

Vivado Design Suite Tutorial

Designing IP Subsystems Using IP Integrator

UG995 (v 2013.2) June 20, 2013





Notice of Disclaimer

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of the Limited Warranties which can be viewed at <http://www.xilinx.com/warranty.htm>; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in Critical Applications: <http://www.xilinx.com/warranty.htm#critapps>.

©Copyright 2013 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, Vivado, ISE, Kintex, Spartan, 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
06/20/2013	2013.2	Minor editorial update.
06/19/2013	2013.2	New in this release.

Table of Contents

Revision History.....	2
Table of Contents	3
Designing IP Subsystems	4
Introduction	4
Tutorial Design Description	4
Software Requirements.....	4
Hardware Requirements	5
Locating Tutorial Design Files.....	5
Lab 1: Designing IP Subsystems in IP Integrator	6
Step 1: Creating a Project.....	6
Step 3: Creating External Connections	13
Step 4: Customize IP	18
Step 5: Running Connection Automation	23
Step 6: Using the Address Editor	26
Step 7: Creating and Implementing the Top-Level Design	28
Conclusion.....	33

Designing IP Subsystems

Introduction

The Xilinx[®] Vivado[®] Design Suite IP integrator feature 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 on this tutorial, you will be introduced to the IP Integrator GUI, run design rule checks (DRC) on your design, and then integrate the design in a top-level design in the Vivado Design Suite. Finally, you will run synthesis and implementation and generate bitstream on the 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 a 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.

Software Requirements

This tutorial requires that the 2013.2 Vivado Design Suite software release or later is installed. For installation instructions and information, see the *Vivado Design Suite User Guide: Release Notes, Installation, and Licensing* ([UG973](#)).

Hardware Requirements

The supported Operating Systems include Redhat 5.6 Linux 64 and 32 bit, and Windows 7, 64 and 32 bit.

Xilinx recommends a minimum of 2 GB of RAM when using the Vivado tool.

Locating Tutorial Design Files

This tutorial requires the standard Vivado Design Suite tutorial files, with the addition of a new constraints file, `top_ipi.xdc`, which is available to download for use with the tutorial.

You can find the standard Vivado Design Suite tutorial files in the `examples` folder of the Vivado tools installation, at the following location:

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

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

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

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

2. **Download** the `ug995-tutorial.zip` file from the Xilinx website:

<http://www.xilinx.com/cgi-bin/docs/rdoc?v=2013.2;t=vivado+tutorials>

3. **Extract** the `ug995-tutorial.zip` file into *<Extract_Dir>/Vivado_Tutorial/Sources* folder.

The new `top_ipi.xdc` file is added to the `Sources` folder. You are now ready to begin the labs.

Lab 1: Designing IP Subsystems in IP Integrator

Step 1: Creating a Project

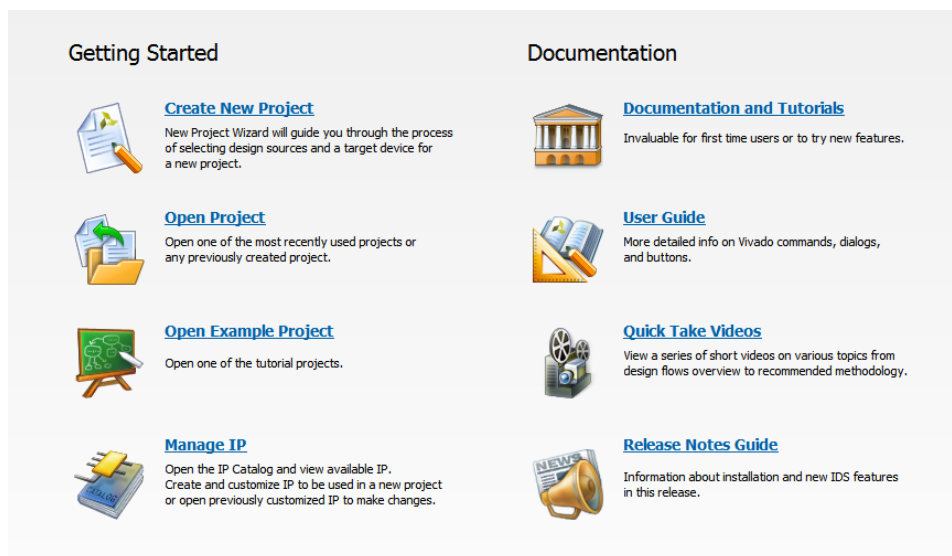


Figure 1: Vivado IDE Getting Started Page

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 2013.2 > Vivado 2013.2¹
 2. As an alternative, click the **Vivado 2013.2** Desktop icon to start the Vivado IDE.

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

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 Getting Started 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:

`<Extract_Dir>/Vivado_Tutorial`

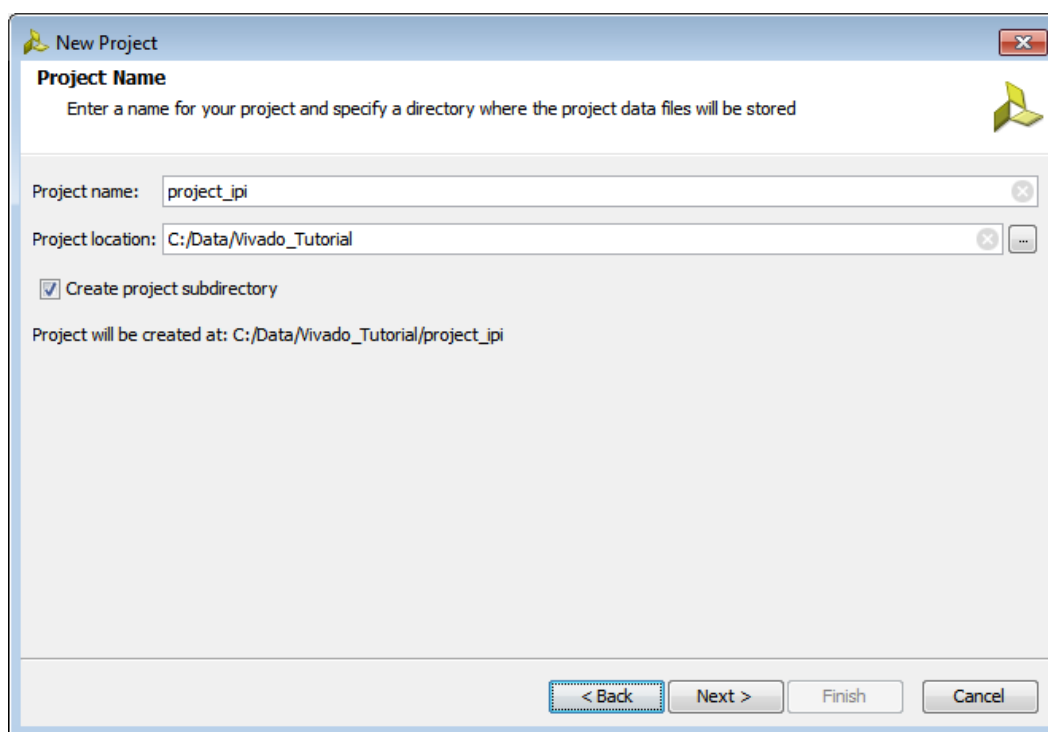


Figure 2: Create Project

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

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

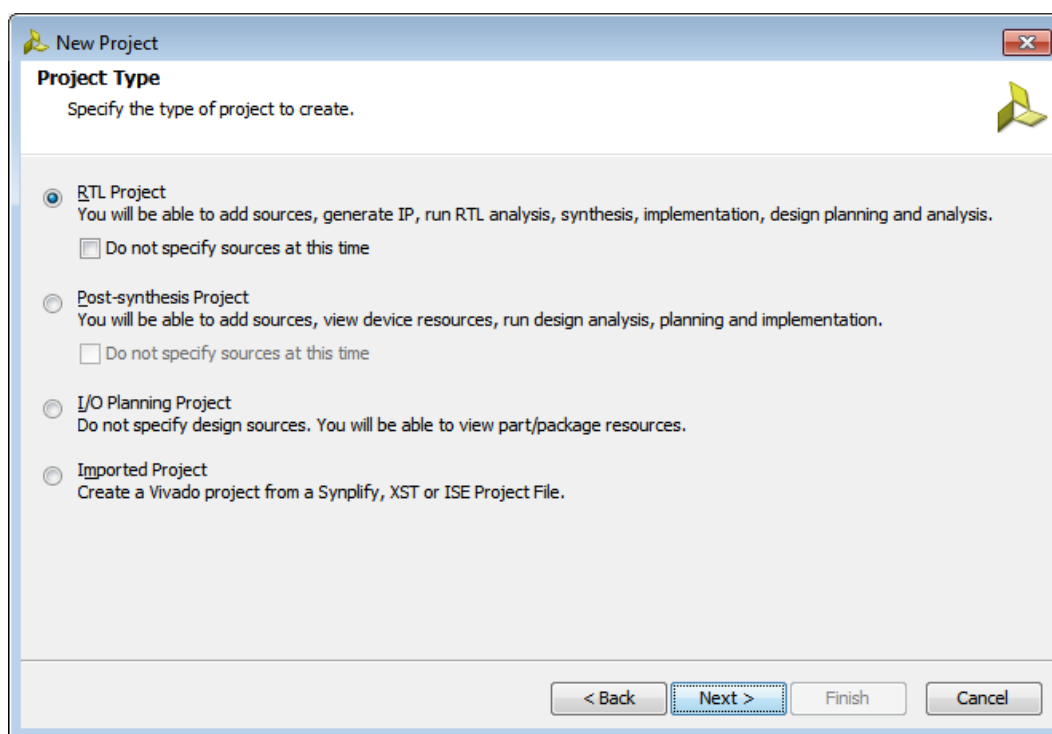


Figure 3: Specify Project Type

8. In the **Add Sources** dialog box, ensure that the **Target language** is set to **VHDL**, and 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

12. **Choose** the **xc7k325tffg900-2** part from the listed parts, and click **Next**.

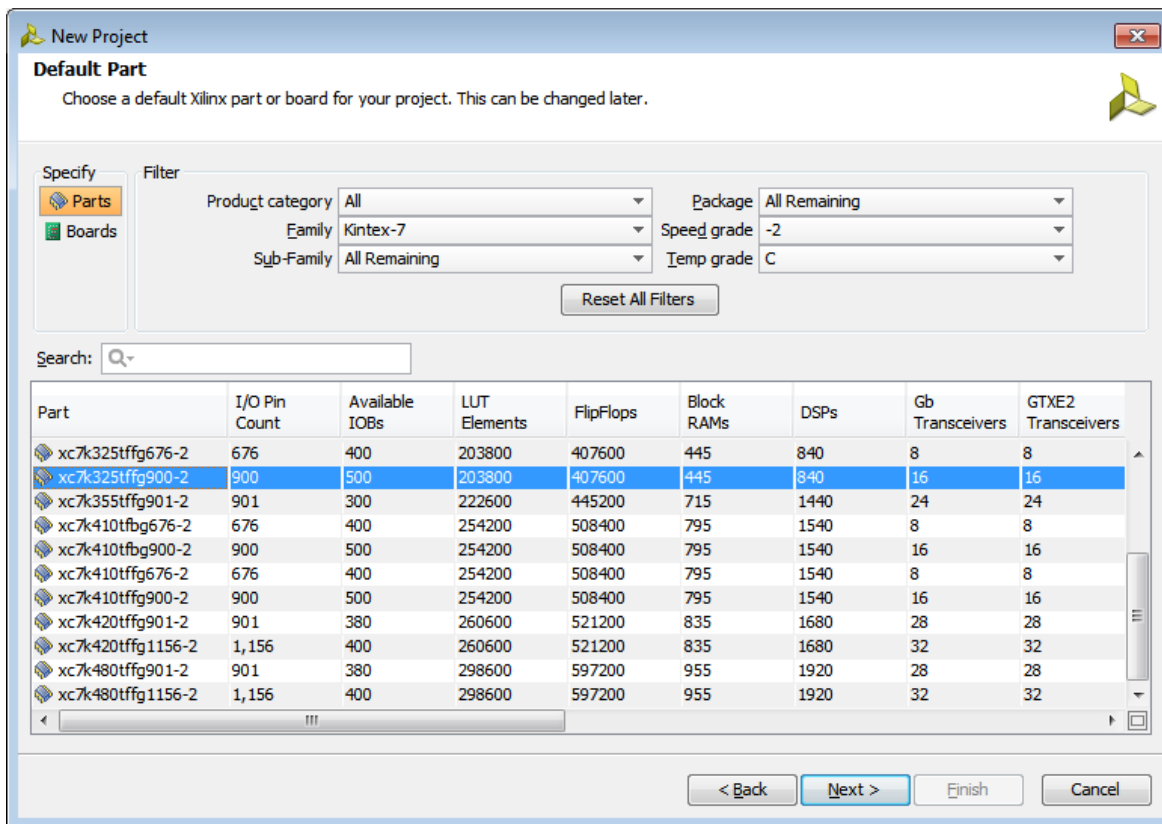


Figure 4: Select Default Part

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.
2. On the **Create Block Design** dialog box, specify **subsystem_1** as the Design Name.

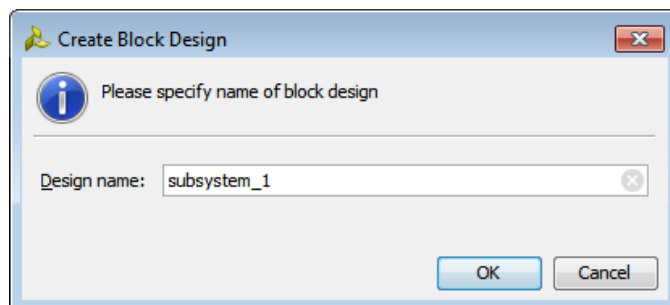


Figure 5: Create Block Design dialog box

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

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

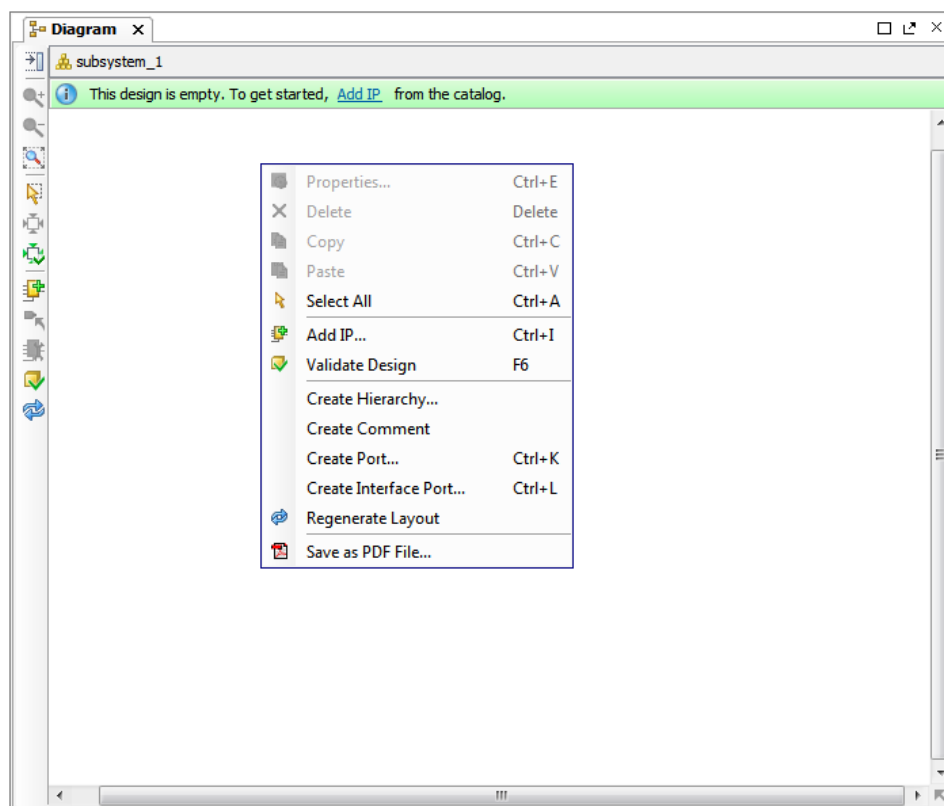

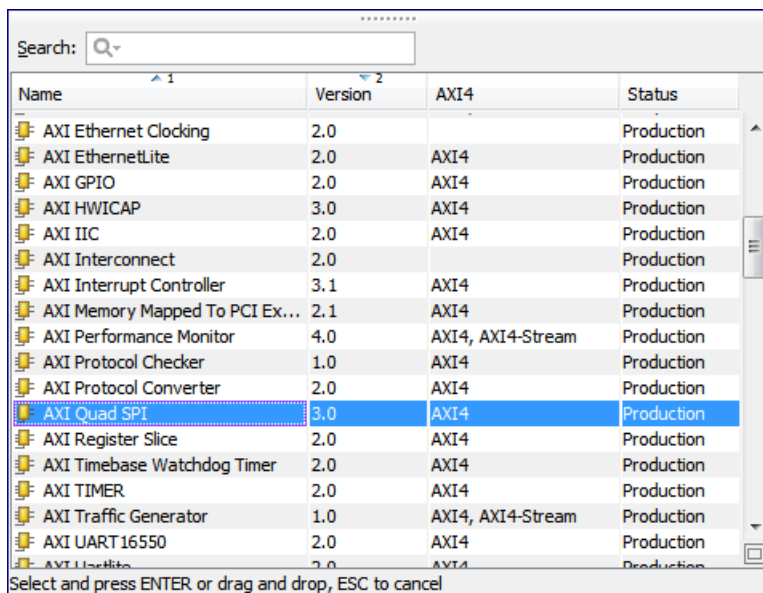


Figure 6: Add IP in Design Canvas

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.



Name	Version	AXI4	Status
AXI Ethernet Clocking	2.0		Production
AXI EthernetLite	2.0	AXI4	Production
AXI GPIO	2.0	AXI4	Production
AXI HWICAP	3.0	AXI4	Production
AXI IIC	2.0	AXI4	Production
AXI Interconnect	2.0		Production
AXI Interrupt Controller	3.1	AXI4	Production
AXI Memory Mapped To PCI Ex...	2.1	AXI4	Production
AXI Performance Monitor	4.0	AXI4, AXI4-Stream	Production
AXI Protocol Checker	1.0	AXI4	Production
AXI Protocol Converter	2.0	AXI4	Production
AXI Quad SPI	3.0	AXI4	Production
AXI Register Slice	2.0	AXI4	Production
AXI Timebase Watchdog Timer	2.0	AXI4	Production
AXI TIMER	2.0	AXI4	Production
AXI Traffic Generator	1.0	AXI4, AXI4-Stream	Production
AXI UART16550	2.0	AXI4	Production
AXI4-Lite	2.0	AXI4	Production

Select and press ENTER or drag and drop, ESC to cancel

Figure 7: Instantiate AXI Quad SPI IP



TIP: If your 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.

- In the search field, type **spi** to find the AXI Quad SPI.
- 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 into the IP integrator design canvas.

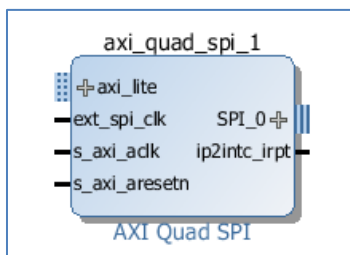


Figure 8: An Instantiated AXI Quad SPI

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

9. Use the Add IP command to instantiate the:

- AXI Uartlite,
- AXI BRAM Controller,
- Block Memory Generator
- AXI Interconnect.

At this point, the IP integrator canvas should look as shown in Figure 9.

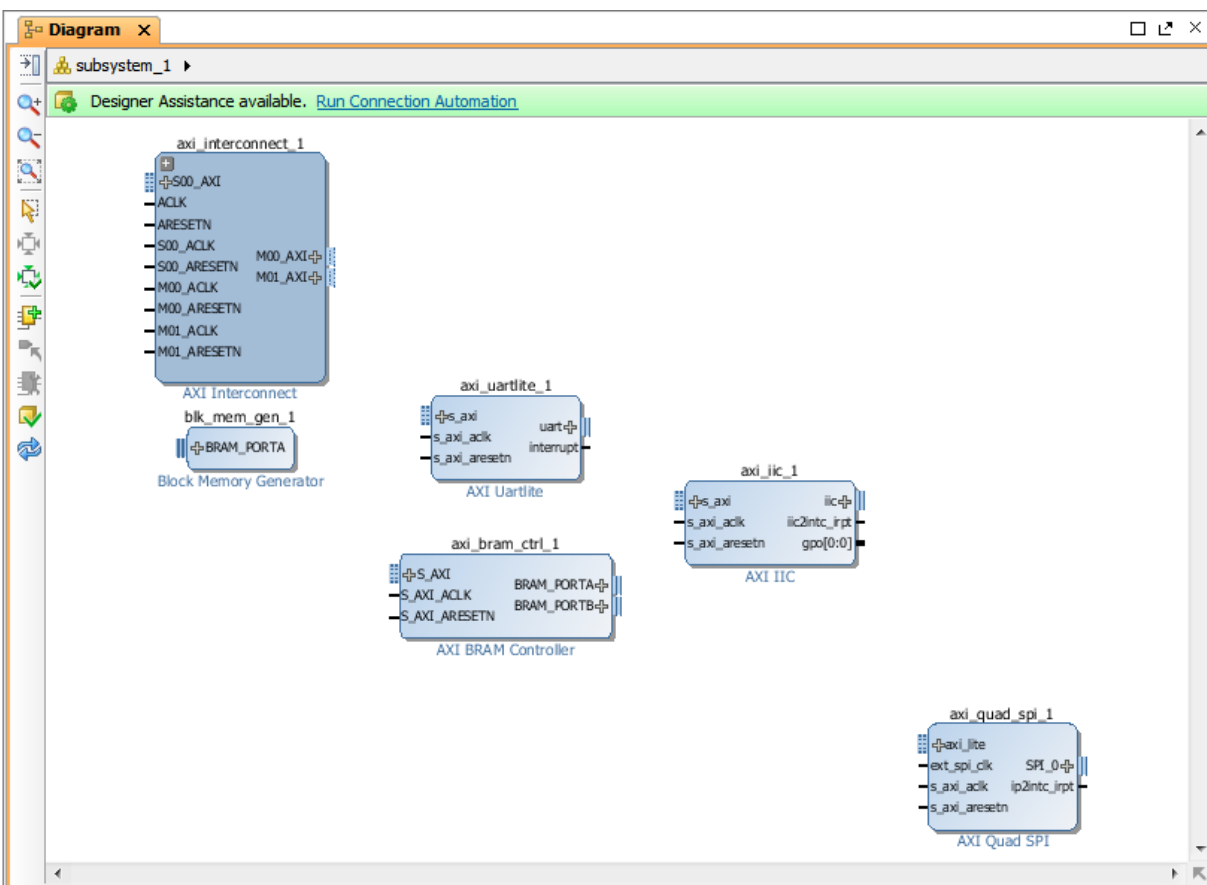


Figure 9: IP Integrator Design Canvas

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 indicate that there is 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 to open the popup menu, and select **Create Interface Port**.

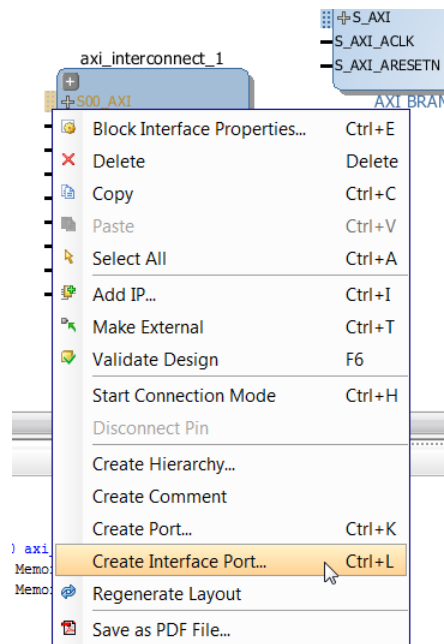


Figure 10: Right Click on S00_AXI Interface Pin

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.

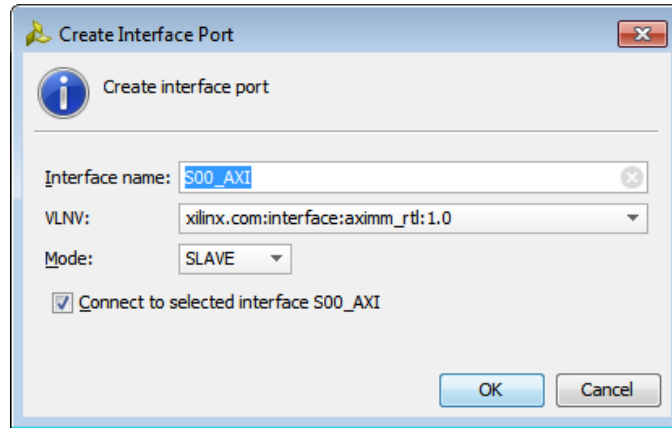


Figure 11: Create Interface Port

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**.

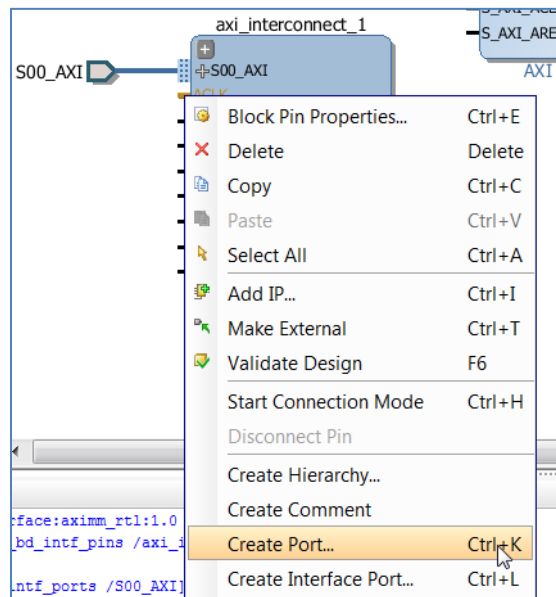


Figure 12: Create Port

5. In the **Create Port** dialog box, as shown in [Figure 13](#), specify the **frequency** as **200 MHz**.
6. Click **OK**.

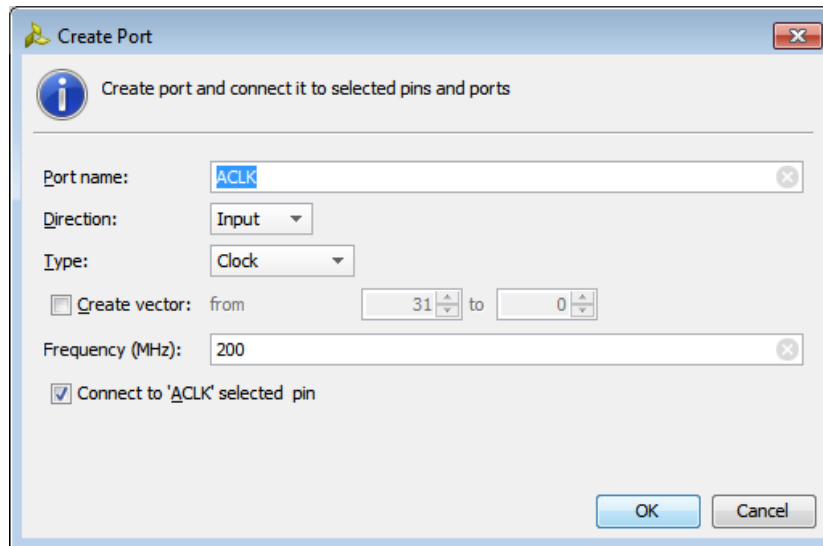


Figure 13: Create Port - ACLK

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.

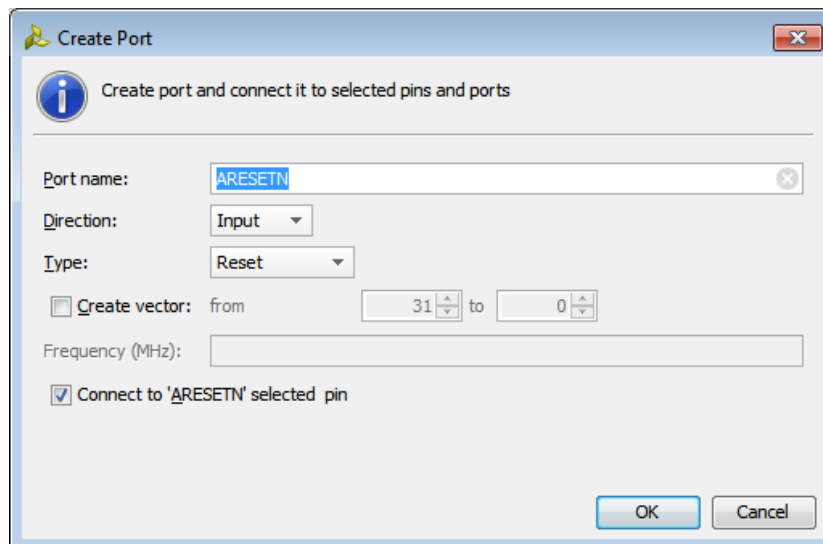


Figure 14: Create Port - ARESETN

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.

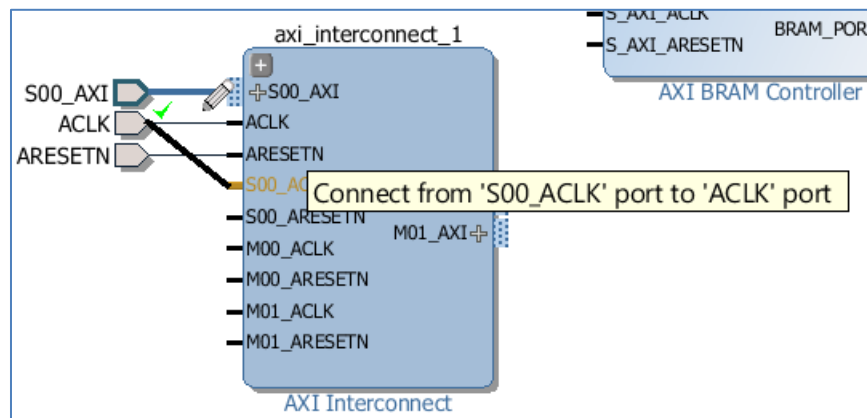


Figure 15: Connect the S00_ACLK pin to the ACLK port

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](#).

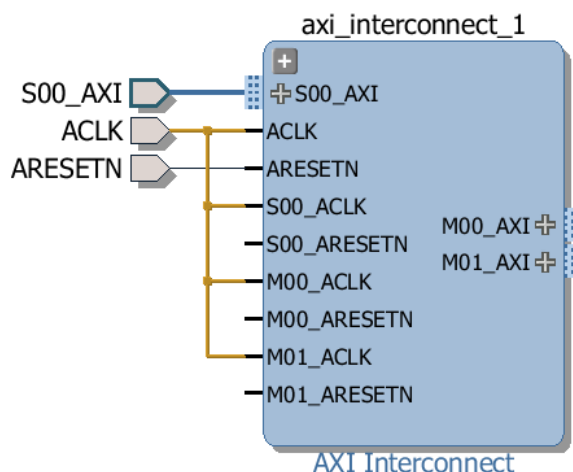


Figure 16: ACLK connections

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

13. Place the cursor over the **S00_ARESETN** pin.
14. **Click and drag** the cursor to the **ARESETN** port.
15. **Release** the mouse button to make the connection.

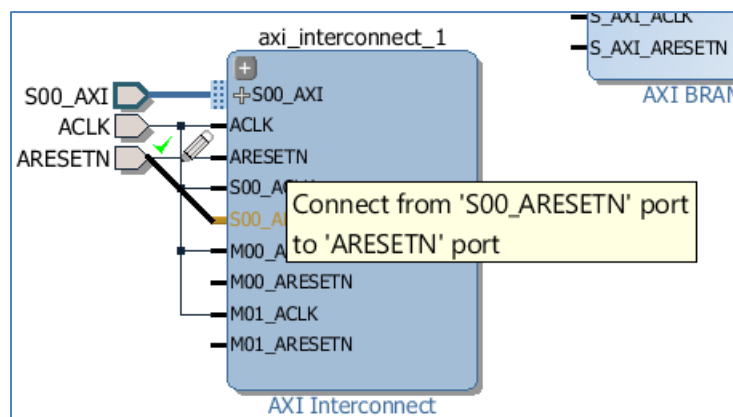


Figure 17: Connect S00_ARESETN to the ARESETN port

16. Repeat the steps to **connect** the **M00_ARESETN** and the **M01_ARESETN** pins of the AXI Interconnect to the ARESETN port.

The AXI Interconnect should now look as shown in Figure 18.

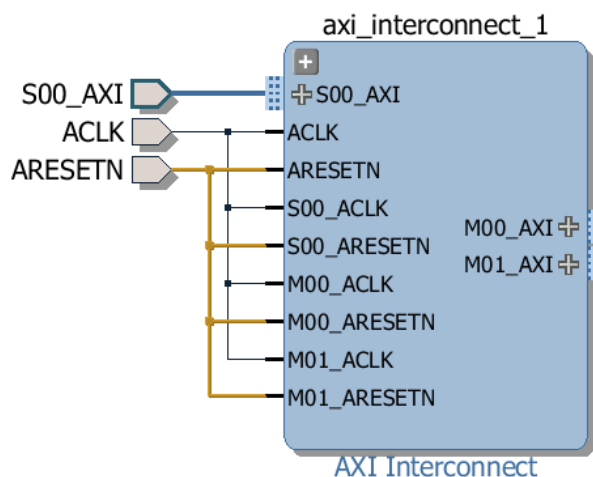


Figure 18: ARESETN Connections

Step 4: Customize IP

1. **Double click** on the **Block Memory Generator** IP to customize it.

The Re-customize IP dialog box opens as shown in Figure 19.

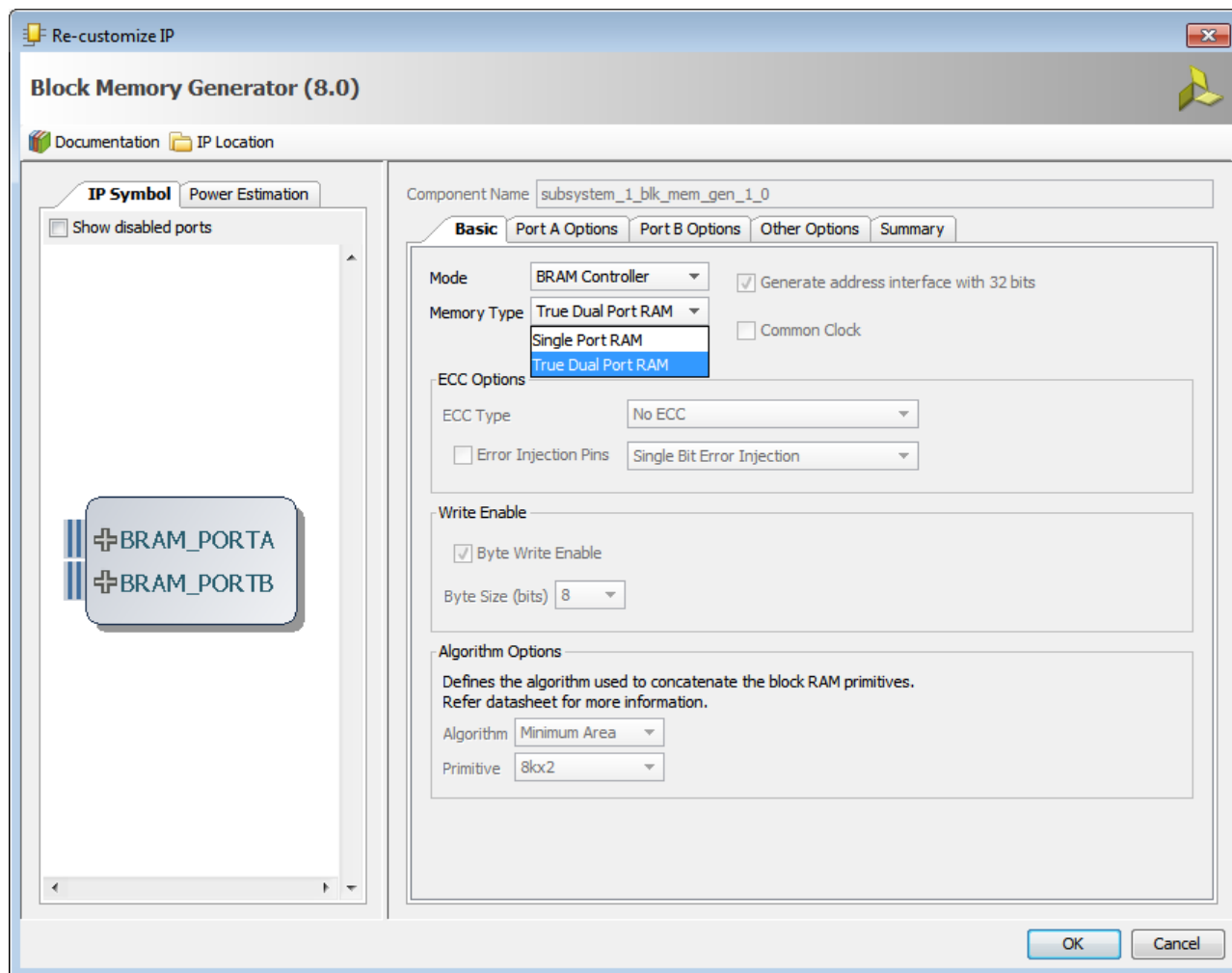


Figure 19: Customize the Block Memory Generator as a True Dual Port RAM

2. On the Basic tab of the Re-Customize IP dialog box, ensure that the **Mode** field is set to **BRAM Controller**.
3. Change the **Memory Type** to **True Dual Port RAM** using the pull-down menu.
4. Leave all the **other options** with their **default** values, and **click OK** to implement the changes in the instantiated Block Memory Controller.

5. **Connect** the **BRAM_PORTA** and **BRAM_PORTB** pins of the AXI BRAM Controller **to** the **BRAM_PORTA** and **BRAM_PORTB** pins of the Block Memory Generator respectively, as seen in XXX.

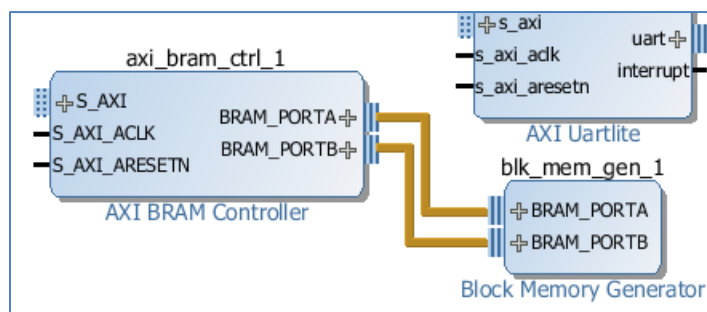


Figure 20: Connecting Interface Pins

6. On the IP integrator design canvas sidebar menu, **click** the **Regenerate Layout** button to redraw the subsystem design.

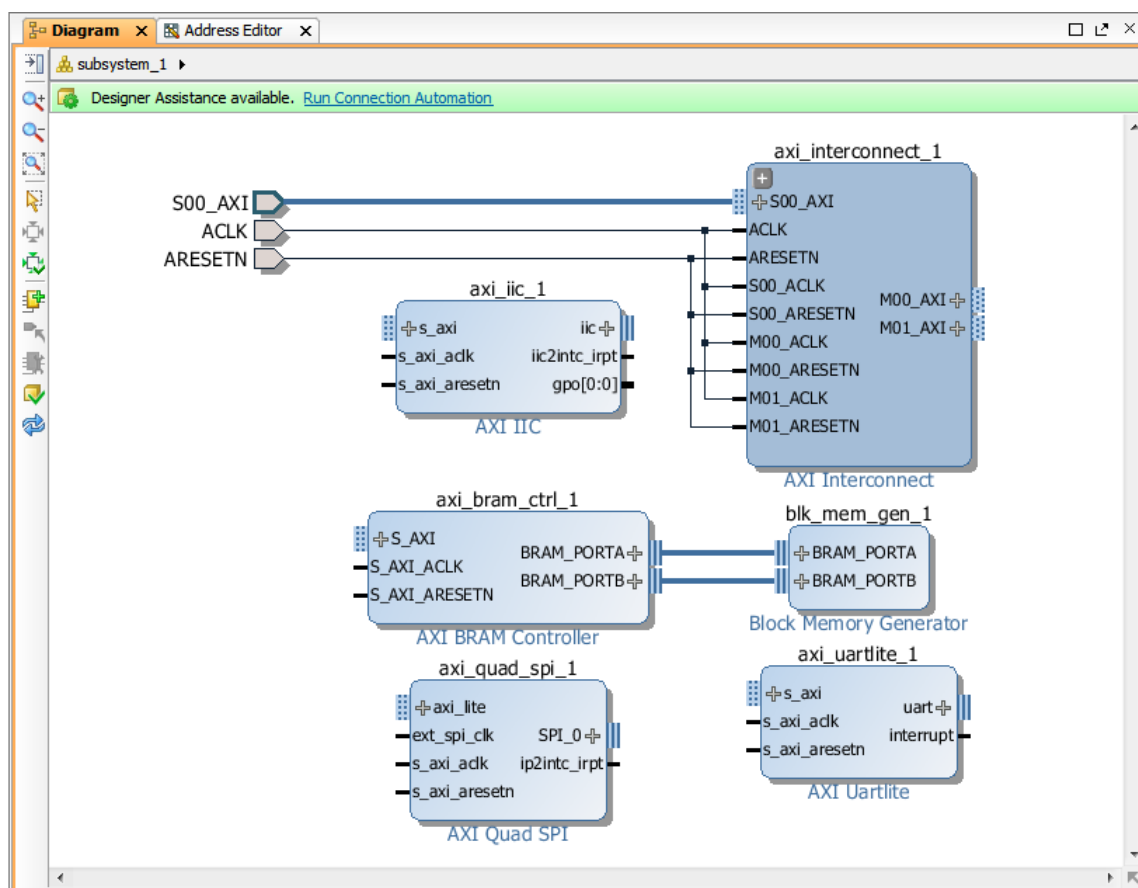


Figure 21: Master Ports of the AXI Interconnect

Looking at the IP integrator design canvas, you should see that there are four slave cores that need to be connected to the AXI Interconnect: AXI BRAM Controller, AXI Uartlite, AXI Quad SPI and AXI IIC. However, there are currently only two master interfaces on the AXI Interconnect. You must re-customize the AXI Interconnect to increase the number of master interfaces to four.

7. **Double-click** on the **AXI Interconnect** core to open the Re-Customize IP dialog box as seen in Figure 22.

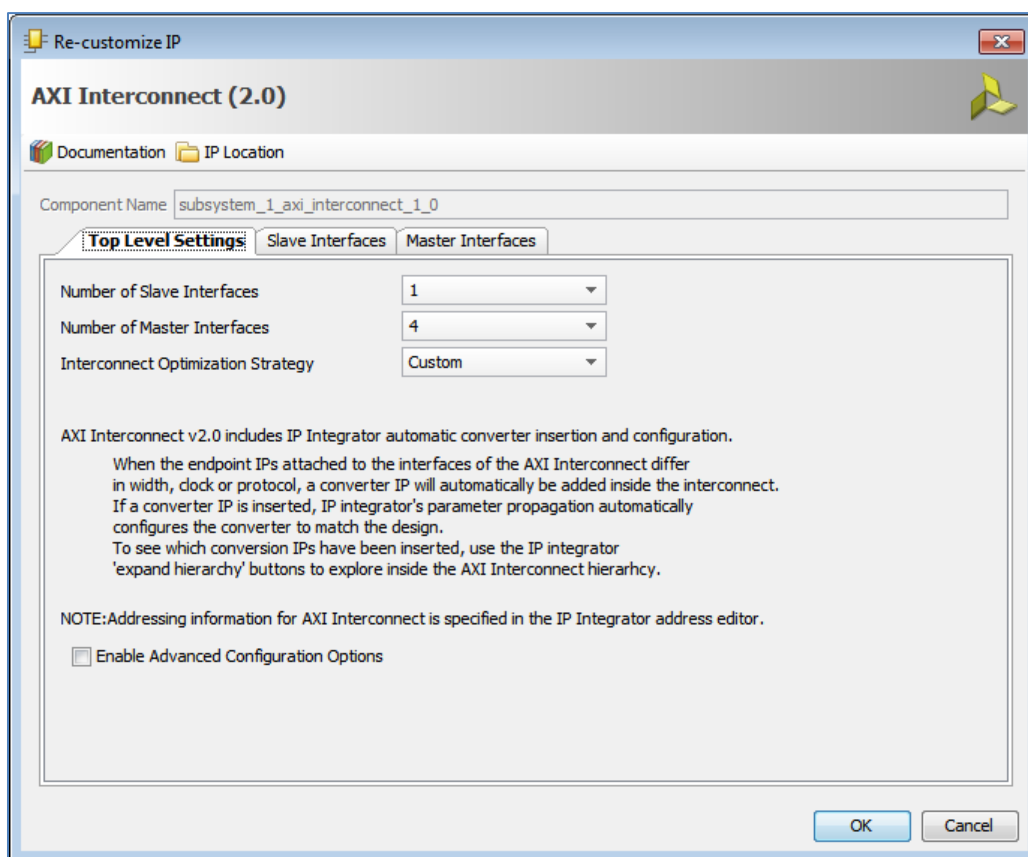


Figure 22: Re-customize the AXI Interconnect

8. From the Top Level Settings Tab, set the **Number of Master Interfaces** to **4** from the pull down menu.
9. 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 four, and adding the necessary clock and reset pins to support these new master interfaces, as shown in Figure 23.

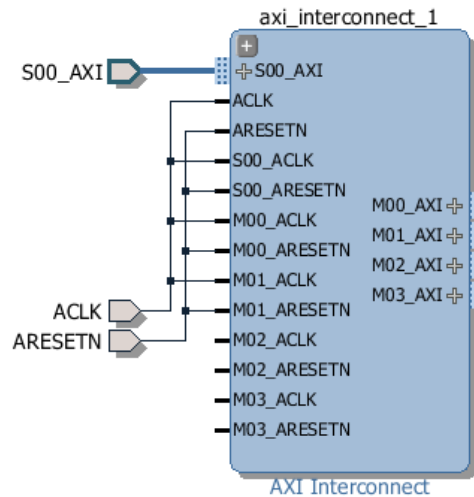


Figure 23: AXI Interconnect with 4 Master Interfaces

10. **Connect** all the **new clocks** to the **ACLK** port, **and** the new **resets** to the **ARESETN** port.
Now you can connect all four slave IP cores through the AXI Interconnect.

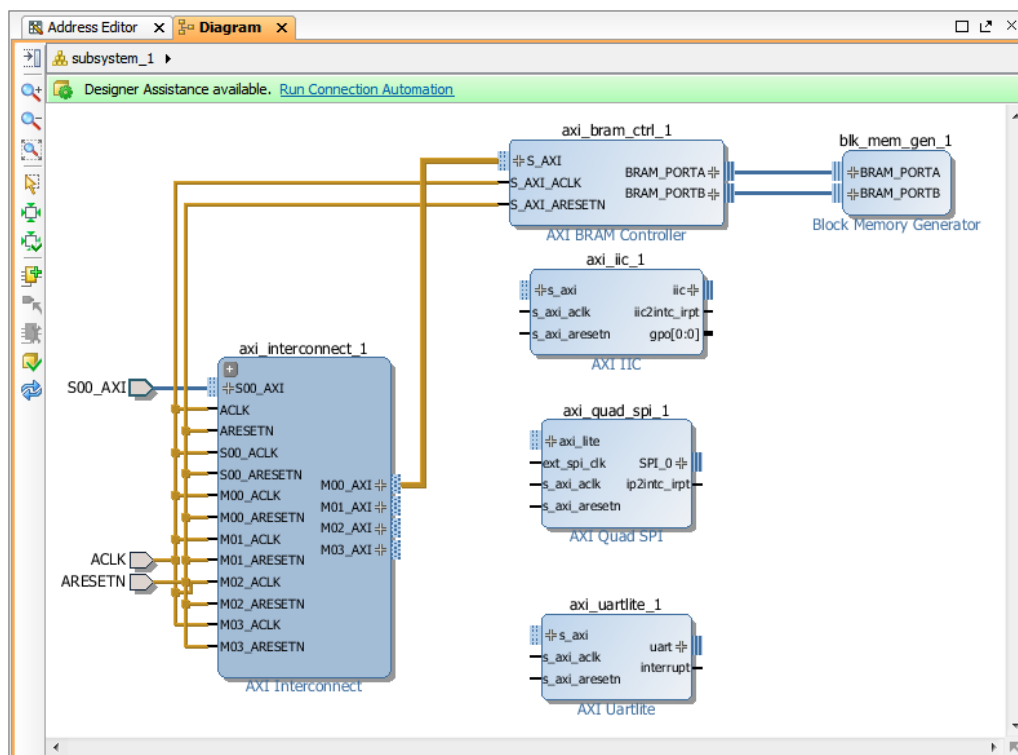


Figure 24: Connect the AXI BRAM Controller to the AXI Interconnect

11. **Connect** the **S_AXI** interface of the AXI BRAM Controller **to M00_AXI** interface of the AXI Interconnect, as shown in Figure 24.

12. **Connect** the **S_AXI_CLK** and the **S_AXI_RESETN** pins of the AXI BRAM Controller **to** the **ACLK** and **ARESETN** ports.
13. Using the same steps, **connect** the remaining **slave IP** cores in the design **to** the **AXI Interconnect**.

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.

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

The IP integrator design canvas should look similar to what is shown in Figure 25.

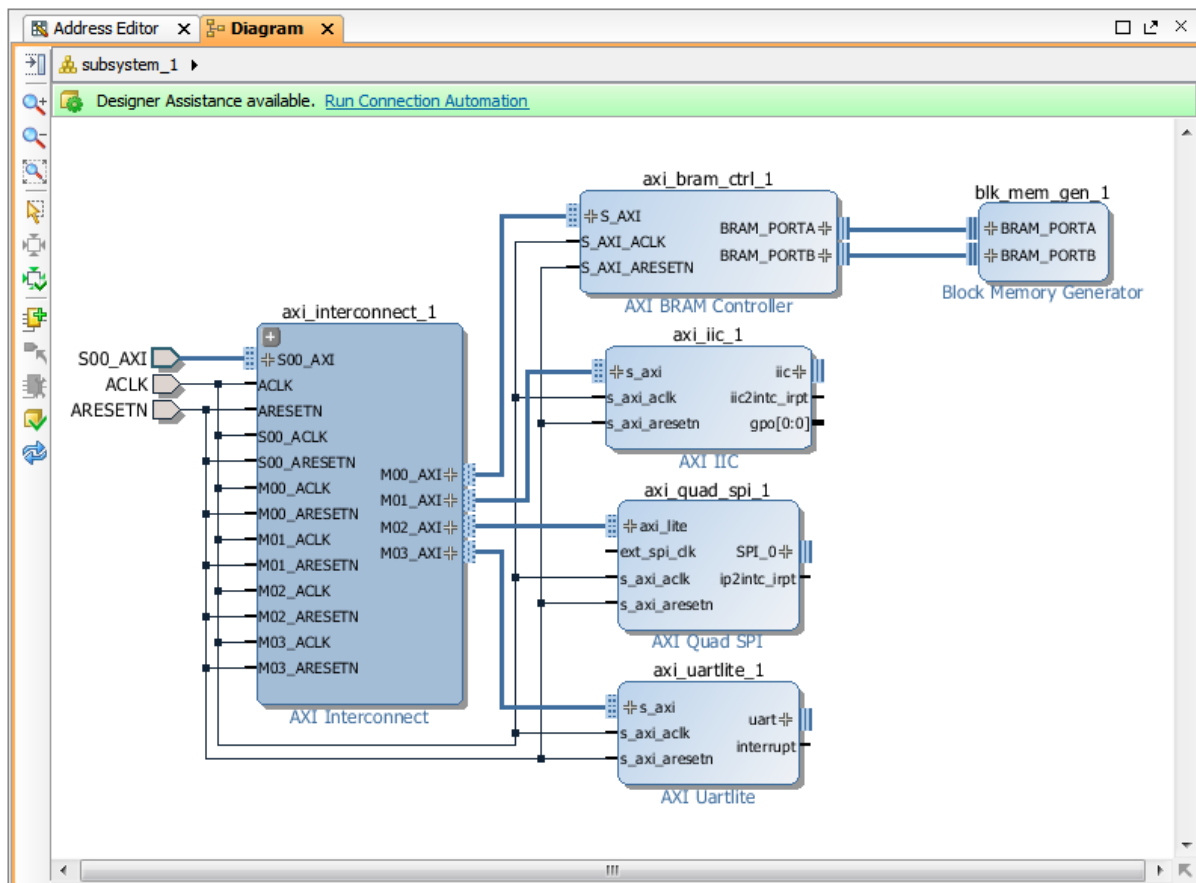


Figure 25: Connecting to the AXI Interconnect

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

15. **Click** the **Tcl console** at the bottom of the Vivado IDE to make it the active window.
16. **Enter** the following **command** at the **Tcl prompt** to save the current subsystem design:

```
save_bd_design
```

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.

IP integrator offers the Design Assistance feature to automate certain kinds of connections. For the current subsystem design, you can connect the UART and IIC interfaces to external ports using connection automation.

1. Click on **Run Connection Automation** link, in the banner at the top design canvas, and select **/axi_iic_1/iic**, as shown in Figure 26.

Notice the selected interface pin is highlighted on the AXI IIC core.

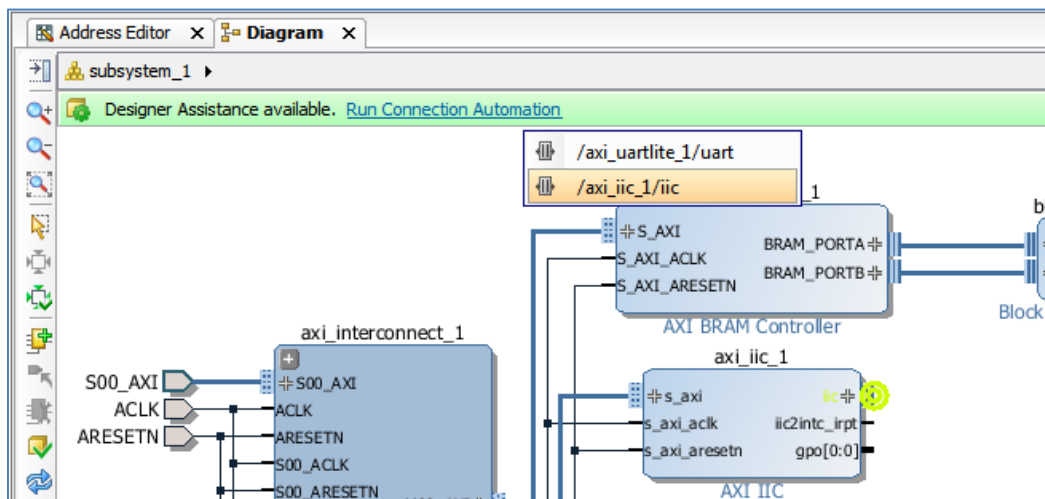


Figure 26: Run Connection Automation

The Run Connection Automation dialog box opens.

2. Click **OK**.

An external port is added to the subsystem design, and a wire connects the specified pin to the port.

3. Click the **Run Connection Automation** link a second time, and select **/axi_uartlite_1/uart**.

The Run Connection Automation dialog box opens a second time.

4. Click **OK**.

5. **Right-click** on the **SPI_0** interface of the AXI Quad SPI, and select **Create Interface Port**.

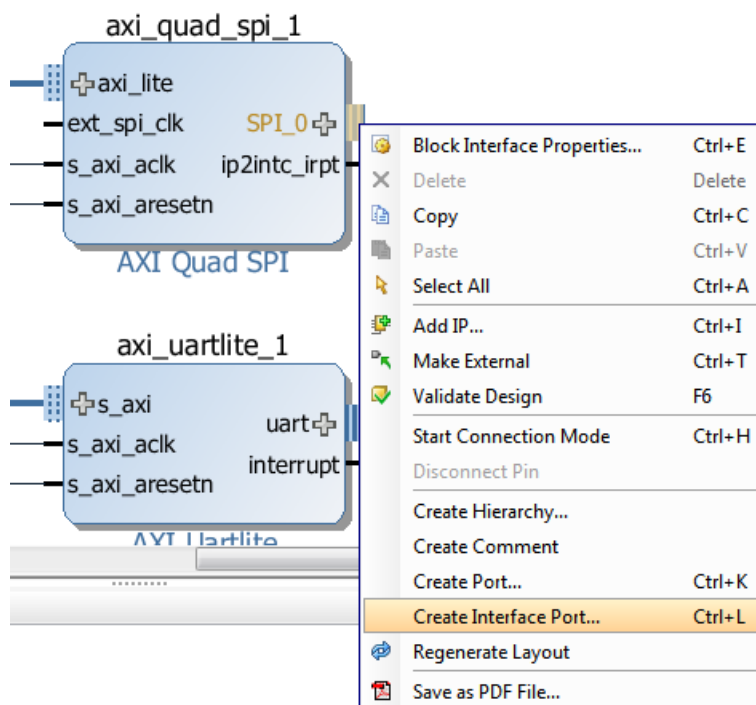


Figure 27: Create SPI_0 Interface Port

The Create Interface Port dialog box opens.

6. **Click OK** to accept the defaults.
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 28.

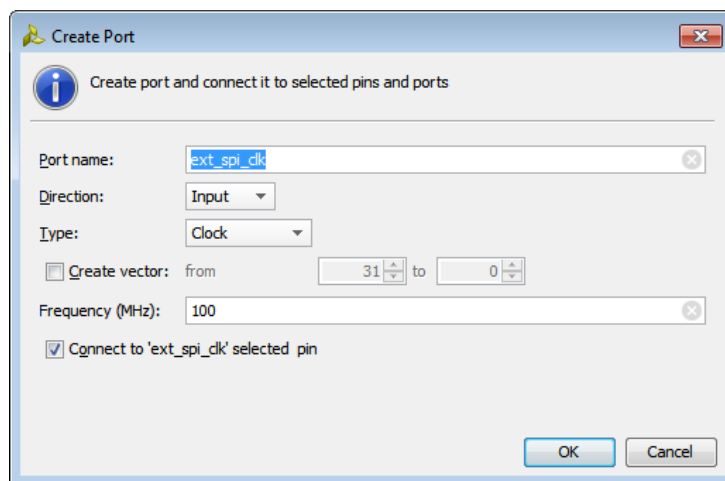


Figure 28: Create ext_spi_clk Port

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

At this time, you have completed the needed connections to the IP integrator subsystem design.

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 29.

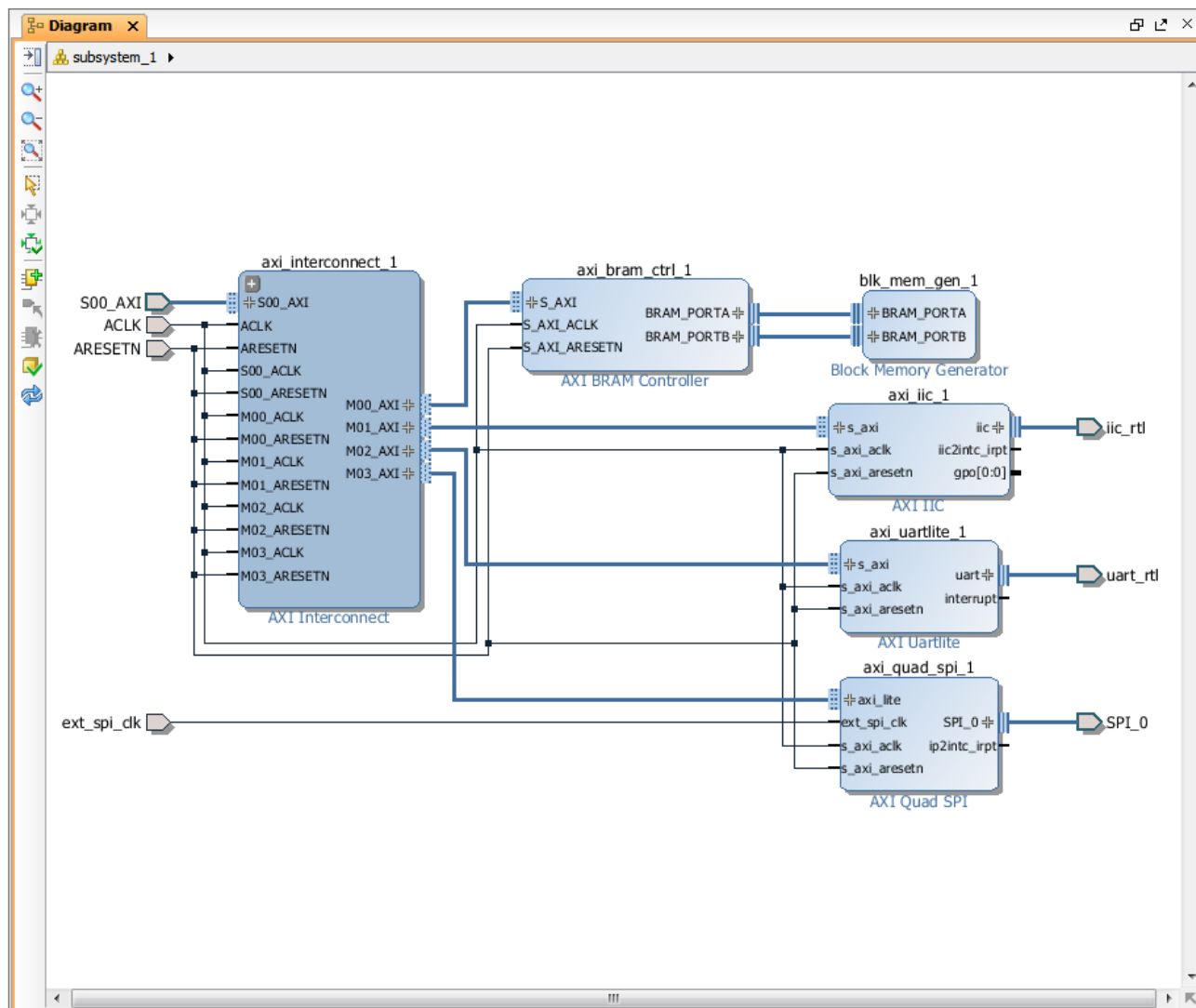


Figure 29: Completed IP Subsystem Design

Step 6: 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.

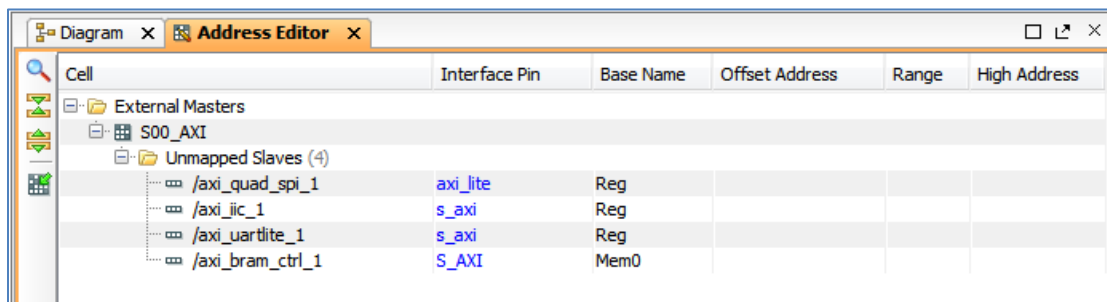


Figure 30: Address Editor

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

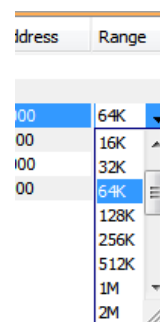
This command maps slave address segments to master address spaces, thereby creating address segments in the master. You can change these automatic addresses later by clicking in the corresponding column and changing the values.

The Address Editor should now look similar to Figure 31.

Cell	Interface Pin	Base Name	Offset Address	Range	High Address
External Masters					
S00_AXI					
/axi_bram_ctrl_1	S_AXI	Mem0	0xC0000000	4K	0xC0000FFF
/axi_iic_1	s_axi	Reg	0x40800000	64K	0x4080FFFF
/axi_quad_spi_1	axi_lite	Reg	0x44A00000	64K	0x44A0FFFF
/axi_uartlite_1	s_axi	Reg	0x40600000	64K	0x4060FFFF

Figure 31: Automatically Assigned Addresses

- 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.
- Select** the **Diagram** tab, to return to the IP Integrator design canvas.
- From the sidebar menu of the design canvas, run the IP subsystem design rule checks by clicking the **Validate Design** button.



Alternatively, you can do the same by selecting **Tools > Validate Design** from the main menu, or by right-clicking in the design canvas and selecting **Validate Design** from the popup menu.

The Validate Design dialog box opens.

- Click OK** to validate the design.

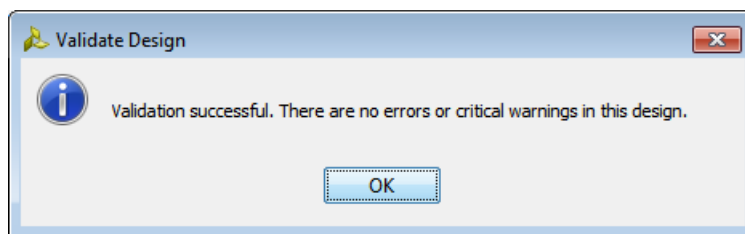


Figure 32: Validate Design dialog box

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

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

Step 7: 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 of the top-level design, or may be the only block of 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.

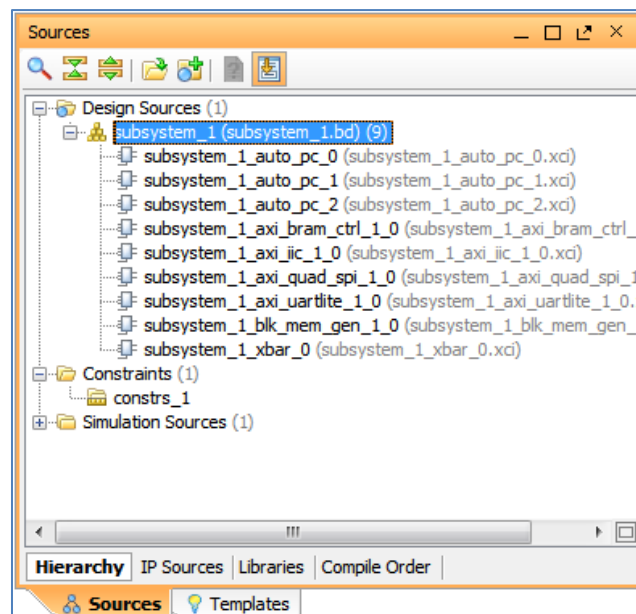


Figure 33: Generate Output Products

The Manage Output Product dialog box opens, as shown in Figure 34, to regenerate the various output products associated with the subsystem design.

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

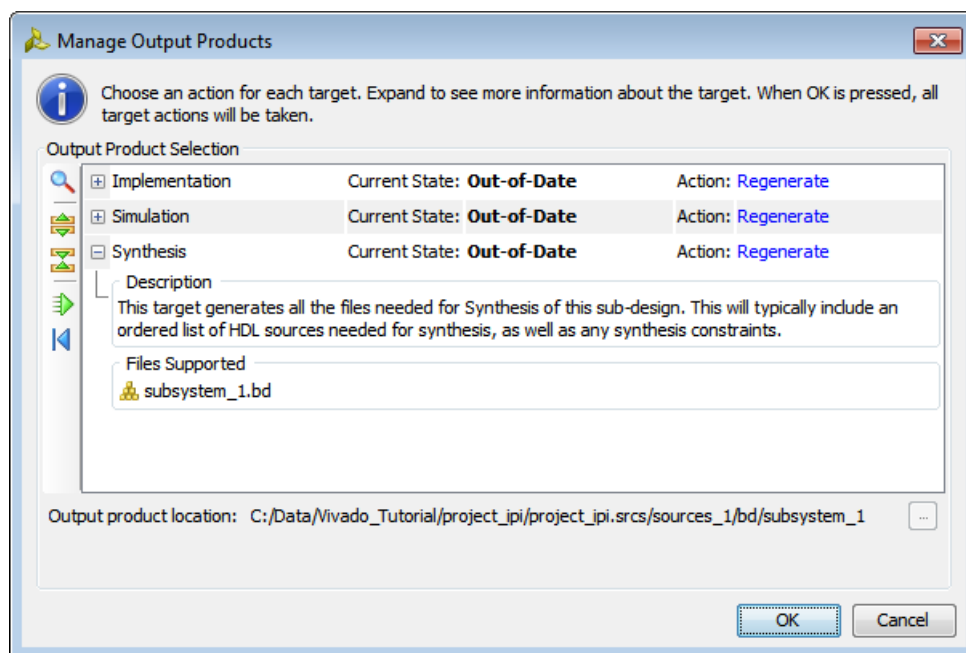


Figure 34: Manage Output Products

3. 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 to report that a Verilog wrapper file has been created and added to the top-level of the design.

4. **Click OK** to close the dialog box.

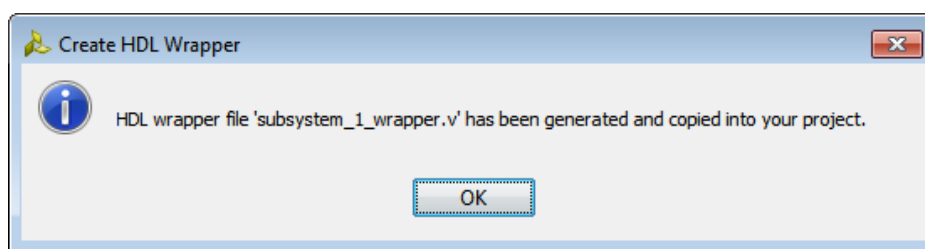


Figure 35: Create HDL Wrapper dialog box

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

5. From the Project Manager section of the **Flow Navigator**, click **Add Sources**.

The Add Sources dialog box opens up.

6. Select the **Add or Create Constraints** option and click **Next**.
7. In the Add or Create Constraints dialog box, click **Add Files**.

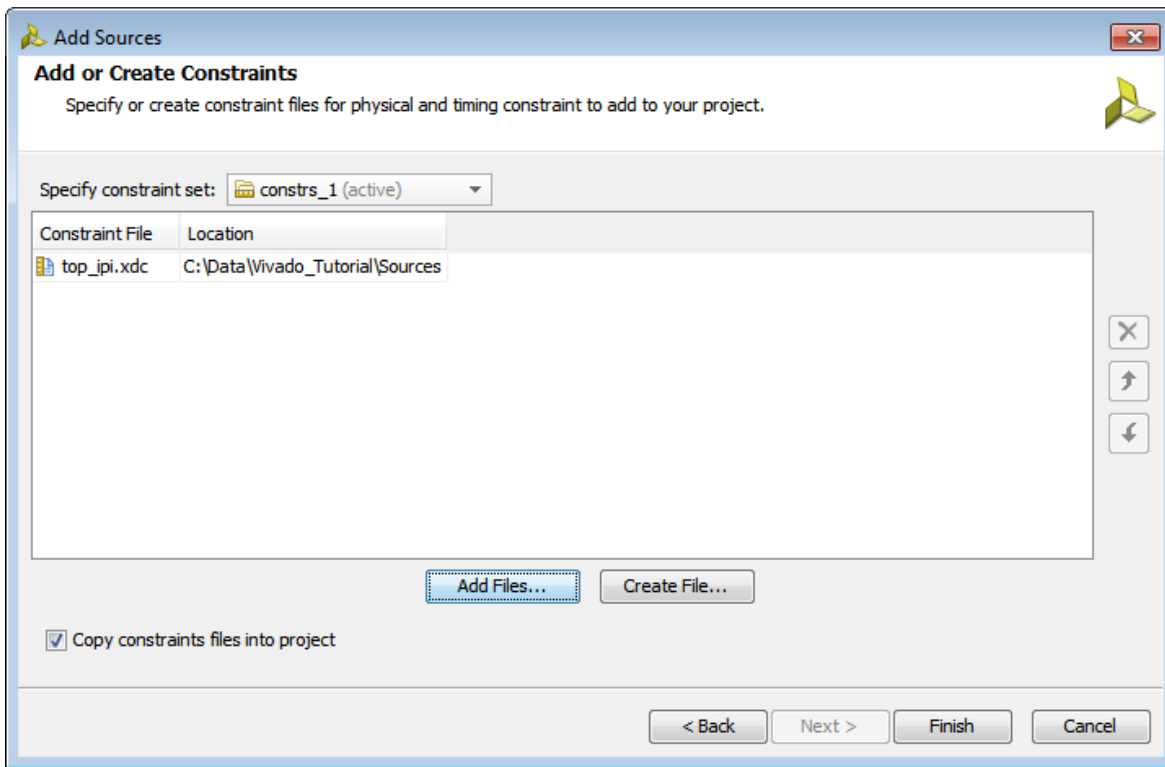


Figure 36: Add Constraints

The Add Constraints Files dialog box opens.

8. **Select** the **top_ipi.xdc** file you extracted to the *<Extract_Dir>/Vivado_Tutorial/Sources* folder at the start of this tutorial, and click **OK**.
9. In the Add or Create Constraints dialog box, make sure that **Copy constraints files into project** is checked.
10. **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.

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

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.

12. **Click Yes**.

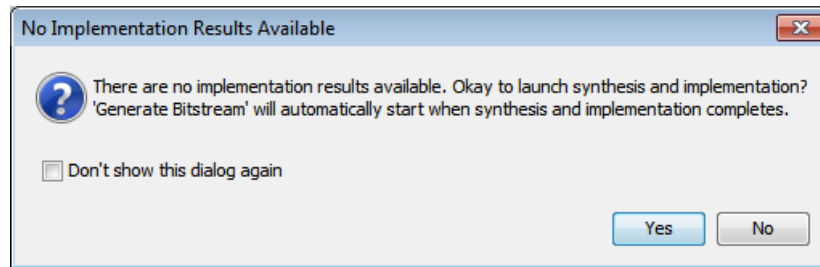


Figure 37: No Implementation Results Available

Synthesis will run for a few minutes as can be seen in the top right corner of Vivado IDE. You may see some Warnings that might pop-up while the design is trying to go through the flow. You can close and ignore these warnings.

After bitstream is generated, the Bitstream Generation Completed dialog box opens. View Reports is selected by default.

13. Select **Open Implemented Design**, and click **OK**.

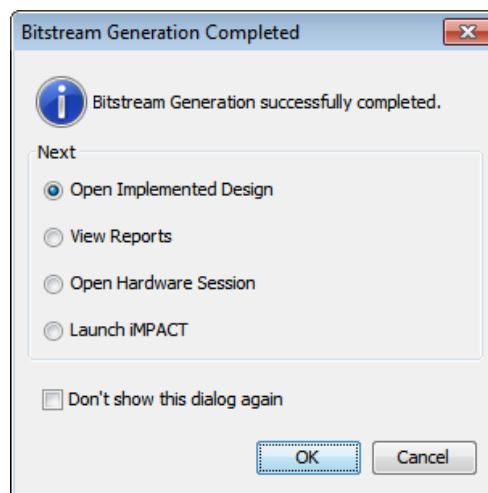


Figure 38: Bitstream Generation Completed

14. In the Implemented Design section of the **Flow Navigator**, click on **Report Timing Summary**.

The Report Timing Summary dialog box opens as seen in Figure 39.

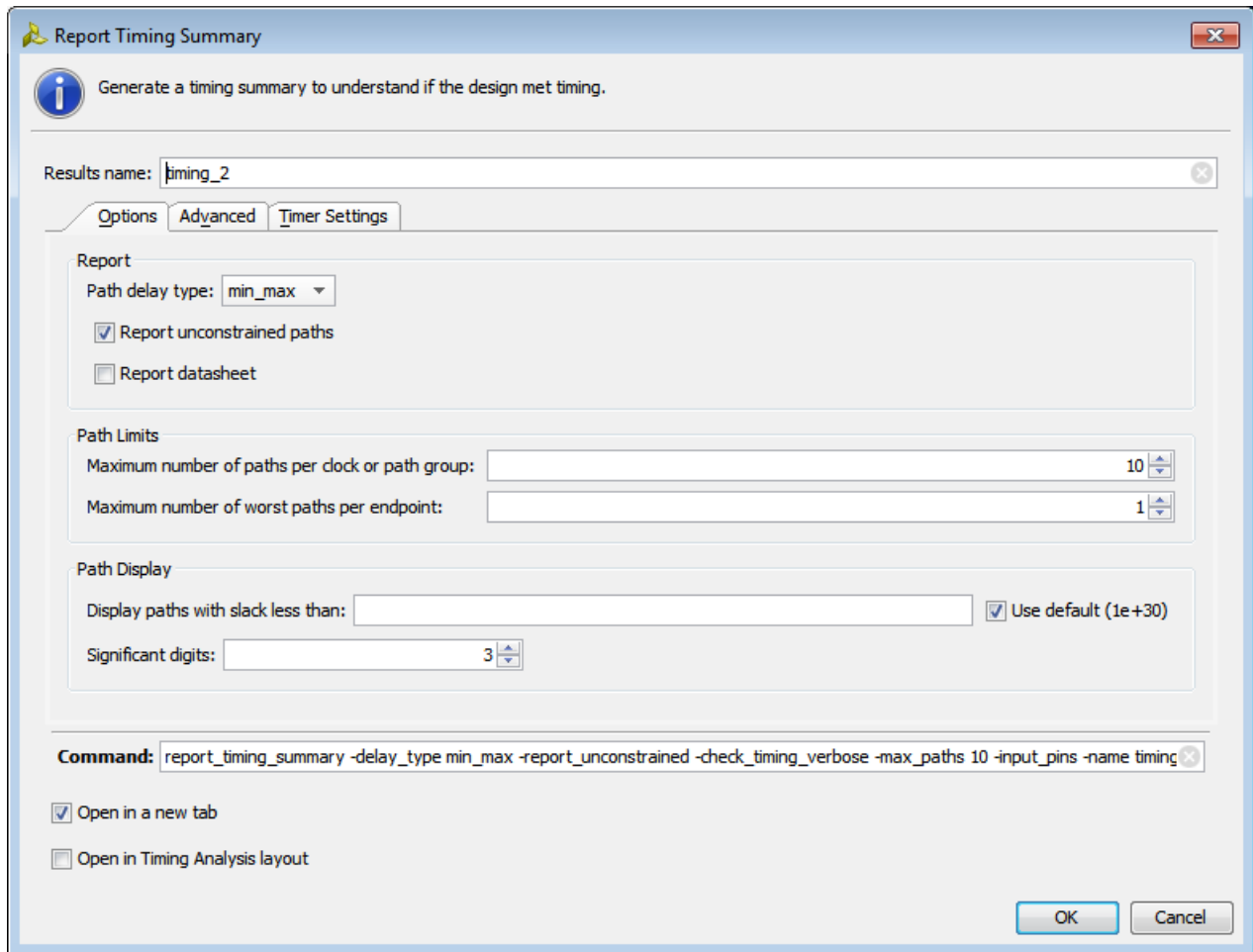


Figure 39: Report Timing Summary

15. **Click OK** to accept the defaults, and run timing analysis.
16. **Ensure** that all **timing constraints** have been met by looking at the Timing Summary report.

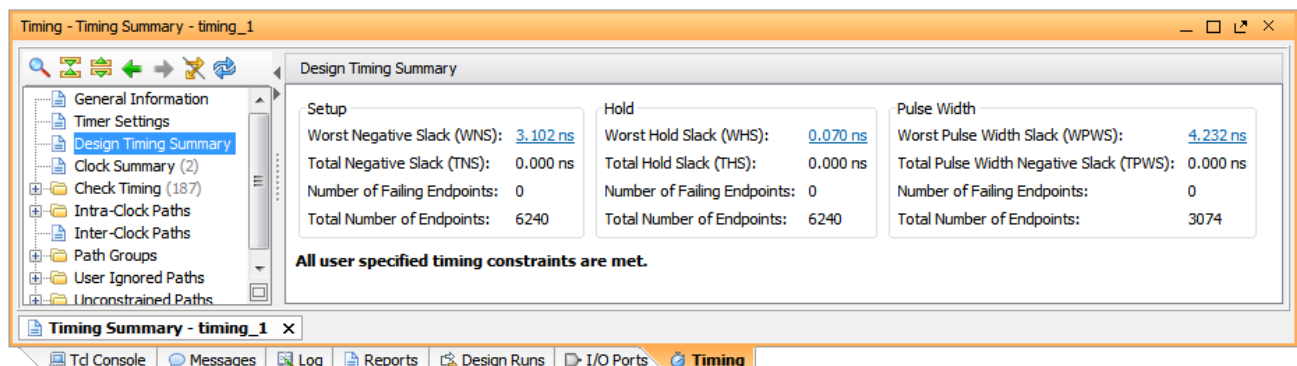


Figure 40: Timing Summary Report

Conclusion

This tutorial walks you through creating a simple IP Integrator subsystem design by instantiating some 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.

The tutorial gives you hands-on experience, and a basic understanding of many of the features of the Vivado IP integrator.