Building Custom AXI IP

2016.3

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.

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 Xilinx 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 integrated design environment (IDE). 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 IDE project file that can be used as a design environment.

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

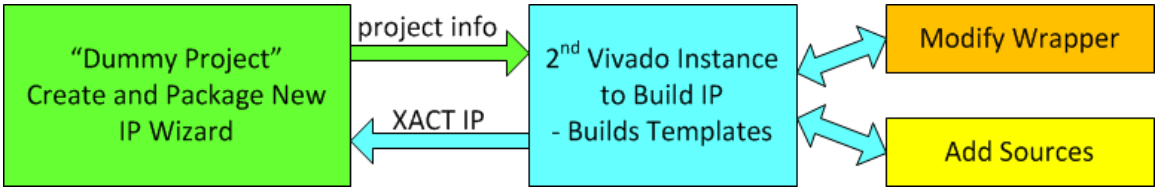

Figure 4-1: How the Instances of the Vivado IDE Are Used

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.

You will use the Create and Package IP Wizard to create the user (peripheral top-level AXI slave interface) skeleton files and then add the custom LED controller user logic (provided).

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.

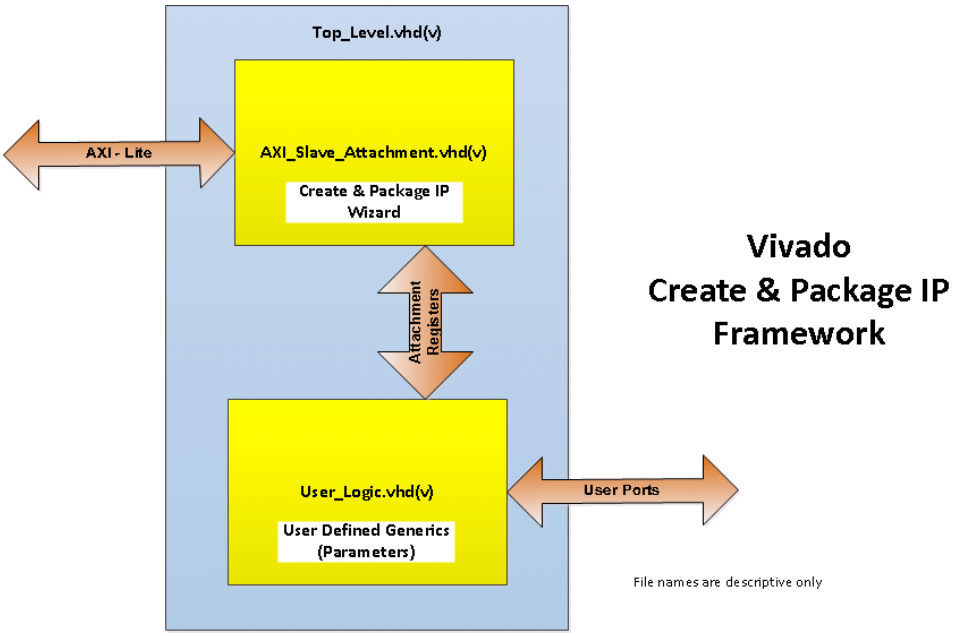

Figure 4-2: Create and Package IP Block Diagram

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.

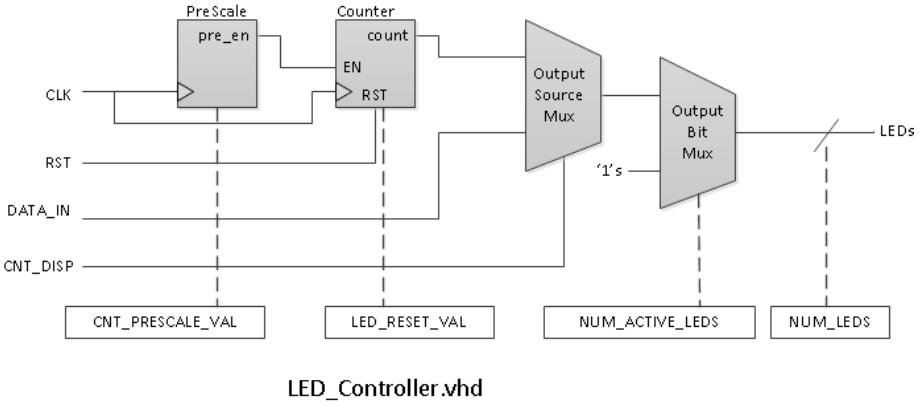

Figure 4-3: LED Controller Block Diagram

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 one can actually see LEDs blink in 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 it is 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.

General Flow

Creating the Vivado IDE Project and Running the Create and Package IP Wizard Step 1

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 as a platform to launch the Create and Package IP Wizard.

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

There are a number of ways to launch the Vivado Design Suite. The two most popular mechanisms are shown here.

1-1. Launch the Vivado Design Suite.

This can be done in two standard ways, use your preferred method.

1-1-1. Select **Start** > **All Programs** > **Xilinx Design Tools** > **Vivado 2016.3** > **Vivado 2016.3**.

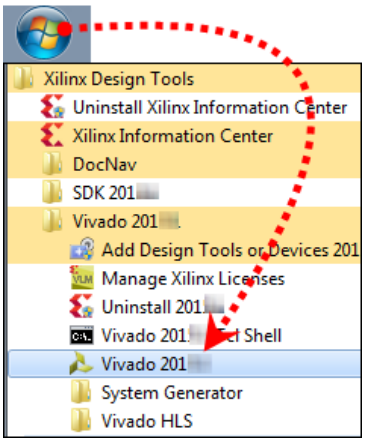

Figure 4-4: Launching the Vivado Design Suite from the Start Menu

-- OR --

Double-click the **Vivado Design Suite** shortcut icon () on the desktop.

The Vivado Design Suite opens to the Welcome window. From the Welcome window you can create a new project, open an existing project, or enter Tcl commands directly into the Vivado Design Suite as well as access documentation and examples.

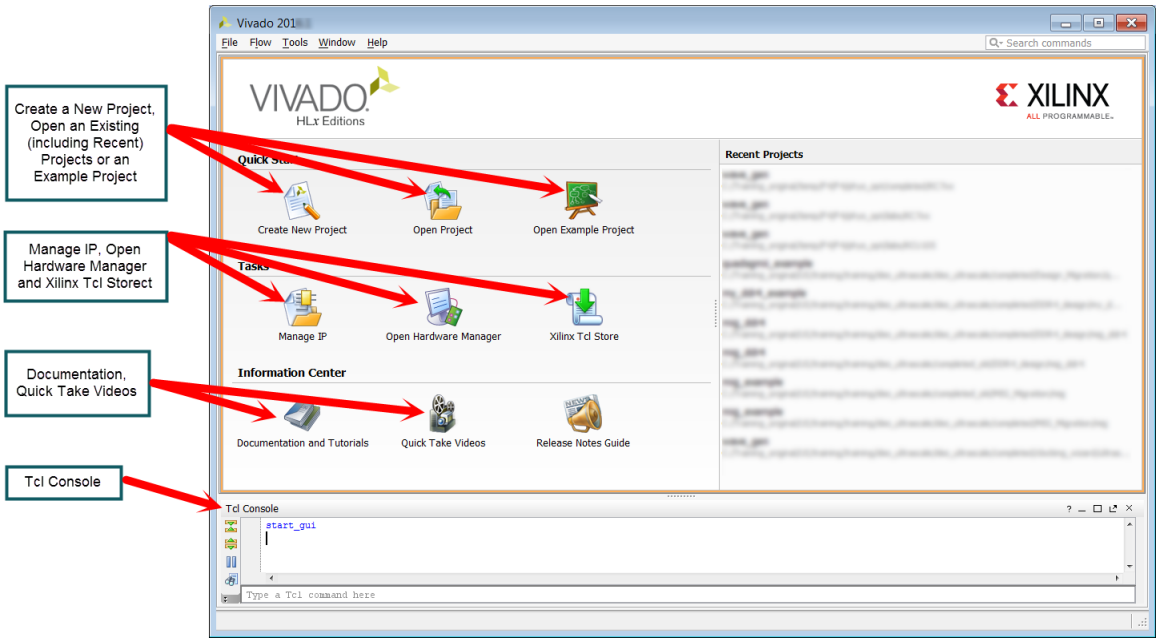

Figure 4-5: Vivado Design Suite Welcome Screen

Since this is an intermediate-level lab, you are expected to know how to create a Vivado Design Suite project.

Here are the following parameters for the project:

- Project name: **bldLEDperiph**
- Platform (board): Any (you will be specifying families for the IP to target later in this lab)
- Project location: *C:\training\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. Scripted control takes the form of Tcl commands. These Tcl commands can be entered directly into the tool one at a time, or an entire Tcl script can be loaded and executed.

1-2. Run a Tcl script.

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

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

From the Welcome screen:

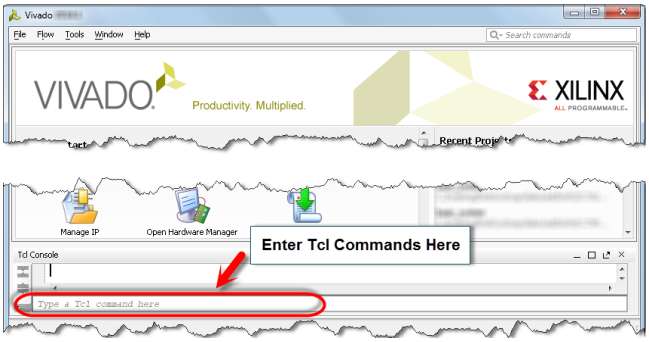

Figure 4-6: Accessing the Tcl Console from the Getting Started Page

From an opened project:

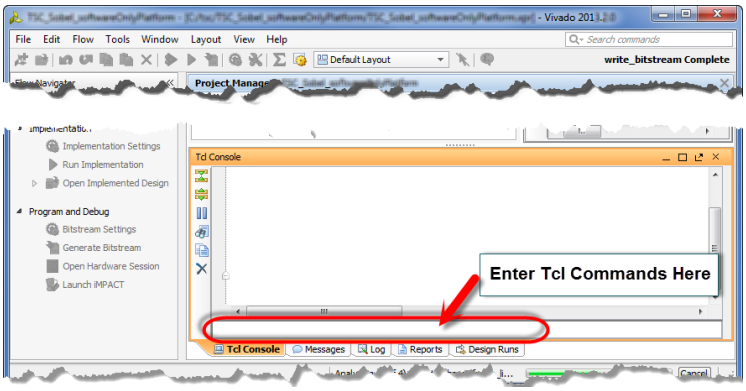

Figure 4-7: Entering Commands into the Tcl Console from an Open Project

The default directory for the Tcl environment is nested within the Xilinx installation directory. This placement, however, is often disadvantageous. In most cases, you will want to navigate to a more useful path. To do this, use the `cd` command to change directory to the user directory.

1-2-2. Change the current working directory to where the Tcl script is located by entering:

```
cd C:\training\AXIbldPeriph\support
```

Remember that the Tcl environment is based on Linux and requires the '/' character to delimit hierarchical paths.

1-2-3. Verify that you are now where you want to be by entering the following into the Tcl command line:

```
pwd
```

The current working directory is displayed. If you are not where you want to be, use the `cd` command to change to `C:\training\AXIbldPeriph\support`.

1-2-4. Enter the following Tcl command:

```
source AXI_buildPeriph_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 <language>
```

Where `<language>` is either `VHDL` or `Verilog`. This is not case sensitive.

Alternately, you can set your language preference by clicking **Flow Navigator** > **Project Manager** > **Project Settings** > **General** > **Target Language** and selecting your preferred language.

1-3-2. Enter the following into the Tcl command line to run the proc that will build the project:

```
createProject
```

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 IP**.

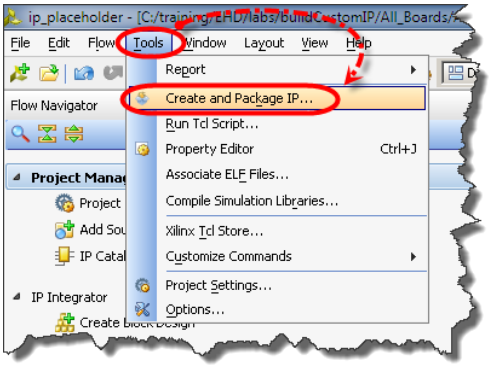

Figure 4-8: Selecting Create and Package IP

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.

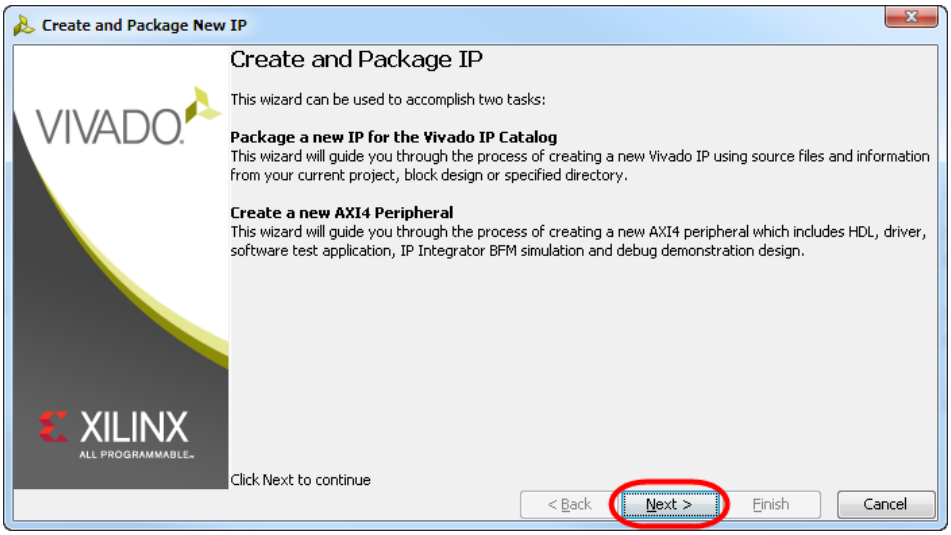

Figure 4-9: Create and Package IP Dialog Box

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

1-4-3. Select the **Create new AXI peripheral** option.

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

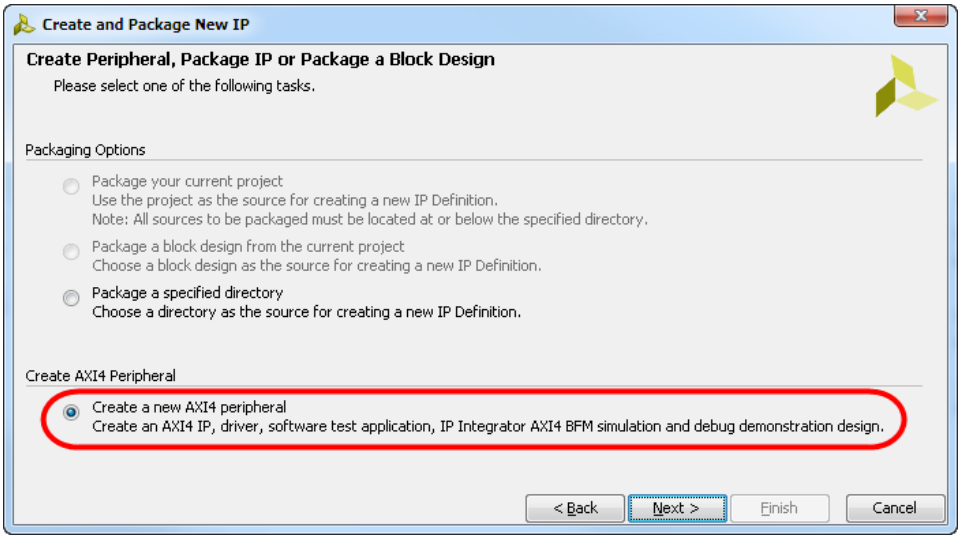

Figure 4-10: Create Peripheral, Package IP or Package a Block Design Dialog Box

Note that the default location of the IP definition will be within the directory structure of the dummy Vivado IDE project.

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

1-4-5. Enter **LEDcntrl** as the peripheral name.

1-4-6. Enter **LED Controller** as the display name.

1-4-7. Enter **Simple LED controller example** in the Description field.

1-4-8. Set the IP's location to **C:\training\AXIbldPeriph\lab\LEDcntrl**.

Note that the default IP location *(../ip_repo)*, is one directory level above the current Vivado Design Suite project. This is where the IP will be located and customized. Currently this directory does not exist. You can edit the *IP location* path provided to replace this default directory with **/LEDcntrl**.

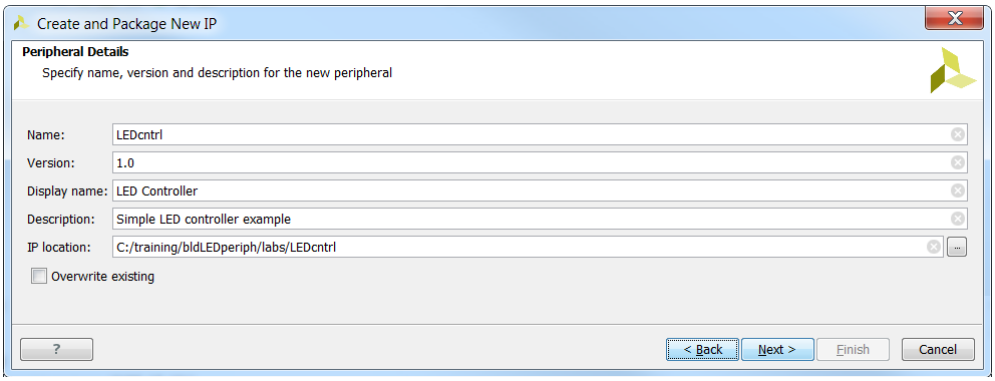

Figure 4-11: Peripheral Details Dialog Box

1-4-9. Click **Next** to accept the peripheral details and proceed to the Add Interfaces dialog box.

1-4-10. Enter **S00_AXI_LEDs** for the interface name.

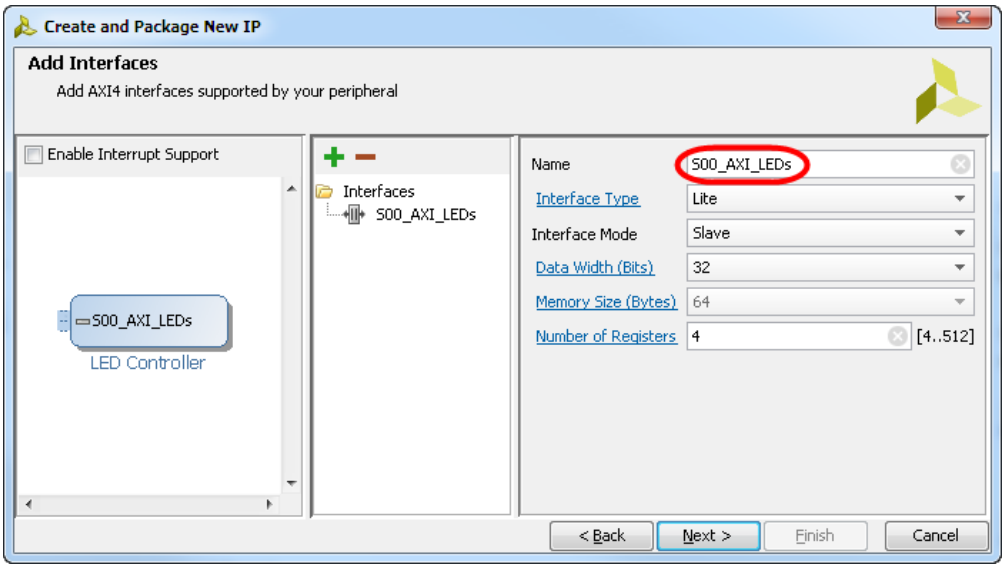

Figure 4-12: Add Interfaces Dialog Box

Leave the remainder of the entries at their default, as a 32-bit AXI 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.

1-4-11. Click **Next** to accept the added AXI interfaces and proceed to the Create Peripheral dialog box.

1-4-12. Select **Edit IP** to open the new IP in its own Vivado IDE project for further development to complete the development of the custom peripheral.

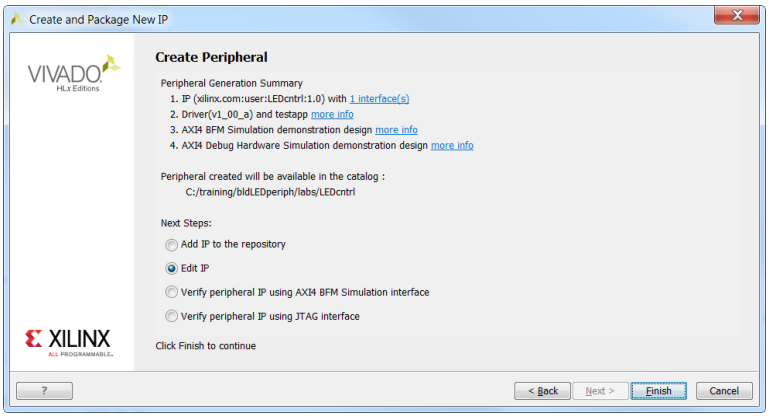

Figure 4-13: Summary Dialog Box

1-4-13. Click **Finish**.

A new Vivado IDE 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.

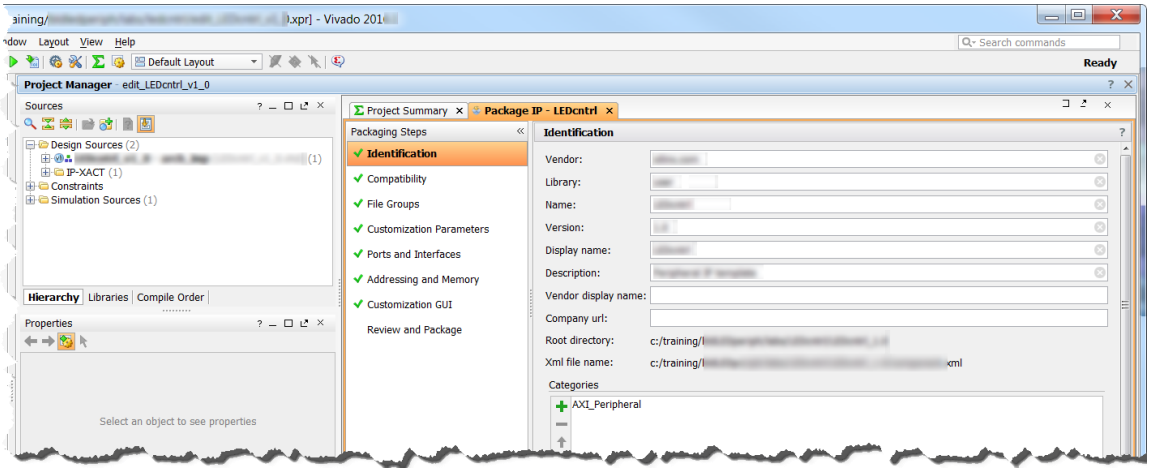

Figure 4-14: Wizard-Generated Vivado IDE Project

Question 1

How many Vivado IDE projects are now open? How do they differ?

Adding User HDL to the Peripheral Project Step 2

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 will illustrate how the templates can be customized and how user code can be added.

One of the aspects 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 *edit* 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 is capable of supporting mixed-language synthesis, this does not pose a problem. Additionally, the only code modification that you will perform is the modification of the generated templates (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 enter `addSources` into the Tcl command line of the Tcl Console to launch the proc to perform 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.

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**.

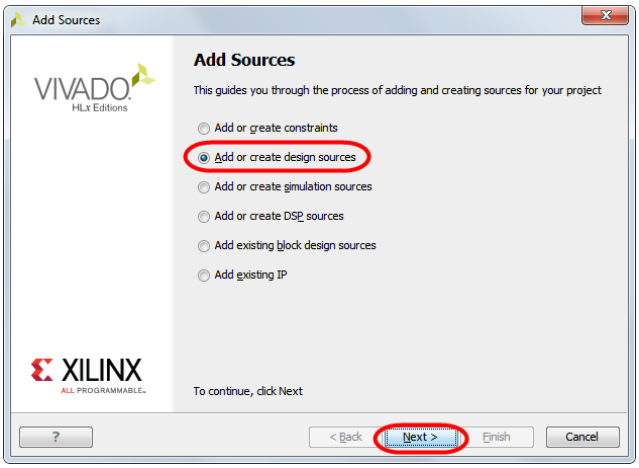

Figure 4-16: Selecting Add or Create Design Sources

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

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 and select **Add Files**.

2-1-5. Browse to the *C:\training\AXIbldPeriph\support* if it is not open already.

2-1-6. Select **LED_Controller.vhd**.

2-1-7. Double-click the source file name in the Add Source Files dialog box to select the file(s) or click **OK**.

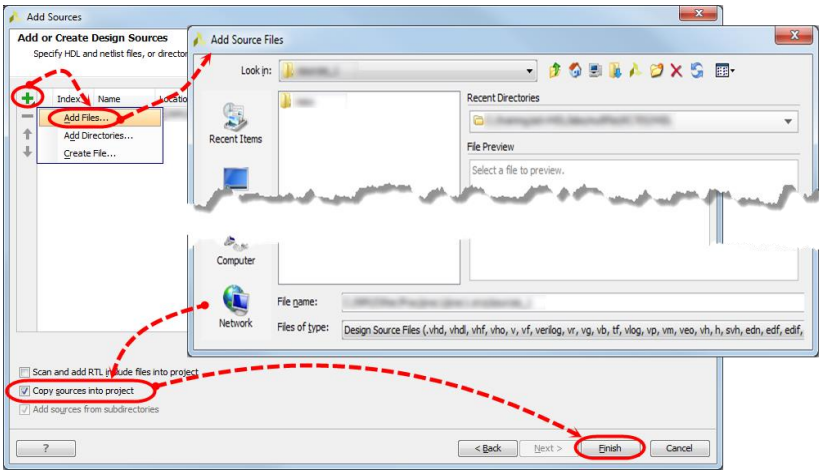

Figure 4-17: Selecting Add Files

2-1-8. Ensure that the **Copy sources into project** option is selected (when building IP this will be listed as **Copy sources into IP Directory**).

2-1-9. Click **Finish** in the Add or Create Design Sources dialog box to add the HDL sources to the project.

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.[v | vhd]**
- **LEDcntrl_v1_0_S00_AXI_LEDs. [v | 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.

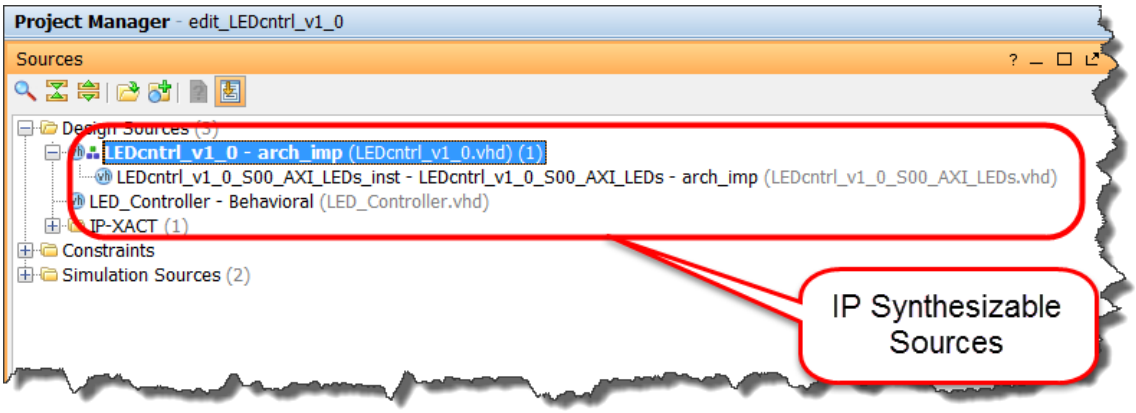

Figure 4-18: Open IP Source Files

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

Question 2

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 3

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

- **Type the code as it appears in the screenshots below.**
- **Cut-and-paste the code from the provided *LED_Controller_RTL_Snippets.txt* file in the *C:\training\AXIbldPeriph\support\bldLEDperiph* directory.**

Note: You can select File > Open File to use Vivado Design Suite's editor.

Also note that only the VHDL version is available for the 2016.3 release. Future releases may contain the Verilog version of the LED controller.

- **Run the helper Tcl script by entering `modifyTemplates` into the Tcl command line of the Tcl Console.**

Note that 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.

2-3-1. 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)

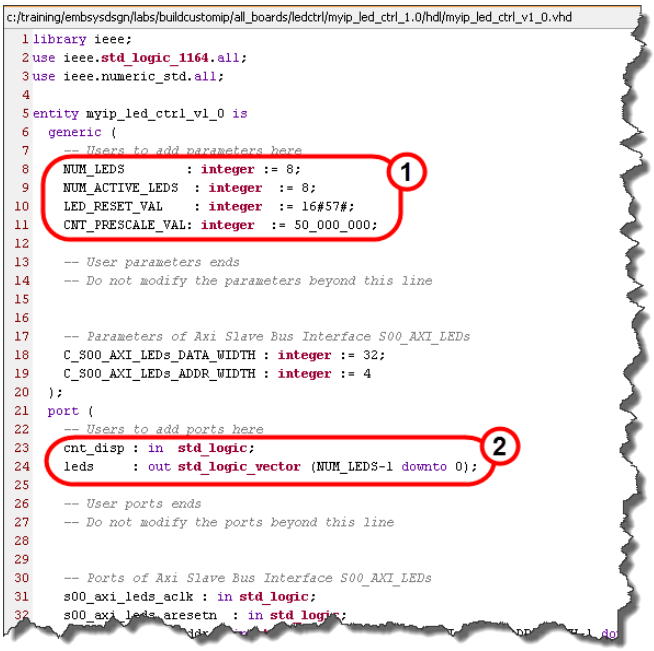

Figure 4-19: Addition to Entity Block

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

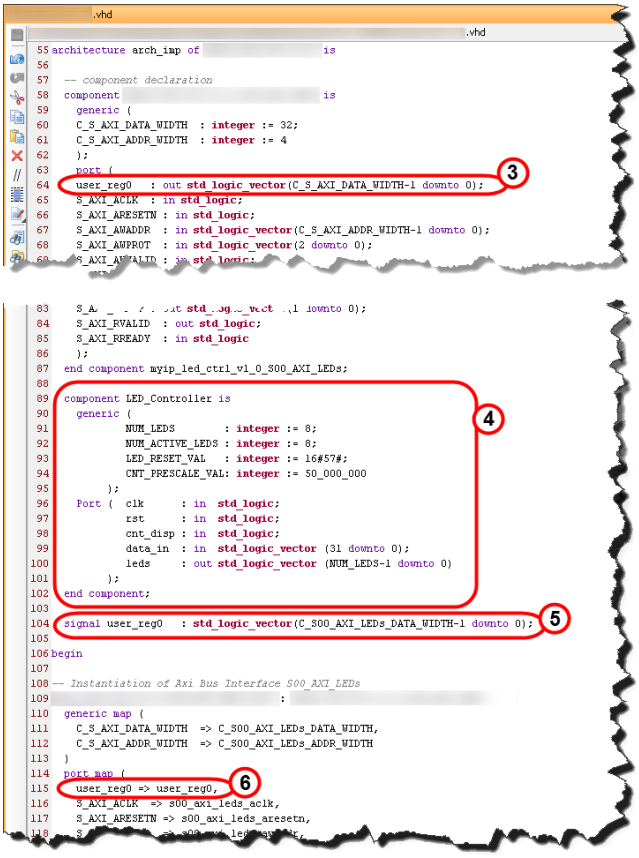

Figure 4-20: Instantiation Additions

- User RTL component instantiation (7)

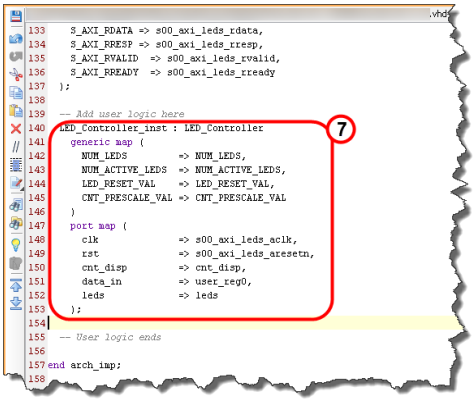

Figure 4-21: Instantiating the Peripheral Core

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

2-3-3. Cut-and-paste the following sections from the *LED_Controller_RTL_Snippets.txt* file in the *C:\training\AXIbldPeriph\support\bldLEDperiph* directory 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)

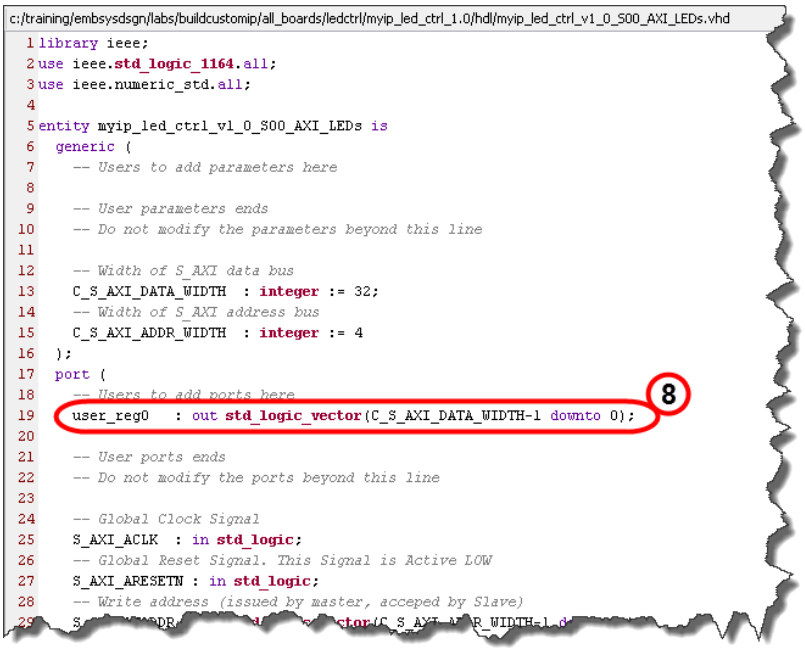

Figure 4-22: myip_led_ctrl_v1_0_S0_AXI_LEDs.vhd File Additions 1

- Map internal AXI register to entity port (9)

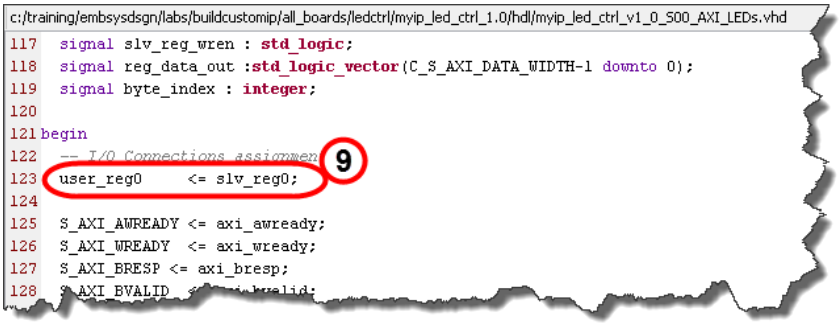

Figure 4-23: myip_led_ctrl_v1_0_S0_AXI_LEDs.vhd File Additions 2

2-3-4. Press <**Ctrl** + **S**> to save your work.

2-3-5. Close the **LED_Controller_RTL_Snippets.txt** file if you used the cut-and-paste method, as you will not need it any more.

Use the Vivado synthesis tool to checking 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 `runSynth` 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. Click **Run Synthesis** in the Flow Navigator under Synthesis.

Alternatively, you can also select **Flow** > **Run Synthesis** or press <**F11**>.

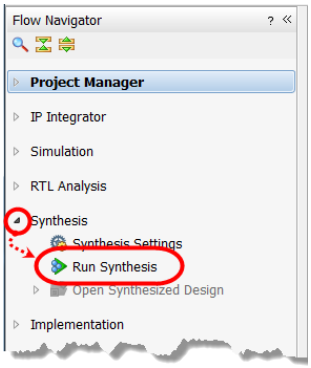

Figure 4-24: Selecting Run Synthesis

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

After the synthesis process completes, the Synthesis Completed dialog box opens. The dialog box prompts you to run implementation, open the synthesized design, or view reports.

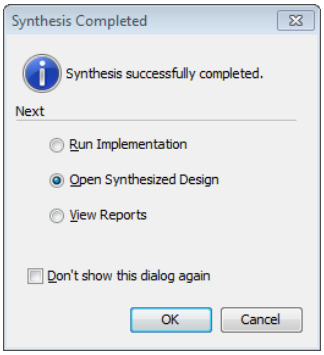

Figure 4-25: Synthesis Completed Dialog Box

2-4-3. Select whichever option best suits your needs.

Remember that any of these choices can be accessed from other places.

Note: If you do not want to do any of these operations options, you can click **Cancel**. This will not undo the synthesis results.

2-4-4. Click **OK** to continue with your preferred choice or **Cancel** to simply close the dialog box and return to the normal view of the Vivado Design Suite.

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*

Importing User Ports and Parameters Step 3

For custom AXI user IP to be included in the IP catalog and used in the Vivado Design Suite IDE and block diagram editor, various Tcl scripts and XML files must exist. 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 turned red as a result of modifying and adding files, parameters, and ports to the project and design.

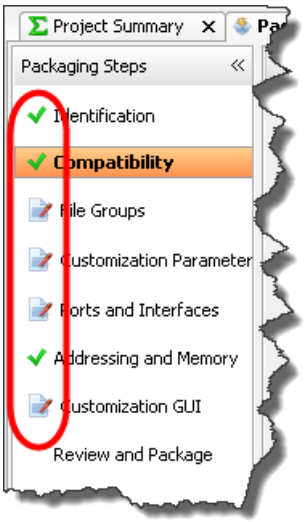

Figure 4-26: Package IP Tab - Checks

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 is generic (meaning that it uses logic resources common to all Xilinx families) you will now configure this piece of IP to be supported in all families 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 (1).

3-1-2. Select **Compatibility** as this provides the list of compatible Xilinx product families that will be supported by this IP (2).

3-1-3. Click the **Add** (+) icon to add an entry to the list (3).

The Zynq® family was added by default as this family was selected when the project was created. You can only add the families that were installed during the initial installation of the Vivado Design Suite or when Add Design Tools or Devices was run.

That is, if you only chose one or two families to install (typically done to save space on the hard drive) only these families can be selected. Add Design Tools or Devices can be run after the initial install to add additional tools or families.

3-1-4. Click **Add Family Explicitly** (4) to open the Add Family dialog box (5).

3-1-5. Select **All Families and Parts** to indicate that this custom AXI IP peripheral is compatible with all families (6).

Because the Zynq family is already part of the list, it must be deselected or a duplicate entry will be created, which will cause an error.

3-1-6. Deselect the following families (7):

- **kintexu (Kintex UltraScale)**
- **virtexu (Virtex UltraScale)**
- **zynq (Zynq-7000)**
- From the Life-cycle drop-down list (8), select **Production** (9).

This sets all of the selections to consider this IP in the *production* state.

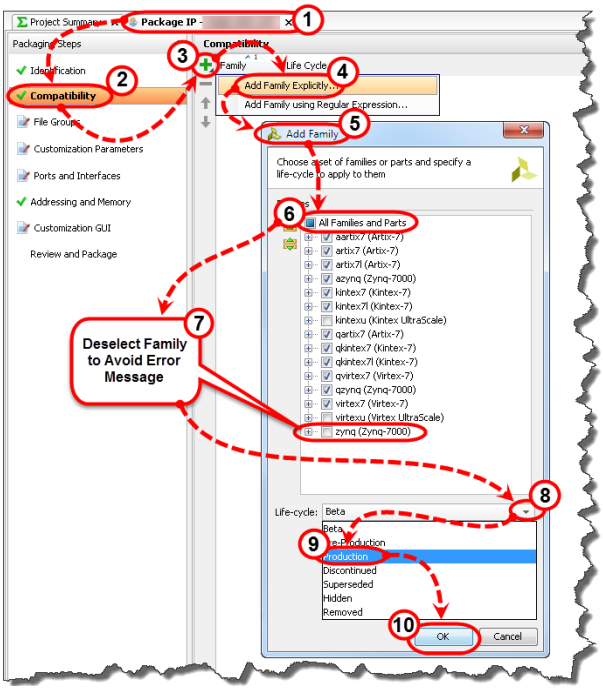

Figure 4-27: IP Compatibility - Family Support

3-1-7. Click **OK** to accept all of the selections and populate the Compatibility list (10).

The life cycle for the Zynq family is at its default selection of Pre-Production.

3-2. Change the Life Cycle property for the Zynq family from Pre-Production to Production.

3-2-1. Click the **Life Cycle** column's drop-down list next to zynq to list all of the life cycle options.

3-2-2. Select **Production**.

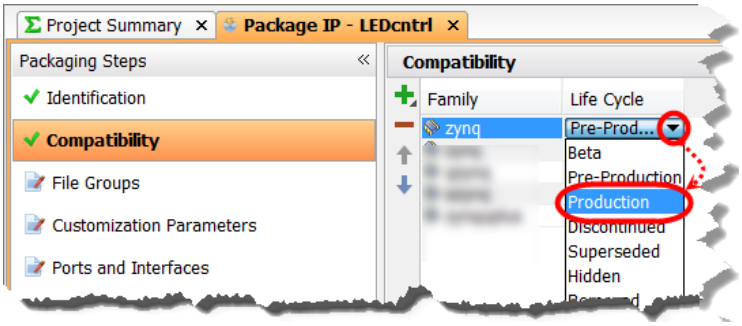

Figure 4-28: Selecting Production

3-3. 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.

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

3-3-2. 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.

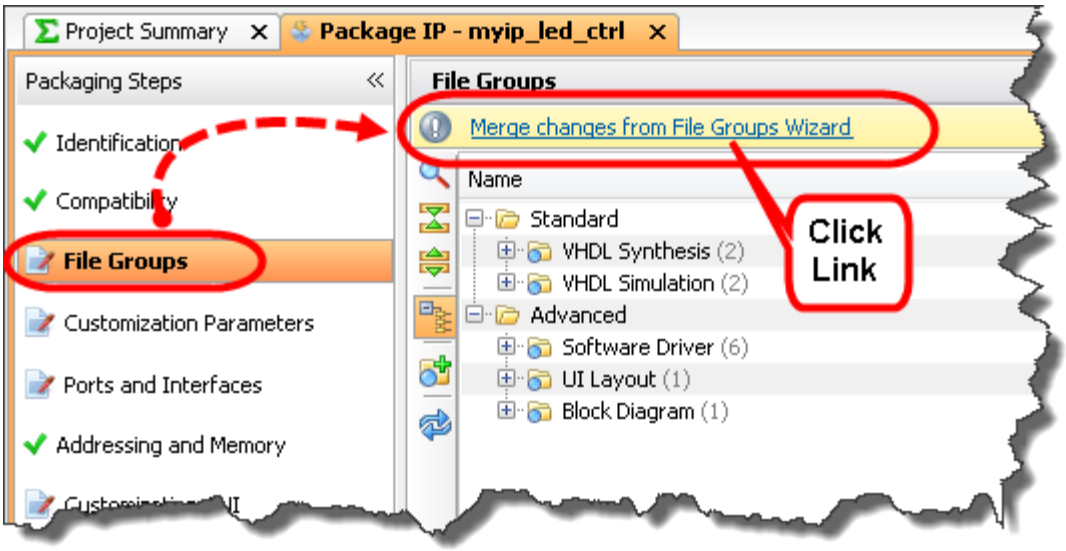

Figure 4-29: Updating File Groups

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

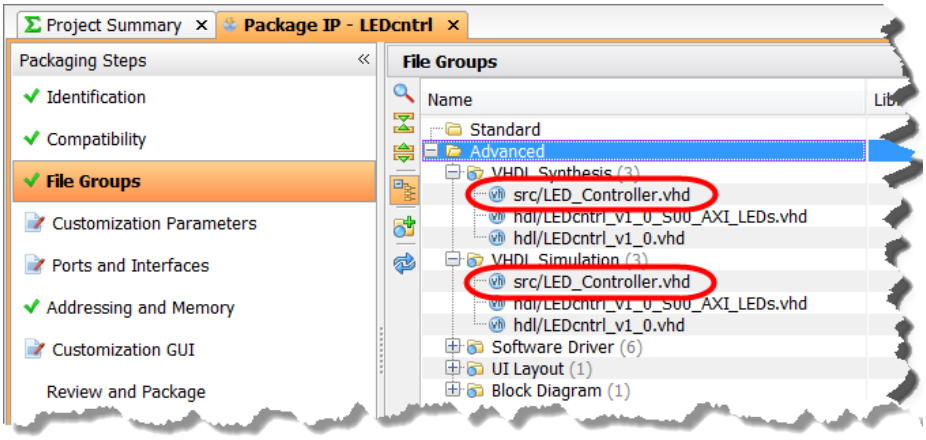

Figure 4-30: Verifying Source File Added to Project

3-4. 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.

3-4-1. 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.

3-4-2. Click the **Merge changes from Customization Parameters Wizard** link.

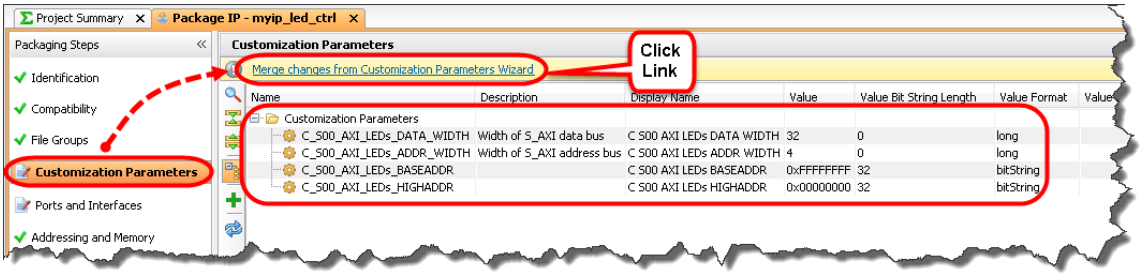

Figure 4-31: Initial IP Customization Parameters

3-4-3. 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.

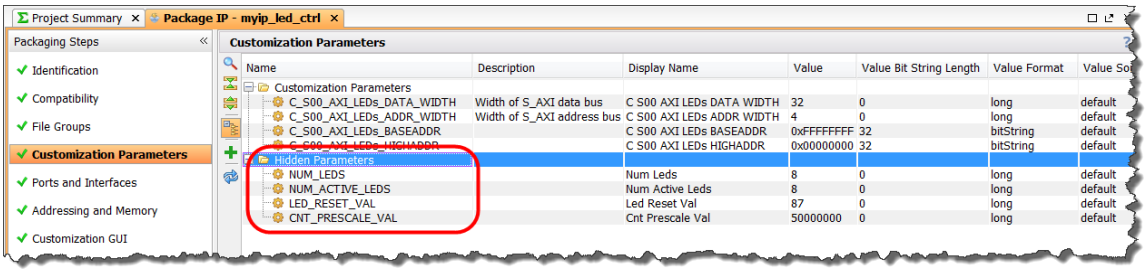

Figure 4-32: User Parameters Now Visible

3-5. 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.

3-5-1. Select **Ports and Interfaces** to view the list.

3-5-2. Verify that the custom added ports were recognized.

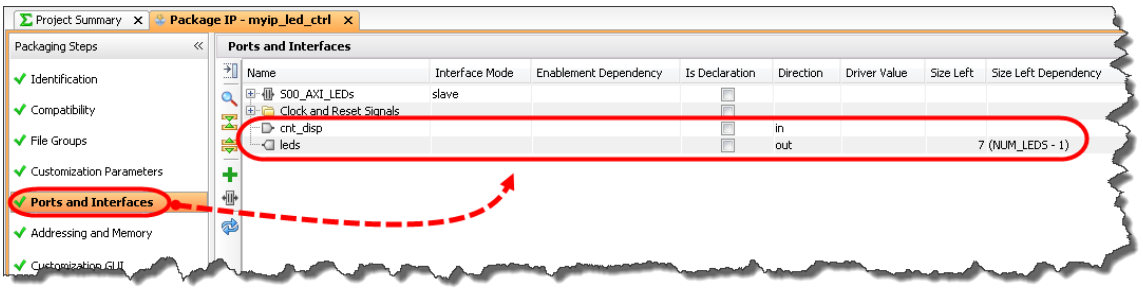

Figure 4-33: User Ports Now Visible

3-6. Update the IP Schematic GUI menu for the user-added generic parameters and ports. In this window you will also configure the schematic GUI to display parameters on a custom pane.

3-6-1. Select **Customization GUI** to access the controls for the generation of the schematic symbol and IP configuration GUI.

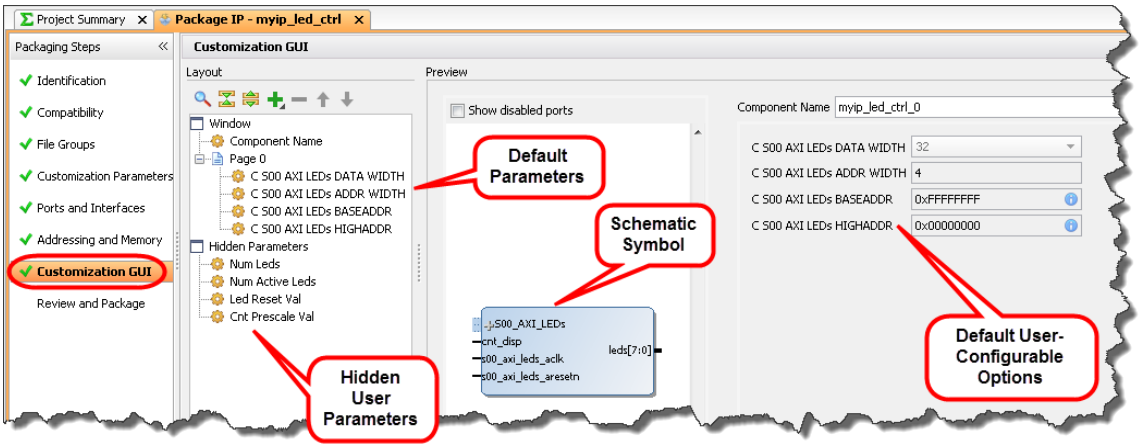

Figure 4-34: GUI Customization - Merge Changes

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.

3-6-2. Select **Layout** pane > **Window**.

3-6-3. Right-click anywhere in the Layout window and select **Add Page**.

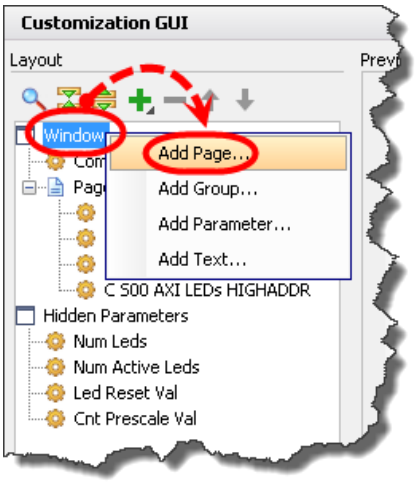

Figure 4-35: Adding a New Parameter Page

3-6-4. Enter **LED Controller Parameters** in the Display name field.

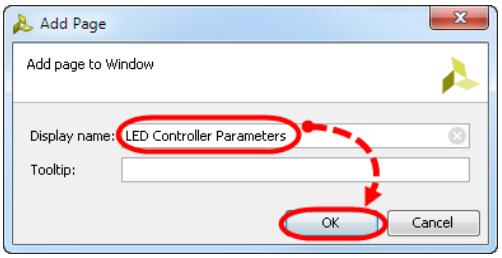

Figure 4-36: Entering New Page Display Name

3-6-5. Click **OK**.

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

3-6-6. Left-click and drag each of the parameters from the Hidden Parameters list to the LED Controller Parameters list.

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

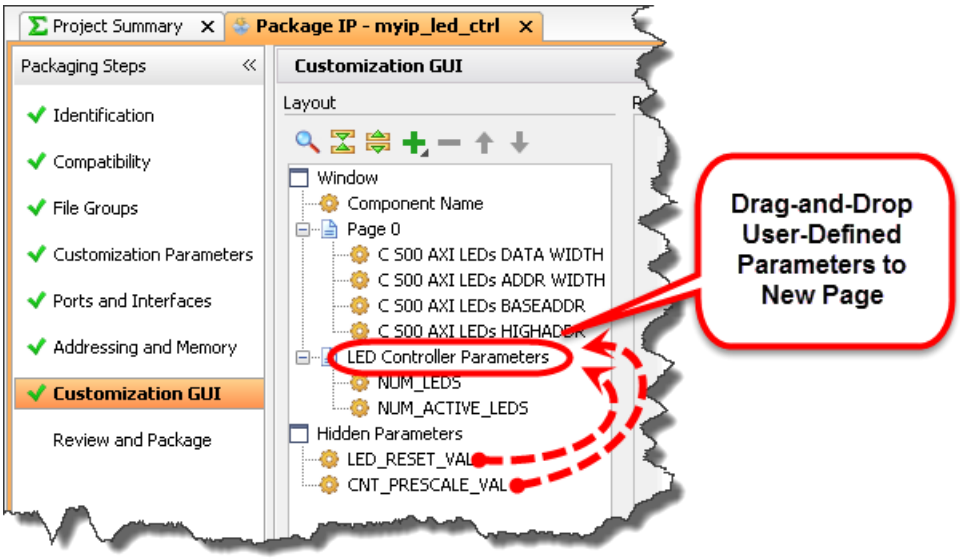

Figure 4-37: Drag & Drop User Parameters to New Page

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.

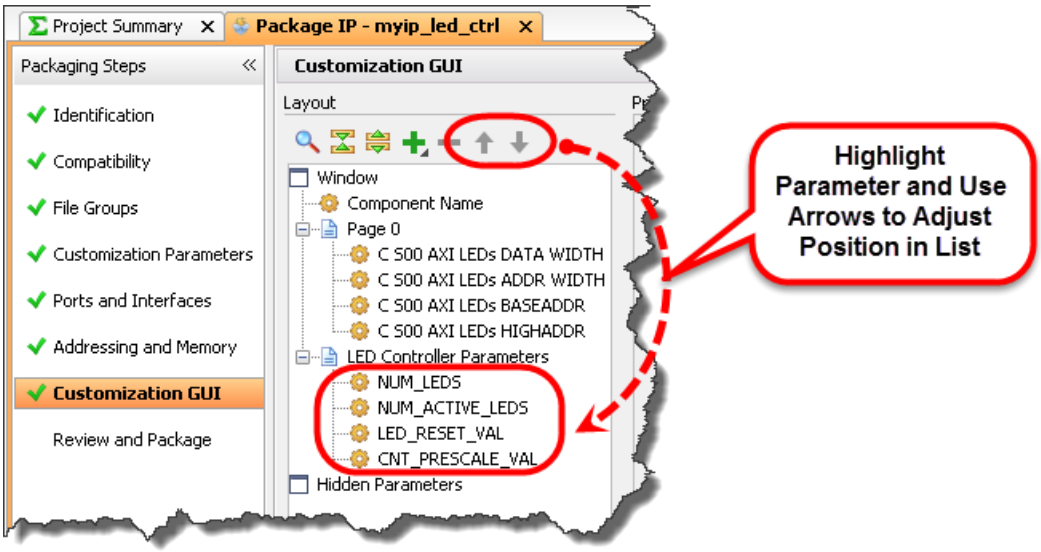

Figure 4-38: Arranging User Parameters on the New Page

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, so, from a minimalists point of view, this page is not really needed. It can be deleted so as not to clutter the schematic symbol GUI.

3-6-7. Right-click the **Page 0** list to select it and open the context menu.

3-6-8. Select **Remove Page**.

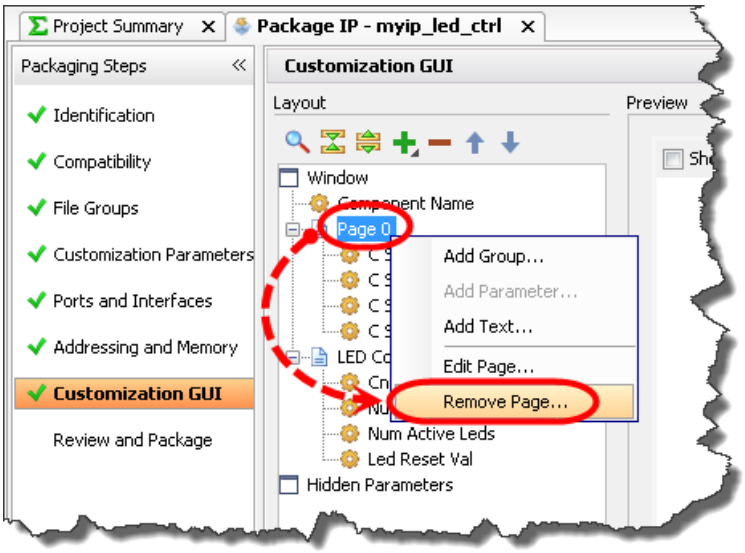

Figure 4-39: Removing Unused Parameter Page

3-6-9. 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.

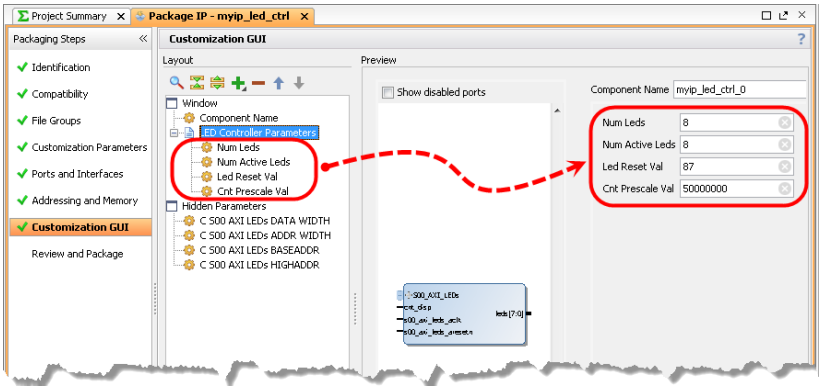

Figure 4-40: Completed GUI Customization

3-7. Re-package the IP to save the changes made. This will update all the necessary scripts and files for the IP catalog.

3-7-1. Select **Review and Package**.

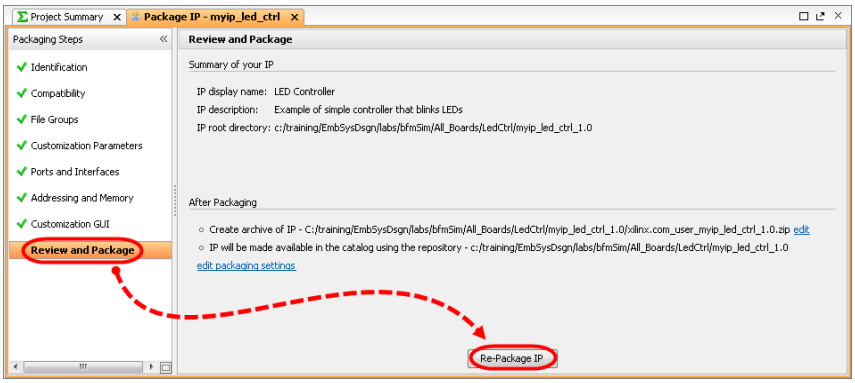

Figure 4-41: Review and Package IP

3-7-2. 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.

3-7-3. Click **Yes** to complete the packaging and close the project.

This Vivado Design Suite projects closes, but the *ip_placeholder* 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**.

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

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

3-8-1. [Optional] Locate the Vivado Design Suite in the operating system task bar and click the **Vivado Design Suite** icon that appears there to bring the tool into view if the *ip_placeholder* 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. [Optional] Select **File** > **Open Project** and select the **placeholder_project.xpr** project located at *C:\training\AXIbldPeriph\lab\placeholder_project*.

3-8-3. [Optional] Enter the following into the Tcl command line of the Tcl Console:

```
source C:/training/AXIbldPeriph/support/
AXI_buildPeriph_completer.tcl
```

This will source the extra Tcl functions provided to automate several processes.

3-8-4. Enter the following into the Tcl command line of the Tcl Console:

```
createBlockDesign; addNewIP
```

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.

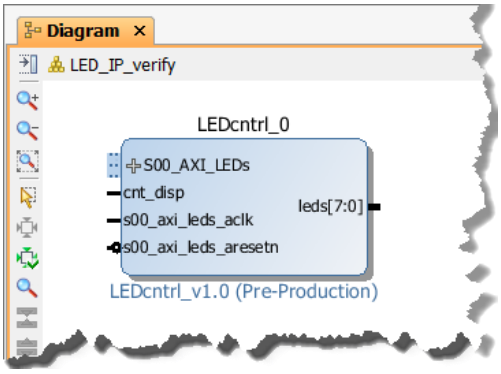

Figure 4-42: Block Diagram with Symbol

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.

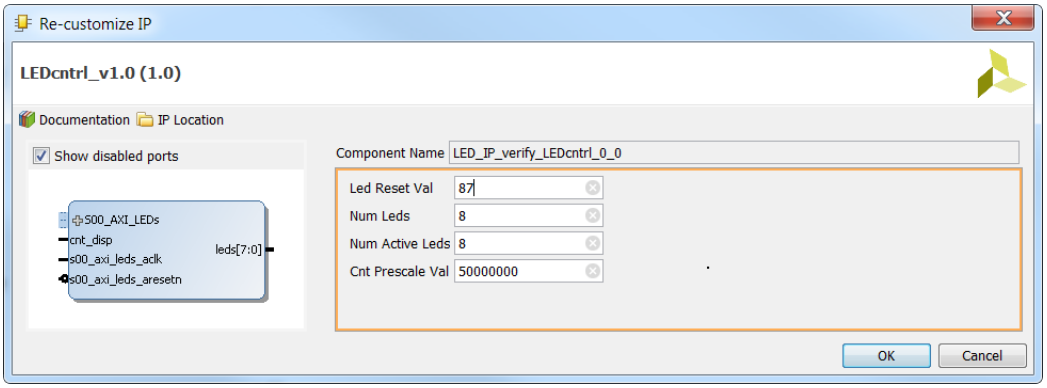

Figure 4-43: Block Diagram IP Configuration

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.

Figure 4-44: Exit Vivado Dialog Box

3-10-2. Click **OK**.

3-10-3. 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 4

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 5

What is the best practice to follow when updating IP?

Summary

Use the Create and Package IP Wizard in the Vivado IDE 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 IDE projects are now open? How do they differ?

   There are two Vivado IDE 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 IDE project, *edit_[your ip name].xpr*, that will be used as the design environment for building the custom IP.

2. 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.

3. 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.

4. 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 IDE 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.

5. 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.