

Vivado Design Suite

Creating and Packaging Custom IP

UG1118 (v2014.3) October 8, 2014

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
10/08/2014	2014.3	<p>This document was distilled from the UG896 Chapter on Packaging.</p> <p>Moved Standard and Advanced File group tables to Appendix A, Standard and Advanced File Groups. Added Important note regarding NGC files to this appendix.</p> <p>Changed Chapter 2, Using the Creating and Package IP Wizard.</p> <p>Added note regarding 80 character limit to page 18.</p> <p>Changed all sections in Chapter 3, Packaging IP.</p> <p>Moved Setting an Enablement Expression, page 63 a common section.</p> <p>Added references to UG994 in Chapter 4, Creating and Packaging Custom IP in IP Integrator.</p> <p>Added Introduction, page 5 and changed Outputs from IP Packager, page 7.</p> <p>Changed Using IP Project Settings, page 8.</p> <p>Captured inferred information in Auto-Inferring an Interface, page 53.</p> <p>Added Packaging a Block Design from IP Integrator in Chapter 4.</p> <p>Added Creating New Interface Definitions in Chapter 5.</p>

Table of Contents

Chapter 1: Creating and Packaging Custom IP

Introduction	5
Available IP Packager Inputs	7
Outputs from IP Packager	7
Using IP Project Settings	8

Chapter 2: Using the Creating and Package IP Wizard

Introduction	12
Using the Create and Package IP Wizard	13
Packaging Your Current Project	15
Packaging a Specified Directory	17
Creating a New AXI4 Peripheral	19

Chapter 3: Packaging IP

Introduction	22
Identification	22
Compatibility	25
File Groups	28
Customization Parameters	33
Ports and Interfaces	45
Addressing and Memory	53
Customization GUI	57
Review and Package	60
Setting an Enablement Expression	63

Chapter 4: Creating and Packaging Custom IP in IP Integrator

Introduction	64
Packaging a Block Design from IP Integrator	64

Chapter 5: Creating New Interface Definitions

Introduction	68
Creating a New Interface Definition	68
Using the Interface Definition Editor	70

Completing the Interface Definition Creation 73
Re-Editing Interface Definitions 74
Using a New Interface Definition 74

Appendix A: Standard and Advanced File Groups

Introduction 75
Standard File Groups 75
Advanced File Groups 76

Appendix B: Additional Resources and Legal Notices

Xilinx Resources 78
Vivado Design Suite Documentation 78
Xilinx IP Documentation 79
Training Resources 79
Please Read: Important Legal Notices 80

Creating and Packaging Custom IP

Introduction

Using the Vivado® IP packaging flow gives you a consistent experience whether using Xilinx® IP, third-party IP, or customer-developed IP.

Figure 1-1, page 6 shows the flow in the IP packaging and usage model. With the Vivado IP packager, you, as an IP Developer, can:

- Create and package files and associated data in an IP-XACT standard format.
- Add IP to the Vivado IP Catalog.
- Deliver packaged IP to an end-user in a repository directory or in an archive (.zip) file.

After you distribute IP, an end-user can create a customization of that IP in their designs.



RECOMMENDED: *Verify IP by running each IP module completely through the IP user flow before you package and distribute any files.*

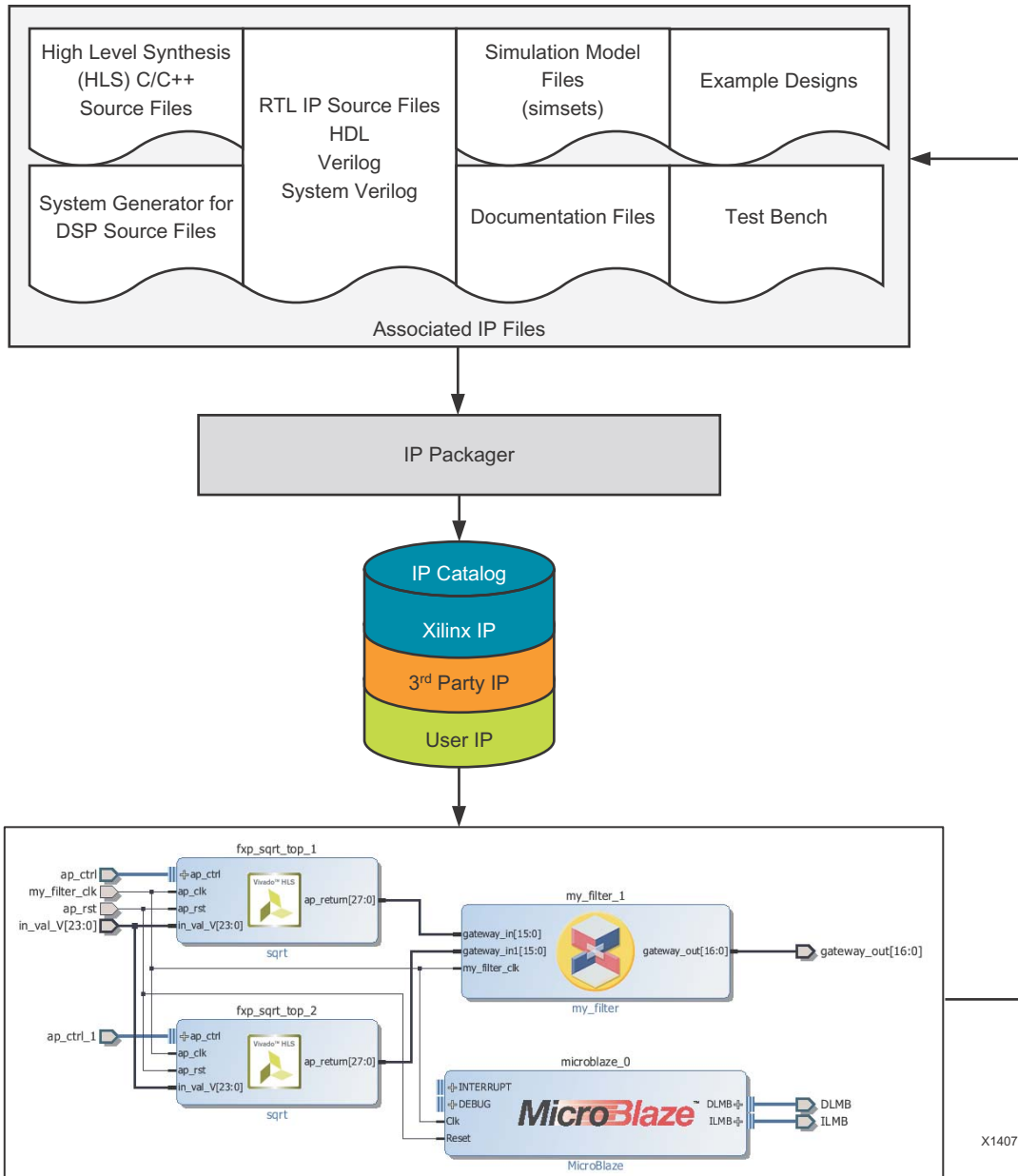


Figure 1-1: IP Packaging and Usage Flow

Available IP Packager Inputs

The Vivado IP packager supports the following input file groups:

- HDL synthesis
- HDL simulation
- Documentation
- HDL test bench
- Example design
- Implementation files (including constraint and structural netlist files)
- Drivers
- GUI customization

IP packager can designate as many or as few file groups as is appropriate to the IP. There is no requirement for a minimum set of file groups; however, the IP packager **IP File Groups** page presents a *typical* set of file groups, based upon the packaged project sources. When any of these file groups are empty, the final Review and Package page issues a warning about missing file content.

Outputs from IP Packager

The IP packager generates an XML file based on the IP-XACT standard, component.xml, and a XGUI customization Tcl file. These two files are generated at the location of the IP root directory. The IP-XACT component XML file is then used to identify the IP definition information. The XGUI customization Tcl file, located in the /XGUI folder of the IP root directory location, displays the customization GUI of the custom IP from the IP Catalog.

The associated files of the custom IP are relatively referenced from the IP-XACT XML file. If the project was packaged remotely, the associated IP files are copied to the selected IP location. The files are categorized in directories based on usage (for example: /src, /sim, /doc).

Using IP Project Settings

When working with IP in a Manage IP project or in an RTL project, you can configure IP-specific project settings using the IP category in the Project Settings dialog box. The following tabs are available:

- **Repository Manager:** Adds IP repositories and specifies the IP to include in the IP catalog.
- **Packager:** Sets the default behavior used by the IP packager when packaging IP.

Note: The IP Project Settings and the Vivado IP Catalog are only available when working with an RTL project or when using Manage IP from the Getting Started page. When using Manage IP, a subset of the IP settings are available unless a project is created.

Using the Repository Manager

To use the Repository Manager, shown in the following figure, do the following:

1. Select **Tools > Project Settings**.
2. In the left pane of the Project Settings dialog box, click **IP**.
3. Click the **Repository Manager** tab, and do the following:
 - a. In the IP Repositories section, click **Add Repository** to specify the directories that contain packaged IP to add to the IP repositories list. The Repository Manager hierarchically searches within the user repository paths for IP definitions.

You can either use your packaged IP or acquire it from a third-party supplier.

- b. In the **IP > Selected Repository** section, click **Add IP** to specify the IP to include in the IP Catalog.

Note: The IP catalog shows the included IP, and you can create a customization of the IP for use in a design.

- c. To update the contents of the IP Catalog with the IP within each repository, click **Apply**.

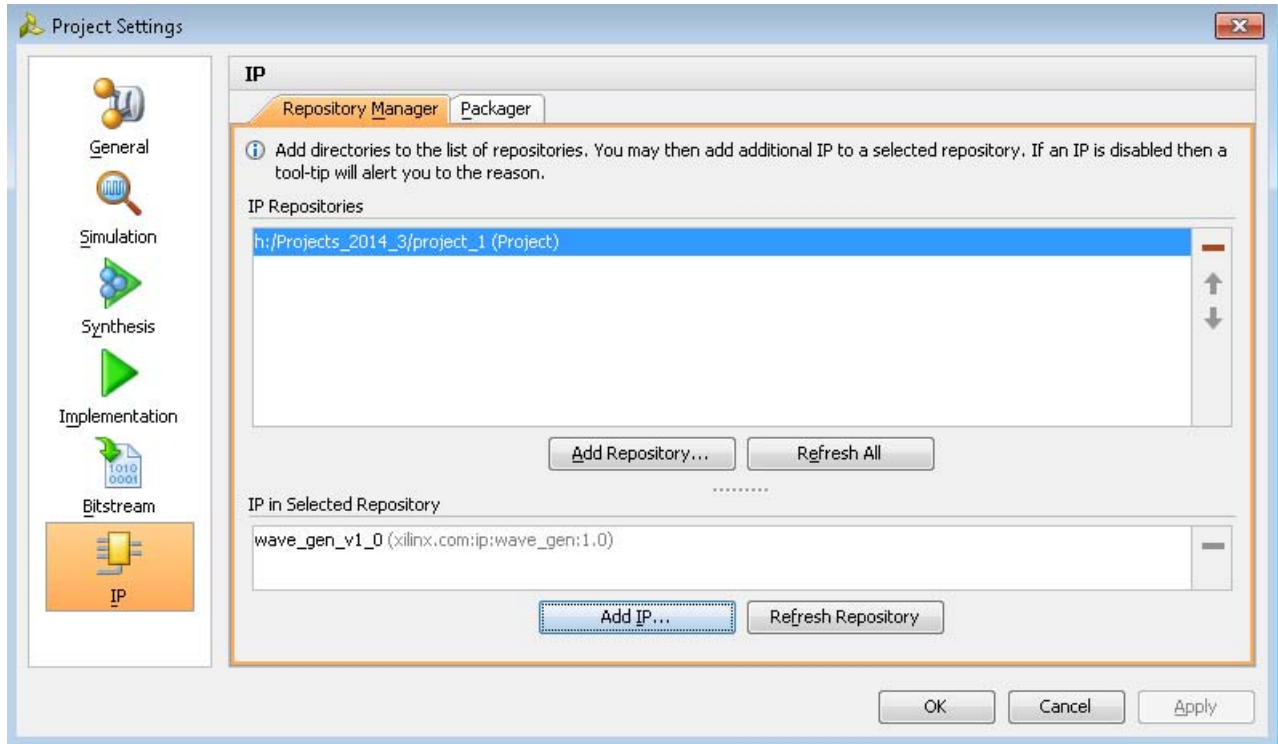


Figure 1-2: IP Project Settings: Repository Manager

Using the Packager Settings

To set the Packager options:

1. Click the Packager tab as shown in the following figure:

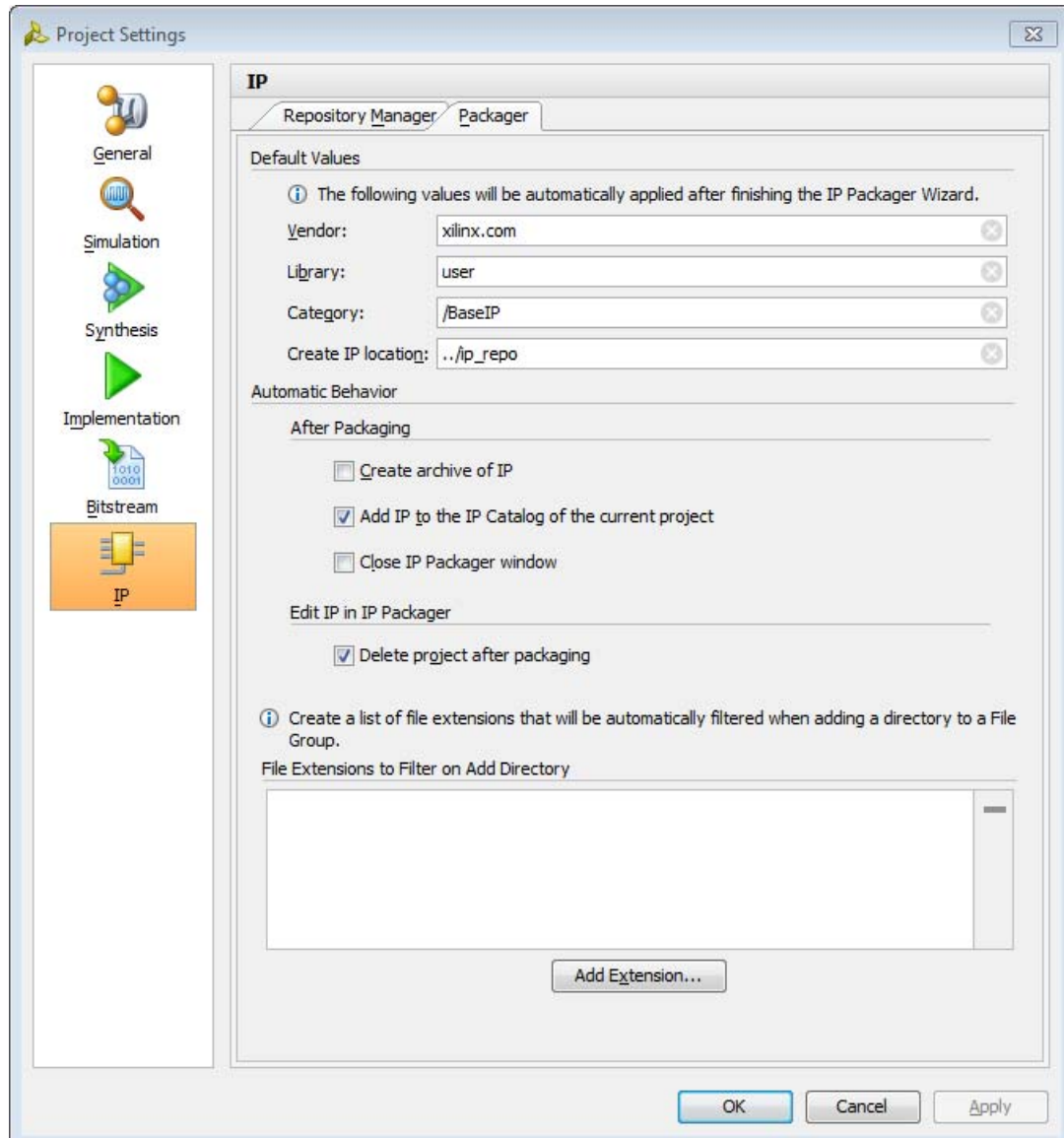


Figure 1-3: IP Project Settings—Packager Tab

2. Fill out the following information:

- In **Default Values**, set the following options:
 - **Vendor**: Sets the vendor name to use when packaging a new IP. This is, for example, the top domain name of a company.
 - **Library**: Sets the associated library for the IP. This category, along with the Vendor are used in conjunction with the IP name to create a unique identifier.
 - **Taxonomy**: Specifies the categories in the IP Catalog in which to place the IP. For example, /BaseIP.

Note: If necessary, you can change the default values for packaging IP during the IP packaging process.

3. In **Automatic Behavior**, check or uncheck the options you want:
 - **After Packaging:**
 - **Close IP Packager window:** Closes the Package IP window automatically when IP packaging is complete.
 - **Add IP root directory to current project's IP repository paths:** Adds the current IP to the IP repository.
 - **Create archive of IP:** After packaging an IP, automatically create an archive (ZIP format) of the IP.
 - **Edit IP in IP Packager:**
 - **Delete project after packaging:** Removes the iterative editing project after the IP is re-packaged.
 - In **Filtered Extensions**, add extensions (for example .txt) to automatically filter when selecting a directory to include in a **File Group** when packaging an IP.

Using the Creating and Package IP Wizard

Introduction

The Vivado® Integrated Design Environment (IDE) Create and Package IP wizard lets you create and package the following:

- IP using source files and information from a Vivado Design Suite project
- IP from a specified directory
- A template AXI4 peripheral that includes:
 - HDL files
 - Drivers
 - A test application
 - A bus functional model (BFM) (which requires special licensing)
 - An example template

The Create and Package IP wizard can generate Xilinx-supported AXI interfaces. These are:

- **AXI4:** For memory-mapped interfaces, which allows burst of up to 256 data transfer cycles with a single address phase.
- **AXI4-Lite:** A light-weight, single transaction memory-mapped interface.
- **AXI4-Stream:** For high-speed streaming data.

For more information on the Xilinx adoption of AXI, see the Vivado *AXI Reference Guide* (UG1037) [Ref 16].

Using the Create and Package IP Wizard

The Create and Package New IP wizard takes you step-by-step through the IP creation and packaging steps.

To run the Create and Package New IP wizard:

1. From the **Tools** menu, select **Create and Package IP**, as shown in the following figure.

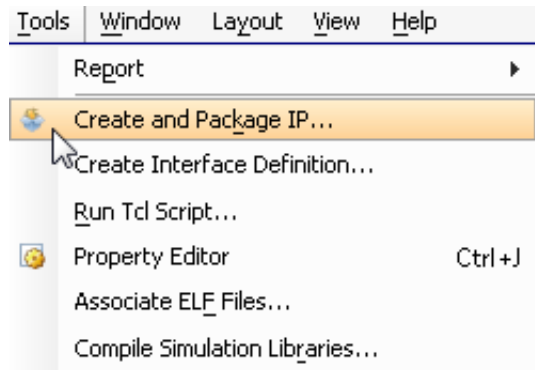


Figure 2-1: Create and Package IP Option 09/17

The first page of the Create And Package IP wizard opens, as shown in the following figure.

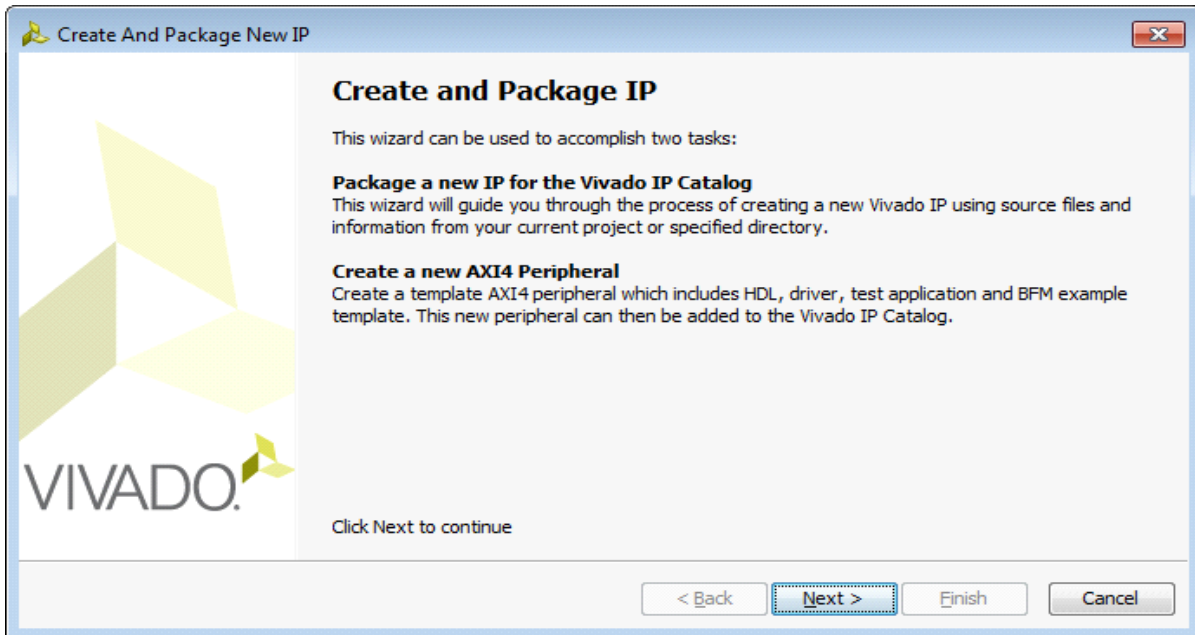


Figure 2-2: Create and Package IP Dialog Box

Use the wizard to accomplish one of these tasks:

- **Package a new IP for the Vivado IP Catalog:** Guides you through the process of creating a new Vivado IP using source files and information from your current project or specified directory.
- **Create a new AXI4 Peripheral:** Create a template AXI4 peripheral which includes HDL, driver, test application, and BFM example template.



IMPORTANT: *You must acquire a BFM licensing file if you intend to use BFM models.*

2. Click **Next**.

The Choose Create Peripheral or Package IP dialog box opens, (Figure 2-3).

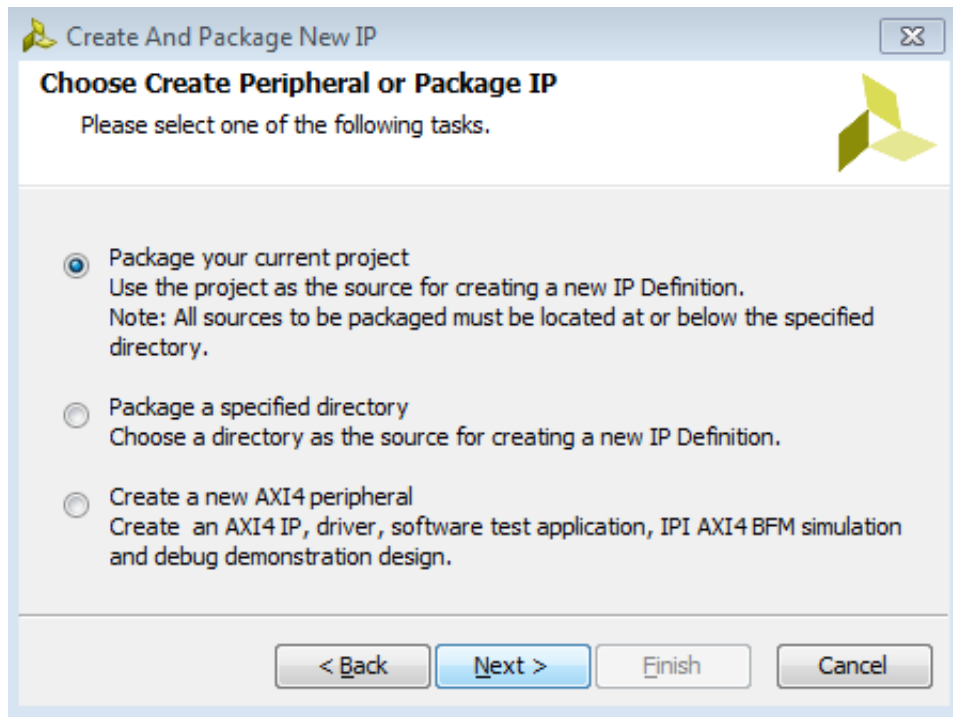


Figure 2-3: Choose Create or Package IP Dialog Box

3. Select from the options:
 - **Package your current project:** See [Packaging Your Current Project](#), page 15.
 - **Package a specified directory:** See [Packaging a Specified Directory](#), page 17.
 - **Create a new AXI4 peripheral:** See [Creating a New AXI4 Peripheral](#), page 19.
4. Make your selection, and click **Next**.

The next dialog box option differs, based upon the **Choose Create or Package** option you selected.

Packaging Your Current Project

The **Package Your Current Project** option lets you package the files associated with your current Vivado project. The IP packager attempts to gather the necessary information about your IP and create a basic IP package in the staging area.

When you select this option, the dialog updates with the available options for packaging the current project.

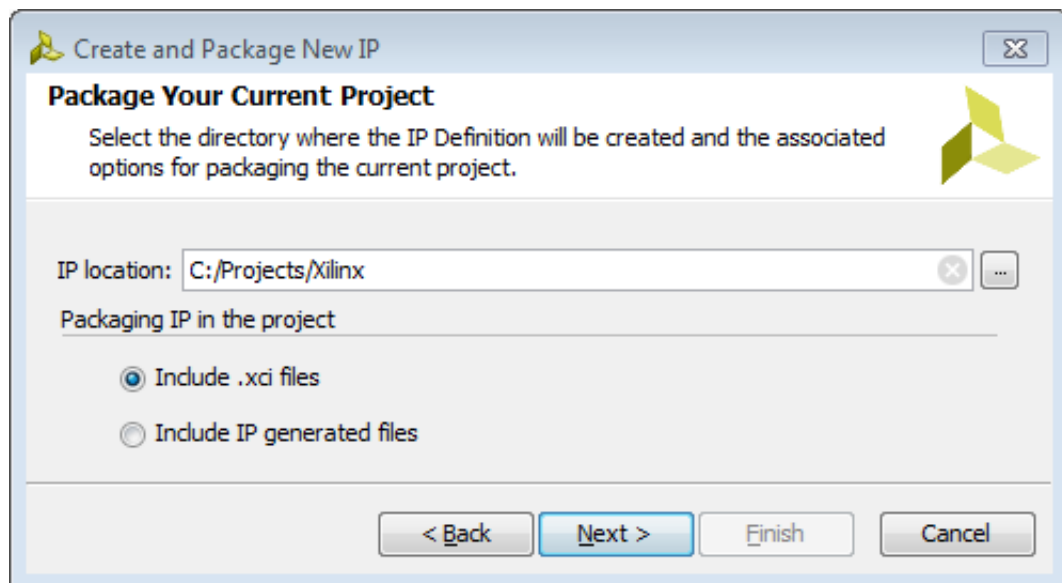


Figure 2-4: Create and Package New IP: Package Your Current Project

- In the Package Your Current Project dialog box, (Figure 2-4), make your selections from the following options:
 - IP Location:** The directory in which the IP Packager creates the IP Definition. The default is the project sources directory.
 - Packaging IP in the project:** If the project you are packaging includes IP, the following options determine how the IP is included in the newly packaged IP. See [Packaging IP in the project, page 16](#) for more information.
 - Include .xci files:** Packages only the IP customization file. The Vivado IDE generates the IP output products with the newly created parent IP.
 - Include IP generated files:** Packages the generated HDL and XDC sources from the IP customization.

Selecting an IP Location

The IP Packager gives the user options on how they would like to package files in the current project. Generally, the IP location is the `/sources` directory of the project.

This location, if all the files are copied into the project, can relatively reference all the project files. If files are stored remotely from the project source directory, the IP location is determined based upon where the majority of project files are relatively referenced.

If a location is selected outside of the hierarchical file paths of the project files, the Vivado IDE prompts you to copy the project source files into the indicated IP location directory. This process copies all the remote files to the IP location into a directory based on their category (for example, `src/`, `sim/`).

The Vivado IDE creates a new temporary editing project in the IP location for editing and modifying the newly created packaged project.

Packaging IP in the project

When including only the XCI files in the packaged IP, this creates an association between the parent IP and enables the packaged XCI files to be managed by the Vivado IDE.

The advantage of the Vivado IDE managing the packaged XCI is that the IP can be upgraded to the latest release by using the IP upgrade instructions as described in *Vivado Design Suite User Guide: Designing with IP* (UG896)[Ref 6].



IMPORTANT: *As only one version of Xilinx IP is delivered in each release, the parent IP can become locked if the associated XCI has a new release. This requires that you repackage the parent IP with an upgraded XCI from the latest Vivado Design Suite.*

In the case of including the IP generated files, all the generated HDL and XDC output products of the IP customization are packaged. This removes any reference of the original IP customization and treats the IP as project source files.

2. Click **Next**.

The New IP Creation page summarizes the information that the Create and Package IP wizard heuristically gathers about the design.

3. Click **Finish** to complete packaging and open the IP Packager.

The *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 6] describes the XCI files and IP output products.

When you include the IP generated files, all the generated HDL and XDC output products of the IP customization are packaged. This removes any reference of the original IP customization and treats the IP as project source files.

Packaging a Specified Directory

The packaging a specified directory option lets you package the files underneath a specific directory within the file system. When specifying a directory for packaging, there are inference rules which assist in packaging the IP correctly. The following table describes the directory structure recommended for inferring an IP:

Table 2-1: Directory Inference Recommendation

Source Type	Directory Inference
Synthesizable Sources	src/, hdl/
Simulation Sources	sim/, simulation/
Example Sources	example/, ex/, examples
Testbenches	testbench/, tb/, test/
C Sim Models	cmodel/, c/
Documents	docs/, doc/, documents/

Using the directory structure described in [Table 2-1](#), the IP packager attempts to populate the contents into each corresponding file group. In the synthesizable sources directory, the files are filtered by the .sv, .v* and .xdc extensions. For all other directories, files are populated.

If the directory structure of the specified directory cannot be recognized, the IP packager recursively searches the synthesizable source files and adds files to the synthesis and simulation file groups.

When you select the **Package a Specified Directory** option, the Package Your Current Project dialog box opens, as shown in the following figure.

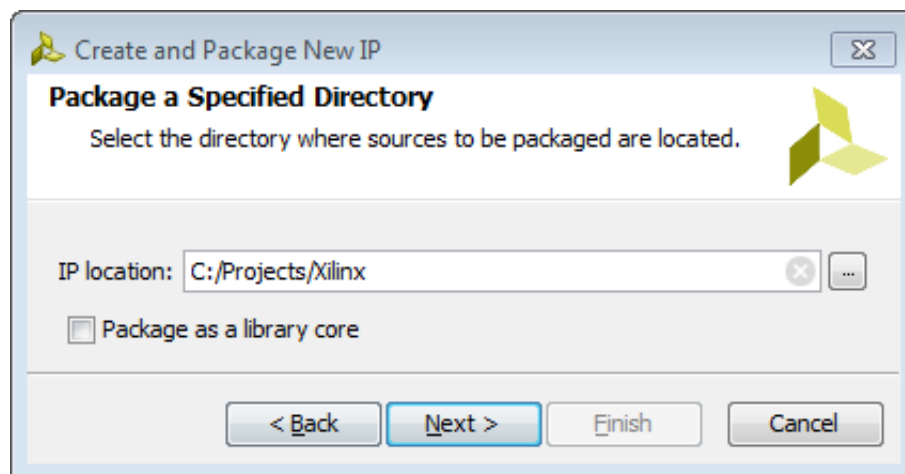


Figure 2-5: Create and Package New IP: Package a Specified Directory

1. Make your selections from the following options:
 - **IP Location:** The design directory containing the related files for the IP.
 - **Package as a library core:** Defines the IP as a library core, which is IP that is available in the IP repository, but is not visible in the IP Catalog.



IMPORTANT: Use the Packaging as a library core for IP that is not to be used as a standalone IP. This option lets you reference the IP from the IP Repository, but does not make the IP visible in the IP Catalog.

2. Click **Next**.
3. In the **Edit in IP Packager Project Name** dialog box, as shown in the following figure, set the following information:
 - **Project name:** Name of the project created with the generated IP definition.
 - **Project location:** Directory where the design sources exist and the Edit IP Packager project is located.



RECOMMENDED: Keep file path lengths under 80 characters.

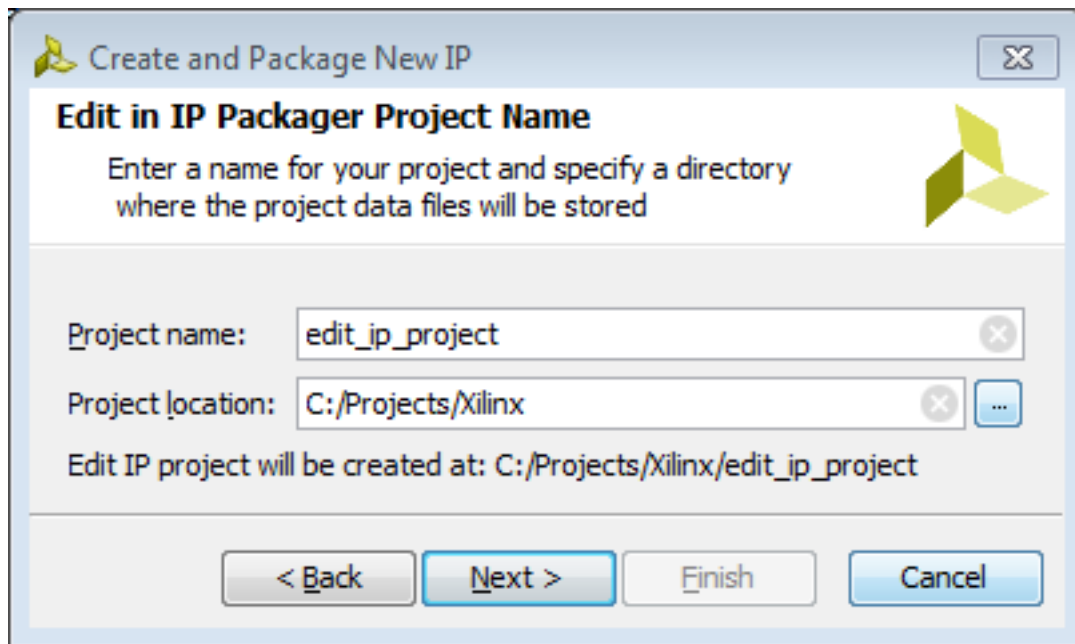


Figure 2-6: Create and Package New IP: Edit in IP Packager Project Name

4. Click **Next**.
The New IP Creation page summarizes the information that the Create and Package IP wizard heuristically gathers about the design.
5. Click **Finish** to complete packaging and open the Edit IP Packager project.

Creating a New AXI4 Peripheral

To create a new AXI4 peripheral:

1. From the Choose Create Peripheral or Package IP dialog box, select **Create a new AXI4 peripheral**, and click **Next**.
2. Enter the IP peripheral details:
 - **Name:** The name of the IP.
 - **Version:** IP version that reflects the <major#.minor#.Rev#> version scheme.
 - **Display Name:** The name of the IP that shows in the IP Catalog.

You can have different names in the **Name** and **Display Name** fields; however, the **Display Name** must be intuitive enough that any change in **Name** can reflect automatically in the **Display Name**.

 - **Description:** The IP description to share with an end-user of the IP.
 - **IP Location:** IP packager automatically adds the IP repository location.
3. Click **Next**.
4. Add interfaces to your IP based on the functionality and the required AXI type, shown in the following figure.

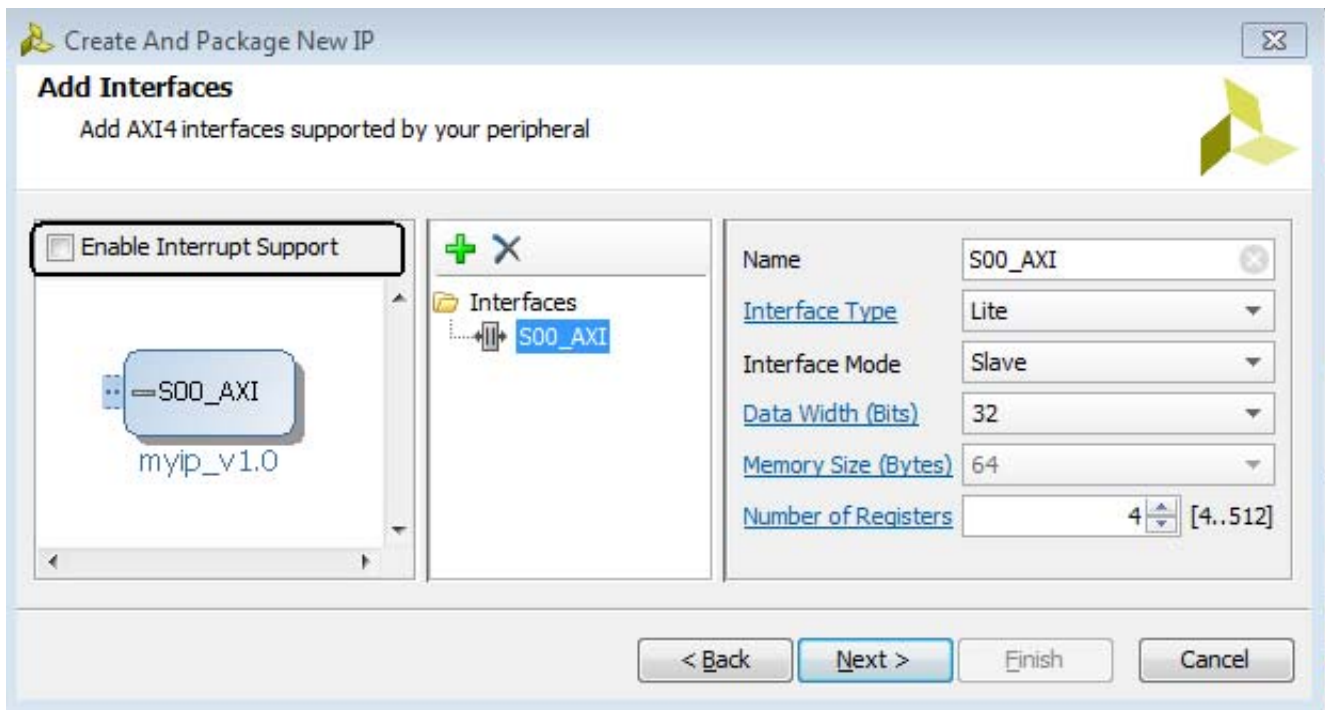




Figure 2-7: Create and Package New IP: Add Interfaces Dialog Box

- To include interrupts to be available in your IP, check the **Enable Interrupt Support** option (highlighted in [Figure 2-7, page 19](#)).

The figure shows that generated IP supports edge or level interrupt (generated locally on the counter) and those interrupts can be extended to input ports by user and IRQ output.

- Add an interface using the  mark.
- Delete an interface using the  mark.

The data width and the number of registers vary, based upon the AXI4 selection type.

- Click **Next**.
- Review your selections.

The details of your IP are listed in the final wizard page, as shown in the following figure.

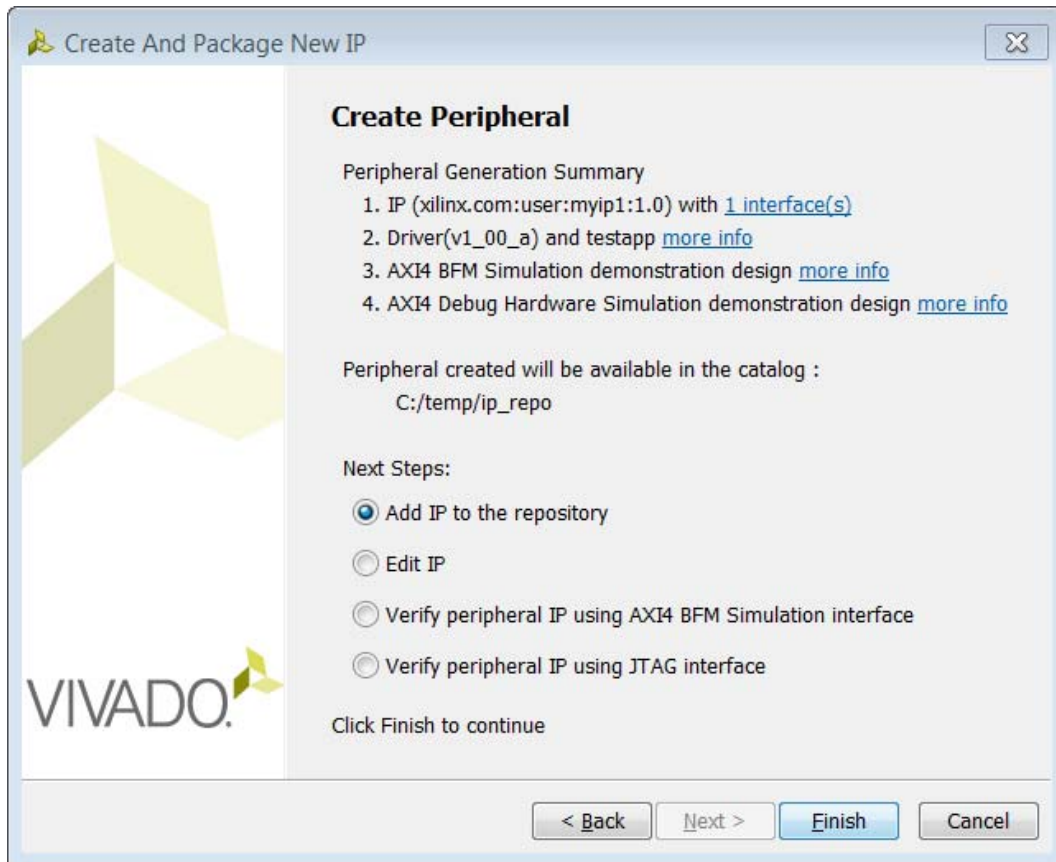


Figure 2-8: Create and Package New IP: Peripheral Summary Dialog Box

The following options are available after you generate the IP:

- **Add IP to the repository:** Lets you add IP to the IP repository.
- **Edit IP:** Lets you edit the IP.

- **Verify peripheral IP using AXI4 BFM Simulation Interface:** Lets you use an AXI4 BFM simulation interface (licensing is required to use AXI4 BFM simulation).
- **Verify peripheral IP using JTAG interface:** Creates a block design with which you can debug your IP module in hardware for a system with JTAG-to-AXI IP. See the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 18] for more information about the Vivado debug tools.

You can generate a bitstream and then validate the register writes and reads (from the sample Tcl script generated by the tool for your design) in the debug mode after the targeted device is programmed. You can do so by connecting to the board server from hardware manager, programming the board, and then sourcing the Tcl script. See the *Vivado Design Suite User Guide: Using Tcl Scripting* (UG894) [Ref 13] for more information.

After you create the peripheral, have the option to add custom logic and make the peripheral a custom IP. See *Vivado Design Suite Tutorial: Creating and Packaging Custom IP* (UG1119) [Ref 20] for a demonstration.

Packaging IP

Introduction

This chapter describes the features for adjusting and packaging a custom IP. The Vivado® IP packager is the interface between the Vivado integrated design interface (IDE) and the IP-XACT component file.

In the Vivado IP packager window, the following packaging steps are available to customize the custom IP definition:

- **Introduction**: The identification information for custom IP.
- **Compatibility**: The device support for custom IP.
- **File Groups**: The location and behavior of the files for custom IP.
- **Customization Parameters**: The parameters related to customizing IP.
- **Ports and Interfaces**: The list of top-level ports or interfaces of custom IP.
- **Addressing and Memory**: The address space and memory maps required for custom IP.
- **Customization GUI**: The GUI customization layout of the custom IP.
- **Review and Package**: A summary and packaging of the custom IP.

After packaging the custom IP definition, you can use the IP within your current project or point the custom IP definition repository path to use in another Vivado project.

Identification

The Identification page, ([Figure 3-1, page 23](#)), is the first section of the IP packager. The information is initially populated based on the information described in the **Project Settings > IP Settings**, and determined heuristically by the tool during the Create and Package IP wizard.

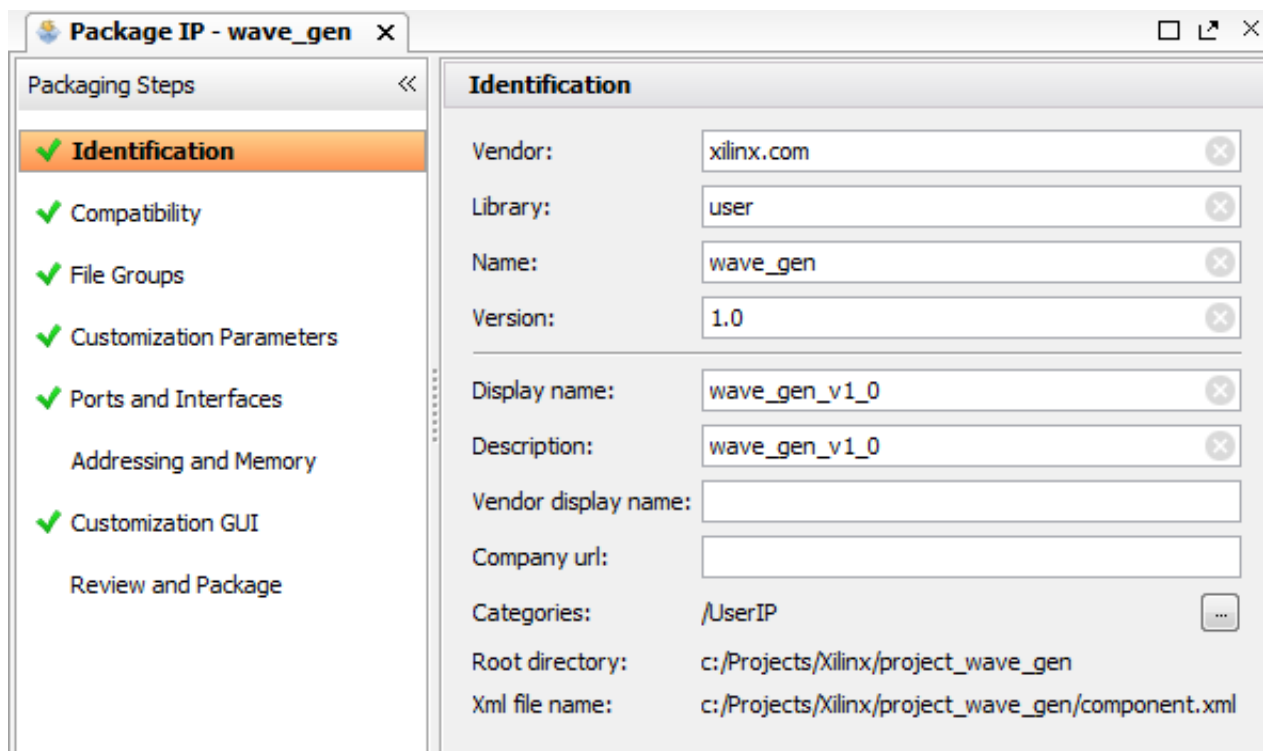


Figure 3-1: Package IP: Identification Page

The following fields are available to describe the identification of package IP:

- **Vendor:** The vendor of the IP. This is also the identifier for the vendor that displays in the VLVN of the IP definition.
- **Library:** The library in which the IP belongs. This is also the identifier for the library that displays in the VLVN of the IP definition.
- **Name:** The name of the IP. This is also the identifier for the name that displays in the VLVN of the IP definition.
- **Version:** The version of the IP. This is also the identifier for the version that displays in the VLVN of the IP definition.
- **Core Revision:** The IP core revision.
- **Display Name:** The Vivado IP Catalog display name.
- **Description:** The Vivado IP Catalog description.
- **Vendor Display Name:** The Vivado IP Catalog Vendor display name.
- **Company URL:** The Vivado IP Catalog display of the company URL.
- **Categories:** The list of category names in which the IP belongs.
- **Root Directory:** The working directory of the packaged IP. The directory controls both the location of the input and the output files.
- **XML File Name:** The name and location of the IP-XACT standard XML file.

The **Vendor, Library, Name, and Version (VLNV)** of the IP definition uniquely identifies the IP in the Vivado IP Catalog.



IMPORTANT: Only one VLNV can exist within the IP Repository. The IP names need to be concise with words separated by underscores.

Each IP within the IP Catalog have taxonomy for organization purposes, as described by the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 6]. These classifications are controlled by the categories set during IP packaging.

In the Categories list, each category is separated by the forward slash (/) character. Initially, Vivado defaults the custom IP to the UserIP category.

To add or remove categories for your IP, press the button in the **Categories** line of the IP Identification section to open the Choose IP Categories dialog box, as shown in the following figure.

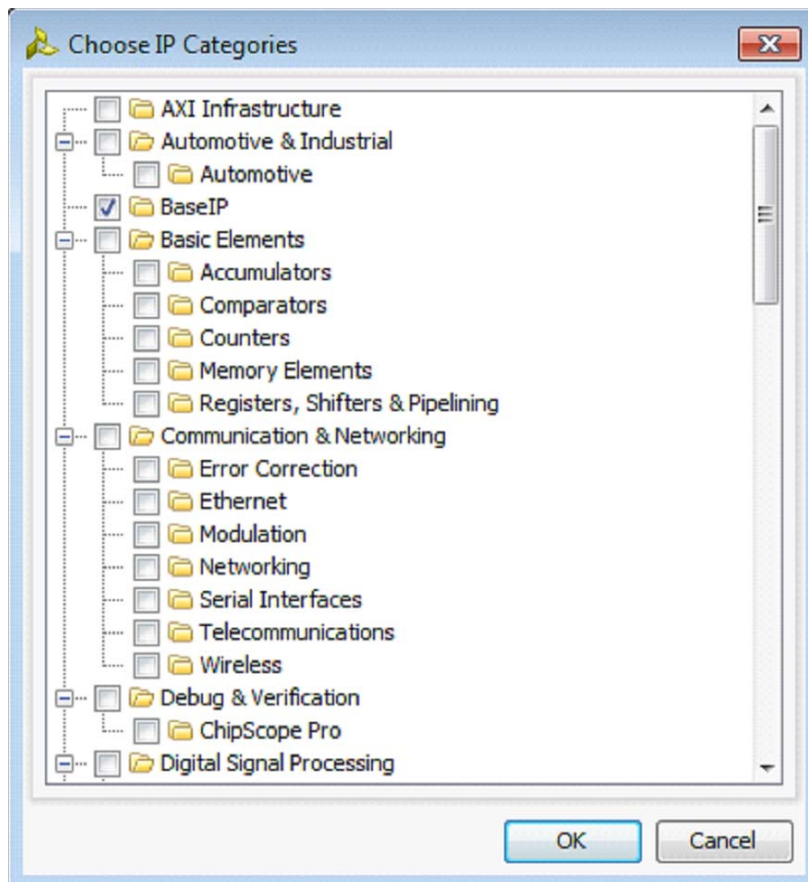


Figure 3-2: Choose IP Categories Dialog Box

Select the categories from the IP Definition to display in the Vivado IP Catalog.

Compatibility

The IP Compatibility section, (shown in the following figure), configures the specific Xilinx parts or device families compatible with your custom IP.

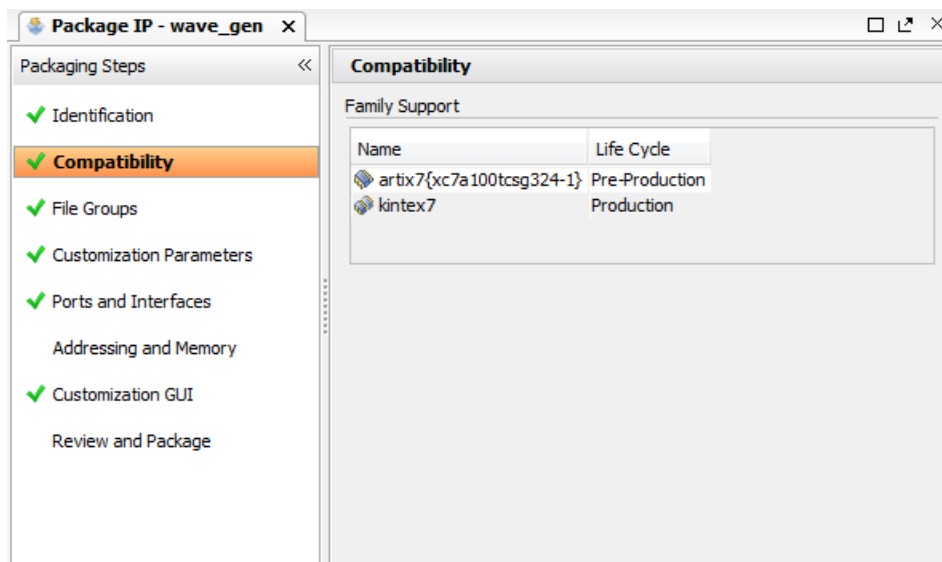


Figure 3-3: Packing IP: Compatibility

The information is initially populated with the 7 series Kintex[®] device family with the **Life Cycle** set to **Production**. Any Xilinx family and/or devices supported by the Vivado IDE can be included within the family support list of custom IP.

The Life Cycle properties inform the end-user of the IP compatibility with the selected IP use case. Within the Family Support list, each family or part added can have their own Life Cycle property for specific granularity in describing the IP compatibility with the selected family or part. Any parts or families added are set automatically to a Life Cycle of **Pre-Production**.

Adding a Xilinx Family or Part

1. In the Family Support bounding box, right-click and select **Add Family**, (Figure 3-4, page 26).



Figure 3-4: Add Family Option

2. In the Choose Family Support window, shown in the following figure, select the desired families or parts available in the Vivado IDE in the **Manual** selection mode.

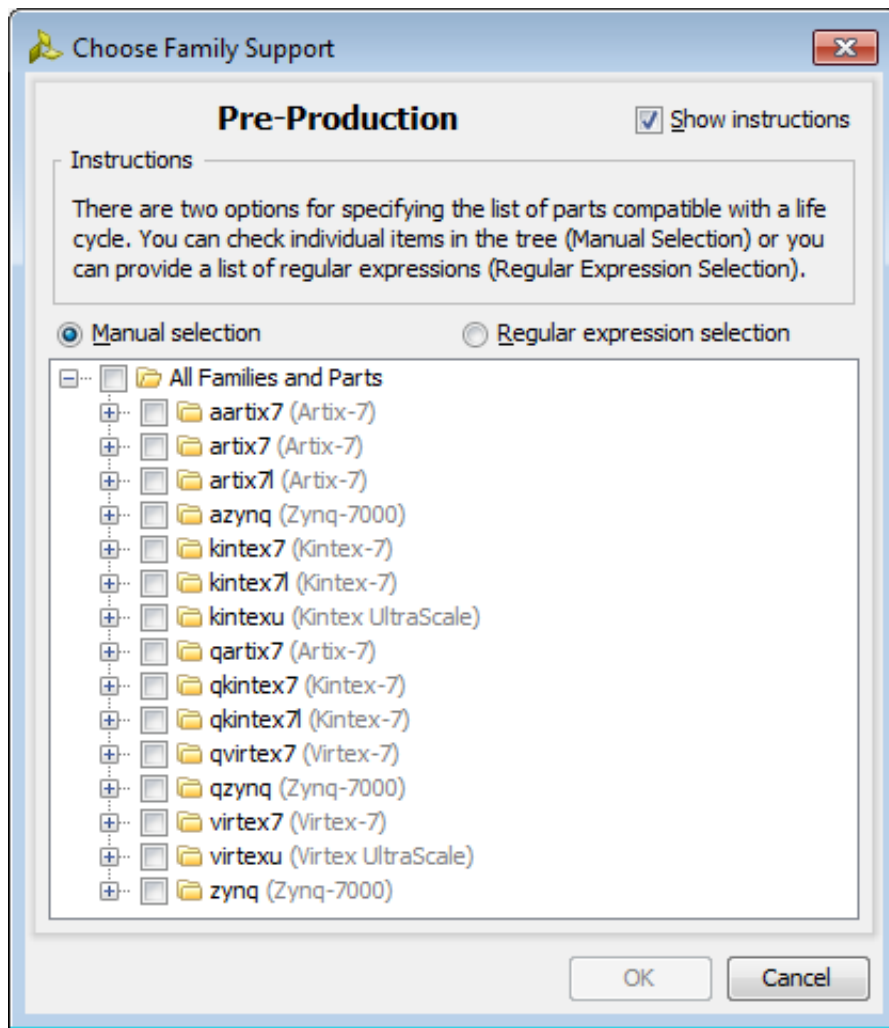


Figure 3-5: Choose Family Support Dialog Box: Manual Expression

3. Add a part using either the **Manual selection** or the **Regular expression selection**.

Optionally, besides directly selecting parts or families compatible with the custom IP, the Regular expression selection mode shown in the following figure, lets you add specific parts or families through a regular expression.



Figure 3-6: Regular Expression Selection

To add a regular expression selection to the search, add the required syntax in the **Expression** text field. When finished creating the expression, click **Add** to add the regular expression to the expression list.

The syntax for the regular expression search is `familyName{regex}`. For example:

```
virtex7{xc7v[hx].*}
```

This regular expression example returns all Virtex®-7 XT and Virtex®-7 HT devices.

Setting Life Cycle Properties

To set the Life Cycle Properties:

1. Select the Life Cycle value for the family, part, or regular expression and you can select a new Life Cycle from the drop-down menu.

Alternatively, you can right-click and choose the **Select Set Life Cycle** from the selected item in the list and choose a **Life Cycle**.

The following choices are available to describe the Life Cycle for a given part or family.

- Beta
- Discontinued
- Hidden
- Pre-Production
- Production
- Removed
- Superseded

Setting the property to **Discontinued**, **Hidden**, **Removed**, or **Superseded** ensures that the IP does not appear in the IP Catalog.

File Groups

The File Group step, shown in the following figure, provides a listing of files of your custom IP.

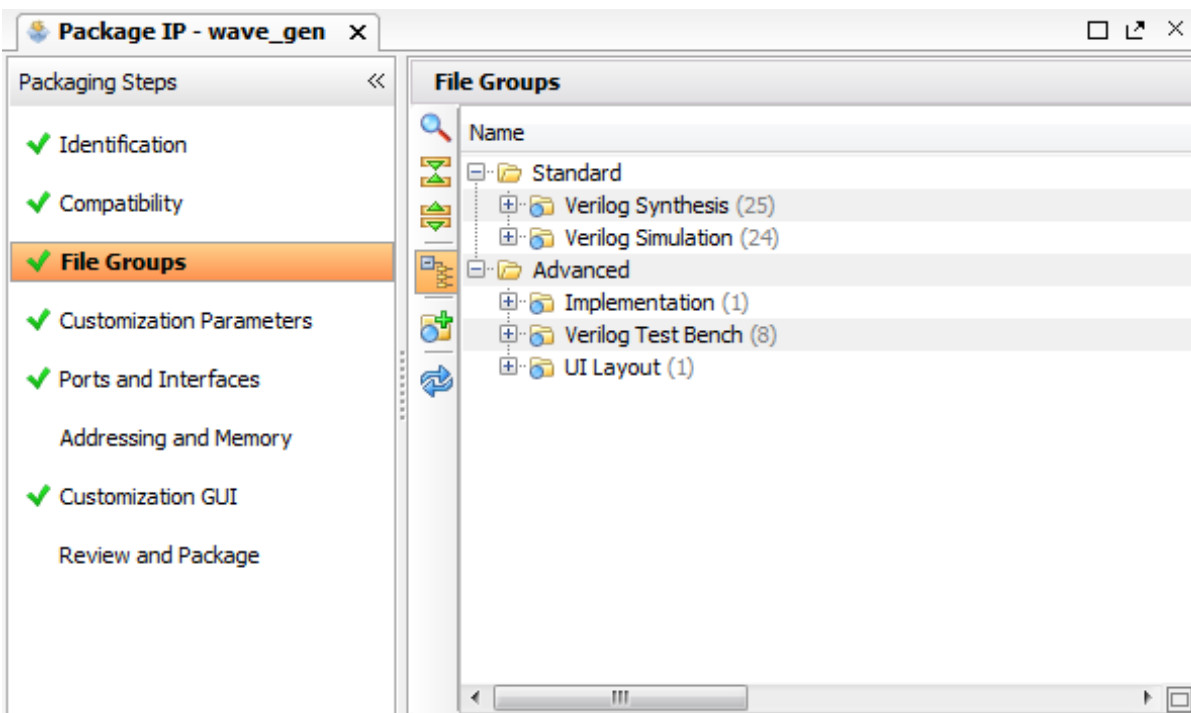


Figure 3-7: Package IP – File Groups

Each file is grouped into specific file groups that alter the files behavior and use.

For example, a file located in the Synthesis file group is used for synthesis, but not simulation.

The files locations in the specific file groups are initially determined when you run the Create and Package IP wizard. The wizard determines the file groups, either by:

- Using the file sets of the current project.
- Heuristically determining the information from the directory structure of the IP location directory.

Two sections exist for separating files into different file groups:

- **Standard:** Table A-1, page 75 describes these options. For many custom IP, the **Standard** file group list contains the necessary file groups required for packaging an IP for reuse.
- **Advanced:** Table A-2, page 76 describes the advanced features options. If you want to add additional advanced features, the **Advanced** file groups are available.

The file group types are defined by a specific category. The groups are initially collapsed at the name of the file group followed by a number in parenthesis. This number corresponds to the total number of files in the file group. Expand the file group to expose the list of associated files with the specific file group.

The expanded IP File Groups window shows the list of files as illustrated in Figure 3-8, page 29.

Name	Library Name	Type	Is Include	File Group Name	Model Name
Standard			<input type="checkbox"/>		
Verilog Synthesis (4)			<input type="checkbox"/>		wave_gen
src/wave_gen_timing.xdc		xdc	<input type="checkbox"/>	xilinx_verilogsynthesis	
src/dk_core/dk_core.xci		xci	<input type="checkbox"/>	xilinx_verilogsynthesis	
src/dlogb2.vh		verilogSource	<input checked="" type="checkbox"/>	xilinx_verilogsynthesis	
src/wave_gen.v		verilogSource	<input type="checkbox"/>	xilinx_verilogsynthesis	
Verilog Simulation (4)			<input type="checkbox"/>		wave_gen
Advanced			<input type="checkbox"/>		
Implementation (1)			<input type="checkbox"/>		
Verilog Test Bench (8)			<input type="checkbox"/>		test_wave_gen
UI Layout (1)			<input type="checkbox"/>		

Figure 3-8: File Groups List Expansion

The following properties in the column list are associated with the files of the file group:

- **Name:** File name within the hierarchy tree.
- **Library Name:** Determines the library name used by Vivado synthesis or simulation.
- **Type:** Type of the file in the file group (for example; VerilogSource, XDC, or TclSource).
- **Is Include:** Mark the file as an include file (for example, a Verilog Header file).
- **File Group Name:** The file property to determine to which file group the file is associated. This property is read-only.
- **Model Name:** Top-level design name.



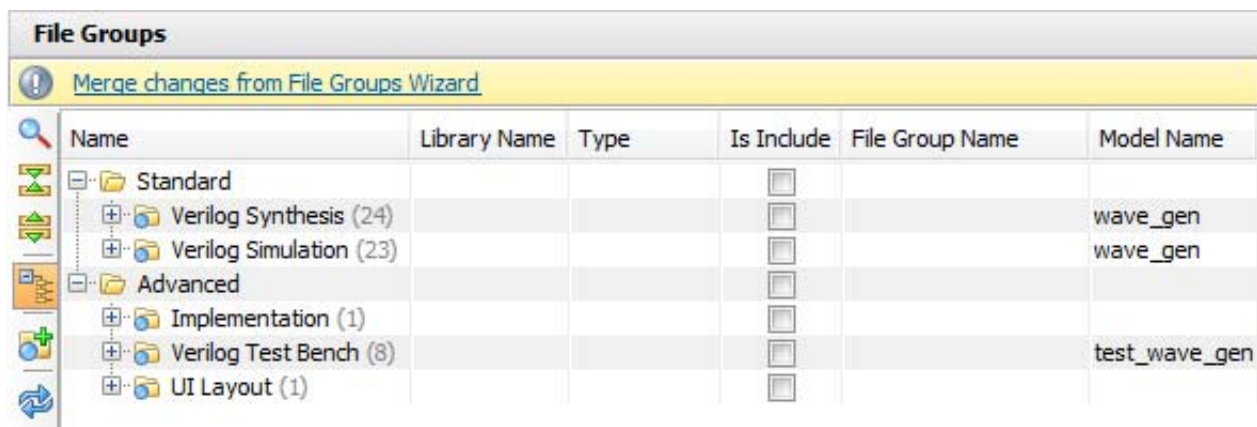
IMPORTANT: *The Model Name is a required property and applies directly to the file group. This value is set of any Synthesis, Simulation, and Implementation file groups.*

Adding Files to File Groups

Add files to a project with the Package IP window open. After you add the files, the updated files can be merged into the IP packager. For more information on how to add files to your Vivado project, see *Vivado Design Suite User Guide: System-Level Design Entry* (UG895) [Ref 15].

When you add files to your Vivado project with the Package IP window open, the File Groups Wizard hyperlink, opens, as shown in the following figure, which imports or removes files associated with the Vivado project.

Similar to the Create and Package IP wizard, the imported files are associated with the correct file group.



File Groups						
Merge changes from File Groups Wizard						
	Name	Library Name	Type	Is Include	File Group Name	Model Name
	Standard			<input type="checkbox"/>		
	Verilog Synthesis (24)			<input type="checkbox"/>		wave_gen
	Verilog Simulation (23)			<input type="checkbox"/>		wave_gen
	Advanced			<input type="checkbox"/>		
	Implementation (1)			<input type="checkbox"/>		
	Verilog Test Bench (8)			<input type="checkbox"/>		test_wave_gen
	UI Layout (1)			<input type="checkbox"/>		

Figure 3-9: File Group Wizard

You can manually add files to the IP packager, but this is not recommended as the wizard ensures that the files are located in the correct file group as well as relatively referenced in the package.



IMPORTANT: *Ensure that you organize necessary files for the IP in the appropriate file group type. If you add a file to an incorrect file group, the IP might not build or work properly.*

1. To manually add a file, right-click the respective file group, and select **Add Files**.
2. In the Add IP Files dialog box, add or create the files for the file group, and click **Finish**.

Note: The Add IP Files dialog box only allows files to one file group at a time.

Copying Files from a File Group

For files that already exist in a file group, you can copy a file or file group using the **Copy To** option, which extends the list of available file groups. The list contains the file groups listed within the File Groups window as well as additional, commonly used groups.

For a single file, selecting one of the files in the list copies the file to the specified file group. If selecting the file group, the **Copy To** option copies the child files under the file group to the specified file group.

1. In the File Groups window, select a file or file group, right-click and select **Copy To**.
2. Select the destination file group from the list in the extension menu.

Adding a File Group

1. In the File Groups window, right-click and select **Add File Group**.

Note: Alternatively, you can click the **Add File Group** button on the sidebar .

2. In the Add IP File Group window, shown in the following figure, select the file group type to add from the table.

Optionally, select the **show hyperlink**, shown in the following figure, in the Advanced header of the Add IP File Group window to display the list of Advanced file groups.

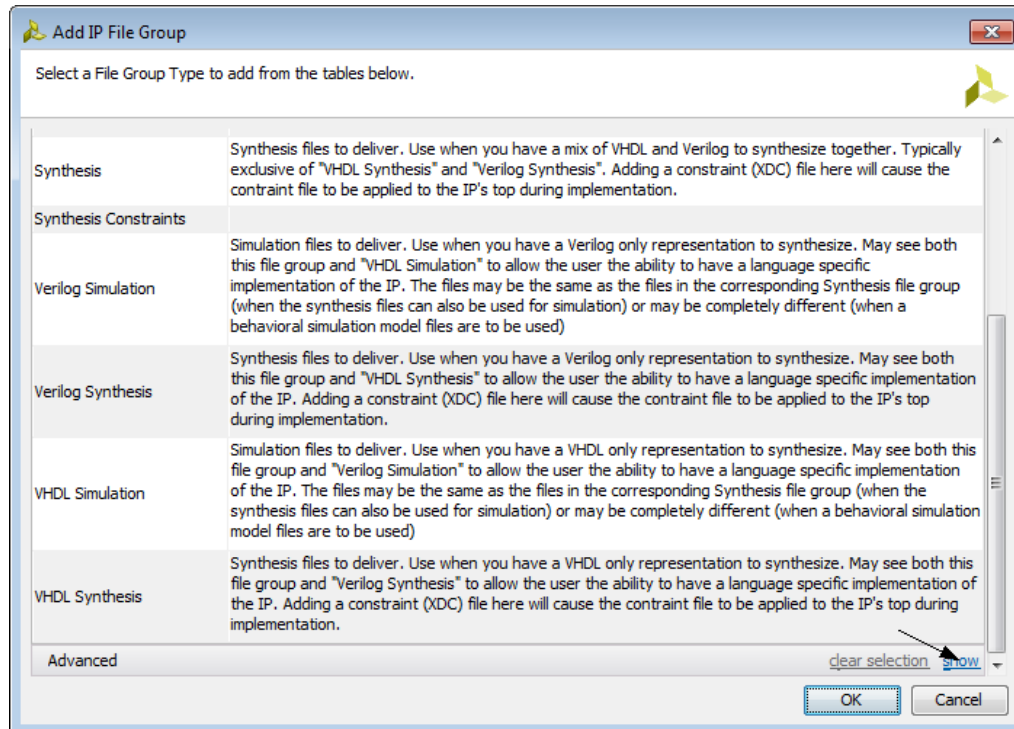


Figure 3-10: Add IP File Group

Customization Parameters

The IP Customization Parameters step, shown in the following figure, provides a listing of parameters of your custom IP. You can use these parameters specifically for GUI customization or customization through the top-level HDL source file.

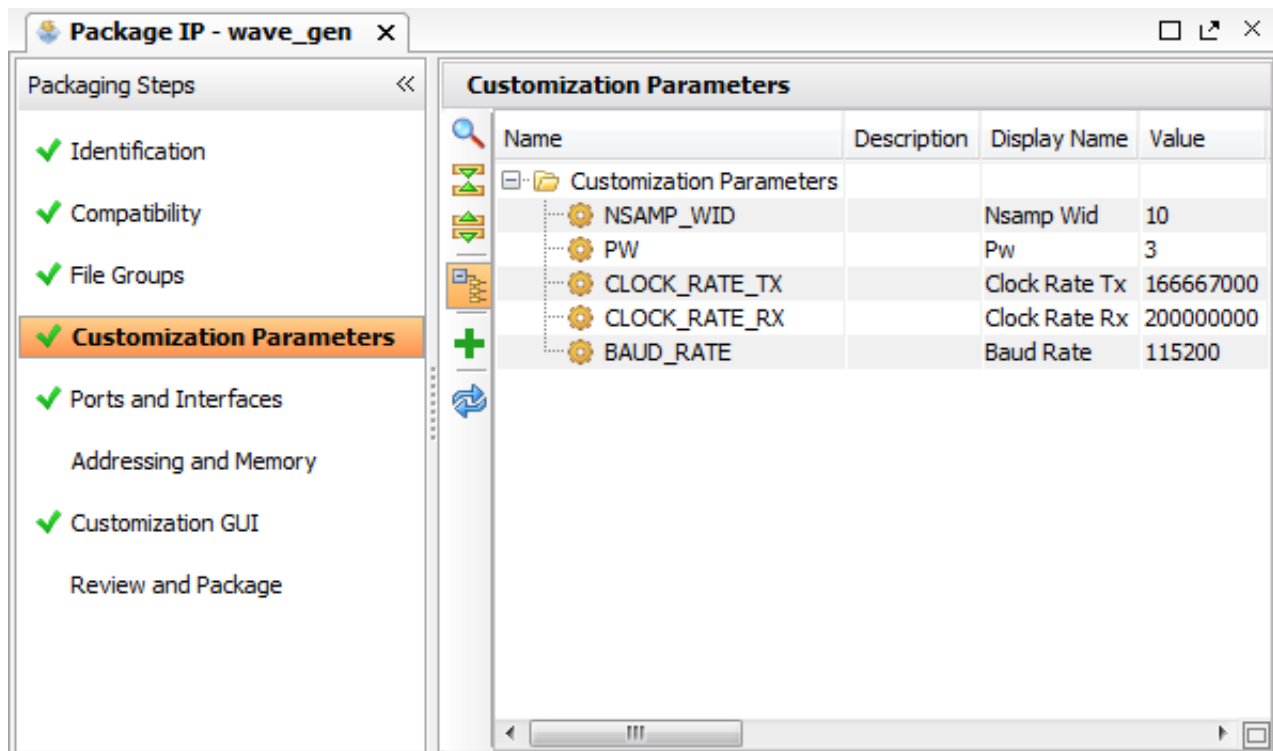


Figure 3-11: Package IP - Customization Parameters

After completing the Create and Package IP wizard, the customization parameters list is populated based on the parsing of the top-level HDL source file.

There are two folders for parameters displayed in the Customization Parameters list:

- **Visible:** Customization parameters shown in the IP Customization GUI.
- **Hidden:** Customization parameters not visible in the IP Customization GUI.

By default, parameters parsed from the wizard are visible and organized in the `/Customization Parameters` folder. Hidden parameters are not intended for direct editing. These parameters are generally dependent upon the visible parameters to determine their values.

Importing Parameters from Top-Level HDL Files

The parameters are initially determined during the Create and Package IP wizard. When you want to import parameters from the top-level HDL source file due to a change to the HDL after packaging, you can re-import the IP parameters.

1. From the Customization Parameters window, right-click and select **Import IP Parameters**.
2. From the HDL dialog box, select the following options from **Import Parameters**, and click **Finish**.
 - **Top-Level source file**: The top-level source HDL file that contains the top-level entity or module of the custom IP.
 - **Top entity name**: The name of the top-level entity or module that contains the parameters of the custom IP.
 - **Make all imported HDL parameters visible**: Sets all the imported parameters to be visible in the customization GUI. Unchecking this box places all the imported parameters into the `/Hidden` folder.

Adding or Removing a Parameter

You can create or remove parameters for use in IP GUI customization. These parameters, because they do not have a reference to an HDL parameter, must be visible.

Adding a Parameter

1. In the Customization Parameters window, right-click and select **Add Parameter**.

Note: Alternatively, click the **Add Parameter** button on the sidebar. 

2. In the Add New Parameter dialog box, select the name of the parameter, and click **OK**.

Note: This parameter name is not the display name of the parameter in the customization GUI.

After creating the new parameter, the Edit IP Parameter dialog box opens for specific parameter customization.

Removing a Parameter

You can remove any parameter from the list regardless if it is referenced from the top-level HDL by right-clicking the parameter and selecting **Remove Parameter**.

Note: Any parameter dependent on the removed parameter flags as an error.

Editing a Parameter

After you add the parameters to the IP Customization Parameters page, you can customize how the parameter displays in the IP Customization GUI. You can edit the display name, data format, default value, and other options.

To edit the parameter, open the Edit IP Parameter dialog box by right-clicking on a parameter and selecting the **Edit Parameter** from the context menu, as shown in the following figure.

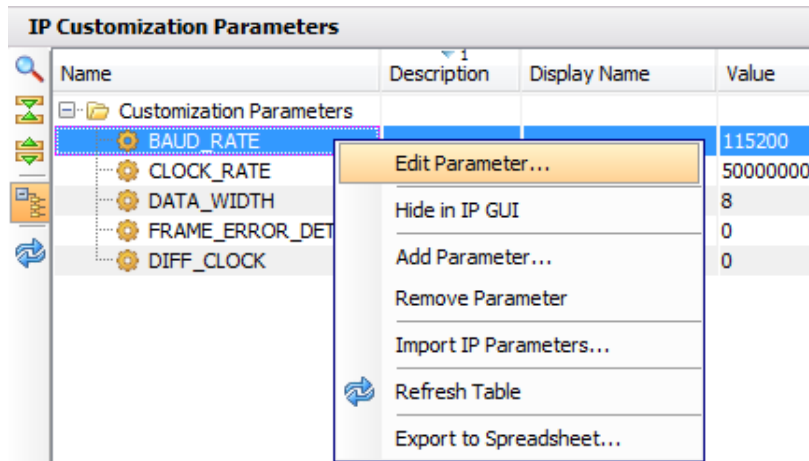


Figure 3-12: Edit Parameter Option

The Edit IP Parameter dialog box opens with content populated from the data that is parsed from either the Create and Package IP wizard, or the Import Parameters wizard. An example of an Edit IP Parameter dialog box with all available options is shown in [Figure 3-13](#), [page 36](#).

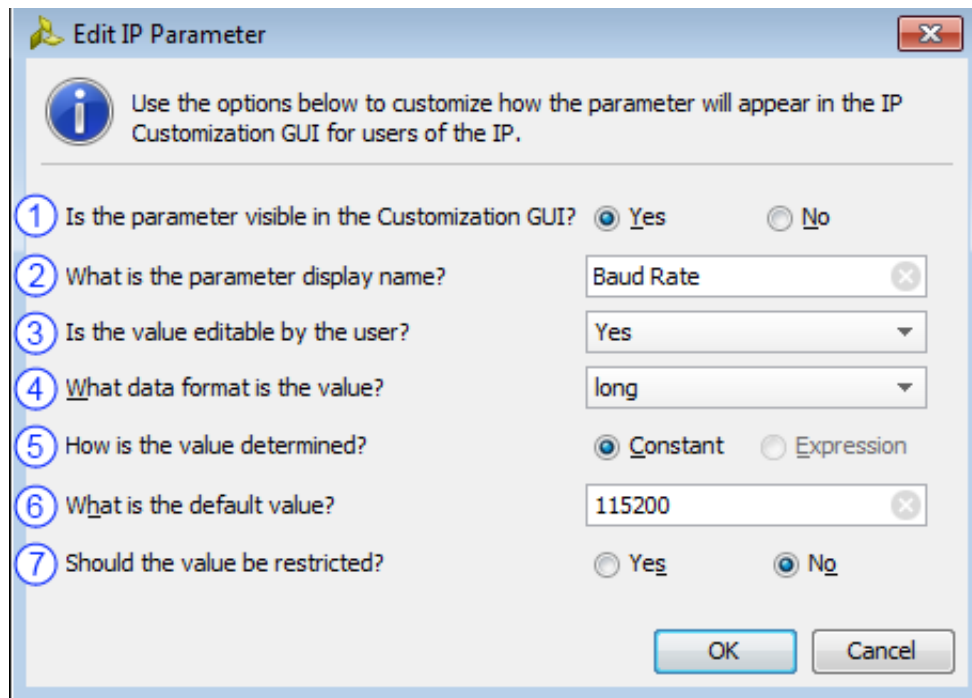


Figure 3-13: Edit IP Parameter Dialog Box

The available options depend on the selections you make within the dialog box. The following options are available:

1. **Is the parameter visible in the Customization GUI?** This option controls whether the parameter is visible or hidden in the Customization GUI of the IP.

A parameter populated from the top-level RTL has the option of being marked as No for visibility in the Customization GUI. This parameter can then be made dependent on other parameters visible during customization or set as a static value.

In the case that the parameter is not visible, the Edit IP Parameter dialog box limits the rest of the Edit IP Parameter customizations, as shown in [Figure 3-14, page 37](#).

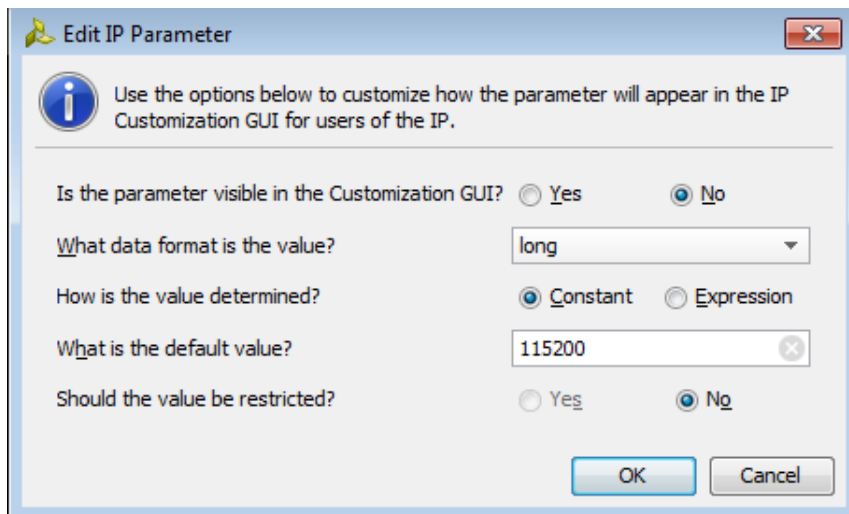


Figure 3-14: Edit IP Parameter Marked as Hidden

2. **What is the parameter display name?** This option controls the display name of the parameter in the IP Customization GUI. The Vivado IDE tool tries to heuristically determine a proper name from the parameter listed in the top-level RTL. When the IP is being customized, this is the text that displays next to the value that an end-user can set. This can be a simple name, or a small description of the parameter.
3. **Is the value editable by the user?** This option controls how the parameter is edited by the end IP user. The available selections for this option are: **Yes**, **Dependent**, and **No**.
 - **Yes** is the default setting, indicating you can edit the parameter directly in the Customization GUI.
 - When set to **No**, you cannot edit the parameter with the Customization GUI. If displayed, the parameter is read-only during customization. The value of the parameter is determined by a constant value or expression.
 - When set to **Dependent**, a text box displays for you to provide Parameter Enable Expressions, as shown in [Figure 3-15, page 38](#).

The parameter enablement depends upon previous decisions made in the IP Customization GUI. For example, you can make a frequency parameter editable by the end-user only if a certain protocol is selected.

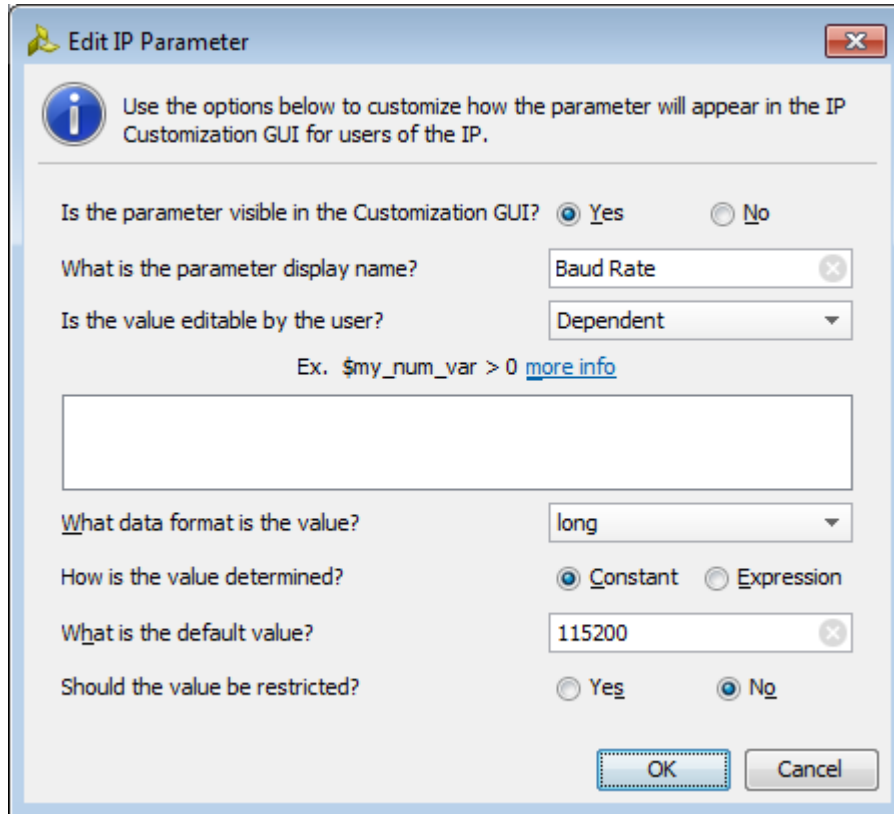


Figure 3-15: Edit IP Parameter Marked as Dependent

Parameter enablement is based upon Tcl expression rules.

For more information, see [Setting an Enablement Expression, page 63](#).

4. **What data format is the value?** Controls the format of the parameter data. The selections available are: `long`, `float`, `bool`, `bitString`, and `string`.

The `long` option is the default selection for a parameter. The data format selection determines the supported values the end-user can use to customize the IP.

Only the `bool` format, when selected, changes the Edit IP Parameter dialog box because the value becomes restricted. Either use all uppercase letters when describing the logic state of a signal or use all lowercase letters when describing the logic state of a signal value.

The following figure shows the resulting dialog box when you select a boolean format.

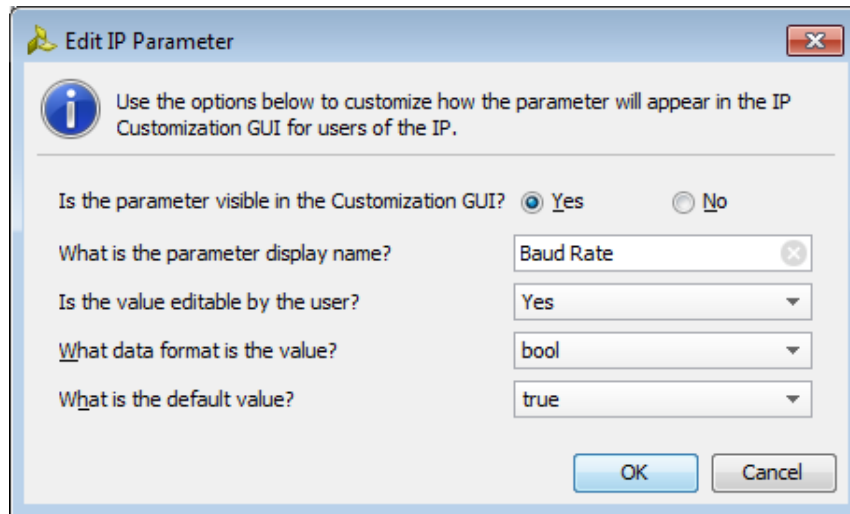


Figure 3-16: Edit IP Parameter Marked as Bool

5. **How is the value determined?** This option controls how the value is determined during IP customization.

The available selections for restricting the value of the parameters are: `Constant` (the default) and `Expression`.

- As a `Constant`, the value has a single, known value.
- When the value is not editable, or is set as `Dependent`, you can determine the value by an expression.
- When you select `Expression`, the dialog box adds another text box for the Tcl code required to evaluate the value, as shown in [Figure 3-17, page 40](#).

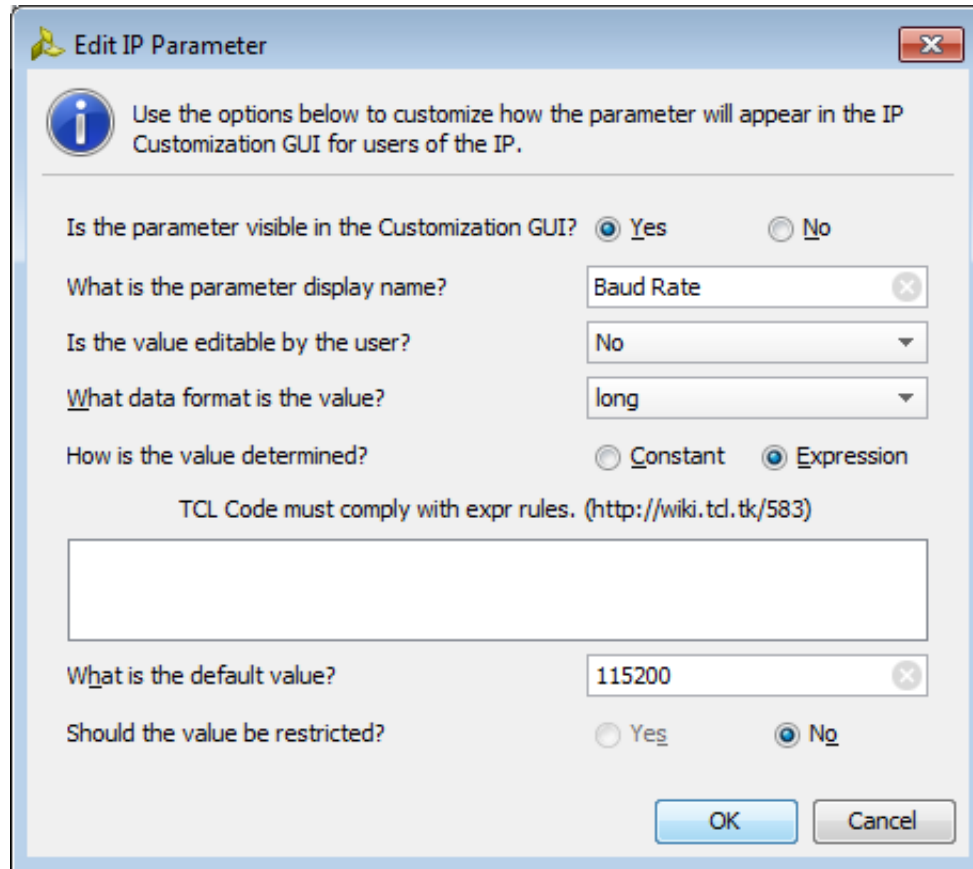


Figure 3-17: Edit IP Parameter with Expression option

The expression that evaluates the value can reference parameters defined through the IP packager.

For more information, see [Setting an Enablement Expression, page 63](#).

6. **What is the default value?** This option controls the default value for the specified IP parameter. If a user generates the IP without changing any of the customization, the value of the parameter is the one specified within this text field.
7. **Should the value be restricted?** This controls the bounds by which the parameter value can be set.
 - When the selection is `No`, the value can be any value in the selected data format. The IP Customization GUI *does not* ensure that the value is within the expected bounds for usage.
 - When the selection is `Yes`, the dialog box adds additional options on how to restrict the value, ([Figure 3-18, page 41](#)). The initial selection for the restriction is set to a list of values.

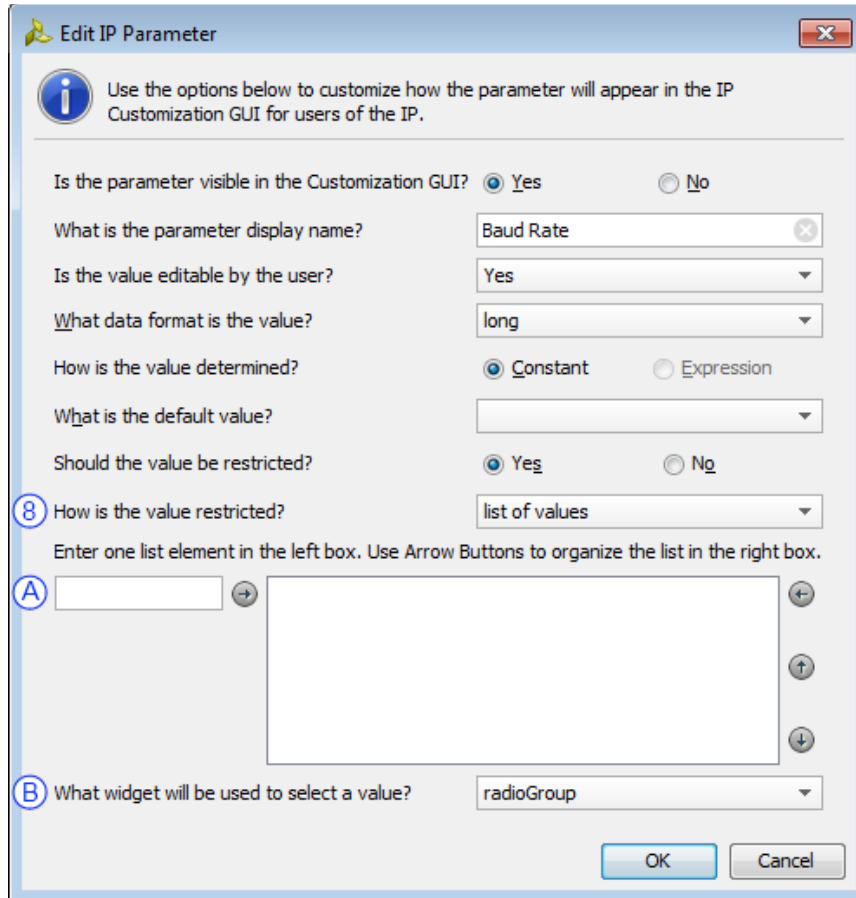


Figure 3-18: Edit IP Parameter Dialog Box with List of Value Restrictions

8. **How is the value restricted?** This option is only shown if you set the value to be restricted, and controls how the value is restricted in the IP Customization GUI.


The selections on for the restricting of the value of the parameter are: list of values, range of integers, or pairs. The dialog box changes depending on the values selected for this option.

- A. **List of Values:** Limits the value choices for the parameter from a predefined list.

To control the list of values, two text boxes open near the bottom of the dialog box. These text boxes control the valid list of values for the parameter.



- The text box on the left is the input field for the value list.
- The text box on the right is to organize the list of values.


The text boxes have four arrow icons within the surrounding area, as seen in [Figure 3-19, page 42](#).

To add a value to the list, enter the value in the input text field and click .

This button turns green when the text field contains a valid value.

After you place the set of values within the organization list, you can move the values up and down the list to control the order in which the values display in the IP Customization GUI.

The  and  buttons control the direction the value moves within the list by one. The button turn green only if the move is valid for that direction.

To edit or remove a value from the list, select the value of the list to modify and click the  button. This moves the value from the list back to the editable input text field.

After you complete the list of values, you can adjust the default value to select the value from the restriction list. As an example, a dialog box with a list of values organized is shown in the following figure.

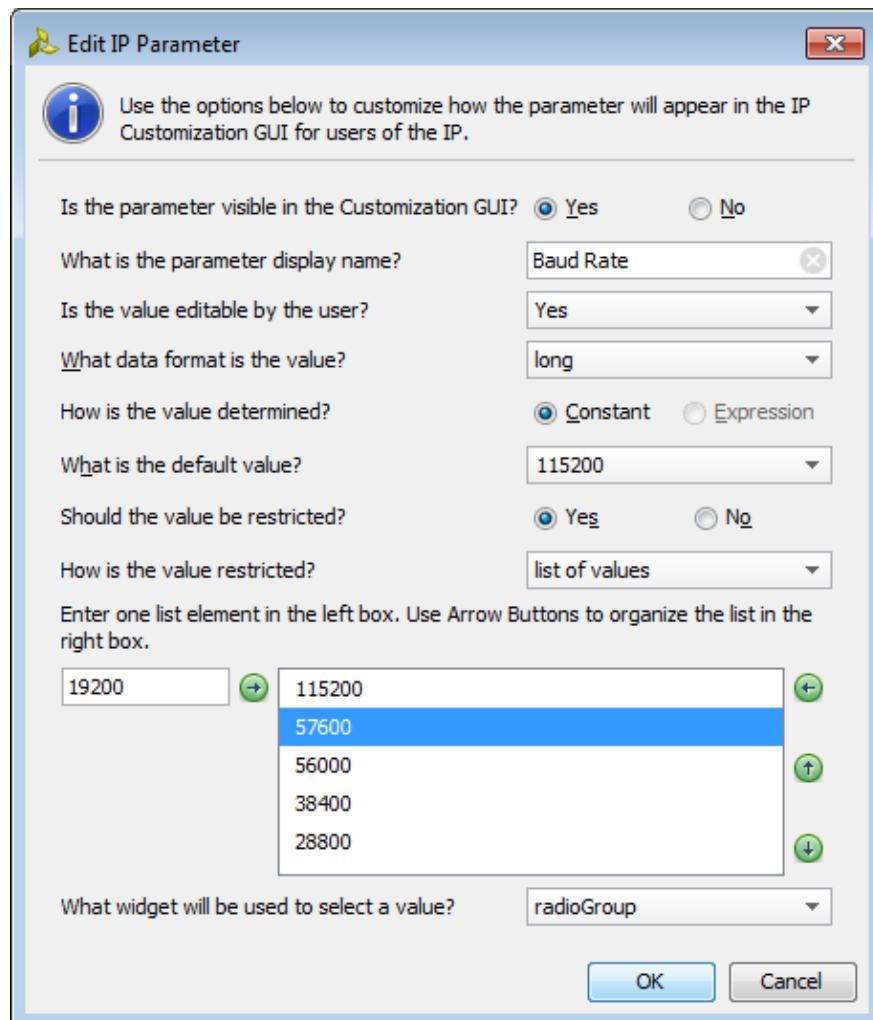


Figure 3-19: Edit IP Parameter Dialog Box with Organized List Values

- B. **What widget will be used to select a value?** Controls the type of GUI widget that displays the values in the IP Customization GUI.

The two available selections are: a radioGroup or comboBox.

- C. **Range of Integers:** The range of integers restriction limits the values of the parameter to a specified range of numeric integer values. The input boxes, shown in the following figure, control the values.

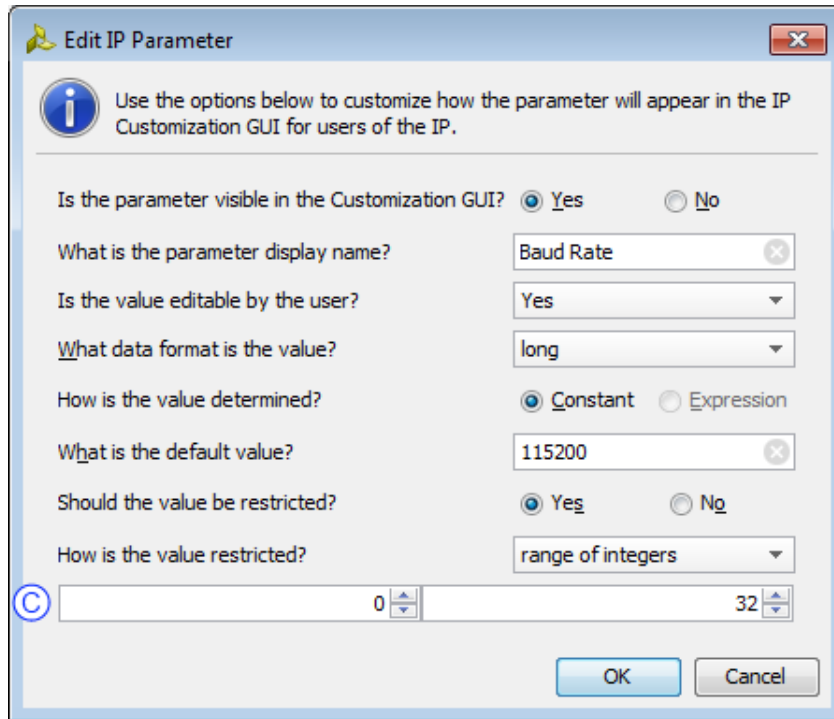


Figure 3-20: Edit IP Parameter Dialog Box with a Range of Integers Restriction

The supported range of integer values is from -2,147,483,648 to 2,147,483,647. A value set outside the specified range within the IP Customization GUI reports as an error.

- D. **Pairs:** The pairs restriction limits the value choices for the parameter from a predefined list similar to the list of values restriction; however, you can show the selections in the IP Customization GUI in a different format. A Key/Value pair list controls the values, as shown in the following figure.

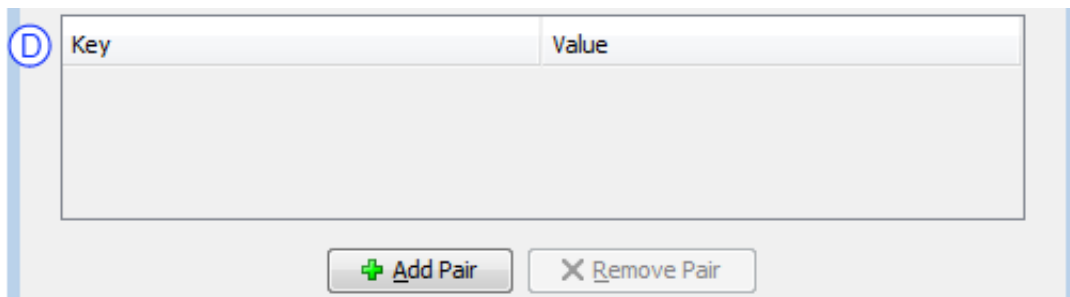


Figure 3-21: Edit IP Parameter Dialog Box with Pairs Restriction

To add a pair to the Key/Value list, click the **Add Pair** button in the dialog box. This creates a generic key/value pair in the list with Key set as **key** and Value set as **value**. To make the Key or Value field editable, double-click the text to change.

The text of the Key field is the value that displays in the IP Customization GUI. This is the value shown to the end-user of the IP. The text of the Value field is the value of the parameter passed to the RTL.

Note: The order in which the Key/Value pairs display in the list is the order the keys appear in the IP Customization GUI.

To remove a pair from the Key/Value list, select the pair to remove and click the **Remove Pair** button.

As an example, the following figure shows a dialog box with a list of Key/Value pairs.

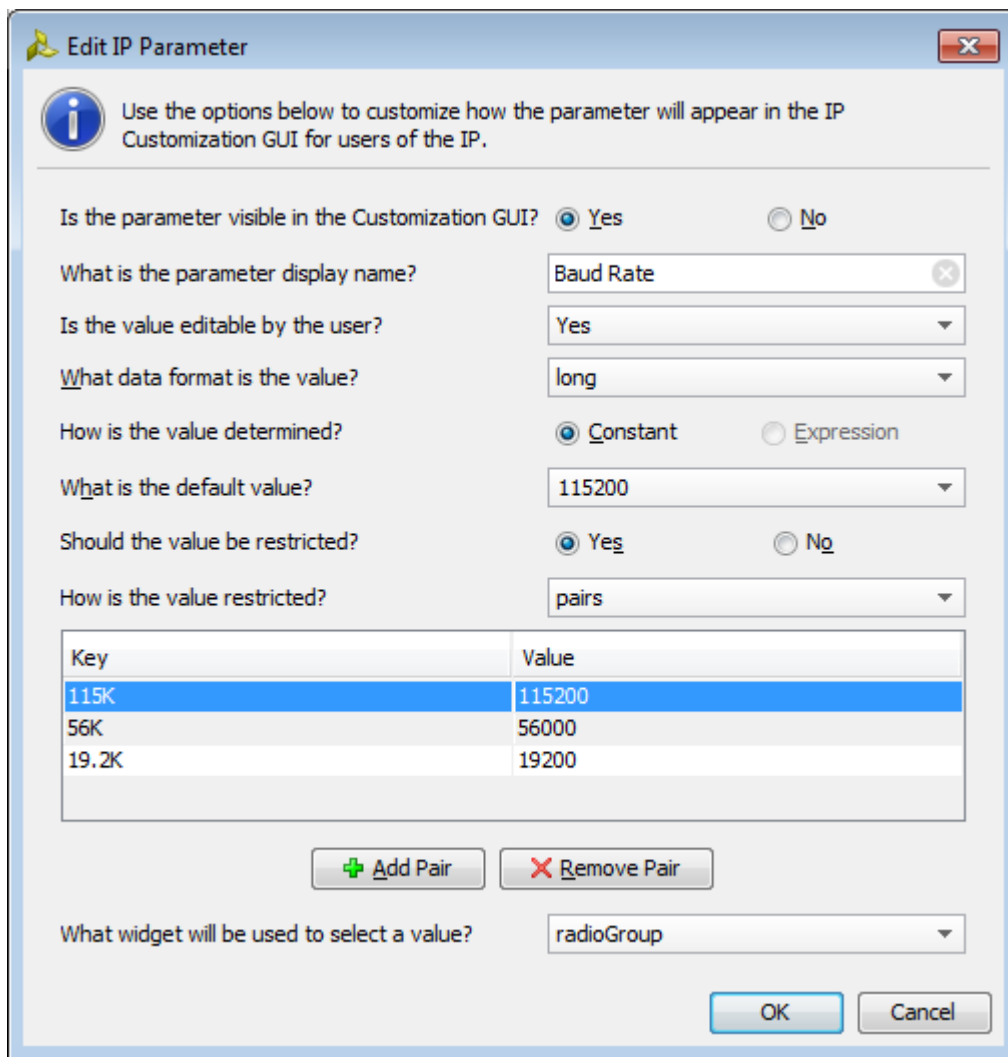


Figure 3-22: Edit IP Parameter Dialog Box with Key/Value Pair List

Similar to the list of values restriction, you can select the type of GUI widget that displays the values in the IP Customization GUI. The two available selections are: **radioGroup** or **comboBox**. For more information on the display of these widgets, see the [Customization Parameters, page 33](#).

Ports and Interfaces

The Ports and Interfaces step, shown in [Figure 3-23, page 45](#), provides a listing of ports and interfaces of the custom IP.

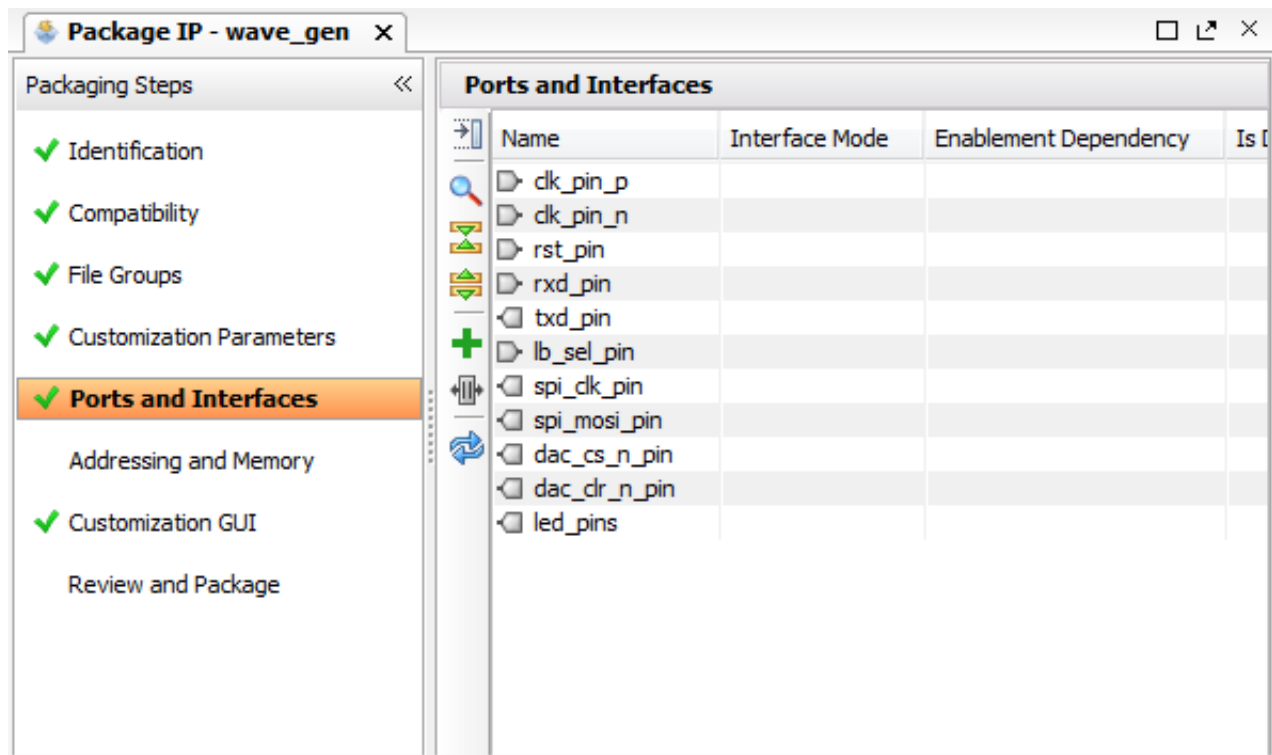


Figure 3-23: Package IP – Ports and Interfaces

The ports and interfaces list is initially populated at the end of the Create and Package IP wizard from parsing the top-level HDL source file. During the parsing of the top-level ports of the custom IP, if an interface can be heuristically determined, the interface will be inferred automatically by the IP Packager.

The ports and interfaces are listed in a table format that contain the following default columns of information of the port or interface:

- **Name:** The port or interface name.
- **Interface Mode:** The interface mode (master or slave)

- **Enablement Dependency:** The parameter equation to control whether the interface or port is enabled. See [Setting an Enablement Expression, page 63](#).
- **Direction:** The port direction.
- **Driver Value:** The default driver value for the port if that port is disabled.
- **Size Left:** The evaluated value of the left side of the vector bus.
- **Size Left Dependency:** The dependency parameter equation to determine the value of the left side of the vector bus.
- **Size Right:** The evaluated value of the right side of the vector bus.
- **Size Right Dependency:** The dependency parameter equation to determine the value of the right side of the vector bus.
- **Type Name:** The port type (`std_logic` or `std_logic_vector`).

To import ports from the top-level HDL source file due to a change to the HDL after initial packaging or just to initially populate the ports, open the IP Ports and Interface wizard, which is available from the IP Ports and Interfaces header

Importing Ports from a Top-Level HDL

The ports are initially determined during the Create and Package IP wizard.

To import the ports from the top-level HDL source file due to a change to the HDL after packaging, re-import the HDL ports.

1. From the Ports and Interfaces window, right-click and select **Import IP Ports**.
2. In the Import Ports from HDL dialog box, select the following options, and click **Finish**.
 - **Top-Level source file:** The top-level source HDL file that contains the top-level entity or module of the custom IP.
 - **Top entity name:** The name of the top-level entity or module that contains the ports of the custom IP.

Editing an Existing Port

1. Select the port, right-click and select **Edit Port**.

Note: You can also double-click on the port.
2. In the Edit IP Port dialog box, [Figure 3-24, page 47](#), set the following options and click **OK**.
 - **Driver Value:** If the port is disabled during customization, this is the default value that is assigned to the port.

- **Is this port optionally present?:** Determines whether the port can be disabled based on customization based on an enablement expression. For more details, see the [Setting an Enablement Expression, page 63](#).

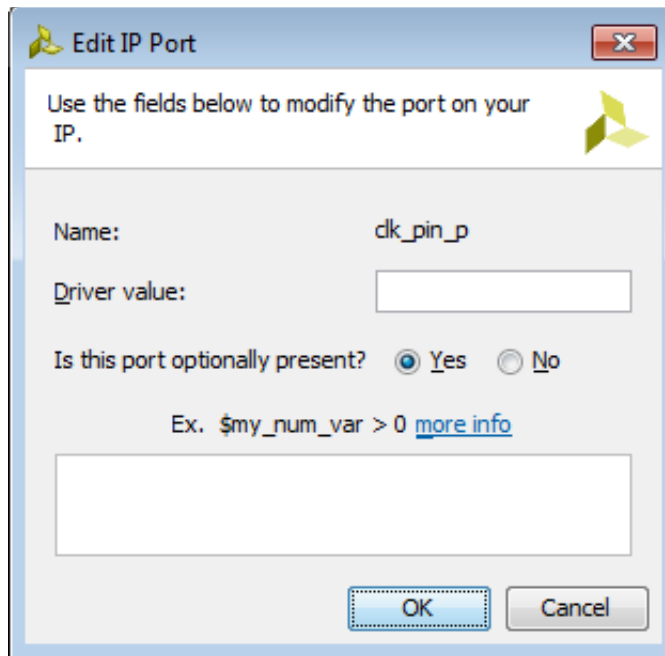


Figure 3-24: Edit IP Port

Adding or Removing Bus Interfaces

To add a bus interface, from the Ports and Interfaces window, right-click and select **Add Bus Interface**.

Note: Alternatively, you can click the **Add Bus Interface** button on the sidebar.

The Edit IP Bus Interface dialog opens for you to modify the information related to the interface. For more information on the Edit IP Bus Interface dialog box, see the following section, [Editing an Existing Interface](#).

To remove an interface, select the interface from the list, and select **Remove Interface**.

After you remove the interface, the ports previously associated to the interface return to the Ports and Interfaces list as un-associated ports.

Editing an Existing Interface

Create an interface by adding a new interface, using auto inference, or start during the Create and Package IP Wizard. In each case, the interface is initially populated with data that you can adjust based on your needs.

To edit the interface:

1. In the Ports and Interfaces window, right-click, select the interface, then select **Edit Interface**.

Note: You can also double-click on the interface you want to edit.

2. In the Edit IP Bus Interface dialog box select the options you want, and click **OK**.

Editing the Interface Information

The General tab, shown in [Figure 3-25](#), lets you set the definition information for the interface. This tab controls the name, type, mode, and description information.

1. From the Edit IP Bus Interface dialog box, select the following options.
 - **Interface Definition:** The selected interface to be created.
 - **Name:** This is the display name of the interface in the customization GUI as well as the IP integrator block design cell.
 - **Mode:** Defines the mode of the interface: Master or Slave.
 - **Display Name:** The display name of the interface.
 - **Description:** The description of the interface.
 - **Is this interface optionally present?:** Determines whether the interface can be disabled based on customization based on an enablement expression. For more details, see [Setting an Enablement Expression, page 63](#).

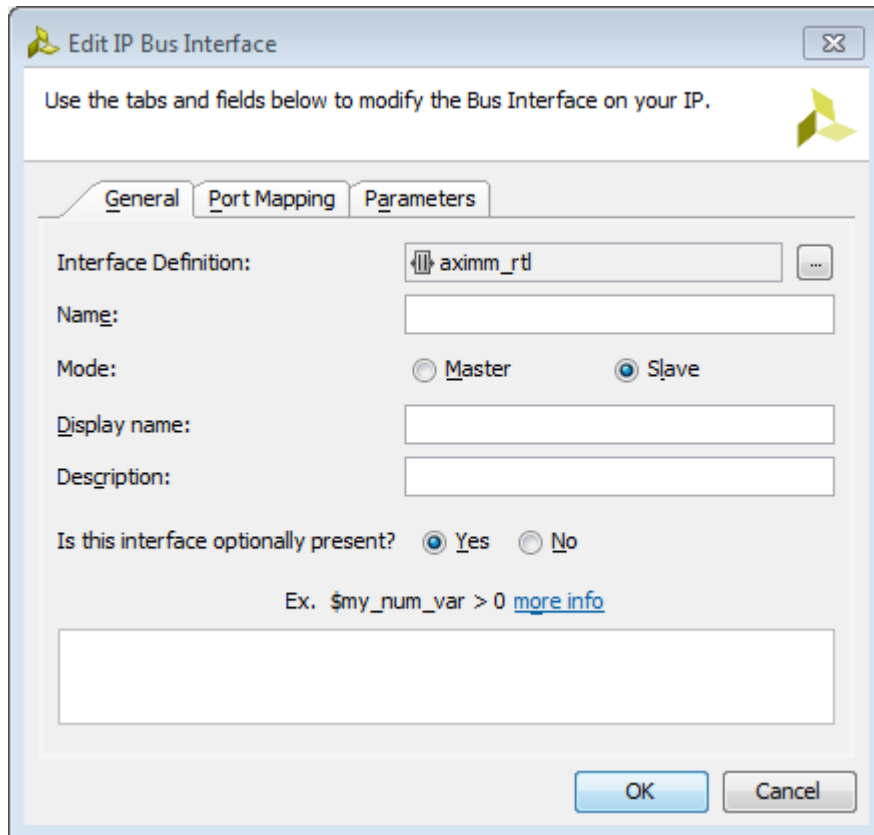



Figure 3-25: Edit IP Bus Interface – General Tab

2. To change the interface definition, select the **More Options**  button on the Interface Definition.
3. In the Interface Definition Chooser dialog, [Figure 3-26, page 50](#), select the required interface, and click **OK**.

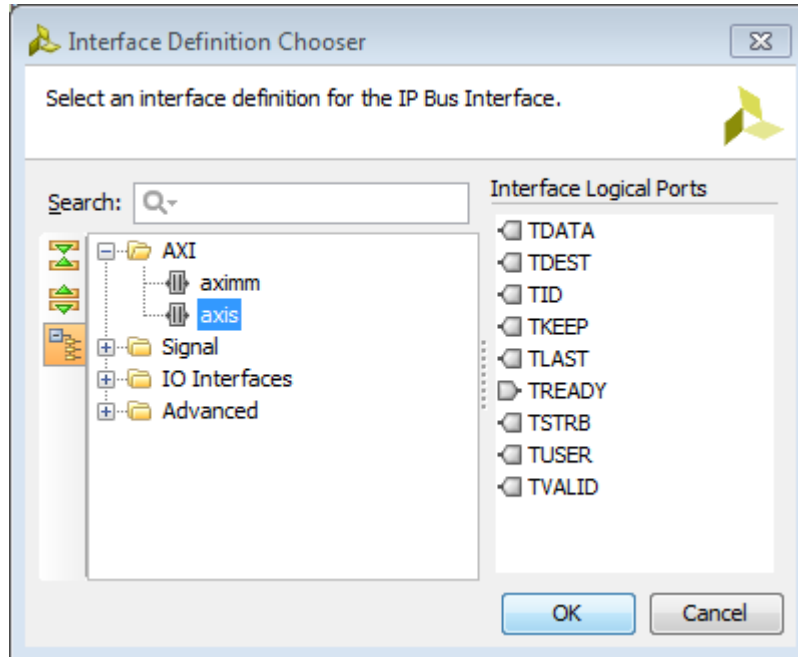


Figure 3-26: Edit IP Bus Interface – Interface Definition Chooser

Mapping the Interface Ports

Use the Port Mapping tab, shown in [Figure 3-27, page 51](#), to map the interface ports to the ports of the custom IP. The interface ports are listed in table format with the columns representing the logical ports of the interface and the physical ports of the IP. All ports mapped for the interface display in the summary are at the bottom of the window.

1. Select an interface port in the left column, and select the desired port from the physical port list in the right column.
2. After selecting a port from each column, click the **Map Ports** button.

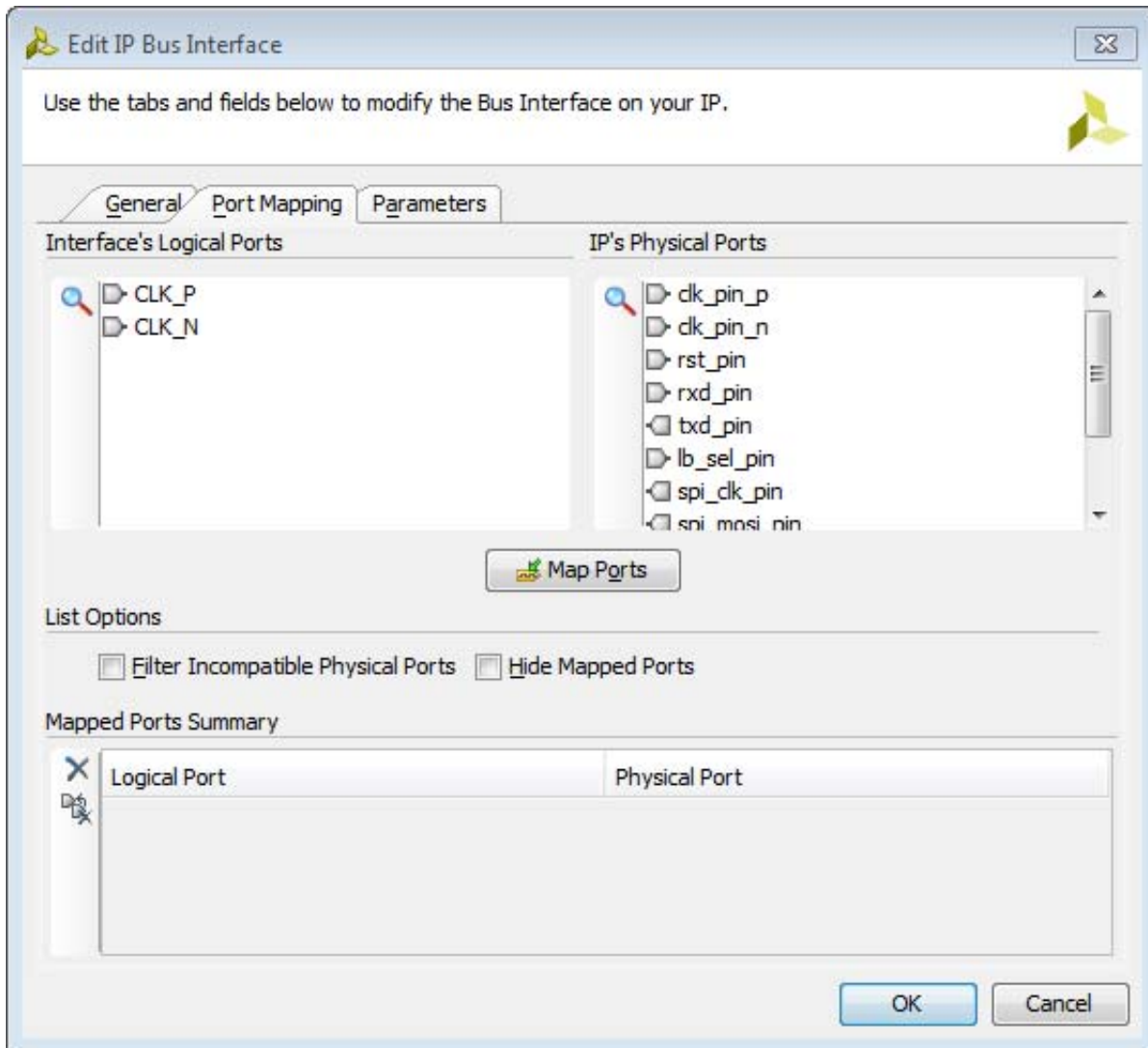


Figure 3-27: Edit IP Bus Interface – Port Mapping Tab

For large interfaces, the **List Options** gives you the ability to control the ports displayed in the columns. You can use the following checkboxes as follows:

- **Filter Incompatible Physical Ports:** Hide physical ports that do not match the correct direction of the selected interface port.
- **Hide Mapped Ports:** Hide physical and logical ports that have been previously mapped.

Each mapped port is listed in the **Mapped Port Summary** section of the window. you can un-map all the previously mapped ports, or select individual ports to be unmapped in the summary window.

Adding and Removing Interface Parameters

The Parameters tab, shown in the following figure, lets you add parameters to the interface. Some bus interfaces require associated parameters. Depending on the bus interface, the Vivado IDE identifies some parameters automatically, and recommends those parameters.

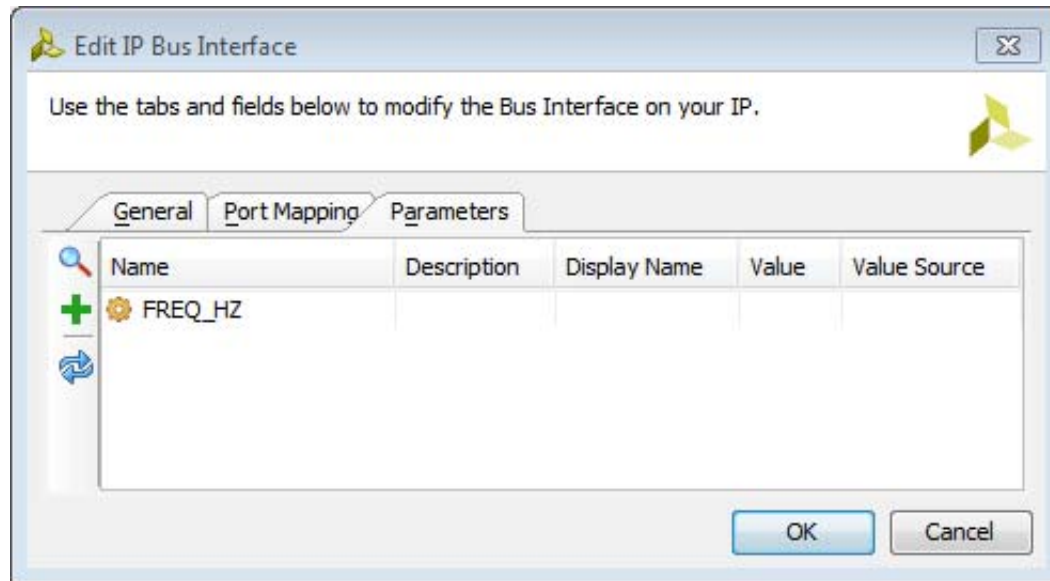


Figure 3-28: Edit IP Bus Interface – Parameters Tab

To add a recommended parameter to the list, do the following:

1. In the **Edit IP Bus Interface > Parameters** tab, right-click and select **Add Recommended Parameter**.
2. In the Add IP Parameter dialog box, select the required parameter, and click **OK**.

Optionally, you can add your own parameter to the interface by right-clicking selecting **Add Bus Parameter**.

3. After you add the parameters to the list, set the following options for each parameter:
 - **Description:** The description of the parameter.
 - **Display Name:** The display name of the parameter.
 - **Value:** The value of the parameter.
 - **Value Source:** The source value of the parameter.

To remove a parameter, select the parameter in the list, right-click and select **Remove Bus Parameter**.

Auto-Infering an Interface

There are two options for automatically inferring an interface: single-bit or bus.

- Infer the single-bit interface from a list of commonly used single-bit interfaces.
- Infer the bus interface by selecting from the full interface list.

To infer the bus interface:

1. Select all the ports related to the interface you want to infer, and select **Auto Infer Interface** from the popup menu.

Alternatively, you can click the **Auto Infer Interface** button on the sidebar.



RECOMMENDED: Name the ports exactly as specified in the interface, so the auto inference can correctly identify the bus interface ports.

2. Select the name of the interface you want to infer and click **OK**.

If IP packager is able to infer the interface, the Edit IP Bus Interface dialog box opens with the selected interface type, name, and port mapping. For more detailed information on the Edit IP Bus Interface dialog box, see [Editing an Existing Interface](#), page 48.

Note: If the auto-inference is unable to identify all the ports, you must manually identify the bus interface ports through the **Add Bus Interface** option. See [Adding or Removing Bus Interfaces](#), page 47.

3. To infer a single-bit interface, select the port, right-click and select **Auto Infer Single Bit Interface**.
4. From the extended popup menu, select the interface you want to infer.

Addressing and Memory

The Addressing and Memory section, ([Figure 3-29](#), page 54), lets you add a memory-map or address space to the IP.

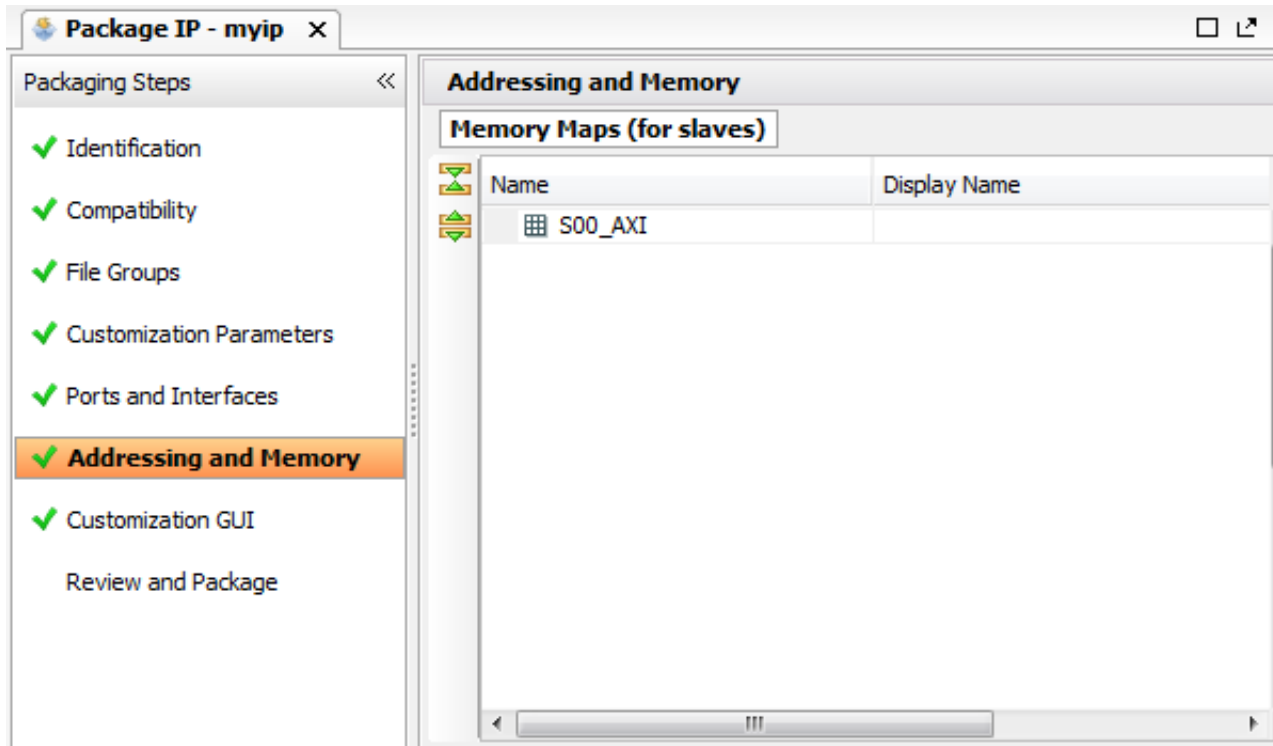


Figure 3-29: IP Addressing and Memory

If the Create and Package IP wizard was able to automatically infer an interface for your custom IP, the Addressing and Memory step is populated also if an address mapping requirement for the interface was identified.

Identifying an Interface for Address Mapping

1. Click the Addressing and Memory Map Wizard link if an interface has not been mapped previously.

Note: Alternatively, right-click and select IP Addressing and Memory Map Wizard from the popup menu if an interface has already been mapped.
2. In the Addressing and Memory Wizard, click **Next**.
3. In the Choose IP Interface page, shown in [Figure 3-30, page 55](#), select the interface to add a memory map and click **Next**.

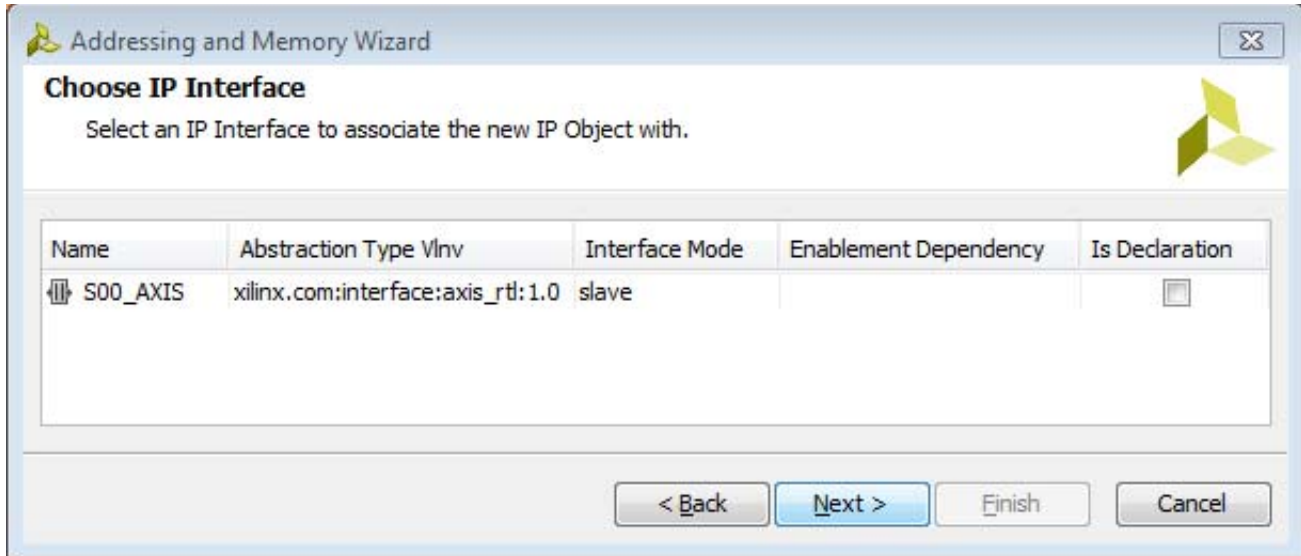


Figure 3-30: IP Addressing and Memory Configuration Wizard - Choose IP Interface

4. In the Choose IP Object Name page, select the name for the memory map, and click **Next**.
5. On the Summary page, click **Finish** to complete the wizard.

Adding an Address Block

1. Select a Memory Map from the list, then right-click and select **Add Address Block**.
2. In the Add Address Block dialog box, select the name of the new Address Block, and click **OK**.

The selected memory map in the Addressing and Memory window now contains a child address block section with the newly-created address block. The address block list contains the following columns:

- **Name:** Address block name.
- **Display Name:** Address block display name.
- **Description:** Detailed description of the address block.
- **Base Address:** Base address of the address block.
- **Range:** Range of the address block.
- **Range Dependency:** The dependency expression for the address block range.

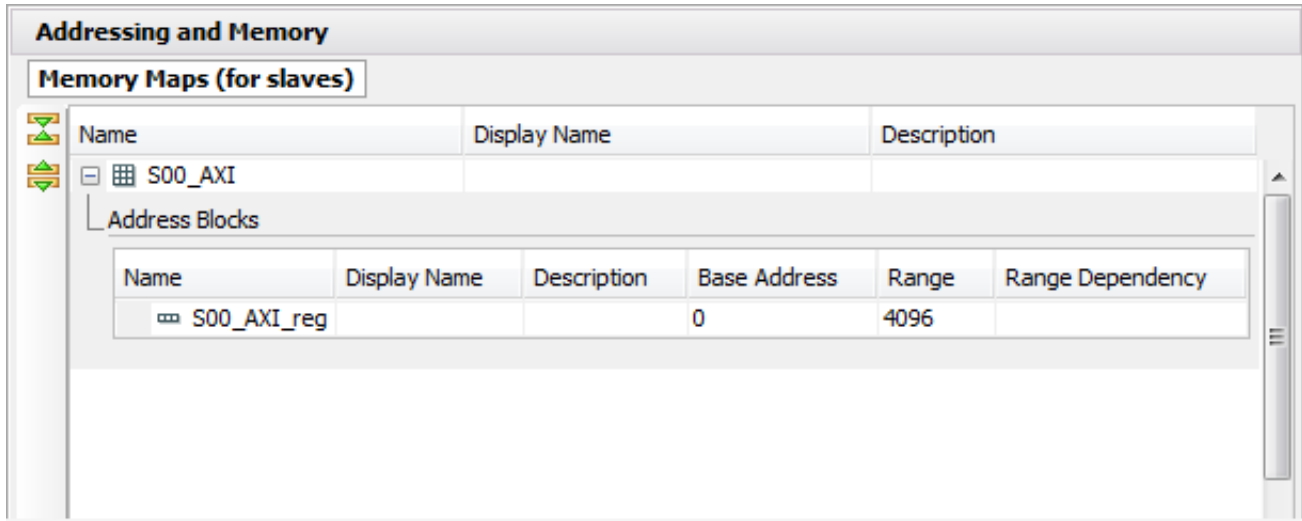


Figure 3-31: Memory Map with Address Block

3. Select the cell of the address block to enter the desired information.

Add or Remove an Address Block Parameter

To add an address block parameter:

1. Select the address block in the list, right-click select **Add Address Block Parameter**.
2. In the Add Address Block Parameter dialog box, select the name and click **OK**.
3. Select the cell of the address block parameter to enter the information.



RECOMMENDED: Give the interface in a meaningful name that reflects the functionality.

To remove an address block parameter from an address block, select the address block parameter in the list, and right-click and select **Remove Address Block Parameter**.

The IP Addressing and Memory Wizard Summary page opens (Figure 3-32, page 57). This describes the name of the new memory-map as well as to which interface the memory-map references.

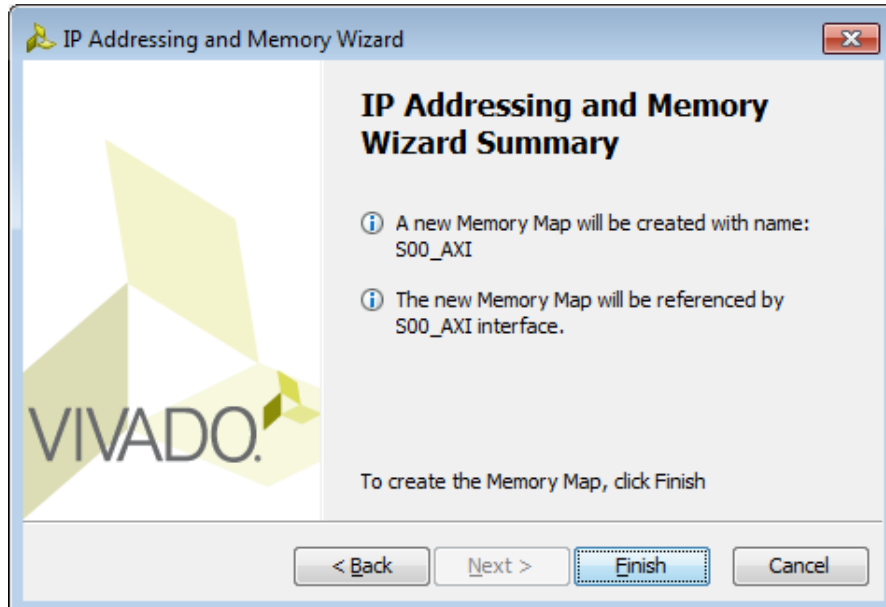


Figure 3-32: IP Addressing and Memory Wizard Summary

4. Review the summary, and click **Finish**.

Customization GUI

The Customization GUI step, [Figure 3-33, page 58](#), provides an environment for the GUI customization of your custom IP. The Customization GUI section lets you customize the layout by adding display pages, parameter groupings, and text fields.

After you setup the parameters of your IP in the Customization Parameters section, you can customize the GUI to change how a user interacts with your custom IP.

Initially, the Customization GUI generates a layout with all the viewable parameters displayed on a single page of the GUI.

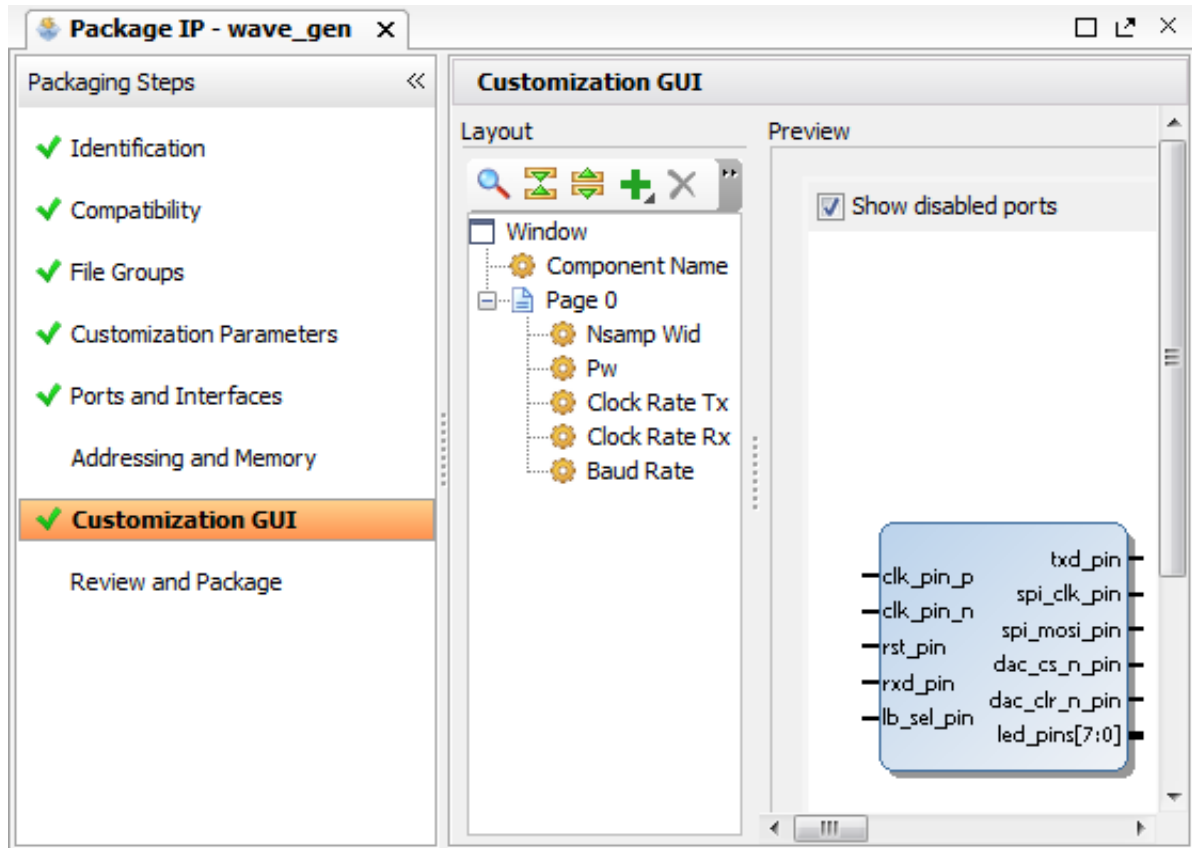


Figure 3-33: Package IP – Customization GUI

The two columns that display in the Customization GUI window are, as follows:

- **Layout:** A hierarchical display of the layout which allows for modification of the custom IP GUI.
- **Preview:** A preview display of the Customization GUI of the custom IP.

The **Layout** column displays a hierarchical view of the customization GUI components. Within a **Window** component, which is the top-level to which to associate the customization components, there are a total of four components that you can create to customize the display of your custom IP GUI:

- **Page:** An individual page to display the parameters of the custom IP.
- **Group:** A collection of parameters to display in a single group.
- **Parameter:** A parameter of the custom IP.
- **Text:** A text field to display any necessary information in the GUI.

Each component can be associated hierarchically to other components within the customization layout. The **Group**, **Parameter**, and **Text** components can be within a **Page** or **Group**. The **Page** component can only be a child of the **Window** component.

The preview window shows you a real-time feedback view of the customization GUI as it would appear if the IP was customized through the IP Catalog.


The components in the preview display in the same order in which they are arranged in the layout. You can change the order in which the components display by dragging the components in the layout column to the location you want.

The Customization GUI section retains its information, thereby allowing for a simple iterative process for updating the custom IP because it only affects the parameters that were added or removed.

Adding Parameters to the Layout

To add parameters to the layout, do the following:

1. Select the hierarchical level (page or group) you want to add the parameter, right-click and select **Add Parameter**.


Note: Alternatively, you can select the **Add** button  from the toolbar.

2. In the Add Parameter dialog box, select the following options, and click **OK**.
 - **Available Parameters:** The available parameters of the custom IP. These are parameters that have not been previously added to the Customization GUI.
 - **Display Name:** The label text displayed for the parameter in the Customization GUI.
 - **Tooltip:** The text displayed for the tooltip when hovering over the parameter in the Customization GUI.
 - **Show Label:** The option to disable the label (Display Name) of the parameter.

Adding Groups to the Layout

To add groups to the layout, do the following:

1. Select the hierarchical level (page or group) you want to add the group, right-click and select **Add Group** from the popup menu.


Note: Alternatively, you can select the **Add** button  from the toolbar.

2. In the Add Group dialog box, select the following options, and click **OK**.
 - **Display Name:** The label text displayed as a header for the group in the Customization GUI.
 - **Tooltip:** The text displayed for the tooltip when hovering over the group in the customization GUI.
 - **Layout:** The option to display the components of the group vertically or horizontally.

Adding Pages to the Layout

To add pages to the layout, do the following:

1. Select the window component, right-click, and select **Add Page**.

Note: Alternatively, you can select the **Add** button  from the toolbar.

2. In the Add Page dialog box, select the following options, and click **OK**.
 - **Display Name:** The label text displayed on the tab for the page in the Customization GUI.
 - **Tooltip:** The text displayed for the tooltip when hovering over the page contents in the Customization GUI.

Adding Text to the Layout

To add text to the layout, do the following:

1. Select the hierarchical level (page or group) to which you want to add the text, then right-click and select **Add Text**.

Note: Alternatively, you can select **Add** button  from the toolbar.

2. In the Add Text dialog box, select the following options, and click **OK**.
 - **Display Name:** The label text displayed in the layout and the tooltip for the text in the customization GUI.
 - **Text:** The text displayed in the Customization GUI.

Review and Package

The Review and Package section, shown in the following figure, provides a summary of the IP and information about the settings you selected after packaging.

The IP is initially packaged at the end of the Create and Package IP wizard. If any changes occur to any of the packaging steps of the custom IP, the custom IP must be repackaged for the changes to go into effect.

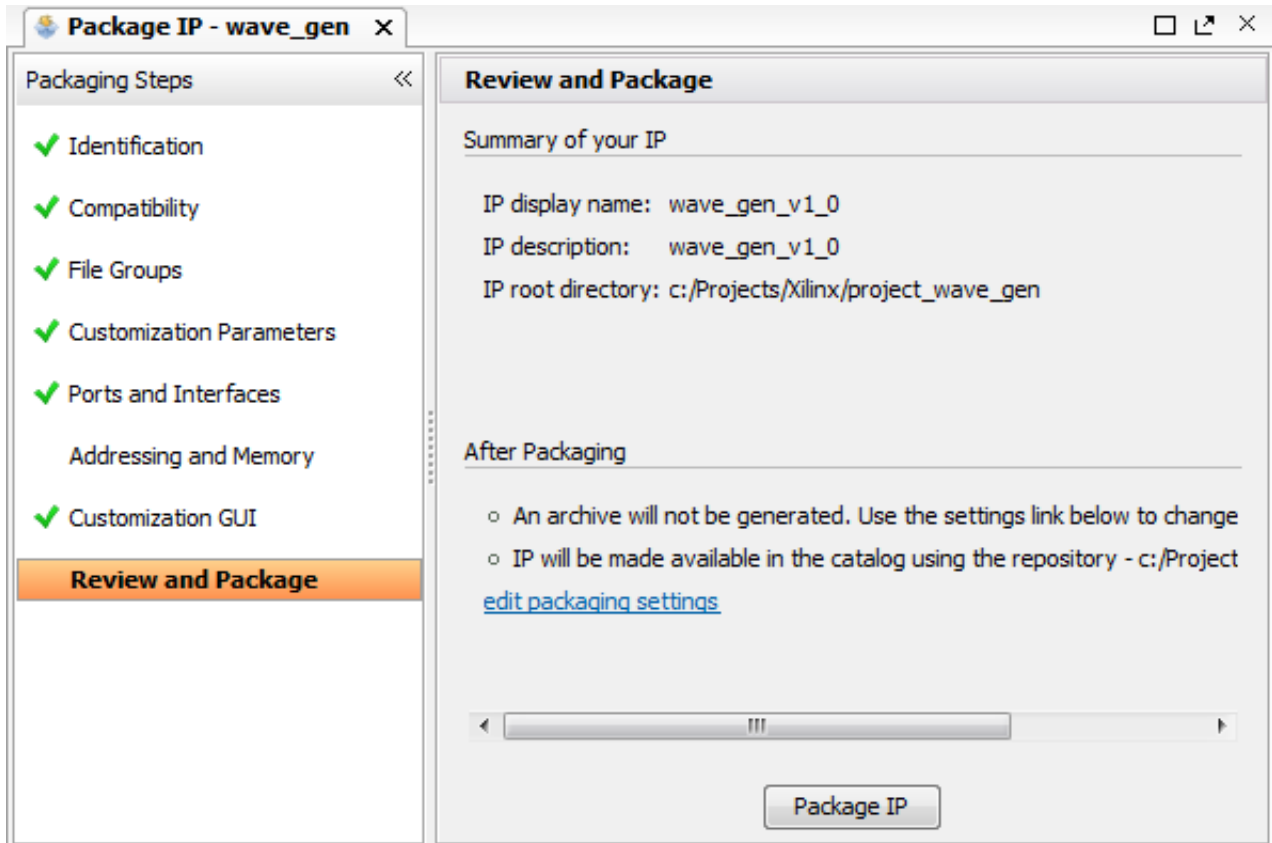


Figure 3-34: Package IP – Review and Package

The summary information is the identification data of the custom IP from the Identification section. To change the information, make the modifications in the Identification section, as described in [Identification, page 22](#).

The After Packaging section contains information about the Vivado IDE actions after packaging is complete.

The information that describes how the packager behaves is based on the IP Packager Settings, described in the [Using IP Project Settings, page 8](#), are, as follows:

- **Archive:** Select whether to create an archive of the IP.
- **Repository:** The location of the IP for the Vivado repository manager.

In the IP Project Settings, you have an option of creating an archive of the custom IP definition.

By default, the custom IP is generated at the IP root directory in which the source files are referenced relatively from the IP root directory location, if possible.

When an archive is created, the Vivado IDE compresses all the relatively referenced files into a single archive file. This archive file can be used in a Vivado repository by using the single archive file or extracting the archive file in a desired repository location.

After the IP is packaged, the custom IP is available in the IP Catalog of the current Vivado project. The Vivado IDE automatically adds the IP root directory of the custom IP to the Repository Manager in the IP Project Settings.

The IP root directory is the location of custom IP definition and the location required to add to the Repository Manager of other Vivado projects that require the custom IP.

Repackaging the IP

To repackage IP, do the following:

1. Click the Re-Package IP button at the bottom of the Review and Package window.
2. If successful, click **OK** on the Package IP popup dialog.



IMPORTANT: *Repackaging an IP and changing the default parameter values does not change the customization parameters in the IP instance. Each IP instance must be manually changed to the new default value if that change is required.*

Creating an Archive of the IP

To create an archive of the IP:

1. Select the **Create archive of IP** option in the IP Packager Project settings, as described in [Using IP Project Settings, page 8](#).

Note: You can open the IP Packager Project Settings by clicking the **edit packaging setting** hyperlink.

2. In the **After Packaging** section, select the **edit** hyperlink to change the name and location of the archive.

Note: By default, the archive name is <Vendor>_<Library>_<Name>_<Version#>.zip.

3. In the Package IP dialog box, select the following options, and click **OK**.
 - **Archive name:** The name of the archive file.
 - **Archive location:** The location where the archive file is created.

Setting an Enablement Expression

For ports and interfaces, you can use an enablement expression to enable or disable top-level ports of the custom IP based on the customization. The syntax that evaluates the expression can reference parameters defined through the IP packager Customization Parameters.

The variable name of the parameter is the parameter name, which is case-sensitive and requires a (\$) sigil to be prefixed before the variable name.

For example, a parameter named `BAUD_RATE` has an expression variable reference of `$BAUD_RATE`.

To create a functioning expression, use the parameter name with numeric and comparison operators to form a Boolean expression.

Note: See <http://wiki.tcl.tk/583> to ensure conformity with expression rules.

Creating and Packaging Custom IP in IP Integrator

Introduction

This chapter provides a brief summary of the Vivado® IP integrator features for creating and packaging IP.

VIDEO: See the Quick Take Video: [Packaging Custom IP for use with IP Integrator](#) for a demonstration of how to use this feature.

Packaging a Block Design from IP Integrator

When you create an IP integrator design, implement it, and test it on target hardware and are satisfied with the functionality, you might want to *Package* that design, and convert it to an IP that can be reused in another design.

When you package a design, it gets converted into an IP and is available for you in the IP Catalog. You can instantiate that IP as part of a different design.

To package a block design:

1. In the Vivado IDE Sources window, right-click the block design and select **Package Block Design** ([Figure 4-1, page 65](#)).

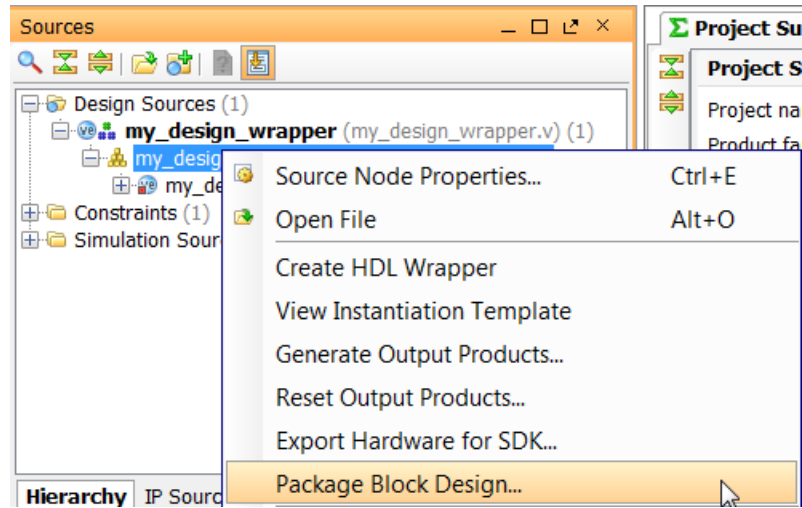


Figure 4-1: Package Block Design

The Package IP window opens, as shown in the following figure.

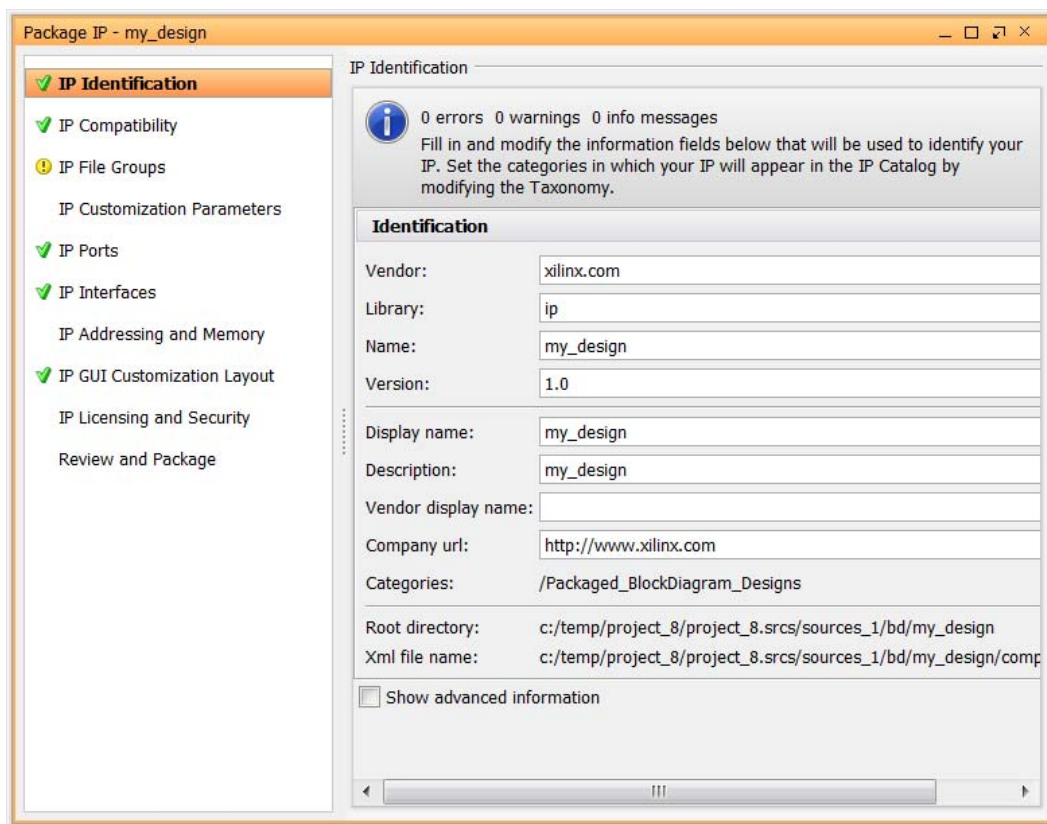


Figure 4-2: Package IP Window

2. Select the Review and Package menu item in the dialog box, then click **Package IP**.

See [Chapter 3, Packaging IP](#) for more information about the packaging options.

After the IP is packaged, the IP is available in the IP integrator catalog, as shown in the following figure.

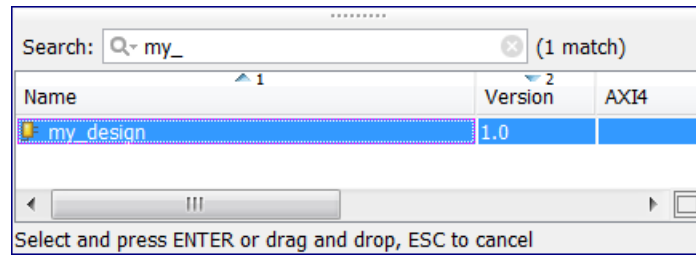


Figure 4-3: Packaged Design in the IP Integrator Catalog

The newly packaged design is also available in the Vivado IP catalog under the category **Packaged Block Diagram Designs**, as shown in the following figure. This category can be changed and renamed while packaging the block design.

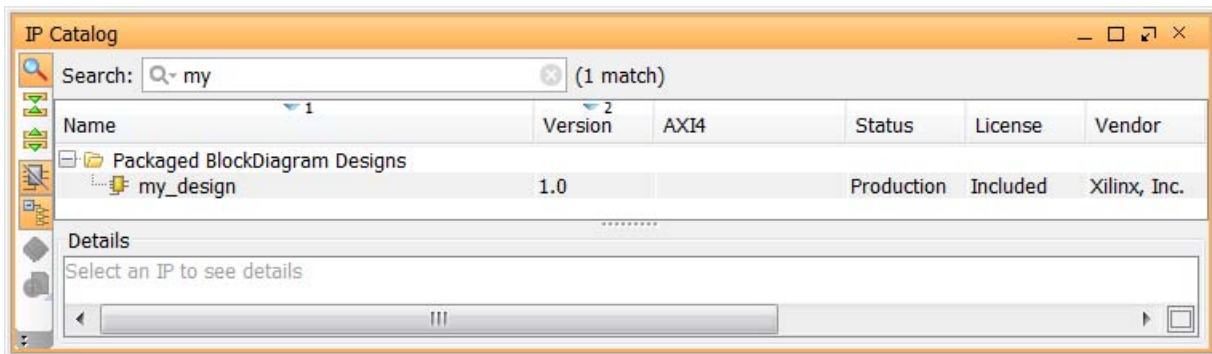


Figure 4-4: Packaged Block Diagram Designs Category

Packaging Block Design Limitations

The limitations to packaging a block design are:

- If a packaged block design contains a MicroBlaze™ processor or a Zynq®-7000 processor, and then the packaged design is instantiated in a different project, the **Export Hardware to SDK** option poses some issues. Essentially, SDK does not recognize the embedded objects in the packaged block design.
- When a block design is packaged, a package (XML) file is generated that contains references to lower-level IP XCI files. When this packaged diagram is ported to a future release of the software, this could pose problems with upgrades as the IP can get locked if there are newer versions of the IP in the current release.

Revision Control for IP Integrator Designs

The Vivado Design Suite is designed to work with any revision control system.

To ensure that designs are suitable for any revision control system, Vivado supports the following:

- Timestamp update only with file modification. Opening a project does not change the file timestamp.
- ASCII-based project files
- Tcl scripting capabilities

Projects can include multiple design sources and configuration files; however, only a subset of files require revision control to re-create a project and reproduce implementation results.

For more information, see this [link](#) in the *Vivado Design Suite User Guide: Designing IP Subsystems Using IP Integrator* (UG994) [Ref 15] for more information.

Creating New Interface Definitions

Introduction

Interface definitions provide the capability to group functional signals into a common interface grouping to use between IP in a Vivado IP integrator diagram. The interface definition give you more a comprehensible diagram, and also enforces a standardized expectation that signals are designed to work between IP pairs.

Xilinx provides many interface definitions, including standardized AXI protocols and other industry standard signaling; however, some legacy or custom implementations have unique IP signaling protocols.

You can define your own interface and capture the expected set of signals, and ensure that those signals exist between IP.

The Create Interface Definition option uses the IP-XACT industry standard specification. The interface definitions use two files that together correspond to a bus definition file (`myInterface.xml`) and an abstraction definition file (`myInterface_rtl.xml`).

The Vivado IDE uses the created interface definition to support mapping logical ports and inferring bus interfaces on IP Definition in the IP packager.

Creating a New Interface Definition

1. Select **Tools > Create Interface Definition**.

The Create Interface Definition dialog opens, as shown in [Figure 5-1, page 69](#).

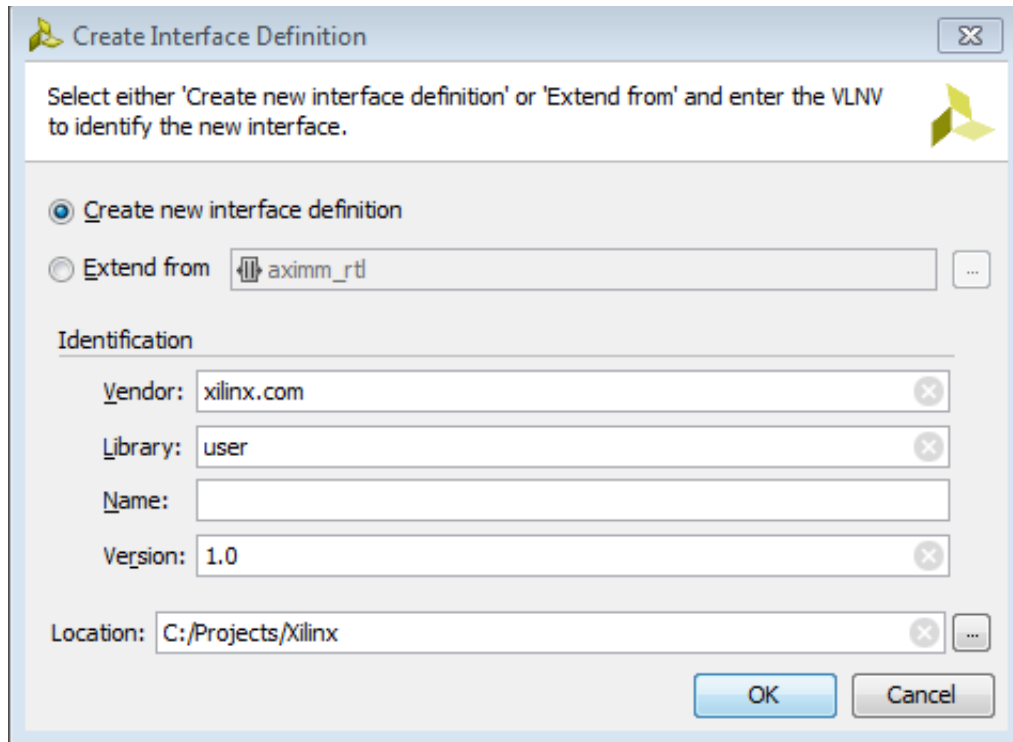


Figure 5-1: Create Interface Definition

2. Select one of the interface types from the following options:
 - **Create new interface definition:** Creates an empty interface, which is the typical use case.
 - **Extend from:** Creates a new interface with the selected interface ports list as a template. This is an advanced feature.

By creating a new interface definition, the Interface tab opens without any pre-determined ports defined. You have a blank state for creating the new interface.

If you extend from a previously selected interface, the Interface tab opens with the ports pre-populated in the ports list. You can add or delete the ports as necessary.

3. In the Identification fields, enter the following information for the interface:
 - **Vendor:** The vendor of the interface. This is also the identifier for the vendor that displays in the VLNV of the interface definition. Use the standard internet domain order to provide the vendor information.
 - **Library:** The library in which the interface belongs. This is also the identifier for the library that displays in the VLNV of the interface definition. By convention, Xilinx interface definitions use **interface** for the library field.
 - **Name:** The name of the interface. This is also the identifier for the name that displays in the VLNV of the interface definition.

- **Version:** The version of the interface. This is also the identifier for the version that displays in the VLNV of the interface definition.
 - **Location:** The directory where the pair of interface XML files are created.
4. Click **OK** to complete the interface creation.

Using the Interface Definition Editor

After you finish creating the new interface definition, the workspace editor for the interface opens in Interface Definition Editor for modifications, as shown in the following figure.

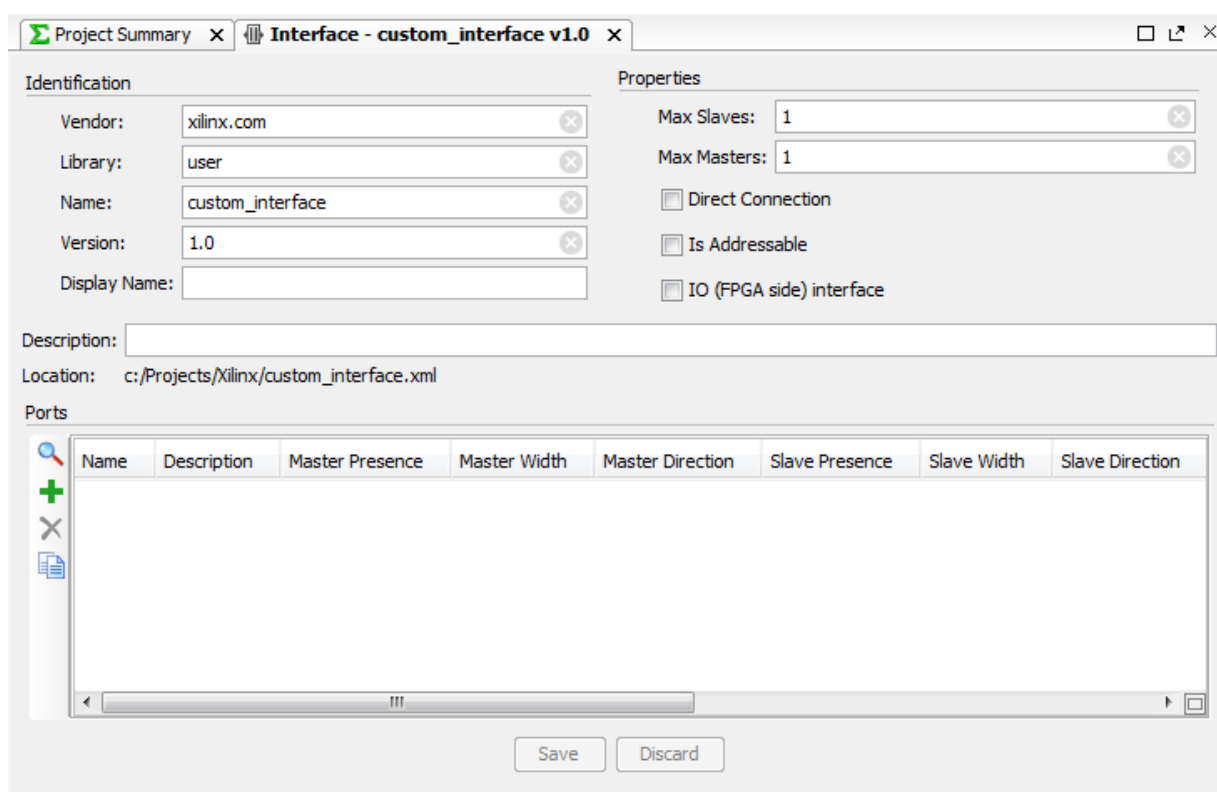


Figure 5-2: Interface Definition Editor

- **Identification**

This content field is pre-populated if you are re-defining an existing interface, as described in [Creating a New Interface Definition, page 68](#).

- **Vendor:** Vendor name.
- **Display Name:** A human-readable string for a short description, which is shown in Bus Interface related windows in the Vivado IP packager.


- **Description:** A longer description of the interface. This field also shows in Bus Interface related windows in the IP packager.
- **Properties**

Enter the following information:

 - **Max Slaves:** This is an advanced property.
 - A **0** setting restricts the bus interface from being exposed in a slave mode IP Definition.
 - A **-1** instructs the Vivado tools to not check DRCs on the number of slaves.
 - **Max Masters:** This is an advanced property.
 - Setting this field to **0** restricts the bus interface from being exposed on an IP Definition with the mode of master.
 - A value of **-1** instructs the IP integrator to not check DRCs on the number of masters.
 - **Direct Connection:** This advanced property, when checked, allows the IP integrator to create the connection of wires from master to slave in the netlist.
 - **Is Addressable:** This advanced property, when checked, indicates that the interface can participate in address space mapping.
 - **IP (FPGA side) interface:** This advanced property, when checked, allows no connects between IP inside an IP Integrator diagram. It automatically suppresses showing slave interface in the logical port table.

Adding a Logical Port to the Interface

1. On the ports list, right-click and select **Add Port**.

Alternatively, you can click the  button on the sidebar.

2. Type the name of the port in the pop-up field, and press **Enter**. In the Xilinx convention, these names are uppercase and are not prefixed with the name of the interface.

The physical ports on IP that map to this interface often contain a prefix, but it is unnecessary for that prefix in the interface definition.

Adding Logical Ports from a Previously Defined Interface

As when creating the definition of the interface, you can additionally extend interfaces to the custom interface.

1. Right-click the ports list, and select **Copy Ports From**.

Alternatively, click the **Copy Ports From**  button on the sidebar.

2. In the Copy Interface Ports dialog box, select the interface to which to add ports.
3. Click **OK**.

All the ports from the selected interface are populated in the Ports list.

Note: Any previously defined port with the same name as one of the imported port is overwritten.

Editing Logical Ports

After you create a logical port, a new row appears in the Create Interface Definition table. Edit following cells:

- **Description:** Enter a human-readable description for the logical port. Any transaction signaling must be described.
- **Master Presence:** Set to either **required** or **optional**, indicating if the logical port is required to be mapped in master mode bus interfaces exposed on an IP. Allowed values are: **optional**, **required**, and **illegal**.
- **Master Width:** Required bit width of the mapped port for master mode bus interfaces exposed on an IP. Values are: **-1** (undefined width), **1** (single bit signal), or a fixed value.
- **Master Direction:** Required port mode when mapped for master mode bus interfaces exposed on an IP. Values are **in**, **out**, and **inout**.
- **Slave Presence:** Set to either **required** or **optional**, indicating if the logical port is required to be mapped in slave mode bus interfaces exposed on an IP. Values are **optional**, **required**, and **illegal**.

Note: Often the slave and master presence are the same.

- **Slave Width:** Indicates the required bit width of this mapped port for slave mode bus interfaces exposed on an IP. Values are: **-1** (undefined width), **1** (single bit signal), or a fixed value.
- **Slave Direction:** Required port mode when mapped for master mode bus interfaces exposed on an IP. Values are: **in**, **out**, and **inout**.

Note: Typically the slave and master directions are opposite. If the directions are the same, IP Integrator might not be able to properly directly connect the master and slave.

Editing Documentation Cells

Edit the following cells to provide additional documentation of the role that the logical ports play in the connection.

- **Is Address:** Check this option when the logical port serves as an address port.
- **Is Data:** Check this option when the port carries data and is not a control, addressing nor clock/reset port.
- **Is Clock:** Check this option when the port is a clock line.
- **Is Reset:** Check this option when the port is a reset line.
- **Default Value:** When the port is optional, the IP integrator uses this default value to drive the `in` mode port when unconnected.

Note: If this port is part of the control logic, and is optional, it is important that the value is set to allow transactions to continue and not stall.

Setting Tristate Signaling

The following cells provide advanced information for tristate signaling and can be specified to allow a cross-over like connection between masters and slaves with a triple of tristate signals.

Because a device does not have true tristate lines inside the fabric, a triple of signals (**in**, **out**, **tristate**) represents a tristate pair (**in**, **out**, **tristate**).

When you create an interface with the triple of signals, add the following information, which is required to allow connectivity between masters and slaves.

- **Tristate Role:** This indicates if the port is part of a tristate triple of signals, and which role. Non-tristate ports leave this cell blank. Values are **blank**, **in**, **out**, and **tristate**.
- **Group:** When there are multiple tristate triples in the interface, it is important that the three signals be grouped together. This field should have the same string identifier entered for the three ports that form the same unique tristate.

Completing the Interface Definition Creation

When you have completed providing the required information for the Create Interface Definition option, click the **Save** button centered at the bottom of the window to save the pair of XML files.

To throw away edits, click the **Discard** button.

Re-Editing Interface Definitions

To edit an interface definition, in **File > Open IP-XACT File**, browse to either of the bus definition file (`myInterface.xml`) or abstraction definition file (`myInterface_rtl.xml`). Both files must be in the same directory for the Vivado GUI to edit the definition correctly.

Using a New Interface Definition

Similar to IP Definitions, the IP-XACT bus definition files must be in a repository path for the IP packager and IP integrator tools to use the files.



RECOMMENDED: Structure your IP repository to have a single directory with an `/interfaces` subdirectory to contain the interface definition files, and a peer IP subdirectory to contain your IP definition files. A repository path **must be added at, or preceding, the directory** containing the IP definition file.

To see the interface definition files in the IP Catalog:

1. In the IP Catalog, ensure that the catalog is organized to display **by repository**.
2. Select the user repository with the new interface definition files. A tab displays the names and number of interfaces.

Note: After the interface definitions are recognized the projects catalog, the IP packager lets the IP definition create bus interfaces based on the bus definition, including the ability to infer the bus interface port mapping.



IMPORTANT: The IP packager uses a name-matching heuristic which is based on the logical port names matching the physical ports in the IP. IP packager automatically detects AXI interfaces when the names of the HDL modules ports follow the `<interface_name>_<AXI signal name>` convention. For example, `s_axi_awvalid`, where `s_axi` is the interface name and `awvalid` is one of the AXI signal names used in the interface. Module ports with the same `<interface_name>` are mapped into the named interface.



RECOMMENDED: Use short interface names because the IP block diagram has limited space.

The IP packager bus interface algorithms are almost completely data-driven, based on the bus definition and the algorithm works similarly with Xilinx provided definitions as well as user created interfaces.

AXI4 memory-mapped and AXI4-Stream interfaces have additional DRC checks beyond the data-driven heuristics.

Standard and Advanced File Groups

Introduction

This appendix lists the Standard and Advanced file group categories and descriptions.



IMPORTANT: *NGC format files are not supported in the Vivado Design Suite for UltraScale devices. It is recommended that you regenerate the IP using the Vivado Design Suite IP customization tools with native output products. Alternatively, you can use the NGC2EDIF command to migrate the NGC file to EDIF format for importing. However, Xilinx recommends using native Vivado IP rather than XST-generated NGC format files going forward*

Standard File Groups

Table A-1 lists the **Standard** file group types and a description.

Table A-1: Standard File Group Types and Descriptions

Standard File Groups	
Examples	Files that make up an example. Typically contains a constraint (XDC), HDL, and/or XIT files. Vivado uses these files to seed a new Vivado example project and shows this to the end-user for their exploration. The files are available for both synthesis and simulation.
Product Guide	The production guide for an IP.
Readme	Any required <code>readme.txt</code> file.
Simulation	Simulation files to deliver. Use when you have a mix of VHDL and Verilog to simulate together. Typically, exclusive of "VHDL Simulation" and "Verilog Simulation." The files could be the same as the files in the corresponding synthesis file group (when the synthesis files can also be used for simulation) or could be different (when a behavioral simulation model files are to be used).
Synthesis	Synthesis files to deliver. Use when you have a mix of VHDL and Verilog to synthesize to together. Typically exclusive of "VHDL Simulation" and "Verilog Simulation." Adding a constraint (XDC) file here causes the constraint file to be applied to the top-level of the IP during implementation.

Table A-1: Standard File Group Types and Descriptions (Cont'd)

Standard File Groups	
Verilog Simulation	Simulation files to deliver. Use when you have a Verilog only representation to simulate. You might see both this file group and "VHDL Simulation" to allow the ability to have a language-specific IP simulation. The files could be the same as the files in the corresponding synthesis file group (when the synthesis files can also be used for simulation) or might be completely different (when a behavioral simulation model files are to be used).
Verilog Synthesis	Synthesis files to deliver. Use when you have a Verilog only representation to synthesize. You might see both this file group and "VHDL Synthesis" to allow the ability to have a language specific implementation of the IP. Adding a constraint (XDC) file here applies the constraint file to the top-level of the IP during implementation.
VHDL Simulation	Simulation files to deliver. Use when you have a VHDL only representation to simulate. You might see both this file group and "Verilog Simulation" to allow the ability to have a language specific IP simulation. The files could be the same as the files in the corresponding synthesis file group (when the synthesis files can also be used for simulation) or might be completely different (when a behavioral simulation model files are to be used).
VHDL Synthesis	Synthesis files to deliver. Use when you have a VHDL only representation to synthesize. You might see both this file group and "Verilog Synthesis" to allow the ability to have a language specific implementation of the IP. Adding a constraint (XDC) file here causes the constraint file to be applied to the top-level of the IP during implementation.

Advanced File Groups

Table A-2 contains a listing and a description of each **Advanced** file group type.

Table A-2: Advanced File Group Types and Descriptions

Advanced File Groups	
Catalog Disabled Icon	GUI icon to show in the IP Catalog when the IP is disabled. (For example, when the IP does not support the selected device family).
Catalog Icon	GUI icon to show in the IP Catalog.
C Simulation	Use to deliver files for c-model simulation. These files are copied out without any further processing (excepting Tcl and XIT which is always evaluated). If you are delivering pre-compiled libraries, it is suggested to deliver these in machine specific directories following the Vivado convention (for example; lnx32, lnx64, win32, and win64).
Custom UI Layout	Tcl file to control custom GUI layout and customization. Only a single file is allowed, additional files should be added to the Utility XIT file group.
Data sheet	IP Data sheet.
Encrypted Data sheet	Encrypted IP Data sheet.
Example Implementation	Files to apply for implantation, post synthesis in an example design. Typically only XDC files should be added.
Examples Script	Script to create example project. Typically only Tcl files should be added.

Table A-2: Advanced File Group Types and Descriptions (Cont'd)

Advanced File Groups	
Examples Script Extension	Script to extend our default example project script. Typically only Tcl files should be added.
Examples Simulation	Files that make up a simulation example design. Typically exclusive of the "Examples" and "Example Synthesis" file groups.
Getting Started Guide	Getting Started Guide.
Implementation	Files that make up an implementation design. Typically this file group contains only implementation (XDC) constraints.
MATLAB Simulation	MATLAB® software simulation file.
MIF Files	MIF file.
Miscellaneous	Vivado copies any files in the group to disk during generation. It is recommended that you use another, appropriate file group.
Reference Design	Files that make up a reference design.
Software Drivers	Any created software drivers.
System C Simulation	Use to deliver files for System-C Simulation. If you are delivering pre-compiled libraries, it is suggested to deliver these in machine-specific directories following the Vivado convention (such as lnx32, lnx64, win32, and win64).
System Generator Simulation	Any System Generator simulation.
System Verilog Simulation	Use to deliver files for System-Verilog simulation.
Test Bench	One or more test bench files written in both VHDL and Verilog. Add all mixed language test bench files for delivery to the end-user, even if there are multiple top modules.
UI DRCs	User Interface Design Rule Checks.
UI Icon	GUI icon to use in IP Customization GUI.
UI Layout	Tcl file to control GUI layout and customization. Only a single file is allowed, additional files should be added to the Utility XIT file group.
Upgrade Tcl Functions	Xilinx scripts supporting upgrades from previous versions of IP. These allow the later version of provide an equivalent instance to the earlier version of IP.
Utility XIT/TTCL	Add any utility files used during TTCL, XIT, or XSpice generation. These can either be data files, include files or extra evaluation Tcl files. For XSpice, only a single Tcl file is allowed in its file group, therefore any add any additional supporting Tcl files here.
Verilog Instantiation Template	Verilog instantiation template.
Verilog Test Bench	Test bench files written in Verilog.
Version Information	Version information.
VHDL Instantiation Template	VHDL instantiation template.
VHDL Test Bench	Test Bench files written in VHDL.

Additional Resources and Legal Notices

Xilinx Resources

Support resources such as Answers, Documentation, Downloads, and Forums, are documented in [Xilinx Support](#).

Solution Centers

See the [Xilinx Solution Centers](#) for support on devices, software tools, and intellectual property at all stages of the design cycle. Topics include design assistance, advisories, and troubleshooting tips.

Vivado Design Suite Documentation

The following documents are cited within this guide:

1. *Vivado Design Suite User Guide: Design Flows Overview* ([UG892](#))
2. *Vivado Design Suite User Guide: Hierarchical Design* ([UG905](#))
3. *Vivado Design Suite User Guide: Using Constraints* ([UG903](#))
4. *Vivado Design Suite User Guide: Logic Simulation* ([UG900](#))
5. *Vivado Design Suite Tutorial: Designing with IP Tutorial* ([UG939](#))
6. *Vivado Design Suite User Guide: Designing with IP* ([UG896](#))
7. *Vivado Design Suite User Guide: System-Level Design Entry* ([UG895](#))
8. *Vivado Design Suite User Guide: Getting Started* ([UG910](#))
9. *Vivado Design Suite User Guide: Release Notes, Installation, and Licensing* ([UG973](#))
10. *Vivado Design Suite Tcl Command Reference Guide* ([UG835](#))
11. *Vivado Design Suite Tutorial: Logic Simulation* ([UG937](#))
12. *Vivado Design Suite Tutorial: Design Flows Overview* ([UG888](#))

13. *Vivado Design Suite User Guide: Using Tcl Scripting* ([UG894](#))
 14. *Vivado Design Suite User Guide: Using the Vivado IDE* ([UG893](#))
 15. *Vivado Design Suite User Guide: Designing IP Subsystems Using IP Integrator* ([UG994](#))
 16. *Vivado AXI Reference Guide* ([UG1037](#))
 17. *Vivado Design Suite Properties Reference Guide* ([UG912](#))
 18. *Vivado Design Suite Tutorial: Programming and Debugging* ([UG936](#))
 19. *Vivado Design Suite User Guide: Programming and Debugging* ([UG908](#))
 20. *Vivado Design Suite Tutorial: Creating and Packaging Custom IP* ([UG1119](#))
 21. [Vivado Design Suite QuickTake Video Tutorials](#)
 22. [Vivado Design Suite Documentation](#)
-

Xilinx IP Documentation

23. *LogiCORE IP Integrated Logic Analyzer Product Guide* ([PG172](#))
 24. *LogiCORE IP IBERT for 7 Series GTX Transceivers* ([PG132](#))
 25. *LogiCORE IP IBERT for 7 Series GTP Transceivers* ([PG133](#))
 26. *LogiCORE IP IBERT for 7 Series GTH Transceivers* ([PG152](#))
 27. *LogiCORE IP Virtual Input/Output Product Guide* ([PG 159](#))
-

Training Resources

Xilinx provides a variety of training courses and QuickTake videos to help you learn more about the concepts presented in this document. Use these links to explore related training resources:

1. [Essentials of FPGA Design](#)
2. [Embedded Systems Software Design](#)

Please Read: Important Legal Notices

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third-party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at <http://www.xilinx.com/legal.htm#tos>; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at <http://www.xilinx.com/legal.htm#tos>.

AMBA, AMBA Designer, ARM, ARM1176JZ-S, CoreSight, Cortex, and PrimeCell are trademarks of ARM in the EU and other countries.

© Copyright 2012-2014 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, UltraScale, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.