Vivado Design Suite Tutorial

Creating and Packaging Custom IP

UG1119 (v2020.2) February 4, 2021

XILINX®

Revision History

The following table shows the revision history for this document.

Section	Revision Summary
02/04/2021 Version 2020.2	
General updates	Validated for release 2020.2.
06/24/2020 Version 2020.1	
General updates	Validated for release 2020.1.

Table of Contents

Introduction to Creating and Packaging Custom IP

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. 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 Vivado IP packager is a unique design reuse feature, which is based upon the IP-XACT standard. The IP packager 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.*

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

1. Download the design files from the Reference Design Files on the Xilinx website.
2. Extract the zip file contents into any write-accessible location.

Lab 1

Packaging a Project

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

1. Launch the Vivado® IDE.
   - On Linux:
     - Change to the directory where the lab materials are stored: `cd <Extract_Dir>/lab_1`.
     - Launch the Vivado Design Suite IDE: `vivado`.
   - On Windows:
     - Launch the Vivado Design Suite IDE:

       Select **Start → All Programs → Xilinx Design Tools → Vivado 2020.x → Vivado 2020.x**.

       Or

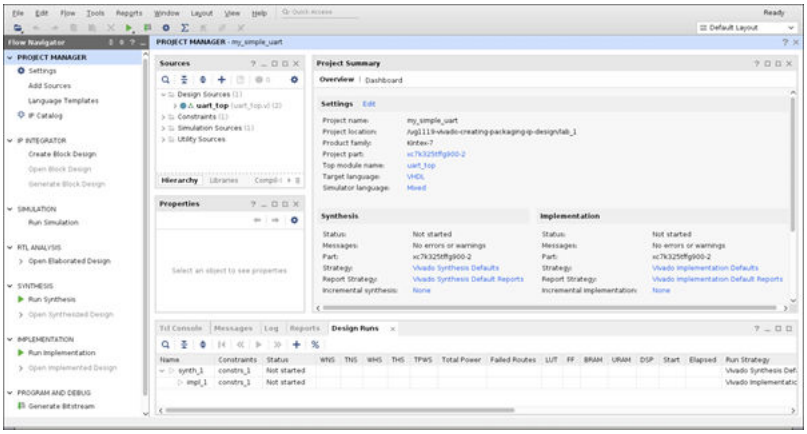
       Click the Vivado 2020.x desktop icon to start the Vivado IDE.

   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.
2. Click **Open Project**, and browse to: `<Extract_Dir>/lab_1`.

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

3. 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.



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 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.

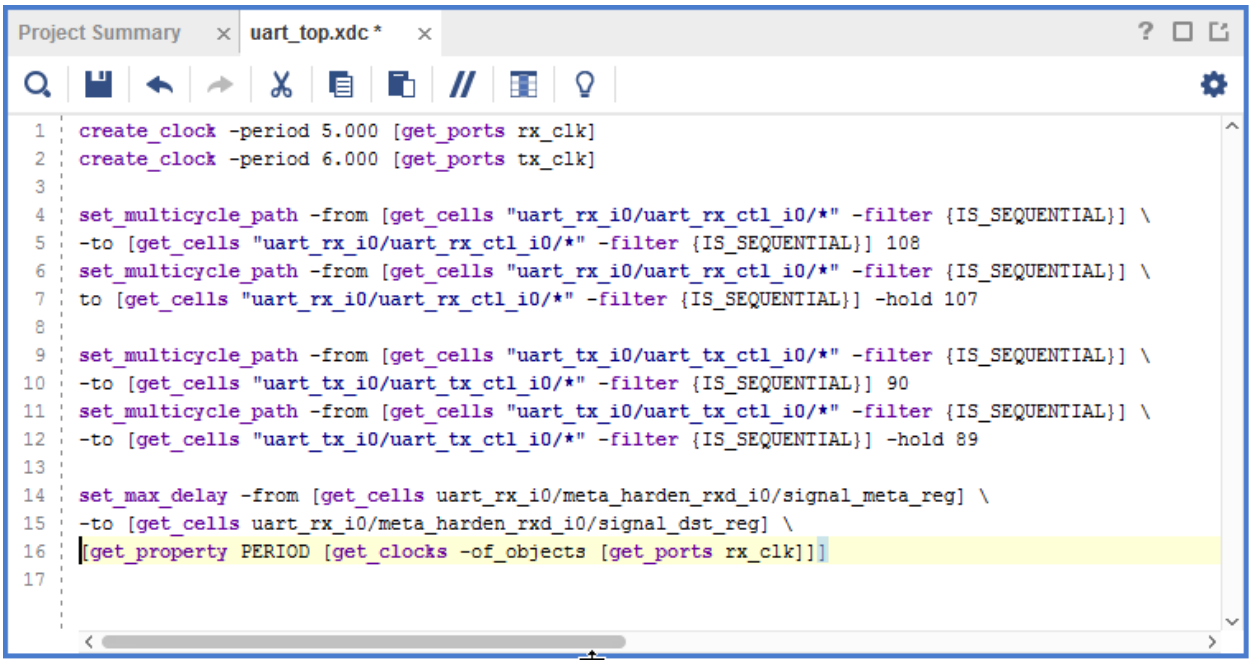
When used in the context of a top-level design, the parent XDC file provides the clock constraints and the standalone OOC XDC file is not needed.

For more information on the 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, as shown in the following figure:



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

   - create_clock constraints (lines 1 and 2)
   - set_max_delay constraint relying on the clock object period value (line 14).

   Note: The line numbers referenced in the above figure 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 necessary 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 the parent design provides the clock.

   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.

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, mark the XDC files of the UART IP for LATE processing.

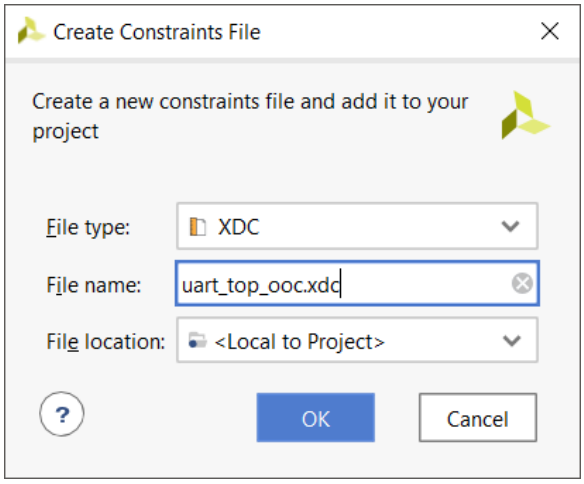
TIP: *Xilinx® delivered IP with `_clock` appended to the XDC filename are all marked for LATE processing.*

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

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

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



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.

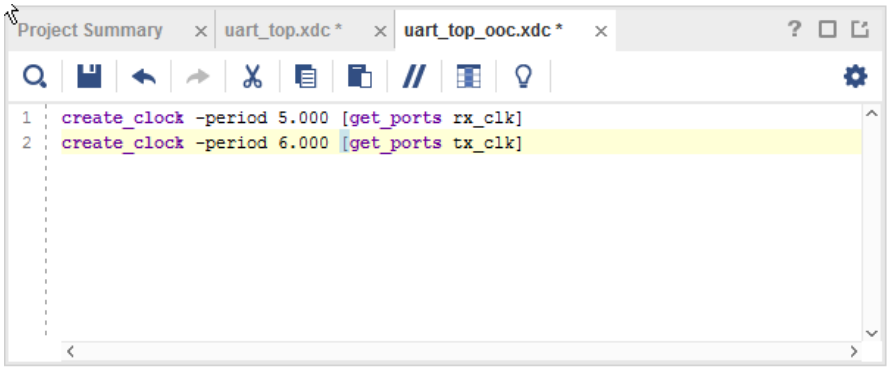
   The Vivado® tools create a new XDC file in the project and displays the file under the Constraints section in the Hierarchy pane of the Sources window.

   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.

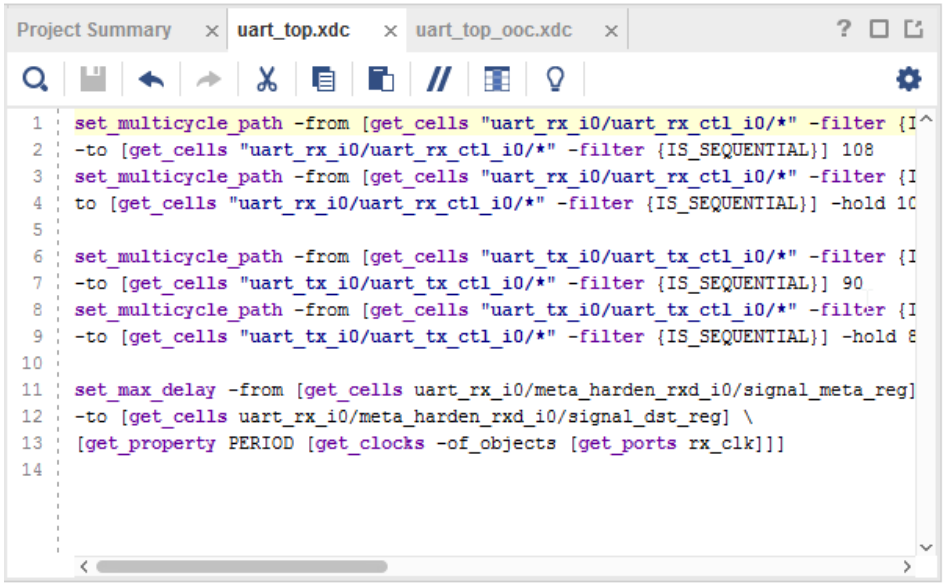


9. Right-click in the text area and select **Save All Files**.

   This saves both XDC files that are currently open.

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.



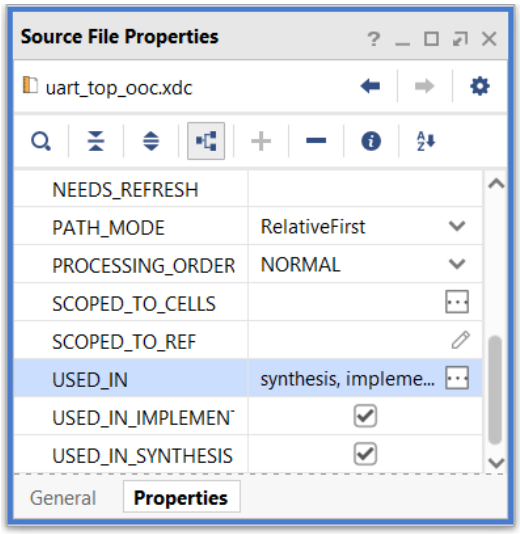
11. Close the two open XDC files.

    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 tools processes the constraint files for the IP correctly.

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

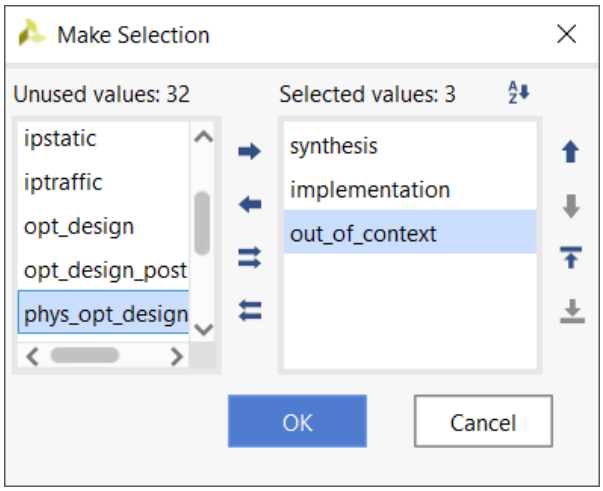
    The Source File Properties window displays the file automatically.

13. In the Properties pane of the Source File Properties window, scroll down to find the Used In selection, shown in the figure below.



The Make Selection dialog box opens.

14. Select **out_of_context** in the unused values and select the **Move Right** button, to add the value to the USED_IN property, shown in the following figure.



15. (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:

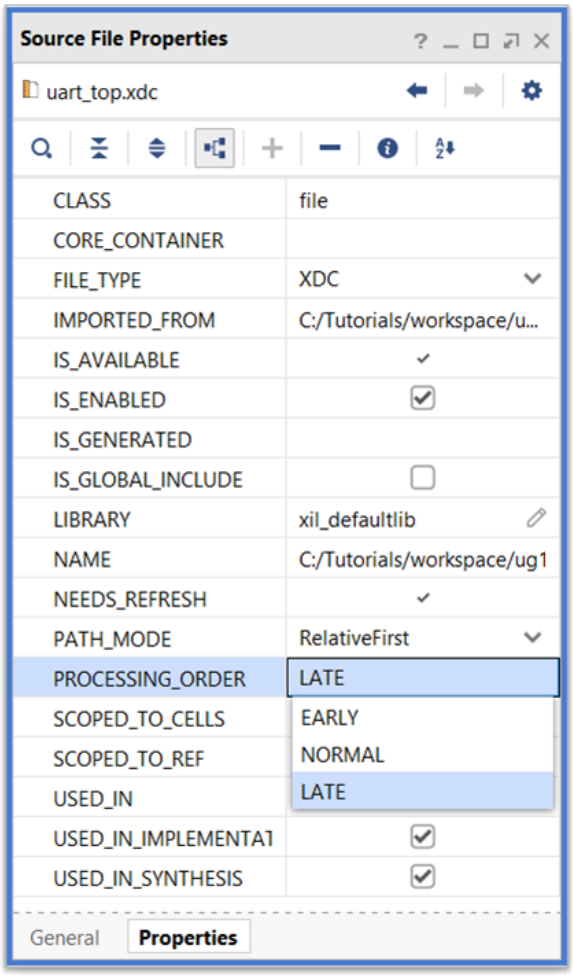
```
set_property USED_IN {synthesis implementation out_of_context}\
[get_files uart_top_ooc.xdc]
```

16. 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 (OOC) 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 OOC, 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. In the Source File Properties window, scroll down and change the PROCESSING_ORDER property value to **LATE**, as shown in the following figure.



You could also change the property value 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

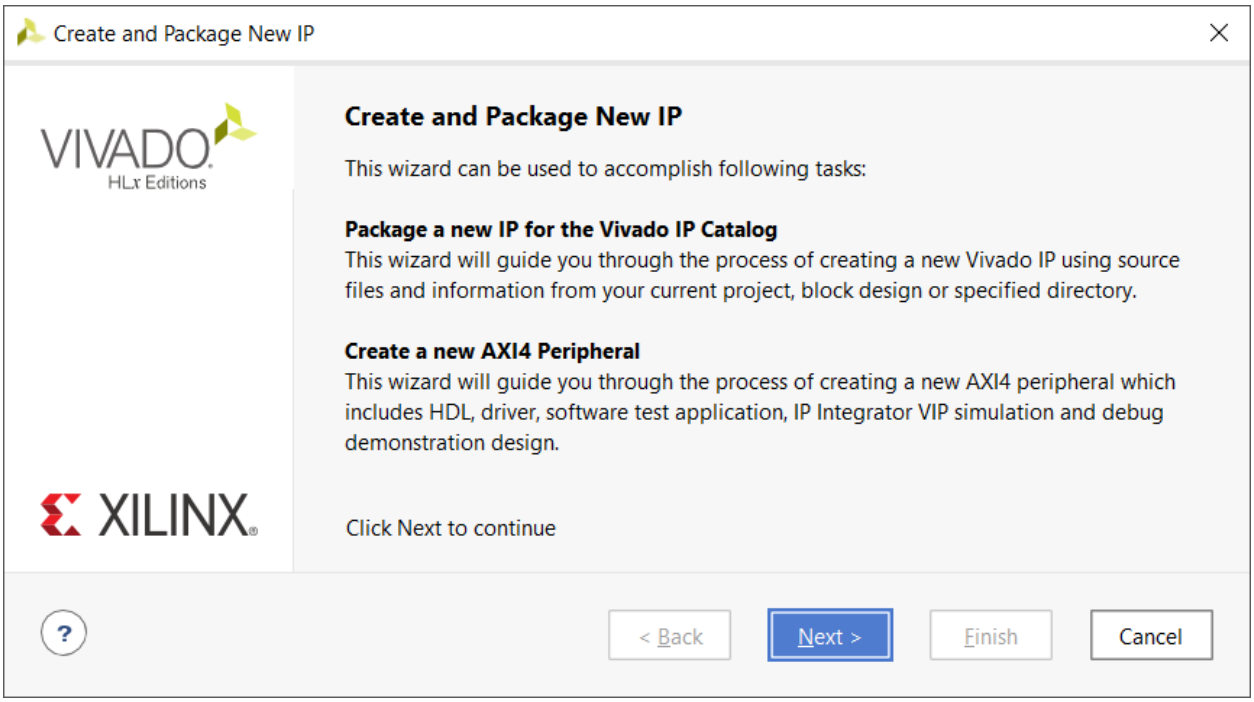
After setting up the design and supporting constraint files, the next step is to create and package the new IP Definition, and add it to the IP catalog.

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

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

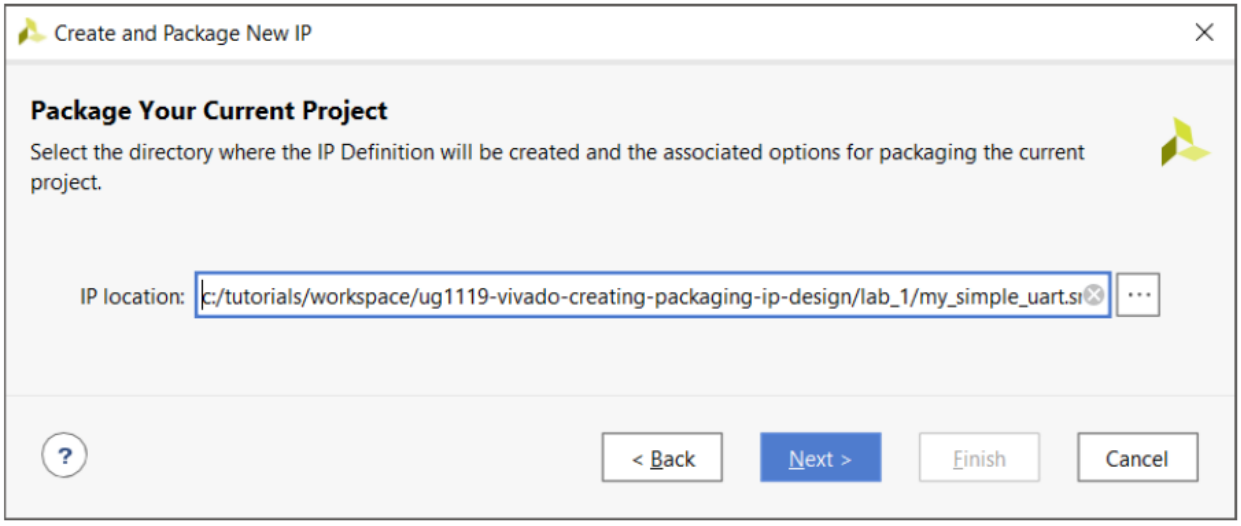
2. Click **Next**.

The Create and Package new IP dialog box opens, as shown in the following figure.



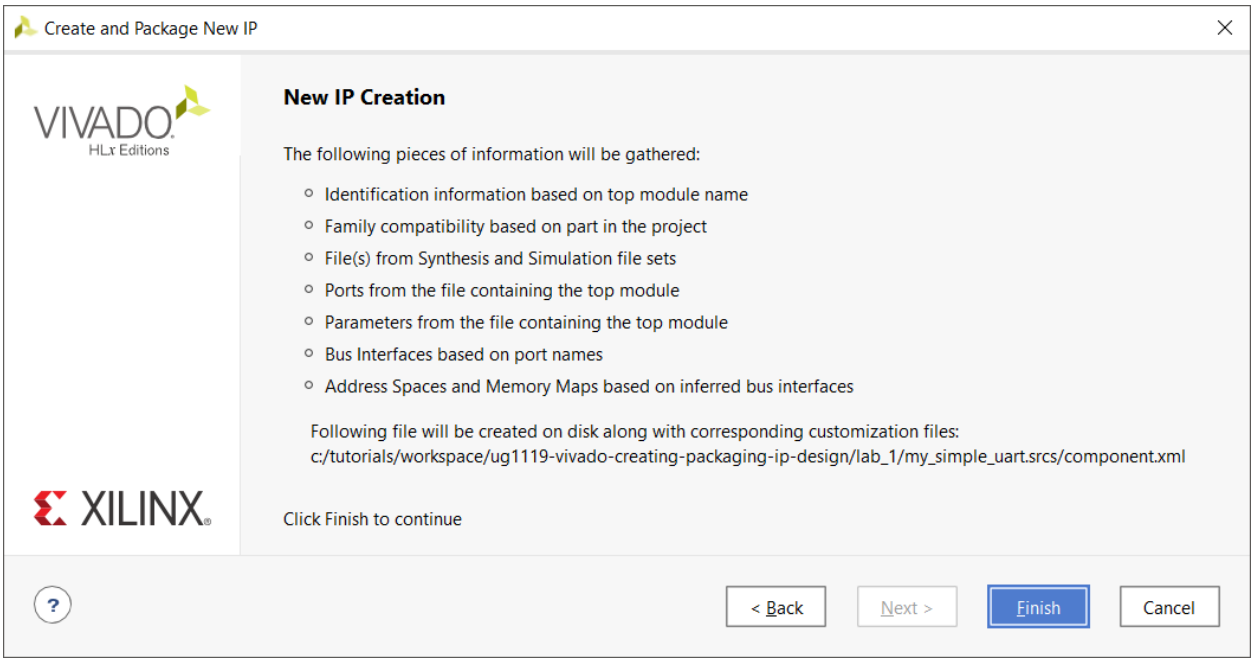
3. Click **Next**.
4. Select the **Package your current project** option to use the current project as the source for creating the new IP Definition.
5. Click **Next**.

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



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

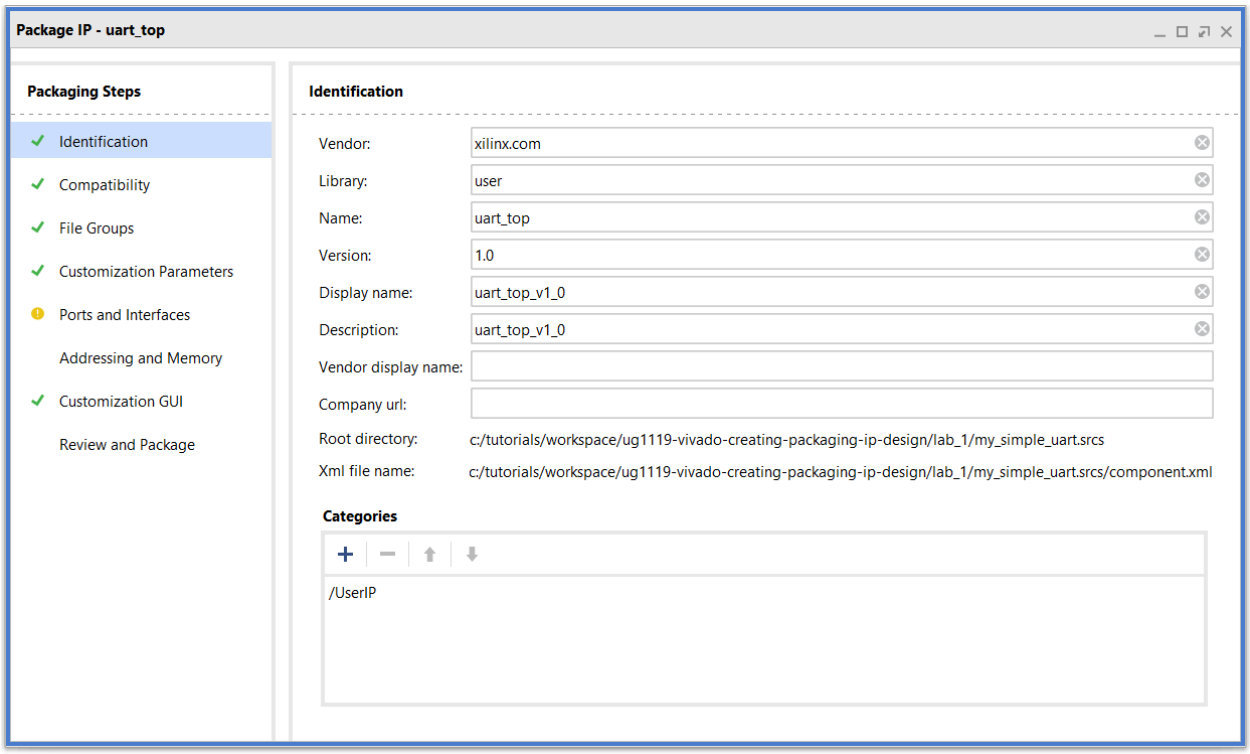
The New IP Creation dialog box, as shown in the following figure, opens with a summary of the information the wizard will automatically gather from the project.



7. 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.

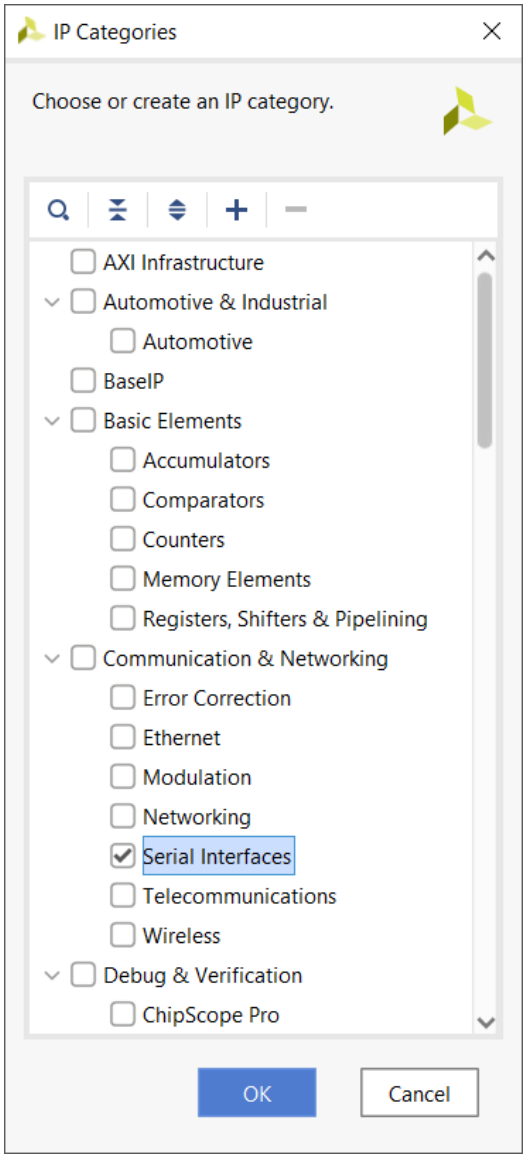
   The Package IP window opens and displays the basic IP package in a staging area for editing and repackaging, as seen in the following figure.



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`
   - **Version:** `1.0`
   - **Display name:** `My Simple UART`
   - **Description:** `My simple example UART interface`
   - **Vendor display name:** `<My Company>`
   - **Company url:** `<company_URL>`
2. For the Categories option, select the **Add** button + to open the Choose IP Categories dialog box, as shown in the following figure.



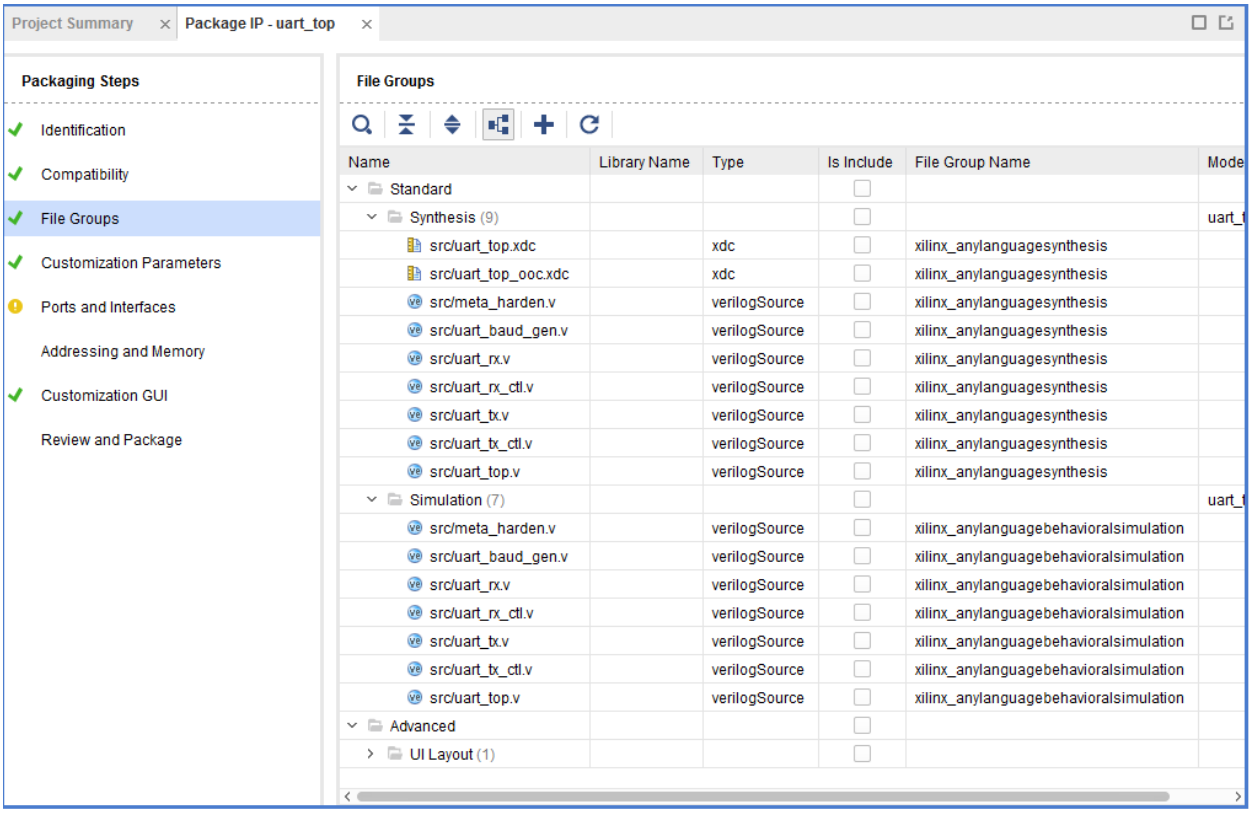
The Choose IP Categories dialog box lets you select various appropriate categories to help classify the new IP definition. When you add the IP definition to the IP catalog, the IP lists under the specified categories.

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

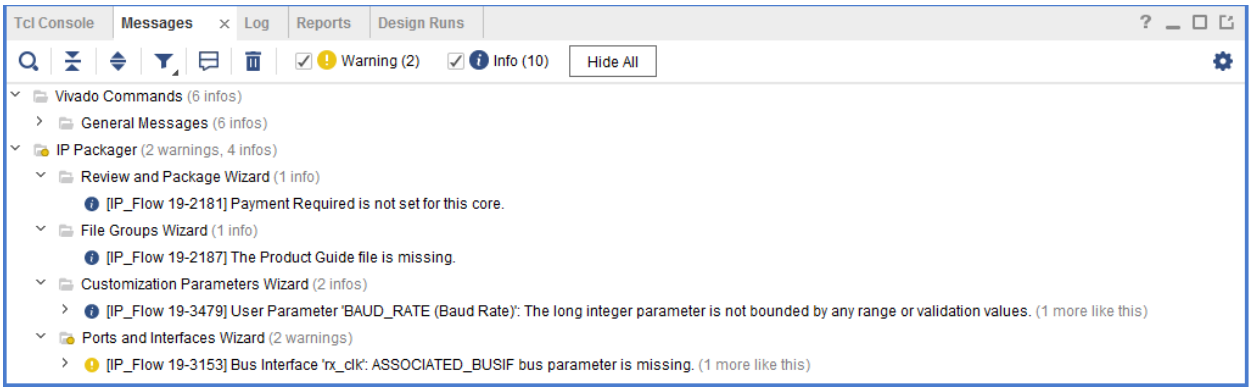
Step 5: Add a Product Guide to the IP

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 page provides a listing of the files to package as part of the IP, as shown in the following figure:



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



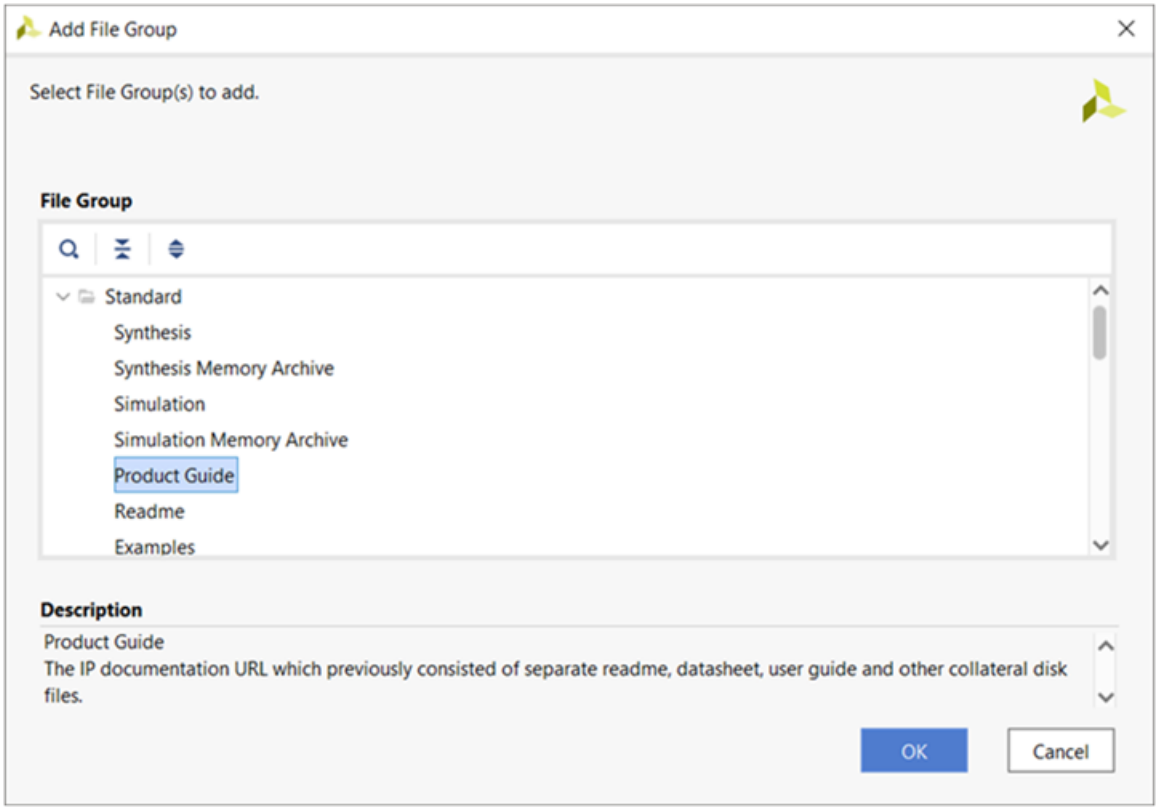
The IP packager messages inform you of the state of the IP. The File Groups Wizard message indicates that the IP definition does not include any documentation.

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. The reason for the warning is that the IP integrator tool works best with interfaces, and it expects that you would typically be using AXI interfaces. You do not have any bus interfaces in your design, and therefore, you can safely ignore this warning.

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 the following figure:

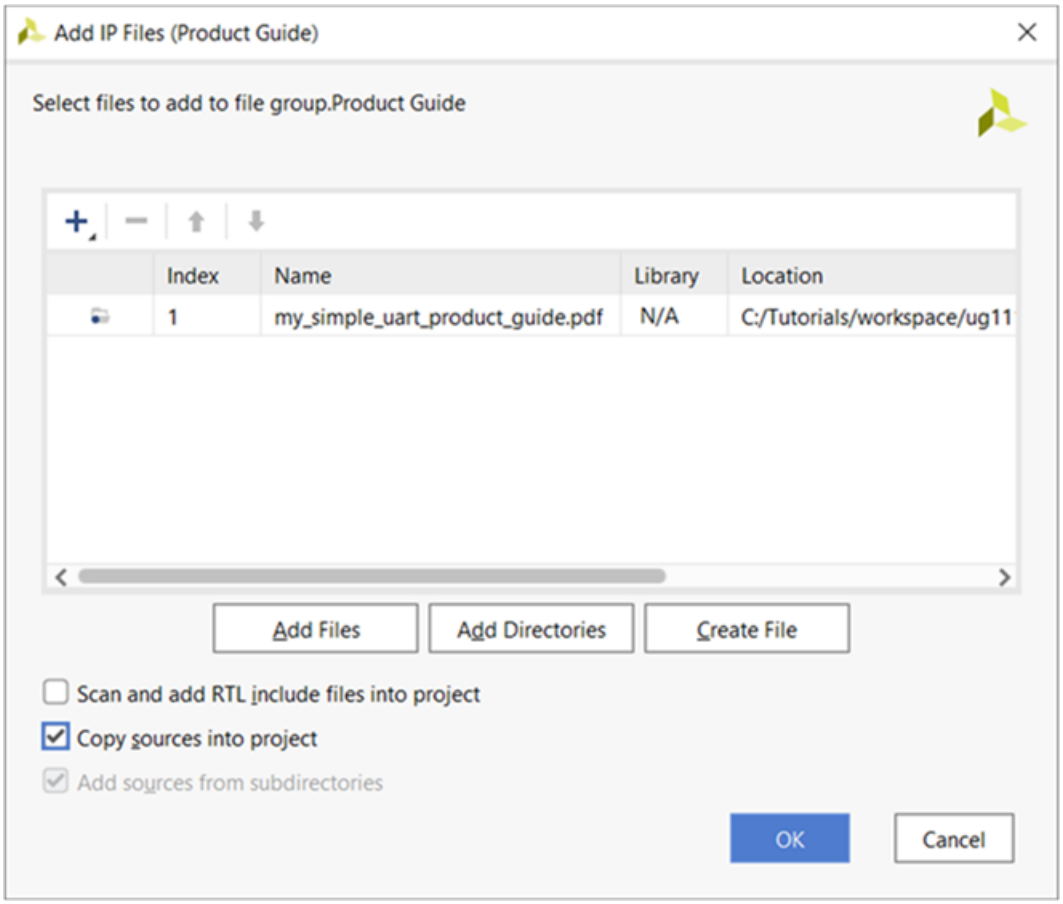


5. Click **OK**.

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.

6. Right-click the **Product Guide** file group, and select **Add Files**.
7. In the opened Add IP Files (Product Guide) dialog box, click **Add Files**.
8. Browse to `<Extract_Dir>/lab_1/docs`, and select **All Files** in the Files of type: entry line.
9. Select **my_simple_uart_product_guide.pdf**, and click **OK**.
10. 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 imports into the project sources directory, and not remotely referenced by the IP packager.



11. Click **OK**.

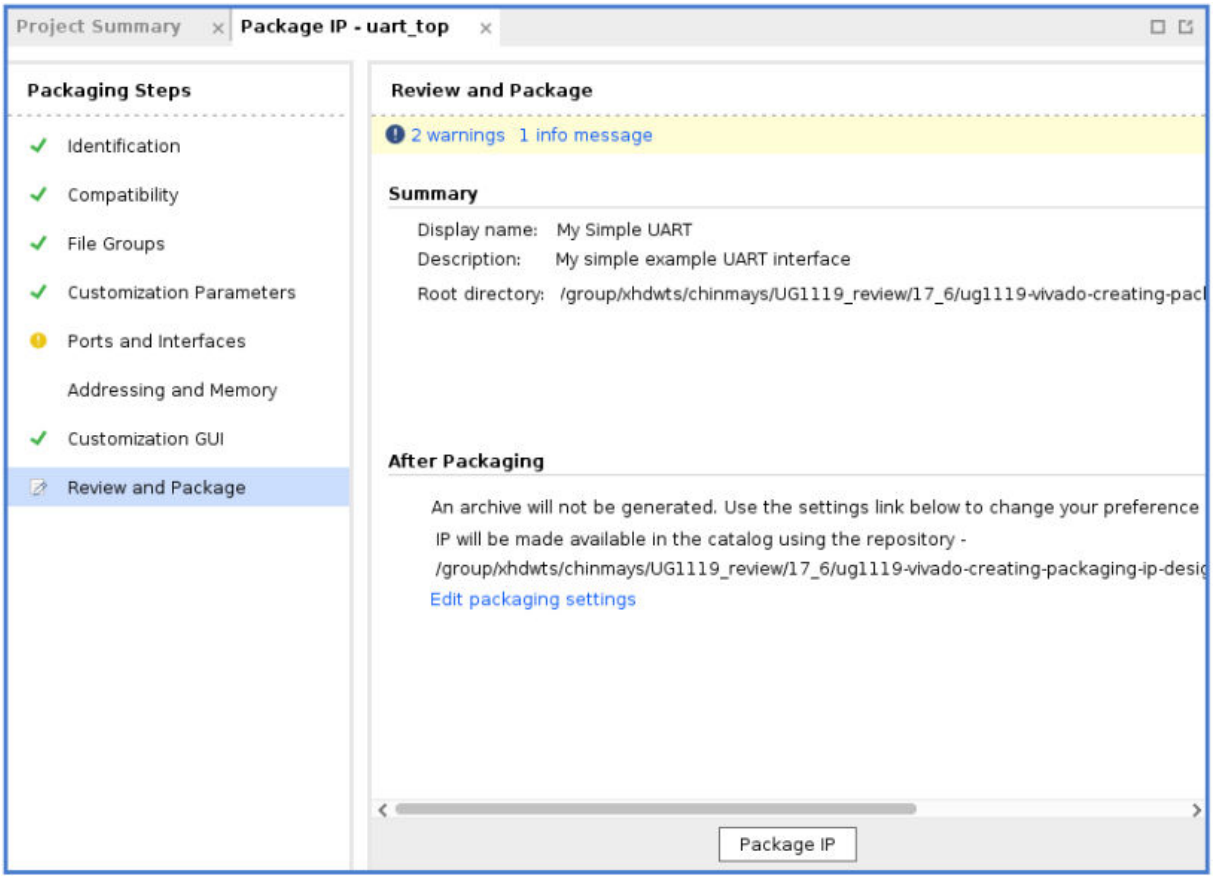
    The IP packager adds the PDF file of the Product Guide to the files defined as part of the IP, and resolves the Documentation Info check.

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:



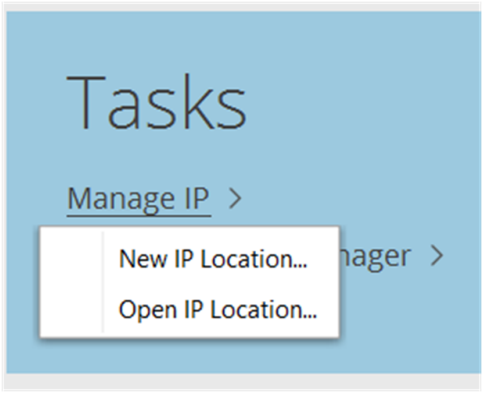
   With default settings of the current project, Vivado does not generate an archive for this IP after packaging. This is reflected in the After Packaging section of the Review and Package pane of the Package IP window.

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 from the File menu.

Step 7: Validate the New IP

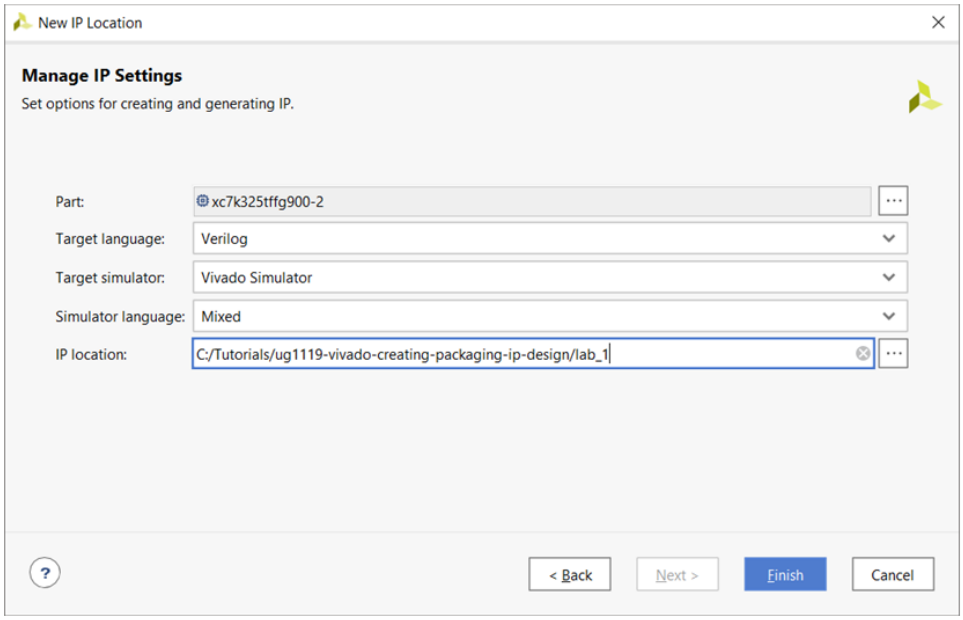
With the new custom IP definition packaged and added to the IP Catalog, you can validate that the IP works as expected when added to designs. To validate the IP, add a new customization of the UART IP to a project, and synthesize the design.

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



> **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 the following figure.
   - **Part:** xc7k325tffg900-2
   - **Target language:** Verilog
   - **Target Simulator:** Vivado simulator
   - **Simulator Language:** Mixed
   - **IP Location:** `<Extract_Dir>/lab_1`



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

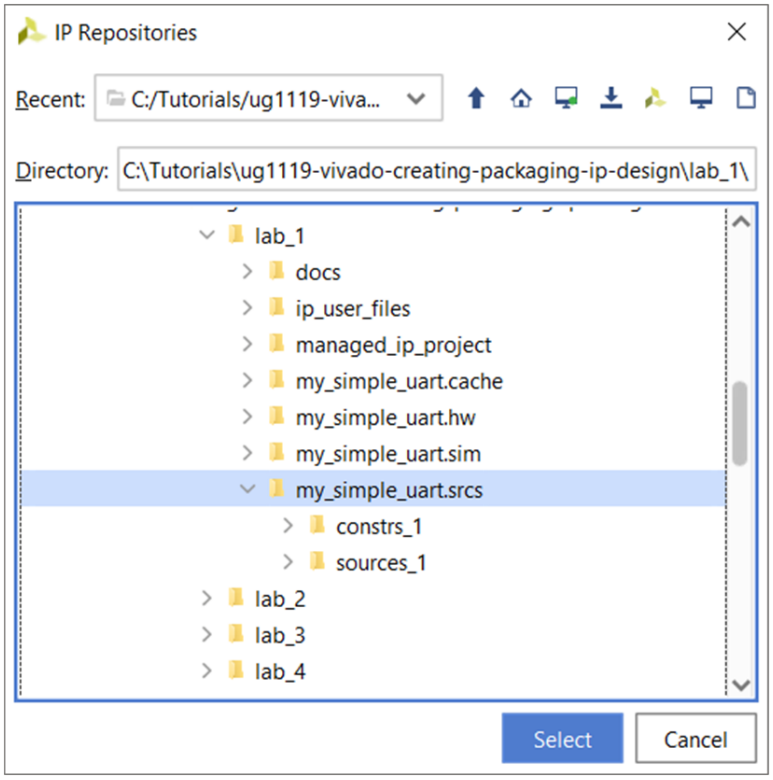
   A new Manage IP project opens in the Vivado IDE. The IP catalog opens automatically in a Manage IP project; however, the IP Catalog does not contain the repository used to package the custom UART IP.

   You now add the IP repository to the IP catalog.

5. In the IP Catalog window, right-click and select **IP Settings**, and expand IP to show Repository.
6. In the Repository Manager tab, click the **Add Repository** button to show and then select the IP Repositories Dialog Box.
7. In the IP Repositories dialog box, browse to and select the following location:

   `<Extract_Dir>/my_simple_uart.srcs/`

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



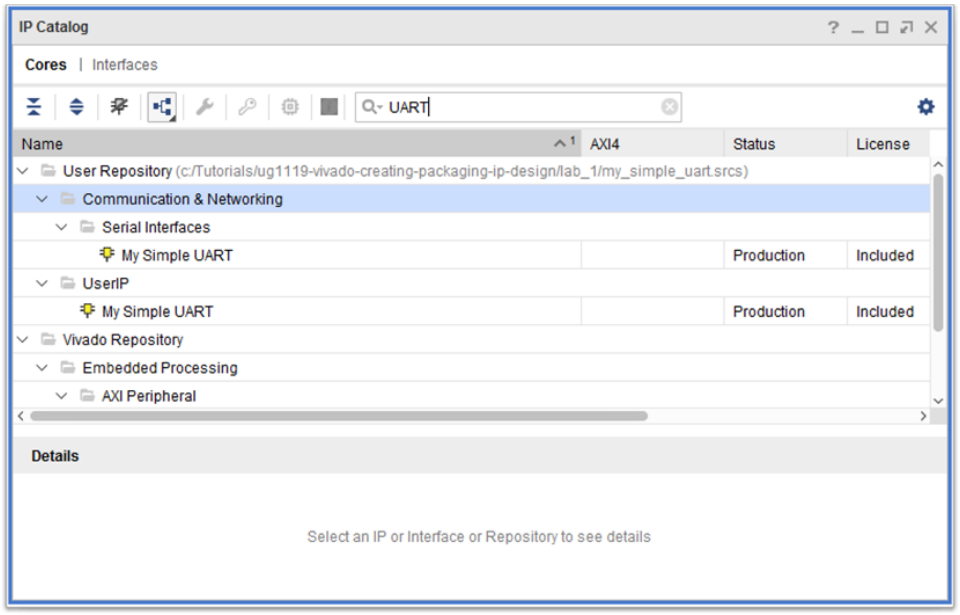
The added location displays in the IP Repositories section, and any packaged IP found in the repositories displays under the IP in Selected Repository. The My Simple UART IP definition that you packaged in Step 3: Package the IP is listed.

9. Click **OK** twice 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*** *→* ***Settings*** *command in the Vivado IDE to set the Default IP Repository Search Paths under the General IP options. The default IP repository search path is stored in the* `vivado.xml` *file, and added to new projects using the IP_REPO_PATHS property for the current_fileset :* `set_property IP_REPO_PATHS {…} [current_fileset]`*. (See the Vivado Design Suite Properties Reference Guide (UG912) for more information.)*

10. 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, as shown in the following figure.

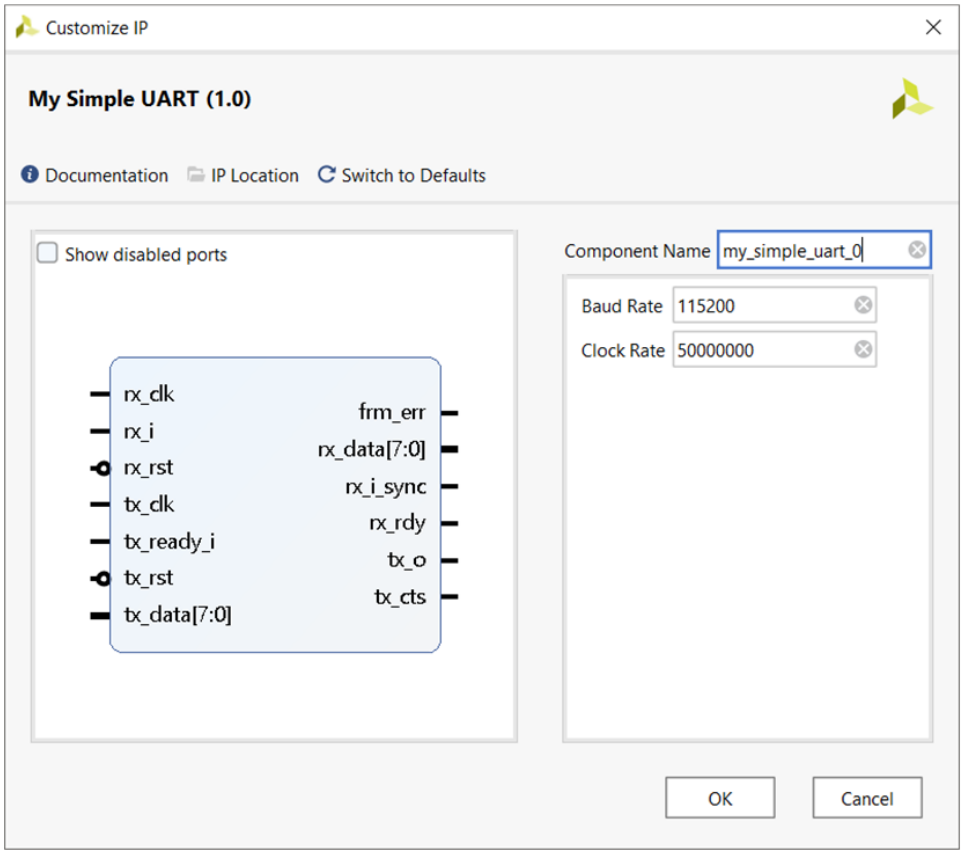


***Note*:** This IP Catalog view shows when the Taxonomy and the Repository options are selected for grouping the IP. See the*Vivado Design Suite User Guide: Creating and Packaging Custom IP* (UG1118) for more information about IP Groups.

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

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

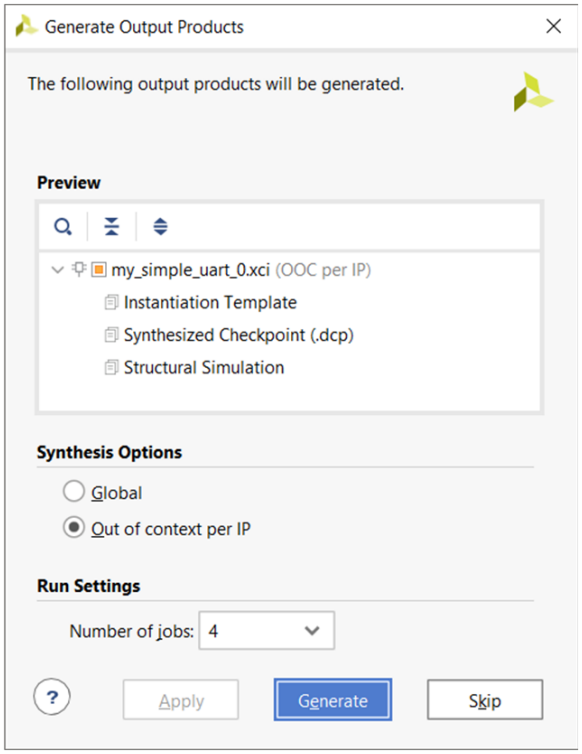
13. In the IP catalog, double-click **My Simple UART** to open the Customize IP dialog box, shown in the following figure.



14. (Optional) In the Customize IP dialog box, click **Documentation** and open the Product Guide.
15. 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.



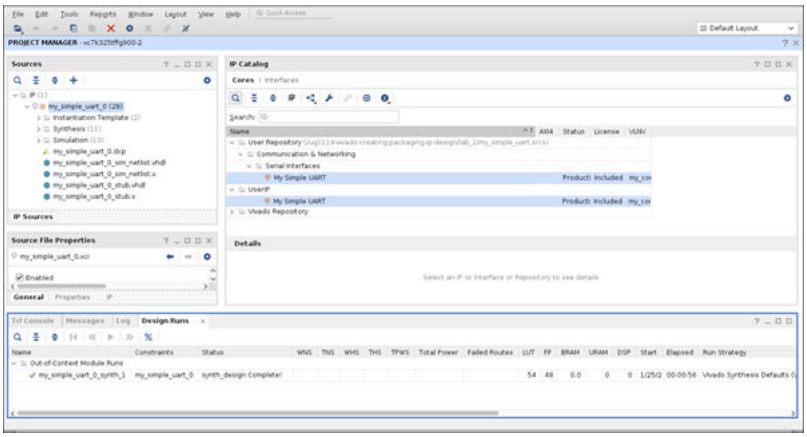
16. Click **Generate**.

    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 generated successfully.

17. Click **OK**.
18. Examine the IP Sources window and the various design and simulation source files that are added to the project.
19. In the Design Runs window, shown in the following figure, verify that the Out-Of-Context synthesis run was successful.



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.

Lab 2

Packaging a Specified Directory

Introduction

In this lab, you 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 test bench for the custom IP.

   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 User Guide: Creating and Packaging Custom IP* (UG1118).

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

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_2`.
- Launch the Vivado® IDE: `vivado`.

On Windows:

- Launch the Vivado Design Suite IDE:

  Select **Start → All Programs → Vivado 2020.x → Vivado 2020.x**.

  Or

  Click the Vivado 2020.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 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`
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**.

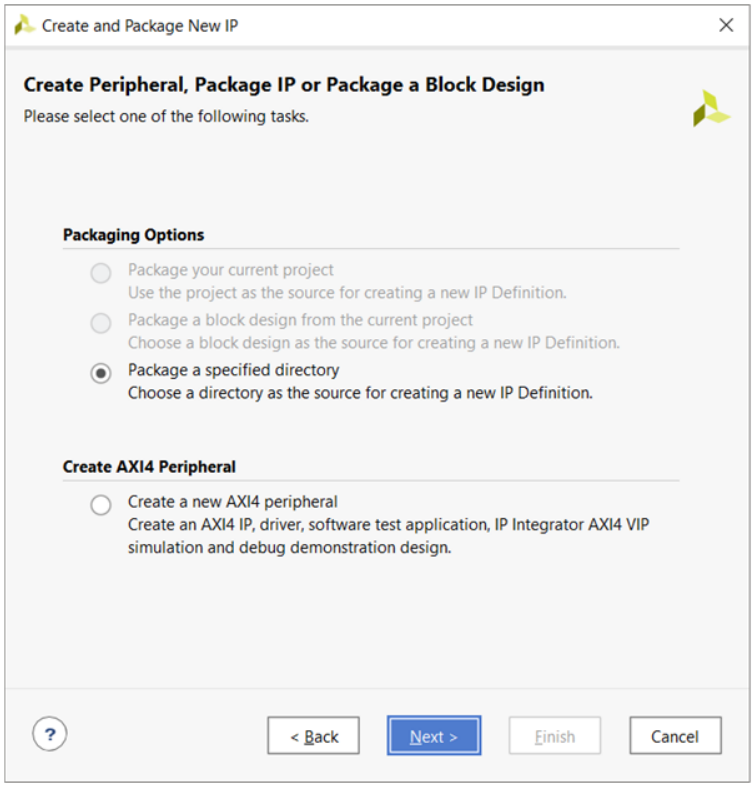
   For this lab, you select a Kintex®-7 device. This device family is used for the initial compatibility of the custom IP.
8. Click **Finish** to close the New Project Summary page, and create the project.

The Vivado IDE opens `project_lab2`, with the default layout.

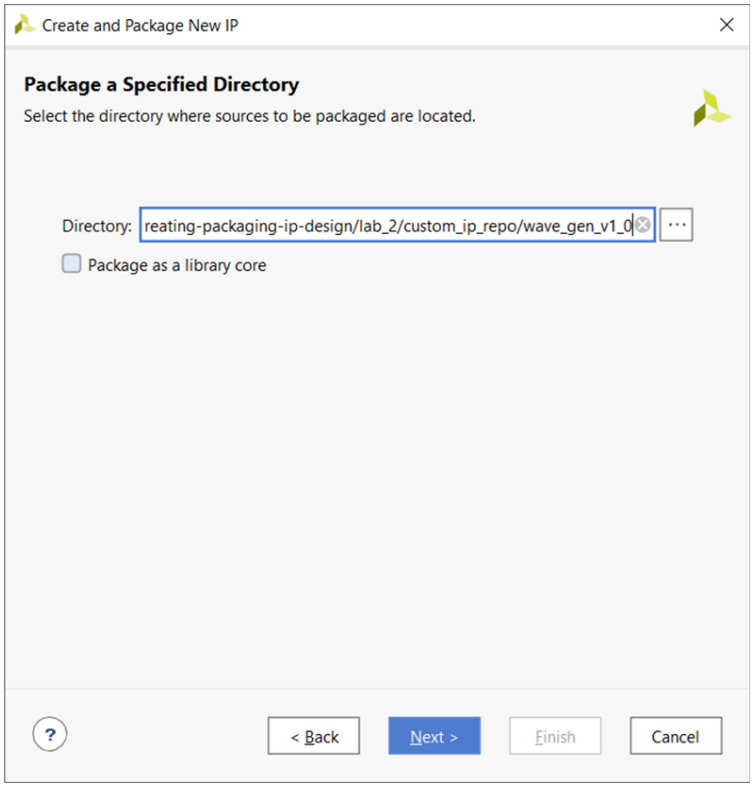
Step 3: Package the IP Directory

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 New 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**.



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



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

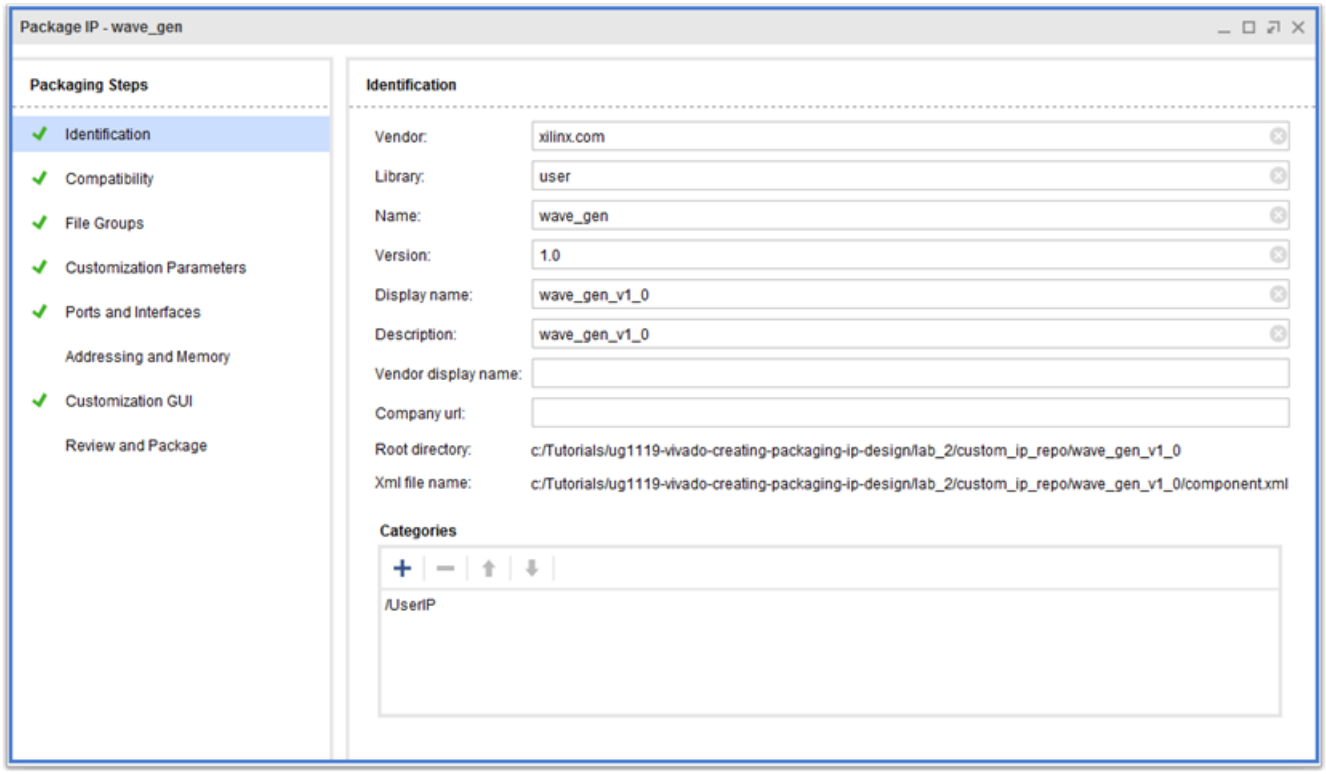
   When packaging a specified directory, the custom IP is packaged through an edit IP project. The default options create an edit IP project in the project temporary location. The edit IP project can be saved for future editing, but a new edit IP project can always be created later.
7. Click **Finish**.

   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 shown below, lists the current IP identification information.

Figure 1: **Package IP**



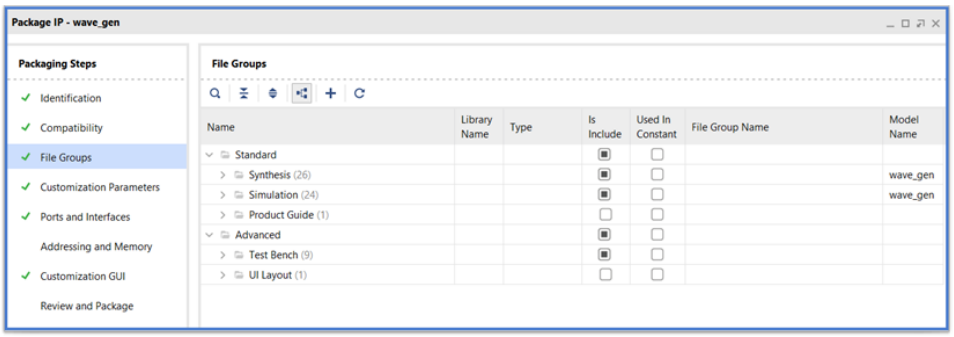
Update the IP Identification

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:** `<company_URL>`
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, set the option to My Company, and click **OK**.
5. Click **OK** to close the Add IP Categories dialog box.

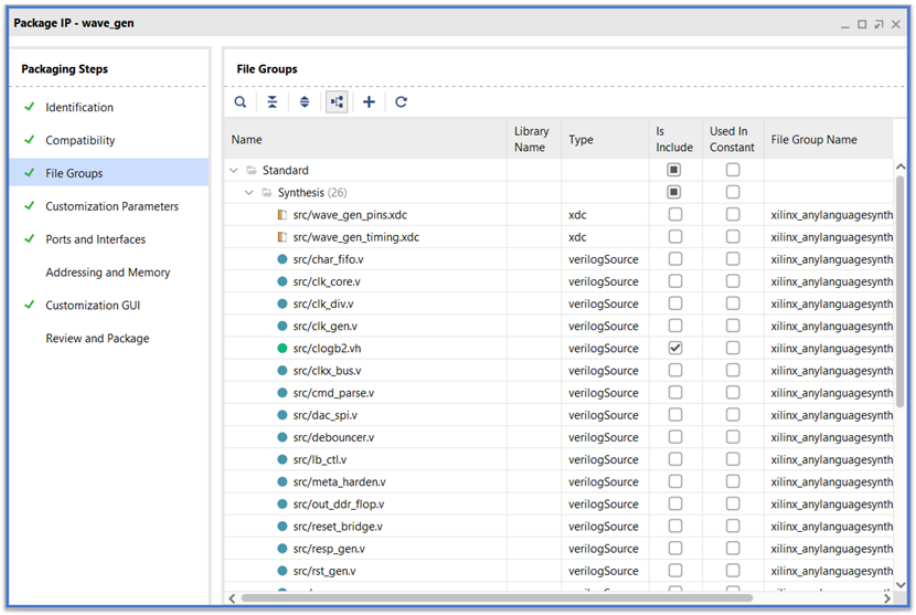
Examine the IP File Groups

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, shown in the following figure, to understand how the IP directory correlates to the 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.



   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.

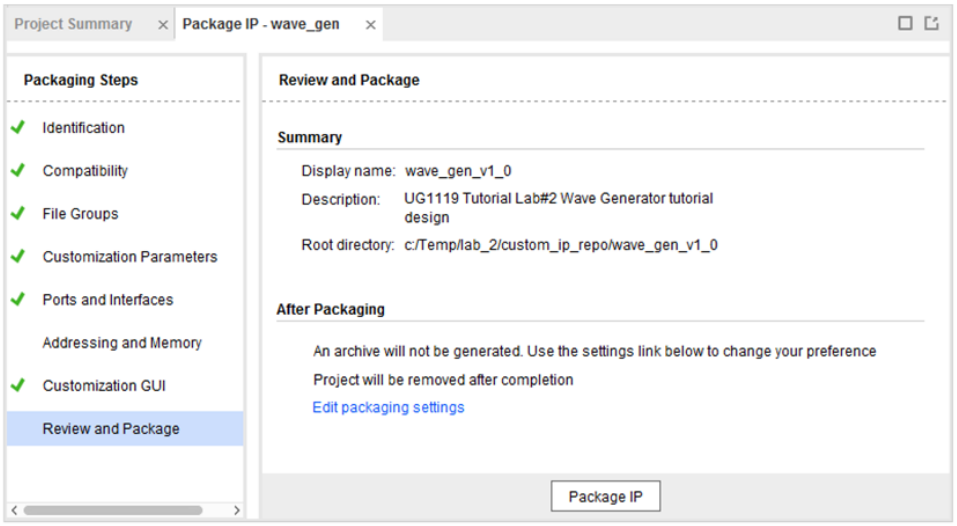
   The Product Guide file group is populated with the PDF from the `/doc` directory and the Testbench file group is populated with the `/tb` directory.

4. Notice that the test benches 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, shown in the following figure, select the **Review and Package** page.



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

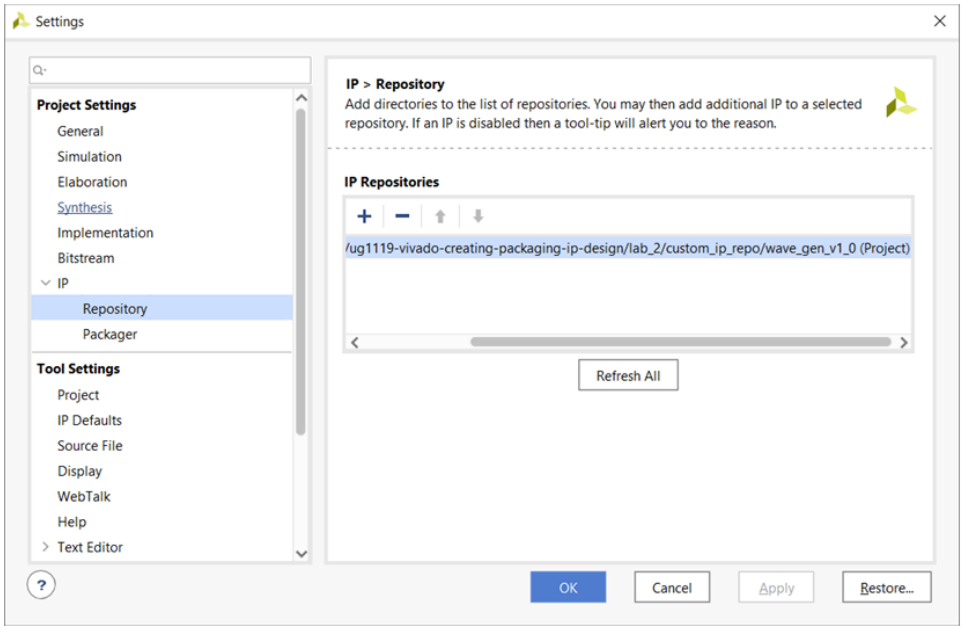
With the new custom IP packaged, the next step is to verify the repository in the IP catalog and validate the generation of the custom IP. You can use the `project_lab2` created in the earlier steps to validate the IP.

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 **Settings**.
2. In the Settings dialog box, expand **IP** and select **Repository**.
3. In the Repository Manager tab, check for 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, as shown below.

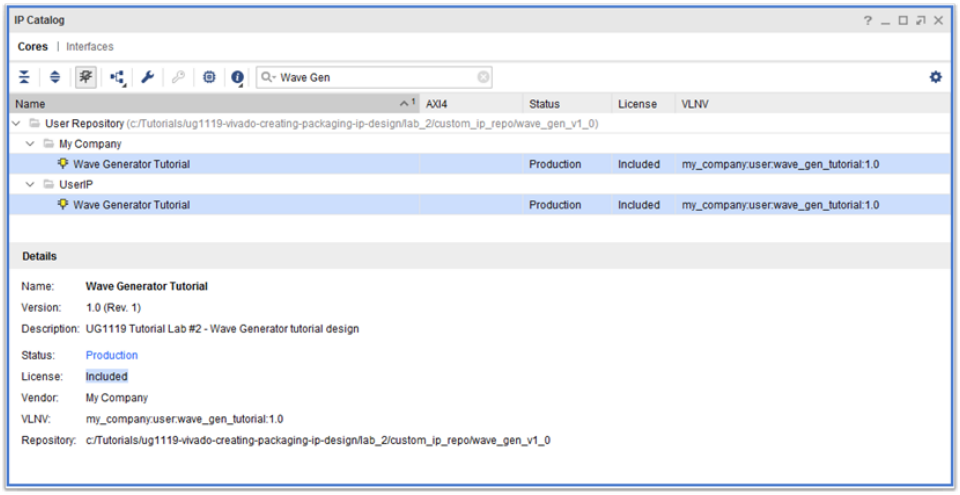


***Note*:** The Vivado tool selects the IP directory location as the repository. You can select the parent repository directory and Vivado traverses the subdirectories for packaged IP.

4. Click **OK**.

Customize the IP

1. Select **Flow Navigator → Project**, and then select **IP Catalog**.
2. In the search field at the top of the IP Catalog, type `Wave Generator`.

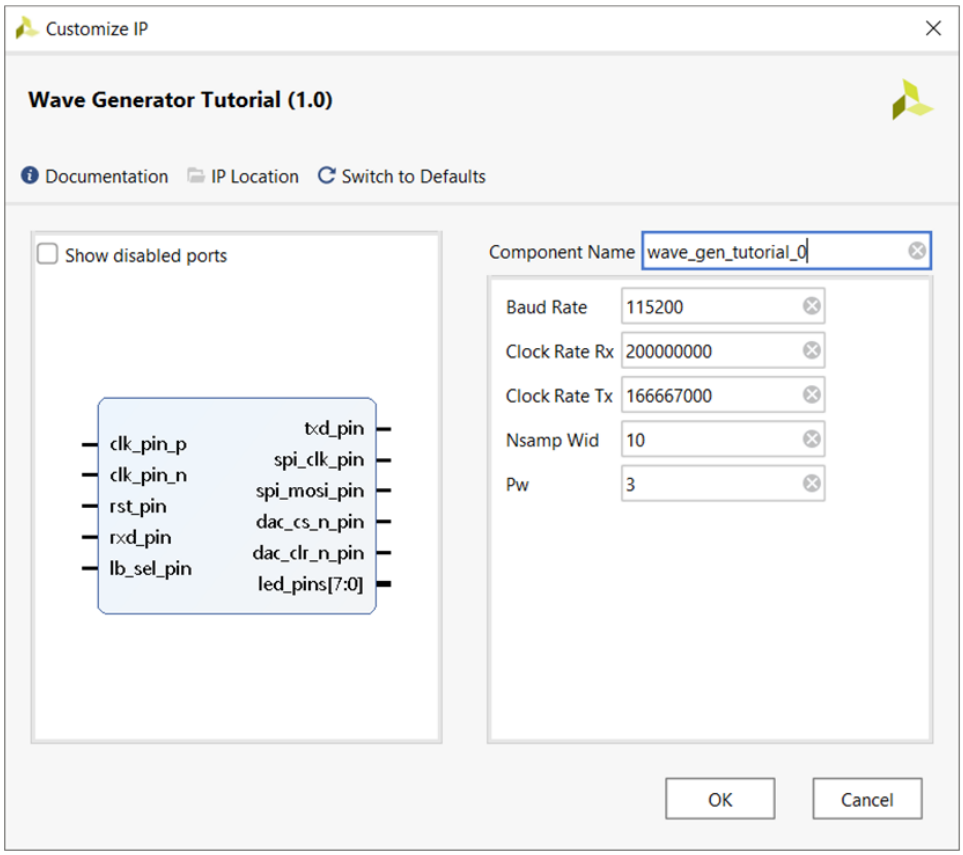


The Wave Generator Tutorial IP shows under the UserIP category, as well as the custom category My Company, that was created during packaging.

Note: This IP catalog view shows when you select Taxonomy and the Repository options for grouping the IP. See the *Vivado Design Suite User Guide: 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.



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 generated successfully.
6. Close the Vivado tool.

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.

Lab 3

Packaging Legacy IP

Introduction

You might need to use a legacy core in Vivado® 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 AXI 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 AXI GPIO IP, package them as library cores (or sub-cores), add the sub-cores to the IP catalog, and then package the AXI 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:

Select **Start → All Programs → Xilinx Design Tools → Vivado 2020.x → Vivado 2020.x**.

Or

Click the Vivado 2020.x desktop icon.

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.

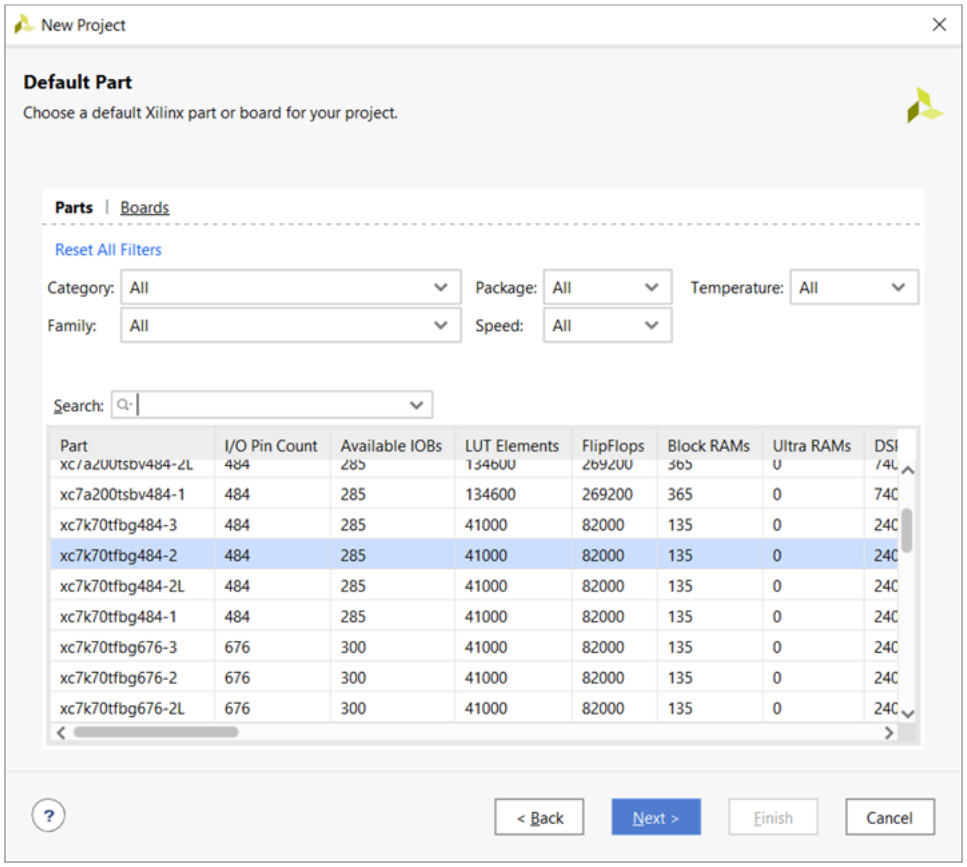
Create a New Project

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.
4. Click **Next**.
5. Select **RTL Project** for Project Type 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.



You 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`, with the default layout.

Step 2: Package a Library Core

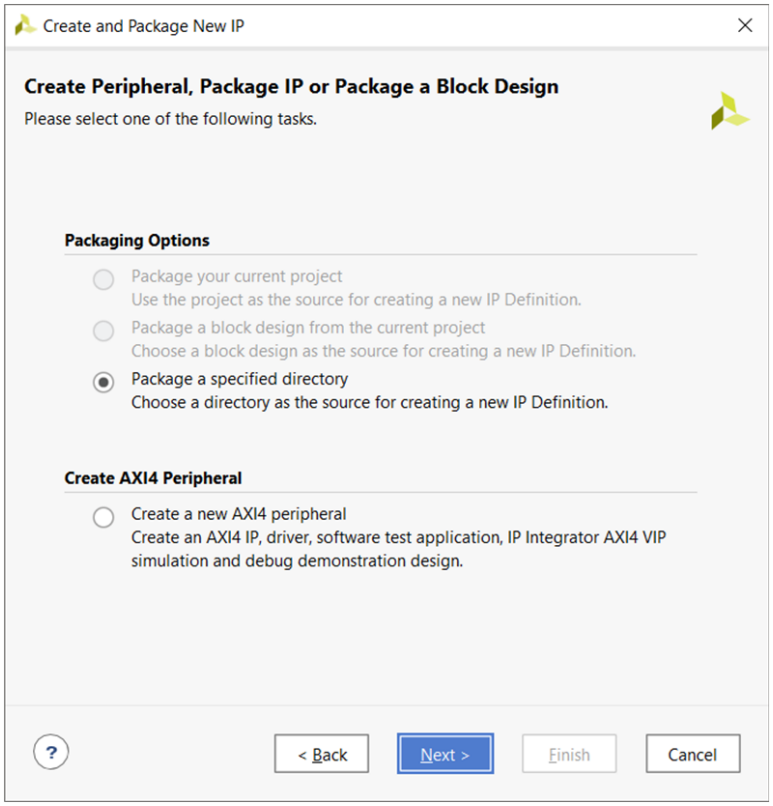
As discussed in the Introduction of this lab, the AXI GPIO Pcore requires several library references (sub-cores) to function.

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

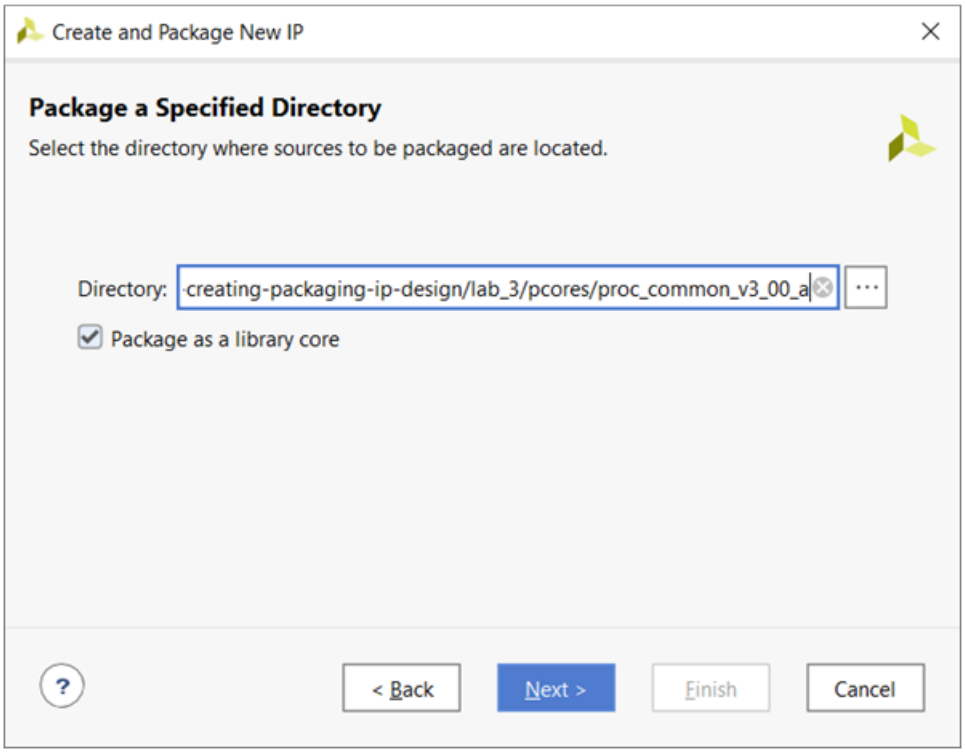
Use the Create and Package Wizard

1. From the Tools menu, select **Create and Package New 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**.



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.



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

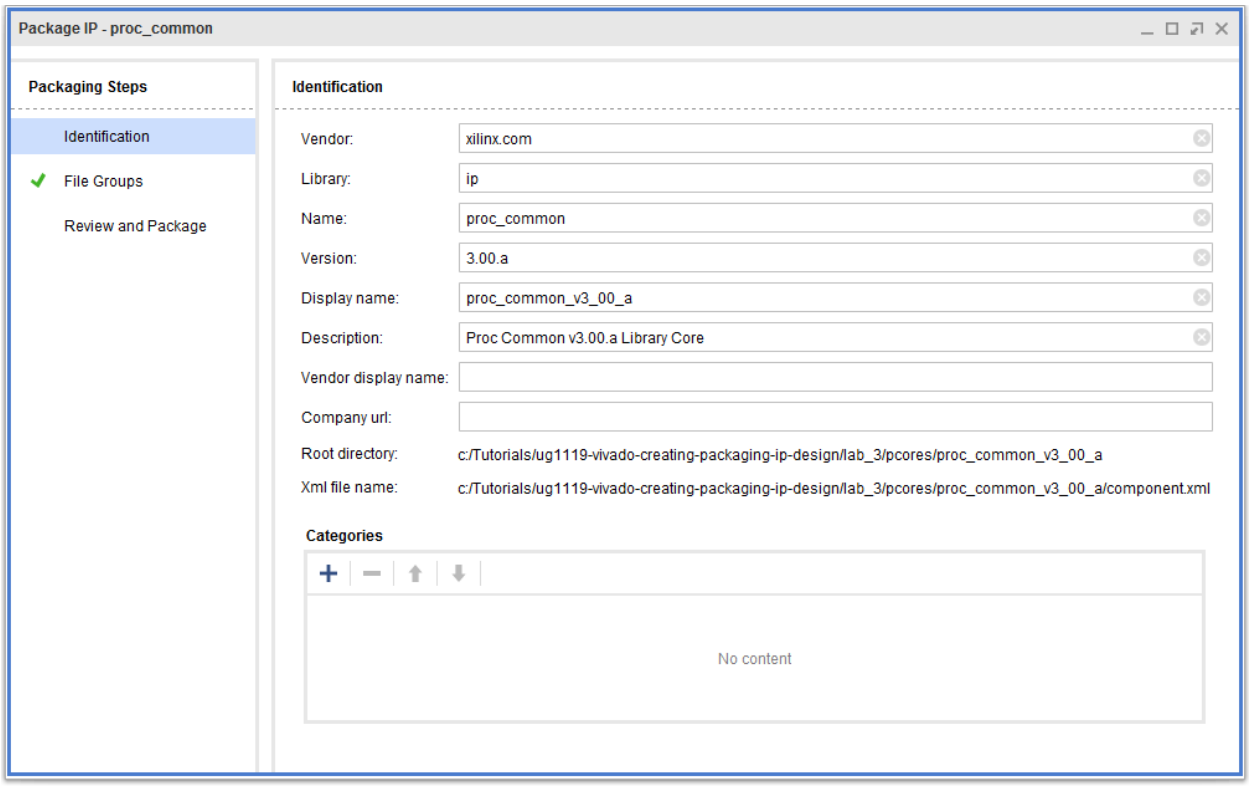
   An edit IP project opens in a new Vivado window with the Package IP window opened. The Package IP window displays the basic IP package in a staging area for editing and repackaging.

Update the IP Information

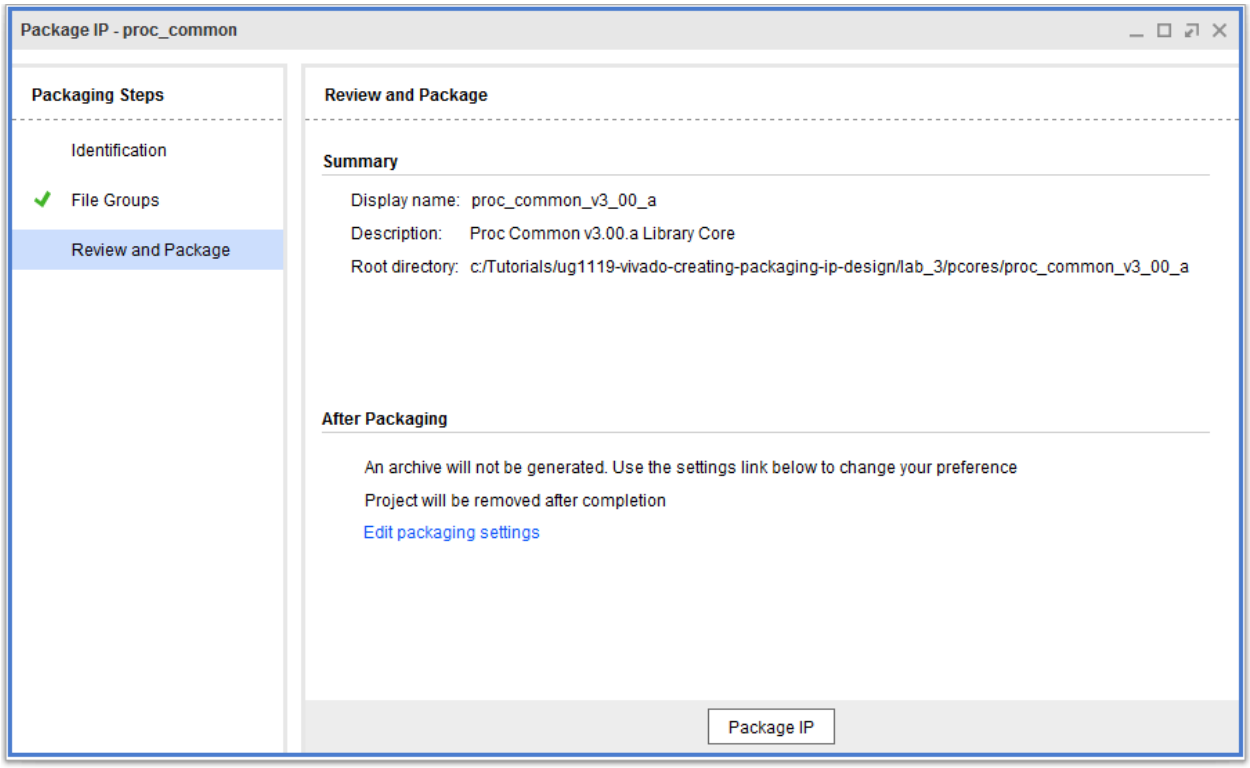
Because you selected the library core option, the Package IP window has as subset of available options for the custom IP, as shown in the following figure.

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`

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



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



4. Click **Package IP**.

   This completes the packaging for the `proc_common_v3_00_a` library core. If prompted, 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

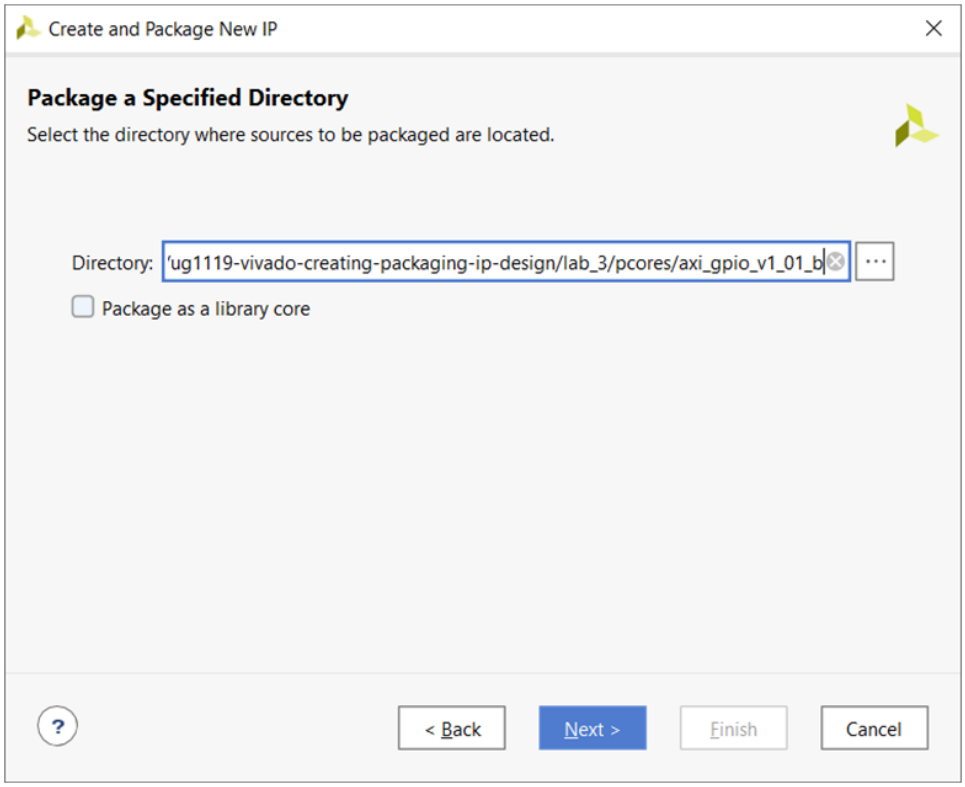
IMPORTANT! *When packaging the additional library cores, the `axi_lite_ipif` and the `interrupt_control_v2_01_a` libraries will display a green checkmark for the File Group page.*

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 New 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`.

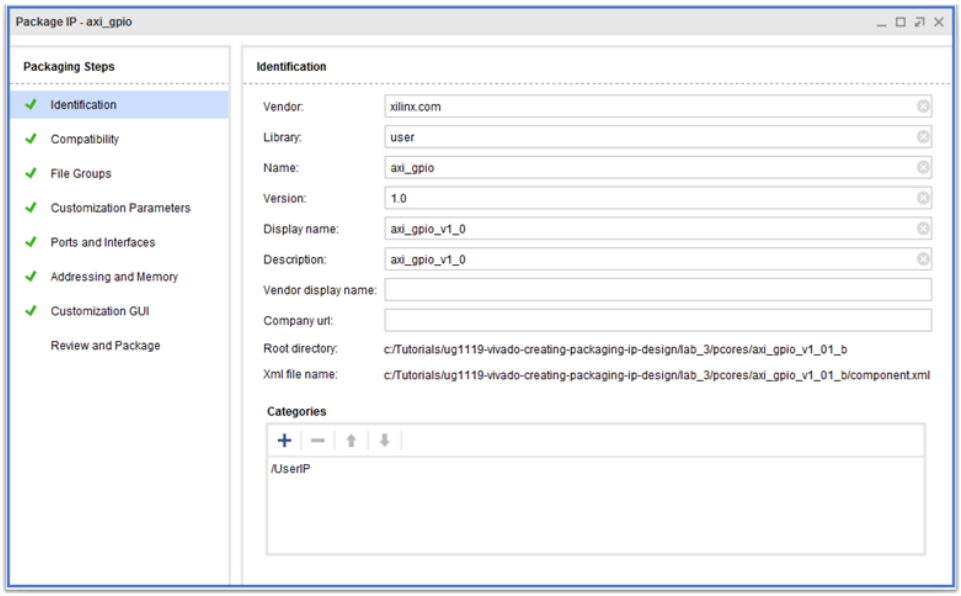


5. Click **Next**.
6. On the Edit in IP Packager Project Name page, leave the default locations, click **Next**, and then click **Finish**.

   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 processor cores (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.

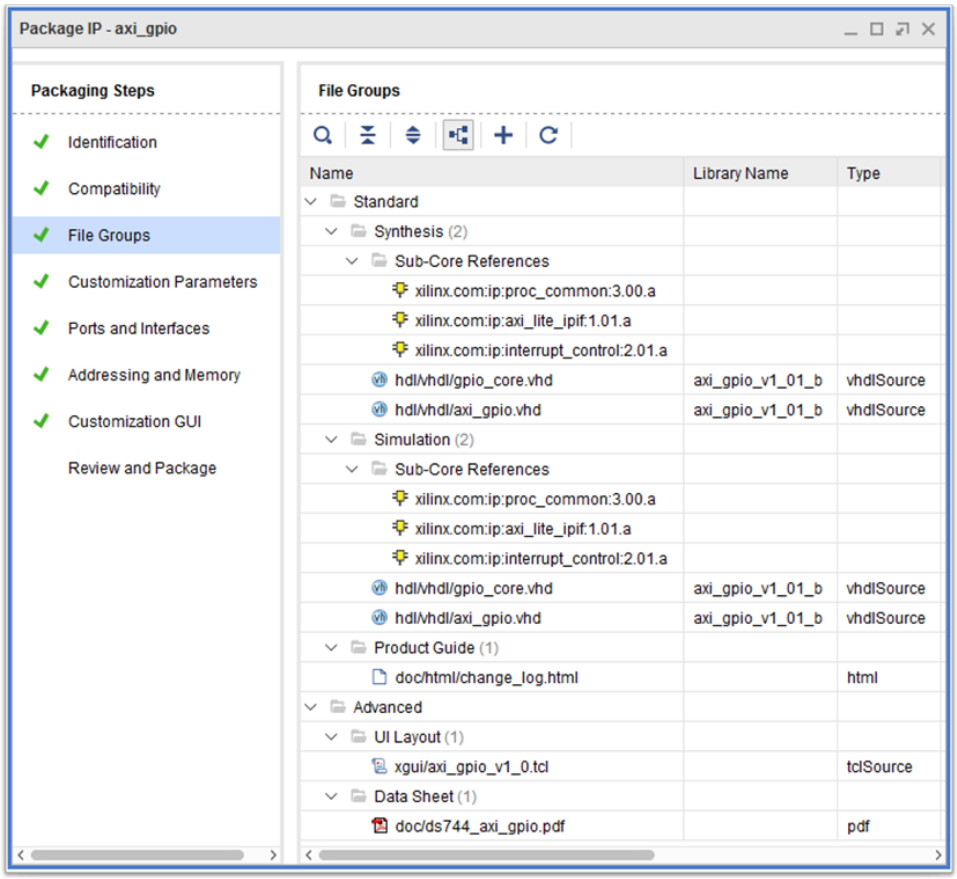
   An edit IP project opens in a new Vivado® packaging window with the Package IP window opened, as shown below.



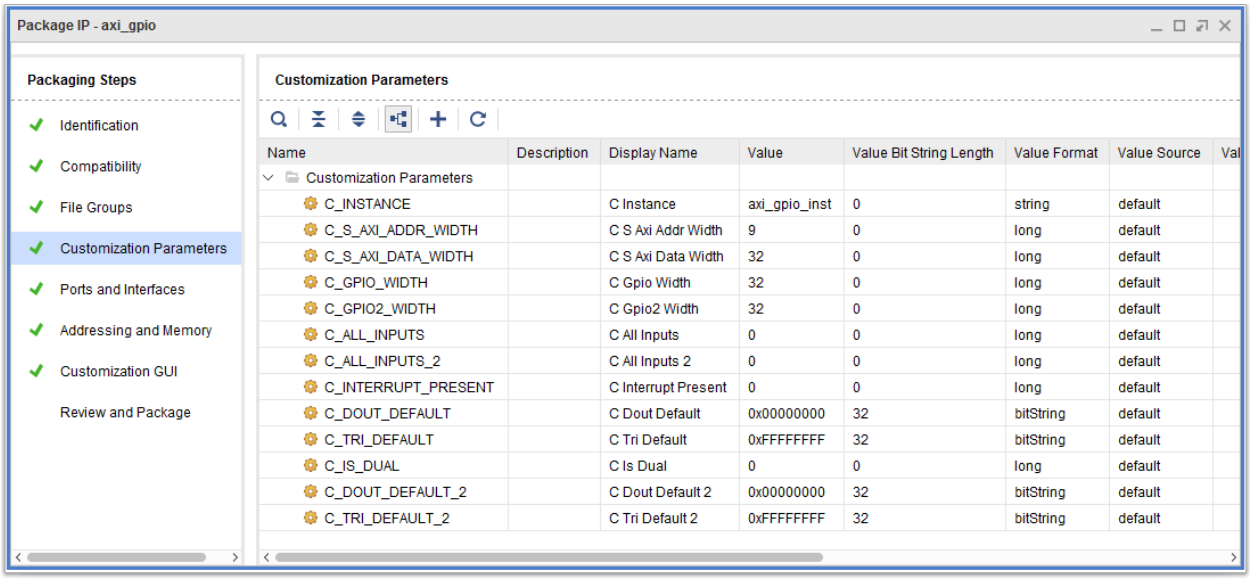
Update the IP Identification

1. In the Package IP window, update the following information:
   - **Vendor:** `my_company`
   - **Display name:** `My AXI GPIO EDK Pcore Tutorial`
   - **Description:** `UG1119 Tutorial Lab #3 - AXI GPIO EDK Pcore`
   - **Vendor display name:** `My Company`
   - **Company url:** `<company_URL>`
2. Click the **File Groups** page to validate that the proper Sub-Core References (Library Cores) are added to the Package IP window.

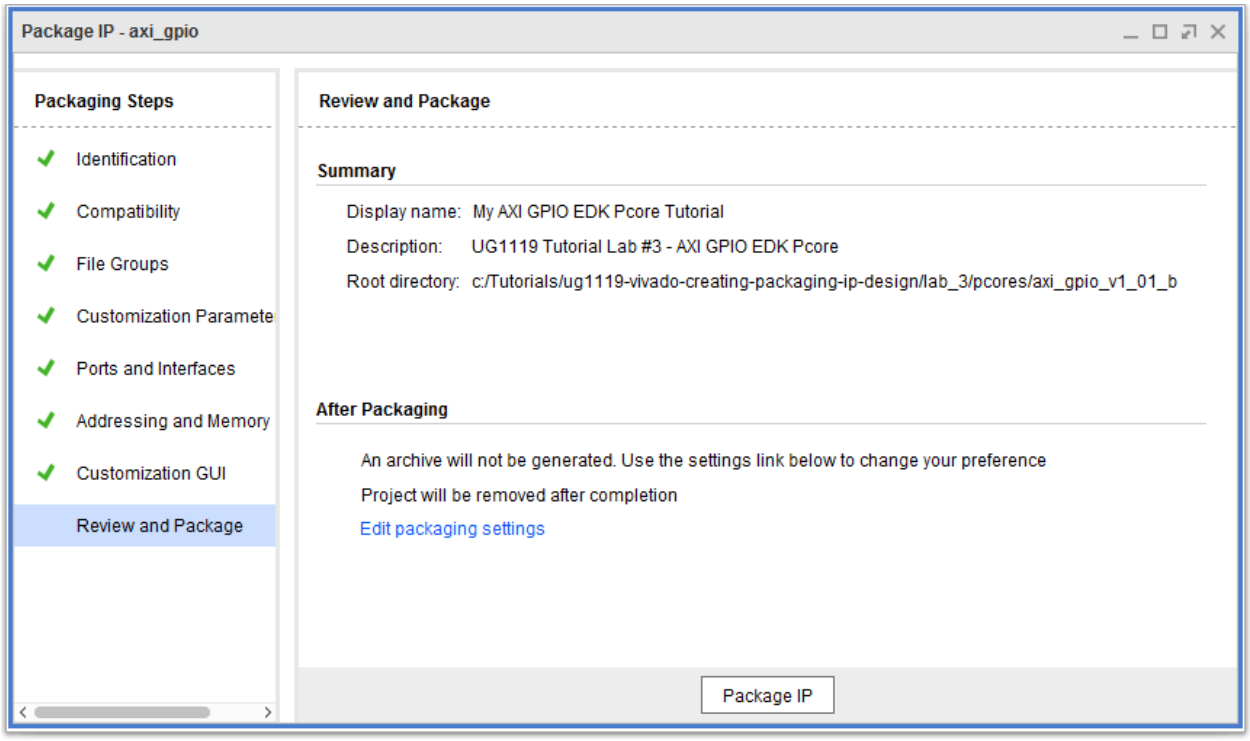
   In this case, the Interrupt Controller, the AXI4-Lite IPIG and the Proc Common display in the `/Sub-Core References` directories for Synthesis and Simulation.



3. Click the **Customization Parameters**, as shown below, to explore the parameters defined for the custom IP.



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



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`.

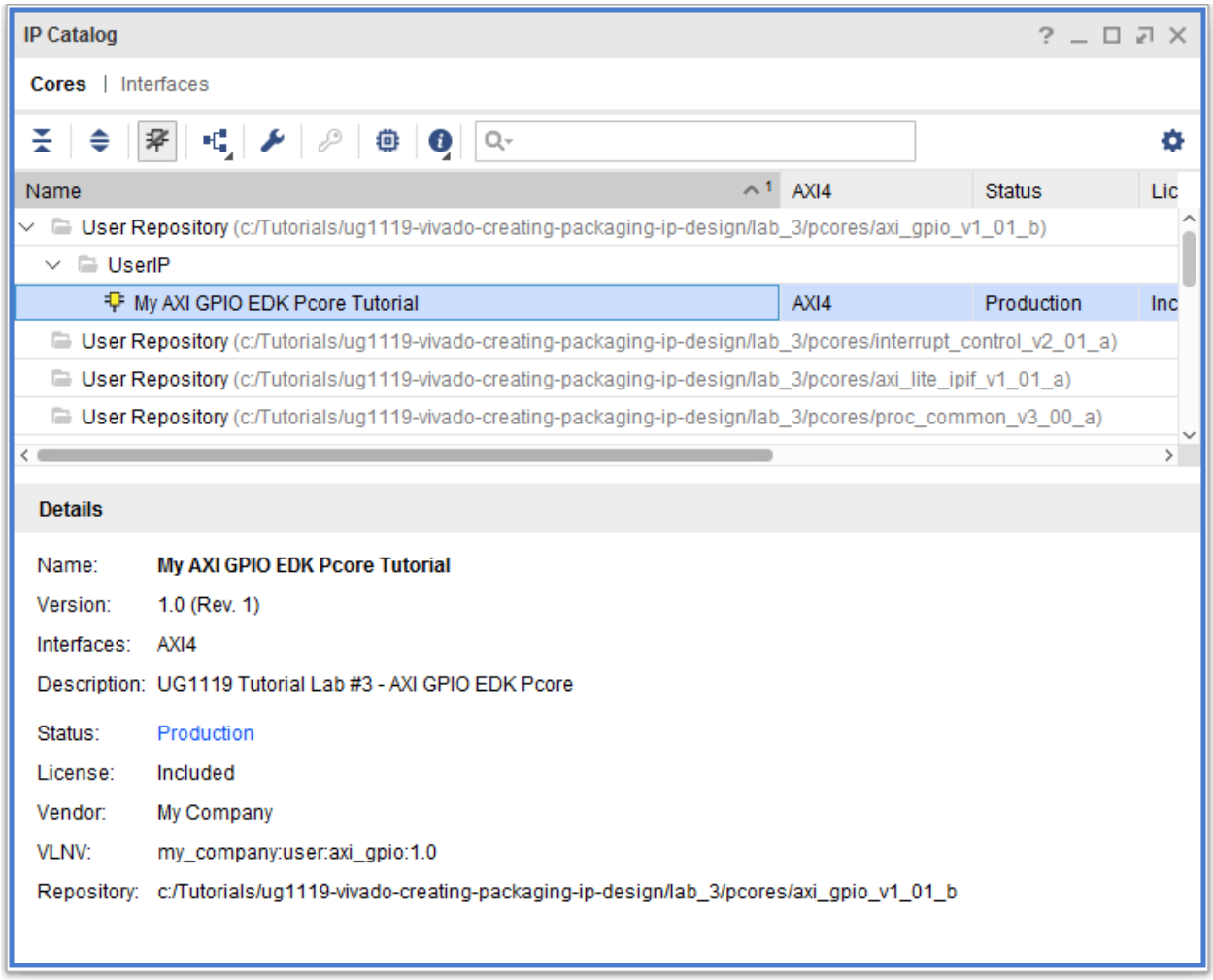
Step 4: Validate the New Custom IP

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, shown in the figure.



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 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 AXI GPIO Pcore.
- Validated the generation of the AXI GPIO Pcore custom IP.

Lab 4

Packaging IP in a Revision Source (Trunk)

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.

   - `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.

   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:

  Select **Start → All Programs → Xilinx Design Tools → Vivado 2020.x → Vivado 2020.x**.

  Or

  Click the Vivado 2020.x desktop icon.

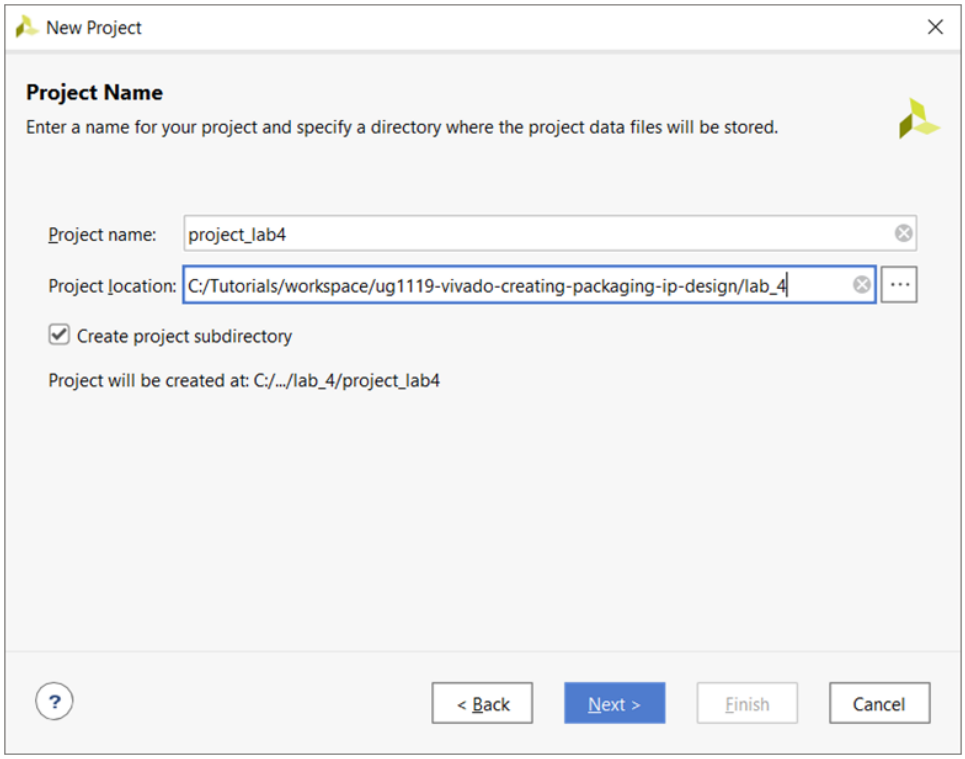
The Vivado IDE Getting Started page displays with links to open or create projects, and to view documentation. For either Windows or Linux, continue the lab from this point.

Create a New Project

1. From the Vivado IDE Getting Started page, select **Create 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`



4. Click **Next**.
5. Select RTL Project as the **Project Type** and select **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™ architecture 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.

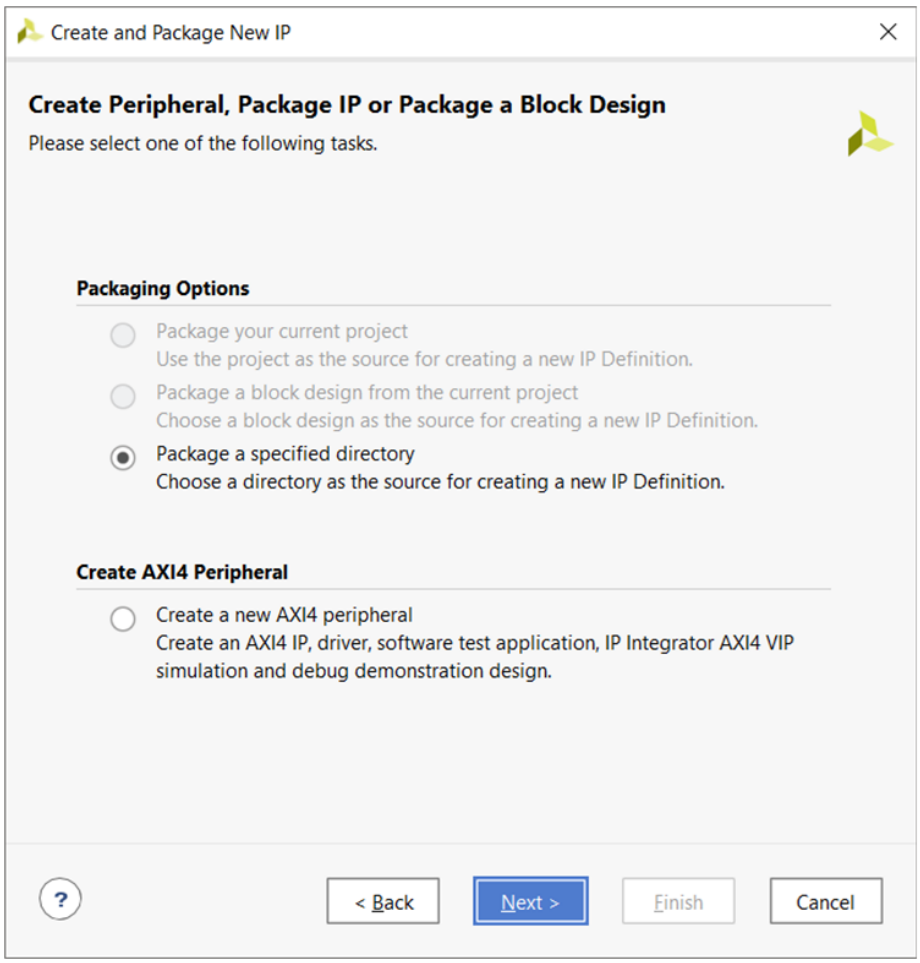
   The Vivado IDE `opens project_lab4`, with the default layout.

Step 3: Package the Library Core

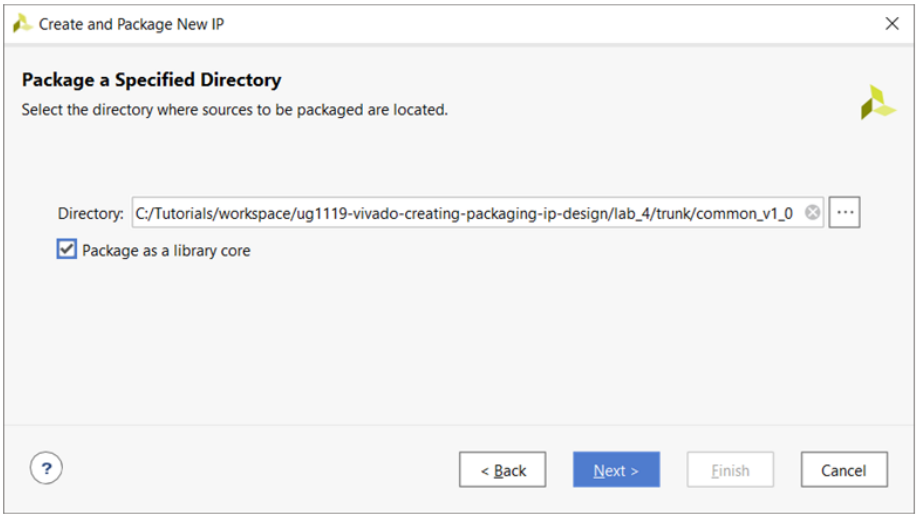
After creating the new empty project, the next step is to create and package the common IP directory. The order of the packaging the IP directories are important because they need to be packaged in the order of dependency. All of the child IP must be packaged prior to packaging the parent IP. You will 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 New 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.



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.



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.

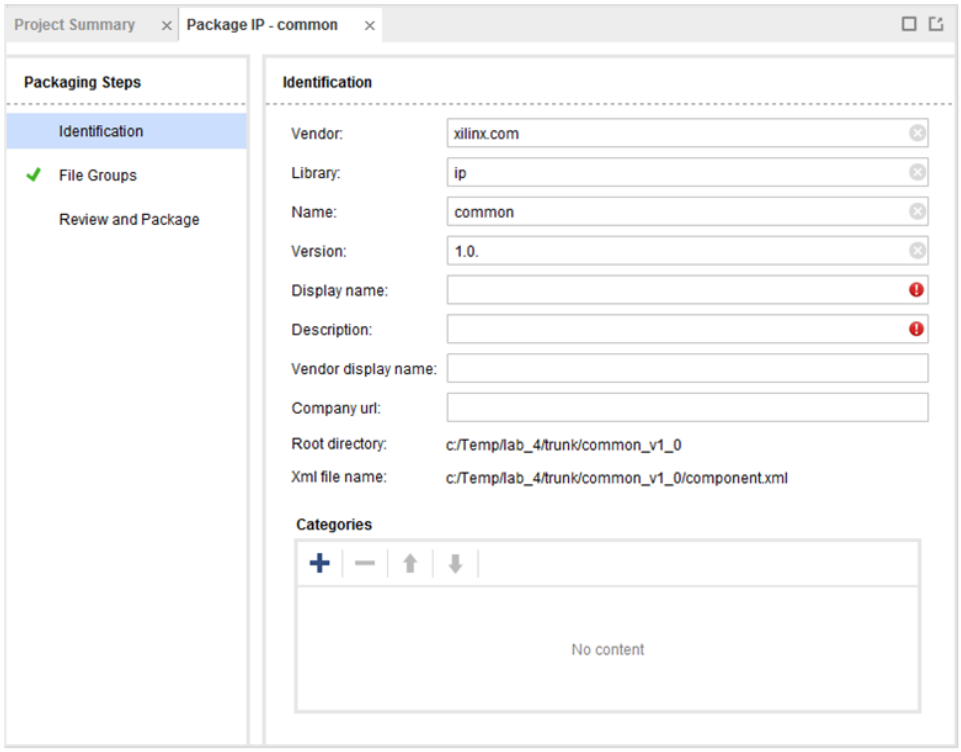
However, any Custom IP can be a child to another IP. If your Custom IP is not a library core, the process for referencing a child IP is the same. This option is just used to mitigate confusion of which IP should be used and hidden in the Vivado IP Catalog.

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 opens 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.



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:** `<company_URL>`
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

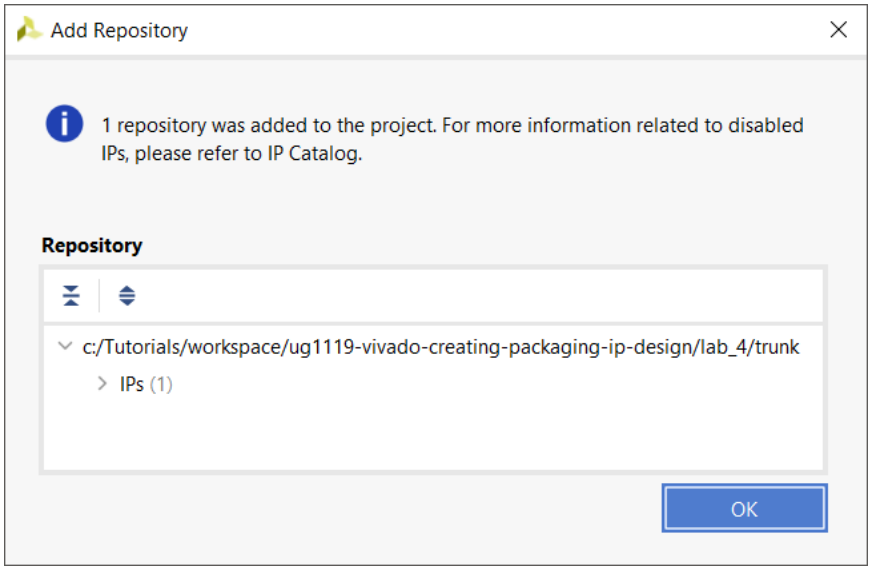
Using the same project previously created, `project_lab4`, you will create and package the `myip_v1_0` IP directory. Because the common IP directory has already been packaged, all the required dependencies are available to package the parent IP.

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. Select **Flow Navigator → Project Manager → Settings → IP**.
2. Expand **IP** and select the **Repository**. In the view, the repository in which the previously packaged IP should automatically show up, if not click the **Add Repository** button.
3. In the IP Repositories dialog box, select the path `<Extract_Dir>/lab_4/trunk` and press **Select** to add the repository.

   The Add Repository dialog box opens to display that the trunk repository was added to the project and one IP was found, as shown in the following figure.



4. Click **OK**.

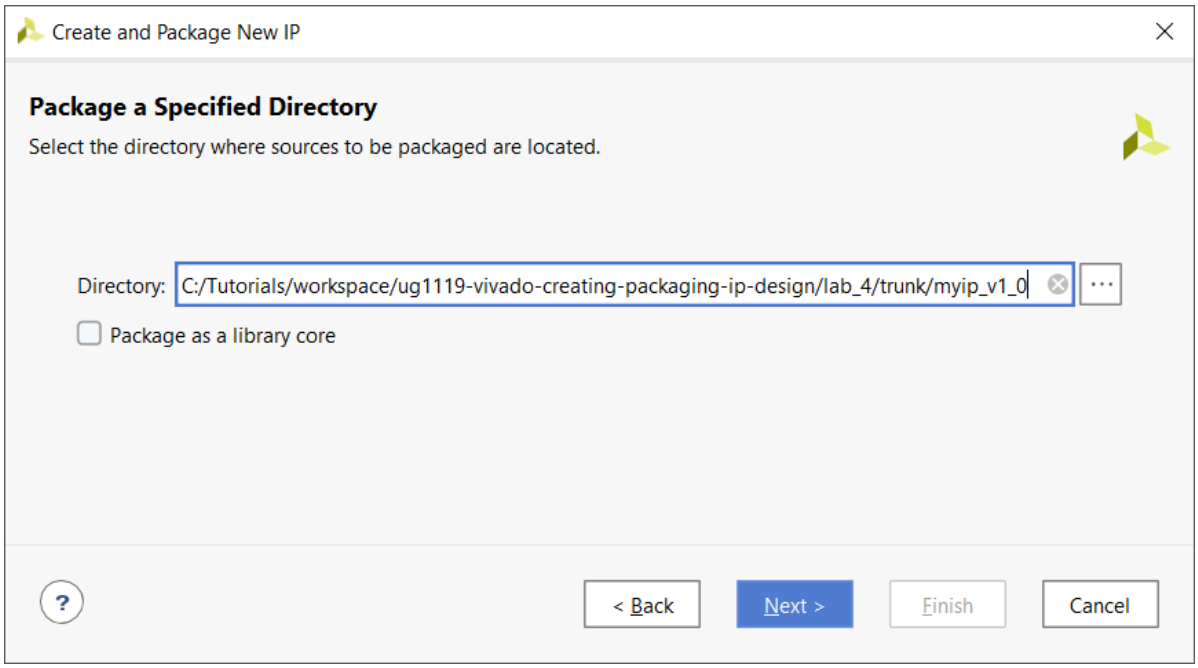
   The Repository Manager is now populated with the selected IP repository, in addition to the `common_v1_0` repository.

5. To remove the `common_v1_0` repository, select it and then select the **Remove** button .
6. Click **OK** to close the IP Setting dialog box.

Use the Create and Package IP Wizard

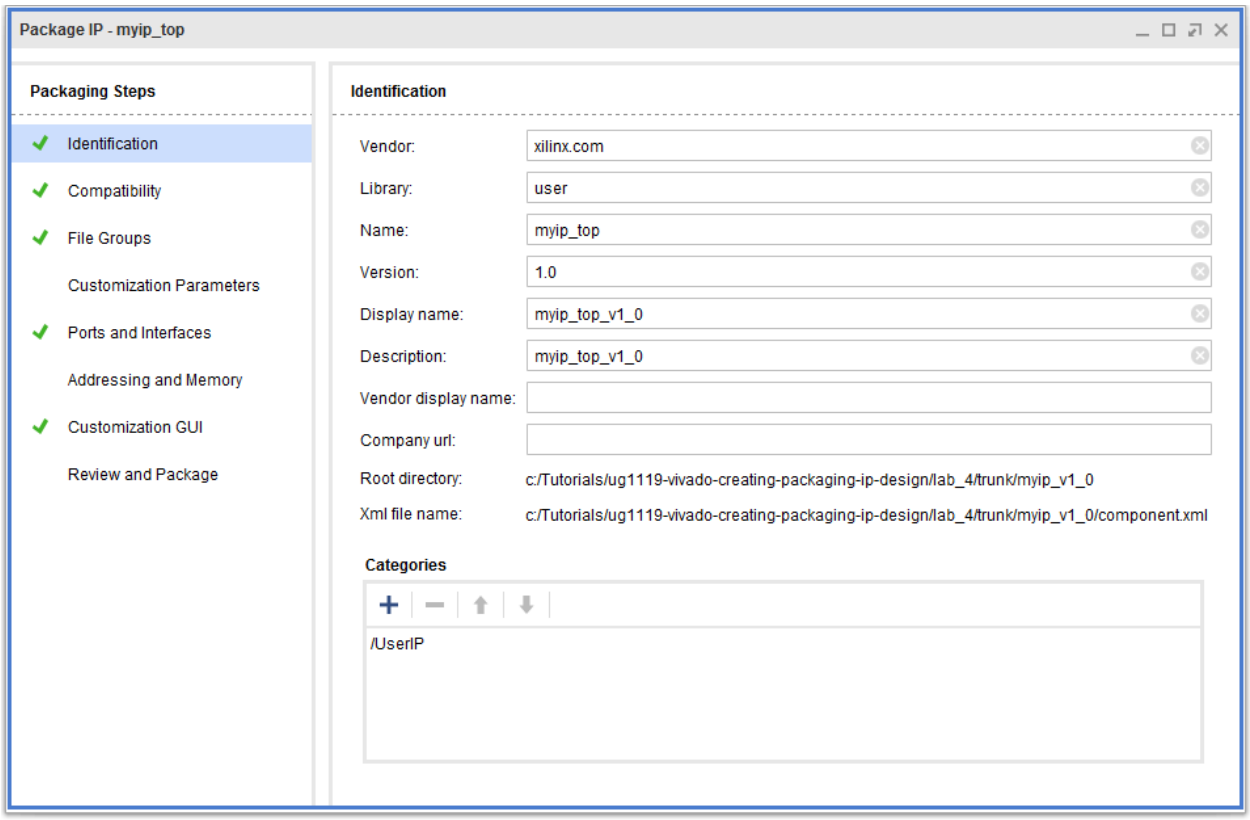
1. From the Tools menu, select **Create and Package New IP** to open the Create and Package IP Wizard.
2. Click **Next** at the Welcome screen for the Create and Package New IP dialog box.
3. In the Create Peripheral, Package IP, or Package Block Design dialog box, select **Package a specified directory**.

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 select* the **Package as a library core** option.

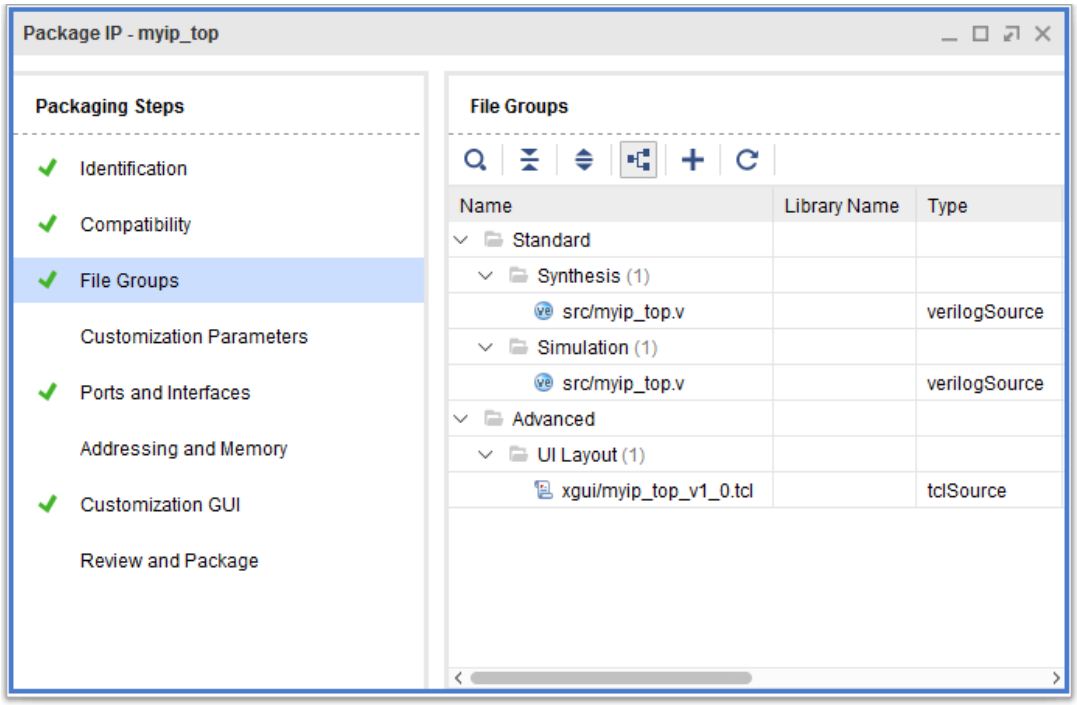


6. Click **Next**.
7. On the Edit in IP Packager Project Name window, leave the default locations, and click **Next**.
8. Click **Finish**.

   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.

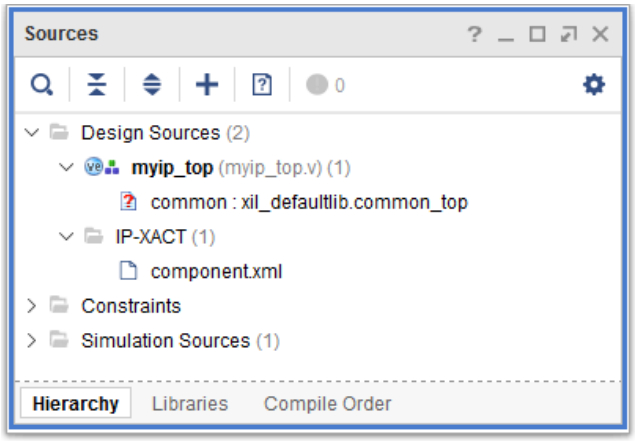


9. Update the IP Information and Contents In the Package IP window, update the following information on the Identification page:
   - **Vendor:** `my_company`
   - **Display Name:** `My IP`
   - **Description:** `UG1119 Tutorial Lab #4 - My IP`
   - **Vendor Display Name:** `My Company`
   - **Company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



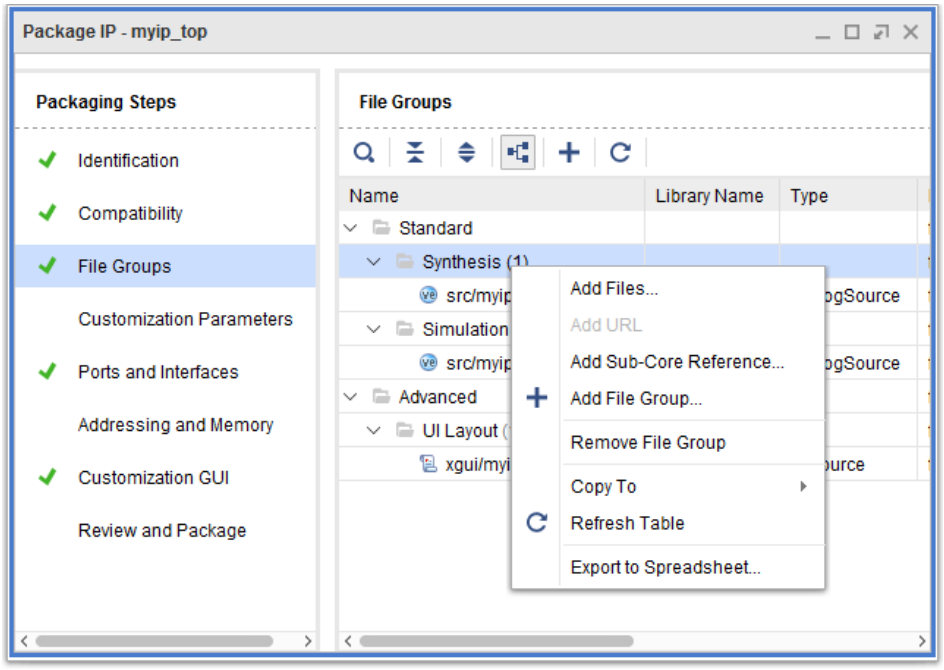
The packaged IP only contains the top-level source file, `myip_top`, as this was the only file in the selected IP directory `<Extract_Dir>/lab_4/trunk/myip_v1_0`. This file instantiates the IP `common_v1_0`.

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

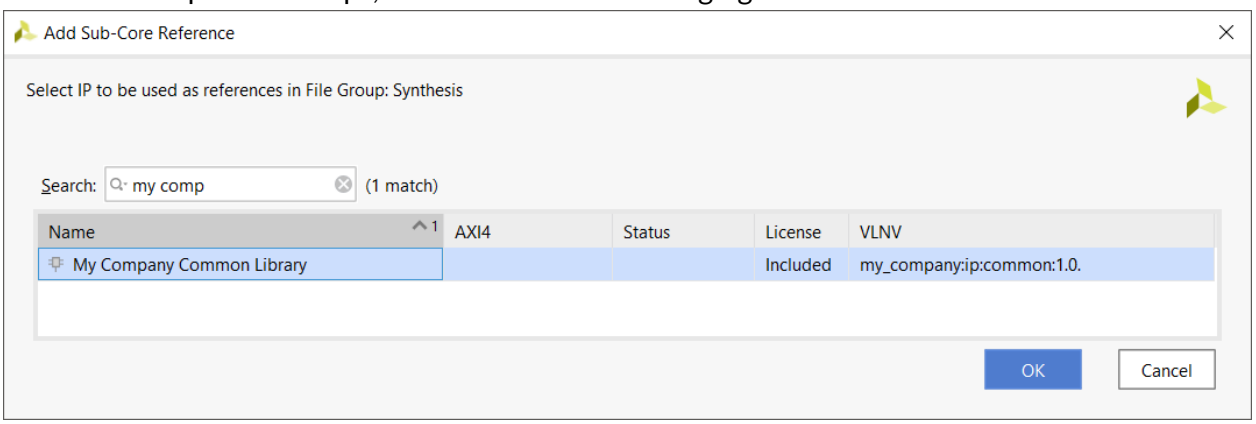


This is expected behavior, because you add the missing IP source files through the Package IP window.

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



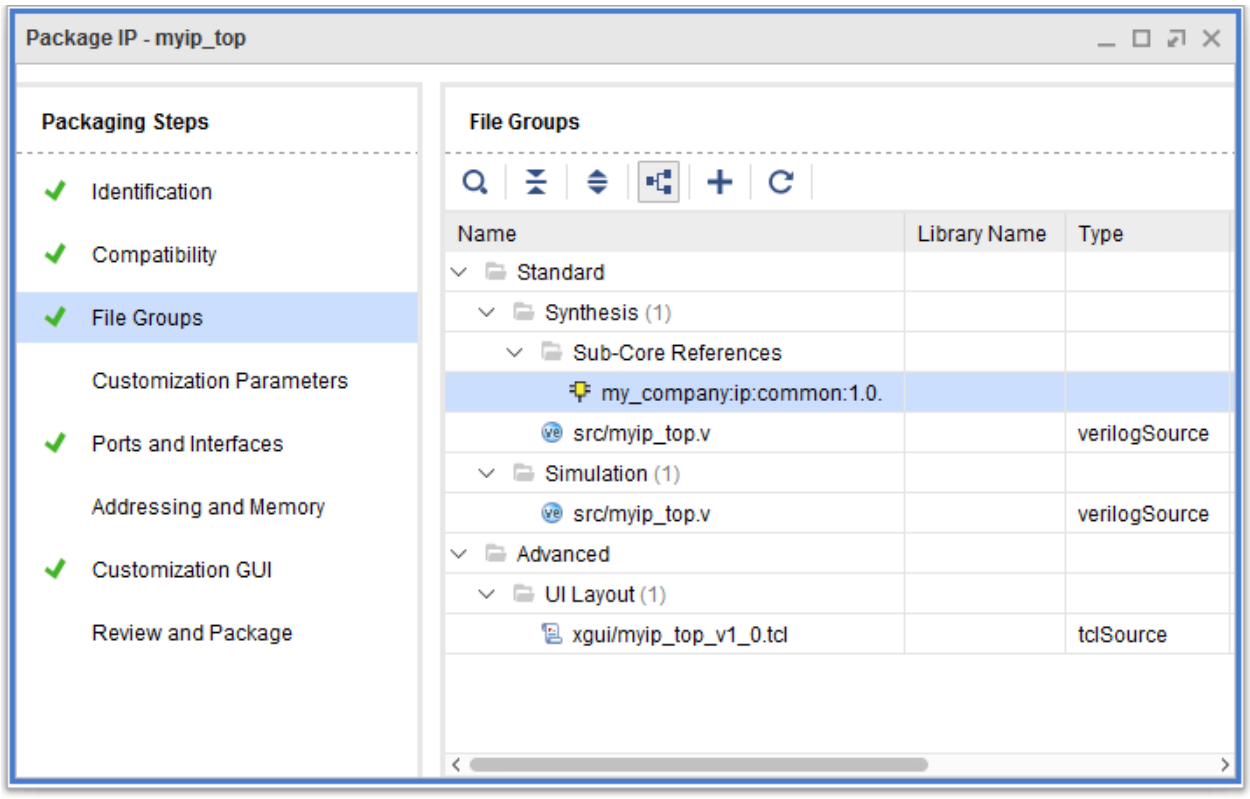
13. In the Add Sub-Core Reference dialog box, select the **My Company Common Library** that you created in the previous steps, as shown in the following figure.



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.

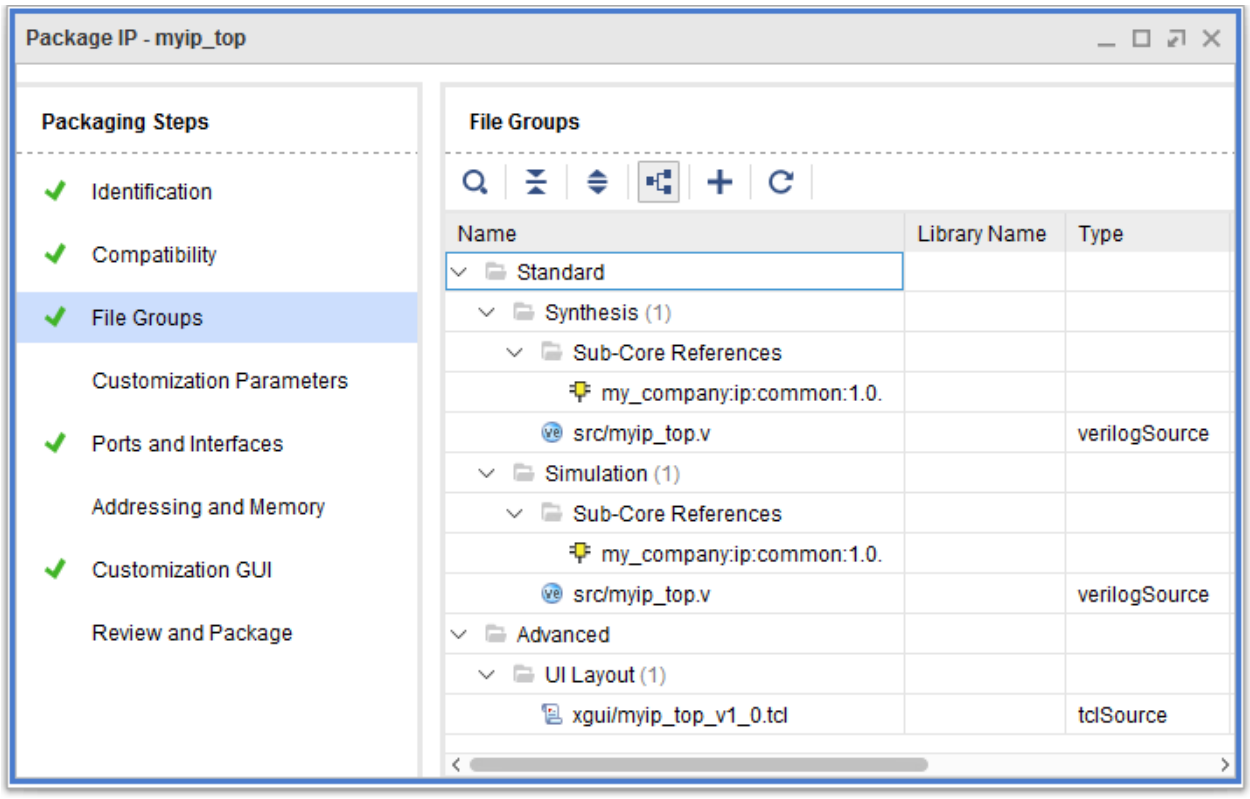
    Adding an IP as a Sub-Core Reference informs the Vivado IDE to copy the files associated IP to the parent IP during generation; therefore, when `myip_v1_0` is generated, the `common_v1_0` files are copied to the location with the rest of the generated output products. This mechanism allows users to systematically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



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 **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.



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`.

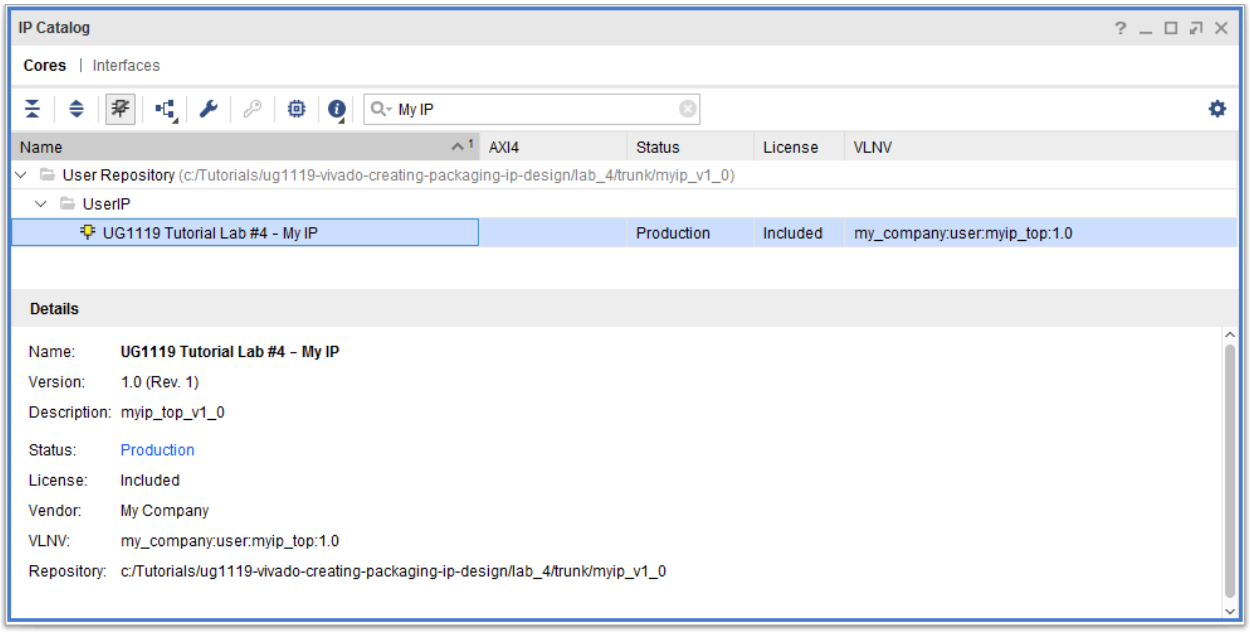
    Note: Adding a sub-core reference in the Package IP window does affect the state of the edit IP project. The Hierarchy Sources window continues to display the missing modules located within the sub-core reference. This information only exists within the Package IP window and `component.xml`. If you want 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.



3. Right-click the **My IP core** and select **Customize IP**.
4. In the Customization IP dialog box, 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 Lab 1: Packaging a Project.
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 the Vivado tool.

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.

XILINX

Appendix A

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

Documentation Navigator and Design Hubs

Xilinx® Documentation Navigator (DocNav) provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open DocNav:

- From the Vivado® IDE, select **Help → Documentation and Tutorials**.
- On Windows, select **Start → All Programs → Xilinx Design Tools → DocNav**.
- At the Linux command prompt, enter `docnav`.

Xilinx Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In DocNav, click the **Design Hubs View** tab.
- On the Xilinx website, see the Design Hubs page.

Note: For more information on DocNav, see the Documentation Navigator page on the Xilinx website.

References

Xilinx Web Resources

1. Vivado IP Versioning
2. Xilinx Answer Record 68071
3. Vivado Design Suite Documentation

Vivado Design Suite Documentation

These documents provide supplemental material useful with this guide:

1. *Vivado Design Suite Tcl Command Reference Guide* (UG835)
2. *Vivado Design Suite Tutorial: Design Flows Overview* (UG888)
3. *Vivado Design Suite User Guide: Design Flows Overview* (UG892)
4. *Vivado Design Suite User Guide: Using the Vivado IDE* (UG893)
5. *Vivado Design Suite User Guide: Using Tcl Scripting* (UG894)
6. *Vivado Design Suite User Guide: System-Level Design Entry* (UG895)
7. *Vivado Design Suite User Guide: Designing with IP* (UG896)
8. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
9. *Vivado Design Suite User Guide: Synthesis* (UG901)
10. *Vivado Design Suite User Guide: Using Constraints* (UG903)
11. *Vivado Design Suite User Guide: Implementation* (UG904)
12. *Vivado Design Suite User Guide: Hierarchical Design* (UG905)
13. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
14. *Vivado Design Suite Properties Reference Guide* (UG912)
15. *Vivado Design Suite Tutorial: Programming and Debugging* (UG936)
16. *Vivado Design Suite Tutorial: Logic Simulation* (UG937)
17. *Vivado Design Suite Tutorial: Designing with IP* (UG939)
18. *UltraFast Design Methodology Guide for Xilinx FPGAs and SoCs* (UG949)
19. *Vivado Design Suite User Guide: Release Notes, Installation, and Licensing* (UG973)
20. *UltraScale Architecture Libraries Guide* (UG974)
21. *Vivado Design Suite User Guide: Designing IP Subsystems Using IP Integrator* (UG994)
22. *Vivado Design Suite: AXI Reference Guide* (UG1037)

23. *Vivado Design Suite User Guide: Creating and Packaging Custom IP* (UG1118)
24. *Vivado Design Suite Tutorial: Creating, Packaging Custom IP* (UG1119)

Xilinx IP Documentation

1. *Integrated Bit Error Ratio Tester 7 Series GTX Transceivers LogiCORE IP Product Guide* (PG132)
2. *Integrated Bit Error Ratio Tester 7 Series GTP Transceivers LogiCORE IP Product Guide* (PG133)
3. *Integrated Bit Error Ratio Tester 7 Series GTH Transceivers LogiCORE IP Product Guide* (PG152)
4. *Virtual Input/Output LogiCORE IP Product Guide* (PG159)
5. *Integrated Logic Analyzer LogiCORE IP Product Guide* (PG172)
6. *AXI Verification IP LogiCORE IP Product Guide* (PG267)
7. *AXI4-Stream Verification IP LogiCORE IP Product Guide* (PG277)
8. *AXI4-Stream Verification IP LogiCORE IP Product Guide* (PG277)
9. *Zynq-7000 SoC Verification IP Data Sheet* (DS941)

Third-Party Documentation

1. IEEE 1735-2014 - IEEE Recommended Practice for Encryption and Management of Electronic Design Intellectual Property (IP)
2. IEEE Standard for IP-XACT, Standard Structure for Packaging, Integrating, and Reusing IP within Tool Flows

Please Read: Important Legal Notices

Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at https://www.xilinx.com/legal.htm#tos; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at https://www.xilinx.com/legal.htm#tos.

AUTOMOTIVE APPLICATIONS DISCLAIMER

AUTOMOTIVE PRODUCTS (IDENTIFIED AS "XA" IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE ("SAFETY APPLICATION") UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD ("SAFETY DESIGN"). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.

Copyright