

# Vivado Design Suite Tutorial

## *Designing IP Subsystems Using IP Integrator*

Vivado Design Suite

UG995 (v2024.2) November 13, 2024

AMD Adaptive Computing is creating an environment where employees, customers, and partners feel welcome and included. To that end, we're removing non-inclusive language from our products and related collateral. We've launched an internal initiative to remove language that could exclude people or reinforce historical biases, including terms embedded in our software and IPs. You may still find examples of non-inclusive language in our older products as we work to make these changes and align with evolving industry standards. Follow this [link](#) for more information.





# Table of Contents

- Designing IP Subsystems..... 3**
  - Introduction..... 3
  - Navigating Content by Design Process..... 3
  - Tutorial Design Description..... 4
  - Locating Tutorial Design Files.....4
  - Hardware and Software Requirements.....4
- Chapter 1: Designing IP Subsystems in IP Integrator..... 5**
  - Step 1: Creating a Project..... 5
  - Step 2: Creating an IP Integrator Design..... 8
  - Step 3: Creating External Connections..... 11
  - Step 4: Customizing IP..... 18
  - Step 5: Running Connection Automation.....22
  - Step 6: Managing Signals with CONCAT and CONSTANT Blocks.....27
  - Step 7: Using the Address Editor.....32
  - Step 8: Validating the Design.....34
  - Step 9: Creating and Implementing the Top-Level Design..... 35
  - Conclusion.....40
- Appendix A: Additional Resources and Legal Notices..... 41**
  - Finding Additional Documentation.....41
  - Support Resources.....42
  - References.....42
  - Revision History.....42
  - Please Read: Important Legal Notices..... 42

# Designing IP Subsystems

## Introduction



**IMPORTANT!** This tutorial requires the use of the AMD Kintex™ 7 family of devices. You need to update your AMD 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.

The AMD Vivado™ Design Suite IP integrator 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 typically construct designs at the AXI-interface level for greater productivity; but you can 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. You instantiate a few IPs in the IP integrator and stitch them up to create an IP subsystem design. While working through this tutorial, you are introduced to the IP integrator GUI, run design rule checks (DRC) on your design, and integrate the design into a top-level design in the Vivado Design Suite. Finally, you 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.

## Navigating Content by Design Process

AMD Adaptive Computing documentation is organized around a set of standard design processes to help you find relevant content for your current development task. You can access the AMD Versal™ adaptive SoC design processes on the [Design Hubs](#) page. You can also use the [Design Flow Assistant](#) to better understand the design flows and find content that is specific to your intended design needs. This document covers the following design processes:

- **Hardware, IP, and Platform Development:** Creating the PL IP blocks for the hardware platform, creating PL kernels, functional simulation, and evaluating the AMD Vivado™ timing, resource use, and power closure. Also involves developing the hardware platform for system integration. Topics in this document that apply to this design process include:

- [Step 1: Creating a Project](#)
- [Step 2: Creating an IP Integrator Design](#)
- [Step 4: Customizing IP](#)
- **System Integration and Validation:** Integrating and validating the system functional performance, including timing, resource use, and power closure. Topics in this document that apply to this design process include:
  - [Step 7: Using the Address Editor](#)
  - [Step 8: Validating the Design](#)
  - [Step 9: Creating and Implementing the Top-Level Design](#)

---

## 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, and also 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 AMD Artix™ 7 device for use with the version of the Vivado Design Suite. The tutorial results should be similar.

---

---

## Locating Tutorial Design Files

1. Download the [Reference Design Files](#) from the AMD website.
2. Extract the file named `top_ipi.xdc` to a directory. That directory is called the `<Extract_Dir>` in this tutorial.

---

## 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.

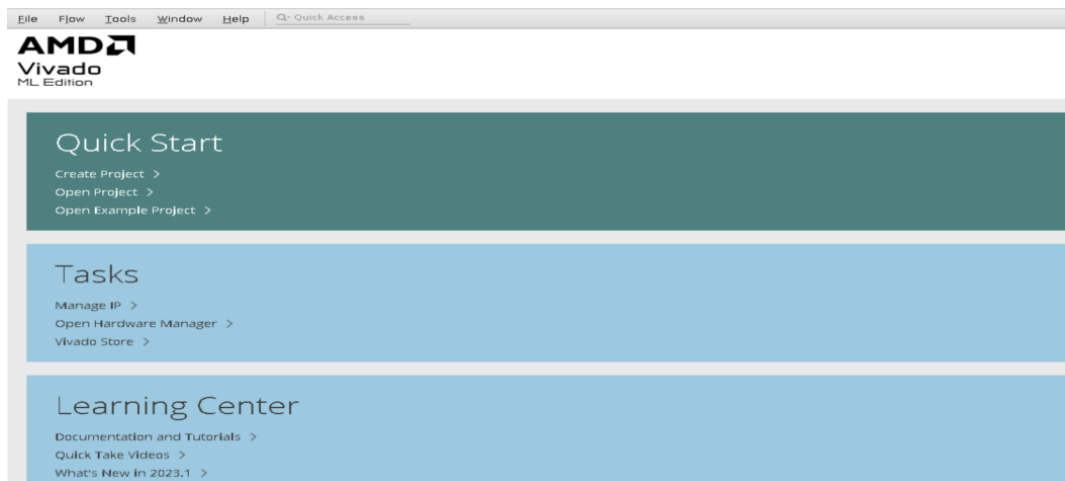
# Designing IP Subsystems in IP Integrator

## Step 1: Creating a Project

1. Open the AMD Vivado™ Integrated Design Environment (IDE).
  - On Linux, change to the directory where the Vivado tutorial design file is stored: `cd <Extract_Dir>/Vivado_Tutorial`. Then, launch the Vivado Design Suite: Vivado.
  - On Windows, launch the Vivado Design Suite: **Start → All Programs → Xilinx Design Tools → Vivado 2024.x**.

As an alternative, click **Vivado 2024.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:



**Note:** Your Vivado Design Suite installation can be called something different from AMD Design Tools on the Start menu.

2. Under the Quick Start section, select **Create Project**.
3. The New Project wizard opens. Click **Next** to confirm the project creation.
4. In the Project Name page shown in the following figure, set the following options:

- a. In the Project name field, enter `project_ipi`.
- b. In the Project location field, enter `<project_directory>`.

**New Project**

**Project Name**  
Enter a name for your project and specify a directory where the project data files will be stored.

Project name:

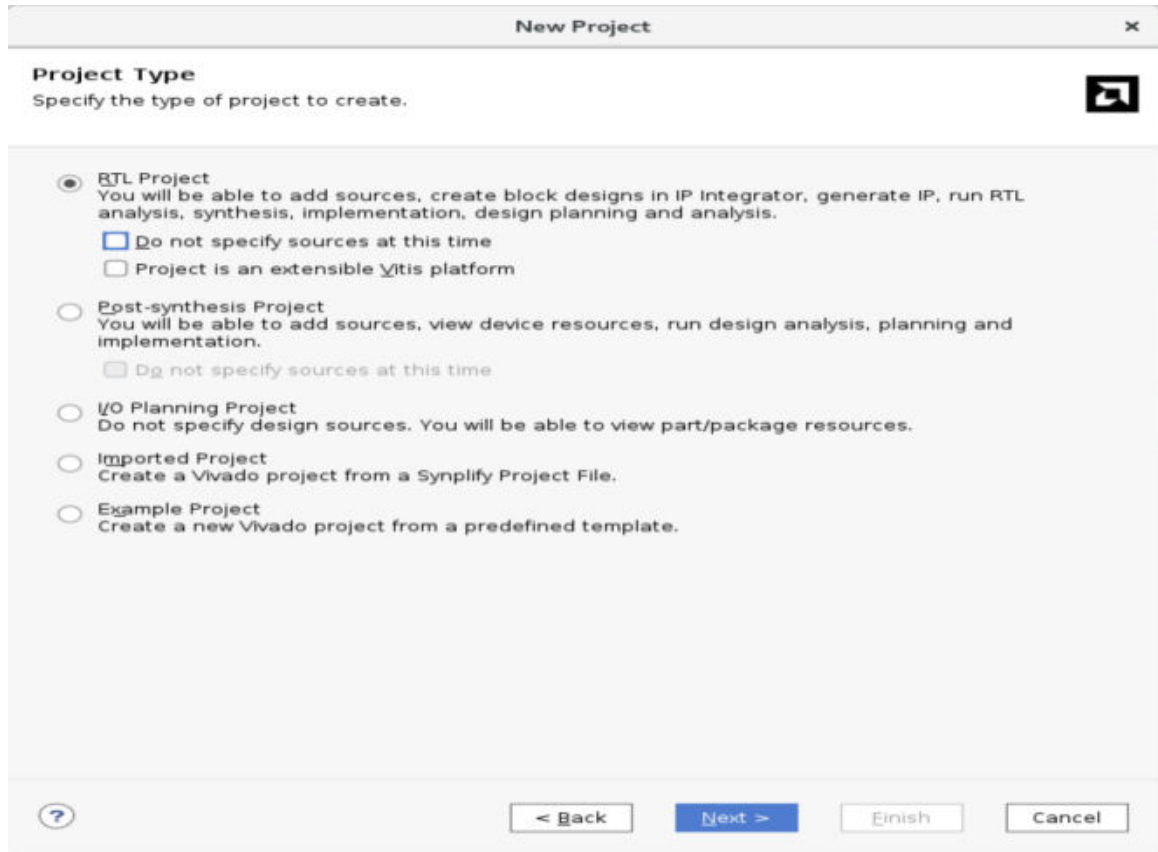
Project location:

☒ Create project subdirectory

Project will be created at: /tutorials/UG995/project\_ipi

[?](#) [< Back](#) [Next >](#) [Finish](#) [Cancel](#)

5. Ensure that Create Project Subdirectory is checked, and click **Next**.
6. In the Project Type page, select **RTL Project**, and click **Next**, as shown in the following figure:

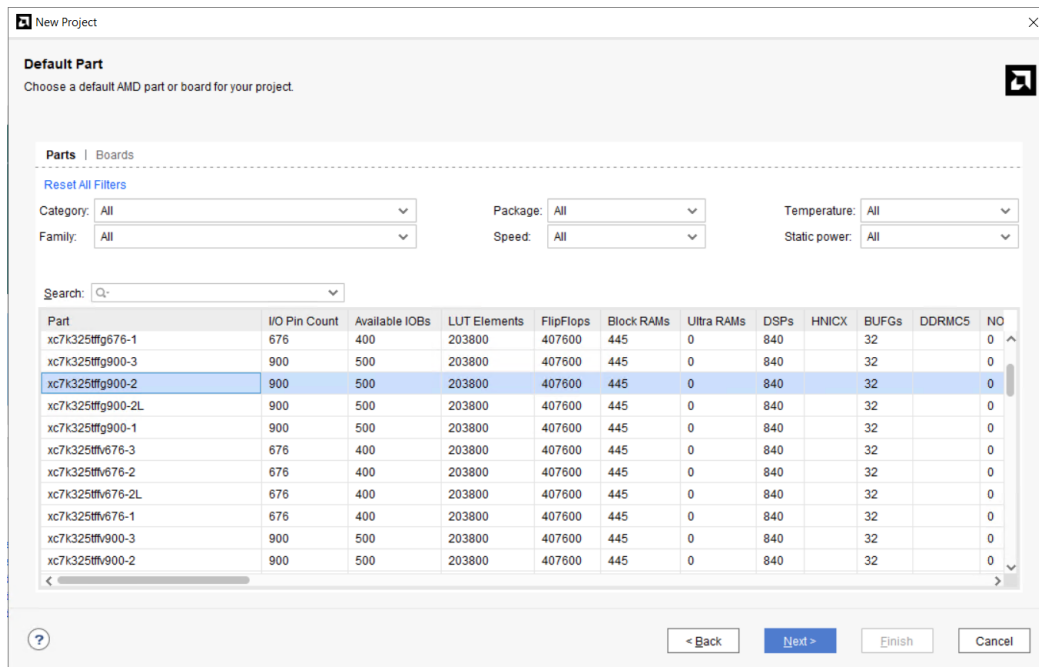


Ensure that the **Do not specify sources at this time** is unchecked. If you check this box, you do not see the other options mentioned below until you get to the **Default Part** page.

7. In the Add Sources page:
  - a. For Target language, select **VHDL** or **Verilog**.
  - b. For Simulator Language, select **Mixed**.
8. Click **Next**.

You add sources later using the design canvas in the Vivado IP integrator to create a subsystem design.

9. In the Add Constraints page, click **Next**.
10. In the Default Part page shown in the following figure, make the following entries:
  - a. Select **Parts**.
  - b. For Family, select **Kintex-7**.
  - c. For Speed, select **-2**.



11. Select **xc7k325tffg900-2** part from the listed parts, and click **Next**.

12. Review the project summary in the New Project Summary page.

13. Click **Finish** to create the project.

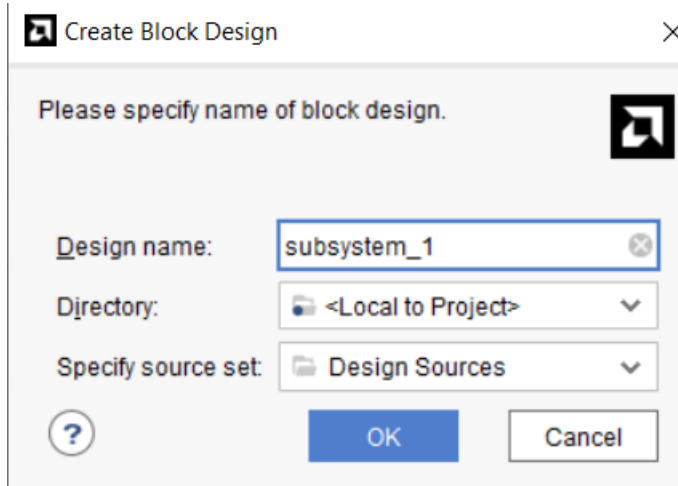
The new project opens in the Vivado IDE.

## 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:





2. In the Create Block Design dialog box, set the following:
  - a. For Design Name, select **subsystem\_1**.
  - b. For Directory, select **<Local to Project>**.
  - c. For Specify source set, select: **Design Sources**.
3. Click **OK**.

The Vivado IP integrator displays a design canvas to let you quickly create complex subsystem designs by integrating IP cores.

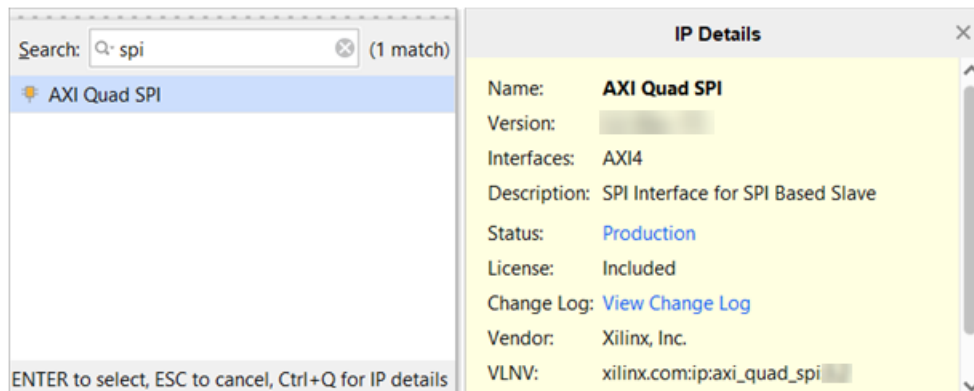
4. Click **Add IP** button  in the block design canvas.

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

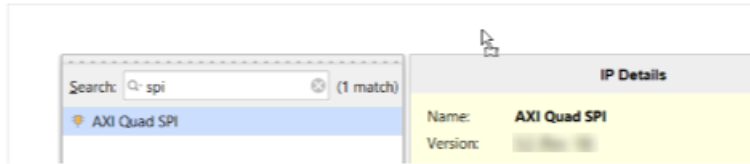


**TIP:** To open the IP Details window beside the IP catalog, as shown in the following figure, type *Ctrl-Q* as described at the bottom of the IP catalog window. This window lets you see details of the currently selected IP in the catalog.

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



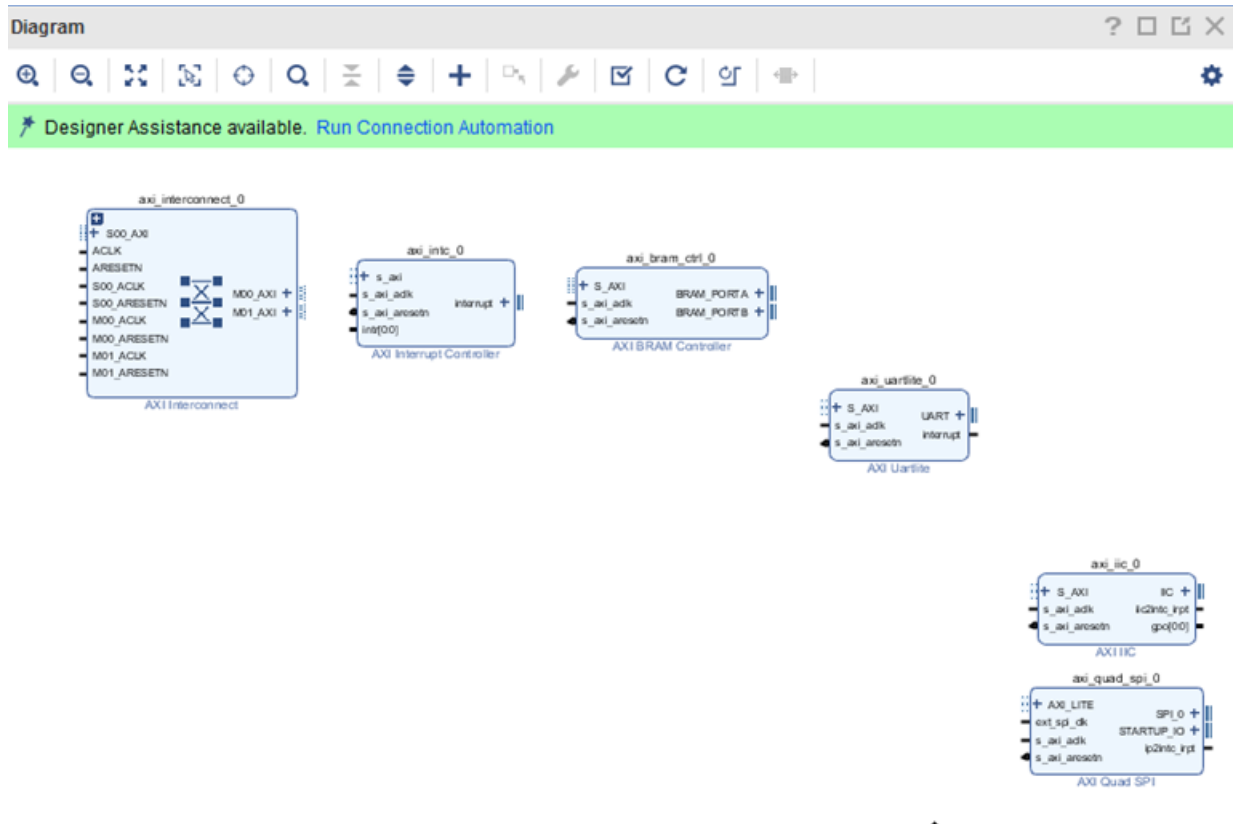
6. Select **AXI Quad SPI** core and press **Enter** on the keyboard, or double-click the core in the IP catalog. One more way of adding an IP is dragging and dropping the IP from the IP catalog to the block design canvas. In this case, you would search for the IP, select it and drag-and-drop it on the block design canvas.



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

7. Right-click the IP integrator canvas to open the pop up menu, and select **Add IP**.
8. In the Search field of the IP integrator catalog, type **IIC**.
9. Either double-click or press **Enter** on your keyboard to instantiate the AXI IIC IP.
10. Use the Add IP command to instantiate the following IP cores:
  - AXI UART Lite
  - AXI Block RAM (BRAM) Controller
  - AXI Interrupt Controller (INTC)
  - AXI Interconnect

The IP integrator canvas should look similar to the following figure. The relative positions of the blocks placed on the canvas can be slightly different.

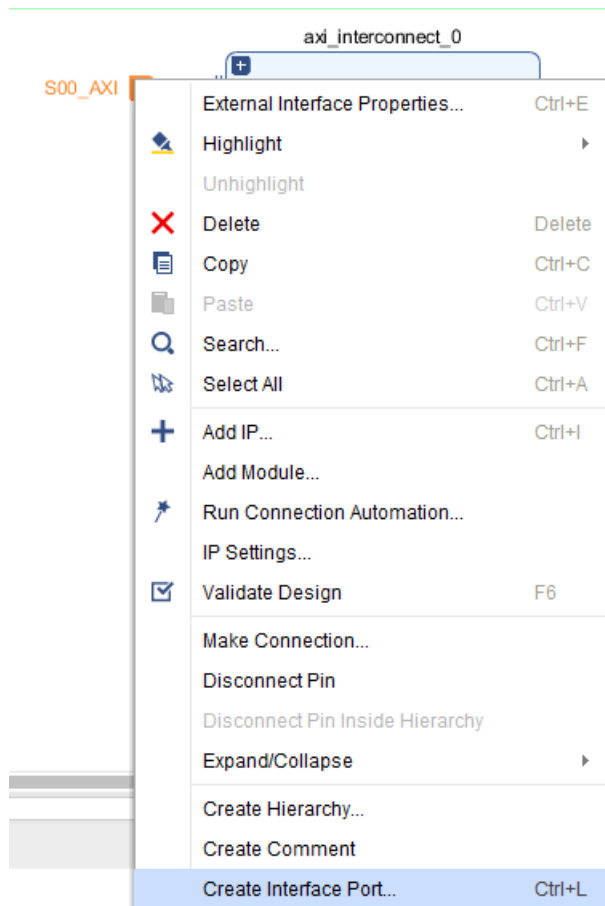


## Step 3: Creating External Connections

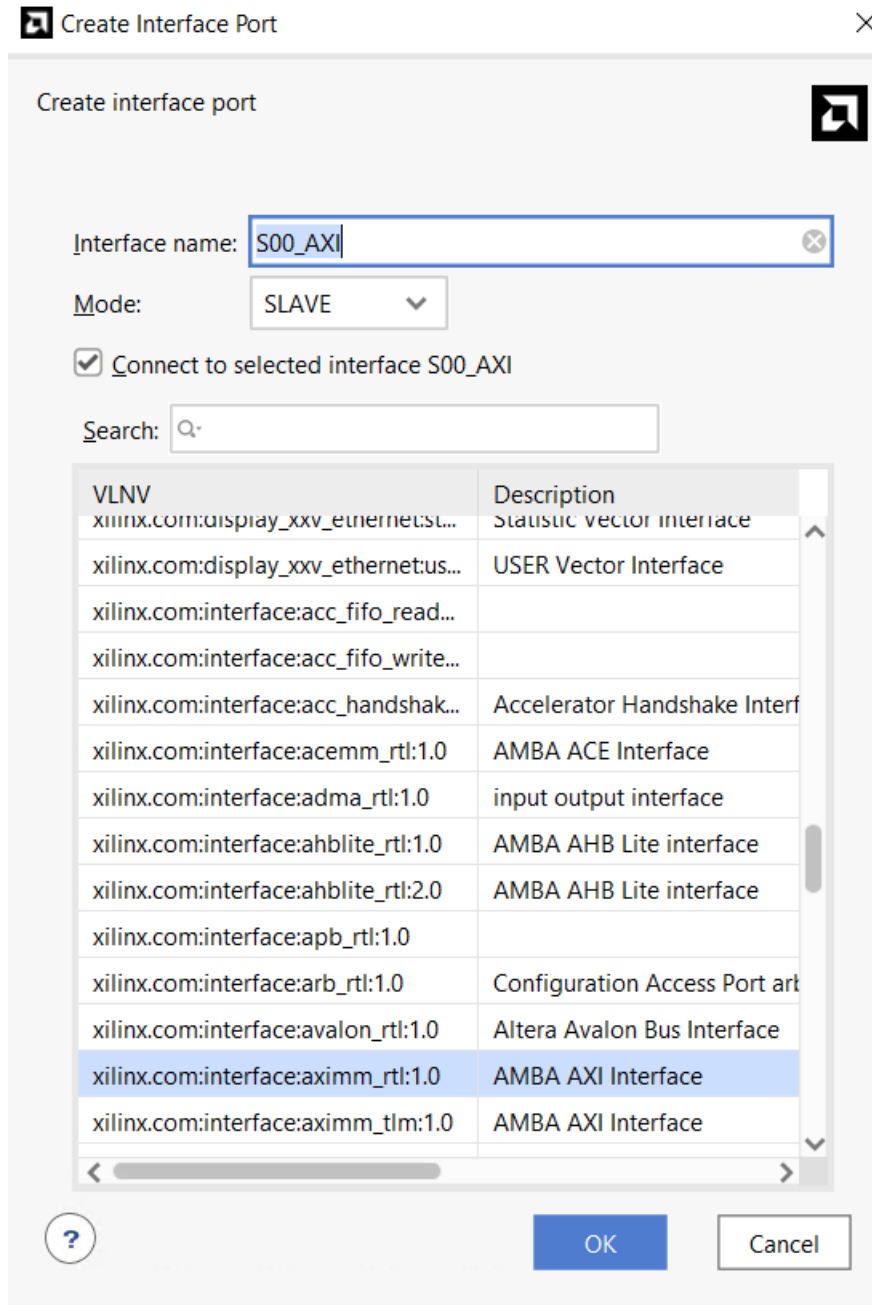
At this point, you have instantiated several AXI slaves that you can access through an external master, such as an on-board processor. To connect to an external master controlling these slaves, you connect the `S00_AXI` interface pin on the AXI Interconnect to an external port.

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 pop up menu and select **Create Interface Port**.



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

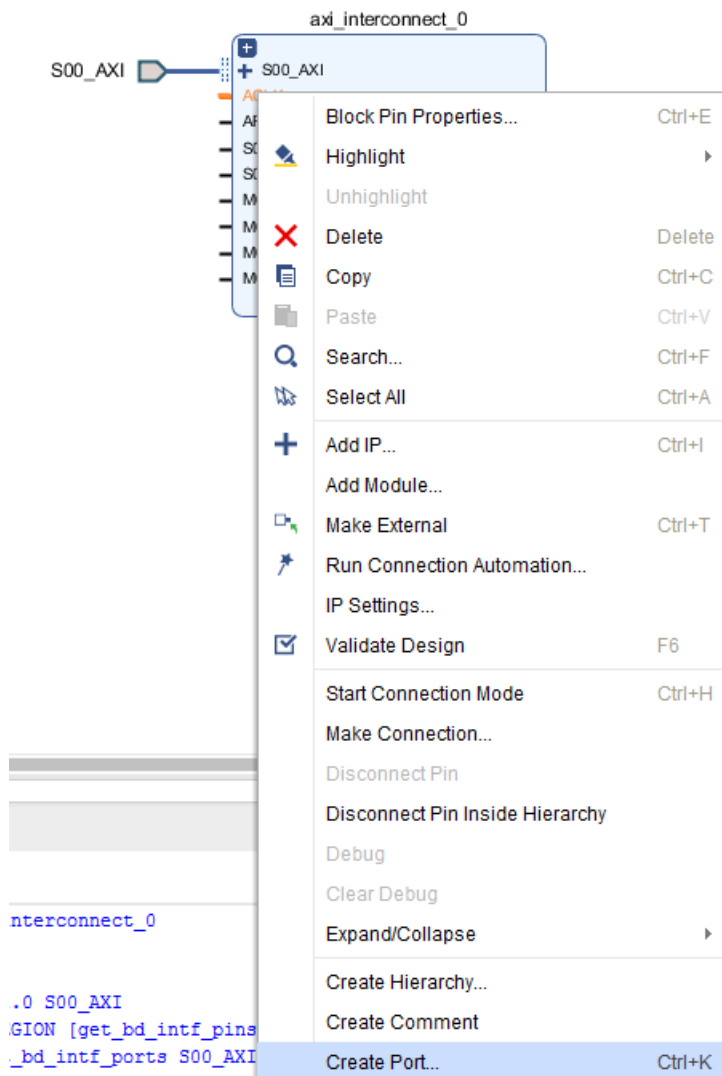


- Click **OK** to accept the default settings.

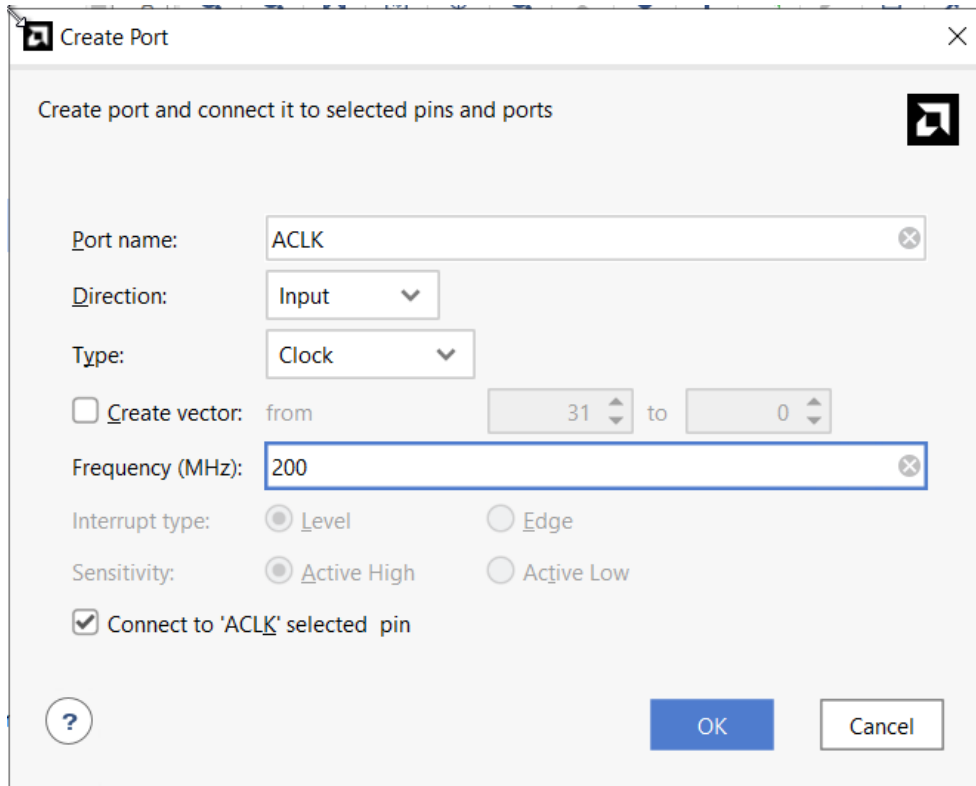
The Vivado IP integrator adds the external `S00_AXI` interface port to the subsystem design, and automatically connects it to the `S00_AXI` interface pin on the AXI Interconnect core.

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

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



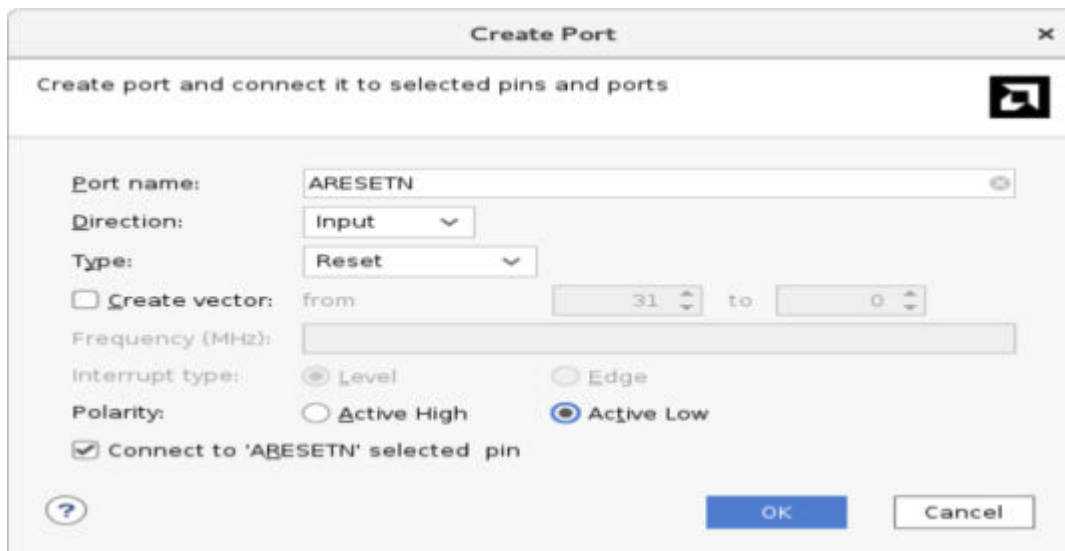
- In the Create Port dialog box, as shown in the following figure, for Frequency (MHz), enter 200, and leave the remaining fields set to the default values.
- Click **OK**.



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

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

- For Polarity, select **Active Low**.
- Click **OK**.



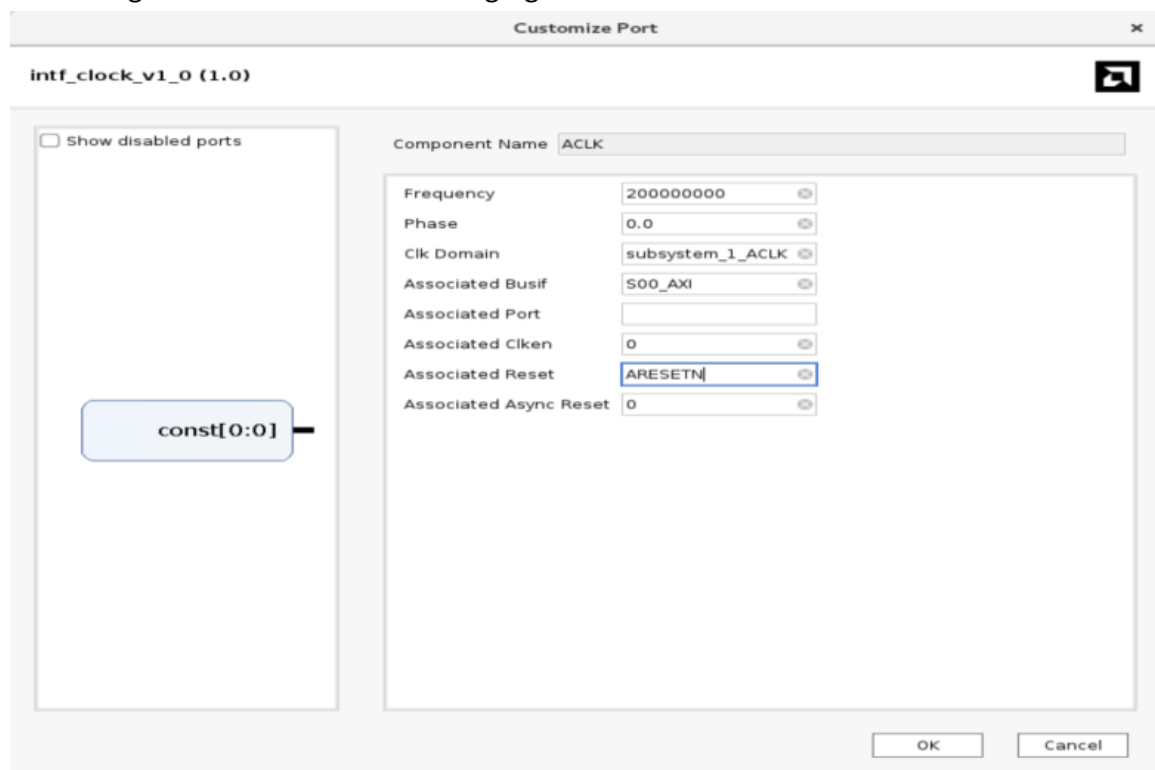


**IMPORTANT!** IP integrator treats an external reset coming into the block design as asynchronous to the clocks. You should always synchronize the external resets with a clock domain in the IP subsystem to help the design meet timing.

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 synchronized external to the block design, you can associate the reset signal with the clock on the external port. You do not need to use the Processor System Reset block in such cases.

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, you can associate this clock pin to the `S00_AXI` interface. In the Associated Busif field, type `S00_AXI`.
11. For the Associated Reset field, enter `ARESETN`.
12. Click **OK**.

The dialog box looks like the following figure:



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.

**Note:** The cursor changes into a pencil indicating that you can make a connection 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.

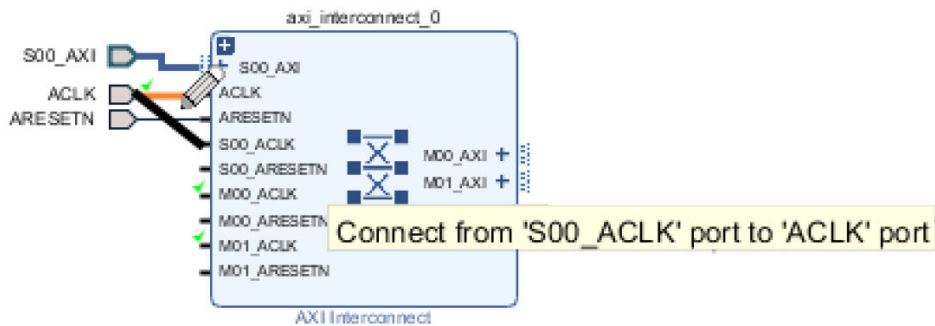




**TIP:** You must press and hold down the mouse button while dragging the connection from the `S00_ACLK` pin to the `ACLK` port.

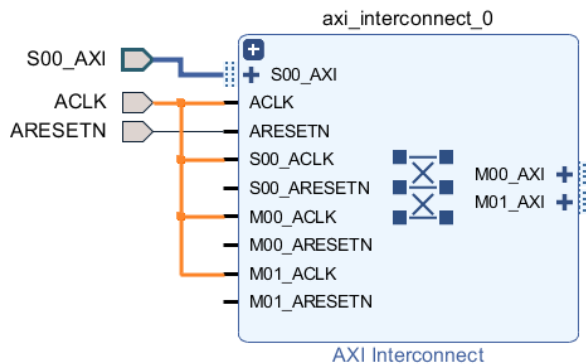
As you drag the connection wire, a green checkmark appears on the `ACLK` port indicating that you can make a valid connection between these points. The Vivado IP integrator highlights all possible connection points in the subsystem design as you interactively wire the pins and ports.

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:



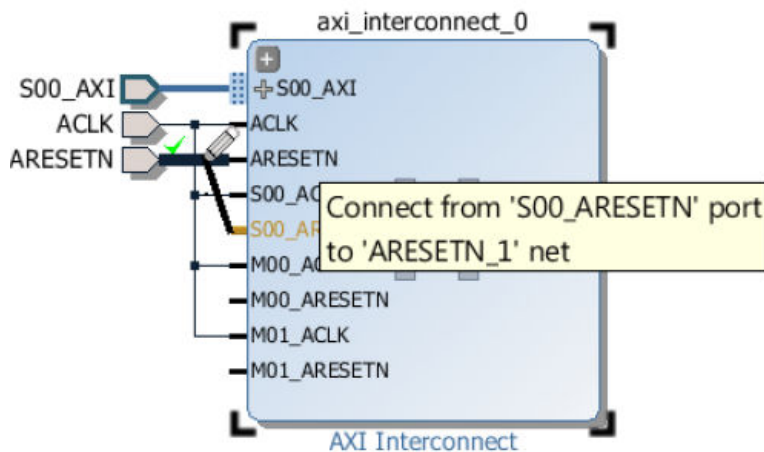
16. Repeating the steps outlined above, connect the `M00_ACLK` and the `M01_ACLK` to the `ACLK` port.

The connections to the AXI Interconnect should now appear as shown in the following figure:

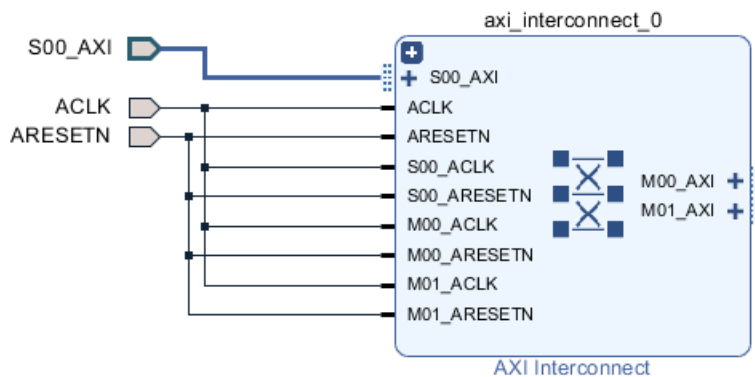


Similarly, connect the reset pins of all the masters and slaves to the `ARESETN` port.

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.

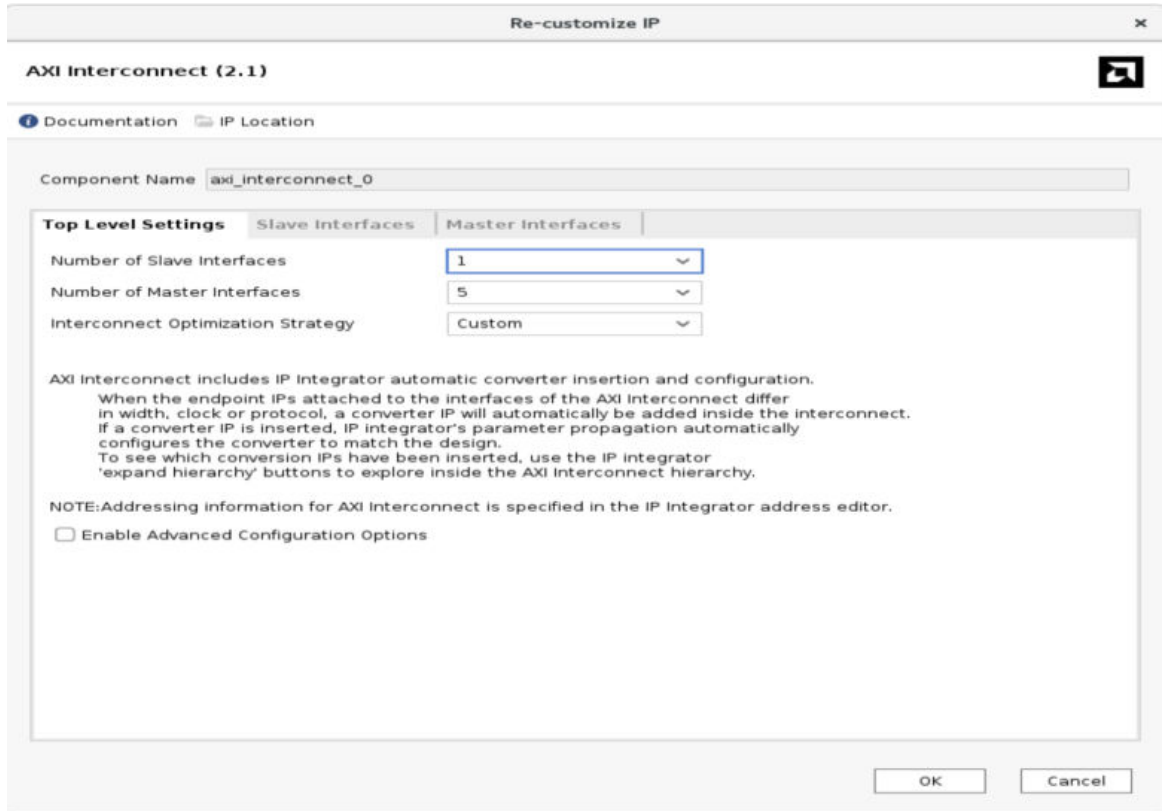


- Repeat the steps to connect the M00\_ARESETN and the M01\_ARESETN pins of the AXI Interconnect to the ARESETN port, as shown in the following figure:



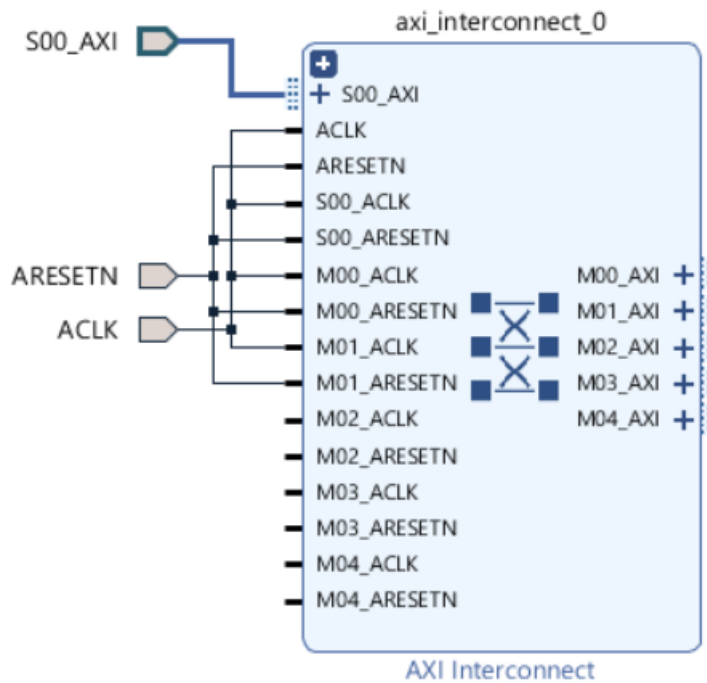
## Step 4: Customizing IP

- Double-click the AXI Interconnect core to open the Re-customize IP dialog box, as shown in the following figure:

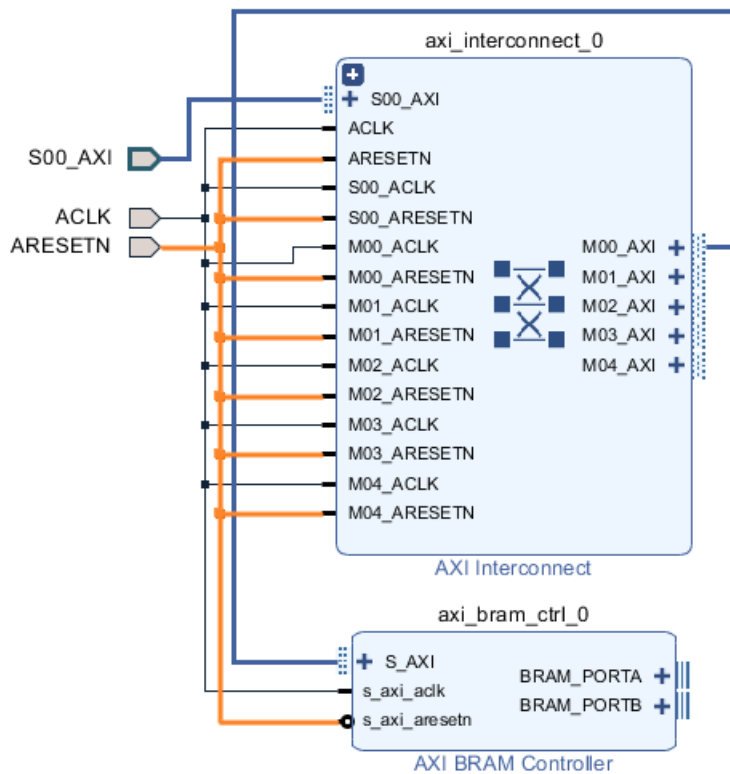


2. From the Top Level Settings Tab, for the Number of Master Interfaces field, select **5** from the drop-down menu.
3. Leave all the remaining options set to their default values and click **OK**.

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:



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. The following figure shows these connections.



- Using the same steps, connect the remaining slave IP cores in the design to the AXI Interconnect.

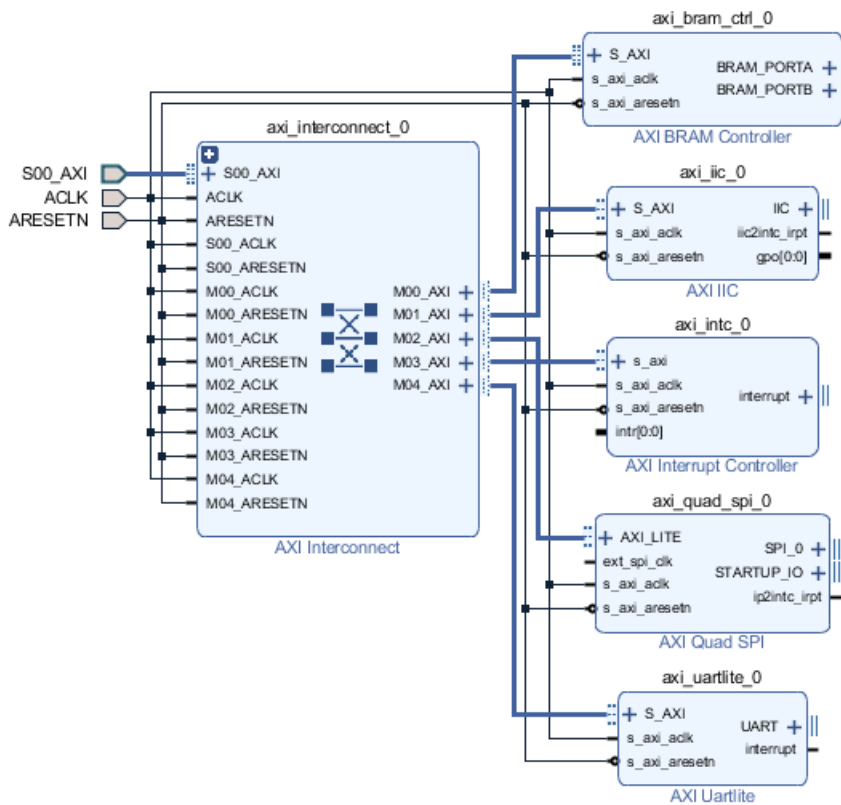


**TIP:** The order of connections between the *S\_AXI* interface pins on the IP slaves and the *M\_AXI* interface pins on the AXI Interconnect does not matter.

- Click **Regenerate Layout** on the menu at the top of the banner.



The IP integrator design canvas should look similar to what shows in the figure below.



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

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

## Step 5: Running Connection Automation

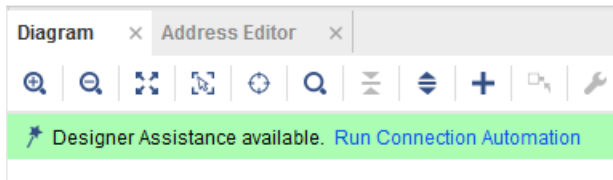
At this point, there are still some output interface pins that you must connect external to the subsystem design, such as the following:

- UART interface of the AXI UART Lite
- `SPI_0` interface of the AXI Quad SPI
- IIC interface of the AXI IIC

**Note:** The AXI Block RAM Controller is not connected to a Block Memory Generator.

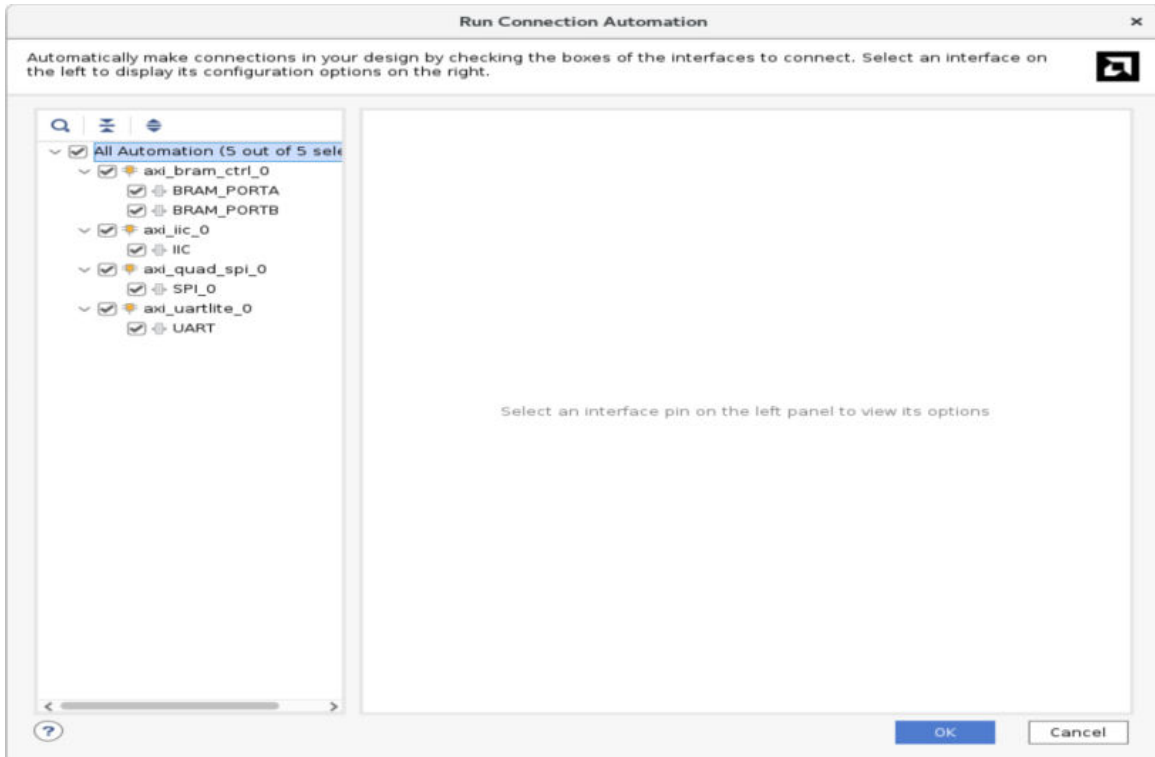
IP integrator offers the Designer Assistance feature to automate certain kinds of connections. For the current subsystem design, you can connect the UART, SPI, and IIC interfaces to external ports using connection automation. You can also use the Designer Assistance feature to connect a Block Memory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

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

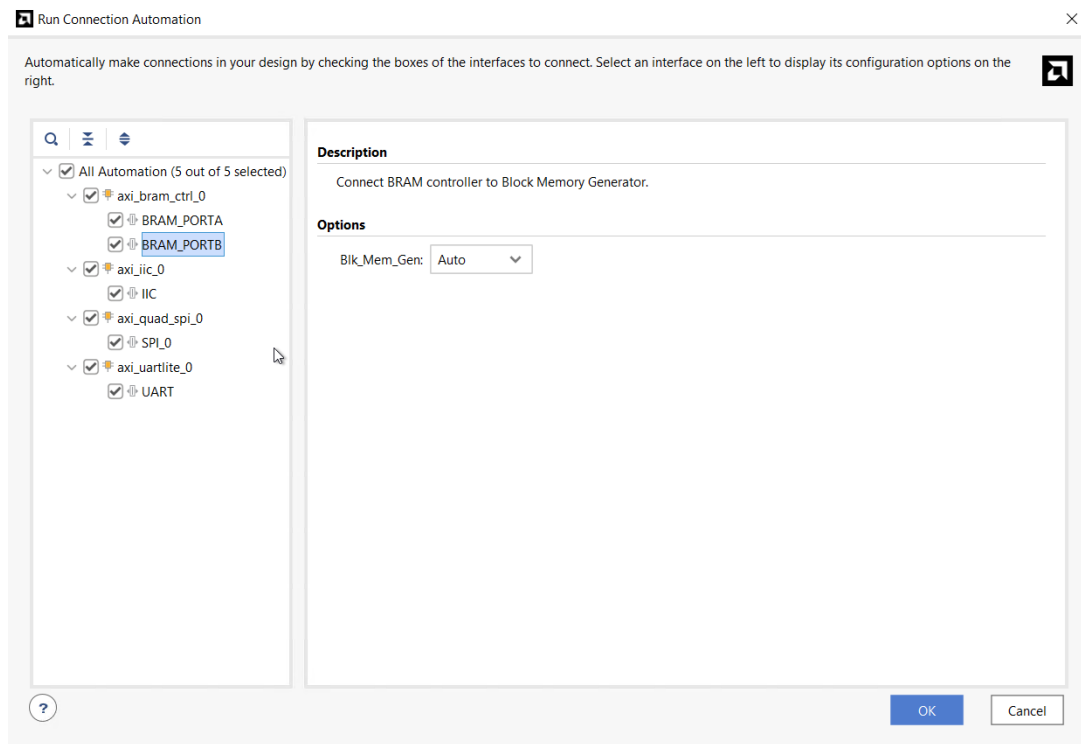


The Run Connection Automation dialog box opens.

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



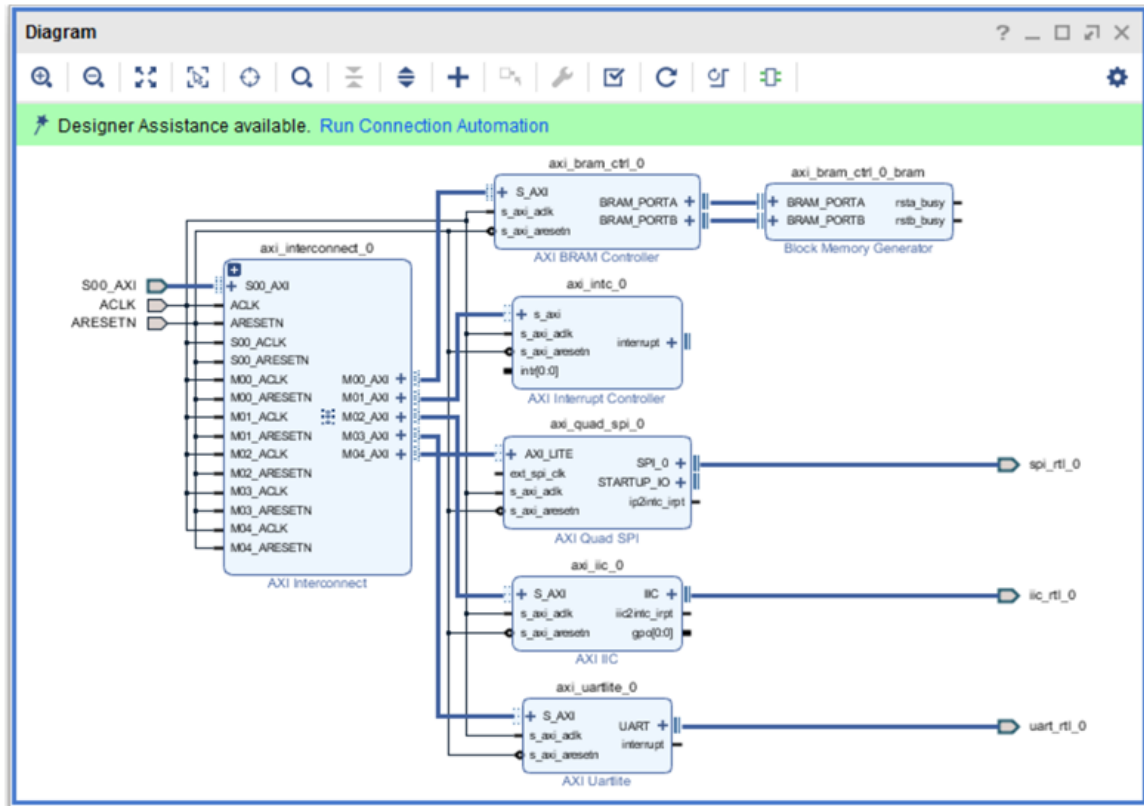
3. Select and highlight the interfaces, as shown in the following figure, to see a description of the automation that the tool offers, also any options needed to connect these interfaces.



4. Click **OK**.



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



6. Right-click the newly added `spi_rtl_0` port to open the popup 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 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:

Create port and connect it to selected pins and ports

Port name:

Direction:

Type:


☐ Create vector: from  to

Frequency (MHz):

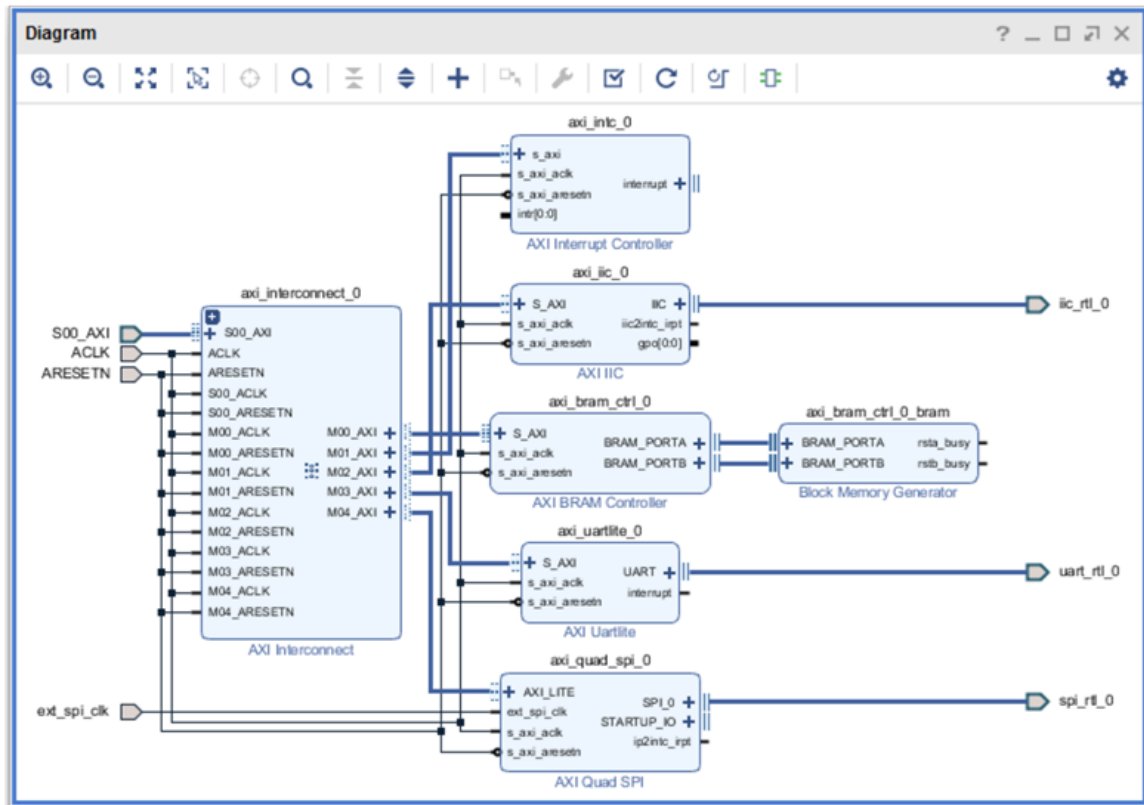
Interrupt type: ☒ Level ☐ Edge

Sensitivity: ☒ Active High ☐ Active Low

☒ Connect to 'ext\_spi\_clk' selected pin

8. For the Frequency (MHz) field, enter 100, if it is not already set, and click **OK**.
9. Click **Regenerate Layout** button  to redraw the subsystem design.

The optimized layout of the design should now look similar to the figure below:

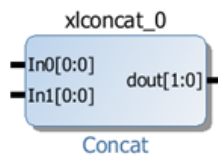


## Step 6: Managing Signals with CONCAT and CONSTANT Blocks

Now you connect the interrupt signals on the various IP slaves to an interrupt controller through an AMD 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.

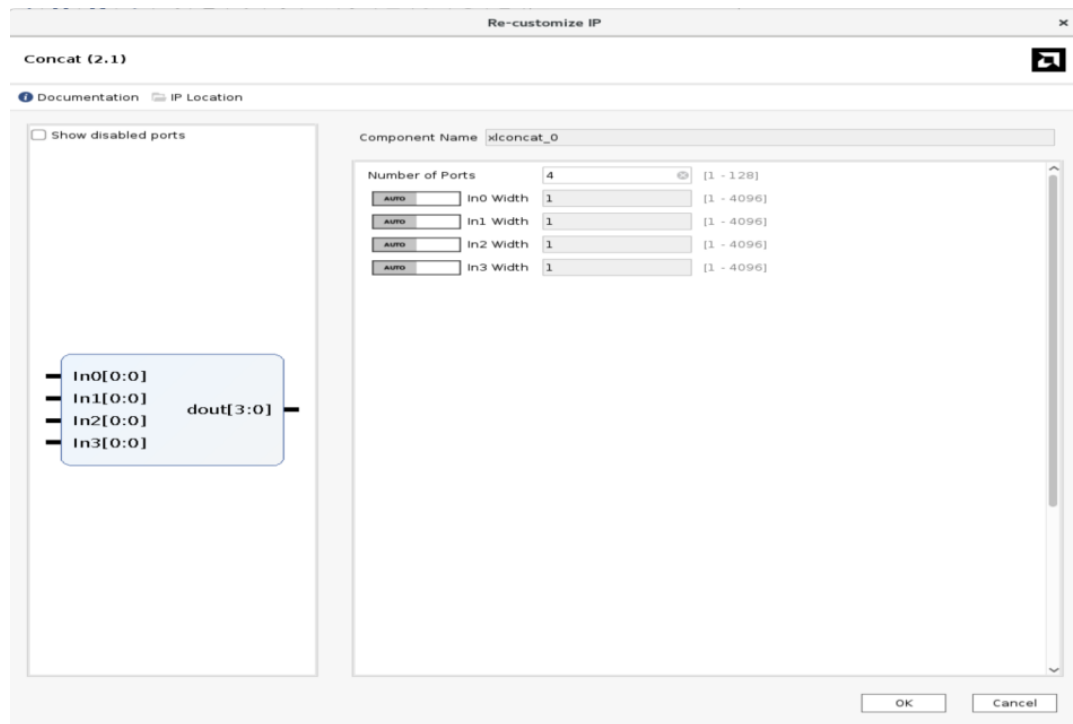
1. Right-click the design canvas to open the popup menu and select **Add IP**.
2. In the search field, type `concat` to find the AMD Concat block, and double-click the core.

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



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:



- For the Number of Ports field, enter 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 use the fourth port to demonstrate tying a signal high or low with the Constant block.

- 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.

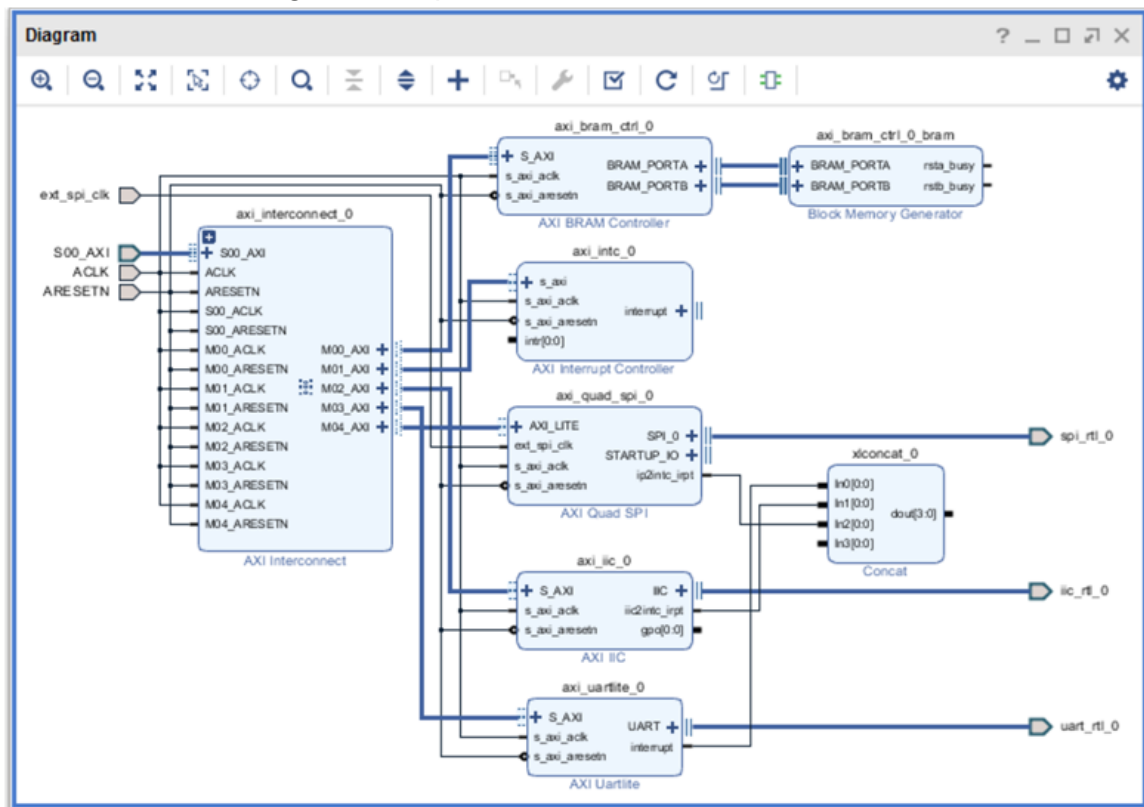
- Place the cursor on top of the `interrupt` pin of the AXI UART Lite.

**Note:** The cursor changes into a pencil indicating that a connection can be made from that pin.

- Click and drag the cursor from the `interrupt` pin to an `input` pin on the Concat block.

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.

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



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, you 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 AMD Constant block, and double-click the core in the IP catalog.

The AMD 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.

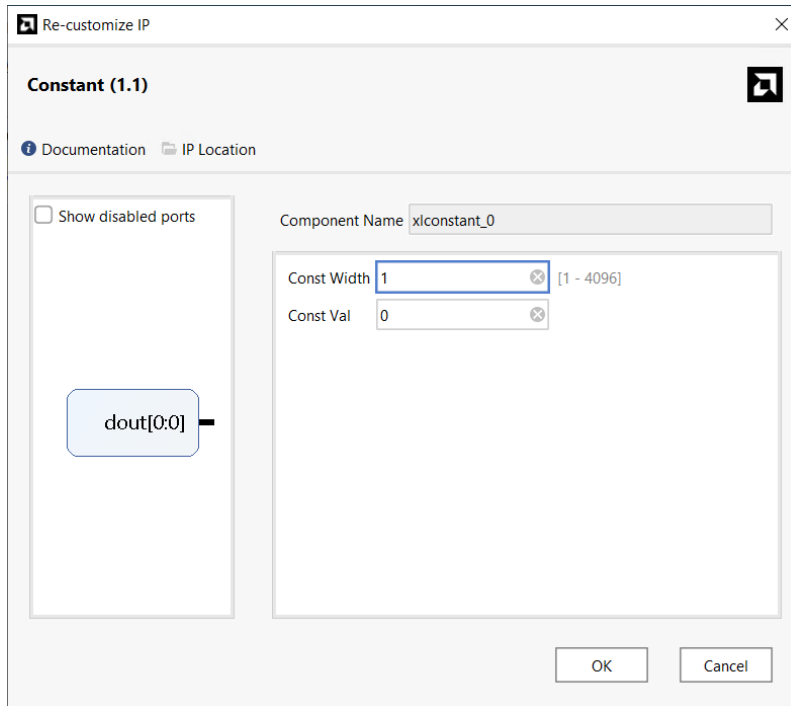
14. Release the mouse button and Vivado IP integrator makes a connection between the pins.

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.

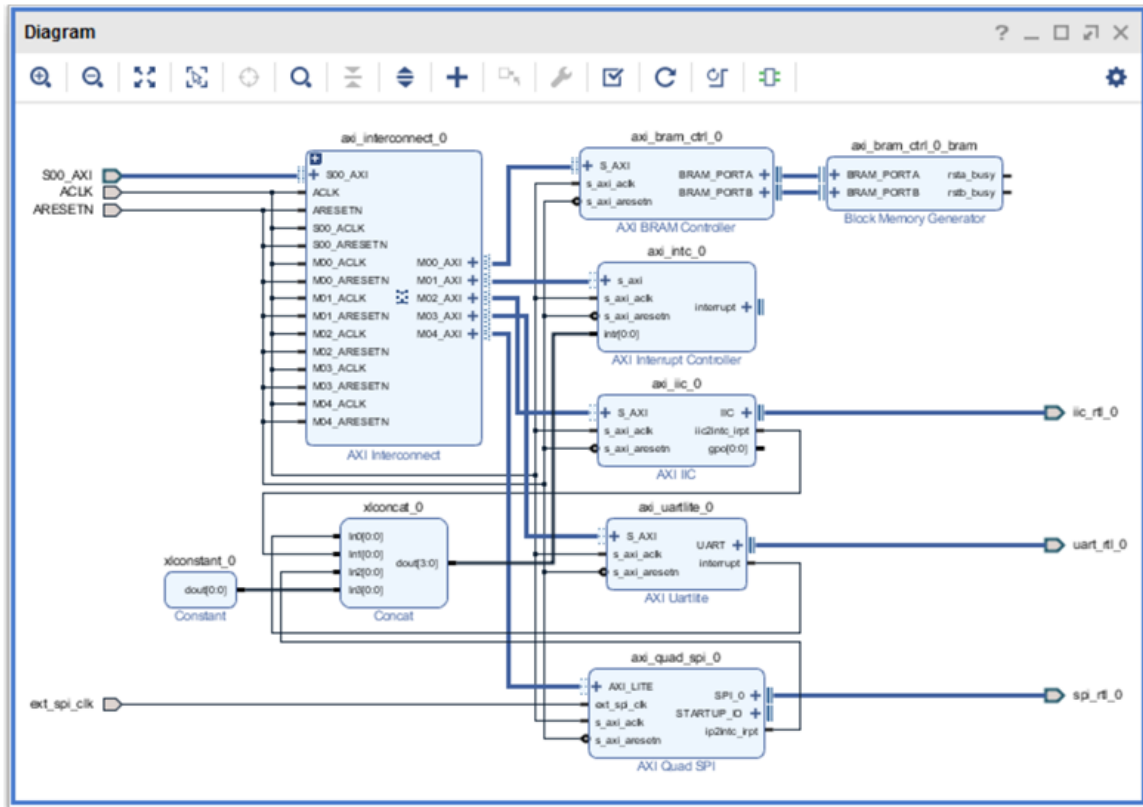
16. For the Const Val field, enter `0`, as shown in the following figure, and click **OK**.



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:



**Note:** 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 has only one address segment, representing a range of addresses. However, some slaves, like a bridge, 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 select **Window → Address Editor** from the main menu.

Address Editor						
<input type="checkbox"/> Assigned (0) <input checked="" type="checkbox"/> Unassigned (5) <input checked="" type="checkbox"/> Excluded (0) <input type="button" value="Hide All"/>						
Name	Interface	Slave Segment	Master Base Address	Range	Master High Address	
Network 0						
External Masters (1)						
/S00_AXI (32 address bits : 0x00000000 [ 4G ])						
Unassigned (5)						
/axi_bram_ctrl_0	S_AXI	Mem0				
/axi_iic_0	S_AXI	Reg				
/axi_intc_0	s_axi	Reg				
/axi_quad_spi_0	AXI_LITE	Reg				
/axi_uartlite_0	S_AXI	Reg				



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

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 **Assign All** button on the Address Editor toolbar to automatically assign the addresses.

The Auto Assign Address dialog box is displayed.

- Click **OK**.

The Address Editor should now look like the following figure:

The screenshot shows the Address Editor window with the following data:

Name	Interface	Slave Segment	Master Base Address	Range	Master High Address
Network 0					
External Masters (1)					
/S00_AXI (32 address bits : 0x00000000 [ 4G ])					
/axi_bram_ctrl_0	S_AXI	Mem0	0xC000_0000	8K	0xC000_1FFF
/axi_iic_0	S_AXI	Reg	0x4080_0000	64	0x4080_FFFF
/axi_intc_0	s_axi	Reg	0x4120_0000	64	0x4120_FFFF
/axi_quad_spi_0	AXI_LITE	Reg	0x44A0_0000	64	0x44A0_FFFF
/axi_uartlite_0	S_AXI	Reg	0x4060_0000	64	0x4060_FFFF

- Change the size of the address segments for the AXI BRAM Controller core.


Click the **Range** column, and select **64K** from the drop-down menu, shown in the following figure:

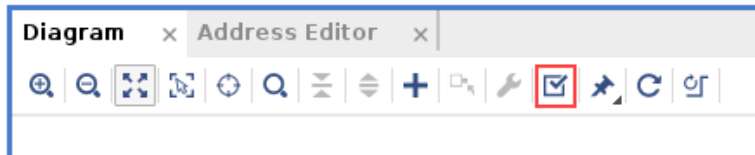
The screenshot shows the Address Editor window with the Range column dropdown menu open for the /axi\_bram\_ctrl\_0 entry. The menu options are:

- 8K
- 16K
- 32K
- 64K (selected)
- 128K
- 256K
- 512K
- 1M

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

## Step 8: Validating the Design

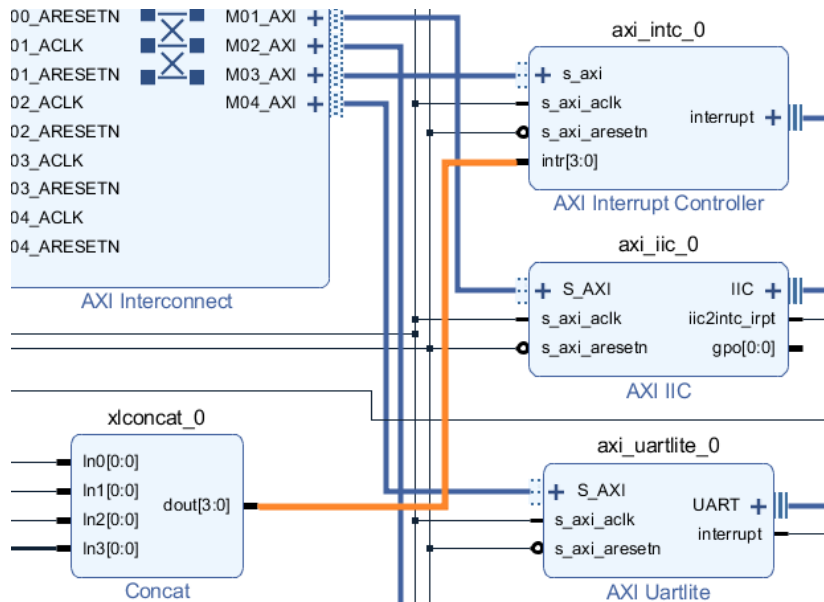
- From the menu at the top of the IP integrator design canvas, run the IP subsystem design rule checks (DRCs) by clicking the **Validate Design** button .



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

- Click **OK**.
- 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.



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

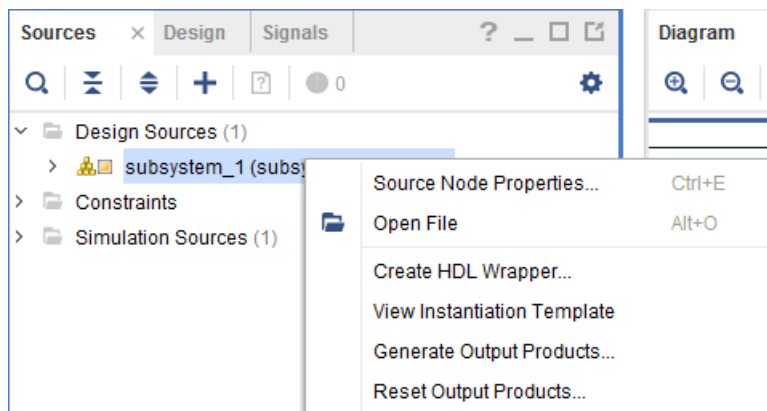
- Select **File → Save Block Design** command from the main menu to save the design.

## 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 can 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**.

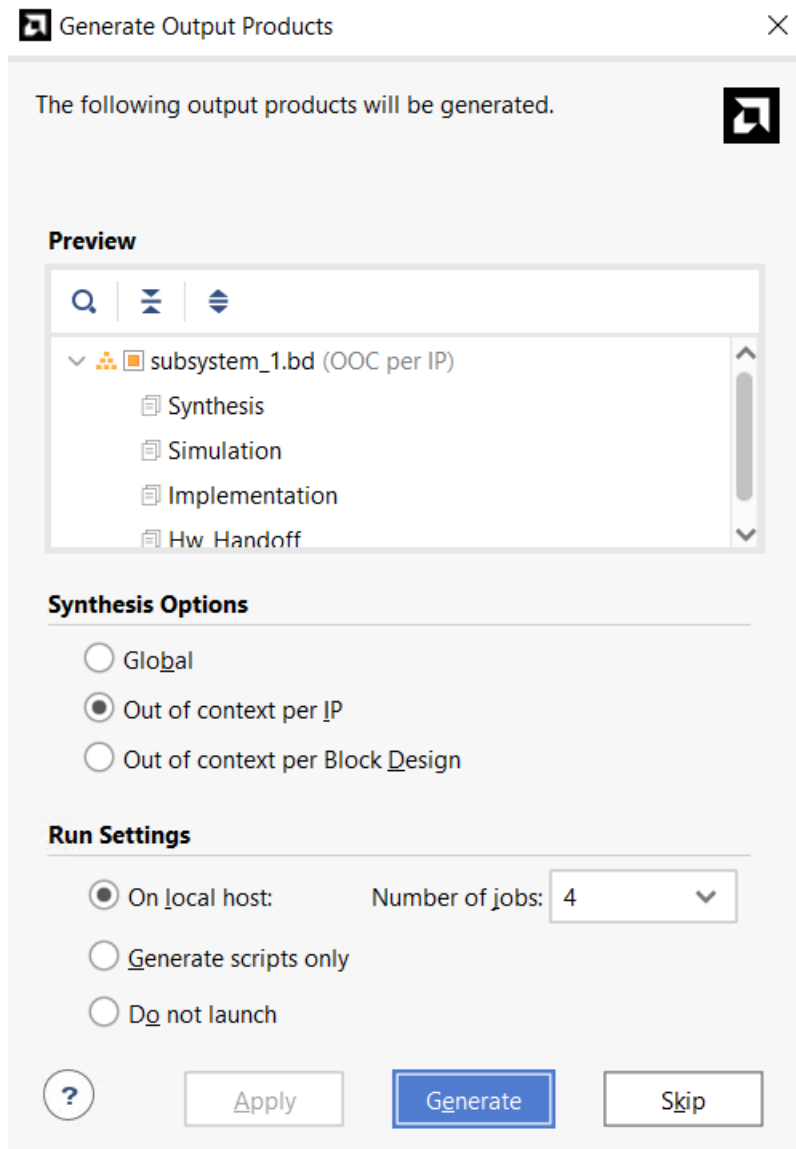
This command generates the source files for the IP cores used in the subsystem design, and any related constraints file.



The Generate Output Product dialog box opens, as shown in the figure below, 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.

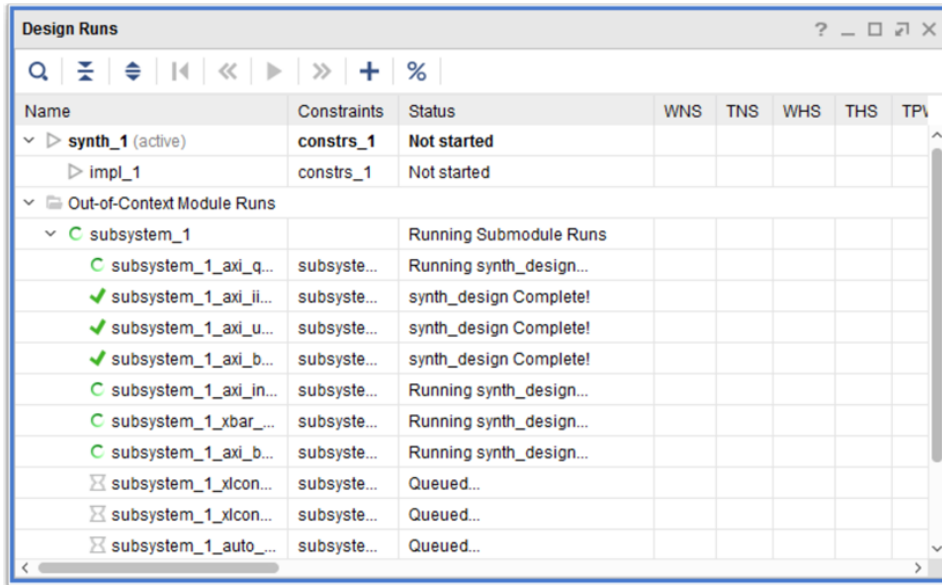


2. Leave the default selection of Out of context per IP.
3. Click **Generate** to generate all output products.

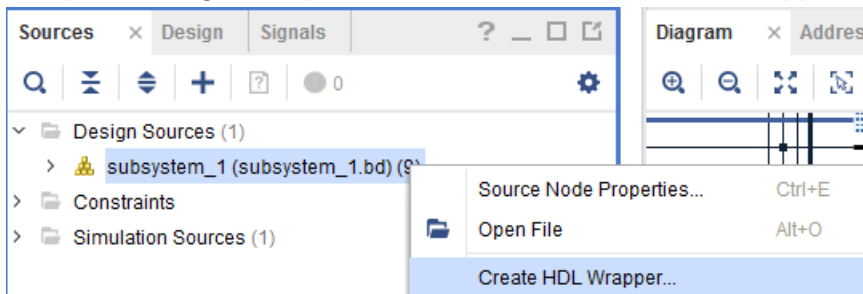
Alternatively, you can click **Generate Block Design** in the Flow Navigator, under the IP integrator drop-down menu.

The Generate Output Products dialog box opens to confirm the output products are generated.

4. Click **OK**.
5. The Out-of-context (OOO) runs for each IP in the design launch, shown in the Design Runs window below. OOO runs can take a few minutes to finish.



- 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**.



The Create HDL Wrapper dialog box opens, and offers two choices:

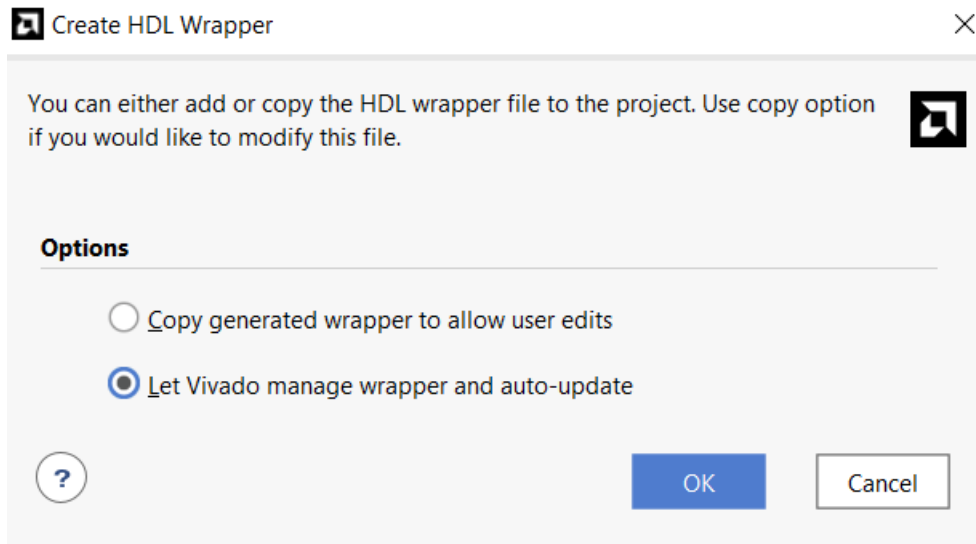
- **Copy generated wrapper to allow user edits:**

Choose this option if you modify the wrapper file. Often 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.

- Select the default option, **Let Vivado manage wrapper and auto-update**, as shown in the following figure:



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.

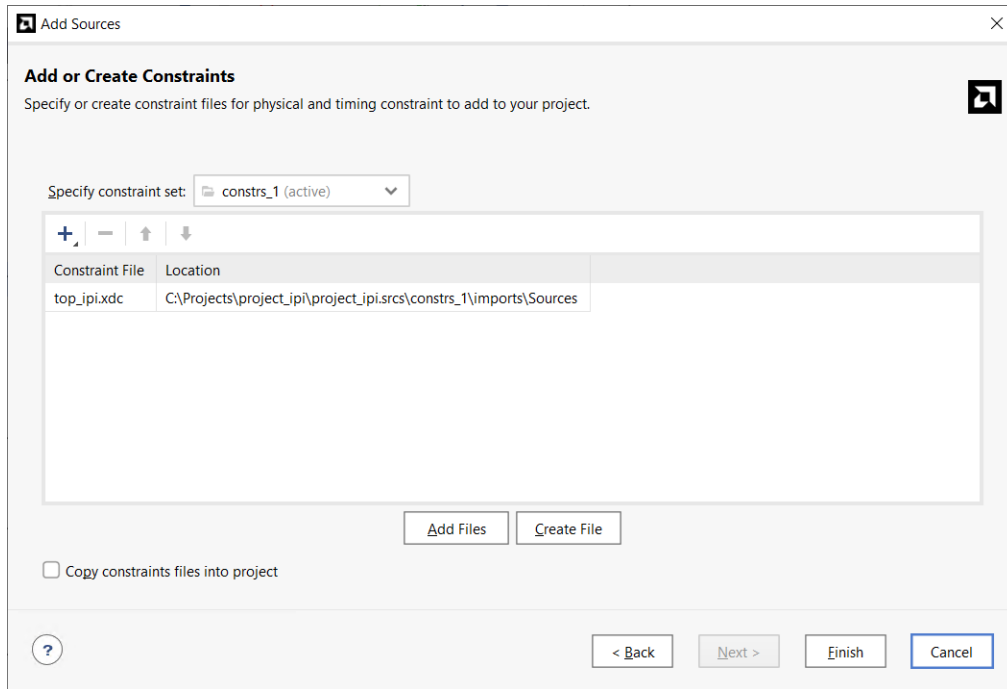
With the top-level HDL source added to the project, you must now add design constraints to the project prior to implementation.

9. From the Flow Navigator, click **Add Sources**.

The Add Sources wizard opens.

10. Select the **Add or Create Constraints** option and click **Next**.

11. In the Add or Create Constraints page, click  , and select **Add Files**, or click the **Add Files** button.



The Add Constraints Files dialog box opens.

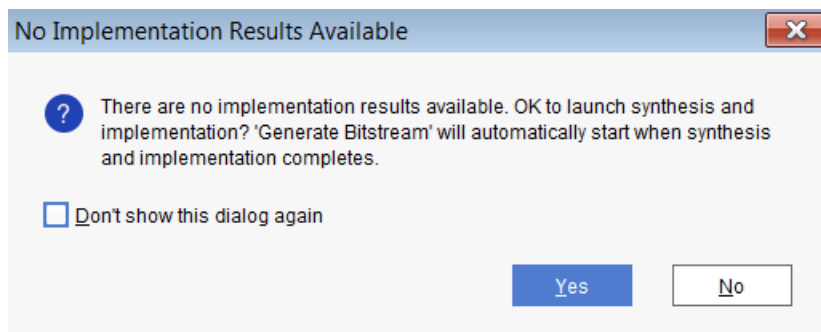
12. Select **top\_ipi.xdc** file in <Extract\_Dir>, and click **OK**.
13. In the Add or Create Constraints page, make sure that Copy constraints files into project is selected.
14. Click **Finish** to add the constraints to the project.

You are now ready to synthesize, implement, and generate the bitstream for the top-level design.

15. In the Flow Navigator, click **Generate Bitstream**.

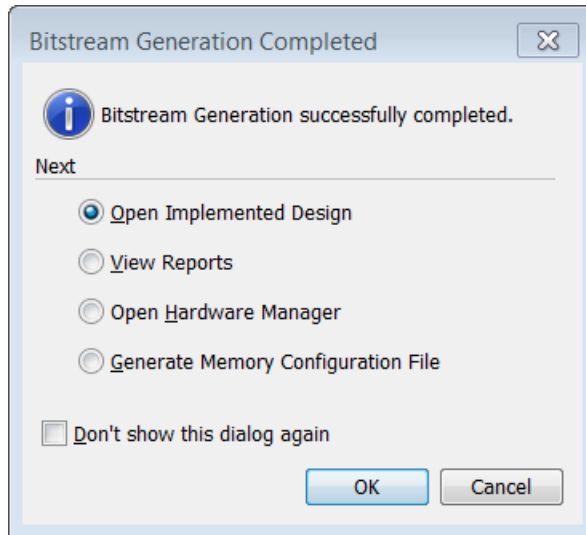
With a single click, this completes all the steps needed to synthesize, implement, and generate the bitstream for the design.

The No Implementation Results Available dialog box opens as seen in the following figure:



16. Click **Yes**.

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



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

Design Timing Summary		
Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 0.979 ns	Worst Hold Slack (WHS): 0.072 ns	Worst Pulse Width Slack (WPWS): 1.732 ns
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS): 0.000 ns
Number of Failing Endpoints: 0	Number of Failing Endpoints: 0	Number of Failing Endpoints: 0
Total Number of Endpoints: 6844	Total Number of Endpoints: 6818	Total Number of Endpoints: 3289
All user specified timing constraints are met.		



**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_output_delay` commands. See 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 take 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.



# Additional Resources and Legal Notices

---

## Finding Additional Documentation

### Technical Information Portal

The AMD Technical Information Portal is an online tool that provides robust search and navigation for documentation using your web browser. To access the Technical Information Portal, go to <https://docs.amd.com>.

### Documentation Navigator

Documentation Navigator (DocNav) is an installed tool that provides access to AMD Adaptive Computing documents, videos, and support resources, which you can filter and search to find information. To open DocNav:

- From the AMD Vivado™ IDE, select **Help** → **Documentation and Tutorials**.
- On Windows, click the **Start** button and select **Xilinx Design Tools** → **DocNav**.
- At the Linux command prompt, enter `docnav`.

**Note:** For more information on DocNav, refer to the *Documentation Navigator User Guide* ([UG968](#)).

### Design Hubs

AMD Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In DocNav, click the **Design Hubs View** tab.
- Go to the [Design Hubs](#) web page.

---

## Support Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see [Support](#).

---

## References

These documents provide supplemental material useful with this guide:

1. *Vivado Design Suite User Guide: Release Notes, Installation, and Licensing* ([UG973](#))
  2. *Vivado Design Suite User Guide: Designing IP Subsystems Using IP Integrator* ([UG994](#))
  3. *Vivado Design Suite User Guide: Using Constraints* ([UG903](#))
- 

## Revision History

The following table shows the revision history for this document.

Section	Revision Summary
11/13/2024 Version 2024.2	
<a href="#">Step 3: Creating External Connections</a>	Updated the section
N/A	Updated reference design file
05/30/2024 Version 2024.1	
N/A	Updated reference design file

---

## Please Read: Important Legal Notices

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions, and typographical errors. The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. Any computer system has risks of security vulnerabilities that cannot be completely prevented or mitigated. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes. THIS INFORMATION IS PROVIDED "AS

IS." AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS, OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION. AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY RELIANCE, DIRECT, INDIRECT, SPECIAL, OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

#### **AUTOMOTIVE APPLICATIONS DISCLAIMER**

AUTOMOTIVE PRODUCTS (IDENTIFIED AS "XA" IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE ("SAFETY APPLICATION") UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD ("SAFETY DESIGN"). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.

#### **Copyright**

© Copyright 2013-2024 Advanced Micro Devices, Inc. AMD, the AMD Arrow logo, Artix, Kintex, Versal, Vivado, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.