# **Vivado Design Suite**

# Creating, Packaging Custom IP Tutorial

UG1119 (v2019.2) March 2, 2020





# **Revision History**

The following table shows the revision history for this document.

Section	Revision Summary
03/02/2020 V	ersion 2019.2
General updates	Updates for 2019.2 release and editorial edits only.
06/12/2019 V	ersion 2019.1
General updates	Updates for 2019.1 release and editorial edits only.





# **Table of Contents**

Revision History	
Introduction to Creating and Packaging Custom IP	
Introduction	
Software Requirements	5
Tutorial Design Description	5
Locating Tutorial Design Files	5
Lab 1: Packaging a Project	
Introduction	
Step 1: Open the Vivado Project	
Step 2: Preparing Design Constraints	7
Step 3: Package the IP	14
Step 4: Modify the IP Definition	
Step 5: Add a Product Guide to the IP	
Step 6: Review and Package the IP	
Step 7: Validate the New IP	
Conclusion	
Lab 2: Packaging a Specified Directory	
Introduction	
Step 1: Examine the IP Directory	
Step 2: Create a New Vivado Project	
Step 3: Package the IP Directory	
Step 4: Examine and Update the Packaged IP	
Step 5: Validate the Custom IP	
Conclusion	
Lab 3: Packaging Legacy IP	
Introduction	
Step 1: Create a New Vivado Project	

www.xilinx.com





Step 2: Package a Library Core	
Step 3: Package the GPIO IP	45
Step 4: Validate the New Custom IP	
Conclusion	
Lab 4: Packaging IP in a Revision Source (Trunk)	51
Introduction	51
Step 1: Examine the Repository Trunk Directory	51
Step 2: Create a New Vivado Project	
Step 3: Package the Library Core	
Step 4: Package the IP	
Step 5: Validate the IP	63
Conclusion	64
Legal Notices	65
Please Read: Important Legal Notices	65





## 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<sup>®</sup> 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<sup>®</sup> IP, third-party IP, and end-user designs targeted for reuse as IP into a single environment.

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.

### **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 the design files from the Reference Design Files on the Xilinx website.

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 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 2019.x> Vivado 2019.x

Or

• Click the Vivado 2019.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:

Eile Edit Flow Iools Rep	grts Window Layout	View Help	Quick Act	201 -																R	Ready
B + + B B X I	. # Φ Σ × I	5.00																	III Defaul	t Layout	Y
Flow Navigator = 0 ?	PROJECT MANAGER - mj	_simple_uart																			?
PROJECT MANAGER	Sources	? _	n c ×	Proj	ject Sun	nmary														7 [	o s ×
Settings	Q 2 0 +	0 • r	•	Ov	rview	Dash	board														
Add Sources Language Templates				Se	ettings	Edit															î
P IP Catalog	> = Constraints (1) > = Simulation Sources				roject na roject lo			my_simple_uar C:/Tutorials/wo	t orkspace/ug1119-	wvado-	creat	ing-packag	ing-ip-di	rsign/lai	b_1						
<ul> <li>IP INTEGRATOR</li> </ul>	>  Ubility Sources				roduct fa			Kintex-7							0000						- 1
Create Block Design					roject pa			xc7k325tffg900	5-2												
Open Block Design	Hierarchy Libraries	Compile Order			op modi arget lar			uart_top VHDL													
Generate Block Design	Properties		n n ×	Si	imulator	langua	ge:	Mixed													
✓ SIMULATION		(.de. ] (	+ 0	Sy	nthesis							Imp	emnt	ation							
Run Simulation				St	tatus:		Not	started				Sta	tus:			Not	started				
<ul> <li>RTL ANALYSIS</li> </ul>	Select an obje	ct to see properties			lessages art:	i.		errors or warnie 325tffg900-2	ngs			Me Par	ssages: t:				errors or wan 325tffg900-1				
Open Elaborated Design				St	trategy: enort St		Viva	do Synthesis D	lefaults lefault Reports			Str	ategy: nort Strat			Viva	ido Impleme	ntation Defa			
<ul> <li>SYNTHESIS</li> </ul>				10	eoon su	concerns:	100	on sympess o	epida senansi.			.66	000.504	eux.			and encoverore	nzanon cieta	OIL APORTS		
Run Synthesis	Tcl Console Message	Log Reports	Design	Runs	×															? -	_ 🗆 🖾
<ul> <li>Open Synthesized Design</li> </ul>	Q 至 0 H	« » » •	• %																		
		raints Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP	Start	Elapsed					
<ul> <li>IMPLEMENTATION</li> </ul>	✓ D synth_1 const																		ults (Vivado Syr		
Run Implementation	⊳ impl_1 const	rs_1 Not started															vivado im	plementatio	n Defaults (Viva	Joo Impien	nentation
Open Implemented Design																					
PROGRAM AND DEBUG																					

#### Figure 1: 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 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.

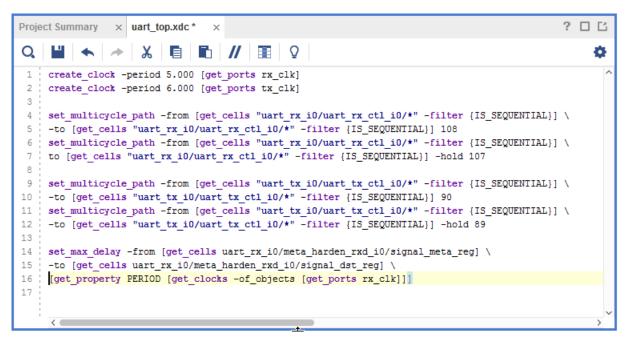
When used in the context of a top-level design, the parent XDC file provides the clock constraints and the standalone OOC XDC file is not needed.

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:



#### Figure 2: File Contents of uart\_top.xdc

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 Figure 2 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.

0

**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 Figure 3:
  - File type: **XDC**
  - File name: uart\_top\_ooc.xdc
  - File location: <Local to Project>
- 5. Click **OK**.





🙏 Create Constraints File						
Create a new co project	4					
<u>F</u> ile type:	D XDC	~				
File name:	uart_top_ooc.xdc	⊗				
Fil <u>e</u> location:	<local project="" to=""></local>	~				
?	OK Can	cel				

Figure 3: 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.

Project Summary × uart_top.xdc * × uart_top_ooc.xdc * ×	? 🗆 🖒
	۵
1 create_clock -period 5.000 [get_ports rx_clk]	^
<pre>2 create_clock -period 6.000 [get_ports tx_clk]</pre>	
	~
<	>

Figure 4: OOC XDC





- 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.

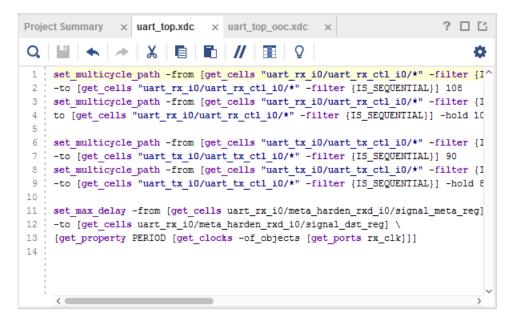


Figure 5: Updated uart\_top.xdc

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 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 Figure 6.





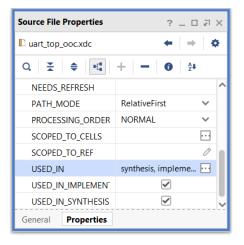


Figure 6: Used In Property

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.

🍌 Make Selecti	on					×
Unused values: 32			Selected	values: 3	₽Z₽	
ipstatic	^	•	synthesi	s		t
iptraffic		-	implem	entation		1
opt_design		_	out_of_o	ontext		-
opt_design_post		4				Ť
phys_opt_design	~	Ħ				+
< — >						
			ОК	Ca	ncel	
< <b>—</b> >	~		OK	Ca	ncel	

Figure 7: Make Selection Dialog Box

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:

Source File Properties	? _ D @ X				
uart_top.xdc	<b>←</b>   →   <b>⊅</b>				
Q   ¥   ♦   ■   +	- 0 Å				
CLASS	file				
CORE_CONTAINER					
FILE_TYPE	XDC 🗸				
IMPORTED_FROM	C:/Tutorials/workspace/u				
IS_AVAILABLE	~				
IS_ENABLED	$\checkmark$				
IS_GENERATED					
IS_GLOBAL_INCLUDE					
LIBRARY	xil_defaultlib				
NAME	C:/Tutorials/workspace/ug1				
NEEDS_REFRESH	~				
PATH_MODE	RelativeFirst 🗸 🗸				
PROCESSING_ORDER	LATE				
SCOPED_TO_CELLS	EARLY				
SCOPED_TO_REF	NORMAL				
USED_IN	LATE				
USED_IN_IMPLEMENTA1	$\checkmark$				
USED_IN_SYNTHESIS	$\checkmark$				
General <b>Properties</b>					

**Figure 8: Source File Properties** 

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:

🝌 Create and Package New I	p	×
VIVADO.	Create and Package New IP This wizard can be used to accomplish following tasks:	
	<ul> <li>Package a new IP for the Vivado IP Catalog</li> <li>This wizard will guide you through the process of creating a new Vivado IP using source files and information from your current project, block design or specified directory.</li> <li>Create a new AXI4 Peripheral</li> <li>This wizard will guide you through the process of creating a new AXI4 peripheral which includes HDL, driver, software test application, IP Integrator VIP simulation and debug demonstration design.</li> </ul>	
<b>£</b> XILINX.	Click Next to continue	
(?)	< <u>B</u> ack <u>Einish</u> Cancel	

Figure 9: Choose Create Peripheral or Package IP Window

- 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 Figure 10.





À Create and Package New II				×
Package Your Current	roject			
Select the directory where the project.	IP Definition will be cre	ated and the associate	ed options for packaging the curre	ent 🥕
IP location: c:/tutorials,		do-creating-packagin	g-ip-design/lab_1/my_simple_uarl	t.sr⊗ ····
<ul> <li>Include .xci files</li> </ul>				
O Include IP gener	ted files			
?		< <u>B</u> ack	<u>N</u> ext > <u>Finish</u>	Cancel

Figure 10: Package Current Project

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.

🝌 Create and Package New	IP X
	New IP Creation
HLx Editions	The following pieces of information will be gathered:
	<ul> <li>Identification information based on top module name</li> </ul>
	<ul> <li>Family compatibility based on part in the project</li> </ul>
	<ul> <li>File(s) from Synthesis and Simulation file sets</li> </ul>
	<ul> <li>Ports from the file containing the top module</li> </ul>
	<ul> <li>Parameters from the file containing the top module</li> </ul>
	<ul> <li>Bus Interfaces based on port names</li> </ul>
	<ul> <li>Address Spaces and Memory Maps based on inferred bus interfaces</li> </ul>
	Following file will be created on disk along with corresponding customization files: c/tutorials/workspace/ug1119-vivado-creating-packaging-ip-design/lab_1/my_simple_uart.srcs/component.xml
<b>£</b> XILINX.	Click Finish to continue
?	< <u>Back</u> Next > Einish Cancel

Figure 11: Begin IP Creation

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 Figure 12.





Package IP - uart_top			_ D 7 X
Packaging Steps	Identification		
<ul> <li>Identification</li> <li>Compatibility</li> <li>File Groups</li> <li>Customization Parameters</li> <li>Ports and Interfaces         Addressing and Memory</li> <li>Customization GUI         Review and Package</li> </ul>		xilinv.com user uart_top 1.0 uart_top_v1_0 uart_top_v1_0 c/tutorials/workspace/ug1119-vivado-creating-packaging-ip-di c/tutorials/workspace/ug1119-vivado-creating-packaging-ip-di	

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
  - 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 Figure 13.





🝌 IP Categories X
Choose or create an IP category.
Q   ¥   \$   +   =
AXI Infrastructure
— Automotive & Industrial
Automotive
BaselP
✓ □ Basic Elements
Accumulators
Comparators
Counters
Memory Elements
Registers, Shifters & Pipelining
Communication & Networking
Error Correction
Ethernet
Modulation
Networking
Serial Interfaces
Telecommunications
Wireless
Debug & Verification
ChipScope Pro 🗸
OK Cancel

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

Packaging Steps	File Groups					
/ Identification	Q   ¥   ♦   ■ +	G				
Compatibility	Name	Library Name	Туре	Is Include	File Group Name	M
Company	Standard					
File Groups	Synthesis (9)					u
	src/uart_top.xdc		xdc		xilinx_anylanguagesynthesis	
Customization Parameters	src/uart_top_ooc.xdc		xdc		xilinx_anylanguagesynthesis	
Ports and Interfaces	@ src/meta_harden.v		verilogSource		xilinx_anylanguagesynthesis	
	e src/uart_baud_gen.v		verilogSource		xilinx_anylanguagesynthesis	
Addressing and Memory	@ src/uart_rx.v		verilogSource		xilinx_anylanguagesynthesis	
Customization GUI	e src/uart_rx_ctl.v		verilogSource		xilinx_anylanguagesynthesis	
	e src/uart_tx.v		verilogSource		xilinx_anylanguagesynthesis	
Review and Package	<pre>w src/uart_tx_ctl.v</pre>		verilogSource		xilinx_anylanguagesynthesis	
	<pre>we src/uart_top.v</pre>		verilogSource		xilinx_anylanguagesynthesis	
	<ul> <li>Simulation (7)</li> </ul>					u
	src/meta_harden.v		verilogSource		xilinx_anylanguagebehavioralsimulation	
	e src/uart_baud_gen.v		verilogSource		xilinx_anylanguagebehavioralsimulation	
			verilogSource		xilinx_anylanguagebehavioralsimulation	
	<pre>w src/uart_rx_ctl.v</pre>		verilogSource		xilinx_anylanguagebehavioralsimulation	
	@ src/uart_tx.v		verilogSource		xilinx_anylanguagebehavioralsimulation	
	<pre>we src/uart_tx_ctl.v</pre>		verilogSource		xilinx_anylanguagebehavioralsimulation	
	@ src/uart_top.v		verilogSource		xilinx_anylanguagebehavioralsimulation	
	Advanced					
	> 📄 UI Layout (1)					

#### Figure 14: File Groups

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

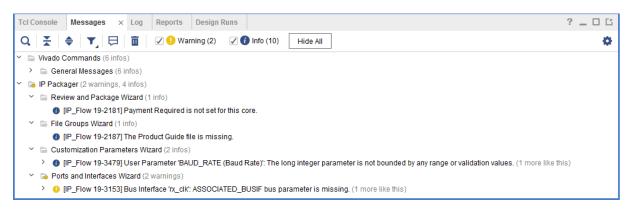


Figure 15: IP Packager Messages





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 the following figure:

🝌 Add File Group	×
Select File Group(s) to add.	4
File Group	
Q <u>X</u> <del>\$</del>	
<ul> <li>Standard</li> <li>Synthesis</li> <li>Synthesis Memory Archive</li> <li>Simulation</li> <li>Simulation Memory Archive</li> <li>Product Guide</li> <li>Readme</li> <li>Examples</li> </ul>	Î ×
Description Product Guide The IP documentation URL which previously consisted of separate readme, datasheet, user guide and other collateral disk files. OK Can	~

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.

- 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.

**Creating, Packaging Custom IP Tutorial** UG1119 (v2019.2) March 2, 2020 www.xilinx.com





- 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.

A	Add IP File	es (Product	Guide)			$\times$
:	Select files t	o add to file	e group.Product Guide		4	
	+, -	1	Ļ			
		Index	Name	Library	Location	
	-	1	my_simple_uart_product_guide.pdf	N/A	C:/Tutorials/workspace/ug11	
	_	nd add RTL <u>i</u>	Add Files Add Directories	<u>C</u> re	> rate File	
			-			
	Md so	urces from s	ubdirectories		OK Cancel	

Figure 17: Add Product Guide

#### 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:

Proje	ect Summary × Package IP - uar	rt_top ×	00
Pac	kaging Steps	Review and Package	
-	Identification	3 warnings 3 info messages	
4	Compatibility	Summary	
9	File Groups	Display name: uart_top_v1_0	î
4	Customization Parameters	Description: uart_top_v1_0 Root directory: c/temp/my_simple_uart	~
9	Ports and Interfaces	After Packaging	
	Addressing and Memory	An archive will not be generated. Use the settings link below to change your pre Project will be removed after completion	eferi
4	Customization GUI	<	>
	Review and Package	Package IP	

#### 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 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.

 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





🝌 New IP Location		×
Manage IP Settings Set options for creating an	d generating IP.	4
Part:	@ xc7k325tffg900-2	
Target language:	Verilog	~
Target simulator:	Vivado Simulator	~
Simulator language:	Mixed	~
IP location:	C:/Tutorials/ug1119-vivado-creating-packaging-ip-design/lab_1	⊗ …
(?)	< Back Next > Einish	Cancel

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, 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:

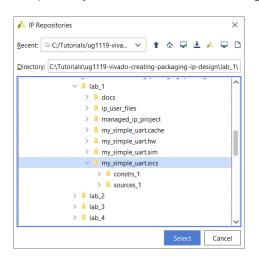


Figure 21: Manage IP Repository

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.

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 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 Figure 22.





IP Catalog				? _ 🗆	ъ×
Cores   Interfaces					
ב   \$   \$   \$   <		۲			Φ
Name	, 1	AXI4	Status	Licen	se
User Repository (c:/Tutorials/ug1119-vivado-creating-packaging-ip-design/l	lab_	_1/my_simple_uart.si	rcs)		^
Communication & Networking					
Serial Interfaces					
🍄 My Simple UART			Production	Inclue	led
V SerIP					
👎 My Simple UART			Production	Inclue	led
Vivado Repository					
Embedded Processing					
🗸 🖨 AXI Peripheral					~
					>
Details					
Select an IP or Interface or Repository f	to s	ee details			

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.

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

Details	
Name:	My Simple UART
Version:	1.0 (Rev. 2)
Description:	My simple example UART interface
Status:	Production
License:	Included
Vendor:	My Company
VLNV:	my_company:user:my_simple_uart:1.0
Repository:	c:/Tutorials/ug1119-vivado-creating-packaging-ip-design/lab_1/my_simple_uart.srcs

Figure 23: My Simple UART – Details





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

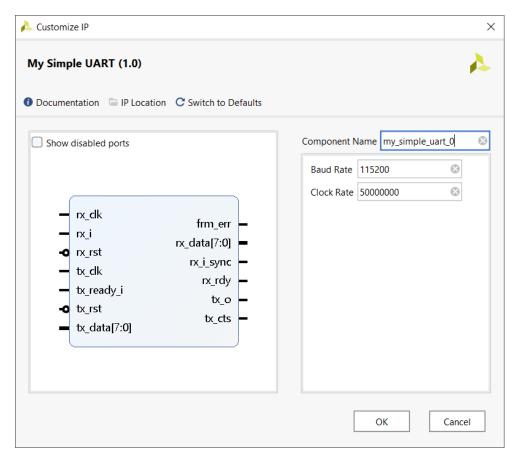


Figure 24: Customize IP – My Simple UART

- 14. Optionally: 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:

🙏 Generate Output Products	×
The following output products will be generated.	4
Preview	
Q ≚ ♦	
Instantiation Template	
Synthesized Checkpoint (.dcp)	
Structural Simulation	
Synthesis Options Global Out of context per IP Run Settings	
Number of jobs: 4	
Apply     Generate     Ski	р

Figure 25: Generate Output Products

#### 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.

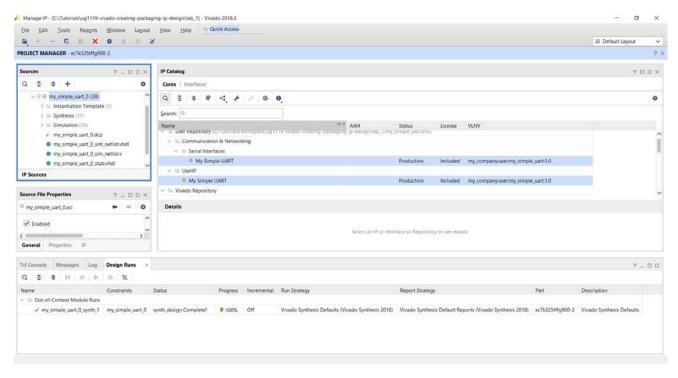
#### 17. Click **OK**.

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.



#### 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.





# 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 testbench 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: 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

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 2019.x > Vivado 2019.x

Or

• Click the Vivado 2019.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<sup>®</sup>-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**.

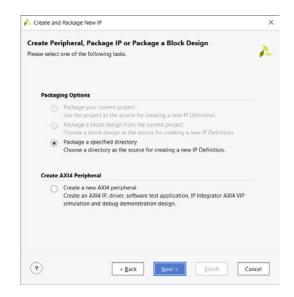


Figure 27: Create Peripheral, Package IP or Package a Block Design

 Set Directory to <Extract\_Dir>/lab2/custom\_ip\_repo/wave\_gen\_v1\_0, as shown in the following figure:

Create and Package N	rw IP	>
ackage a Specified	Directory	
elect the directory when	e sources to be packaged are located.	1
Directopr reating	packaging-ip-design/lab_2/custom_ip_repo/wave_ge	w v1 d63
		ICHONE
Package as a lib	ary core	
~		
(?)	< Back Next > Einish	Cancel

Figure 28: Package a Specified Directory

www.xilinx.com





- 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.

Pack	age IP - wave_gen			_ D @ X
Pac	kaging Steps	Identification		
-	Identification	Vendor:	xilinx.com	8
-	Compatibility	Library:	user	۲
4	File Groups	Name:	wave_gen	8
4	Customization Parameters	Version:	1.0	۲
4	Ports and Interfaces	Display name:	wave_gen_v1_0	۲
	Addressing and Memory	Description:	wave_gen_v1_0	8
	Customization GUI	Vendor display name:		
1		Company url:		
	Review and Package		c:/Tutorials/ug1119-vivado-creating-packaging-ip-design/lab_2/custom_ip_repo/wave_gen_v1_0	
		Xmi lile name.	c:/Tutorials/ug1119-vivado-creating-packaging-ip-design/lab_2/custom_ip_repo/wave_gen_v1_0/com	iponent.xmi
		Categories		
		+  -  +		
		/UserIP		

Figure 29: 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 **t** to add a new category.
- 3. In the IP Categories dialog box, click the **Add** button **t** 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.

Packaging Steps	File Groups						
✓ Identification	Q   ≚   ♦   •€   +   C						
✓ Compatibility	Name	Library Name	Туре	ls Include	Used In Constant	File Group Name	Model Name
<ul> <li>File Groups</li> </ul>	Standard						
	> 🗁 Synthesis (26)						wave_g
<ul> <li>Customization Parameters</li> </ul>	> 🖻 Simulation (24)						wave_g
<ul> <li>Ports and Interfaces</li> </ul>	> 🗁 Product Guide (1)						
	Advanced						
Addressing and Memory	> 🗁 Test Bench (9)						
<ul> <li>Customization GUI</li> </ul>	> 🖨 UI Layout (1)						

#### Figure 30: 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:

Packaging Steps	File Groups           Q   X   \$\Rightarrow\$   \$\Rightarrow\$							
<ul> <li>Identification</li> </ul>								
<ul> <li>Compatibility</li> </ul>	Name	Library Name	Туре	ls Include	Used In Constant	File Group Name		
<ul> <li>File Groups</li> </ul>								
<ul> <li>Customization Parameters</li> </ul>	V 🖻 Synthesis (26)							
	src/wave_gen_pins.xdc		xdc			xilinx_anylanguagesynt		
<ul> <li>Ports and Interfaces</li> </ul>	src/wave_gen_timing.xdc		xdc			xilinx_anylanguagesynt		
Addressing and Memory	src/char_fifo.v		verilogSource			xilinx_anylanguagesynt		
	src/clk_core.v		verilogSource			xilinx_anylanguagesynt		
<ul> <li>Customization GUI</li> </ul>	src/clk_div.v		verilogSource			xilinx_anylanguagesynt		
Review and Package	src/clk_gen.v		verilogSource			xilinx_anylanguagesynt		
	src/clogb2.vh		verilogSource			xilinx_anylanguagesynt		
	src/clkx_bus.v		verilogSource			xilinx_anylanguagesynt		
	src/cmd_parse.v		verilogSource			xilinx_anylanguagesynt		
	src/dac_spi.v		verilogSource			xilinx_anylanguagesynt		
	src/debouncer.v		verilogSource			xilinx_anylanguagesynt		
	src/lb_ctl.v		verilogSource			xilinx_anylanguagesynt		
	src/meta_harden.v		verilogSource			xilinx_anylanguagesynt		
	src/out_ddr_flop.v		verilogSource			xilinx_anylanguagesynt		
	src/reset_bridge.v		verilogSource			xilinx_anylanguagesynt		
	src/resp_gen.v		verilogSource			xilinx_anylanguagesynt		
	src/rst_gen.v		verilogSource			xilinx_anylanguagesynt		

Figure 31: 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, shown in the following figure, select the **Review and Package** page.

Pr	oject Summary × Package I	P-wave_gen ×	
P	Packaging Steps	Review and Package	
•	Identification	Summary	
1	Compatibility	Display name: wave_gen_v1_0	
1	File Groups	Description: UG1119 Tutorial Lab#2 Wave Generator tutorial design	
•	Customization Parameters	Root directory: c:/Temp/lab_2/custom_ip_repo/wave_gen_v1_0	
•	Ports and Interfaces	After Packaging	
	Addressing and Memory	An archive will not be generated. Use the settings link below to change your preference	
1	Customization GUI	Project will be removed after completion Edit packaging settings	
	Review and Package		
<	> >	Package IP	

Figure 32: 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 Settings.
- 2. In the Settings dialog box, expand IP and select Repository.





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, as seen below:

Q		IP > Repository			
Project Settings General	Î	Add directories to the list of repositories. You may then add additional IP to a selected repository. If an IP is disabled then a tool-tip will alert you to the reason.			
Simulation Elaboration		IP Repositories			
<u>Synthesis</u>		$+  -  \pm   \mp$			
Implementation Bitstream		/ug1119-vivado-creating-packaging-ip-design/lab_2/custom_ip_repo/wave_gen_v1_0 (Project)			
∨ IP					
Repository					
Packager		<			
Tool Settings		Refresh All			
Project					
IP Defaults					
Source File					
Display					
WebTalk					
Help					
> Text Editor	~				

Figure 33: 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**.





### 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**.

IP Catalog				?	_ 🗆 🔊 ×
Cores   Interfaces					
王   🖨 😰 🕫 🎤 🖉 🕲 🔍 🔍 Wave Gen	۲				٥
Name	1 AXI4	Status	License	VLNV	
User Repository (c:/Tutorials/ug1119-vivado-creating-packaging-ip-design/i	ab_2/custom_ip_repo/	wave_gen_v1_0)			
🗸 🚍 My Company					
Wave Generator Tutorial		Production	Included	my_company:user:wave_gen_tutorial:1.0	
✓					
👎 Wave Generator Tutorial		Production	Included	my_company:user:wave_gen_tutorial:1.0	
Details					
Name: Wave Generator Tutorial					
Version: 1.0 (Rev. 1)					
Description: UG1119 Tutorial Lab #2 - Wave Generator tutorial design					
Status: Production					
License: Included					
Vendor: My Company					
VLNV: my_company:user.wave_gen_tutorial:1.0					
Repository: c:/Tutorials/ug1119-vivado-creating-packaging-ip-design/lab_2/cus	tom_ip_repo/wave_ge	1_v1_0			

Figure 34: IP Catalog

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: 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.

À Customize IP			×
Wave Generator Tutorial (1.0)  Documentation PLocation C Switch to Defa	ults		4
Show disabled ports	Component Name	wave_gen_tutorial_0	0
	Baud Rate 1	15200	
	Clock Rate Rx 2	00000000 📀	
	Clock Rate Tx 1	66667000 🛞	
	Nsamp Wid 1	0 🛞	
– clk_pin_n – spi_mosi_pin –	Pw 3	$\otimes$	
dac_cs_n_pin rxd_pin dac_ctr_n_pin lb_sel_pin led_pins[7:0]			
		ОК	Cancel

Figure 35: 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 reopens to report the output products generated successfully.

6. Close Vivado.

# 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 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 2019.x > Vivado 2019.x
- Click the Vivado 2019.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.
- 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.

oose a defa	ault Xilinx par	t or board for yo	ur project.						
Parts	<u>Boards</u>								
Reset All	Filters								
Category:	All		~	Package:	All	~	Temperat	ure: All	~
Family:	All		~	Speed:	All	~			
	tsdv484-2L	I/O Pin Count 484	Available IOBs 285	LUT Elemer 134600	nts	FlipFlops 269200	Block RAMs	Ultra RAMs U	DSI /40
	tsbv484-2L tsbv484-1	484	285	134600		269200	365	0	740 ~
xc7k70tfl		484	285	41000		82000	135	0	240
xc7k70tfl	-	484	285	41000		82000	135	0	240
xc7k70tfl	- bg484-2L	484	285	41000		82000	135	0	240
xc7k70tfl	bg484-1	484	285	41000		82000	135	0	240
xc7k70tfl	bg676-3	676	300	41000		82000	135	0	240
xc7k70tfl	bg676-2	676	300	41000		82000	135	0	240
	bg676-2L	676	300	41000		82000	135	0	240 🗸
<									>

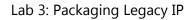
Figure 36: New Project – Default Part

You selected a Kintex<sup>®</sup>-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 (subcores) 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 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.**

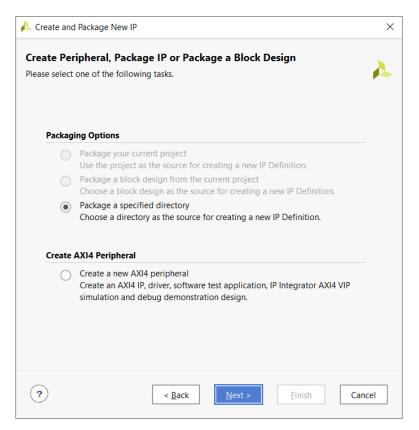


Figure 37: Create Peripheral, Package IP or Package a Block Design





- 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.

Create and Package New IP			×		
Package a S	ecified Di	rectory			
Select the direct	ory where sou	irces to be package	d are located.		A 19
Directory:	creating-pac	kaging-ip-design/l	ab_3/pcores/pro	pc_common_v3_00_	a 🛛 \cdots
🕑 Packag	as a library	core			
2		< <u>B</u> ack	<u>N</u> ext >	Einish	Cancel
$\odot$			<u>IN</u> EXL >	Linisi	Carleer

Figure 38: Package a Specified Directory

- 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

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

Package IP - proc_common		הם	×
Packaging Steps	Identification		
Identification	Vendor:	xilinx.com	
✓ File Groups	Library:	ip ©	
Review and Package	Name:	proc_common ©	
	Version:	3.00.a	
	Display name:	proc_common_v3_00_a	
	Description:	Proc Common v3.00.a Library Core	
	Vendor display name:		]
	Company url:		]
	Root directory:	c:/Tutorials/ug1119-vivado-creating-packaging-ip-design/lab_3/pcores/proc_common_v3_00_a	
	Xml file name:	c:/Tutorials/ug1119-vivado-creating-packaging-ip-design/lab_3/pcores/proc_common_v3_00_a/component.xml	(
	Categories		
	+ - +	l l	
		No content	

Figure 39: Package IP





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

Package IP - proc_common	א הם _ X
Packaging Steps	Review and Package
Identification File Groups Review and Package	Summary Display name: proc_common_v3_00_a Description: Proc Common v3.00.a Library Core Root directory: c/Tutorials/ug1119-vivado-creating-packaging-ip-design/lab_3/pcores/proc_common_v3_00_a
	After Packaging         An archive will not be generated. Use the settings link below to change your preference         Project will be removed after completion         Edit packaging settings

Figure 40: Review and Package

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.

**Creating, Packaging Custom IP Tutorial** UG1119 (v2019.2) March 2, 2020 www.xilinx.com





- 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**.
- In the Package a Specified Directory dialog box, set the following option: Directory: <Extract\_Dir>/lab3/pcores/axi\_gpio\_v1\_01\_b.

Create and Package New IP			×
Package a Specified Direct Select the directory where sources	-		A
	reating-packaging-ip-design,	/lab_3/pcores/axi_gpio_v1_01	_b ⊗ ····
Package as a library core			

Figure 41: 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 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:

Package IP - axi_gpio			_ 🗆 🖉 🗙
Packaging Steps	Identification		
<ul> <li>Identification</li> </ul>	Vendor:	xilinx.com	8
<ul> <li>Compatibility</li> </ul>	Library:	user	$\otimes$
✓ File Groups	Name:	axi_gpio	$\otimes$
<ul> <li>Customization Parameters</li> </ul>	Version:	1.0	$\otimes$
<ul> <li>Ports and Interfaces</li> </ul>	Display name:	axi_gpio_v1_0	
	Description:	axi_gpio_v1_0	$\otimes$
<ul> <li>Addressing and Memory</li> </ul>	Vendor display name:		
<ul> <li>Customization GUI</li> </ul>	Company url:		
Review and Package	Root directory:	c:/Tutorials/ug1119-vivado-creating-packaging-ip-design/lab_3/pcores/axi_gpio_v1_01_b	
	Xml file name:	c:/Tutorials/ug1119-vivado-creating-packaging-ip-design/lab_3/pcores/axi_gpio_v1_01_b/comp	onent.xml
	Categories		
	+  -  +	1	
	/UserIP		

Figure 42: Package IP

### 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 AXI Lite IPIG and the Proc Common display in the /Sub-Core References directories for Synthesis and Simulation.

Package IP - axi_gpio			_ 🗆 🖓 🗙		
Packaging Steps	File Groups				
<ul> <li>Identification</li> </ul>	Q 素 ≑ ∎ + C				
<ul> <li>Compatibility</li> </ul>	Name V 🖨 Standard	Туре			
<ul> <li>File Groups</li> </ul>	Synthesis (2)				
<ul> <li>Customization Parameters</li> </ul>	Sub-Core References				
Customization Parameters	xilinx.com:ip:proc_common:3.00.a				
<ul> <li>Ports and Interfaces</li> </ul>	xilinx.com:ip:axi_lite_ipif:1.01.a				
	xilinx.com:ip:interrupt_control:2.01.a				
<ul> <li>Addressing and Memory</li> </ul>	M hdl/vhdl/gpio_core.vhd	axi_gpio_v1_01_b	vhdlSource		
<ul> <li>Customization GUI</li> </ul>	M hdl/vhdl/axi_gpio.vhd	axi_gpio_v1_01_b	vhdlSource		
	Simulation (2)				
Review and Package	Sub-Core References				
	xilinx.com:ip:proc_common:3.00.a				
	xilinx.com:ip:axi_lite_ipif:1.01.a				
	xilinx.com:ip:interrupt_control:2.01.a				
	M hdl/vhdl/gpio_core.vhd	axi_gpio_v1_01_b	vhdlSource		
	hdl/vhdl/axi_gpio.vhd	axi_gpio_v1_01_b	vhdlSource		
	Product Guide (1)				
	doc/html/change_log.html		html		
	Advanced				
	V 🖨 UI Layout (1)				
	xgui/axi_gpio_v1_0.tcl		tclSource		
	Data Sheet (1)				
	🔀 doc/ds744_axi_gpio.pdf		pdf		
<	<		>		

Figure 43: Package IP: File Groups

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

Packaging Steps	Customization Parameters						
<ul> <li>Identification</li> </ul>	Q   ¥   ♦   ■ +   C						
Commentative	Name	Description	Display Name	Value	Value Bit String Length	Value Format	Value Source
Compatibility	Customization Parameters						
File Groups	C_INSTANCE		C Instance	axi_gpio_inst	0	string	default
	C_S_AXI_ADDR_WIDTH		C S Axi Addr Width	9	0	long	default
Customization Parameters	C_S_AXI_DATA_WIDTH		C S Axi Data Width	32	0	long	default
<ul> <li>Ports and Interfaces</li> </ul>	C_GPIO_WIDTH		C Gpio Width	32	0	long	default
	C_GPIO2_WIDTH		C Gpio2 Width	32	0	long	default
Addressing and Memory	C_ALL_INPUTS		C All Inputs	0	0	long	default
Customization GUI	C_ALL_INPUTS_2		C All Inputs 2	0	0	long	default
Customization GOI	C_INTERRUPT_PRESENT		C Interrupt Present	0	0	long	default
Review and Package	C_DOUT_DEFAULT		C Dout Default	0x00000000	32	bitString	default
	C_TRI_DEFAULT		C Tri Default	0xFFFFFFFF	32	bitString	default
	C_IS_DUAL		C Is Dual	0	0	long	default
	C_DOUT_DEFAULT_2		C Dout Default 2	0x00000000	32	bitString	default
	C_TRI_DEFAULT_2		C Tri Default 2	0xFFFFFFFF	32	bitString	default

Figure 44: Package IP – Customization Parameters





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

Packa	Package IP - axi_gpio _ 🗆 🖓				
Pac	kaging Steps	Review and Package			
•	Identification	Summary			
	Compatibility	Display name: My AXI GPIO EDK Pcore Tutorial			
	File Groups	Description: UG1119 Tutorial Lab #3 - AXI GPIO EDK Pcore Root directory: c:/Tutorials/ug1119-vivado-creating-packaging-ip-design/lab_3/pcores/axi_gpio_v1_01_b			
	Customization Paramete				
	Ports and Interfaces				
	Addressing and Memory	After Packaging			
	Customization GUI	An archive will not be generated. Use the settings link below to change your preference Project will be removed after completion			
	Review and Package	Edit packaging settings			
<	<b></b> >	Package IP			

#### Figure 45: 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, shown in Figure 46.



# **E** XILINX.

IP Catalog			? _ 🗆	₽ X
Cores   Inte	rfaces			
¥   \$	ኛ ⊨] ≁   ∅   ❹   Q-			•
Name	×1	AXI4	Status	Lic
🗸 🗎 User Re	pository (c:/Tutorials/ug1119-vivado-creating-packaging-ip-design/lab	_3/pcores/axi_g	pio_v1_01_b)	^
🗸 🖨 User	IP			
👎 Mj	y AXI GPIO EDK Pcore Tutorial	AXI4	Production	Inc
🖨 User Re	pository (c:/Tutorials/ug1119-vivado-creating-packaging-ip-design/lab	_3/pcores/interru	upt_control_v2_01_a)	
🖨 User Re	epository (c:/Tutorials/ug1119-vivado-creating-packaging-ip-design/lab	_3/pcores/axi_lit	e_ipif_v1_01_a)	
🖨 User Re	epository (c:/Tutorials/ug1119-vivado-creating-packaging-ip-design/lab	_3/pcores/proc_	common_v3_00_a)	~
<				>
Details				
Name:	My AXI GPIO EDK Pcore Tutorial			
Version:	1.0 (Rev. 1)			
Interfaces:	AXI4			
Description:	UG1119 Tutorial Lab #3 - AXI GPIO EDK Pcore			
Status:	Production			
License:	Included			
Vendor:	My Company			
VLNV:	my_company:user:axi_gpio:1.0			
Repository:	c:/Tutorials/ug1119-vivado-creating-packaging-ip-design/lab_3/pcores	s/axi_gpio_v1_0	1_b	

#### Figure 46: 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.





# 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 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.

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 2019.x > Vivado 2019.x
- Click the Vivado 2019.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.

**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

New Project		
Project Name		
nter a name for yo	ur project and specify a directory where the project data files will be stored.	
Project name:	project_lab4	0
Project location:	C/Tutorials/workspace/ug1119-vivado-creating-packaging-ip-design/lab_4	0
Create projec	t subdirectory	
Project will be cr	eated at: C://lab_4/project_lab4	
	< Back Next > Finish	Cancel

Figure 47: 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<sup>™</sup> 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:

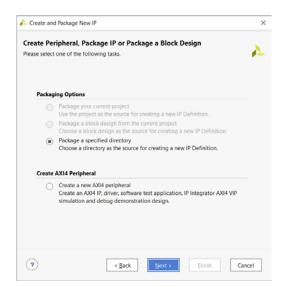


Figure 48: Create Peripheral, Package IP, or Package Block Design

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.

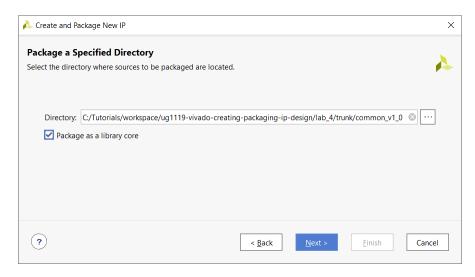


Figure 49: Package a Specified Directory

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 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:

Ргој	ect Summary × Package IP	common ×		06
Pa	ckaging Steps	Identification		
	Identification	Vendor:	xilinx.com	8
~	File Groups	Library:	İp	۲
	Review and Package	Name:	common	8
		Version:	1.0.	$\otimes$
		Display name:		0
		Description:		0
		Vendor display name:		
		Company url:		
		Root directory:	c:/Temp/lab_4/trunk/common_v1_0	
		Xml file name:	c:/Temp/lab_4/trunk/common_v1_0/component.xml	
		Categories		
		+  -  +	+	
			No content	

#### Figure 50: 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: <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. 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. 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.
- 4. The Add Repository dialog box opens to display that the trunk repository was added to the project and 1 IP was found, as shown in the following figure:

🝌 Add Repository	×
1 repository was added to the project. For more information related to disabled IPs, please refer to IP Catalog.	
Repository	
*	
v c:/Tutorials/workspace/ug1119-vivado-creating-packaging-ip-design/lab_4/trunk IPs (1)	
ОК	

Figure 51: Add Repository

#### 5. Click OK.

The Repository Manager is now populated with the selected IP repository, in addition to the common\_v1\_0 repository.

- 6. Remove the common\_v1\_0 repository by selecting it and selecting the **Remove** button
- 7. 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 check the Package as a library core option.

À Create and Package New IP		×
Package a Specified Directory		
Select the directory where sources to be	iged are located.	
Directory: C:/Tutorials/workspace	19-vivado-creating-packaging-ip-design/lab_4/trunk/myip_v1_0 💿 …	
Package as a library core		
		1
?)	< <u>B</u> ack <u>Next</u> Einish Cancel	]

Figure 52: 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.

Package IP - myip_top		_ D & X
Packaging Steps	Identification	
✓ Identification	Vendor:	xilinx.com
<ul> <li>Compatibility</li> </ul>	Library:	user
✓ File Groups	Name:	myip_top 📀
Customization Parameters	Version:	1.0
<ul> <li>Ports and Interfaces</li> </ul>	Display name:	myip_top_v1_0
Addressing and Memory	Description:	myip_top_v1_0
	Vendor display name:	
<ul> <li>Customization GUI</li> </ul>	Company url:	
Review and Package	Root directory:	c:/Tutorials/ug1119-vivado-creating-packaging-ip-design/lab_4/trunk/myip_v1_0
	Xml file name:	$c/Tutorials/ug1119-vivado-creating-packaging-ip-design/lab_4/trunk/myip\_v1\_0/component.xml = 0.0000000000000000000000000000000000$
	Categories	
	$+  -  \uparrow  $	ŧ
	/UserIP	

#### Figure 53: 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: <company\_URL>





10. Click the **File Groups** to examine the files included in the packaged IP, as shown in the following figure:

Package IP - myip_top			_ D @ X
Packaging Steps	File Groups		
<ul> <li>Identification</li> </ul>	Q   素   ≑   €   +   C		
<ul> <li>Compatibility</li> </ul>	Name V   Standard	Library Name	Туре
✓ File Groups	✓		
Customization Parameters	src/myip_top.v     Simulation (1)		verilogSource
<ul> <li>Ports and Interfaces</li> </ul>	src/myip_top.v     Advanced		verilogSource
Addressing and Memory	V DI Layout (1)		
<ul> <li>Customization GUI</li> </ul>	😢 xgui/myip_top_v1_0.tcl		tclSource
Review and Package			
	<		>

Figure 54: 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 in the project, you can see that the common module is missing, as shown in the following figure:

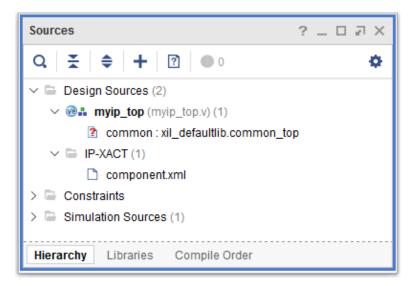


Figure 55: Hierarchy Sources Viewer

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 the Synthesis file group and select **Add Sub-Core Reference**, as shown in the following figure:

Packaging Steps	File Groups		
<ul> <li>Identification</li> </ul>	Q   素   €   ■	+ C	
<ul> <li>Compatibility</li> </ul>	Name   V  Standard	Library Name	Туре
<ul> <li>File Groups</li> </ul>	<ul> <li>Synthesis (1)</li> </ul>		
Customization Parameters	<ul> <li>in src/myip</li> <li>✓  ☐ Simulation</li> </ul>	Add Files Add URL	ogSource
<ul> <li>Ports and Interfaces</li> </ul>		Add Sub-Core Reference. Add File Group	bgSource
Addressing and Memory	V 🖨 UI Layout (	Remove File Group	
<ul> <li>Customization GUI</li> </ul>	🖹 xgui/myi	Сору То	)urce
Review and Package	C	Refresh Table	
		Export to Spreadsheet	

Figure 56: 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:

Add Sub-Core Reference					
Select IP to be used as references in File C	Group: Synthesis				2
Search: Q- my comp	(1 match)				
Name	^1 AXI4	Status	License	VLNV	
My Company Common Library			Included	my_company:ip:common:1.0.	
				ОК	Cancel

Figure 57: 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.

Package IP - myip_top _ D 2 X						
Packaging Steps	File Groups					
<ul> <li>Identification</li> </ul>	Q ≚ ≑ ⊑ + C					
<ul> <li>Compatibility</li> </ul>	Name	Library Name	Туре			
✓ File Groups	✓					
Customization Parameters	<ul> <li>Sub-Core References</li> <li>my_company:ip:common:1.0.</li> </ul>					
<ul> <li>Ports and Interfaces</li> </ul>			verilogSource			
Addressing and Memory	<pre>@ src/myip_top.v</pre>		verilogSource			
<ul> <li>Customization GUI</li> </ul>	Advanced Il Layout (1)					
Review and Package	🗟 xgui/myip_top_v1_0.tcl		tclSource			
	<	-	>			

Figure 58: 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.

Package IP - myip_top			_ D @ X
Packaging Steps	File Groups		
<ul> <li>Identification</li> </ul>	Q   ¥   ♦   •   +   C		
<ul> <li>Compatibility</li> </ul>	Name	Library Name	Туре
✓ File Groups	✓ ➡ Synthesis (1)		
Customization Parameters	✓ Sub-Core References ♥ my_company:ip:common:1.0.		
<ul> <li>Ports and Interfaces</li> </ul>	ege src/myip_top.v  Getain(1)		verilogSource
Addressing and Memory	Sub-Core References		
<ul> <li>Customization GUI</li> </ul>	<pre></pre>		verilogSource
Review and Package	Advanced		
	V 🖨 UI Layout (1)		
	xgui/myip_top_v1_0.tcl		tclSource
	<		>

Figure 59: 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 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:

IP Catalog							? _ D @ X
Cores   Inte	erfaces						
¥   \$	≆∣-⊈ ⊁ ₽ @ 0	Q~ My IP		8			0
Name		~1	AXI4	Status	License	VLNV	
🗸 🗎 User Re	epository (c:/Tutorials/ug1119-viva	do-creating-packagi	ing-ip-design/lab_4/tr	unk/myip_v1_0)			
🗸 🗎 User	rIP						
<b>₽</b> U	G1119 Tutorial Lab #4 - My IP			Production	Included	my_company:user:myip_top:1.0	
Details							
Name:	UG1119 Tutorial Lab #4 - My IP						î
Version:	1.0 (Rev. 1)						
Description:	myip_top_v1_0						
Status:	Production						
License:	Included						
Vendor:	My Company						
VLNV:	my_company:user:myip_top:1.0						
Repository:	c:/Tutorials/ug1119-vivado-creatin	ig-packaging-ip-de:	sign/lab_4/trunk/myip	_v1_0			

Figure 60: 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**

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at 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 2013-2020 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.

