Vivado Design Suite Tutorial

Designing IP Subsystems Using IP Integrator

UG995 (v2017.3) October 13, 2017

XILINX
ALL PROGRAMMABLE™

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
10/13/2017	2017.3	Changes are: Updated and validated for the 2017.3 release.
04/05/2017	2017.1	Updated and validated for 2017.1 release.

Table of Contents

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 view the Designing with Vivado IP Integrator quick take video to learn more about this feature of the Vivado Design Suite.*

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®-7 device**.** The tutorial uses a small design with minimal hardware requirements and to enable timely completion of the tutorial, as well as to minimize the data size.

TIP: Although the tutorial design targets an xc7k325 Kintex-7 device, you can choose another part, such as the xc7a35 Artix®-7 device for use with the WebPack version of the Vivado Design Suite. The tutorial results should be similar.

Hardware and Software Requirements

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 for the Vivado Design Suite.

Locating Tutorial Design Files

You can find the design file for this tutorial on the Xilinx website: Reference Design File.

Extract the file named `top_ipi.xdc` to a directory. That directory is called the `<Extract_Dir>` in this tutorial.

Step 1: Creating a Project

1. Open the Vivado IDE:
2. On Linux, change to the directory where the Vivado tutorial design file is stored:

    cd <Extract_Dir>/Vivado_Tutorial

3. Then launch the Vivado Design Suite: **vivado**

    On Windows, launch the Vivado Design Suite:

    Start > All Programs > Xilinx Design Tools > Vivado 2017.x > Vivado 2017.x[1]

    As an alternative, click the **Vivado 2017.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 the following figure:

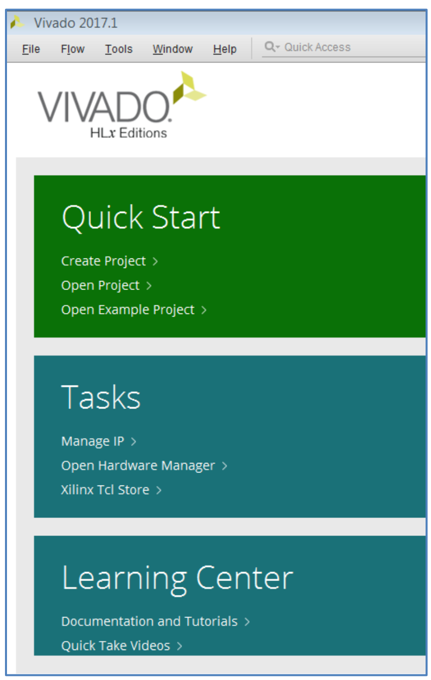


Figure 1: Vivado IDE Getting Started Page

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

4. In the Quick Start page, select **Create Project**.
5. The New Project wizard opens. Click **Next** to confirm the project creation.
6. In the Project Name page, shown in the following figure, set the following options:
   - Type the project name: `project_ipi`
   - Enter the project location: `<project_directory>`

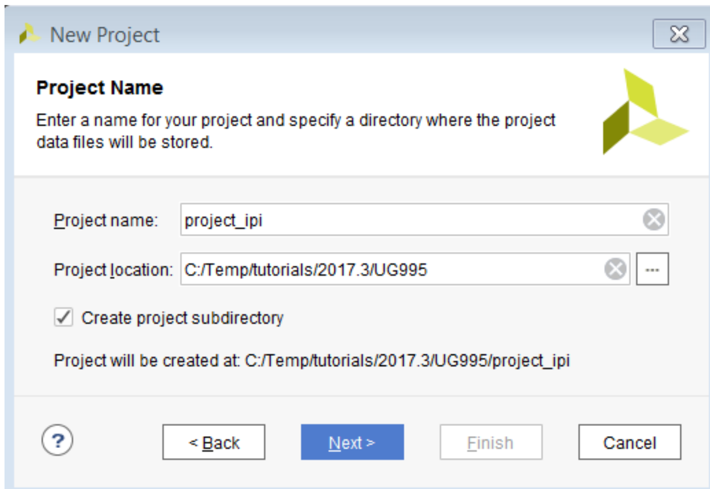


Figure 2: Create Project

7. Ensure that **Create project subdirectory** is checked, and click **Next**.
8. In the Project Type dialog box, select **RTL Project**, and click **Next**, as shown in the following figure:

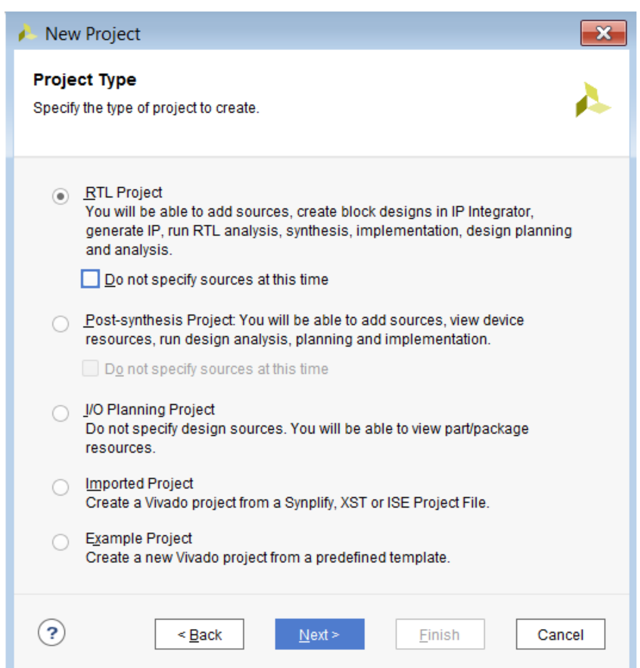


Figure 3: Specify Project Type

9. In the Add Sources dialog box:
   - Set the Target language to **VHDL** or **Verilog**
   - 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.

10. In the Add Constraints dialog box, click **Next**.

11. In the **Default Part** dialog box, shown in the following figure, make the this entries:
    - Select **Parts**
    - Set **Family:** Kintex-7
    - Set **Speed grade:** -2

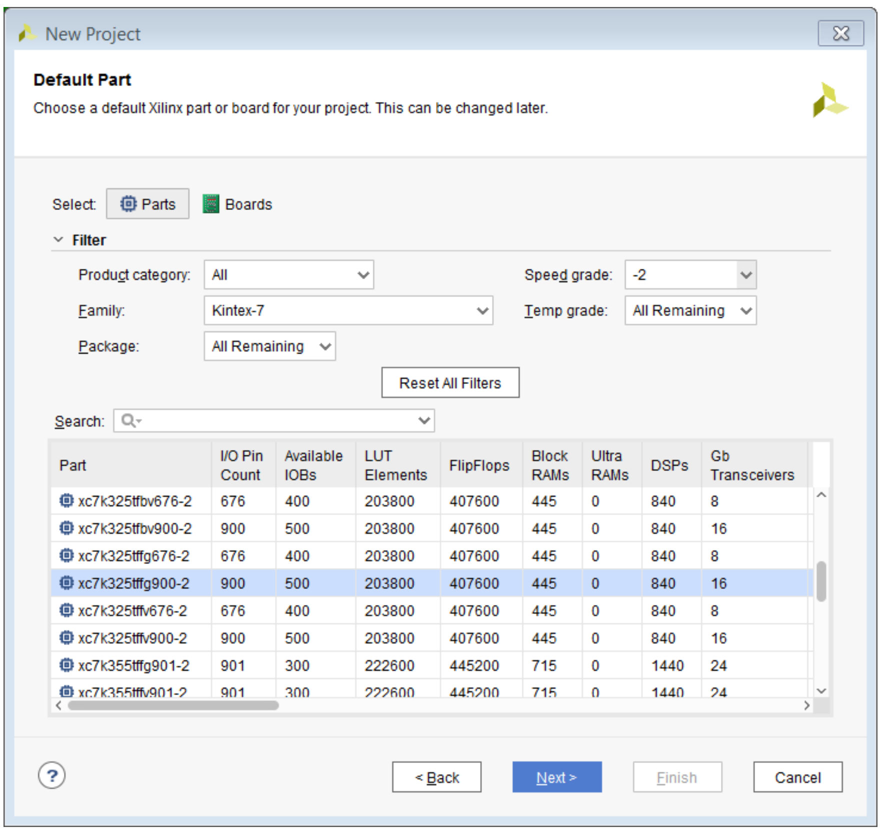


Figure 4: Select Default Part

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: Creating an IP Integrator Design

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

   The Create Block Design dialog box opens, as shown in the following figure:

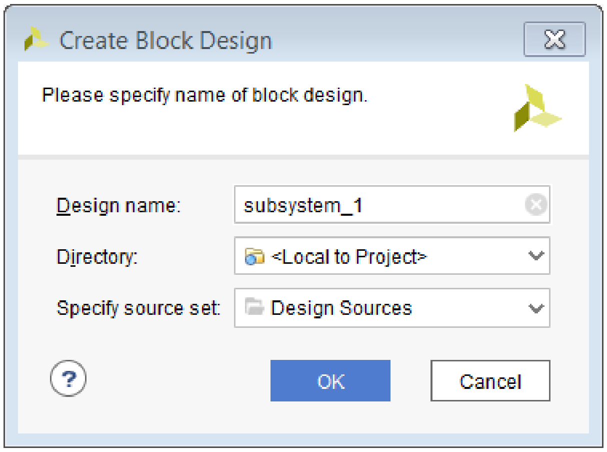


Figure 5: Create Block Design Dialog Box

2. In the Create Block Design dialog box, set the following:

   - Design Name: **subsystem_1**
   - Set Directory: **Local to Project**
   - Specify source set: **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. Click the **Add IP** button in the block design canvas + .

   Alternatively, you can also right-click on the design canvas to open the context menu and select **Add IP**.

TIP: *To open the IP Details window beside the IP catalog, as shown in Figure 6, type 'Ctrl-Q' as described at the bottom of the IP Catalog window. This window lets you see details of the currently selected IP in the catalog.*

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

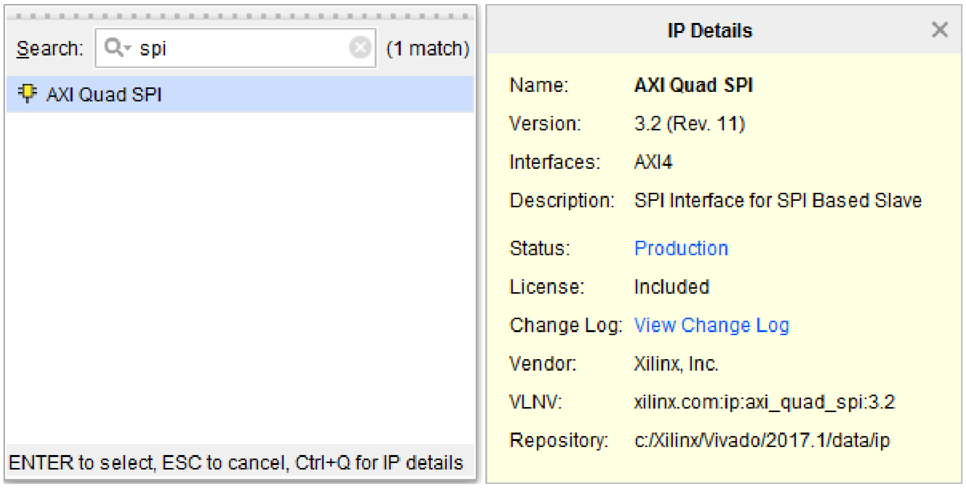
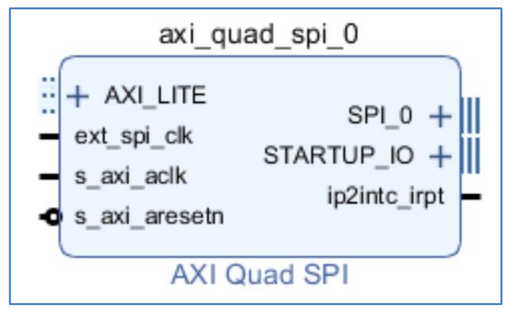


Figure 6: Instantiate AXI Quad SPI IP

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.



6. Right-click 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 the following figure. The relative positions of the blocks placed on the canvas might be slightly different.

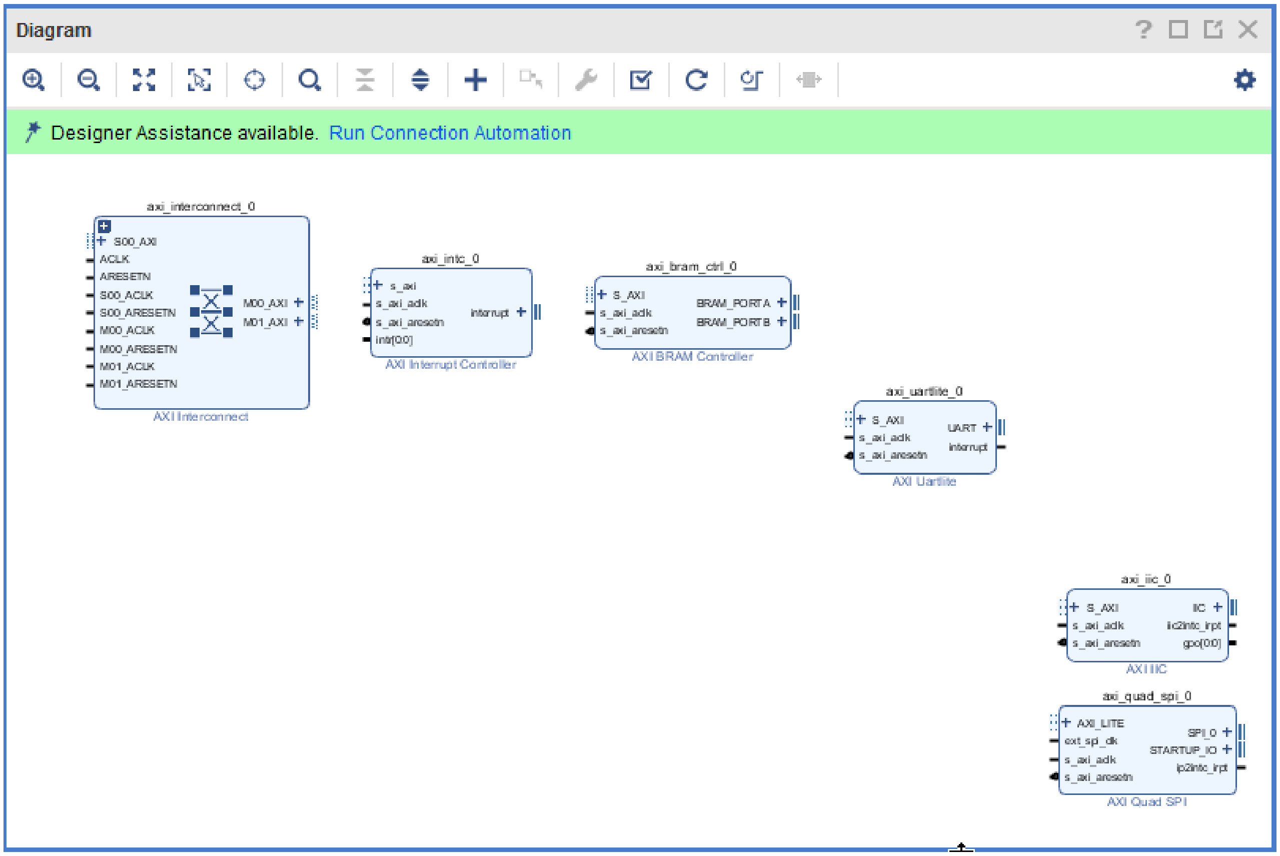


Figure 7: IP Integrator Design Canvas

Step 3: Creating External Connections

At this point, you have instantiated several AXI slaves that you can access through 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 the **S00_AXI** interface pin on the AXI Interconnect to open the popup menu, and select **Create Interface Port**.

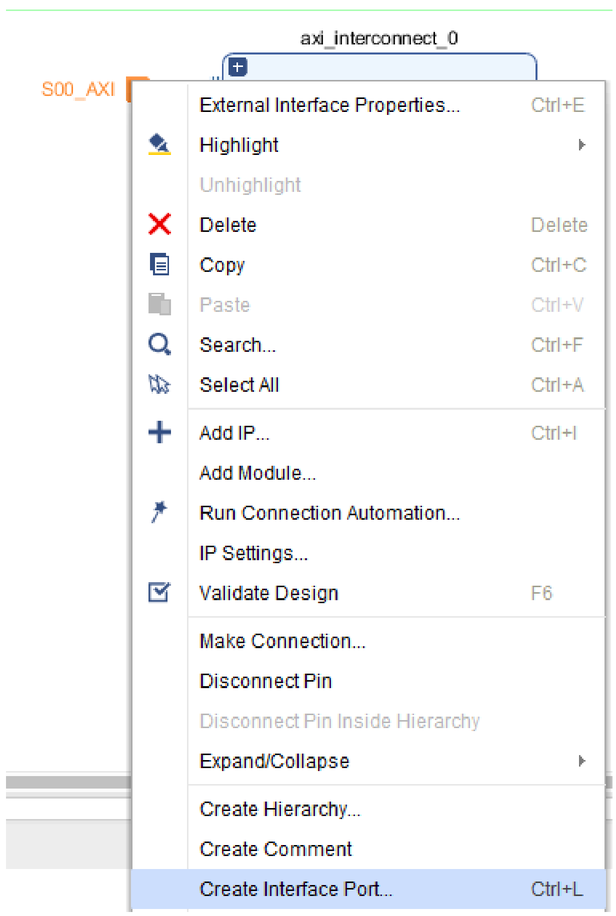


Figure 8: Right-Click S00_AXI Interface Pin

The Create Interface Port dialog box opens, as shown in the following figure.

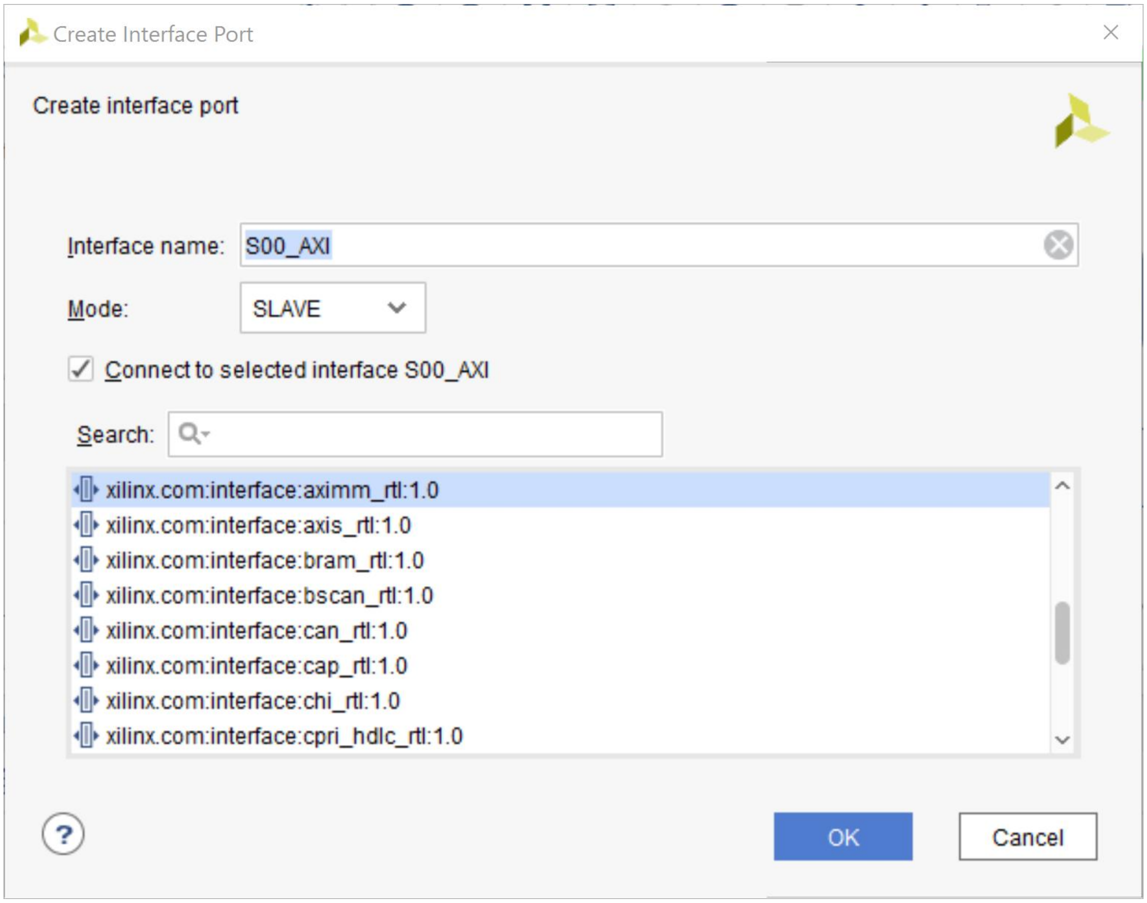


Figure 9: Create Interface Port

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

   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. Because these are not interface pins, you will not need an interface port to connect them.

3. Right-click the `ACLK` pin of the AXI Interconnect, and select **Create Port**, as shown in the following figure:

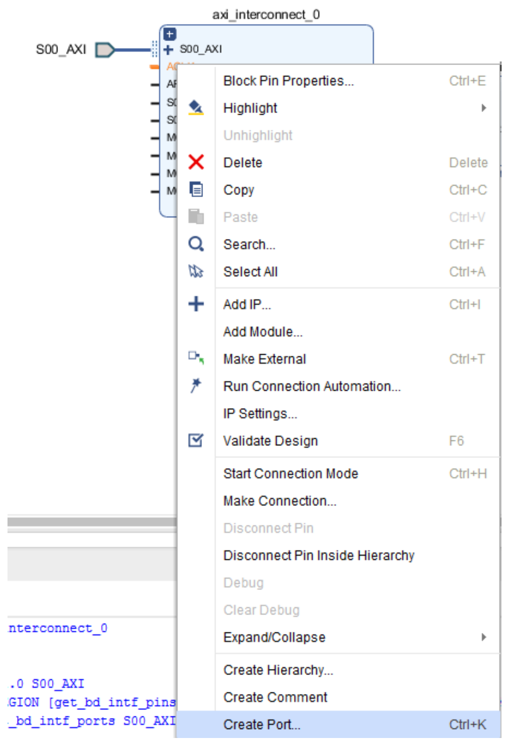


Figure 10: Create Port

4. In the Create Port dialog box, shown in Figure 11, specify the **Frequency** as **200 MHz**, leaving everything else at its default value.
5. Click **OK**.

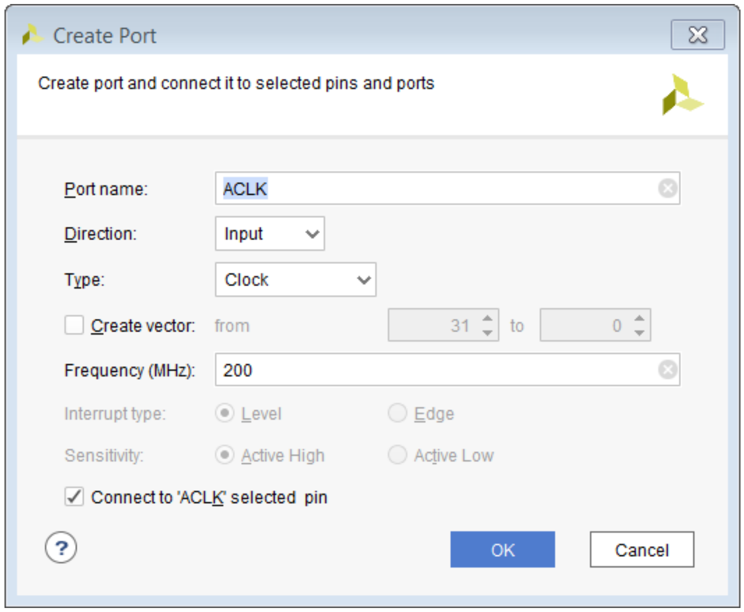


Figure 11: Create Port - ACLK

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

   The Create Port dialog box opens as seen in in the following figure.

7. Set the **Polarity** to **Active Low**.
8. Click **OK** to accept the default settings.

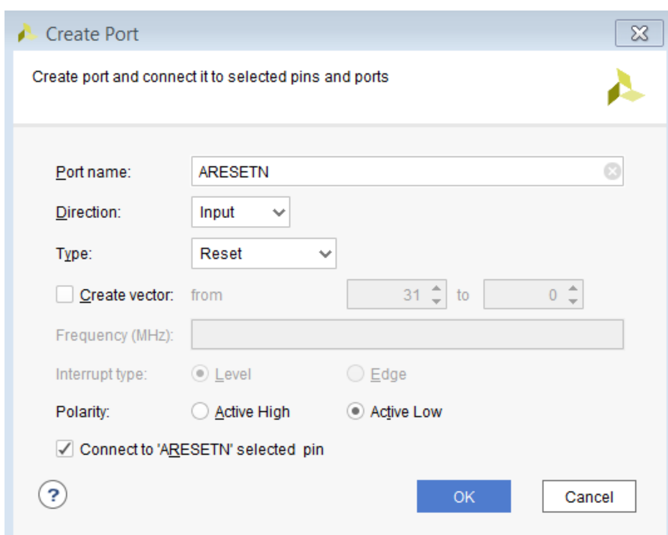


Figure 12: Create Port - ARESETN

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.*

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 synchronizes externally to the clock, then you need to associate the related clock with the reset. This does not require the Processor System Reset block.

9. Double click the **ACLK** port to open the Customize Port dialog box.
10. A clock is typically associated with a Bus Interface. In this case, we can associate this clock pin to the `S00_AXI` interface. In the **Associated Busif** field, type **S00_AXI**.
11. Set the **Associated Reset** field to **ARESETN**.
12. Click **OK.**

The dialog box looks like the following figure:

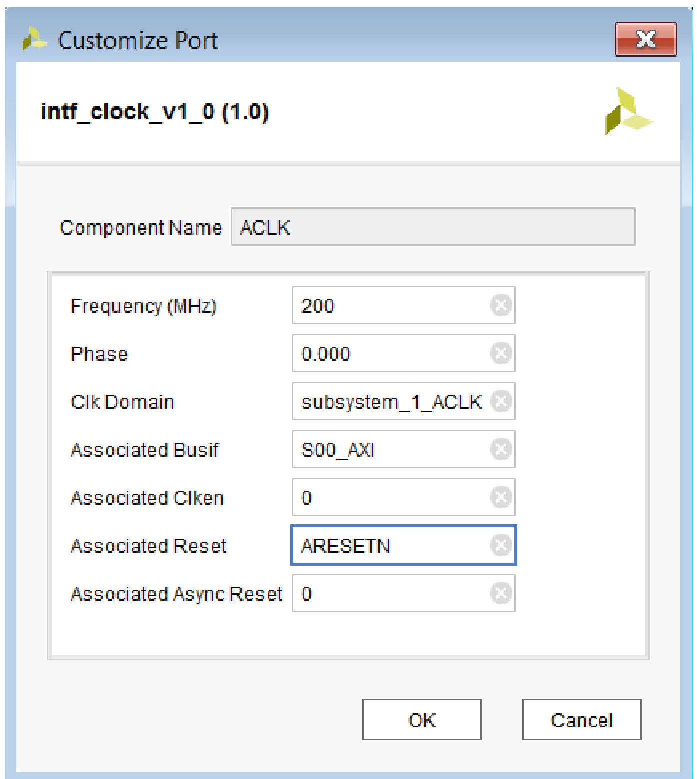


Figure 13: Customize Port – Associated Reset

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

13. 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.

14. Click and drag the cursor from the `S00_ACLK` pin to the `ACLK` port, as shown in the following figure.

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 you can make a valid connection between these points. The Vivado IP integrator highlights all possible connection points in the subsystem design as you interactively wire the pins and ports.

15. Release the mouse button and Vivado IP integrator makes a connection between the `S00_ACLK` pin and the `ACLK` port, as shown in the following figure:

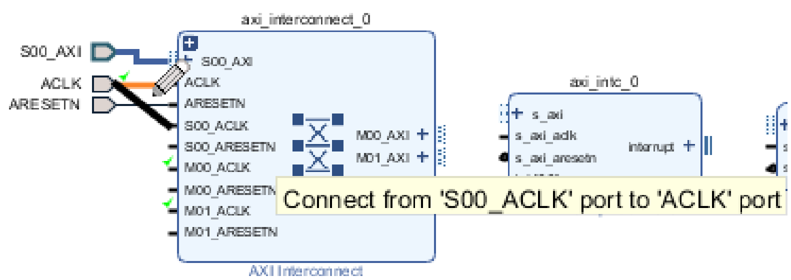


Figure 14: Connect the S00_ACLK Pin to the ACLK Port

16. 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 the following figure:

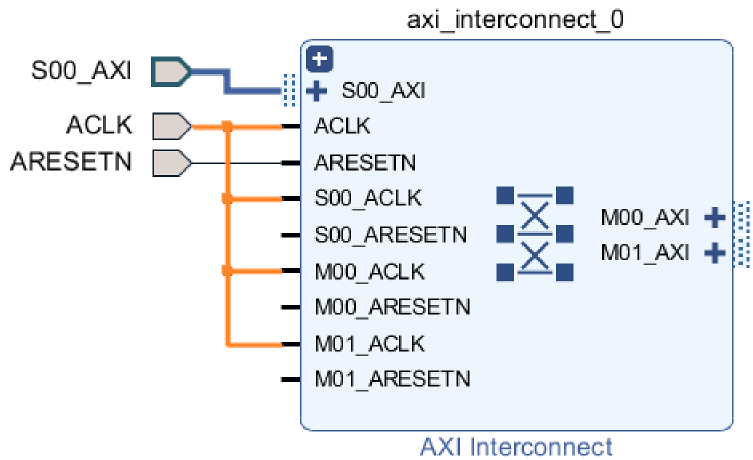


Figure 15: ACLK Connections

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

17. Place the cursor on the **S00_ARESETN** pin, then click and drag the cursor to the **ARESETN** port, as shown below.
18. Release the mouse button to make the connection.

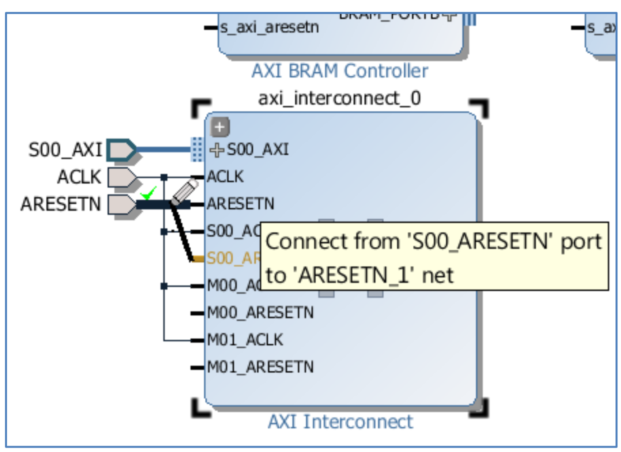


Figure 16: Connect S00_ARESETN to the ARESETN port

19. Repeat the steps to connect the **M00_ARESETN** and the **M01_ARESETN** pins of the AXI Interconnect to the **ARESETN** port, as shown in the following figure:

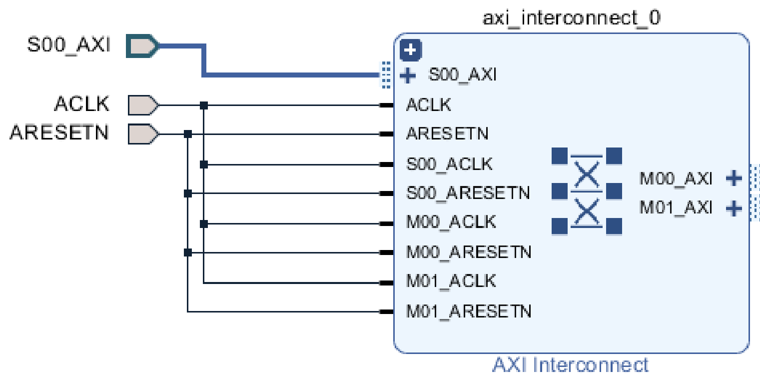


Figure 17: ARESETN Connections

Step 4: Customizing IP

1. Double-click the **AXI Interconnect** core to open the Re-Customize IP dialog box as seen in the following figure:

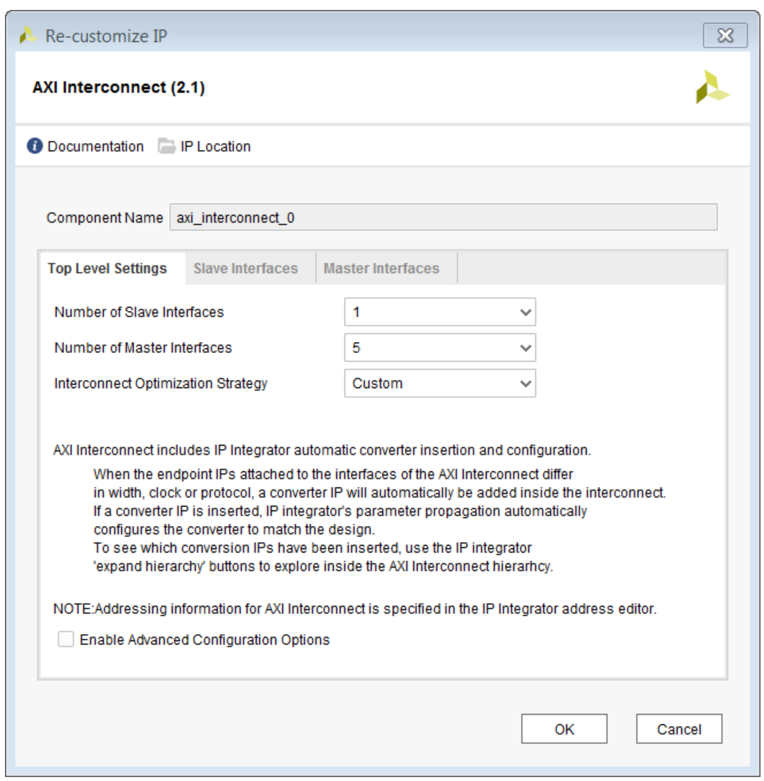


Figure 18: Re-customize the AXI Interconnect

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 the following figure:

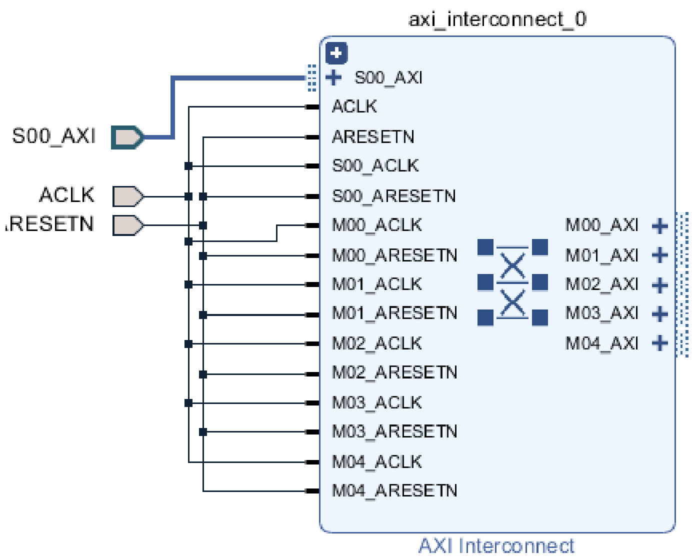


Figure 19: AXI Interconnect with 5 Master Interfaces

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

   Now you can connect the 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.

6. Connect the **s_axi_aclk** and the **s_axi_aresetn** pins of the AXI BRAM Controller to the **ACLK and ARESETN** ports. Figure 20 shows these connections.

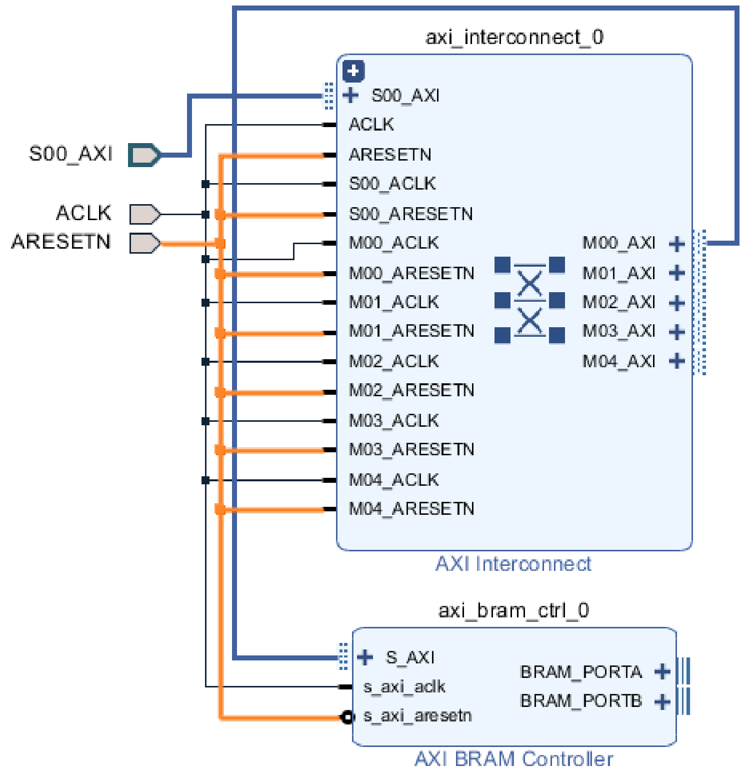


Figure 20: Connect the AXI BRAM Controller to the AXI Interconnect

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 shows in Figure 21.

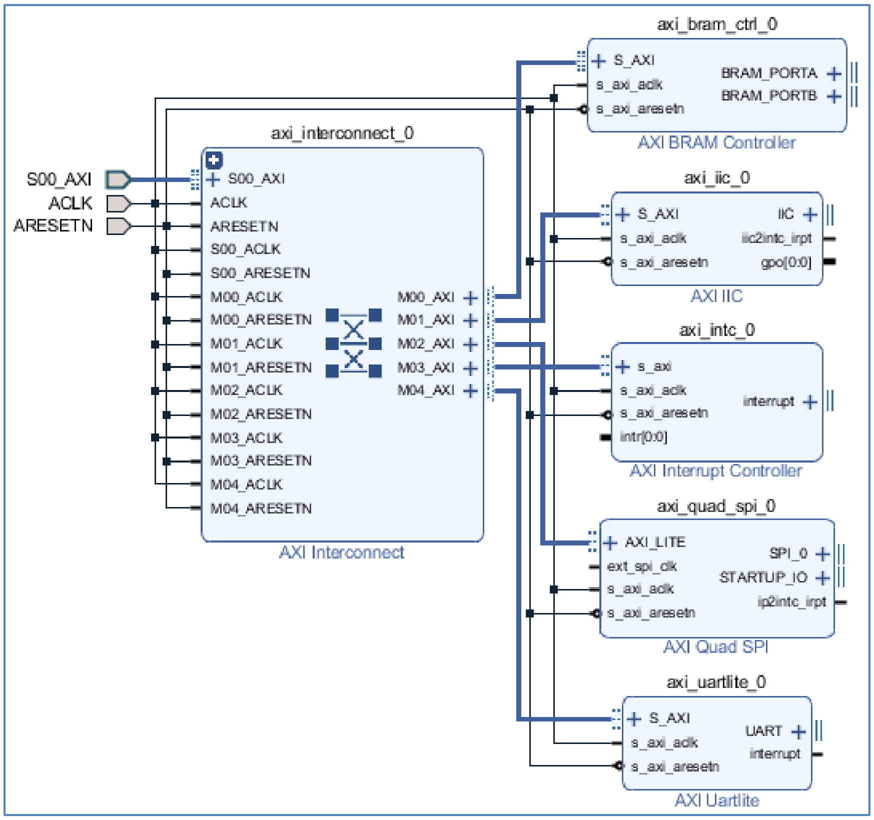


Figure 21: Connecting to the AXI Interconnect

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 you must connect external to the subsystem design, such as the following:

- UART interface of the AXI Uartlite
- `SPI_0` interface of the AXI Quad SPI
- 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 the **Run Connection Automation** link, in the banner at the top of the design canvas.

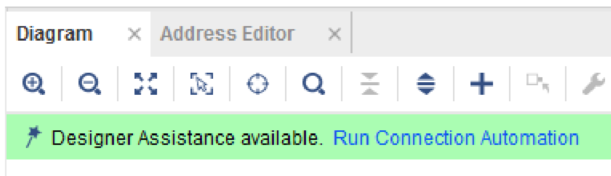


Figure 22: Run Connection Automation for /axi_quad_spi_0/SPI_0

The Run Connection Automation dialog box opens.

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

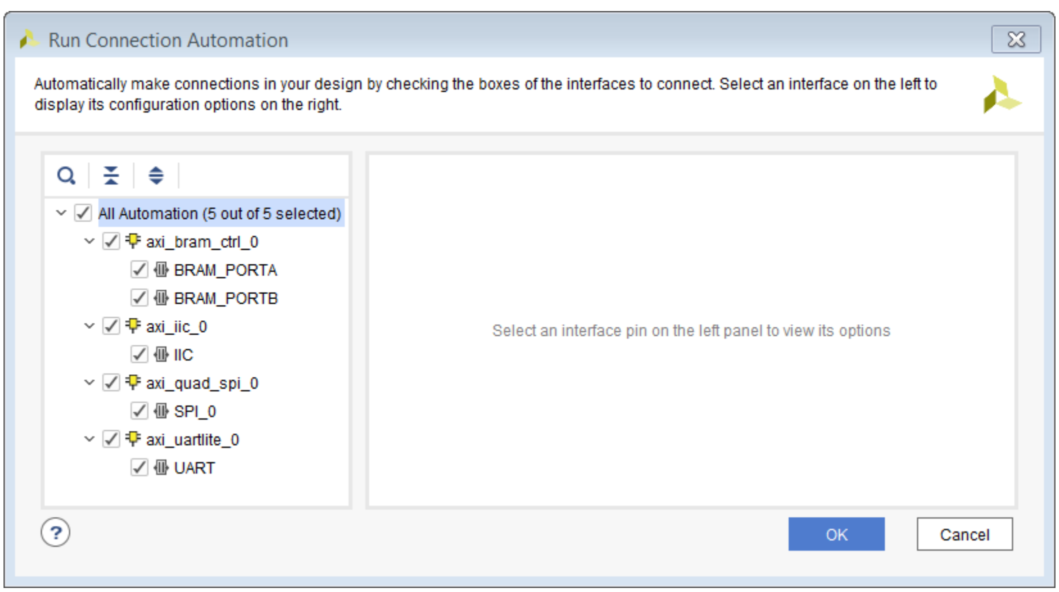


Figure 23: Run Connection Automation Dialog Box

3. Select and highlight the interfaces, as shown in the following figure, to see a description of the automation that the tool offers as well as any options needed to connect these interfaces.

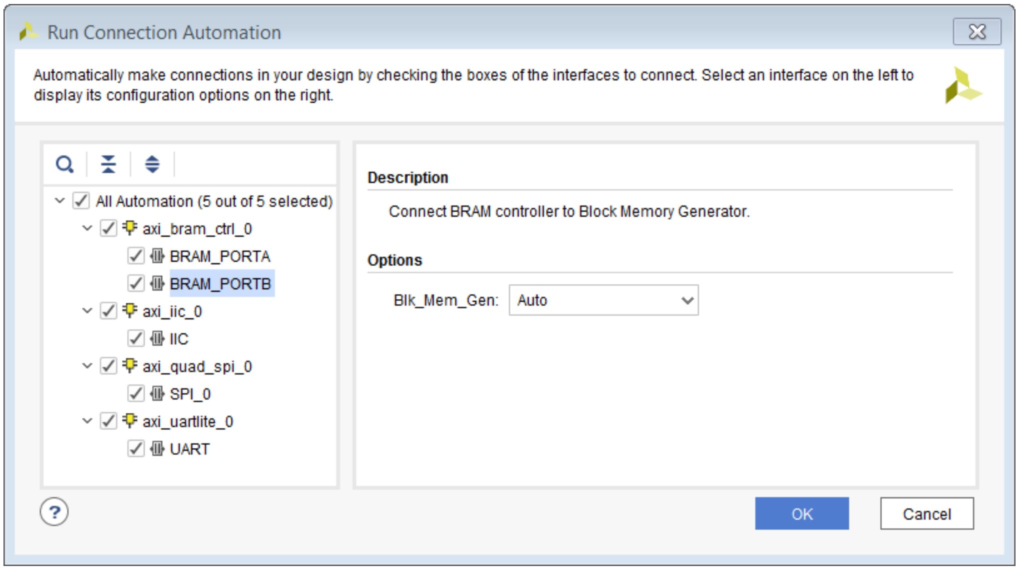


Figure 24: The Run Connection Automation Dialog Box Options

4. Click **OK**.

5. All the external interfaces connect to I/O ports, and the BRAM Controller connects to the Block Memory Generator, as shown in the following figure:

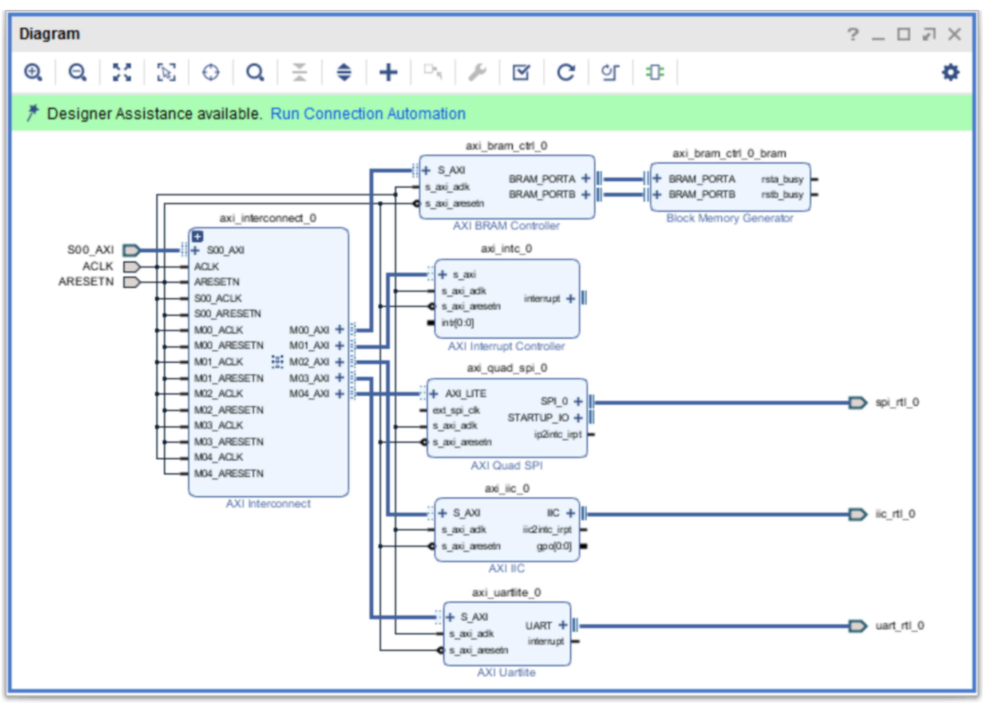


Figure 25: Port Connections after Running Connection Automation

6. Right-click 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_0` port named as it is.

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

   The Create Port dialog box opens as shown in the following figure:

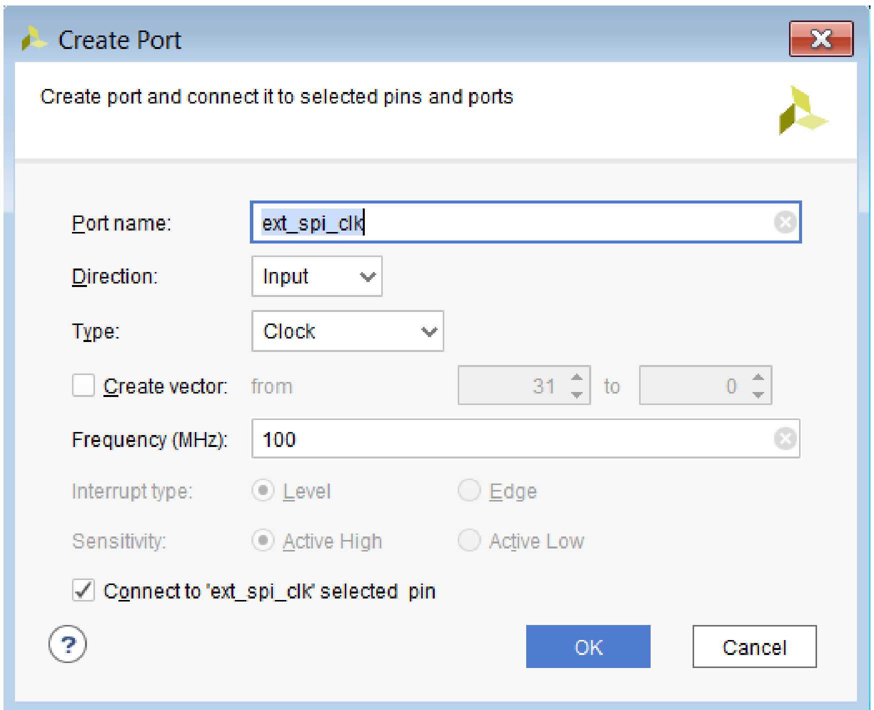


Figure 26: Create ext_spi_clk Port

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

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

   The optimized layout of the design should now look similar to Figure 27:

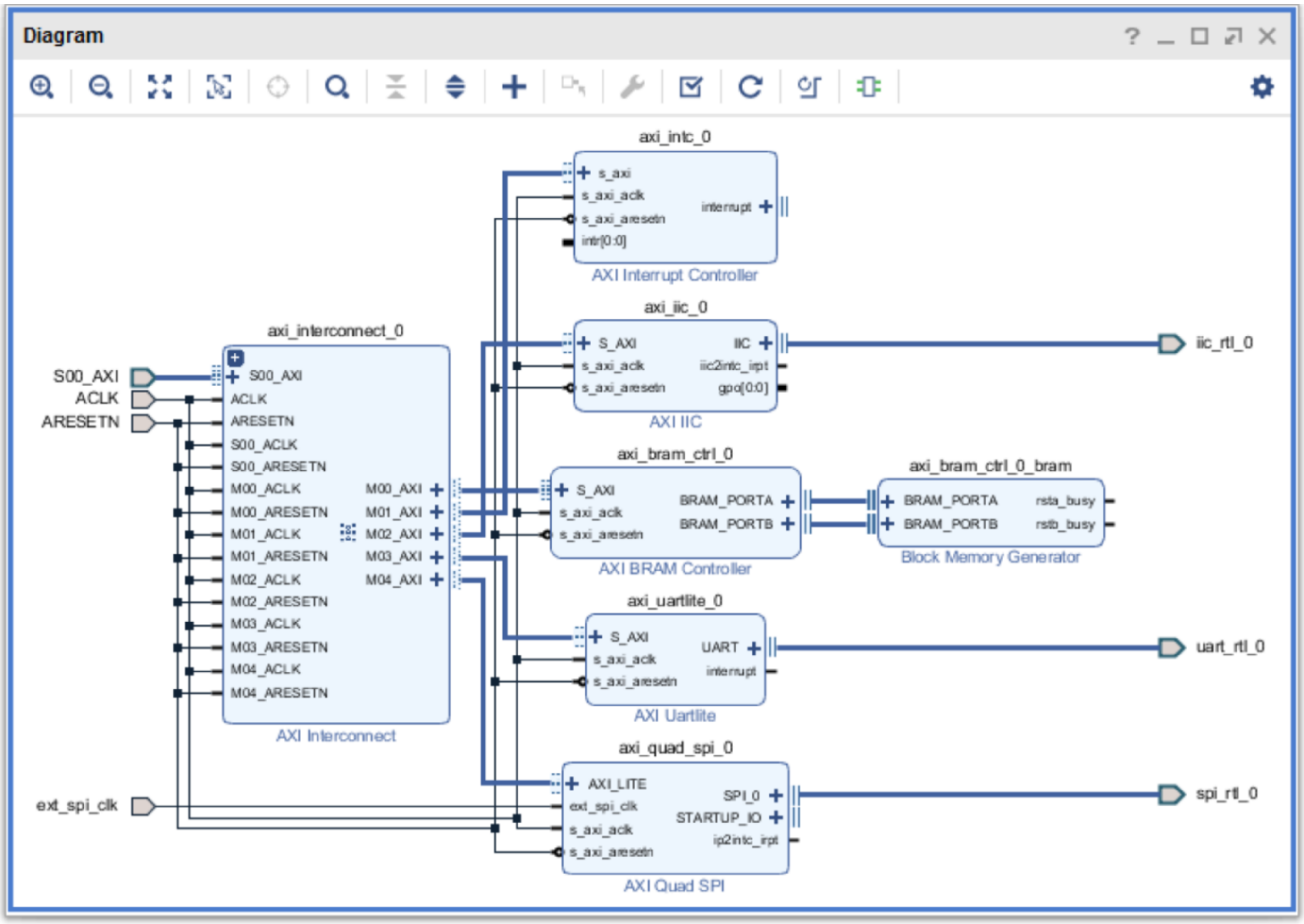


Figure 27: Completed IP Subsystem Design

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 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 the core.

The Xilinx Concat core is instantiated onto the IP integrator design canvas, as shown in the following figure:

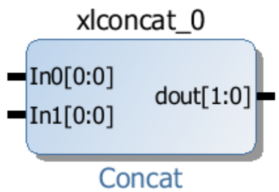


Figure 28: Xilinx Concat Block

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

   The Re-customize IP dialog box opens as shown in the following figure:

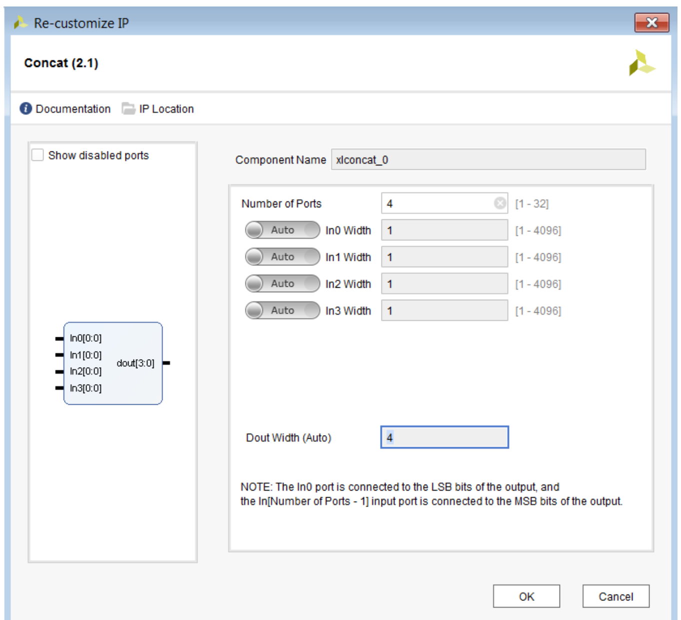


Figure 29: Customize Concat Block

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. You will use the fourth port to demonstrate tying a signal high or low with 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 30.

   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.

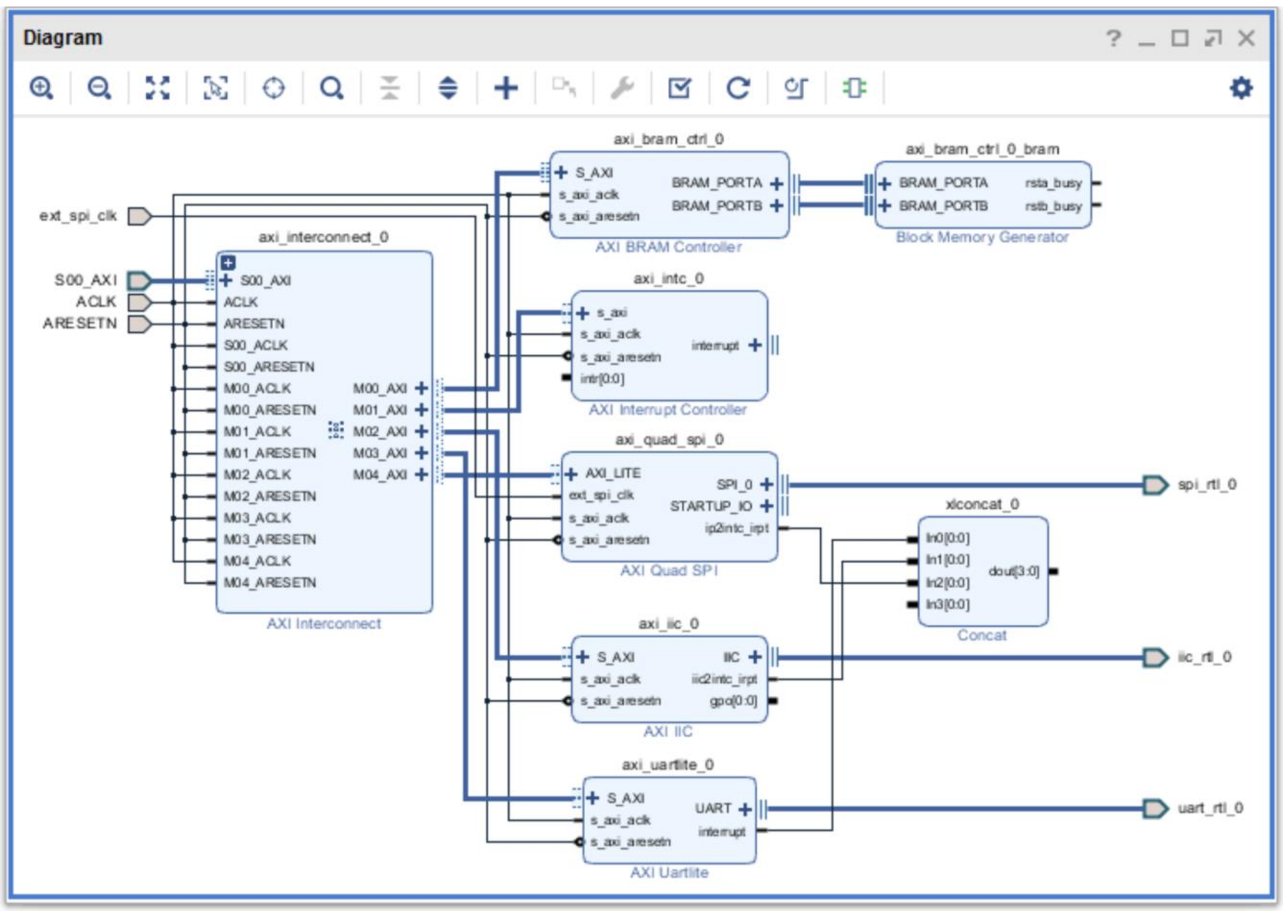


Figure 30: Connect the Interrupt Signals to Concat Block

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 and select **Add IP**.

    The IP catalog opens.

11. In the search field, type **cons** to find the Xilinx Constant block, and double-click 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 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, as shown in the following figure.

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

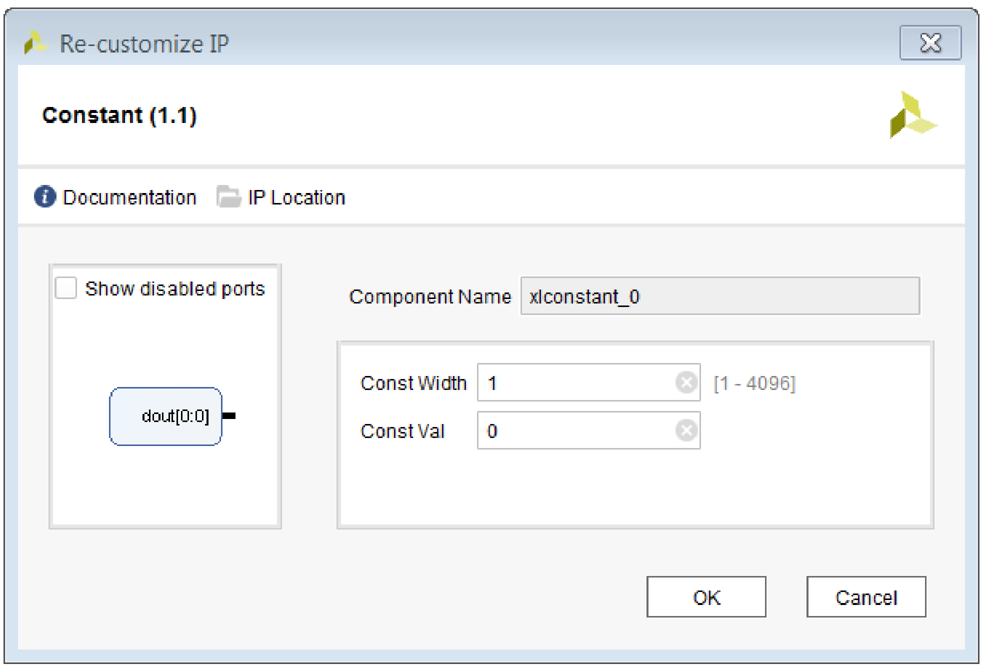


Figure 31: Set Constant Value

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 the following figure:

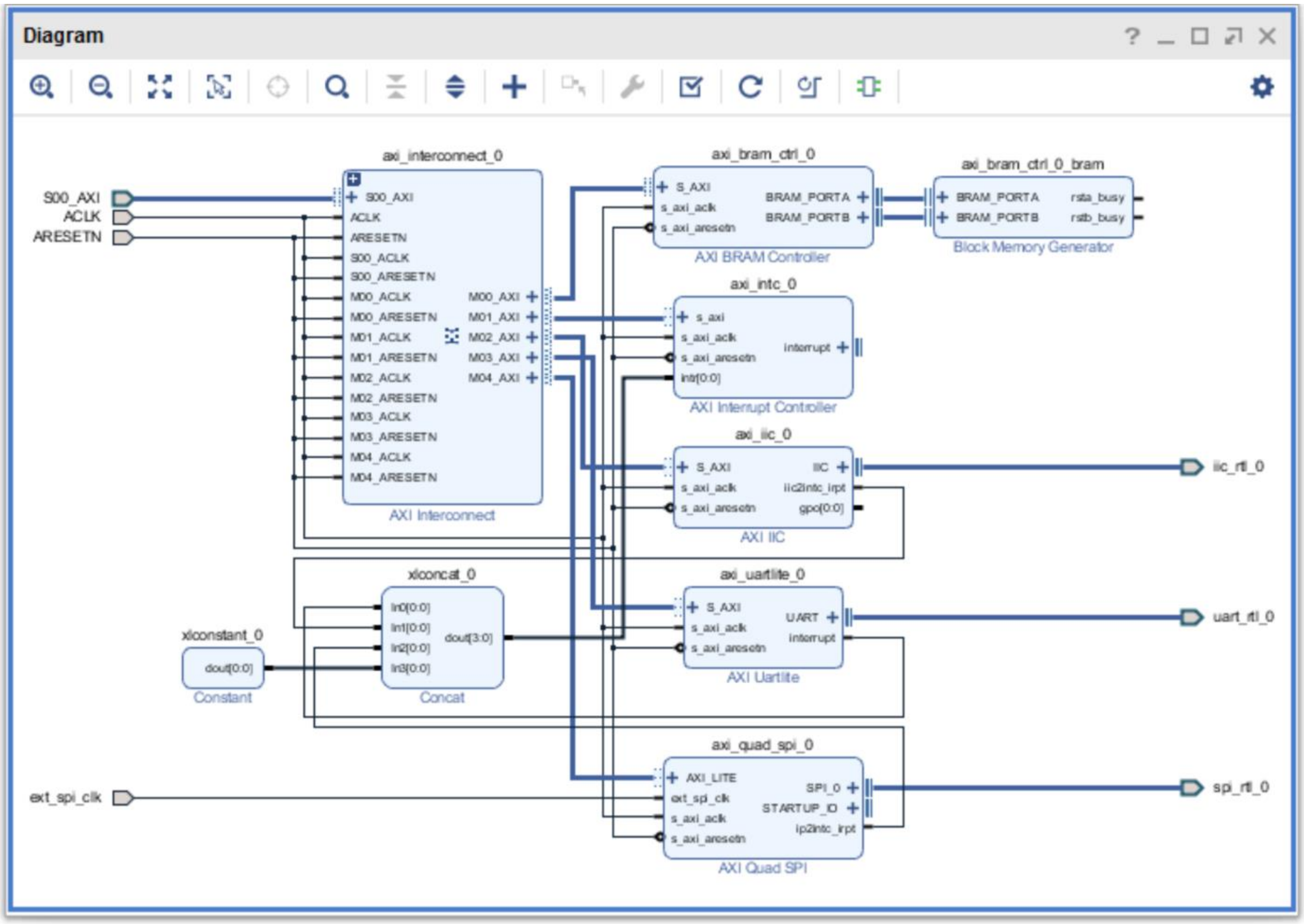


Figure 32: Connect Concat Block to Interrupt Controller

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. The Vivado tools correct this automatically 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 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. Typically, these memory maps are 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 you map a slave 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.

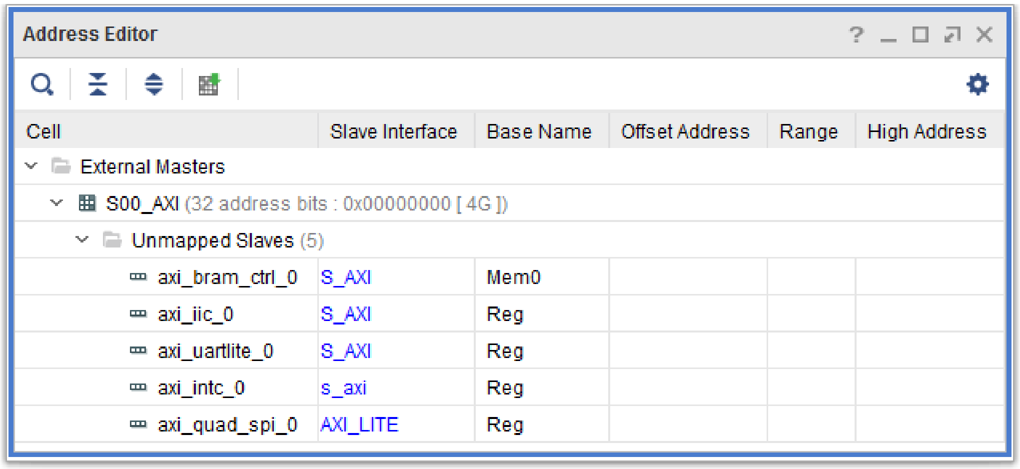


Figure 33: Address Editor

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** button, 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 the following figure:

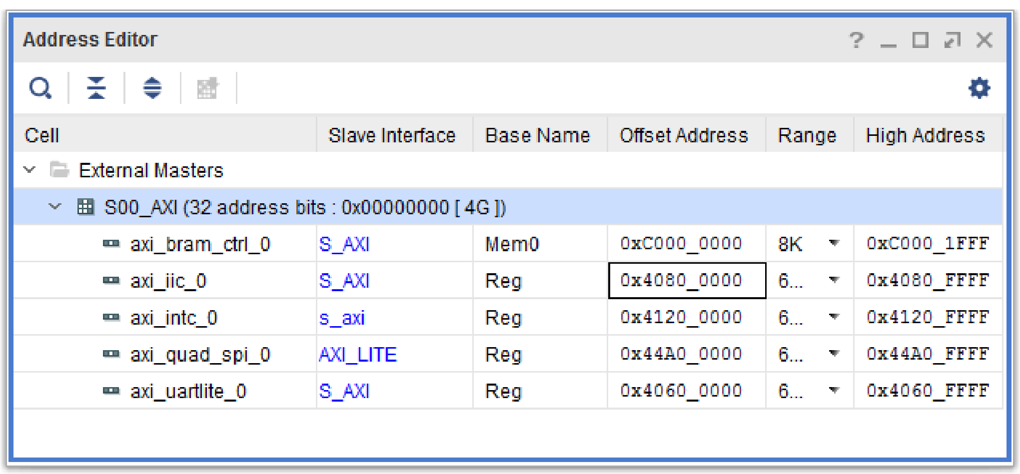


Figure 34: Automatically Assigned Addresses

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, shown in the following figure:

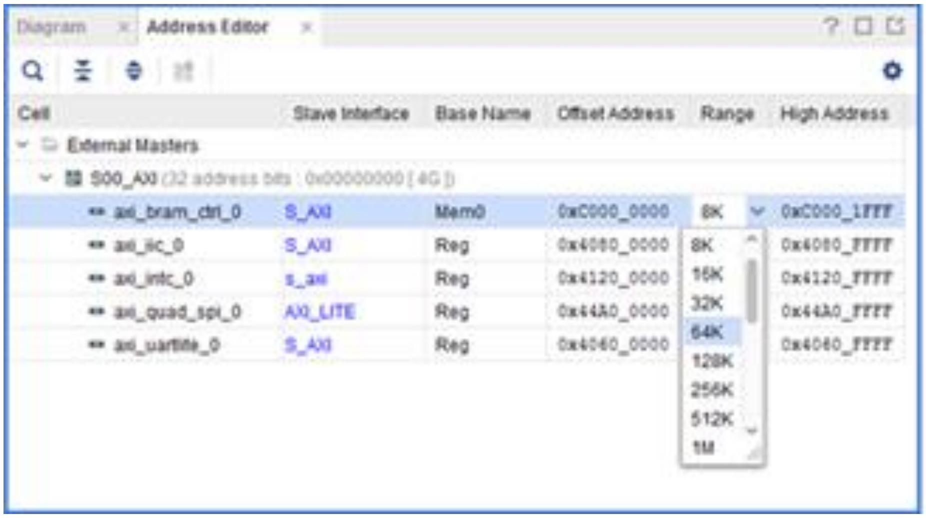


Figure 35: Change Range Field of the AXI BRAM Controller

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 (DRCs) by clicking the **Validate Design** button .

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

2. Click **OK**.
3. Examine the interrupt bus width on the **Interrupt Controller** block.

   Notice, as shown in the following figure, 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.

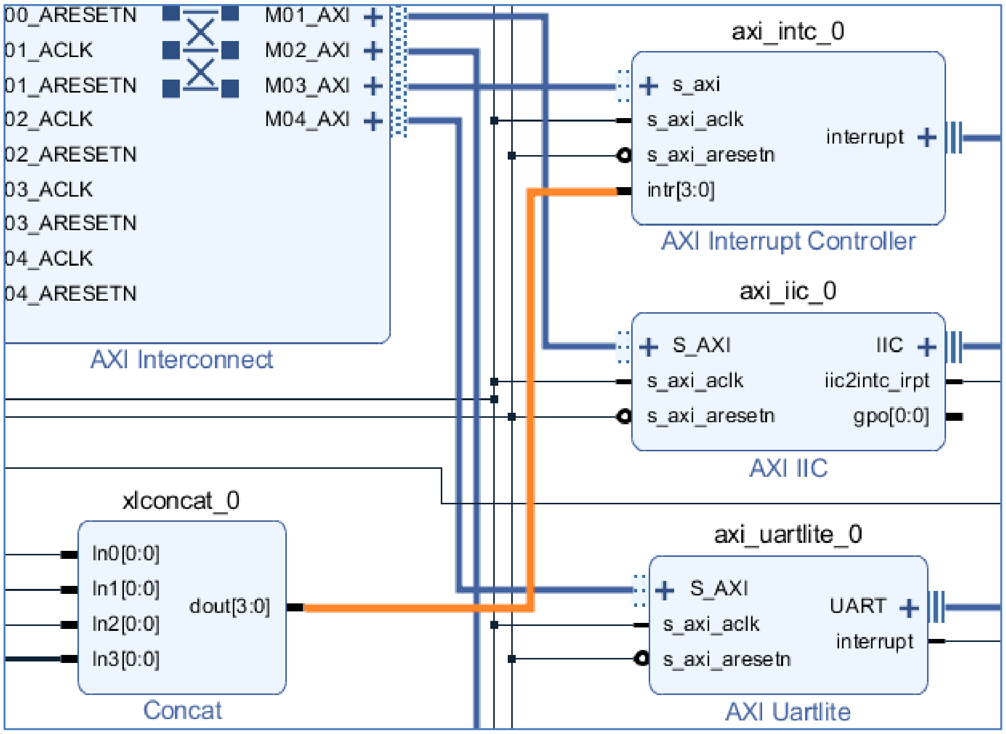


Figure 36: Parameter Propagation Matches Bus Width

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

4. 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.

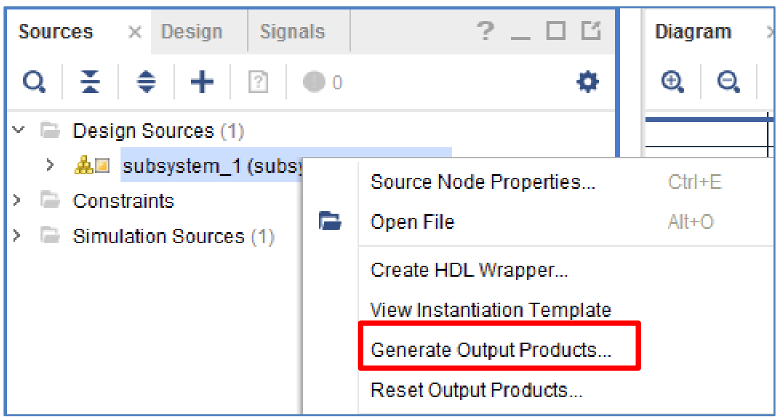


Figure 37: Generate Output Products

The Generate Output Product dialog box opens, as shown Figure 38, to regenerate the various output products associated with the subsystem design.

The Vivado IP integrator lets you choose how to handle the synthesis of the block design. The three **Synthesis Options** include:

- **Global**: Synthesizes the block design as part of the top-level project, rather than as an out-of-context block.
- **Out-of-Context per IP**: Synthesizes each IP in the block design separately, out-of-context of the block design or the top-level design. This prevents each IP from being synthesized unnecessarily, but requires updating and re-synthesizing each IP when it is updated.
- **Out-of-context per Block Design**: Synthesizes the entire block design at one time, but out-of-context from the global or top-level design. This prevents the block design from being synthesized unnecessarily when the top-level design is synthesized, but requires updating and re-synthesizing the block design when any of the IP in it are updated.

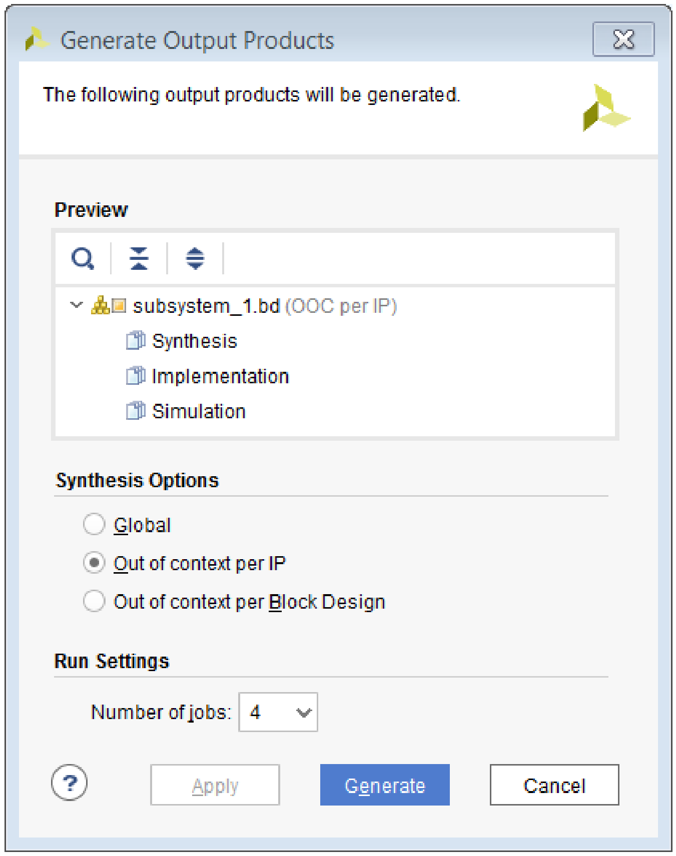


Figure 38: Generate Output Products dialog box

2. Leave the default selection of **Out of context per IP**.
3. Click **Generate** to generate all output products.

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

   The Generate Output Products dialog box opens to confirm the output products are generated.
4. Click **OK**.

5. The Out-of-context (OOC) runs for each IP in the design launch, shown in the Design Runs tab below. OOC runs can take a few minutes to finish.

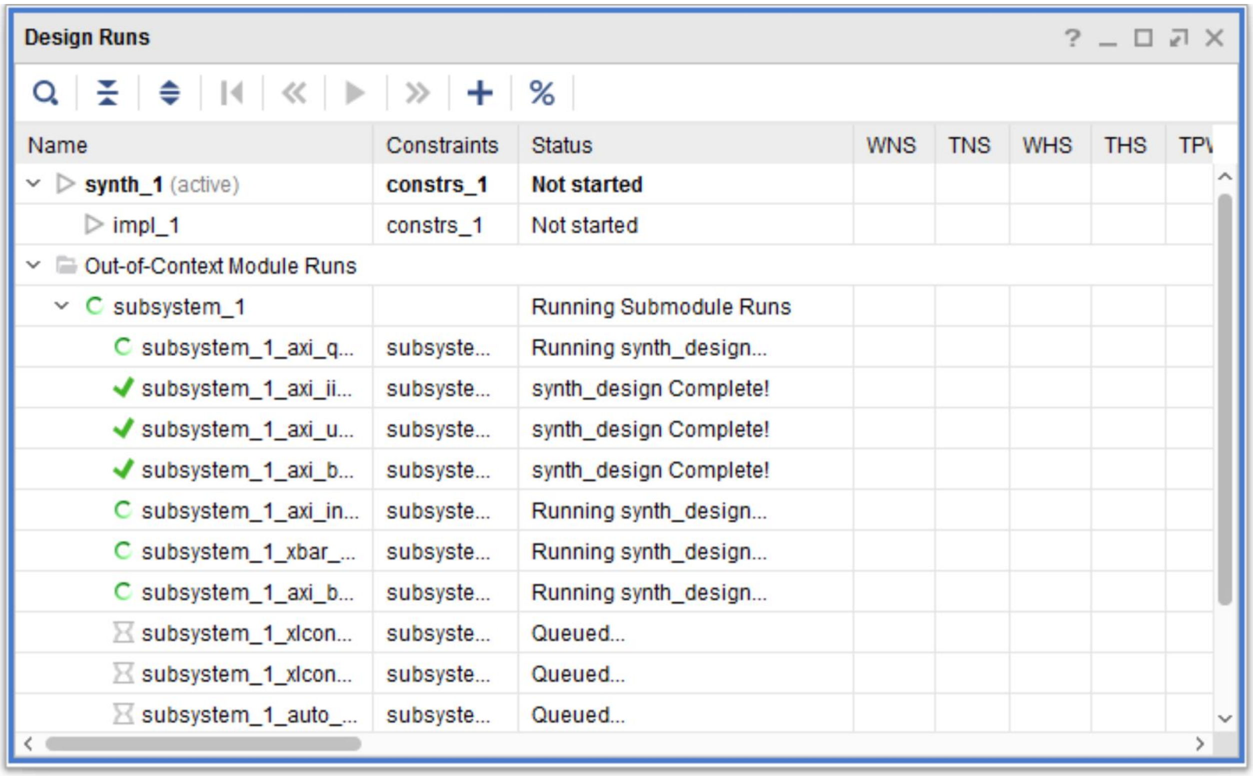


Figure 39: Out of context runs

6. After the Out-of-context runs are finished, in the Sources window, right-click the top-level subsystem design, **subsystem_1**, and select **Create HDL Wrapper**.

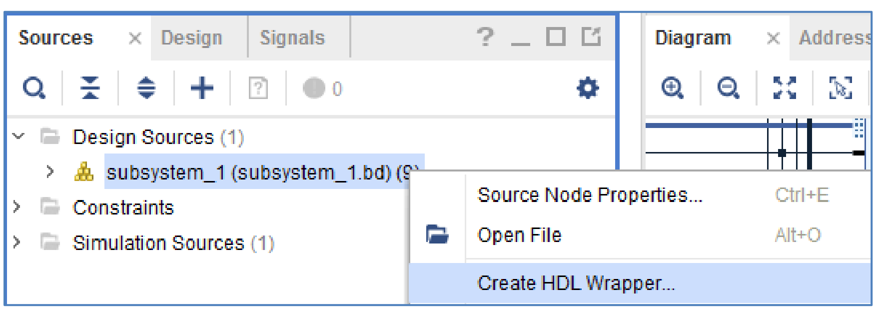


Figure 40: Create Top-Level HDL Wrapper

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 might 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.

7. Choose the **default** option, **let the Vivado manage wrapper and auto-update**, as shown in the following figure:

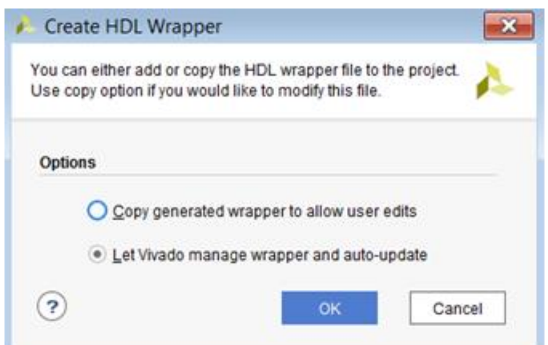


Figure 41: Create HDL Wrapper dialog box

8. 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.

9. From the **Flow Navigator**, click **Add Sources**.
10. Select the **Add or Create Constraints** option and click **Next**.

    The Add Sources dialog box opens as seen in Figure 42.

11. In the Add or Create Constraints dialog box, click the + icon select **Add Files** or click the **Add Files** button to add files.

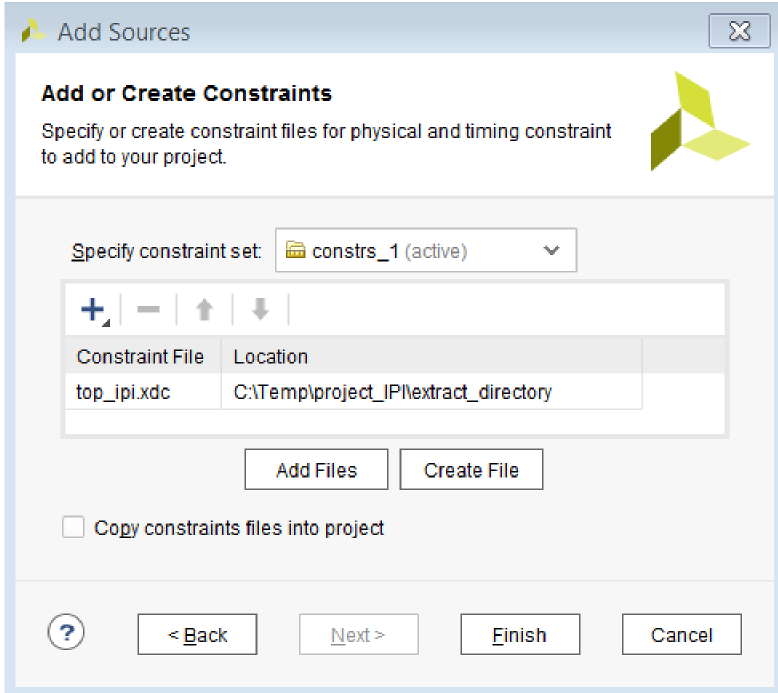


Figure 42: Add Constraints File

The Add Constraints Files dialog box opens.

12. Select the `top_ipi.xdc` file in `<Extract_Dir>`, and click **OK**.
13. In the Add or Create Constraints dialog box, make sure that **Copy constraints files into project** is checked.
14. 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.

15. 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 the following figure:

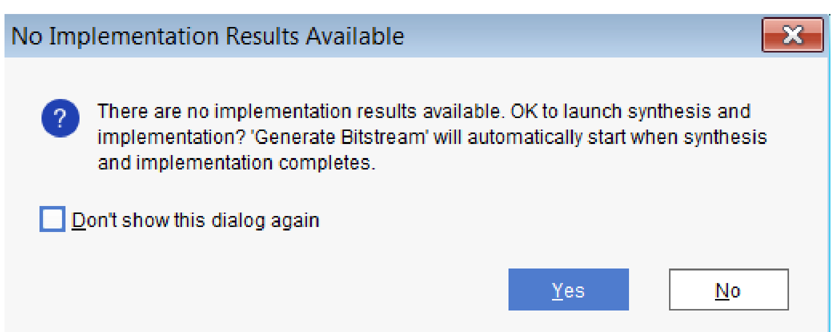


Figure 43: No Implementation Results Available

16. Click **Yes**.
17. The Launch Runs dialog box pops up where you can specify various options for launching the runs. Click **OK**.
18. After the Vivado Design Suite generates the bitstream, the Bitstream Generation Completed dialog box opens, as shown in the following figure:

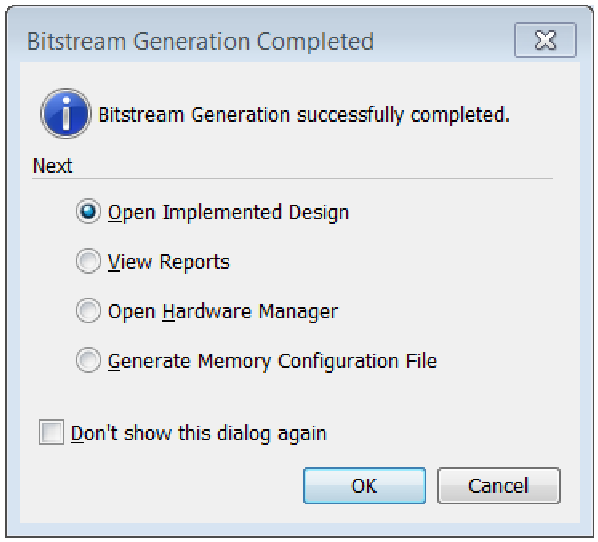


Figure 44: Bitstream Generation Completed

19. Click **OK**.
20. Verify that the design meets timing by looking at the Timing window as seen in the following figure:

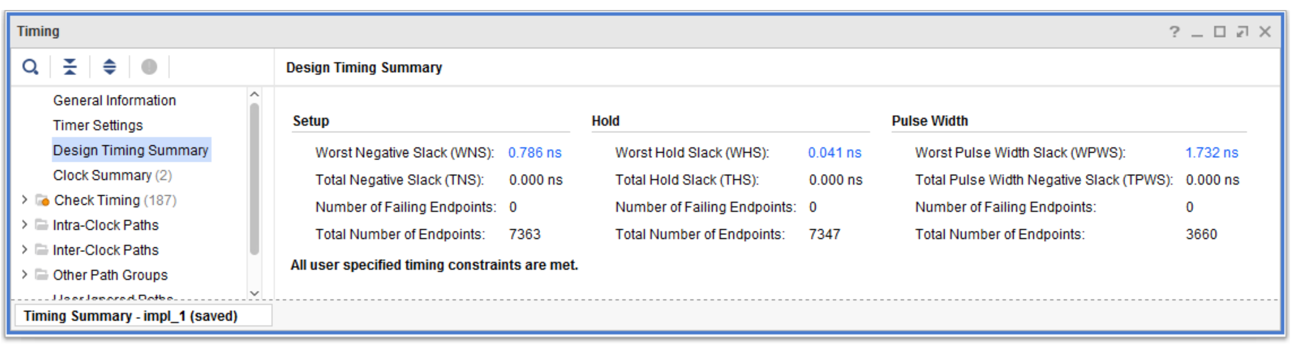


Figure 45: Timing Summary Report

TIP: *The Timing Summary also reports warnings related to the lack of input and output delays for the primary input and output ports of the design. You can add these delays as design constraints using the set_input_delay and set_ouptut_delay commands. See this link in the Vivado Design Suite User Guide: Using Constraints (UG903) for more information on setting input and output delays.*

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.

Please Read: Important Legal Notices