# Building Custom AXI IP

2023.2

## Abstract

This lab guides you through the process of creating and adding a custom AXI peripheral to the Vivado™ IP catalog by using the Create and Package IP Wizard. The focus is on the process of adding an AXI interface onto an existing peripheral—not the actual design of the peripheral logic.

This lab should take approximately 45 minutes.

## CloudShare Users Only

You are provided with three attempts to finish a lab, where the time allotted to complete each lab is twice the expected completion time. Once the timer starts, you cannot pause the timer. Each lab attempt resets the previous attempt—your work from previous attempts is not saved.

## Objectives

After completing this lab, you will be able to:

* Create a custom AXI peripheral accessible for future design use from the IP catalog.
* Modify the top-level and AXI interface skeleton files created by the wizard to add custom functionality.
* Create and import user-defined peripheral port signals and parameters using the Package IP Wizard.

## Introduction

The purpose of this lab is for you to use the Create and Package IP Wizard to wrap an existing peripheral (in this case, a simple LED controller) with the AXI peripheral template and export the wrapped peripheral as an XACT IP.

The lab will illustrate a design flow targeted to building AXI interface slave peripherals. A project will be created in the Vivado Design Suite. A project could be the primary design project, or it could be for the sole purpose of launching the Create and Package IP Wizard to wrap the provided peripheral with an AXI interface.

Here you will use the Vivado Design Suite project as the starting point to launch the Create and Package IP Wizard. The Create and Package IP Wizard will be used to generate the peripheral directory structure, skeleton design files, and a Vivado project file that can be used as a design environment.

Once the IP is completed in the second instance of the Vivado Design Suite, the IP is passed back to the original project as an XACT IP.

A sign with text and arrows

Description automatically generated

Figure 4-1: How the Instances of the Vivado IDE Are Used

The actual peripheral design will be developed in the Vivado Design Suite project (which is created by the Create and Package IP Wizard). The purpose of the Create and Package IP Wizard-created project is to provide a design-authoring environment and the ability to check HDL syntax.

Synthesis will be performed in this lab only as a means to verify the syntax of the HDL additions. Implementation should not be performed in this lab because the IP RTL will be synthesized and implemented at the time that the IP is instantiated into a design.

The skeleton files will be modified to include the user-defined ports and generic parameters. The user-defined ports and parameters will then be imported via the Vivado IP packager with the peripheral becoming a member of the Vivado IP catalog.

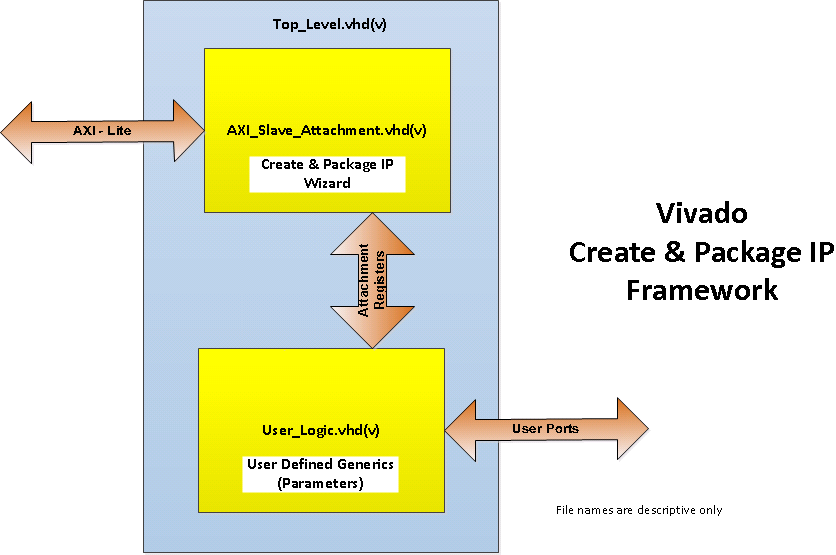


Figure 4-2: Create and Package IP Block Diagram

The Create and Package IP Wizard generates the above framework, which includes the directory structure, Top\_Level.vhd/v, AXI\_Slave\_Attachment.vhd/v, BFM simulation model, software driver skeleton, and example files.

Note that the skeleton.vhd/v filenames are just descriptive, as the actual names are based on the IP name and version and the type of AXI attachment selected in the wizard menu.

The User\_Logic.vhd/v file is the user-defined part of the peripheral IP. For this lab, a simple LED controller is provided for you. The block diagram is illustrated in the figure below.

A diagram of a computer component

Description automatically generated

Figure 4-3: LED Controller Block Diagram

The operation of the IP is far from being an actual controller, but some basic concepts such as custom user ports and parameterization are demonstrated.

The operation of the controller is simple. The intended output, LEDs, is vector-sized parameterized by NUM\_LEDS.

An output source multiplexer, controlled by user input CNT\_DISP, selects between a DATA\_IN input or a counter that will drive the LEDs. It is intended that the DATA\_IN input will be supplied from an AXI attachment register.

The counter is driven by a prescaler that has a parameterized count value, CNT\_PRESCALE\_VAL, so that the speed of the counter can be set to count fast for simulation viewing and slow so that one can see the LEDs blink in the hardware.

The counter also has a reset parameter LED\_RESET\_VAL, which is loaded into the counter when RST is asserted.

The table below summarizes the user-defined ports and parameters.

| User Signal | Direction | Source/Destination |
| --- | --- | --- |
| CLK | in | Driven by AXI attachment |
| RST | in | Driven by AXI attachment |
| DATA\_IN | in | Driven by AXI attachment |
| CNT\_DISP | in | External input |
| LEDs | out | External output – LEDs on board |

User-defined parameters are set in the IP Configuration panel for each IP instance when they are later instantiated into a design.

| User Parameter | Function |
| --- | --- |
| CNT\_PRESCALE\_VAL | Counter clock rate divider |
| LED\_RESET\_VAL | LED display reset value |
| NUM\_ACTIVE\_LEDS | LSB number of active LEDs output bits |
| NUM\_LEDS | LED vector width (number of LEDs) |

The NUM\_ACTIVE\_LEDS parameter will not be critical during this lab, but it is included for a simulation model for other labs. The NUM\_ACTIVE\_LEDS is an integer value that must be less than or equal to NUM\_LEDS, the number of LEDs on the board.

If equal to the number of LEDs, then either the count or DATA\_IN will be displayed on the LEDs (determined by input CNT\_DISP).

If NUM\_ACTIVE\_LEDS is less than NUM\_LEDS, then only the NUM\_ACTIVE\_LEDS number of LSB will be displayed with the remaining MSB bits (NUM\_LEDS – NUM\_ACTIVE\_LEDS) and will be forced to '1' (always on).

While this feature is not practical, it is ideal for a simulation waveform demonstration.

Understanding the Lab Environment

The labs and demos provided in this course are designed to run on a Linux® platform.

One environment variable is required: TRAINING\_PATH, which points to the location of the lab files. This variable comes configured in the CloudShare/CustEd\_VM environments.

Some tools can use this environment variable directly (that is, $TRAINING\_PATH is recognized and automatically expanded), and some tools require manual expansion (/home/amd/training for the CloudShare/CustEd\_VM environments). The lab instructions describe what to do for each tool. Other environments require the definition of this variable for the scripts to work properly.

The Vivado™ Design Suite and the Vitis™ Unified IDE offer a Tcl environment used in many labs. When either tool is launched, it starts with a clean Tcl environment with none of the procs or variables remaining from any previous launch of the tools.

If you sourced a Tcl script or manually set any Tcl variables and you closed the tool, when you reopen the tool, you will need to re-source the Tcl script and set any variables that the lab requires. This is also true of terminal windows—any variable settings will be cleared when a new terminal opens.

Nomenclature

Formal nomenclature is used to explain how different arguments are used. The following are some of the more commonly used symbols:

| Symbol | Description | Example | Explanation |
| --- | --- | --- | --- |
| <text> | Indicates a field | cd <dir> | <dir> represents the name of the directory. The < and > symbols are NOT entered. If the directory to change to is XYZ, then you would enter cd XYZ into the environment. |
| [text] | Indicates an optional argument | ls [ | more] | This could be interpreted as ls <Enter> or ls | more <Enter>. The first instance lists the files in the current Linux directory, and the second lists the files in the current Linux directory, but additionally runs the output through the more tool, which paginates the output. Here, the pipe symbol (|) is a Linux operator. |
| | | Indicates choices | cmd <ZCU104 | VCK190> | The cmd command takes a single argument, which could be ZCU104 OR VCK190. You would enter either cmd ZCU104 or cmd VCK190. |

## General Flow

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step 1:  Running  the  Wizard |  | Step 2:  Adding  HDL to the Peripheral Project | A grey arrow pointing to the right  Description automatically generated | Step 3:  Importing  Ports and Parameters |

Running the Create and Package IP Wizard Step

You will begin the lab by creating a new Vivado Design Suite project. As described in the introduction, you can create a project to be the main working project in which all your design work is performed, or solely as a platform to launch the Create and Package IP Wizard.

Here you will create a project for the sole purpose of launching the Create and Package IP Wizard, which you will in turn use to create the skeleton of an AXI-based peripheral that will be the base connection between the user IP and AXI port.

Here are two ways to open the Vivado Design Suite.

1-1. Open the Vivado Design Suite.

1-1-1. Click the Vivado icon () from the taskbar.

A red arrow pointing to a black box

Description automatically generated

Figure 4-4: Launching the Vivado Design Suite from the Taskbar

Note: It takes a few moments to launch. The order of the icons in your environment may be different.

Alternatively, open the Linux terminal window (<Ctrl + Alt + T>) and enter the following:

source /opt/amd/Vivado/2023.2/settings64.sh; vivado

Note: This installation path is valid for the CustEd VM and CloudShare environments. Use the proper path for your environment.

The tool opens with a Welcome window. From here you can create a new project, open an existing project, enter Tcl commands, and access documentation and examples.

1-1-2. [Optional] Maximize the window as there is a lot of information to see.

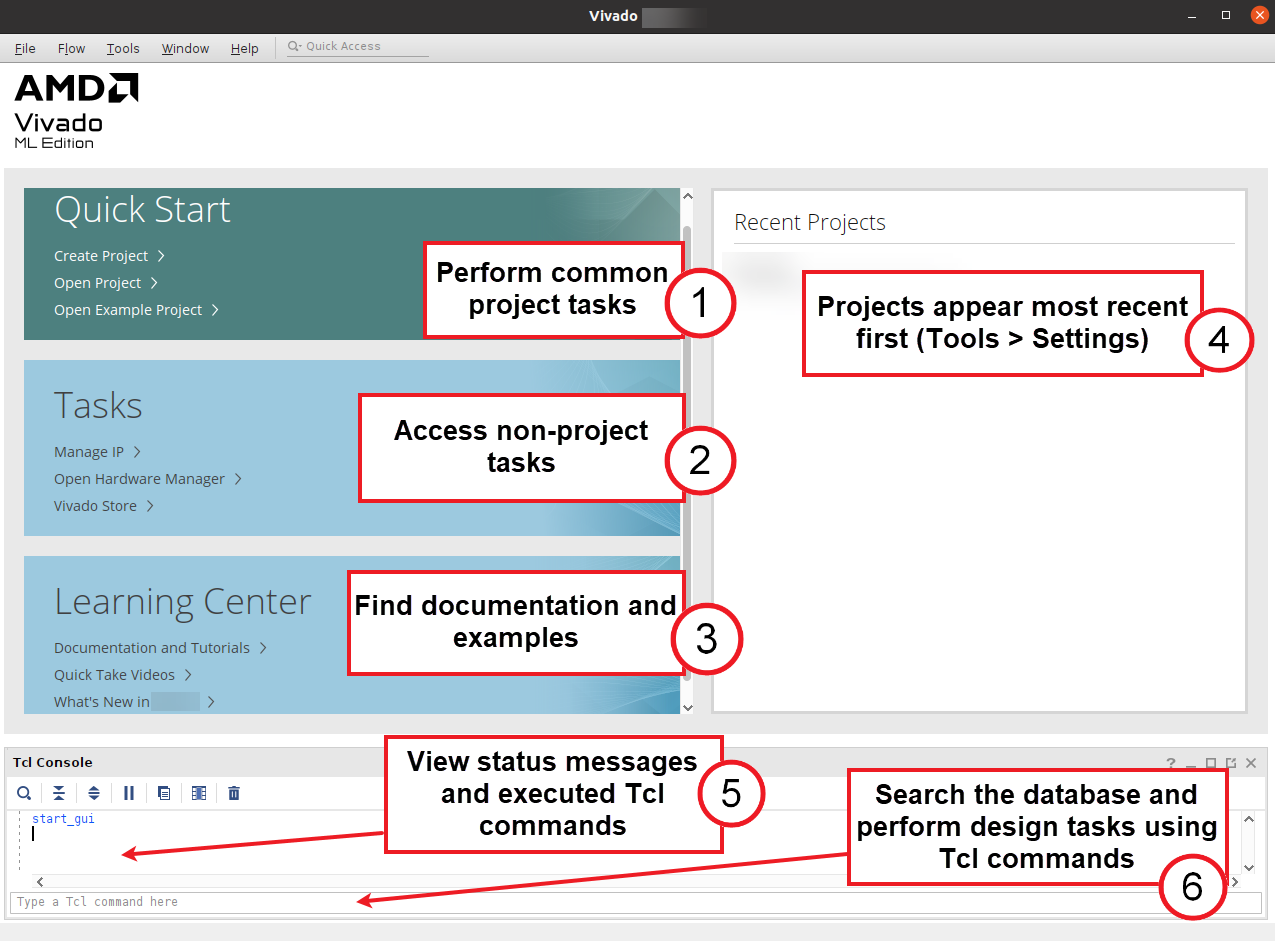


Figure 4-5: Vivado Design Suite Welcome Screen

Hint: If the Tcl Console is not visible, double-click the Tcl tab to make it visible.

This intermediate-level lab assumes that you are familiar with creating a Vivado Design Suite project. Refer to the Lab Reference Guide for more details if necessary.

Here are the following parameters for the project:

* Project name: LEDcntrl
* Platform (board): Any (this lab does not require hardware; additionally, you will be specifying families for the IP to target)
* Project location: <TRAINING\_PATH>/AXIbldPeriph/lab

To simplify and accelerate getting to the interesting portion of the lab, a completer Tcl script has been provided for you in this lab's support directory. Using this completer script, you can build this lab to any step, allowing you to back up to a previous step, or jump ahead to skip material you are already familiar with. Here you will load the completer script.

The Vivado Design Suite offers both GUI and scripted control. Tcl provides scripted control. These Tcl commands can be entered directly into the tool one at a time (or block copied), or an entire Tcl script can be loaded and executed (sourced).

1-2. Run a Tcl script.

1-2-1. Select the Tcl Console tab to view the console.

Hint: If the Tcl Console is minimized, clicking the tab will partially expand the view.

1-2-2. Locate the Tcl command line entry.

The command line entry can be found either on the Welcome page before a project is opened, or once a project has been opened.

From the Welcome screen:

A screenshot of a computer

Description automatically generated

Figure 4-6: Accessing the Tcl Console from the Getting Started Page

You can always click the Tcl tab to maximize/restore it.

From the Tcl Console tab in an open project:

A screenshot of a computer

Description automatically generated

Figure 4-7: Entering Commands into the Tcl Console from an Open Project

Initially, the directory location begins within the tool installation directory.

1-2-3. Enter the following command to change the current working directory to where the Tcl script is located:

cd $::env(TRAINING\_PATH)/AXIbldPeriph/support

Note: The Tcl proc env reaches out to the operating system and returns the value of the environment variable given by TRAINING\_PATH. The $ indicates that the value of that variable should be returned and the two colons (::) indicate from which namespace the information should be pulled. Because there is nothing in front of the ::, the global or base layer of the namespace is to be used.

1-2-4. Verify that you are now where you want to be by entering the following into the Tcl command line:

pwd

This should show that you are in the support directory under the AXIbldPeriph directory in the training directory.

Note that the training directory can be anywhere in the system. What is essential is that you are currently working from the support directory under AXIbldPeriph.

1-2-5. Run the following Tcl command to run the helper Tcl script for this lab:

source AXIbldPeriph\_completer.tcl

The Tcl script is run as though you typed each command included in the Tcl script into the Tcl command line. You can follow the execution of the script and monitor for any errors or warnings in the Tcl Console.

1-3. Now that the Tcl script is loaded, create the top-level project, or dummy project, by using the appropriate Tcl proc.

The project needs to know what language you will be using so that the templates that are created will be created in your preferred language.

1-3-1. Enter the following into the Tcl command line of the Tcl Console:

use VHDL

Only VHDL sources are provided for this lab. All the processes shown in this lab can be performed with Verilog as well.

You have the option of targeting the ZCU104 board (which contains an MPSoC device) or the VCK190 board (which contains a Versal device).

1-3-2. Enter the following into the Tcl command line to select the platform:

[MPSoC users]: use ZCU104

[Versal users]: use VCK190

1-3-3. Enter the following into the Tcl command line to run the proc that will build the project:

projectCreate

1-4. Use the Create and Package IP Wizard to create a new AXI peripheral.

The Create and Package IP Wizard provides a graphical means for creating a design environment platform on which a custom AXI peripheral can be implemented.

The wizard performs the mundane tasks of creating the directory structure (to ensure that the peripheral will appear in the IP catalog), setting up IP parameters (so that the peripheral can be further parameterized), and creating an appropriate AXI interface (master, slave, or stream).

Skeleton structures are also generated for IP HDL, bus functional modeling (BFM), JTAG-based hardware debug design, and a software device driver.

1-4-1. Select Tools > Create and Package New IP.

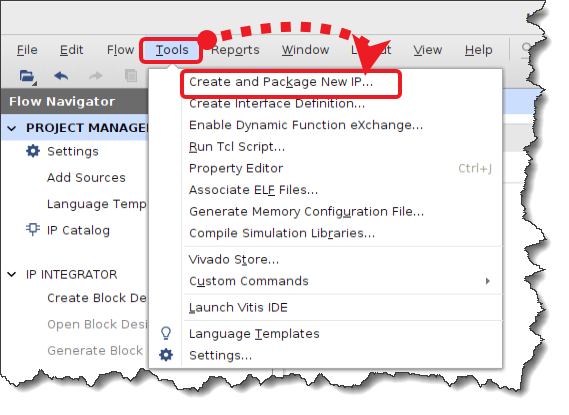


Figure 4-8: Selecting Create and Package New IP

The Create and Package IP Wizard opens. This wizard is used to package existing IP and to create a skeleton for a new AXI-based peripheral. Once packaged or created, the custom IP will appear in the Vivado IP catalog as a selectable item.

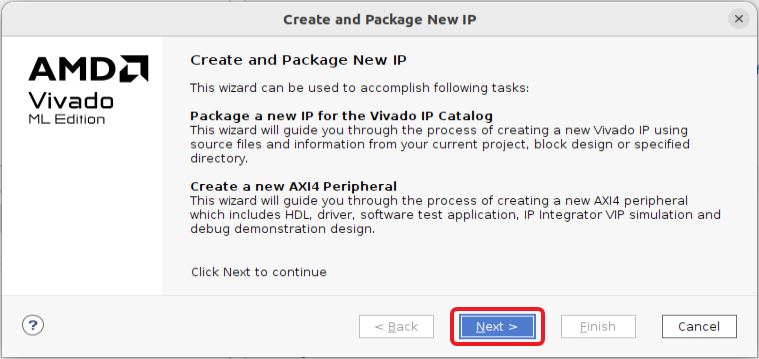


Figure 4-9: Create and Package IP Wizard

1-4-2. Click Next to continue past the welcome screen.

1-4-3. Select the Create new AXI4 peripheral option.

The first several options allow you to package finished designs as IP components that will be placed into the IP catalog.

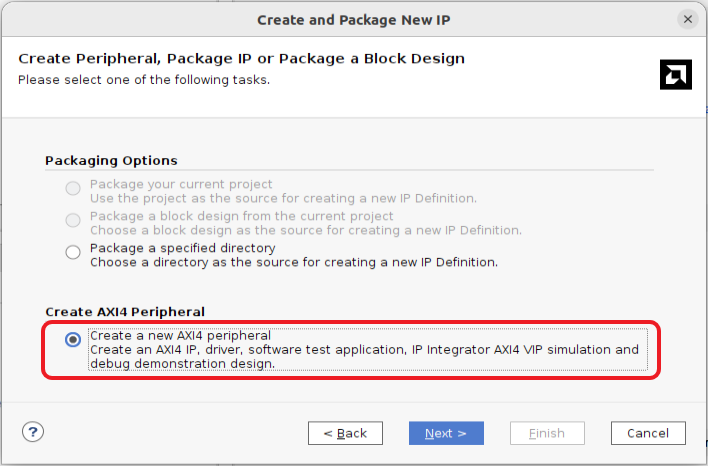


Figure 4-10: Selecting to Create an AXI4 Peripheral

Note: The default location of the IP definition will be within the directory structure of the dummy Vivado Design Suite project.

1-4-4. Click Next to continue to the Peripheral Details dialog box.

1-4-5. Enter LEDcntrl as the peripheral name (1).

1-4-6. Leave the version number at its default of 1.0 (2).

1-4-7. Enter LED Controller as the display name (3).

1-4-8. Enter Simple LED controller example in the Description field (4).

1-4-9. Set the location of the IP to $TRAINING\_PATH/AXIbldPeriph/lab/LEDcntrl (5).

Hint: This is most easily done by removing ip\_repo from the IP location and adding lab/LEDcntrl to the path.

Alternatively, the $TRAINING\_PATH will be automatically expanded for you, and the tools will provide a list for the next level of hierarchy as you type.

1-4-10. Optionally, select the Overwrite existing option (6).

This is convenient if you are iterating through getting a release exported.

A screenshot of a computer

Description automatically generated

Figure 4-11: Setting the Peripheral Details

1-4-11. Click Next to accept the peripheral details and proceed to the Add Interfaces dialog box (7).

1-4-12. Enter S00\_AXI\_LEDs for the interface name (1).

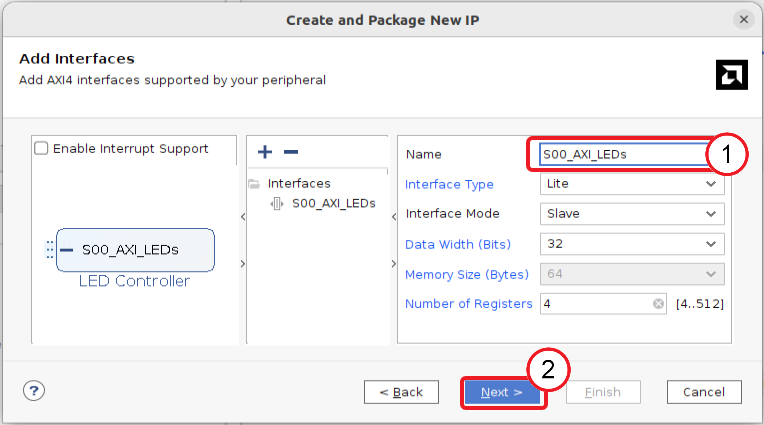


Figure 4-12: Adding the Interface

Leave the remainder of the entries at their default, as a 32-bit AXI4-Lite slave peripheral is what you will be building. If the peripheral required multiple AXI interfaces, you could click the Add Interface button to add them. Each interface would then be assigned unique names and properties.

1-4-13. Click Next to accept the added AXI interfaces and proceed to the Create Peripheral dialog box (2).

1-4-14. Select Edit IP to open the new IP in its own Vivado Design Suite project for further development to complete the development of the custom peripheral (1).

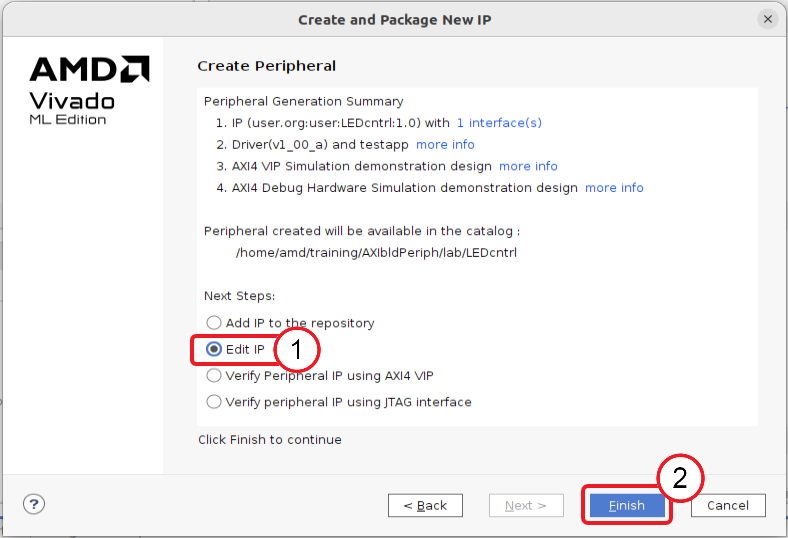


Figure 4-13: Selecting Edit IP

1-4-15. Click Finish (2).

A new Vivado Design Suite project opens. This project will be used as the base design environment where the new peripheral will be built.

Note the Package IP tab that will be used in the next step to customize the IP.

A screenshot of a computer

Description automatically generated

Figure 4-14: Vivado Design Suite Project Generated by the Wizard

Question

How many Vivado Design Suite projects are now open? How do they differ?

Adding HDL Code to the Peripheral Project Step

After this new IP project has been created, user code can be added and attached to the AXI interface, and the automatically generated templates modified to support the user code. This step illustrates how the templates can be customized and how user code can be added.

One aspect of this process will be creating custom user ports and parameters in addition to the basic AXI interface.

You will first begin this step by working in the automatically launched Vivado Design Suite that is now configured to help you build your IP. You will use this project to complete the AXI simple LED controller IP. You should note the process of creating custom user ports and parameters in addition to the basic AXI interface.

The source code is provided as a VHDL file. Since the Vivado Design Suite can support mixed-language synthesis, this does not pose a problem as you can use Verilog as well for your own projects. The only code modification that you will perform is the modification of the generated templates, which is performed in the next step.

Begin by adding the provided source file to the project.

Note: If you are already familiar with the process of adding files, you can return to the Tcl Console tab and enter sourcesAdd LED\_Controller into the Tcl command line to launch the proc that performs this task.

HDL source files can be added to the design at any time.

2-1. Add an HDL source file to the design.

2-1-1. Select Add Sources under the Flow Navigator tab in the Project Manager.

A screenshot of a computer

Description automatically generated

Figure 4-15: Selecting Add Sources

The Add Sources dialog box opens, allowing you to add HDL source files to the project.

2-1-2. Select Add or create design sources (1).

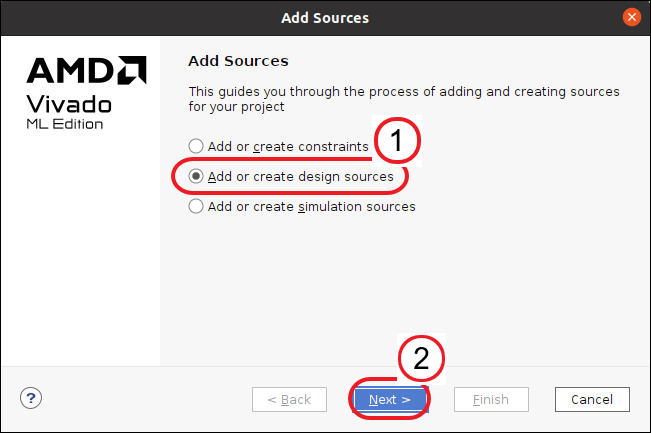


Figure 4-16: Selecting Add or Create Design Sources

2-1-3. Click Next to begin selecting source files (2).

The Add or Create Design Sources dialog box opens and prompts you to add existing HDL source files or to create new HDL sources files.

2-1-4. Click the Plus () icon to open the context menu (1).

2-1-5. Select Add Files to begin adding the source files to the project (2).

2-1-6. Browse to the following directory if it is not open already:

$TRAINING\_PATH/AXIbldPeriph/support

2-1-7. Select LED\_Controller.vhd.

2-1-8. Double-click the source file name in the Add Source Files dialog box to select the file(s) or click OK (3).

2-1-9. Ensure that the Copy sources into project option is selected (when building IP this will be listed as Copy sources into IP Directory) (4).

A screenshot of a computer

Description automatically generated

Figure 4-17: Adding Files

2-1-10. Click Finish in the Add or Create Design Sources dialog box to add the HDL sources to the project (5).

2-1-11. If asked to overwrite the source file, select Overwrite existing files and click OK.

2-2. Open the IP top-level AXI interface and newly added resource HDL files.

2-2-1. Expand LEDcntrl\_v1\_0 under the Sources > Hierarchy pane.

2-2-2. Double-click the following resource files to open them:

* LEDcntrl\_v1\_0.vhd
* LEDcntrl\_v1\_0\_S00\_AXI\_LEDs.vhd
* LED\_Controller.vhd (this is the added source code and is currently only available as VHDL)

Each will be opened in a new tab. HDL files are checked for syntax when they are added to a project. If there is an error, there will be an error generated with a listing of the files that have the incorrect syntax.

A screenshot of a computer

Description automatically generated

Figure 4-18: Open IP Source Files

2-2-3. Examine the contents of each of the files and answer the questions below.

Question

What is the function of each of the files from a design environment standpoint? Which file is the top level of the custom IP?

Question

Examine the file hierarchy. Why is LED\_Controller.vhd not under the top-level file?

2-3. Instantiate the custom user RTL LED\_Controller.vhd into the design top-level and AXI IP components.

You have several choices as to how you can instantiate the LED controller in the top-level code:

* Type the code segments as they appear in the following figures.
* Copy-and-paste the code segments from the provided LED\_Controller\_RTL\_Snippets.txt file from $TRAINING\_PATH/AXIbldPeriph/support.
* Hint: You can open this snippets file by selecting File > Text Editor > Open File to use the Vivado Design Suite's editor.
* Note: Currently only the VHDL version is available for this release. Future releases may contain the Verilog version of the LED controller.
* Copy the entire (completed) contents from the following directory into the editor's LED\_Controller.vhd file:
* $TRAINING\_PATH/AXIbldPeriph/support/golden
* Copy the entire (completed) file using the file browser from the golden directory into the following directory:
* $TRAINING\_PATH/AXIbldPeriph/lab/LEDcntrl/edit\_LED\_cntrl\_v1\_0.srcs/sources\_1/imports/support
* Run the helper Tcl script by entering modifyTemplates into the Tcl command line of the Tcl Console.
* Note: When the script completes, you will need to reload the editor with the updated code. Just click "Reload" in the message bar of the editor.

Important note: The line numbers in the screenshots vary slightly from release to release, so use the line numbers as just a general reference as to where the line(s) of code should be updated.

2-3-1. From the Sources > Hierarchy view, double-click the LEDcntrl\_v1\_0 source file to open it in the editor.

2-3-2. Update the code to add ports and instantiate the LED controller by using one of the methods described above.

* Inclusion of generics which become parameters when this IP is defined (1)
* Creation of the ports that bring in an LED pattern and the port that connects to the LEDs (2)

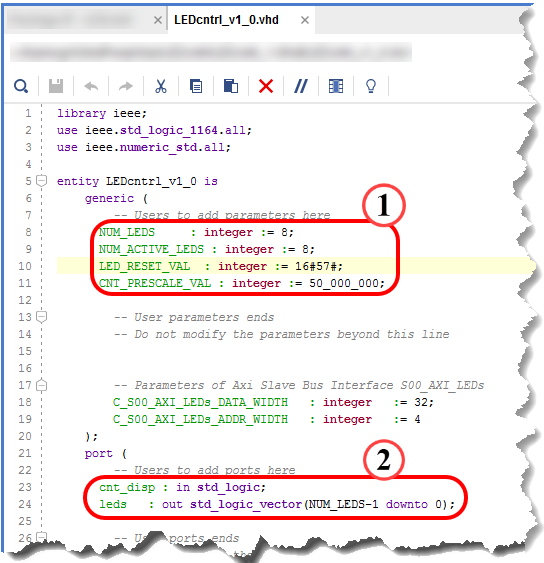


Figure 4-19: Addition to Entity Block

* Addition of external port to component declaration of AXI interface IP to expose the AXI register (3)
* User RTL component declaration (4)
* Additional internal signal declaration for the AXI register (5)
* Addition of external port to component instantiation of AXI interface IP to expose the AXI register (6)

A screenshot of a computer program

Description automatically generated

Figure 4-20: Instantiation Additions

* User RTL component instantiation (7)

A screenshot of a computer

Description automatically generated

Figure 4-21: Instantiating the Peripheral Core

2-3-3. Press <Ctrl + S> to save your work if you manually typed this information in or used the cut-and-paste method.

2-3-4. From the Sources > Hierarchy view, double-click LEDcntrl\_v1\_0\_S00\_AXI\_LEDs.vhd under LEDcontrol\_v1\_0 to open it in the editor.

2-3-5. Cut-and-paste the following sections from the LED\_Controller\_RTL\_Snippets.txt file in $TRAINING\_PATH/AXIbldPeriph/support to LEDcntrl\_v1\_0\_S00\_AXI\_LEDs.vhd.

The wizard generated the AXI4 Lite Slave Attachment logic.

Note that the line numbers shown are approximate.

* Expose internal slave register for user RTL access (8)

A screenshot of a computer

Description automatically generated

Figure 4-22: myip\_led\_ctrl\_v1\_0\_S0\_AXI\_LEDs.vhd File Additions 1

* Map internal AXI register to entity port (9)

A screenshot of a computer

Description automatically generated

Figure 4-23: myip\_led\_ctrl\_v1\_0\_S0\_AXI\_LEDs.vhd File Additions 2

2-3-6. Press <Ctrl + S> to save your work.

2-3-7. Close the LED\_Controller\_RTL\_Snippets.txt file if you used the copy-and-paste method, as you are finished using it.

Use the Vivado synthesis tool to check the syntax of the design. The actual results of this synthesis will not be used here. Instead, whenever this core is instantiated into a project, it will be synthesized as part of that project.

The purpose of checking the syntax here is to identify any problems early (while still developing the core). It is easier to identify synthesis syntax and language errors at this point rather than during your first use of the IP in a design effort.

As always, the completer Tcl script is available to you. Just enter synthesisRun into the Tcl command line of the Tcl Console and the synthesis tool will be run. Alternately, you can follow the manual process shown below.

2-4. Run synthesis.

2-4-1. Expand Synthesis in the Flow Navigator (1).

2-4-2. Click Run Synthesis (2).

A screenshot of a computer

Description automatically generated

Figure 4-24: Running Synthesis

Alternatively, you can also press <F11>.

The Launch Runs dialog box opens.

2-4-3. Ensure that the Launch runs on local host option is selected (1).

2-4-4. Select the largest values from the Number of jobs drop-down list (2).

This allocates the number of processor cores recruited to run synthesis. Generally, the more cores, the faster synthesis completes.

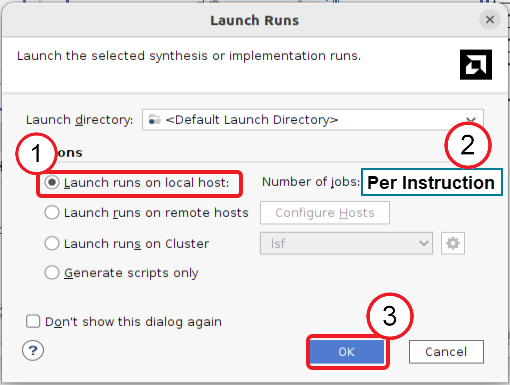


Figure 4-25: Setting the Launch Run Configuration

2-4-5. Click OK to launch the runs (3).

2-4-6. Click Save if you are asked to save your files.

Once synthesis completes, you are asked what task you want to perform next: run implementation, open the synthesized design, view reports, or none of the above.

2-4-7. Select Cancel (1).

2-4-8. [Optional] If you are familiar with accessing these various capabilities, you can disable this dialog box from appearing again by selecting Don't show this dialog again (2).

A screenshot of a computer

Description automatically generated

Figure 4-26: Selecting Post-Synthesis Options

2-4-9. Click OK to take the action you just selected (3) or click Cancel to simply close the dialog box.

2-5. Close the three VHDL source files as they no longer need to be open.

2-5-1. Click the X to the right of each of the following filename tabs to close them:

* LEDcntrl\_v1\_0.vhd
* LEDcntrl\_v1\_0\_S00\_AXI\_LEDs.vhd
* LED\_Controller.vhd

Note: Do NOT close this Vivado Design Suite project as there are a few more things to do to complete the IP.

Importing Ports and Parameters Step

For custom AXI user IP to be included in the IP catalog and used in the Vivado Design Suite and block diagram editor, various Tcl scripts and XML files must be present. The Package IP tab contains a checklist of items required to successfully export an XACT-compliant IP and integrate it into the Vivado Design Suite environment.

You will complete the custom IP creation by indicating the newly added user ports and generic parameters that need to be imported into the Vivado IP packager environment.

Pay particular attention to how user-defined ports and parameters are added. Although beyond the scope of this lab, the IP packager will also allow further control and graphical layout presentation of user-defined parameters. This same process can be later used to edit, add, or delete parameters or ports at which time the Tcl and XML IP project files will be updated.

Note the different check marks associated with the Package IP tab. Green is a finished item, while red indicates that required work is needed on a topic. The Vivado IP packager is sensitive to new project files and changes in HDL source code that add parameters and ports.

You may have noticed that all checks were originally green when the skeleton peripheral was first created, and some green-checked items are now marked as edited as a result of modifying and adding files, parameters, and ports to the project and design.

A screenshot of a computer

Description automatically generated

Figure 4-27: Package IP Tab - Checks

Items with a green check can be changed to expand or modify peripheral features. Items with a red check require attention. In many cases the Vivado IP packager will provide a link to automatically perform the required update. All of this is illustrated in the following steps.

3-1. Configure family and life cycle support.

Since this LED controller design uses logic resources common to all AMD devices, you can target it to any or all devices.

For the purpose of this lab, you will now configure this piece of IP to be supported by MPSoC and Versal devices rather than just the family you selected during the project creation. This will ensure that your IP will show up in the IP catalog regardless of which device family is selected for the user design.

Along with the family selection, you should set the Life Cycle property. The Life Cycle property indicates the use status of the IP.

For this peripheral, this status will be set to production (rather than beta, pre-production, discontinued, superseded, hidden, or removed).

A status other than production will show on the component's schematic symbol when it is later instantiated in the block diagram editor.

3-1-1. Select the Package IP - LEDcntrl tab.

Note: If this tab is not visible, you can open it by selecting Flow Navigator > PROJECT MANAGER > Edit Packaged IP.

3-1-2. Select Compatibility from the Packaging Steps on the left as this provides the list of compatible AMD product families that will be supported by this IP.

Often, the family that was originally selected has the IP marked as Pre-Production.

3-1-3. Click the field under the Life Cycle column next to any family member not marked for production (1).

3-1-4. Select Production from the drop-down list (2).

A screenshot of a computer screen

Description automatically generated

Figure 4-28: Selecting Production

Note: The figure above shows the view when a Versal device was originally selected; that is, the Versal family needs to be updated to Production.

3-2. The File Groups step allows you to keep track of files used in the peripheral design, simulation model, and software drivers. Since a source file was added to the design, this must be updated.

3-2-1. Select File Groups as this is where the list of source files in the IP design is managed.

3-2-2. If presented, click the Merge changes from File Groups Wizard link to update the project files list to include the newly added LED\_Controller.vhd source.

You can also right-click the related file group and add files manually.

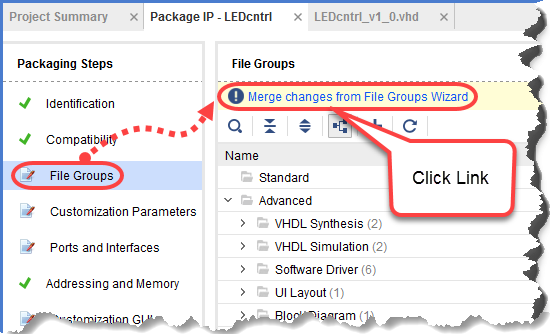


Figure 4-29: Updating File Groups

3-2-3. Expand the VHDL Synthesis and VHDL Simulation groups under the Advanced branch and ensure that the source file is shown in both groups.

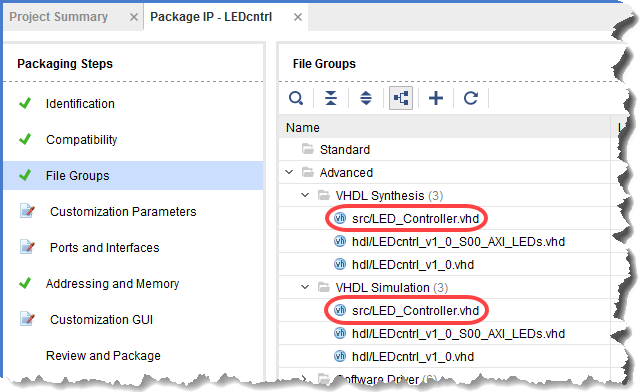


Figure 4-30: Verifying Source File Added to Project

If the LED\_Controller.vhd file is NOT listed, you will need to add it manually with the following process. Otherwise, continue with step 3-3.

3-2-4. Right-click VHDL Synthesis to open the context menu.

3-2-5. Select Add Files to open the Add Files Wizard.

3-2-6. Click Add Files and browse to the $TRAINING\_PATH/AXIbldPeriph/lab/LEDcntrl/LEDcntrl\_1.0/src directory.

3-2-7. Select VHDL file (.vhd, vhdl, vhdp) from the Files of type drop-down list.

3-2-8. Double-click LED\_Controller.vhd to add it to the VHDL Synthesis list.

3-2-9. Repeat tasks 3-2-4 through 3-2-8 for the VHDL Simulation group (instead of VHDL Synthesis).

3-2-10. Click OK to exit the Add IP Files process.

3-3. Import the added user generics into the IP packager. This will make the generic parameters visible in the schematic symbol when the IP is instantiated from the IP catalog.

Parameters can also be added manually by right-clicking in the IP configuration pane and selecting the appropriate add function. The newly added generic parameters can be configured in each instance of this IP when it is later instantiated in a block diagram design.

3-3-1. Select Customization Parameters from the Package IP tab and make sure that the Customization Parameters list is expanded in the Customization Parameters pane.

The default parameters will be displayed.

Note that the only user-controllable parameters shown are the defaults provided by the skeleton RTL.

3-3-2. Click the Merge changes from Customization Parameters Wizard link.

A screenshot of a computer

Description automatically generated

Figure 4-31: Initial IP Customization Parameters

3-3-3. Expand Hidden Parameters and verify that the custom added generic parameters are present.

Newly added parameters are hidden by default, meaning that they will not show in the IP Configuration GUI to be available for modification. You will make them visible shortly.

A screenshot of a computer

Description automatically generated

Figure 4-32: User Parameters Now Visible

3-4. View the newly added user ports available in the IP packager. These user ports will be present on the block diagram IP's schematic block.

This action was completed automatically when the custom user generic parameters were imported in the previous step. If this was not the case, a wizard link, similar to the one for importing parameters would have been visible.

Ports and interfaces can be added manually by right-clicking in the pane and selecting the appropriate add function.

3-4-1. Select Ports and Interfaces to view the list.

3-4-2. Verify that the custom added ports were recognized.

A screenshot of a computer

Description automatically generated

Figure 4-33: User Ports Now Visible

3-5. Update the IP Schematic GUI menu for the user-added generic parameters and ports.

You will also configure the schematic GUI to display parameters on a custom pane as part of this process.

3-5-1. Select Customization GUI to access the controls for the generation of the schematic symbol and IP configuration GUI.

A screenshot of a computer

Description automatically generated

Figure 4-34: GUI Customization - Merge Changes

Note that the added custom parameters show as hidden in the pane. They must be added to a configuration GUI page to become visible on the schematic symbol.

You will begin by adding a new page to the schematic symbol GUI and then add each of the custom user-defined parameters individually.

Also note that the skeleton parameters that are defined for the AXI interface are contained in a page named Page 0.

3-5-2. From the Layout pane in the Customization GUI, expand the Window field.

3-5-3. Right-click Window in the Layout window to open the context menu.

3-5-4. Select Add Page to add an additional Page to the Window entries.

This page will be used for adding LED controller-specific parameters for later user customization.

A screenshot of a computer

Description automatically generated

Figure 4-35: Adding a New Parameter Page

3-5-5. Enter LED Controller Parameters in the Display name field.

A screenshot of a computer

Description automatically generated

Figure 4-36: Entering the New Page Display Name

3-5-6. Click OK.

Now that a new page has been created, add parameters to it.

3-5-7. Click and drag each of the four parameters from the Hidden Parameters list to the LED Controller Parameters list.

Alternatively, you can select all of the Hidden Parameters using multi-select and drag them as a group into the LED Controller Parameters field.

A screenshot of a computer

Description automatically generated

Figure 4-37: Drag & Drop User Parameters to New Page

Note that you can select the parameters that you just moved (individually or as a group) and use the up/down arrows to order them for your preferred viewing in the IP Configuration GUI.

A screenshot of a computer

Description automatically generated

Figure 4-38: Arranging User Parameters on the New Page

As previously mentioned, the Page 0 parameter page contains the default AXI interface parameters. These AXI parameters are typically not configured as they must follow strict rules so that the AXI ports can be connected to other devices; therefore, this page is not really needed. It can be deleted so as not to clutter the schematic symbol GUI.

3-5-8. Right-click the Page 0 list to select it and open the context menu.

3-5-9. Select Remove Page.

A screenshot of a computer

Description automatically generated

Figure 4-39: Removing Unused Parameter Page

3-5-10. Click OK to confirm removing the page.

Notice that the Page 0 parameters are now part of the hidden list. All that should remain visible under the LED controller parameters are the user-defined parameters on their own user-defined page in the order that they were positioned in when created or arranged via the up and down arrows in the Layout window.

A screenshot of a computer

Description automatically generated

Figure 4-40: Completed GUI Customization

3-6. Re-package the IP to save the changes made. This will update all the necessary scripts and files for the IP catalog.

3-6-1. Select Review and Package.

A screenshot of a computer

Description automatically generated

Figure 4-41: Review and Package IP

3-6-2. Click Re-Package IP.

The script files are created (first use) or updated (when the IP is modified).

A dialog box should appear asking to close the project.

3-6-3. Click Yes to close the project.

This Vivado Design Suite projects closes, but the LEDcntrl project will still be open.

This dialog box only appears the first time that the IP is packaged. If it does not appear, it is probably because the project was previously closed and reopened. You can close the project by selecting File > Exit.

Now that the IP is created and packaged, you can now import the IP into a project.

IP can be imported from a number of tools provided that they adhere to the IP-XACT standard. Once created, the Vivado Design Suite must be made aware of the presence of the IP. This is typically performed from an open project.

The instructions below describe the process of importing a single IP; however, the steps can be repeated as necessary to collect different IPs from various locations.

3-7. Import LEDcntrl into the open project.

3-7-1. Using the Flow Navigator, select Project Manager > Settings (1).

The Project Settings Dialog box opens with the General tab open.

3-7-2. Expand IP (2) and click Repository to access the IP settings (3).

A screenshot of a computer

Description automatically generated

Figure 4-42: Accessing the Project's IP Settings

3-7-3. Click the Plus icon () to open the file browser to point to the IP repository in which your IP is located (1).

3-7-4. Browse to the repository where you created the IP ($TRAINING\_PATH/AXIbldPeriph/lab).

3-7-5. Click Select.

IMPORTANT: If the IP repository that you have selected has the IP in zip format, then you will need to continue as described below. If, however, your IP has been expanded (for example, if you are revisiting this step or adding another piece of IP), then it will automatically appear in the IP in Selected Repository field. You can simply click OK to complete the repository inclusion process.

The selected IP repository is scanned for all IP that is listed in the Select IP To Add To Repository dialog box.

A screenshot of a computer

Description automatically generated

Figure 4-43: Adding the IP Repositories and the Specific IP

A window opens, showing the number of IPs being added to the project.

3-7-6. Click OK to close the Add Repository dialog box.

3-7-7. Click OK again to exit the project settings.

You have successfully completed building a custom AXI IP peripheral. Now verify that the IP is available for use.

3-8. Verify that the IP was created properly.

3-8-1. Locate the Vivado Design Suite in the operating system task bar and click it to bring the tool into view if the LEDcntrl project is not visible.

The next set of steps is straightforward: create a new block design, open the IP catalog, locate the new IP, and add it to the canvas. This process should be quite familiar to you by now, so a Tcl proc has been created to remove this tedium for you.

3-8-2. Enter the following into the Tcl command line of the Tcl Console:

IPverify

This will create a new block design and add the LED IP that you just created.

The schematic symbol appears in the block diagram and is ready to be included in your design.

A screenshot of a computer

Description automatically generated

Figure 4-44: Block Diagram with Symbol

3-9. View the re-customization options for the IP.

3-9-1. Double-click the IP symbol in the Diagram tab to open the LED controller's Re-customize IP dialog box.

All of the user generic parameters are present and can be adjusted for each instance of the IP.

A screenshot of a computer

Description automatically generated

Figure 4-45: Re-customization of the IP

The user's custom AXI IP built in this lab is now in the IP catalog and can be added to a block diagram and its parameters modified.

3-9-2. Click Cancel to close the Re-customize IP dialog box.

3-10. Close the Vivado Design Suite.

3-10-1. Select File > Exit.

The Exit Vivado dialog box opens.

A screenshot of a computer

Description automatically generated

Figure 4-46: Exit Vivado Dialog Box

3-10-2. If you are asked to save the project or a portion of the project, select whichever elements of the project you want to save, then click Save to save the selected elements; otherwise, click Don't Save.

3-10-3. Click OK when you are asked to exit the Vivado Design Suite.

Note: You can choose to select the Don't show this dialog again option to avoid being asked for confirmation when exiting the Vivado Design Suite.Click Don't Save when exiting as the demo block diagram is not needed.

3-10-4. Answer the questions below to enhance your understanding of the operation of the Vivado Design Suite regarding custom IP.

These topics were not specifically addressed in this lab but are important in the design environment.

Question

What is the procedure when custom IP, AXI, or any other aspect of the IP, needs to be later updated or the HDL changed in the Vivado IP design project?

Question

What is the best practice to follow when updating IP?

Some systems (particularly VMs) may be memory constrained. Removing the workspace frees a portion of the disk space, allowing other labs to be performed.

You can delete the directory containing the lab you just ran by using the graphical interface or the command-line interface. You can choose either mechanism. Both processes will recursively delete all the files in the $TRAINING\_PATH/AXIbldPeriph directory.

3-11. [Optional] [Only for local VMs—not for CloudShare] Clean up the file system.

Using the GUI:

3-11-1. Navigate to $TRAINING\_PATH/AXIbldPeriph.

3-11-2. Select AXIbldPeriph.

3-11-3. Press <Delete>.

-- OR --

[Linux users]: Using the command line:

3-11-4. Press <Ctrl + Alt + T> to open a terminal window.

3-11-5. Enter the following command to delete the contents of the workspace:

[host]$ rm -rf $TRAINING\_PATH/AXIbldPeriph

## Summary

Use the Create and Package IP Wizard in the Vivado Design Suite to design your custom AXI peripheral and add it to the IP catalog. It will build a skeleton AXI interface design environment that your user logic can be added to.

The wizard also creates the necessary folder structure, skeleton HDL files, and adds the necessary Tcl scripts to the project directory, making the IP accessible from the IP catalog and usable in the block diagram editor.

After creating a peripheral, use the resulting Vivado Design Suite project that the Create and Package IP Wizard generated to add additional files and code to the peripheral design. Modify the skeleton HDL files to bring up any user-defined parameters and I/O to the top-level project file.

The Vivado synthesis tool can also be used to check the design HDL code syntax. Lastly, the Package IP tab will be used to import user-added ports and parameters to the IP Catalog symbol and customization GUI.

## Answers

1. How many Vivado Design Suite projects are now open? How do they differ?

There are two Vivado Design Suite projects now open. The originally created project, placeholder\_project.xpr, is just a dummy project that is created only so that the Create and Package IP Wizard can be launched. After that, this project is not used.

The Create and Package IP Wizard created the IP Vivado project, edit\_[your ip name].xpr, that will be used as the design environment for building the custom IP.

1. What is the function of each of the files from a design environment standpoint? Which file is the top level of the custom IP?

LEDcntrl\_v1\_0.vhd: Top-level file of the design. This file was generated by the Create and Package IP Wizard. It houses the AXI port interface and user logic RTL. This file is modified to include user RTL components. User ports and generic parameters are added to its entity statement.

LEDcntrl\_v1\_0\_S00\_AXI\_LEDs.vhd: AXI port interface logic. This file was generated by the Create and Package IP Wizard. The contents of this file are generated based on the type of AXI interface and options chosen in the wizard. User modification is necessary to expose internal registers and needed AXI signals via its entity statement to communicate to the user logic component in the top-level entity.

LED\_Controller.vhd: Custom user RTL. In this lab, this file is provided for you, as it is not generated by the Create and Package IP Wizard.

1. Examine the file hierarchy. Why is LED\_Controller.vhd not under the top-level file?

The user RTL is yet to be instantiated as a component into the design. This will be performed in the next steps of the design.

1. What is the procedure when custom IP, AXI, or any other aspect of the IP, needs to be later updated or the HDL changed in the Vivado IP design project?

The HDL code for the IP is modified in the edit\_LEDcntrl\_v1\_0.xpr Vivado Design Suite project. In the Package IP tab, under Review and Package, use the Re-Package IP selection to update the IP catalog. When the block diagram (in the Vivado Design Suite project that uses the IP) is opened, a message will be reported that the instance needs to be updated. Its schematic symbol will be locked to any changes in parameters until the instance is updated.

1. What is the best practice to follow when updating IP?

When IP is updated, it is best practice to change the version in the Identification section of the Package IP tab.

# 