Vivado Design Suite Tutorial

Designing IP Subsystems Using IP Integrator

UG995 (v2014.3) October 1, 2014

XILINX
ALL PROGRAMMABLE™

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

©Copyright 2013-2014 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, Vivado, ISE, Kintex, Spartan, UltraScale, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.

Revision History

Date	Version	Revision
10/01/2014	2014.3	Validated with this release.
06/04/2014	2014.2	Validated with this release.
04/02/2014	2014.1	Validated with this release. Added AXI Interrupt Controller block in Step 1: Creating a Project. Added CONCAT and CONST cores in Step 6: Managing Signals with CONCAT and CONSTANT blocks. Associated ACLK to ARESETN in Step 8: Validating the Design.

Table of Contents

Revision History 2

Table of Contents 3

Designing IP Subsystems 4

Introduction 4

Tutorial Design Description 4

Hardware and Software Requirements 5

Locating Tutorial Design Files 5

Lab 1: Designing IP Subsystems in IP Integrator 6

Step 1: Creating a Project 6

Step 2: Create an IP Integrator Design 9

Step 3: Creating External Connections 14

Step 4: Customize IP 20

Step 5: Running Connection Automation 24

Step 6: Managing Signals with CONCAT and CONSTANT blocks 29

Step 7: Using the Address Editor 33

Step 8: Validating the Design 35

Step 9: Creating and Implementing the Top-Level Design 38

Conclusion 43

Designing IP Subsystems

IMPORTANT: *This tutorial requires the use of the Kintex®-7 family of devices. You will need to update your Vivado tools installation if you do not have this device family installed. Refer to the Vivado Design Suite User Guide: Release Notes, Installation, and Licensing (UG973) for more information on Adding Design Tools or Devices.*

Introduction

The Xilinx® Vivado® Design Suite IP integrator tool lets you create complex system designs by instantiating and interconnecting IP cores from the Vivado IP catalog onto a design canvas. You can create designs interactively through the IP integrator design canvas GUI, or programmatically using a Tcl programming interface. You will typically construct designs at the AXI-interface level for greater productivity; but you may also manipulate designs at the port level for more precise design control.

This tutorial walks you through the steps for building a basic IP subsystem design using the IP integrator tool. You will instantiate a few IP in the IP integrator tool and then stitch them up to create an IP sub-system design. While working through this tutorial, you will be introduced to the IP integrator GUI, run design rule checks (DRC) on your design, and then integrate the design into a top-level design in the Vivado Design Suite. Finally, you will run synthesis and implementation and generate a bitstream on the design.

VIDEO: *You can also learn more about the Vivado Design Suite IP integrator feature by viewing the quick take video at http://www.xilinx.com/training/vivado/designing-with-vivado-ip-integrator.htm.*

Tutorial Design Description

This tutorial is based on a simple non-processor based IP integrator design. It contains a few peripheral IP cores and an AXI interconnect core, which connects to an external on-board processor.

The design targets an xc7k325 Kintex device**.** A small design is used to allow the tutorial to be run with minimal hardware requirements and to enable timely completion of the tutorial, as well as to minimize the data size.

Hardware and Software Requirements

This tutorial requires that the 2014.3 Vivado Design Suite software release or later is installed. The following partial list describes the operating systems that the Vivado Design Suite supports on x86 and x86-64 processor architectures:

Microsoft Windows Support:

- Windows 8.1 Professional (32-bit and 64-bit), English/Japanese
- Windows 7 and 7 SP1 Professional (32-bit and 64-bit), English/Japanese

Linux Support:

- Red Hat Enterprise Workstation 6.4 and 6.5 (32-bit and 64-bit)
- SUSE Linux Enterprise 11 (32-bit and 64-bit)
- Cent OS 6.4 and 6.5 (64-bit)

Refer to the *Vivado Design Suite User Guide: Release Notes, Installation, and Licensing (UG973)* for a complete list and description of the system and software requirements.

Locating Tutorial Design Files

You can find the files for this tutorial in the Vivado Design Suite examples directory at the following location:

- **<Vivado_install_area>/Vivado/<version>/examples/Vivado_Tutorial**

You can also extract the provided zip file, at any time, to write the tutorial files to your local directory, or to restore the files to their starting condition.

Extract the zip file contents from the software installation into any write-accessible location.

- **<Vivado_install_area>/Vivado/<version>/examples/Vivado_Tutorial.zip**

The location of the extracted Vivado_Tutorial directory is referred to as the <*Extract_Dir*> in this Tutorial.

Lab 1: Designing IP Subsystems in IP Integrator

Step 1: Creating a Project

Figure 1: Vivado IDE Getting Started Page

Open the Vivado IDE:

- On Linux,

1. Change to the directory where the Vivado tutorial designs are stored:

   cd <Extract_Dir>/Vivado_Tutorial

2. Launch the Vivado Design Suite: **vivado**

- On Windows,

1. Launch the Vivado Design Suite:

   Start > All Programs > Xilinx Design Tools > Vivado 2014.x > Vivado 2014.x[1]

[1] Your Vivado Design Suite installation may called something different than **Xilinx Design Tools** on the **Start** menu.

2. As an alternative, click the **Vivado 2014.x** Desktop icon to start the Vivado IDE.

   The Vivado IDE Getting Started page contains links to open or create projects and to view documentation, as shown in Figure 1.

3. In the Quick Start page, select **Create New Project**.
4. The New Project wizard opens. Click **Next** to confirm the project creation.
5. In the Project Name page, set the following options, and click **Next**:
   - Type the project name: `project_ipi`
   - Enter the project location:

     `<project_directory>`

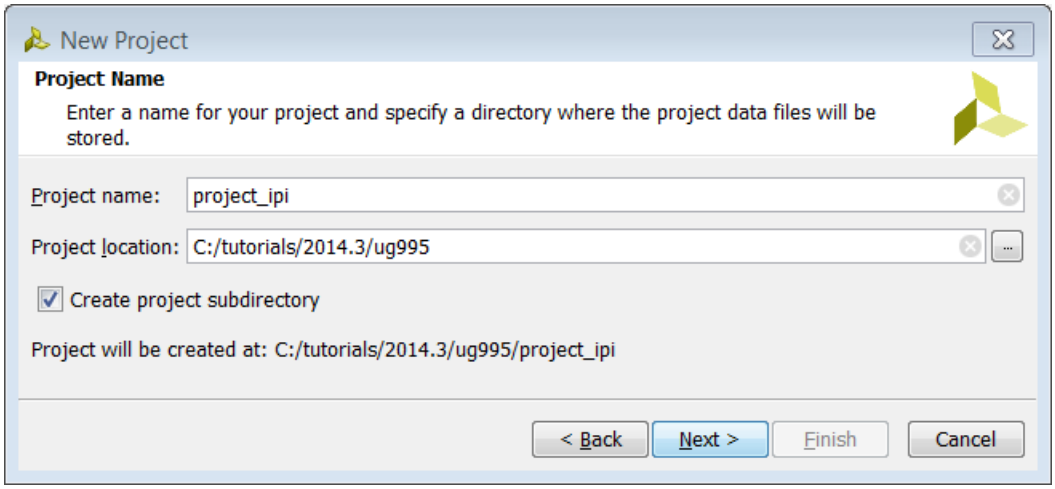

Figure 2: Create Project

6. Ensure that **Create project subdirectory** is checked, and click **Next**.

7. In the **Project Type** dialog box, select **RTL Project**. Click **Next**.

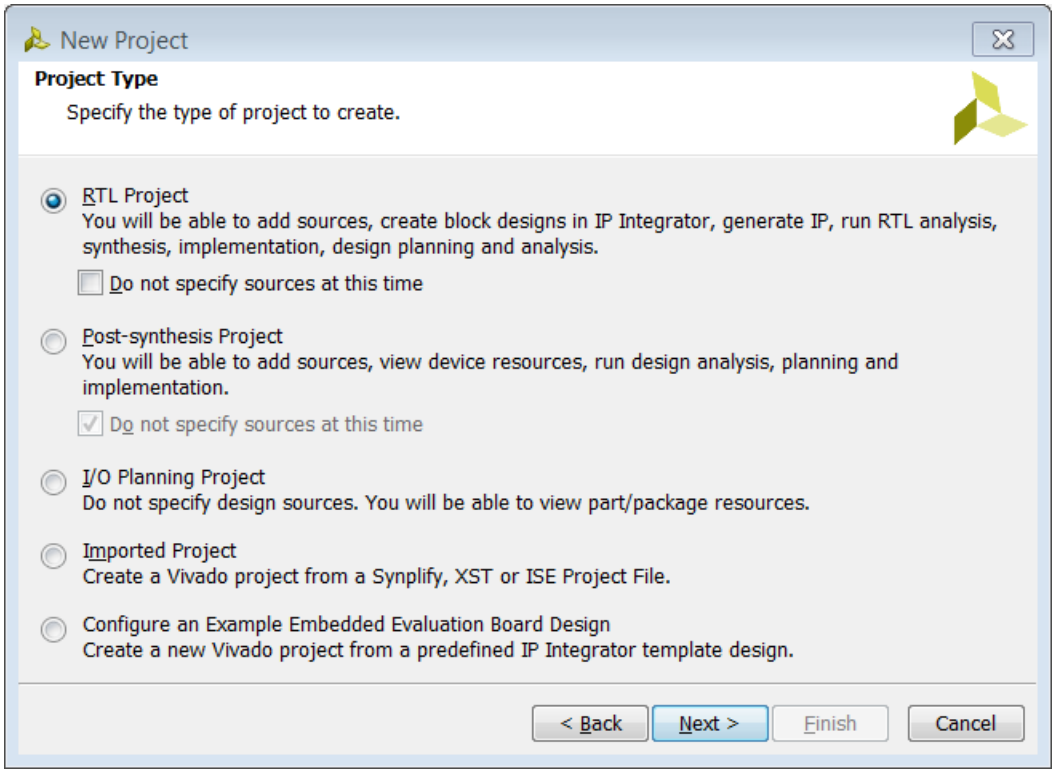

Figure 3: Specify Project Type

8. In the **Add Sources** dialog box:
   - Set the Target language to VHDL,
   - Set the Simulator Language to Mixed,
   - Click Next.

   You will add sources later using the design canvas in the Vivado IP integrator to create a subsystem design.
9. In the **Add Existing IP** dialog box, click **Next**.
10. In the **Add Constraints** dialog box, click **Next**.
11. In the **Default Part** dialog box:
    - Specify **Parts**
    - Set **Family:** Kintex-7
    - Set **Speed grade:** -2

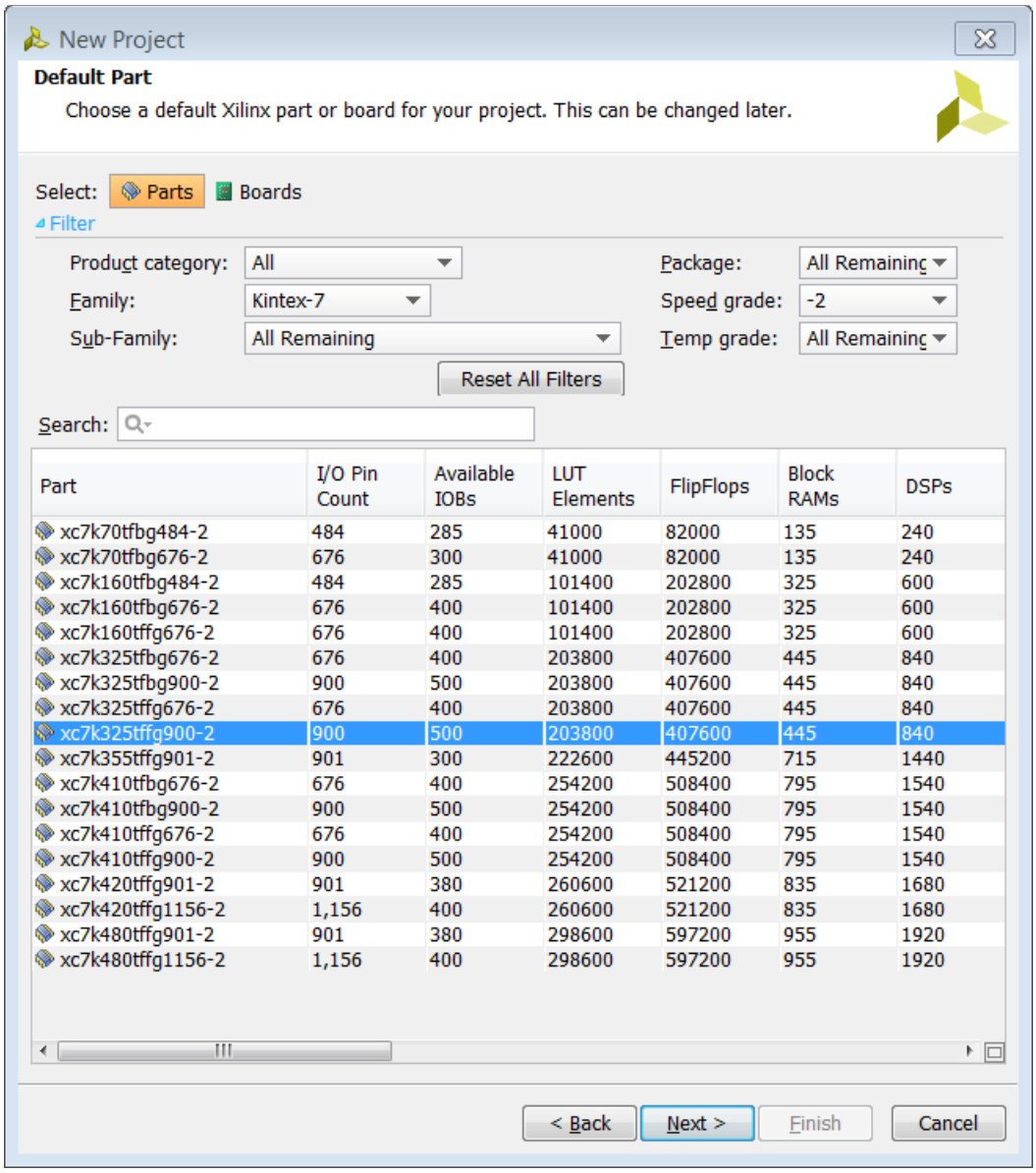

Figure 4: Select Default Part

12. Choose the **xc7k325tffg900-2** part from the listed parts, and click **Next**.
13. Review the project summary in the **New Project Summary** dialog box.
14. Click **Finish** to create the project.

    The new project opens in the Vivado IDE.

Step 2: Create an IP Integrator Design

1. In the Flow Navigator, select the **Create Block Design** option.

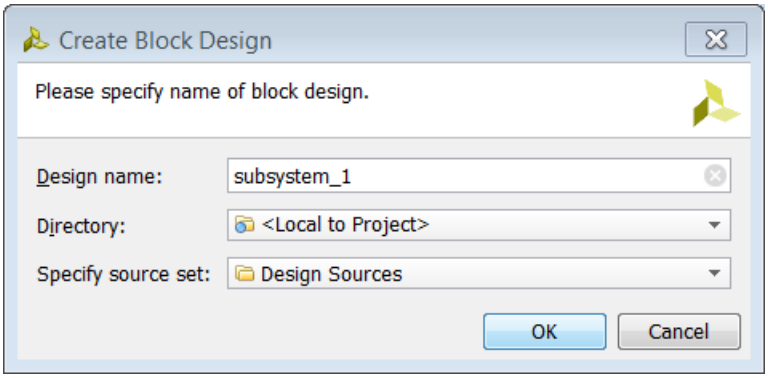

Figure 5: Create Block Design dialog box

2. On the **Create Block Design** dialog box:
   - Specify **Design Name** as **subsystem_1**,
   - Set **Directory** to **Local to Project**,
   - Set **Specify source set** to **Design Sources**
   - Click **OK**.

   The Vivado IP integrator displays a design canvas to let you quickly create complex subsystem designs by integrating IP cores.
3. Right-click on the design canvas to open the popup menu and select **Add IP**.

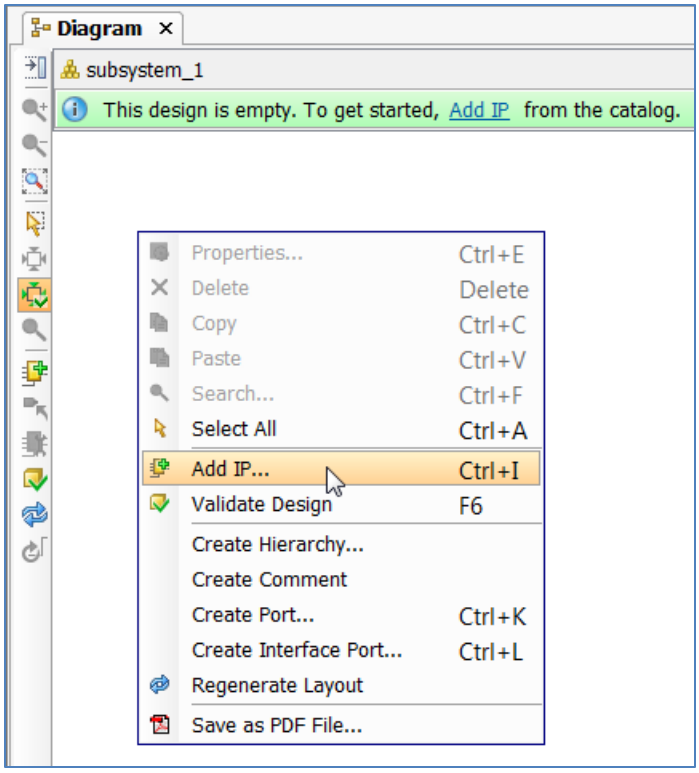

Figure 6: Add IP in Design Canvas

Alternatively, you can also click the **Add IP** link in the IP integrator canvas, or click on the **Add IP** button in the IP integrator sidebar menu.

The IP Catalog opens as seen in Figure 7.

TIP: *If the IP Catalog does not display the same number of columns as shown in Figure 7, place your mouse in the column headers of the IP catalog, right click to open the menu, and select which fields of the IP catalog you would like to view.*

4. In the search field of the IP Catalog, type **spi** to find the AXI Quad SPI.

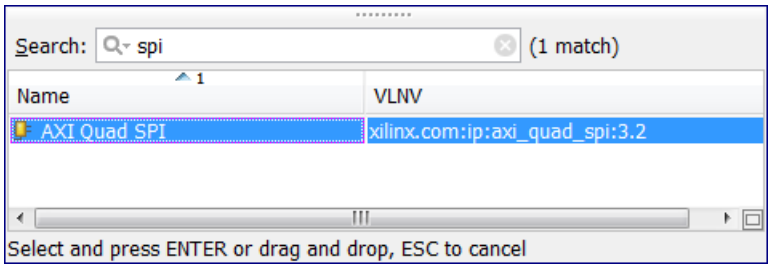

Figure 7: Instantiate AXI Quad SPI IP

5. Select the **AXI Quad SPI** core and press enter on the keyboard, or simply double click the core in the IP Catalog.

The AXI Quad SPI core is instantiated onto the IP integrator design canvas.

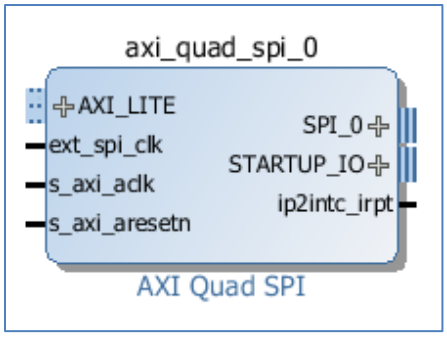

Figure 8: An Instantiated AXI Quad SPI

6. Right-click on the IP integrator canvas to **open the popup menu**, and select **Add IP**.
7. In the **Search** field of the IP integrator catalog, type **IIC**.
8. Either double-click or press ***Enter*** on your keyboard to instantiate the AXI IIC IP.

9. Use the **Add IP** command to instantiate the flowing IP cores:
   - AXI Uartlite,
   - AXI BRAM Controller,
   - AXI Interrupt Controller
   - AXI Interconnect

   The IP integrator canvas should look similar to Figure 9. The relative positions of the blocks placed on the canvas may be slightly different.

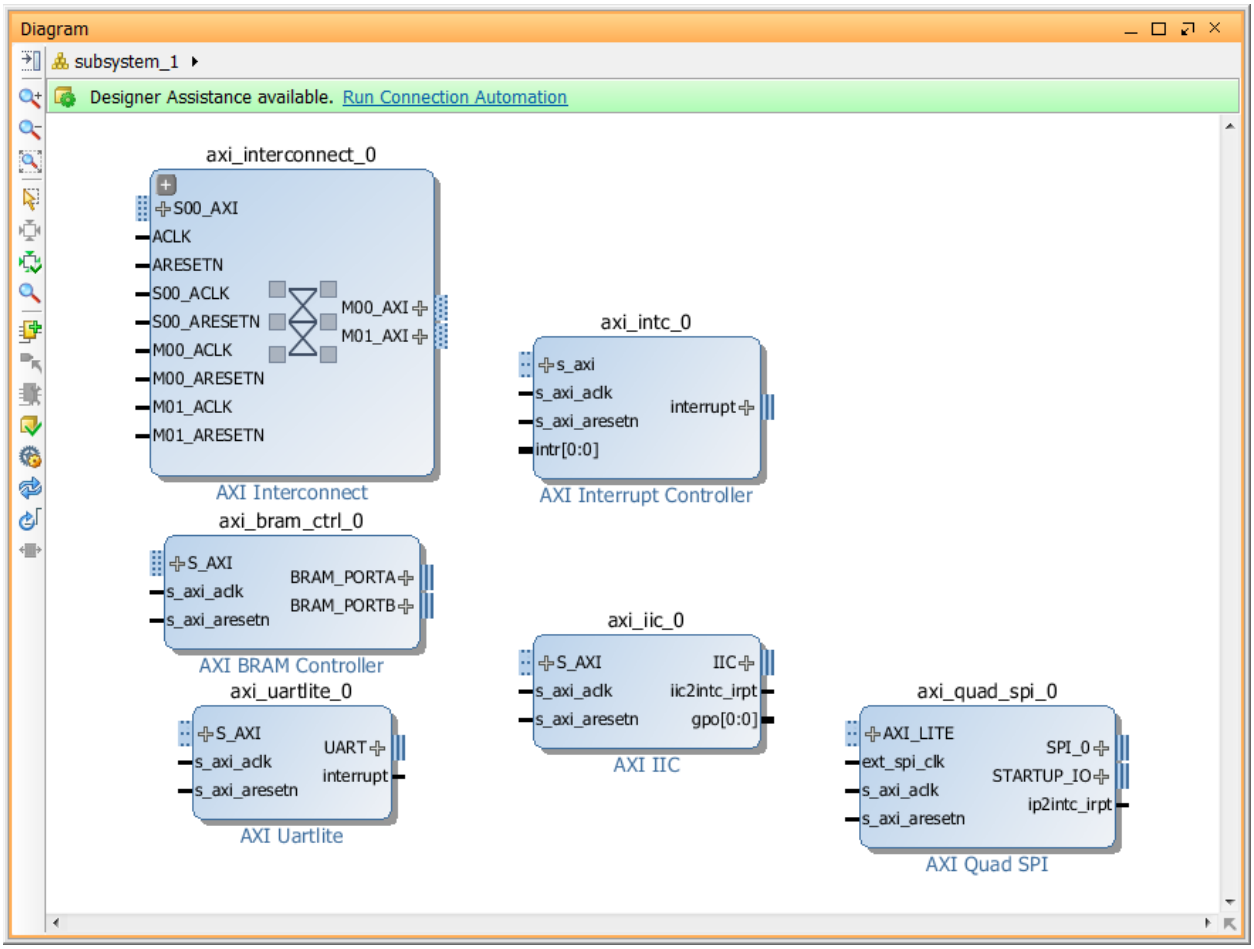

Figure 9: IP Integrator Design Canvas

Step 3: Creating External Connections

At this point, you have instantiated several AXI slaves that can be accessed using an external master such as an on-board processor. To connect to an external master controlling these slaves, you will connect the S00_AXI interface pin on the AXI Interconnect to an external port.

An interface is a grouping of signals that share a common function, containing both individual signals and multiple buses. By grouping these signals and buses into an interface, the Vivado IP integrator can identify common interfaces and automatically make multiple connections in a single step. See the *Vivado Design Suite User Guide: Designing IP Subsystems Using IP Integrator* (UG994) for more information on interface pins and ports.

1. Right-click on the **S00_AXI** interface pin on the AXI Interconnect to open the popup menu, and select **Create Interface Port**.

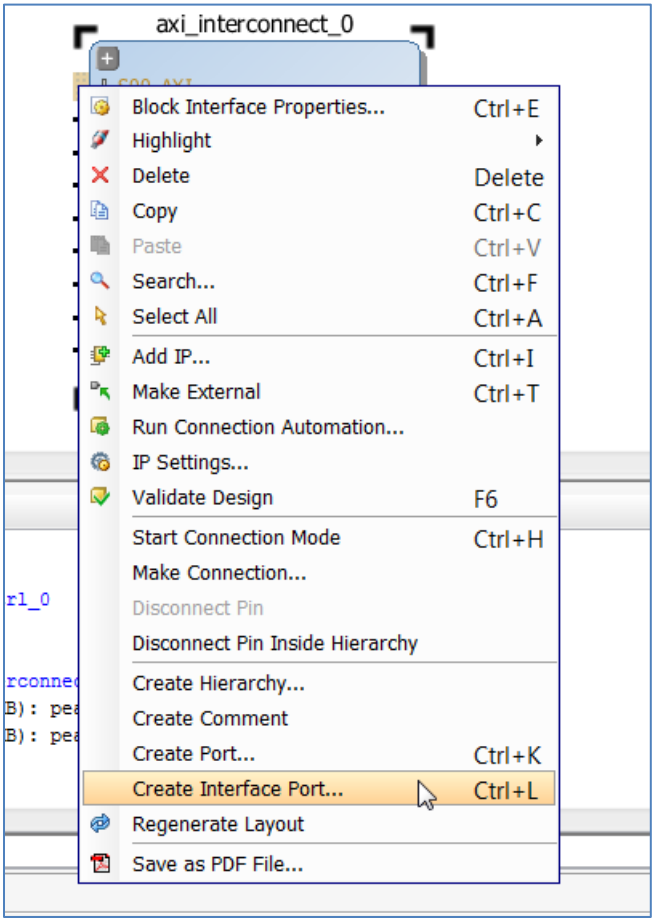

Figure 10: Right Click on S00_AXI Interface Pin

The Create Interface Port dialog box opens up, as shown in Figure 11.

2. Make sure the checkbox **Connect to selected interface S00_AXI** is selected.

3. Click **OK** to accept the default settings.

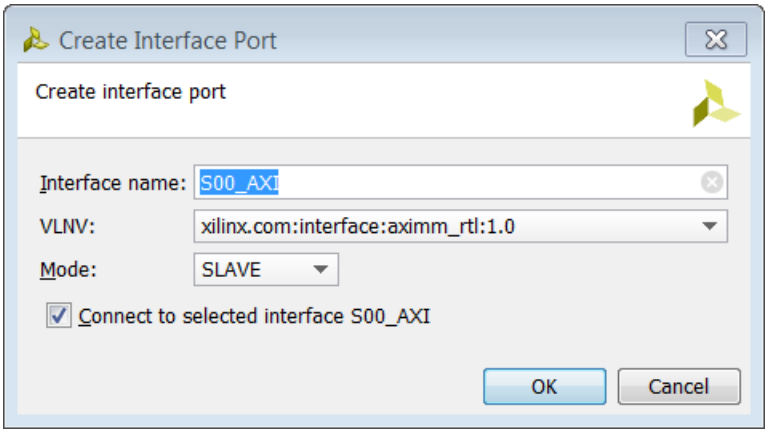

Figure 11: Create Interface Port

The Vivado IP integrator adds the external S00_AXI interface port to the subsystem design, and automatically connects it to the S00_AXI interface pin on the AXI Interconnect core.

On the AXI Interconnect, connect the Clock and the Reset pins to external ports using the Create Port command. Since these are not interface pins, you will not need an interface port to connect them.

4. Right-click the ACLK pin of the AXI Interconnect, and select **Create Port**.

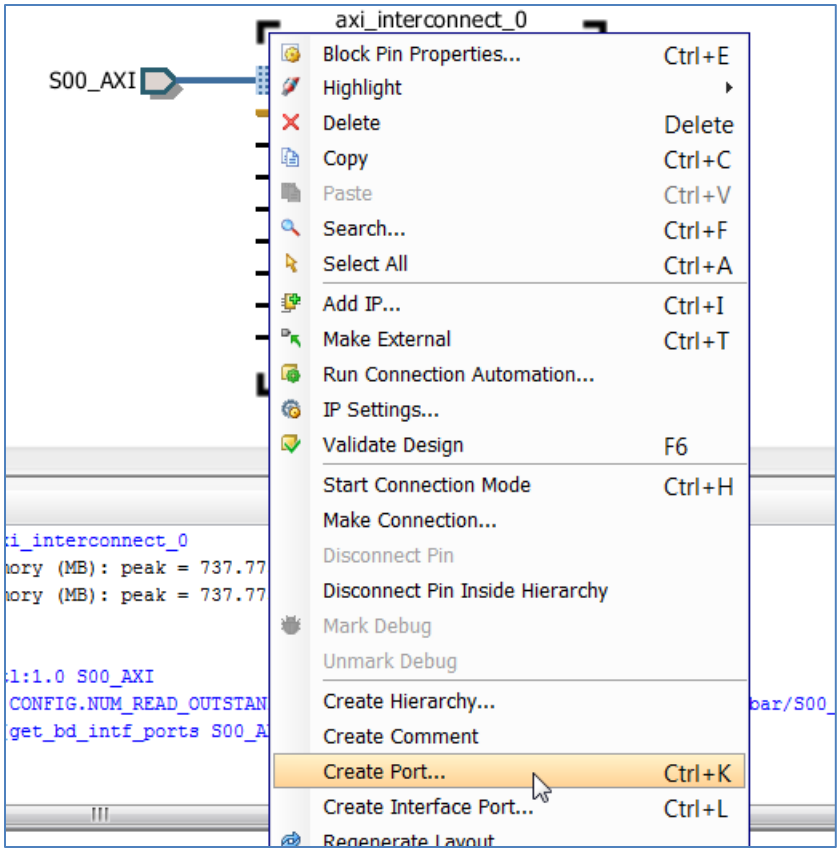

Figure 12: Create Port

5. In the **Create Port** dialog box, as shown in Figure 13, specify the **frequency** as **200 MHz**, leaving everything else at its default value.
6. Click **OK**.

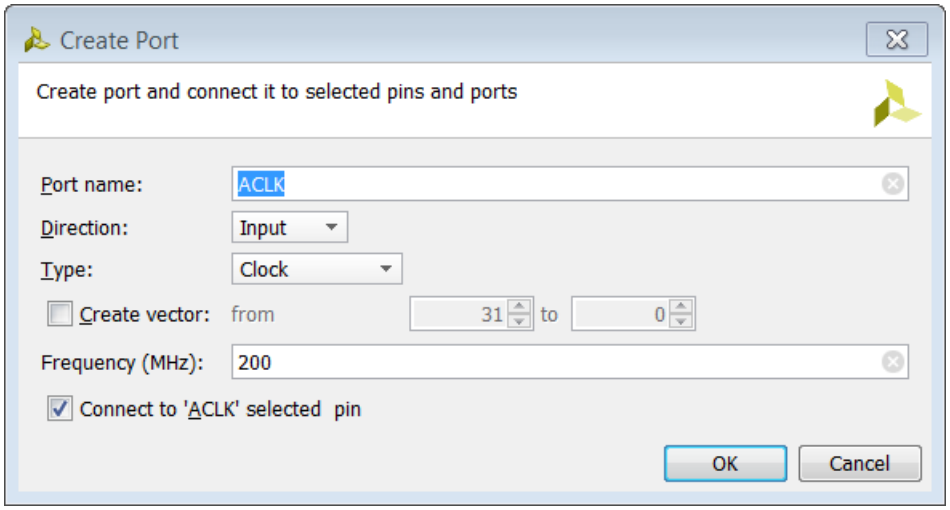

Figure 13: Create Port - ACLK

7. Right-click the **ARESETN** pin of the AXI Interconnect and select **Create Port.**

   The Create Port dialog box opens.

8. Click **OK** to accept the default settings.

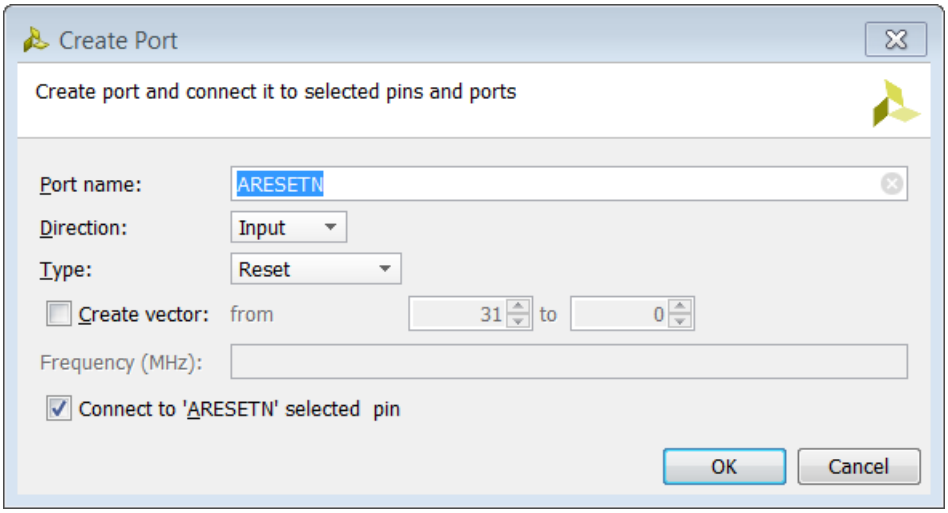

Figure 14: Create Port - ARESETN

Now you will connect the AXI clock and reset nets to the remaining master and slave clocks and resets of the AXI Interconnect.

9. Place the cursor on top of the **S00_ACLK** pin of the AXI Interconnect.

   Notice that the cursor changes into a pencil indicating that a connection can be made from that pin. Clicking the mouse button here starts a connection on the S00_ACLK pin.

10. Click and drag the cursor from the S00_ACLK pin to the ACLK port as shown in Figure 15.

TIP: *You must press and hold down the mouse button while dragging the connection from the S00_ACLK pin to the ACLK port.*

As you drag the connection wire, a green checkmark appears on the ACLK port indicating that a valid connection can be made between these points. The Vivado IP integrator highlights all possible connection points in the subsystem design as you interactively wire the pins and ports.

11. Release the mouse button and Vivado IP integrator makes a connection between the S00_ACLK pin and the ACLK port, as shown in Figure 15.

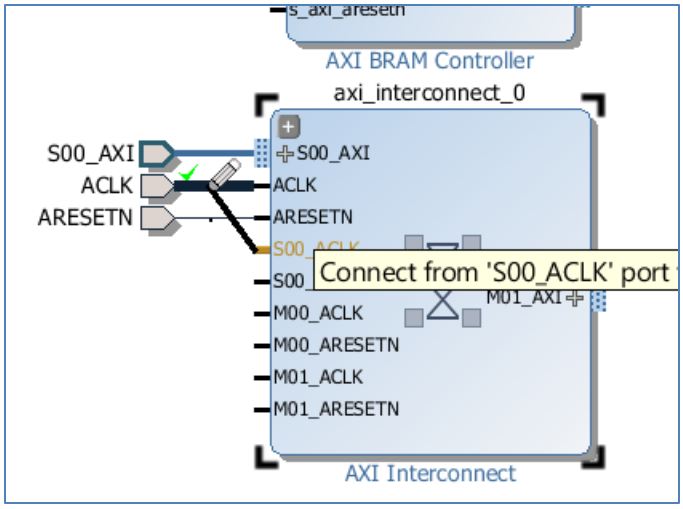

Figure 15: Connect the S00_ACLK pin to the ACLK port

12. Repeating the steps outlined above, connect the **M00_ACLK** and the **M01_ACLK** to the ACLK port.

    The connections to the AXI Interconnect should now appear as shown in Figure 16.

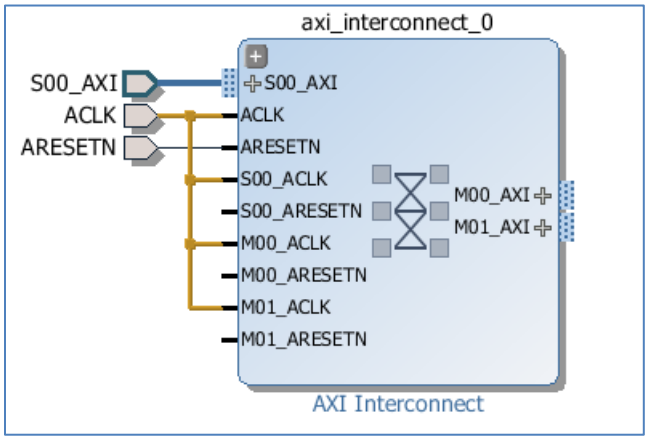

Figure 16: ACLK connections

    Similarly connect the reset pins of all the masters and slaves to the ARESETN port.

13. Place the cursor on the **S00_ARESETN** pin, click and drag the cursor to the **ARESETN** port.
14. Release the mouse button to make the connection.

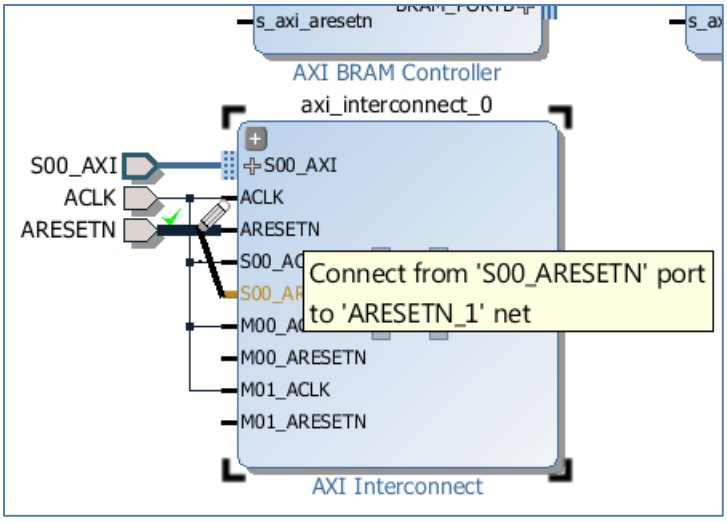

Figure 17: Connect S00_ARESETN to the ARESETN port

15. Repeat the steps to connect the **M00_ARESETN** and the **M01_ARESETN** pins of the AXI Interconnect to the ARESETN port, as shown in Figure 18.

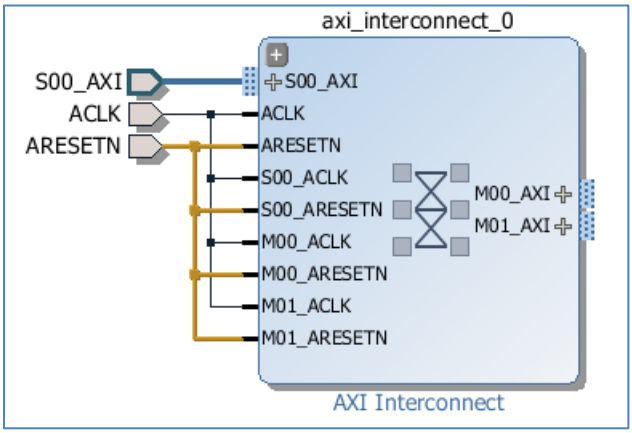

Figure 18: ARESETN Connections

Step 4: Customize IP

1. Double-click on the **AXI Interconnect** core to open the Re-Customize IP dialog box as seen in Figure 19.

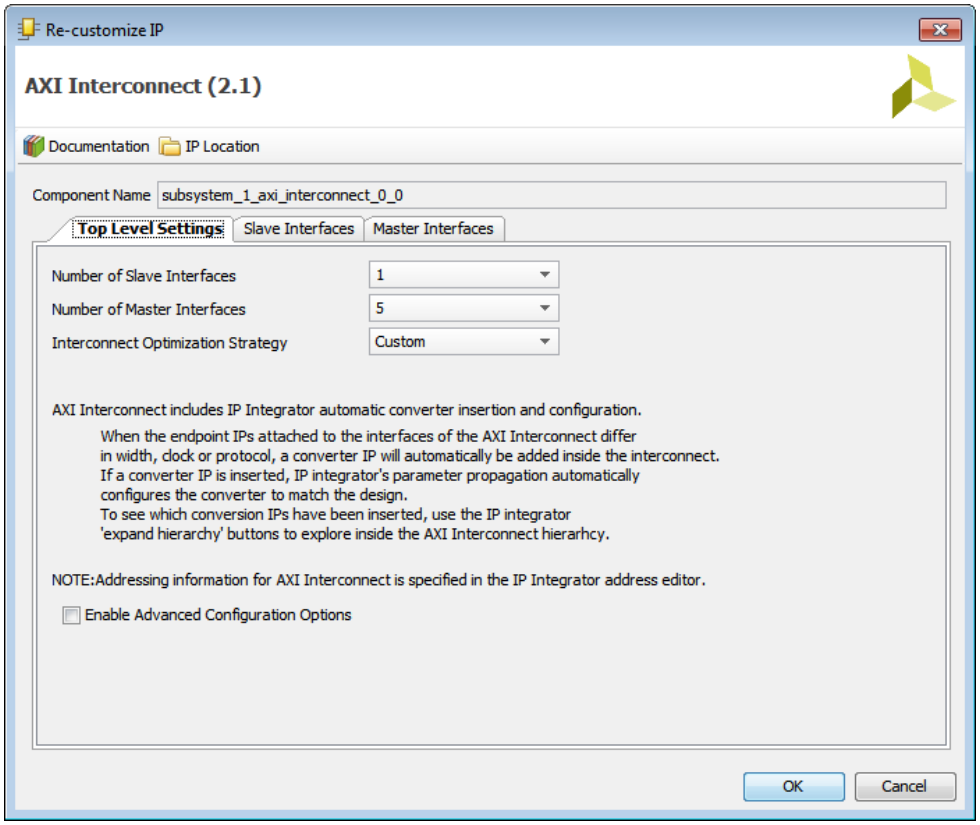

Figure 19: Re-customize the AXI Interconnect

2. From the Top Level Settings Tab, set the **Number of Master Interfaces** to **5** from the pull down menu.
3. Leave all the remaining options set to their default values, and click **OK**.

   The Vivado IP integrator re-customizes the AXI Interconnect, changing the number of master interfaces to five, and adding the necessary clock and reset pins to support these new master interfaces, as shown in Figure 20.

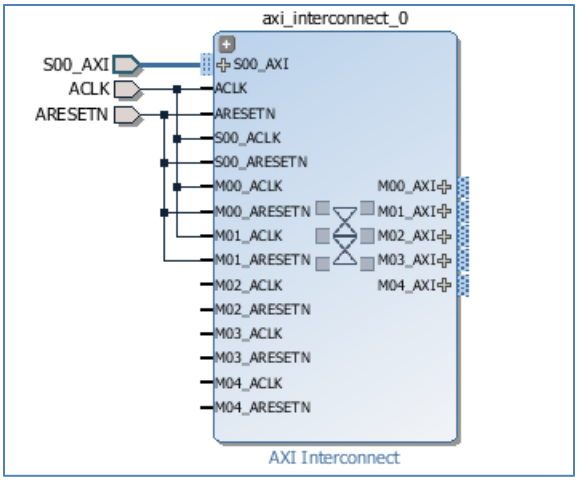

Figure 20: AXI Interconnect with 5 Master Interfaces

4. Connect all the **new clocks** to the ACLK port, and the new **resets** to the ARESETN port.

   Now you can connect all five slave IP cores to the AXI Interconnect.

5. Connect the **S_AXI** interface of the AXI BRAM Controller to **M00_AXI** interface of the AXI Interconnect, as shown in Figure 21.

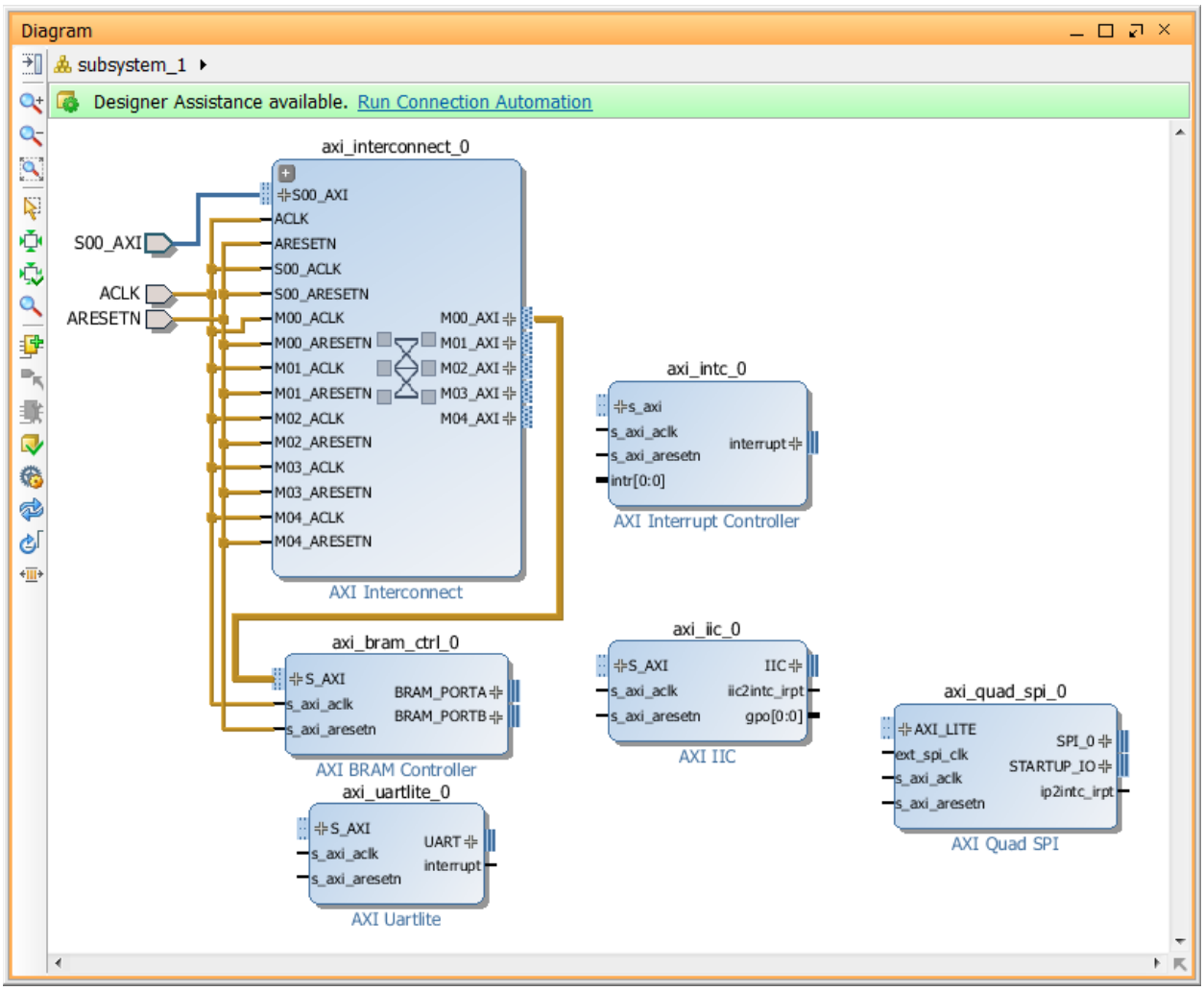

Figure 21: Connect the AXI BRAM Controller to the AXI Interconnect

6. Connect the **s_axi_aclk and** the **s_axi_aresetn** pins of the AXI BRAM Controller to the **ACLK and ARESETN** ports.
7. Using the same steps, connect the remaining **slave IP** cores in the design to the **AXI Interconnect**.

TIP: *The order of connections between the S_AXI interface pins on the IP slaves and the M_AXI interface pins on the AXI Interconnect does not matter.*

---

8. Click the **Regenerate Layout** button on the sidebar menu.

   The IP integrator design canvas should look similar to what is shown in Figure 22.

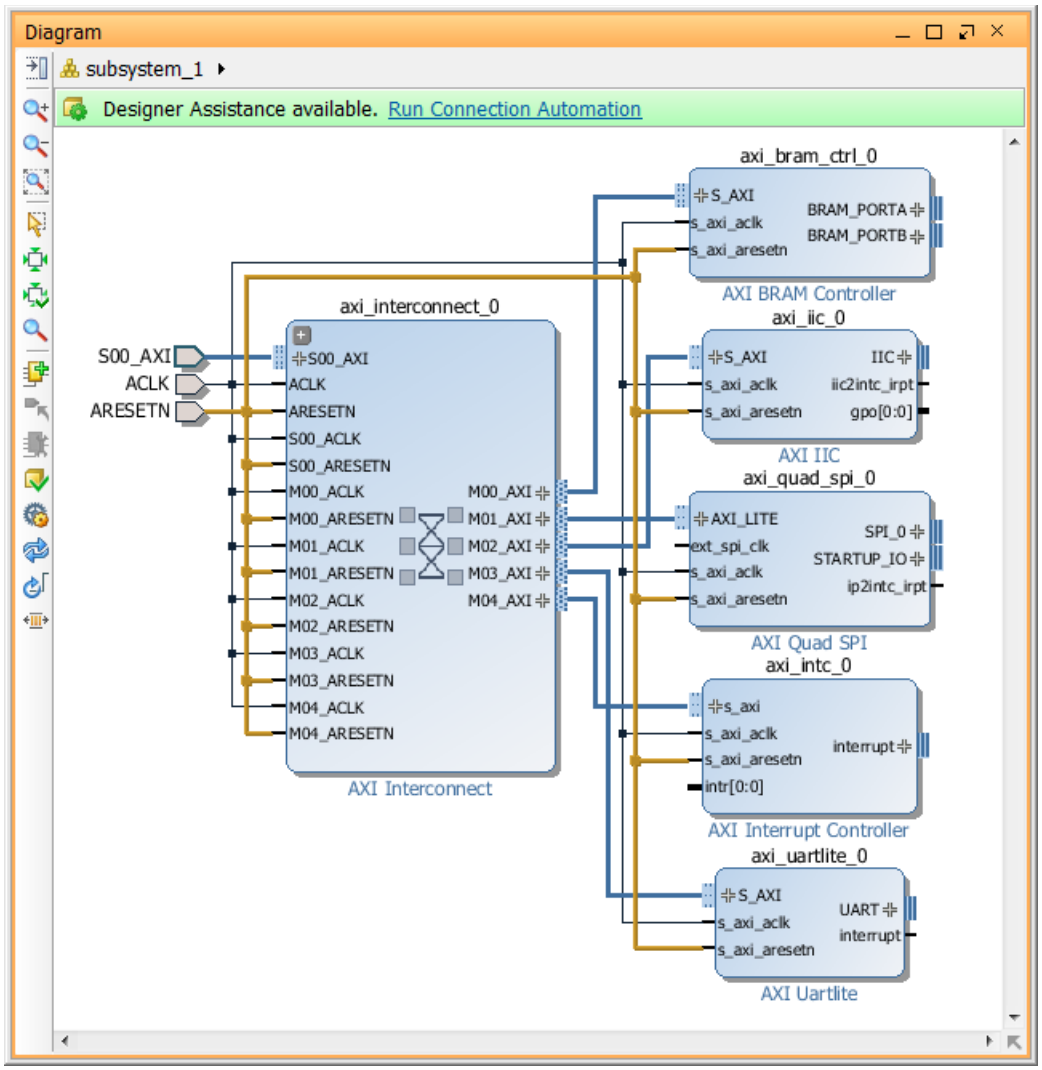

Figure 22: Connecting to the AXI Interconnect

At this point, you should save the IP integrator subsystem design.

9. Click the **File > Save Block Design** command from the main menu.

Step 5: Running Connection Automation

At this point, there are still some output interface pins that need to be connected external to the subsystem design, such as the UART interface of the AXI Uartlite, the SPI_0 interface of the AXI Quad SPI, and the IIC interface of the AXI IIC. Also, note that the AXI BRAM Controller is not connected to a Block Memory Generator.

IP integrator offers the Designer Assistance feature to automate certain kinds of connections. For the current subsystem design, you can connect the UART, SPI and IIC interfaces to external ports using connection automation. You can also use the Designer Assistance feature to connect a Block Memory Generator to the BRAM Controller.

1. Click on **Run Connection Automation** link, in the banner at the top of the design canvas.

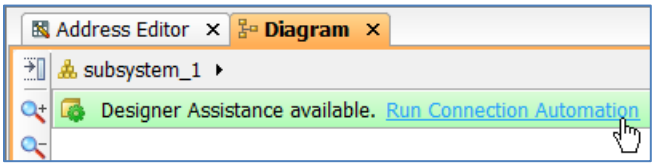

Figure 23: Run Connection Automation for /axi_quad_spi_0/SPI_0

   The Run Connection Automation dialog box opens.

2. Select **All Automation (0 out of 5 selected)**. This selects all external interfaces and the BRAM Controller for auto connection.

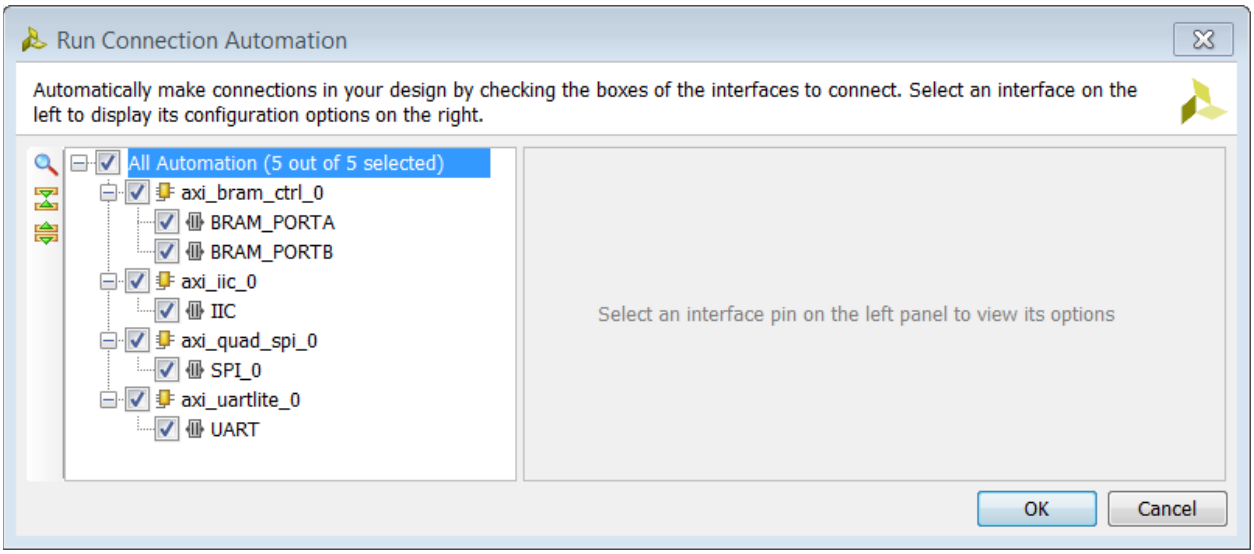

Figure 24: Run Connection Automation dialog box

3. Select and highlight the interfaces to see a description of the automation being offered as well as any options that may be needed to connect these interfaces.

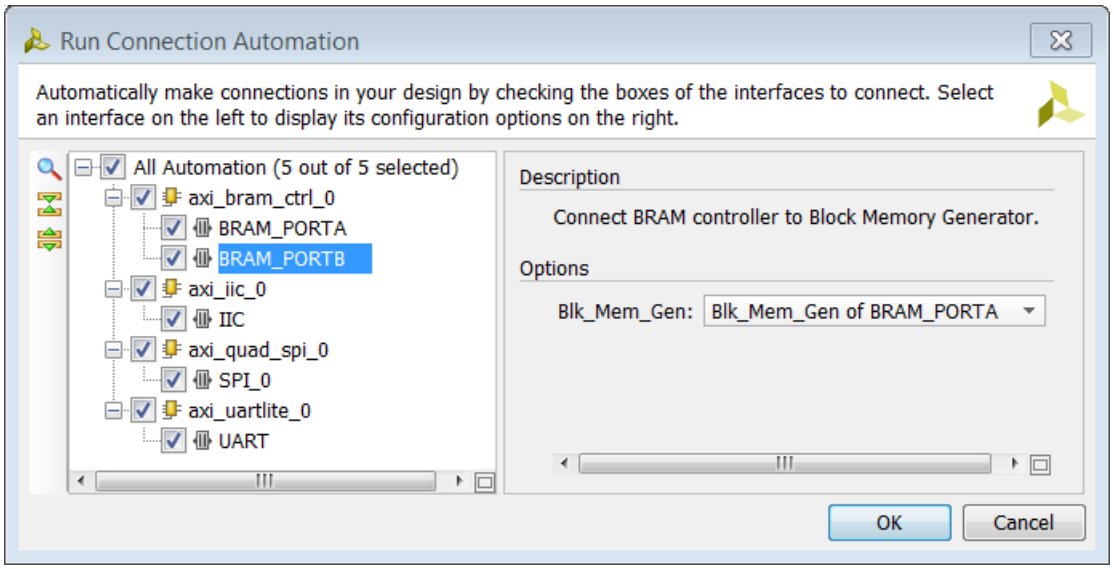

Figure 25: The Run Connection Automation dialog box automation options

4. Click **OK**.
5. All the external interfaces are connected to I/O ports as well as the BRAM Controller is connected to the Block Memory Generator.

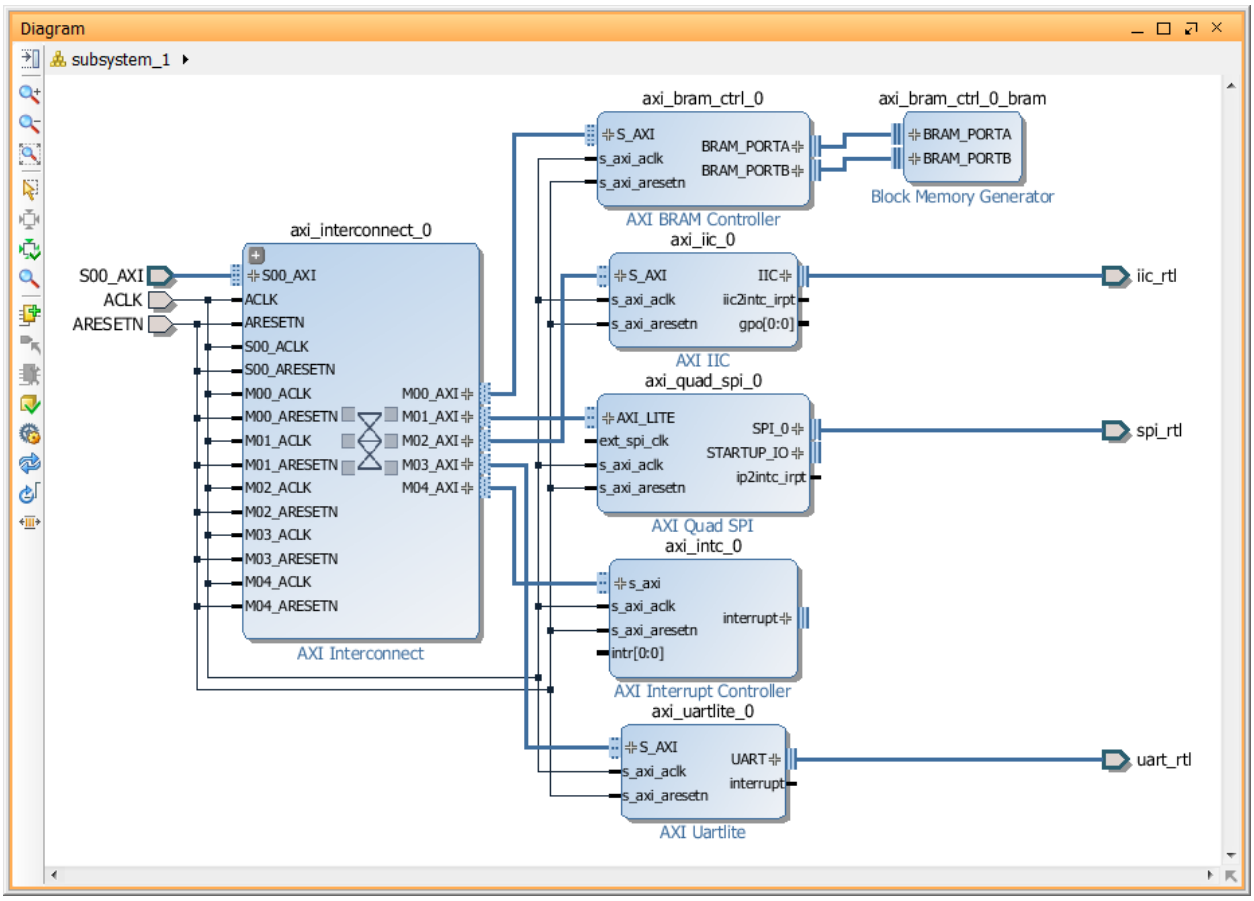

Figure 26: spi_rtl port connection after running connection automation

6. Right-click on the newly added **`spi_rtl`** port to open the pop-up menu, and select the **External Interface Properties** command.

   In the External Interface Properties window, you can change the Name of the port if needed. The Vivado IP integrator feature automatically assigns the name of the port when connection automation is run. For now, leave the `spi_rtl` port named as it is.

7. Right-click on the **ext_spi_clk** pin of the AXI Quad SPI, and select **Create Port**.

   The Create Port dialog box opens as shown in Figure 26.

8. Set the **frequency** to 100 MHz, if it is not already set, and click **OK.**

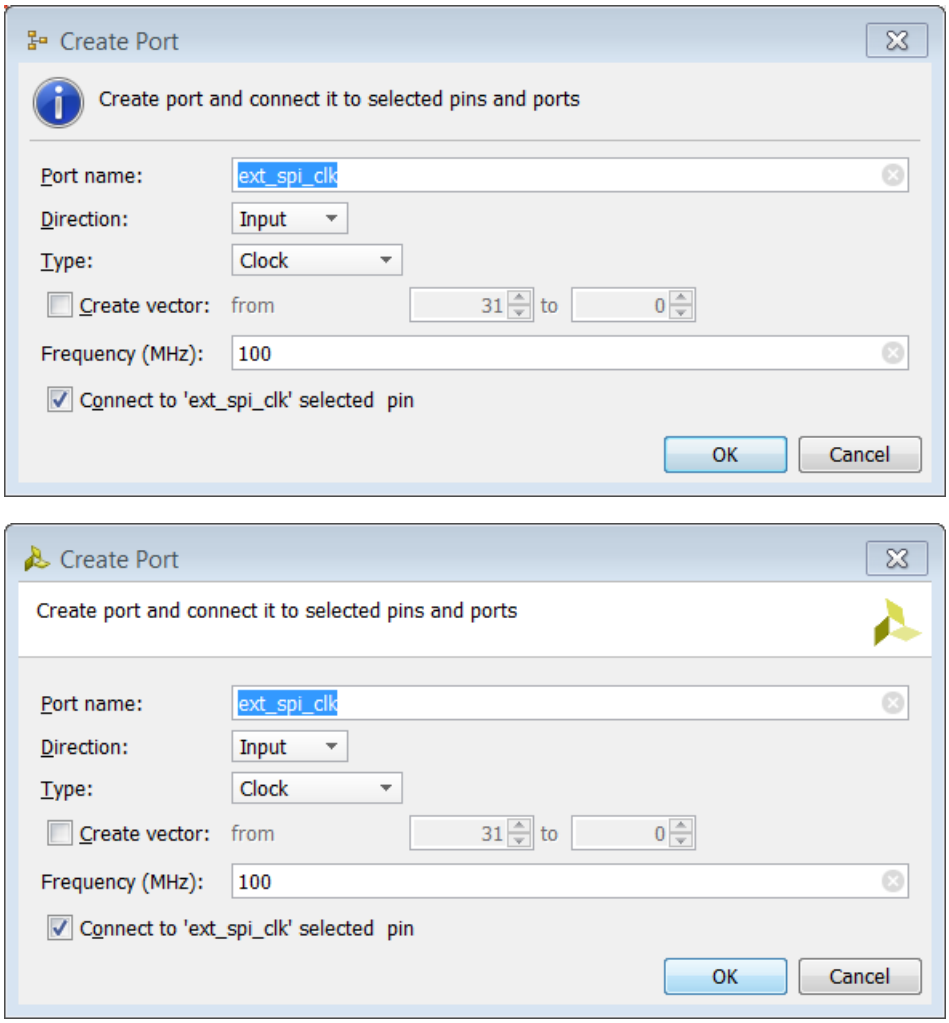

Figure 27: Create ext_spi_clk Port

9. Click the **Regenerate Layout** button to redraw the subsystem design.

   The optimized layout of the design should now look similar to what is shown in Figure 32.

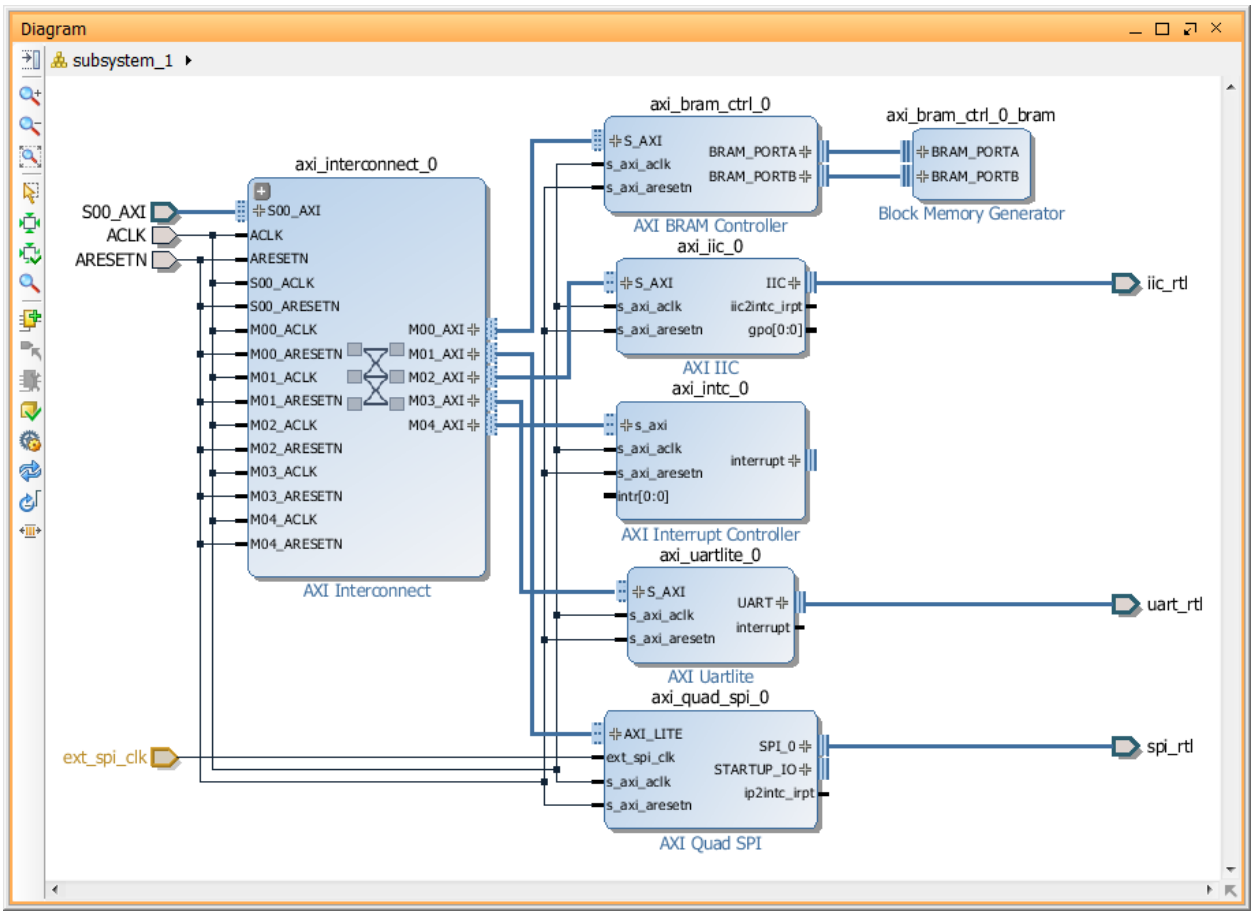

Figure 28: Completed IP Subsystem Design

Step 6: Managing Signals with CONCAT and CONSTANT blocks

Now you will connect the interrupt signals on the various IP slaves to an interrupt controller through a Xilinx Concat block, to concatenate the individual interrupt signals into a bus. The Concat block is a general-purpose block to combine multiple inputs into a single bus output. The individual interrupt signals from different AXI slave cores need to be combined into a bus because the Interrupt Controller takes a bus at its input.

1. Right-click on the design canvas to open the popup menu and select **Add IP**.
2. In the search field, type **concat** to find the Xilinx Concat block, and double-click on the core.

   The Xilinx Concat core is instantiated onto the IP integrator design canvas.

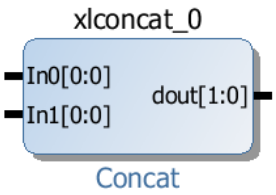

Figure 29: Xilinx Concat Block

3. Right-click on the **Concat** block to open the popup menu, and select **Customize Block**.

   The Re-customize IP dialog box opens as shown in Figure 34.

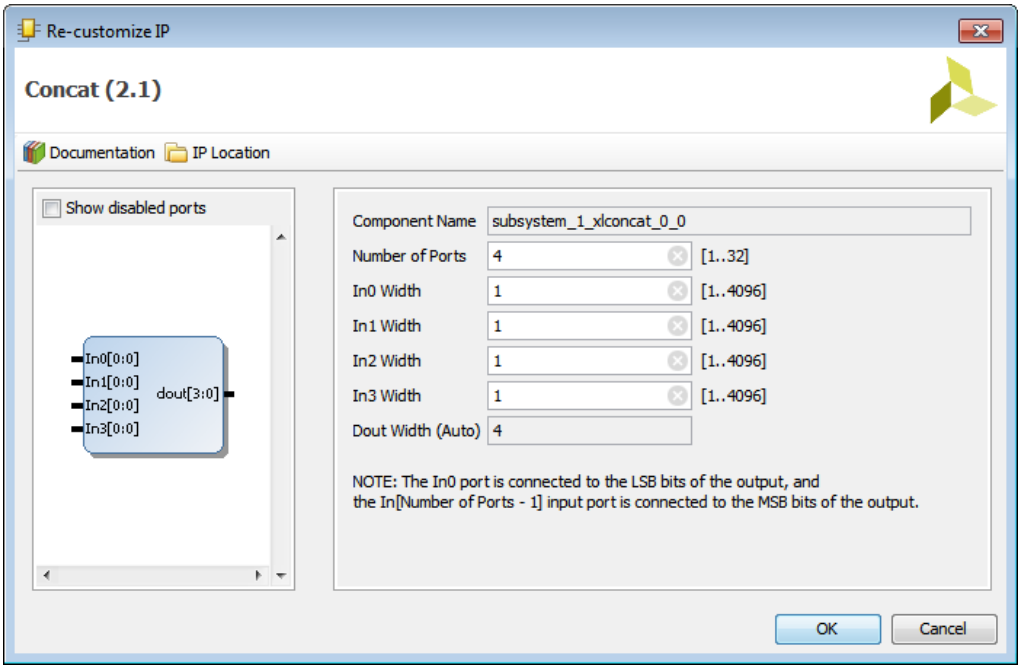

Figure 30: Customize Concat Block

4. Change the **Number of Ports** to **4**, and click the mouse in the **In0 Width** field to accept the change.

   The dialog box is updated to reflect the new number of input ports. Three ports are needed to connect the interrupt pins on the various slave IP blocks into the Interrupt Controller. The fourth port is used to demonstrate tying a signal high or low using the Constant block.

5. Click **OK**.

   Now connect the interrupt signals of the AXI slaves to the Concat block to create an interrupt bus.

TIP: *To pull signals out of a bus, use the Slice block instead of the Concat block.*

6. Place the cursor on top of the **interrupt** pin of the AXI UART-Lite.

   Notice that the cursor changes into a pencil indicating that a connection can be made from that pin.

7. Click and drag the cursor from the interrupt pin to an input pin on the Concat block, as shown in Figure 35.

   As you drag the connection wire, a green checkmark appears on the Input port indicating that a valid connection can be made between these points.

8. Release the mouse button and Vivado IP integrator makes a connection between the pins.

9. Connect the **interrupt pins** on the AXI IIC block and the AXI Quad SPI block to input pins on the Concat block, using the same process.

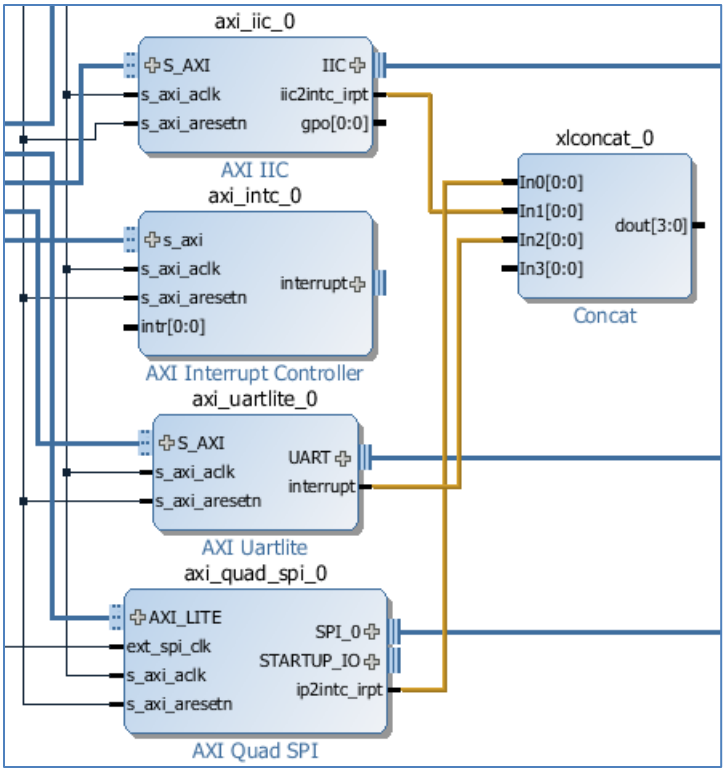

Figure 31: Connect the Interrupt signals to Concat Block

At this point, you have connected the interrupt signals to the Concat block, and there is a remaining unconnected input pin. You could re-customize the block to include only the required number of inputs. For this tutorial though, you will use the Constant block to tie the extra input down instead.

10. On the design canvas, right-click to open the popup menu and select **Add IP**.

    The IP Catalog opens.

11. In the search field, type **cons** to find the Xilinx Constant block, and double-click on the core in the IP Catalog.

    The Xilinx Constant block is instantiated into the subsystem design.

12. Relocate the **block** as needed to be close to the Concat block.

13. Click and drag the cursor from the output pin on the Constant block, and connect it to the dangling fourth input pin on the Concat block.

    As you drag the connection wire, a green checkmark appears on the input pin indicating that a valid connection can be made between these points.

14. Release the mouse button and Vivado IP integrator makes a connection between the pins.

    Double-clicking on any of the interrupt pins on the various AXI slaves shows that by default they are active high, or triggered on the rising edge. In this case, you must use the Constant block to tie down the fourth input on the Concat block to prevent needless interrupt.

15. Double-click the **Constant** block to re-customize the IP.

    The Re-customize IP dialog box opens.

16. Set the **Const Val** field to **0**, as shown below, and click **OK**.

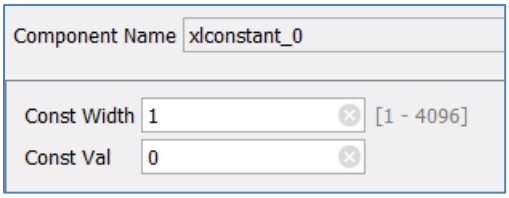

Figure 32: Set Constant Value

The fourth input of the Concat block is no longer floating. Now you can connect the concatenated interrupt signals from the Concat block to the Interrupt Controller.

17. Click and drag the cursor from the output pin on the Concat block, dout[3:0], and connect it to the intr[0:0] pin of the Interrupt Controller block.

18. Click the **Regenerate Layout** button to redraw the subsystem design.

    The optimized layout of the design should now look similar to what is shown in Figure 37.

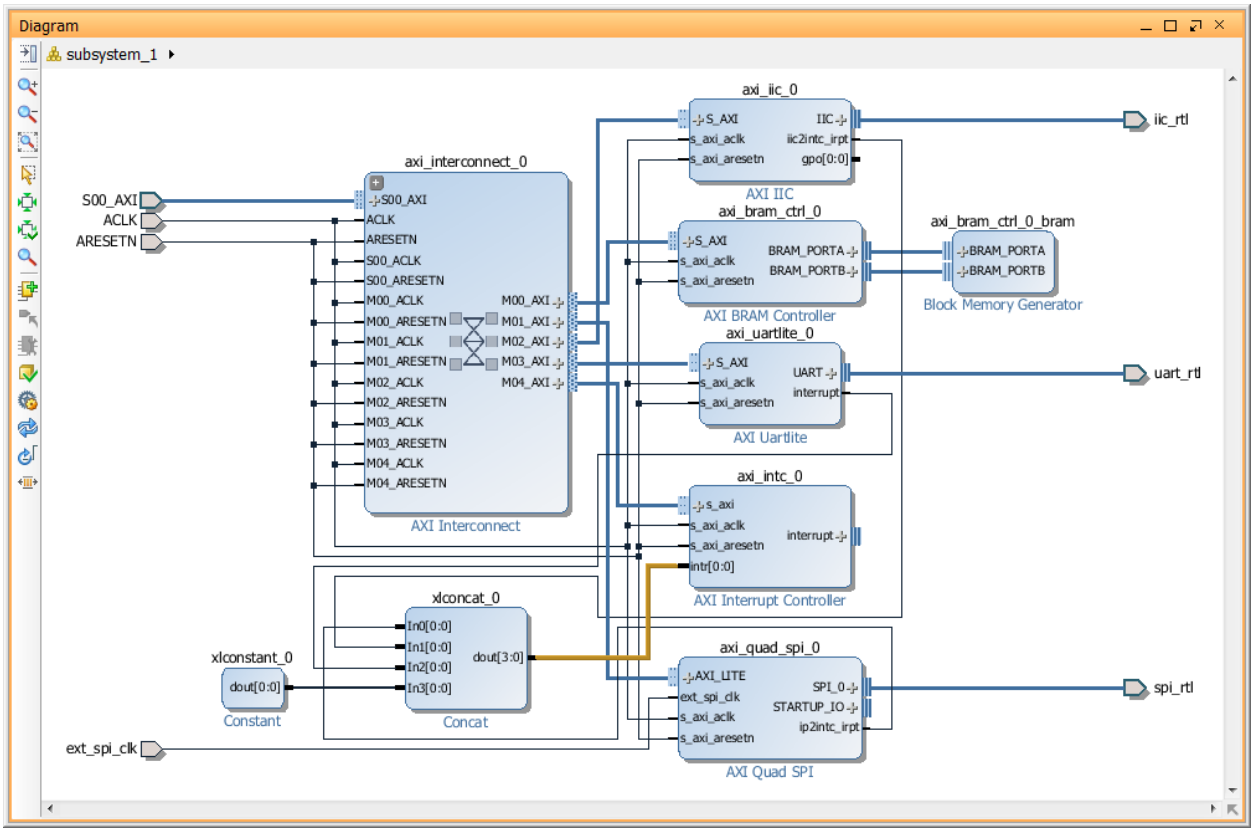

Figure 33: Connect Concat block to Interrupt Controller

Notice that the 1-bit bus width of the interrupt signal on the Interrupt Controller block does not match the 4-bit signal width from the Concat block. This is automatically corrected during design validation.

TIP: *When you instantiate an Interrupt Controller block, the interrupt port by default is 1-bit. During design validation, parameter propagation passes the width of the output signal from the Concat block to the input signal of the Interrupt Controller, and the port width on the interrupt controller changes automatically.*

19. Right-click the **interrupt pin** of the Interrupt Controller, to open the popup menu, and select **Make External**.

    This connects the interrupt output pin to an output port that is taken outside the IP subsystem design, to a processor for example.

    All the interrupts on the Interrupt Controller have a priority that is determined based on the order of connection to the Concat block. Bit-0 of the interrupt bus has the highest priority. When the interrupt port goes high, or active, the processor determines which slave is causing the interrupt. Multiple interrupts are handled according to their priority.

20. Click the **File > Save Block Design** command from the main menu.

Step 7: Using the Address Editor

For various memory mapped master and slave interfaces, IP integrator follows the industry standard IP-XACT data format for capturing memory requirements and capabilities of endpoint masters and slaves. This section provides an overview of how IP integrator models address information on a memory-mapped slave.

Master interfaces have address spaces, or address_space objects. Slave interfaces have an address_space container, called a memory map, to map the slave to the address space of the associated master. These memory maps are usually named after the slave interface pins, for example S_AXI, though that is not required.

The memory map for each slave interface pin contains address segments, or address_segment objects. These address segments correspond to the address decode window for that slave. A typical AXI4-Lite slave will have only one address segment, representing a range of addresses. However, some slaves, like a bridge, will have multiple address segments; or a range of addresses for each address decode window.

When a slave is mapped to the master address space, a master address_segment object is created, mapping the address segments of the slave to the master. The Vivado IP integrator can automatically assign addresses for all slaves in the design. However, you can also manually assign the addresses using the Address Editor. In the Address Editor, you see the address segments of the slaves, and can map them to address spaces in the masters.

TIP: *The Address Editor tab only appears if the subsystem design contains an IP block that functions as a bus master. In the tutorial design, the external processor connecting through the AXI Interconnect is the bus master.*

1. Click the **Address Editor** tab to show the memory map of all the slaves in the design.

   Note: If the Address Editor tab is not visible then open it with the **Window > Address Editor** command from the main menu.

   You can expand the Unmapped Slaves folder by clicking on the '+' sign if necessary.

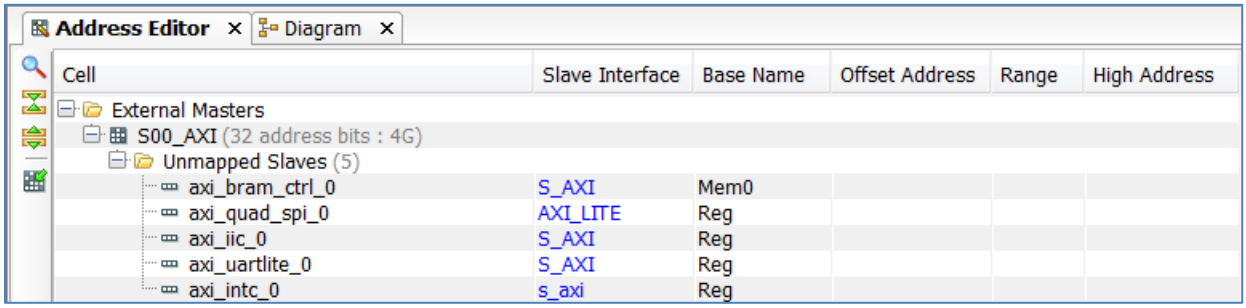

Figure 34: Address Editor

2. Right-click anywhere in the Address Editor and select **Auto Assign Address**.

   This command maps slave address segments to master address spaces, thereby creating address segments in the master. You can change these automatic addresses later by clicking in the corresponding column and changing the values.

   Alternatively, you can also click on the Auto Assign Address icon, on the block design canvas toolbar to automatically assign the addresses.

3. The Auto Assign Address dialog box pops-up. Click **OK**.

   The Address Editor should now look similar to Figure 39.

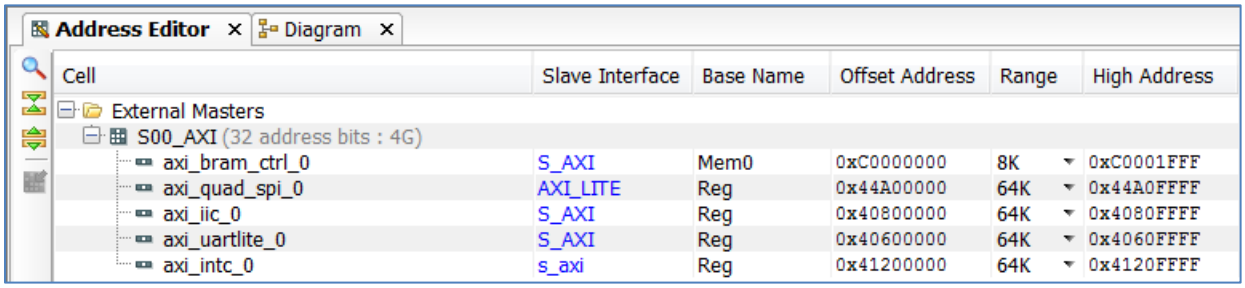

Figure 35: Automatically Assigned Addresses

4. Change the size of the address segments for the **AXI BRAM Controller** core by clicking in the **Range** column and selecting **64K** from the pull-down menu.

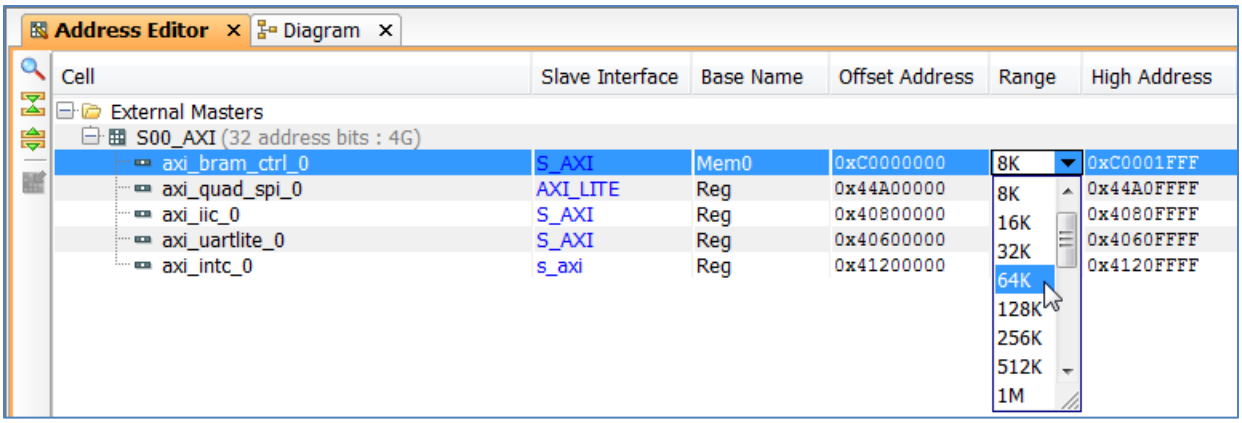

Figure 36: Change Range field of the AXI BRAM Controller

5. Select the **Diagram tab**, to return to the IP integrator design canvas.

Step 8: Validating the Design

1. From the sidebar menu of the design canvas, run the IP subsystem design rule checks by clicking the **Validate Design** button.

   The design is checked, and a number of Critical Warnings are returned as seen in Figure 40, indicating problems with the design. In this case, the problems are related to the asynchronous reset in the subsystem design.

IMPORTANT: *IP integrator treats an external reset coming into the block design as asynchronous to the clocks. You should always synchronize the external resets with a clock domain in the IP subsystem to help the design meet timing.*

The Critical Warning message includes a suggestion to "`add proc_sys_reset module to create a reset, that is synchronous to the associated clock source /ACLK`."

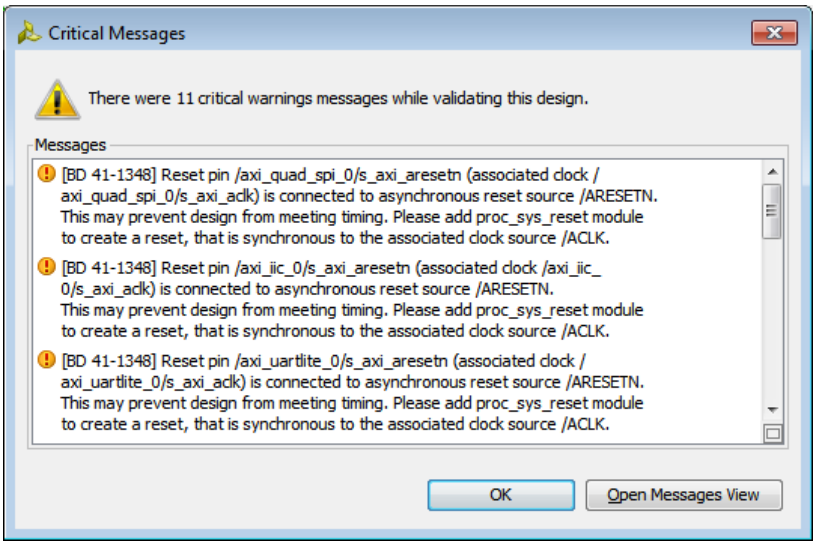

Figure 37: Critical Warnings – Asynchronous Reset

You can use a Processor System Reset block (`proc_sys_reset`) to synchronize the reset. The Processing System Reset is a soft IP that handles numerous reset conditions at its input and generates appropriate system reset signals at its output. However, if a clock and a reset are external inputs to the block design, and the reset signal is externally synchronized, then you simply need to associate the related clock with the reset. This does not require the Processor System Reset block.

2. Double click the **ACLK port** to open the Customize Port dialog box.

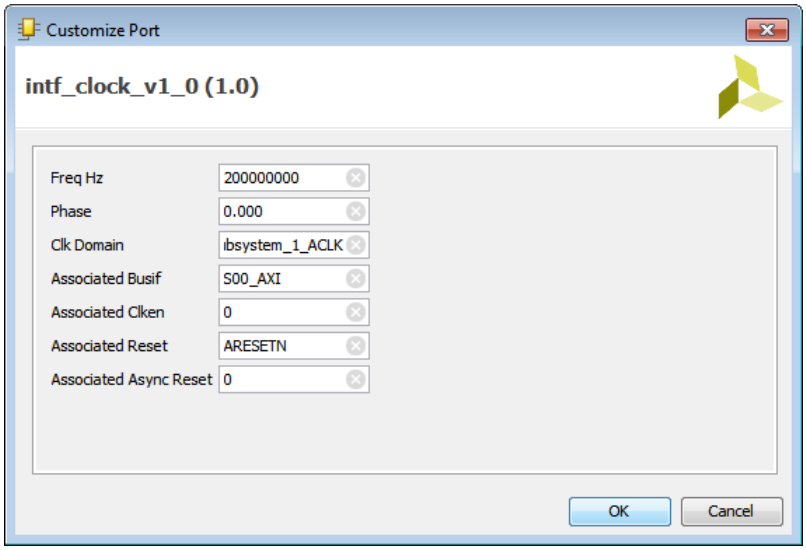

Figure 38: Customize Port – Associated Reset

3. Set or Change the **Associated Reset** field to **ARESETN**.
4. Click **OK.**

5. Select **Tools > Validate Design** to rerun the validation.

   The Validate Design dialog box opens, and validation should be successful.

6. Click **OK**.

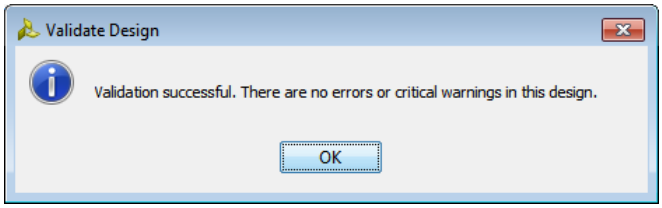

Figure 39: Validate Design dialog box

7. Examine the interrupt bus width on the **Interrupt Controller** block.

   Notice that the width now matches the width of the bus signal coming from the Concat block. Parameter propagation has allowed the bus width to propagate through the subsystem design as needed.

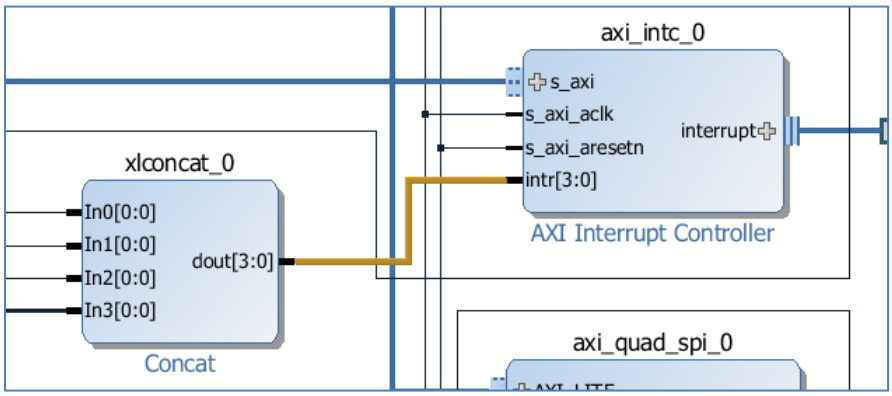

Figure 40: Parameter Propagation Matches Bus Width

At this point, you should save the IP integrator subsystem design again.

8. Use the **File > Save Block Design** command from the main menu to save the design.

Step 9: Creating and Implementing the Top-Level Design

With the IP subsystem design completed and validated, you need to prepare it for inclusion into the top-level HDL design. The subsystem can be included as a module or block in the top-level design, or may be the only block in the top-level design. In either case, you need to generate the HDL files for the subsystem design.

1. In the Sources window, right-click the top-level subsystem design, **subsystem_1**, and select **Generate Output Products**.

   This command generates the source files for the IP cores used in the subsystem design, and any related constraints file.

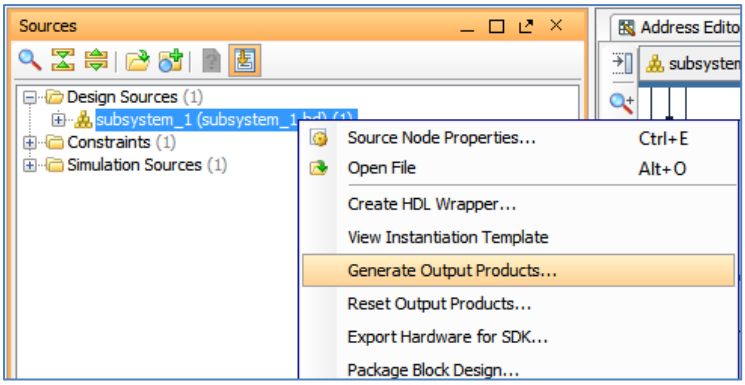

Figure 41: Generate Output Products

   The Generate Output Product dialog box opens, as shown in Figure 45, to regenerate the various output products associated with the subsystem design.

2. Click **Generate** to generate all output products.

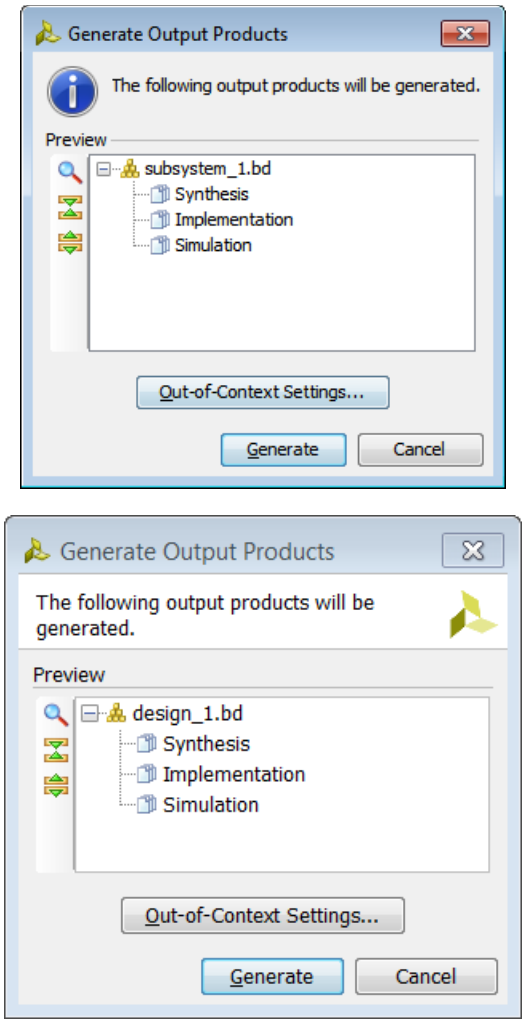

Figure 42: Generate Output Products

Alternatively, you can click on **Generate Block Design** in the Flow Navigator, under the IP integrator drop-down menu.

3. The Generate Output Products dialog box pops up after the output products are generated. Click **OK**.
4. In the Sources window, right-click the top-level subsystem design, **subsystem_1**, and select **Create HDL Wrapper**.

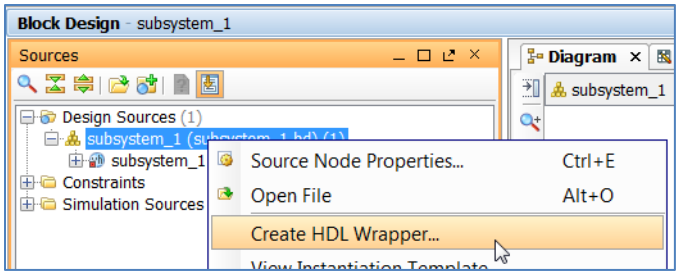

Figure 43: Create top-level HDL Wrapper

The Create HDL Wrapper dialog box opens and offers two choices.

- **Copy generated wrapper to allow user edits**: Choose this option if you will modify the wrapper file. Often times a block design is a subset of an overall project. In cases like these, you may need to edit the wrapper file and instantiate other design components in the wrapper. If the I/O interface of the block design changes in any manner, you must manually update the wrapper file to reflect those changes. The wrapper file created using this method is written to the `<project_name>.srcs/sources_1/imports/hdl` directory.
- **Let Vivado manage wrapper and auto-update**: Choose this option if you want the Vivado IDE to generate and update the wrapper file as needed. The wrapper file created using this method is automatically updated every time output products for the block design are generated, to reflect the latest changes. The wrapper file is written to the `<project_name>.srcs/sources_1/bd/<bd_name>hdl` directory.

5. Choose the **default** option to let the Vivado IDE create and manage the wrapper file.

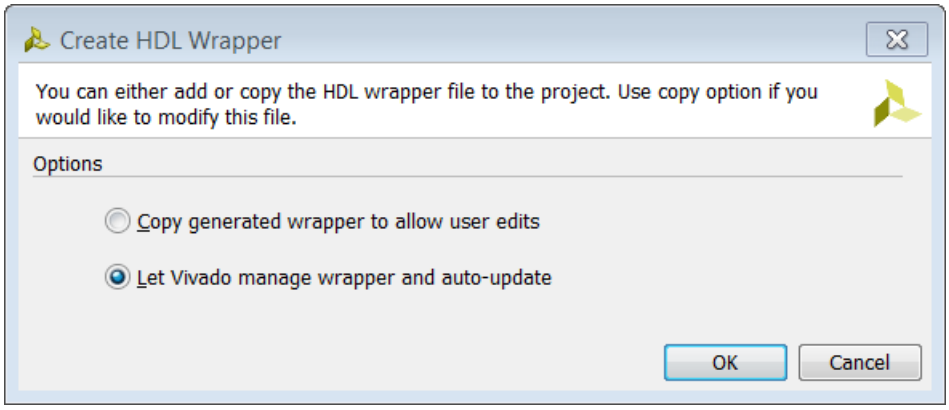

Figure 44: Create HDL Wrapper dialog box

6. Click **OK**.

The Vivado IDE creates a top-level HDL wrapper for the `subsystem_1` block design and adds it to the design sources.

With the top-level HDL source added to the project, you must now add design constraints to the project prior to implementation.

7. From the **Flow Navigator**, under Project Manager, click **Add Sources**.
8. Select the **Add or Create Constraints** option and click **Next**.

   The Add Sources dialog box opens as seen in Figure 48.
9. In the Add or Create Constraints dialog box, click **Add Files**.

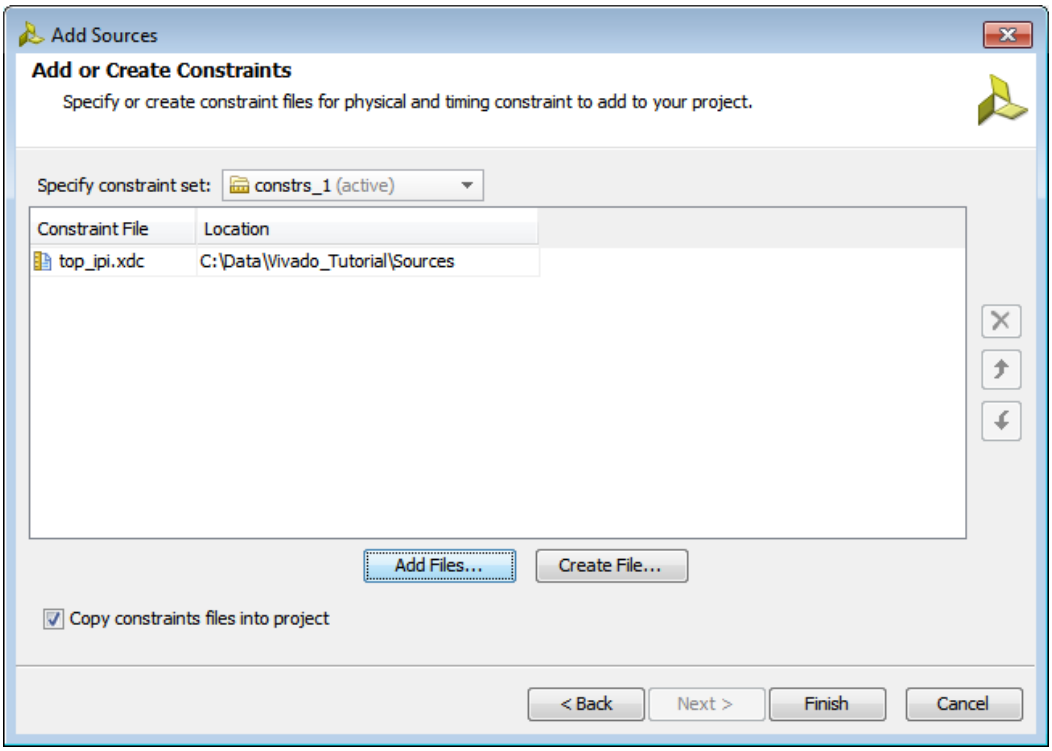

Figure 45: Add IPI Constraints

The Add Constraints Files dialog box opens.

10. Select the **`top_ipi.xdc`** file in *<Extract_Dir>/Vivado_Tutorial/Sources*, and click **OK**.
11. In the Add or Create Constraints dialog box, make sure that **Copy constraints files into project** is checked.
12. Click **Finish** to add the constraints to the project.

    You are now ready to synthesize, implement, and generate the bitstream for the top-level design.
13. In the **Flow Navigator**, click **Generate Bitstream**.

With a single click, this will complete all of the steps needed to synthesize, implement and generate the bitstream for the design.

The No Implementation Results Available dialog box opens as seen in Figure 49.

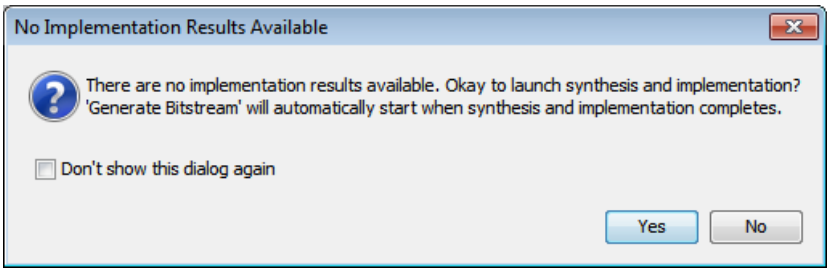

Figure 46: No Implementation Results Available

14. Click **Yes**.

    Synthesis will run for a few minutes as can be seen in the top-right corner of Vivado IDE, then place and route, and finally bitstream generation. You may see some warnings that pop-up while the design is going through the flow. You can close and ignore these warnings.

    After the Vivado Design Suite generates the bitstream, the Bitstream Generation Completed dialog box opens.

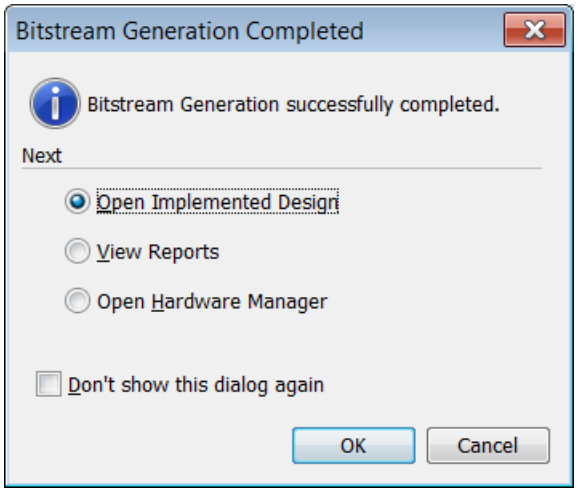

Figure 47: Bitstream Generation Completed

15. Click **OK**.
16. In the Implemented Design section of the **Flow Navigator**, click on **Report Timing Summary**.

    The Report Timing Summary dialog box opens.

17. Click **OK** to accept the defaults, and run timing analysis.

After timing analysis completes, the Timing Summary window opens as seen in Figure 51.

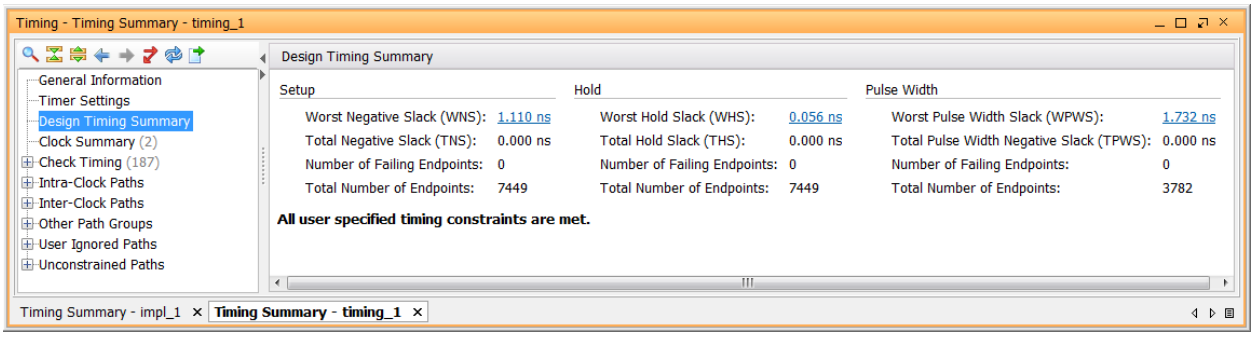

Figure 48: Timing Summary Report

Conclusion

This tutorial walked you through creating a simple IP integrator subsystem design by instantiating common peripherals and local memory cores, and connecting them through an AXI Interconnect. You then took the completed subsystem design into a top-level HDL design for implementation in the Vivado Design Suite. This tutorial gave you hands-on experience and a basic understanding of many of the features of the Vivado IP integrator tool.