Vivado Design Suite

Creating, Packaging Custom IP Tutorial

UG1119 (v2016.1) May 5, 2016

XILINX
ALL PROGRAMMABLE.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
05/05/2016	2016.1	Added Lab 4: Packaging IP located in a Trunk.

Table of Contents

Revision History 2
Introduction to Creating and Packaging Custom IP 5
Introduction 5
Software Requirements 6
Tutorial Design Description 6
Locating Tutorial Design Files 6
Lab 1: Packaging a Project 7
Introduction 7
Step 1: Open the Vivado Project 7
Step 2: Preparing Design Constraints 8
Step 3: Package the IP 15
Step 4: Modify the IP Definition 17
Step 5: Add a Product Guide to the IP 19
Step 6: Review and Package the IP 22
Step 7: Validate the New IP 23
Conclusion 29
Lab 2: Packaging a Specified Directory 30
Introduction 30
Step 1: Examine the IP Directory 30
Step 2: Create a New Vivado Project 31
Step 3: Package the IP Directory 33
Step 4: Examine and Update the Packaged IP 35
Step 5: Validate the Custom IP 39
Conclusion 41
Lab 3: Packaging Legacy IP 42
Introduction 42
Step 1: Create a New Vivado Project 42

Step 2: Package a Library Core 45
Step 3: Package the GPIO IP 49
Step 4: Validate the New Custom IP 53
Conclusion 54
Lab 4: Packaging IP located in a Trunk 55
Introduction 55
Step 1: Examine the Repository Trunk Directory 55
Step 2: Create a New Vivado Project 56
Step 3: Package the Library Core 58
Step 4: Package the IP 61
Step 5: Validate the IP 68
Conclusion 69
Legal Notices 70
Please Read: Important Legal Notices 70

Introduction

This tutorial takes you through the required steps to create and package a custom IP in the Vivado® Design Suite IP packager tool.

The Vivado Design Suite provides an IP-centric design flow that helps you quickly turn designs and algorithms into reusable IP. As shown in the following figure, the Vivado IP Catalog is a unified IP repository that provides the framework for the IP-centric design flow. This catalog consolidates IP from all sources including Xilinx® IP, third-party IP, and end-user designs targeted for reuse as IP into a single environment. The following figure provides a diagram of Vivado Design Suite IP design flow.

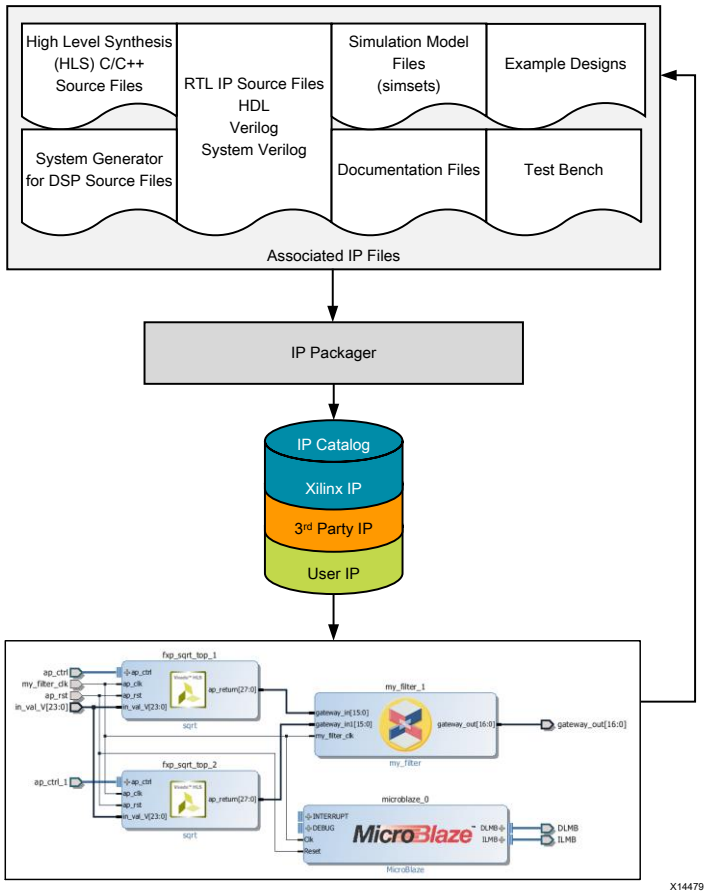


Figure 1: Vivado Design Suite IP Design Flow

The Vivado IP packager tool is a unique design reuse feature, which is based upon the IP-XACT standard. The IP packager tool provides you with the ability to package a design at any stage of the design flow and deploy the core as system-level IP.

See the *Vivado Design Suite User Guide: Creating and Packaging Custom IP* (UG1118) for more information about the Vivado IP packager.

VIDEO: *You can also learn more about the creating and using IP cores in Vivado Design Suite by viewing the quick take videos:* Configuring and Managing Custom IP *and* Customizing and Instantiating IP.

TRAINING: Xilinx provides training courses that can help you learn more about the concepts presented in this document. Use these links to explore related courses:

Essentials of FPGA Design

Embedded Systems Software

Software Requirements

See the *Vivado Design Suite User Guide: Release Notes, Installation, and Licensing* (UG973) for a complete list and description of the system and software requirements.

Tutorial Design Description

The small sample design used in this tutorial has a set of RTL design sources consisting of Verilog files, along with a PDF that describes how to add a document file to your IP.

Locating Tutorial Design Files

Download `ug1119-vivado-creating-packaging-ip-design.zip`, from the Reference Design Files on the Xilinx website.

Extract the zip file contents into any write-accessible location.

Introduction

In this lab, you define a new custom IP from an existing Vivado project, using the Create and Package IP wizard.

You start with an existing design project in the Vivado IDE, define identification information for the new IP, add documentation to support its use, and add the IP to the IP Catalog.

After packaging, you verify the new IP through synthesis in a separate design project.

The lab project contains Verilog source files for a simple UART interface.

Step 1: Open the Vivado Project

Launch Vivado.

On Linux:

- Change to the directory where the lab materials are stored: `cd <Extract_Dir>/lab_1`
- Launch the Vivado IDE: **`vivado`**

On Windows:

- Launch the Vivado Design Suite IDE:

Start > All Programs > Xilinx Design Tools > Vivado 2016.1> Vivado 2016.1

Or

- Click the Vivado 2016.1 desktop icon to start the Vivado IDE

The Vivado IDE Getting Started page displays with links to open or create projects, and to view documentation. For either Windows or Linux, continue the lab from this point.

1. Click Open Project, and browse to: <Extract_Dir>/`lab_1/my_simple_uart`

   Note: *Your Vivado Design Suite installation might have a different name on the Start menu.*

2. Select the `my_simple_uart.xpr` project and click **OK**.

   The design loads, and you see the Vivado IDE in the default layout view, with the Project Summary information as shown in the following figure.

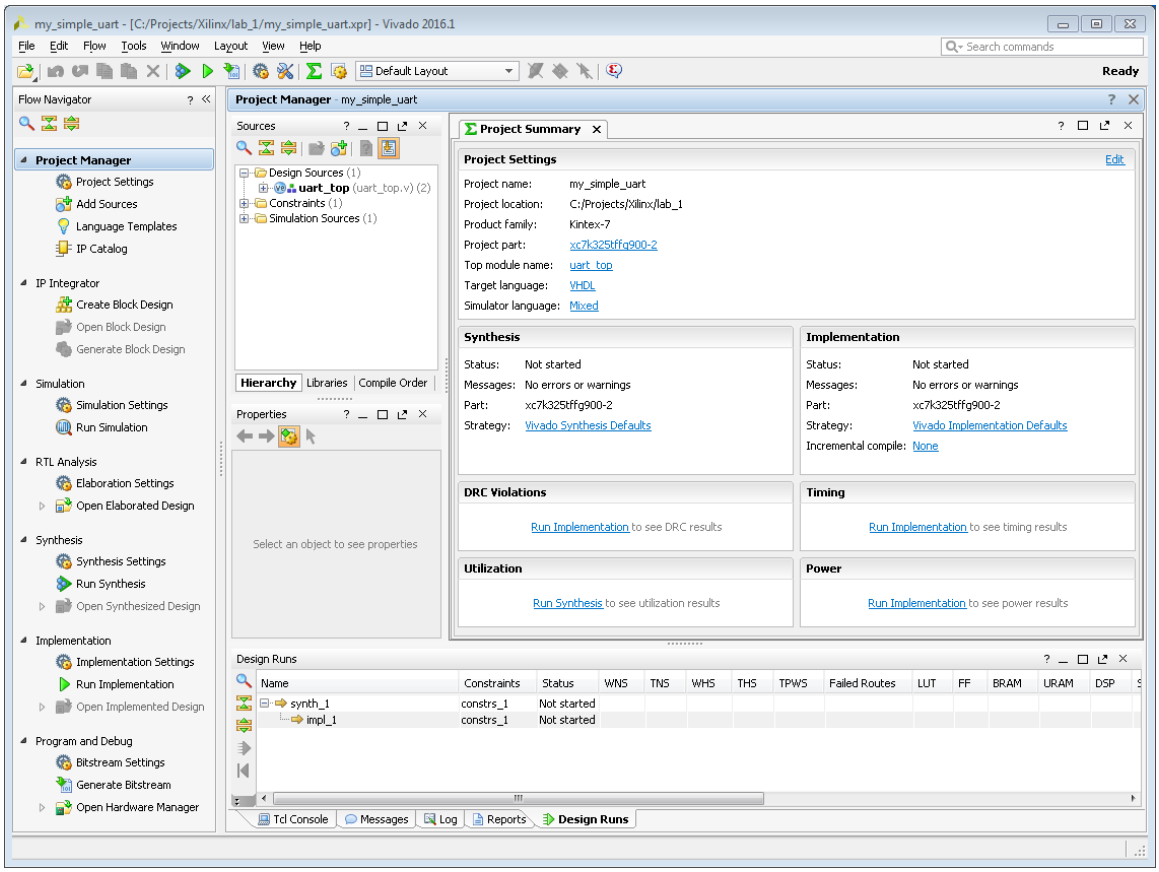


Figure 2: Project Default View Layout

Step 2: Preparing Design Constraints

The existing design includes timing constraints defined in an XDC file (`uart_top.xdc`). These constraints were defined for the UART design as a standalone design. However, when packaged as an IP, the design inherits some of the needed constraints from the parent design. In this case, you must modify the XDC file to separate constraints the IP requires when used in the context of a parent design, and the constraints the IP requires when used out-of-context (OOC) in a standalone capacity. This requires splitting the current XDC file.

You should prepare the design constraints prior to packaging the design for inclusion in the IP catalog; however, you can also perform these steps after packaging the IP.

IMPORTANT: The Vivado tools create a synthesized design checkpoint (DCP) as part of the default Out-of-Context (OOC) design flow for IP packaging and use.

To ensure that the packaged IP functions properly in the default Out-of-Context (OOC) design flow, the IP packaging must include a standalone XDC file to define all external clocking information for the IP.

Vivado synthesis uses the standalone XDC file in the OOC synthesis run to constrain the IP to the recommended clock frequency.

When used in the context of a top-level design, the parent XDC file provides the clock constraints and the standalone OOC XDC file is not needed.

For more information on the Out-Of-Context (OOC) design flow, and the use of the DCP file, see the *Vivado Design Suite User Guide: Designing with IP* (UG896).

TIP: *Depending on the function and use of the packaged IP, you might need to adjust the design constraints to ensure proper scoping. For more information, See "Constraints Scoping" in the Vivado Design Suite User Guide: Using Constraints (UG903).*

Analyze the Current Constraints Files

1. In the Hierarchy pane of the Sources window, open the target XDC file (`uart_top.xdc`) under the `/Constraints` folder.

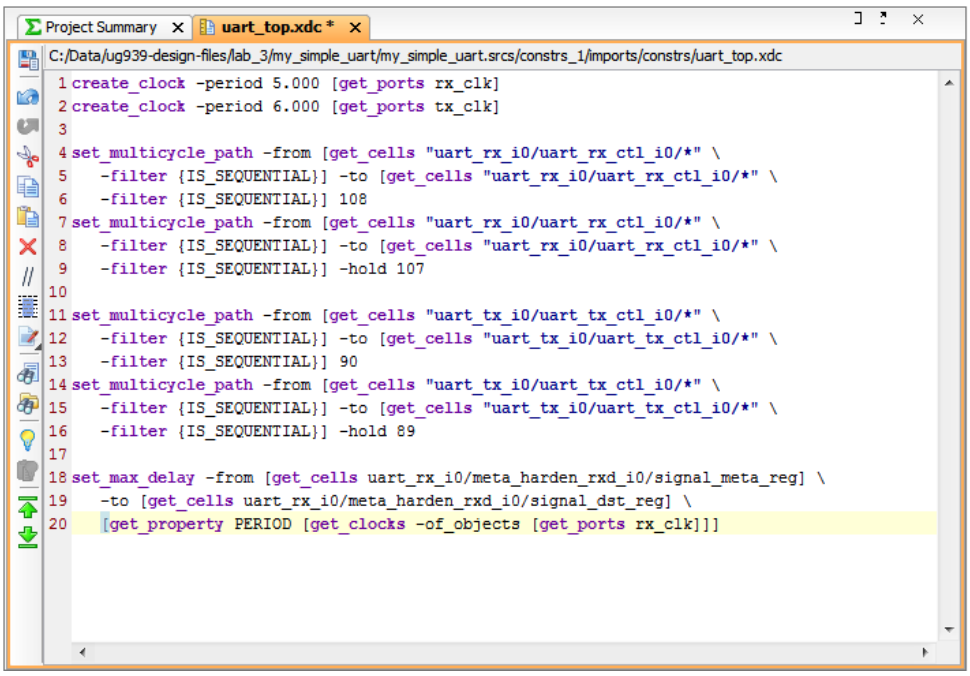


Figure 3: File Contents of uart_top.xdc

There are two items to take note of in the XDC file, as seen in , above.

- `create_clock` constraints (Lines 1 and 2)
- `set_max_delay` constraint relying on the clock object period value (line 18).

Note: *The line numbers referenced in Figure 3 might differ from the line numbers in your XDC file because the constraints were edited for easier viewing in this tutorial.*

2. Examine all `create_clock` constraints prior to packaging the new IP definition.

If the created clock is internal to the IP (GT), or if the IP contains an input buffer (IBUF), the `create_clock` constraint should stay in the IP XDC file because it is needed to define local clocks.

In the next sub-step, you move clocks that are not internal, or local, to the IP from the IP XDC file to an OOC XDC file, because they are provided by the parent design.

For this example, you move the `create_clock` constraints on line 1 and 2 from the design XDC file to an OOC XDC file. When a user instantiates the IP you are packaging from the IP catalog into a design, the IP inherits the clock definitions from the parent design.

The `set_max_delay` constraint is also noteworthy in that it has a dependency on the PERIOD property of defined clocks, (`get_clocks -of_objects`). This dependency is affected by the order of processing of the constraints of the IP and top-level design.

By default, when IP customizations are instantiated into a design, the Vivado IDE processes the XDC files of an IP before the XDC files of the top-level design. This is known as EARLY processing, and is defined by the `PROCESSING_ORDER` property on the XDC file.

By default, the XDC files of the top-level design are marked for **NORMAL** processing. This means that the processing of XDC files for IP constraints happens before the top-level design constraints created by the user. However, in the case of the `set_max_delay` constraint, the dependency on the clock `PERIOD` will cause errors in processing the IP constraints early and defining the clock later.

3. To resolve this issue, you mark the XDC files of the UART IP for LATE processing.

> ***TIP:*** *Xilinx delivered IP with* _clock *appended to the XDC filename are all marked for* LATE *processing.*

Creating an Out-Of-Context (OOC) XDC file

1. From the Flow Navigator, or from the File menu, select **Add Sources**, or select the **Add Sources** button .

   The Add Sources dialog box opens.

2. Select Add or Create Constraints, and click **Next**.

3. In the Add or Create Constraints dialog box, click the **Add an Existing or Create file** button .

4. In the Create Constraints File dialog box, fill in the constraints file information, as shown in the following figure.
   - File type: **XDC**
   - File name: `uart_top_ooc.xdc`
   - File location: **<Local to Project>**
5. Click **OK**.

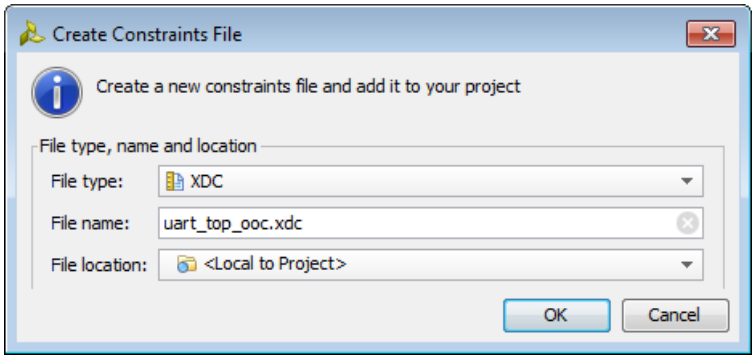


Figure 4: Create Constraints File Dialog Box

TIP: For Xilinx-delivered IP, the out-of-context XDC file has _ooc appended to the filename; however, it is the USED_IN property of the file that determines if it is an OOC XDC file, not the filename.

6. Click **Finish** to complete the Add Sources dialog box.

   The Vivado tools create a new XDC file in the project and displays the file under the Constraints section in the Hierarchy pane of the Sources window.

   You now move the create_clock constraints from the XDC file of the original design (`uart_top.xdc`) into the OOC XDC file (`uart_top_ooc.xdc`).
7. In the Sources window, open the new OOC XDC file (`uart_top_ooc.xdc`) by double-clicking the file. The file is empty.
8. Cut and paste the `create_clock` constraints, from lines 1 and 2 of the IP XDC file (`uart_top.xdc`) into the empty OOC XDC file.

   The OOC XDC file contains only the two `create_clock` constraints.

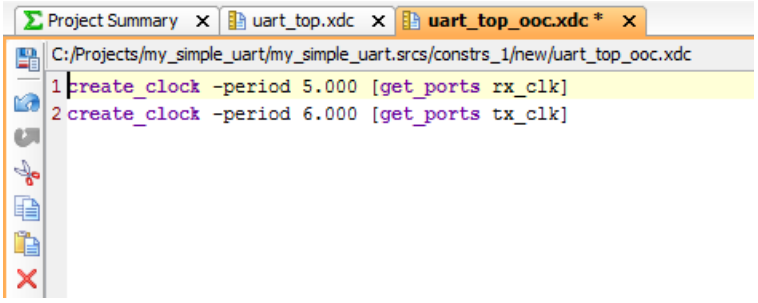


Figure 5: OOC XDC

9. Select the **Save File** button to save the updated contents of the OOC XDC file.
10. Check to be sure that the `create_clock` commands are removed from the IP XDC file (`uart_top.xdc`), and save the file.

    The `create_clock` constraints are not necessary because parent design defines the clocks. The IP XDC file should now only contain the constraints, as shown in the following figure. The OOC XDC file defines the clocks needed for standalone processing.

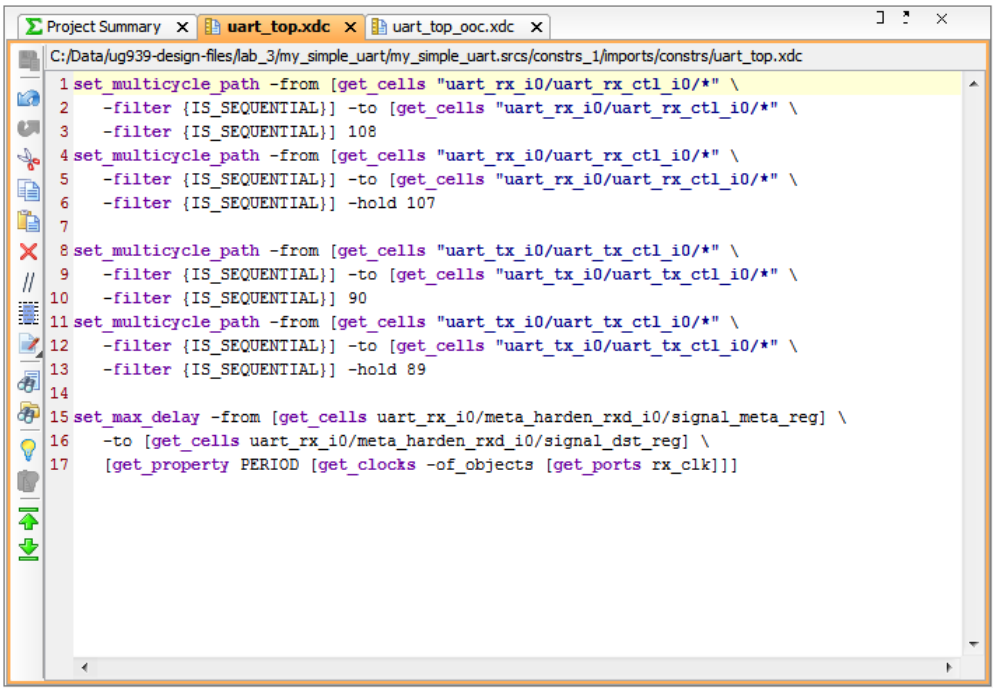


Figure 6: Updated uart_top.xdc

11. Close the two open XDC files.

    With the OOC and IP XDC files defined, you must set the USED_IN and PROCESSING_ORDER properties on the XDC files so that the Vivado Design Suite correctly processes the constraint files for the IP.

12. In the Hierarchy pane of the Sources window, select the OOC XDC file (`uart_top_ooc.xdc`) listed under the Constraints section.

13. Right-click the file, and select **Source File Properties**.

14. From the Source File Properties window, scroll down and select the USED_IN property value to open the **Make Selection** dialog box.

15. Select **out_of_context** in the unused values and select the Move right button, to add the value to the USED_IN property.

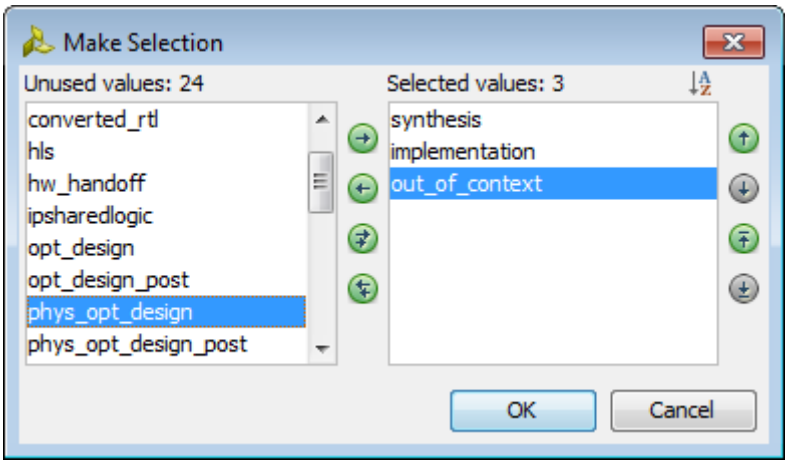


Figure 7: Make Selection Dialog Box

16. *Optional*: You can adjust the `USED_IN` property in the Tcl Console. To set the `USED_IN` property of the OOC XDC file to include the "out_of_context" using the following Tcl command:

```
set_property USED_IN {synthesis implementation out_of_context} \
[get_files uart_top_ooc.xdc]
```

17. When the `USED_IN` property includes the `out_of_context` setting, the XDC file is only used for synthesis or implementation in Out-of-Context runs (`-mode out_of_context`).

IMPORTANT: *The `USED_IN` property for an OOC XDC file should be {synthesis implementation out_of_context}. If it is just out_of_context, it is not used during synthesis or implementation.*

Setting the Processing Order for the IP XDC

1. In the Hierarchy pane of the Sources window, select the IP XDC file (`uart_top.xdc`) listed under the Constraints section.
2. Right-click the file, and select **Source File Properties** from the menu.
3. From the Source File Properties window, scroll down and change the **PROCESSING_ORDER** property value to **LATE**, as shown in the following figure.

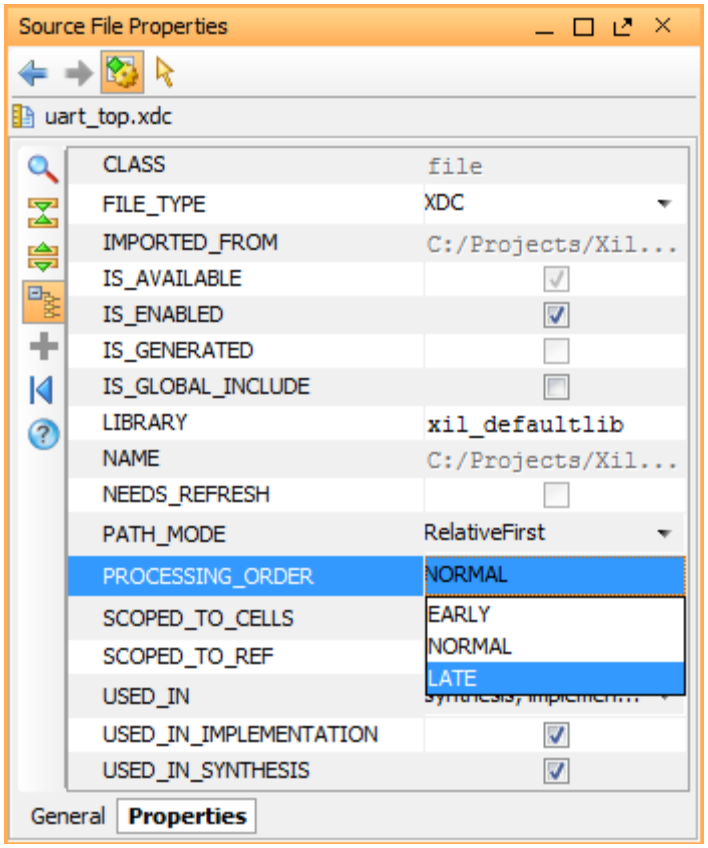


Figure 8: Source File Properties

The property value can also be changed in the Tcl Console with the following Tcl command:

```
set_property PROCESSING_ORDER LATE [get_files uart_top.xdc]
```

After completing the above steps, the XDC files are correctly prepared for packaging and the OOC design flow.

Step 3: Package the IP

After setting up the design and supporting constraint files, the next step is to create and package the new IP Definition, and add it to the IP Catalog.

1. From the Tools menu, select the **Create and Package IP** command to open the Create and Package IP Wizard.

   The Welcome window opens for the Create and Package New IP dialog box.

2. Click **Next**.

   The Choose Create Peripheral or Package IP dialog box opens, as shown in the following figure.

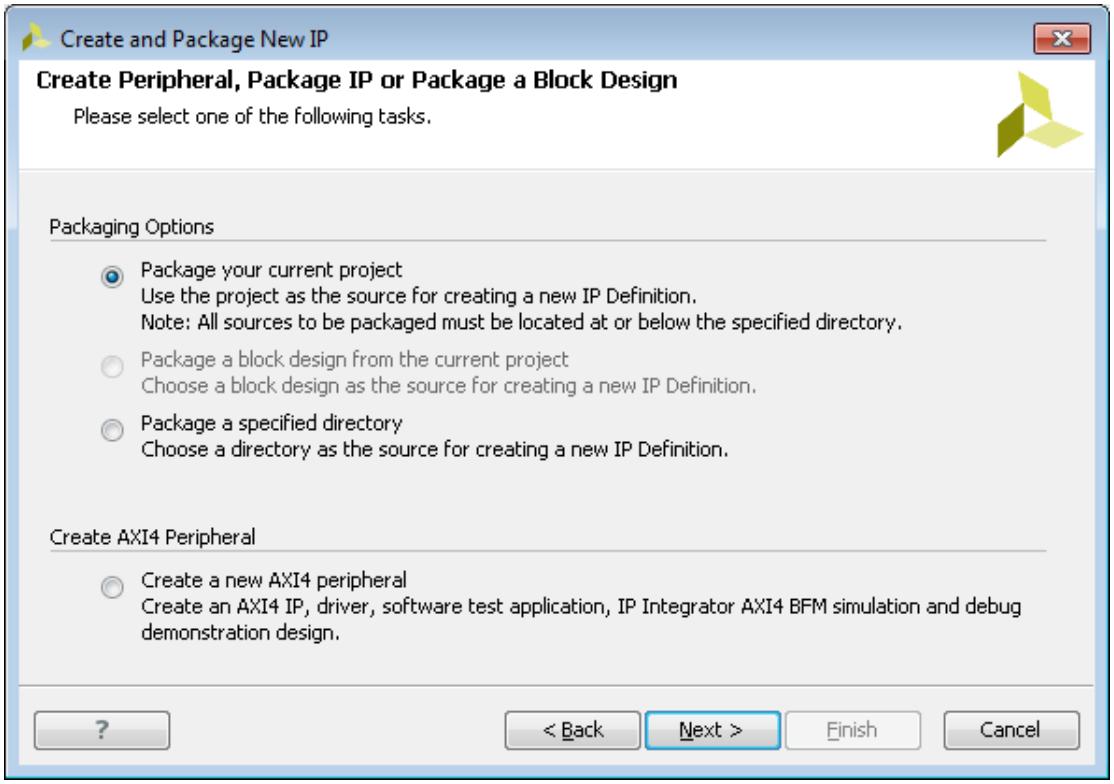


Figure 9: Choose Create Peripheral or Package IP Window

3. Select the **Package your current project** option to use the current project as the source for creating the new IP Definition.

4. Select **Next**.

The Package Your Current Project dialog box opens, as shown in the following figure.

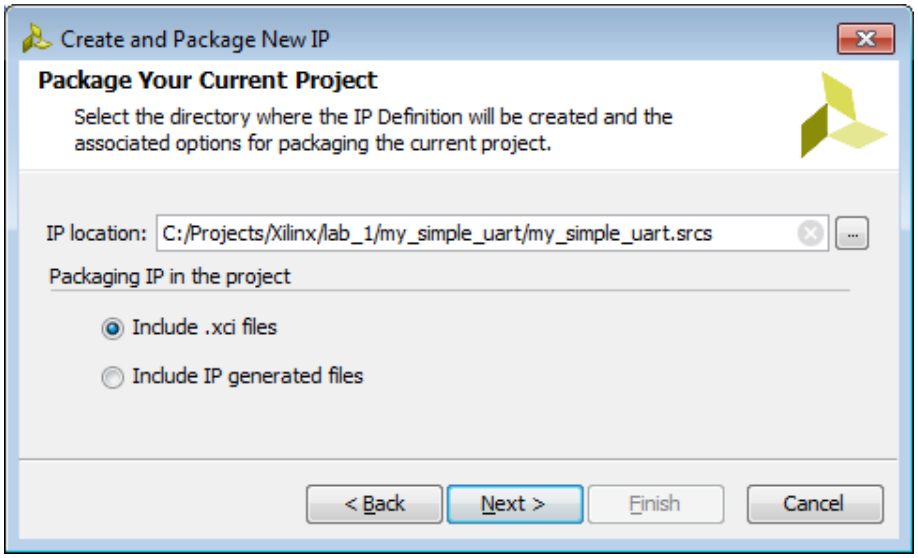


Figure 10: Package Current Project

5. Click **Next** to accept the defaults.

   The New IP Creation dialog box, as shown in the following figure, opens with a summary of the information the wizard will automatically gather from the project.

Figure 11: Begin IP Creation

6. Click **Finish**.

After the wizard completes, the Vivado IDE initially packages the current project as an IP for inclusion in the IP repository, and the Package IP dialog box opens to report success.

7. Click **OK**.

   The Package IP window opens and displays the basic IP package in a staging area for editing and repackaging, as seen in the following figure.

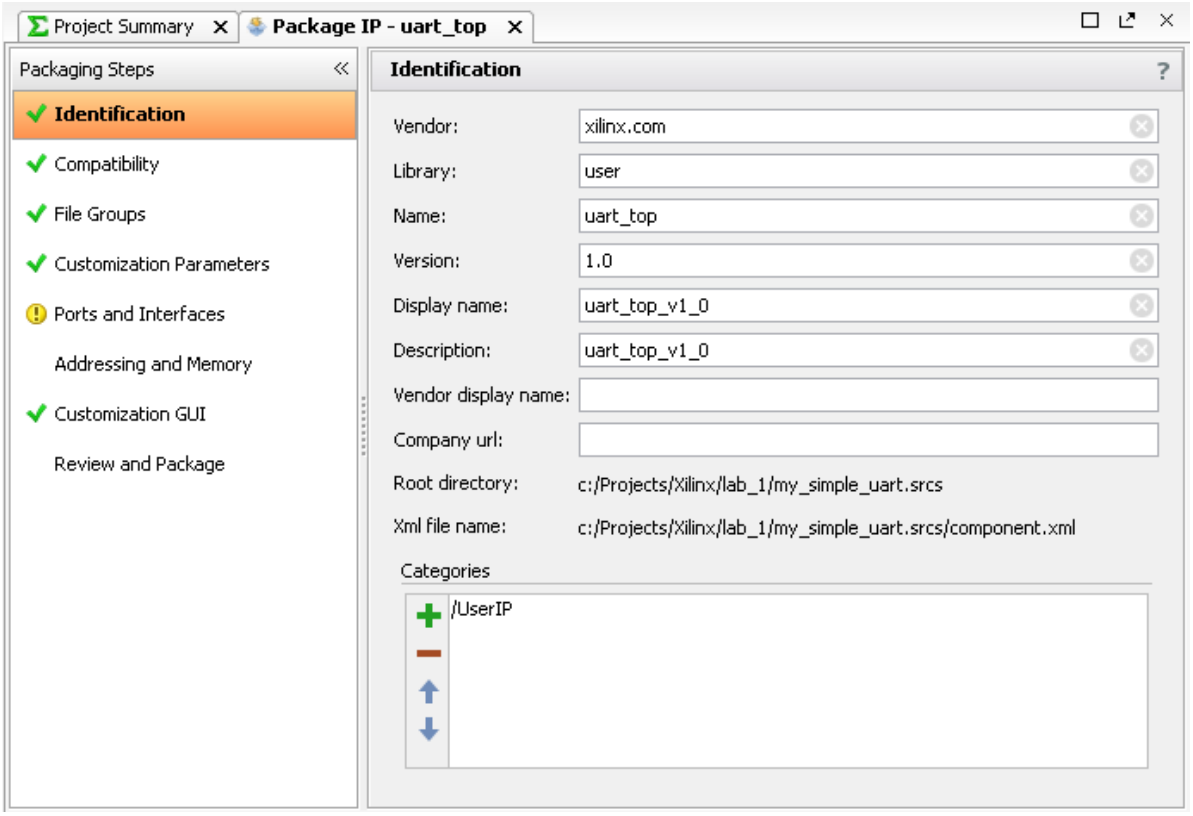


Figure 12: Editing the Default IP Definition

Step 4: Modify the IP Definition

The Package IP window shows the current IP identification information, including Vendor, Library, Name, and Version (VLNV) attributes of the newly packaged IP.

1. In the Package IP window, select the **Identification pane** in the left side panel, and fill in the right side with the following information:
   - **Vendor:** my_company
   - **Library**: user
   - **Name:** my_simple_uart
   - **Display name:** My Simple UART

- **Version**: 1.0
- **Description:** My simple example UART interface
- **Vendor display name:** My Company
- **Company url:** http://www.my_company_name.com

2. For the **Categories** option, select the Add button to open the Choose IP Categories dialog box, as shown in the following figure.

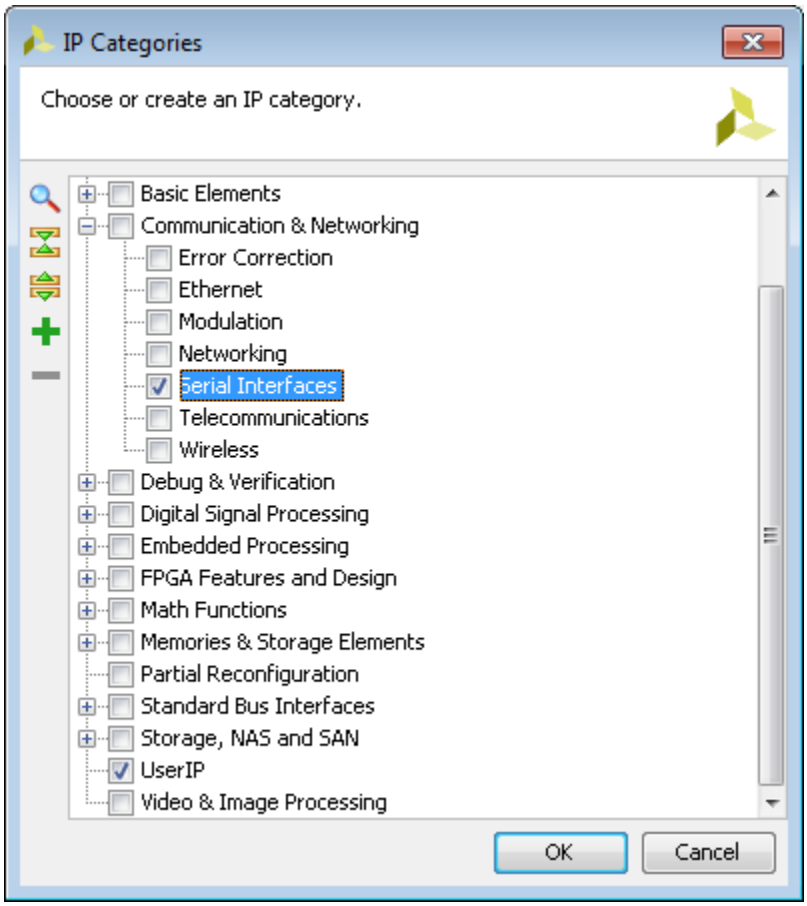


Figure 13: Choose IP Categories

The Choose IP Categories dialog box lets you select various appropriate categories to help classify the new IP definition. When you add the IP definition to the IP Catalog, the IP lists under the specified categories.

3. Select the **Serial Interfaces** box under **Communications & Networking** because the IP is a UART interface.
4. Click **OK**.

Step 5: Add a Product Guide to the IP

1. On the left side of the Package IP window, select the **File Groups** item to display the File Groups panel on the right side.

   The File Groups pane provides a listing of the files to be packaged as part of the IP.

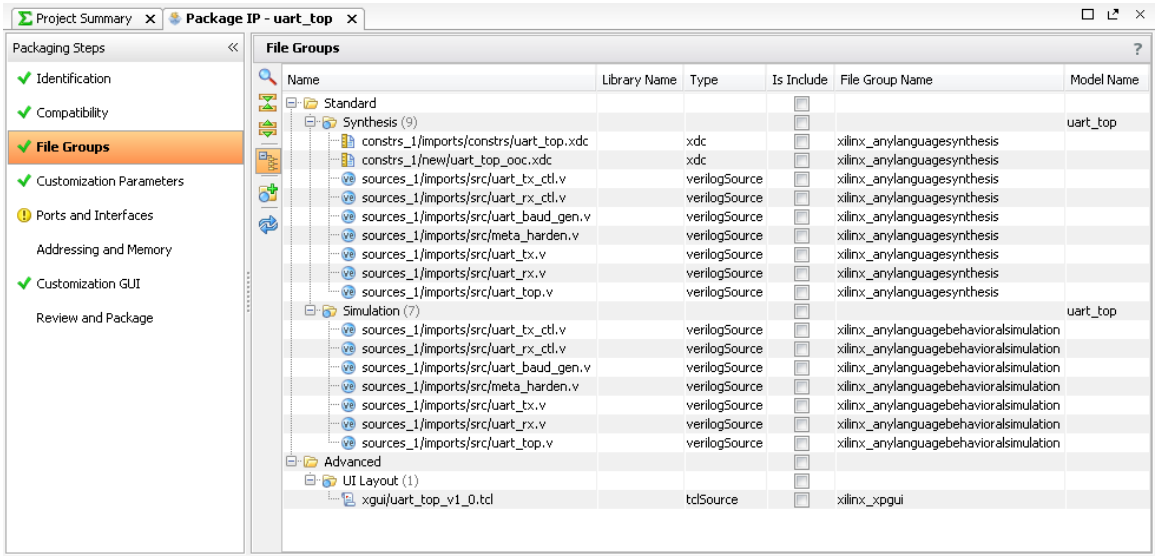


Figure 14: File Groups

2. Open the Messages window, and review the IP Packager messages as seen in the following figure.

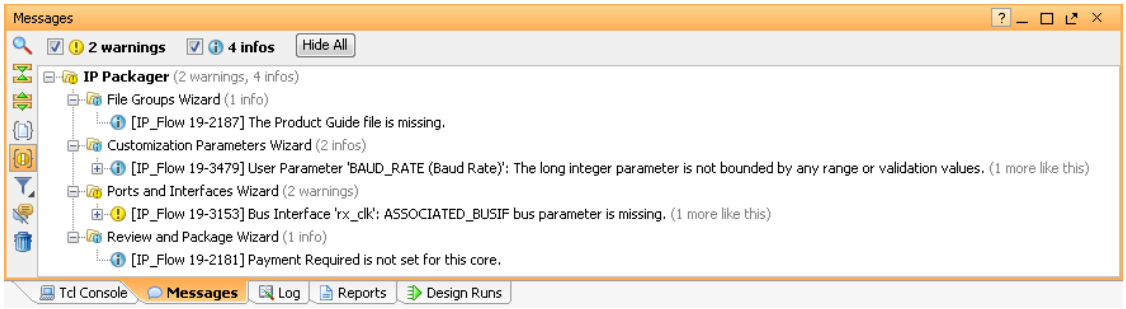


Figure 15: IP Packager Messages

The IP Packager messages inform you of the state of the IP. The File Groups Wizard message indicates that the IP definition does not include any documentation.

The Customization Parameters Wizard informs you that specific parameters of the IP do not have range values.

As `INFO` messages, these are quick checks of the IP definition that do not prevent you from moving forward if you choose. However, in the next step you add the product guide to the IP definition.

The Ports and Interfaces wizard has warnings related to the inferred single-bit clock interfaces inferred by the IP Packager for missing `ASSOCIATED_BUSIF` parameters. These parameters are required for AXI interfaces in the Vivado IP integrator, but you can ignore them for this exercise.

3. In the Package IP window, right-click in the File Groups pane, and select **Add File Group**.
4. In the Add IP File Group dialog box, select **Product Guide** from the Standard File Groups section, as shown in in the following figure.

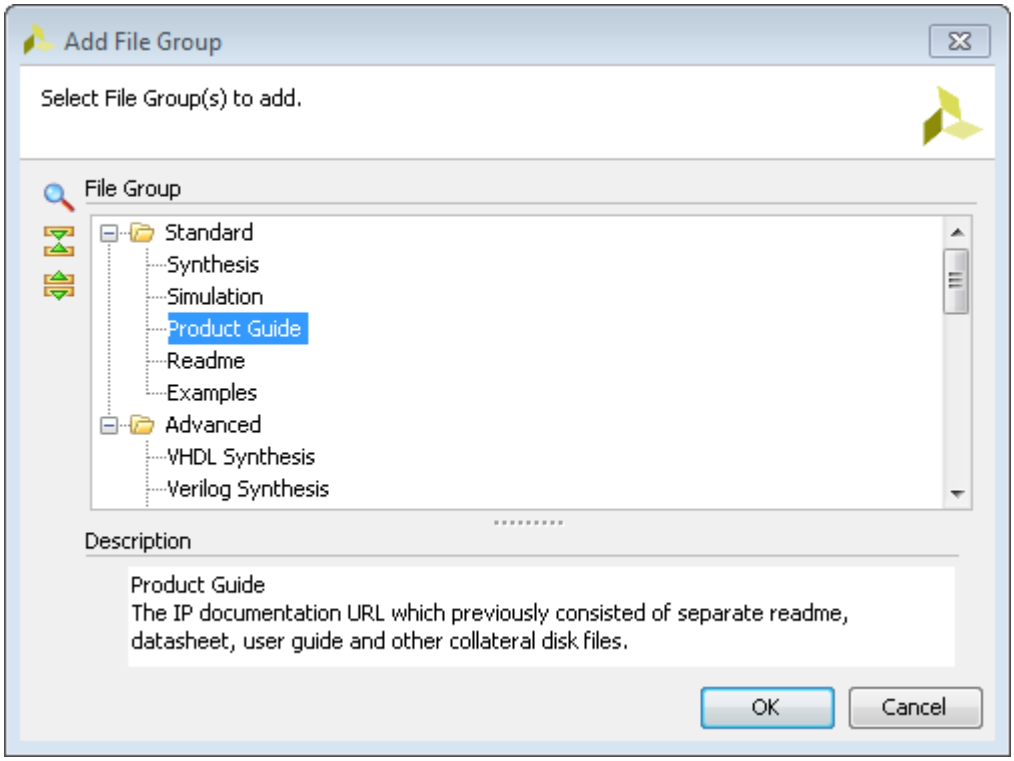


Figure 16: Add IP File Group – Product Guide

5. Click **OK**.

   The IP File Groups pane now updates with the Product Guide group in the list. There is a 0 next to the Product Guide name because there are no files added to the newly created group.

   Note: *A critical warning opens when you add the Product Guide file group, noting that the file group is empty.*

5. Right-click the **Product Guide** file group, and select **Add Files**.
6. In the opened Add IP Files (Product Guide) dialog box, click **Add Files**.
7. Browse to `<Extract_Dir>/lab_1/my_simple_uart/docs`, and select **All Files** in the **Files of type**: entry line.
8. Select my_simple_uart_product_guide.pdf, and click **OK.**

9. In the Add IP Files (Product Guide) dialog box, shown in the following figure, ensure that **Copy sources into project** is selected.

   The option ensures that the file is imported in the project sources directory, and not remotely referenced by the IP packager.

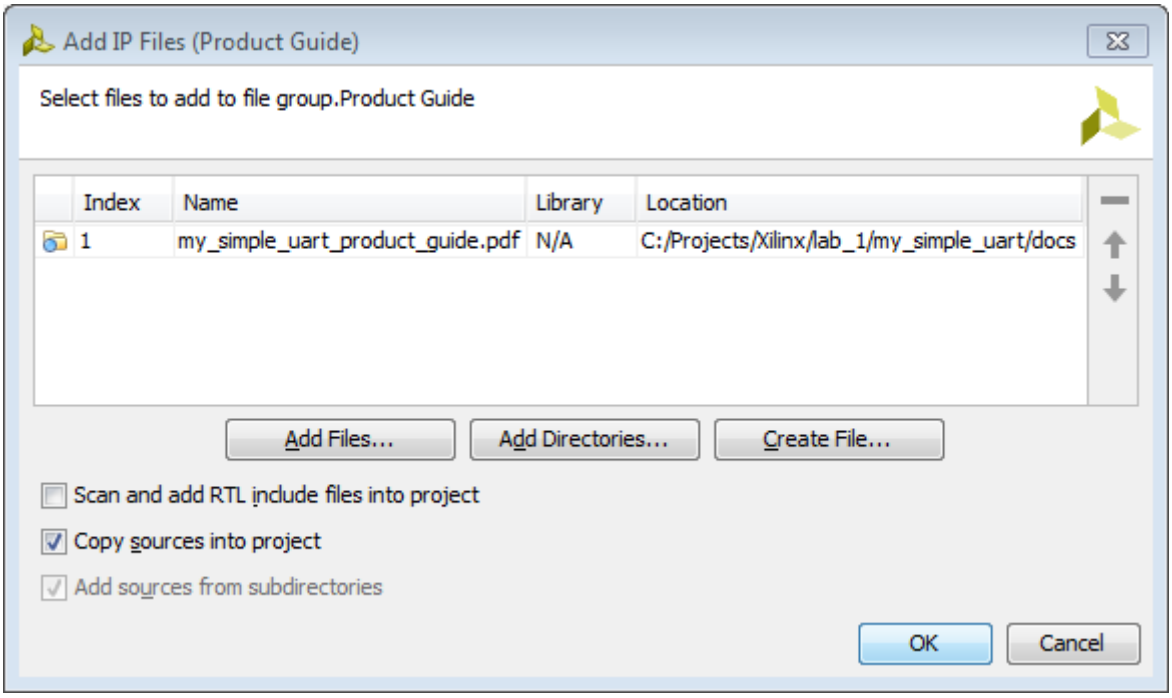


Figure 17: Add Product Guide

10. Click **OK**.

    The IP Packager adds the PDF file of the Product Guide to the files defined as part of the IP, and resolves the Critical Warning.

Step 6: Review and Package the IP

The custom IP was initially packaged at the end of the Create and Package IP wizard, but because changes were made in the Package IP window, the custom IP must be re-packaged for the changes to take effect.

1. On the left side of the Package IP window, select the **Review and Package** panel.

   The Review and Package pane provides a summary of the IP being packaged, as shown in the following figure.

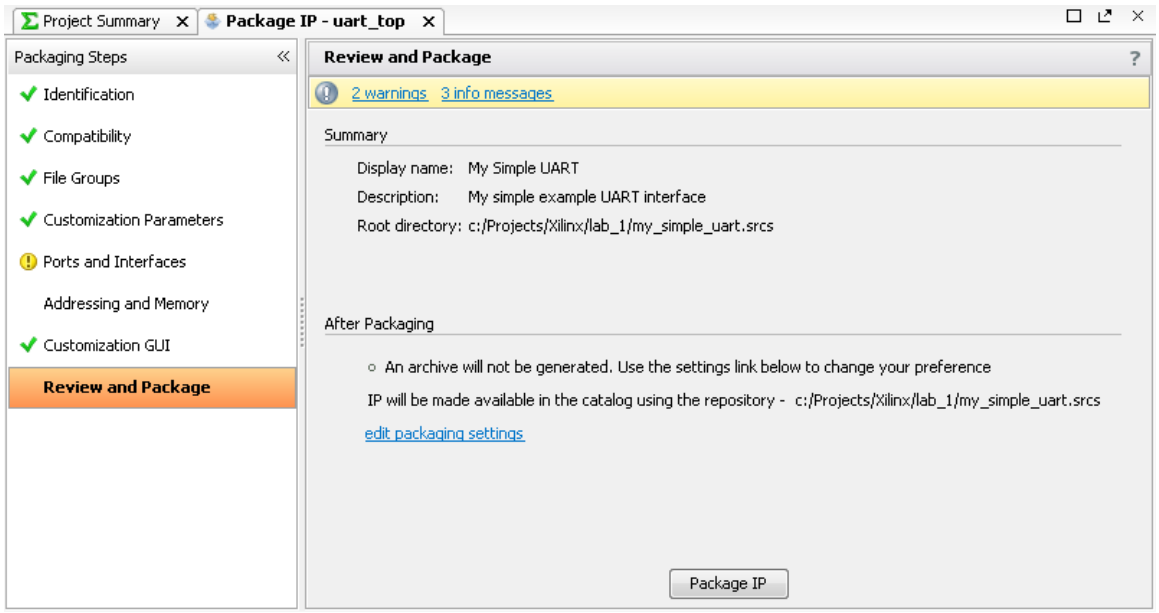


Figure 18: Review and Package IP

   With default settings of the current project, Vivado does not generate an archive for this IP after packaging. This is reflected in the **After Packaging** section of the Review and Package pane of the Package IP window.

2. Make a note of the location of the IP repository in the After Packaging section. This is necessary to validate the custom IP in the next step.
3. In the Package IP window, click **Package IP** to package the current project and add it to the IP Catalog.
4. After the packaging process completes, close the Vivado project.

Step 7: Validate the New IP

With the new custom IP definition packaged and added to the IP Catalog, you can validate that the IP works as expected when added to designs. To validate the IP, add a new customization of the UART IP to a project, and synthesize the design.

1. From the Vivado IDE Getting Started page, select **Manage IP** > **New IP Location** to create a new project.

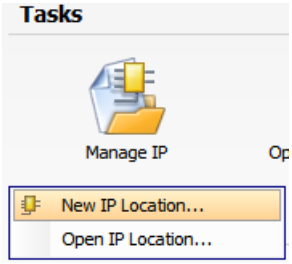


Figure 19: New Manage IP Project

TIP: *You can use either an RTL project or a Manage IP project to validate IP.*

2. Click **Next** in the New IP Location dialog box.
3. In the Manage IP Settings dialog box, set the following options as they appear in Figure 20.
   - **Part**: xc7k325tffg900-2
   - **Target language**: Verilog
   - **Target Simulator**: Vivado Simulator
   - **Simulator Language**: Mixed
   - **IP Location:** <Extract_Dir>/lab_1

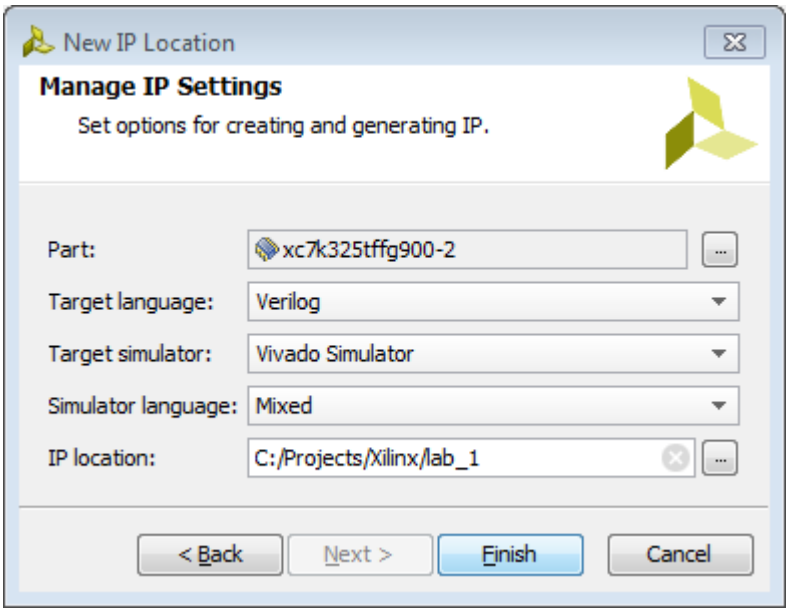


Figure 20: Manage IP Settings

4. Click **Finish** to create the Manage IP project.

   A new Manage IP project opens in the Vivado IDE. The IP Catalog opens automatically in a Manage IP project; however, the IP Catalog does not contain the repository used to package the custom UART IP.

   You now add the IP repository to the IP Catalog.

5. In the IP Catalog window, right-click and select **IP Settings**.

   The **Tools > Project Settings > IP dialog** box opens.

6. In the Repository Manager tab, click the **Add Repository** button to open the IP Repositories Dialog Box.

7. In the IP Repositories dialog box, **browse** to and **select** the following location:

   `<Extract_Dir>/lab_1/my_simple_uart/my_simple_uart.srcs`

5. Click **Select** to add the selected repository, as shown in the following figure.

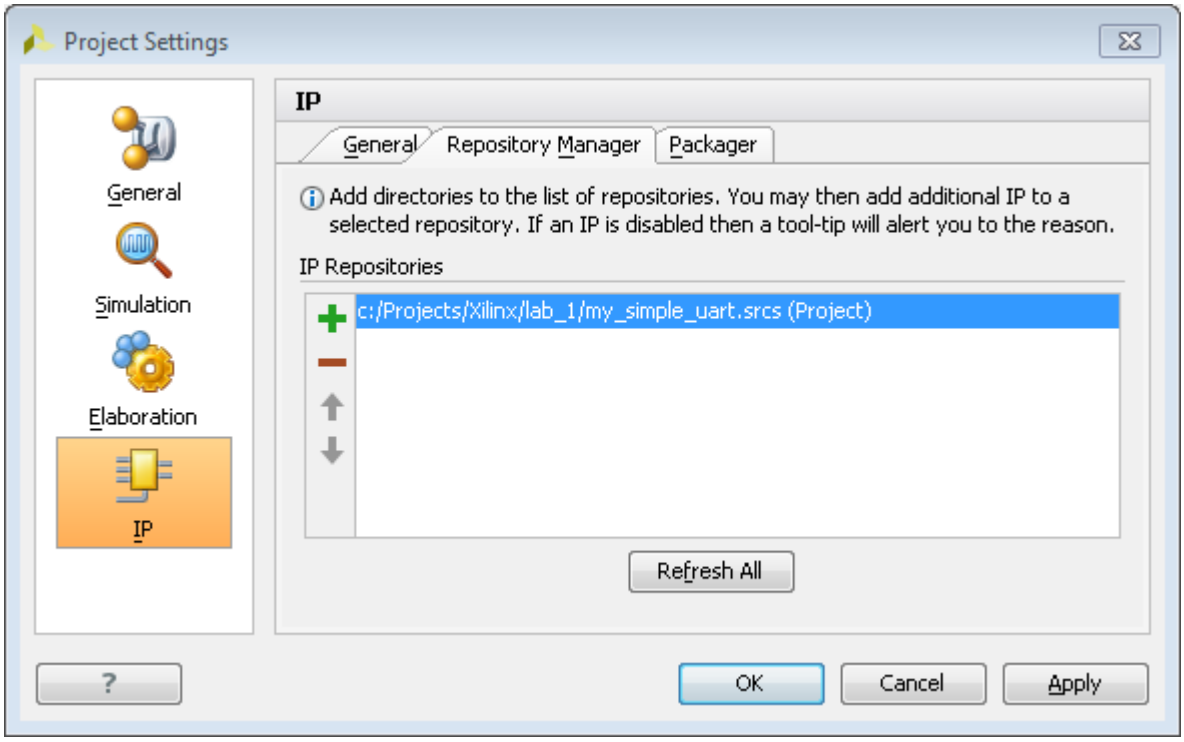


Figure 21: Manage IP Repository

As seen in the previous figure, the added location displays in the **IP Repositories** section, and any packaged IP found in the repositories displays under the **IP in Selected Repository**. The **My Simple UART** IP definition that you packaged in Step 3: Package the IP Directory is listed.

6. Click **OK** to add the IP repository to the IP Catalog and close the dialog box.

TIP: To define a custom IP repository for use across multiple design projects you can use the **Tools > Options** command in the Vivado IDE to set the Default IP Repository Search Paths under the General options. The default IP repository search path is stored in the `vivado.ini` file, and added to new projects using the IP_REPO_PATHS property for the current_fileset:

```
set_property IP_REPO_PATHS {…} [current_fileset]
```

the Vivado Properties Reference Guide (UG912) *for more information.*

---

8. In the search field at the top of the **IP Catalog**, type **UART**.

The My Simple UART is reported under the /UserIP and Serial Interfaces categories that it was previously assigned to during packaging.

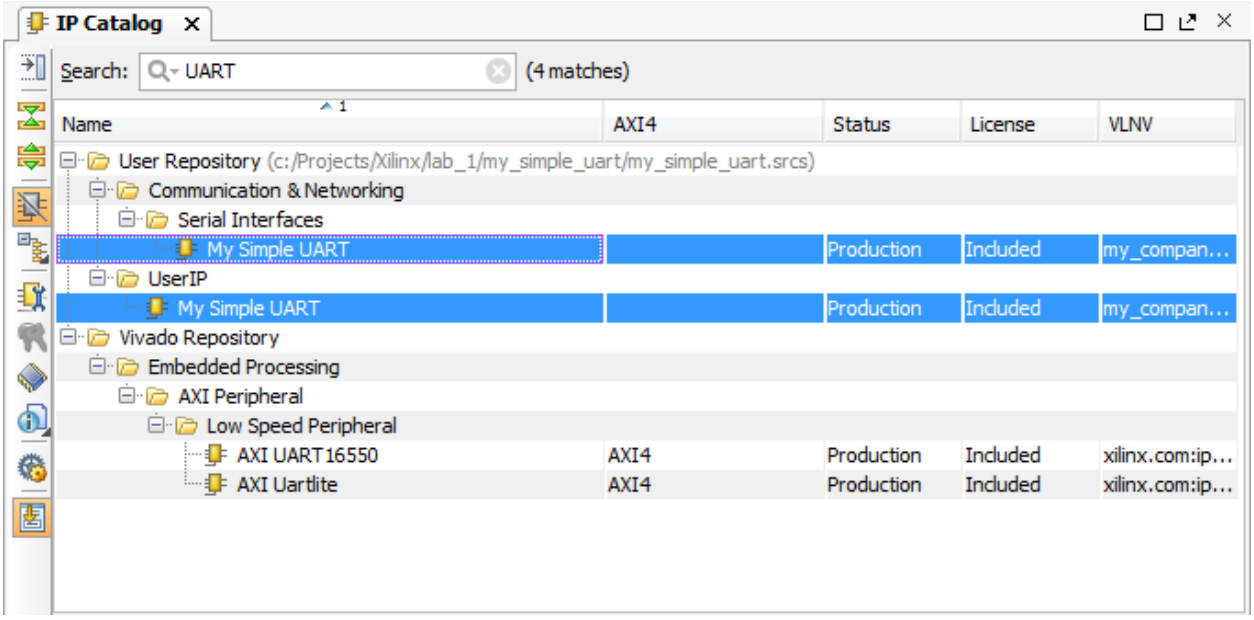


Figure 22: Search IP Catalog for UART

Note: *This IP Catalog view shows when the Taxonomy and the Repository options are selected for Grouping the IP. See the Vivado Design Suite: Creating and Packaging Custom IP (UG1118) for more information about IP Groups.*

9. Select the **My Simple UART** by clicking it under either the `/UserIP` or `/Serial Interfaces` category.
10. Examine the Details pane of the IP Catalog window, as shown in the following figure.

    Notice the details match the information provided when you packaged the IP.

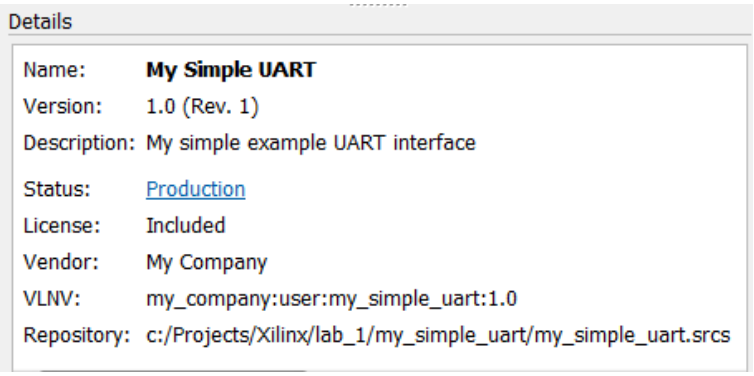


Figure 23: My Simple UART - Details

11. In the IP Catalog, double-click **My Simple UART** to open the Customize IP dialog box, shown in the following figure.

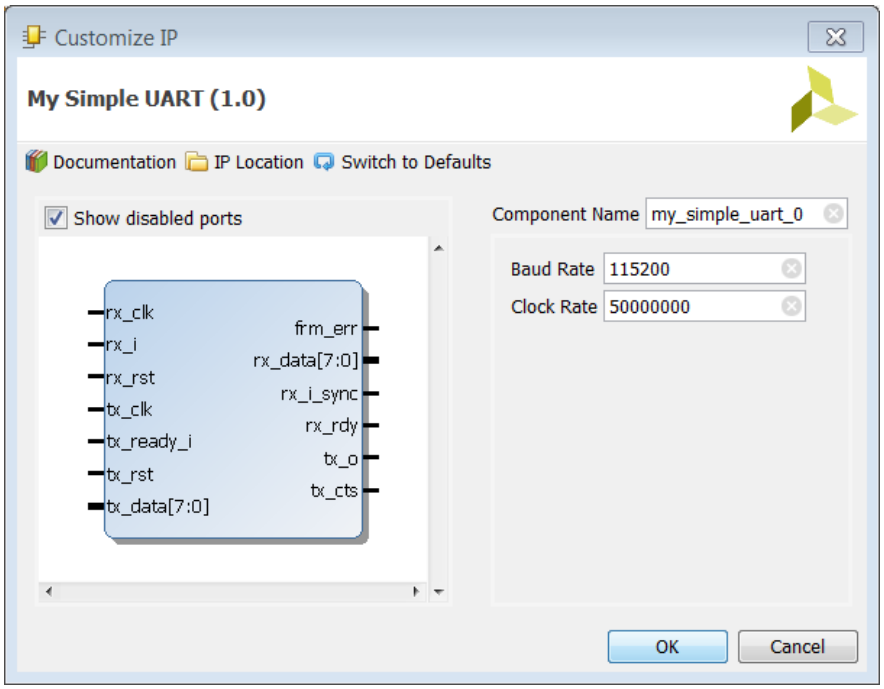


Figure 24: Customize IP – My Simple UART

12. *Optionally*: In the **Customize IP** dialog box, click **Documentation** and open the **Product Guide**.
13. Click **OK**, accepting the default Component Name and other options.

    The Vivado packager adds the customized IP to the current project, and displays the IP in the IP Sources window.

The Generate Output Products dialog box opens, as shown in the following figure.

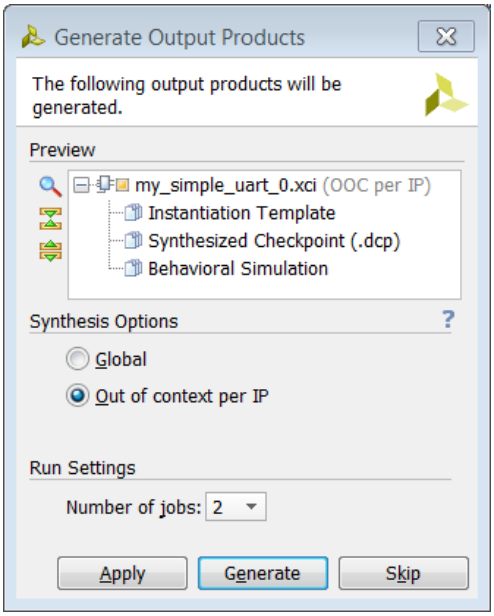


Figure 25: Generate Output Products

14. Click **Generate**.

    This generates the various files required for this IP in the current Manage IP project, and launches an out-of-context (OOC) synthesis run for the IP, which creates a design checkpoint (DCP) file.

    Recall this OOC synthesis run uses the OOC XDC file that defines the necessary clocks for the standalone IP.

    The Generate Output Products dialog box re-opens to report the output products were generated successfully.

16. Click **OK**.

17. Examine the IP Sources window and the various design and simulation source files that are added to the project.

18. In the Design Runs window, shown in the following figure, verify that the Out-Of-Context synthesis run was successful.

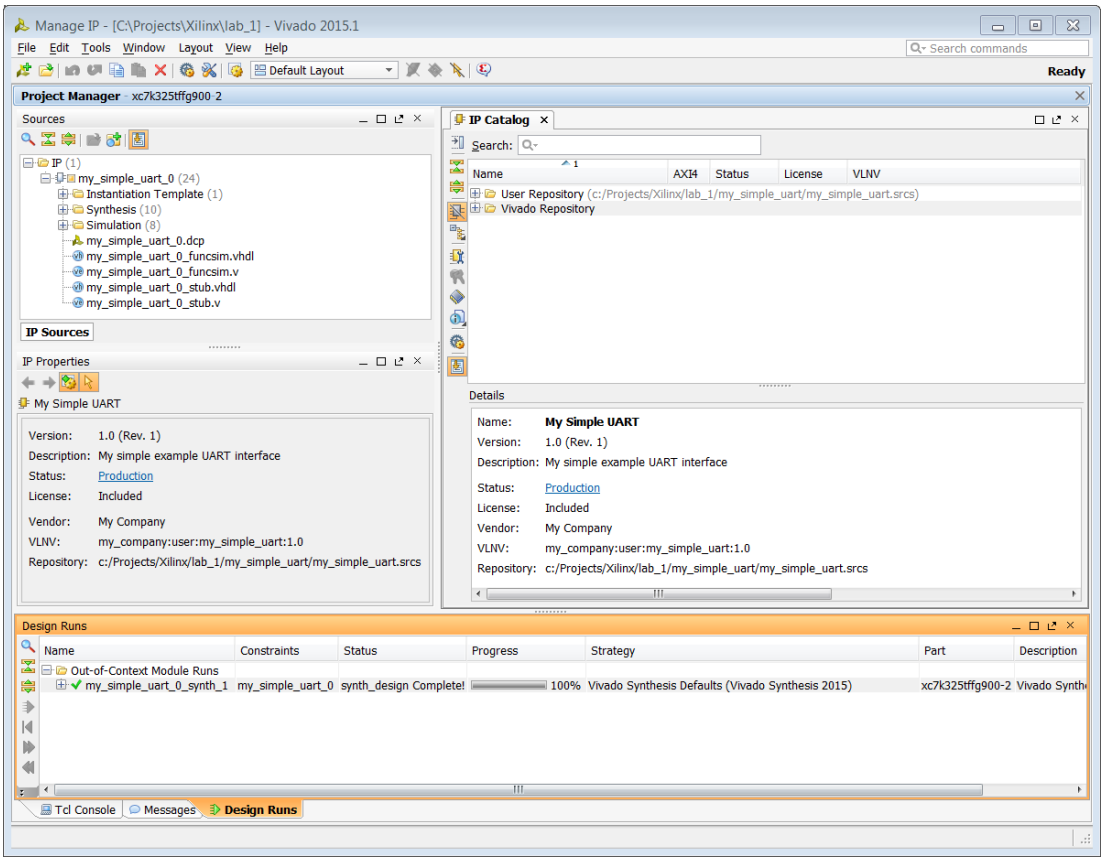


Figure 26: Validate IP in Managed IP Project

Conclusion

In this Lab, you did the following:

- Used the Create and Package IP wizard to create a custom IP definition for the tutorial project, `my_simple_uart`.
- Setup the XDC files to support the processing order requirements as well as Out-Of-Context synthesis.
- Validated the packaged IP by creating a Managed IP project, and then adding the new IP repository to the IP Catalog.
- Created a customization of the IP, and generated a DCP of the IP to validate that the IP definition was complete and included all the necessary files to support using the IP in other designs.

Introduction

In this lab, you will create a new Vivado project and package a custom IP from a specified directory.

You start with an IP repository directory and create a new Vivado project. In the Vivado project, you package the custom IP in the repository using the Create and Package Wizard, define the identification information, and verify the packaged files.

After packaging, you validate the IP was created successfully by completing Synthesis in the created Vivado project.

The lab project contains source files for a non-working version of the Wave Generator example design.

Step 1: Examine the IP Directory

1. Examine the `<Extract_Dir>/lab_2/custom_ip_repo/wave_gen_v1_0` location.

   This directory contains the custom IP files required for packaging the IP. Notice the three directories are created, as shown in the following figure:

   - `doc`: Directory contains the documentation related to the custom IP.
   - `src`: Directory contains the synthesis and simulation sources for the custom IP.
   - `tb`: Directory contains the testbench for the custom IP.

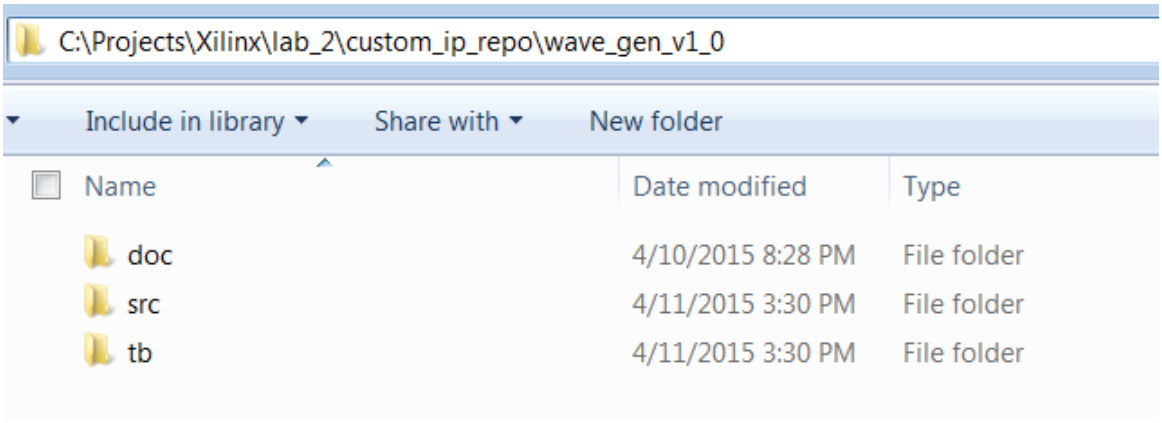


Figure 27: Lab 2 Directory Structure

The directory containing the custom IP should be organized to ensure proper packaging.

When specifying a directory for packaging, there are inference rules that assist in packaging the IP correctly. For more information, see the *Vivado Design Suite: Creating and Packaging Custom IP* (UG1118).

Examine the files in each of the directories for more information about the custom IP.

Step 2: Create a New Vivado Project

Launch Vivado

Launch Vivado.

On Linux:

- Change to the directory where the lab materials are stored: cd `<Extract_Dir>/lab_2`
- Launch the Vivado IDE: **vivado**

On Windows:

- Launch the Vivado Design Suite IDE:
  Start > All Programs > Xilinx Design Tools > Vivado 2016.x > Vivado 2016.x

Or

- Click the Vivado 2016.x desktop icon to start the Vivado IDE.

The Vivado IDE Getting Started page displays with links to open or create projects, and to view documentation. For either Windows or Linux, continue the lab from this point.

Create a New Project

1. From the Vivado IDE Getting Started page, select **Create New Project** to create an empty Vivado project.

   A new or existing project is required to creating and packaging a custom IP. The project information is used for populating certain fields in the Package IP window.

2. Click **Next** at the New Project wizard dialog box.

3. In the Project Name page, as shown in the following figure, set the following options for the project location:
   - **Project name**: `project_lab2`
   - **Project location**: `<Extract_Dir>/lab_2`

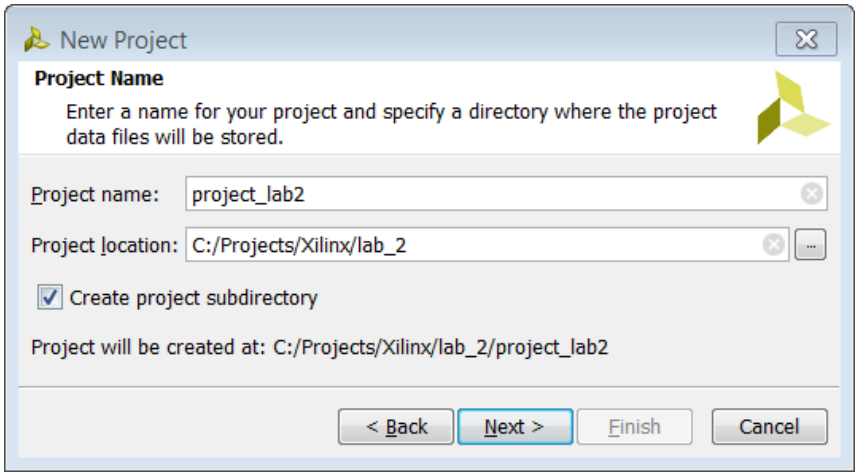


Figure 28: New Project – Project Name

4. Click **Next**.
5. Select RTL Project as the **Project Type** and **Do not specify sources at this time**.
6. Click **Next**.
7. In the Default Part dialog box, select the **xc7k70tfbg484-2** part, and click **Next**.

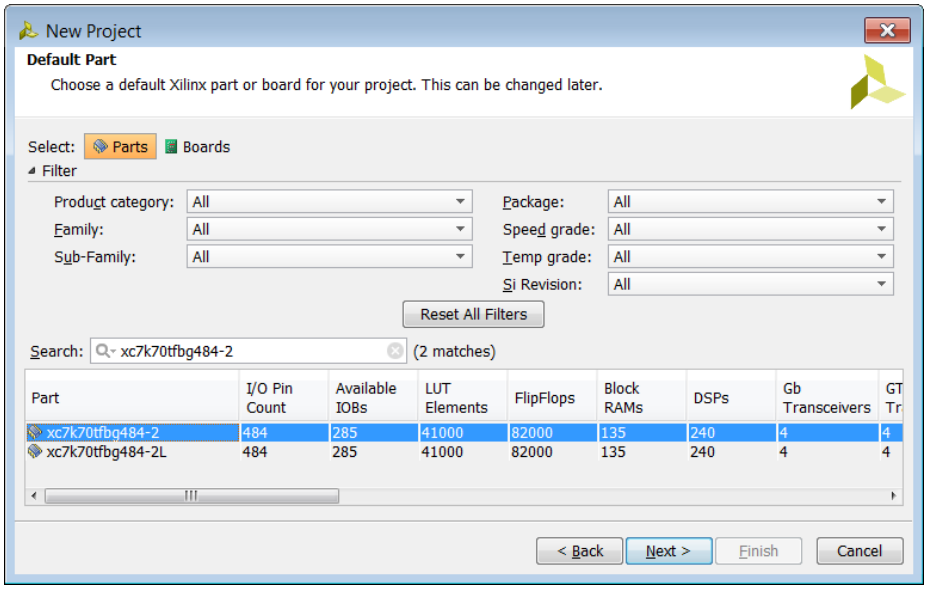


Figure 29: New Project – Default Part

8. For this lab, you select a Kintex®-7 device. This device family is used for the initial compatibility of the custom IP.
9. Click **Finish** to close the New Project Summary page, and create the project.

   The Vivado IDE opens project_lab2, with the default layout.

Step 3: Package the IP Directory

After creating the new empty project, the next step is to create and package the custom IP directory.

1. From the Tools menu, select **Create and Package IP** to open the Create and Package IP Wizard.
2. Click **Next** at the Welcome screen for the Create and Package New IP dialog box, shown in the following figure.
3. In the Create Peripheral, Package IP, or Package a Block Design dialog box, select **Package a specified directory**, and click **Next**.

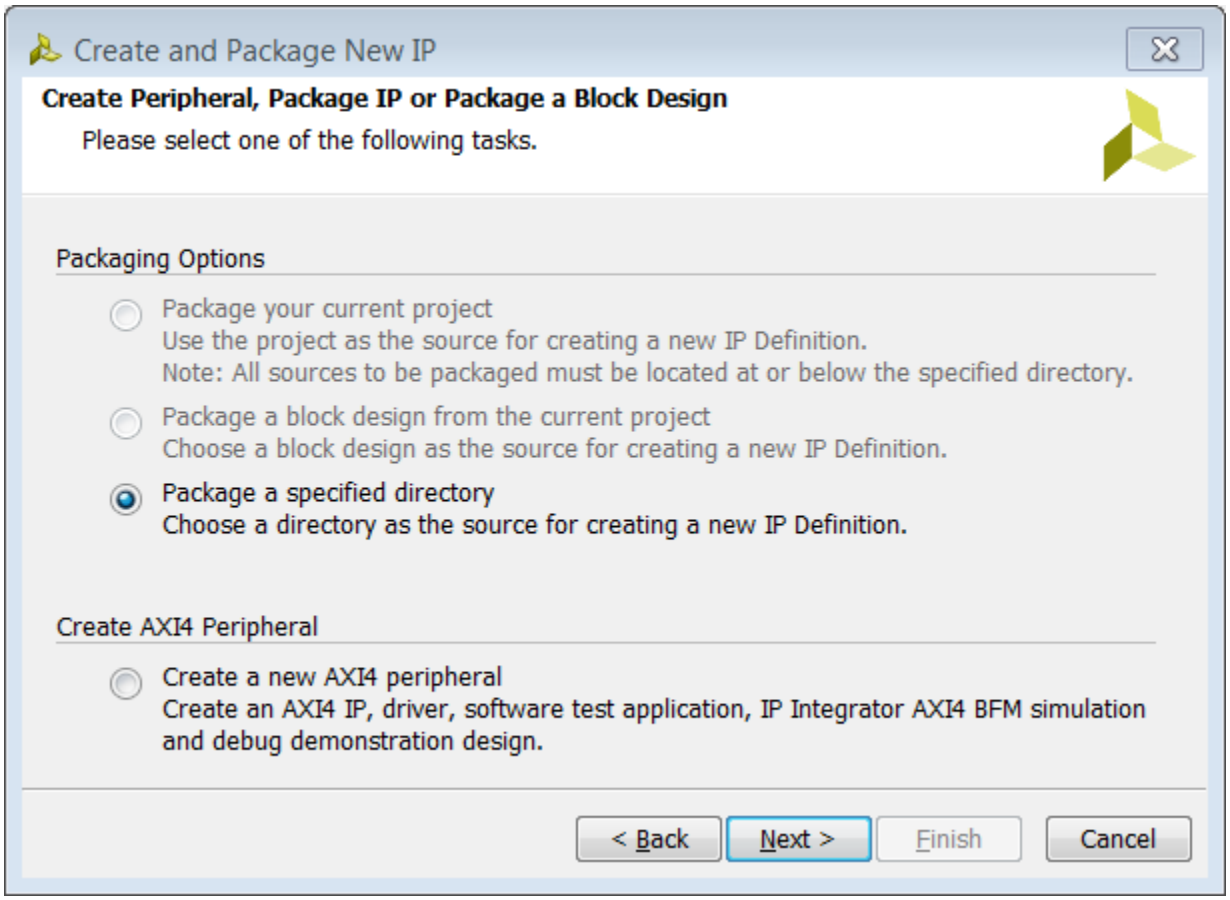


Figure 30: Create Peripheral, Package IP or Package a Block Design

4. Set **Directory** to `<Extract_Dir>/lab2/custom_ip_repo/wave_gen_v1_0`, as shown in the following figure.

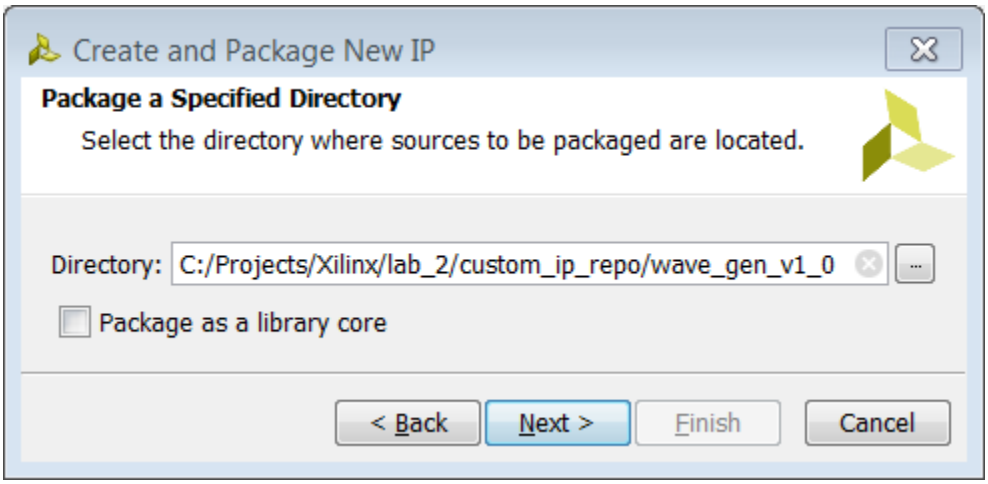


Figure 31: Package a Specified Directory

5. Click **Next**.
6. On the Edit in IP Packager Project Name page, leave the default locations, and click **Next**.

   When packaging a specified directory, the custom IP is packaged through an edit IP project. The default options create an edit IP project in the project temporary location. The edit IP project can be saved for future editing, but a new edit IP project can always be created later.
7. Click **Finish**.

   An edit IP project opens in a new Vivado window with the Package IP window opened. The Package IP window displays the basic IP package in a staging area for editing and repackaging.
8. Leave `project_lab2` open during this process.

Step 4: Examine and Update the Packaged IP

The edit IP project is created as a standard RTL project with the directory sources included. The Package IP window shows the current IP identification information.

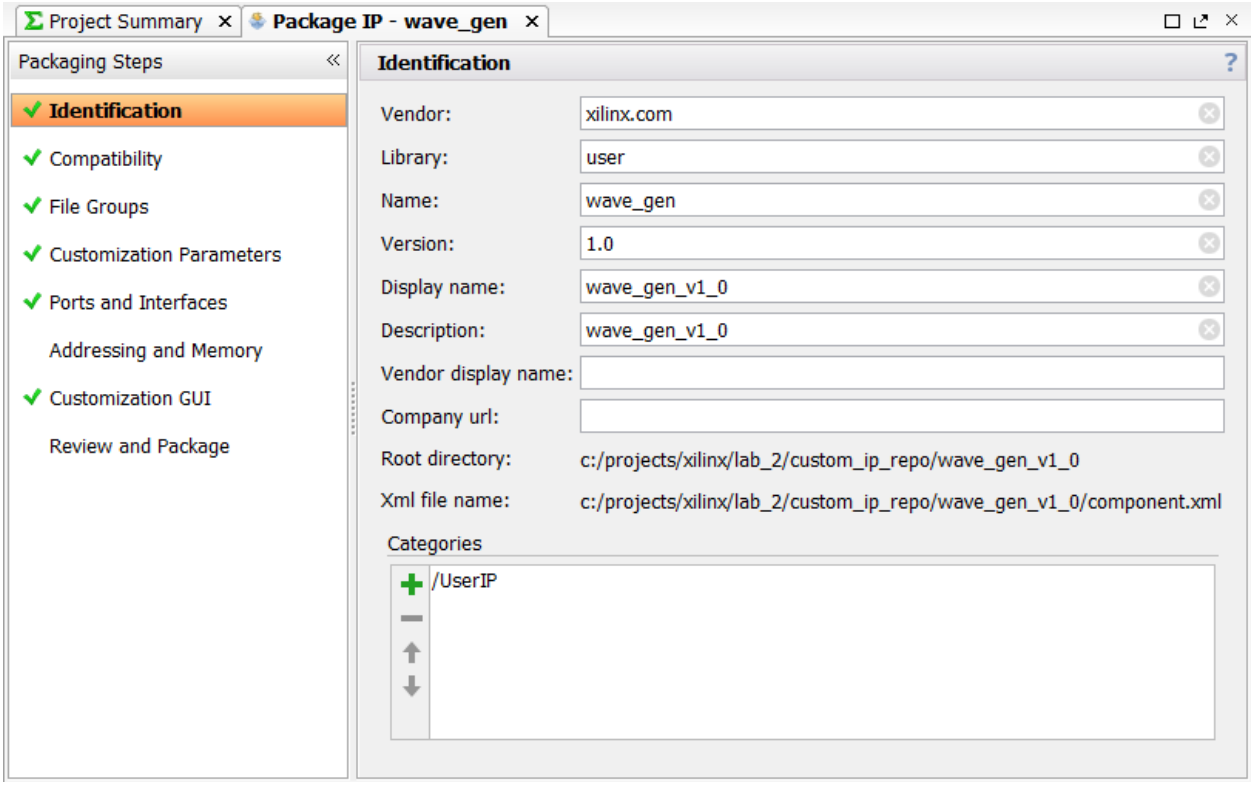


Figure 32: Package IP

Update the IP Identification

1. In the Identification page, set the following options:
   - **Vendor**: my_company
   - **Name**: wave_gen_tutorial
   - **Display name**: Wave Generator Tutorial
   - **Description**: UG1119 Tutorial Lab #2 - Wave Generator tutorial design
   - **Vendor display name**: My Company
   - **Company url**: http://www.my_company_name.com
2. In the Categories section, click the Add button to add a new category.
3. In the **IP Categories** dialog box, click the Add button to add a custom category.

4. In the Add IP Category dialog box, shown in the following figure, set the option to **My Company** and click **OK**.

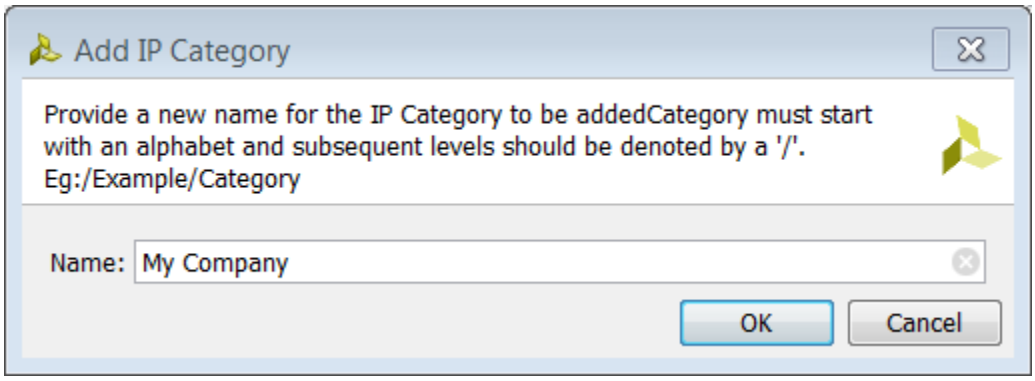


Figure 33: Add IP Category

5. Click **OK** to close the Add IP Categories dialog box.

Examine the IP File Groups

The File Groups page provides a listing of the files to be packaged as part of the custom IP.

1. Examine the files packaged as part of the custom IP to understand how the IP directory correlates to the File Groups.

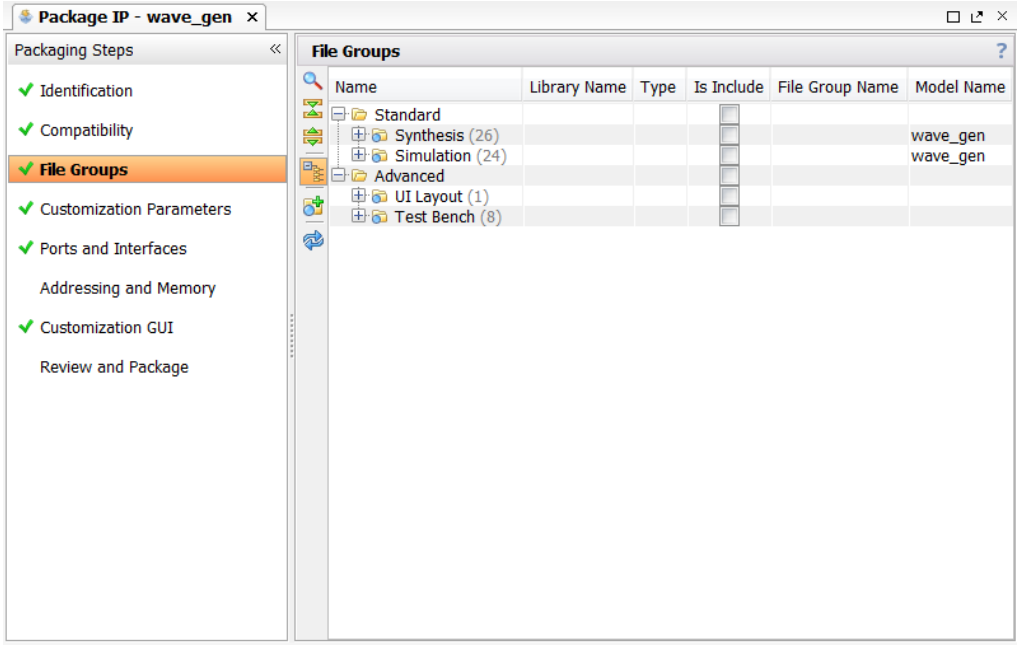
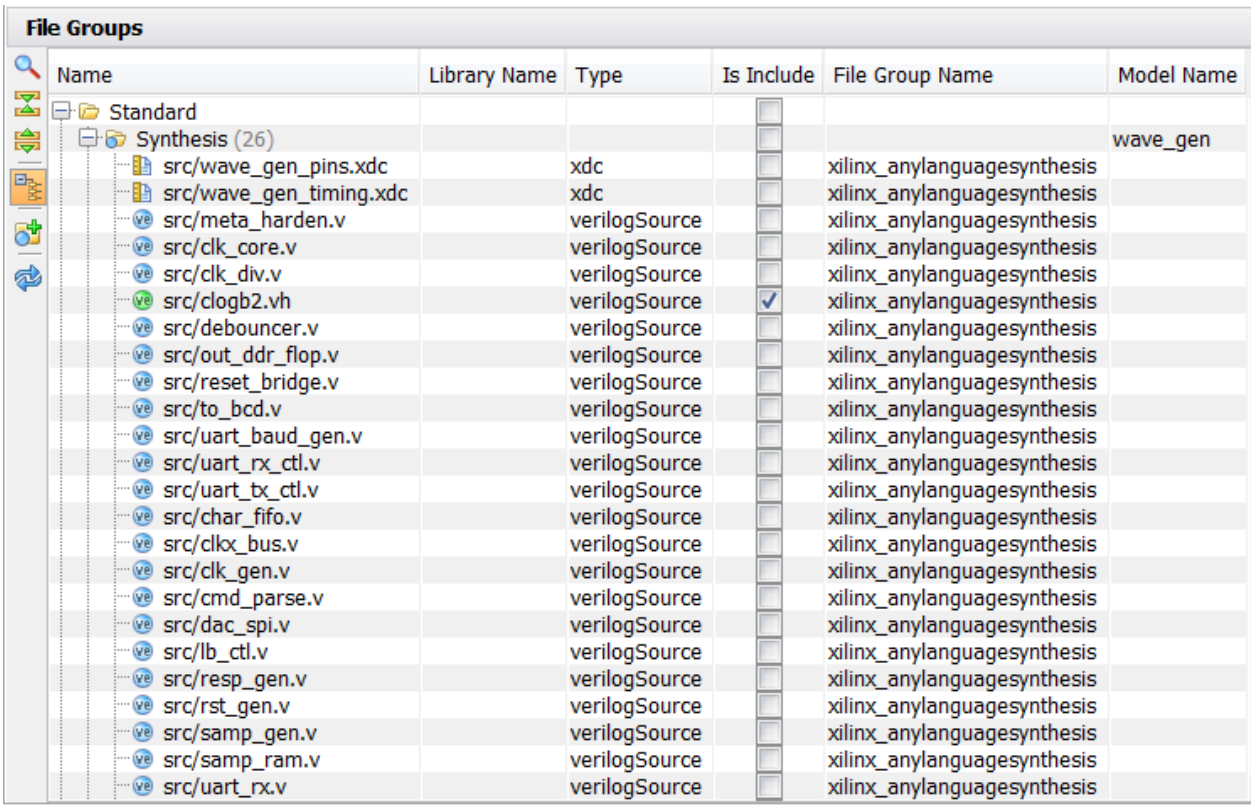


Figure 34: Package IP – File Groups

2. In the Packaging Steps toolbar, select the **File Groups** page.
3. Expand the file group folders as shown in the following figure.



File Groups

Name	Library Name	Type	Is Include	File Group Name	Model Name
Standard					
Synthesis (26)					wave_gen
src/wave_gen_pins.xdc		xdc		xilinx_anylanguagesynthesis	
src/wave_gen_timing.xdc		xdc		xilinx_anylanguagesynthesis	
src/meta_harden.v		verilogSource		xilinx_anylanguagesynthesis	
src/clk_core.v		verilogSource		xilinx_anylanguagesynthesis	
src/clk_div.v		verilogSource		xilinx_anylanguagesynthesis	
src/clogb2.vh		verilogSource	✓	xilinx_anylanguagesynthesis	
src/debouncer.v		verilogSource		xilinx_anylanguagesynthesis	
src/out_ddr_flop.v		verilogSource		xilinx_anylanguagesynthesis	
src/reset_bridge.v		verilogSource		xilinx_anylanguagesynthesis	
src/to_bcd.v		verilogSource		xilinx_anylanguagesynthesis	
src/uart_baud_gen.v		verilogSource		xilinx_anylanguagesynthesis	
src/uart_rx_ctl.v		verilogSource		xilinx_anylanguagesynthesis	
src/uart_tx_ctl.v		verilogSource		xilinx_anylanguagesynthesis	
src/char_fifo.v		verilogSource		xilinx_anylanguagesynthesis	
src/clkx_bus.v		verilogSource		xilinx_anylanguagesynthesis	
src/clk_gen.v		verilogSource		xilinx_anylanguagesynthesis	
src/cmd_parse.v		verilogSource		xilinx_anylanguagesynthesis	
src/dac_spi.v		verilogSource		xilinx_anylanguagesynthesis	
src/lb_ctl.v		verilogSource		xilinx_anylanguagesynthesis	
src/resp_gen.v		verilogSource		xilinx_anylanguagesynthesis	
src/rst_gen.v		verilogSource		xilinx_anylanguagesynthesis	
src/samp_gen.v		verilogSource		xilinx_anylanguagesynthesis	
src/samp_ram.v		verilogSource		xilinx_anylanguagesynthesis	
src/uart_rx.v		verilogSource		xilinx_anylanguagesynthesis	

Figure 35: Package IP – File Groups Expanded

The File Groups page is the listing of the files for the custom IP. The file groups for the custom IP match with directory structure of the IP directory.

The synthesis and simulation file groups contain the HDL files associated with the `/src` directory. The synthesis file group contains two additional files from the `/src` directory, the XDC files.

The Product Guide file group is populated with the PDF from the `/doc` directory and the Testbench file group is populated with the `/tb` directory.

4. Notice that the testbenches are located within its own file group and not in the Simulation file group.

Repackage the IP

The custom IP was packaged at the end of the Create and Package IP wizard. Because changes occurred in the Package IP window, the custom IP must be repackaged for the changes to take effect.

1. In the Packaging Steps toolbar, select the **Review and Package** page.

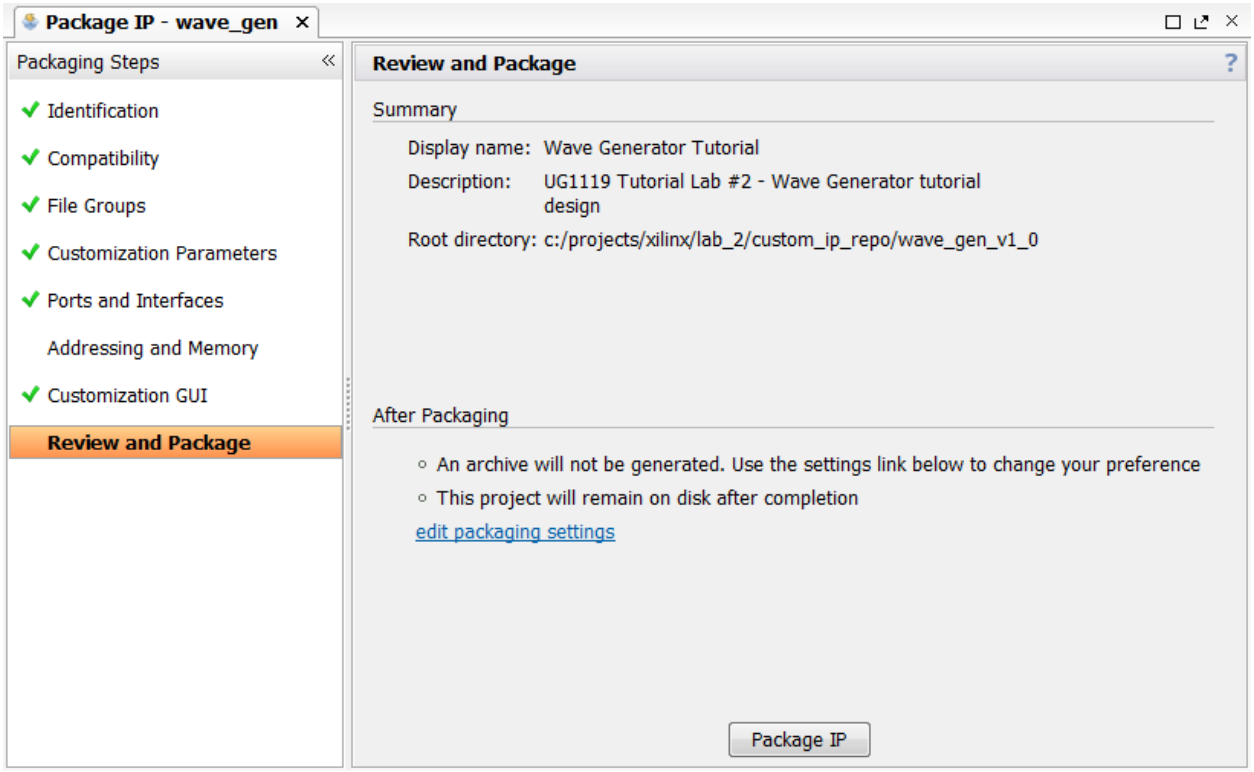


Figure 36: Review and Package

2. Click the **Package IP** button to repackage the IP.
3. After the packaging process completes, close the Vivado edit IP project.

Step 5: Validate the Custom IP

With the new custom IP packaged, the next step is to verify the repository in the IP Catalog and validate the generation of the custom IP. You can use the project_lab2 created in the earlier steps to validate the IP.

Check the IP Repository Project Settings

The project that packaged the specified directory has the IP repository path in the project repository manager. You can validate the IP repository in the project settings at this time.

1. In **Flow Navigator > Project Manager**, select **Project Settings**.
2. In the Project Settings dialog box, select **IP** in the sidebar.
3. In the Repository Manager tab, check the existence of the IP repository `<Extract_Dir>/lab_2/custom_ip_repo/wave_gen_v1_0`.

   The Wave Generator Tutorial IP shows in the IP in Selected Repository list.

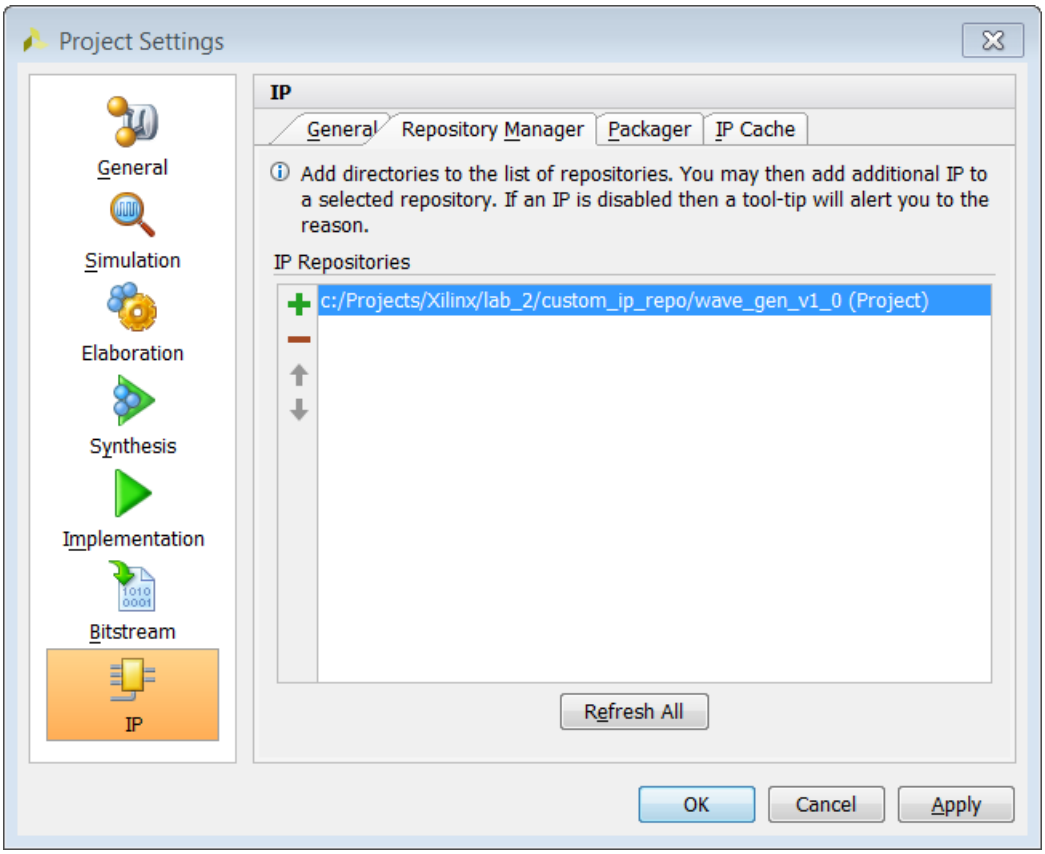


Figure 37: IP Project Settings

***Note**: Vivado selects the IP directory location as the repository. You can select the parent repository directory and Vivado traverses the subdirectories for packaged IP.*

4. Click **OK** to close the Project Settings dialog box.

Customize the IP

1. In Flow Navigator > Project Manager, select IP Catalog.
2. In the search field at the top of the IP Catalog, type **Wave Generator**.

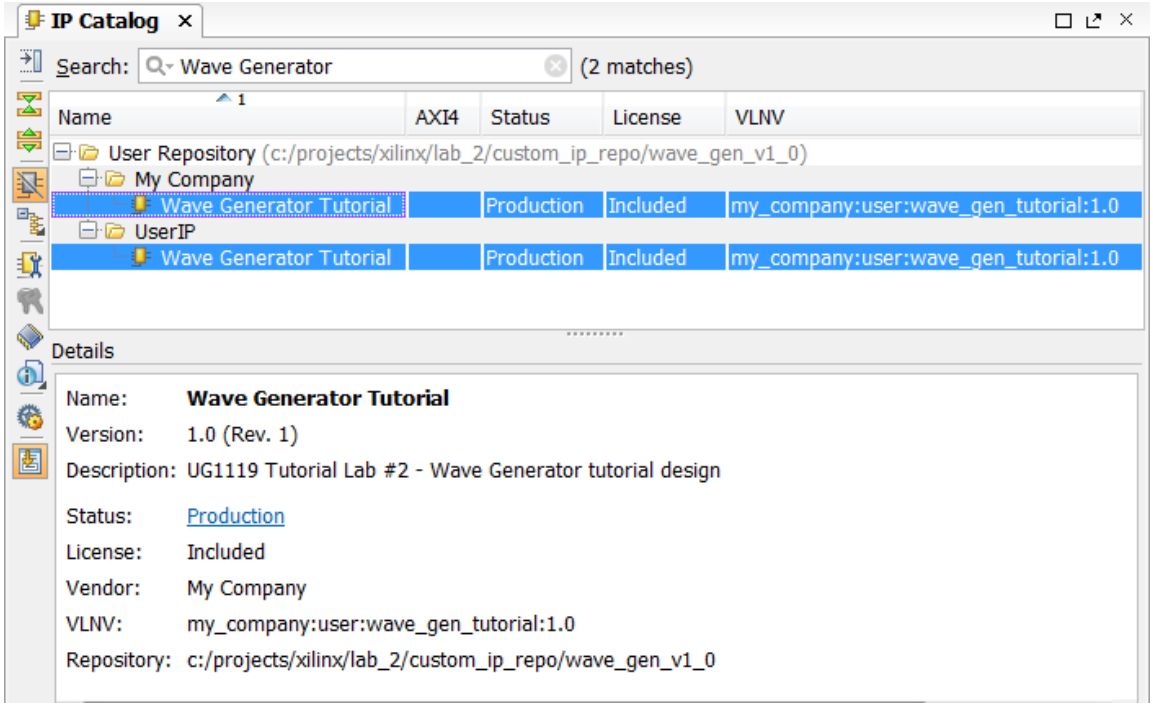


Figure 38: IP Catalog

The Wave Generator Tutorial IP is reported under the **UserIP** category as well as the custom category **My Company** that was created during packaging.

> ***Note***: *This IP Catalog view shows when the Taxonomy and the Repository options are selected for Grouping the IP. See the Vivado Design Suite: Creating and Packaging Custom IP (UG1118) for more information about IP Groups.*

3. Right-click the Wave Generator Tutorial IP and select **Customize IP**.

   The following figure shows the Wave Generator Tutorial IP view.

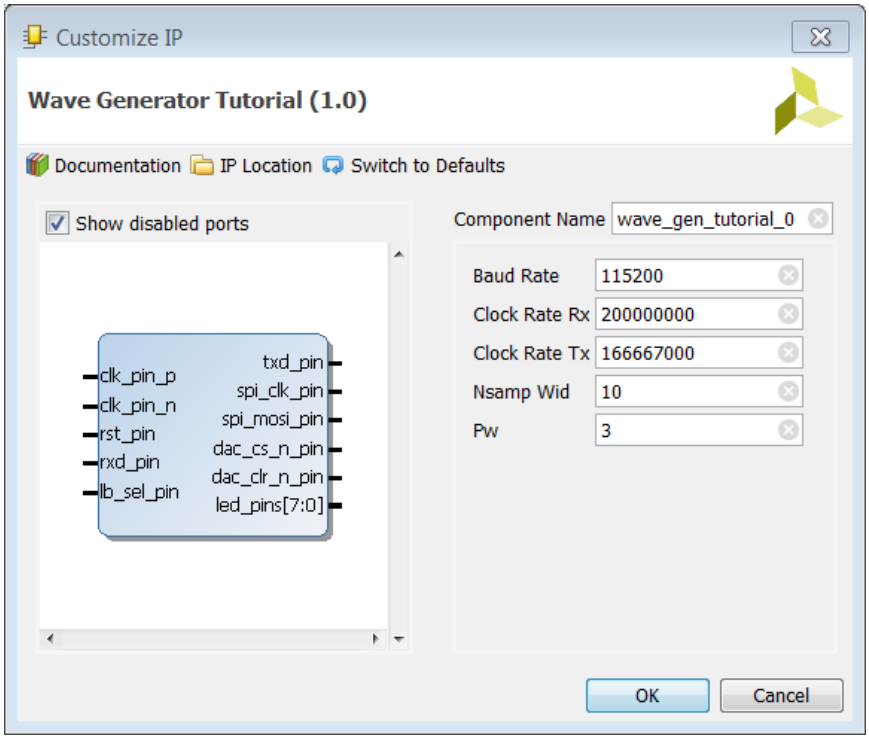


Figure 39: Customize IP – Wave Generator Tutorial

4. Click **OK** to accept the default configuration options.
5. In the Generate Output Products dialog box, select **Generate**.

   This generates the various files required for this IP in the current Manage IP project, and launches an Out-Of-Context synthesis run for the IP to create a DCP. The Generate Output Products dialog re-opens to report the output products were generated successfully.

Conclusion

You have successfully created the Wave Generator Tutorial IP by packaging a specified directory. Close the project and exit the Vivado tool. You cannot continue further with this design because it will not complete implementation. In this lab, you did the following:

- Used the Create and Package IP wizard to package a specified directory for the Wave Generator Tutorial design.
- Validated the generation of the Wave Generator Tutorial IP output products.

Introduction

You might need to use a legacy core in Vivado that was originally created in the Xilinx Platform Studio (XPS) tool.

In this lab, you learn how to convert an XPS processor core, or Pcore, to a Vivado Design Suite native IP for use in IP integrator. To migrate a legacy core, you need all the libraries on which the main core is dependent. This lab uses a simple GPIO Pcore from an XPS project. This core has several dependencies on the following libraries:

- proc_common_v3_00_a
- axi_lite_ipif_v1_01_a
- interrupt_control_v2_01_a
- axi_gpio_v1_01_b

To migrate this Pcore, you must determine all the files that are needed for the GPIO IP, package them as library cores (or sub-cores), add the sub-cores to the IP Catalog, and then package the GPIO IP.

Step 1: Create a New Vivado Project

Launch Vivado

On Linux:

- Change to the directory where the lab materials are stored: cd `<Extract_Dir>/lab_3`.
- Launch the Vivado IDE: **vivado.**

On Windows:

- Launch the Vivado Design Suite IDE, by using either of the following methods:
  Start > All Programs > Xilinx Design Tools > Vivado 2016.x > Vivado 2016.x
- Click the Vivado 2016.x desktop icon to start the Vivado IDE.

The Vivado IDE Getting Started page displays with links to open or create projects, and to view documentation. For either Windows or Linux, continue the lab from this point.

Create a New Project

1. From the Vivado IDE Getting Started page, select **Create New Project** to create an empty Vivado project.

   A new or existing project is required to creating and packaging a custom IP. The project information populates certain fields in the Package IP window.

2. In the New Project Wizard dialog box, click **Next**.
3. As shown in the following figure, set the following options:
   - **Project name**: `project_lab3`
   - **Project location**: `<Extract_Dir>/lab_3`
   - Check the Create Project subdirectory box.

   The following figure shows these settings.

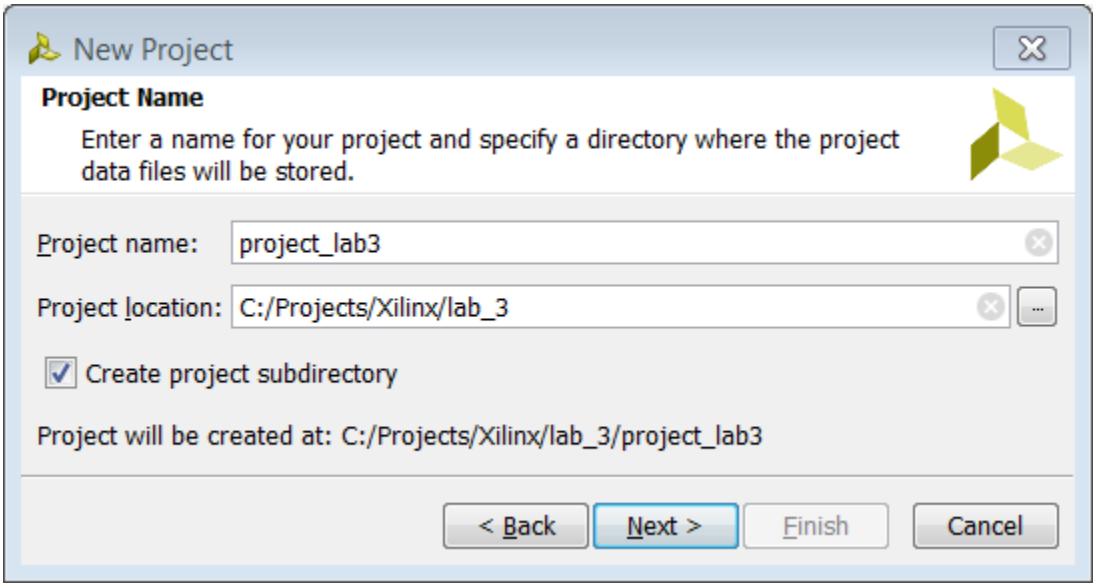


Figure 40: New Project – Project Name

4. Click **Next**.
5. Select a Project Type of **RTL Project** and **Do not specify sources at this time**.
6. Click **Next**.
7. On the Default Part page, select the `xc7k70tfbg484-2` part, and click **Next**.

The following figure shows the New Project: Default Part dialog box.

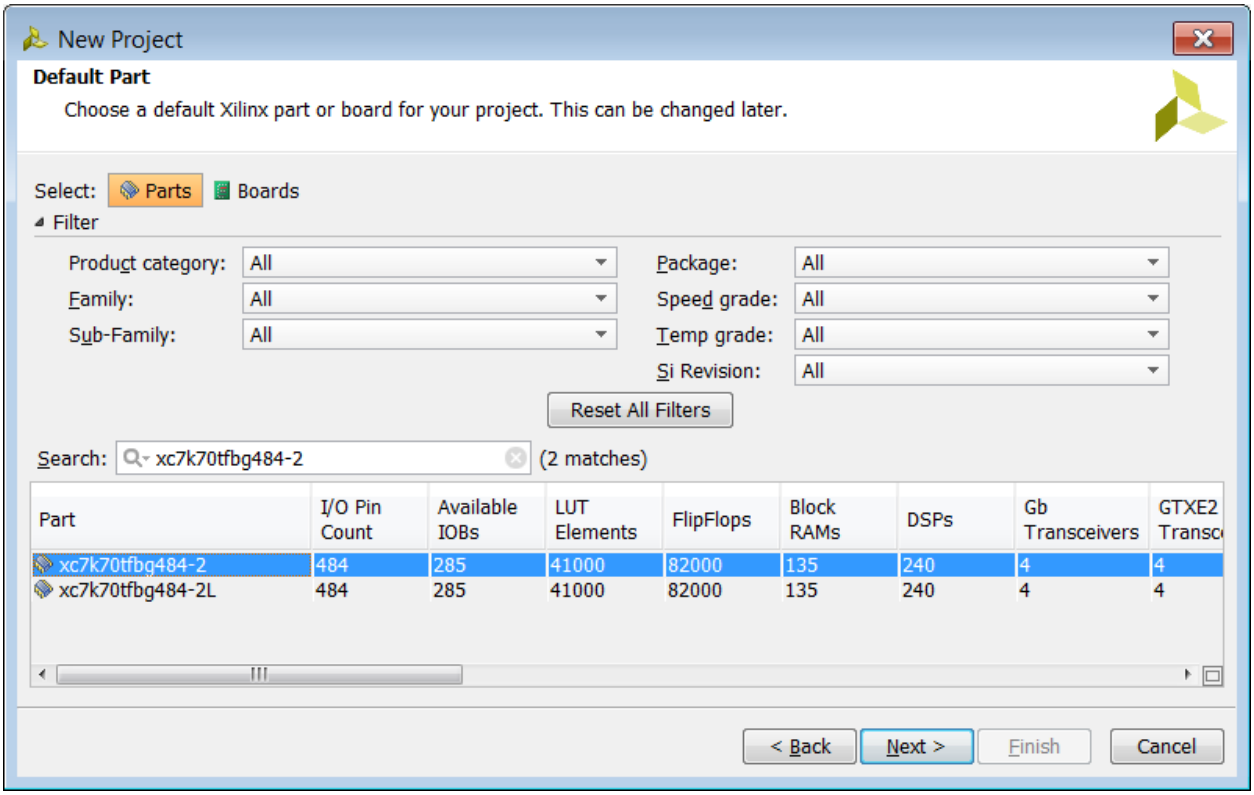


Figure 41: New Project – Default Part

You have selected a Kintex®-7 device. This device family is used for the initial compatibility of the custom IP.

8. In the New Project Summary page, which opens, click **Finish** to create the project.

   The Vivado IDE opens `project_lab3`, the default layout.

Step 2: Package a Library Core

As discussed in the Introduction of this lab, the GPIO Pcore requires several library references (sub-cores) to function.

Because these library cores do not exist in the latest Vivado releases, start by packaging the libraries before you package the GPIO Pcore.

Use the Create and Package Wizard

1. From the Tools menu, select Create and Package IP to open the Create and Package IP wizard.
2. In the Create and Package New IP dialog box welcome screen, click **Next**.
3. In the Create Peripheral, Package IP, or Package a Block Design screen, select **Package a specified directory.**

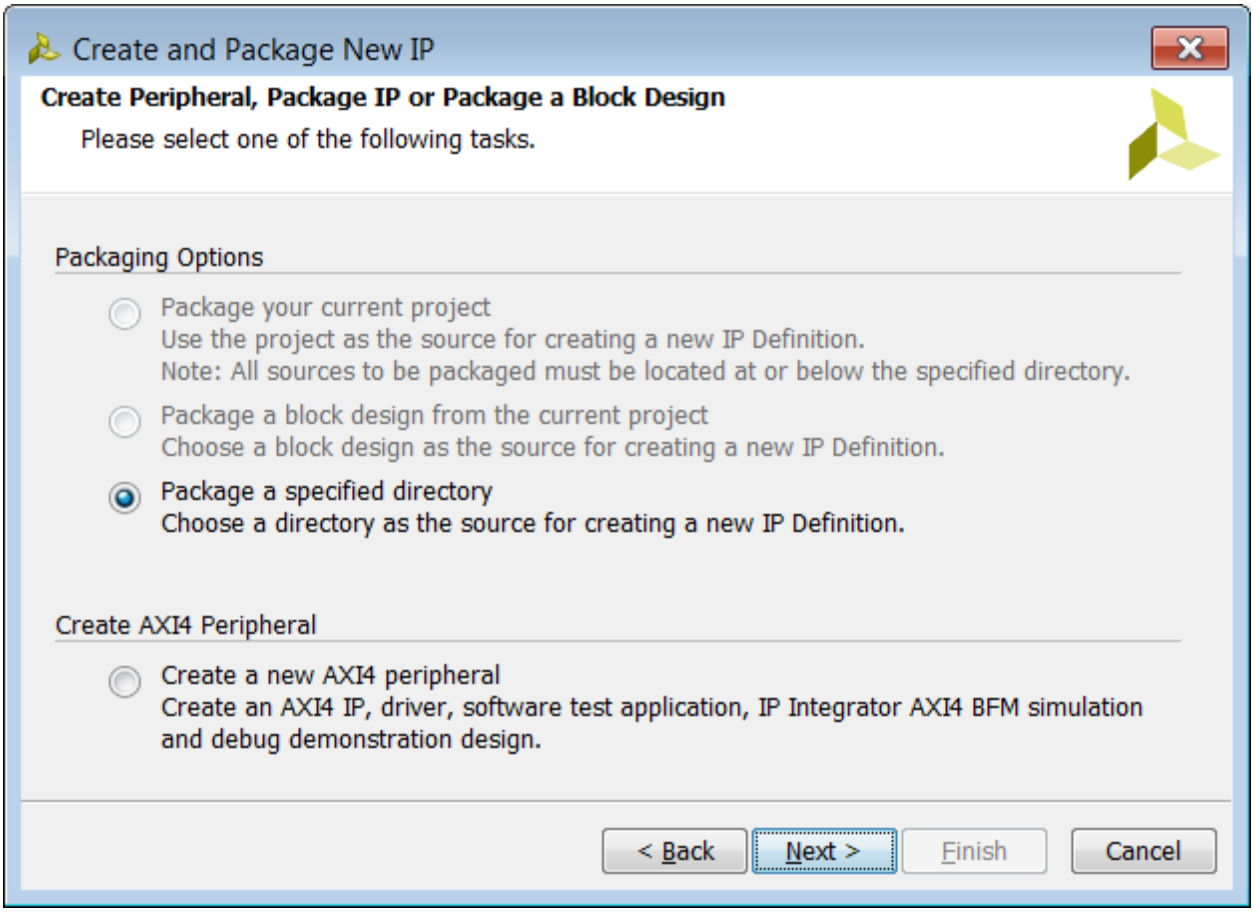


Figure 42: Create Peripheral, Package IP or Package a Block Design

4. In the Package a Specified Directory dialog box, shown in the following figure, set the options as follows:
   - **Directory**: `<Extract_Dir>/lab3/pcores/proc_common_v3_00_a`
   - Check the **Package as a library core** option.

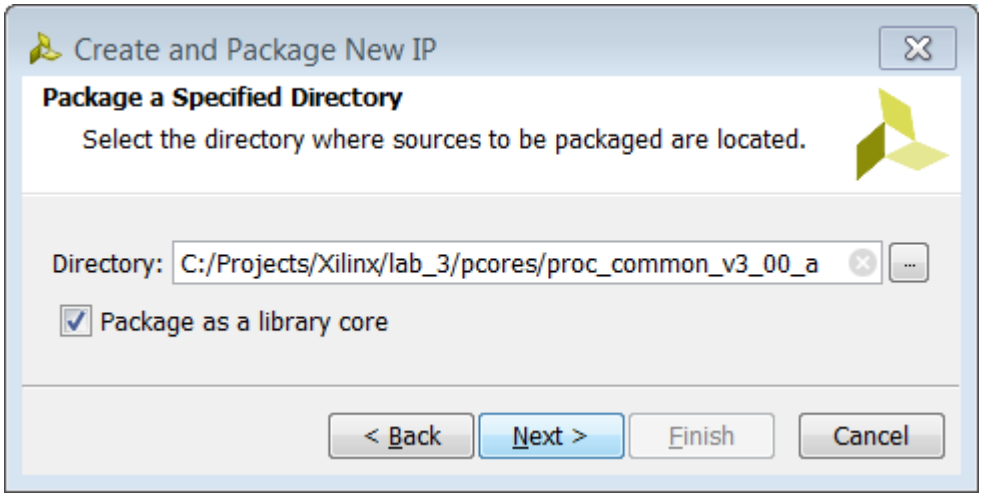


Figure 43: Package a Specified Directory

5. Click **Next**.
6. In the Edit in IP Packager Project Name page, leave the default locations, and click **Next**.
7. Click **Finish**.

   An edit IP project opens in a new Vivado window with the Package IP window opened. The Package IP window displays the basic IP package in a staging area for editing and repackaging.

Update the IP Information

Because you selected the library core option, the Package IP window has as subset of available options for the custom IP, as shown in the following figure.

1. Update the library core with the necessary information, as follows:
2. Select the Identification page, and fill in the following fields:
    - **Display name**: proc_common_v3_00_a
    - **Description**: Proc Common v3.00.a Library Core

    Note: *Notice that the Vendor and Library fields are auto-populated.*

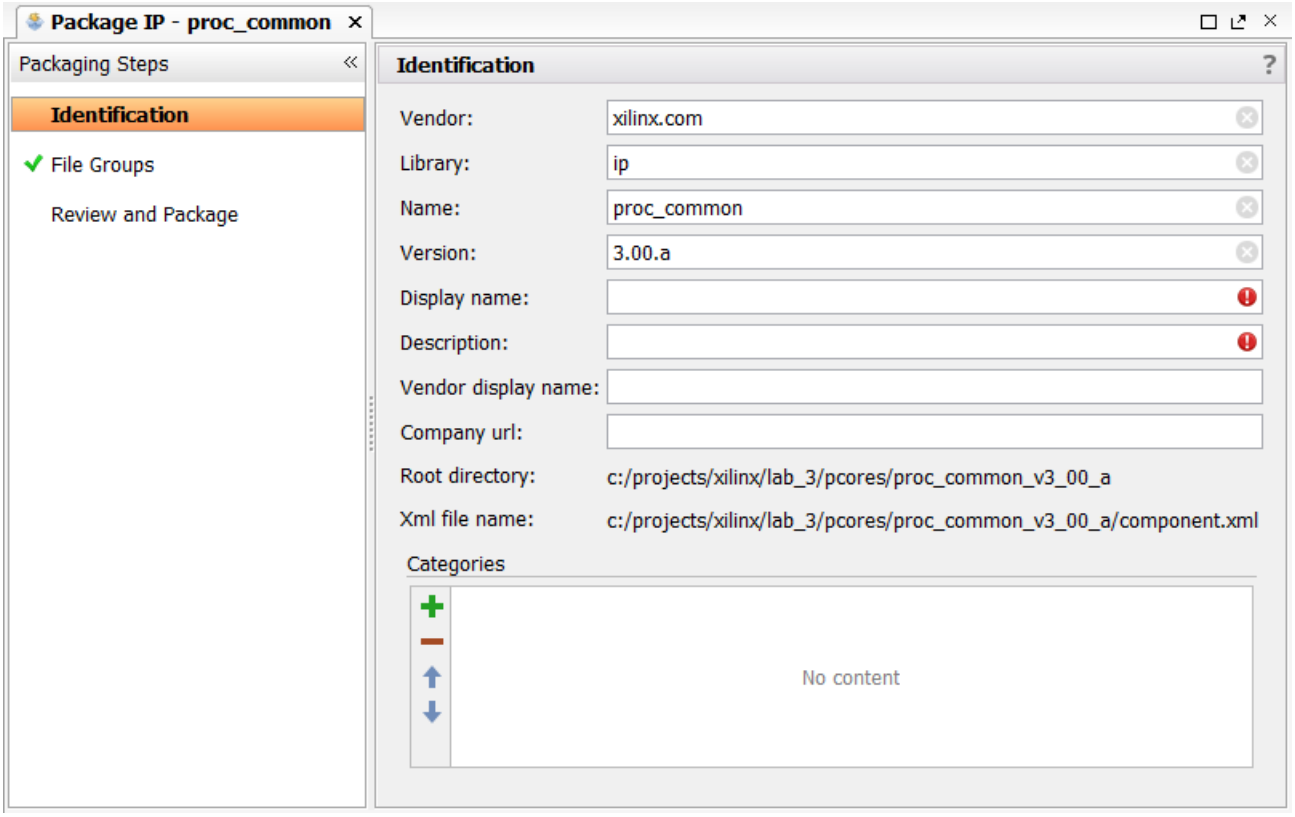


Figure 44: Package IP

Select Review and Package to view the name, location, and Root directory information about the library core, as shown in the following figure.

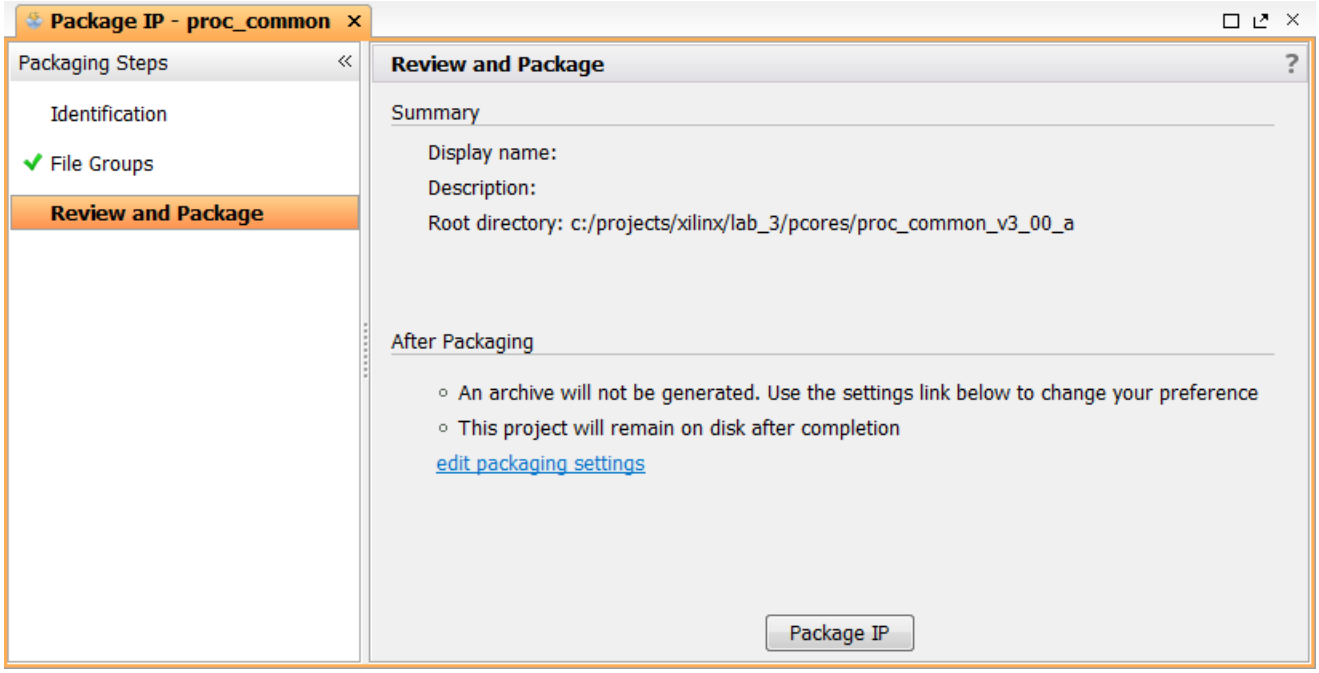


Figure 45: Review and Package

3. Click Package IP.

   This completes the packaging for the `proc_common_v3_00_a` library core. If prompted, you can close the edit_ip_project.

Package Additional Library Cores

Repeat the steps to package the `axi_lite_ipif_v1_01` library and the `interrupt_control_v2_01_a` libraries. When packaging these two library cores, ensure that the display name and descripts for each of the library cores are as follows:

Library Core	Display Name	Description
axi_lite_ipif	axi_lite_ipif_v1_01_a	AXI Lite IPIF v1.01.a Library Core
interrupt_control	interrupt_control_v2_01_a	Interrupt Control V2.01.a Library Core

IMPORTANT: *When packaging the additional library cores, the `axi_lite_ipif` and the `interrupt_control_v2_01_a` libraries will display a green checkmark for the File Group page.*

Step 3: Package the GPIO IP

Now that all the library cores are properly packaged, you can package the GPIO IP from the originally created `lab_3` project.

1. From the Tools menu, select Create and Package IP to open the Create and Package IP wizard.
2. Click **Next** at the Welcome screen for the Create and Package New IP dialog box.
3. In the Create Peripheral, Package IP, or Package a Block Design dialog box, select **Package a specified directory**.
4. In the Package a Specified Directory dialog box, set the following option:
   Directory: `<Extract_Dir>/lab3/pcores/axi_gpio_v1_01_b`.

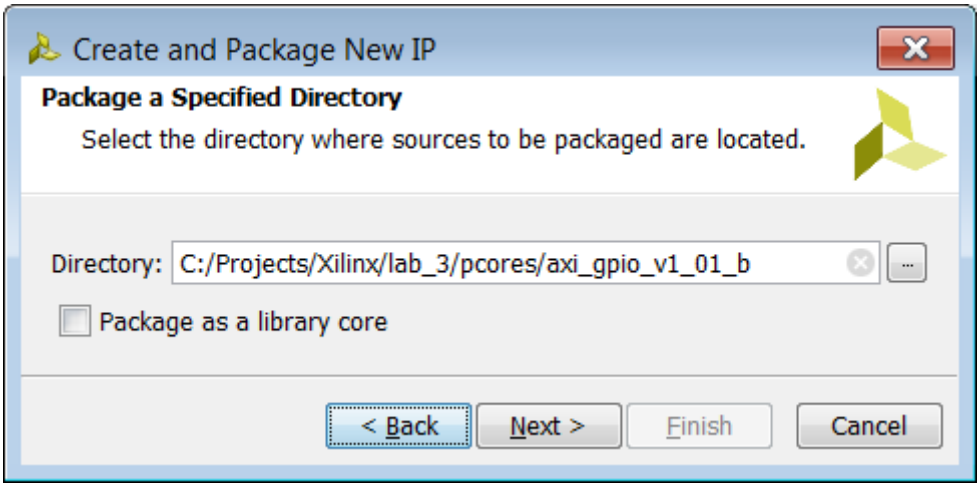


Figure 46: Package a Specified Directory

5. Click **Next**.
6. On the Edit in IP Packager Project Name page, leave the default locations, click **Next**, and then click **Finish**.

   The Create and Package IP wizard collects the available information from the specified location. When specifying a directory for packaging, there are inference rules that assist in packaging the IP correctly.

   For XPS Pcores, if a peripheral analyze order file (PAO file) exists in the data directory, the wizard reads this file and uses the associated library information.

An edit IP project opens in a new Vivado packaging window with the Package IP window opened.

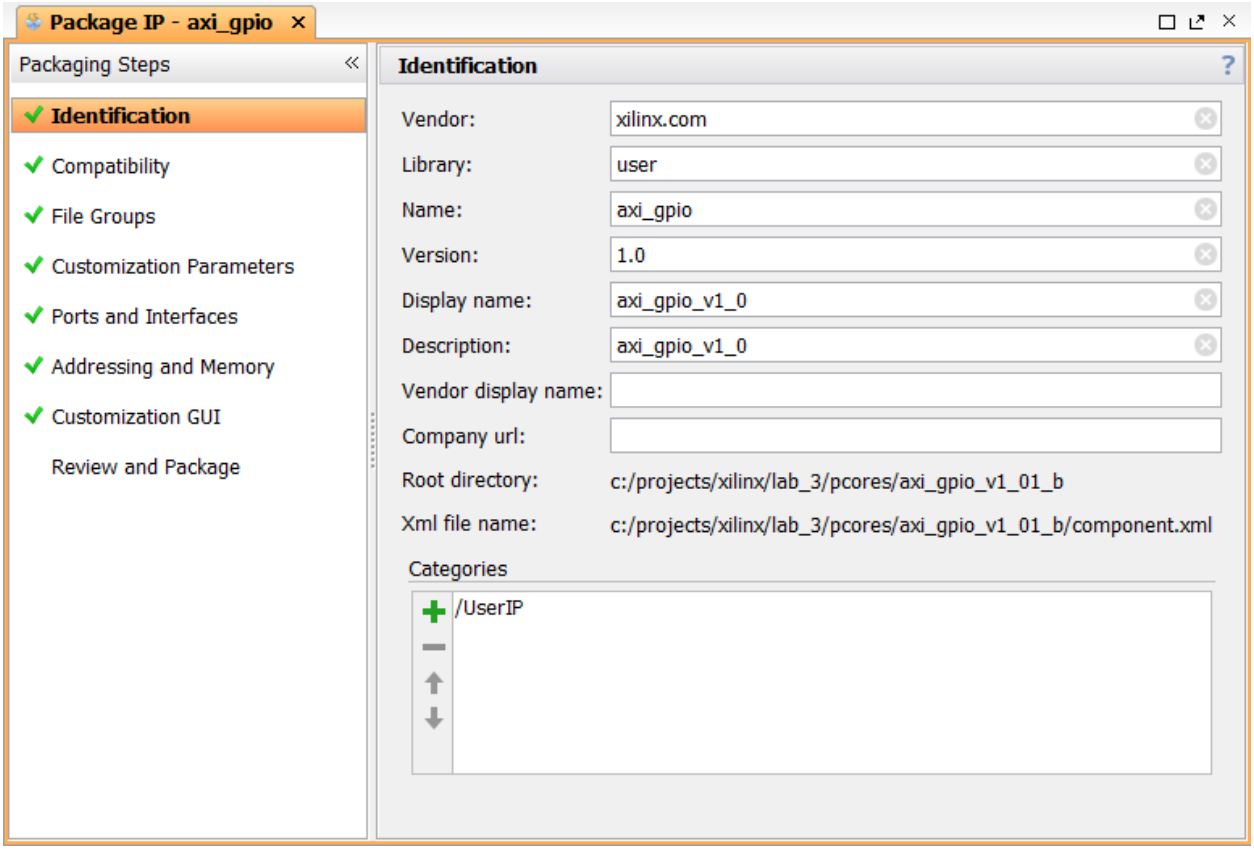


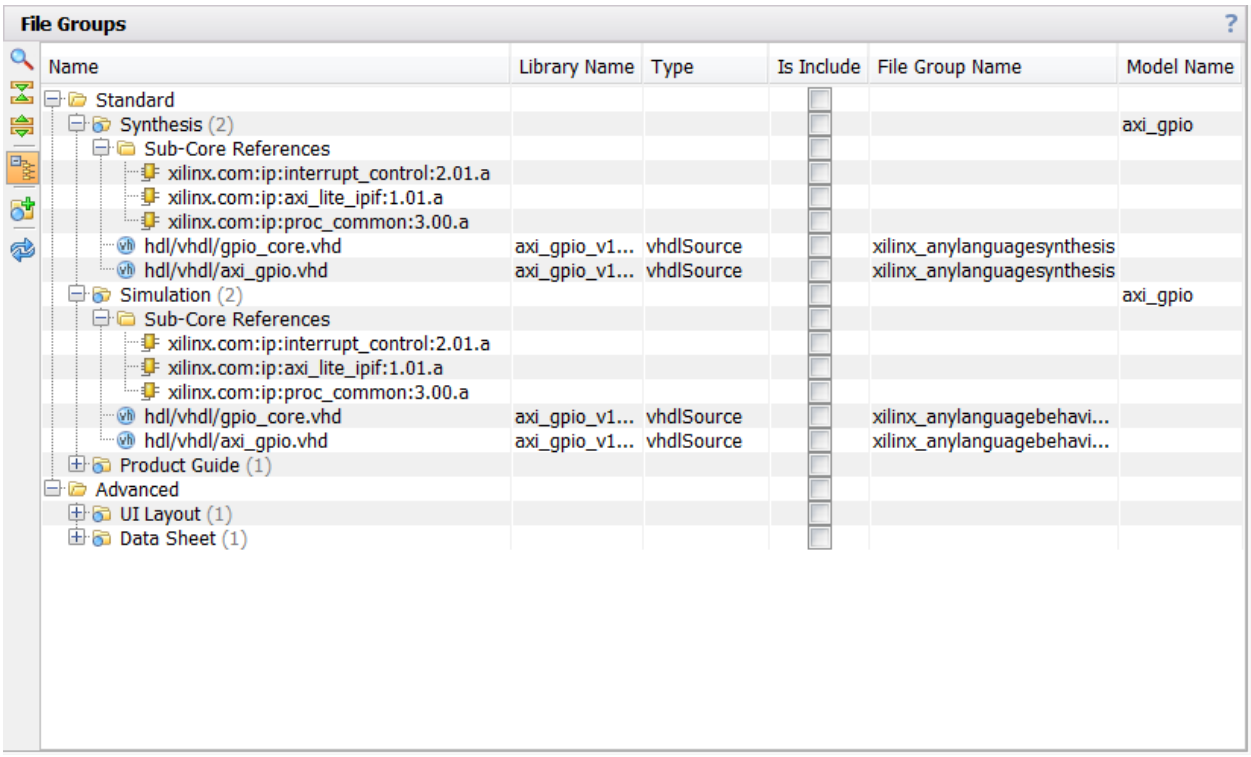
Figure 47: Package IP

Update the IP Identification

1. In the Package IP window, update the following information:
    - **Vendor**: my_company
    - **Name**: axi_gpio
    - **Display name**: My AXI GPIO EDK Pcore Tutorial
    - **Description**: UG1119 Tutorial Lab #3 - AXI GPIO EDK Pcore
    - **Vendor display name**: My Company
    - **Company url**: http://www.my_company_name.com

2. Click the **File Groups** page to validate that the proper Sub-Core References (Library Cores) were added to the Package IP window.

In this case the Interrupt Controller, the AXI Lite IPIG and the Proc Common display in the `/Sub-Core References` directories for Synthesis and Simulation.

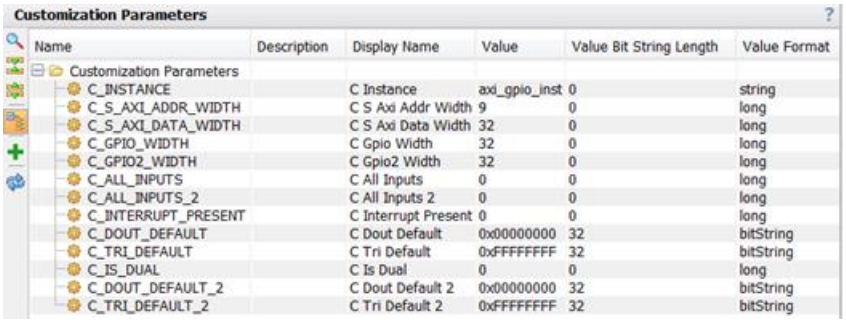


File Groups

Name	Library Name	Type	Is Include	File Group Name	Model Name
Standard					
Synthesis (2)					axi_gpio
Sub-Core References					
xilinx.com:ip:interrupt_control:2.01.a					
xilinx.com:ip:axi_lite_ipif:1.01.a					
xilinx.com:ip:proc_common:3.00.a					
hdl/vhdl/gpio_core.vhd	axi_gpio_v1...	vhdlSource		xilinx_anylanguagesynthesis	
hdl/vhdl/axi_gpio.vhd	axi_gpio_v1...	vhdlSource		xilinx_anylanguagesynthesis	
Simulation (2)					axi_gpio
Sub-Core References					
xilinx.com:ip:interrupt_control:2.01.a					
xilinx.com:ip:axi_lite_ipif:1.01.a					
xilinx.com:ip:proc_common:3.00.a					
hdl/vhdl/gpio_core.vhd	axi_gpio_v1...	vhdlSource		xilinx_anylanguagebehavi...	
hdl/vhdl/axi_gpio.vhd	axi_gpio_v1...	vhdlSource		xilinx_anylanguagebehavi...	
Product Guide (1)					
Advanced					
UI Layout (1)					
Data Sheet (1)					

Figure 48: Package IP: File Groups

3. Click the **Customization Parameters** to explore the parameters defined for the custom IP.



Customization Parameters

Name	Description	Display Name	Value	Value Bit String Length	Value Format
Customization Parameters					
C_INSTANCE		C Instance	axi_gpio_inst	0	string
C_S_AXI_ADDR_WIDTH		C S Axi Addr Width	9	0	long
C_S_AXI_DATA_WIDTH		C S Axi Data Width	32	0	long
C_GPIO_WIDTH		C Gpio Width	32	0	long
C_GPIO2_WIDTH		C Gpio2 Width	32	0	long
C_ALL_INPUTS		C All Inputs	0	0	long
C_ALL_INPUTS_2		C All Inputs 2	0	0	long
C_INTERRUPT_PRESENT		C Interrupt Present	0	0	long
C_DOUT_DEFAULT		C Dout Default	0x00000000	32	bitString
C_TRI_DEFAULT		C Tri Default	0xFFFFFFFF	32	bitString
C_IS_DUAL		C Is Dual	0	0	long
C_DOUT_DEFAULT_2		C Dout Default 2	0x00000000	32	bitString
C_TRI_DEFAULT_2		C Tri Default 2	0xFFFFFFFF	32	bitString

Figure 49: Package IP – Customization Parameters

4. Click **Review and Package** to view the Summary of the custom IP, as shown in the following figure.

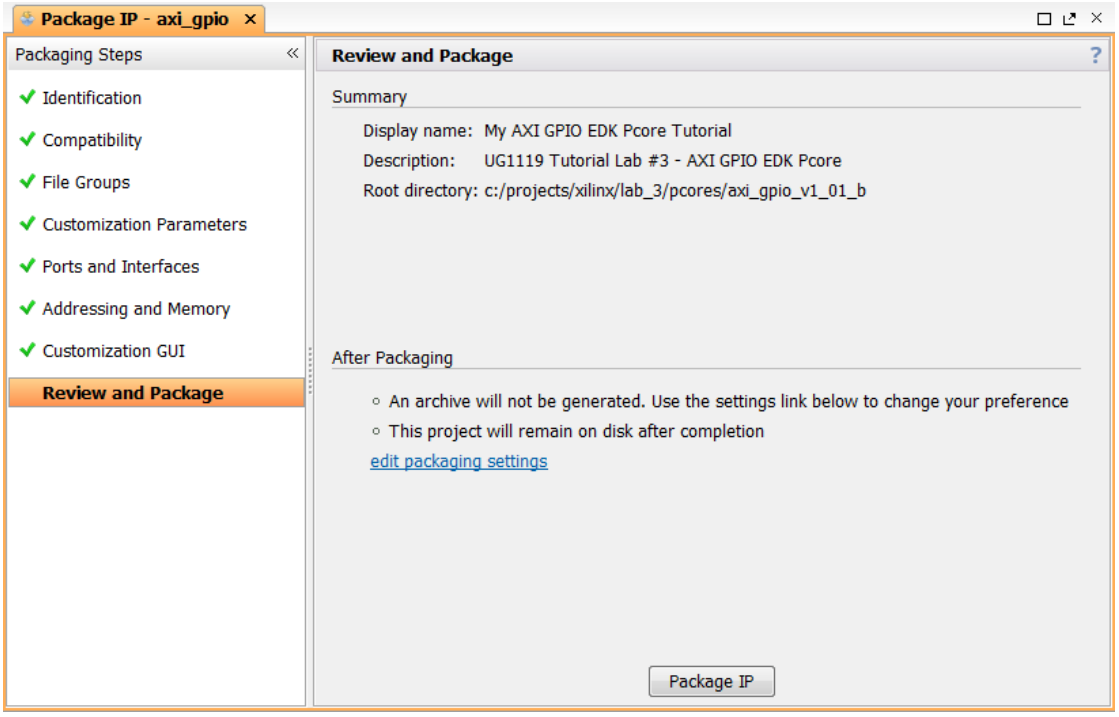


Figure 50: Package IP – Review and Package

5. Click the **Package IP** button to update the IP with the changes you made in the Package IP window.
6. After packaging is complete, close the `edit_ip_project`.

Step 4: Validate the New Custom IP

After completing packaging of the library cores and the AXI GPIO IP, you can use `project_lab3` that you created to validate the generation of the custom IP.

IMPORTANT: Because you packaged the custom IP and library cores in this lab, the Repository Manager already contains the paths to the custom IP. If you use another project for validation, the repository paths for the custom IP and the library cores must be set.

1. In the **Flow Navigator** > **Project Manager**, select **IP Catalog**.
2. In the search field at the top of the IP Catalog, type AXI GPIO.

   The My AXI GPIO EDK Pcore Tutorial IP shows under the `/UserIP` directory.

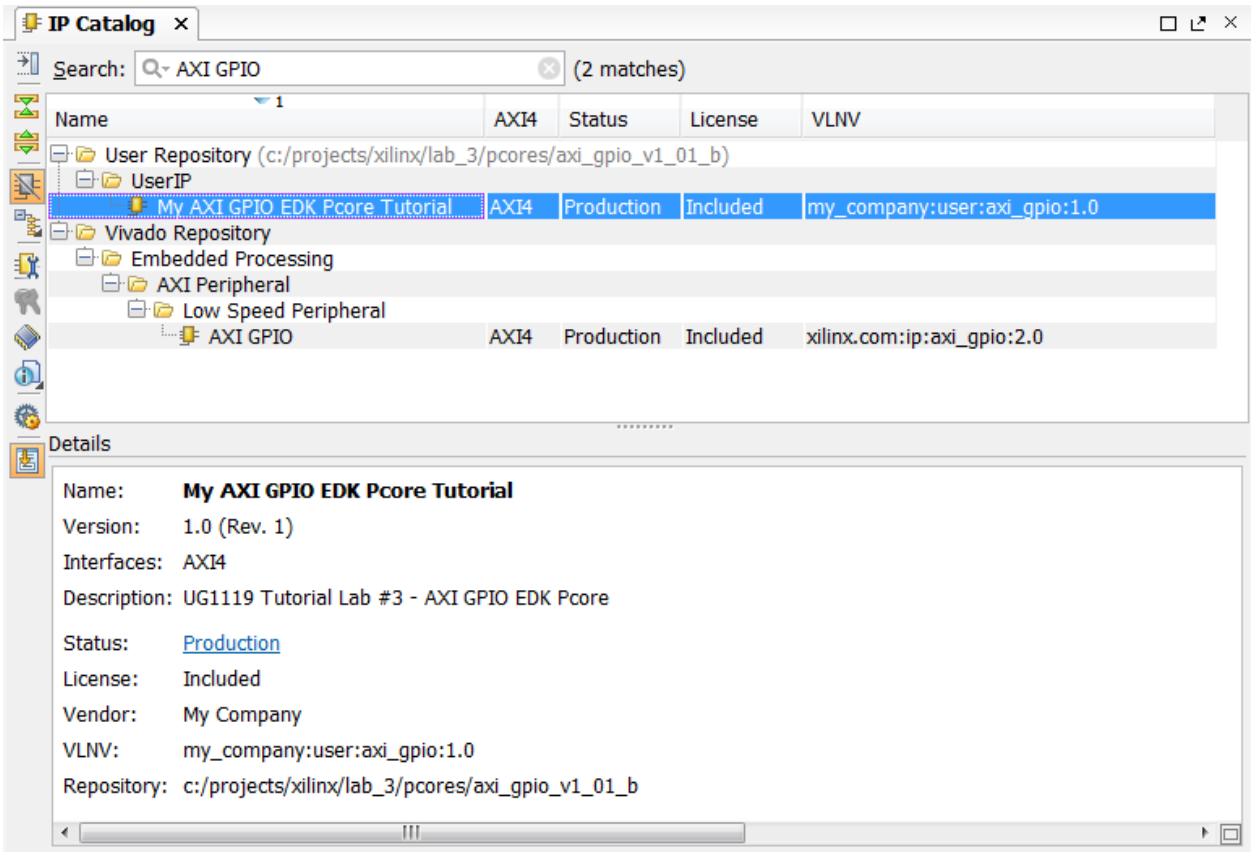


Figure 51: IP Catalog

3. Right-click the My AXI GPIO EDK Pcore Tutorial IP and select Customize IP.
4. Click **OK** to accept the default configuration options.
5. In the Generate Output Products dialog box, select **Generate**.

   The files required for this IP in the current Manage IP project generate, and an out-of-context (OOC) synthesis run for the IP generates and creates a DCP file.

   The Generate Output Products dialog re-opens to report that the output products generated successfully.
6. Close the project and exit the Vivado tool.

Conclusion

This concludes Lab #3.

You have successfully created the AXI GPIO Pcore IP by packaging the /Pcore directory as well the library dependencies. In this lab, you did the following:

- Used the Create and Package IP Wizard to package a specified directory for each of the library cores.
- Used the Create and Package IP Wizard to package a specified directory for the GPIO Pcore.
- Validated the generation of the GPIO Pcore custom IP.

Introduction

In this lab, you define new custom IP from a set of example files that mimic a repository development trunk. In addition, this lab describes the process for creating custom IP that depend on files from other IP within the repository trunk.

You start with an IP repository trunk and create a new Vivado project. In the Vivado project, you package the different custom IP in the repository using the Create and Package IP Wizard. You also identify which need to be library cores, and verify the packaged files. The lab project contains Verilog source files.

Step 1: Examine the Repository Trunk Directory

1. Examine the <Extract_Dir>/lab_4/trunk location.

   The directory contains the files for the respective custom IP that would exist in the repository. In particular, there are two source directories as shown in the following figure:

   - `common_v1_0`: Directory contains source for logic common to the IP within the repository trunk.
   - `myip_v1_0`: Directory contains source for custom IP.

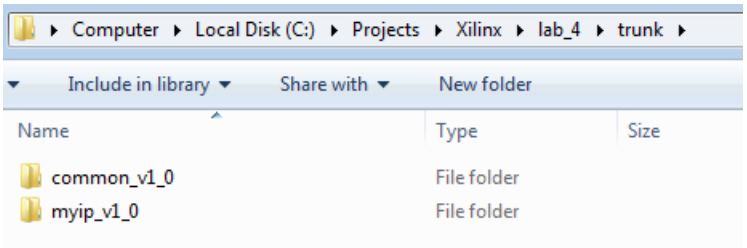


Figure 52: Lab 4 Directory Structure

The `common_v1_0` directory contains a source file that is required by `myip_v1_0`. Because the `component.xml` file for `myip_v1_0` cannot reference a source file from outside the IP root directory, the source file from `common_v1_0` must be referenced differently.

Note: *The directories containing the source files should be organized to ensure proper packaging. For an example on how to properly organize your source files, see Lab 2: Packaging a Specified Directory. Although not described in this lab, if your repository trunk does not have the same structure, you can package each source directory by packaging the associated Vivado project.*

2. Examine the files in each of the directories for more information about the structure of the repository trunk.

Step 2: Create a New Vivado Project

Launch Vivado

On Linux:

- Change to the directory where the lab materials are stored: cd `<Extract_Dir>/lab_4`.
- Launch the Vivado IDE: **vivado.**

On Windows:

- Launch the Vivado Design Suite IDE, by using either of the following methods:
  Start > All Programs > Xilinx Design Tools > Vivado 2016.x > Vivado 2016.x
- Click the Vivado 2016.x desktop icon to start the Vivado IDE.

The Vivado IDE Getting Started page displays with links to open or create projects, and to view documentation. For either Windows or Linux, continue the lab from this point.

Create a New Project

1. From the Vivado IDE Getting Started page, select **Create New Project** to create an empty Vivado project.

   Note: *A new or existing project is required to creating and packaging a custom IP. The project information is used for populating certain fields in the Package IP window.*

2. Click **Next** at the New Project wizard dialog box.
3. In the Project Name page, as shown in the following figure, set the following options for the project location:
   - **Project name:** `project_lab4`
   - **Project location:** `<Extract_Dir>/lab_4`

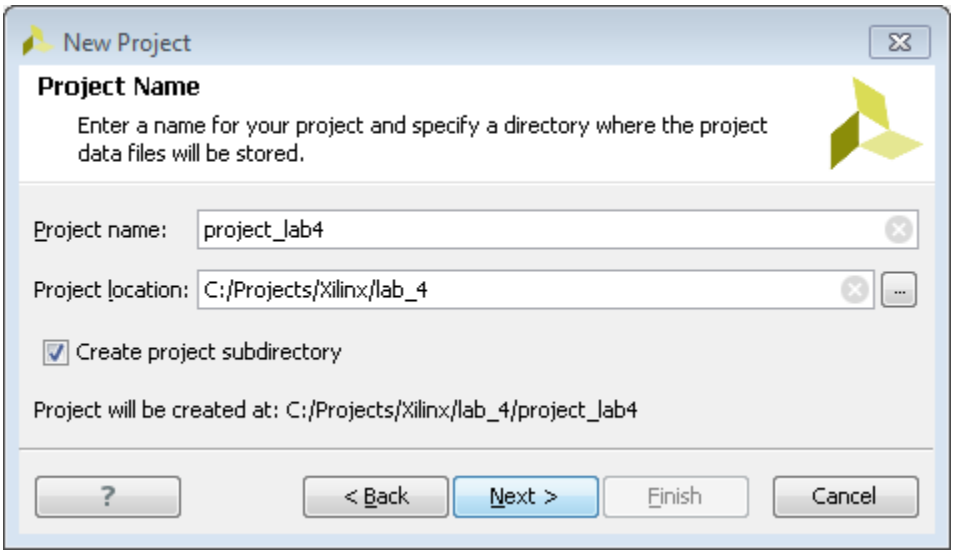


Figure 53: New Project – Project Name

4. Click **Next**.
5. Select RTL Project as the **Project Type** and **check Do not specify sources at this time**.
6. Click **Next**.
7. In the Default Part dialog box, select the **xcku040-ffva1156-2-e** part and click **Next**.
8. For this lab, you select an Ultrascale device. This device family is used for the initial compatibility of the custom IP.
9. Click **Finish** to close the New Project Summary page, and create the project.

   The Vivado IDE opens `project_lab4`, with the default layout.

Step 3: Package the Library Core

After creating the new empty project, the next step is to create and package the common IP directory. The order of the packaging the IP directories are important because they need to be packaged in the order of dependency. All of the child IP must be packaged prior to packaging the parent IP. You will be setting this IP directory as a library core which is a special kind of IP which is not for standalone use.

Use the Create and Package IP Wizard

1. From the Tools menu, select **Create and Package IP** to open the Create and Package IP Wizard.
2. Click **Next** at the Welcome screen for the Create and Package New IP dialog box.
3. In the Create Peripheral, Package IP, or Package Block Design dialog box, select **Package a specified directory**, as shown in the following figure.

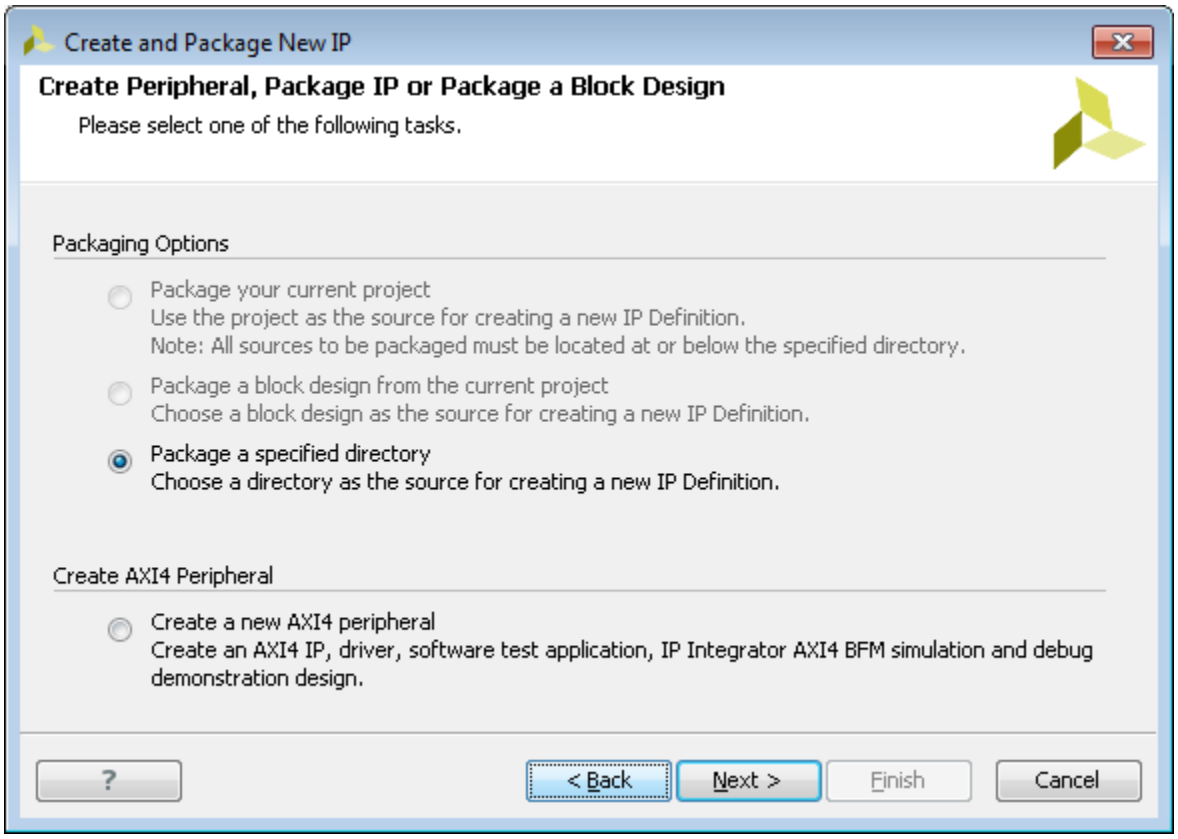


Figure 54: Create Peripheral, Package IP, or Package Block Design

4. Click **Next**.

5. In the Package a Specified Directory dialog box, shown in the following figure, set the options as follows:
   - **Directory**: `<Extract_Dir>/lab_4/trunk/common_v1_0`
   - Check the **Package as a library core** option.

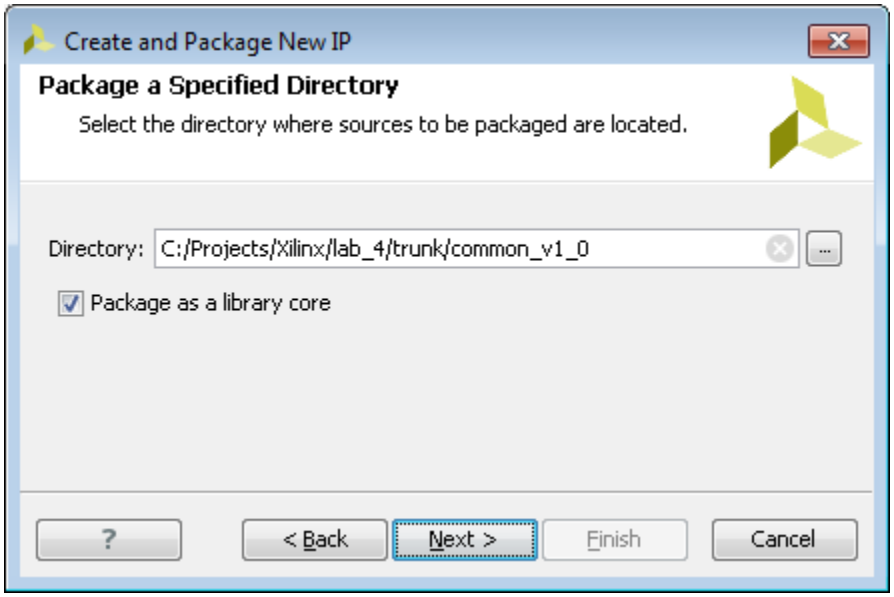


Figure 55: Package a Specified Directory

The **Package as a library core** option is used when the source is not intended to be used as a standalone IP. The option is intended to mark IP in the IP Catalog that can only be used as a child of another IP.

However, any Custom IP can be a child to another IP. If your Custom IP is not a library core, the process for referencing a child IP is the same. This option is just used to mitigate confusion of which IP should be used and hidden in the Vivado IP Catalog.

6. Click **Next**.
7. On the Edit in IP Packager Project Name window, leave the default locations, and click **Next**.
8. Click **Finish**.

   An edit IP project opens in a new Vivado window with the Package IP window opened. The Package IP window displays the basic IP package information determined through the wizard. The project is opened in the staging area for editing and repackaging.

Update the IP Information

Because you selected the library core option, the Package IP window has a subset of available options for packaging, as shown in the following figure.

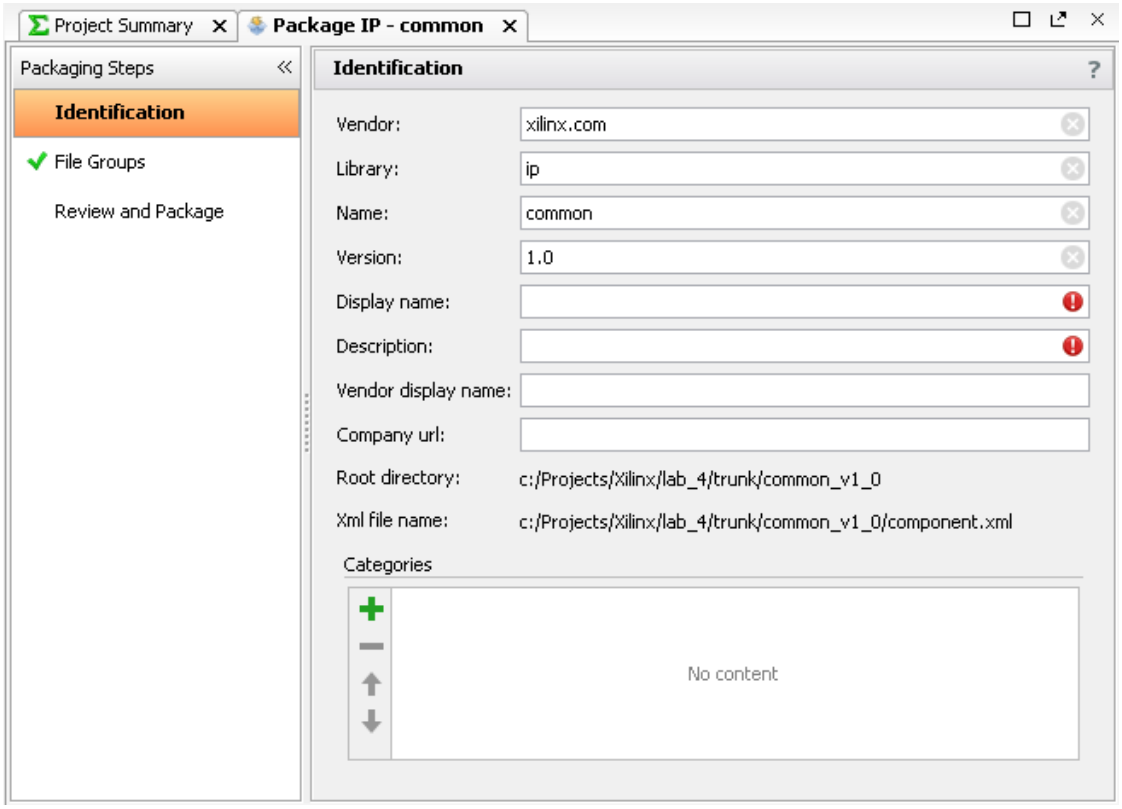


Figure 56: Package IP

1. Update the Identification information as follows:
   - **Vendor**: `my_company`
   - **Display Name**: `My Company Common Library`
   - **Description**: `My Company Common Library Files`
   - **Vendor Display Name**: `My Company`
   - **Company url**: `http://www.my_company.com`
2. Click the **Review and Package** page to view the name, location, and root directory information about the library core.
3. Click **Package IP** to update the IP with the updated identification information.

This completes the packaging for the `common_v1_0` library core. If prompted, you can close the `edit_ip_project`.

Step 4: Package the IP

Using the same project previously created, project_lab4, you will create and package the `myip_v1_0` IP directory. Since the common IP directory has already been packaged, all the required dependencies are available to package the parent IP.

Add the IP Repository

Before you package the parent IP, you must set the repository location in the project settings to include the `common_v1_0` IP that was just created in the IP Catalog.

1. In the **Flow Navigator** > **Project Manager** > select **Project Settings**.
2. In the Project Settings dialog box, select **IP** in the sidebar.
3. Select the **Repository Manager** tab, and click the **Add Repository** button.
4. In the IP Repositories dialog, select the path `<Extract_Dir>/lab_4/trunk` and press **Select** to add the repository.
5. The Add Repository dialog opens to display that the trunk repository was added to the project and 1 IP was found, as shown in the following figure.

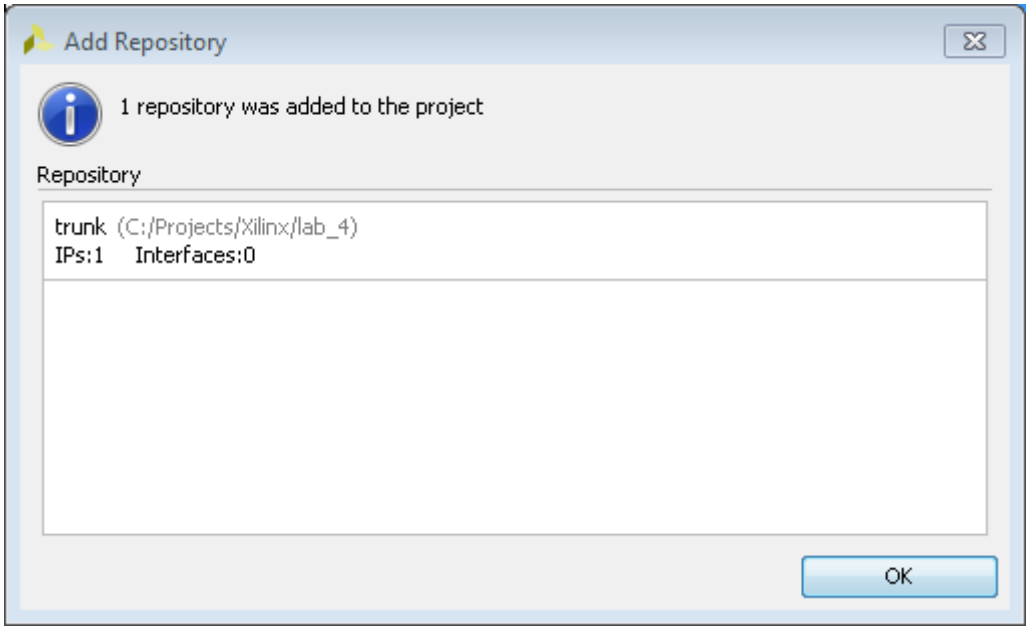


Figure 57: Add Repository

6. Click **OK**.

7. The Repository Manager tab is now populated with the selected IP repository, as shown in the following figure.

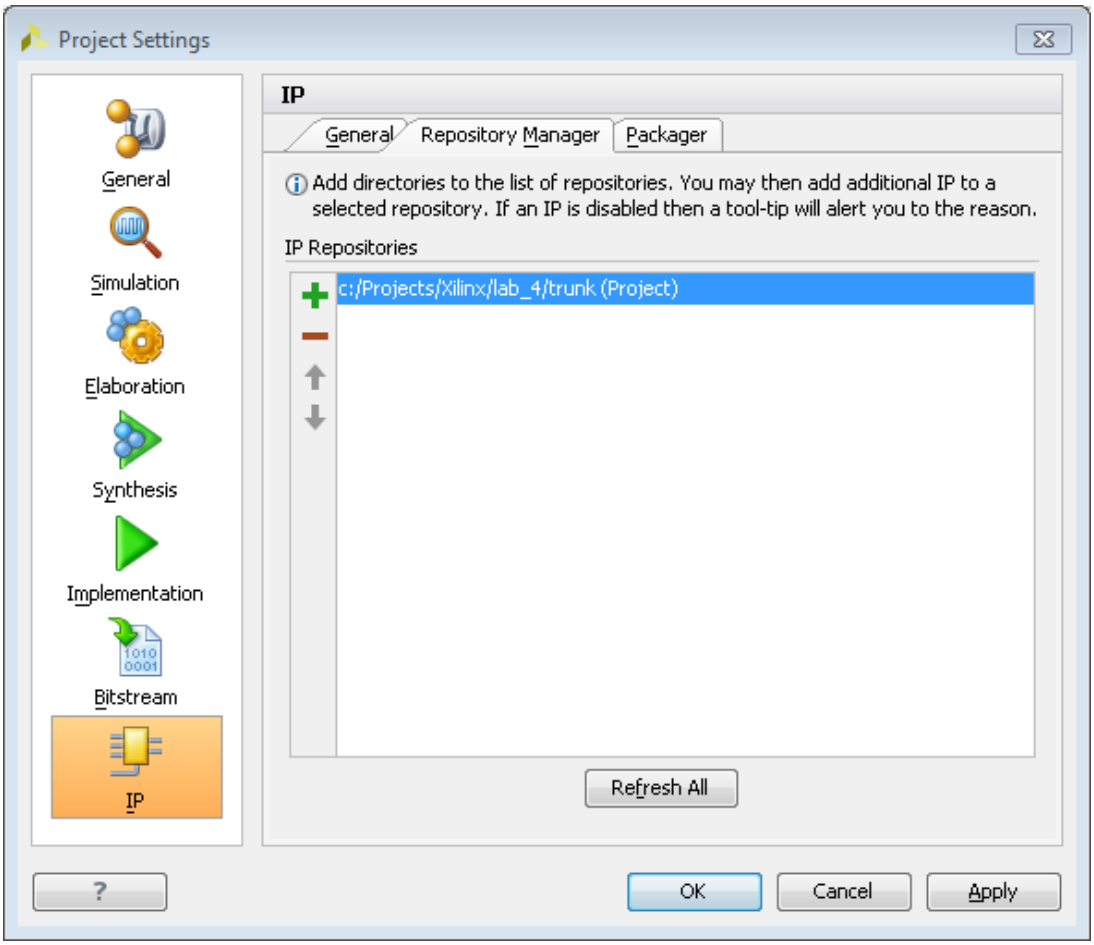


Figure 58: Project Settings – Repository Manager

8. Click **OK** to close the Project Settings dialog.

Use the Create and Package IP Wizard

1. From the Tools menu, select **Create and Package IP** to open the Create and Package IP Wizard.
2. Click **Next** at the Welcome screen for the Create and Package New IP dialog box.
3. In the Create Peripheral, Package IP, or Package Block Design dialog box, select **Package a specified directory**.
4. Click **Next**.

5. In the Package a Specified Directory dialog box, shown in the following figure, set the options as follows:
   - **Directory:** `<Extract_Dir>/lab_4/trunk/myip_v1_0`
   - **Do not check** the **Package as a library core** option.

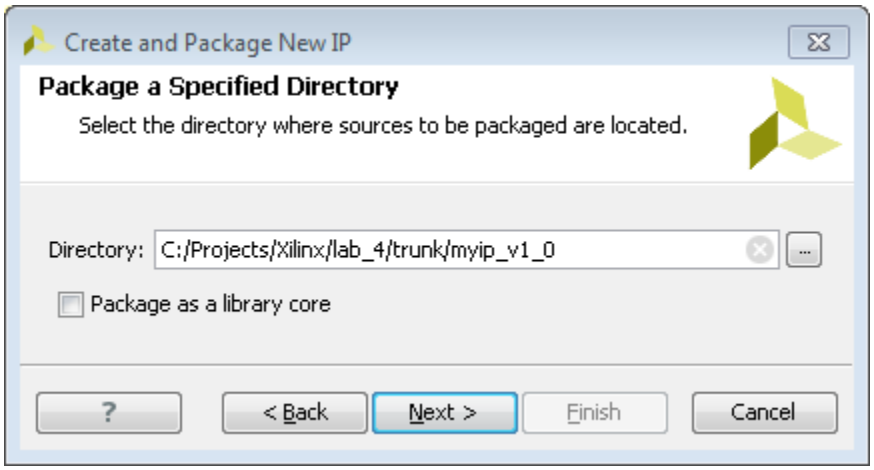


Figure 59: Package a Specified Directory

6. Click **Next**.
7. On the Edit in IP Packager Project Name window, leave the default locations, and click **Next**.
8. Click **Finish**.

   An edit IP project opens in a new Vivado window with the Package IP window opened, as shown in the following figure, to continue with the next steps.

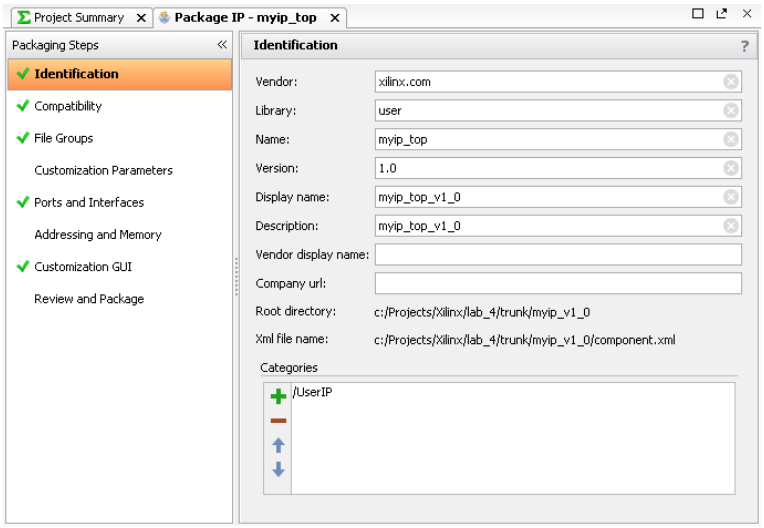


Figure 60: Package IP

9. Update the IP Information and Contents In the **Package IP** window, update the following information on the **Identification** page:
   - **Vendor**: `my_company`
   - **Display Name**: `My IP`
   - **Description**: `UG1119 Tutorial Lab #4 – My IP`
   - **Vendor Display Name**: `My Company`
   - **Company url**: `http://www.my_company.com`
10. Click the **File Groups** to examine the files included in the packaged IP, as shown in the following figure.

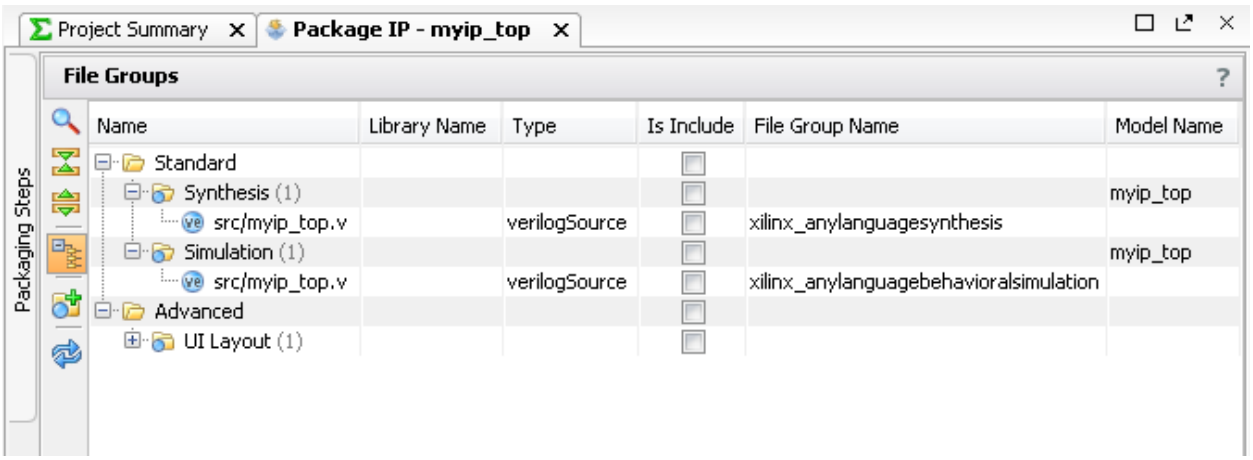


Figure 61: File Groups

The packaged IP only contains the top-level source file `myip_top` as this was the only file in the selected IP directory `<Extract_Dir>/lab_4/trunk/myip_v1_0`. This file instantiates the IP `common_v1_0`.

11. As reference, if you examine the Hierarchy Sources Viewer (HSV) in the project, you can see that the common module is missing, as shown in the following figure.

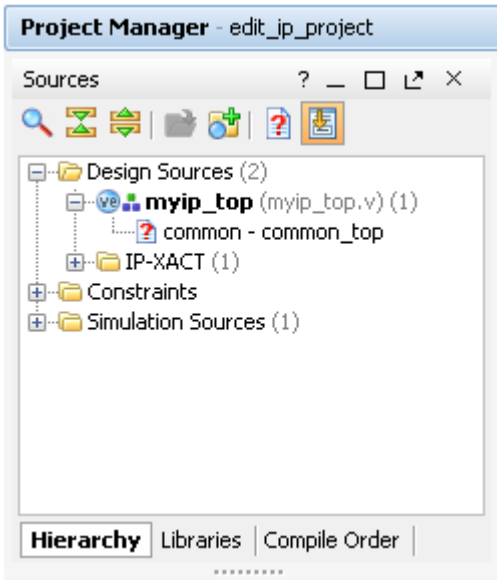


Figure 62: Hierarchy Sources Viewer

This is expected behavior, because you add the missing IP source files through the Package IP window.

12. In the File Group window, right-click on the Synthesis file group and select **Add Sub-Core Reference**, as shown in the following figure.

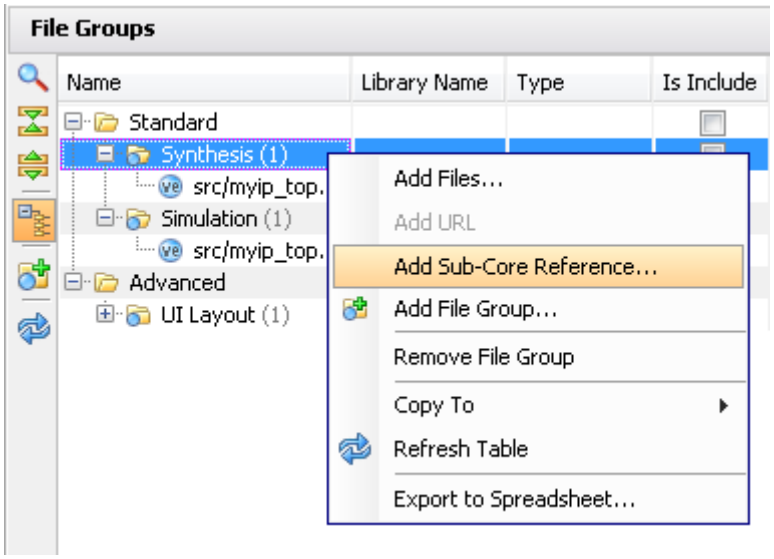


Figure 63: Add Sub-Core Reference Option

13. In the Add Sub-Core Reference dialog box, select the **My Company Common Library** that you created in the previous steps, as shown in the following figure.

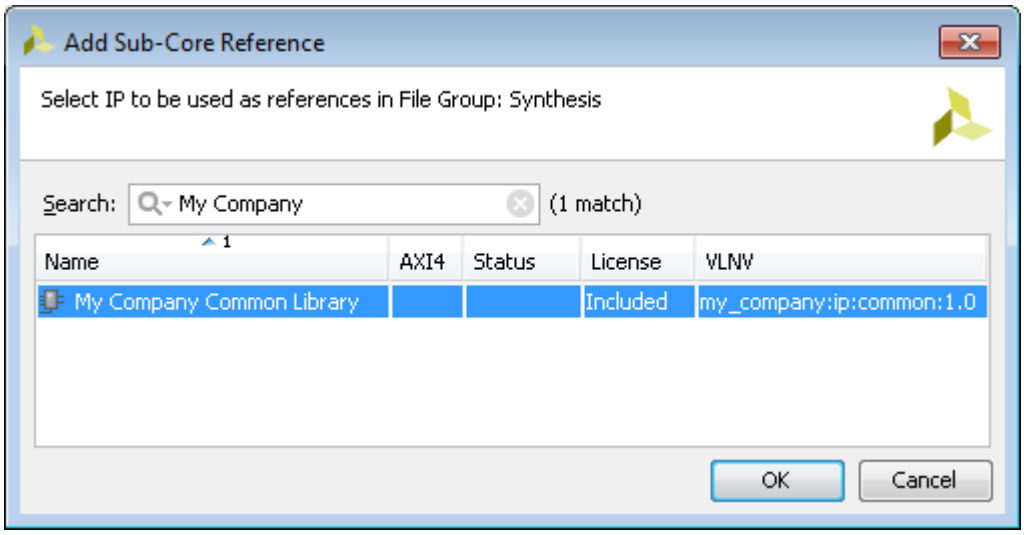


Figure 64: Add Sub-Core Reference (Synthesis)

14. Click **OK**.

The File Groups page is updated with the selected Sub-Core Reference under the Synthesis File Group, as shown in the following figure.

Adding an IP as a Sub-Core Reference informs the Vivado IDE to copy the files associated IP to the parent IP during generation; therefore, when `myip_v1_0` is generated, the `common_v1_0` files are copied to the location with the rest of the generated output products. This mechanism allows users to systematically share IP files.

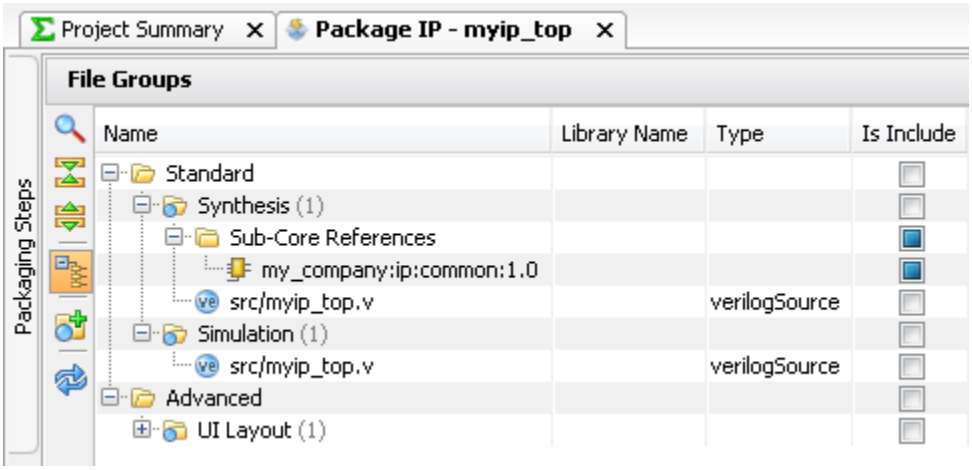


Figure 65: File Groups with Sub-Core Reference

The Sub-Core Reference is added for the Synthesis File Group, and the same process needs to be performed for Simulation.

15. In the File Group window, right-click the Simulation file group, and select **Add Sub-Core Reference**.
16. In Add Sub-Core Reference dialog box, select the **My Company Common Library**.
17. Click **OK**.

    The Sub-Core References are now added to both the Synthesis and Simulation File Groups, as shown in the following figure. The necessary files from the `common_v1_0` IP are available to `myip_v1_0` for both Synthesis and Simulation.

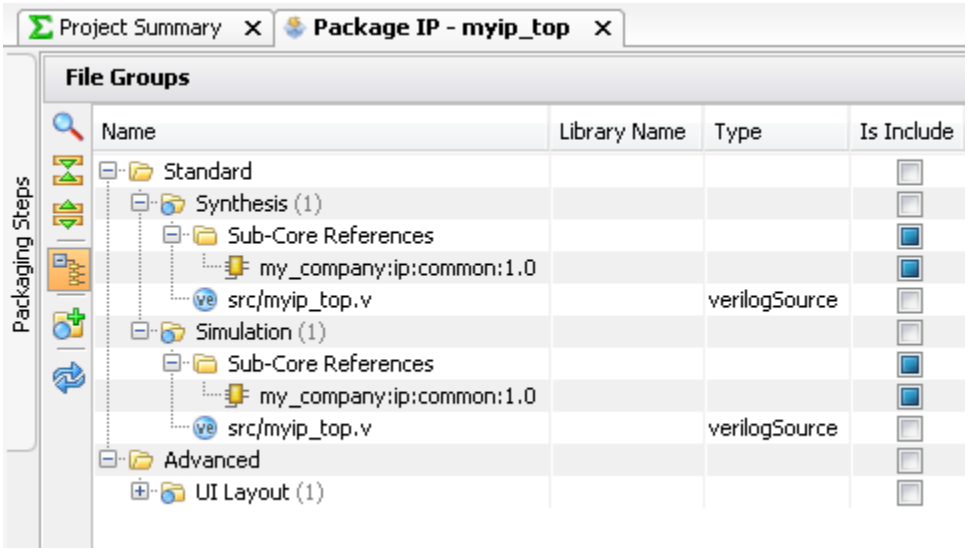


Figure 66: File Groups with Complete Sub-Core References

18. Click the **Review and Package** page to view the name, location, and root directory information about the IP.
19. Click **Package IP** to update the IP with the updated identification and Sub-Core Reference information.

    This completes the packaging for the `myip_v1_0` IP. If prompted, you can close the `edit_ip_project`.

    ***Note**: Adding a sub-core reference in the Package IP window does affect the state of the edit IP project. The HSV continues to display the missing modules located within the sub-core reference. This information only exists within the Package IP window and `component.xml`. If you want to verify the IP with the files from the Sub-Core Reference, you can reopen the packaged IP in an edit IP project through the IP Catalog and the associated Sub-Core Reference files will be present.*

Step 5: Validate the IP

After completing the packaging of the `common_v1_0` and `myip_v1_0` IP, you can use `project_lab4` to validate the generation of `myip_v1_0`.

1. In the **Flow Navigator** > **Project Manager**, select **IP Catalog**.
2. In the search field at the top of the IP Catalog, type **My IP**.

   The My IP core shows under the `/UserIP` directory, as shown in the following figure.

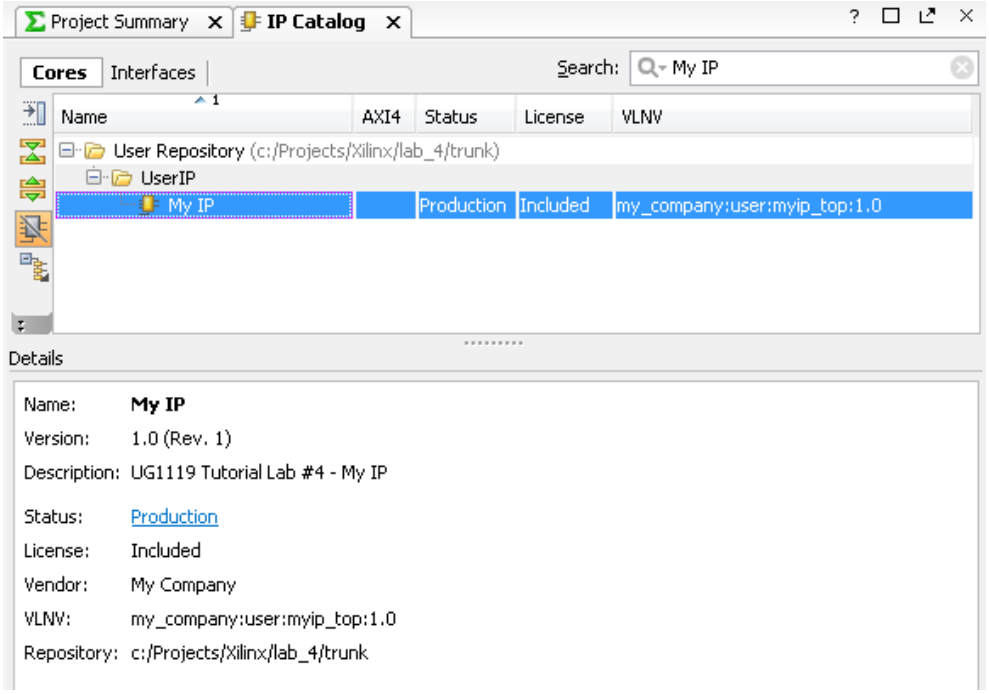


Figure 67: IP Catalog

3. **Right-click** the My IP core and select **Customize IP**.
4. In the Customization IP GUI, click **OK**.
5. In the Generate Output Products dialog box, select **Generate**.

   By default, the IP is generated out-of-context (OOC), which means the IP is synthesized standalone, and producing a DCP file. This IP example has not been optimized for ideal use for OOC synthesis. For more information regarding proper use of your custom IP for OOC synthesis, see the Lab 1: Packaging a Project exercise.
6. Click **OK** to close the Generate Output Products message box.
7. After the Out-of-Context Module Run completes successfully, close the project and exit Vivado.

Conclusion

This concludes Lab #4.

You have successfully created two IP within a repository trunk, and created an IP that referenced another IP through a sub-core reference.

In this lab, you did the following:

- Used the Create and Package IP Wizard to package a specified directory for the `common_v1_0` library core.
- Used the Create and Package IP Wizard to package a specified directory for the `myip_v1_0` IP.
- Referenced the `common_v1_0` IP as a Sub-Core Reference in `myip_v1_0` in the File Groups page.
- Validated the generation of the `myip_v1_0` IP.

Please Read: Important Legal Notices