Vivado Design Suite User Guide

Designing with IP

UG896 (v2014.1) May 1, 2014

XILINX
ALL PROGRAMMABLE™

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
05/01/2014	2014.1	Initial 2014.1 release.

Table of Contents

Appendix C: IP Optimization (FMax Characterization)

Appendix D: AXI Naming Conventions

Appendix E: Adding User Logic to Your AXI4 Peripheral

Appendix F: Additional Resources and Legal Notices

IP-Centric Design Flow

Introduction

The Xilinx® Vivado® Integrated Design Environment (IDE) provides an IP-centric design flow that lets you add IP modules to your design from various design sources. Central to the environment is an extensible IP catalog that contains Xilinx-delivered "Plug-and-Play" IP. The IP catalog can be extended by adding the following:

- Modules from System Generator for DSP designs (MATLAB/Simulink algorithms) and Vivado High-Level Synthesis designs (C/C++ algorithms)
- Third-party IP
- Designs packaged as IP using the Vivado IP packager

Figure 1-1 illustrates the IP-Centric design flow.

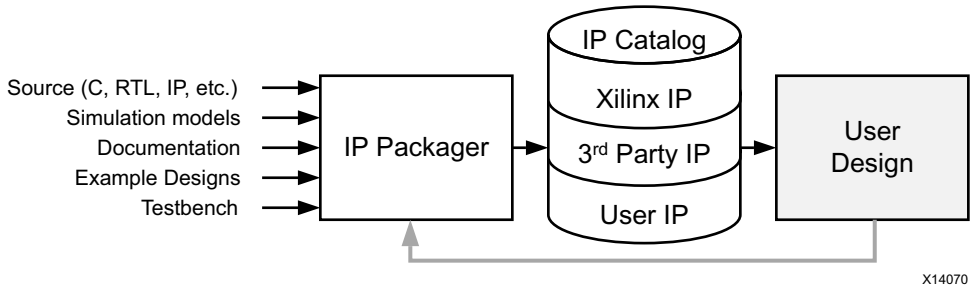


Figure 1-1: **IP-Centric Design Flow**

Note: In some cases, third-party providers offer IP as synthesized EDIF netlists. You can load these files into a Vivado IDE design using the **Add Sources** command.

The available methods to work with IP in a design are as follows:

- Use the Managed IP Flow to customize IP and generate output products, including a synthesized Design Checkpoint (DCP).
- Use IP in either Project or Non-Project modes by referencing the created Xilinx Core Instance (XCI) file, which is a recommended method for large projects with many team members.
- Create and add IP within a Vivado Design Suite project. Access the IP catalog in a project to create and add IP to design. Store the IP either inside the project or save it externally to the project, which is the recommended method for projects with small team sizes.
- Create and customize IP and generate output products in a Non-Project script flow, including generation of a DCP.

IP Terminology

The Vivado IDE uses the following terminology to describe IP, where it is stored, and how it is represented

- **IP Definition**: The description of the IP-XACT characteristics for IP.
- **IP Customization**: Customizing an IP from an IP definition, resulting in an XCI file.
- **IP Location**: A directory that contains one or more customized IP.
- **IP Repository**: A unified view of a collection of IP definitions.
- **Output Products:** Generated files produced for an IP customization. They can include HDL, constraints, and simulation targets. The XCI file stores the configuration and constraint options that are user-specified during customization. During generation, these customizations are used to produce the files that are used during synthesis and simulation.

IP Catalog

The IP Catalog allows for the exploration of Xilinx "Plug-and-Play" IP, as well as other IP-XACT compliant Intellectual Property. This can include designs that you package as IP. See Chapter 2, IP Basics for more information.

IP Packager

The Vivado IP packager lets you create plug-and-play IP and add that IP to the extensible Vivado IP Catalog. Figure 1-1, page 6, illustrates the IP packager, based on the IEEE Standard for IP-XACT (IEEE Std 1685), Standard Structure for Packaging, Integrating, and Reusing IP within Tool Flows.

After you have assembled your Vivado IDE user design, the IP packager lets you turn your design into a reusable IP module that you can then add to the Vivado IP Catalog, and that others can use for design work. You can use packaged IP within a Project Mode-based IP flow.

You can locate the IP within the project or use a remote location (recommended).

See Chapter 8, Creating and Packaging IP for more information about using the Packaging feature.

IP Basics

Introduction

This chapter describes the basic features of the IP catalog, how to create and instantiate IP, and how to generate output products for use within a design.

Working with Xilinx® IP consists of first creating a customization of the IP. You can create an IP customization in various ways using the Vivado® Integrated Design Environment (IDE), as follows:

- Directly in a project using the IP Catalog
- Using the Manage IP project flow
- Using a Tcl script to create IP customization

After creating a customization, you can either automatically generate output products, or defer until later.

- In a Project Mode flow, if the output products are not present, they are generated automatically prior to synthesis or simulation.
- In a Non-Project Mode flow, you must generate the output products prior to synthesis or simulation.

To use a customization in a design, instantiate the IP in the HDL code. One of the IP output products is an instantiation template based upon the target language set in the project settings.

IP Catalog Features

The key features of the Vivado IP catalog include:

- Consistent, easy access to Xilinx IP, including building blocks, wizards, connectivity, DSP, embedded, AXI infrastructure, and video IP from a single common repository regardless of the end application being developed.

- Support for multiple physical locations, including shared network drives, allowing users or organizations to leverage a consistent IP deployment environment for third-party or internally-developed IP.
- Access to the latest version of Xilinx-delivered IP, which is rigorously tested prior to inclusion in the IP catalog.
- Access to IP customization and generation using the Vivado IDE or an automated script-based flow using Tcl.
- On-demand delivery of IP output products such as instantiation templates and simulation models (HDL, C, or MATLAB® software).
- IP example designs that provide capability to evaluate IP directly as an instantiated source in a Vivado Design Suite project.
- Access to version history details as recorded in the Change Log.
- A `<major#.minor#.Rev#>` numbering scheme unifies the IP version numbers. For information, see the Xilinx IP Versioning page available from the Xilinx website: www.xilinx.com/ipcenter/vivado_ip_versioning.htm
- Catalog filter options that let you filter by Supported Output Products, Supported Interfaces, Licensing, Provider, or Status.

For information on IP that supports the Vivado Design Suite, see www.xilinx.com/support/index.html/content/xilinx/en/supportNav/ip_documentation.html.

For information on specific IP, see www.xilinx.com/ipcenter or look at the IP catalog.

See www.xilinx.com/ipcenter/axi4.htm for information on AXI IP.

Using the IP Catalog

The IP catalog contains categories of IP that you can filter and search.

Figure 2-1 shows the IP Catalog Filter options that let you filter the IP catalog by categories. Select the **Filter** button  in the top left corner to open the filter options.

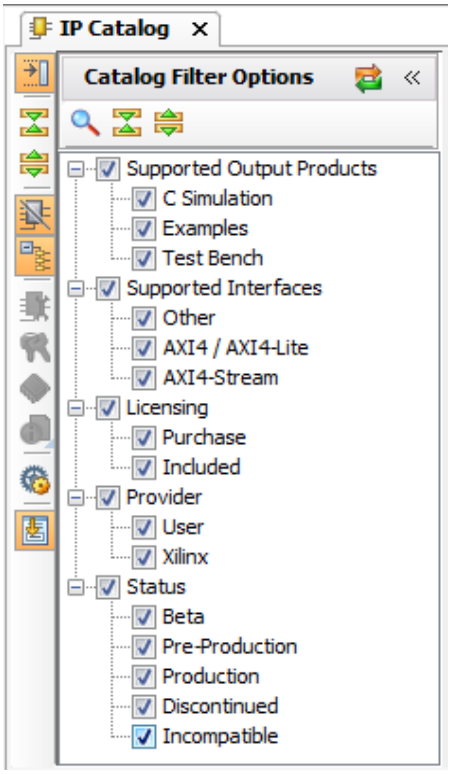
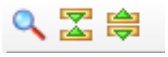


Figure 2-1: **IP Catalog Filter Options**

Uncheck the filter options to filter out IP that meets unwanted criteria.

IP Catalog options give you the ability to expand, collapse, and zoom the IP catalog content.  These icons are consistent with other Vivado IDE.

Other filter options are:

Option	Description
	Hide Incompatible IP
	Group by Category

Option	Description
	Customize IP
	License Status
	Show Compatible Families
	View Product Guide, Change Log, Product Webpage, and Answer Records
	Settings for IP catalog, IP generation, and IP packager
	Auto-scroll to Selected Items

IMPORTANT: *It is possible to create duplicate IP. The Vivado tools issue a warning. See Resolving Duplicate IP, page 20 for the possible resolutions.*

Synthesis Options for IP

When generating the output products for an IP, the default behavior is to produce an out-of-context (OOC) synthesized design checkpoint (DCP). Alternatively, you can choose to synthesize the IP along with the top level user logic, which is called *global synthesis*.

Figure 2-2, page 13, shows the two flows possible for synthesizing IP:

- The default OOC flow
- Synthesizing the IP HDL along with the top-level user HDL (global synthesis)

In either flow, Vivado IDE generates HDL and XDC files for the IP, and uses those files during synthesis and during implementation.

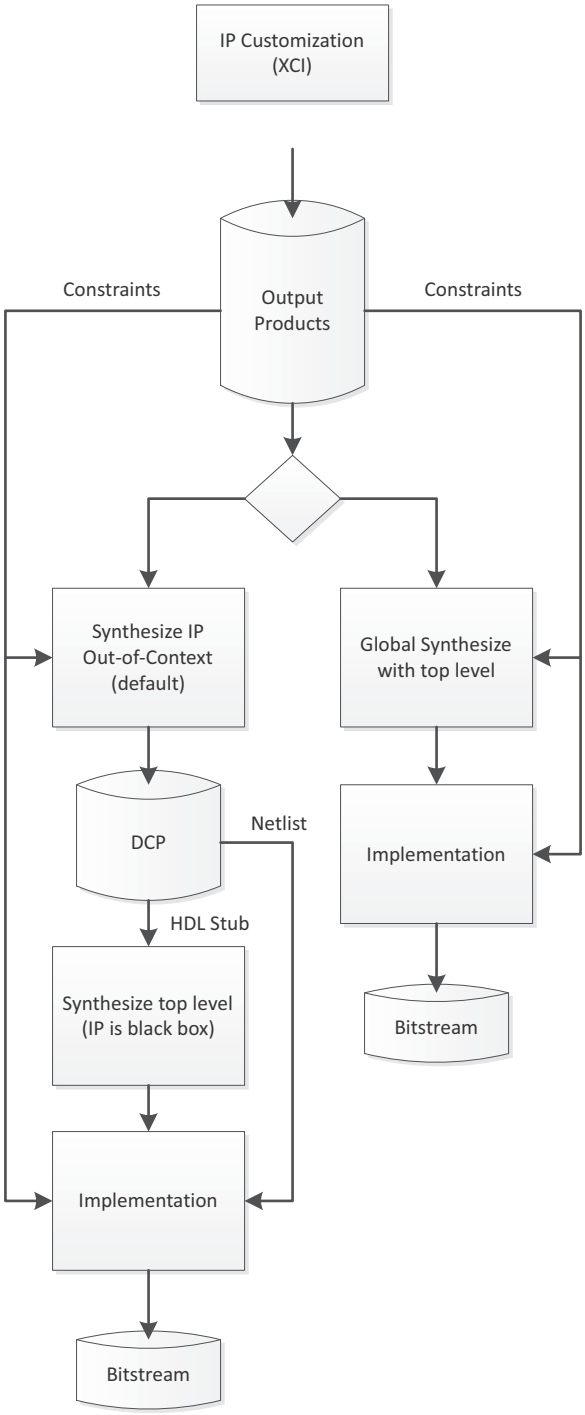


Figure 2-2: **Out-Of-Context (OOC) and Global Synthesis Flow**

Out-of-Context Flow

In the OOC flow, the IP is synthesized alone and an OOC DCP is produced. A special OOC flow-only XDC output product file (the `_ooc.xdc`) is used when synthesizing the IP, which provides default input clock definitions. The produced DCP is a container file, and includes a netlist as well as constraints.

When synthesizing the entire design, an HDL stub module is provided in the DCP which causes a black box to be inferred for the IP.

Also, during synthesis of the entire design, the DCP provides an XDC file which defines any clocks an IP might output (the `_in_context.xdc`).

During implementation, the netlist from the IP DCPs are linked with the netlist produced when synthesizing the top-level design files, and the Vivado IDE resolves the IP black boxes. The IP XDC output products that were generated for use during implementation are applied along with any user constraints.

The OOC flow is the default flow because of two main benefits:

- It improves synthesis run times because you synthesize the IP only once.
- It produces either a `_funcsim.v or a _funccsim.vhdl` structural simulation model. You can use these files during simulation if you use a single language simulator and the IP does not deliver behavioral HDL in that language.

Implementing IP OOC

By default, the Vivado IDE creates a synthesized design checkpoint (DCP) file automatically during the customization process for most Vivado Design Suite IP.

When performing synthesis of the top-level design, IP with an associated DCP file is a black box because it is being synthesized OOC.

Use this procedure to perform manual analysis of the IP standalone after implementation:

1. In the Design Runs window, right-click the IP design run, and select **Launch Runs** to launch the implementation run as shown in Figure 2-3.

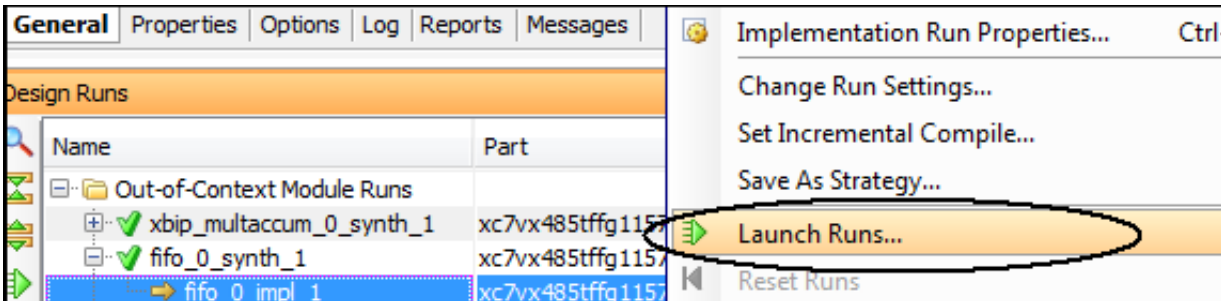


Figure 2-3: **Implementing an IP Standalone**

2. After implementation completes, right-click the IP design run, and select **Open Implemented Design** to open the design for analysis, as shown in Figure 2-4.

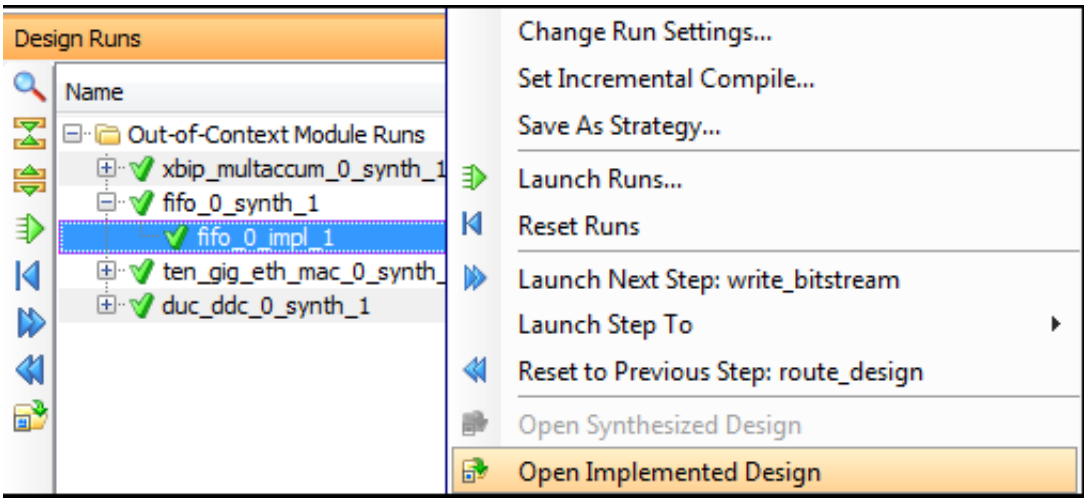


Figure 2-4: **Opening an Implemented IP**

When performing timing analysis, the results are not accurate because the clocks are not yet routed and ideal clocks are used. This is most obvious when performing hold analysis, because the router cannot fix hold violations.

Some IP include the `HD.CLK_SRC` property in the `<IP_Name>_ooc.xdc` file, which improves the accuracy of post-implementation timing analysis. This property provides a location for a clock buffer, and the timer is able to more accurately model SLEW.

IMPORTANT: *The implemented IP is for analysis only and is not used during synthesis or implementation of the top-level design. For information on using an implemented version of the IP, see the Vivado Design Suite User Guide: Hierarchical Design (UG905)* [Ref 2].

Out-Of-Context Settings

When working with Vivado Design Suite IP, you can disable the generation of a DCP and instead synthesize the IP RTL with the top-level RTL.

RECOMMENDED: *Use the OOC flow when generating IP. The OOC flow speeds up run time for the complete project, and lets you avoid re-synthesizing IP when doing project runs.*

In the Generate Output Products dialog box, click **Out-of-Context Settings**.

In the Out-of-Context Settings dialog box (Figure 2-5), you can:

- Uncheck the check box. This prevents the IP files from being automatically generated out-of-context to the design for synthesis.
- Specify the number of DCP generation jobs to run at one time.

By default, one job is specified, and the design runs launch sequentially. A higher number specifies the maximum number of design runs that can be running in parallel.

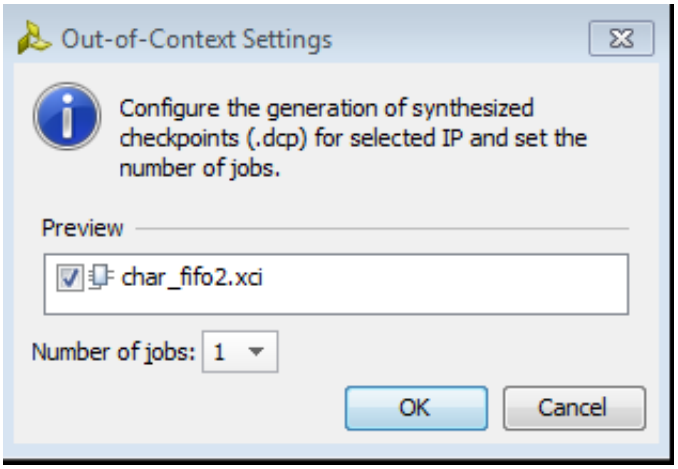


Figure 2-5: **Out of Context Dialog Box**

After changing these settings, you can either generate the output products or delay output product generation. DCP generation preferences are preserved whether or not you generate output products now.

For a scripted flow, you can disable the automatic generation of the DCP in either Project Mode or Non-Project Mode by setting the `GENERATE_SYNTH_CHECKPOINT` property to `FALSE` for the XCI file, as shown in the following Tcl command:

Tcl Command to Disable OOC Options on IP

```
set_property GENERATE_SYNTH_CHECKPOINT FALSE \
  [get_files <IP_Name>.xci]
```

Global Synthesis Flow

In the global synthesis flow, the IP is synthesized along with user HDL. Any changes made to user HDL result in the IP being re-synthesized as well.

During implementation, the IP XDC output products that were generated for use during implementation are applied along with any user constraints.

RECOMMENDED: *Always reference the IP using the XCI file. It is not recommended to read just the IP DCP file, either in a Project Mode or Non-Project Mode flow. While the DCP does contain constraints, it does not provide other output products that an IP could deliver and that could be needed, such as ELF, COE, and Tcl scripts.*

Generating Output Products

After IP customization is complete, the Generate Output Products dialog box opens, as shown in Figure 2-6.

Output products delivered by the IP are listed in the Preview area, and you can do any of the following:

- To generate the listed output products, click **Generate**. By default, the Vivado IDE creates an XCI and a DCP for the IP when you generate the output products, as well as a change log, a behavioral simulation model, and an instantiation template.
- To delay generation of output products, click **Skip**. This lets you select multiple IP customizations and generate all output products at one time, including launching parallel synthesis runs for IP DCP files.

  Note: When working in Project Mode, output products are automatically generated as needed prior to synthesis of the top-level design. This includes any specified DCP files.
- The Vivado IDE generates the synthesized IP DCPs out-of-context. See the Synthesis Options for IP, page 12.

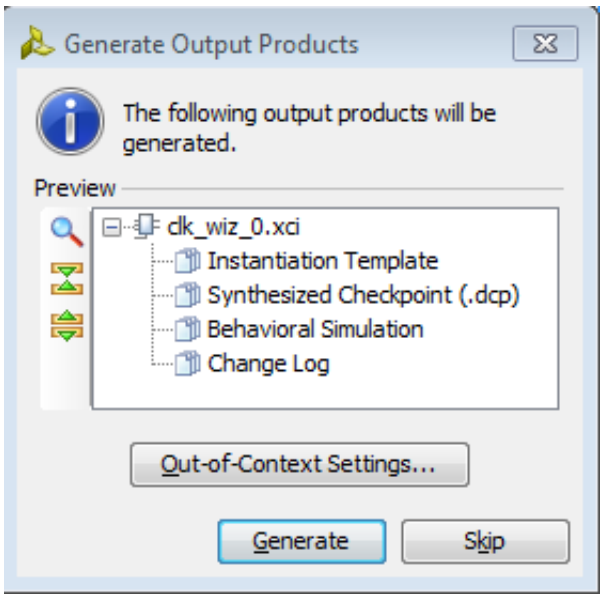


Figure 2-6: **Generated Output Products**

Examining Generated Output Products

The IP Sources window shows the generated output products for all IP in the project. If you skipped generation of the output products an instantiation template is the only generated product (besides the XCI and BOM files, which are not displayed) as shown in Figure 2-7.

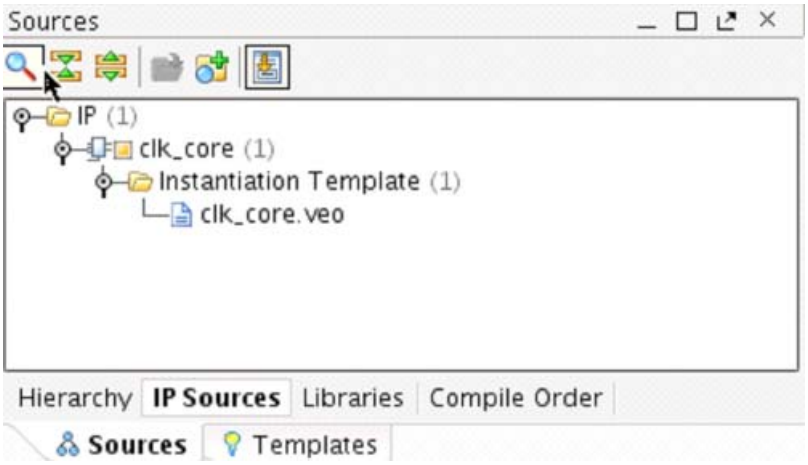


Figure 2-7: **IP Customization with Generation of Output Products Skipped**

If the output products are generated, all unencrypted files are listed as shown in (Figure 2-8).

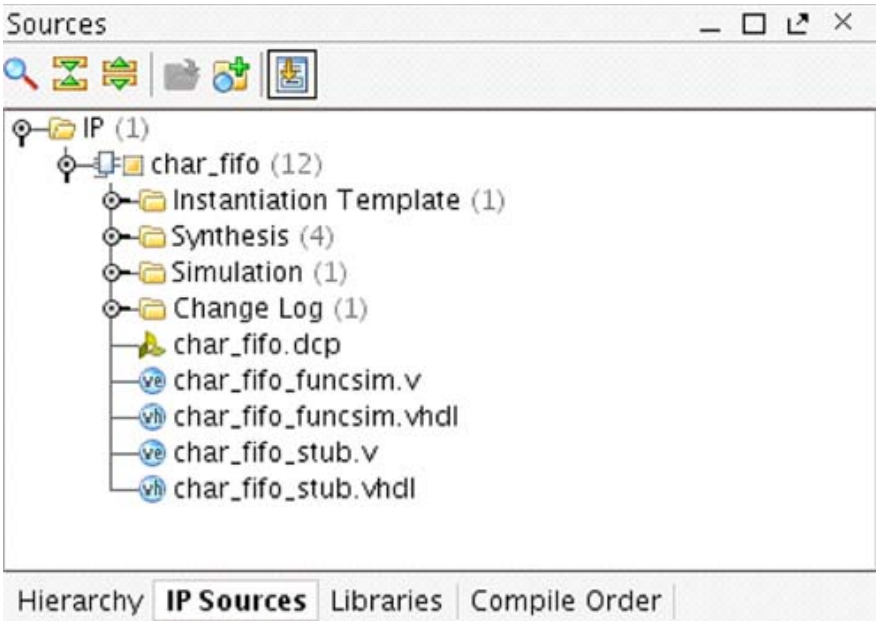


Figure 2-8: **IP Customization with Output Products Generated, Including OOC DCP**

These include: an instantiation template, synthesis and simulation targets, XDC constraints, a change log, and other products.

By default the Vivado IDE creates an OOC DCP along with structural simulation netlists (`_funcsim.v`/`_funcsim.vhdl`). Stub files (`*_stub.v`/`*_.vhdl`) are created for use with third-party synthesis tools to infer a black box for the IP.

After the synthesis output products are generated, the Vivado IDE creates and launches a design run to produce the OOC DCP.

While the synthesis run is processing, the OOC related files are shown as missing (Figure 2-9).

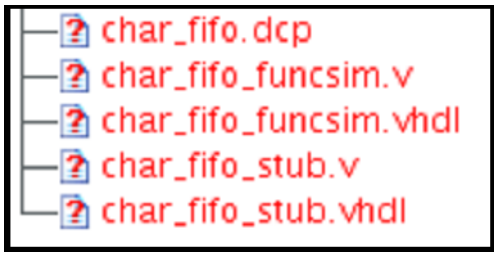


Figure 2-9: **OOC Flow Output Products Pending Creation**

If you elected to not generate the OOC DCP, the DCP, the Vivado IDE does not create `_funcsim`, and stub files.

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 a project as an 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 is available unless a project is created.

Using the Repository Manager

1. Select **Tools > Project Settings**.
2. In the Project Settings dialog box, click **IP** in the left pane.
3. Click the **Repository Manager** tab (Figure 2-10), 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. You can either package 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 as described in Creating an IP Customization.
   c. To see the IP within each repository, click **Apply**.

Resolving Duplicate IP

If you enter an IP that is a duplicate of an existing IP, the Repository Manager issues a warning banner.

- Review the duplicated IP in the IP Catalog.
- Review the directories in the description.
- If necessary, remove the IP or remove the repository that contains the IP.

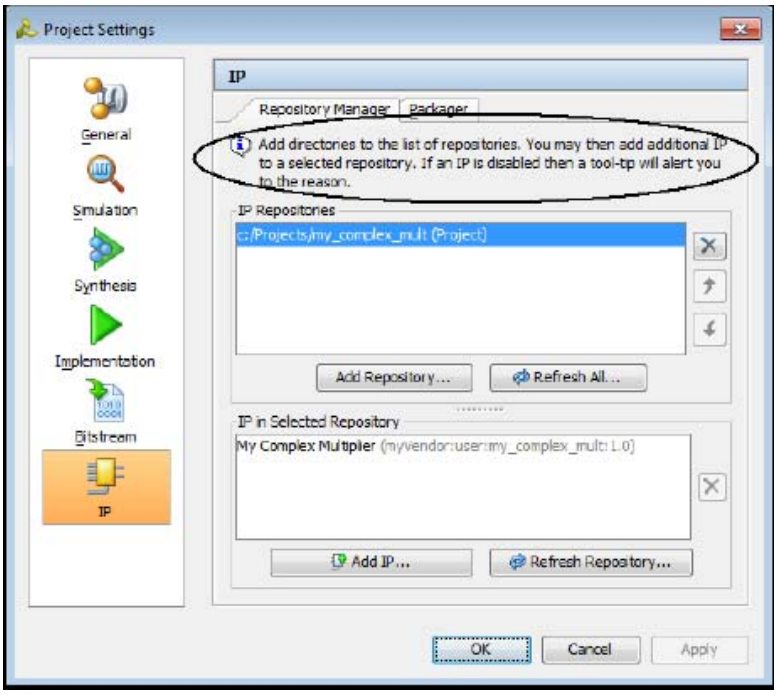


Figure 2-10: **IP Project Settings-Repository Manager Tab**

Using the Packager Settings

To set the Packager options:

1. Click the **Packager** tab (Figure 2-11):

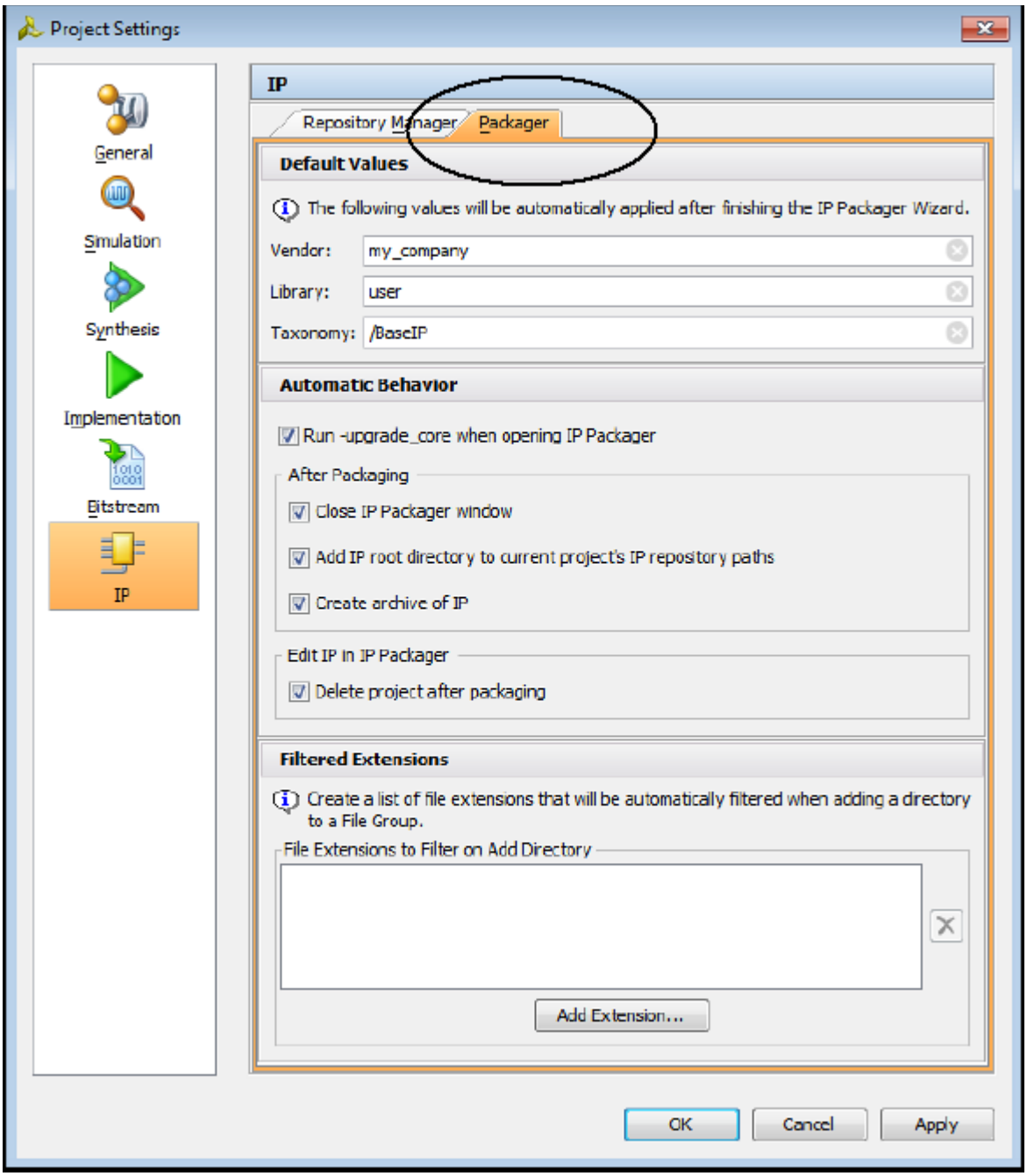


Figure 2-11: **IP Project Settings—Packager Tab**

2. Fill out the following information:
   a. 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 sections 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. For more information on using the IP Packager, see the *Vivado Design Suite Tutorial: Designing with IP* (UG939) [Ref 5].

   b. In **Automatic Behavior**, check or uncheck the options you want:

   ◦ **After Packaging**:
      - **Close IP Packager window**: Closes the 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.

   c. 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.

Creating an IP Customization

You can create an IP customization using the IP Catalog. The steps are:

1. Locate an **IP Definition** that matches your design requirements.
2. Customize that IP for your design or for the general use of a design team, thus creating an *IP Customization*.
3. Save the IP customization inside a project directory (default), or into a centralized directory.
4. Select an IP from the IP Catalog and customize the IP for use in your design.
   a. From the **IP Catalog**, select the IP.

b. Double-click the selected IP, or select the **Customize IP** command from the toolbar.

The Customize IP dialog box shows the various parameters available for you to customize the IP. This interface varies, depending on the IP you select, and can include one or more tabs in which to enter values.

The Customize IP dialog box includes the following:

- An IP symbol
- Various tabs for setting configuration options for the specific IP

The IP symbol supports zoom, re-size, and auto-fit options that are consistent with the schematic viewer canvas in Vivado IDE. Figure 2-12 shows the Customize IP interface for the FIFO Generator IP.

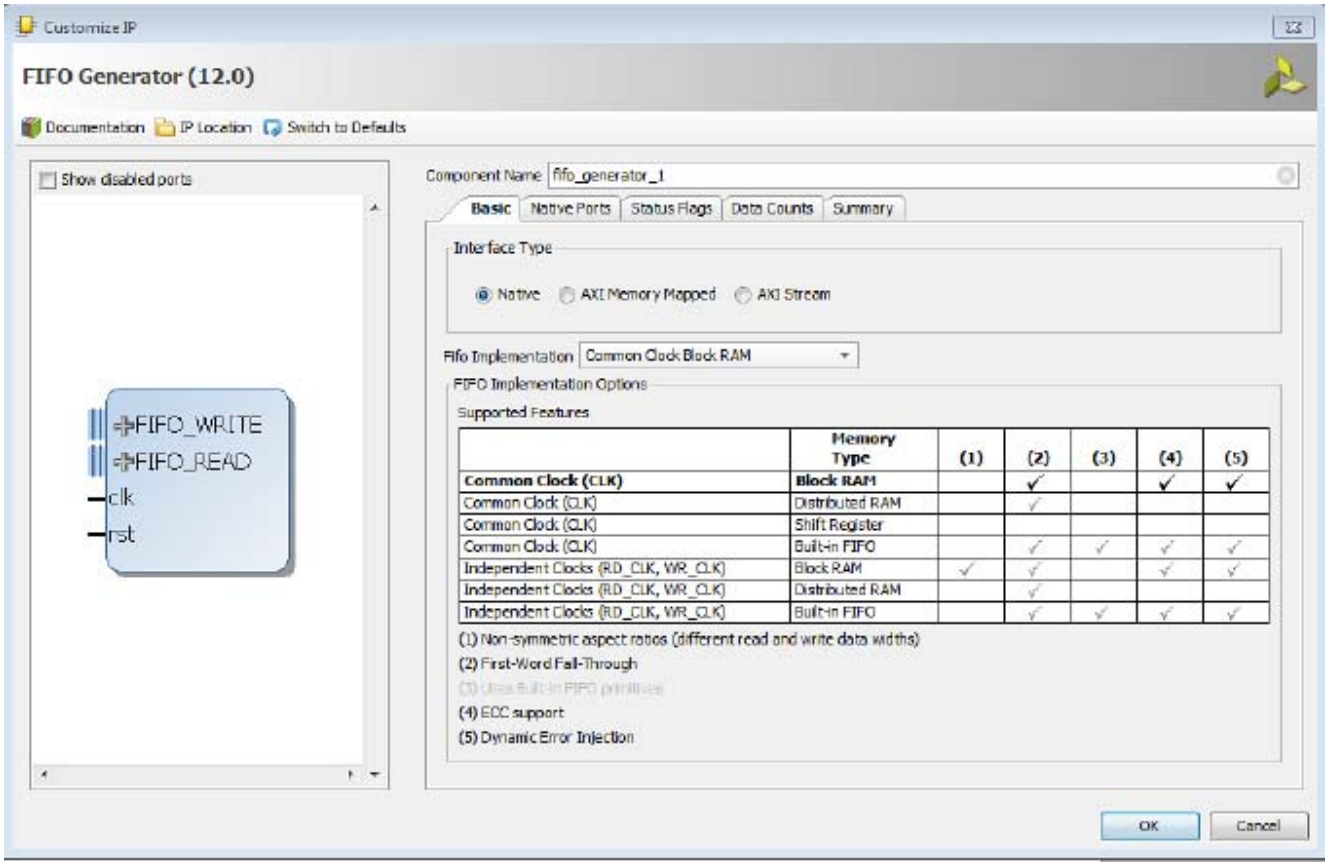


Figure 2-12: **FIFO Generator Dialog Box**

5. In the IP dialog box:
   - Click **Documentation > Product Guide** to open the product guide for the selected IP.
   - Click **IP Location** to specify the location on disk to store the IP. You can only change this location when using the IP catalog in an RTL project. This is *not an option* in a Manage IP project.
   - Click **Switch to Defaults** to display a window asking if you want to reset all configuration options back to their default starting point.
6. Customize the IP as needed for your design, and click **OK**.

VIDEO: *See the* Vivado Design Suite QuickTake Video: Customizing and Instantiating IP *for a demonstration of the IP Customization and Instantiation process.*

Tcl Command Example for Creating an IP Customization

You can also create IP customizations using the `create_ip` Tcl command. For example:

```
create_ip -name fifo_generator -version 12.0 -vendor xilinx.com -library ip
-module_name fifo_gen
```

You must specify either `-vlnv` or all of `-vendor`, `-library`, `-name`, and `-version`.

Note: Executing the `create_ip` Tcl command creates the IP customization file (XCI), which is the configuration for the IP, as well as the instantiation template and BOM (XML) file, but does not create any other output products.

The default configuration for the IP is set with the `create_ip` command.

Tcl Command Example for Setting IP Properties

To configure with different setting use the `set_property` command. For example:

```
set_property CONFIG.Input_Data_Width 12 [get_ips fifo_gen]
```

Tcl Command Example for Reporting IP Properties

To get a list of properties available for an IP use the `report_property` command. For example:

```
report_property [get_ips fifo_gen]
```

Configuration properties start with `CONFIG`.

Re-Customizing Existing IP

You can re-customize existing IP in either an RTL project or in a Manage IP project.

1. To open the IP customization dialog box for an IP, you can either:
   - Double-click the IP.
   - In the **IP Sources** view, right-click the IP, and select **Re-customize IP** from the menu as shown in Figure 2-13.

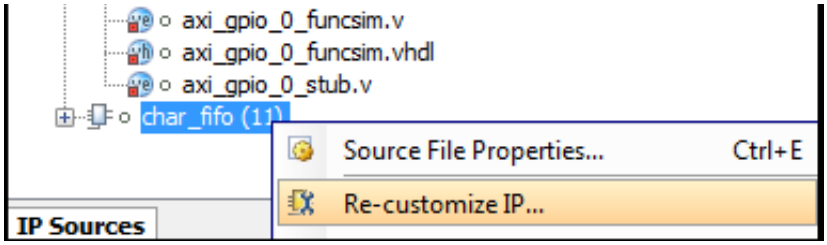


Figure 2-13: **Re-customize IP Option**

2. Change IP parameterization options, and press **OK**.

   The Generate Output Products dialog box opens.

Alternatively, you can view the current settings, and click **Cancel** to keep the settings intact.

When you do make changes to the IP configuration and generate the output products, the existing products are reset and, if enabled, the design run resets and launches again.

RECOMMENDED: *Always generate the output products for IP, including the synthesized DCP.*

Manually Generating Output Products

At any point you can manually generate output products as follows:

1. In the IP Sources or the Hierarchy view, select the IP.
2. Right-click and select **Generate Output Products**.

The default flow when generating output products is to create and launch an Out-Of-Context (OOC) synthesis design run for the IP. This results in the inference of a black box for the IP when synthesizing the rest of the design.

IMPORTANT: *The implementation stage resolves black boxes by extracting the netlists from the DCP of the IP.*

Note: If you do not want to automatically create an OOC run during synthesis, uncheck the **Generate Synthesized Checkpoint** check box and press **Generate,** as described in Re-Customizing Existing IP, page 25.

RECOMMENDED: *OOC is the recommended flow. Using the OOC reduces the run time on synthesizing your design: you do not need to synthesize the IP every time you run synthesis during development.*

Instantiating an IP

After you create an IP Customization, you can then instantiate that IP in a design. An instantiation template is created after IP customization, regardless of whether you generated output products. The instantiation template is available in the IP Sources tab of the Sources window in the Instantiation Template folder as shown in Figure 2-14.

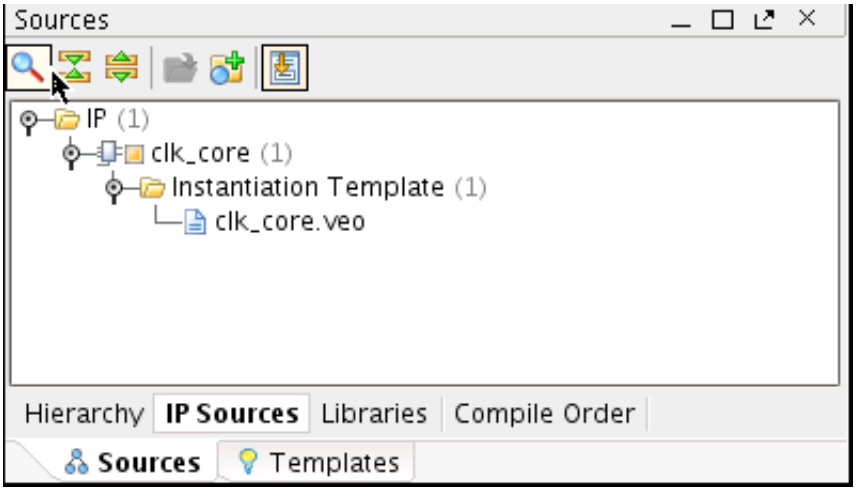


Figure 2-14: **Location of Instantiation Template**

Depending on the set project target language, either a VHO or VEO file is present, which contains the instantiation template that you can copy and paste into your RTL design. Figure 2-15, page 27 shows the instantiation template for a Clocking Wizard IP customization.

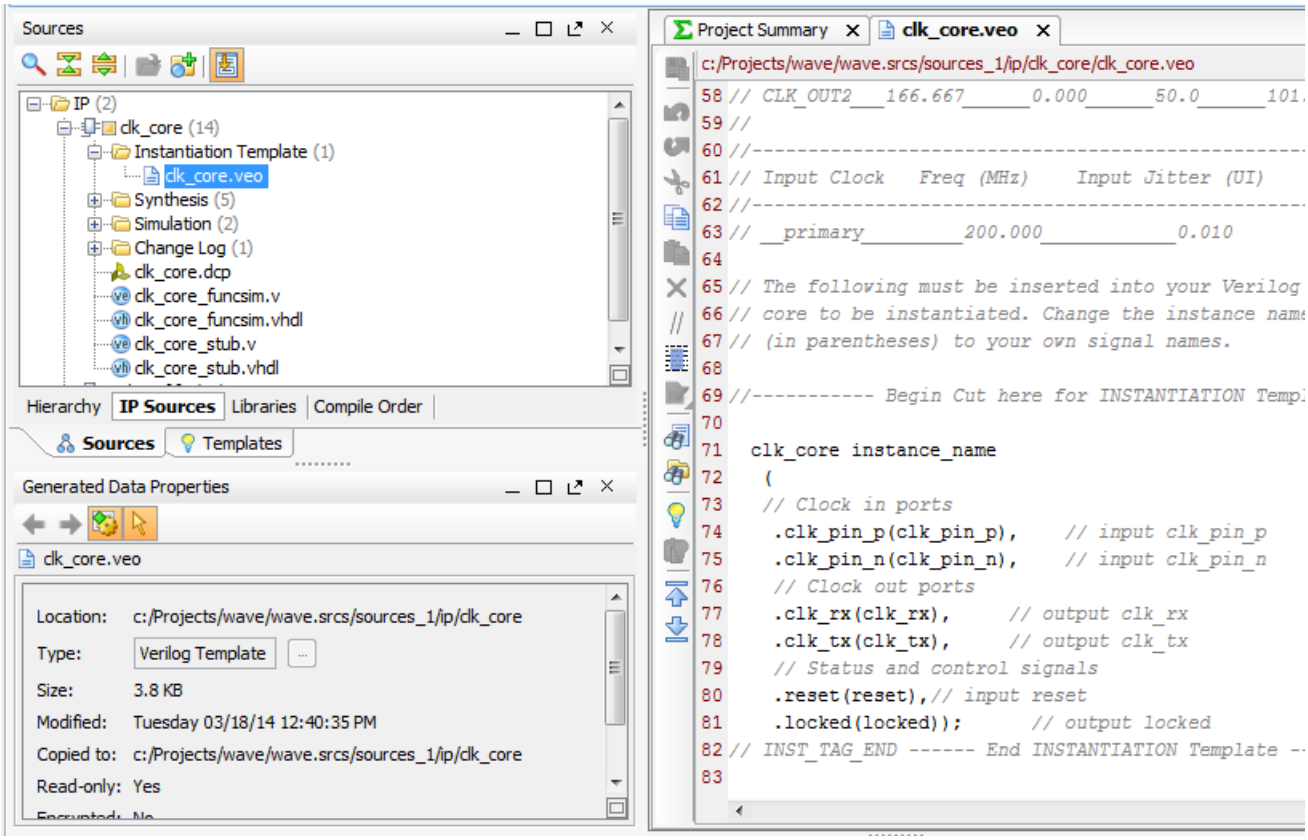


Figure 2-15: **Instantiation Template**

To use the instantiation template in your design:

1. Open either the VEO or VHO template file for the IP customization by double-clicking the file in the Sources view, or by selecting the file using the **Open Files** option.
2. Highlight the Instantiation Template between the comments as indicated and copy the section.
3. Open HDL file in which you want to instantiate the IP.
4. Paste the copied template to the location of your choice.
5. Edit the HDL to integrate the template into your design as needed; for example, change the port connections.

   After the IP is instantiated in a design, the IP files show in the Hierarchy tab Sources window at the location in the design where it is instantiated, as shown in Figure 2-16, page 28, which shows two IP instantiated in a design.

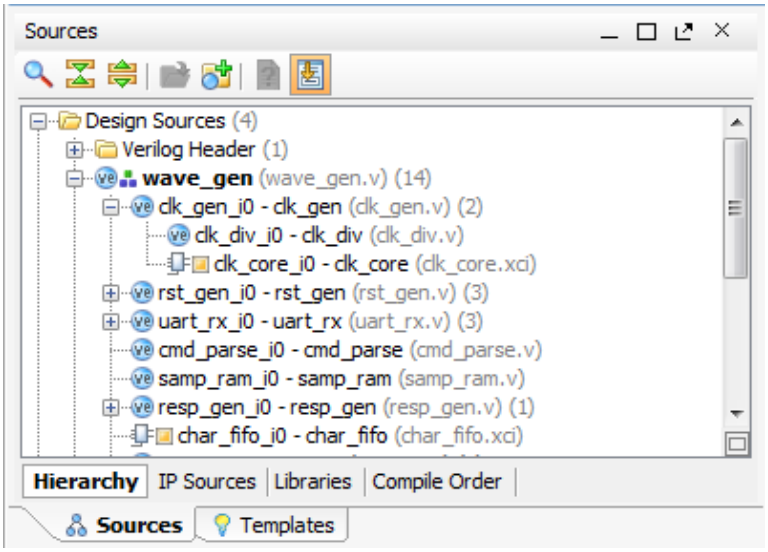


Figure 2-16: **IP Instantiated in a Design**

With the IP customization properly instantiated into your design, you are ready to synthesize the IP along with the rest of your design, either as a black box if the OOC flow is used or together you are using global synthesis. See Synthesis Options for IP, page 12 for more details.

If you expand the IP hierarchy by right-clicking the IP, you may see this icon , which denotes an encrypted IP source. This source cannot be viewed.

Adding Existing IP to a Project

You can add previously created CORE Generator™ IP (`<IP_Name>.xco` files) or Vivado IP (`<IP_Name>.xci` files) by using the **Add Existing IP** option in the Add Sources dialog box, as shown in Figure 2-17, page 29.

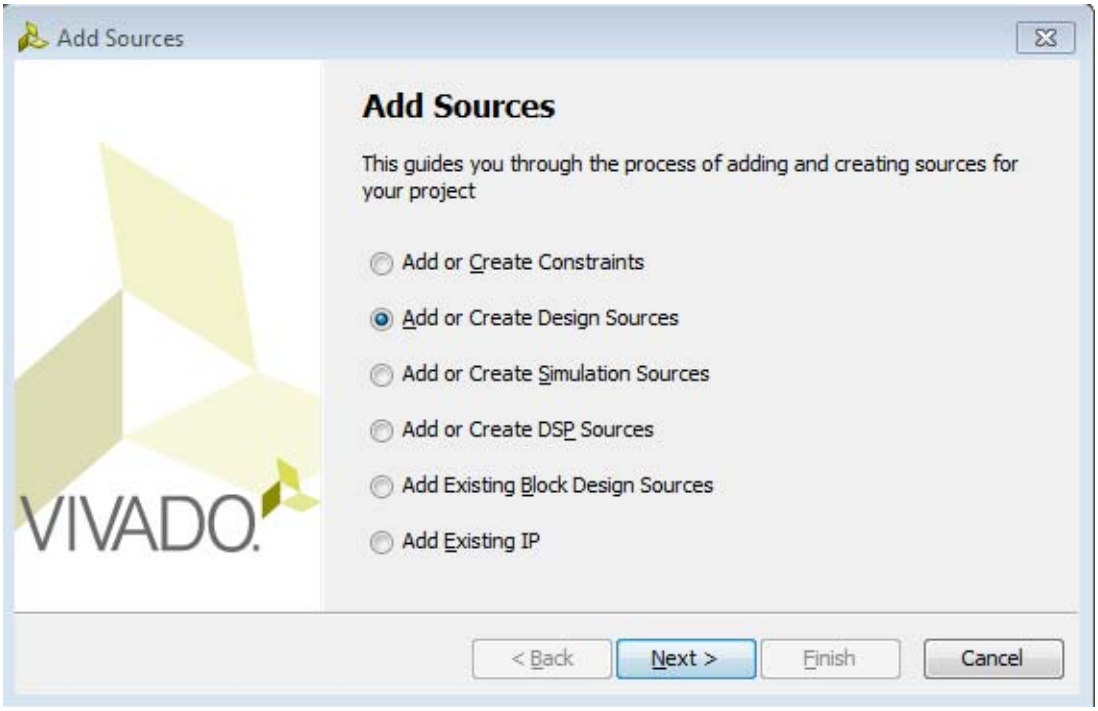


Figure 2-17: **Add Sources Dialog Box**

You can either reference the IP and any generated output products from the location specified or copy the IP and any generated output products into your project.

See Chapter 3, Using Manage IP Projects for details on creating IP using the Managed IP flow.

After you click **Next**, the Add Existing IP wizard opens, as shown in Figure 2-18, page 29.

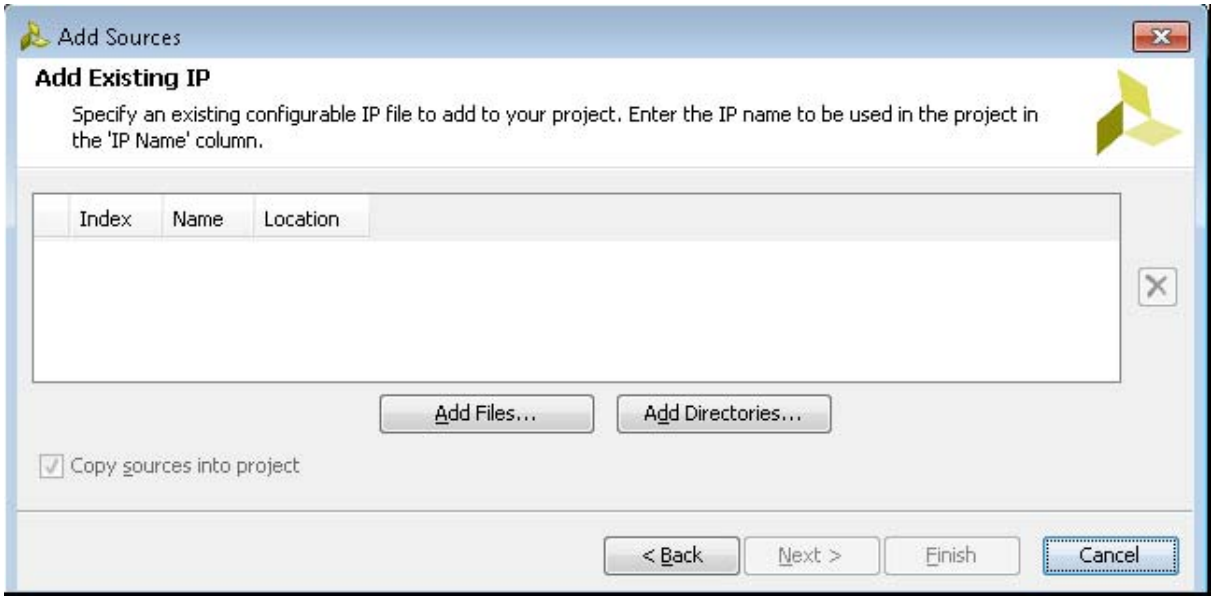


Figure 2-18: **Add Existing IP**

The added IP and any output products are shown in the IP Sources tab of the Sources view, as well as with other source files in the Hierarchy, Libraries, and Compile Order views.

You can select the IP in the IP Sources and view the properties for it in the Source File Property window.

Note: You can also add EDIF, Verilog or SystemVerilog netlists, or NGC files for IP cores into either RTL or netlist-based projects. For more information, see "Creating a Post-Synthesis Project" in Chapter 2 of the *Vivado Design Suite User Guide: System-Level Design Entry* (UG895) [Ref 7].

IMPORTANT: *The Vivado Design Suite does not support UCF constraints. When using an NGC or an XCO file, you must provide XDC constraints.*

Tcl Command Examples

The following are commonly used IP Tcl command examples.

Tcl Command Example to Add Existing IP

You can add existing IP using the following Tcl Command:

```
import_ip
```

Tcl Command Examples to Import IP

The following are examples of `import_ip`:

```
import_ip -files C:/vivado_ip/aurora_8b10b_v7_1.xci -name aurora_1234_v7_1
import_ip -files C:/vivado_ip/blk_mem_gen_v6_1.xci
```

Tcl Command Example to Read IP

To remotely access an IP, use:

```
read_ip <IP_Name>
```

This command does not copy IP into the project.

Tcl Command Example to Add Files

```
add_files
```

Upgrading IP

Future versions of, or patches for, the Vivado Design Suite might have newer versions of IP that are used in a project. Only one version of an IP is delivered in each release of the Vivado Design Suite.

Prior to moving to a new Vivado Design Suite release, do one or more of the following:

- Generate all the output products for the IP, including the DCPs. This lets you use the old version of the IP in the new release of the Vivado Design Suite, if needed.

IMPORTANT: *It is especially important for IP that have a major revision change between Vivado Design Suite releases because these IP typically require RTL changes.*

Note: You cannot generate output products, including DCPs, for IP that is not the current version.

- If you are using Manage IP projects, copy the entire Manage IP project location as a backup.
- Archive design projects that contain IP.

RECOMMENDED: *When migrating the project to the newer version of the Vivado Design Suite, some IP might become locked because they are not current. It is recommended that the IP be upgraded.*

Before upgrading IP, view the change log for information on the changes.

When an upgrade is available for an IP, you can upgrade the IP using either of the following methods:

- In the IP Sources window or the Hierarchy window, right-click the IP, and select **Upgrade IP** as shown Figure 2-19.
- In the IP Status window (Figure 2-21, page 33) that opens when you run the **Report IP Status** command, click the **Upgrade** option next to the IP.

Figure 2-19: **Upgrade IP Option**

Note: When upgrading IP that is in a board design, see the *Vivado Design Suite User Guide: Designing IP Subsystems Using IP Integrator* (UG994) [Ref 17] for upgrade instructions.

VIDEO: *See the* Vivado Design Suite QuickTake Video: Managing Vivado IP Version Upgrade *for a demonstration of upgrading IP.*

Upgrading IP Using a Tcl Command

You can also use the `upgrade_ip` Tcl command to upgrade IP, as follows:

```
upgrade_ip [get_ips clk_core]
```

Note: If no argument is given, the command upgrades all IP in the project to the latest version, if an upgrade path exists.

Upgraded IP creates an update log automatically. These logs are available from the `/IP Update Log` folder in the Design Sources window (Figure 2-20).

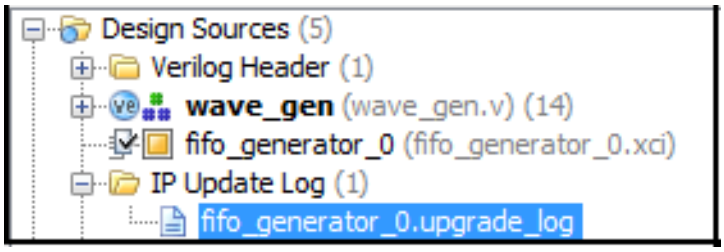


Figure 2-20: **IP Update Log Folder**

This log contains information about the IP upgrade, such as:

- If the upgrade is successful
- If user intervention is required

  Note: If the upgraded IP requires user intervention, the log lists the issues encountered, such as parameters that no longer exist.
- The original version and revision of the IP
- The upgraded version and revision of the IP
- Additional information as appropriate:
  - A description of changes made by the upgrade script.

    Examples of such information are: "`Renamed parameter DATA_W to C_DATA_WIDTH`", and "`Set parameter BUFFER_LENGTH to value 32`").
  - Warnings issued during customization of the IP, such as ports added, changed, and removed during an upgrade.
  - Warnings associated with the upgrade; either general issues relating to the upgraded version, or specific issues related to the current parameterization.
  - Any warnings issued during customization of the IP.

CAUTION! *When upgrading an IP, all previously generated output products are removed, including the DCPs and any associated design runs.*

Viewing the IP Status Report

You can view the status report of IP in a project using the **Tools > Report > Report IP Status** pull-down menu. A new tab titled **IP Status** displays the report results. Multiple runs cause additional reports to display in the IP Status tab (Figure 2-21).

Figure 2-21: **Report IP Status View in GUI**

When you bring in older projects that contain IP, the Vivado IDE prompts you to open the **IP Status Report** to review the status of the IP in the design.

The change log provides information about changes to the IP for each release.

To view the change log, either:

- Right-click the IP sources in the Report IP Status view, and select **IP Documentation > View Change Log**,
- On the Report IP Status view, click the **More Info** link in the **Change Log** column.

The pop-up menu has only the latest changes. Use the **More info** link to see all the information, as shown in Figure 2-22.

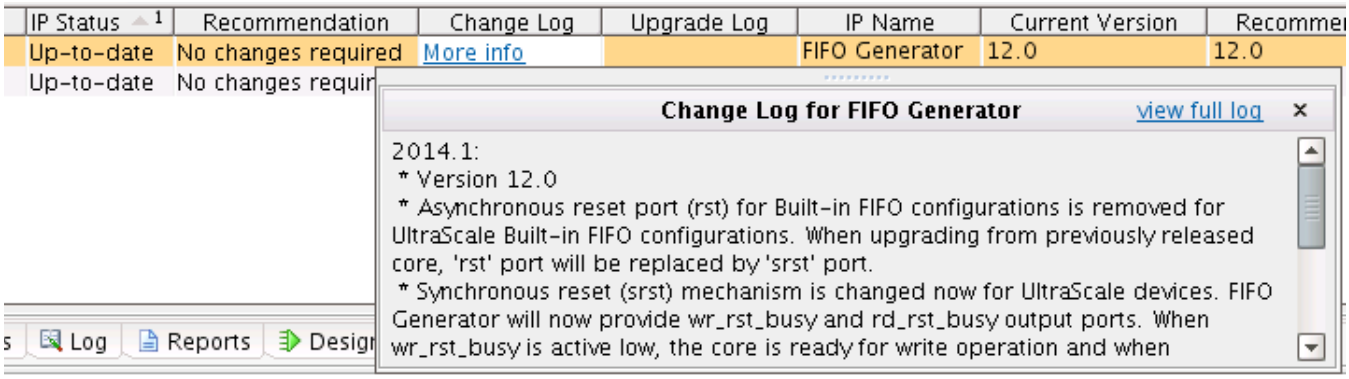


Figure 2-22: **Change Log from More info Link**

Reporting IP Status using a Tcl Command

You can also generate this information using the following Tcl command:

```
report_ip_status
```

Optionally, use the following command to create a text file:

```
-file <file_name>
```

Copying an IP

You can copy an existing IP customization to use as a starting point for a new IP. This is useful when you already have customized an IP and need only make small or simple customization changes and do not want to start from the default customization.

To copy an IP:

1. In the IP Sources tab, select the IP, right-click and select **Copy IP,** as shown in Figure 2-23, page 35.

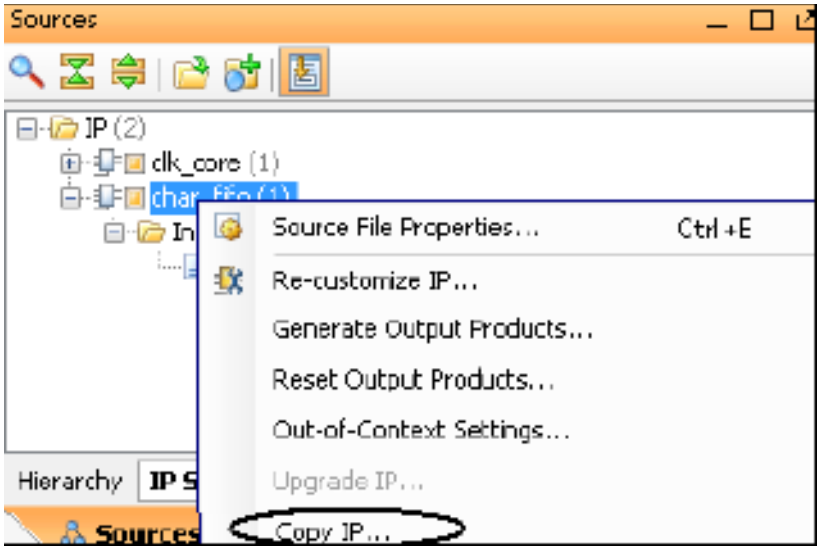


Figure 2-23: **Select IP to Copy**

2. Provide a destination name and location for the copy, as shown in Figure 2-24.

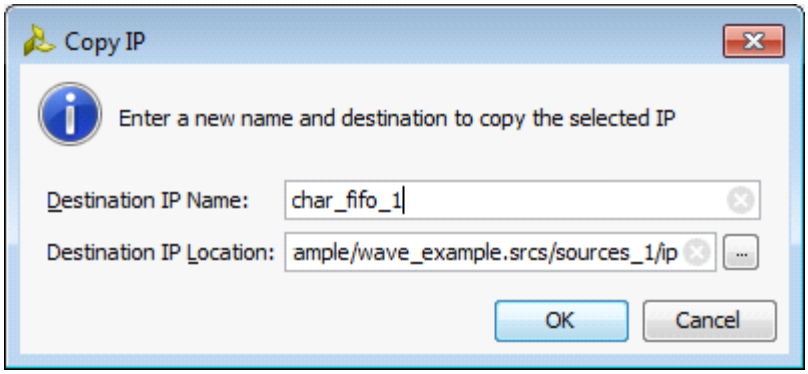


Figure 2-24: **Copy IP Dialog Box**

You can save the copied IP into a project directory structure, which by default is located at `<project_name>.src/sources_1/ip/`.

When working with a **Manage IP** project the default location is the same location as the `/mange_ip_project` directory.

You can now make changes to the copied IP customization by either double-clicking the IP, or right-clicking and selecting **Re-customize IP** from the IP Sources tab. The copied IP customization window opens with the customization settings from the original IP. You can now make edits.

CAUTION! *It is possible to have two versions of the same IP that reference different subcore versions; an older version that is locked and an another, newer version. In such a case, a synthesis tool could produce errors or logic bugs because files exist with the same module names but with different contents.*

Tcl Command Example for Copying IP

You can use the `copy_ip` command to create a copy of an IP customization: For example:

```
copy_ip -name newFIFo [get_ips char_fifo]
```

This command creates a copy of the `char_fifo` IP, names it `newFIFO`, and adds it to the project. Because no directory was specified using the `-dir` option, the IP is created inside the project directory structure.

Understanding IP States Within a Project

There are several states in which an IP can display within in a project, depending if it is the current version in the catalog and if you have generated output products.

IMPORTANT: *For imported IP cores with versions that are not accessible from the Vivado IP catalog, re-customizing, re-setting, and re-generating the IP is not enabled.*

When you add existing IP (either in the XCI or XCO form), if present, the output products (such as NGC and HDL files) are also added. Table 2-1 shows the buttons that represent the states of the Vivado IDE IP:

Table 2-1: **IP States in a Project**

Button	Description
	IP in an RTL project to be synthesized out-of-context (OOC).
	Customized IP which is in the IP catalog that is to be synthesized with the project (global synthesis).
	Unmanaged IP.

Table 2-1: **IP States in a Project *(Cont'd)***

Button	Description
	Locked IP Customization. You can use the IP, but it cannot be re-customized and new output products cannot be generated. For example, simulation targets, cannot be generated if they are missing. When IP has the `IS_LOCKED` property set to `TRUE`, it could be for various reasons: ◦ Because the IP is not current. ◦ The IP is locked avoid any further auto-generation from taking place.
	The IP cannot be used in its current state; otherwise you would encounter errors, because: ◦ The IP must be upgraded to the current version in the IP catalog. ◦ The IP is no longer in the IP catalog. You must bring in the original output products; otherwise you cannot use the IP. See Determining Why IP is Locked, page 37 for more information.

IMPORTANT: *When working with Xilinx-delivered patches, you might notice IP locking due to changes in the IP definitions from the patch.*

Determining Why IP is Locked

IP becomes locked for several reasons. The Vivado IDE provides an IP status report that provides the reason and a recommendation. Table 2-2, page 38 lists the locked IP messages and recommendations. See Viewing the IP Status Report, page 33.

Table 2-2: **IP Locked Reasons and Recommendations**

Brief Reason	Verbose Reason	Brief Recommendation	Verbose Recommendation
IP file read-only	IP <IPNAME> has a read-only file <IPXMLFILE>. IP is write protected.	Check file and project permissions	Review your project and file system permissions causing the IP to be read-only.
	IP <IPNAME> has a read-only file <IPXCIFILE> with restricted functionality. Commands to change the configuration of this IP are disallowed.		
Shared output directory	IP <IPNAME> shares a common output directory with other IP. Xilinx recommends that you place each IP in its own directory.	Move IP	• Manually remove the IP from the project with the remove_files command (see the *Vivado Design Suite Tcl Command Reference Guide* (UG835) [Ref 12]. • Import it back into the project with a unique directory import_IP command (see Tcl Command Examples to Import IP, page 30).
IP definition not found	IP definition <CURRENT_IPDEF> for IP <IPNAME> (customized with software release <SWVERSION>) was not found in the IP Catalog.	Add IP definition to catalog.	Consult the IP Catalog for the replacement IP.
IP major version change	IP definition <CURRENT_IPDEF> for IP <IPNAME> (customized with software release <SWVERSION>) has a newer major version in the IP Catalog.	Upgrade IP.	Target IP definition <TARGET_IPDEF> requires a major version change. Review the impact on the design before upgrading the IP
IP minor version change	IP definition <CURRENT_IPDEF> for IP <IPNAME> (customized with software release <SWVERSION>) has a newer minor version in the IP Catalog.		Target IP definition <TARGET_IPDEF> requires a minor version change. Review the change log before upgrading the IP.
IP revision change	IP definition <CURRENT_IPDEF> for IP <IP_NAME> (customized with software release <SW_VERSION>) has a different revision in the IP Catalog.		Target IP definition <TARGET_IPDEF> requires a revision change. Review the change log before upgrading the IP.
Incompatible IP data detected	The IP Data in the repository is not compatible with the current instance (despite having identical Version and Revision). This typically occurs if you are using IP that is currently under development. You cannot able to view the customization or generate outputs until it is updated.		Upgrade the IP.
IP unsupported part	IP <IPNAME> does not support the current project part <CURRENT_PART>. However part differences can result in undefined behavior.	Unsupported Upgrade IP.	Target IP definition <TARGET_IPDEF> does not support the current project part <CURRENT_PART>. Select a supported project part before upgrading the IP

Table 2-2: **IP Locked Reasons and Recommendations *(Cont'd)***

Brief Reason	Verbose Reason	Brief Recommendation	Verbose Recommendation
IP license not found	IP <IPNAME> requires one or more mandatory licenses but no valid licenses were found. However license checkpoints could prevent the use of this IP in some tool flows.	Unlicensed Upgrade IP	Target IP definition <TARGET_IPDEF> requires a valid license. Obtain a valid license before upgrading the IP.
		Check IP license	IP <IPNAME> requires a valid license. Obtain a valid license or review your licensing environment.
IP board change (1)	This IP has board specific outputs. Current project board <CURRENT_BOARD> and the board <ORIGINAL_BOARD> used to customize the IP <IPNAME> do not match.	Retarget IP	Change the project part or re-target this IP using the upgrade flow to the current project part or board.
IP part change	Current project part <CURRENT_PART> and the part <ORIGINAL_PART> used to customize the IP <IPNAME> do not match		Change the project part or re-target this IP using the upgrade flow to the current project part or board.
IP contains locked subcore	IP <IPNAME> contains one or more locked sucrose.	Upgrade parent IP	Upgrade the parent IP <PARENTNAME>

1. When an is locked due to a part or board change and is upgraded, you need to review the ports. Some IP have port differences based upon the part selected. For example, debug ports names and functions change when you update from 7 series to UltraScale for the QSGMII IP. You must make RTL changes to avoid encountering errors during synthesis and or implementation. See the appropriate IP product guide for more details.

Understanding IP Constraints

The Vivado IDE manages both user-defined XDC timing and physical constraints for the entire design, as well as for Xilinx IP. It handles the association and the unification of constraints for Xilinx IP instantiated multiple times within a project.

Most IP in the IP catalog deliver IP-specific XDC constraints based on user customization. The constraints delivered by the IP are optimized using the default synthesis settings.

RECOMMENDED: *Do not change these settings for any of the IP design runs because you could encounter issues with applying constraints. To take ownership of constraining an IP, disable the XDC file(s) that are delivered with an IP. If you must change the synthesis settings for an IP, select the IP run from the* ***Out-of-Context Module Runs*** *in the Design Runs tab, then make changes in the Options tab of the* ***Synthesis Run Properties****. You can also script this action using the Tcl command:* `set_property <synthesis_option> <value [get_runs <IP_Name>_synth_1].`

Tcl Command Example for Changing Synthesis Run Properties

```
set_property STEPS.SYNTH.DESIGN.ARGS.FSM_EXTRACTION sequential /
[get_runs my_IP_synth_1]
```

During design synthesis and implementation, the Vivado IDE processes the IP-delivered XDC constraints before processing the user-defined constraints, or after, depending on the constraint file.

Tcl Command Example to Report Constraint Compile Order

```
report_compile_order -constraints
```

This command provides the procession order of synthesis HDL files as well as constraints used for synthesis and implementation of user logic, and provides a breakdown of the constraints used for each IP synthesis run used for the generation of the IP DCP.

Ordering Multiple Constraint Files

Vivado IP can generate multiple XDC constraints files. By default, IP constraints are processed before user constraints because of the following possibilities:

- The IP might produce a clock that must be available to the end-user constraints.
- If the IP delivers physical constraints, the end-user can override them if necessary.

Some constraints that an IP delivers could have a dependency on a clock object that comes from either the end-user or another IP. These constraints are provided in a separate XDC file and are processed after the end-user constraints.

Typically, an IP delivers a core XDC file that can contain clock creation commands as well as commands without external clock dependencies. The customization file name is `<IP_Name>.xdc`, and is referred to as the *core* XDC file. By default, the Vivado IDE processes the core XDC files before any user constraints.

Some IP can also include another XDC file that contains clock-dependent commands. Because top-level clock can come from other constraints, or from other IP with a dependency, you must place any command that need those clocks to be defined first in the `<IP_Name>_clocks.xdc`. By default, the Vivado IDE processes the `<IP_Name>_clocks.xdc file` after user constraints and other IP core XDC files.

Most IP deliver an OOC XDC file as well, (`<IP_Name>_OOC.xdc`). This file contains default top-level definitions for input clocks to the IP. This file is only used in the DCP creation when using the recommended default flow (IP synthesized OOC to the top-level design). When the Vivado Design Suite synthesizes the IP OOC of the top-level design, clocks that are created by the end-user or other IP are not available; consequently, this file is necessary to provide the clock definitions for synthesizing the IP.

The `<IP_Name>_OOC.xdc` is not needed during implementation of the user logic with the IP, because all the netlists are linked together before constraints are applied. At that point a user-created clock or an IP-created clock is available to any IP that requires a clock.

Some IP support a board flow, see Appendix A, Using the IP Board Flow. When creating a project, if you select a development board during part selection, is available during the customization of the IP that lets you specify which connections on the board to which to connect the IP. This produces an `<IP_Name_board>.xdc` file which contains `PACKAGE_PIN`, `IOSTANDARD`, and other physical constraints.

Some IP can deliver additional XDC files. This might be because they deliver constraints that are to be used only during synthesis or only during implementation. For a list of possible XDC files that an IP can deliver, see Appendix B, IP Files and Directory Structure.

For more detailed information on XDC constraints, including specifics about XDC constraints for IP, see *Vivado Design Suite User Guide: Using Constraints* (UG903) [Ref 3].

After some constraints are processed for a project, those constraints can become project *Properties*. For more information regarding properties, see the *Vivado Design Suite Properties Reference Guide* (UG912) [Ref 20].

The following subsections briefly describe some of the constraint files that the Vivado IDE creates when processing IP.

VIDEO: *See the Vivado Design Suite QuickTake Video: IP Constraints Overview, for a demonstration of how the following constraints are used during IP flow.*

dont_touch.xdc

The Vivado Design Suite uses the `dont_touch.xdc` to set `DONT_TOUCH` properties on the IP top-level during synthesis of the IP. This prevents interface ports from being removed.

You can see this constraint file being processed in the synthesis log file, either for the IP when it is synthesized using OOC (the default flow), or in the global synthesis log file when synthesizing the IP with the end-user RTL.

dont_buffer.xdc

The Vivado Design Suite uses the `dont_buffer.xdc` file when sythesizing the end-user logic when IP is present that uses the OOC (default) flow.

The IP is treated as a black box during top-level synthesis, and the `dont_buffer.xdc` file sets the `BUFFER_TYPE` to `NONE` for all the interface pins of the IP. Setting the property prevents I/O buffer instantiations in the RTL sources, and this property prevents an additional buffer from being placed at the top-level. Later, if an I/O buffer is necessary, any required I/O buffer is inserted.

in_context.xdc

During the creation of the IP DCP, if the IP output any clocks, an `<IP_Name>_in_context.xdc` file is created and stored in the IP DCP file.

When synthesizing the top-level, any IP that has had a generated DCP file has the `<IP_Name>_in_context.xdc` file processed before the end-user constraints.

The clock objects are created on the boundary pins of the IP black box cell. This file is not necessary during implementation because the netlist is linked and is no longer a black box.

Determining Clocking Constraints and Interpreting Clocking Messages

Vivado can contain hierarchical constraints, top-level user constraints, and constraints that are delivered by an IP. These constraints can have dependencies which must be met to work correctly. One such constraint is clock creation.

- Some IP create clocks that other IP or the end-user top-level need.
- Some IP require a clock to exist to function correctly and not produce critical warnings.

If the necessary clock constraint is not being provided, then the IP at the top-level of the design issues a CRITICAL WARNING as described in Examples of Critical Warnings and Warnings on Clocking, page 44.

Tcl Example of an IP Clock Dependency

The following is an example of a constraint that an IP could deliver that has a dependency on a top-level clock which enters the IP on the port `ref_clk`:

```
set_max_delay -from [get_cells data_reg] -to [get_cells synchro_stage0_reg]
-datapath_only [get_property PERIOD [get_clocks -of_objects [get_ports
ref_clk]]
```

As explained in the *Constraints Guide* (UG625) [Ref 3], the `get_ports` command is converted into a `get_pins` command on the IP cell instance. Depending upon how the IP is connected in a design, the clock could come either from a user-supplied clock or from another IP.

- In the case of another IP supplying the clock, the clock is provided and no critical warnings are produced.
- If the clock is provided in the end-user logic, a critical warning is produced if no clock object is created (using the `create_clock` or `create_generated_clock` command).

Tcl Command Examples for Clocking

One way to find clocks in your design that are not being properly generated is to use:

```
report_clock_networks
```

This command produces a clock module for the design, including constrained and unconstrained clocks. You can use the report to determine if the clock module connected to your IP is missing a clock definition.

Another useful command is:

```
report_clocks
```

This command returns a table showing all the clocks in a design, including propagated clocks, generated and auto-generated clocks, virtual clocks, and inverted clocks in the current synthesized or implemented design.

As well as the command:

```
report_compile_order -constraints
```

This command shows which XDC files the design is using for synthesis and implementation and in what order they are processed. If an IP XDC that is creating a clock comes after an IP XDC that needs the clock, this clarifies the relationship.

Often you can resolve issues of a missing clock coming from an IP by adding a constraint to your top-level XDC timing constraints file.

This could be the case when working with an XPS design where there are no XDC files present for some of IP that could be creating a clock, such as a GT.

For more information about working with the designs with clocking requirements, see the *Vivado Design Suite User Guide: Using Constraints* (UG903) [Ref 3]

Examples of Critical Warnings and Warnings on Clocking

The following are examples of the errors for a design that fails to find the clock constraint needed by a piece of IP.

```
CRITICAL WARNING: [Vivado 12-259] No clocks specified, please specify clocks
using -clock, -fall_clock, -rise_clock options [C:/Design/v_tc.xdc:1]INFO:
[Vivado 12-1399]
There are no top level ports directly connected to pins of cell
'system/v_tc', returning the pins matched for query '[get_ports s_axi_aclk]'
of cell 'system/v_tc'.
[C:/Design/v_tc.xdc:1]Resolution: The get_ports call is being converted to a
get_pins call as there is no direct connection to a top level port. This
could be due to the insertion of IO Buffers between the top level terminal
and cell pin. If the goal is to apply constraints that will migrate to top
level ports it is required that IO Buffers manually be instanced.

CRITICAL WARNING: [Vivado 12-1387] No valid object(s) found for
set_max_delay constraint with option 'from'.
[C:/Design/v_tc.xdc:1]Resolution:
Check if the specified object(s) exists in the current design. If it does,
ensure that the correct design hierarchy was specified for the object.

WARNING: [Vivado 12-584] No ports matched 's_axi_aclk'.
[C:/Design/v_tc.xdc:2]

WARNING: [Vivado 12-1008] No clocks found for command 'get_clocks -of
[get_ports s_axi_aclk]'. [C:/Design/v_tc.xdc:2]INFO: [Vivado 12-626] No
clocks found. Please use 'create_clock' or 'create_generated_clock' command
to create clocks. [C:/Design/v_tc.xdc:2]
```

Using Fee-Based Licensed IP

The Vivado IP catalog displays either **Included** or **Purchase** under the License column in the IP catalog. The following definitions apply to IP offered by Xilinx:

- **Included**: The Xilinx End User License Agreement applies to Xilinx LogiCORE™ IP cores that are licensed within the Xilinx Vivado Design Suite software tools at no additional charge.

- **Purchase**: The Core License Agreement applies to fee-based Xilinx LogiCORE IP, and the Core Evaluation License Agreement applies to the evaluation of fee-based Xilinx LogiCORE IP.

For more information on how to obtain IP licenses, see the Xilinx Licensing site at http://www.xilinx.com/ipcenter/ip_license/ip_licensing.htm

For fee-based IP, the **OK** button on the Customize IP dialog box is disabled until an evaluation or a paid license is found as shown in the Figure 2-25, page 45.

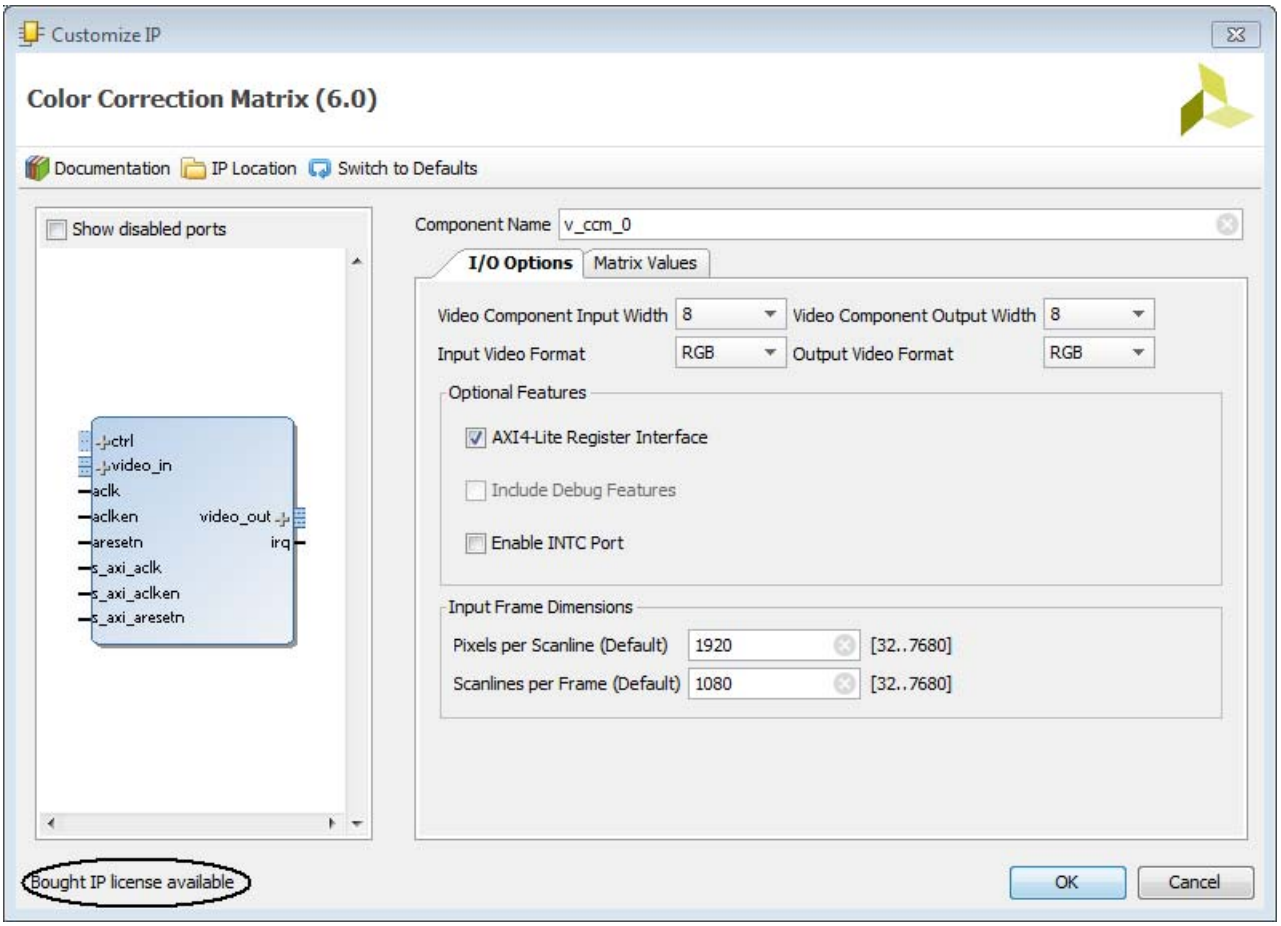


Figure 2-25: **Fee-Based IP "OK" Button Enabled after License Found**

Using Debug IP

The Vivado lab tools is the term that represents the debugging tools that are available in the Vivado Design Suite. The features that are included in the Vivado lab tools include:

- Vivado logic analyzer (see ILA, page 46)

- Vivado serial I/O analyzer (see VIO, page 46)
- IBERT serial analyzer (see IBERT, page 47)
- JTAG to AXI (see JTAG-to-AXI, page 47)

You can use the Vivado lab tools to test and verify the IP capabilities when attached to a Xilinx board with a JTAG connection.

See the following documents for more information:

- *Vivado Design Suite Tutorial: Programming and Debugging* (UG936) [Ref 21]
- *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 22]

Also, simulating customized IP lets you see if you are achieving the results you expect. See the following for more information about simulation options:

- Chapter 6, Simulating IP
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 13]

The following sections describe the IP that assist with debugging customized Vivado IP.

ILA

The integrated logic analyzer (ILA) also called *Vivado logic analyzer*, lets you perform in-system debugging of post-implemented designs on an FPGA. Use this feature when you need to monitor signals in a design.

You can also use this feature to trigger on hardware events and capture data at system speeds.

You can instantiate the ILA core in your RTL code or insert the core, post-synthesis, in the Vivado design flow. See the *LogiCORE IP Integrated Logic Analyzer Product Guide* (PG172) [Ref 25].

VIO

The virtual input/output (VIO) debug feature, also called the *Vivado serial I/O analyzer* can both monitor and drive internal FPGA signals in real time.

In the absence of physical access to the target hardware, you can use this debug feature to drive and monitor signals that are present on the real hardware.

IMPORTANT: *This debug core must be instantiated in the RTL code; consequently, you need to know what nets to drive.*

The IP catalog lists this core under the Debug category. See the *LogiCORE IP Virtual Input/Output Product Guide (PG 159)* [Ref 29].

IBERT

The integrated bit error ratio tester (IBERT) serial analyzer enables in-system serial I/O validation and debug. This allows you to measure and optimize your high-speed serial I/O links in your FPGA-based system.

RECOMMENDED: *Use the IBERT Serial Analyzer when you are interested in addressing a range of in-system debug and validation problems from simple clocking and connectivity issues to complex margin analysis and channel optimization issues. Use the Vivado IBERT serial analyzer design when you are interested in measuring the quality of a signal after a receiver equalization is applied to the received signal. This ensures that you are measuring at the optimal point in the TX-to-RX channel and thereby real and accurate data.*

You can access this design by selecting configuring, and generating the IBERT core from the IP catalog and selecting the **Open Example Design** feature of this core.

See details on the this IP in the following documents:

- *LogiCORE IP IBERT for 7 Series GTX Transceivers (PG132)* [Ref 26]
- *LogiCORE IP IBERT for 7 Series GTP Transceivers (PG133)* [Ref 27]
- *LogiCORE IP IBERT for 7 Series GTH Transceivers (PG152)* [Ref 28]

JTAG-to-AXI

- The JTAG-to-AXI debug feature generates AXI transactions that interact with various AXI4 nd AXI4-Lite slave cores in a system that is running in hardware.

IMPORTANT: *Use this core to generate AXI transactions and debug and to drive AXI signals internal to an FPGA at run time. You can use this core in IP designs without processors as well. The IP Catalog lists the core under the **Debug** category.*

Using Manage IP Projects

Introduction

The Vivado® Integrated Design Environment (IDE) provides mechanisms for:

- Exploring IP in the IP catalog
- Customizing IP
- Managing centralized location of customized IP.

The Vivado IDE can create a special project, referred to as a *Manage IP* project, which facilitates the management of IP customizations.

From this project type, you can view the IP catalog, customize IP, and generate output products. The IP customization (XCI) and generated output products are stored in separate directories located outside of the Manage IP project. The Manage IP project manages the IP design runs for the generation of the synthesized Design Checkpoint (DCP) files.

RECOMMENDED: *When working in teams, or if the design uses many Xilinx® IP, Xilinx recommends that you create and maintain your customized IP in a location outside of the Vivado project structure. This method makes revision control more straightforward and allows for ease of sharing customized IP with others. This is also the recommended methodology for working with IP in a non-project, script-based flow.*

Managed IP Features

When you use the Manage IP flow in the Vivado IDE, the following features are available:

- Simple IP project interface
- Direct access to the Xilinx IP catalog
- Ability to customize multiple IP
- Separate, unique directories for each IP instantiation with all related IP files
- Option to generate or skip generation of the Design CheckPoint (DCP) file. A DCP file consists of both a netlist and constraints for the IP, and creating this file is the default flow.

Using the Manage IP Flow

1. Invoke the Vivado IDE, and from the **Getting Started** page, select **Manage IP** (Figure 3-1).

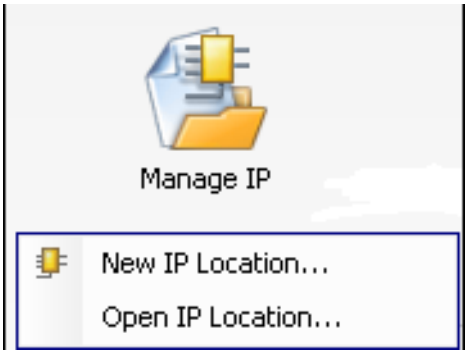


Figure 3-1: **Invoking the Manage IP Flow**

2. Open a new or an existing IP location.
   - **New IP Location**: Opens a new IP project at the location specified for exploring the IP catalog and customizing IP, including generation of output products.
   - **Open IP Location**: Lets you navigate to a location from which to open an IP.
   - **Recent Projects:** Lists recent locations from which to open an IP Project. If the specified location already contains an IP Project, then a window opens asking if you would like to open the existing project.

     When you select the **New IP Location** option, the **Create a New Customized IP Location** menu informs you that a wizard will guide you through creating and managing a new customized IP location, as shown in Figure 3-2, page 50.

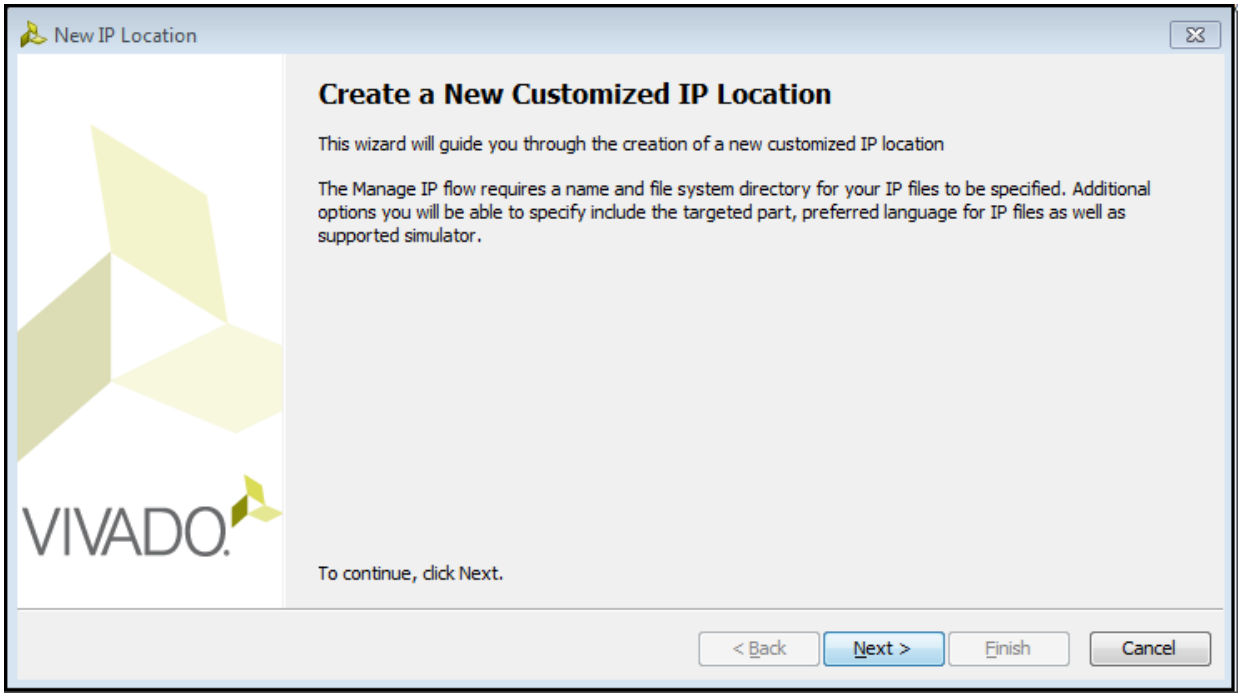


Figure 3-2: **Create a New Customized IP Location**

If the location specified to create a new manage IP project already contains a project, a dialog box opens (see Figure 3-3).

You can either choose to open the existing project or cancel to go back to the **New IP location** and specify a different location.

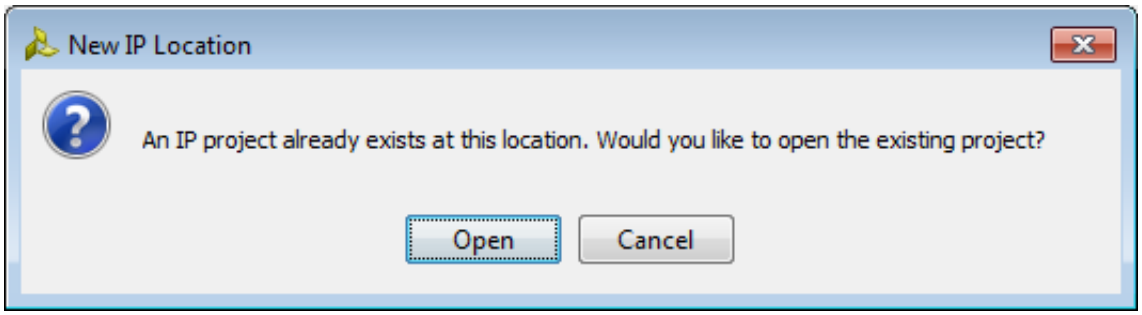


Figure 3-3: **New IP Location Dialog Box**

3. Select **Next**.

   The Manage IP Settings dialog box opens (Figure 3-4, page 51).

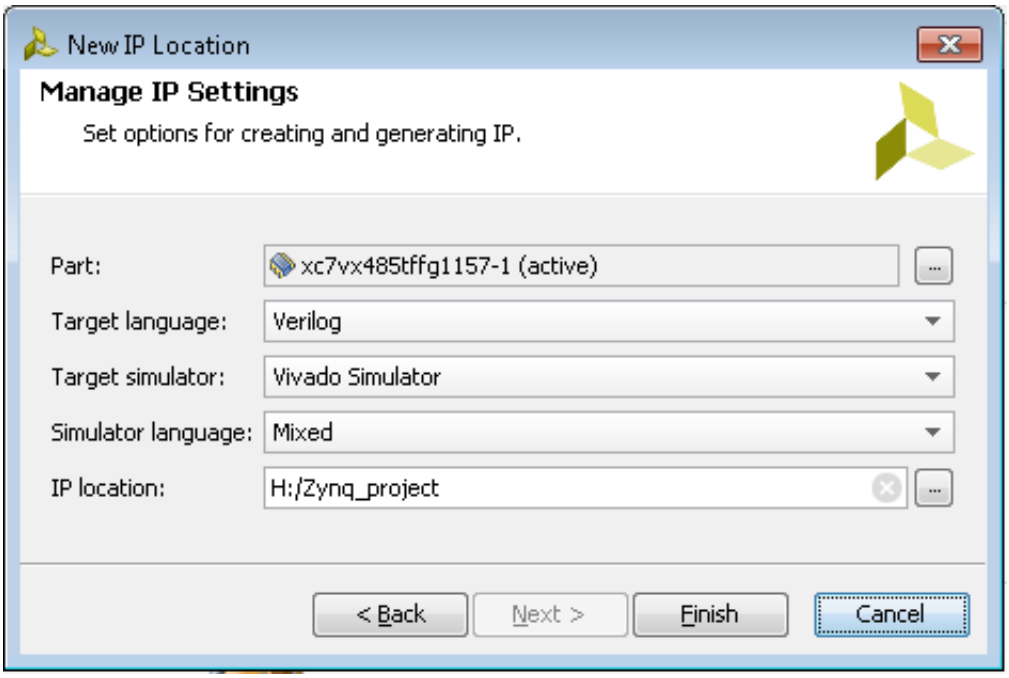


Figure 3-4: **Manage IP Settings Dialog Box**

Managing IP Settings

To manage the IP settings:

1. Enter the following:
   - **Part**: Select the active part.
   - **Target language:** Set the target language to VHDL or Verilog depending on your design top.
   - **Simulator language:** Options are **VHDL**, **Verilog**, or **Mixed** depending on the license that you have available for the simulator. For Vivado Simulator, the default is **Mixed**.
   - **IP location**: The location where the Vivado IDE creates the `managed_ip_project` directory.
2. Click **Finish**.

   The Manage IP window opens, and you can now select and customize IP.

You have access to the full IP catalog, including IP Product Guides, Change Logs, Product web pages, and Answer Records.

After you customize an IP, the Sources and Properties windows display, providing information about the IP created in the project.

Figure 3-5 shows the Manage IP Project window where you customize and manage multiple IP.

Each IP customization has a directory created under the specified manage IP location. This directory contains the XCI file and any generated output products.

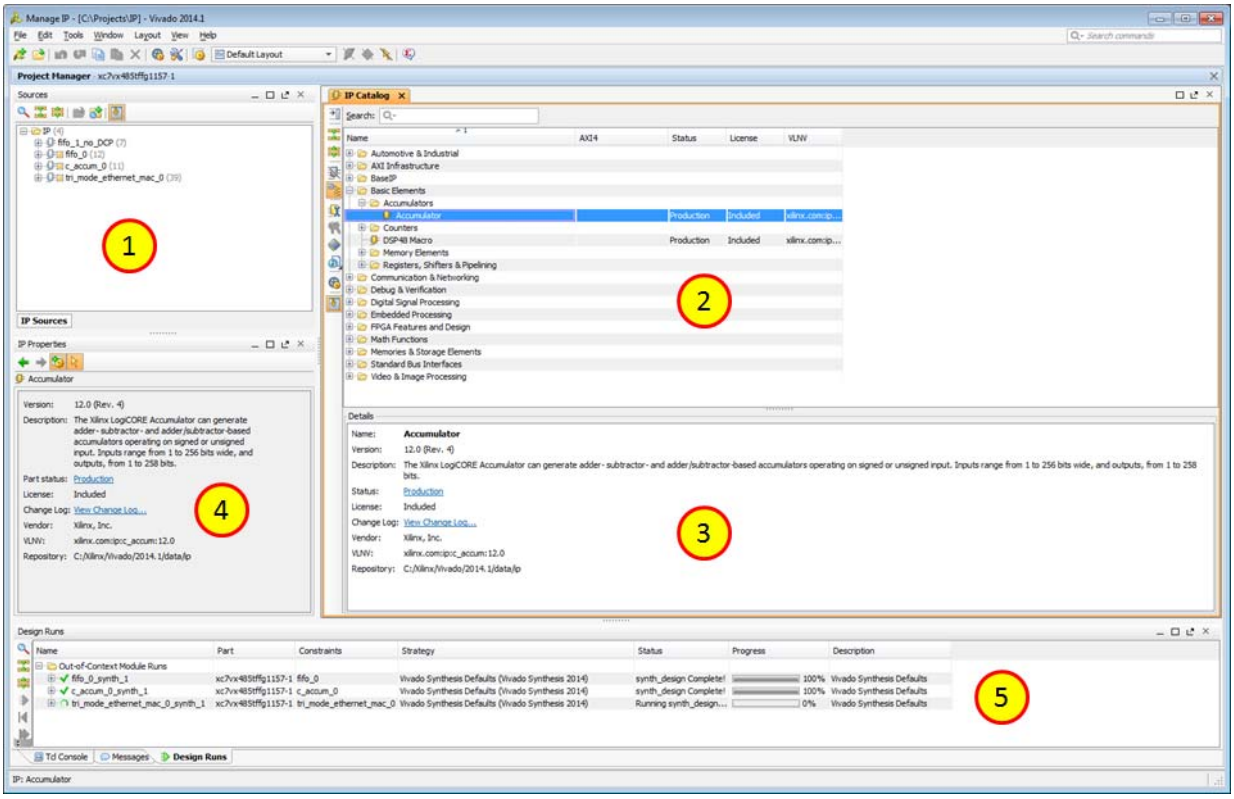


Figure 3-5: **Manage IP Project Window Containing Three IP**

The main sections to an IP Project window are as follows:

1. **IP Sources**: Lists the IP that are customized for the project, where you can view the output products and manage the generation of additional output products.
2. **IP Catalog**: Lets you explore the IP catalog and create customized IP to add to the IP Project.
3. **Details**: Displays the details of the selected IP.
4. **IP Properties**: Displays the properties and general detail information for the selected IP.
5. **Design Runs**: If you generate a synthesis design checkpoint (DCP) output product, a run for the IP shows in this view.

See Appendix B, IP Files and Directory Structure for a list of possible files and directories that the Vivado IDE can create for IP, depending upon the requirements.

Using IP Example Designs

Introduction

Many Xilinx® IP deliver an example design project. The example design project consists of top-level logic and constraints that interact with the created IP customization. These example designs typically come with an example test bench that helps simulate the design.

Opening an Example Design

To open an example design project for an IP, select the IP customization in the IP Sources tab, then right-click and select **Open IP Example Design** from the context menu.

This opens the **Open IP Example Design** window (Figure 4-1) for you to specify the location. The project is called `<IP_Name>_0_example`.

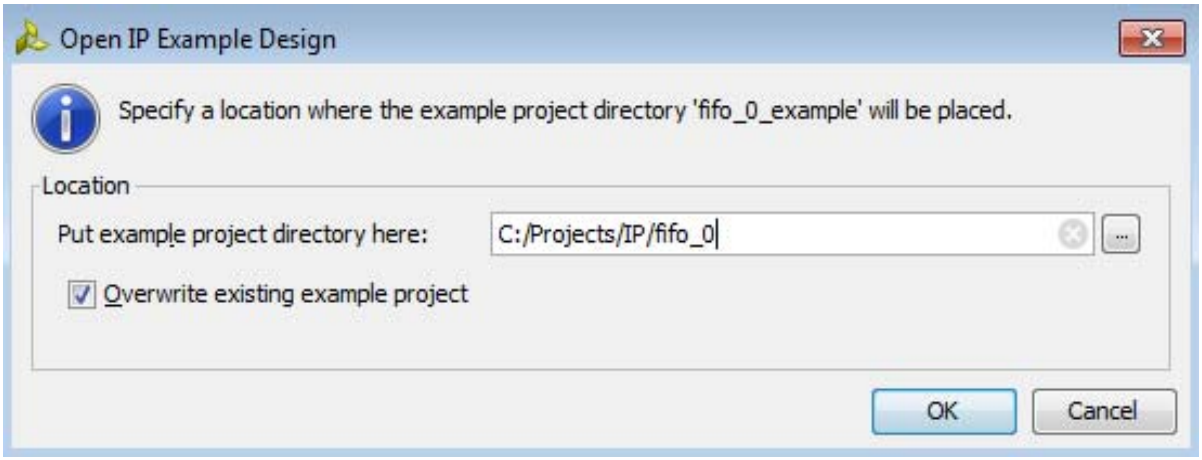


Figure 4-1: **Open IP Example Design Dialog Box**

A new session of the Vivado IDE opens that shows the example design in the Design Sources window (Figure 4-2, page 54).

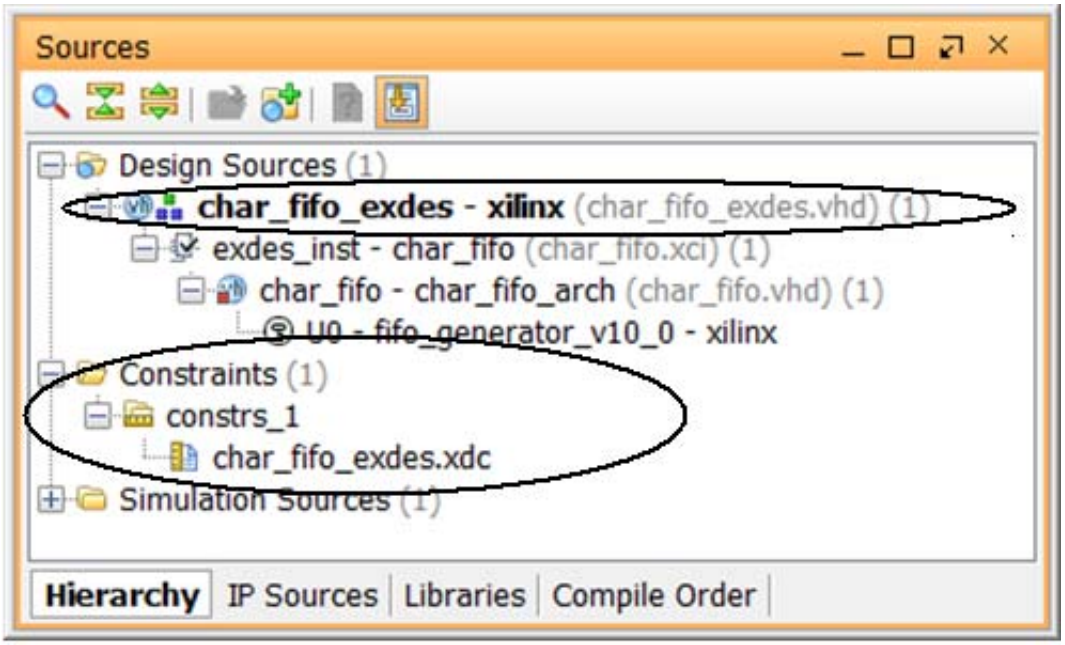


Figure 4-2: **IP Example Design Instance with Constraint File**

The IP is instantiated in the example design with an example XDC constraint file to enable further evaluation of the IP.

Tcl Command to Open a Project

Alternatively, you can use the following Tcl command to open a project:

```
open_example_project [get_ips <IP_Name>]
```

You can use the Tcl help to find other available options.

Chapter 5

Using Xilinx IP with Third-Party Synthesis Tools

Introduction

Xilinx® supports netlists created by third-party synthesis tools for user logic. When using Xilinx IP the only supported synthesis tool is the Vivado® synthesis tool. During logic synthesis, a black box is inferred for Xilinx IP. A file is provided to infer the black box as described in the following sections. The Vivado IDE resolves the black boxes during implementation.

Synthesis of Xilinx IP is supported only with the Vivado synthesis tool, including the IP core and any example design files an IP might deliver.

IMPORTANT: *Xilinx encrypts IP HDL files with the IEEE Recommended Practice for Encryption and Management of Electronic Design Intellectual Property (IP) (IEEE Std P1735) encryption: consequently those files are readable only when using the Vivado IDE for synthesis. You can use a third-party synthesis tool for the end-user logic and generate a netlist that Vivado implementation can use. Support is enabled for third-party simulation tools to perform behavioral simulations using the encrypted RTL.*

Synthesis Flow

When using a Synopsys® Synplify Pro or a Mentor® Graphics Precision netlist for synthesis with a design that has Xilinx IP, the recommended flow is:

1. Use the **Manage IP** flow in to create and customize IP.
2. Generate all the output products for the IP, including the synthesis design checkpoint (DCP) for each IP.

   When you generate the DCP file, two additional files are created to infer the black box when used with the third-party synthesis tool: `<IP_Name>_stub.v` and `<IP_Name>_sub.vhdl`.

3. Add the Verilog stub file or the VHDL stub file to the project with the third-party synthesis tool.

   The Verilog or VHDL stub file infers a black box for the Xilinx IP, and prevents the synthesis tool from adding I/O buffers. The `<IP_Name>_stub.v`, and the `<IP_Name>_stub.vhdl` contain synthesis directives that instruct the third-party synthesis tool to not infer I/O buffers for the IP if the IP connects to top-level ports. Change these directives as required for the third-party synthesis tool being used.

4. Generate a netlist with the third-party synthesis tool.
5. Create a Vivado netlist project. See *Vivado Design Suite User Guide: System-Level Design Entry* (UG895) [Ref 7] for more information about netlist projects.
6. Add the following into the Vivado netlist project:
   - The netlist from the third-party synthesis tool
   - User-level top-level constraints
   - The XCI files for the IP (one XCI file per IP)

   The netlist in the IP DCP as well as the XDC output products are used automatically during implementation when using the XCI file for the IP.

   When you use the **Add Sources** button to add design sources, select **Add Files** and change the **Files of type** to **All Files** so that the XCI files are listed.

7. Implement the design in the Vivado IDE.

Vivado implementation adds any required I/O buffers if they are not already present in the DCP of the IP.

RECOMMENDED: *Xilinx recommends that you use the IP XCI file when referencing Xilinx IP in either a Project Mode or Non-Project Mode and not the DCP file directly. While the DCP does contain constraints, they are resolved Out-Of-Context of the end-user constraints. Using the XCI results in the XDC output product for the IP being applied after all the netlists are combined (end-user and IP). Additionally, any Tcl script in the IP XDC is then evaluated in context of the end-user constraints and netlist.*

Using a Third-party Synthesis as a Black Box Flow in Project Mode

In Project Mode, to synthesize top-level logic separately with a third-party synthesis tool, create an EDIF, then include that EDIF into Vivado, along with the IP-level DCP files, and run implementation.

Example Tcl Script for Project Mode

```
# Create a project on disk
create_project <name> -part <part>

# configure as a netlist project
set_property design_mode "GateLvl" [current_fileset]

# Add in the netlist from third-party synthesis tool
add_files top.edif

# Add in XCI files for the IP
add_files {ip1.xci ip2.xci ip3.xci}

# Add in top level constraints: this might include XDC files from the third-party
# synthesis tool
add_files top.xdc
# Launch implementation
launch_runs impl_1 -to write_bitstream
```

Using a Black Box Flow in Non Project Mode

To synthesize top-level logic separately with a third-party synthesis tool, create an EDIF, then include that EDIF back into Vivado, along with the IP-level DCP files, and run implementation.

Example Tcl Script for Non Project Mode

```
# Create an in memory project and set part
create_project -in_memory -part <part>

# Read the netlist from third-party synthesis tool
read_edif top.edif

# Read in the IP XCIs
read_ip ip1.xci
read_ip ip2.xci

# read in top level constraints
read_xdc top.xdc
# Implement the design
link_design -top <top>
opt_design
place_design
phys_opt_design
route_design
write_bitstream -file <name>
```

Simulating IP

Introduction

Design simulation is an important and necessary step in the design flow to verify the functionality and performance of the design. To enable this, IP in the Vivado® IP catalog delivers a simulation model that you can include in the simulation of the overall design.

The simulation model delivered for the IP can be one of the following:

- Custom behavioral simulation model
- Plain text or encrypted synthesizable RTL sources used for simulation
- Structural simulation model

Note: Some IP (for example, the FIR Compiler IP) deliver IP-level test benches that you can directly use to simulate the IP.

IMPORTANT: *Vivado provides a Tcl command, `export_simulation`, that lets you write out a complete behavioral/RTL simulation script for use in standalone mode or integrated into your custom simulation environment.*

For details on how to use the `export_simulation` command, see the section "Using the `export_simulation` Command" in the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 4].

Delivering IP Simulation Models

Most Xilinx® IP deliver RTL sources for a single language only, effectively disabling simulation for "language locked" simulators if you do not have licensing for the appropriate language.

To simulate your design and include IP, Vivado ensures the availability of an appropriate simulation model for the IP using the project property **Simulator language** setting. The values are **Verilog**, **VHDL**, and **Mixed**. Set this property in one of available flows: **Manage IP,** Project-based, and Non-project based flows.

Some IP deliver simulation files for VHDL and some for Verilog. When the simulator language is set to **Mixed** it is possible that the same module for both languages can be sent to the simulator by different IP.

The Vivado simulator is a mixed language simulator and can handle simulation models in both VHDL and Verilog. If you are using other simulators and have license for a single language only, change the Simulator language to match your license.

If the IP does not deliver a behavioral model or does not match the chosen and licensed simulator language, the Vivado tools automatically generate a structural simulation model (`_funcsim.v` or `_funcsim.vhdl`) to enable simulation.

RECOMMENDED: *When you generate IP output products, enable the synthesized Design Checkpoint (.dcp) option to ensure that the Vivado IDE can deliver a structural simulation netlist for the IP. For more information, see Re-Customizing Existing IP in Chapter 2.*

Tcl Command to Set Simulator Language

To set the simulator language property use:

```
set_property simulator_language <language_option> [current_project]
```

Table 6-1 illustrates how the `simulator_language` property controls the delivery of the IP simulation model.

Table 6-1: **Simulator Language Property**

Simulation Model	Language	Simulation Model
IP delivers VHDL and Verilog behavioral models	Mixed	Behavioral model (target_language)
	Verilog	Verilog behavioral model
	VHDL	VHDL behavioral model

Table 6-1: **Simulator Language Property *(Cont'd)***

Simulation Model	Language	Simulation Model
IP delivers Verilog behavioral model only	Mixed	Verilog behavioral model
	Verilog	Verilog behavioral model
	VHDL	VHDL simulation netlist generated from the IP DCP
IP delivers VHDL behavioral model only	Mixed	VHDL behavioral model
	Verilog	Verilog simulation netlist generated from the IP DCP
	VHDL	VHDL behavioral model
IP deliver no behavioral models	Mixed/Verilog/VHDL	Netlist generated from DCP (target_language)

Note: Where available, a "Behavioral Simulation" always takes precedence over a "Structural Simulation". Vivado does not offer a choice of simulation model.

Note: The setting for the project property `Target_language` is used to deliver simulation models when the IP supports both; either language could be used for simulation, as shown in the following example:

Tcl Command to Set Properties on the Target Language

```
set_property target_language <language_option> [current_project]
```

Simulating IP with Vivado Simulator

The Vivado simulator is integrated in the Vivado IDE when you are in Project Mode. Figure 6-1 shows the Vivado simulator settings.

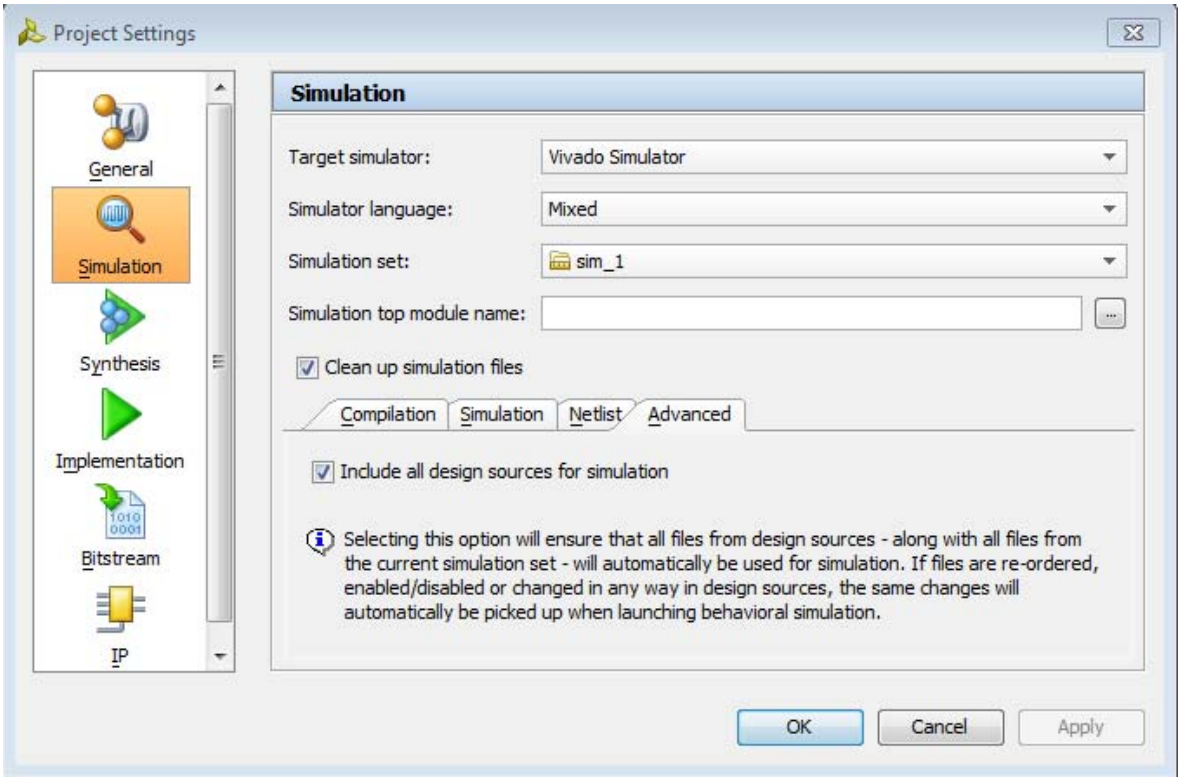


Figure 6-1: **Simulation Project Settings**

To simulate your design instantiating Xilinx IP:

1. In **Project Settings > Simulation**, set the **Target simulator** to **Vivado simulator**.
2. From the **Flow Navigator**, select **Run Behavioral Simulation** to start the simulation.

Alternatively, you can run simulations outside of Vivado simulation environment, using the `xsim` command.

Tcl Command to Launch Vivado Simulation

You can produce a run script using the `launch_xsim` command:

```
launch_xsim -scripts_only -mode behavioral
```

This produces a script for use with the `xsim` executable from the command line with the required file and library information.

Simulating IP with QuestaSim/ModelSim Simulator

Support for QuestaSim/ModelSim is integrated in the Vivado IDE when you are using the Project Mode.

To simulate your design instantiating Xilinx IP:

1. In **Project Settings> Simulation**, set the **Target simulator** to **QuestaSim/ModelSim**. as shown in Figure 6-2.

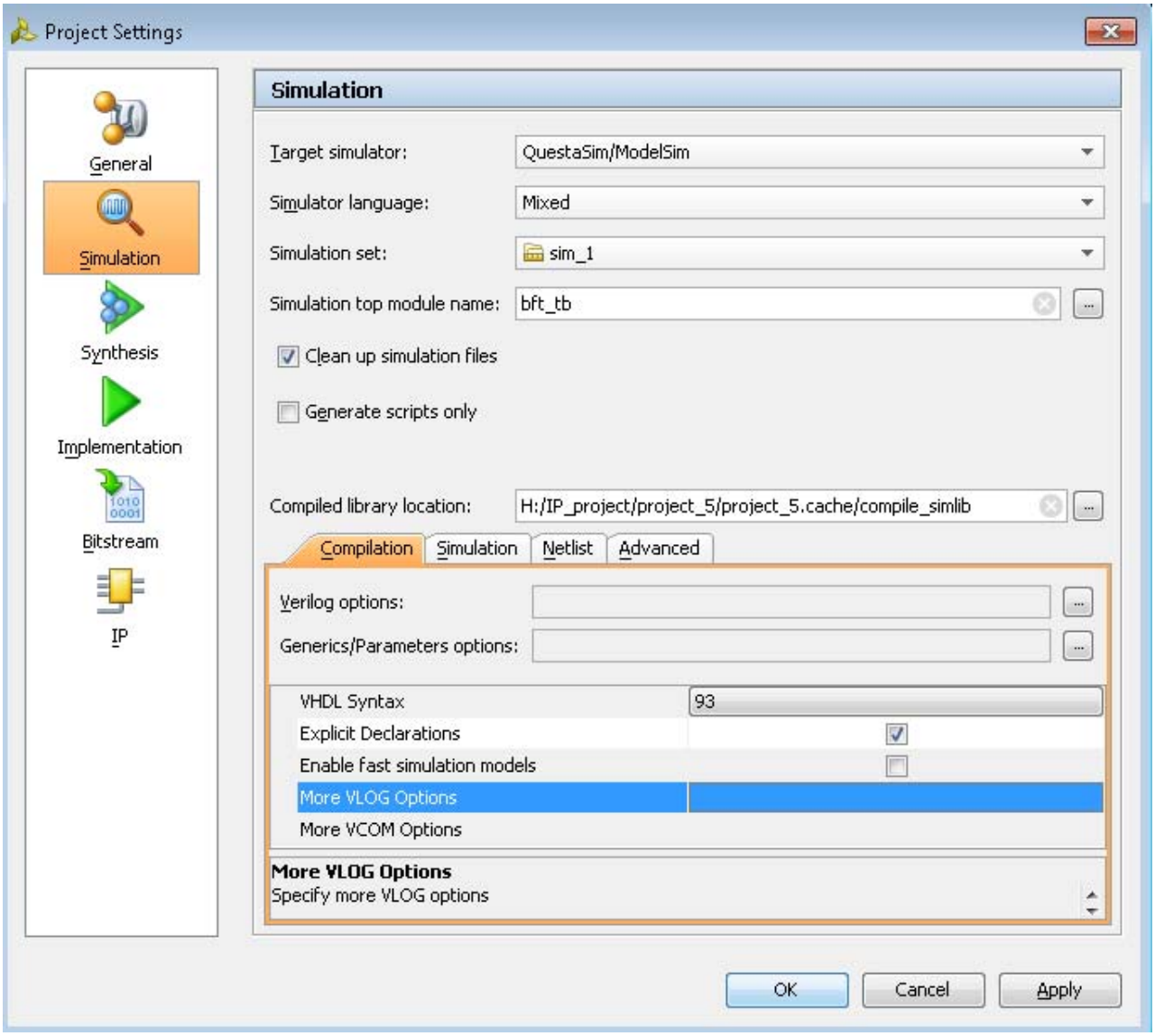


Figure 6-2: **QuestaSim/ModelSim Settings**

When you select QuestaSim/ModelSim, the following dialog box opens, as shown in Figure 6-3.

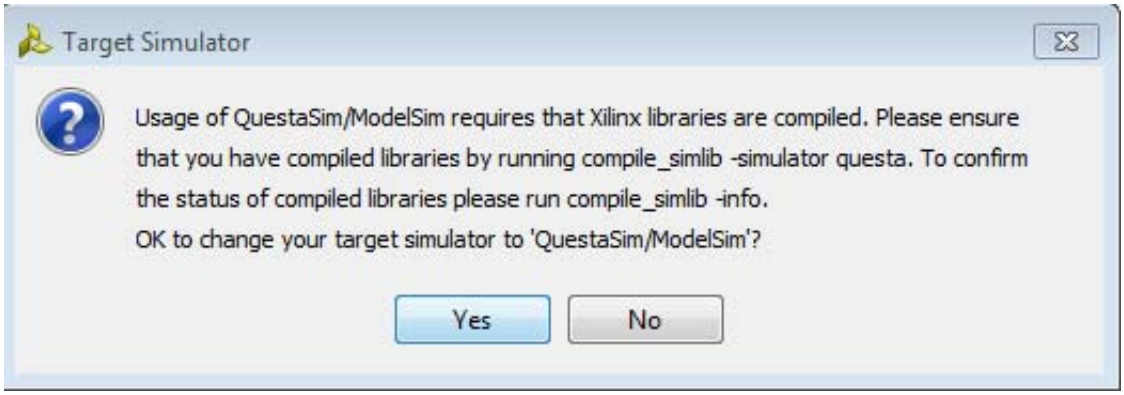


Figure 6-3: **Target Simulator Library Compilation Dialog Box**

2. Choose **Yes** and then ensure that the following actions are completed:
   a. Compile simulation libraries for QuestaSim/ModelSim.
   b. Specify the path in the **Compiled Library** Location.
3. Select **Flow Navigator > Run Behavioral Simulation** to start the simulation.

Tcl Command to Launch QuestaSim/ModelSim

To launch the Mentor Graphics ModelSim or Questa Advanced Simulator tool, the specified simulator must be installed, and appear in your `$PATH` to be properly invoked when launching simulation.

To launch ModelSim or QuestaSim, you must first set the target simulator property for the project:

```
set_property -target_simulator ModelSim [current_project]
```

Tcl Command to Compile Simulation Libraries

To support the use of ModelSim/QuestaSim you must compile the Xilinx simulation libraries for use with the target simulator:

```
compile_simlib
```

After the libraries are compiled, the simulator references these compiled libraries using the `modelsim.ini` file. The `modelsim.ini` file is the default initialization file and contains control variables that specify reference library paths, optimization, compiler and simulator settings. The `modelsim.ini` is located as follows:

- The path specified by the `-directory` argument at the time `compile_simlib` is run.
- The path defined by `$MODELSIM` environment variable.

- The path defined by `$MGC_WD` environment variable.
- The simulation run directory of the project.

Note: If the `modelsim.ini` file is not found at any of these locations a warning message is returned by the simulator. The command returns the transcript of the simulator.

For more details on using QuestaSim/ModelSim, see *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 4].

Tcl Script to Launch QuestaSim/ModelSim

Alternatively, to run simulations outside of Vivado with ModelSim/Questa you can produce a run script using the `launch_modelsim` command:

```
launch_modelsim –scripts_only –mode behavioral
```

This produces a script for use with ModelSim/QuestaSim with the required file and library information.

Tcl Commands for IP Simulation

If you want to manage your own scripts for simulating IP, you can use the `get_files` command to get all the files that an IP delivers for simulation.

```
get_files -compile_order sources -used_in simulation \
-of_objects [get_files <IP name>.xci]
```

Where:

- `compile_order` supports RTL sources only.
- `used_in` lets you specify files used in Vivado simulation or synthesis.
- `of_objects` takes the IP XCI file object from which all the files related to the IP can then be filtered.

For VHDL, you must specify the library associated with each file also. When you use a Tcl script, you can determine this information quickly.

The following script prints out each simulation file, and its file path, as well as the associated library:

```
# Get the list of files required for simulation
set ip_files [get_files -compile_order sources -used_in simulation \
-of_objects [get_files <IP_Name>.xci]
# For each of these files, get the library information
foreach file $ip_files {
puts "[get_property LIBRARY $file] $file"
}
```

The `LIBRARY` specifies the default library for the project. All files without an explicit library specification are compiled in the `xil_defaultlib`, which is created by the Vivado IDE. You can select a library name, or specify a new library name by typing in the **Library** text field.

Using Cadence Incisive Enterprise Simulator and Synopsys VCS MX Simulator

Xilinx encrypts the Vivado IDE IP using industry standard IEEE P1735 encryption; you can use simulators that support this encryption to do behavioral simulation. Behavioral simulation requires a list of simulation files and the libraries to which they belong.

The Vivado IDE can generate scripts for Cadence® Incisive Enterprise Simulator, (ies) and the Synopsys® VCS MX simulator (`vcs_mx`). The generated scripts contains simulator commands for compiling, elaborating, and simulating the design. The script commands retrieve the simulation compile order of specified objects, and export this information in a text file with the compiler commands and default options for the target simulator.

The specified object can be either a simulation file set or an IP.

If the object is not specified, then the `export_simulation` command generates the script for the simulation top. Any verilog `` `include `` directories or file paths for the files containing verilog define statements are added to the compiler command line.

Working with Revision Control

Introduction

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

For IP, there are strategies to consider, and each has a trade-off on either run time or the number of files that must be managed. This chapter discusses these strategies in detail.

For more information on how to use Vivado Design Suite with version control systems, see *Using Vivado Design Suite with Version Control Systems*, (XAPP1165) [Ref 18].

Using Revision Control

When working with revision control systems and Xilinx IP there are a few possible options:

- Full Revision Control
- Partial Revision Control

Full Revision Control

Place the entire IP directory into revision control, including all output products as well as the Synthesized Design Checkpoint (DCP).

RECOMMENDED: *This is the recommended option because it gives the flexibility to decide when and if to upgrade the IP at a future point. It also provides better run time as the IP does not have to be regenerated every time it is used.*

The Vivado IDE only supports one version of an IP in the IP catalog. If you upgrade the tool and the IP is no longer current, it is still usable. The IP is locked and you are not able to recustomize or generate output products; however, if all the output products are present, the IP can be used.

Partial Revision Control

Though it is not recommended, you can put a subset of the files associated with an IP into revision control. The following are some options and their limitations:

- Only the IP XCI file. This is the absolute minimum of what could be placed in revision control. If the IP is current, from the XCI all required output products can be generated, including the DCP. If the IP is not current your only option is to upgrade.
- The IP XCI file plus a subset of the output products which could include a structural simulation model and, or, the synthesis Design Checkpoint (DCP). This way the customization of the IP is retained and IP can be upgraded to the latest version if necessary. If you do not want to upgrade the IP, you can continue to use the DCP. The structural simulation model is used for simulating the IP.

One limitation is that the XCI file does not track the output product generation; this is done by the BOM XML file. You must use the DCP or any other output products directly, and not through the XCI. The XCI would be used only to track the IP upgrade status.

Another limitation is that the if constraints in the DCP are applied standalone, any Tcl queries would be resolved out-of-context of the entire design. When the IP is fully generated, only the netlist is used from the DCP; the constraints are applied from the XDC output product files.

Tcl Script to Get All IP Files

To get a list of all the output products generated for an IP, use the following Tcl command:

```
get_files -all –of_objects [get_ips <IP_Name>]
```

This commands lists all files generated for the IP including:

- BOM XML synthesis and simulation targets
- Created DCPs
- Third-party synthesis stubs
- Structural simulation models

Tcl Script for Synthesis-Only Files

To get a list of the files used for synthesis only, use the following Tcl command:

```
get_files -used_in synthesis -compile_order sources –of_objects [get_ips <IP_Name>]
```

IMPORTANT: *Without the BOM XML file the* `-used_in *` *commands are not associated with the IP XCI.*

Tcl Script for Simulation-Only Files

To get a list of the files used for simulation you can use the following Tcl command:

```
get_files -used_in simulation -compile_order sources -of_objects [get_ips <IP_Name>]
```

Tcl Command to Archive a Project

Note: To create an archive of the entire project, including any IP, use the following command:

```
archive_project
```

This command consolidates all the project files, bringing external ones into the archive project to ensure a complete archival. The original project is not modified.

Creating and Packaging IP

Introduction

The Vivado® Design Suite IP packager lets you use the Vivado Integrated Design Environment (IDE) to prepare IP designs to add to the Vivado IP Catalog. End-users can then instantiate this custom IP into their designs when using Vivado Design Suite tools.

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

Figure 8-1 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.*

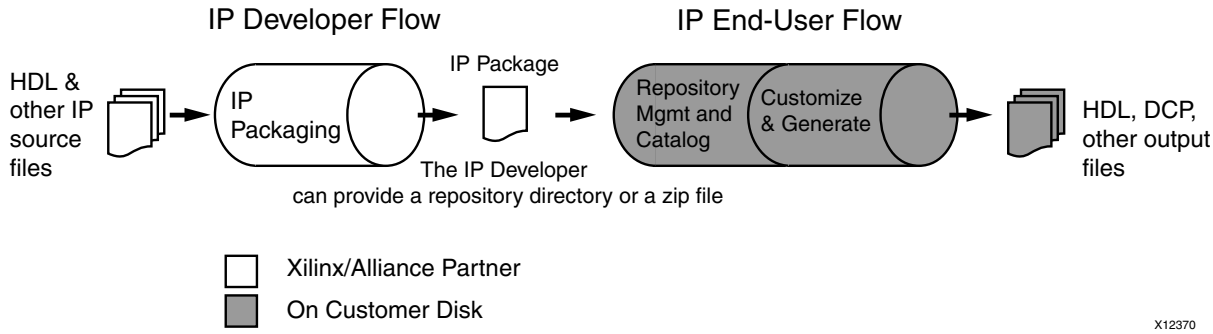


Figure 8-1: **IP Packaging and Usage Flow**

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 outputs files organized in sub-folders identical to the organization of the input directory. Files are not re-organized to match the logical groupings specified in the IP File Groups page.

In addition to the IP-XACT component files, the IP packager delivers a `/XGUI` folder. This folder contains the files for displaying the IP customization GUI in the Vivado IDE.

Using the Create and Package IP Wizard

The Vivado 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.

Appendix D, AXI Naming Conventions lists the naming conventions required for AXI interfaces. For more information on the Xilinx adoption of AXI, see the Vivado *AXI Reference Guide* (UG1037) [Ref 19].

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 Figure 8-2, page 71.

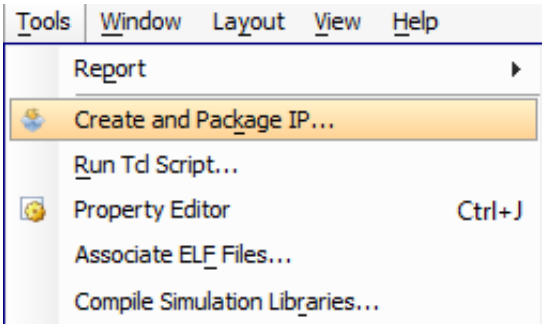


Figure 8-2: **Create and Package IP Option**

The first page of the Create And Package IP wizard opens (Figure 8-3).

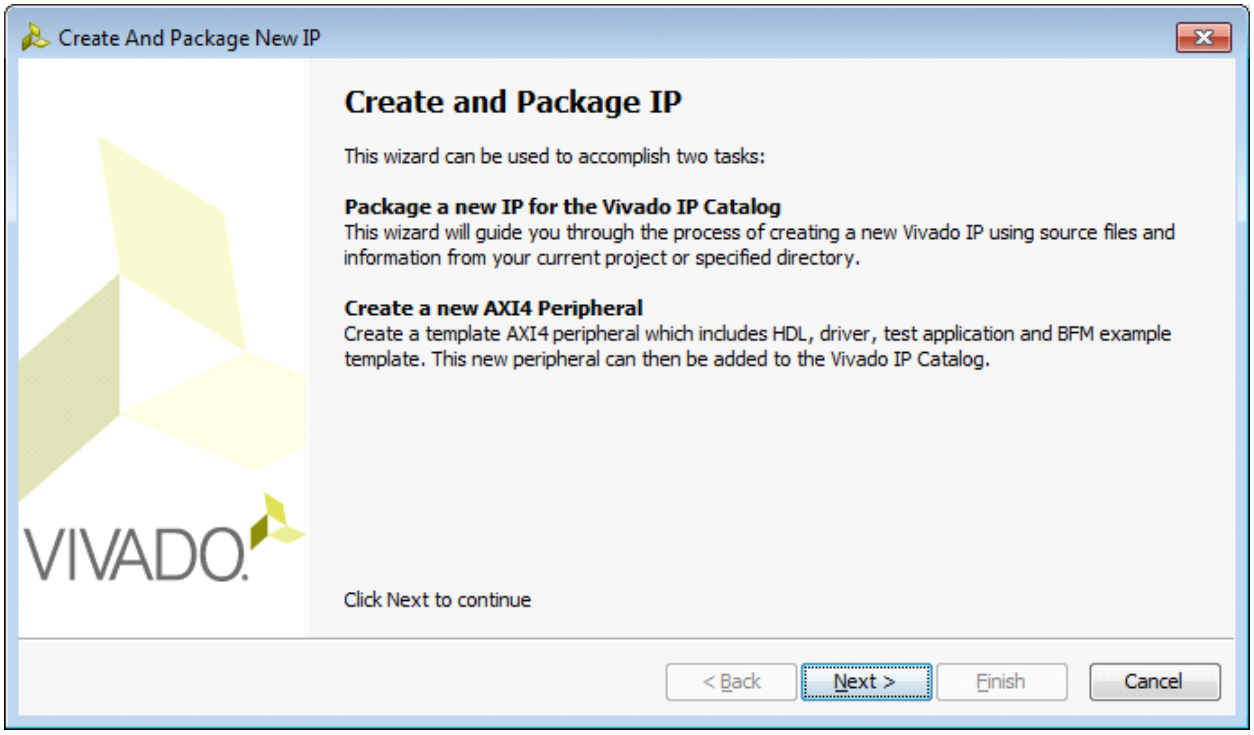


Figure 8-3: **Create and Package IP Dialog Box**

2. The wizard can be used 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.*

3. Click **Next**.

The Choose Create Peripheral or Package IP dialog box opens, (Figure 8-4).

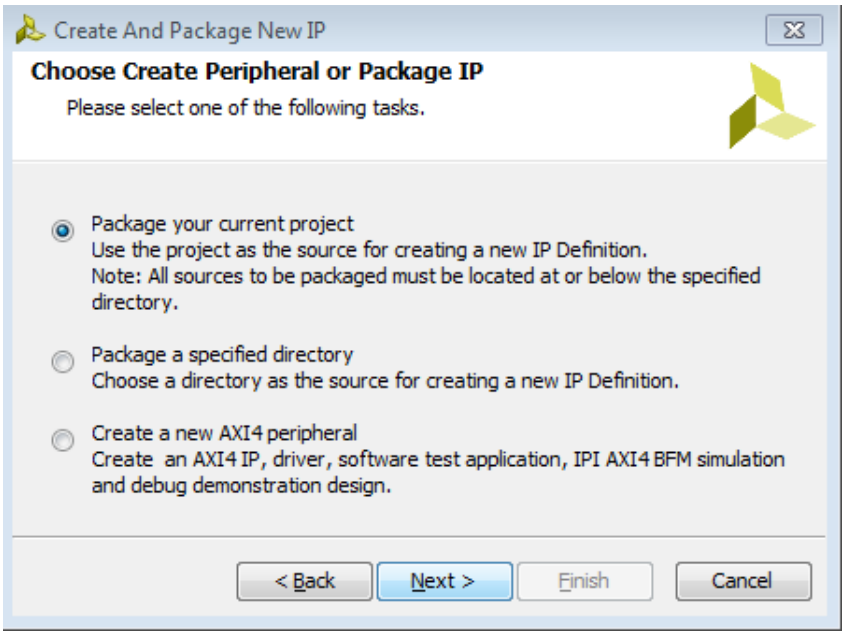


Figure 8-4: **Choose Create or Package IP Dialog Box**

4. Select from the options:
   - **Package your current project:** See Packaging Your Current Project for the option description.
   - **Package a specified directory:** See Packaging a Specified Directory, page 76 for the option description.
   - **Create a new AXI4 peripheral:** See Creating a New AXI4 Peripheral, page 78 for more information.
5. Make your selection, and click **Next**.

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

Packaging Your Current Project

Figure 8-5 shows the IP Location dialog box that displays after you select **Package your current project**.

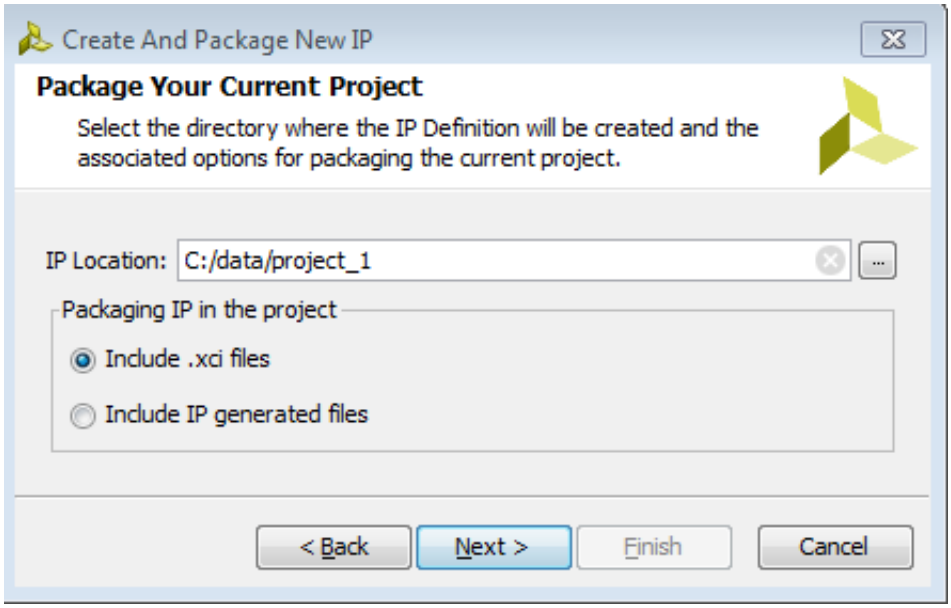


Figure 8-5: **Create and Package New IP: Package Your Current Project**

1. Provide the following settings:
   - **IP Location**: The directory in which the tool creates the IP Definition. The default is the project sources directory.
   - **Packaging IP in the project:**
     - **Include `.xci` files**: If the project you are packaging includes subcores, package only the IP customization XCI file. The tool generates the IP output products with the output products of the parent IP customization.
     - **Include IP generated files:** Packages the generated RTL and XDC sources of the IP customization.

     By deciding to include the XCI files, the IP Packager packages only the XCI file of the IP customizations. This creates a subcore reference to the parent IP and allows the packaged XCI file to be managed by the Vivado tool. The advantage is that the IP can easily be upgraded to the latest release by using the Vivado IP Upgrade methodology. This also means that the parent IP can become *locked* if the subcore IP has a new release. This will require the parent IP to be repackaged with an upgraded subcore for the latest Vivado Design Suite. See Determining Why IP is Locked in Chapter 2 for more information.

     In the case of including the IP generated files, the IP Packager packages all the generated RTL and XDC sources of the IP Customization.

This removes any reference of Vivado managing the IP customization and treats the IP as standard project sources. The advantage is that the IP can be transferred among versions easily as the IP definition versioning information is lost.

When you include the IP generated files, the IP packager packages the generated RTL and XDC sources of the IP customization. This removes any reference to the Vivado tool to manage the IP customization, and the Vivado tool treats the IP as standard project sources.

2. Click **Next**.

The New IP Creation dialog box, (Figure 8-6), summarizes the information that the Create and Package IP wizard gathers about the design automatically, and creates a basic IP package in a staging area.

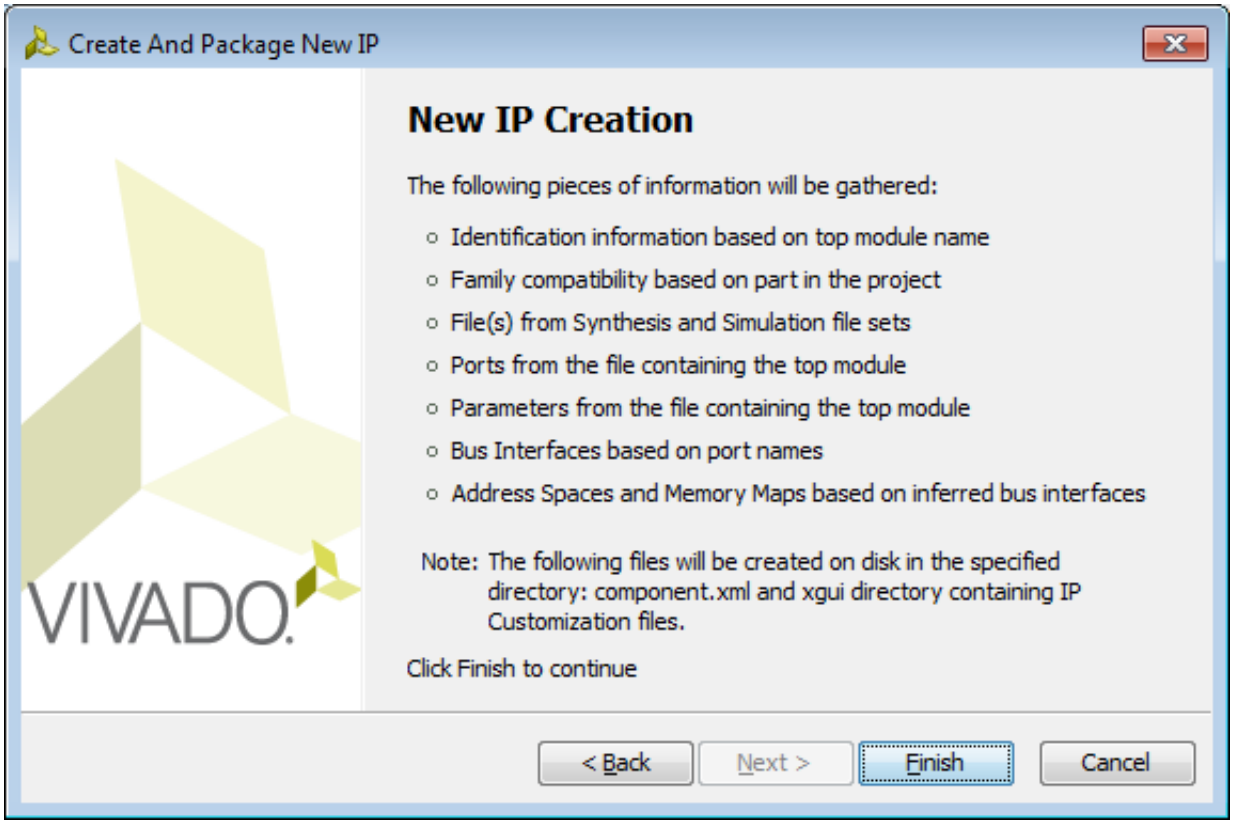


Figure 8-6: **New IP Creation Dialog Box**

3. Click **Finish**.

This process creates files on disk in the specified IP Location directory:

- The `component.xml` containing the packaging information set in the IP-XACT standard.
- The `xgui` directory containing Tcl procedures for handling the proper customization of the IP.

The Create and Package New IP wizard creates the files and directory initially and updates them accordingly, after re-packaging the IP in the Package IP window.

Packaging a Specified Directory

When you specify that a directory to package, the dialog box opens as shown in Figure 8-7.

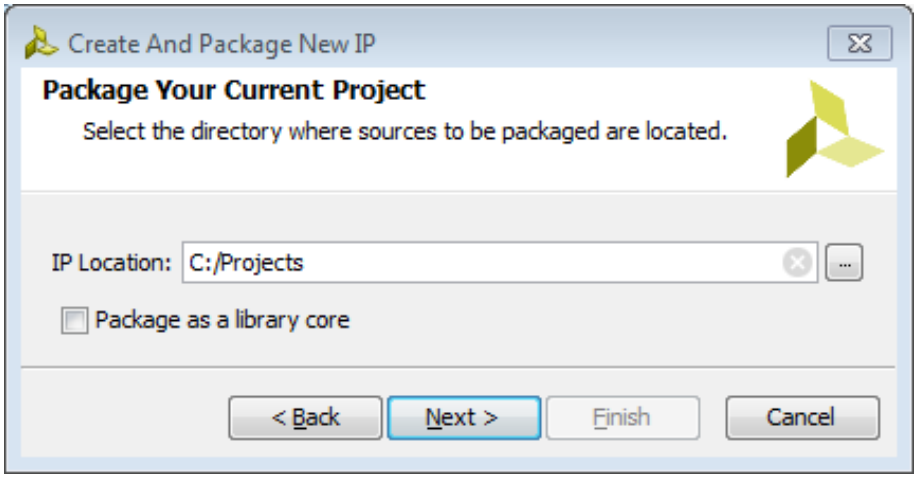


Figure 8-7: **Package Your Current Project Dialog Box**

1. Make your selections from the following options:
   - **IP Location**: The design source directory and location of the IP definitions.
   - **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**.

   The Edit in IP Packager Project Name dialog box displays as shown in Figure 8-8, page 77.

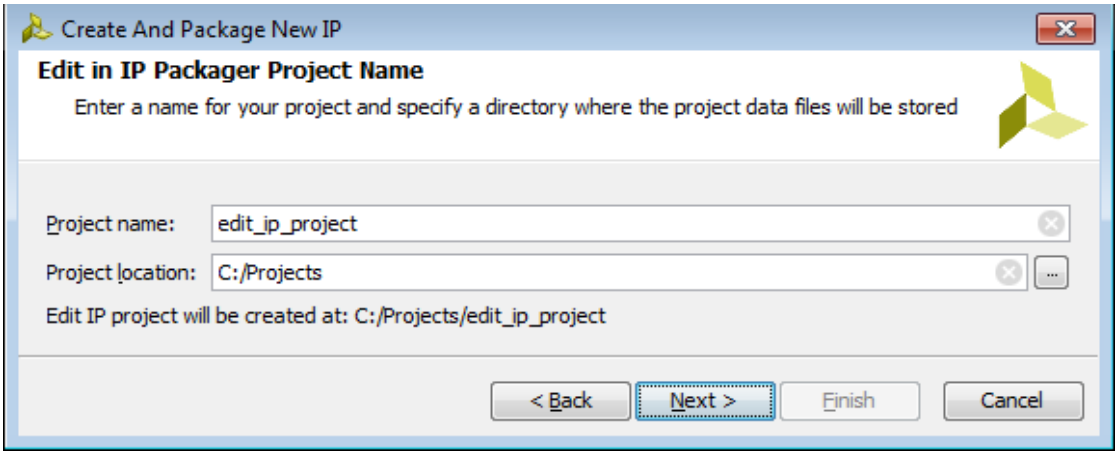


Figure 8-8: **Edit in IP Packager Project Name and Location**

3. Enter the following information:
   - **Project name**: Project name created with the generated IP definition.
   - **Project location**: The directory where the design sources exist and the Edit IP Packager Project will be located.
4. Click **Next.**

The next dialog box, (Figure 8-9), summarizes the information that the wizard gathers about the design automatically for the New IP Creation, and creates a basic IP package in a staging area.

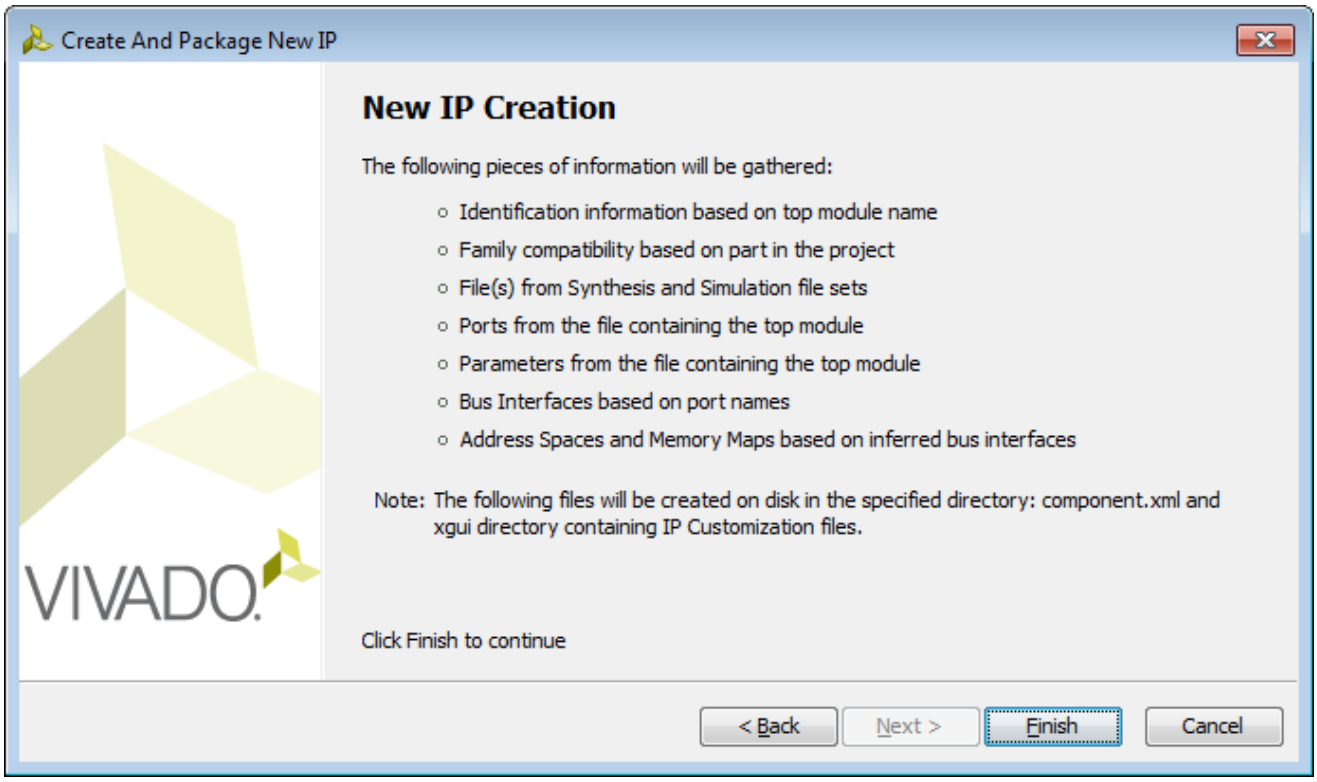


Figure 8-9: **New IP Creation Dialog Box**

5. Review your IP creation details, and click **Finish**.

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 (Figure 8-10).

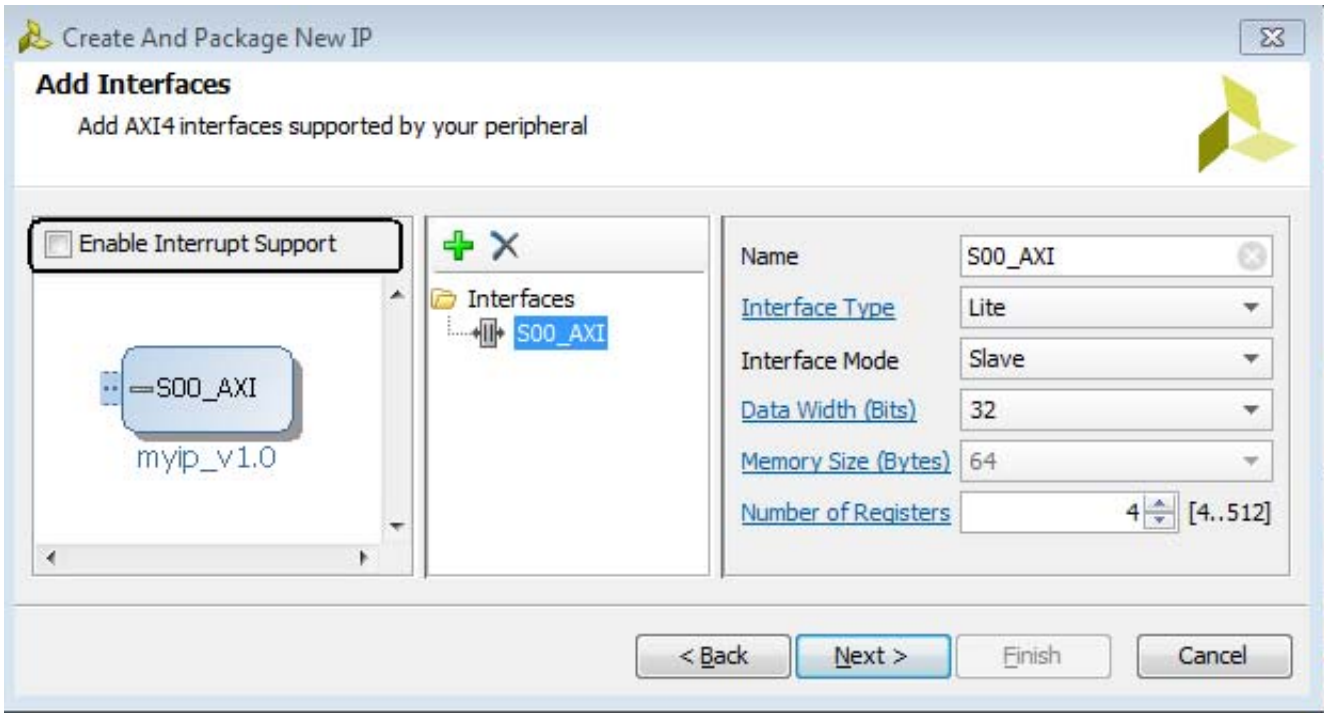


Figure 8-10: **Create and Package New IP: Add Interfaces Dialog Box**

5. To include interrupts to be available in your IP, check **Enable Interrupt Support** (highlighted in Figure 8-10).

   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.

6. Click **Next**.
7. Review your selections.

   The details of your IP are listed in the final wizard page (Figure 8-11).

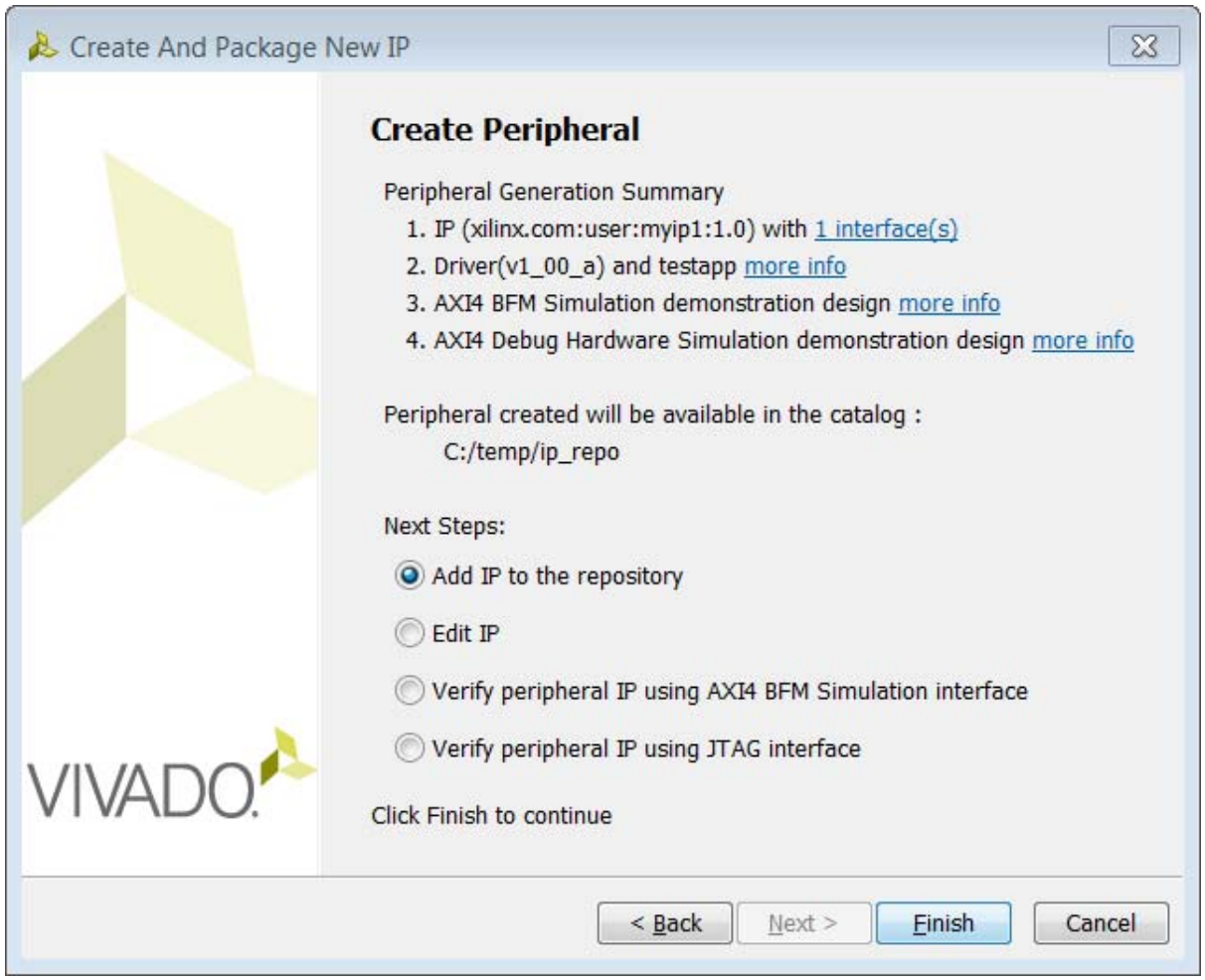


Figure 8-11: **Create and Package New IP: Peripheral Summary Dialog Box**

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 22] 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 15] for more information.

After you create the peripheral, have the option to add custom logic and make the peripheral a custom IP. See Appendix E, Adding User Logic to Your AXI4 Peripheral for a brief explanation.

Using the Vivado IP Packager

After you finish using the Create and Package IP wizard and have input the required information the Vivado IP packager, (Figure 8-12), opens with the details about the packaging steps that were performed.

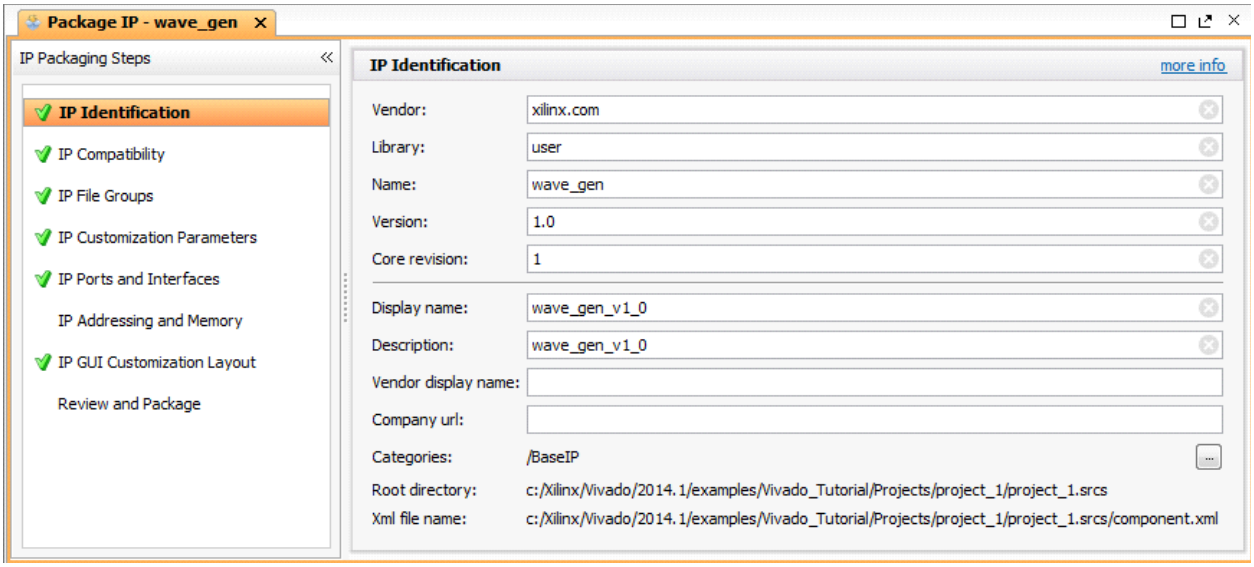


Figure 8-12: **IP Identification Page**

1. Review the IP Packaging steps in the Package IP page:
   - **IP Identification**: Information used to identify your IP. See IP Identification, page 82.
   - **IP Compatibility**: Configure the parts and/or families of Xilinx devices that are compatible with your IP. See IP Compatibility, page 85.
   - **IP File Groups**: Individual files for your IP are grouped into specific file groups. See IP File Groups, page 88.
   - **IP Customization Parameters**: Specify the parameters to customize your IP. See IP Customization Parameters, page 96.
   - **IP Ports and Interfaces**: Top-level ports and interfaces for your IP. See Defining IP Ports and Interfaces, page 111.
   - **IP Addressing and Memory:** Specify the memory-maps or address spaces. See Setting IP Addressing and Memory, page 130.

- **IP GUI Customization Layout:** Configure the parameters that appear on each page of the Customization GUI. See IP GUI Customization, page 135.
- **Review and Package**: Summary of the IP and repackaging. See Review and Package, page 145.

For each section, an icon describes the status of the information. Table 8-1 describes the status icons.

Table 8-1: **Packager Summary Icons and Descriptions**

Packager Icon	Description
	The section is complete with no errors or warnings.
	The section contains a warning that should be examined before completing packaging of the IP.
	The section contains an error that requires a resolution before completing the IP packaging.
	The section has data changes that require regeneration

Use these icons to help you determine if the section requires additional attention.

IP Identification

The IP Identification page, (Figure 8-12, page 81), is the first section of the IP Packager. The information is initially populated based on the information described in the IP Settings of the Project Settings, and determined heuristically by the tool during the Create and Package IP wizard.

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

- **Vendor**: The vendor of the IP. This is also the identifier for the vendor that displays in the VLNV of the IP definition.
- **Library:** The library in which the IP belongs. This is also the identifier for the library that displays in the VLNV of the IP definition.
- **Name:** The name of the IP. This is also the identifier for the name that displays in the VLNV of the IP definition.
- **Version**: The version of the IP. This is also the identifier for the version that displays in the VLNV 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.*

The **Display Name** and **Description** options give additional detailed information about the IP in the Vivado IP Catalog. The Vendor, Library, and Category information is initially populated by the information set in the IP Packager Settings of the Project Settings.

Each category is separated in the list by the forward slash (/) character. Initially, Vivado defaults the IP to the **BaseIP** 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 Figure 8-13, page 84.

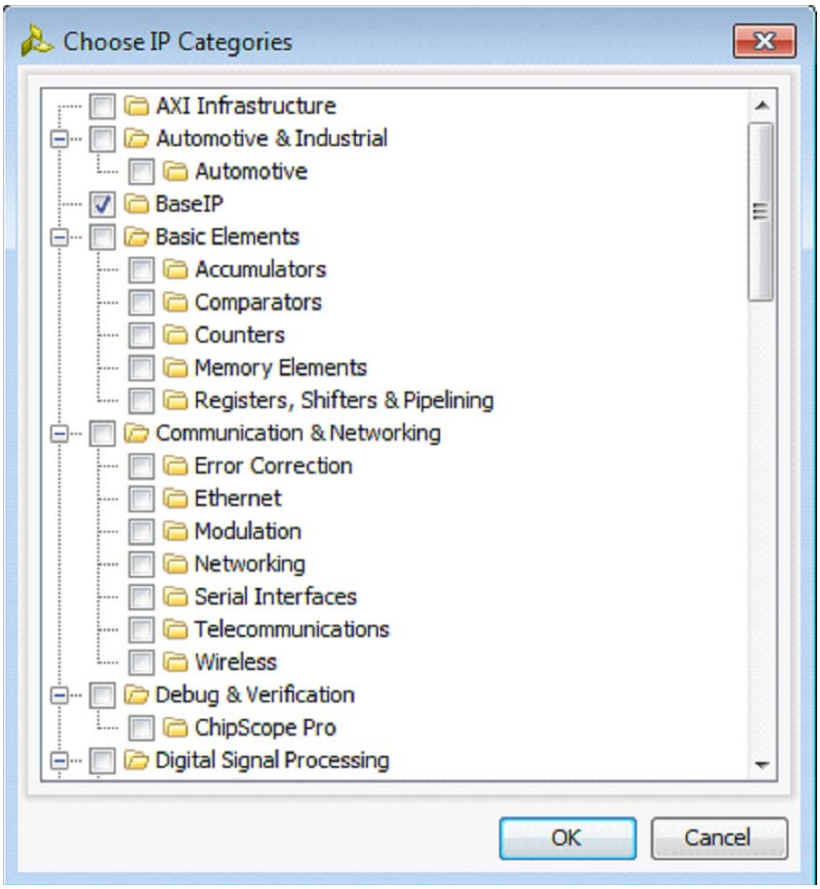


Figure 8-13: **Choose IP Categories Dialog Box**

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

IP Compatibility

The IP Compatibility section, (Figure 8-14), configures which parts or Xilinx device families are compatible with IP you are packaging.

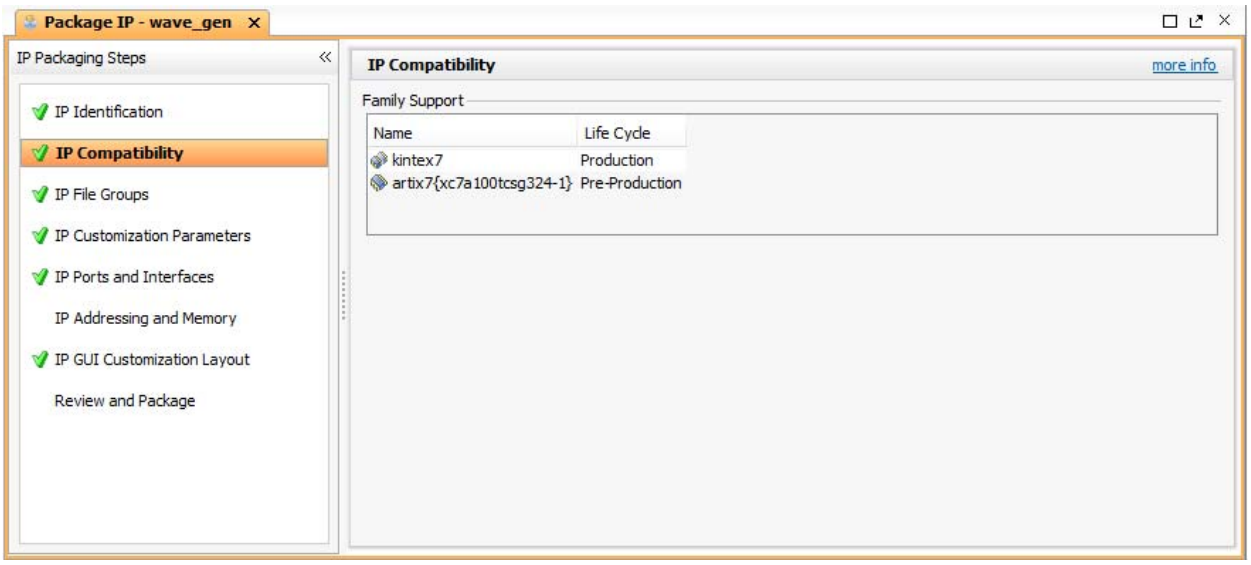


Figure 8-14: **IP Compatibility Page**

The information is initially populated with the 7 series Kintex® device family with the Life Cycle set to Production.

To add a family or part to the supported list of Xilinx devices for the IP:

1. From the IP Compatibility page, open the Choose Family Support dialog box, or right-click the IP Compatibility window, and select **Add Family**, as shown in Figure 8-15, page 85.

Figure 8-15: **Add Family Option**

The Choose Family Support dialog box opens, (Figure 8-16, page 86), and lists all the family and parts available in the Xilinx device catalog.

This dialog box opens in the **Manual selection** mode for specifying the Family support list.

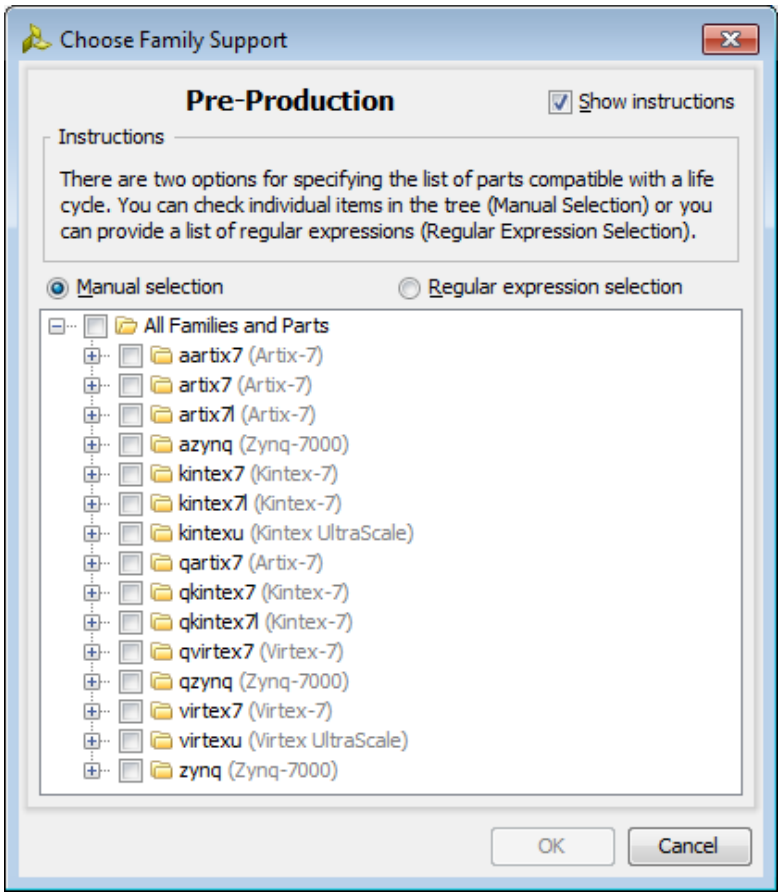


Figure 8-16: **Choose Family Support Dialog Box**

2. Add a part using either the **Manual selection** or the **Regular expression selection**. Figure 8-17 shows the Expression option selected.

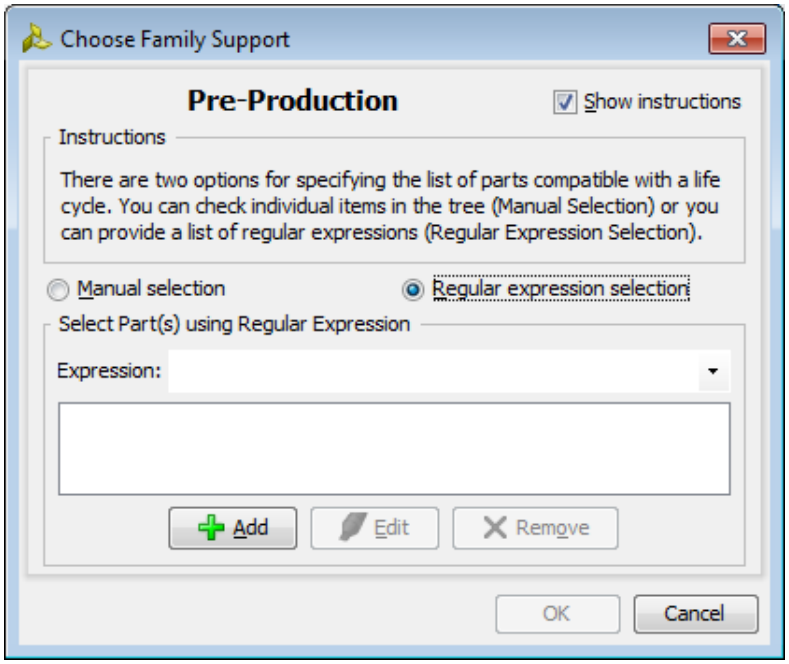


Figure 8-17: **Choose Family Support with Regular Expression Selected**

- In **Manual selection** mode, you can expand the Family folders to select individual parts, or select the entire Family folder.
- In the **Regular expression selection** mode, you can add specific parts through a regular expression.

To add a regular expression selection to the search, add the required syntax in the **Expression** text field. When finished selecting the compatible parts for the IP, and click **OK**.

The syntax for the regular expression search is `familyName{regexp}`, for example:

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

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

Setting Life Cycle Properties

Each family or part has a Life Cycle property to describe the current status of the family or part for this IP.

Any parts or families you add are set automatically to a Life Cycle set to Pre-Production. Adjust the Life Cycle as needed. The Life Cycle properties inform the end-user of the IP status against the part or family being used.

The following options are available to describe the Life Cycle for a given part or family. More details about the IP Life Cycle definitions can be found on the Xilinx LogiCORE IP Life Cycle Definitions page of the Xilinx website.

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

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

To change the Life Cycle property, select the value in the Family or Part row you would like to change, and use the drop-down menu to open the values, as shown in Figure 8-18.

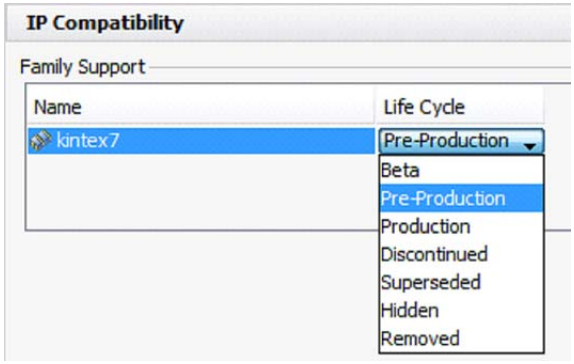


Figure 8-18: **Life Cycle Property Change**

IP File Groups

The IP File Groups section, (Figure 8-19, page 89), provides a listing of individual files for your IP and how they are grouped into specific file groups.

You determine the file groups initially in the Create and Package IP wizard. The wizard determines file groups, either by:

- Using the file sets of the current project
- Heuristically determining the data from the packaged directory

Two sections separate the file groups:

- Standard: Table 8-2, page 92 describes these options.

- Advanced: Table 8-3, page 92 describes these options

For most IP developers, the Standard file group contains the necessary groups for packaging IP for reuse. In case you want to use additional advanced features, the Advanced file group exists.

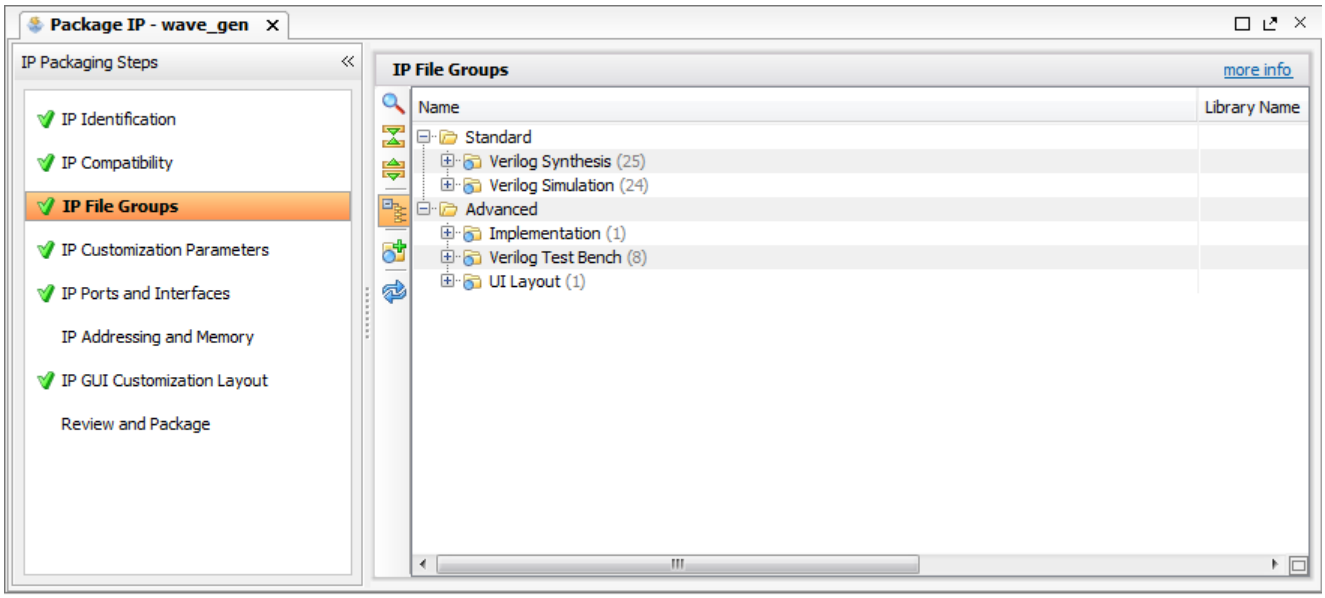


Figure 8-19: **IP File Groups**

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 page shows the list of files as illustrated in Figure 8-20.

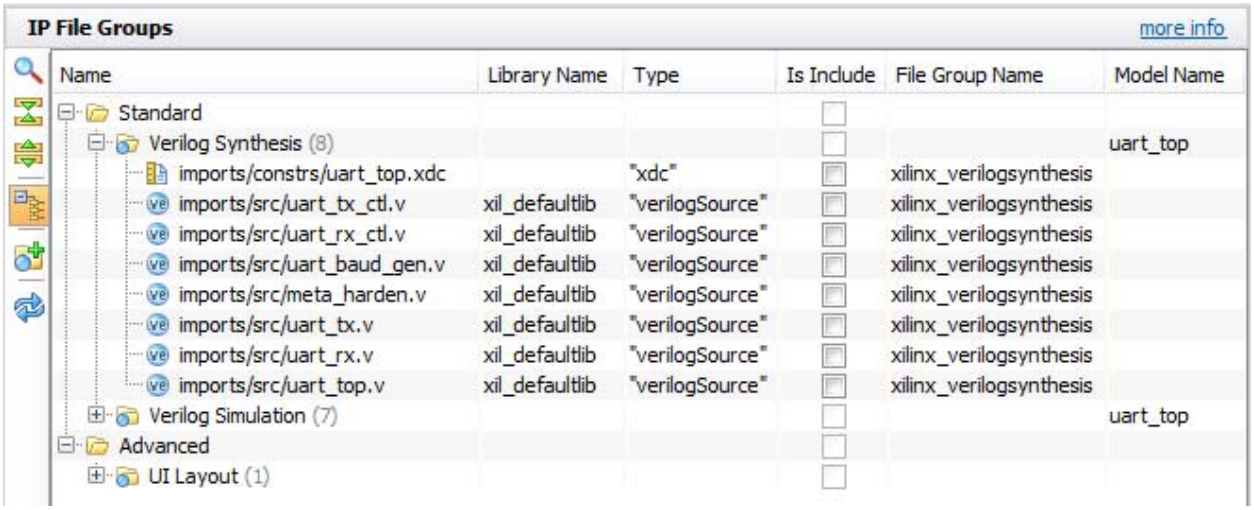


Figure 8-20: **IP File Groups List Expansion**

As shown in Figure 8-20, IP File Groups data properties contains information about the individual files properties and for the file group.

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**: The 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.

The **Model Name** is a *required property* and the last property of the column list. This property applies directly to file group and is the top-level HDL file (the name of the design). This value is set for **Synthesis**, **Simulation**, and **Implementation** file groups. The IP packager reports an error in the message window if you do not set a property.

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.*

RECOMMENDED: *To add files to the IP Packager, the recommendation is to use the Vivado IDE* ***Add Sources*** *option to add the files to the packaged Vivado project with the IP Packager window open.*

If you have followed the recommendation and added files to the project with the IP Packager window open, the updated files can be merged into the IP packager as shown in Figure 8-21. For additional information on how to add files to your Vivado project, see the *Vivado Design Suite User Guide: System-Level Design Entry* (UG895) [Ref 7].

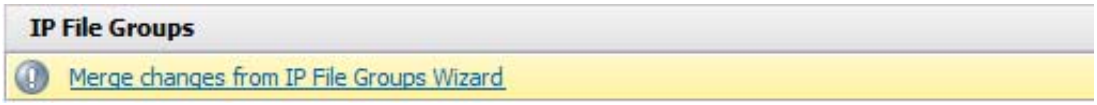


Figure 8-21: **IP File Groups Wizard**

Selecting the IP File Groups Wizard hyperlink imports or removes the files associated with the Vivado project. After the wizard is complete, the new files are visible in the IP File Groups window.

To manually add a file group to the IP File Groups window:

1. Right-click **IP File Groups**, and select **Add File Group**, as shown in Figure 8-22.

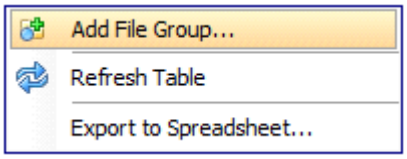


Figure 8-22: **Add File Group Option**

The Add IP File Group dialog box opens.

2. Choose which **File Group** types to add to the IP, as shown in Figure 8-23.

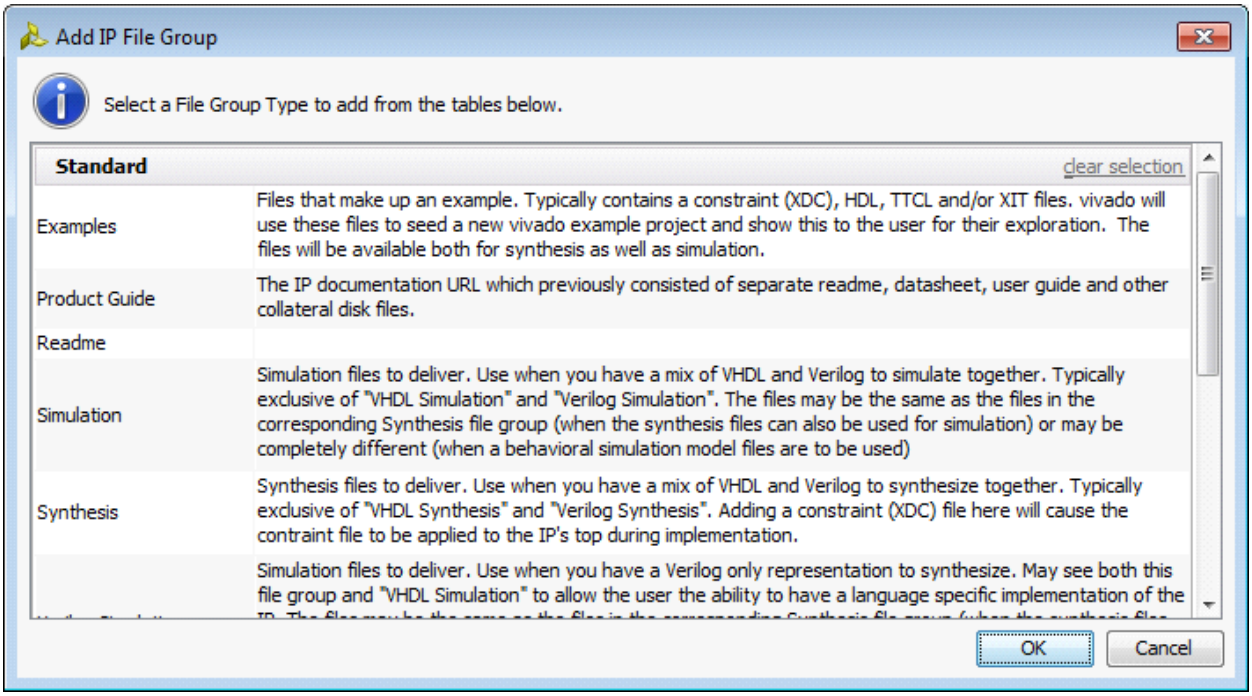


Figure 8-23: **Add IP File Group Dialog Box**

By default, the Standard file groups are displayed. Each file group has a specific name and function that is described in the dialog box.

Table 8-2 lists the Standard file group types and a description.

Table 8-2: **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 `readme.txt` 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.
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.

Table 8-3 contains a listing and a description of each **Advanced** file group type.

Table 8-3: **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.

Table 8-3: **Advanced File Group Types and Descriptions *(Cont'd)***

Advanced File Groups	
C Simulation	Use to deliver files for c-model simulation. These files are copied out without any further processing (excepting tcl/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.
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.

Table 8-3: **Advanced File Group Types and Descriptions *(Cont'd)***

Advanced File Groups	
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.

3. After selecting the file groups you want, click **OK**.

Adding Files to File Groups

After you add the IP File Groups to the IP File Group window, you can now add files to the different file groups. You can add files as follows:

- To a file group from the file system
- From a previous file group
- From a sub-core reference

To add files to a specific file group from the file system:

1. Open the Add IP Files dialog box, right-click the respective file group, and select **Add Files**, as shown in Figure 8-24.

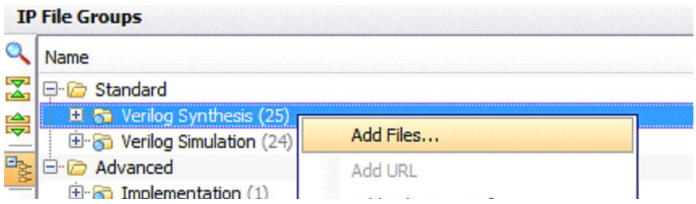


Figure 8-24: **Add Files Option**

The Add IP File Group dialog box opens and lets you choose which File Group types you would like to add to the IP, as shown in Figure 8-25, page 95.

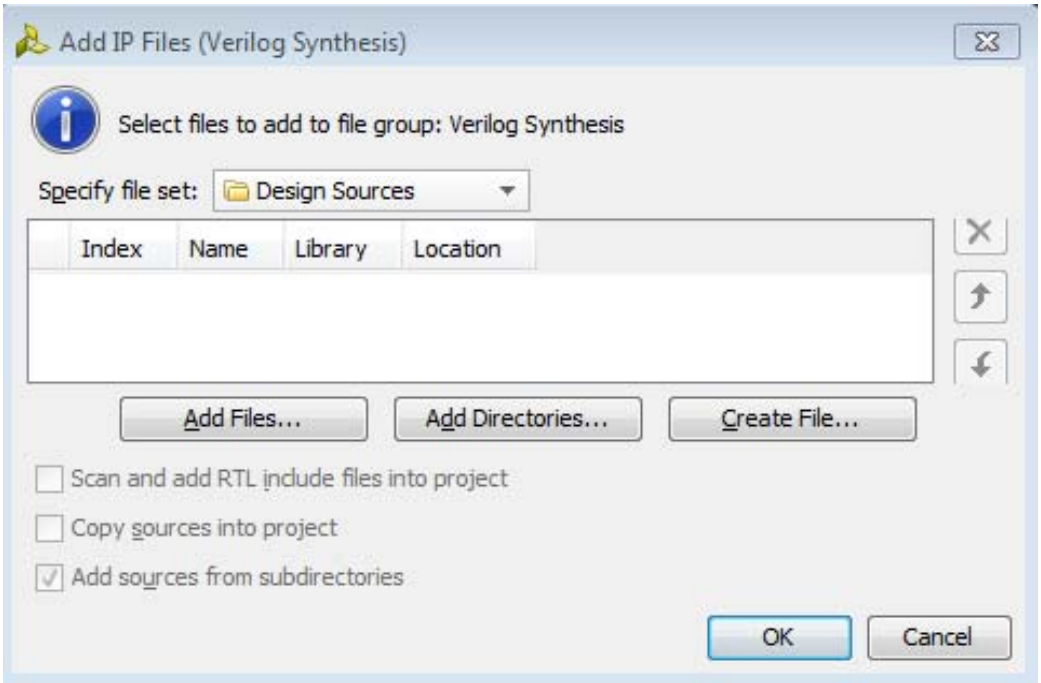


Figure 8-25: **Add IP Files Dialog Box**

The Add IP Files dialog box includes the associated file group in parenthesis in the title to help avoid confusion about which files are associated to which file group. The Add Files dialog box adds to one file group at a time.

You can directly add files or directories, or create one of the four types of files: Verilog, Verilog Header, System Verilog, or VHDL.

RECOMMENDED: *Check the **Copy sources into project** option before adding files to the IP Packager project. Without this option, the IP Packager references files are as an absolute path based on the your file system. This causes issues for IP used outside the original file system.*

2. After selecting the files to add to the IP File Group, click **OK**.

Adding Files to Multiple File Groups

To add files to more than one file group, such as synthesis and simulation, the options are

- Add the files using the Add IP Files dialog box to each of the file groups.
- For files that already exist in a file group, you can copy a file or entire file group using the **Copy To** option.

Right-click the file or file group, and select **Copy To**, as shown in Figure 8-26.

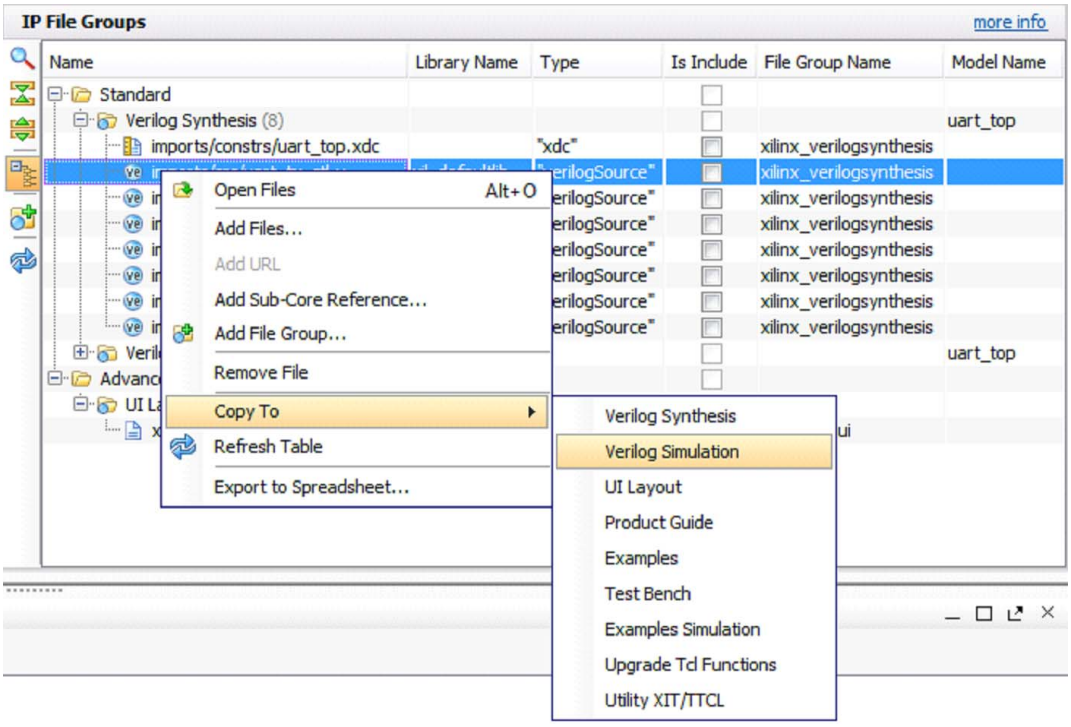


Figure 8-26: **Copy To Option**

The **Copy To** option extends the dialog box with the list of available IP File Groups. The list contains the file groups listed in the IP File Groups window as well as some 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 to copy, the **Copy To** option copies the child files under the file group to the specified file group.

IP Customization Parameters

The IP Customization Parameters section, (Figure 8-27, page 97), provides a listing of parameters passed to the top-level HDL source file.

Using the Create and Package IP wizard, the customization parameters populate based on the heuristic parsing of the top-level HDL source file.

You can then add, edit, or remove parameters in the IP Customization Parameters page.

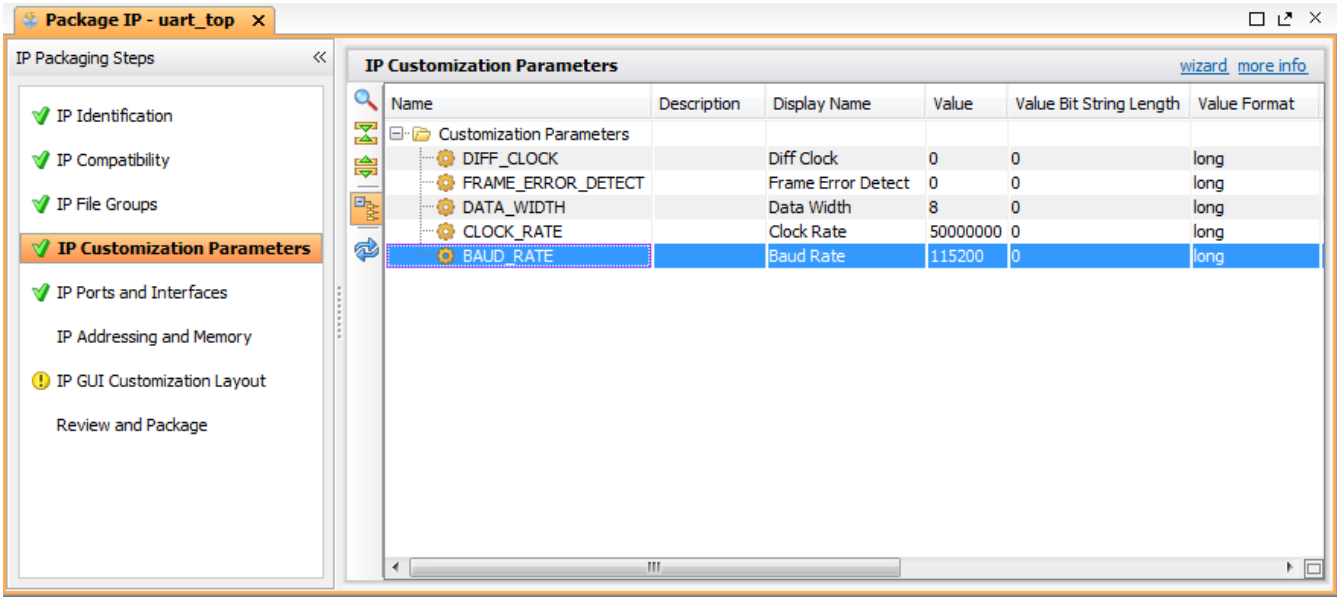


Figure 8-27: **IP Customization Parameters**

There are two folders for parameters in the list: visible, and hidden. By default, the parameters parsed from the wizard are visible and organized in the `/Customization Parameters` folder.

- Visible customization parameters show in the IP Customization GUI.
- Hidden customization parameters are for parameters not visible to the end-user, and not intended for direct editing. These parameters are generally dependent upon the selected parameters to determine their values.

In the case that you want to import parameters from the top-level HDL source file due to a change to the HDL after initial packaging, or to initially populate the values, open the IP Customization Parameter wizard.

The wizard is available from the IP Customization Parameters header, (Figure 8-28).

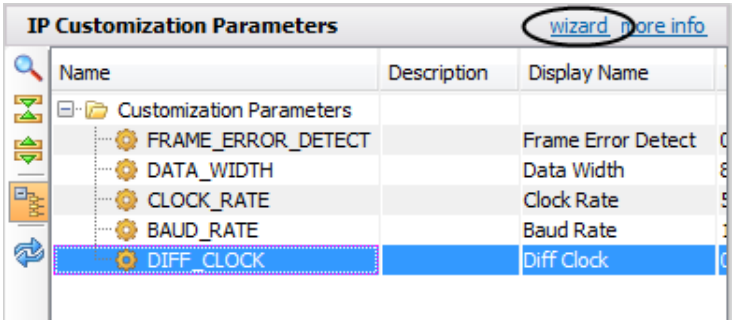


Figure 8-28: **IP Customization Parameter Wizard**

The wizard hyperlink opens the Import Parameters from HDL dialog box, as shown in Figure 8-29.

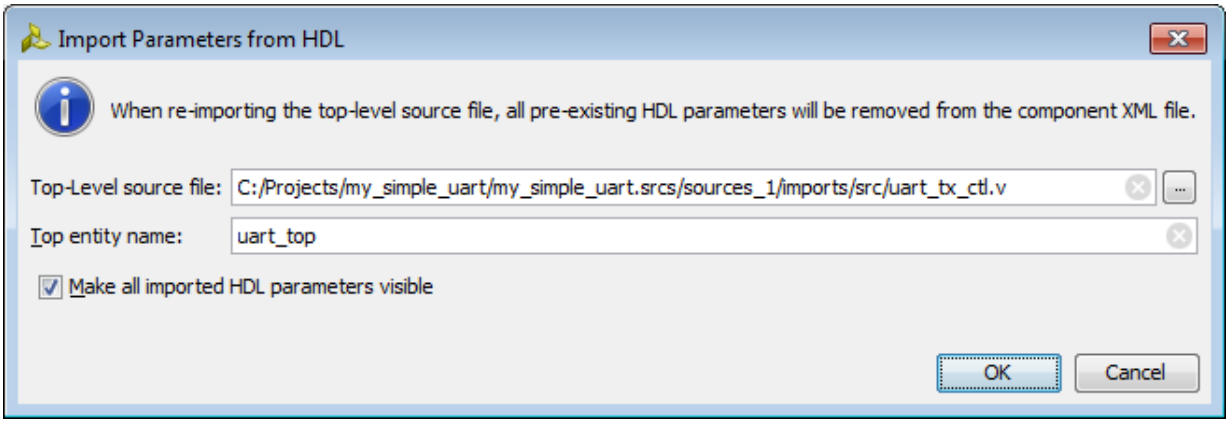


Figure 8-29: **Import Parameters from HDL Dialog Box**

Set the following options:

- **Top-Level source file**: The top-level source HDL file that contains the top-level entity or module.
- **Top entity name**: The name of the top-level entity or module that contains the parameters for the wizard to parse.
- **Make all imported HDL parameters visible**: Enables all the imported parameters to be visible to the end-user. Deselecting this check box places all the imported parameters into the `Hidden` folder.

You can reference the wizard from **IP Customization Parameters** by right-clicking on the window and selecting **Import IP Parameters** as shown in Figure 8-30.

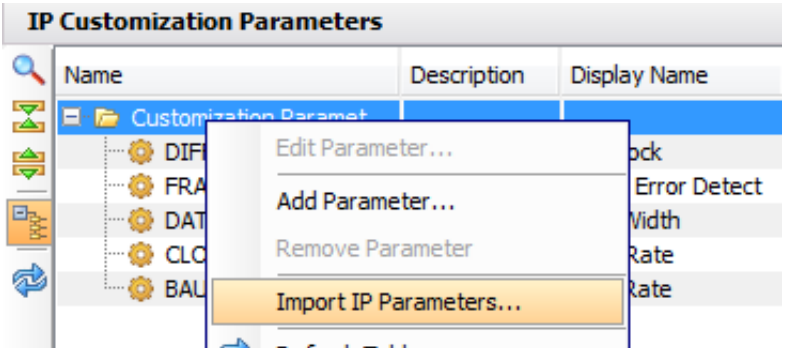


Figure 8-30: **Import IP Parameters Option**

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 Figure 8-31.

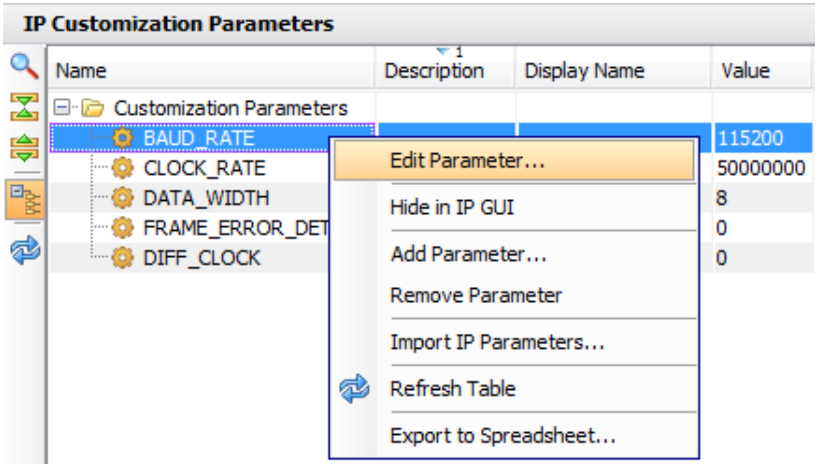


Figure 8-31: **Edit Parameter Option**

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 8-32.

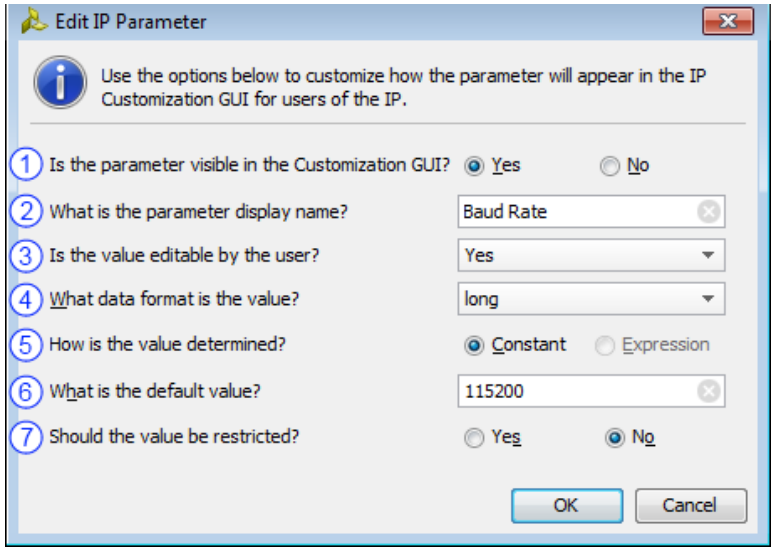


Figure 8-32: **Edit IP Parameter Dialog Box**

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 8-33.

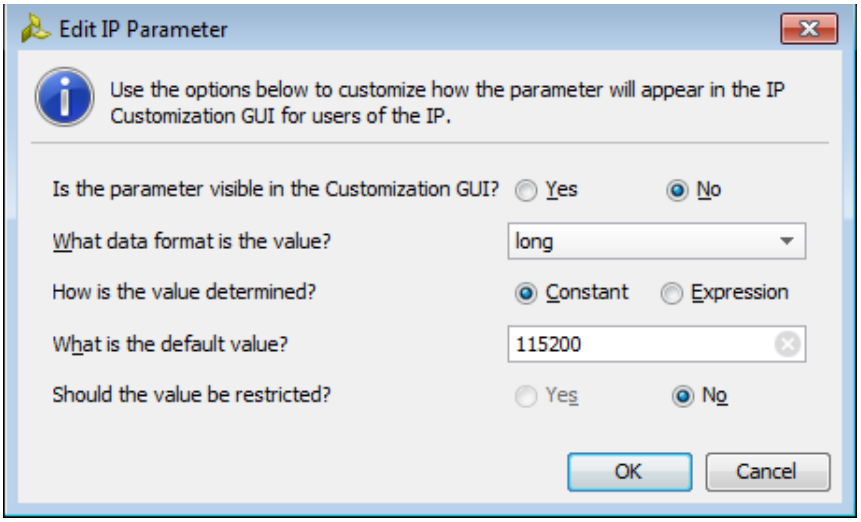


Figure 8-33: **Edit IP Parameter Marked as Hidden**

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 8-34, page 101.

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.

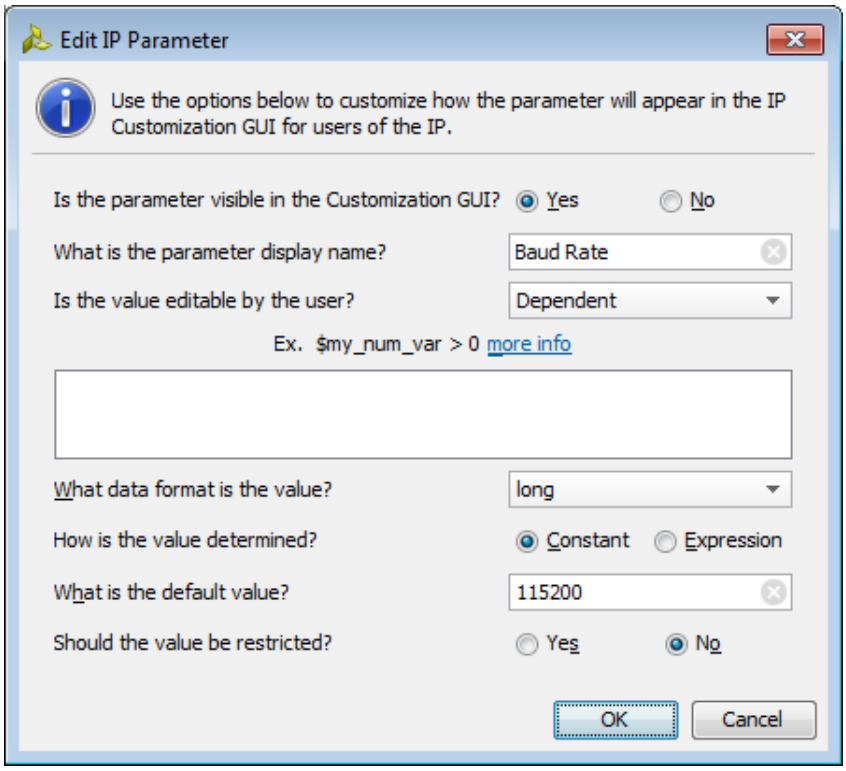


Figure 8-34: **Edit IP Parameter Marked as Dependent**

Parameter enablement is based on Tcl expression rules. To create a proper enablement expression, the parameter name is used with numeric and comparison operators to form a boolean expression.

Note: See the http://wiki.tcl.tk/583 to ensure conformity with `expr` rules

The Tcl variable to reference the parameter is the parameter name. This name is case-sensitive and requires prefix of the Tcl variable (`$`) syntax. For example, if the parameter is named `BAUD_RATE`, the enablement expression variable reference would be `$BAUD_RATE`.

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 a `TRUE` or `FALSE` value.

Figure 8-35 shows the resulting dialog box when you select a boolean format.

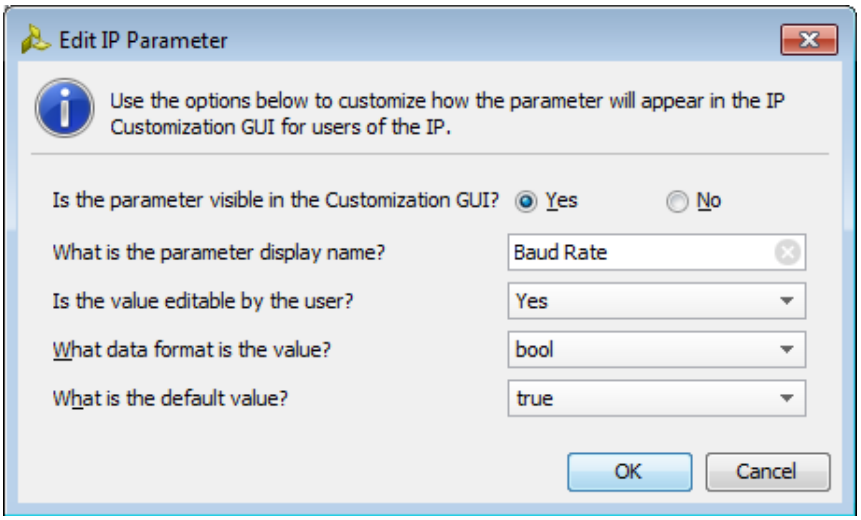


Figure 8-35: **Edit IP Parameter Marked as Bool**

5. **How is the value determined?** This option controls how the value is determined during IP customization. The available selections 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 8-36, page 103.

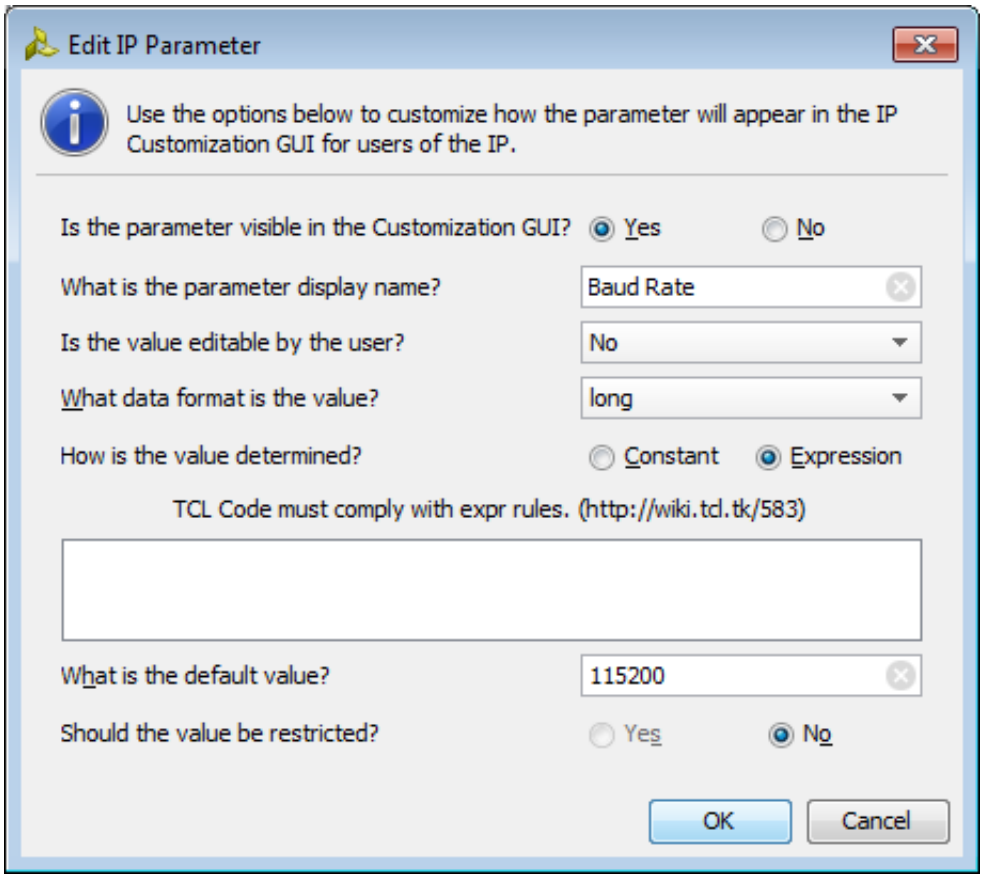


Figure 8-36: **Edit IP Parameter with Expression option**

The expression that evaluates the value can reference parameters defined through the IP Packager. The Tcl variable to reference the parameter is the parameter name. The name is case-sensitive and requires a prefix of the Tcl variable (`$`) syntax.
For example, a parameter named `BAUD_RATE` has an enablement expression variable reference of `$BAUD_RATE`.

Note: See http://wiki.tcl.tk/583 to ensure conformity with `expr` rules.

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 8-37, page 104). The initial selection for the restriction is set to a list of values.

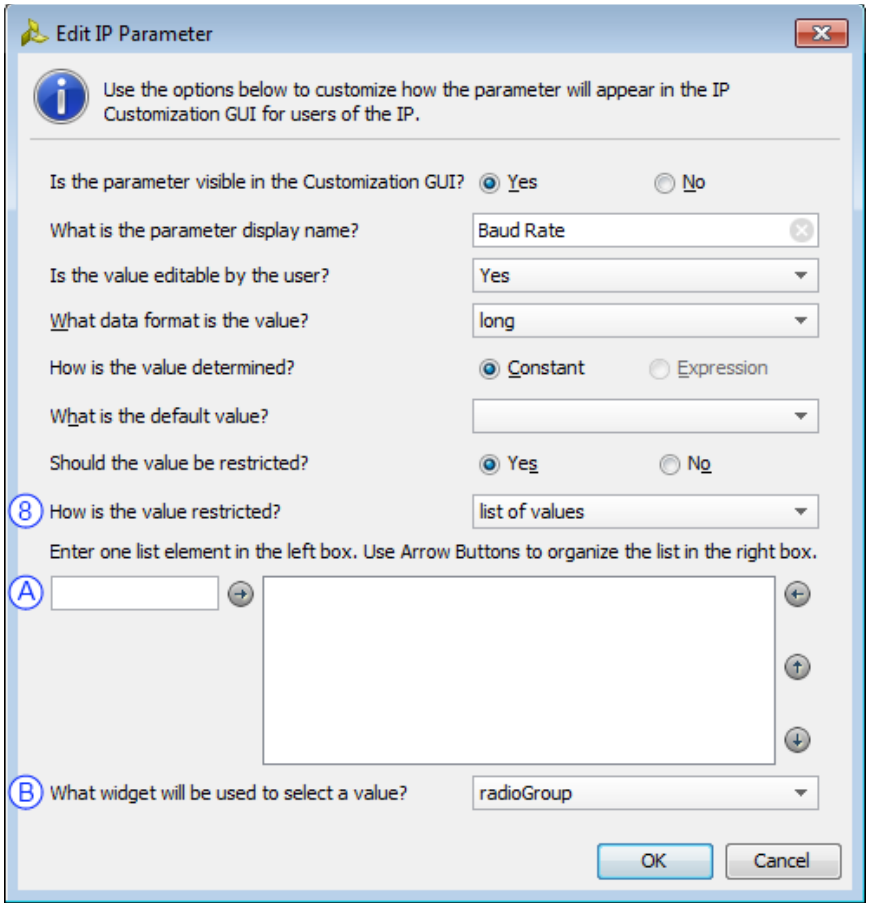


Figure 8-37: **Edit IP Parameter Dialog Box with List of Value Restrictions**

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 8-38, page 105.

      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 Figure 8-38.

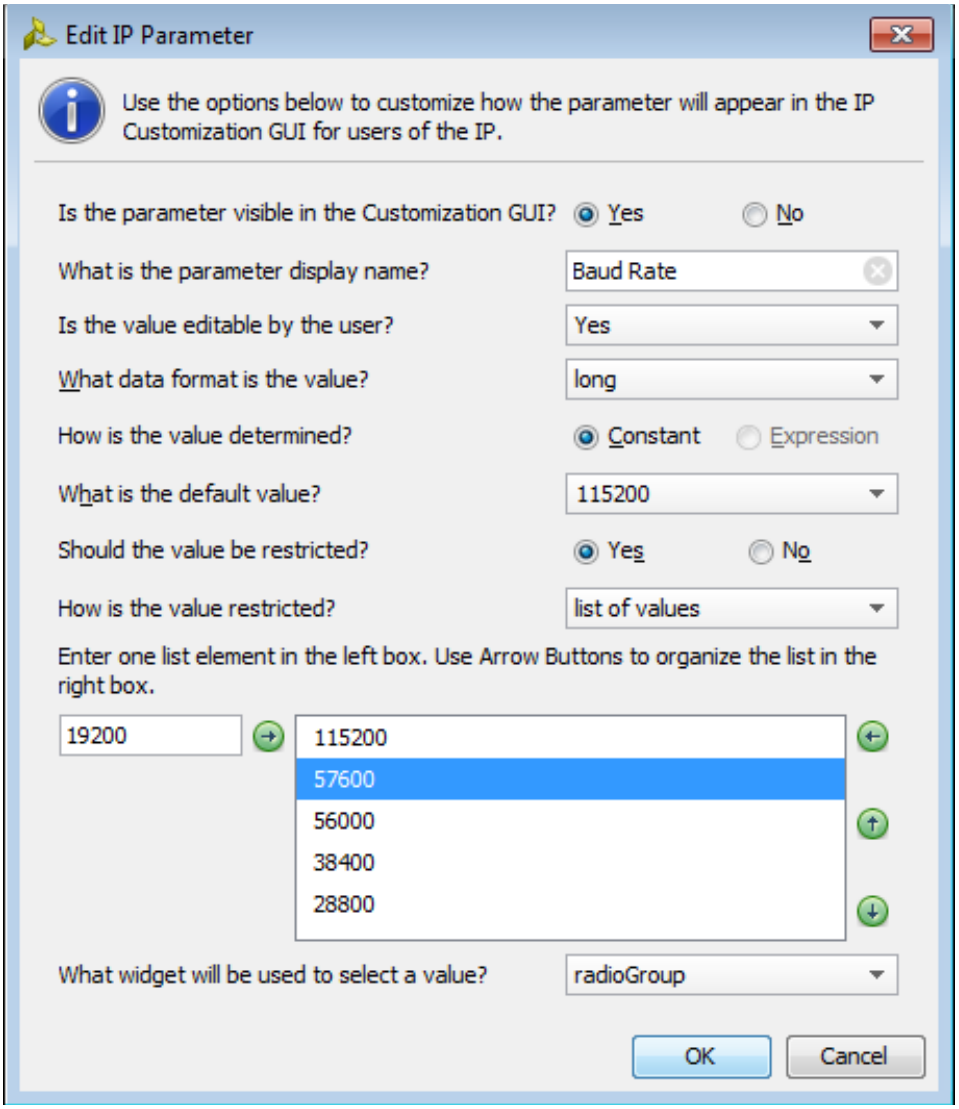


Figure 8-38: **Edit IP Parameter Dialog Box with Organized List Values**

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. For more information on the display of these widgets, see IP GUI Customization, page 135.

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 Figure 8-39, control the values.

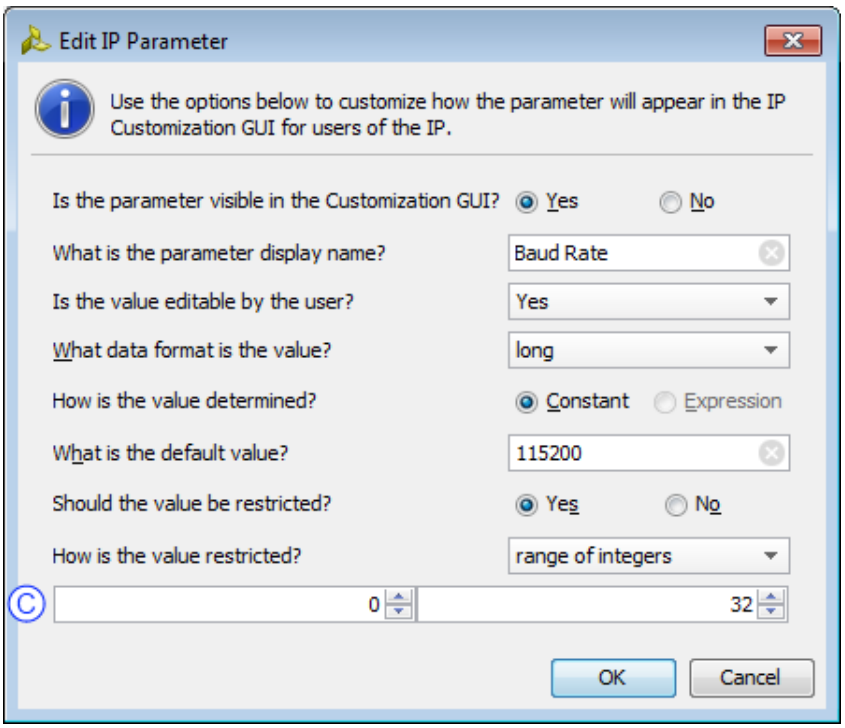


Figure 8-39: **Edit IP Parameter Dialog Box with a Range of Integers Restriction**

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 Figure 8-40, page 107.

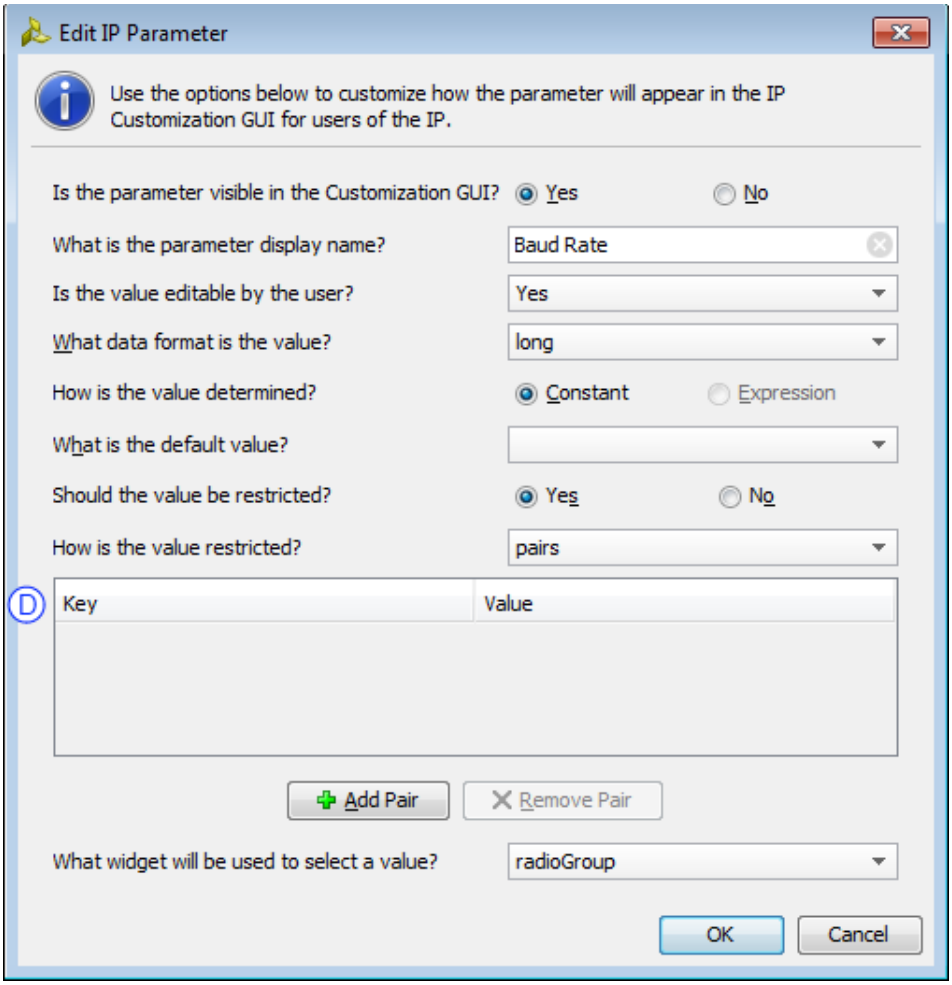
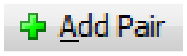


Figure 8-40: **Edit IP Parameter Dialog Box with Pairs Restriction**

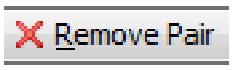
To add a pair to the Key/Value list, click the **Add Pair** button in the dialog box.This creates a generic key/value pair in the list with Key set as **key** and Value set as **value**. To make the Key or Value field editable, double-click the text to change.



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 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, Figure 8-41 shows a dialog box with a list of Key/Value pairs.

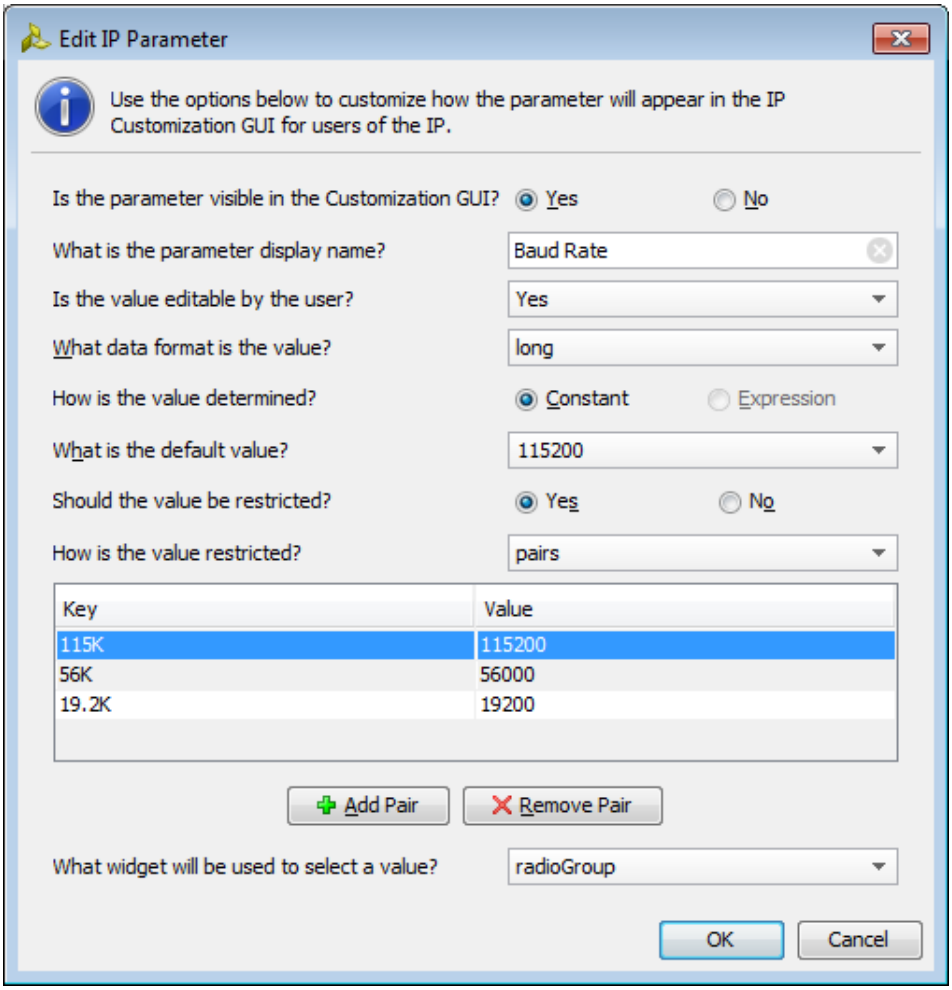


Figure 8-41: **Edit IP Parameter Dialog Box with Key/Value Pair List**

Similarly to the list of values restriction, you can select the type of GUI widget that is used to display the values in the IP Customization GUI. The two available selections are a radioGroup or comboBox.

For more information on the display of these widgets, see the IP GUI Customization, page 135.

Adding a Parameter

You can create parameters for use in the IP GUI Customization. These parameters, because they do not have a reference to the RTL, must be visible.

To add a parameter that is not related to the RTL parameters, do the following:

1. Right-click the IP Customization Parameters window and select **Add Parameter** as shown in Figure 8-42, page 109.

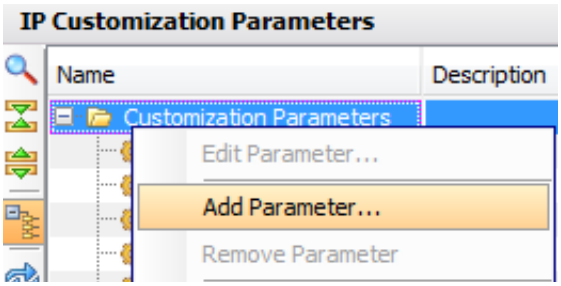


Figure 8-42: **Add Parameter Option**

The Add New Parameter dialog box opens, (Figure 8-43), lets you enter a name for the new parameter. The expected value is the name of the parameter, not the display name that shows in the IP Customization GUI.

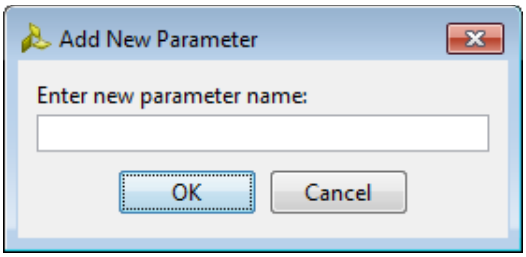


Figure 8-43: **Add New Parameter Dialog Box**

2. Enter the new name of the parameter, and click **OK**.

After creating a new parameter, the Edit IP Parameter dialog box opens populated with the default data values.

Removing a Parameter

To remove a parameter, right-click the parameter, and select **Remove Parameter** as shown in Figure 8-44.

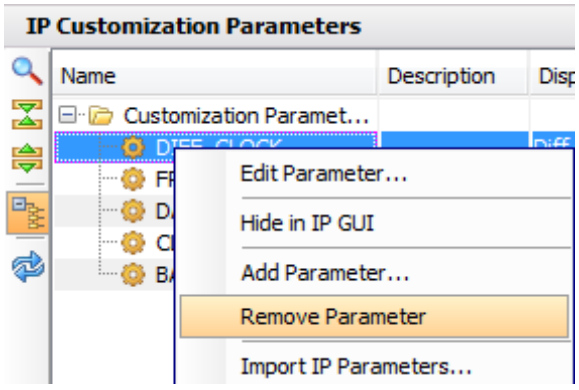


Figure 8-44: **Remove Parameter Option**

This removes the parameter from the IP Customization Parameter and the IP Customization GUI. Any parameter that depended upon the removed parameter is now flagged as an error.

Hiding and Showing a Parameter in the Customization GUI

You can also quickly move a customization parameter between hidden and shown in the IP Customization GUI within the IP Customization Parameters pane.

To hide a Customization Parameter in the list, right-click the parameter, and select **Hide in IP GUI** as shown in Figure 8-45.

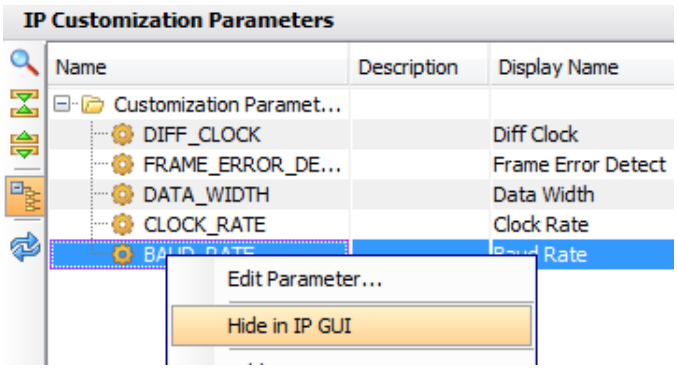


Figure 8-45: **Hide in IP GUI Option**

This option moves the parameter from the `/Customization Parameter` folder to the `/Hidden` folder, as shown in Figure 8-46.

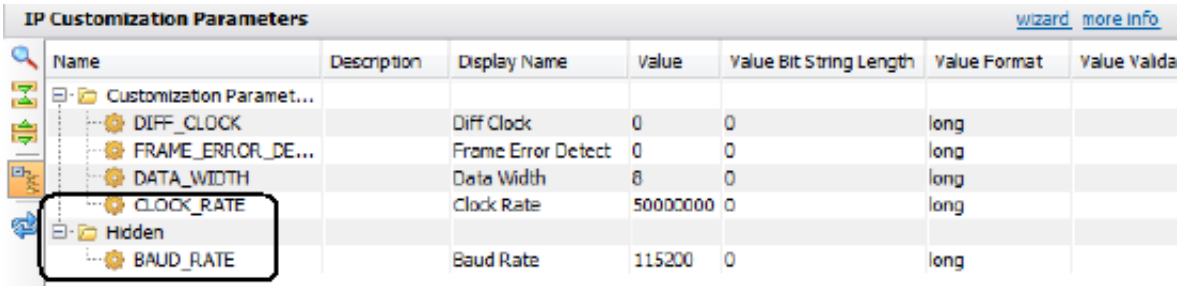


Figure 8-46: **IP Customization Parameters with Hidden Folder**

To move the parameter from the Hidden folder to the Customization Parameters folder, right-click the hidden parameter and select **Show in IP GUI** as shown in Figure 8-47, page 111.

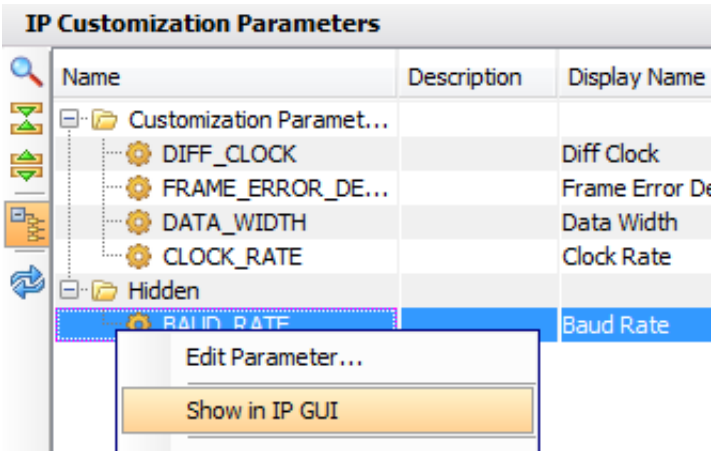


Figure 8-47: **Show in IP GUI Option**

Moving a parameter between Hidden and Shown affects the visibility option of the parameter only. All the other options set for the parameter remain unchanged.

Defining IP Ports and Interfaces

The IP Ports and Interfaces section, as shown Figure 8-48, provides a listing ports and interfaces of the top-level HDL source file.

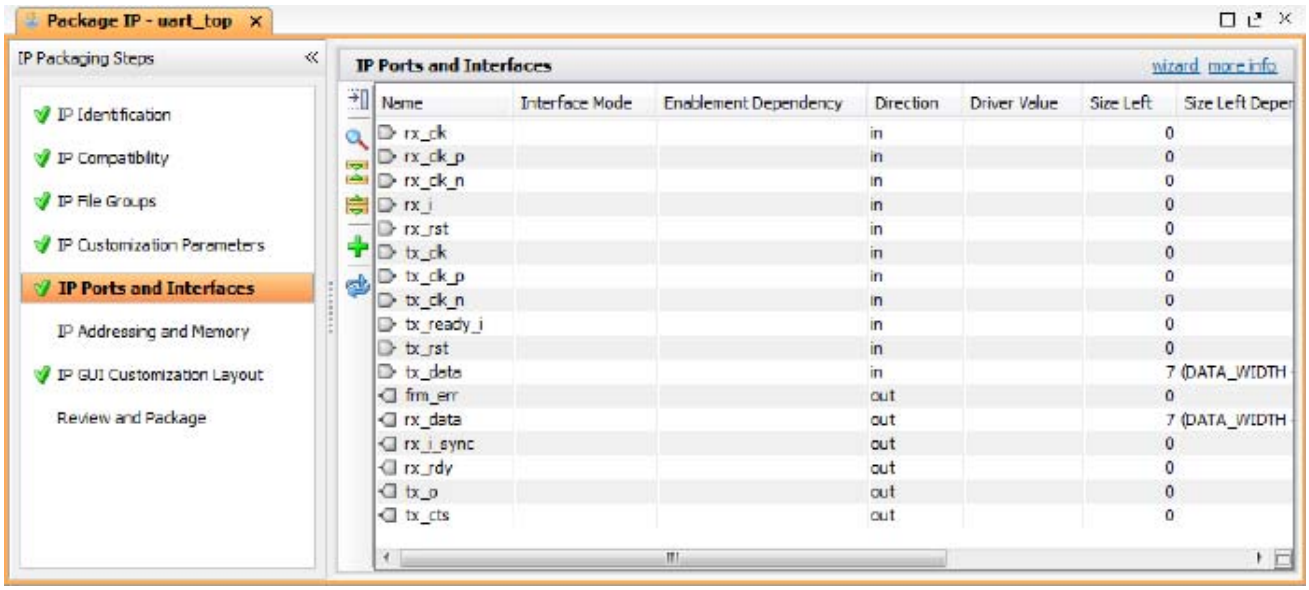


Figure 8-48: **IP Ports and Interfaces Page**

The Create and Package IP wizard, populates the ports and interfaces based upon the heuristic parsing of the top-level HDL source file. You can add interfaces or auto-infer interfaces, based upon the naming convention of the available ports.

By default, the IP Ports and Interfaces table contains the following columns:

- **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.
- **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`).

In the case that you want 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, you can open the wizard.

The IP Ports and Interface wizard is available from the IP Ports and Interfaces header, (Figure 8-49).

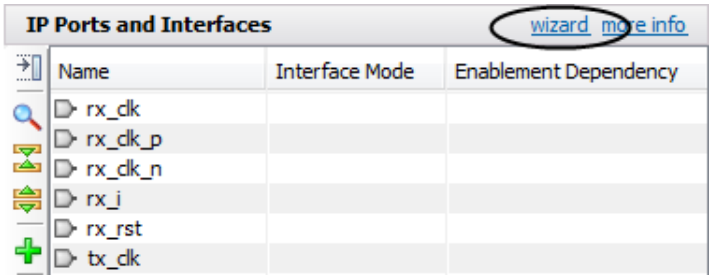


Figure 8-49: **IP Ports and Interfaces Wizard**

The wizard hyperlink opens the Import Ports from the HDL dialog box, (Figure 8-50).

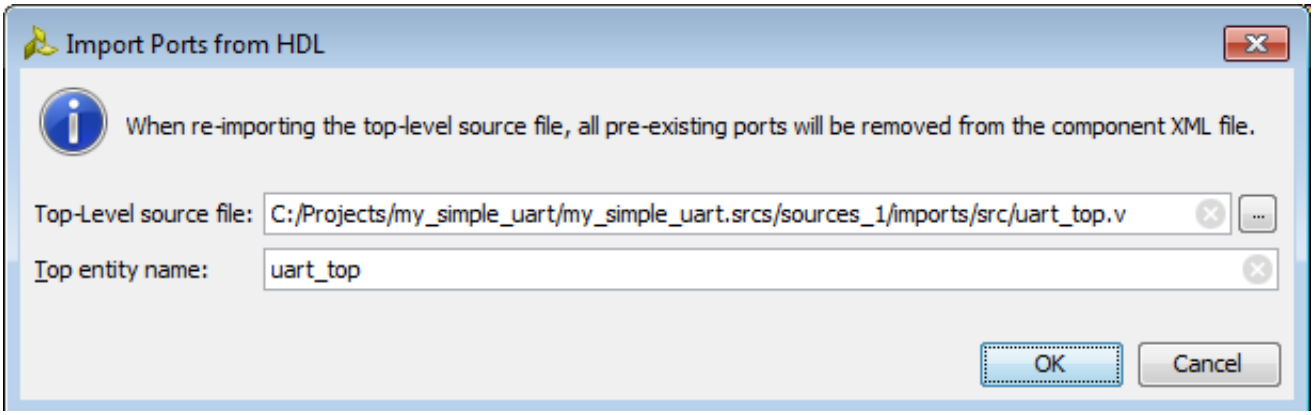


Figure 8-50: **Import Parameters from the HDL Dialog Box**

From this dialog box, set the following options:

- **Top-Level source file**: The top-level source HDL file that contains the top-level entity or module.
- **Top entity name**: The name of the top-level entity or module that contains the ports for the wizard to parse.

The top-level source file and the entity name are populated initially by the top-level information that is inferred when packaging the project.

You can also access this wizard from the IP Ports and Interfaces by right-clicking the window and selecting **Import IP Ports**.

Editing an Existing Port

To edit a port that is listed in the IP Ports and Interface window, double-click the port or right-click the port and select **Edit Port** from the context menu, as shown in Figure 8-51.

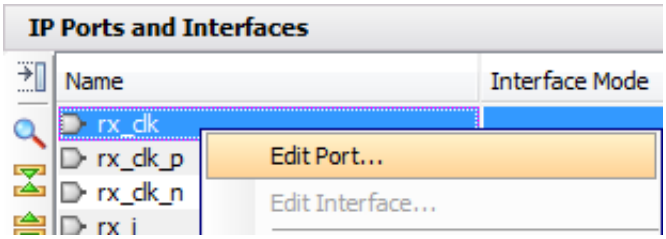


Figure 8-51: **Edit Port Option**

The Edit IP Port dialog box opens, which controls the driver value and the enable expression if the port is optionally present (Figure 8-52).

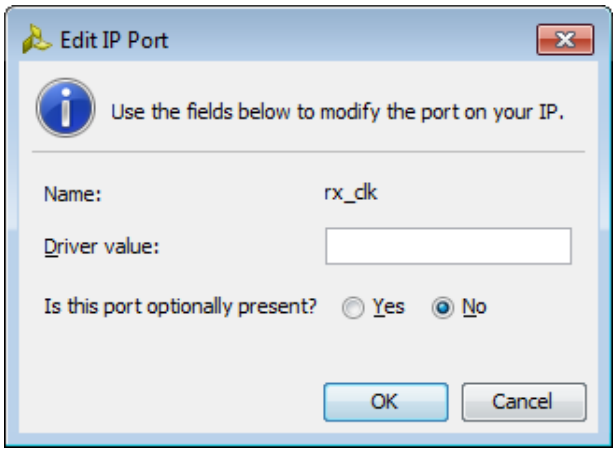


Figure 8-52: **Edit IP Port Dialog Box**

The options are:

- **Name**: The port name (read-only in this dialog box).
- **Driver Value**: If the port is optionally present and disabled during customization, this is the default driver value of the port.
- **Is this port optionally present?**: This option controls whether the port is optionally present. When set to **Yes**, the enablement expression text box becomes visible in the dialog box (Figure 8-53).

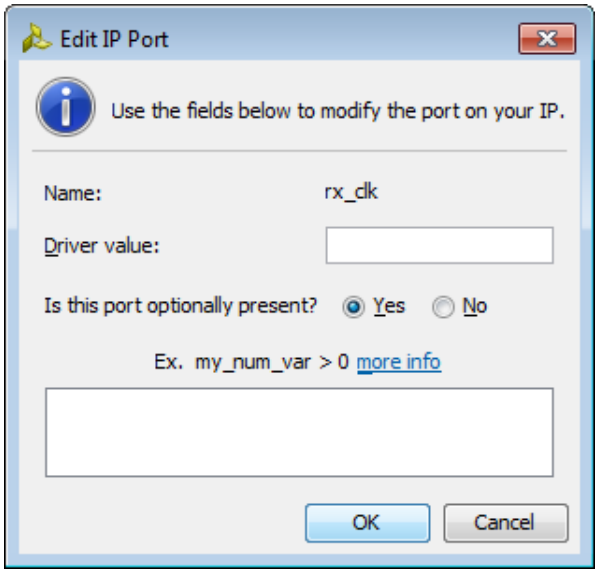


Figure 8-53: **Edit IP Port with Enablement Expression Text Box**

The enablement expression for ports uses the following supported operators (==, >, <, and !=).

The syntax for the support parameters is the name of the parameter. For example, if the parameter is named BAUD_RATE, the enablement expression to ensure the value is 1 is BAUD_RATE==1.

Adding a Bus Interface

You can add bus interfaces in the IP Packager that are supported for your IP. An IP that adds a bus interface must have the ports required by that interface.

IMPORTANT: *The ports do not require specific names, but it is suggested you name the ports exactly as specified in that interface.*

To add a bus interface, right-click the IP Ports and Interfaces window, and select **Add Bus Interface**, (Figure 8-54).

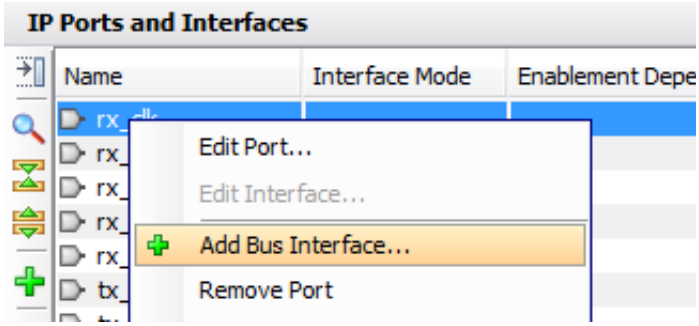


Figure 8-54: **Add Bus Interface Option**

The Edit IP Bus Interface dialog box opens, (Figure 8-55, page 116), where you can set up the IP interface.

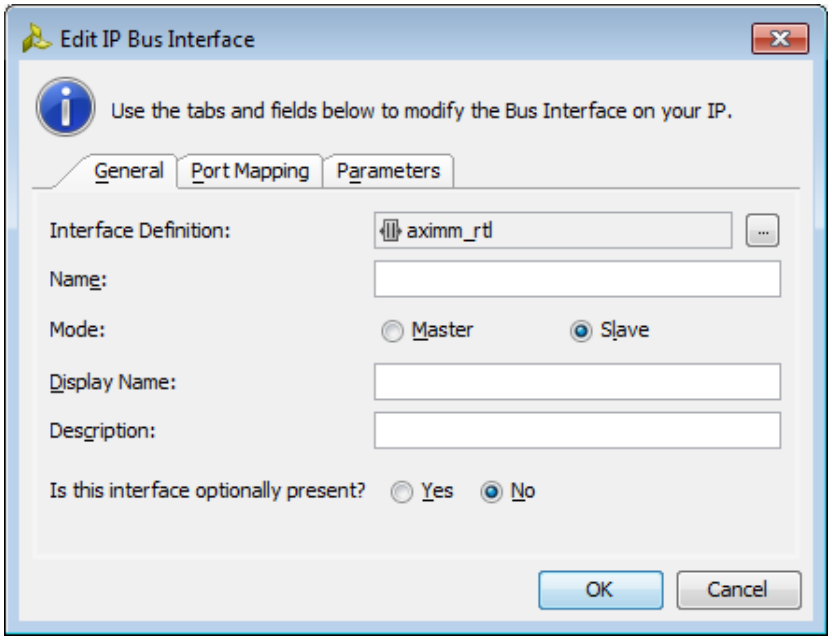


Figure 8-55: **Edit IP Bus Interface Dialog Box**

General Tab

This dialog box opens onto the General tab, which contains the following selections:

- **Interface Definition**: The selected interface to be created. By default, the selected interface is the AXI4 memory-mapped interface (`aximm_rtl`).

  To change the interface, click the ... button at the end of the interface definition input field, which opens the Interface Definition Chooser dialog box, (Figure 8-56, page 117), that lets you select a different interface.

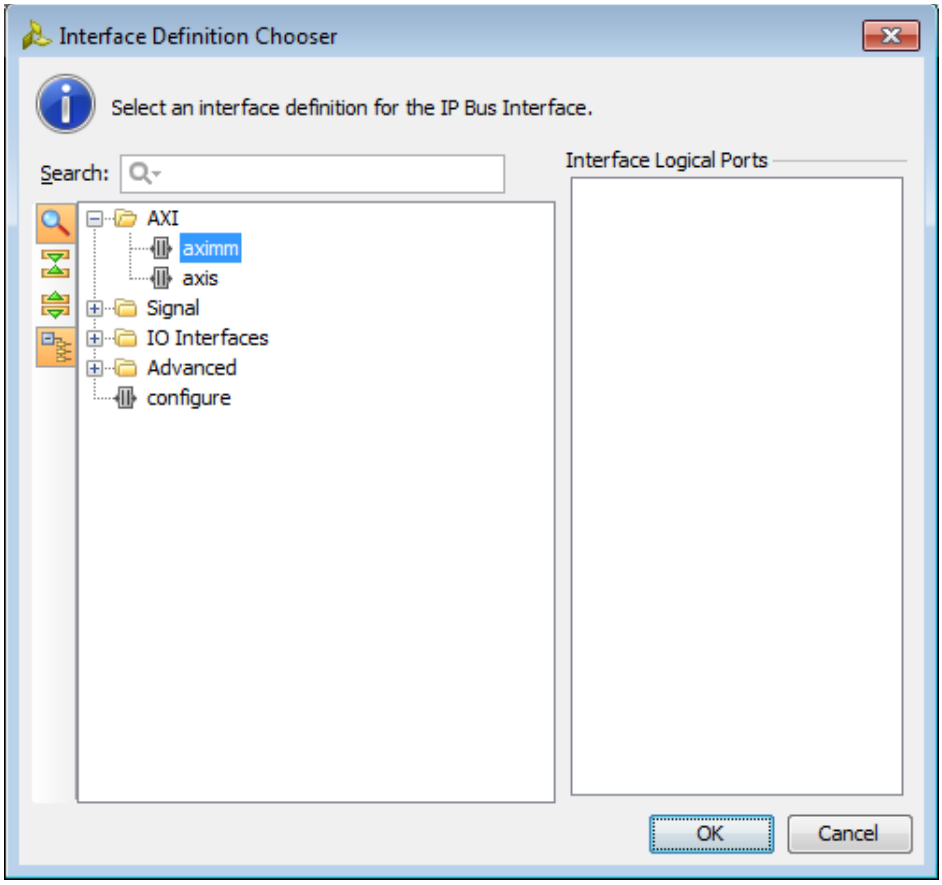


Figure 8-56: **Interface Definition Chooser Dialog Box**

The interfaces are consolidated into the following categories:

- **AXI**: Contains available AXI interfaces.
- **Signal**: Contains available single-bit interfaces.
- **I/O Interfaces**: Contains available external I/O interfaces.
- **Advanced**: Contains available interfaces for advanced users.

After selecting an interface, the Interface Logical Ports dialog box populates the defined ports of the interface (Figure 8-57).

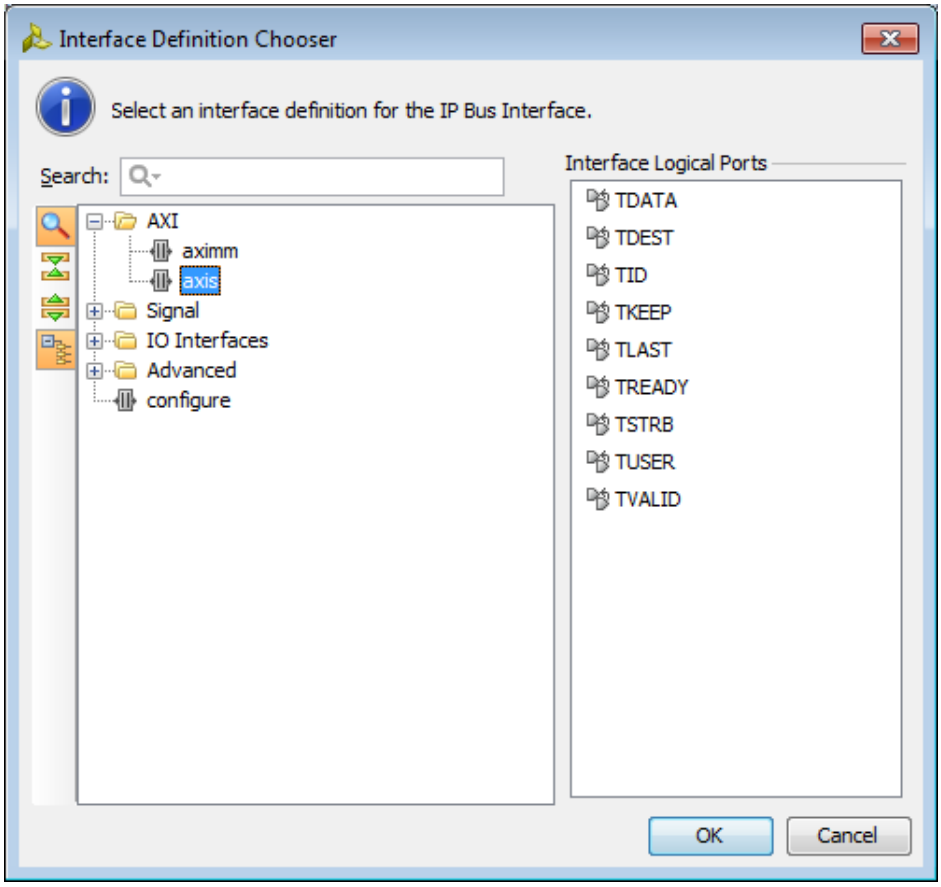


Figure 8-57: **Interface Definition Chooser with Interface Logical Ports Available**

The interface logic ports listed give the naming convention of the respective interface.

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

- **Name**: This is the name of the interface. This name displays for the interface in the customization GUI as a grouping of the mapped ports. This value is required before proceeding to the rest of the Edit IP Bus Interface dialog box tabs.
- **Mode**: Defines the mode of the interface: Master or Slave.
  - Select **Master** if the IP operates as the master for the interface.
  - Select **Slave** if the IP operates as a slave of the interface.
- **Display Name**: The interface display name.
- **Description**: The interface description.

- **Is this interface optionally present?:** Controls if the interface is optionally present.
  - If **Yes**, the enablement expression text box becomes visible in the dialog box (Figure 8-58).

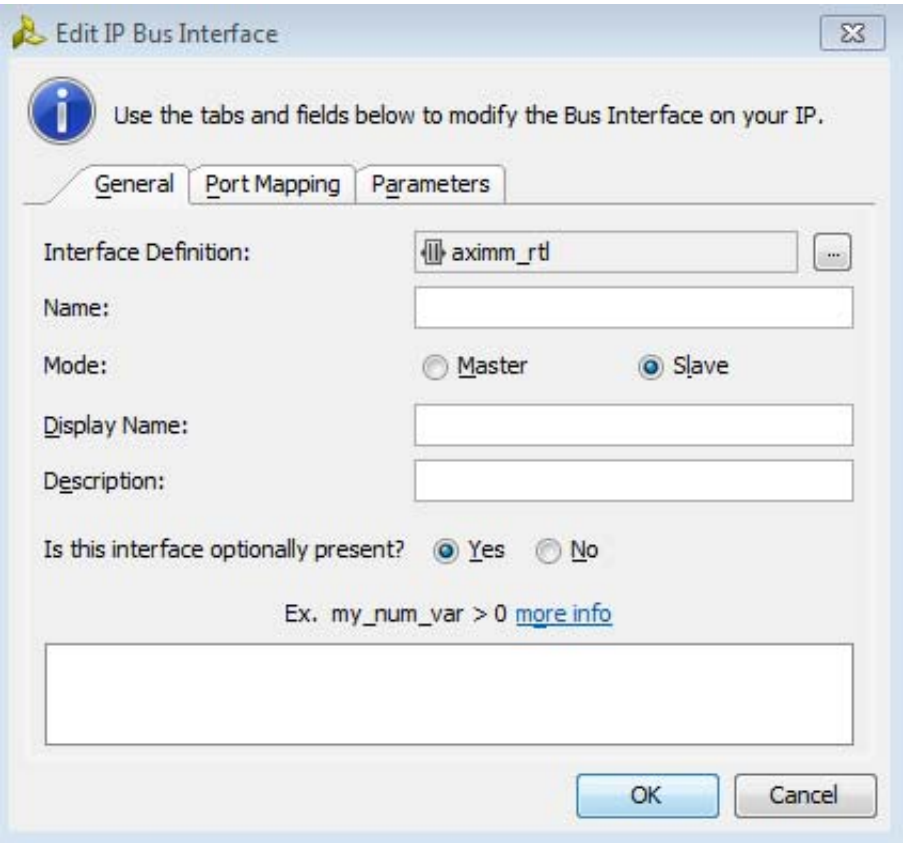


Figure 8-58: **Edit IP Bus Interface with Expression Text Box**

The enablement expression for interfaces uses the supported operators (==, >, <, and !=).

The syntax for the support parameters is the name of the parameter. For example, if the parameter is named `BAUD_RATE`, the enablement expression to ensure the value is 1 is `BAUD_RATE==1`.

Port Mapping Tab

After setting the information in the General tab, select the Port Mapping tab (Figure 8-59).

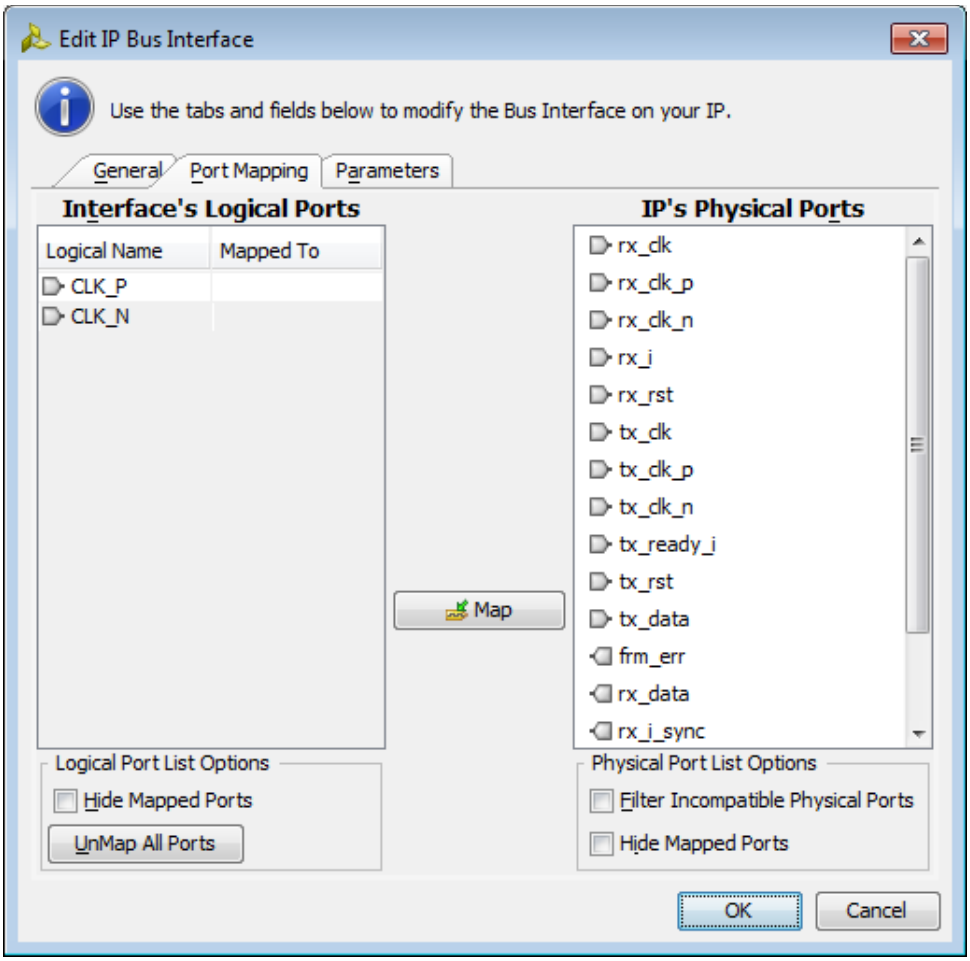


Figure 8-59: **Edit IP Bus Interface Port Mapping Tab**

The Port Mapping tab maps the logical interface ports to the physical ports of the IP.

- The column list on the left contains the list of the logical ports of the interface.
- The column list on the right contains the list of the physical IP ports.

To map the logical ports of the interface to the physical ports of the IP, select the logical port to map in the Logical Ports interface list, then select the corresponding IP port in the Physical Ports list of the IP.

Figure 8-60 shows an example of physical ports on an IP.

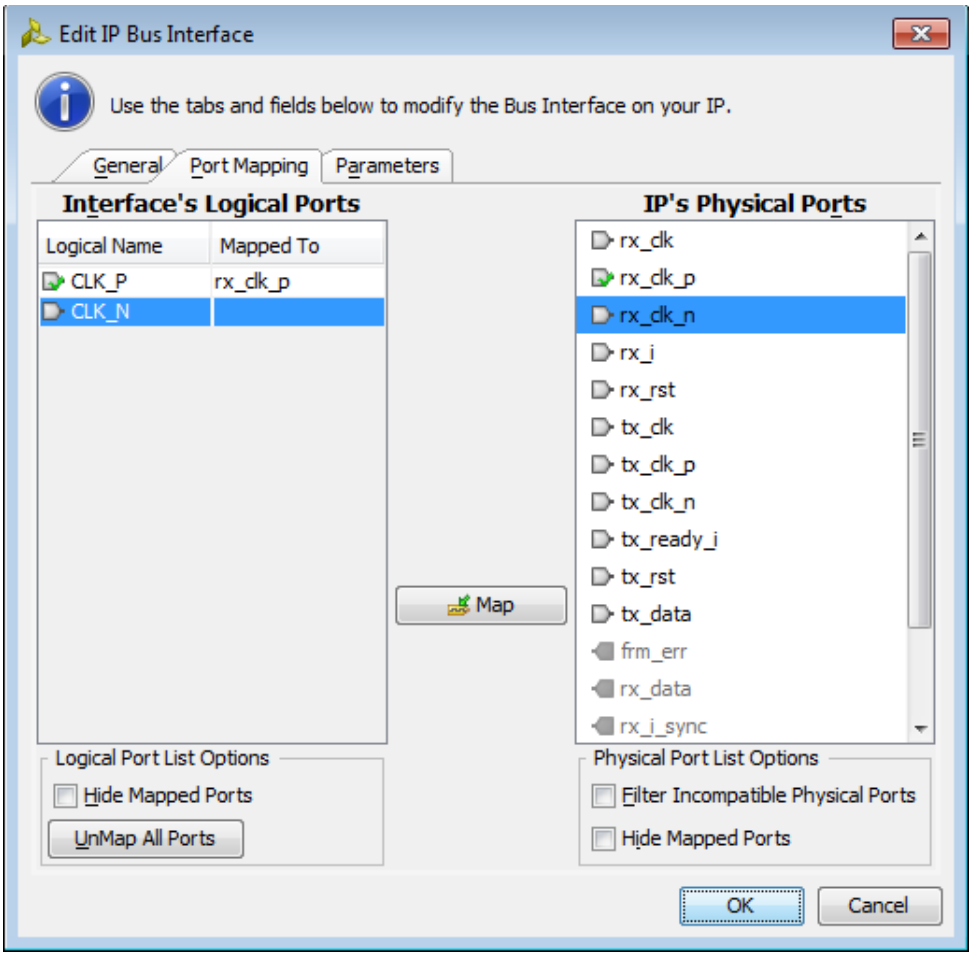


Figure 8-60: **Edit IP Bus Interface with Selected Ports**

After selecting the corresponding logical and physical ports, click the **Map** button in the center of the dialog box. This maps the selected ports and populates the IP physical port to the **Mapped To** column on the Logical Ports interface list.

The options in the dialog box to assist you through the process are, as follows:

- **Logical Port List Options**: The logical port list options are available at the bottom of the Logical Ports list of the interface.
  - **Hide Mapped Ports**: Hides any logical port that are mapped to a physical port of the IP.
  - **UnMap All Ports**: Unmaps all the logical ports of the interface that are mapped to a physical port.

- **Physical Port List Options**: Available at the bottom of the Physical Ports list.
  - **Filter Incompatible Physical Ports**: Filters out any incompatible physical ports.
  - **Hide Mapped Ports**: Hides any physical port that is mapped to a logical port.

Parameters Tab

After mapping the interface ports, select the Parameters tab, (Figure 8-61).

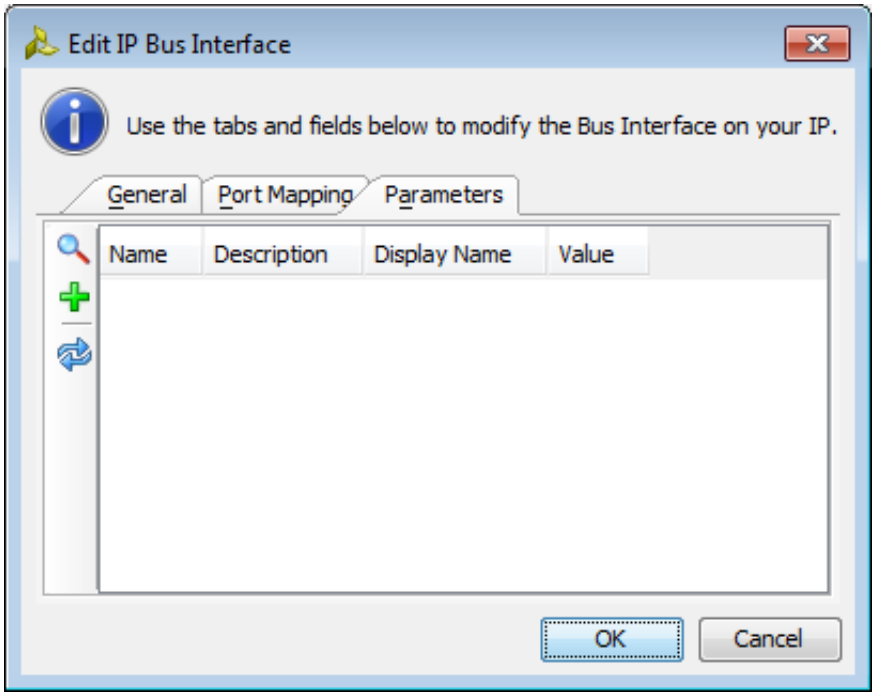


Figure 8-61: **Edit IP Bus Interface – Parameters tab**

The Parameters tab 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 when choosing to add any necessary parameters.

1. To add a recommended parameter, right-click the window, and select **Add Recommended Parameter,** (Figure 8-62, page 123).

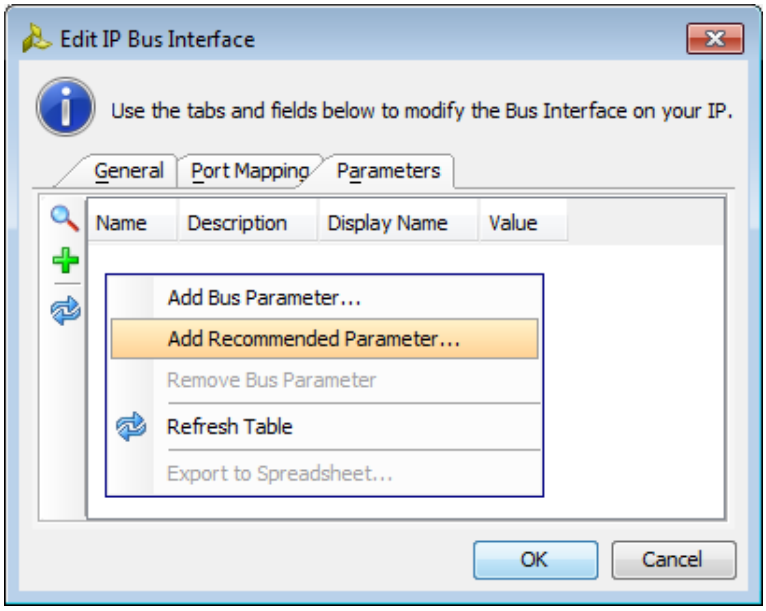


Figure 8-62: **Edit IP Bus Interface – Add Recommended Parameter**

The Add IP Parameter dialog box opens, that contains a list of recommended IP parameters for the respective interface (Figure 8-63).

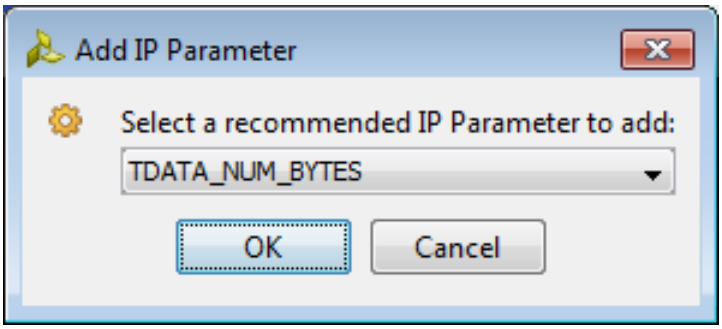


Figure 8-63: **Add IP Parameter: Recommended Parameters**

The Add IP Parameters dialog box gives you a drop-down list of recommended parameters for the interface.

- To add the recommended parameter, select the parameter from the list, and click **OK** to close the dialog box.
- To add a unique parameter for the interface or one not in the recommended list, right-click the window and select **Add Bus Parameter**.

  This option opens a similar Add IP Parameter dialog box, that contains a field to enter a new Bus Parameter name (Figure 8-64).

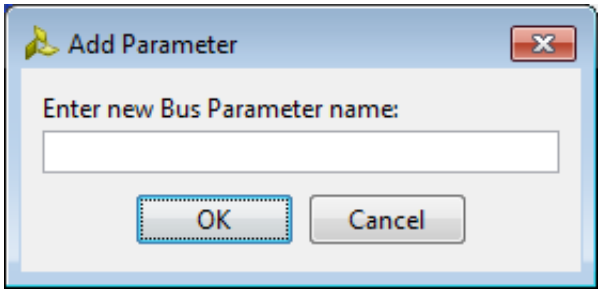


Figure 8-64: **Edit IP Bus Interface: Add Bus Parameter Option**

2. Type the name of the bus parameter, and click **OK**.

The Parameters tab populates with the parameters added from either the recommended list or from the input field.

Figure 8-65 shows an example of a populated parameters tab.

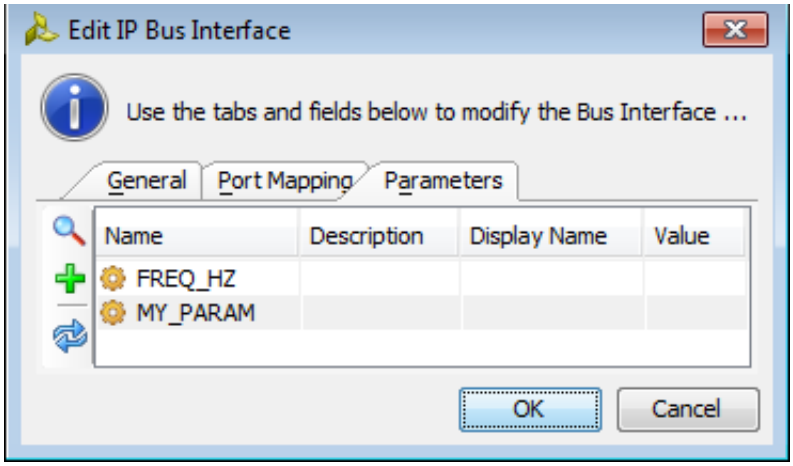


Figure 8-65: **Edit IP Bus Interface: Populated Parameters Tab**

Edit the parameters in the table to populate the data in the available columns: **Description**, **Display Name**, and **Value**.

3. To edit the information, select the cell of the parameter in the column and enter the revised information (Figure 8-66).

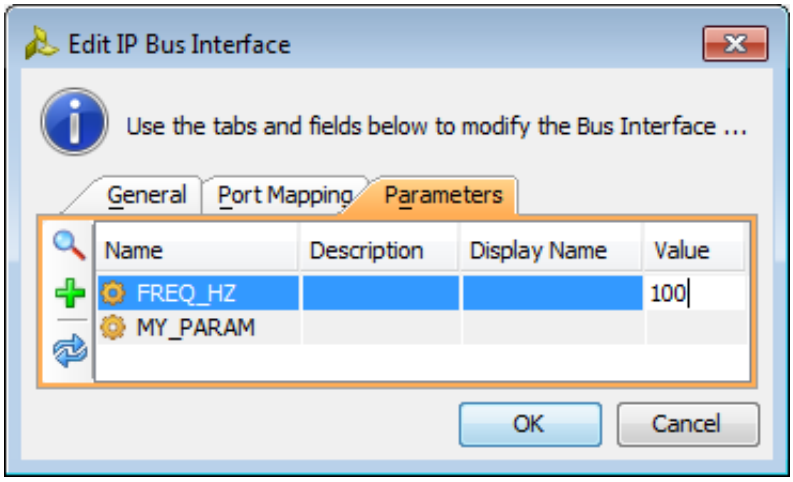


Figure 8-66: **Edit IP Bus Interface: Populated with Parameters**

4. To remove a parameter from the list, select the parameter, right-click and select **Remove Bus Parameter**, (Figure 8-67).

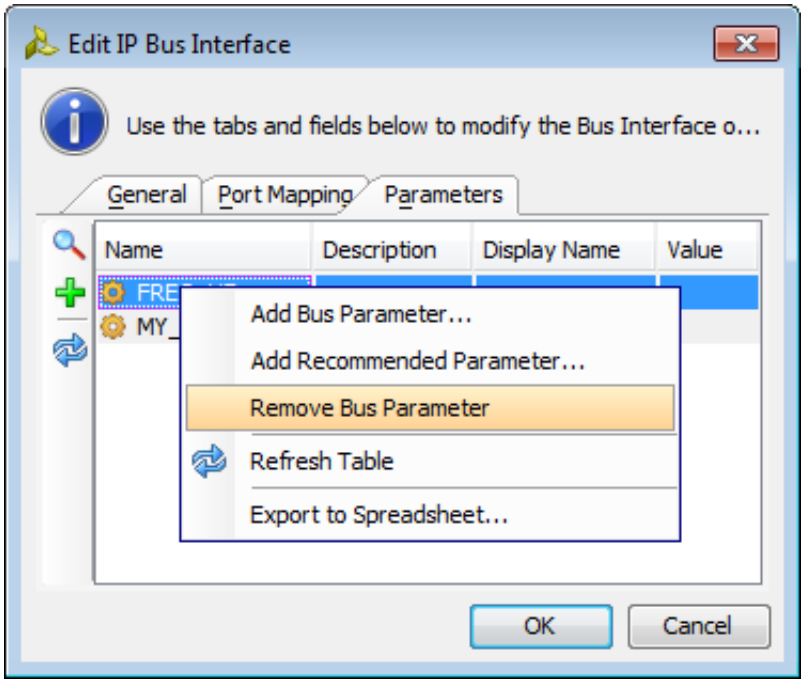


Figure 8-67: **Edit IP Bus Interface: Remove Bus Parameter**

5. After you complete the necessary information for the interface, click **OK**.

The created interface shows in the IP Ports and Interfaces list with the associated ports as children of the interface (Figure 8-68).

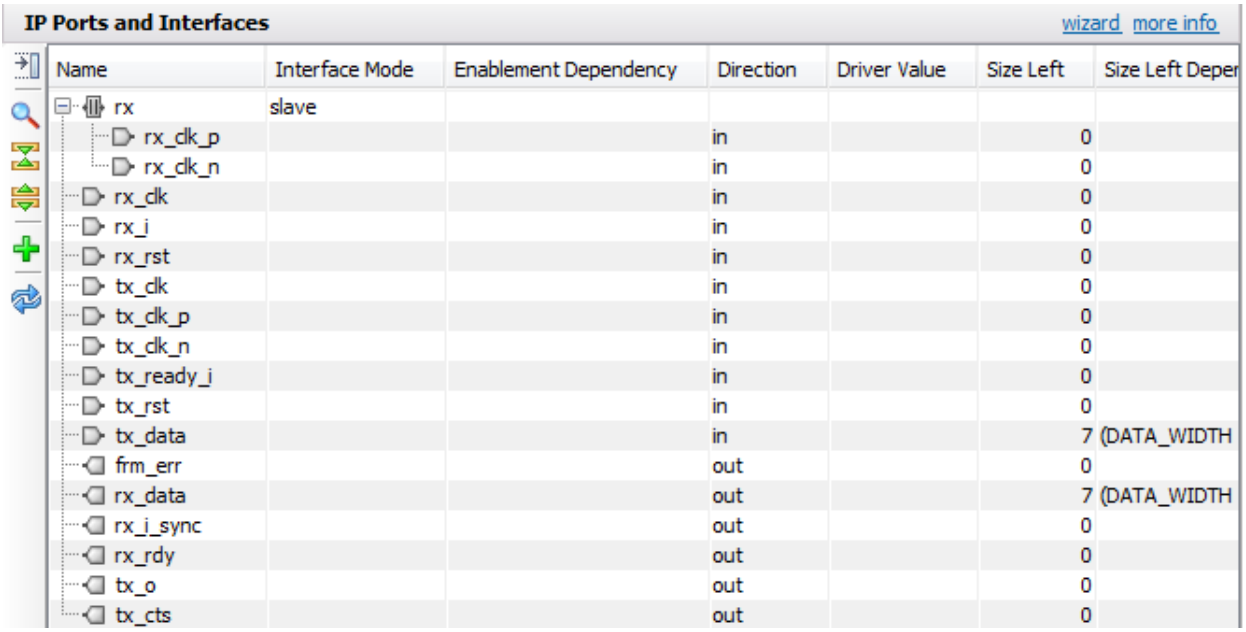


Figure 8-68: **IP Port and Interfaces List with Differential Clock Interface Example**

Auto-Inferring an Interface

The IP packager provides an option to auto-infer bus interfaces, based upon the port defined naming conventions.

RECOMMENDED: *Name the ports exactly as specified in that interface, so the auto inference can correctly identify the bus interface ports. 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 a Bus Interface, page 115*.

---

1. To auto-infer an interface, select the ports of the interface and select **Auto Infer Interface**, (Figure 8-69).

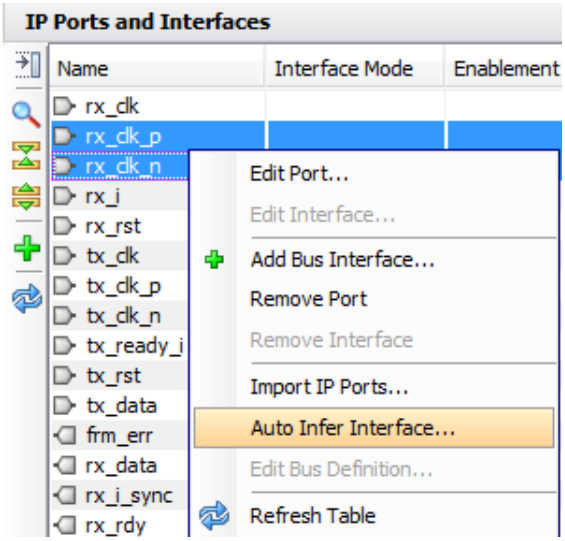


Figure 8-69: **Auto-Infer Interface Option**

The Auto Infer Interface Chooser dialog box opens (Figure 8-70).

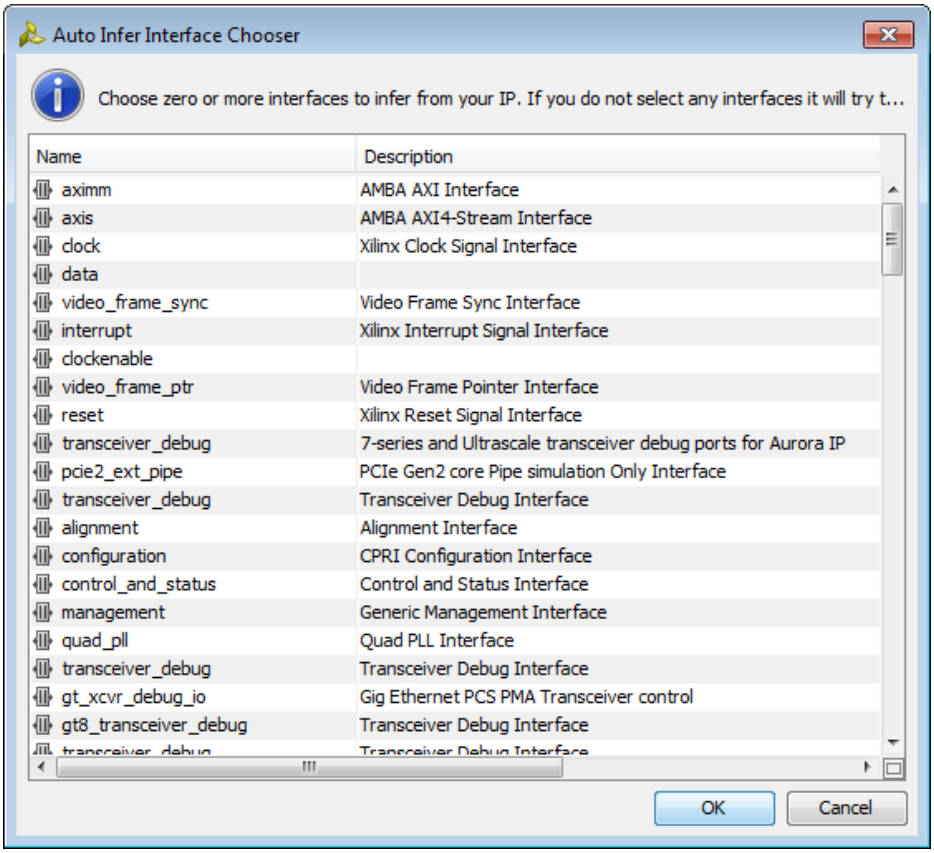


Figure 8-70: **Auto-Infer Interface Chooser Dialog Box**

The Auto Infer Interface Chooser dialog box lets you select the interface for IP packager to attempt to infer. The table contains the name of the interface and the description.

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

   When the IP Packager is unable to infer the interface, the following message displays, (Figure 8-71).

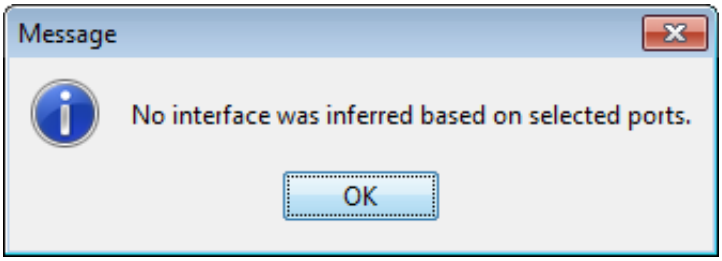


Figure 8-71: **Unsuccessful Auto-Inference of an Interface**

When auto-inference is successful, the Edit IP Bus Interface dialog box opens, populated with information about the interface definition, name, and port mapping (Figure 8-72).

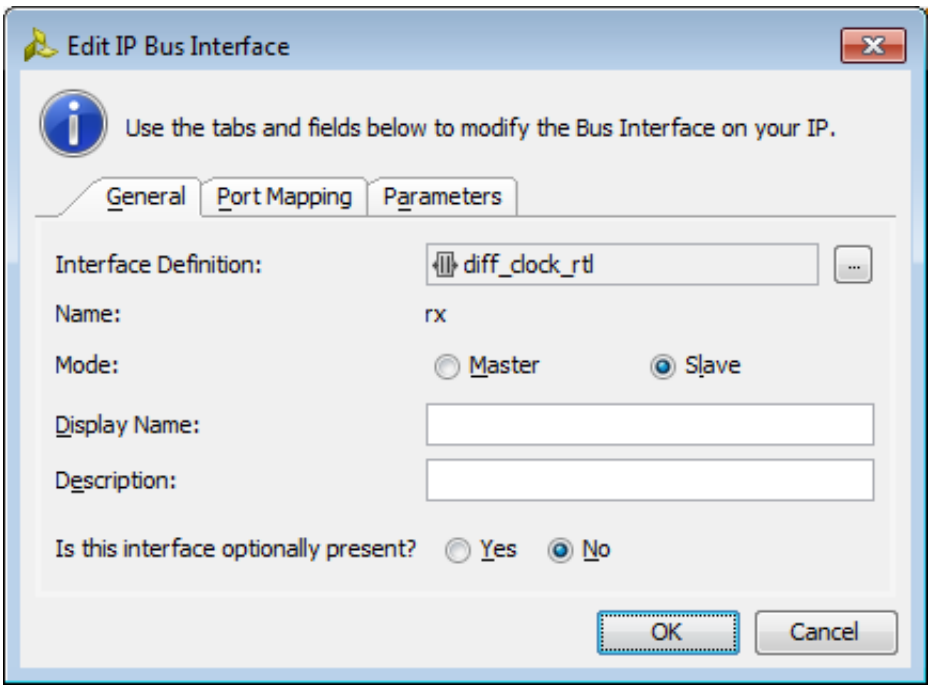


Figure 8-72: **Edit IP Bus Interface: Successful Auto Inference of an Interface**

3. Click **OK**.

For information about the Edit IP Bus Interface dialog box, see Adding a Bus Interface, page 115.

To auto-infer a single bit interface in the Edit IP Ports and Interfaces right-click the port, and use the **Auto Infer Single Bit Interface** option, (Figure 8-73).

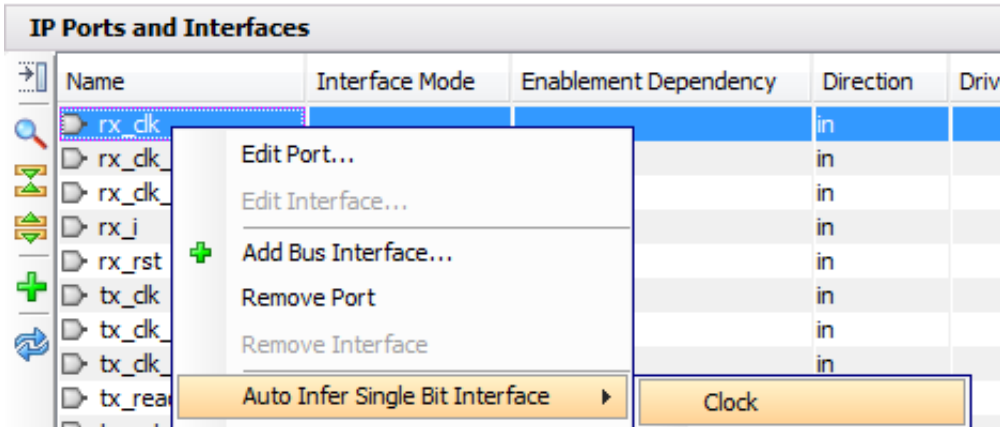


Figure 8-73: **Auto-Infer Single Bit Interface Option**

The single bit interface inference is available for the following interfaces: `Clock`, `Clock Enable`, `Data`, `Interrupt`, `Reset`, `Video Frame Pointer`, and `Video Frame Sync`.

To infer the interface, select the interface from the single bit inference list.

Editing or Removing an Interface

To edit a created interface, select the interface from the list in the IP Ports and Interfaces section, right-click in the interface, and select **Edit Interface** (Figure 8-74).

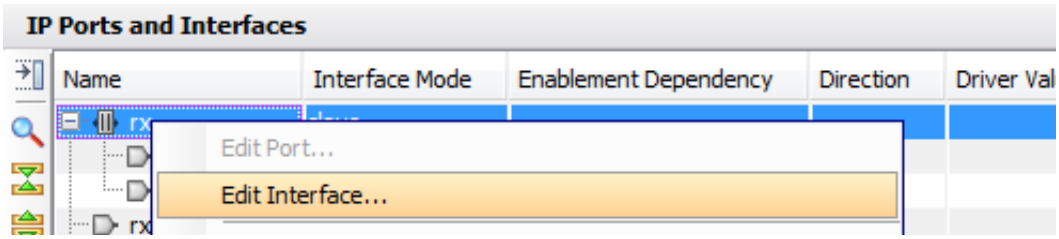


Figure 8-74: **Edit Interface Option**

The Edit IP Bus Interface dialog box opens so you can modify the interface. For information on the Edit IP Bus Interface dialog box, see Adding a Bus Interface, page 115.

To remove the interface, select the interface from the list, right-click and select **Remove Interface** from the context menu.

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

Setting IP Addressing and Memory

The IP Addressing and Memory section, (Figure 8-75), lets you add a memory-map or address space to the IP.

Initially, the Create and Package IP wizard populates the IP Addressing and Memory section, if the Vivado IDE can heuristically determine an address mapping requirement for the interface.

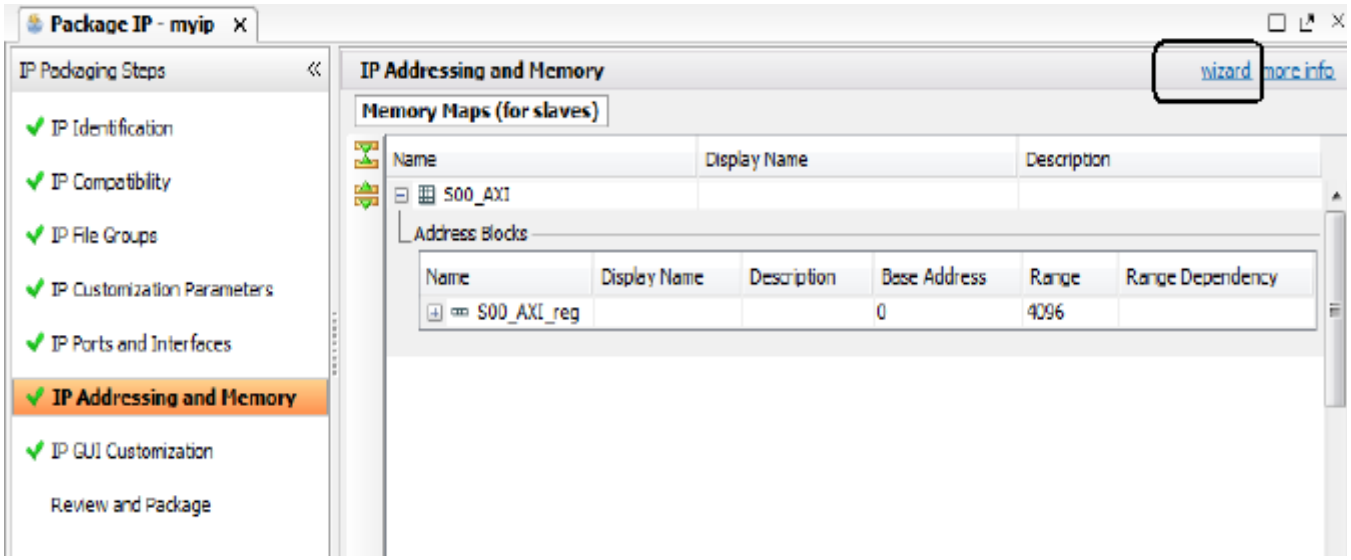


Figure 8-75: **IP Addressing and Memory**

Use the IP Addressing and Memory wizard to identify the IP bus interfaces that require a memory-map or address space.

1. Open the IP Addressing and Memory wizard by clicking the wizard hyperlink, circled in Figure 8-75.

   The IP Addressing and Memory Configuration page of the Wizard (Figure 8-76) guides you through adding an address space or memory-map for any associated bus interfaces.

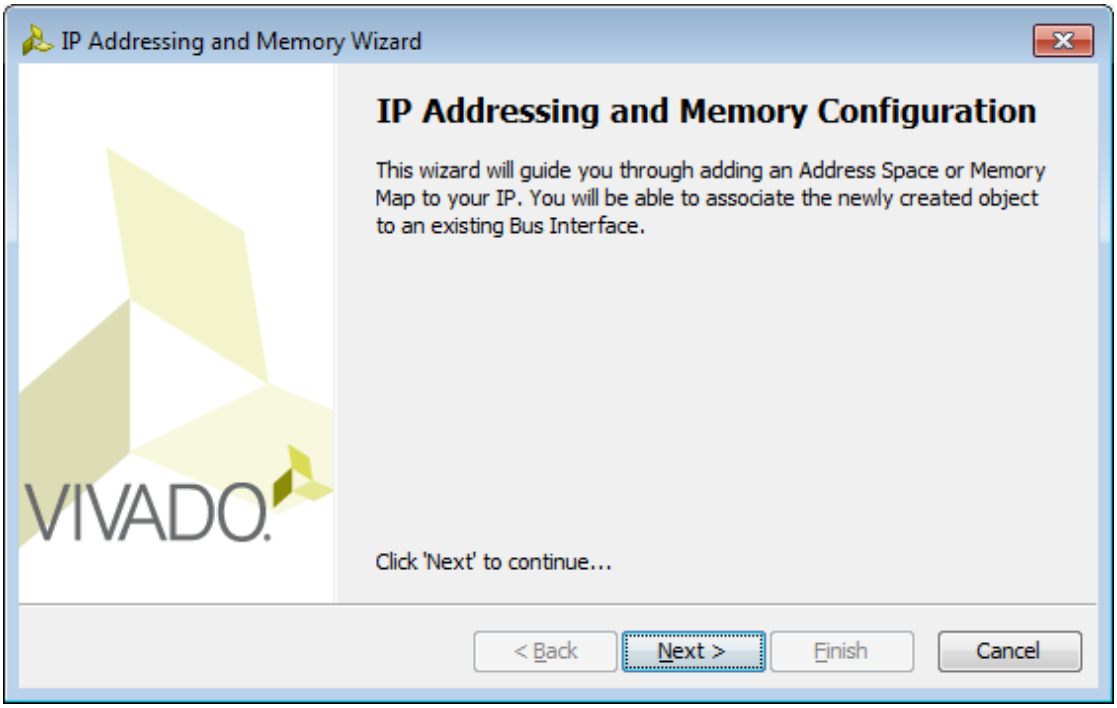


Figure 8-76: **IP Addressing and Memory Configuration Wizard**

2. Open the Choose IP Interface page of the wizard by clicking **Next**.

   The Choose IP Interface page, (Figure 8-77), lists the available bus interfaces to associate to a memory-map or address space. Any bus interface that has already been mapped will not be present in the list of available interfaces.

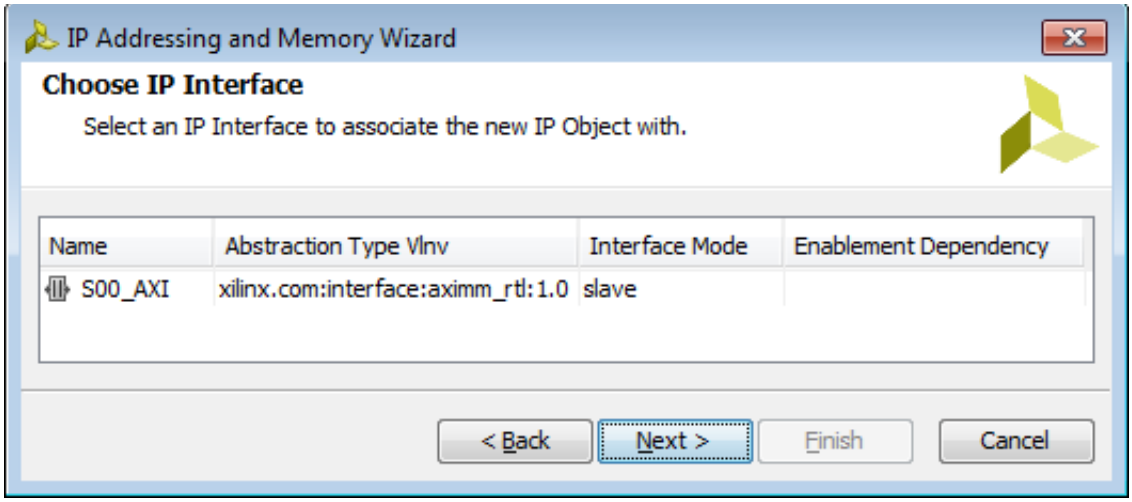


Figure 8-77: **Choose IP Interface**

The interface provides the following information:

- **Name:** The interface name.
- **Abstraction Type VLNV**: **Vendor**, **Library**, **Name**, and **Version** of the interface.
- **Interface Mode:** The interface mode (**Master** or **Slave**).
- **Enablement Dependency:** The enablement expression that defines the interface visibility.

3. Select the required interface, and click **Next**.

   The Choose IP Object Name page opens, (Figure 8-78).

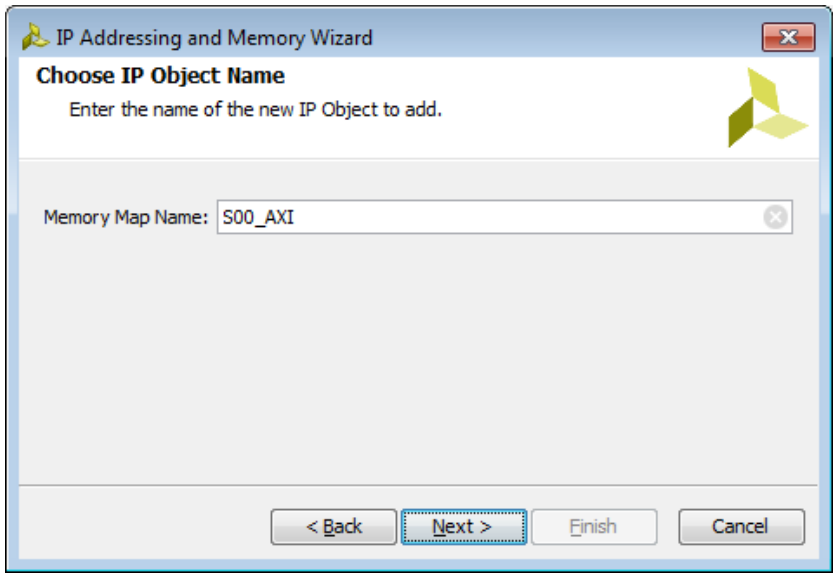


Figure 8-78: **Choose IP Object Name**

   By default, the name of the memory-map is the name of the interface.

4. Set the name and, click **Next**.

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

The IP Addressing and Memory Wizard Summary page opens (Figure 8-79, page 133). This describes the name of the new memory-map as well as to which interface the memory-map references.

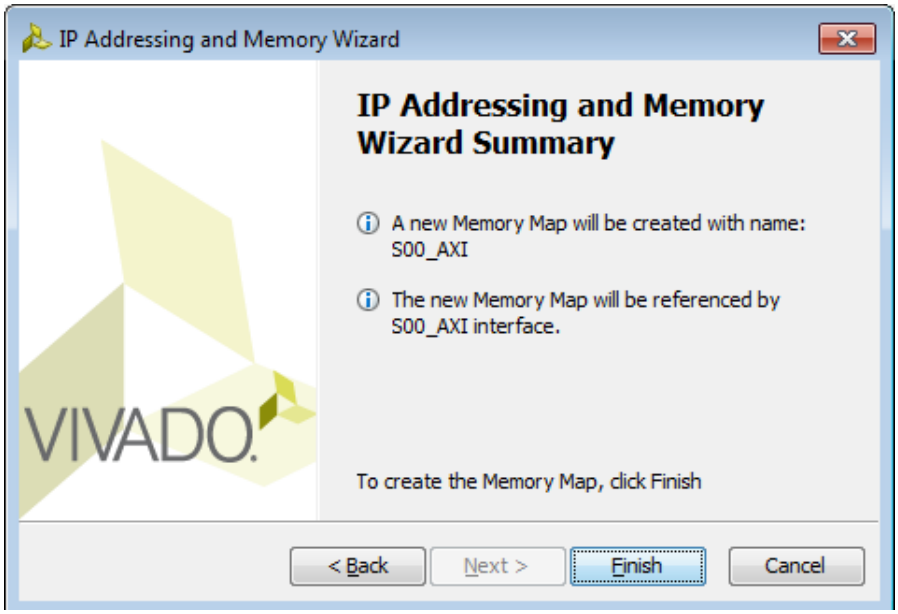


Figure 8-79: **IP Addressing and Memory Wizard Summary**

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

Adding an Address Block

To add an address block to a memory-map:

1. Right-click the selected memory-map and click **Add Address Block** (Figure 8-80).

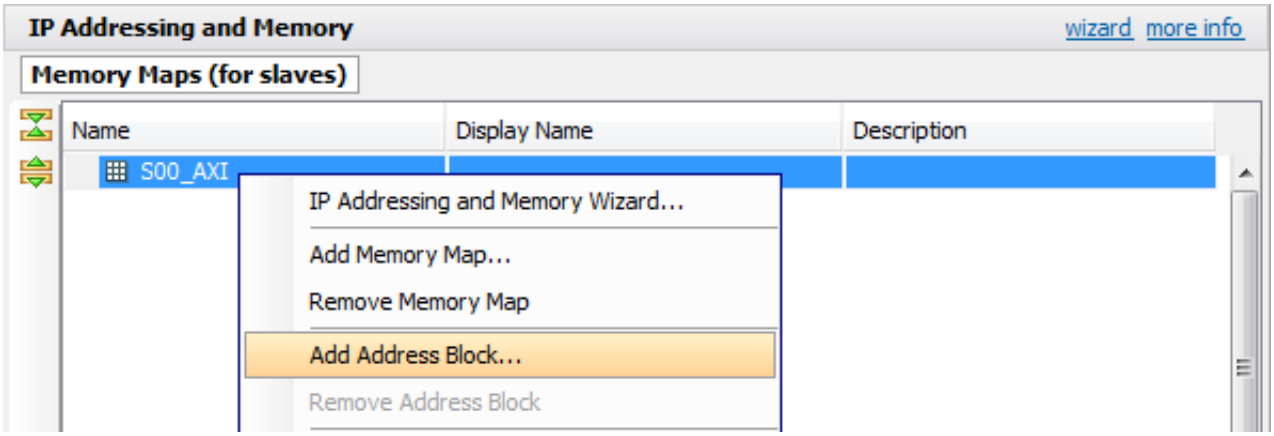


Figure 8-80: **Add Address Block Option**

The Add Address Block dialog box opens (Figure 8-81).

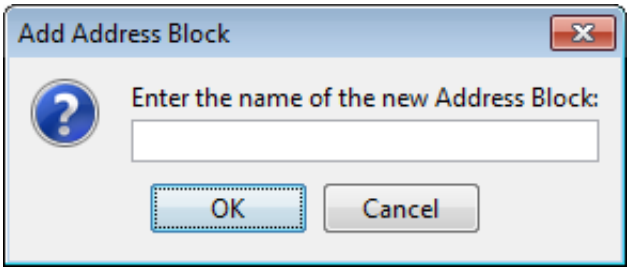


Figure 8-81: Add Address Block Dialog

2. Enter the name of the new address block, and click **OK.**

   The selected memory-map in the IP Addressing and Memory section now contains a child address block section with the newly-created address block (Figure 8-82).

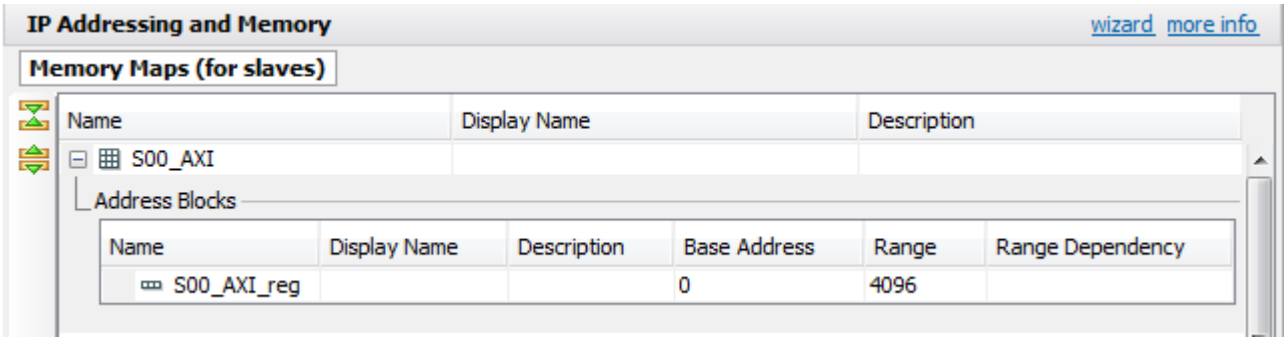


Figure 8-82: **Memory Map with Address Block**

The address block contains the following columns:

- **Name**: address 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, if necessary.

You can edit the parameters in the table and populate that data into the available columns.

3. Select the cell of the address block in the column and enter the information.

IP GUI Customization

The IP GUI Customization section, (Figure 8-83), provides a preview of the GUI customization layout of the IP.

The IP GUI Customization is generated based on the parameters defined in the top-level HDL source file. After you set up the parameters for the IP, you can customize the IP GUI from this option to change how you can interact with the IP representation.

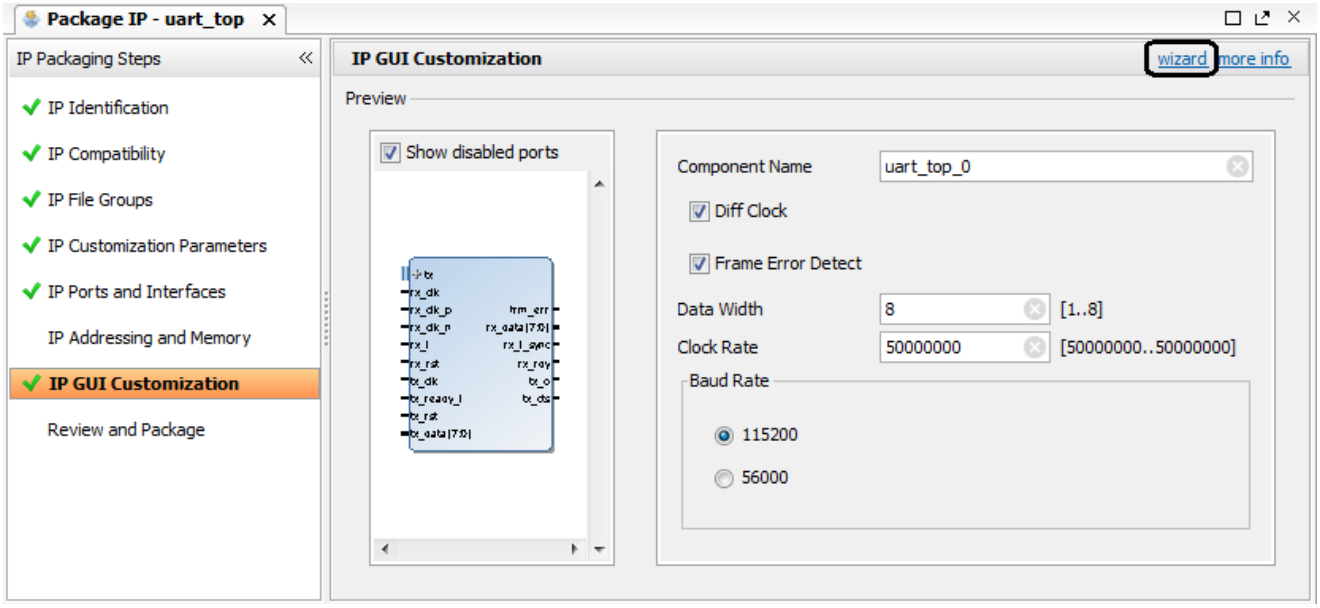


Figure 8-83: **Example IP GUI Customization Window**

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

To customize the GUI, use the wizard IP GUI Customization wizard the display of the viewable parameters. You can access the wizard from the IP GUI Customization header as outlined in Figure 8-83.

1. Click **wizard** to open the IP GUI Customization wizard dialog box, which is shown in Figure 8-84, page 136.

   The IP GUI Customization Wizard dialog box opens, and guides you through the process of organizing the parameters into various pages and groups.

Figure 8-84: **IP GUI Customization Wizard**

Two options exist:

- **Guide me through creating pages and groups**: Has the wizard aid in the creation of pages and groups for the customization parameters of the IP.
- **Jump straight to finish page:** Bypasses the wizard and jumps to the final finish page to regenerate the IP GUI files with the previous settings. The wizard bypasses all the steps and moves to the IP Customization Completion Page.

2. Select **Guide me through creating pages and groups**, and click **Next**.

   The New Configuration Page Creation dialog box opens (Figure 8-85, page 137) The IP GUI Customization has, at minimum, one page of parameter customization. This dialog box lets you control which parameters exist on the first page.

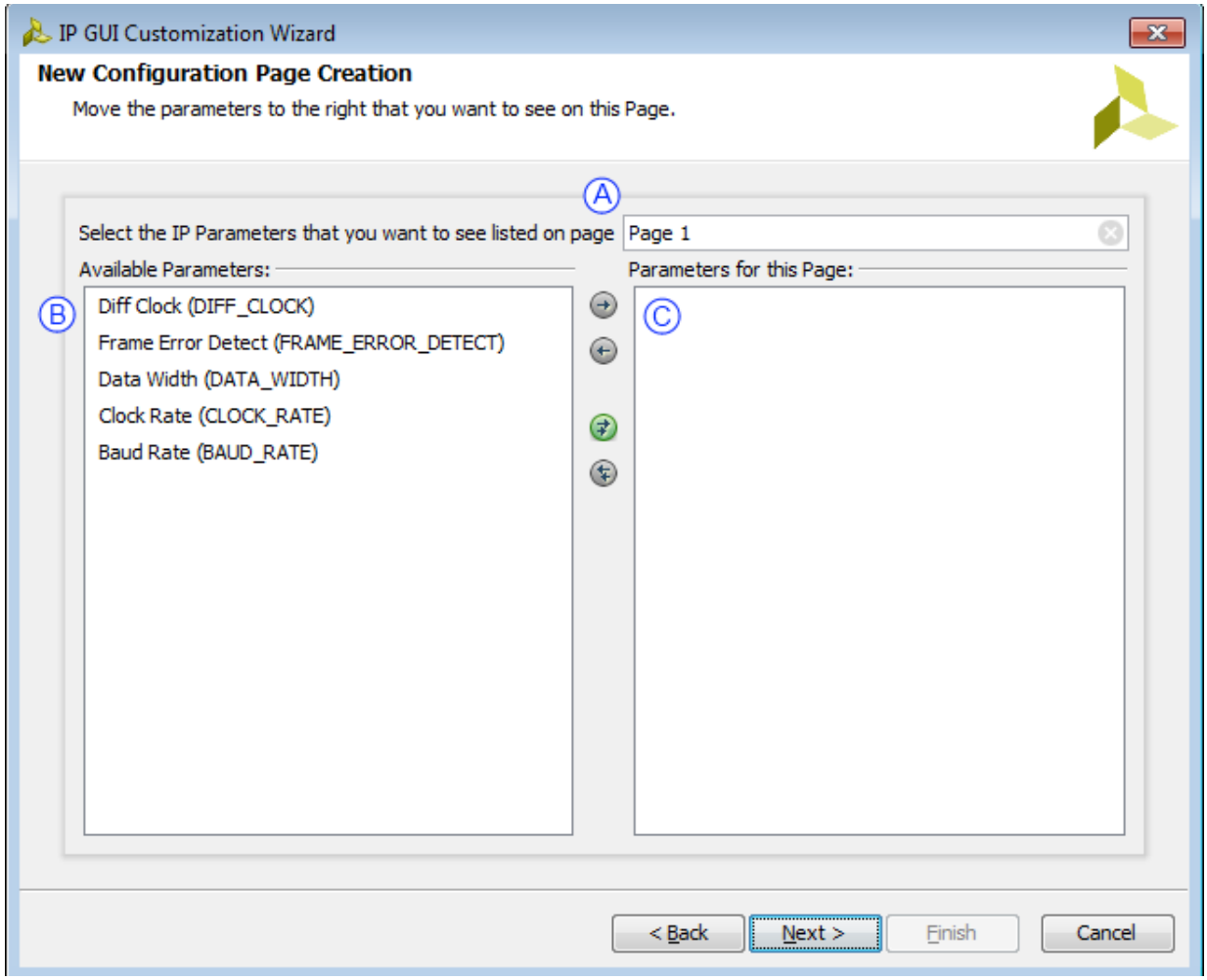


Figure 8-85: **New Configuration Page Creation**

The New Configuration Page Creation dialog box provides three sections:

- By default, the IP Parameters pages are numbered with an incremental count of the customization pages. You can rename a page using the input text field box, as shown in section (A) of Figure 8-85.
- Section (B) shows the list of available parameters within the customization GUI. If a parameter was on a previous page, it is not shown as available in the list. The parameters show with the display name of the parameter first, followed by the parenthesized parameter name.
- Section (C) of the dialog box pane lets you control the parameters to use for the respective page.

  Use the direction buttons in the center panel to control the parameter location by page. The icons change color from gray to green when the you can use the operation that the icon represents.

1. Move parameters as follows:
   - Select an available parameter from section (A) and click the button to move the parameter from the list on the left to the respective page list in section (B).

     You can also select multiple parameters and move them.
   - Move a parameter back from the respective page list to make the parameter available elsewhere in the customization GUI by selecting the parameter in the respective page list and clicking the button.
   - Move all the parameters from available list to the respective page list or back to the available parameter list by clicking the button.
2. After selecting and placing the parameters on the page, click **Next**.

The Group and Parameter Ordering Page (Figure 8-86) controls the order in which the parameters display in the customization GUI, and lets you group and arrange parameters as necessary.

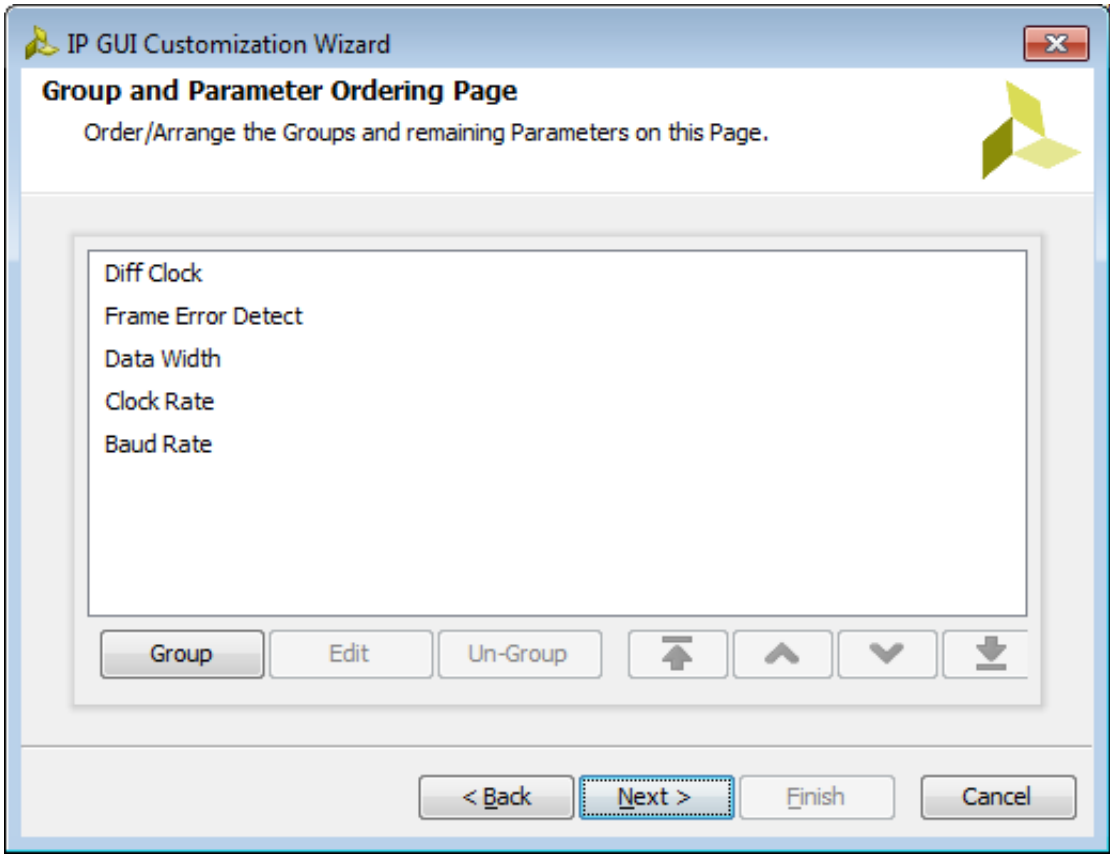


Figure 8-86: **Group and Parameter Ordering Page**

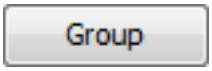
The arrangement of the parameters in the list corresponds to their arrangement in the customization GUI.

Select the parameter you would like to move using the controls on the right bottom of the page. Table 8-4 describes the controls.

Table 8-4: **Group and IP Ordering Page**

Control	Action
	Move the selected parameter to the top of list
	Move the selected parameter up by one
	Move the selected parameter down by one
	Move the selected parameter to the bottom of the list

You can also group certain parameters together using the **Group** button. You can group selected parameters together to organize them separately from the other parameters in the customization GUI.



3. Click **Group** to open the New Group Creation Page (Figure 8-87).

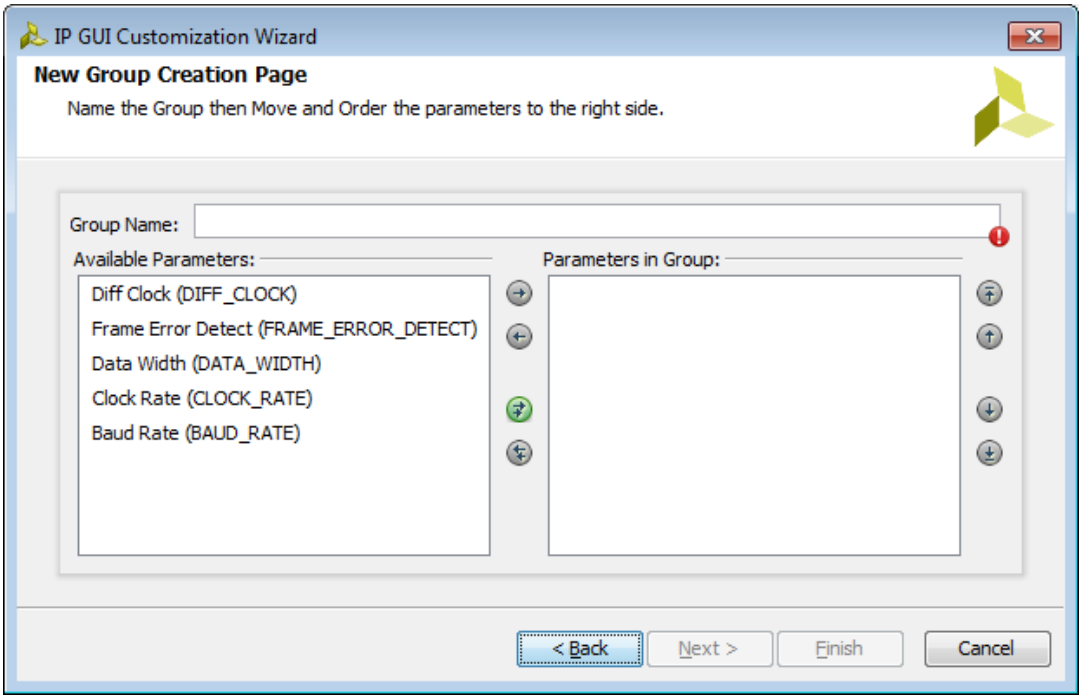


Figure 8-87: **New Group Creation Page**

The New Group Creation Page lets you name the group and control which of the available parameters on the selected page will move into the group.

To use the New Group Creation Page to create groups:

1. Type in a name for the group in the **Group Name** field.

   This creates the display name for the group as seen in the customization GUI.

   You can move the parameters between the two lists using the same control buttons as the New Configuration Page Creation dialog box.

2. Use the direction buttons in the center panel to control the what parameters go into a group. The icons change color from gray to green when the you can use the operation that the icon represents. Move parameters as follows:
   - From the parameter list to the group list, click the button. You can also select multiple parameters and move them.
   - From the group list to the parameter list from the group list, click the button.
   - All the parameters from the available list to the respective page list, click the button.
   - All the parameters from the respective page list to the back to the parameter list click the button.
3. Use the up and down buttons as follows:
   - Move a parameter one position by selecting the parameter and clicking the button.
   - Move a parameter one position down by selecting the parameter and clicking the button.
   - Move a parameter to the top of the list by selecting the parameter and clicking the button.
   - Move a parameter to the bottom of the list by selecting the parameter and clicking the button.
4. Finish group creation by clicking **Next**.

   The wizard returns to the Group and Parameter Ordering Page. The parameter list displays the newly created group or groups with the suffix **(GROUP)** to differentiate from parameters in the list, (Figure 8-88, page 141).

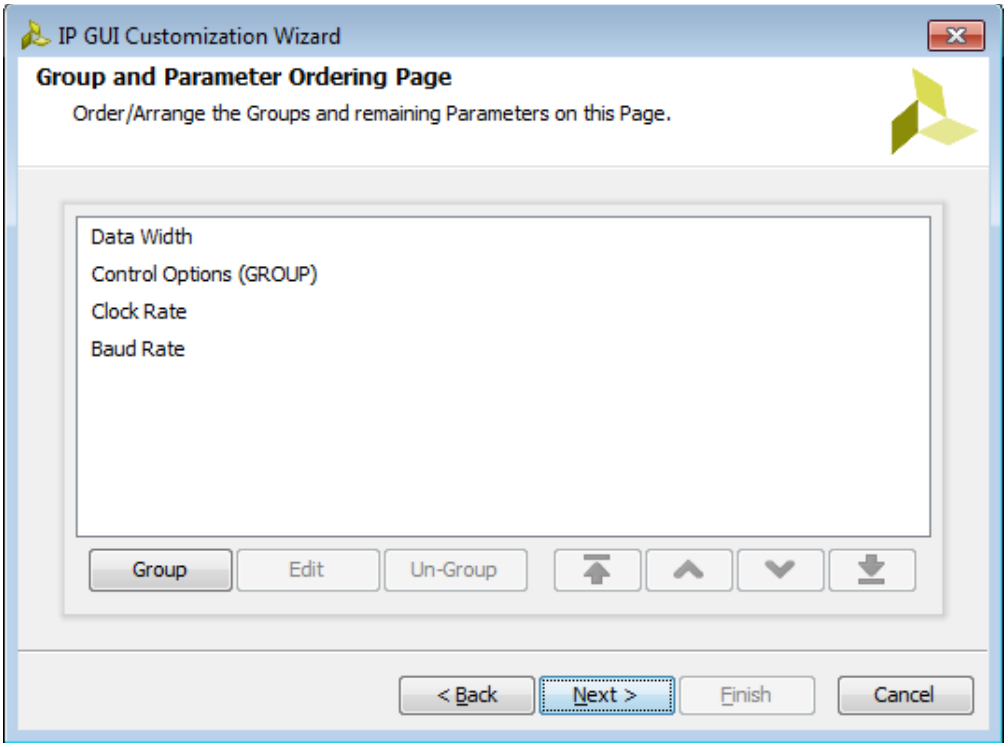
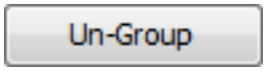


Figure 8-88: **Group and Parameter Ordering Page with Groups**

To edit an existing group, select the group in the Group and Parameters Ordering Page and select the **Edit** button to reopen the New Group Creation Page dialog box for modifications.

To ungroup parameters, select a group and click the **Un-Group** button. This removes the group and adds the parameter back to the bottom of the ordering list.



5. Click **Next**.

The Customization Page Summary page opens (Figure 8-89, page 142).

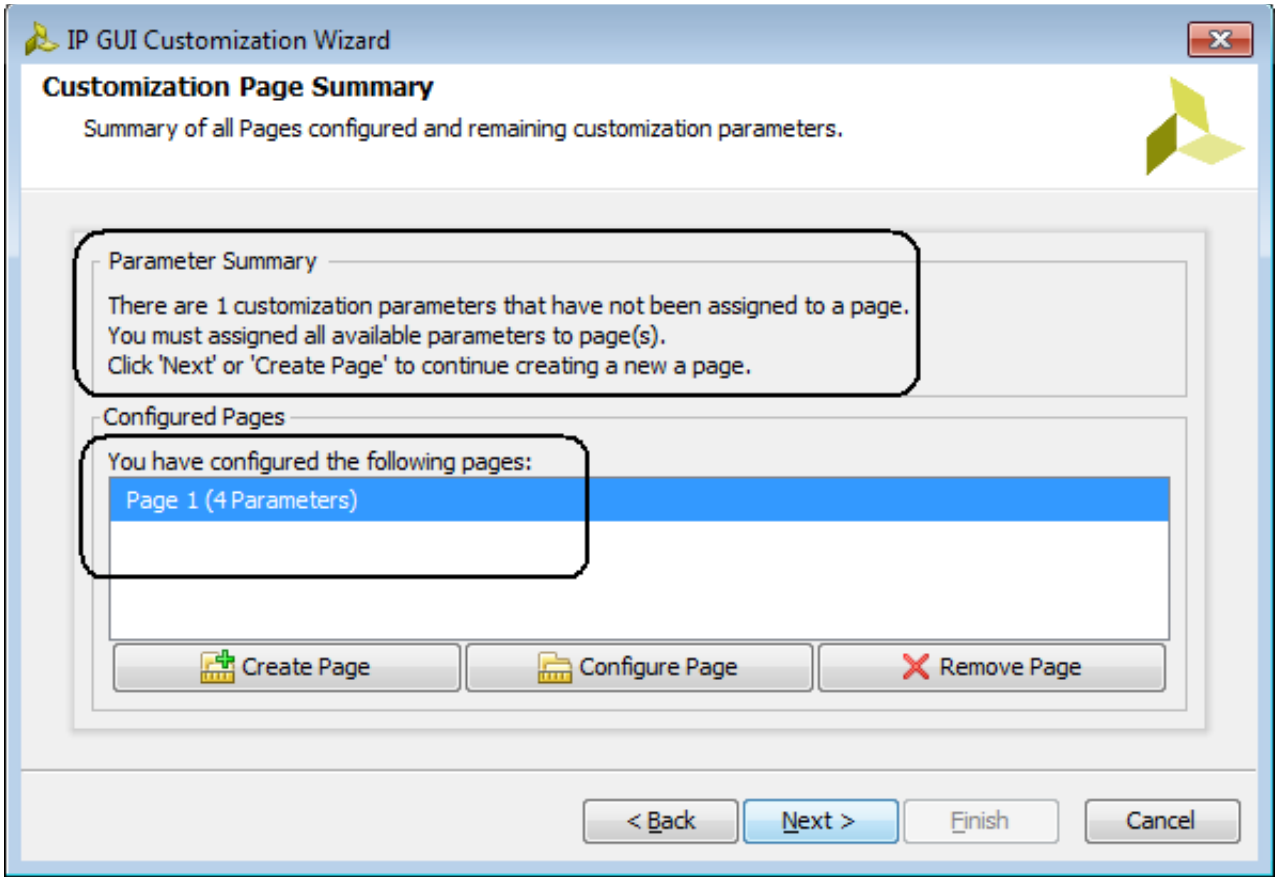
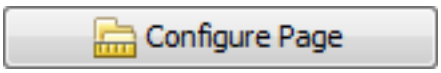


Figure 8-89: **Customization Summary Page with Unassigned Parameter**

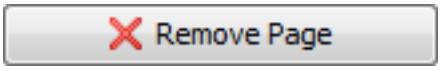
The Customization Page Summary, shows the summary of all the configured pages and any remaining parameters that are not assigned to a specific page. Figure 8-89 shows how the dialog box looks with a single page and at least one parameter unassigned.

- To create a new page to display parameters, click the **Create Page** button. to open the New Configuration Page Creation dialog box.

- To edit an already configured page, select the respective page from the Configured Pages list and click the **Configure Page** button to open the New Configuration Page Creation dialog box with the previous configuration settings allowing additional modifications.



- To remove a page from the Configured Pages list and un-assign the associated parameters associated, click the **Remove Page** button.



After you have assigned the parameters to the pages where you want them to show, the Parameter Summary section of the dialog box displays that you are finished assigning parameters (Figure 8-90).

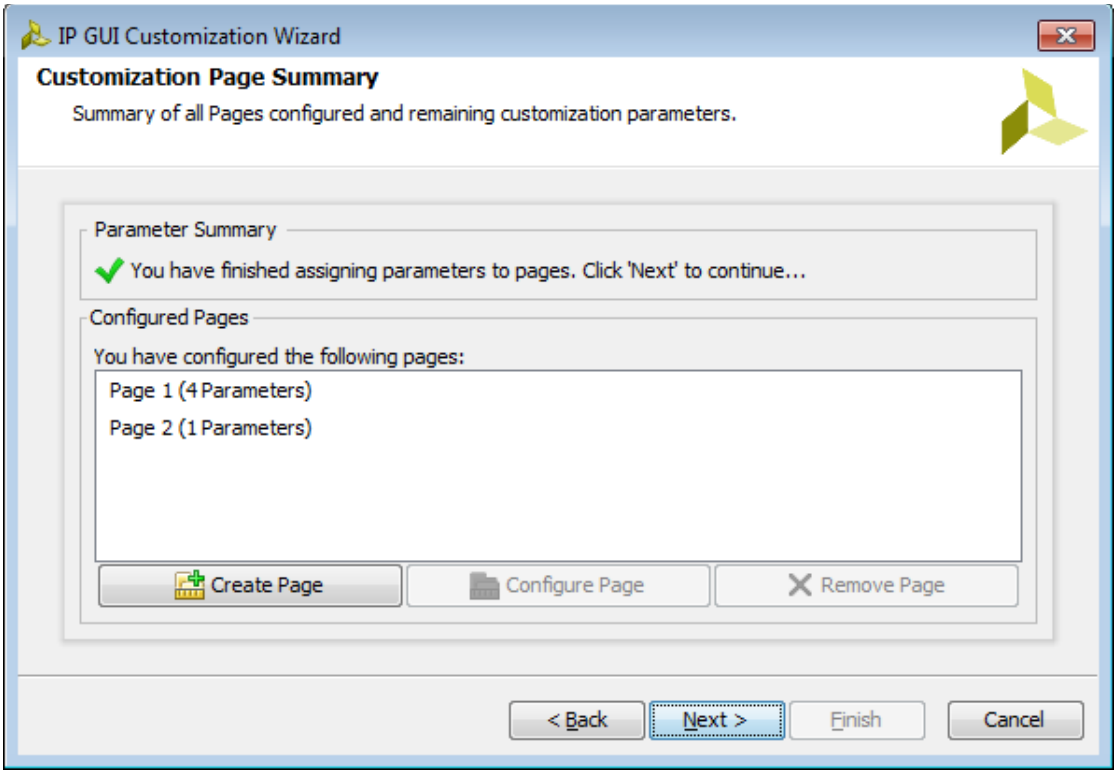


Figure 8-90: **Customization Page Summary with Completed Parameter Assignment**

6. Click **Next**

   The IP Customization Completion Page opens for you to confirm the changes you made to the customization GUI layout, and gives you the option to regenerate the IP GUI files (Figure 8-91, page 144).

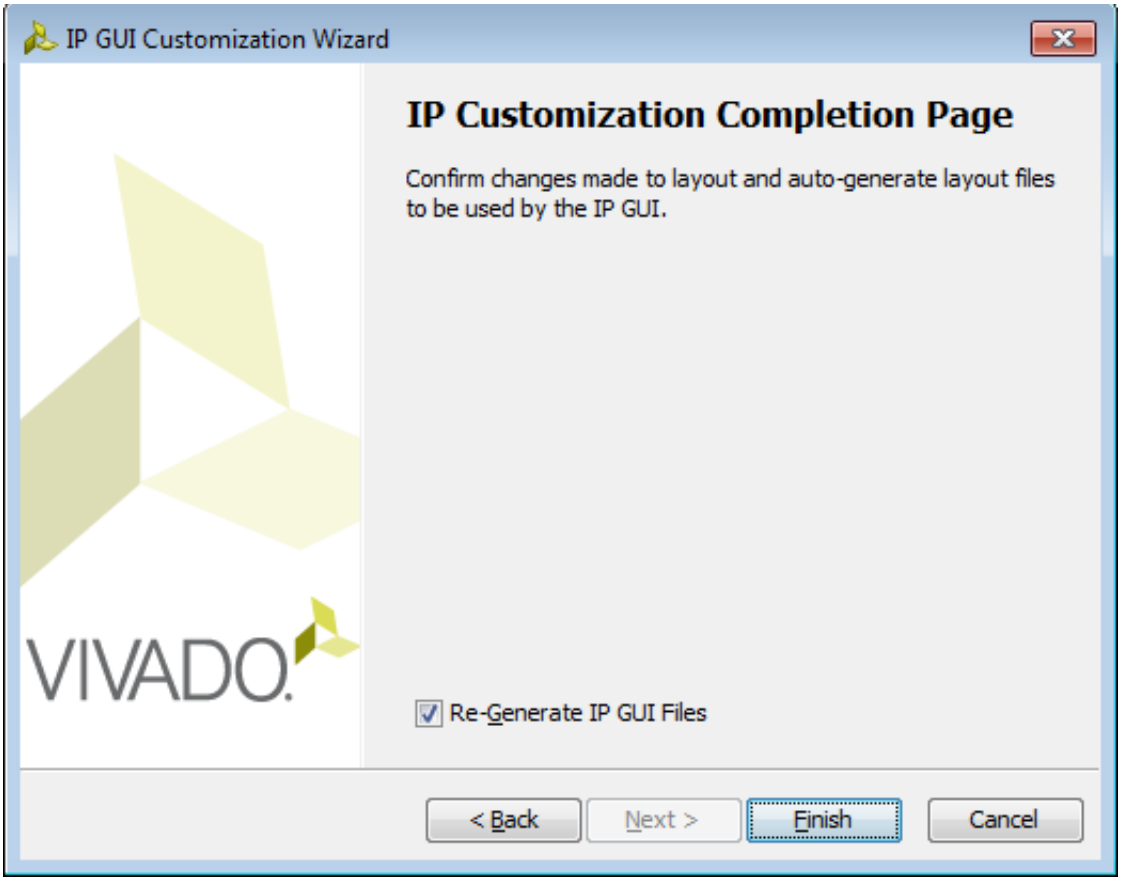


Figure 8-91: **IP Customization Completion Page**

- Check the **Re-Generate IP GUI Files** to have the wizard regenerate all the customization files required to display the IP Customization GUI for the IP.
- If you uncheck the option to regenerate the IP GUI files the wizard still confirms the changes in the IP packager, but does not update the IP GUI files to properly display the changes.

7. To complete the wizard, click **Finish**.

Review and Package

The Review and Package section, (Figure 8-92), provides a summary of the IP, information about the settings you have for **After Packaging**, and a button so you can repackage the IP, if necessary.

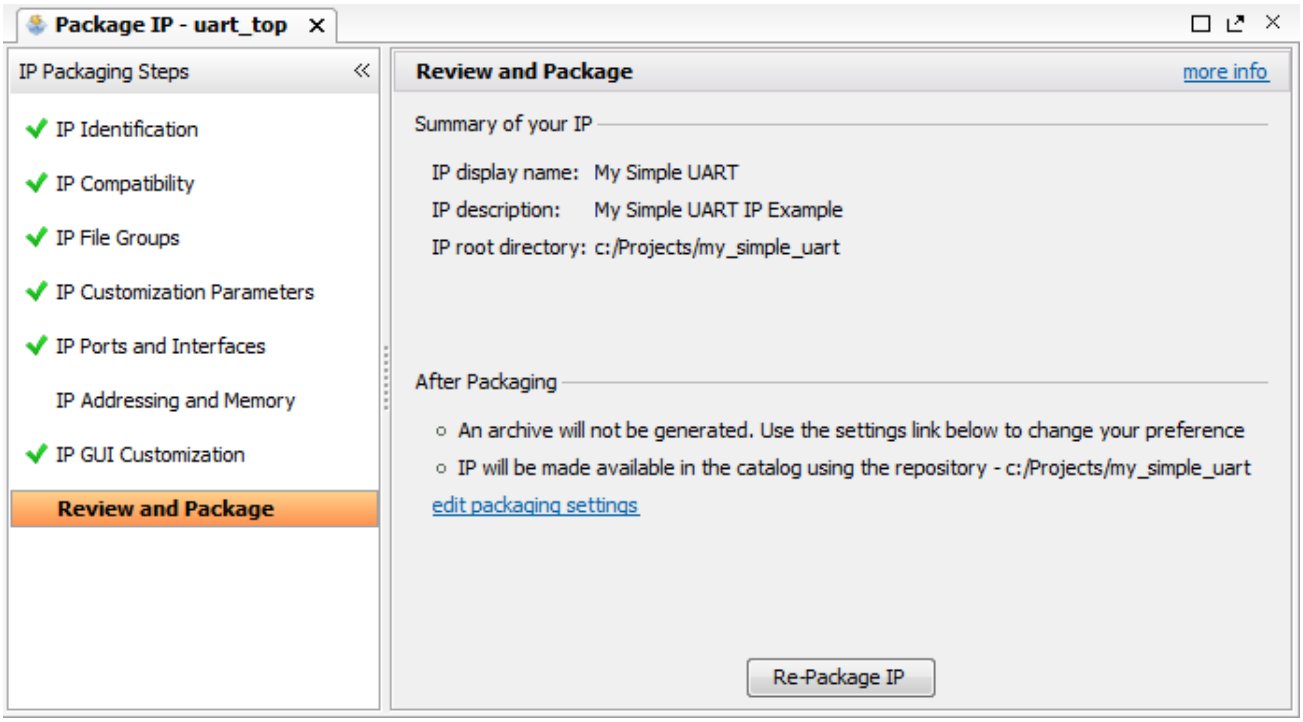


Figure 8-92: **Review and Package**

The Review and Package section information is a summarization from the IP Identification section, as follows:

- **IP display name**: Name that displays in the Vivado IP Catalog.
- **IP description**: Description of the IP that displays in the Vivado IP Catalog.
- **IP root directory**: Working directory of the packaged IP. The directory controls both the location of the input and output files of the IP.

To change the information, make the modifications in the IP Identification section of the IP Packager, as described in IP Identification, page 82.

The **After Packaging** section contains information about what the IP and Vivado tools actions are after you have completed packaging.

The information that describes how the packager behaves based on the IP Packager Settings, which are described in the Project Settings, are, as follows:

- **Archive**: Inform you if an archive will be created. Based upon the IP Packager settings, you can choose to archive the IP Packager outputs.
  - If the archive option is not created, the IP Packager outputs are located in the repository directory defined in the later options.
  - If the archive option is set in the IP Packager settings, the text shows the name and location of the archive output file (Figure 8-93). By default, the archive name is `<Vendor>_<Library>_<Name>_<Version#>.zip`.

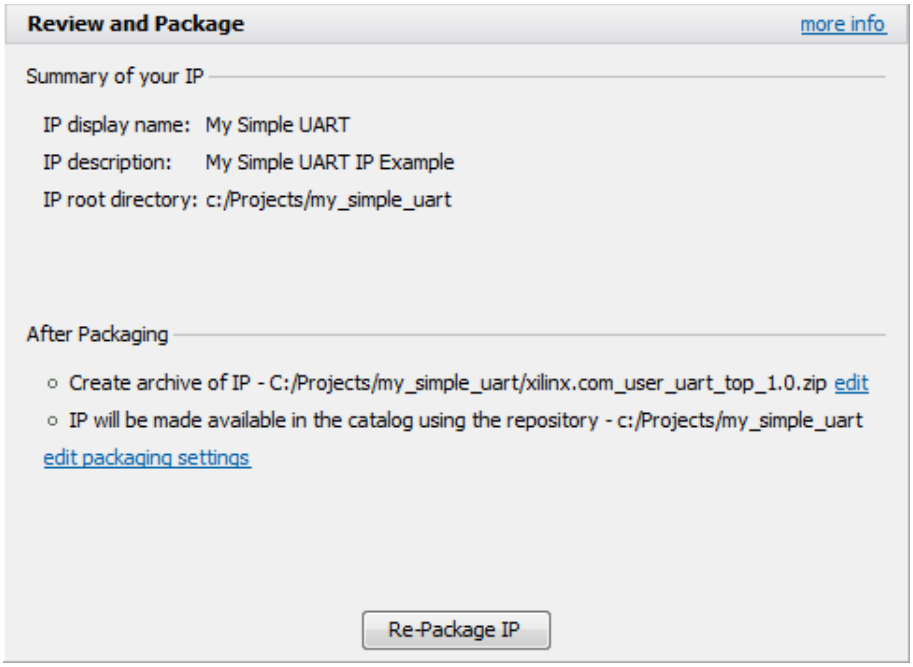


Figure 8-93: **Review and Package with Archive Option**

1. To change the name or location of the archive, click the **edit hyperlink** at the end of the archive name. This opens the archive dialog box (Figure 8-94).

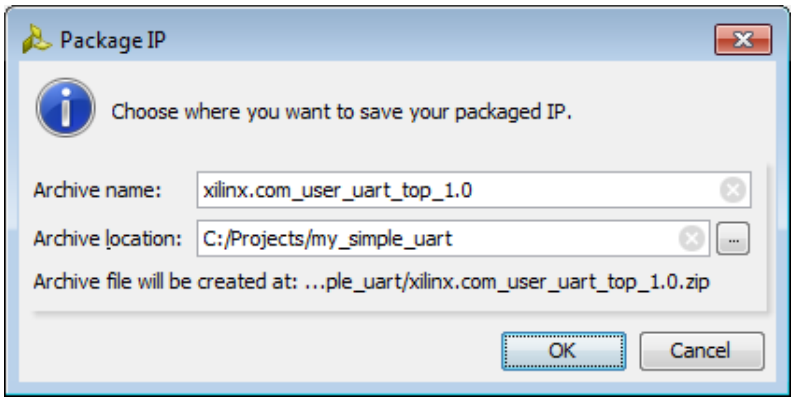


Figure 8-94: **Package IP: Archive Modification Dialog Box**

The dialog box contains the following options to modify:

- **Archive Name**: Name of the archive output file.
- **Archive location**: Location directory where the archive is generated.
- **Archive file will be created at**: Full path of the archive output file based upon the archive name and location.

2. After adjusting the name and location of the archive, click **OK** to return to the Review and Package section with the updated archive information.
   - **Repository**: Informs where the IP repository is made available.

     The repository location described in the summary is the location that other Vivado projects can reference to create your IP.

IMPORTANT: *End-user must set the repository location in the **Project Settings** > **Repository Manager** section of their projects.*

   - **IP Packager Window**: Informs you how the IP Packager window behaves after packaging was successfully completed. In the IP Packager settings, you can chose to close the IP Packager packaging.

     This option might not be visible if you did not set the option to close the IP Package window after packaging. When the option to close the IP Package window after packaging is set, the Review and Package summary contains a bullet describing the behavior of the window (Figure 8-95, page 148).

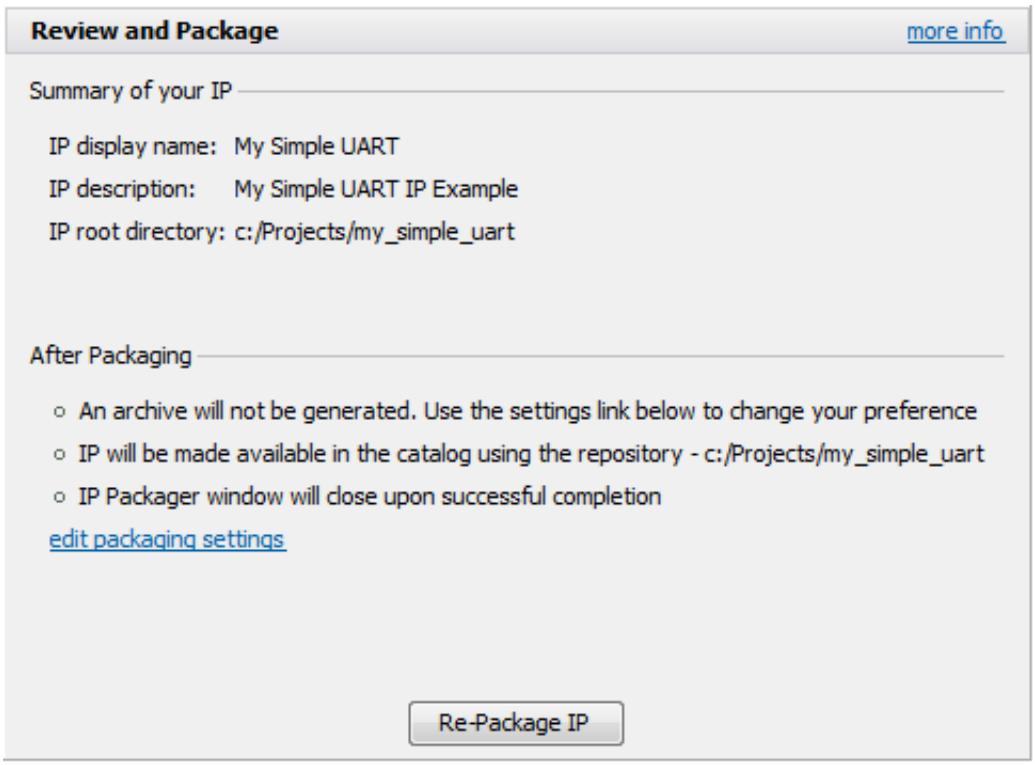


Figure 8-95: **Review and Package Window with IP Packager Window Behavior**

At the end of the Create and Package IP wizard, the IP is packaged based upon the heuristic parsing of the Vivado IDE.

At the bottom of the Review and Package window, the **Re-Package IP** button lets you re-package the IP. If you made any changes the IP

Package window after initially packaging the IP from the Create and Package IP wizard, the **Re-Package** button re-packages the changes. If necessary, select the **Re-Package** button

A finished packaging dialog box (Figure 8-96) displays, when re-packaging is successful.

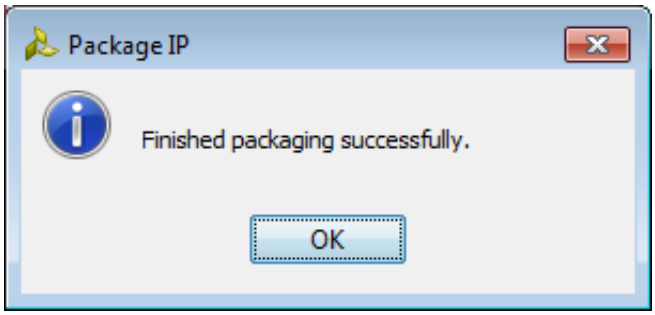


Figure 8-96: **Finished Packaging Successfully Dialog Box**

Chapter 9

Tcl Commands for Common IP Operations

Introduction

This chapter covers the Tcl commands to use for common IP operations.

For more information about using Tcl and Tcl scripting, see the following:

- *Vivado Design Suite User Guide: Using Tcl Scripting* (UG894) [Ref 15]
- *Vivado Design Suite Tcl Command Reference Guide* (UG835) [Ref 12].

For a step-by-step tutorial that shows how to use Tcl in the Vivado tool, see the *Vivado Design Suite Tutorial: Design Flows Overview* (UG888) [Ref 14].

Using IP Tcl Commands In Design Flows

Generally, the IP Tcl commands used for working are consistent between the Project Mode flow and the Non-Project Mode flow with a few exceptions related to setting the part to be used for IP creation and synthesis. Table 9-1 lists the Tcl command in the order that you would use them in a design.

Table 9-1: **IP Tcl Commands in Order of Design Use**

Action	Project Mode Command	Non-Project Mode Command
Set part for IP creation	N/A. Part is a project setting.	create_project –in_memory / –part <*part_name*> ***Note:*** Creates project in memory, not on disk. Commands that have a part option, for example, **`synth_design`** then use the specified part.
Create an IP Customization	`create_ip <IP_Name>`	`create_ip <IP_Name>`
Configure IP Customization	`set_property \ CONFIG.Input_Data_Width 8 \ [get_ips <IP_Name>]`	`set_property \ CONFIG.Input_Data_Width 8 \ [get_ips <IP_Name>]`
Generate output products	`generate_target \ [get_ips <IP_Name>]` ***Note:*** Optionally, you can specify the target(s) you want to generate.	`generate_target \ [get_ips <IP_Name>]` ***Note:*** Optionally, you can specify the target(s) you want to generate.

Table 9-1: **IP Tcl Commands in Order of Design Use *(Cont'd)***

Action	Project Mode Command	Non-Project Mode Command

Synthesize IP to create OOC DCP	`create_ip_run \`
	`  [get_ips <IP_Name>]`
	`launch_runs <IP_Name>_synth_1`
		`synth_ip [get_ips <IP_Name>]`

Read an IP	Copy an IP into a project along with any output products:
	`import_ip <ip_name>.xci`
	Add the IP to a project along with any output products and reference from the specified location. Use either of the following:
	`add_files <ip_name>.xci`
	`read_ip <ip_name>.xci`
	Note: `import_ip` allows you to rename the IP using the `-name` option.
		Read the IP as well as any generated output products. Use either of the following:
		`add_files <ip_name>.xci`
		`read_ip <ip_name>.xci`
		Note: Unlike in the Project Flow, the output products are not generated automatically. You must generate them using the `generate_target` command.
		If you use the `synth_ip` command to produce a DCP for the IP, it is not necessary to generate the output targets first; those targets are generated automatically.

File queries	`get_files -of_objects \` `  [get_ips <IP_Name>]`	`get_files -of_objects \` `  [get_ips <IP_Name>]`
Simulation	See the Chapter 6, Simulating IP.	See the Chapter 6, Simulating IP.

Tcl Commands for Common IP Operations

Within the Vivado IDE, the Vivado IP Catalog can be accessed from the Vivado IDE and the Tcl design environment.

To accommodate end-users that prefer batch scripting mode, every IP Catalog action such as IP creation, re-customization, output product generation, and so forth, which is performed in the Vivado IDE echoes an equivalent Tcl command into the `vivado.log` file; consequently, anything that you can do in the Vivado IDE you can script also.

The Vivado IP catalog provides direct access to IP parameter customization from the integrated Vivado IDE Tcl Console so you can set individual IP parameters directly from the Tcl Console.

The following are examples of common IP operations:

Create a customization of the accumulator IP:

- **Tcl Command**:

```
create_ip -name c_accum -vendor xilinx.com -library ip  \
-module_name c_accum_0
```

To change customization parameter such as input and output widths:

- **Tcl Command**:

```
set_property -dict [list CONFIG.Input_Width {10}
CONFIG.Output_Width {10}] [get_ips c_accum_0]
```

To generate selective output products:

- **Tcl Command**:

```
generate_target {synthesis instantiation_template simulation} \
[get_ips c_accum_0]
```

To reset any output products generated:

- **Tcl Command**:

```
reset_target all [get_ips c_accum_0]
```

You can use a Tcl script to list the user configuration parameters that are available for an IP.

Use either the `list_property` or `report_property` command and reference the created IP.

The difference between these commands is:

- `list_property` returns a list of objects which can be processed with a Tcl script as a list.
- `report_property` returns a text report giving the current value for each parameter, its type, and other parameters.

To get a list of all properties which apply to an IP:

- **Tcl Command**:

```
list_property [get_ips fifo_generator_0]
```

To get an alphabetized list of just the customization parameters you can augment this further:

- **Tcl Command**:

```
lsearch -all -inline [ list_property [ get_ips fifo_generator_0 ] ] CONFIG.*
```

Create a report listing all the properties for an IP, including the configuration parameters:

- **Tcl Command**:

```
report_property [get_ips fifo_generator_0]
```

For more information on the supported IP Tcl commands type, `help -category IPFlow` in the Tcl Console as shown in Figure 9-1.

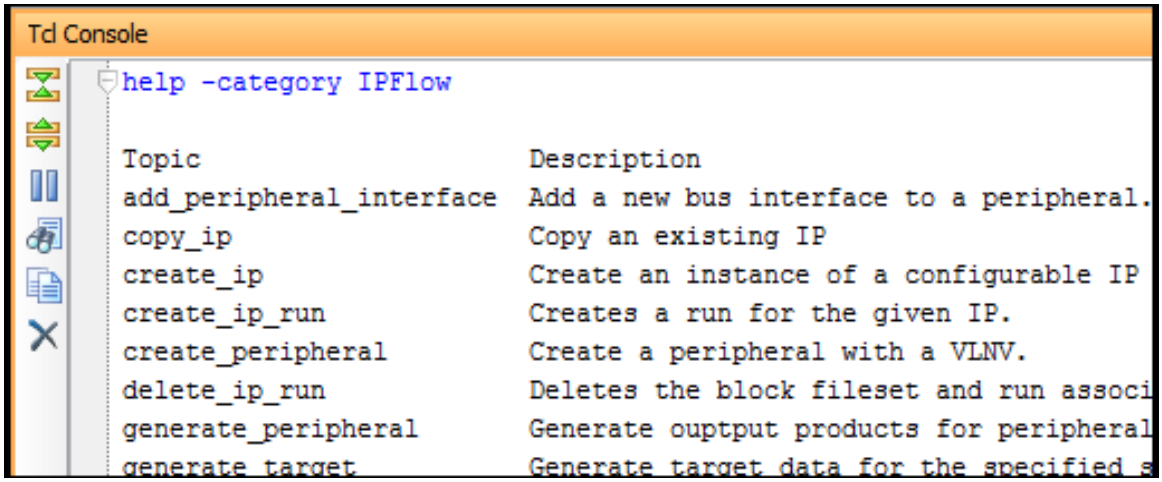


Figure 9-1: **Getting Help on IP Tcl Commands**

Note: The *Vivado Design Suite Tutorial: Designing with IP* (UG939) [Ref 5] contains labs that cover scripting of both Project Mode and Non-Project Mode flows with IP. They include examples of generating output products as well as selectively upgrading IP.

Example IP Flow Commands

This section provides Tcl script examples for some common operations.

Commands to Create IP

The `create_ip` command is used to create IP customizations.

RECOMMENDED: *Xilinx recommends that you perform this operation as described in Using the Manage IP Flow, page 49. When you create IP with the Manage IP flow, you can subsequently use that IP in Project and Non-Project mode.*

The following script shows how to created a manage IP project, create and customize an IP, and generate a DCP:

```
# Create a Manage IP project
create_project <managed_ip_project> ./managed_ip_project -part <part> -ip

# Set the simulator language (Mixed, VHDL, Verilog)
set_property simulator_language Mixed [current_project]

# Target language for instantiation template and wrapper (Verilog, VHDL)
set_property target_language Verilog [current_project]

# Create an IP customization
create_ip -name c_accum -vendor xilinx.com -library ip -module_name c_accum_0

# configure the parameters for the IP customization
```

```
set_property -dict {CONFIG.Input_Width 10 CONFIG.Output_Width 10} [get_ips
c_accum_0]

# Create a synthesis design run for the IP
create_ip_run [get_ips c_accum_0]

# Launch the synthesis run for the IP
# Because this is a project, the output products are generated automatically
launch_run c_accum_0_synth_1
```

IMPORTANT: *Xilinx recommends project-based flows. Project-based flows can run in either the Vivado IDE or using the Tcl commands.*

Tcl Command to Remove a Design Run

```
delete_ip_run
```

Use this command to remove the design run for an IP. If you reset the output products for an IP in the Vivado IDE, two Tcl commands are issued: `reset_target` and `delete_ip_run`.

Querying IP Customization Files

Tcl Script for Getting Files for Source Control

This example script shows how to get all files for a given IP customization. You can use this script to generate a list of files for use with a source control system.

```
# Create a project in memory, no project directory
# created on disk
create_project -in_memory -part <part>
# read an IP customization
read_ip <ip_name>.xci

# Generate all the output products
generate_target all [get_ips <ip_name>]

# Create a DCP for the IP
synth_ip [get_ips <ip_name>]

# Query all the files for this IP
get_files -all -of_objects [get_files <ip_name>.xci]
```

Note: See Table 9-1, page 149 for a more detailed explanation on these Tcl Commands and when to use them.

Querying an Ordered Source List

When creating custom scripts, you can use one of the following commands:

Tcl Command for IP Only: Synthesis

```
get_files -compile_order sources -used_in synthesis \
-of_objects [get_files <ip_name>.xci]
```

Tcl Command for IP Only: Simulation

```
get_files -compile_order sources -used_in simulation \
-of_objects [get_files <ip_name>.xci]
```

Tcl Command for Top-Level Design: Including IP For Synthesis

- **Tcl Command**:

```
get_files -compile_order sources -used_in synthesis
```

Tcl Command for Top-Level Design: Including IP For Simulation

- **Tcl Command**:

```
get_files -compile_order sources -used_in simulation
```

Scripting Examples

Implementing an IP Example Design

Create a project to run implementation on an IP example design.

```
# Create a project
create_project <name> <dir> -part <part>

# Create an IP customization and a DCP
# This will also generate all the output products
create_ip ...
create_ip_run [get_ips <ip>.xci]
launch_runs <ip>_synth_1
wait_on_run <ip>_synth_1

# Open the example design for the IP
# This will use the IP DCP generated
open_example_project -force -dir "." -in_process [get_ips <ip>]
launch_runs synth_1
wait_on_run synth_1
launch_runs impl_1
wait_on_run impl_1 -to write_bitstream
open_run impl_1

# produce some reports
report_timing_summary ...
report_utilization ...
```

Non-Project Synthesis

Synthesize and implement design in a non-project flow with one IP which has an OOC DCP generated and one IP being synthesized along with user logic.

When reading an IP XCI file, all output products present, including an OOC DCP, are used, there is no need to generate them.

If the output products have not been generated for the IP, you must generate the output products (or create a DCP using the `synth_ip` command which generates the output products also).

If you elect to use global synthesis for an IP (see the Synthesis Options for IP in Chapter 2) then you must disable checkpoint support and generate the output products. This Tcl script provides a template for this.

```
#create an in memory project to provide the part to use for IP creation and for
running synthesis

create_project -in_memory -part <part>

# read in sources
read_verilog top.v

# Read in an existing IP customization
# or create an IP from scratch
# create_ip ... or read_ip ip1.xci

# Generate a DCP for the IP
# will generate output products if needed
synth_ip [get_ips ip1]

# Read in an existing IP customization
# or create an IP from scratch
# create_ip ... or read_ip ip2.xci

# Set IP to use global synthesis (no DCP generated)
set_property generate_synth_checkpoint false [get_files ip2.xci]

# Need to generate output products for IP
generate_target all [get_ips ip2]

# synthesis the complete design
synth_design -top top

# run implementation
opt_design
place_design
route_design

# write the bitstream
write_bitstream -file top
```

Simulating an IP Example Design

Create a project to run simulation on an IP example design.

```
#create the project
create_project <name> <dir> -part <part>

# create IP and a synthesis run
create_ip ...
create_ip_run [get_ips <ip_name>]

#launch runs
launch_runs <ip>_synth_1
wait_on_run <ip>_synth_1

#open the example project
open_example_project -force -dir "." -in_process [get_ips <ip>]

#launch simulation
<launch_xsim |launch_modelsim>
```

Synthesizing and Simulating an IP

If an IP does not deliver an example design, but does deliver a test bench, you can perform simulation of just the IP.

```
#create the project
create_project <name> <dir> -part <part>

# create_ip ... or add_files ip.xci

# create an IP design run
create_ip_run [get_ips <ip_name>]

#launch IP synthesis run
launch_runs <ip>_synth_1
wait_on_run <ip>_synth_1

# Setting up simulation test bench
set_property top <tb> [current_fileset -simset]

# Launch simulation
<launch_xsim> | <launch_modelsim>
```

Using the IP Board Flow

Using the IP Board Flow

The IP Board Flow feature is supported by some IP and gives you the ability to select board interfaces while customizing an IP. When you use this feature, the creation of physical constraints for the IP is automated by delivering additional XDC constraints in a special file.

As shown in Figure A-1, when creating a new project, you can select a board as the default part.

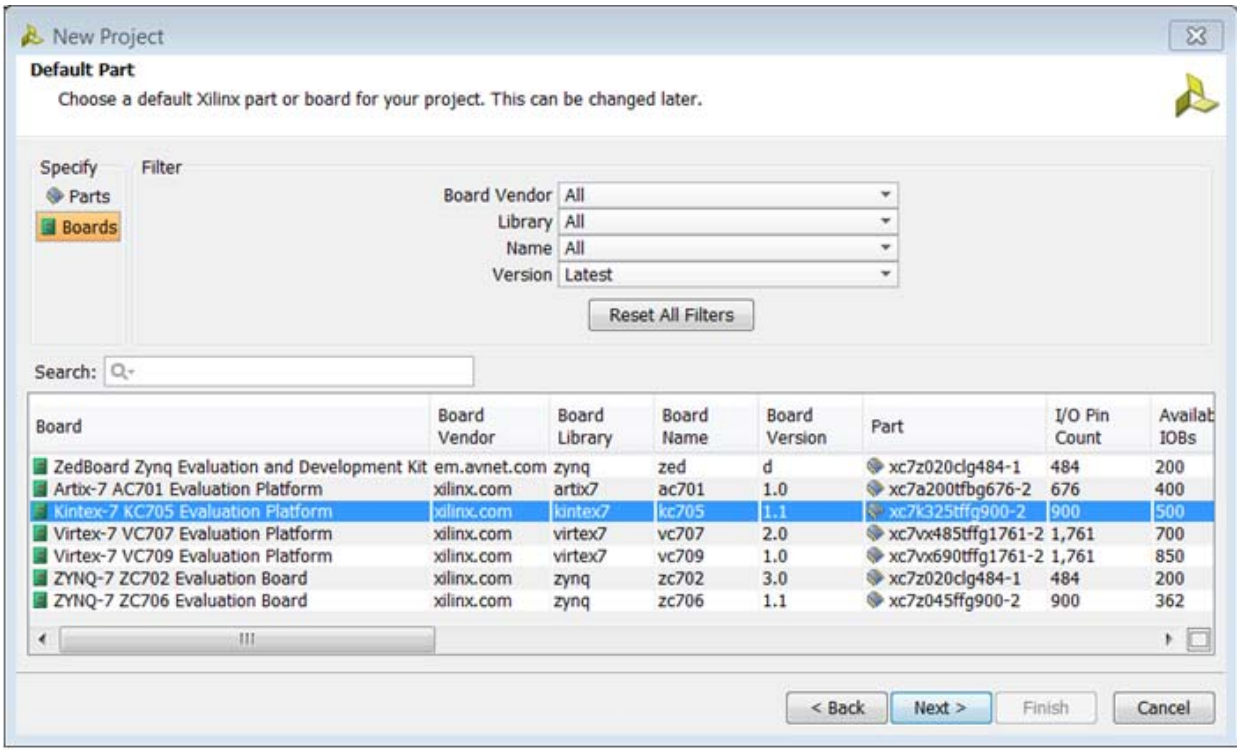


Figure A-1: **Selecting a Board as the Default Part**

Selecting one of the listed boards results in IP that supports the board flow to have a new tab visible during customization as shown in the following figure.

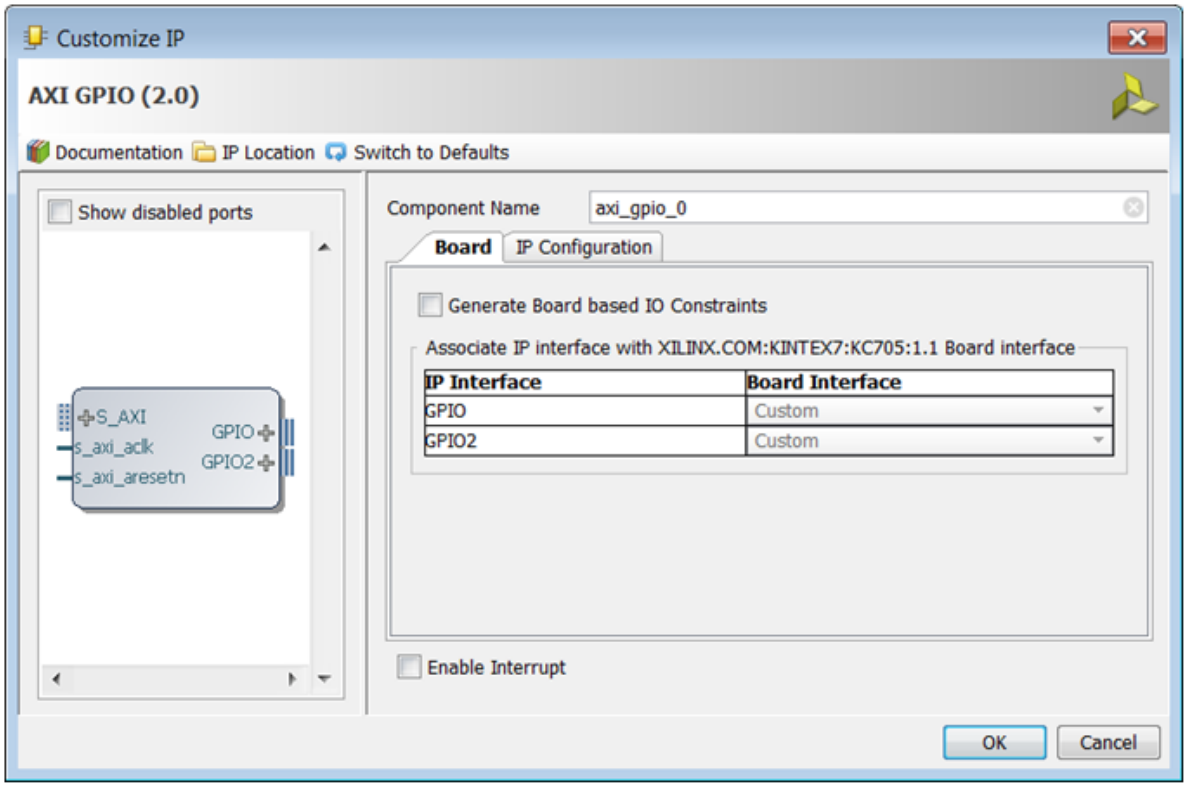


Figure A-2: **Board Type Visible in Supported IP Customization**

The following figure shows that when you check the **Generate Board based IO Constraints** check box, you can associate the IP interface to the available board interface.

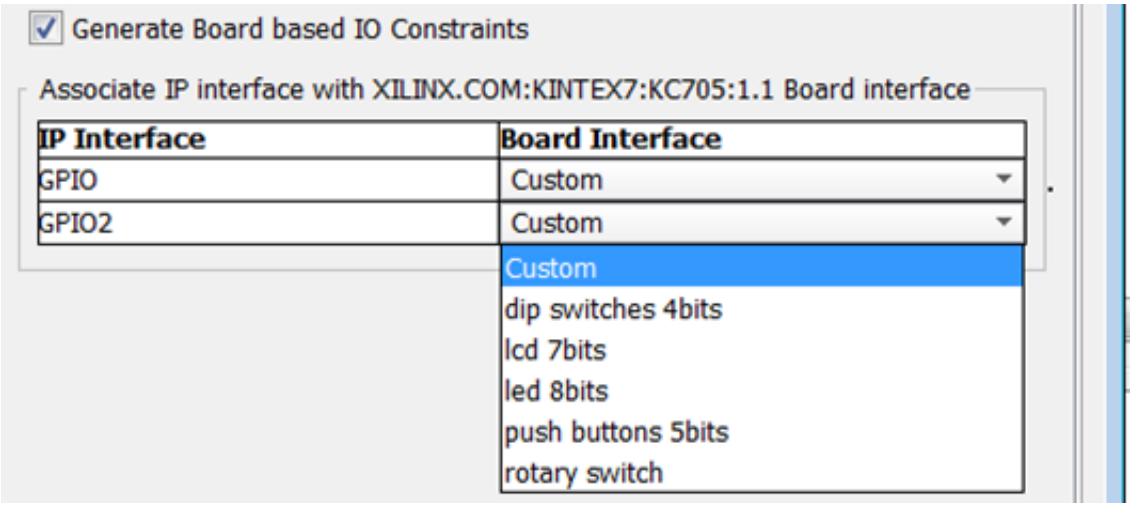


Figure A-3: **Associating the IP Interface with the Board Interface**

When the Vivado IDE generates the output products for the IP in the IP Sources view, you see the `<IP_Name>_board.xdc` file listed.

This file contains physical constraints assigning ports of the IP to the package pins that connect to the related board connector or device such as a USB port, LED, button, or switch.

The following figure shows the XDC constraints created for the GPIO IP when you connect the GPI0 interface to the board LCD interface and the connect the GPIO2 interface to the board push buttons.

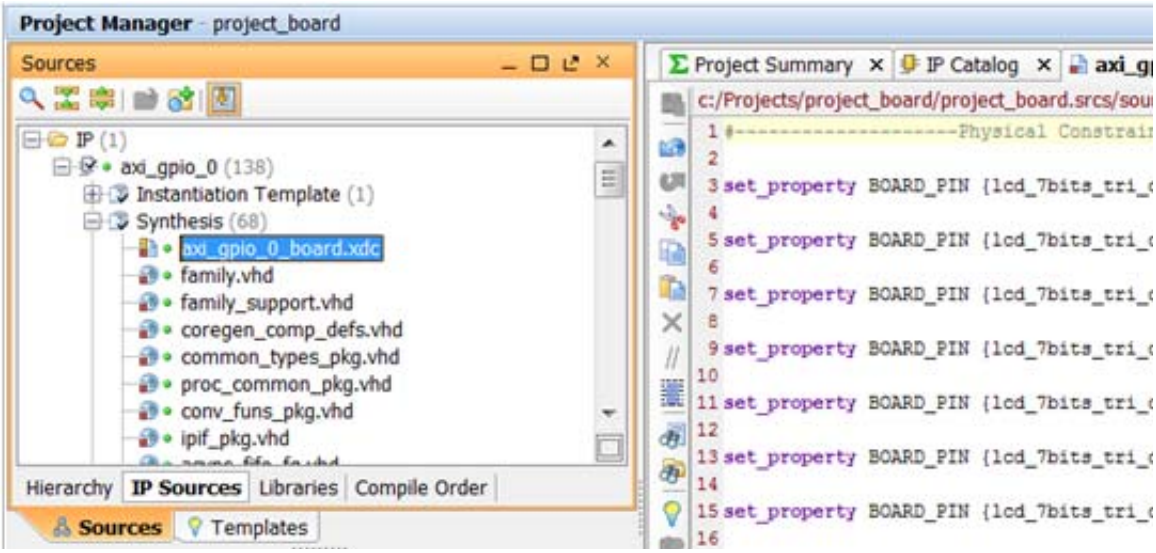


Figure A-4: **IP Board XCD File**

IP that Supports Board Flow

The IP that support the board flow include the following:

- AXI EMC (`axi_emc`)
- AXI Ethernet (`axi_ethernet`)
- AXI Ethernet Buffer (`axi_ethernet_buffer`)
- AXI Ethernet Lite (`axi_ethernetlite`)
- AXI GPIO (axi_gpio)
- AXI IIC (`axi_iic`)
- AXI Quad SPI (`axi_quad_spi`)
- AXI UART 16550 and Lite (`axi_uart16550` and `axi_uartlite`)
- Clocking Wizard (`clk_wiz`)
- Gigabit Ethernet PCS PM (`gig_ethernet_pcs_pm`)
- I/O Module (`iomodule`)
- MicroBlaze™ MCS (microblaze_mcs)
- MIG for 7 series (`mig_7series`)

- Proc Sys Reset (`proc_sys_reset`)
- Processing System 7 series (`processing_system7`)
- Tri-Mode Ethernet Mac (`tri_mode_ethernet_mac`)

When generating IP for use in the Vivado IP integrator, you must be familiar with the parameters and the valid values for key bus interfaces. These interfaces include:

- AXI4-Lite
- AXI4 memory mapped protocol
- AXI4-Stream
- clock
- interrupt
- reset

See the Vivado *AXI Reference Guide* (UG1037) [Ref 19], and the IP-specific data sheets for more information on interface naming conventions. Also, see AXI Naming Conventions, page 167.

XILINX
ALL PROGRAMMABLE

Appendix B

IP Files and Directory Structure

Introduction

When customizing an IP using the IP Catalog, either directly in a project or using the Managed IP Flow, the Vivado® Integrated Development Environment (IDE) creates a unique directory for each IP.

After creating an IP customization, a unique directory is made which contains the Xilinx® Core Instance (XCI) file, instantiation template, BOM file, and any generated output products. In this IP directory, there are a number of additional directories. There is no common structure for the organization of the files that each IP delivers, but there are some common files that are created for each IP.

IP-Generated Directories and Files

The following table lists the IP-generated target directories and files, which are also known as output products.

RECOMMENDED: *Xilinx recommends that you use Tcl commands to access the list of related files rather than view the file and directory structure. For example, you can use the* `get_files` *Tcl commands, which are shown in Querying IP Customization Files in Chapter 9. For more information, see the Vivado Design Suite Tcl Command Reference Guide (UG835)* [Ref 12].

Table B-1: **IP Output Products**

Directory Name, File Name, or File Type	Description
`doc`	Contains the `<Core_Name>_changelog.txt` file that provides information about changes to the IP for each release.
`sim`	Contains a top-level file in the target language specified (Verilog or VHDL). This directory is not present for all IP.
`synth`	Contains a top-level file in the target language specified (Verilog or VHDL). This directory is not present for IP that does not support synthesis, such as simulation-only bus functional model (BFM) IP.

Table B-1: **IP Output Products *(Cont'd)***

Directory Name, File Name, or File Type	Description
<IP_Name>.xci	Contains the IP customization information. You can generate the output products from this file. If an upgrade path exists for the IP in the Catalog, you can upgrade from this file to the latest version.
<IP_Name>.veo|vho	Verilog (VEO) or VHDL (VHO) instantiation template. You would use one of these files to instantiate the IP inside your design.
<IP_Name>.xml	IP Bill of Material (BOM) file that keeps track of the current state of the IP, including generated files, computed parameters, and interface information.
<IP_Name>.dcp*	Synthesized Design Checkpoint file containing constraints and post synthesis netlist for IP.
<IP_Name>_stub.[v|vhdl]*	Instantiation template for use with third-party synthesis tools to infer a black box for IP.
<IP_Name>_funcsim.[v|vhdl]*	Post synthesis structural simulation netlist files
<IP_Name>.xdc	Timing and/or physical constraints. These files are not present for all IP, and their location varies by IP.
<IP_Name>_clocks.xdc	Constraints with a clock dependency. These files are not present for all IP, and their location varies by IP.
<IP_Name>_board.xdc	Constraints used in a board flow. These files are not present for all IP, and their location varies by IP.
<IP_Name>_ooc.xdc	Default clock definitions for use in a Managed IP flow or IP netlists. These files are not present for all IP, and their location varies by IP.
Encrypted HDL for the IP	Files used for synthesizing and simulating the IP. These files are not present for all IP, and their location varies by IP.

* The DCP, _stub, and _funcsim files are created only when using the Out-of-Context flow for synthesis (default). See Synthesis Options for IP in Chapter 2 for more details.

Note: Although example design are not output products, they are commonly generated for IP. The example design files are only available when the example design is opened with one of the following:

- open_example_project Tcl command
- Vivado IDE with the **Open IP Example Design** menu command.

For more information, see Chapter 4, Using IP Example Designs.

Appendix C

IP Optimization (FMax Characterization)

Introduction

This appendix describes the methods used by Xilinx® to determine the maximum frequency (FMax) of IP operation within a system design. The method also enables realistic performance reporting for any Xilinx FPGA architecture. The maximum frequency of a design is the maximum frequency at which the overall system can be run without encountering functional issues. With the current trend toward implementing more complex systems in FPGAs, the FMax frequency of each included IP is an important factor.

Standalone IP Characterization Methodology

There are several factors to consider in determining how fast an IP runs.

1. The HDL coding style of the IP. You can achieve higher performance by pipelining the design to run at higher frequencies.

   There are also other trade-offs that can be made at the micro-architecture level to share resources that affect the frequency.

2. Another factor in determining the overall performance is the option settings used in the implementation tools. By choosing appropriate options, you can achieve required design goals. You can constrain a design for *area* (minimum resource usage) or *performance* (highest achievable frequency).
3. Options in the design tools provide additional control over how you synthesize and implement a design. For example, you could choose to replicate logic, do register re-timing, share resources, and/or provide proper constraints to achieve higher performance.
4. For standalone IP characterization, Xilinx uses the default design tool settings. As shown in the following figure, the methodology involves instantiating the stand-alone IP design Design Under Test (DUT) and then wrapping the design in a scan-register wrapper to limit the use of I/Os.

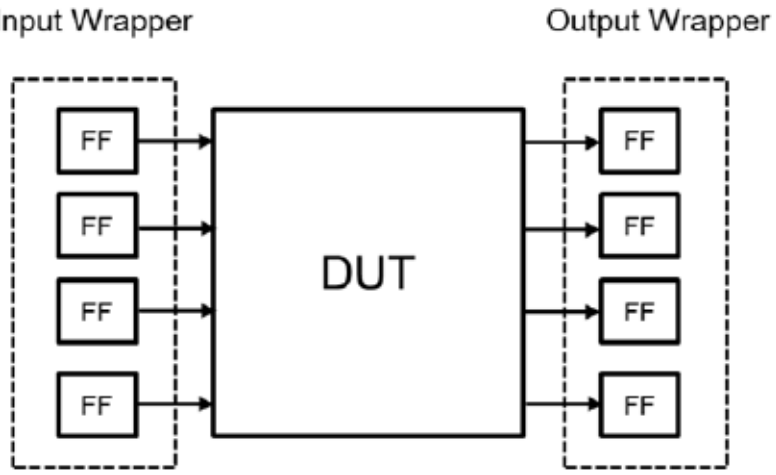


Figure C-1: **Standalone IP Characterization**

In addition to saving I/Os, the wrapper logic also ensures the following:

- The design maps effectively on a large device.
- Spreading the design within a large device is not an issue
- Validates that the IP is ready for use.

The standalone IP characterization methodology helps determine the maximum frequency at which the IP can be run using only the default system constraints. Because this is standalone IP, there are no resource constraints competing for placement and routing resources. Additionally, no I/O constraints, and the presence of the wrapper ensures that I/Os are directly driven or sampled by Flip-Flops (in the wrapper).

Note: The standalone IP performance numbers are based on a full implementation run of the design through Xilinx tools including synthesis, placement and routing.

The Fmax Margin System Methodology

While stand-alone IP characterization helps to determine the maximum limit of IP performance, it is important to determine the IP performance in the context of a user system. In the case of embedded systems, the system typically includes the following items:

- MicroBlaze Processor or Processor System 7 (PS7)
- Caches (IC and DC)
- One or more levels of Interconnect
- Memory controller (MIG)

- Direct Memory Access (DMA) Controller
- On-chip BRAM controller
- Peripherals (such as UART, Timer, GPIO, Interrupt Controller)
- The IP under test (such as DUT)

Determining the Fmax of an Embedded IP with these components provides a more realistic performance target.

In addition, the user system can fill up the device with up to 70-80% logic. The Fmax Margin system methodology stitches the IP in a basic embedded system and fills the rest of the device with available LUTs, BRAMs, and I/Os to ensure that 70-80% of the device is full.

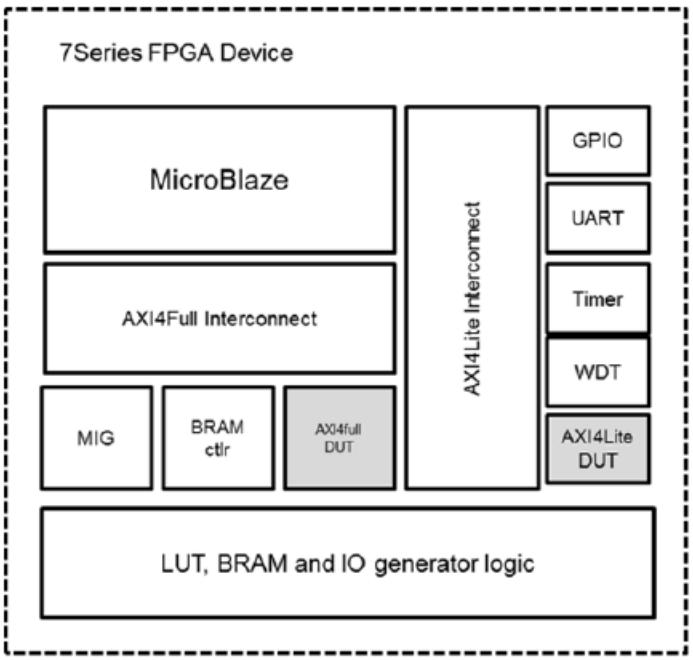


Figure C-2: **Fmax Margin System for 7 Series**

The embedded system shown in the figure above has the following two types of AXI Interconnect:

- **AXI4-Lite**: Used for peripheral command and control. In general, this interconnect runs at a much lower frequency and is designed for minimum area.
- **AXI4**: Used for heavy-duty data-motion-type applications. In general, this interconnect runs at optimum speed and is designed for maximum performance.

For Fmax Margin System Analysis, the AXI4-Lite Interconnect clock frequency is fixed to 150 MHz. The AXI4 Interconnect and the rest of the logic is increments from 150 MHz up to the maximum frequency where the system breaks with timing violations (Worst Case Negative slack). The maximum frequency at which AXI4 runs determines the Fmax of the overall system.

The following block diagram is the Fmax Margin System for the Zynq®-7000 AP SoC processor.

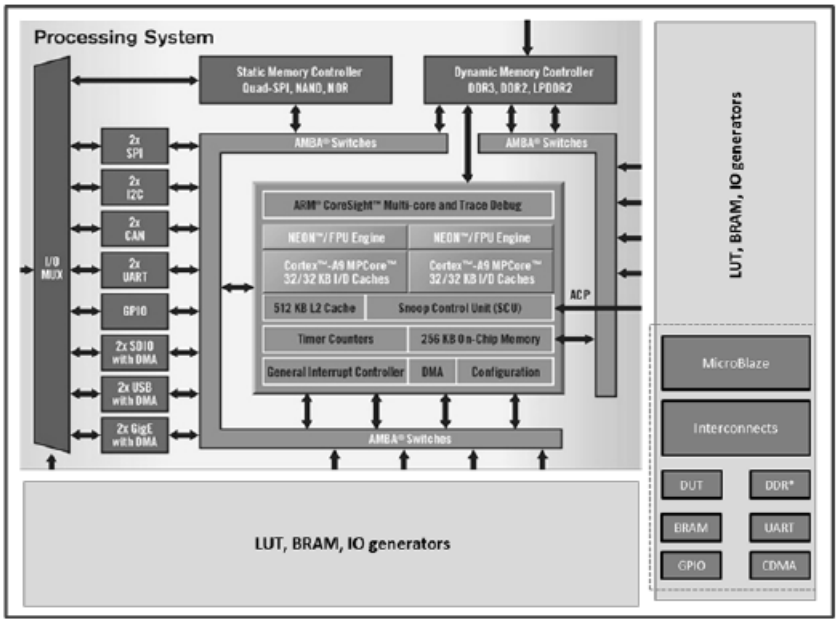


Figure C-3 **Fmax Margin System for Zynq**

Tool Options and Other Factors

Xilinx tools offer a number of options and settings that provide a trade-off between design performance, resource usage, implementation run time, and memory footprint. The settings that produce the best results for one design might not necessarily work for another design.

For the purpose of the Fmax Margin System Analysis, the IP design is characterized with default settings without specific constraints (other than the clocking constraint). The clocking constraint is increased in steps starting from 150 MHz until the system fails with timing violations (Worst Case Negative slack). This analysis is done with different FPGA architectures and speed grades. The results are reported in the individual IP Product Guide for the IP core.

AXI Naming Conventions

Introduction

When renaming IP signals (such as interfaces, ports, memory) for the IP integrator after customization, you must adhere to the required conventions for the IP integrator to infer associations.

- For AXI interface adoption and protocols, see the Vivado *AXI Reference Guide* (UG1037) [Ref 19] for naming conventions.
- For information on the IP integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems Using IP Integrator* (UG994) [Ref 17].

Required Naming Conventions

Use Interfaces - AXI4

- Signal Interfaces: clock, clock enable, reset, reset enable, and interrupt, data are typically single port interfaces. They are identified as an interface to allow for specialized processing.
- Clock, clock enable, and reset must be associated with AXI interface.
- I/O (off chip) interfaces: Many of these are available.

Memory

Slaves

AXI memory-mapped slave IP must declare information about accessible memory to the master. Slave IP must map to the master memory map, and within it, one or more address blocks to declare access.

If your IP has a fixed range (non-parameterizable) under 32 bits, you can skip declaration.

Masters

For memory masters, keep these rules in mind:

- Master interfaces need to declare the address space they would access. Create an address space and set the range and width.
- Some masters, like DMA, must be interconnected to appropriate IP. To assist in additional automation, specify more information, as follows:
  - An additional parameter, `DEPENDENT_ON`, must be set on the address space. Set the value should to the string name of the slave interface that creates the transaction. This allows DMA-like masters to declare that master transactions are based upon a slaved request.
  - An additional parameter `PREFERRED_USAGE` on the address space indicates the type of memory accessed. Possible values are: `MEMORY`, `REGISTER`, and `ALL`. DMAs use `MEMORY`.

Bridges

Multiple address blocks might require dependency expressions.

- Declare the relationship between the slave and the master interface (the *channel*), along with how opaque the address space is from slave interface to master.
- Use the property `bridges`, and on the slave interface:
  - Set the bridges property to a value that is the pair of the name-related master interface.
  - Set the boolean value for the opacity.

Advanced

Enabling Ports and Interfaces

Depending on parameterization, it is possible to have interfaces and single ports appear to be hidden on a block.

- Input ports must have driver values specified as to when they can be disabled (either individually or as part of an interface)
- Variable width ports must use a verilog-like sign extension to drive non-zero values.

Naming Style

Adhere to the following naming style conventions wherever possible:

- Use small, meaningful names where possible: the IP block diagram has limited space.
- For memory maps, name them exactly as the interface to which it refers.
- For address blocks within a memory map, choose short meaningful names such as `Mem0` or `Reg0`.

Guidelines on AXI Parameter Propagation

Table D-1: **Propagation Names with Use Notes**

Propagation Name	Notes
ID_WIDTH	AXI3/AXI4 slaves should propagate in this information and use it to configure their slave ID widths.
NUM_READ_OUTSTANDING	For endpoint AXI masters, this number should be emitted to describe how many outstanding pipelined reads the master is capable of issuing in its current configuration. For endpoint AXI slaves, this number should be emitted to describe how many outstanding pipelined reads the slave is capable of handling. This information is used to tune/optimize AXI Infrastructure IP.
NUM_WRITE_OUTSTANDING	For endpoint AXI masters, this number should be emitted to describe how many outstanding pipelined writes the master is capable of issuing in its current configuration. For endpoint AXI slaves, this number should be emitted to describe how many outstanding pipelined writes the slave is capable of handling. This information is used to tune/optimize AXI Infrastructure IP.
SUPPORTS_NARROW_BURST	For endpoint masters. set to 0 if the masters set cache bits[1:0] to 11 (modifiable and bufferable) and do not issue narrow bursts (`AxiSize <data_width and AxLEN > 0`). Emitting this value as 0 allows AXI infrastructure IP and endpoints to make significant area optimizations.
MAX_BURST_LENGTH	For endpoint masters this should be set represent the longest burst length transactions the IP is capable of generating in its current configuration. Emitting this value correctly allows AXI infrastructure IP and endpoints to make area optimizations, especially to remove burst splitters when converting AXI4 transactions to AXI3. If an AXI4 master sets `MAX_BURST_LENGTH` to 16 or less, it allows AXI3 protocol conversion for a Zynq®-7000 processor to use less area by omitting burst splitting logic.

Reserved Parameters

The following is a set of reserved parameters that Vivado IDE sets automatically during customization/generation.

- `c_family`
- `c_component_name`
- `c_xdevicefamily`
- `c_elaboration_dir`
- `c_elaboration_transient_dir1`

Adding User Logic to Your AXI4 Peripheral

Introduction

This appendix describes how to add user logic to an IP and edit the top module to include new files.

Adding User Logic to IP

To add user logic to generated IP:

1. From the Summary page, select **Open IP in editing project** (Figure E-1).

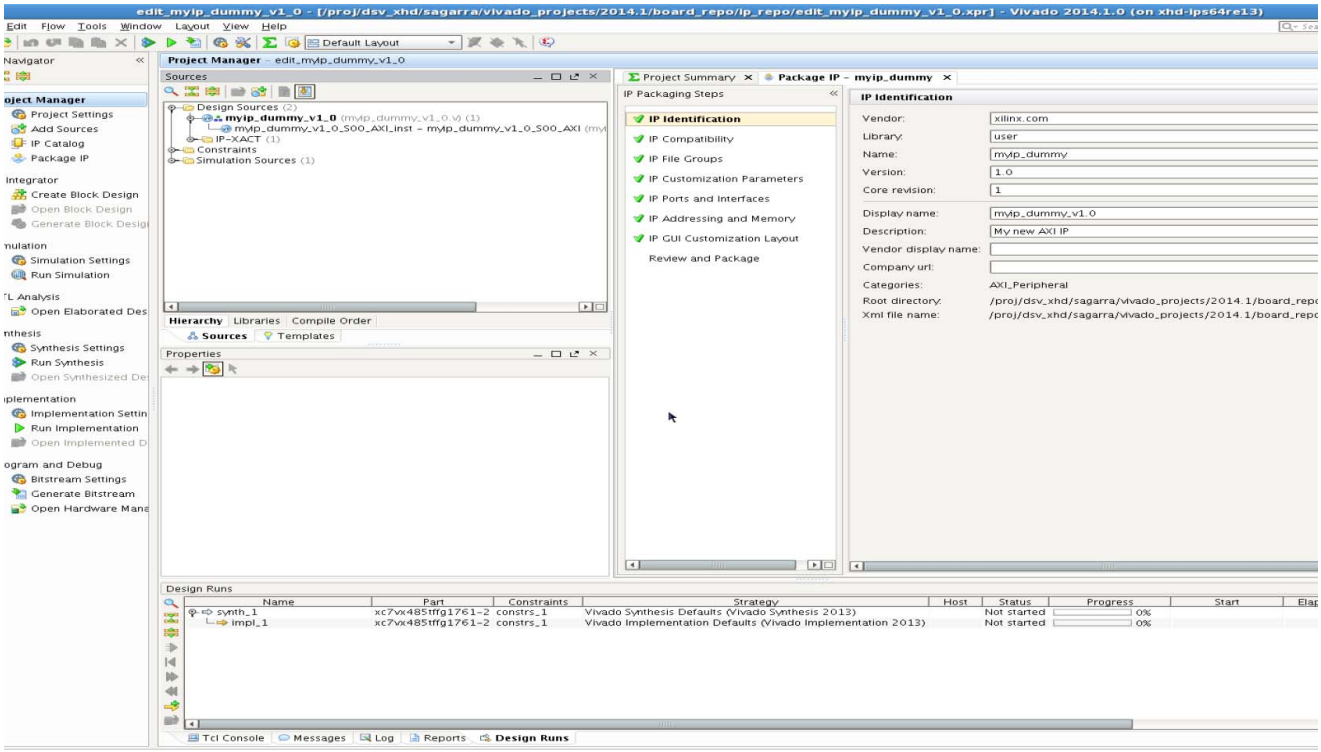


Figure E -5: **Open the IP in Edit Mode**

This opens of another instance of Vivado which creates a new project enabling the end-user to edit the IP (Figure E-2). From the hierarchy pane you can now open the top module and the associated modules.

1. In the Design Sources view, right-click and select **Add Sources**.

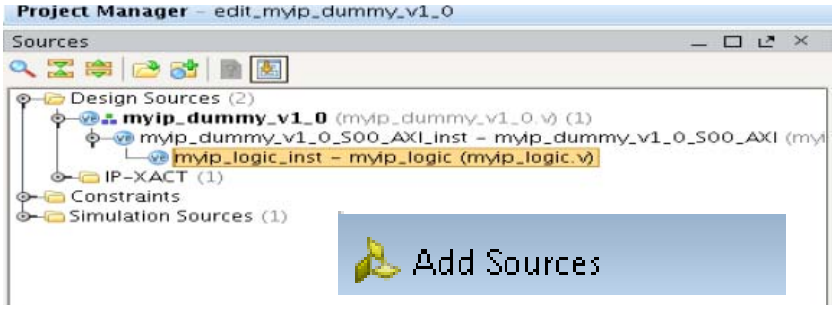


Figure E-6: **Add Sources Option**

The top module contains the instantiation of the sub-modules that include the functionality of the user-selected protocol. In this case, it is the AXI4-Lite protocol.

2. Browse through the tool generated HDL files and you can see user sections where you can add the ports to the IP to define in the top-level, if necessary.

Editing the Top Module for User Sections

To edit the top-module of a design to add source files:

1. Define the end-user logic module port map declaration in the sub-module source file, as shown in Figure E-3, page 175.

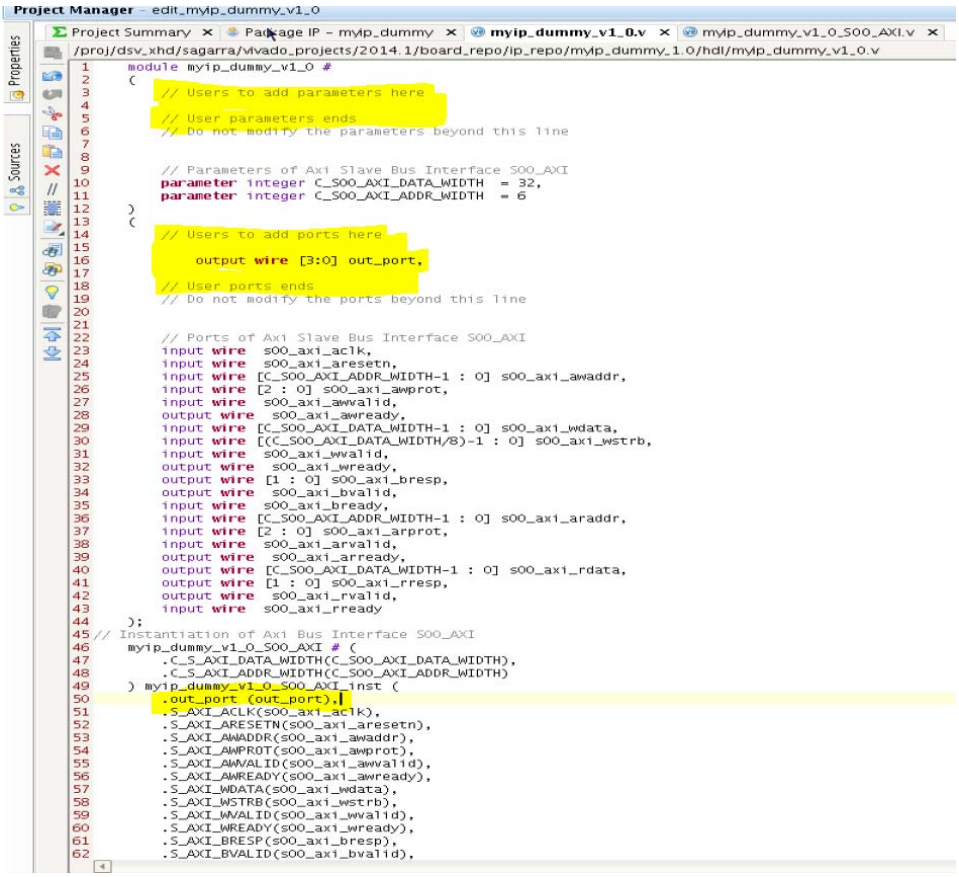


Figure E-3: **Mapping the User Module to the IP**

2. After you make your changes, ensure that these changes are updated in the `component.xml` by updating the information in the Package IP tab.

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see 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. http://www.xilinx.com/support/answers/56487.html
7. *Vivado Design Suite User Guide: System-Level Design Entry* (UG895)
8. *Vivado Design Suite Tutorial: Designing with IP Tutorial* (UG939)
9. http://www.xilinx.com/support/answers/56634.html
10. *Vivado Design Suite User Guide: Getting Started* (UG910)
11. *Vivado Design Suite User Guide: Release Notes, Installation, and Licensing* (UG973)
12. *Vivado Design Suite Tcl Command Reference Guide* (UG835)

13. *Vivado Design Suite Tutorial: Logic Simulation* (UG937)
14. *Vivado Design Suite Tutorial: Design Flows Overview* (UG888)
15. *Vivado Design Suite User Guide: Using Tcl Scripting* (UG894)
16. *Vivado Design Suite User Guide: Using the Vivado IDE* (UG893)
17. *Vivado Design Suite User Guide: Designing IP Subsystems Using IP Integrator* (UG994)
18. *Using Vivado Design Suite with Version Control Systems* (XAPP 1165)
19. *Vivado AXI Reference Guide* (UG1037)
20. *Vivado Design Suite Properties Reference Guide* (UG912)
21. *Vivado Design Suite Tutorial: Programming and Debugging* (UG936)
22. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
23. Vivado Design Suite QuickTake Video Tutorials
24. Vivado Design Suite Documentation

Xilinx IP Documentation

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

Please Read: Important Legal Notices