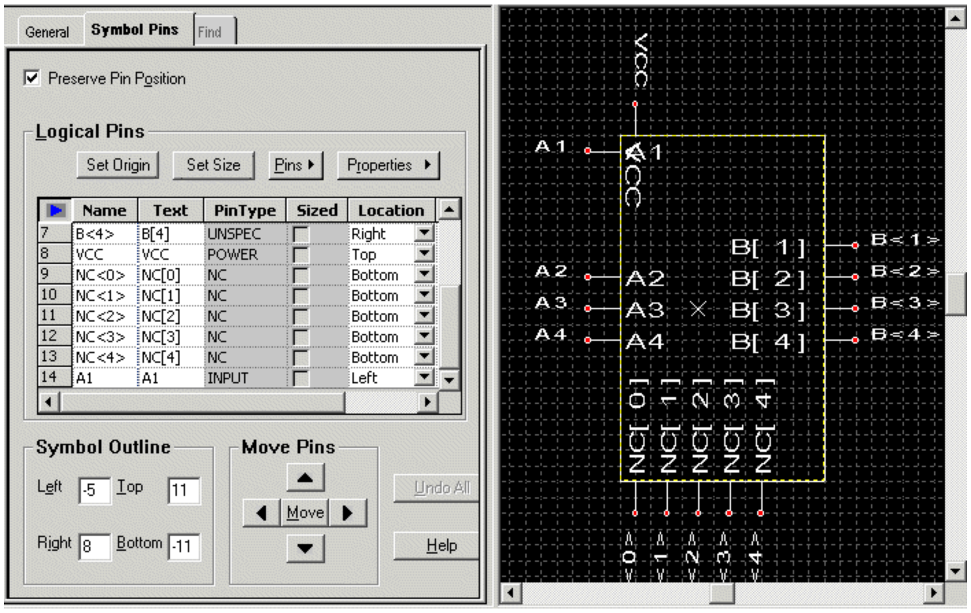
4. Select *INPUT* from the *Type* drop-down list.

5. Click *Add*.

   The input pin A1 is added to the pin grid.

6. Click *OK.*

   The input pin A1 gets added to the symbol.



[✓] Preserve Pin Position

	Name	Text	PinType	Sized	Location
7	B<4>	B[4]	UNSPEC		Right
8	VCC	VCC	POWER		Top
9	NC<0>	NC[0]	NC		Bottom
10	NC<1>	NC[1]	NC		Bottom
11	NC<2>	NC[2]	NC		Bottom
12	NC<3>	NC[3]	NC		Bottom
13	NC<4>	NC[4]	NC		Bottom
14	A1	A1	INPUT		Left

Left -5 Top 11
Right 8 Bottom -11

Caution

Changes to the symbol pin list results in breaking symbol-package associations. You will need to update the package pin list to ensure that the symbol is usable in the flow. You can use Interface Comparator to synchronize the pin lists of packages and symbols. See Interface Comparator on page 147 for more details on interface comparison.

Important

The *Preserve Pin Position* option determines whether the pin positions and their associated properties and pin text will be preserved when the symbol outline is modified. If the option is not selected, the pin positions, their properties, and the pin text are adjusted dynamically when the symbol outline changes. Otherwise, the pin positions, their properties and pin text do not change with the change in the symbol outline.

When a new symbol is created, this option is deselected by default. This allows Part Developer to dynamically shift the pins on the symbol outline as and when new pins are added.

Note: Part Developer extends the symbol outline when new pins are added. However, no symbol outline contraction is done when symbol pins are deleted. You need to manually resize the symbol outline. When an existing symbol is loaded, this option is selected by default. This is done to ensure that graphics associated with a symbol pin are not destroyed by the automatic movement of pins. This also ensures that in the case of horizontal or vertical extension of the symbol outline due to the addition of pins, the pins at the top or bottom or on the left or right of the symbol remain bounded to the symbol outline.

Adding Properties to a Symbol

Add the property `my_symbol_prop` with value `sym_val` in `sym_1`.

1. Select *sym_1* in the cell tree.

   The symbol is loaded in the Symbol Editor.

2. Click on the *General* tab.

3. Right-click on the Properties grid and select *Insert Row After*.

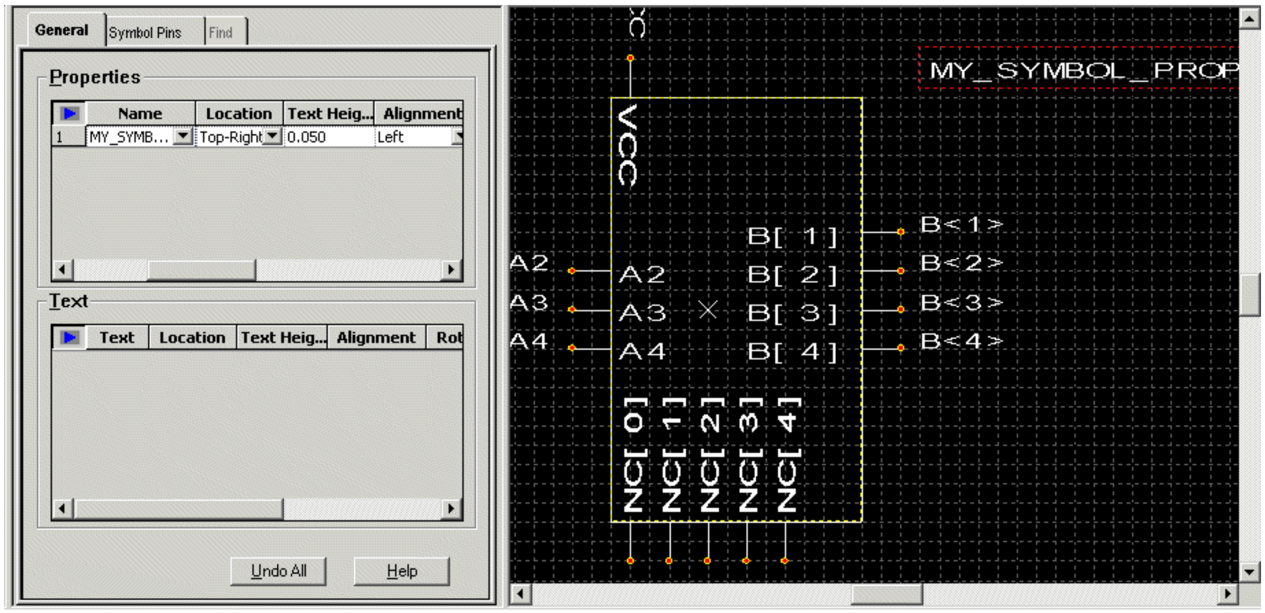
   A blank row is added to the *Properties* grid.

4. Enter `my_symbol_prop` in the *Name* column and `sym_val` in the *Value* column.

Next, you need to determine the appearance of the property and its value on the symbol.

5. To make both the property and its value visible, select *Both* from the *Visibility* drop-down list.
6. To make the property appear in the top-right corner of the symbol, select *Top-Right* from the *Location* drop-down field.

   Similarly, you can specify other values, such as rotation, alignment, and so on. Notice that the changes you make are immediately visible on the Symbol Editor canvas.



	Name	Location	Text Heig...	Alignment
1	MY_SYMB...	Top-Right	0.050	Left

	Text	Location	Text Heig...	Alignment	Rot

Adding Symbol Pin Properties

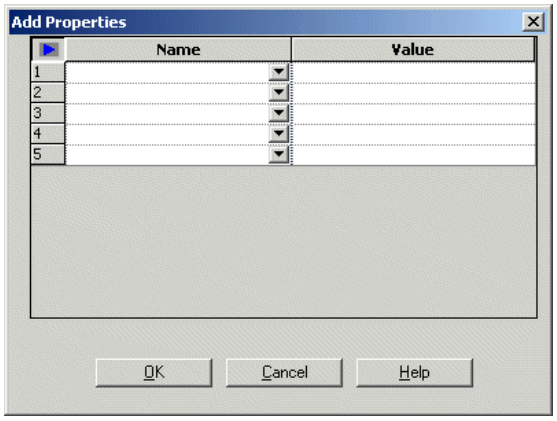
Add the symbol pin property `VHDL_TYPE` with the value `lN` to pins `A2` and `A3`.

1. Select *sym_1* in the cell tree.

   The symbol is loaded in the Symbol Editor.

2. Click on the *Symbol Pins* tab.
3. Select *Properties – Add*.

The Add Properties dialog box appears.



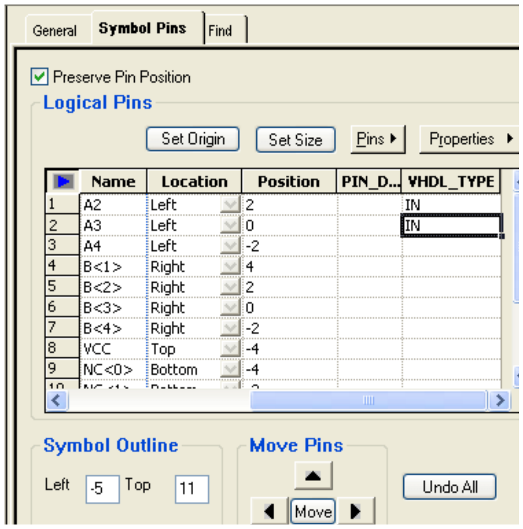
4. Enter `VHDL_TYPE` in the *Name* column and click *OK*.

   The `VHDL_TYPE` property is added. By default, the property will get added to only those pins that have a value. A null value results in the property being omitted from the pins.

5. To add the value of IN to the pin A2, enter `IN` in the A2 row in the `VHDL_TYPE` column.

6. Similarly, specify `IN` as the `VHDL_TYPE` property value for the pin A3.

   The *Logical Pins* grid should appear as follows:



	Name	Location	Position	PIN_D...	VHDL_TYPE
1	A2	Left	2		IN
2	A3	Left	0		IN
3	A4	Left	-2		
4	B<1>	Right	4		
5	B<2>	Right	2		
6	B<3>	Right	0		
7	B<4>	Right	-2		
8	VCC	Top	-4		
9	NC<0>	Bottom	-4		

Modifying Symbol Pin Properties

Modify the `VHDL_TYPE` symbol pin property to make it visible on the symbol pins.

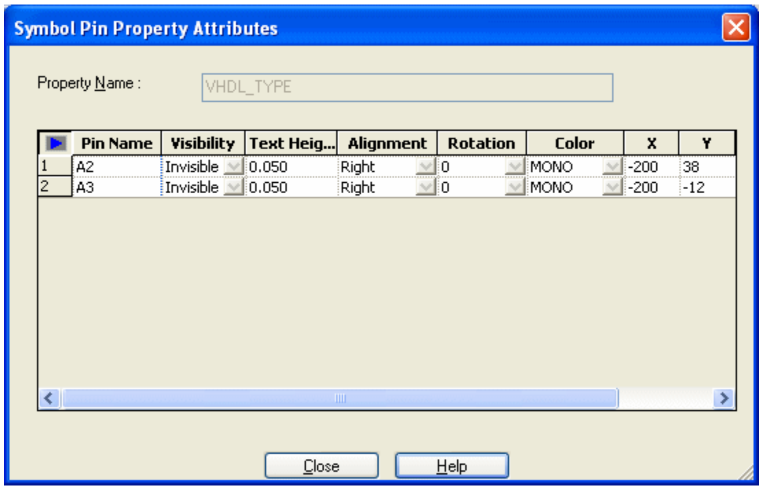
1. Select *sym_1* in the cell tree.

   The symbol gets loaded in the Symbol Editor.

2. Select the `VHDL_TYPE` column in the *Logical Pins* grid.

3. Select *Properties – Attributes.*

   The Symbol Pin Property Attributes dialog box appears.



Property Name : VHDL_TYPE

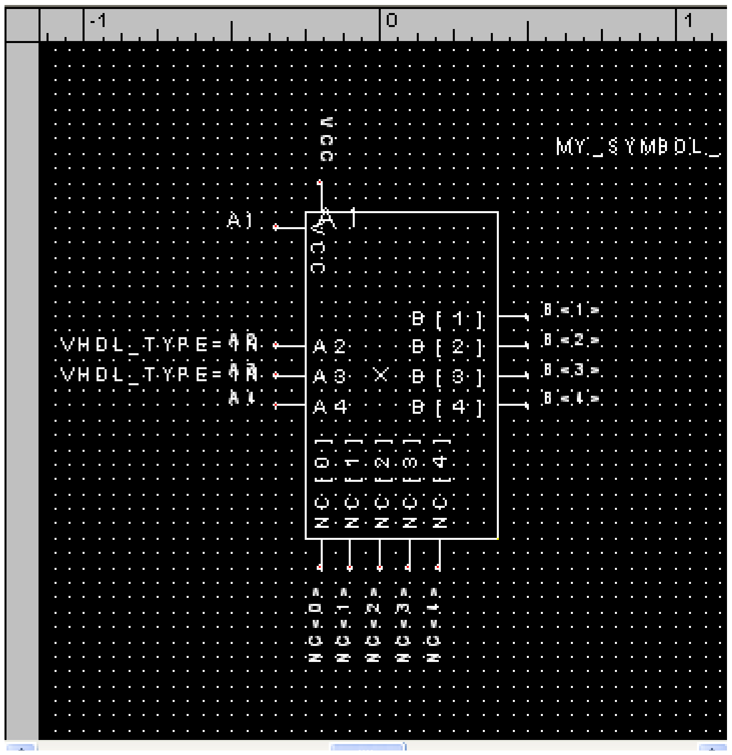
	Pin Name	Visibility	Text Heig...	Alignment	Rotation	Color	X	Y
1	A2	Invisible	0.050	Right	0	MONO	-200	38
2	A3	Invisible	0.050	Right	0	MONO	-200	-12

4. For pins A2 and A3, select *Both* from the *Visibility* drop-down list box.

   Similarly, you can modify the other attributes through this dialog box.

5. Click *Close*.

The symbol pin properties appear next to the pins A2 and A3.



Deleting Pins from a Symbol

Delete the pin A2 from sym_2.

1. Select *sym_2* in the cell tree.

   The symbol gets loaded in the Symbol Viewer.

2. Right-click on the pin A2 and select *Delete Selected Rows*.

   The pin gets deleted from sym_2.

Adding Symbol Text

Add the text *CADENCE* to sym_1.

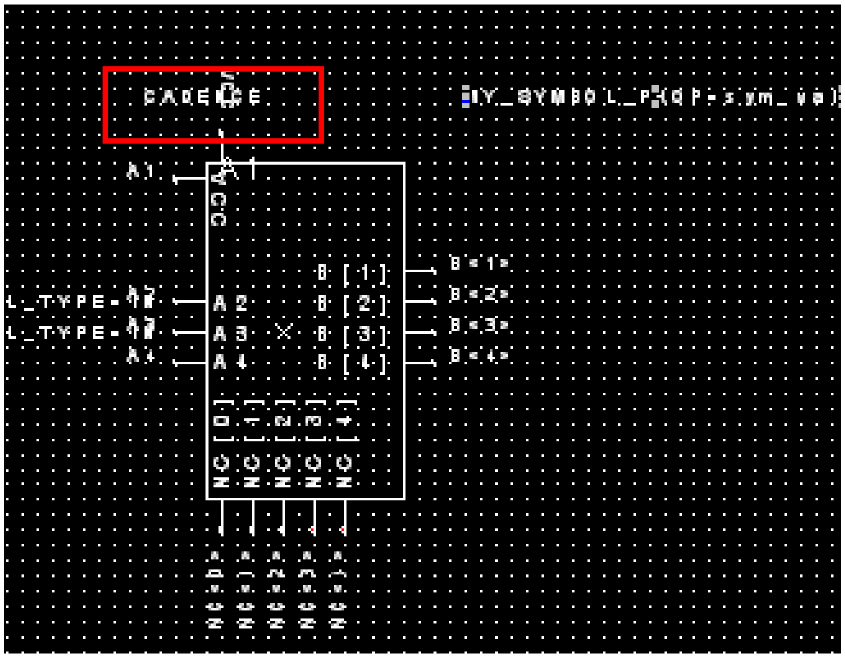
1. Select *sym_1* in the cell tree.

   The symbol gets loaded in the Symbol Viewer.

2. Click on the *General* tab.
3. Right-click on the Text grid and select *Insert Row After*.

   A blank row is inserted in the *Text* grid.
4. Enter *CADENCE* in the text column.
5. To make the text appear in the top-left location of the symbol, select *Top-Left* from the *Location* drop-down box.

   The symbol text appears in the top-left corner of the symbol.



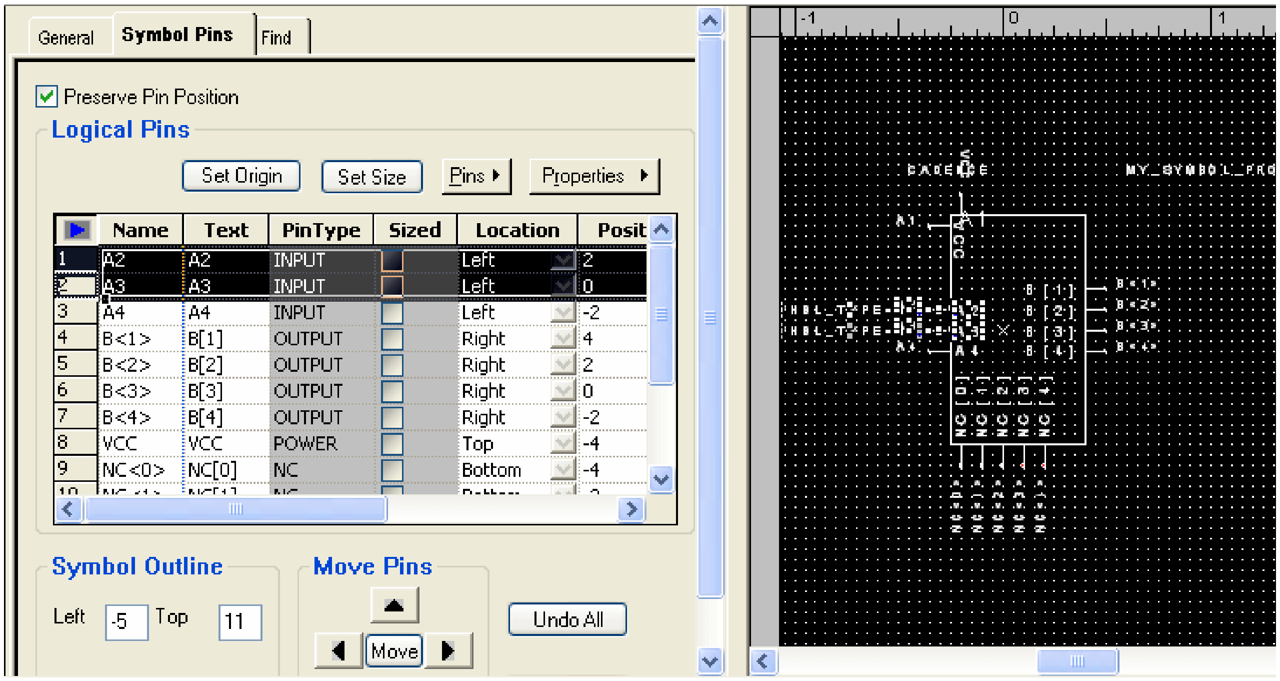
Moving Symbol Pins

Move the pins A2 and A3 by 1 grid to the left and pin A4 by 4 grids to the right.

1. Select *sym_1* in the cell tree.

   The symbol gets loaded in the Symbol Editor.
2. Select the pins A2 and A3.

The Symbol Editor shows the selected pins.



☑ Preserve Pin Position

	Name	Text	PinType	Sized	Location	Posit
1	A2	A2	INPUT	☐	Left	2
2	A3	A3	INPUT	☐	Left	0
3	A4	A4	INPUT	☐	Left	-2
4	B<1>	B[1]	OUTPUT	☐	Right	4
5	B<2>	B[2]	OUTPUT	☐	Right	2
6	B<3>	B[3]	OUTPUT	☐	Right	0
7	B<4>	B[4]	OUTPUT	☐	Right	-2
8	VCC	VCC	POWER	☐	Top	-4
9	NC<0>	NC[0]	NC	☐	Bottom	-4

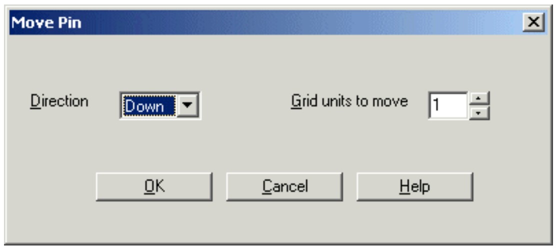
Left -5 Top 11

3. To move the pins by one grid to the left, click the left arrow button in the Move Pins grid.

   The selected pins move to the left by one grid.

4. Next, select pin A4.
5. To move pins by more than one grid at a time, click *Move*.

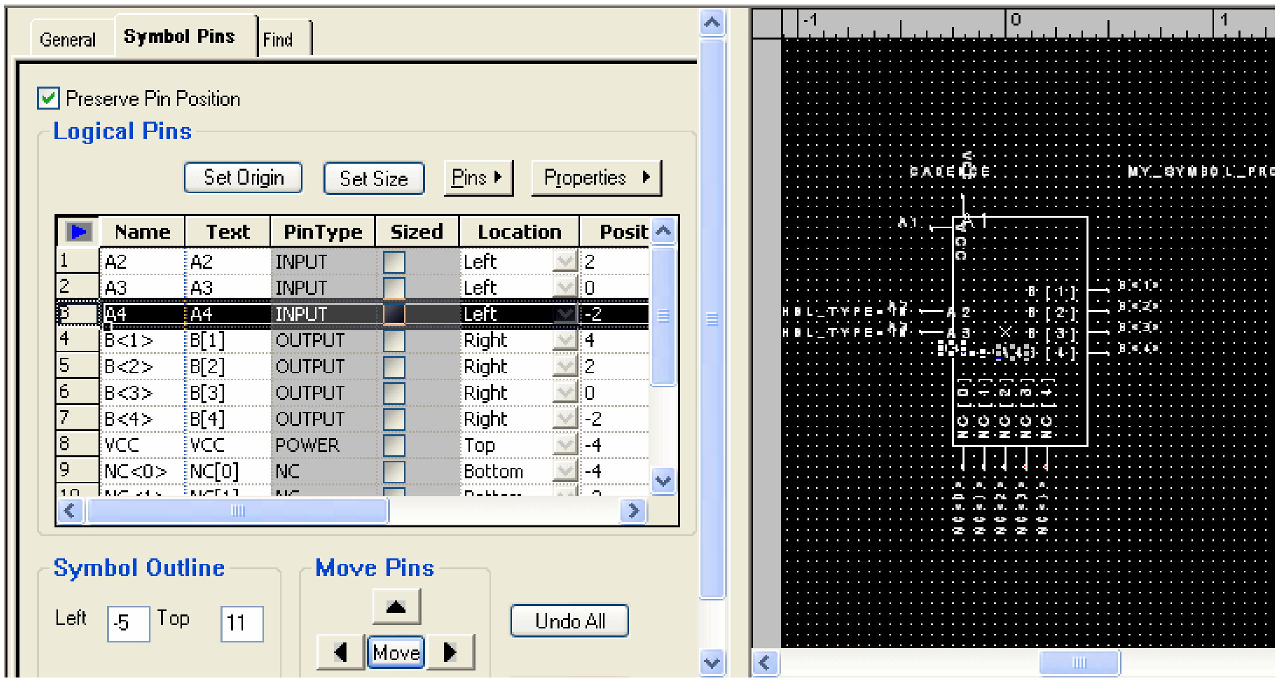
   The Move Pin dialog box appears.



Direction: Down
Grid units to move: 1

6. Select *Right* from the *Direction* drop-down list box.
7. Enter 4 in the *Grid units to move* field.
8. Click *OK*.

The pin A4 moves to the right by 4 grid units.



☑ Preserve Pin Position

	Name	Text	PinType	Sized	Location	Posit
1	A2	A2	INPUT	☐	Left	2
2	A3	A3	INPUT	☐	Left	0
3	A4	A4	INPUT	☐	Left	-2
4	B<1>	B[1]	OUTPUT	☐	Right	4
5	B<2>	B[2]	OUTPUT	☐	Right	2
6	B<3>	B[3]	OUTPUT	☐	Right	0
7	B<4>	B[4]	OUTPUT	☐	Right	-2
8	VCC	VCC	POWER	☐	Top	-4
9	NC<0>	NC[0]	NC	☐	Bottom	-4

Left -5 Top 11

Modifying Symbol Outline

Increase the left and right outline of sym_2 by 5 grids.

1. Select sym_2 in the cell tree.

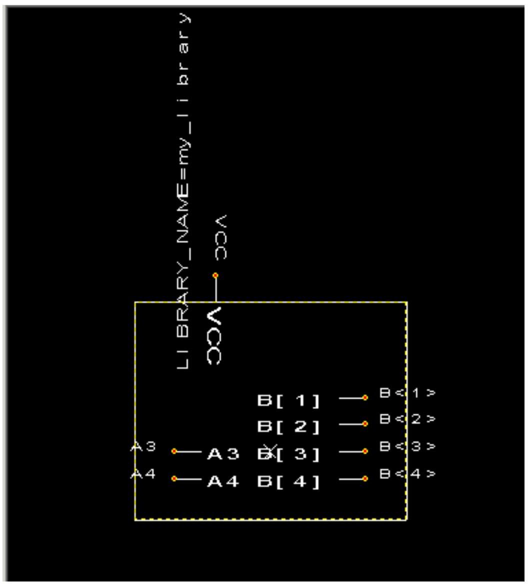
   The symbol loads in the Symbol Editor.

2. Enter the value `-10` in the *Left* field in the *Symbol Outline* group box.

3. Enter the value `10` in the *Right* field in the *Symbol Outline* group box.

   The symbol outline changes as specified. Note that the symbol pin positions are not changing automatically with the change in the symbol outline. This is because the

Preserve Pin Position option is selected. If you want to move the pin automatically with the symbol outline, deselect the *Preserve Pin Position* option.



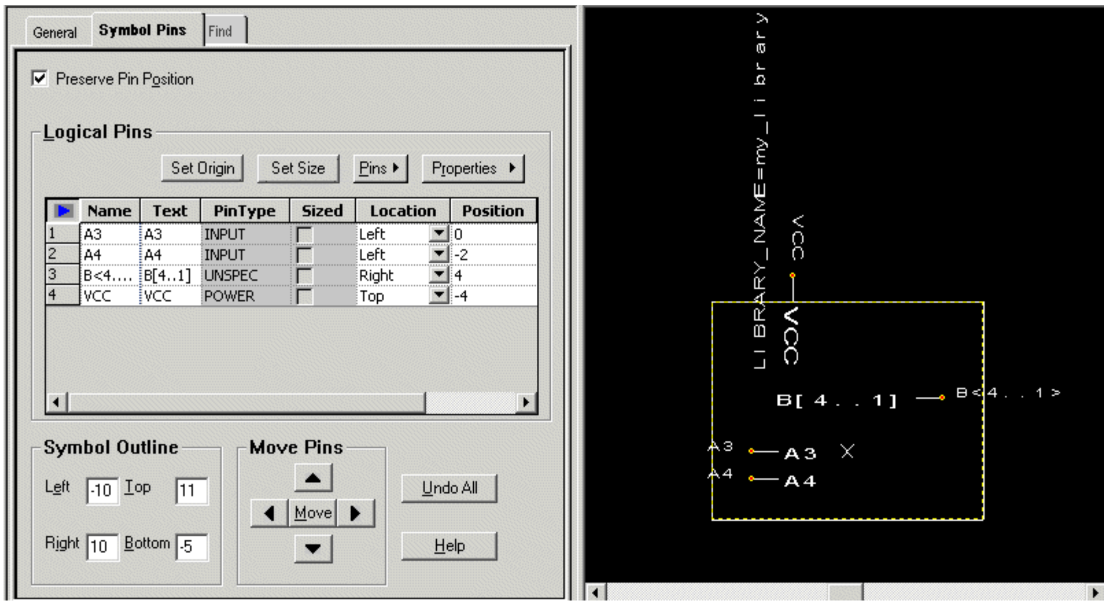
Expanding and Collapsing Vector Pins

Collapse the bits of the vector pin B.

1. Select all the bits of the vector pin B.
2. Right-click on the selection and select *Collapse*.

   Part Developer gives an error message saying that bit-specific properties might get lost on collapsing.
3. Click *Yes*.

The bits collapse to form a vector pin.



Preserve Pin Position ✓

Logical Pins

	Name	Text	PinType	Sized	Location	Position
1	A3	A3	INPUT	☐	Left	0
2	A4	A4	INPUT	☐	Left	-2
3	B<4....	B[4..1]	UNSPEC	☐	Right	4
4	VCC	VCC	POWER	☐	Top	-4

Symbol Outline

Left -10 Top 11

Right 10 Bottom -5

Move Pins

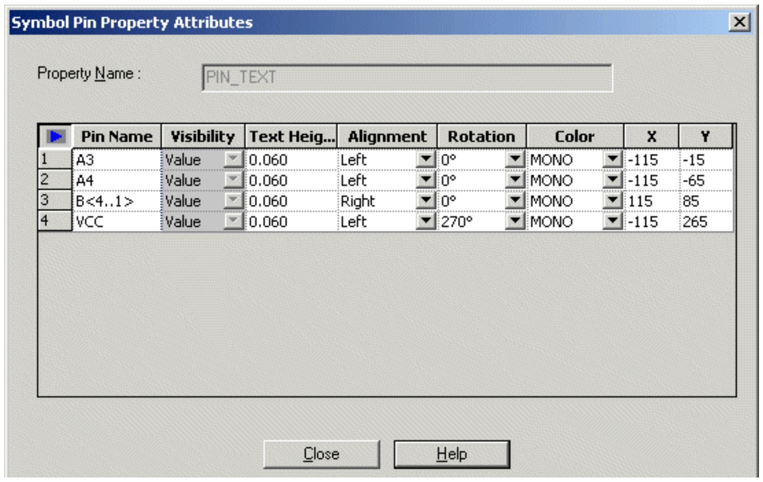
Modifying the PIN_TEXT Property

The value in the *Text* column shows the value of the PIN_TEXT property for the symbol pins. To change the values, simply edit the value for a particular pin.

To change the display attributes of the PIN_TEXT property:

1. Select *Pin – Pin Text Attributes*.

   The Symbol Pin Property Attributes dialog box appears.



Property Name : PIN_TEXT

	Pin Name	Visibility	Text Heig...	Alignment	Rotation	Color	X	Y
1	A3	Value	0.060	Left	0°	MONO	-115	-15
2	A4	Value	0.060	Left	0°	MONO	-115	-65
3	B<4..1>	Value	0.060	Right	0°	MONO	115	85
4	VCC	Value	0.060	Left	270°	MONO	-115	265

2. Change the display attributes as required and click *Close*.

Summary

In this chapter, you learned how to modify symbols.

12 Editing Symbol Graphics

Objective

To become familiar with the steps in editing symbol graphics by using the Symbol Editor.

In this chapter, you will learn to:

- Perform zoom and pan operations on a symbol
- Draw graphical objects to add to a symbol
- Create a group of symbol objects
- Rotate an object
- Align objects
- Add a bitmap to a symbol
- Perform move and stretch operations on symbol objects

Overview

In Part Developer, you can modify a symbol in two ways, by changing the values of various fields in the Symbol Pins panel or by modifying the symbol text and graphics on the Symbol Editor canvas. While some symbol modification tasks, such as changing the location of a pin from left to right, can be done using either the Symbol Pins panel or the Symbol Editor, some other tasks, such as adding graphics to indicate the functionality of the symbol, can be done only using the Symbol Editor. This chapter covers various Symbol Editor features, which you can use to edit symbol graphics. You will use the library part `sample_part` in the `my_lib` library.

The part `sample_part` has six symbols.

Performing Zoom and Pan Operations on a Symbol

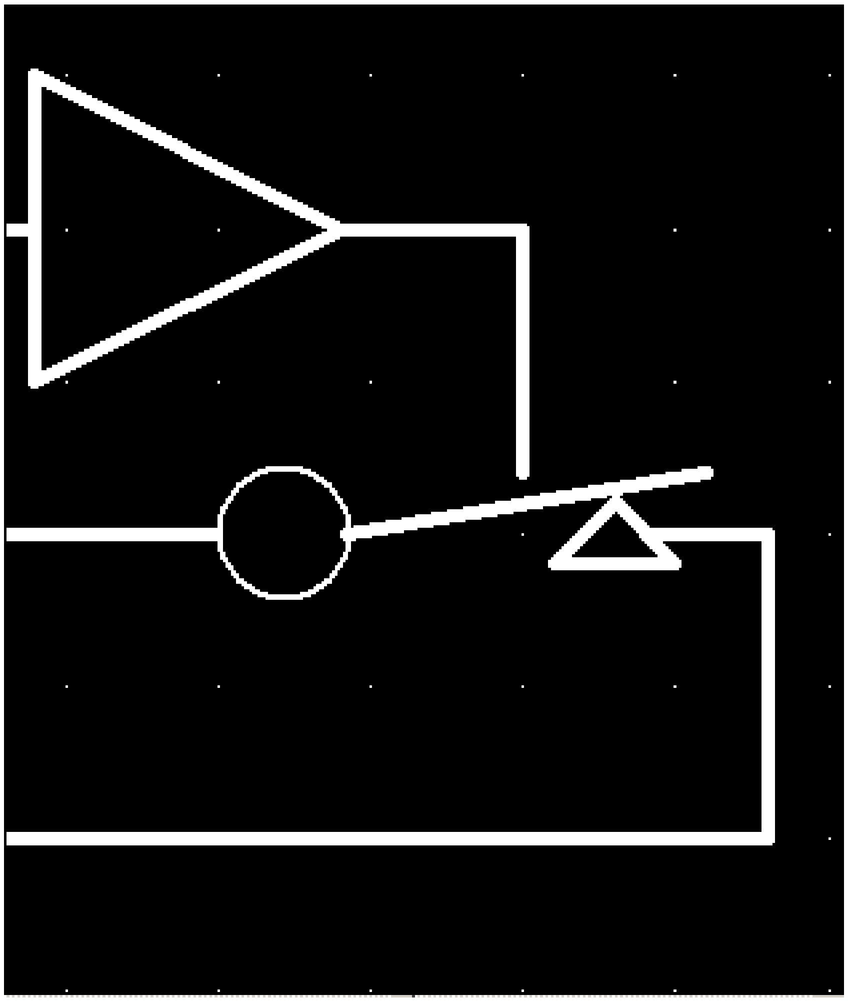
Zoom in on the switch graphics associated with the `PIN1` pin in `sym_6`. After the object has been magnified according to your specification, pan the Symbol Editor canvas to display only the `PIN+` and `PIN-` pins.

1. Click the *Zoom By Points* tool button .

   Notice that the mouse pointer changes to a magnifying glass.

2. Using the mouse, drag and select an area that encloses the switch associated with the `PIN1` pin.

   The switch is magnified.



Note: If you want to edit the graphics, first press Esc to exit the zoom mode.

3. To pan the canvas so that the PIN+ and PIN- pins can be viewed, click the *Pan* tool button .

   The mouse pointer changes to a hand.

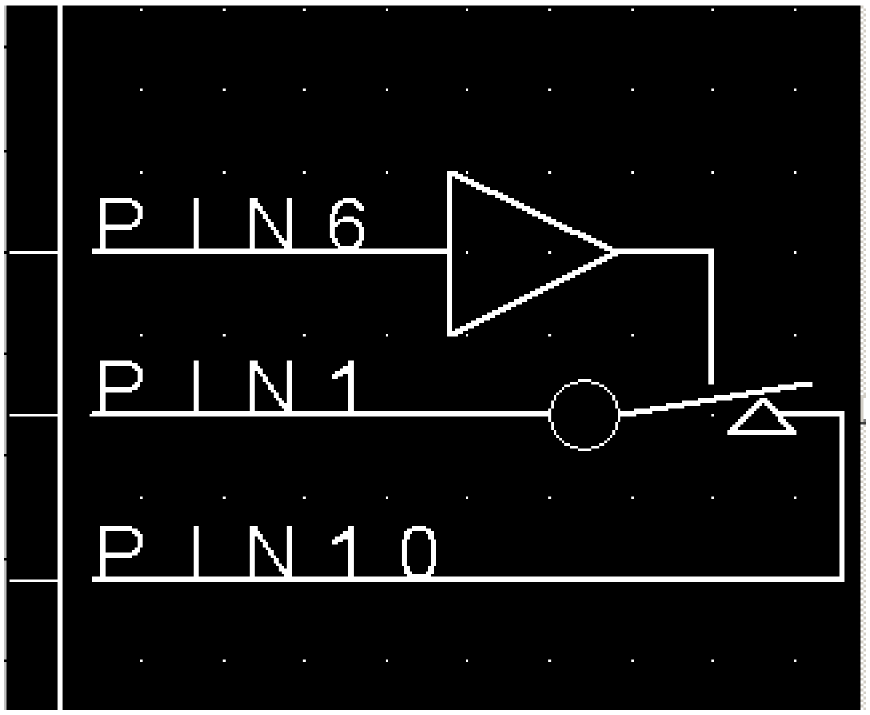
4. Click and release the mouse button only after the PIN+ and PIN- pins are displayed at the desired location on the Symbol Editor canvas.

Tip

If you are using a three-button mouse, you can pan using the middle mouse button.

Drawing Graphical Objects to Add to a Symbol

Draw the following graphic in the sym_1 symbol of sample_part to indicate that pins PIN6, PIN1, and PIN10 are associated with a buffer and a switch.



1. To begin drawing the buffer, click the *Add Line* tool button .

2. Click below the PIN6 text label and drag the mouse pointer to draw a horizontal line of the desired length.

3. To draw the triangle, click the *Add Polygon* tool button .

4. To draw a straight vertical line that is bisected by one end of the `PIN6` horizontal line, click at the desired location in the symbol and drag the mouse pointer vertically until the line is of the desired length.
5. Click and drag the mouse pointer to the apex of the triangle.
6. Double-click to complete the triangle.
7. To complete the buffer, click the *Add Polyline* tool button.
8. Click at the apex of the triangle and drag horizontally up to the desired length.
9. Click and drag vertically down up to the desired length.
10. Double-click to complete the shape.
11. To draw the switch, first draw a horizontal line of the desired length below the `PIN1` text label.
12. To draw a circle at the end of the horizontal line, click the *Add Circle* tool button.
13. Click on the desired location and drag the mouse pointer to draw the diameter of the circle.
14. Release the mouse pointer.

    The circle is drawn.
15. Using the *Add Line* tool button, draw a slanted line from the circle.
16. Using the *Add Polygon* tool button, draw a small triangle, the apex of which touches the slanted line.
17. Draw a horizontal line below the `PIN10` text label and drag it vertically up and then horizontally left to connect to the small triangle.

    The graphic is complete.

Tip

When drawing objects, you can either copy/paste or use the *Make components the same width* and *Make components the same height* tool buttons to create objects of identical dimensions.

Tip

To draw objects that should not be snapped to fit into the grid, make sure the *Snap to Grid* option is not selected. You can access this option from the *Canvas* menu item on the *Graphic Editor* menu or from the toolbar using the *Snap to Grid* tool button .

Creating a Group of Symbol Objects

In `sym_1`, create a group of the two pins `PIN+` and `PIN-` so that the pins can be moved together.

1. To select the `PIN+` pin along with its name and text label, click the pin when the mouse pointer changes to the move icon. You will notice that the same pin gets selected in the Symbol Pins panel.
2. Keeping the Ctrl key pressed, click the `PIN-` pin.

   Both the pins are selected. Notice that each selected pin is displayed in a separate box.
3. Right-click on the selection.
4. Click the Group option.

   Notice that all the selected objects are now displayed in a single box. This indicates that the group is created. You can now do any operation on the grouped object.

Tip

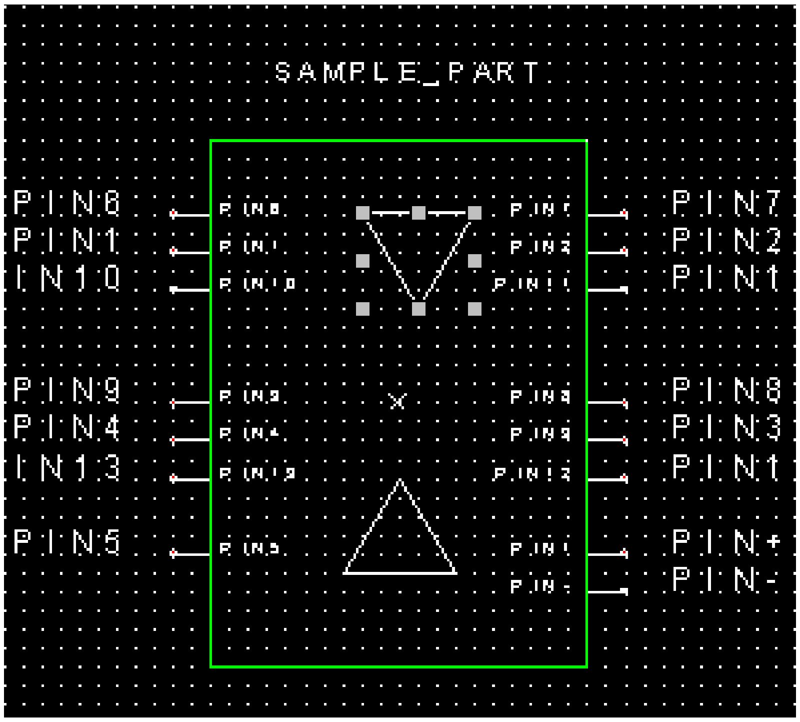
You can also select pins from the Symbol Pins panel by clicking the row number in the first column of the Logical Pins grid. This method of selection cannot be used if you want to select symbol objects other than pins.

Rotating an Object

In the `sym_2` symbol, rotate the first triangle to make it inverted.

1. Create a group of the three lines that constitute the first triangle.
2. Click the *Rotate 90 degrees counter clockwise* tool button two times.

The triangle is inverted.

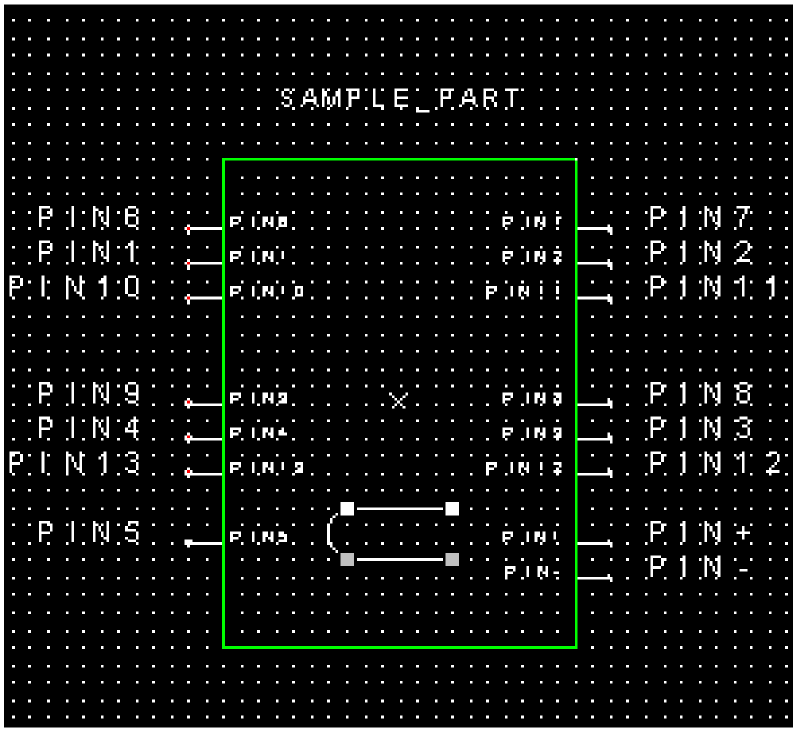


Aligning Objects

In the `sym_3` symbol, left-align the first horizontal line with the second horizontal line.

1. Select the first horizontal line.
2. Keeping the Ctrl key pressed, select the second horizontal line.
3. Click the *Align Left* tool button .

The line selected first is left-aligned with the line selected next. In the case of more than two objects, all objects are aligned with the last selected object.



Tip

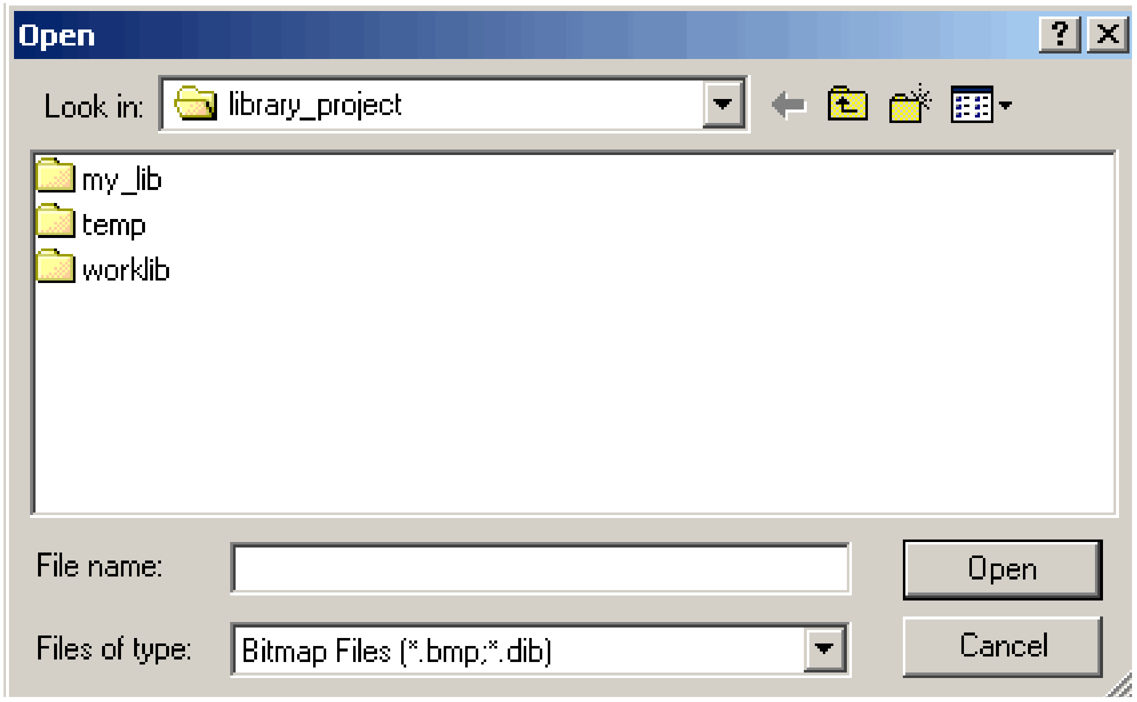
To make a set of circles concentric, select all the circles and then click the *Align Middle* tool button and the *Align Center* tool button.

Adding a Bitmap to a Symbol

Add the bitmap `logo.bmp` to the `sym_4` symbol.

1. Click the *Add Image* tool button .

The Open dialog box displays.



2. The path of the `logo.bmp` file is <*your_work_area*>`/library_project/my_lib/sample_part/images`. Browse the `my_lib` folder to select the bitmap `logo.bmp` and click the *Open* button.

   Notice that the mouse pointer changes to indicate that an image is being added.

3. To add the image, click at the desired location on the canvas.

The image is added to the symbol. You can stretch the image to increase its size.

Important

Make sure that the image to be added is stored in a `.bmp` file or a `.jpg` file.

Performing Move and Stretch Operations on Symbol Objects

Move the group of `PIN+` and `PIN-` pins a few grid points down in the `sym_5` symbol. Increase the symbol size by dragging the symbol outline.

1. Create a group of `PIN+` and `PIN-` pins.

2. Place the mouse pointer on either pin and drag the group to the desired location in the symbol.

   The group is moved to the new location.

3. To increase the length of the symbol, move the horizontal line at the base of the symbol down to the desired location.

   Notice that a yellow dotted line indicating the existing symbol outline is displayed at the original location.

4. Click the dotted line to display stretch handles and drag the middle handle until the outline merges with the horizontal line that you had moved to the new location.

   The symbol size is increased according to your specification.

Tip

To move an object by a fraction of a grid in any direction, you can use *Nudge Up*, *Nudge Down*, *Nudge Left*, or *Nudge Right* tool buttons.

Tip

To move pins from one side of the symbol to another, use the Symbol Pins panel instead of the Symbol Editor. Part Developer automatically adjusts the direction of a pin when the *Location* column value for the pin is changed in the Symbol Pins panel.

Summary

In this chapter, you learned how to edit symbol graphics.

13

Importing and Exporting

Objective

To become familiar with commonly used import and export procedures.

In this chapter, you will learn to:

- Import pin information from a CSV file to create a part.
- Import changed pin information (Import ECO) from a CSV file to update an existing part.
- Export part information to save in CSV format.
- Import pin information from an FPGA component to create a part.
- Import pin information from a die file to create a part.
- Export part information to create a Capture part.
- Export part information to save in ViewLogic format.

Overview

Support for a wide variety of file import and export in Part Developer makes Part Developer a powerful library-development tool. The Import and Export wizard of Part Developer not only speeds up logical-part creation but also eliminates the errors that are often introduced inadvertently when information is copied from one tool to another.

This chapter covers some of the import and export procedures that are commonly used by librarians.

The data files that you can use to try the import procedures covered in this chapter are stored in a directory called `import_files` at <*your_inst_dir*>`/doc/pdv_tut/tutorial_data`.

CSV Import and Export

CSV format enables you to maintain data in tool-independent format. If you need to support design teams that use multiple schematic editors with a layout tool, such as Allegro PCB Editor, you can single-source part data with CSV import and export support.

In this tutorial, you will:

- Import the pentium4_3Ghz.csv file to create a part named p4_3ghz.
- Update the `p4_3ghz` part by importing changed pin information from the CSV file that was used to create the part.
- Export the part details of the `p4_3ghz` part to a CSV file, `p4_3Ghz`.csv.

Importing a CSV File

Import the pentium4_3Ghz.csv file to create a part named p4_3ghz.

Import File Description

The pentium4_3Ghz.csv file contains information about logical and physical pins and symbols. When creating a part from this information, Part Developer maps the logical and physical pins and generates symbols for the part.

Task Overview

When creating the `p4_3ghz` part from the `pentium4_3Ghz.csv` file, do the following:

- Convert the duplicate `BIDIR` pins to the bits of a vector pin.
- Move the duplicate `POWER` pins to the global pin list.
- Change the type of the `UNSPEC` pins to `INPUT`.

Steps

The steps are as follows:

1. Open the `tutorial_project.cpm` project from the `library_project` folder in `tutorial_data`.

2. Choose *File – Import and Export*.

   The Import and Export wizard appears.

3. Choose the *Import Comma Separated Value (.csv) file* option and click *Next*.

   The Select Source page appears.

4. Browse to select the input CSV file `pentium4_3Ghz.csv` from the `import_files` folder and click *Open*.

5. Click *Next*.

   The Select Destination page appears.

   The name of the part to be created is seeded automatically from the CSV filename.

6. Change the part name to `p4_3ghz` by deleting the necessary characters in the displayed part name.

7. Select the library `my_lib` and click *Next*.

   The Preview of Import Data page appears.

8. Change the type of pin `P100` from `UNSPEC` to `INPUT` by choosing the *INPUT* option from the *Type* drop-down list.

9. Click *Finish* to complete the part creation process.

   Duplicate Pin Resolver Dialog displays the duplicate pins in the CSV file and provides options to resolve the duplicate pins. You can convert the duplicate pins into the bits of a vector pin, define them as individual scalar pins, or move them to the global pin list.

10. To convert the six `BPM` pins into a vector pin of six bits, retain the *Vector* option in the *Convert* column.

11. To move the duplicate POWER pins VCC_1 and VSS_1 to the global pin list, select the *Global* option from the *Convert* list for each pin.

12. Click *OK*.

    The Cell Editor appears with the part information.

13. Save the imported part and close the Cell Editor.

Importing a CSV File to Update a Part

Import pin information from a CSV file, `pentium4_3Ghz_eco.csv,` in the ECO process to update the `p4_3ghz part.`

Task Overview

When updating the `p4_3ghz` part, do the following:

- Make sure that all symbol graphic modifications that have been done during ECO are retained.

Steps

1. Choose *File – Import and Export*.

   The Import and Export wizard appears.

2. Choose the *Import ECO - Comma Separated Value (.csv)* file option and click *Next*.

   The Select Source for ECO page appears.

3. Browse to select the CSV file `pentium4_3Ghz_eco.csv` and click *Open*.
4. Click *Next*.

   The Select Destination for ECO page appears.

5. Select `my_lib as` the library and `p4_3ghz` as the destination cell and click *Next*.

   The Preview of Import Data page appears.

6. Click *Next*.

   Duplicate Pin Resolver Dialog displays the duplicate pins in the CSV file.

7. Resolve the duplicate pins and click *OK*.
8. Click *Next*.

   The ECO Messages page appears. You can review the list of differences and select only those that you want to import

9. To retain all symbol graphic modifications that have been done during ECO, make sure that the *Graphic modifications* check box in the Ignore section is not selected.
10. Click *Finish*.

    The Cell Editor appears with the part information.

11. Save the updated part and close the Cell Editor.

Exporting to a CSV File

Export the details of a part named `p4_3ghz` to a CSV file, `p4_3Ghz`.csv.

Steps

The steps are as follows:

1. Choose *File – Import and Export*.

   The Import and Export wizard appears.

2. Choose the *Export Comma Separated Value (.csv) file* option and click *Next*.

   The Select Source page appears.

3. Select `my_lib` as the source library and `p4_3ghz` as the cell to be exported and click *Next*.

   The Select Package page appears.

4. Select `P4_3GHZ` as the package and click *Next*.

5. Specify `import_files` as the directory in which to save the CSV file and click *Finish*.

   The `p4_3ghz` part is exported.

   Note: The CSV filename is derived from the package name of the part.

Importing an FPGA Component

Import the Xilinx place-and-route file `xilinx_7x.pad` to create a library part named `xilinx_7x`.

Task Overview

- Make sure that you create a new schematic symbol for the imported FPGA.
- Specify the default values for the following properties:
  - DESCRIPTION
  - JEDEC_TYPE

Steps

The steps are as follows:

1. Choose *File – Import and Export*.

The Import and Export wizard appears.

2. Choose the *Import FPGA* option and click *Next*.

The Select Source page appears. On this page, you choose the vendor for the place-and-route tool you used to create the place-and-route file for an FPGA.

3. Choose *Xilinx* from the Vendor list.

4. Click the browse button to select the `xilinx_7x.pad` file in the `import_files` folder and click *Open*.

5. Click *Next*.

The Select Destination page appears.

6. To create a new schematic symbol for the FPGA, choose the *Generate Custom Component* option.

7. Click *Default Properties* to display the Default Properties dialog box.

In this dialog box, you specify the default values for the following properties as tabulated below:

Name	**Value**
DESCRIPTION	Xilinx pad file
JEDEC_TYPE	ff668

8. To specify the value for a property, select a property from the *Name* drop-down list and then enter its value in the *Value* field.

Tip

After adding the information for one property, press `Ctrl +I` to add a new row.

9. To close the Default Properties dialog box, click *OK*.

10. Select `my_lib` as the library in which you want the FPGA component to be created and click *Next*.

The Preview of Import Data page appears.

11. Click *Finish*.

The Cell Editor appears with the part information.

12. Save the imported part and close the Cell Editor.

You can now use the Component Browser to add the FPGA component or the block instantiating the FPGA component in your design.

Importing a Die File

Import the die file `transceiver.txt` from the `import_files` folder to create a library part named `transceiver`.

Task Overview

When creating the `transceiver` part from the `transceiver.txt` die file, do the following:

- Change the type of the `UNSPEC` logical pins to `INPUT`.

Steps

The steps are as follows:

1. Choose *File – Import and Export*.

The Import and Export wizard appears.

2. Choose the *Import Die Text* option and click *Next*.

The Select Source page appears.

3. Click the browse button to select the die file `transceiver.txt` and click Open.

4. Click *Next*.

The Select Destination page appears.

The name of the part to be created is seeded automatically from the name of the die file.

5. Select `my_lib` as the destination library and click *Next*.

The Preview of Derived Data page appears.

All the pins are of the UNSPEC type.

6. To change the type of all the pins to INPUT, select all the values in the *Type* column.
7. Type `i`.

   The types of all the pins are changed to `INPUT`.
8. Click *Finish* to complete the part creation process.

   Duplicate Pin Resolver Dialog displays the duplicate pins in the die file.
9. To convert the duplicate pins to vector pins, retain the *Vector* option in the Convert column and click *OK*.

   The Cell Editor appears with the part information.
10. Save the imported part and close the Cell Editor.

Exporting to a Capture Part

Export the details of a part named `p4_3ghz_sym` to create a Capture part.

Task Overview

Make sure that the symbol port names are used to represent the pins in Capture.

Steps

The steps are as follows:

1. Choose *File – Import and Export*.

   The Import and Export wizard appears.
2. Choose the *Export Capture Part (Windows Only)* option and click *Next*.

   The Select Source page appears.
3. Select `my_lib` as the source library and `p4_3ghz_sym` as the part to be exported and click *Next*.

   The Select Package and Symbol(s) page appears.
4. Select the `P4_3GHZ_SYM` package.

5. To use the symbol port names as pin names in Capture, select the *Use Pin Name to write Capture Port name* check box.
6. Click *Next*.
7. Browse to the `import_files` folder, specify `export_p4_3ghz` as the name of the Capture library in which the part is to be created, and click *Open*.
8. Click *Finish*.

   The part details have been exported to a file called `EXPORT_P4_3GHZ.OLB` in the `import_files` folder. You can launch Capture and view the part.

Exporting to a ViewLogic Part

Export the details of a part named `p4_3ghz_sym` to save in ViewLogic format.

Steps

The steps are as follows:

1. Choose *File – Import and Export*.

   The Import and Export wizard appears.
2. Choose the *Export ViewLogic(VL) Part* option and click *Next*.

   The Select Source page appears.
3. Select `my_lib` as the source library and `p4_3ghz_sym` as the cell to be exported and click *Next*.

   The Select Associated Package(s) or Unassociated Symbol(s) page appears.
4. Select the `P4_3GHZ_SYM` package, specify the type of the ViewLogic part as *Non-Hetero,* and click *Next*.

   The Select Destination page appears.
5. Specify the destination directory and click *Finish*.

   The part is exported. If you have ViewLogic installed, you can open the newly created part.

Summary

In this chapter, you learned how to use Part Developer to import pin information from various formats for creating Design HDL parts and export pin information from Design HDL parts to various formats.

14

Part Logging and Versioning

Overview

Part logging and versioning enables you to store and view the following information about a part:

- A log of all actions done on a part, such as symbol creation and pin modifications
- Major and minor revision numbers of all the views

By default, whenever you start part logging, Part Developer assigns the major revision number for all views and the cell as 1. The minor revision number is 0. Then, depending upon the types of changes you make to the part or views, the major or minor number of the part or the views are updated automatically by Part Developer.

Whenever a major or minor revision number of a view is incremented, the major or minor revision number of the cell is also incremented. For example, a part and its package and symbol views have a major revision number 1 and a minor revision number 0. Now, the package undergoes a modification, resulting in a major revision number change for the package. This will cause the major revision number of the part to be incremented as well. So, the part and the package will have the major revision number as 2. Now, the symbol is modified, resulting in the major revision number of the symbol to be incremented. This will also result in the major revision number of the part to be incremented. So, the part will now have the major revision number as 3 while the major revision numbers of the package and the symbol continue to be 2. See Modifications that Result in Major and Minor Number Changes on page 144 for details.

The part log and version information is stored in the metadata view of the part. The metadata view contains the following files:

- ❑ master.tag

  The `master.tag` file is the control file that ensures that Part Developer treats this directory as a valid view.

- ❑ revision.dat

The `revision.dat` file stores the version information for a part. The information stored includes the name of the view, the type of the view, major and minor revision numbers, the date and time of creation, the name of the creator, the library to which the part belongs, and the modification history.

- ❑ revision.log

  The `revision.log` file logs all the activities that are done on the part. The information stored includes the time of the modification, the name of the modifier, the type of modification, and a detailed description of the type of change.

  Both `revision.dat` and `revision.log` files are written when the part is saved.

- ❑ pinlist.txt

  The `pinlist.txt` file stores the list of pins that have been added to the part.

Part Developer will start logging all the changes that you make on the part and also automatically increment the major or minor version of the views and parts as required.

Caution

Do not manually edit the revision.dat and revision.log files outside of Part Developer. If these files are manually edited, Part Developer might fail to give you the correct version information or logging information for the part.

Starting Part Logging and Versioning

Task Overview

Load the `for_baselining` part from the `my_lib` library and baseline it. Modify the package by deleting the input pin A1. View the version numbers after the changes are made to the part.

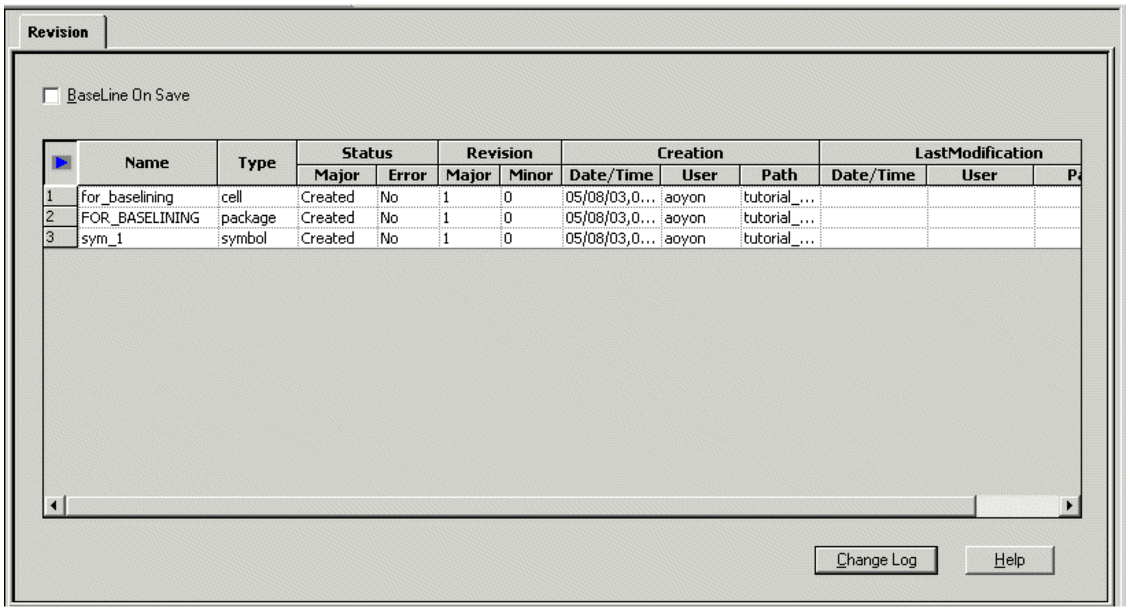
Steps

1. Select *File – Open – Cell.*

   The Open Cell dialog box appears.

2. Select the *for_baselining* part from the *my_lib* library and click *OK*.

The part is loaded in the Cell Editor. Notice that by default, the *Major* revision number is assigned as 1 for the existing views. Also, the *Major* status appears as *Created*.



☐ BaseLine On Save

	Name	Type	Status		Revision		Creation			LastModification		
			Major	Error	Major	Minor	Date/Time	User	Path	Date/Time	User	Pa
1	for_baselining	cell	Created	No	1	0	05/08/03,0...	aoyon	tutorial_...			
2	FOR_BASELINING	package	Created	No	1	0	05/08/03,0...	aoyon	tutorial_...			
3	sym_1	symbol	Created	No	1	0	05/08/03,0...	aoyon	tutorial_...			

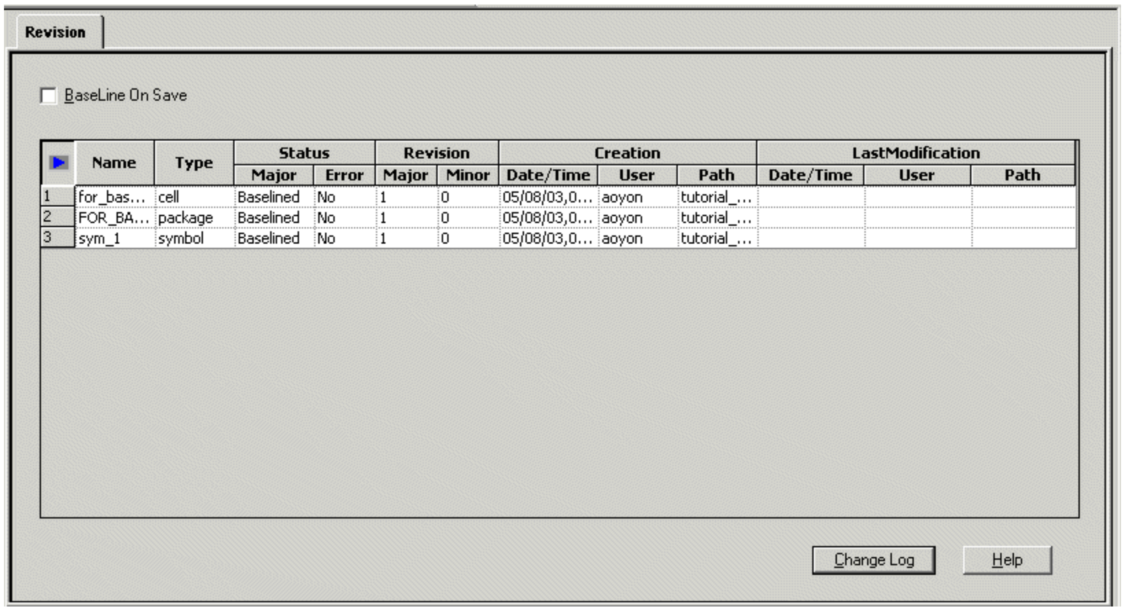
3. Select the *BaseLine on Save* option.

4. Save the part.

5. To view the change log and revision numbers in Part Developer, close the part and reload it.

The Revision Editor appears with the major version appearing as `Baselined` for the cell and all its views. Now, whenever any change is made to the cell, the revision number will get updated as required.

Note: The changes are not dynamically reflected in the Revision Editor. You will need to reload the part to see the revision history.



	Name	Type	Status		Revision		Creation			LastModification		
			Major	Error	Major	Minor	Date/Time	User	Path	Date/Time	User	Path
1	for_bas...	cell	Baselined	No	1	0	05/08/03,0...	aoyon	tutorial_...			
2	FOR_BA...	package	Baselined	No	1	0	05/08/03,0...	aoyon	tutorial_...			
3	sym_1	symbol	Baselined	No	1	0	05/08/03,0...	aoyon	tutorial_...			

Important

The *BaseLine on Save* option is deselected when a part is reloaded. You will need to select the *BaseLine on Save* option before saving the part to ensure that the revision numbers are updated.

In case a baselined part is saved with the *BaseLine on Save* option checked, a `.baselined` file is created in all the views. For example, `chips.prt.baselined` is created in the chips view. Now, if the part is saved with the *BaseLine on Save* option checked, the views are compared against the `.baselined` files and revision numbers are updated accordingly.

Important

Part logging is possible only for error-free parts.

6. Select the *FOR_BASELINING* package in the cell tree.

7. Delete the input pin *A1*.

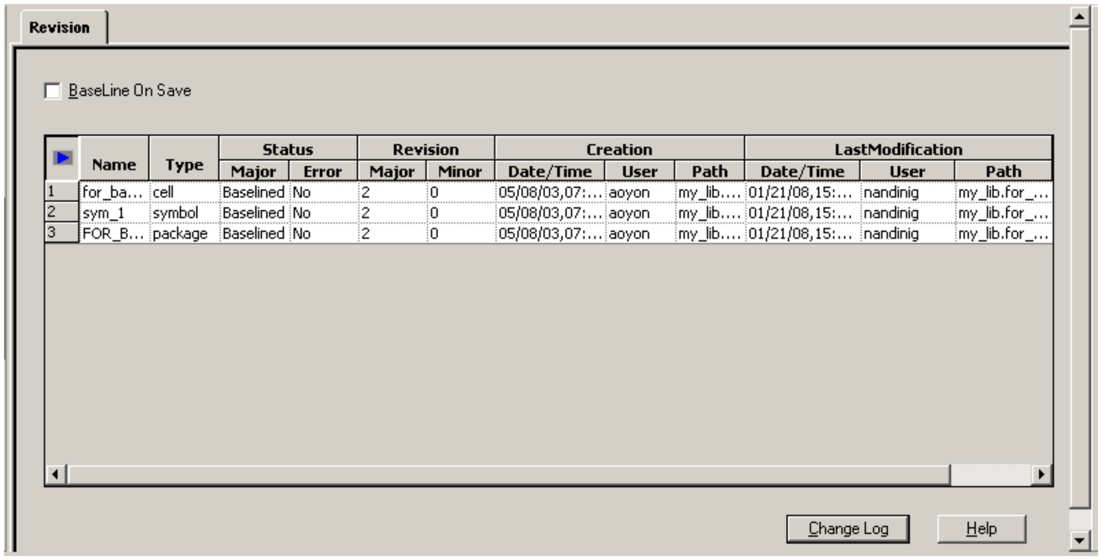
8. Click on the root of the cell tree.

9. Select *BaseLine on Save*.

10. Save the part.

11. Close and reload the part.

The major version is updated for the part.



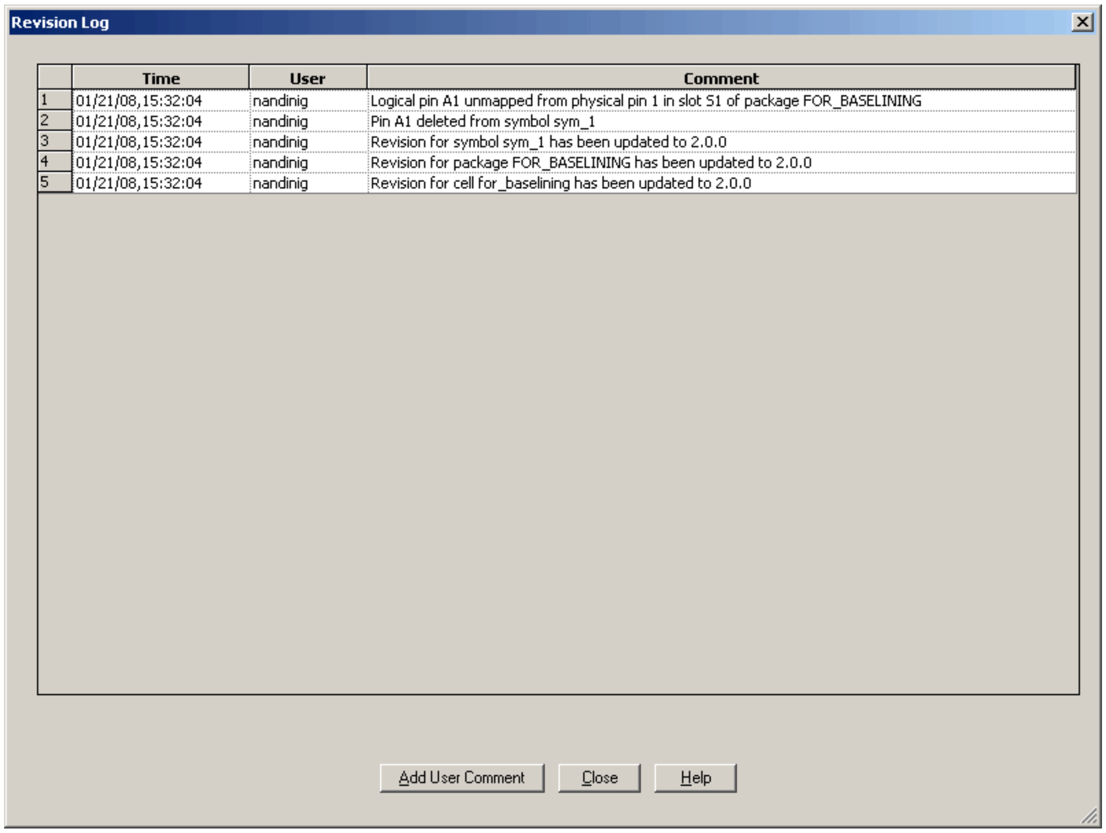
☐ BaseLine On Save

	Name	Type	Status		Revision		Creation			LastModification		
			Major	Error	Major	Minor	Date/Time	User	Path	Date/Time	User	Path
1	for_ba…	cell	Baselined	No	2	0	05/08/03,07:…	aoyon	my_lib.…	01/21/08,15:…	nandinig	my_lib.for_…
2	sym_1	symbol	Baselined	No	2	0	05/08/03,07:…	aoyon	my_lib.…	01/21/08,15:…	nandinig	my_lib.for_…
3	FOR_B…	package	Baselined	No	2	0	05/08/03,07:…	aoyon	my_lib.…	01/21/08,15:…	nandinig	my_lib.for_…

Viewing the Revision Log

1. To view the revision log file, click *Change log*.

The Revision Log dialog box appears displaying the log of changes made to the part and its views.



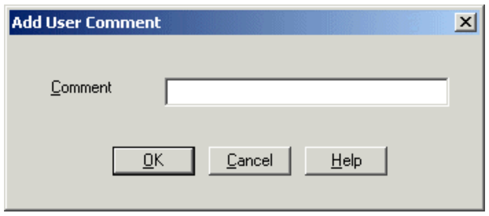
	Time	User	Comment
1	01/21/08,15:32:04	nandinig	Logical pin A1 unmapped from physical pin 1 in slot S1 of package FOR_BASELINING
2	01/21/08,15:32:04	nandinig	Pin A1 deleted from symbol sym_1
3	01/21/08,15:32:04	nandinig	Revision for symbol sym_1 has been updated to 2.0.0
4	01/21/08,15:32:04	nandinig	Revision for package FOR_BASELINING has been updated to 2.0.0
5	01/21/08,15:32:04	nandinig	Revision for cell for_baselining has been updated to 2.0.0

Adding Your Comments to the Revision Log

Part Developer allows you to write to the revision log of a part. To do so:

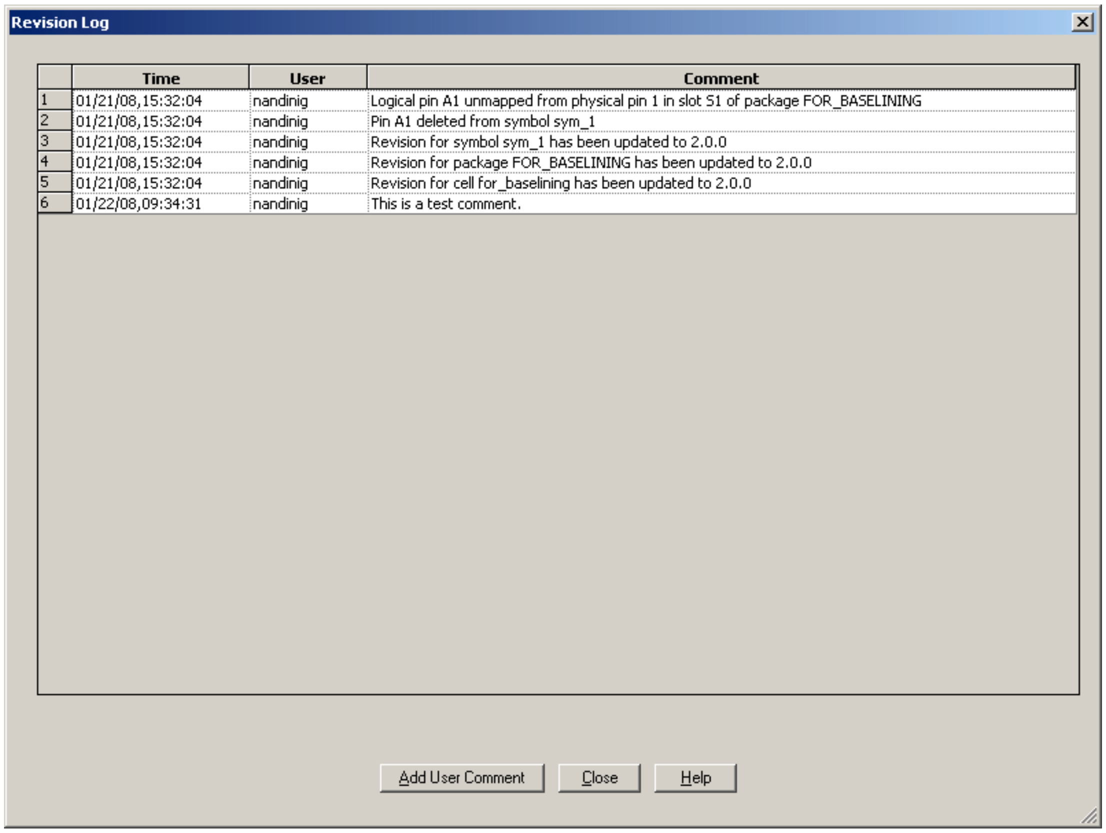
1. Click *Add User Comment*.

   The Add User Comment dialog box appears.



2. Enter the comment that you want to add to the log file and click *OK*.

The comment is written to the `revision.log` file and appears in the Revision Log dialog box.



	Time	User	Comment
1	01/21/08,15:32:04	nandinig	Logical pin A1 unmapped from physical pin 1 in slot S1 of package FOR_BASELINING
2	01/21/08,15:32:04	nandinig	Pin A1 deleted from symbol sym_1
3	01/21/08,15:32:04	nandinig	Revision for symbol sym_1 has been updated to 2.0.0
4	01/21/08,15:32:04	nandinig	Revision for package FOR_BASELINING has been updated to 2.0.0
5	01/21/08,15:32:04	nandinig	Revision for cell for_baselining has been updated to 2.0.0
6	01/22/08,09:34:31	nandinig	This is a test comment.

Stopping Part Logging and Versioning

➤ To stop part logging and versioning, deselect the *BaseLine on Save* option.

Restarting Part Logging and Versioning

➤ To start part logging and versioning, select the *BaseLine on Save* option.

Modifications that Result in Major and Minor Number Changes

The changes that impact the design flow are considered major changes, leading to an increment in the major revision number. All other changes are treated as minor changes and result in incrementing the minor revision number.

Modifications that Result in a Major Number Change for a Part or a View

The following modifications cause a change in the major revision number:

- Addition, deletion, or renaming of any view
- Modification of the JEDEC_TYPE property
- Addition or deletion of pins from packages and symbols
- Modification of the pin type for an existing pin
- Renaming of pins in the package and symbols
- Addition, deletion, or renaming of symbol text
- Modification of a symbol pin position
- Addition or deletion of slots from a package
- Change in the low-assertion character setup for a part, resulting in the change of how the low-assertion character is stored in the chips and the symbol view
- Change in the distribution of pins among slots
- Change in the mapping information of existing pins
- Global renaming of pins
- Global deletion of pins
- Addition or deletion of a model for a wrapper
- Modification of a model for an existing wrapper
- Mapping or unmapping of pin to m-port mapping in a mapfile or a wrapper
- Addition, deletion, or modification of the binding statement for a VHDL wrapper or mapfile

- Addition or deletion of pin-to-model port mapping for primitives from Verilog/VHDL mapfiles
- Modification of the `default_model` property for a primitive in the map view
- Addition, deletion, or modification of a model for a primitive entry in the mapfile
- Annotation of parameters or generics onto symbols
- Deletion of annotated parameters or generics from the symbols
- Updating pin mode or port type of pins during Verilog/VHDL mapfile/wrapper creation
- Modification of the `UPPERCASE` property of primitives

Modifications that Result in a Minor Number Change for a Part or a View

- Addition, deletion, or renaming of an additional package property
- Addition, deletion, or renaming of a package pin property
- Addition, deletion, or renaming of a package alias
- Modification of the electrical class of a package
- Modification of the reference designator prefix of a package
- Addition, deletion, or renaming of a symbol property
- Addition, deletion, or renaming of a symbol pin property
- Modification of the symbol outline (thickness/width/height)
- Modification of the color of the symbol outline
- Change of pin shapes
- Modification of symbol and symbol pin attributes
- Addition or deletion of the model alias from the primitive in the mapfile

Summary

In this chapter, you learned how to create and maintain part versions and change logs.

15

Interface Comparator

Overview

The Interface Comparator is a mechanism by which you can compare the pin lists of two interfaces, such as a package and a symbol, identify the differences between them and then update the pin list of one of the objects to match the other, or both with respect to each other.

The following scenario explains the need and usefulness of this feature. Suppose you create a part where you develop the packages and symbols separately. After creating them, you realize that there are some symbols that are unassociated with any of the packages. Such a part is not usable in the design flow because the symbols must be packageable into one of the packages. To make a symbol packageable, you need to update the pin list of the symbol to match the pin list of the package or vice-versa. For large pin-count parts, this task, when done manually, can be tiresome and error-prone. The Interface Comparator feature enables you to automatically identify the differences between the selected symbol and the package and update the pin list of either the symbol or the package to match the other.

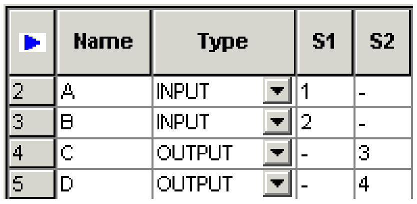
Using Interface Comparator, you can do comparisons between:

- A function group and a symbol
- Two function groups of a package
- Two symbols

After the comparison, you can choose to update the pin list of one of the compared objects with the pin list of the other or both with respect to each other.

A sample part, `part_for_interface_comparison`, is used to explain this. The `part_for_interface_comparison` part has one package with four pins, `A`, `B`, `C`, and

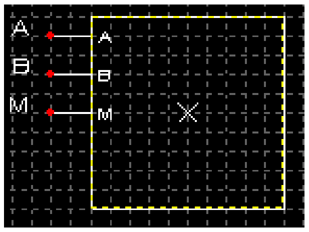
D. These four pins are split into two function groups, with pins A and B forming the first function group, and pins C and D forming the other. The pin list of the package is displayed below:



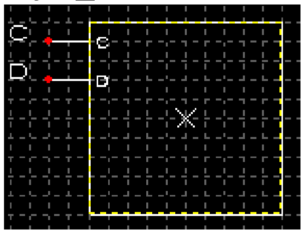
	Name	Type	S1	S2
2	A	INPUT	1	-
3	B	INPUT	2	-
4	C	OUTPUT	-	3
5	D	OUTPUT	-	4

The `part_for_interface_comparison` part also has two symbols, sym_1 and sym_2. The symbols are displayed below:

Sym_1



Sym_2



The symbol `sym_2` is associated with the second function group. The symbol `sym_1` has pins `A` and `B` and an extra pin `M`. This results in `sym_1` being not associated with any function group.

The Interface Comparator will be used to synchronize the function group with the symbol. As a result of this synchronization, the logical pin `M` will be added to the first function group. Part Developer will ensure that the synchronization does not destroy the existing function group and symbol associations, for example, the association of `sym_2` with function group 2.

Running the Interface Comparator

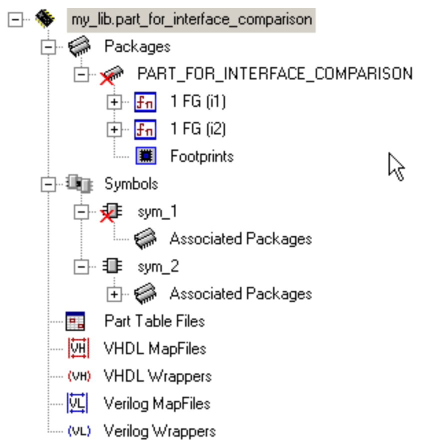
Task Overview

Synchronize the `sym_1` symbol of the `part_for_interface_comparison` part of the `my_lib` library with the `PART_FOR_INTERFACE_COMPARISON` package.

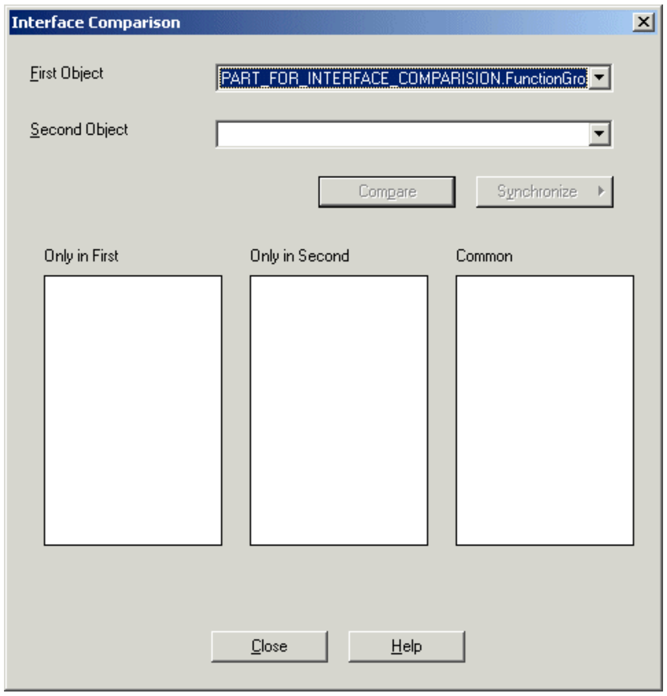
Steps

1. Load the `part_for_interface_comparison` part in Part Developer.

   Notice that `sym_1` is not associated with any package.



2. Right-click on the `sym_1` entry and choose *Interface Comparison* from the pop-up menu.



3. Select *FunctionGroup1* as the first object.

4. Select *sym_1* as the second object.

5. Click *Compare*.

Part Developer runs a comparison check on the logical pin lists of the two objects and displays the results in the following way:

- ❑ The pins that are present only in the first object are displayed in the *Only in First* list box.
- ❑ The pins that are present only in the second object are displayed in the *Only in Second* list box.
- ❑ The pins that are common to both the objects are displayed in the *Common* list box.

Next, you need to determine how to synchronize the pin lists of the two objects. Synchronization is the process by which the pin list of one of the objects is updated to match the pin list of the other object. This is done by either adding or removing pins. Synchronization can be done in one of the following ways:

- ❑ First with Second

  In this method, the second object is treated as the master object and the pin list of the first object is updated to match the pin list of the second object. Extra pins in the first object are deleted from the first object. If the second object has additional pins, they are added to the first object.

- ❑ Second with First

  In this method, the first object is the master object and the pin list of the second object is updated to match the pin list of the first object. The pins that are present in the first object but not in the second object are added to the second object. Pins that are present in the second object but not in the first will be deleted from the second object.

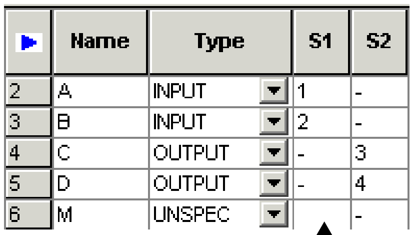
- ❑ Both

  In this method, both objects are treated at par. Only the common pins are retained in both the objects and the extra pins, if any, are deleted.

6. Click *Synchronize*.

7. Select the *First With Second* option.

Now, when you synchronize with the *First with Second* option, pin `M` will get added to the first function group. Pin `M` will appear with a hyphen in the second function group, thus showing that it is not present in the second function group. As shown below, you will need to specify the physical pin numbers for the new logical pin `M` and do the mapping.



▶	Name	Type	S1	S2
2	A	INPUT	1	-
3	B	INPUT	2	-
4	C	OUTPUT	-	3
5	D	OUTPUT	-	4
6	M	UNSPEC		-

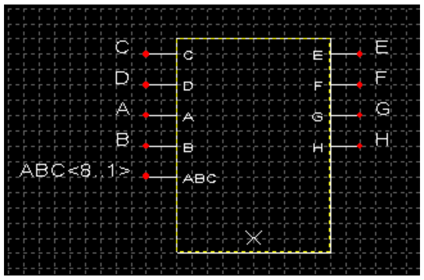
Pin M added after synchronization

Points to Remember when Running Interface Comparator

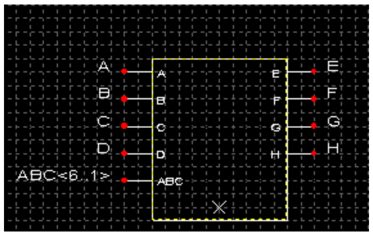
- In case logical pins are added to a package after synchronization, you will need to open the package in the Package Editor and manually specify the logical-to-physical pin mappings for the new pins.
- When a pin is added to the symbol, the location for the pin is read from the setup options.
- When a pin is added to a package, the attributes like pin type, pin location, loading values, and checks are taken from the global data. Therefore, if the package, function group, or symbol from which the pin is added has the attribute values that are different from the global data, those values are ignored and the attribute values in the global data is used.
- If pins are added to a symbol, Part Developer attempts to fit in as many pins as possible in the existing symbol outline. If the pins do not fit into the existing symbol outline, Part Developer automatically extends the symbol outline. In case the symbol outline is extended and the *Preserve Pin Position* option is checked on the Pins page of the Symbol Editor, Interface Comparator will keep the positions of the existing pins intact when adding the new pins.
- For vector pins in a symbol, if some bits are deleted after synchronization, Part Developer displays the vector pin in an expanded format with the remaining bits.

For example, consider the following two symbols. The symbol `sym_1` has a vector pin, ABC, with 6 bits, and `sym_2` has a vector pin, ABC, with 8 bits.

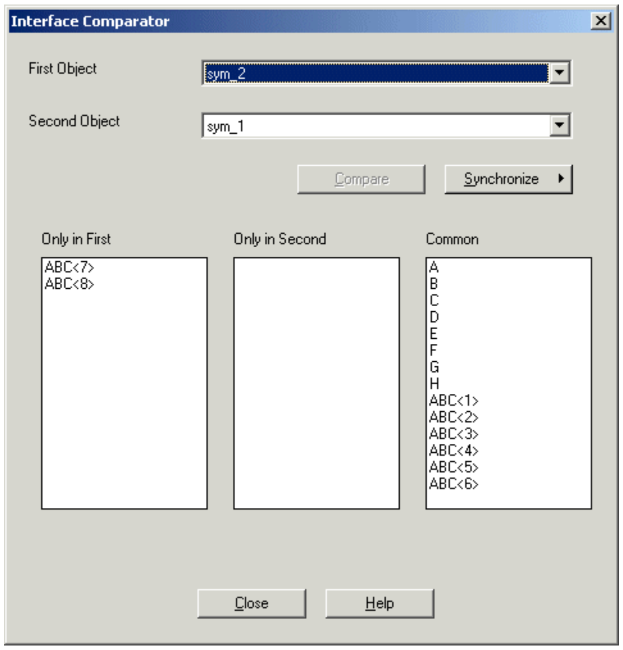
Sym_2



Sym_1



On running the interface comparison on the two symbols, the Interface Comparator dialog box appears as follows:

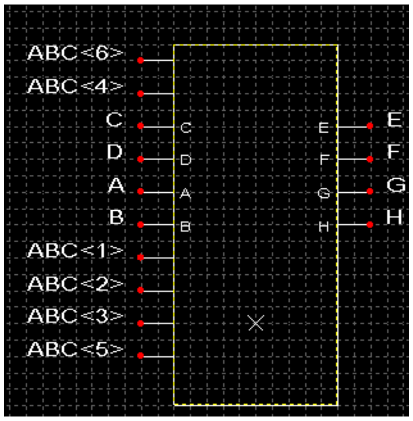


First Object: sym_2
Second Object: sym_1

Only in First	Only in Second	Common
ABC<7>		A
ABC<8>		B
		C
		D
		E
		F
		G
		H
		ABC<1>
		ABC<2>
		ABC<3>
		ABC<4>
		ABC<5>
		ABC<6>

Now, when you synchronize with the *First with Second* option, the bits 7 and 8 will get deleted from `sym_2`. As shown in the following graphic, the vector pin ABC appears in `sym_2` with all the 6 bits expanded.

Sym_2 After Synchronization



- After comparing a function group with another, it may be possible that a pin has hyphens in all the slots. In such a case, the pin is deleted from the package.

Summary

In this chapter, you learned to run Interface Comparison to eliminate errors in your part.