



XAPP1168 (v1.0) June 01, 2013

Packaging Custom AXI IP for Vivado IP Integrator

Author: Dylan Buli and Kyle Corbett

Summary

This application note describes how to bring custom AXI IP into the IP Integrator to fully use the Vivado™ design tools. Six types of AXI-based IP cores are described, including a demonstration for packaging the cores as Vivado-enabled IP. In addition, this document details how to take advantage of the automation features available in the IP Integrator.

This application note enables you to start creating custom IP for an AXI system. Six different IP types are described:

- AXI4 Master
- AXI4 Slave
- AXI4-Lite Master
- AXI4-Lite Slave
- AXI4-Stream Master
- AXI4-Stream Slave

Overview

AXI is a standardized IP interface protocol based on the Advanced Microcontroller Bus Architecture (AMBA 4) specification [Ref 2]. This reference design uses AXI4, AXI4-Lite and AXI4-Stream interfaces as described in the AXI4 specification. These interfaces provide a common IP interface protocol framework for building a system.

Before any custom IP can be used within the IP Integrator, the IP must be packaged. To reduce the barrier of entry into the Vivado design tools chain, Vivado provides a simple way to package the IP. For custom AXI IP, Vivado can automatically infer the AXI interfaces when the IP port naming follows the AXI4 specification.

While Vivado provides multiple ways to approach packaging an IP, this application note provides the following route:

1. Create separate Vivado projects for each IP to be packaged.
2. Import existing RTL which uses a common naming convention for its ports.
3. Run Package IP Wizard, providing additional metadata and parameter associations.
4. Archive the packaged IP into a .zip file, saved in a 'repository' directory.
5. Create a system-level project where the packaged IP will be used.
6. Add the repository directory to the project preferences and add the IP to the repository.
7. Create an IP Integrator block diagram, adding and connecting the packaged IP into a system and simulating it.

Reference Design Contents

The provided ZIP file contains the following:

<https://secure.xilinx.com/webreg/clickthrough.do?cid=338300>

- **Project TCL:** `source.tcl`

This Vivado TCL file runs through the complete application note. To use:

 - a. Open Vivado.
 - b. In the Vivado TCL Console, change the directory to the extracted location of this file.
For example: `cd C:/designs/xapp1168`
 - c. In the Vivado TCL Console, type `source source.tcl`
- **AXI4-Lite Master:** `hdl/verilog/axi_lite_master.v`

The example AXI4-Lite Master has one AXI4-Lite interface with a fixed data width (32 bits) and a fixed address width (32 bits). The IP has been designed to issue a parameterized number of transactions, `C_TRANSACTIONS_NUM`. Following the de-assertion of the reset signal, the IP starts writing at address `0x4000` and continues for `C_TRANSACTIONS_NUM` beats to sequential addresses. Upon completion of the writes, the `WCOMPLETE` signal is asserted. The core then issues reads starting at address `0x4000` and continues for `C_TRANSACTIONS_NUM` beats to sequential addresses. While performing the reads, it checks the data read with the value written to the same address. When the process is completed, the `RCOMPLETE` signal is asserted.

The AXI4-Lite master interface port names begin with `M_AXI_` to indicate that they are master AXI ports.
- **AXI4-Lite Slave:** `hdl/verilog/axi_lite_slave.v`

The example AXI4-Lite slave has one AXI4-Lite interface with a fixed data width (32 bits) and a fixed address width (5 bits). The IP has been designed to have only 32 bytes of addressable memory and will alias all other accesses to the same 32 bytes. Of the 32 bytes, only the bottom 16 bytes have memory elements. Accesses to addresses `0x10 - 0x1F` have no effect.

The example slave will not accept the `AW` or the `W` channel until both channels have presented the appropriate `VALID` signal. After the write occurs, the core issues the `B` channel handshake.

The AXI4-Lite slave interface port names begin with `S_AXI_` to indicate that they are slave AXI ports.
- **AXI4 Master:** `hdl/verilog/axi_master.v`

The example AXI4 master has one AXI4 interface with a fixed address width (40 bits) and a programmable data width, which can be either 64 or 32 bits. The IP has been designed to issue a parameterized number of transfers, `C_TRANSACTIONS_NUM`, with each transfer being `C_BURST_LEN` in length. Following the de-assertion of the reset signal, the IP starts writing at address `0x80000` and continues for `C_TRANSACTIONS_NUM` transfers of `C_BURST_LEN` beats each to sequential addresses. Upon completion of the writes, the `WCOMPLETE` signal is asserted. The core then issues reads starting at address `0x80000` and continues for `C_TRANSACTIONS_NUM` transfers of `C_BURST_LEN` beats each to sequential addresses. While performing the reads, it checks the data read with the value written to the same address. Once the process is completed, the `RCOMPLETE` signal is asserted.

The AXI4 master interface port names begin with `M_AXI_` to indicate that they are master AXI ports.
- **AXI4 Slave:** `hdl/verilog/axi_slave.v`

The example AXI4 slave has one AXI4 interface with a fixed address width (14 bits) and a programmable data width, which can be either 64 or 32 bits. The IP has been designed to have only 16 Kbytes of addressable memory with only 2 Kbytes (at 64 bits) or 1 Kbytes (at 32 bits) containing memory elements. The IP aliases all memory accesses to the memory elements.

The example slave will not accept the AW or W channel until both channels have presented the appropriate `VALID` signal. Once the write occurs, the core issues the B channel handshake.

The AXI4 slave interface port names begin with `S_AXI_` to indicate that they are slave AXI ports.

- **AXI4-Stream Master:** `hdl/verilog/axi_stream_master.v`

The example AXI4-Stream master has one AXI4-Stream interface with a programmable data width, which can range from 1 to 64 bytes. The IP has been designed to continually issue transactions each with a length of `C_PACKET_LENGTH`. Following the de-assertion of the reset signal, the IP starts issuing transactions of `C_PACKET_LENGTH` beats each. Upon completion of the transaction, the IP waits a random number of cycles and issues a new transaction of length `C_PACKET_LENGTH`.

The AXI4-Stream master interface port names begin with `M_AXIS_` to indicate that they are slave AXI4-Stream ports.

- **AXI4-Stream Slave:** `hdl/verilog/axi_stream_slave.v`

The example AXI4-Stream slave has one AXI4-Stream interface with a programmable data width, which can range from 1 to 64 bytes. The IP has been designed to provide a "pseudo-random" back-off applied to the `S_AXIS_TREADY` port.

The AXI4-Stream interface port names begin with `S_AXIS_` to indicate that they are slave AXI4-Stream ports.

Test Benches

Three example block diagrams are created to demonstrate the instantiation of the packaged IP cores. These testbench files act as simple simulation stimulus to these designs. The following test benches are included in the ZIP file:

- `tb/verilog/lite_system_wrapper.v`
- `tb/verilog/axi_system_wrapper.v`
- `tb/verilog/axi_stream_system_wrapper.v`

Package AXI4 IP

This section details the steps to package six types of AXI4 IP, including masters and slaves for AXI4, AXI4-Lite and AXI4-Stream. It describes how to use the Vivado Design Suite to package the IP blocks for re-use in a system.

Note: The following figures may show additional configuration or GUI options than those listed in the instructions. In these cases, leave the option at its default value unless instructed to change the setting.

Packaging instructions for each of the types of AXI4 IP are almost identical. When the instructions vary, the changes are noted by the type of IP that is being packaged.

Start a New Vivado Project and Set the Project Options

This section details the steps to create a Vivado project for each core for the purpose of developing and packaging the RTL IP.

1. Install the Vivado Design Suite (requires Logic Edition at a minimum and an IP Integrator license).
2. Unzip the reference design files into a local folder (referred to as `<design_dir>`).
3. Open the Vivado tools by:
 - Windows: Select **Start > Xilinx Design Tools > Vivado 2013.1 > Vivado**
 - Linux: Enter the `vivado` command in Linux after setting up the Xilinx Vivado design tools.
4. Create a new project by selecting **Create New Project** from the Getting Started section.

5. In the New Project window, click **Next**.
6. Each IP will have a separate project. Enter the appropriate name in the Project Name field.
 - AXI4-Lite Slave: package_axi_lite_slave
 - AXI4-Lite Master: package_axi_lite_master
 - AXI4 Master: package_axi_master
 - AXI4 Slave: package_axi_slave
 - AXI4-Stream Master: package_axi_stream_master
 - AXI4-Stream Slave: package_axi_stream_slave

The selected directory for the new project labeled as <user_dir>. Click **Next**. [Figure 1](#) is using package_axi_lite_slave as the reference.

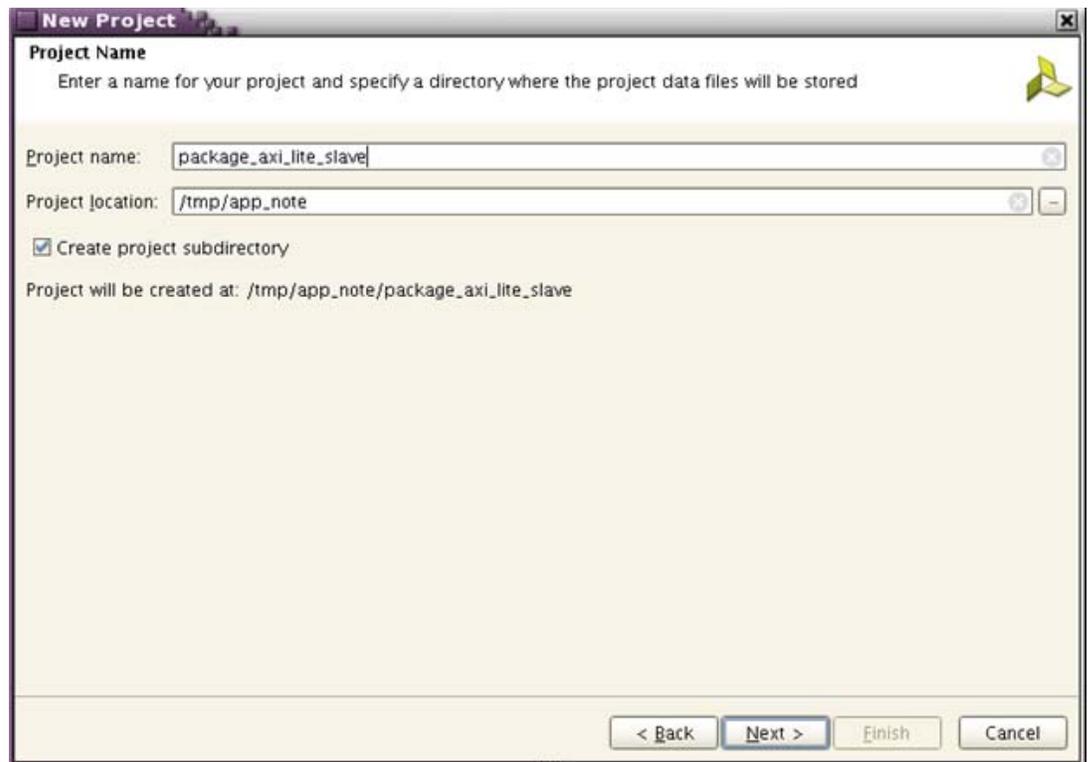


Figure 1: **New Project Window: Project Name**

7. Select **RTL Project** in the Project Type window. Click **Next**.
8. Add sources to the project by clicking **Add Files**.
9. There are six Verilog files in the <design_dir>/hdl/verilog directory.
 - AXI4-Lite Slave: axi_lite_slave.v
 - AXI4-Lite Master: axi_lite_master.v
 - AXI4 Master: axi_master.v
 - AXI4 Slave: axi_slave.v
 - AXI4-Stream Master: axi_stream_master.v
 - AXI4-Stream Slave: axi_stream_slave.v

Select the appropriate source file for the type of IP being packaged. The example shown in [Figure 2](#) shows the packaging of the AXI4-Lite Slave, and the axi_lite_slave.v file is selected.

After selecting the file name, click **OK**.

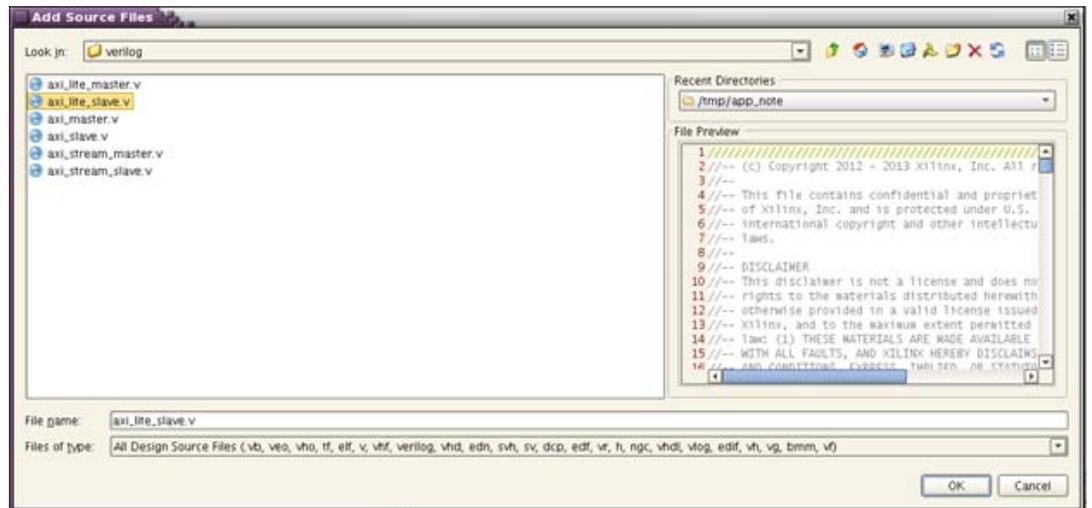


Figure 2: Add Source Files Window: Adding the axi_lite_slave.v Source File

The file, axi_lite_slave.v, is now listed as a source for the new project (Figure 3).

10. Click **Next**.

Note: The file name should correspond to the type of IP being packaged.

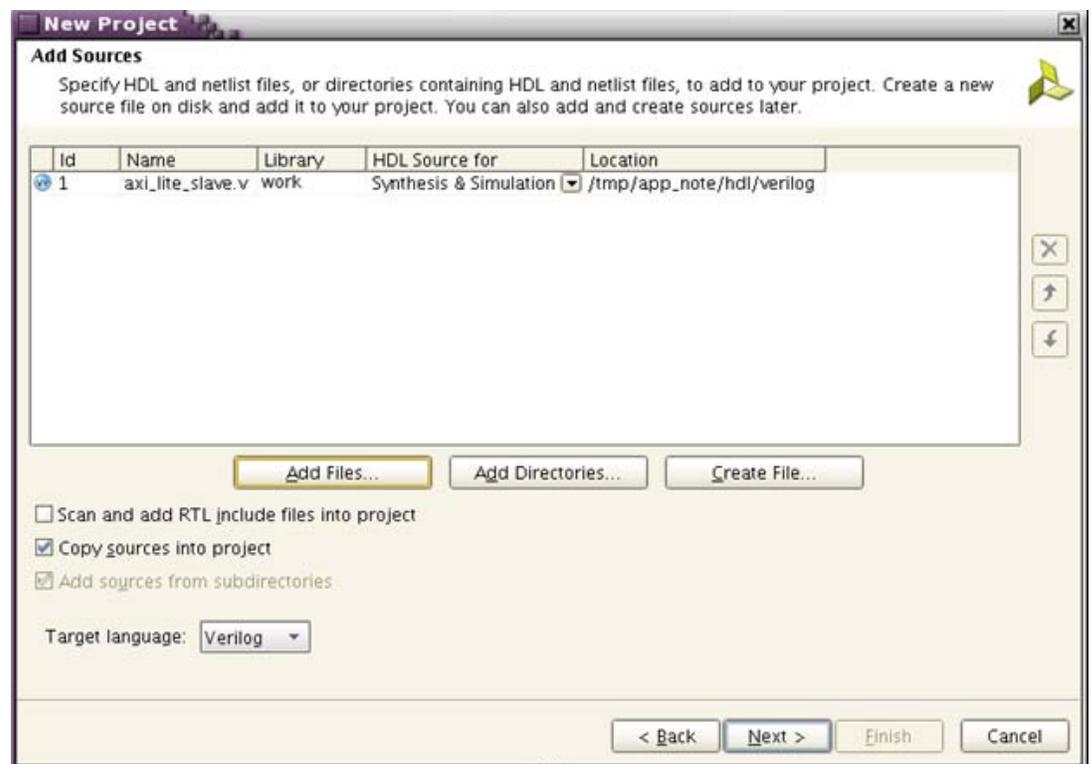


Figure 3: New Project Window: Add Sources with axi_lite_slave.v

11. Click **Next** in the Add Existing IP window.

12. Click **Next** in Add Constraints window.

13. Keep the part selected within the Vivado Design Suite, and click **Next** in the Default Part window (Figure 4).

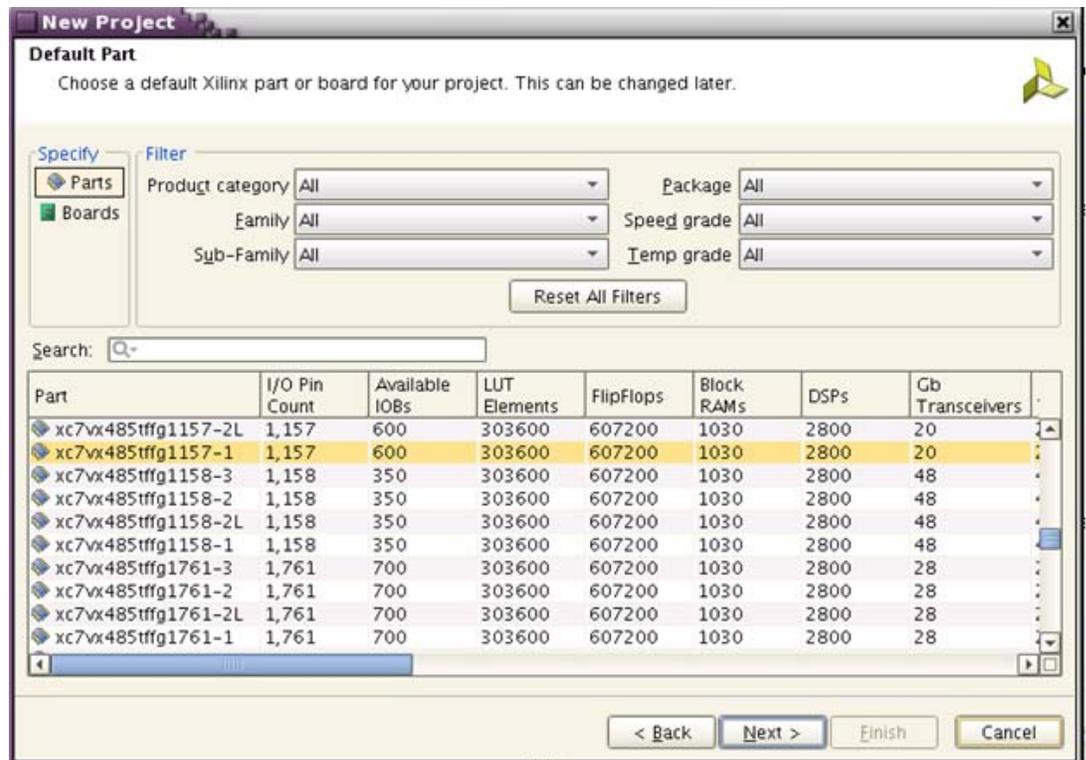


Figure 4: New Project Window: Default Part

The New Project Summary window now appears. Figure 5 shows the summary window for the AXI4-Lite Slave IP. The only difference from the project creation for the other IP is the project name to be created.

14. Ensure that there is one source file to be added and click **Finish**.

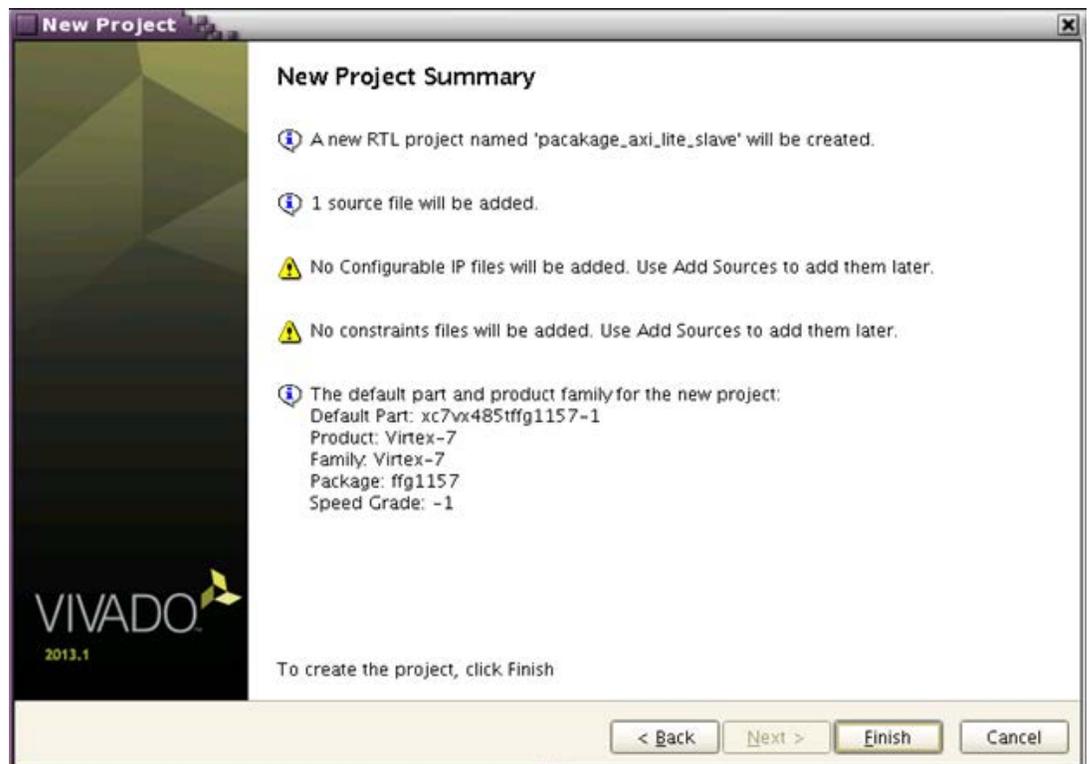


Figure 5: New Project Window: New Project Summary for AXI4-Lite Slave

Packaging a Vivado Project

The Vivado Design Suite offers a quick service to convert an open Vivado project into a packaged IP. The following steps detail the process.

1. Select **Tools > Package IP Wizard** (Figure 6).

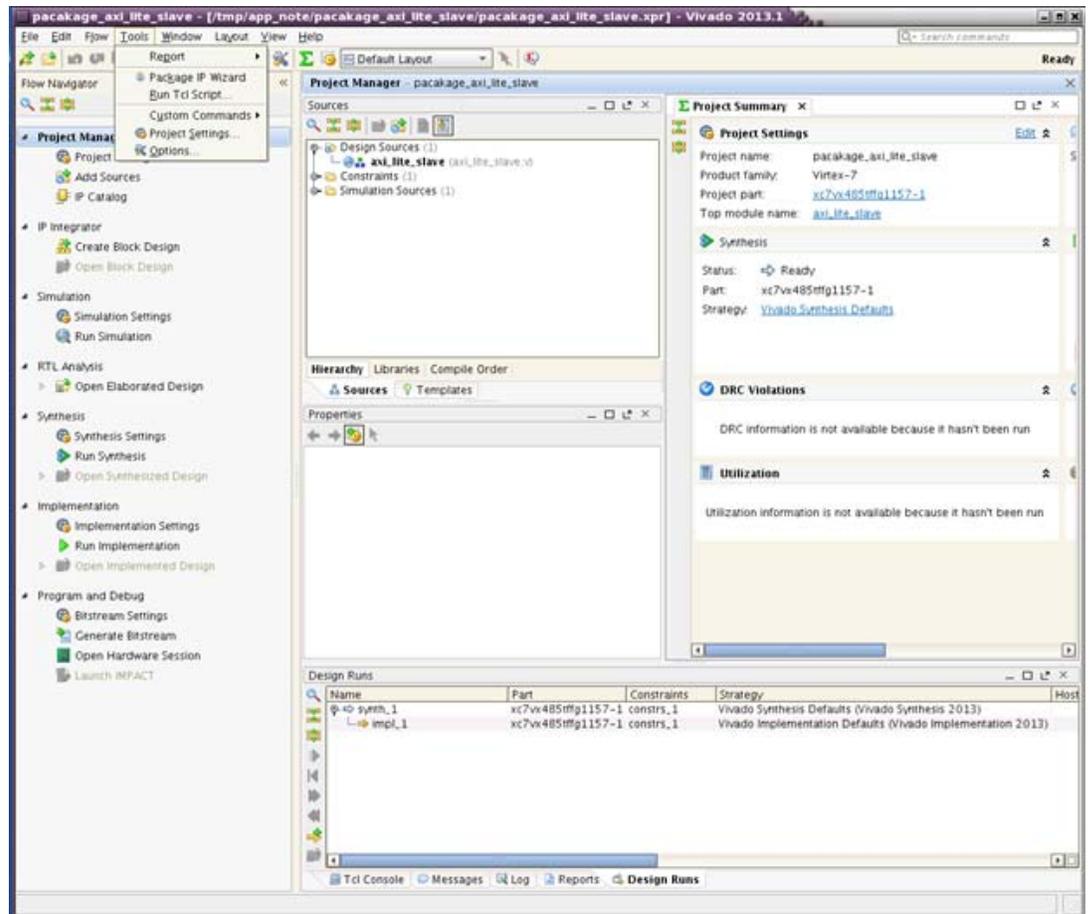


Figure 6: Selecting Tools > Package IP Wizard

2. Click **Next** in the Package New IP window.
3. Select **Package your project** in the Choose IP Source Location window, and click **Next** (Figure 7).

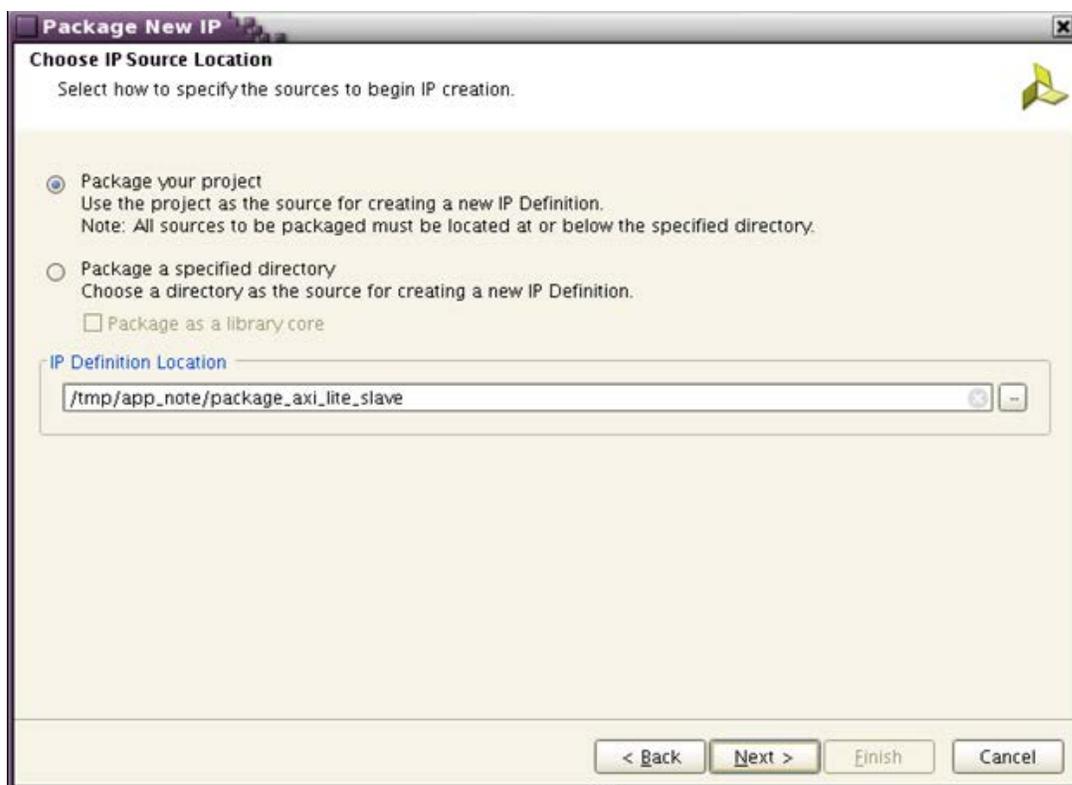


Figure 7: Package New IP Window (Choose IP Source Location)

4. The Begin IP Creation window will appear. Click **Finish**.
5. When the wizard has finished processing, the IP Packager Summary window appears. This window details the number of interfaces, ports and parameters imported from the HDL. Click **OK**. Figure 8 shows the results from packaging the AXI4-Lite Slave IP. Each IP will have different values for each field.

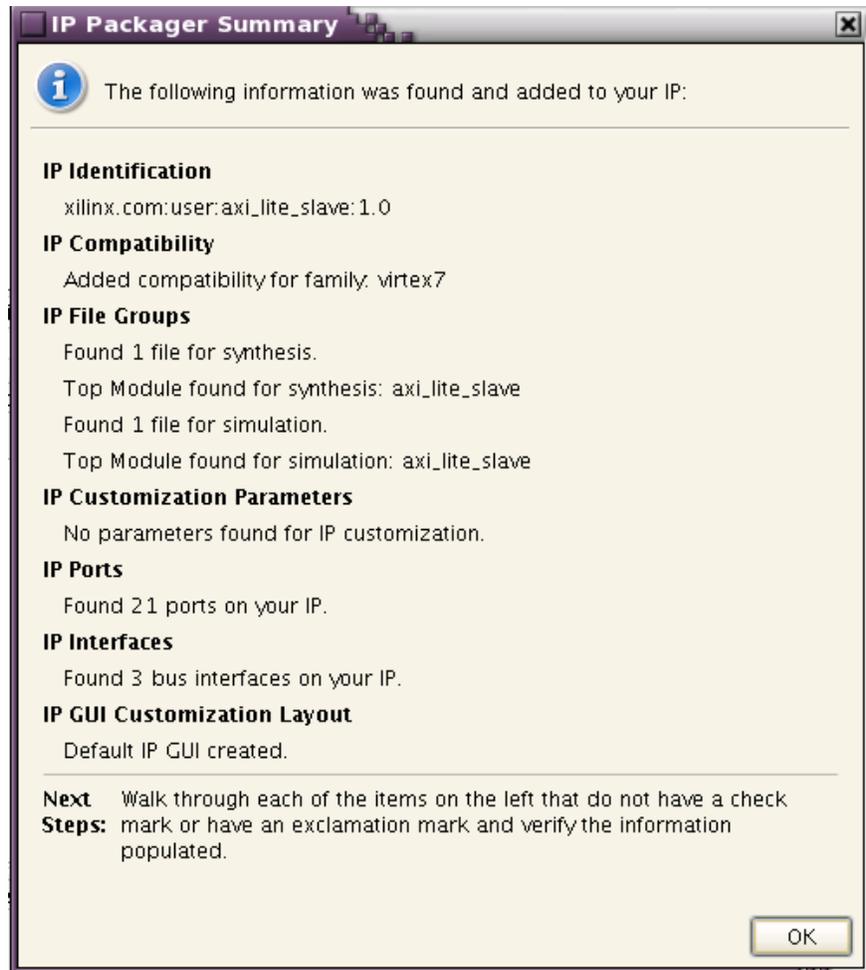


Figure 8: IP Packager Summary Window for AXI4Lite Slave

Updating the Auto-Generated IP Definition

The RTL of the project has been imported and the wizard has estimated the characteristics of the IP. If the import has been successful, a tab named Package IP - <IP_NAME> will be beside the Project Summary Tab. [Figure 9](#) shows the results for the AXI4-Lite Slave IP. The values for Name, Display Name and Description will be different for each packaged IP.

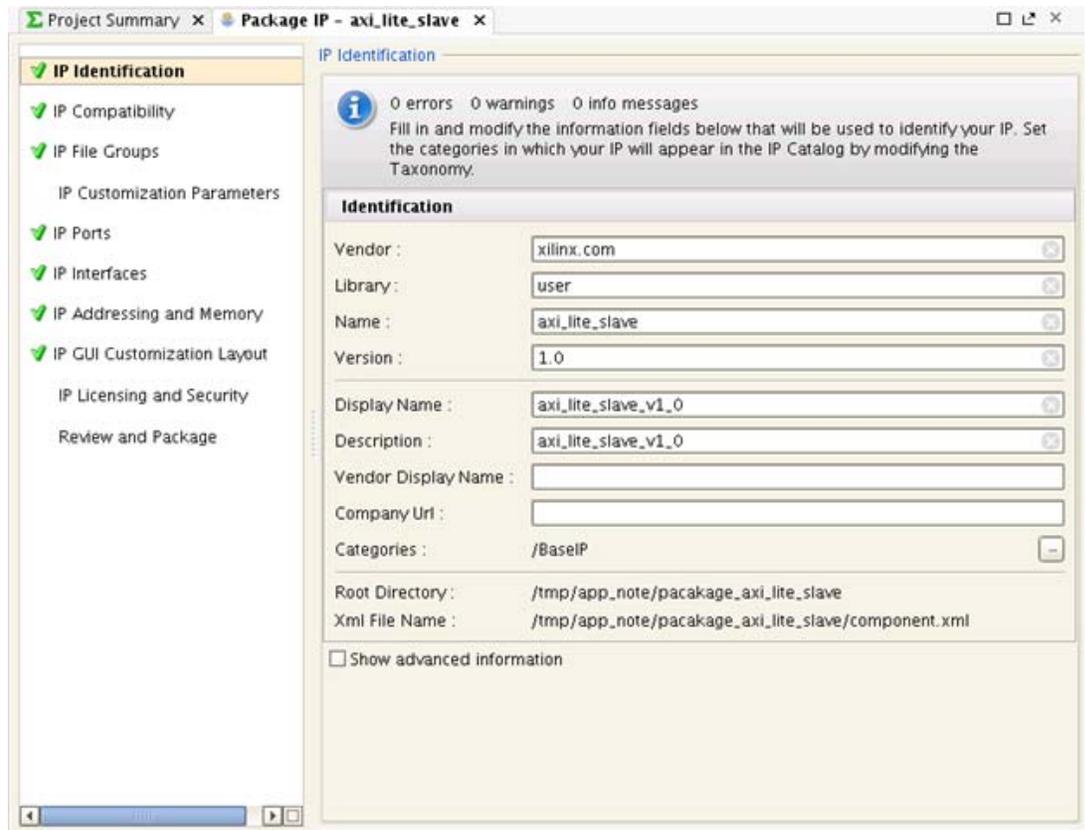


Figure 9: Package IP Tab

6. In the IP Identification window, set the following fields, substituting values if desired:
 - Vendor: YourCompanyName.com
 - Vendor Display Name: Your Company Name
 - Company Url: www.YourCompanyName.com

An example for the AXI4-Lite Slave resulting window is shown in [Figure 10](#).

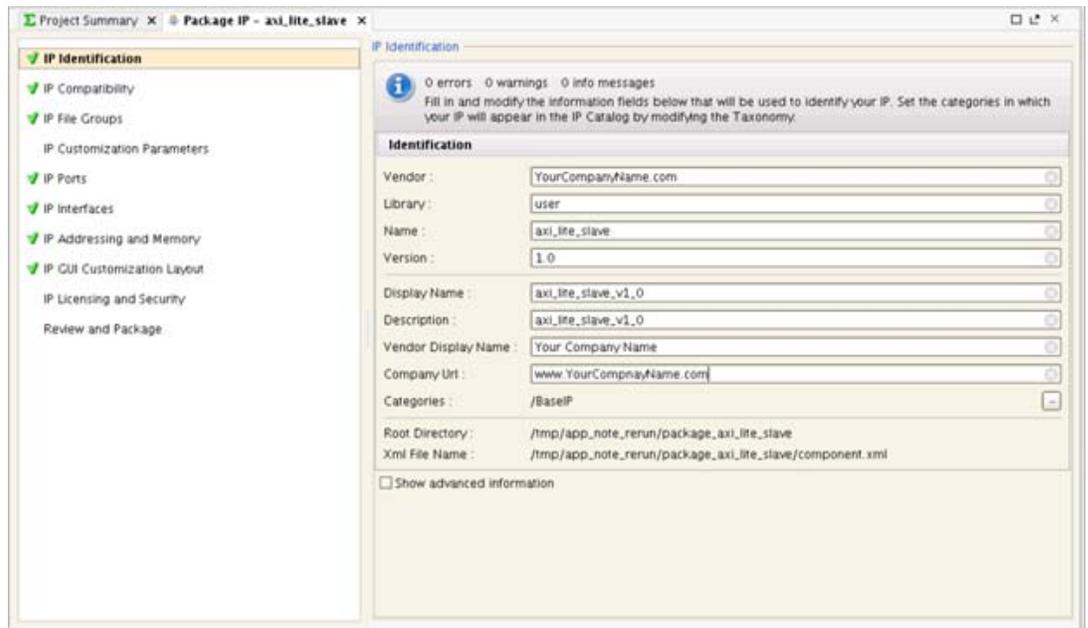


Figure 10: Package IP Tab Following Vendor Customization

7. For the IP to appear in the IP catalog in particular categories, the IP must be configured to be part of those categories. To change which categories the IP will appear in the IP catalog click the ... box on the Categories line. This produces the Choose IP Categories window (Figure 11).

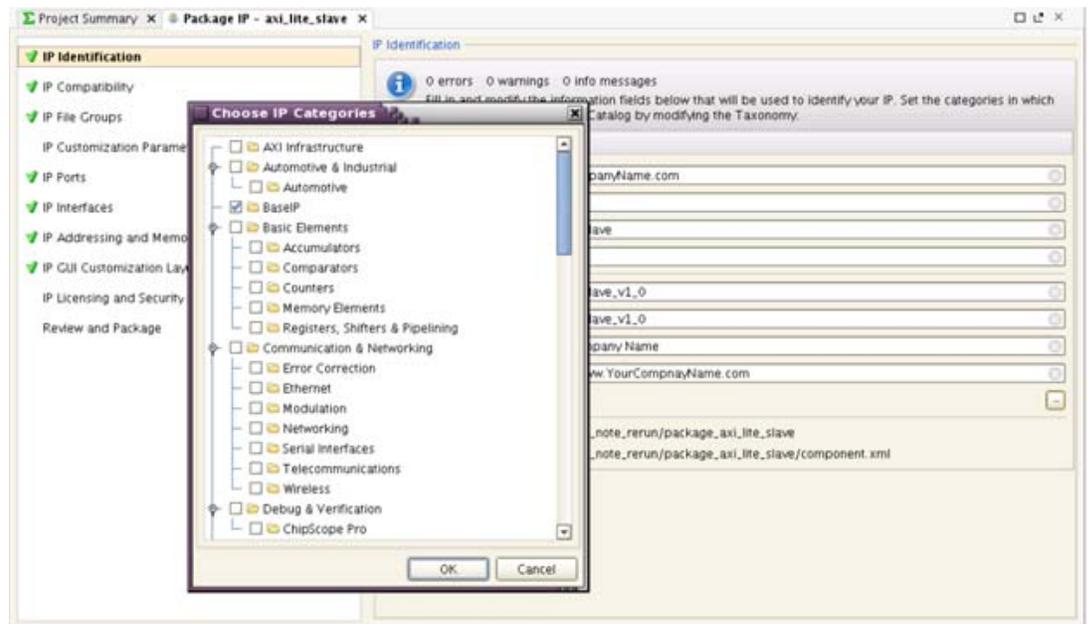


Figure 11: Choose IP Categories Window

In the Choose IP Categories window, perform the following tasks and click **OK**.

- Uncheck the BaselP box.
 - Check the AXI Infrastructure box.
8. Select **IP Compatibility**.
This shows the different Xilinx FPGA Families that the IP supports. The value is inherited from the project and can be changed by right-clicking in the Family Support table and

selecting **Add Family...** from the menu (Figure 12). The IP will not appear to be available to devices which are not on the Family Support list of the IP core.

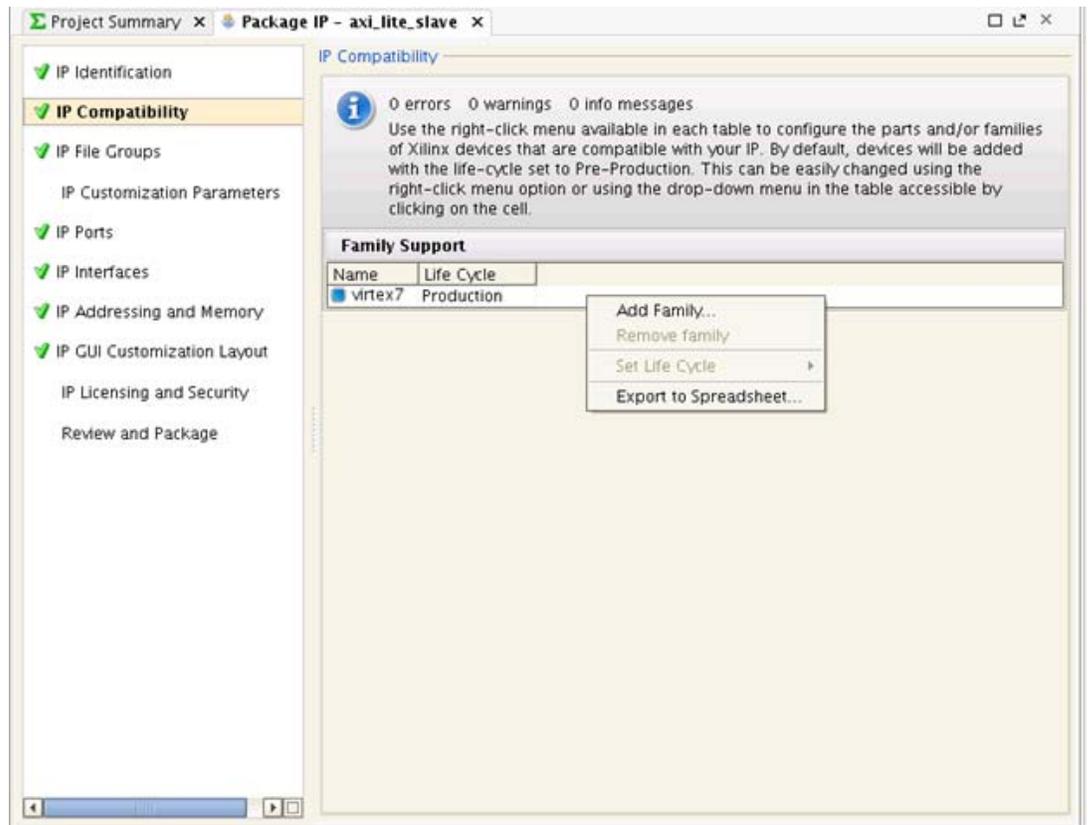


Figure 12: IP Compatibility: Add Family Menu

9. In the Choose Family Support window, select **kintex7** and click **OK** (Figure 13).

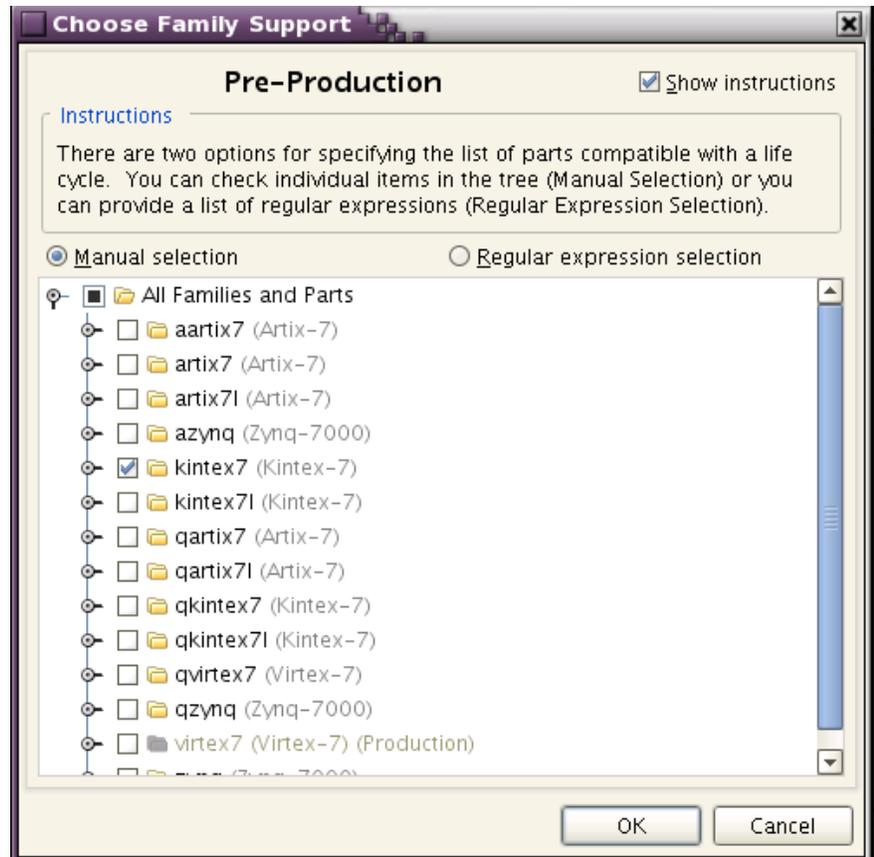


Figure 13: Choose Family Support Window

10. Select **IP Ports**. Note the naming convention used for all ports in the imported verilog. An interface label such as "S_AXI" followed by the standard AXI signal name and vector widths in the Size Left column.

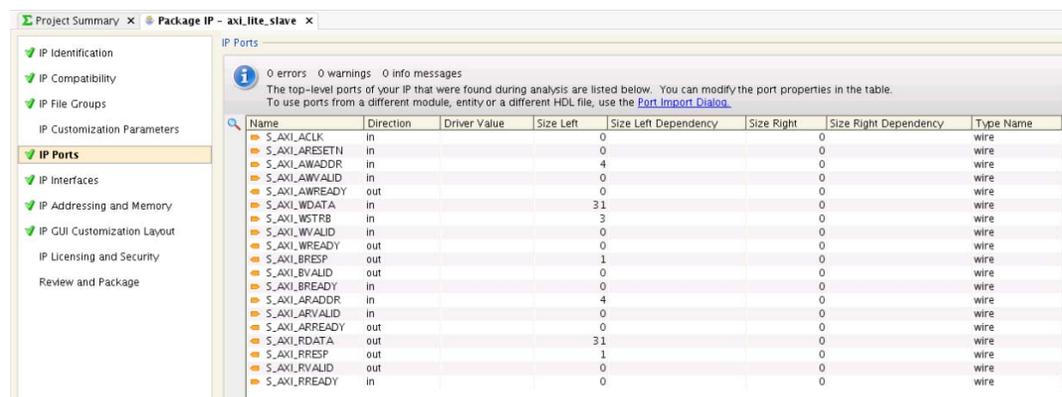


Figure 14: Selecting the IP Ports

11. Select **IP Interfaces**. Note the automatically-inferred clock, reset, and AXI Interface. The AXI Interface collects the consistently-named AXI signals into a single group which can be

handled as an abstracted unit. This allows a simple connection to other AXI Interfaces in a design.

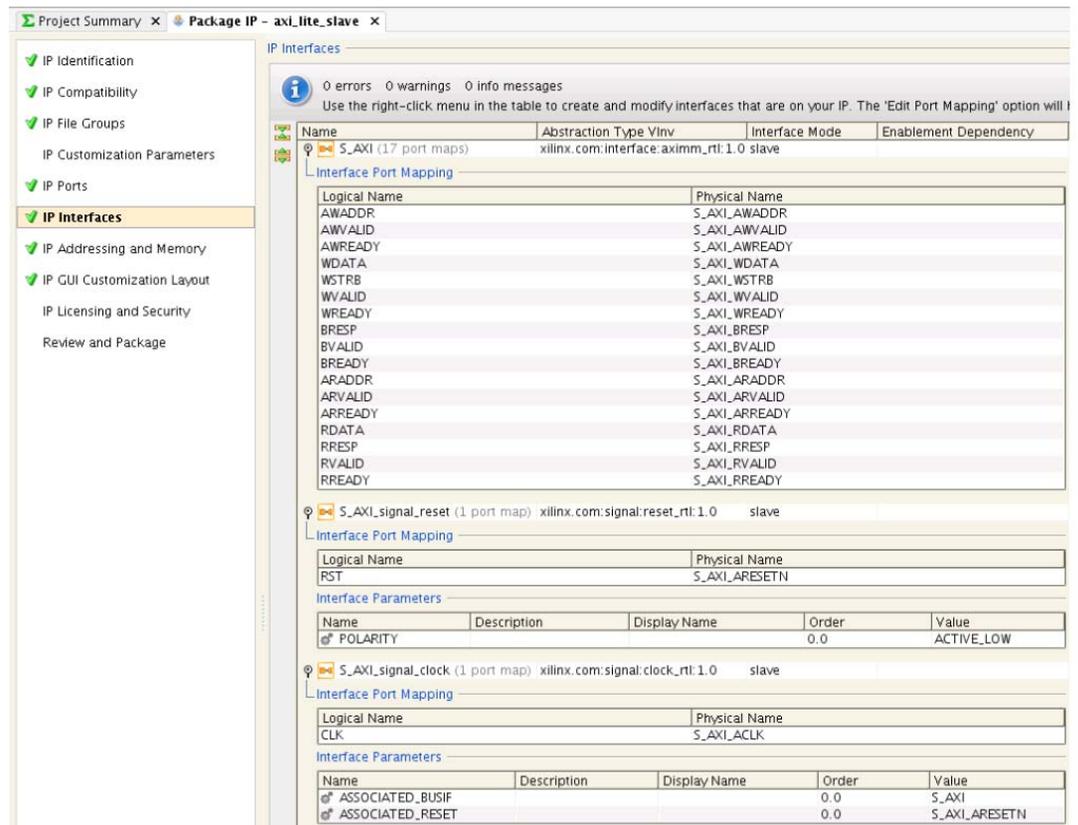


Figure 15: Selecting the IP Interfaces

At this point in the process, customization is distinct to each IP. Choose one of the following to customize and then repeat the previous steps to customize the remaining IP types:

- [AXI4-Lite Slave IP Customization](#)
- [AXI4-Lite Master IP Customization](#)
- [AXI4 Master IP Customization](#)
- [AXI4 Slave IP Customization](#)
- [AXI4-Stream Master IP Customization](#)
- [AXI4-Stream Slave IP Customization](#)

AXI4-Lite Slave IP Customization

The AXI4-Lite Slave IP has a fixed address width of 5 bits on the AxADDR port on the S_AXI interface.

1. Select **IP Addressing and Memory**. The 32 in the range field is the number of addressable bytes in the IP (Figure 16). This is the minimum address space that the peripheral must be assigned.

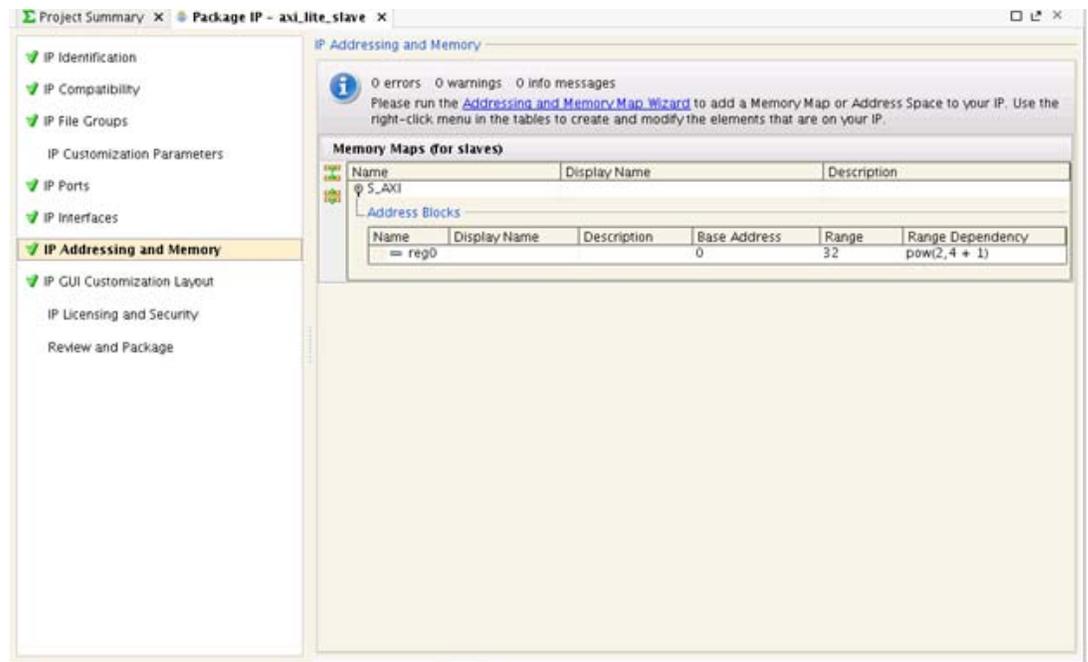


Figure 16: IP Addressing and Memory for AXI4-Lite Slave

2. Go to **Review and Package**, [step 1](#).

AXI4-Lite Master IP Customization

Customizing the AXI4-Lite Master IP will modify the appearance of the configuration GUI of the IP for this HDL parameter. It adds limits to the range of values that the user can input when using the IP.

1. Select **IP Customization Parameters**. In the User Parameters table, update the following fields for C_TRANSACTIONS_NUM ([Figure 17](#)):
 - Description : Number of Transactions
 - Display Name : Number of Transactions
 - Maximum: 64
 - Minimum: 1

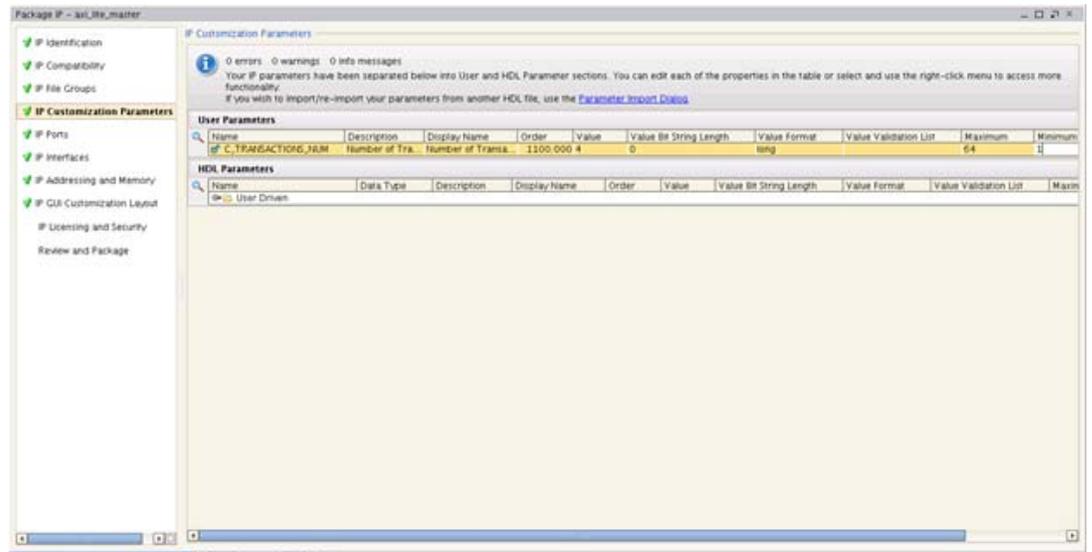


Figure 17: IP Customization Parameters for AXI4-Lite Master

2. Select **IP Addressing and Memory**. The AXI4-Lite Master IP can access up to 4 GBytes of address. This value is shown in Figure 18 in the range field.



Figure 18: IP Addressing and Memory for AXI4-Lite Master

3. Go to **Review and Package**, step 1.

AXI4 Master IP Customization

1. Select **IP Customization Parameters**.
2. Select the C_M_AXI_DATA_WIDTH value in the table. Right-click and select **Edit Parameter...** (Figure 19).

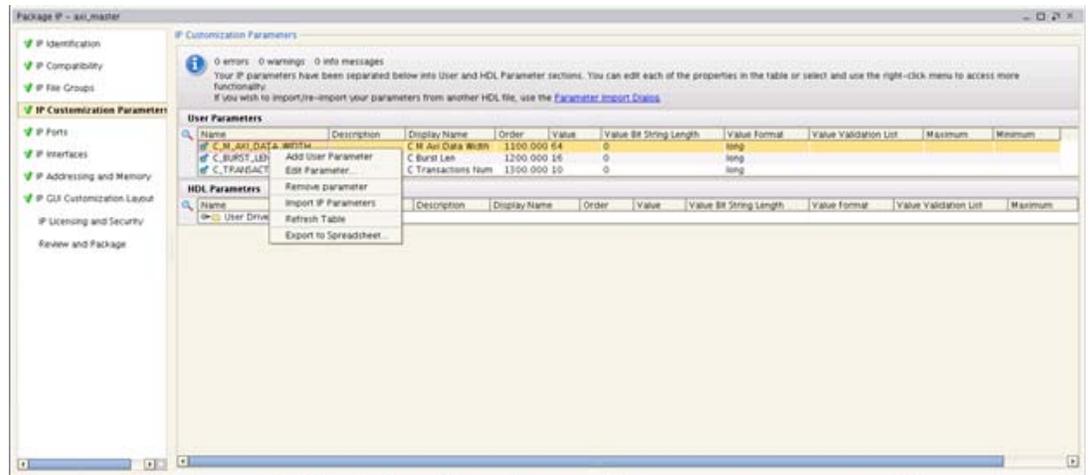


Figure 19: IP Customization Parameters for AXI4 Master IP, C_M_AXI_DATA_WIDTH

- Update the following fields in the Edit Parameter window (Figure 20) and click **OK**. These fields update the GUI. The AXI data width parameter will be limited to two choices.
 - Display name: **Data Width**
 - Should the value be restricted to a list or range?: Select **Yes**.
 - Select **Simple list**. Follow the instructions to enter 32 and 64 into the list.

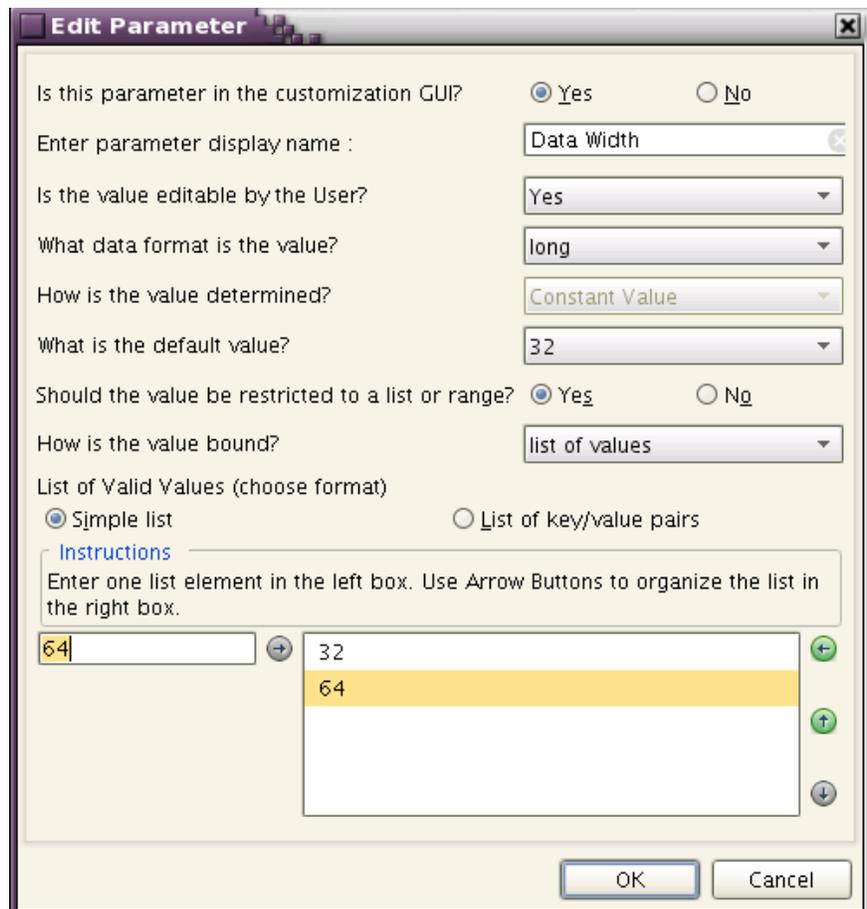


Figure 20: Edit Parameter Window for AXI4 Master

- Select the C_BURST_LEN User Parameter and change the following values:

- Description : Burst Length
 - Display Name: Burst Length
 - Maximum: 256
 - Minimum: 1
5. Select the C_TRANSACTIONS_NUM User Parameter and change the following values:
 - Description: Number of Transactions
 - Display Name: Number of Transactions
 - Maximum: 64
 - Minimum: 1
 6. Select **IP Addressing and Memory**. The AXI4 Master has a 40-bit AxADDR port on the M_AXI interface (Figure 21)

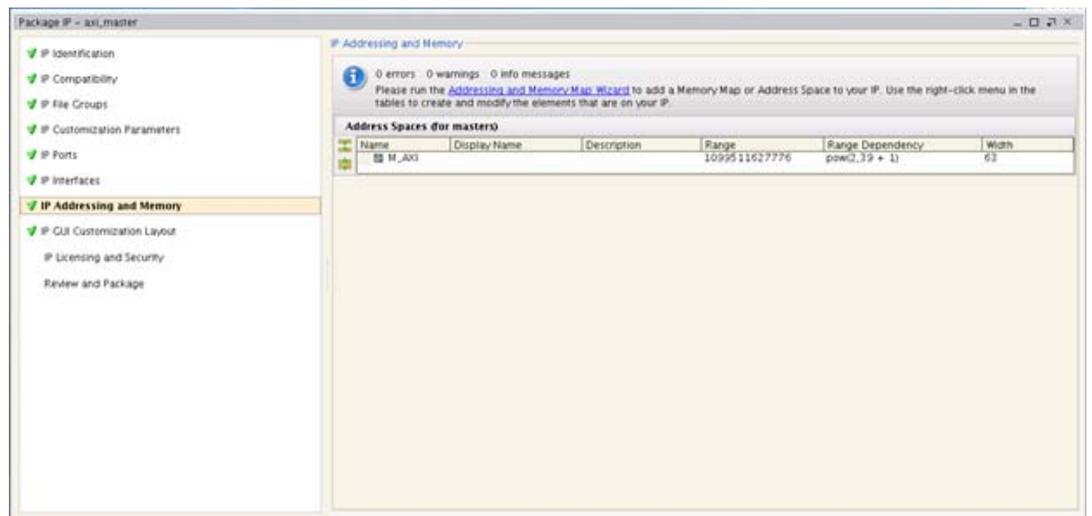


Figure 21: IP Addressing and Memory for AXI4 Master

7. Go to **Review and Package**, step 1

AXI4 Slave IP Customization

1. Select **IP Customization Parameters**.
2. Select the C_S_AXI_DATA_WIDTH value in the table. Right-click and select **Edit Parameter...** (Figure 22)

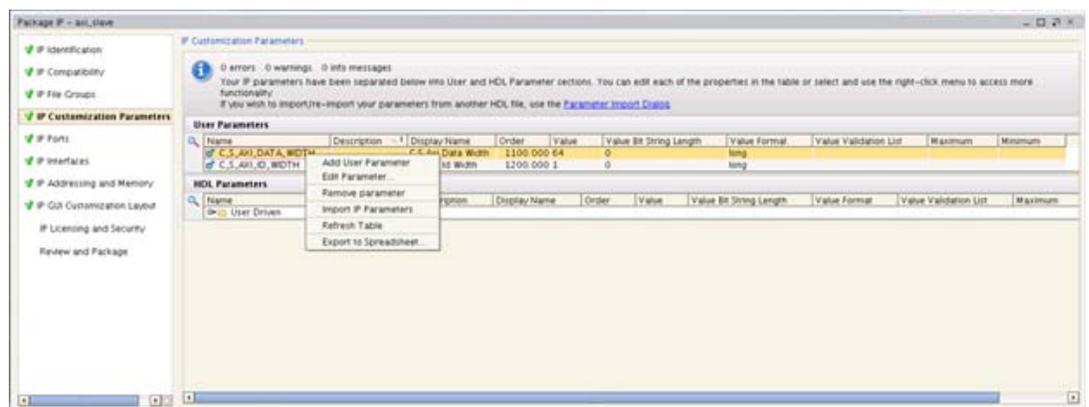


Figure 22: IP Customization Parameters for AXI4 Slave IP, C_S_AXI_DATA_WIDTH

3. Update the following fields in the Edit Parameter window (Figure 23) and click **OK**. These fields update the GUI. The AXI data width parameter will be limited to two choices.
 - Display name: **Data Width**
 - Should the value be restricted to a list or range?: Select **Yes**.
 - Select **Simple list**. Follow the instructions to enter 32 and 64 into the list.

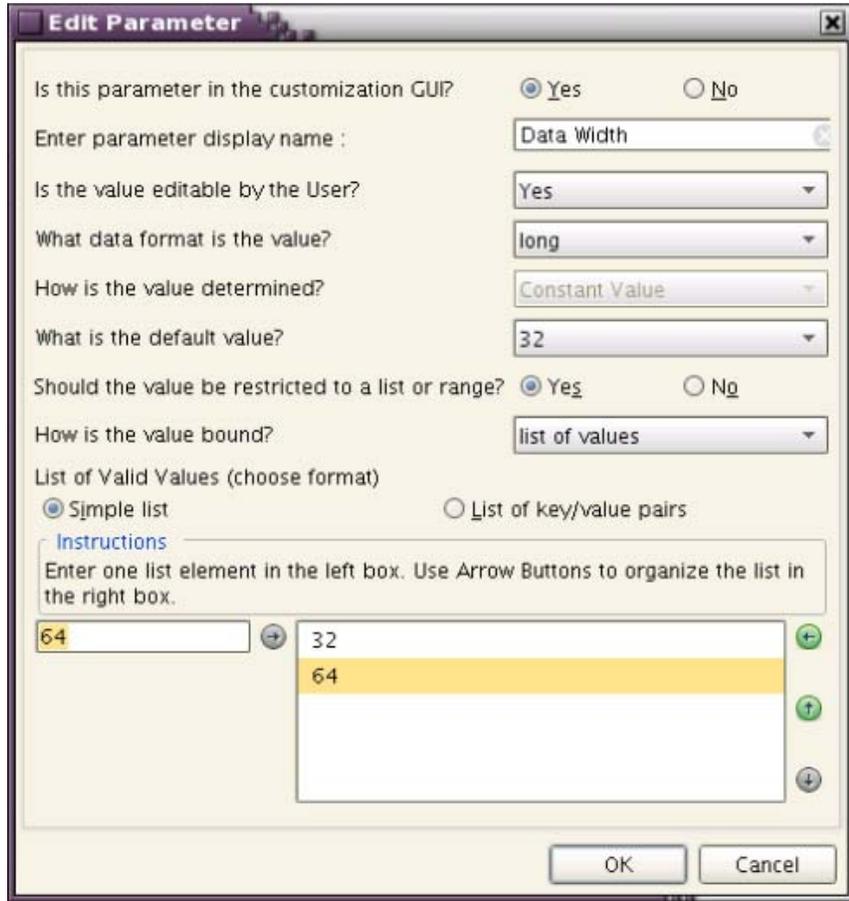


Figure 23: Edit Parameter Window for AXI4 Slave

4. Select the C_S_AXI_ID_WIDTH User Parameter and change the following values:
 - Description: ID Width
 - Display Name: ID Width
 - Maximum: 32
 - Minimum: 1
5. Select **IP Addressing and Memory**. The AXI4 Slave has a 14-bit AxADDR port on the S_AXI interface (Figure 24).



Figure 24: IP Addressing and Memory for AXI4 Slave

- Go to **Review and Package**, [step 1](#)

AXI4-Stream Master IP Customization

- Select IP Customization Parameters option
- Select C_PACKET_LENGTH User Parameter and change the following values:
 - Description: Packet Length
 - Display Name: Packet Length
 - Maximum: 64
 - Minimum: 1
- Select C_M_AXIS_TDATA_NUM_BYTES User Parameter and change the following values:
 - Description: Data Width in bytes
 - Display Name: Data Width in bytes
 - Maximum: 512
 - Minimum: 1

The final result is shown in [Figure 25](#).

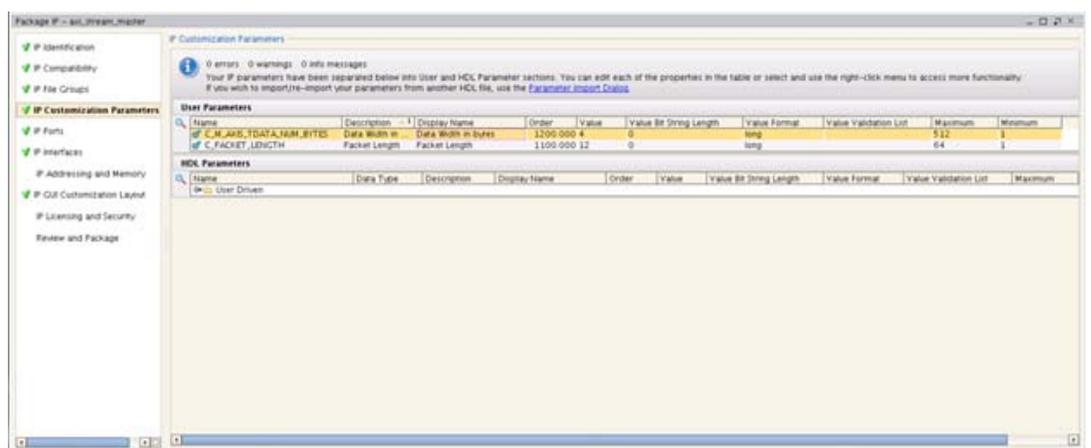


Figure 25: IP Customization Parameters for AXI4-Stream Master

- Go to **Review and Package**, [step 1](#)

AXI4-Stream Slave IP Customization

1. Select **IP Customization Parameters**.
2. Select **C_S_AXIS_TDATA_NUM_BYTES** User Parameter and change the following values:
 - Description: Data Width in bytes
 - Display Name: Data Width in bytes
 - Maximum: 512
 - Minimum: 1

The final result is shown in [Figure 26](#).



Figure 26: IP Customization Parameters for AXI4-Stream Slave

3. Go to **Review and Package**, [step 1](#).

Review and Package

1. Select **Review and Package**, and click **Archive IP** ([Figure 27](#)).

Note: The Archive IP button allows an output directory to be chosen for the packaged IP core, and creates a ZIP archive of the packaged IP. The Add to Catalog button adds the IP inside of the project itself. Since a separate project is created for each IP, the Archive IP selection is used.

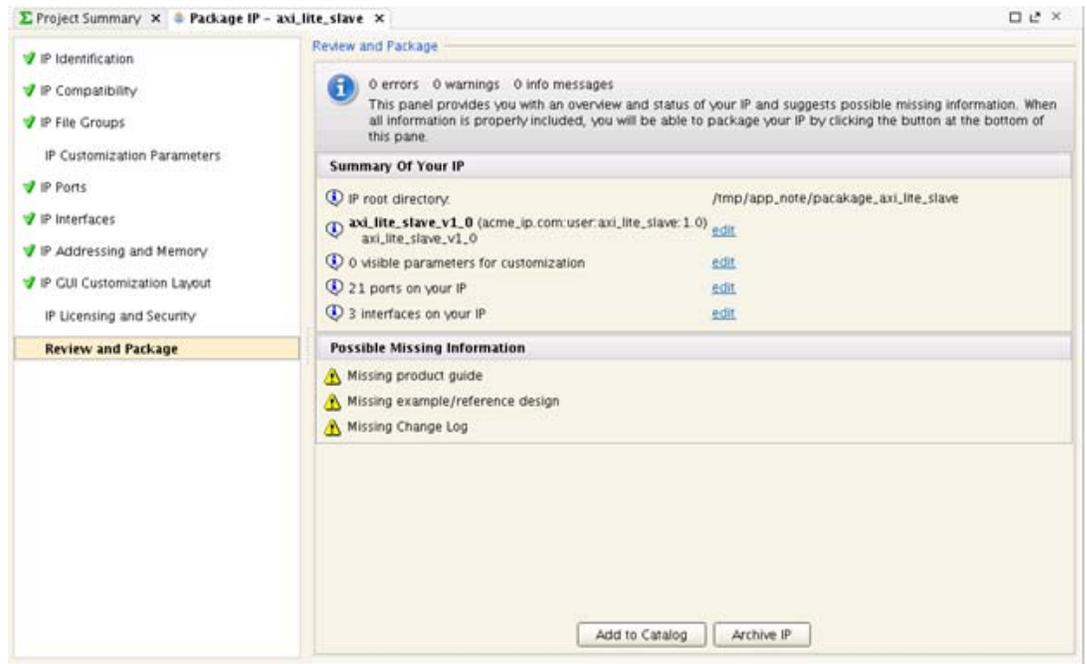


Figure 27: Package IP Tab (Review and Package) for AXI4-Lite Slave

- In the Package IP window, set the archive location: <design_dir>/ip_repo
The example shown is for the AXI4-Lite Slave IP. Click **OK** (Figure 28).

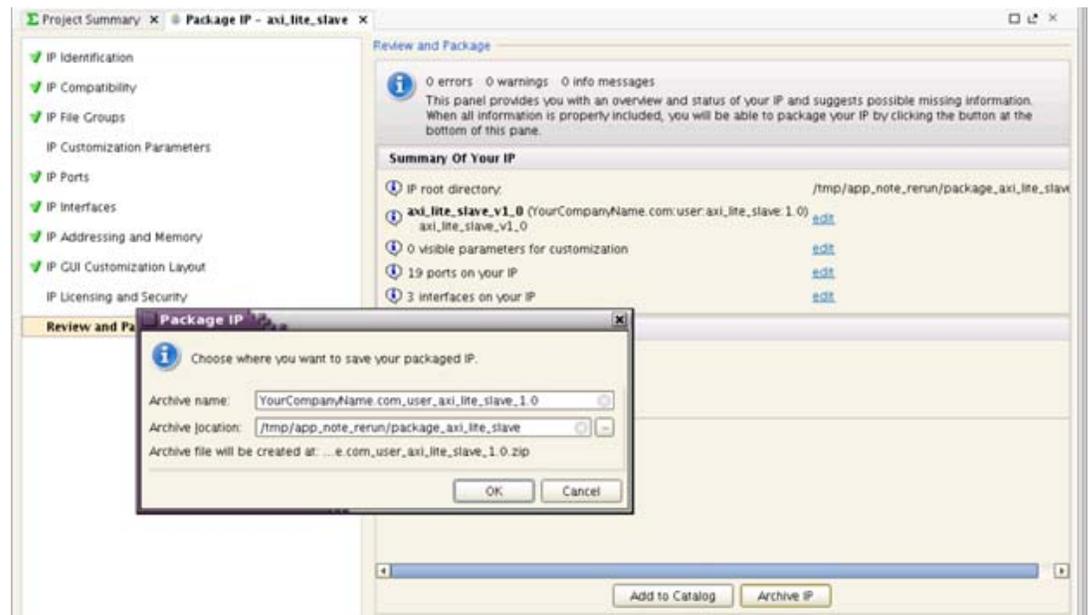


Figure 28: Package IP (Archive IP) Window

- The IP has been successfully packaged. Select **File > Close Project**.
Repeat [Package AXI4 IP](#) for additional cores that need to be packaged.

IP Integrator Project With Generated IP

AXI4-Lite System

This section describes integrating the Vivado packaged IP (created in [Package AXI4 IP](#)) into a project. This flow is similar to sharing IP between multiple teams, projects, or IP customers.

Start a New Vivado Project

1. If the previous section of generating IP was not completed, unzip the reference design files into a local folder (referred to as `<design_dir>`) and rename the `<design_dir>/ip_repo_complete` directory to `<design_dir>/ip_repo`.
2. Open the Vivado tools in Windows by selecting **Start > Xilinx Design Tools > Vivado** or enter the `vivado` command in Linux after setting up the Vivado design tools.
3. Create a new project by selecting **Getting Started > Create New Project**.
4. In the New Project window, click **Next**.
5. Set the project name to `ipi_lite_project`, and click **Next**.
6. Select **RTL Project** in the Project Type window, and click **Next**.
7. In the Add Sources window, select **Add Files...** and select `lite_system_wrapper_tb.v` from the `<design_dir>/tb/verilog` directory. This file is the test bench simulation file. The rest of the design will be created later in this section.
8. Choose **HDL Source for: Simulation only** and click **Next**.
9. Click **Next** in the Add Existing IP window.
10. Click **Next** in the Add Constraints window.
11. Keep the part selected by the Vivado Design Suite, and click **Next** in Default Part window.
12. Click **Finish** in the New Project Summary window.

Including IP Repository

1. Select **Window > IP Catalog**.
2. Click  in the IP Catalog Tab.
3. In the Project Settings window, click **Add Repository** ([Figure 29](#)).

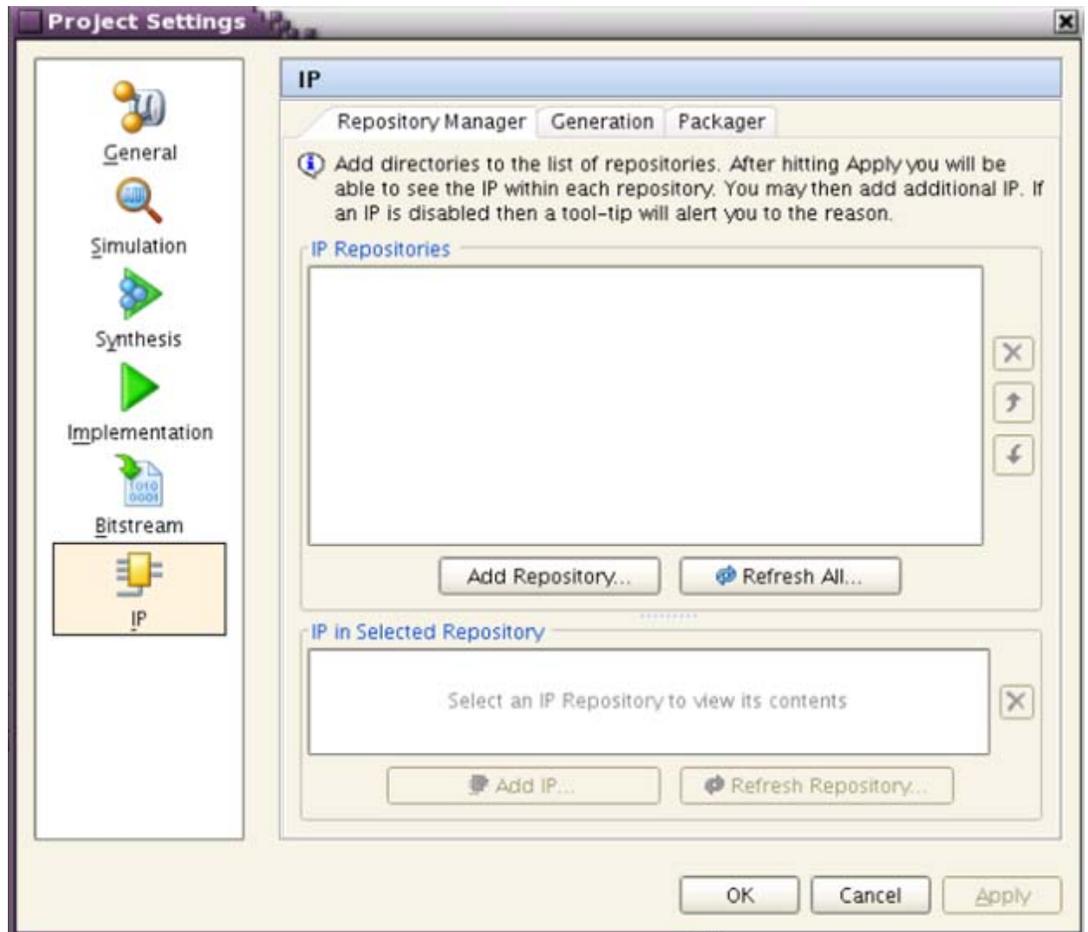


Figure 29: Project Settings: IP Repository Manager

4. If the IP in `<design_dir>/ip_repo` already appears in the IP in the IP Repositories window, click **OK** and skip to [step 1](#).
5. Find and select the `<design_dir>/ip_repo` directory and click **Select**. This makes this directory visible to the IP Catalog.
6. In the Project Settings window, click **Add IP...**
7. In the Select IP To Add To Repository window, select `YourCompanyName.com_user_axi_lite_master_1.0.zip` and click **OK** ([Figure 30](#)). This extracts the archived ZIP file into a usable form in `<design_dir>/ip_repo`.

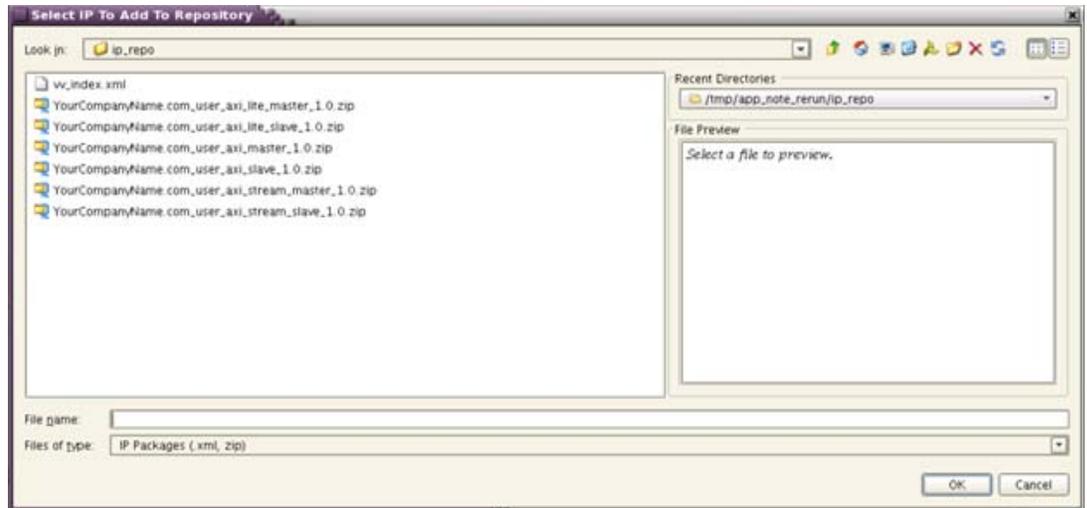


Figure 30: Select IP To Add To Repository Selecting AXI4-Lite Master

Repeat this procedure adding:

- YourCompanyName.com_user_axi_lite_slave_1.0.zip
- YourCompanyName.com_user_axi_master_1.0.zip
- YourCompanyName.com_user_axi_slave_1.0.zip
- YourCompanyName.com_user_axi_stream_master_1.0.zip
- YourCompanyName.com_user_axi_stream_slave_1.0.zip

The resulting Project Settings window is shown in Figure 31. Click **OK**.

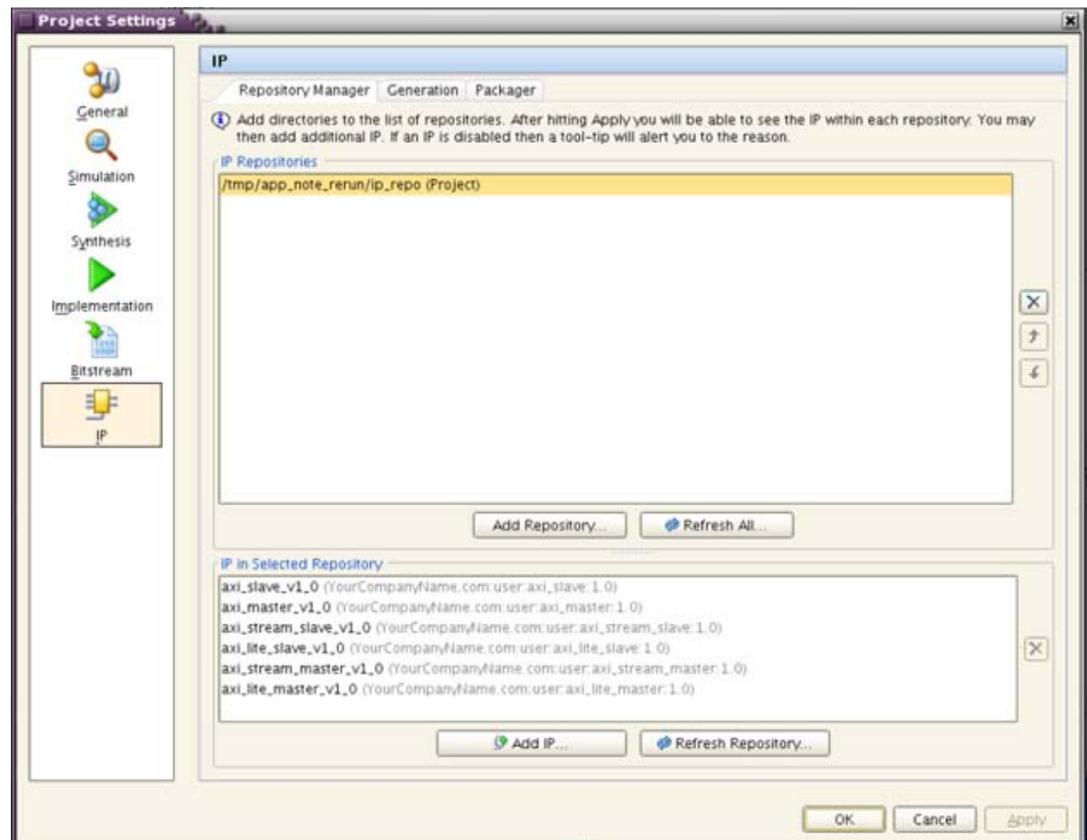


Figure 31: Project Settings Window with All IP Selected

Create Block Diagram System

1. Select **Flow > Create Block Design**.
2. In the Create Block Design window set:
 - Design name: lite_system
3. Right-click **Add IP...** to add IP to the canvas.
4. In the Search box, type axi_lite_master (Figure 32) and press Enter.

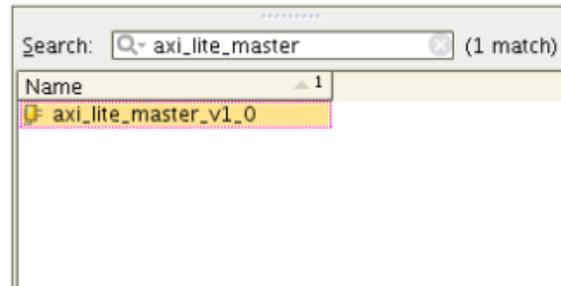


Figure 32: IP Integrator IP Search Box

5. Repeat [step 4](#) for axi_lite_slave.
6. Right-click on the canvas and select **Create Port...**

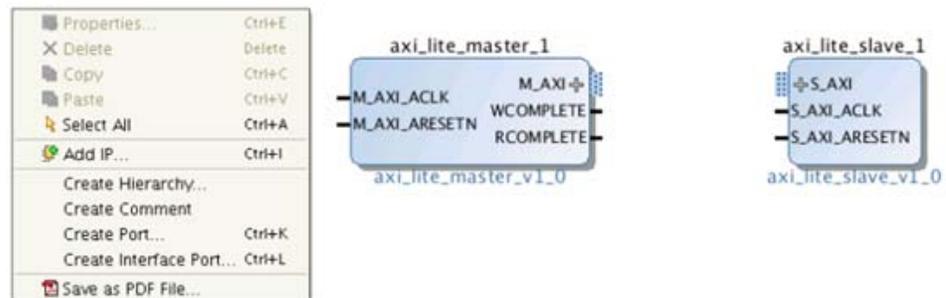


Figure 33: IP Integrator Canvas: Creating Port

7. In the Create Port window, enter the following and click **OK** (Figure 34).
 - Port name: ACLK
 - Type: Clock
 - Frequency: 100

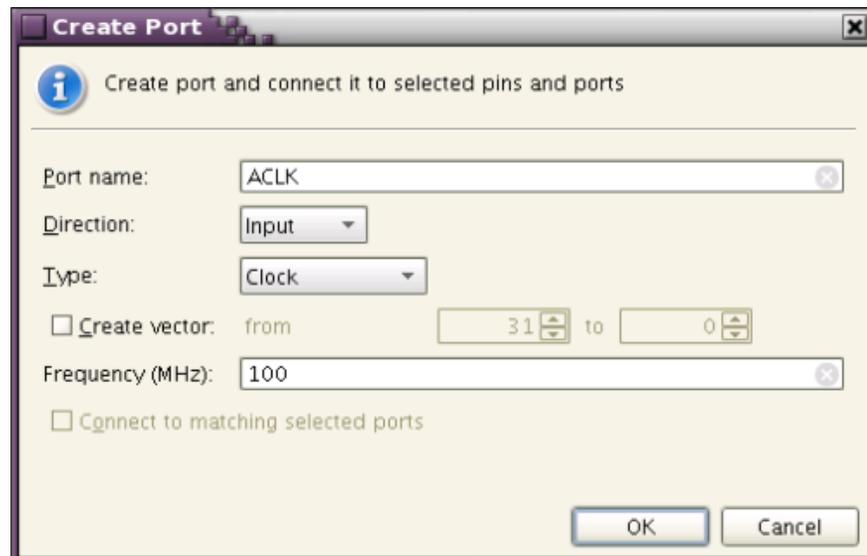


Figure 34: Create Port Window: Adding ACLK Port

8. Right-click on the canvas and select **Create Port...**
9. In the Create Port window, enter the following and click **OK** (Figure 35).
 - Port name: ARESETN
 - Direction: Input
 - Type: Reset



Figure 35: Create Port Window: Adding ARESETN

10. Connect the ACLK port to:
 - M_AXI_ACLK of axi_lite_master_1
 - S_AXI_ACLK of axi_lite_slave_1
11. Connect the ARESETN port to:
 - M_AXI_ARESETN of axi_lite_master_1
 - S_AXI_ARESETN of axi_lite_slave_1
12. Connect the M_AXI interface of axi_lite_master_1 to the S_AXI interface of axi_lite_slave_1.

Figure 36 shows the completed schematic.

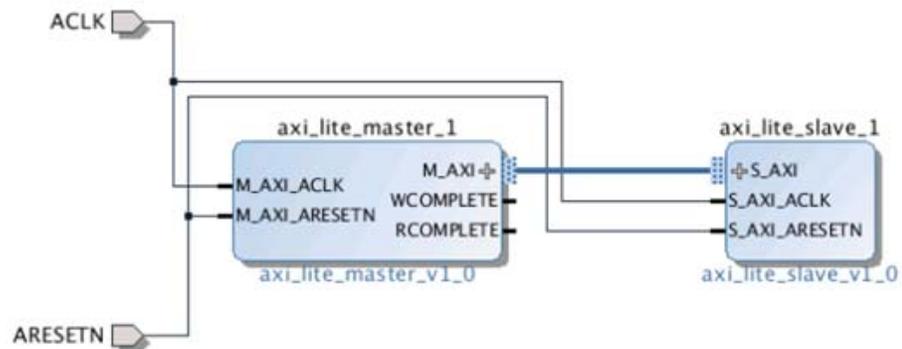


Figure 36: IP Integrator Canvas Connected AXI4-Lite System

13. Select the **Address Editor** tab of the canvas.
14. Select M_AXI, right-click and select **Auto Assign Address** (Figure 37). This chooses an AXI address for the axi_lite_slave.

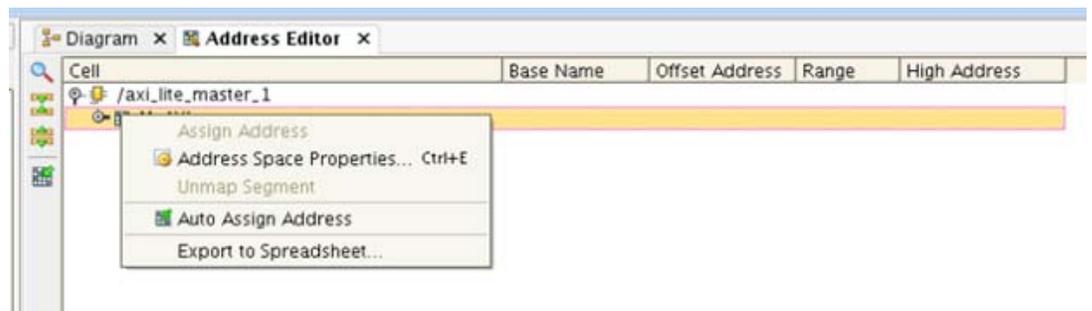


Figure 37: Address Editor Selecting Auto Assign Address for axi_lite_master

15. Select **Tools > Validate Design** (Figure 38) and click **OK**.



Figure 38: IP Integrator Canvas: Validation Successful Window

16. Select **File > Save Block Design**.

System Integration and Simulation

1. Select **Window > Sources**.
2. In the Sources Window, select **lite_system**, right-click and select **Create HDL Wrapper** (Figure 39). This generates a HDL wrapper because the block diagram is not able to be the

top-level object in a design. The generated file will then be replaced with a pre-existing test bench wrapper instead.

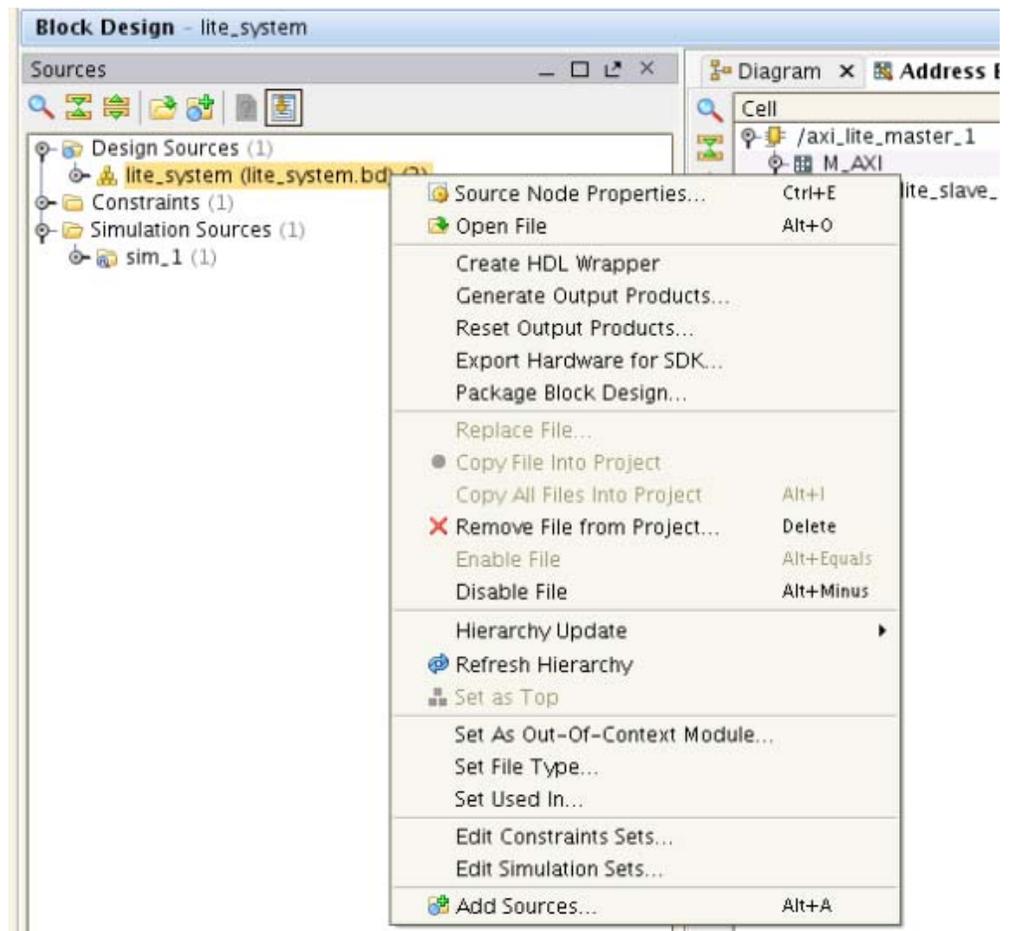


Figure 39: Sources Window: Create HDL Wrapper

3. Select **Flow Navigator > Simulation > Simulation Settings**.

4. On the Simulation tab, change the following value and click **OK** (Figure 40).
 - Simulation Run Time:

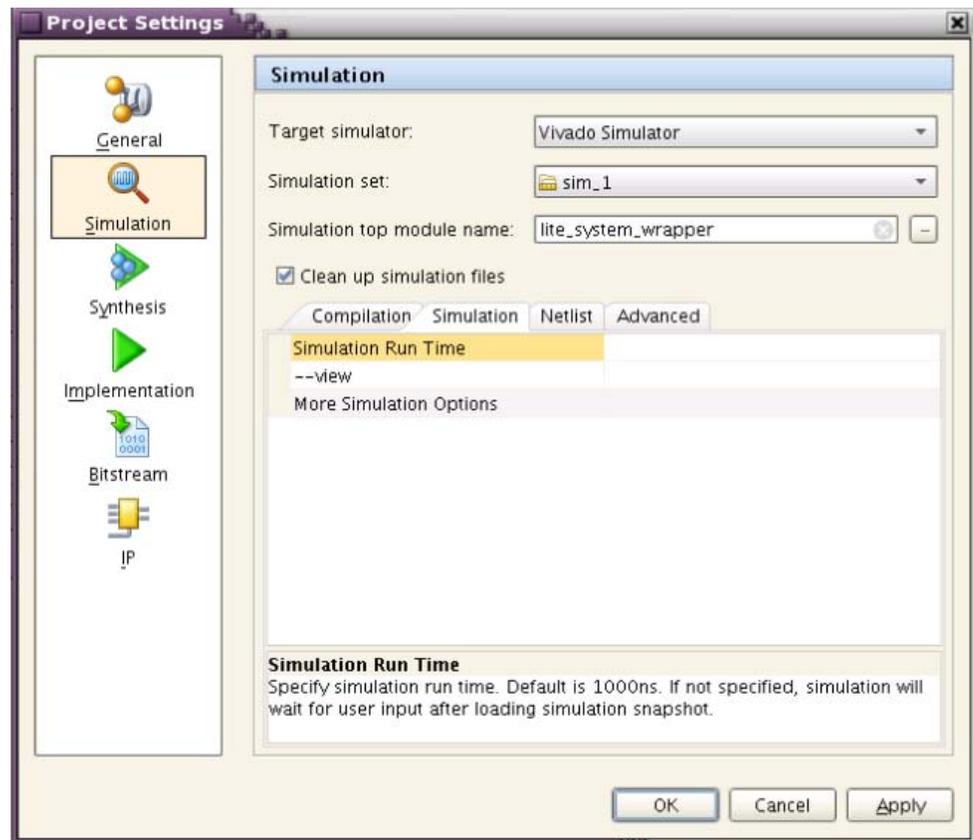


Figure 40: Project Settings Simulation Window: Run Time Option

5. Select **Flow Navigator > Simulation > Run Simulation** and click **Run Behavioral Simulation**.
6. In the Scopes window, expand **lite_system_wrapper**, and drag the **lite_system_wrapper_tb > lite_system_wrapper_i > lite_system_i instance** to the Untitled 1 wave window to allow the AXI Interface from the block diagram to be captured by simulation.
7. Select **Run > Run All**.
8. Select **View > Zoom Fit**. The resulting waveform is shown in Figure 41. Four writes can be seen, followed by four reads.
9. When ready, select **File > Close Design** to prepare for the next section.

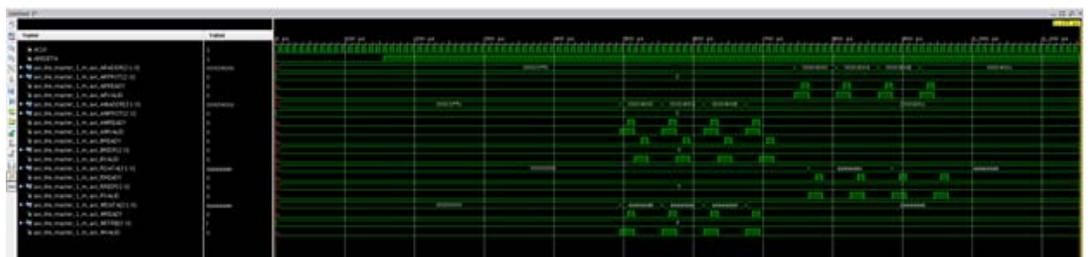


Figure 41: AXI4-Lite IP Integrator System Simulation

AXI4 System

When building an AXI4 system containing an AXI4 Master and an AXI4 Slave, follow the same instructions for building an [AXI4-Lite System](#) with the following changes.

Start a New Vivado Project

- In [step 5](#), set Project name to ipi_axi_project.
- In [step 7](#), select axi_system_wrapper_tb.v in the Add Sources dialog.

Create Block Diagram System

- In [step 2](#), set Design name to axi_system.
- In [step 4](#), type in axi_master.
- In [step 5](#), type: axi_slave.
- In [step 10](#), Connect ACLK port to:
 - M_AXI_ACLK of axi_master_1
 - S_AXI_ACLK of axi_slave_1
- In [step 11](#), Connect ARESETN port to:
 - M_AXI_ARESETN of axi_master_1
 - S_AXI_ARESETN of axi_slave_1
- In [step 12](#), connect the M_AXI interface of axi_master_1 to the S_AXI interface of axi_slave_1.

[Figure 42](#) shows the completed schematic.

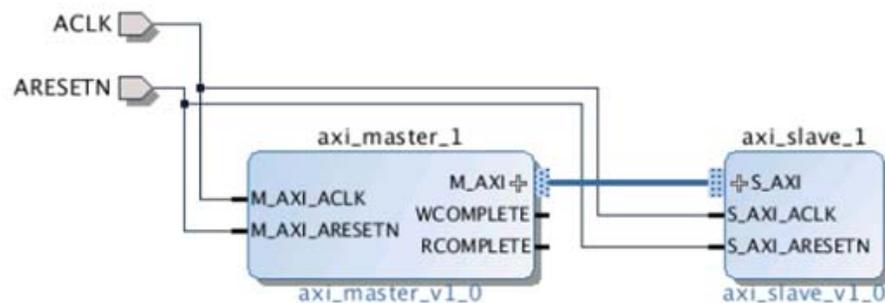


Figure 42: IP Integrator Canvas with Completed AXI4 System

Following [step 16](#), [Figure 43](#) is the result. The warning indicated in the figure is a result of the AXI4 Master and AXI4 Slave having different ID widths. The example AXI4 Master does not have ID bits because it does not support or generate multiple threads. The example AXI4 Slave does have the ports, and the IP integrator indicates a mismatch in width and ties the AxID ports to zero.



Figure 43: IP Integrator Canvas Validation Successful Window with ID_WIDTH Warning

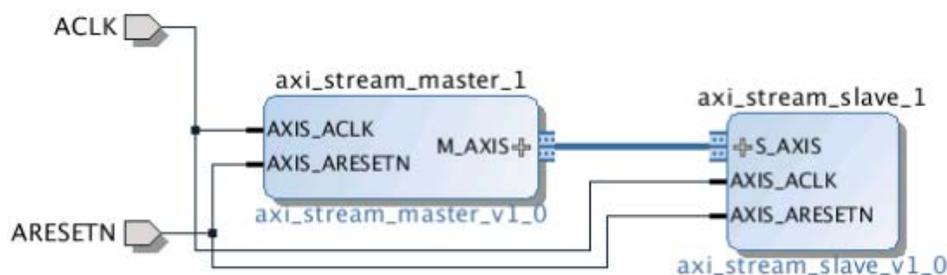


Figure 45: IP Integrator Canvas with Completed AXI4-Stream System

- Skip [step 14](#) because AXI4-Stream does not use addresses.

System Integration and Simulation

- In [step 2](#), in the Sources Window select **axi_stream_system**, right-click and select **Create HDL Wrapper**.
- In [step 6](#), in the Scopes window, expand **axi_stream_system_wrapper_tb**, and drag the **axi_stream_system_wrapper_tb > axi_stream_system_wrapper_i > axi_stream_system_i** instance to the Untitled wave window

The resulting waveform is shown in [Figure 46](#).



Figure 46: AXI4-Stream IP Integrator System Simulation

Reference Design

The reference design files for this application note can be downloaded at:

<https://secure.xilinx.com/webreg/clickthrough.do?cid=338300>

References

This document uses the following references:

1. [Vivado Design Suite Documentation](#)
2. [Advanced Microcontroller Bus Architecture \(AMBA\) ARM AXI4 Specifications](#)
3. AXI Reference Guide ([UG761](#))

Revision History

The following table shows the revision history for this document.

Date	Version	Description of Revisions
06/01/2013	1.0	Initial Xilinx release.

Notice of Disclaimer

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF

MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of the Limited Warranties which can be viewed at <http://www.xilinx.com/warranty.htm>; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in Critical Applications: <http://www.xilinx.com/warranty.htm#critapps>.