

Vivado Design Suite User Guide

Designing IP Subsystems Using IP Integrator

UG994 (v2013.2) June 19, 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, 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.

Date	Changes
06/19/2013	Initial Xilinx Release.

Table of Contents

Designing IP Subsystems Using IP Integrator.....	1
Designing IP Subsystems.....	4
Overview.....	4
Creating a Project	4
Designing with IP Integrator.....	7
Exporting Hardware Definition to SDK	29
Packaging the Block Diagram	31
Adding and Associating an ELF File to an Embedded Design	35

Designing IP Subsystems

Overview

The Vivado IP integrator feature lets you create complex system designs by instantiating and interconnecting IP from the Vivado IP catalog on a design canvas. You can create designs interactively through the IP integrator canvas GUI or programmatically through a Tcl programming interface. Designs are typically constructed at the interface level (for enhanced productivity) but may also be manipulated at the port level (for precision design manipulation).

An interface is a grouping of signals that share a common function. An AXI4-Lite master, for example, contains a large number of individual signals plus multiple buses, which are all required to make a connection. If each signal or bus is visible individually on an IP symbol, the symbol will be visually very complex. By grouping these signals and buses into an interface, the following advantages can be realized. First, a single connection in IP integrator (or Tcl command) will make a master to slave connection. Next, the graphical representation of this connection will be simple – a single connection. Finally, Design Rule Checks (DRCs) that are aware of the specific interface are run to assure that all the required signals are connected properly.

A key strength of IP integrator is that it provides a Tcl extension mechanism for its automation services so that system design tasks such as parameter propagation, can be optimized per-IP or application domain. Additionally, IP integrator implements dynamic, runtime DRCs to ensure, for example, that connections between the IP in an IP integrator design are compatible and that the IP themselves are properly configured.

Creating a Project

While entire designs can be created using IP integrator, the typical design will consist of HDL, IP, and IP integrator block diagrams. This section is an introduction to creating a new IP integrator-based design.

As shown in the figure below, you start by clicking on **Create New Project** in the Vivado GUI to create a new project. You can add VHDL or Verilog design files, any custom IP, and other types of design source files to the project using this wizard.

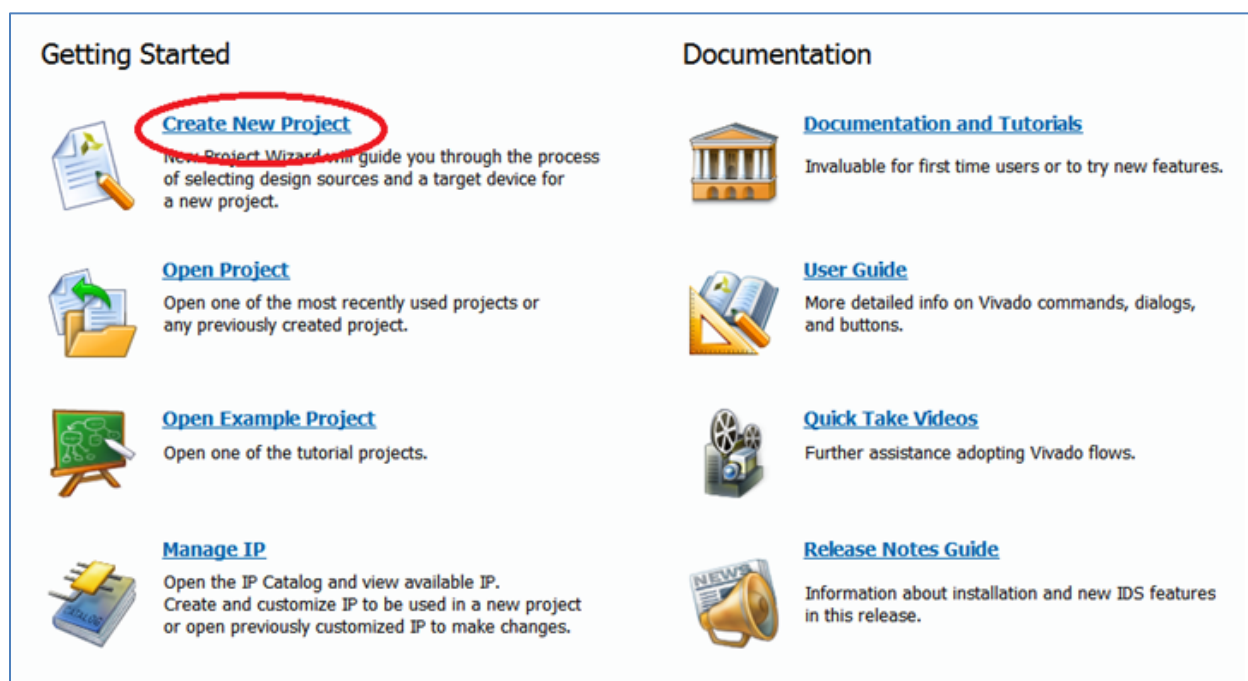


Figure 1: Creating a New Project

As shown in the figure below, you can also select the target device or a Xilinx target board. Vivado also supports multiple versions of Xilinx target boards. So carefully select your target hardware.

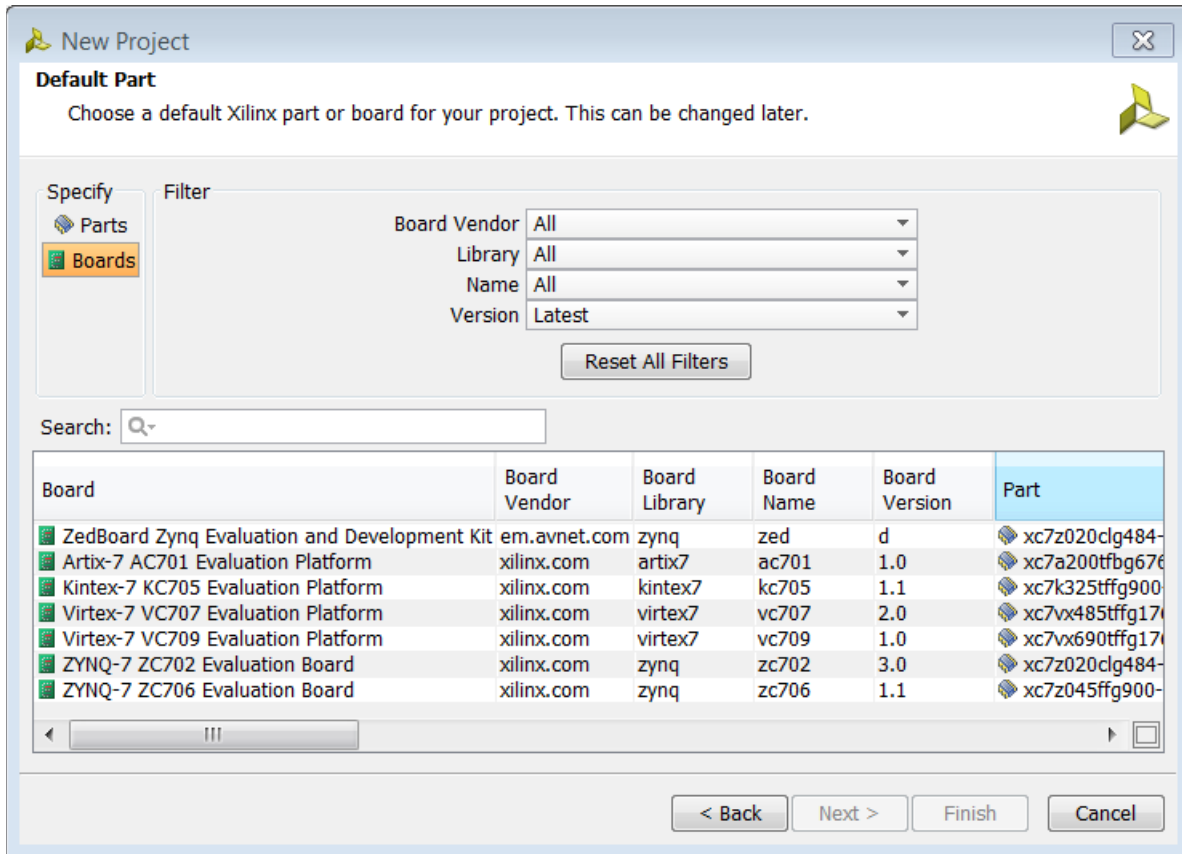


Figure 2: Choosing a New Project Target Device

Note: You can perform the same actions by using the following Tcl commands. When displaying the Tcl commands in this document, the symbols < > are used to surround parameters that are specific to your design. The < > symbols should not be included in the command string.

The Tcl equivalent commands are:

```
create_project xx <your_directory>/xx -part xc7k325tffg900-2
set_property board kc705 [current_project]
set_property target_language VHDL [current_project]
```

Designing with IP Integrator

You create a new block diagram in the Flow Navigator by clicking on **Create Block Design** under the IP Integrator heading.

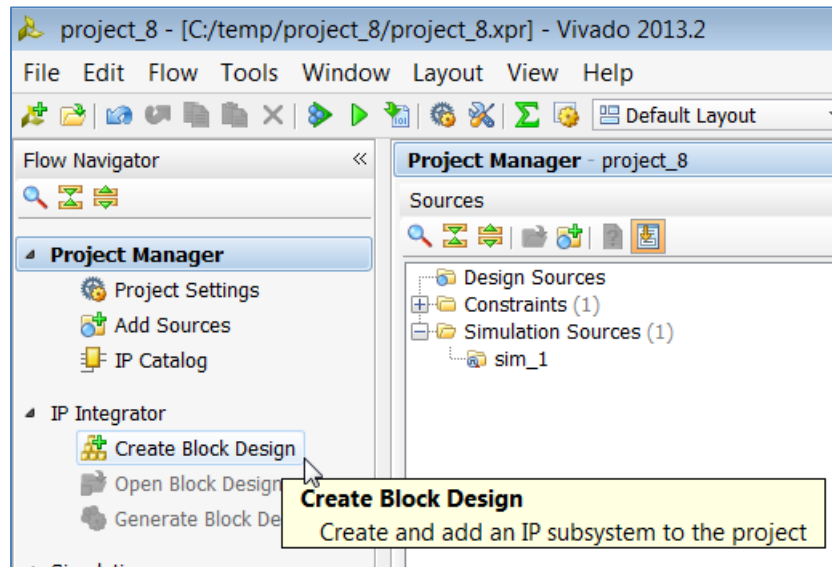


Figure 3: Creating a Block Diagram

The Tcl equivalent command is:

```
create_bd_design "<your_design_name>"
```

Re-sizing the IP Integrator Diagram

Once the design is created, a canvas is presented that you can use to stitch up your design. This canvas can be re-sized as much as possible by re-sizing panes in the Vivado IDE GUI. It is possible to move the diagram to a separate monitor by clicking on the **Float Window** button in the upper-right corner of the diagram. You can also double click on the **Diagram** tab at the upper-left corner of the diagram to increase the size of the diagram. When you double click the tab again, the view returns to the default layout.

Changing the Background Color

You can change the background color of the diagram from the default white color. As shown in the following figure, you can click the **Block Diagram Options** button in the upper-left corner of the diagram to change the color.

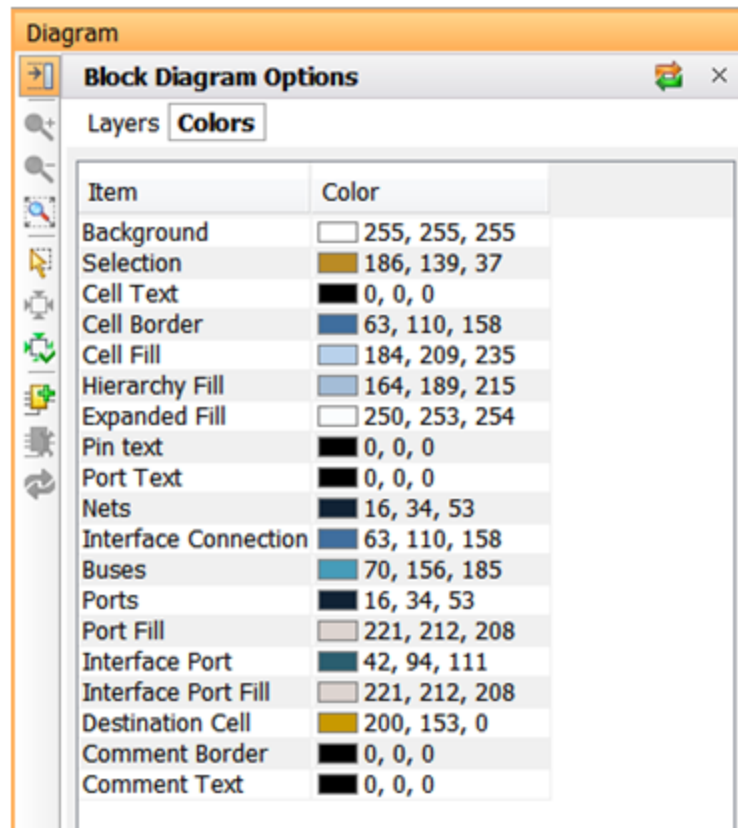


Figure 4: Changing the IP Integrator Background Color

Notice that you can control the colors of almost every object displayed in an IP integrator diagram. For example, changing the background color to 240,240,240 as shown above makes the background light gray. To hide the **Block Diagram Options**, either click the close button in the upper-right corner, or click the **Block Diagram Options** button again.

Using Mouse Strokes and the Left-Button Panel

A northwest stroke (lower-right to upper-left) is **Zoom Fit**

A southwest stroke (upper-right to lower-left) is **Zoom In**

A northeast stroke (lower-left to upper-right) is **Zoom Out**

A southeast stroke (lower-right to upper-left) is **Zoom Area**

The buttons on the left side of the canvas allow the following specific actions to be invoked:

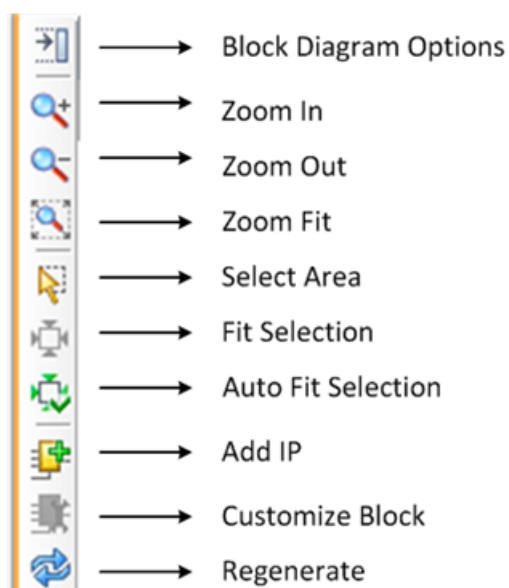


Figure 5: IP Integrator Action Buttons

Adding IP Modules to the Design Canvas

You can add IP modules to a diagram in the following ways:

1. Right click in the diagram and select **Add IP**. A searchable IP Catalog opens.

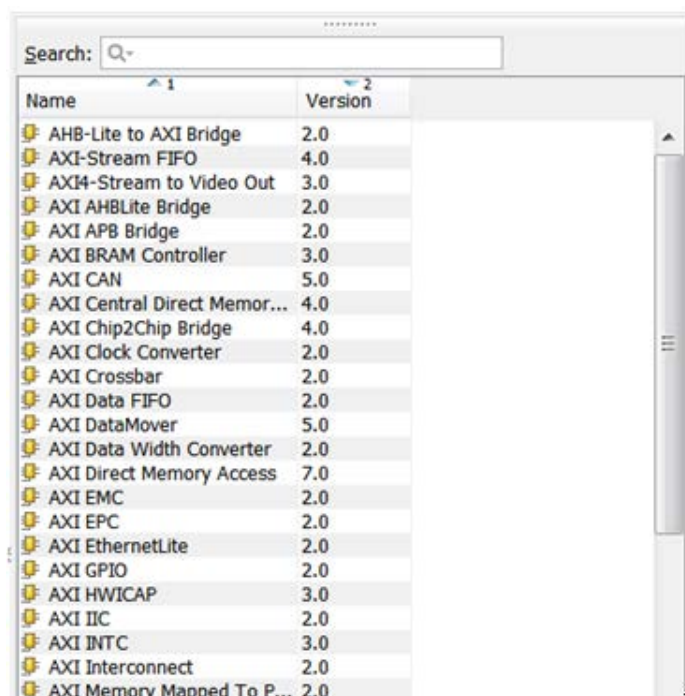


Figure 6: Opening the Vivado IP Catalog

By typing in the first few letters of the IP name in the Search filter, only IP modules matching the string are displayed.

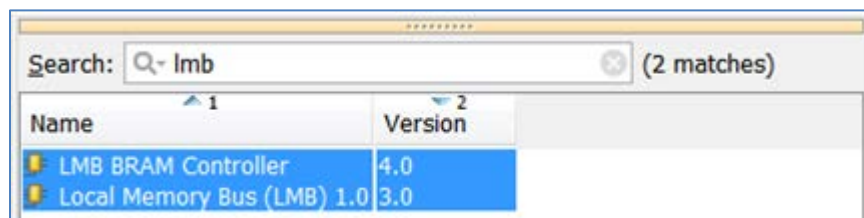


Figure 7: Using the Search Filter in the IP Catalog

2. To add a single IP, you can either click on the IP name and press the **Enter** key on your keyboard, or double click on the IP name.
3. To add multiple IP to the canvas, you can highlight the desired IP (Ctrl-click) and press the **Enter** key.

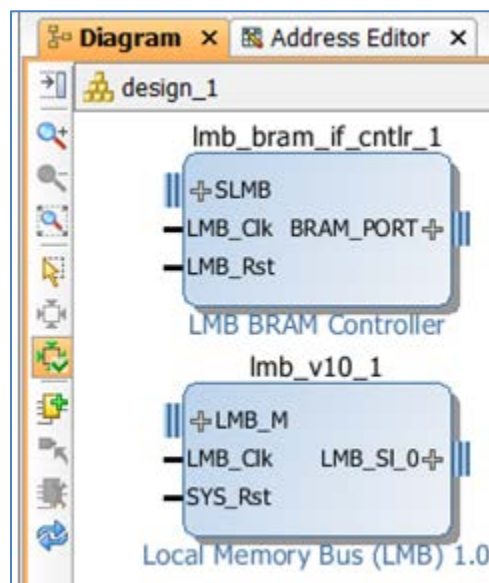


Figure 8: Adding Multiple IP at the Same Time

4. You can also add IP by clicking on the Add IP button on the left side of the canvas.

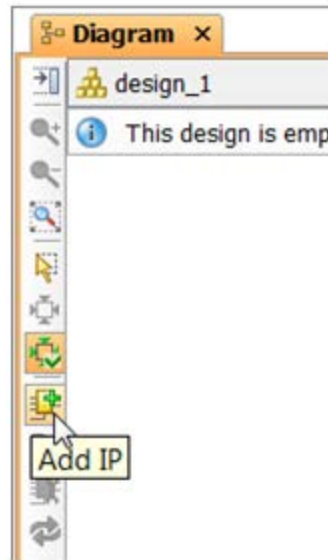


Figure 9: Adding IP by Clicking the Add IP Icon

- Alternatively, IP can also be added by clicking on the link Add IP, in the strip towards the top of the IPI canvas.

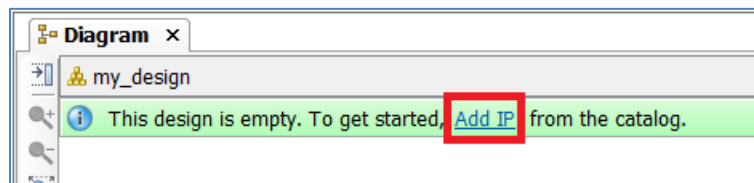


Figure 10: Adding IP by Clicking Add IP Link

For the actions described above, the IP is placed near the cursor location when the **Add IP** command is invoked.

The Vivado IP catalog entry in the Flow Navigator can also be displayed and used. If you are using dual monitors, you can open the IP catalog in its own monitor. If you are using a single monitor, you can float the IP catalog to move it away from the diagram. To add IP from the main IP catalog, you can drag and drop a selected piece of IP from the IP catalog onto the diagram.

Note: If you double click on a piece of IP, that IP will be added to the Vivado project, but not to the block diagram.

Making Connections

When you create a design in IP integrator, you add blocks to the diagram, configure the blocks as needed, make interface-level or simple-net connections, and add interface or simple ports.

Making connections in IP integrator is simple. As you move the cursor near an interface or pin connector on an IP block, the cursor changes into a pencil. You can then click on an interface or pin connector on an IP block, hold down the left-mouse button, and then draw the connection to the destination block.

As shown in the figure below, an interface-level connection is indicated by the more prominent connection box on a symbol.

Clicking on the + symbol on a block expands the interface and displays the associated signals and buses.

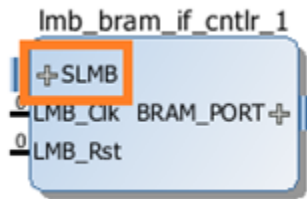


Figure 11: Connection Box on a Symbol

A signal or bus-level connection is shown as a narrow connection line on a symbol. Buses are treated identically to individual signals for connection purposes. As shown in the figure below, when you are making a connection, a green check box appears near the destination connection, indicating a potential connection.

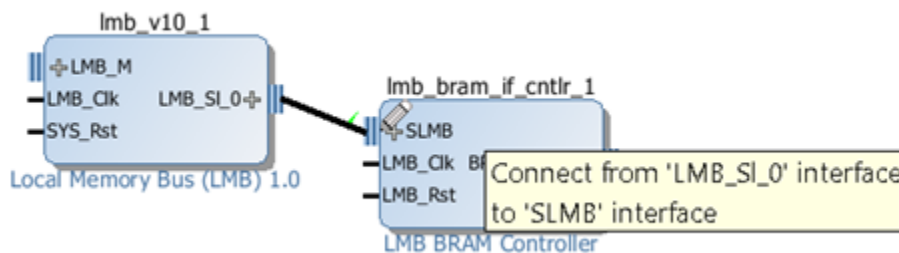


Figure 12: Signal or Bus Connection on a Symbol

6. As shown in the following figure, when signals are grouped as an interface, you should expand the interface first before making individual signal or bus connections.

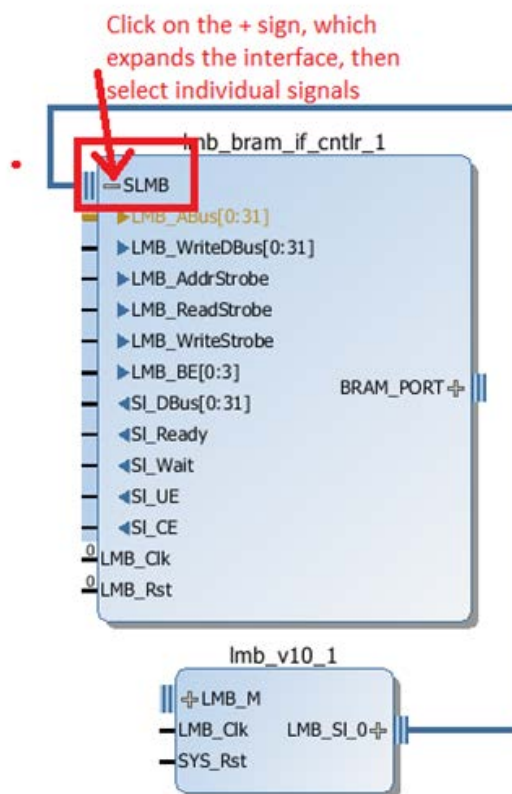


Figure 13: Expanding the Interface Before Making a Connection

7. There are three ways to connect signals and interfaces to external I/O ports.
 - a. As shown the following figure, to connect signals or interfaces to external ports on a diagram, you first select a pin, bus, or interface connection. You can then right-click and select **Make External**. It is possible to Ctrl-click multiple pins and invoke the **Make External** one time.

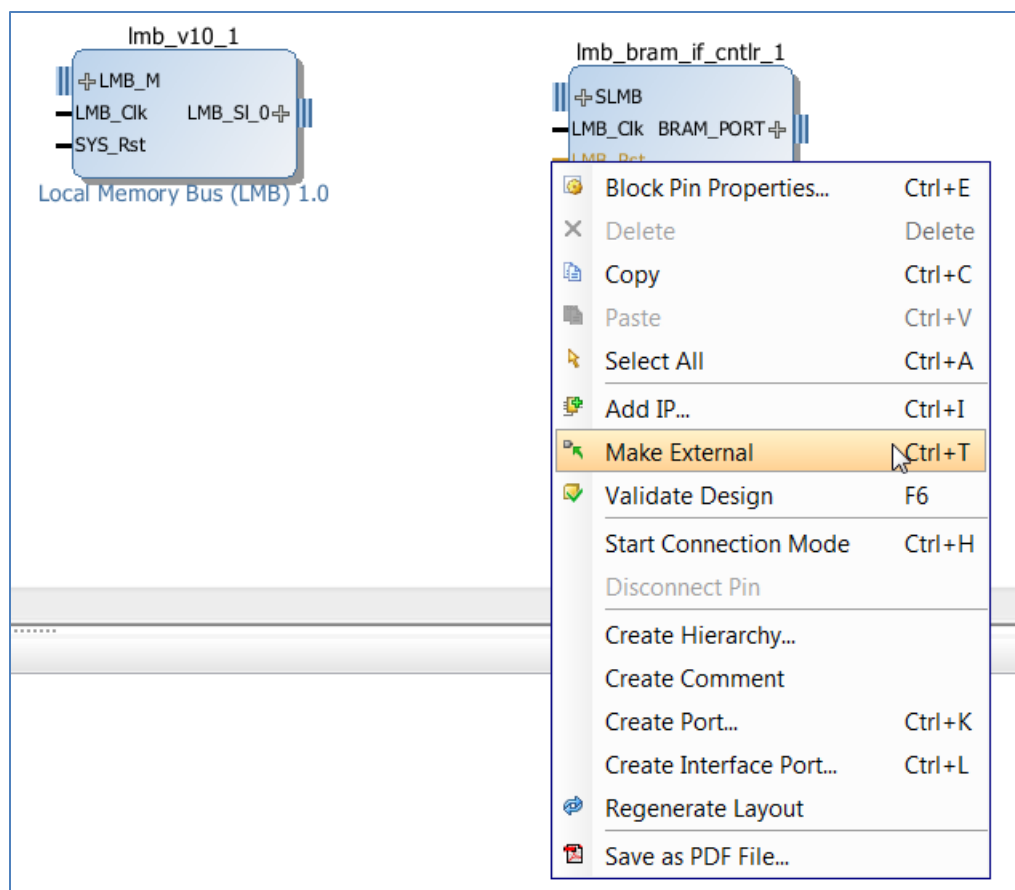


Figure 14: Making External Connections

This command is used to tie a pin on an IP to an I/O port on the block diagram. IP integrator simply connects the port on the IP to an external I/O.

- b. For the second method of making a connection, you can right-click and select **Create Port**, as shown in the following figure. This feature is used for connecting non-interface signals, such as a clock, reset, and uart_txd. **Create Port** gives you more control in terms of specifying the input/output, the bit-width and the type (clk, reset, data) etc. In case of a clock, you can even specify the input frequency.

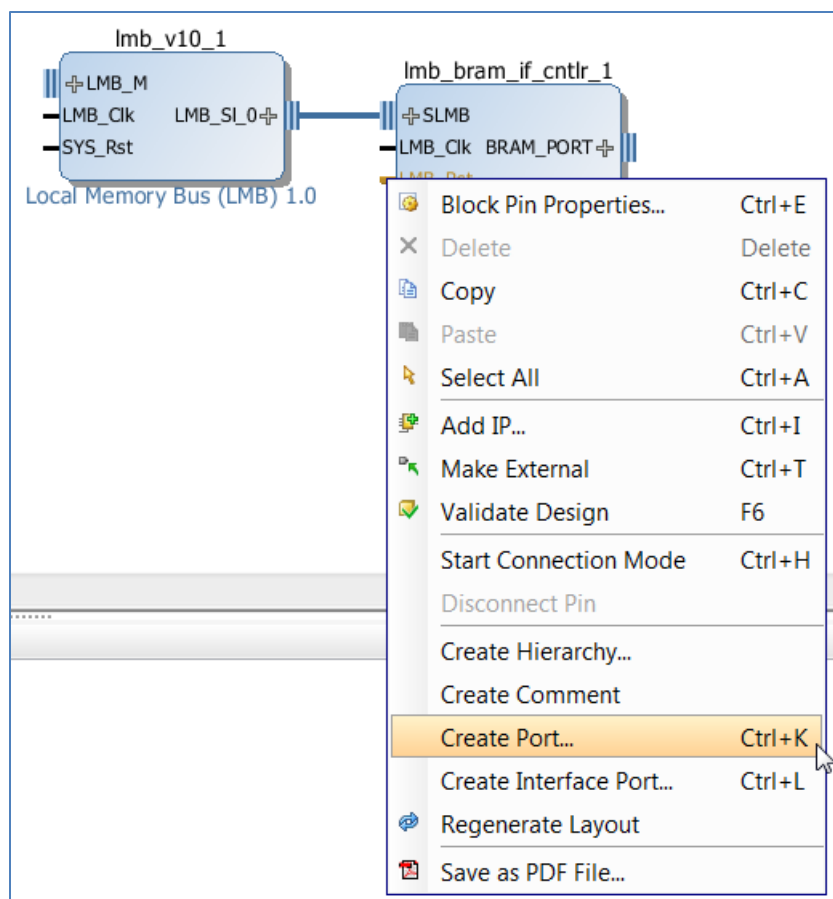


Figure 15: Creating a Port

- c. For the third connection method, you can right-click and select **Create Interface Port**, as shown in the figure below.

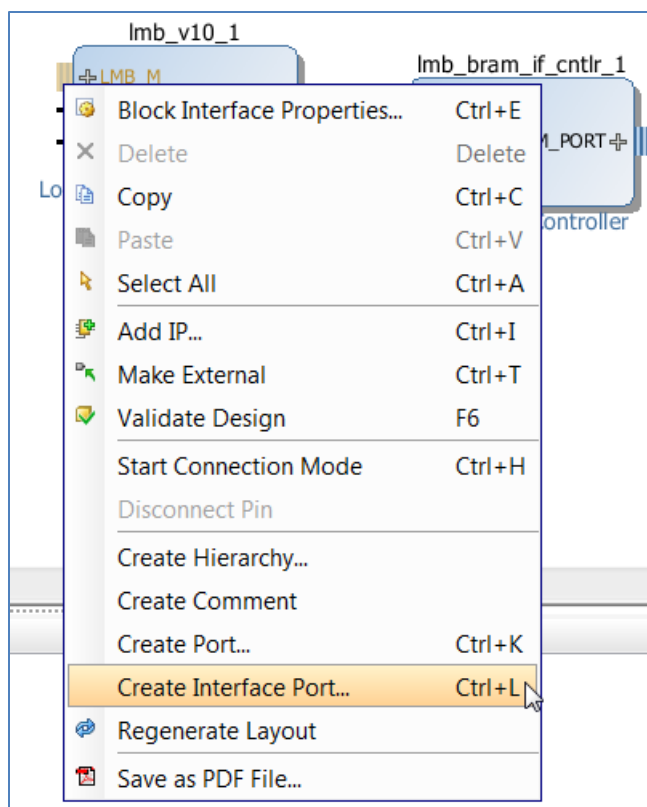


Figure 16: Creating an Interface Port

This command is used to create ports on the interfaces which are groupings of signals that share a common function. For example, the S_AXI is an interface port on several Xilinx IP. The command gives more control in terms of specifying the interface type and the mode (master/slave).

The Board Automation and Connection Automation Feature of IP Integrator

The Block Automation and Connection Automation features are provided in IP integrator to assist you in putting together a basic microprocessor system and/or connecting ports to external I/O ports. The Block Automation Feature is provided when a microprocessor such as the Zynq Processing System 7 or MicroBlaze processor is instantiated in the IP integrator block diagram. You click on **Run Block Automation**, as shown in the following figure, to get assistance with putting together a simple MicroBlaze System.

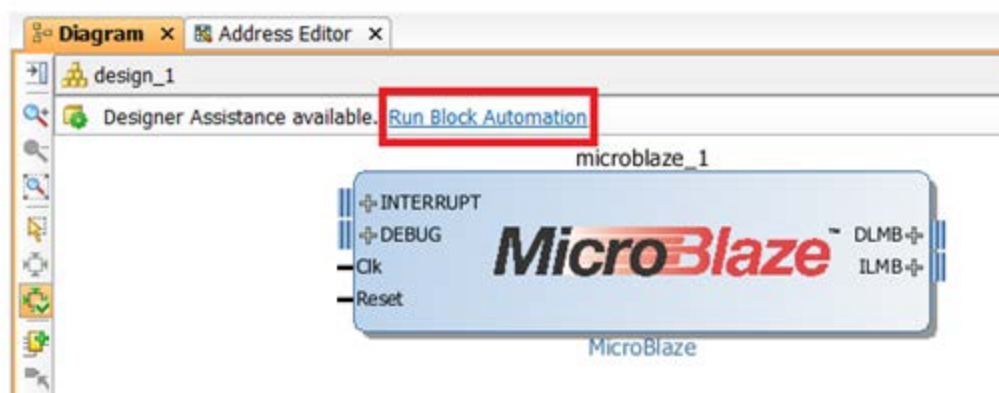


Figure 17: The Run Block Automation Feature

The **Run Block Automation** dialog box allows you to provide input about basic features that the microprocessor system needs.

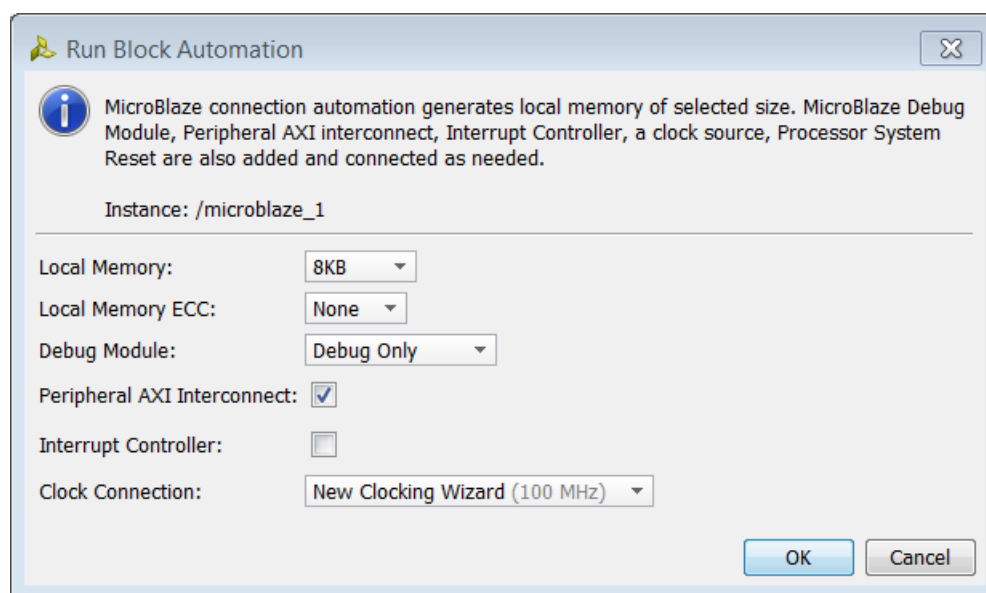


Figure 18: The Run Block Automation Dialog Box

Once you specify the necessary options, the Block Automation feature automatically creates a basic system as shown in the figure below.

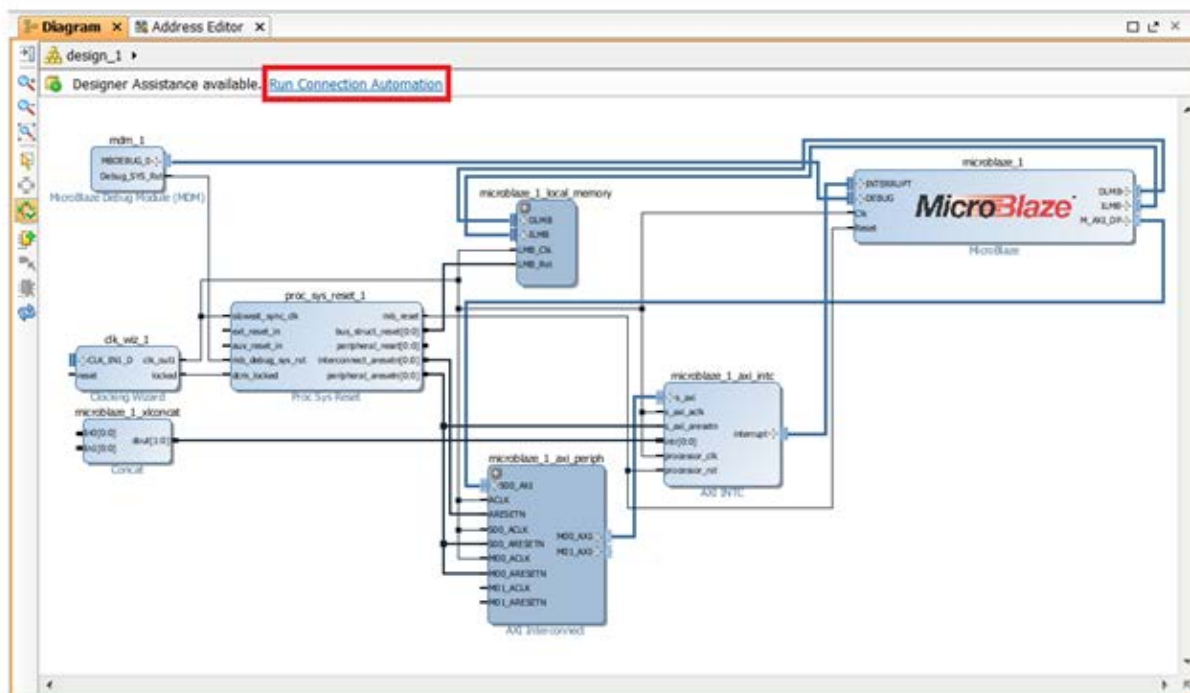


Figure 19: A Basic System Created by the Block Automation Feature

In this case, a basic MicroBlaze System is created that consists of a MicroBlaze Debug Module, a hierarchical block called the `microblaze_1_local_memory` that has the Local Memory Bus, the Local Memory Bus Controller and the Block Memory Generator, a Clocking Wizard, an AXI Interconnect and an AXI Interrupt Controller. Since the design is not connected to any external I/O at this point, IP integrator provides the Connection Automation feature as shown above by the highlighted area in the figure.

When you click on **Run Connection Automation** you'll get assistance in hooking interfaces and/or ports to external I/O ports.

As shown in the following figure, ports/interfaces that can use the Connection Automation feature are listed.

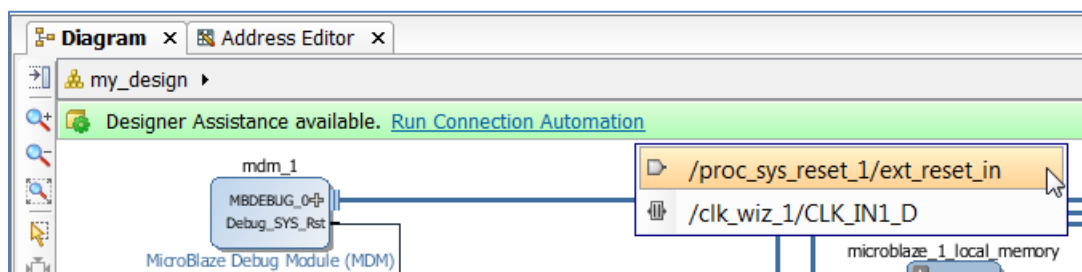


Figure 20: Listing the Ports and Interfaces the can use Connection Automation

For Xilinx's Target Reference Platforms or evaluation boards, the design knows the FPGA pins that are connected /used on the target boards. Based on that information, the IPI connection automation feature can assist the designer in tying the ports in the design to external ports. The IPI then creates the appropriate physical constraints and other I/O constraints required for the I/O port in question. In the design above, the Proc Sys Reset IP needs to be connected to an external reset port and the Clocking Wizard needs to be connected to an external clock source. When the `/proc_sys_reset_1/ext_reset_in` option is selected above then the dialog box shown in the following figure opens up.

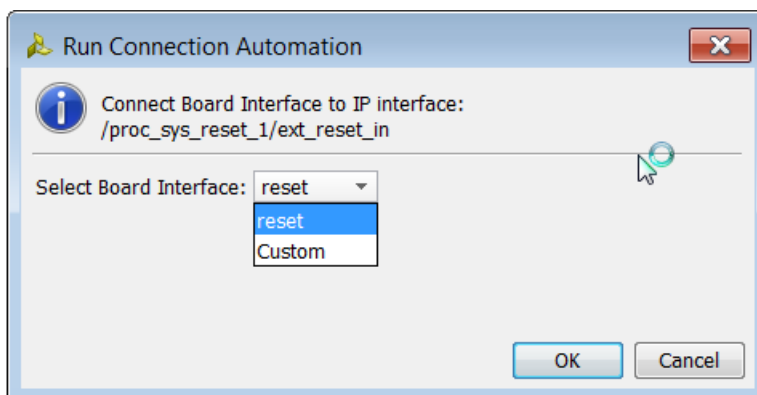


Figure 21: The Run Connection Automation Dialog Box

You can select the reset pin that already exists on the target board, KC705 in this case, or you can specify a custom reset pin to your design. Once specified, the reset pin is tied to the `ext_reset_in` pin of the Proc Sys Rst IP.



The diagram illustrates the hardware configuration of a MicroBlaze system. Key components include:

- mdm_1**: MicroBlaze Debug Module (MDM), connected to MBDEBUG, OPhi, Debug_Sys_Rst, and M_AXI_DP.
- microblaze_1_local_memory**: Local memory block containing DLMB, ILMB, MB_Clk, and MB_Rst.
- microblaze_1**: The main MicroBlaze processor core.
- proc_sys_reset_1**: Processor System Reset block, receiving mb_reset, bus_struct_reset(0:0), peripheral_reset(0:0), interconnect_arresetn(0:0), and peripheral_arresetn(0:0).
- clk_wiz_1**: Clocking Wizard, providing CLK_IN1_P, clk_out, reset, locked, dout[0:0], and in[0:0] signals.
- Concat**: Concatenation block combining dout[0:0] and in[0:0].
- microblaze_1_axi_periph**: AXI peripheral block with multiple ports including S00_AXI, S00_ACLK, S00_ARESETN, M00_ACLK, M00_ARESETN, M01_ACLK, and M01_ARESETN.
- microblaze_1_axi_intc**: AXI Interrupt Controller, featuring phi_s_axi, i_axi_ack, s_axi_aresetn, ver[0:0], processor_clk, processor_rst, and interrupt outputs.
- axi_gpio_1**: AXI GPIO block, highlighted with a red box, showing phi_s_axi, s_axi_ack, s_axi_aresetn, and gpio-phi outputs.

The connections show a complex interplay between the processor core, its local memory, various control and clocking blocks, and external peripherals like the debug module and GPIO controller.

Figure 23: Using Connection Automation to Show Potential Future Connections

When you click on **Run Connection Automation**, the following pop-up box will inform you that the slave AXI port of the GPIO can be connected to the MicroBlaze master. If there are multiple masters in the design, then you will have a choice to select between different masters.

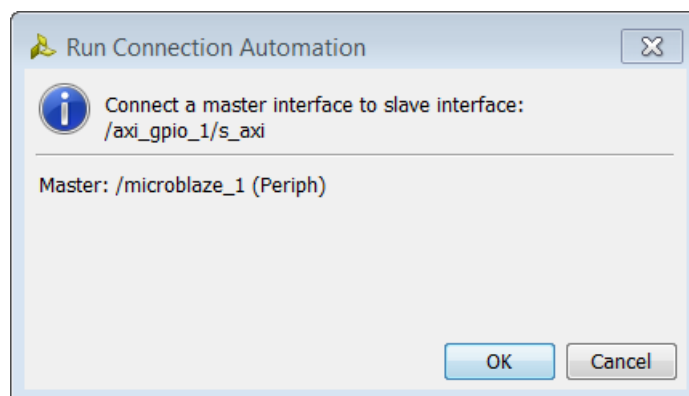


Figure 24: Connecting the Slave Interface s_axi to the MicroBlaze Master

When you click **OK** on the Run Connection Automation dialog box, the connections are made and highlighted as shown in figure below.

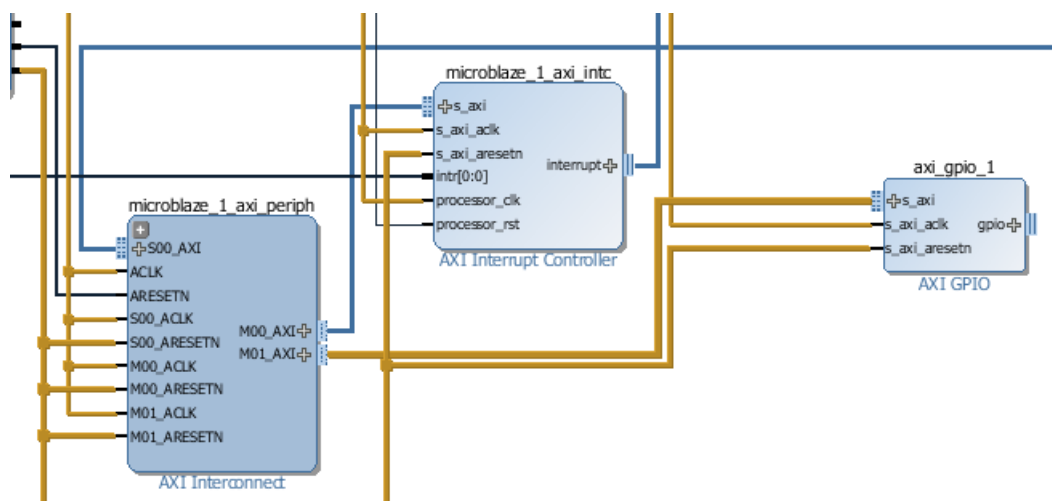


Figure 25: Master/Slave Connections are Made and Highlighted

In case a Xilinx Targeted Reference Platform is used, then further assistance can be available in terms of I/O port connections. Refer to the figure below.

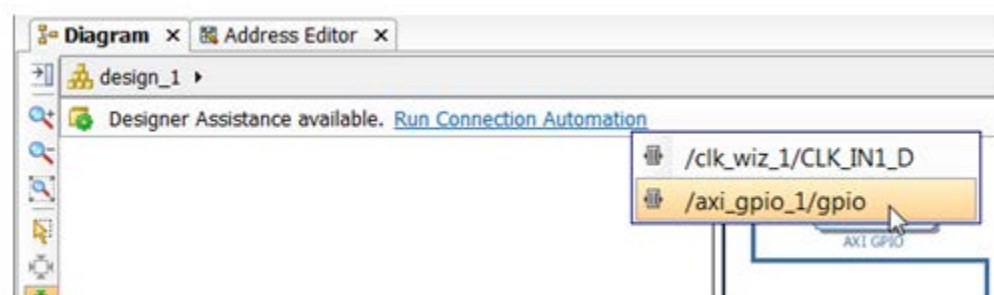


Figure 26: Further Assistance Available for Xilinx Targeted Reference Platforms

When you select the **gpio** port, the following options are presented to the user based on the information available about the board.

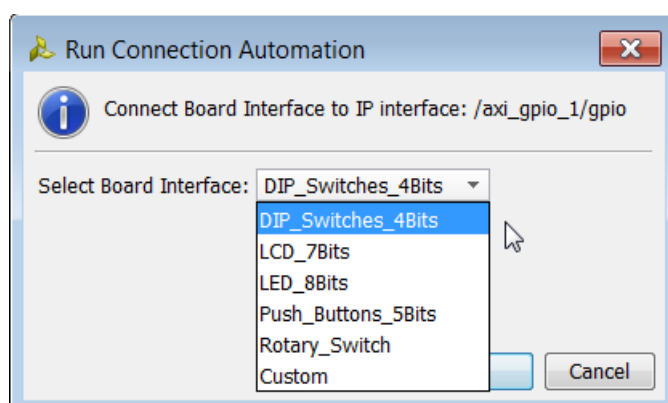


Figure 27: The Options Presented After a Port is Selected

In this case, six different choices are presented. The **gpio** port can be connected to the **Dip Switches** that are 4-bits, connected to the **LCD** that are 7-bits, connected to the **LEDs** that are 8-bits, connected to the 5-bits of Push Buttons, connected to the Rotary Switch on the board or can be connected to a custom interface. Selecting any one of the choice above will connect the **gpio** port to the existing connections on the board.

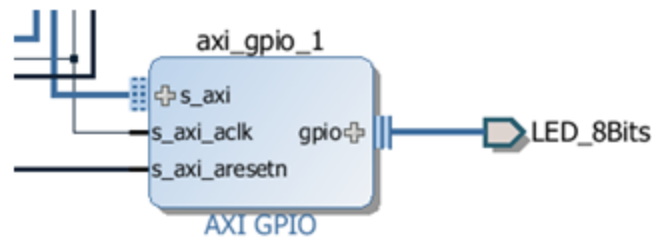


Figure 28: Connecting Board I/O Elements to the Design

Re-arranging the IP Blocks

IP blocks placed on the canvas can be re-arranged to get a better view of the block diagram. To arrange a completed diagram or a diagram in progress, you can click the **Regenerate** button.

It is possible to move blocks manually by clicking on a block, holding the left-mouse button down, and moving the block or you can use the arrow keys. The diagram only allows specific column locations, indicated by the dark gray vertical bars that appear when moving a block. A grid appears on the diagram when moving blocks, which assists you in making better block and pin alignments.

Cutting and Pasting

You can use Ctrl-C and Ctrl-V to copy and paste blocks in a diagram.

Creating Hierarchies

As shown in the figure below, you can create a hierarchical block in a diagram by using Ctrl-click to select the desired IP blocks, right-click and then select **Create Hierarchy**.

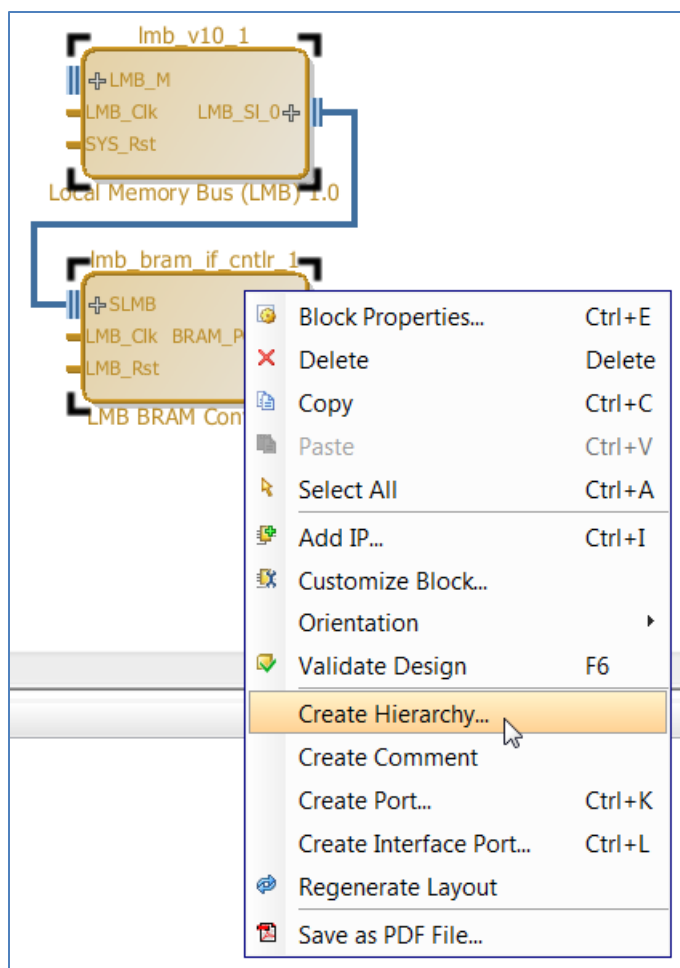


Figure 29: Creating a Hierarchal Block Diagram

The new level of hierarchy is created containing the selected blocks. It is also possible to create an empty level of hierarchy, and later drag existing IP blocks into that hierarchical block. Hierarchies can be expanded when you click on the + sign in the upper-left corner of the block. You can transverse levels of hierarchy in a diagram using the Explorer type path information displayed in the upper-left corner of the IP integrator diagram. Selecting **Ungroup Instances** allows you to move the IP blocks out of a hierarchical grouping.

Clicking on **Create Hierarchy** pops-up the Create Hierarchy dialog box, as shown below, where you can specify the name of the hierarchy.

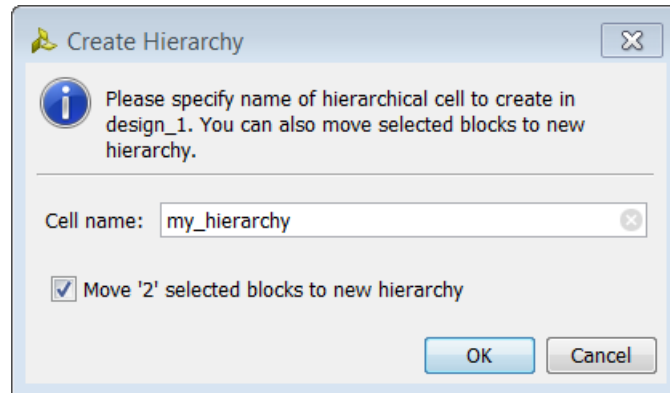


Figure 30: The Create Hierarchy Dialog Box

This action groups the two blocks under one block, as shown below. You can click on the + sign of the hierarchy to view the components underneath. Click on the – sign on the expanded hierarchy to collapse it back to the grouped form.

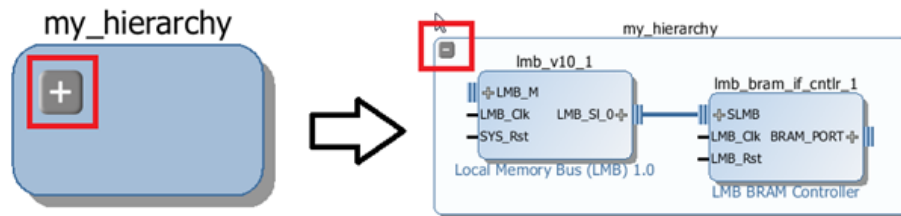


Figure 31: Grouping Two Blocks into One Block

Creating a Memory Map

To generate the address map for this design, you can click on the **Address Editor** tab above the diagram. Typically, the addresses are mapped automatically as you instantiate slaves in your block diagram. However, you can also click the **Auto Assign Address** button (the bottom button on the left side). If you generate the RTL from an IP integrator diagram without first generating addresses, a prompt will appear allowing the selection of automatically assigning addresses. You can also manually set Addresses by entering values in the **Offset Address** and **Range** columns.

Note: The Address Editor tab only appears if the diagram contains an IP block that functions as a bus master (such as the MicroBlaze processor in the following diagram).

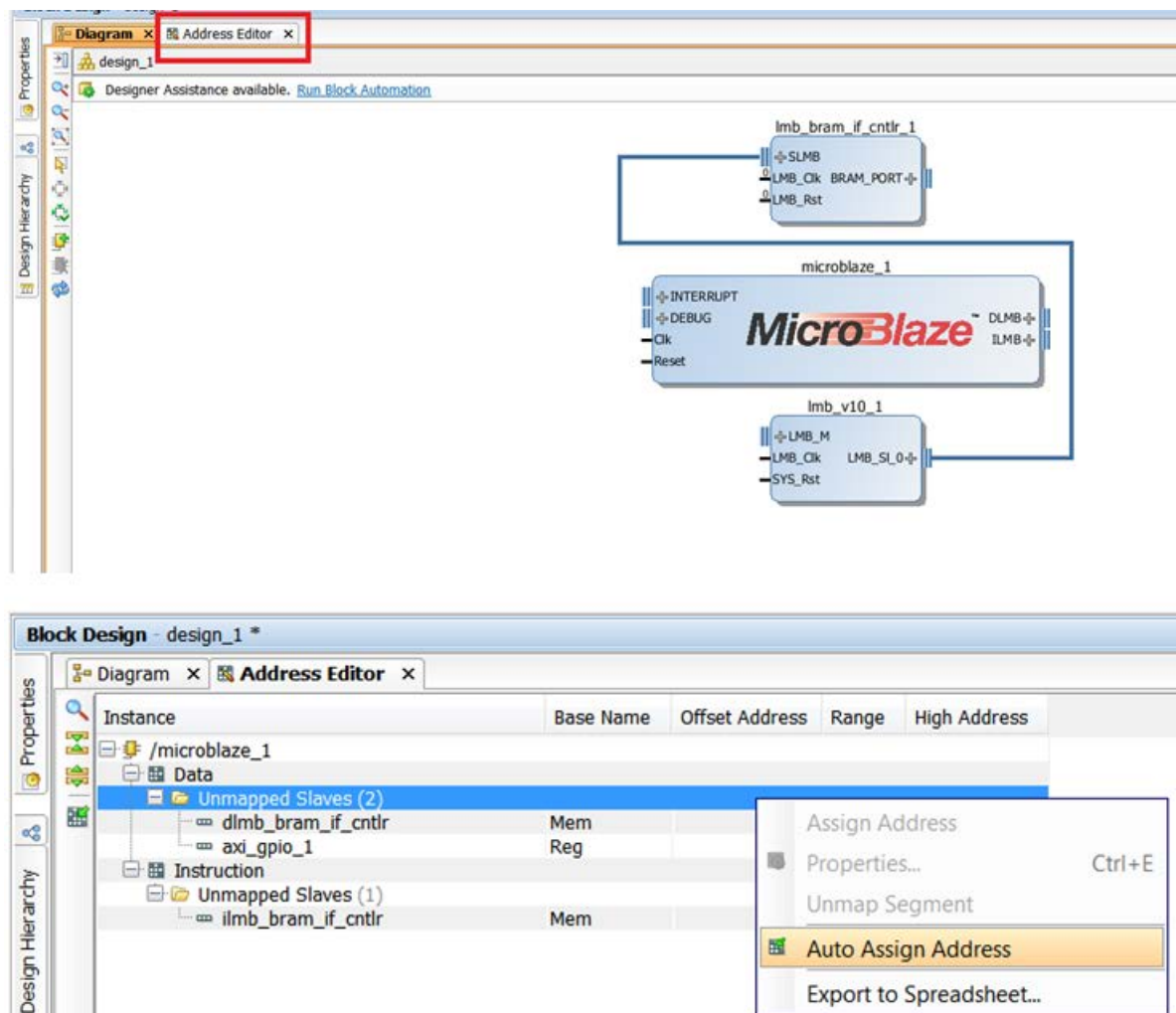


Figure 32: Using the Address Editor Tab

Running Design Rule Checks

IP integrator runs basic design rule checks in real time as the design is being put together. However, there is a potential for something to go wrong during design creation. As an example, the frequency on a clock pin may not be set right. As shown in the following figure, you can run a comprehensive design check on the design by clicking on **Validate Design**.

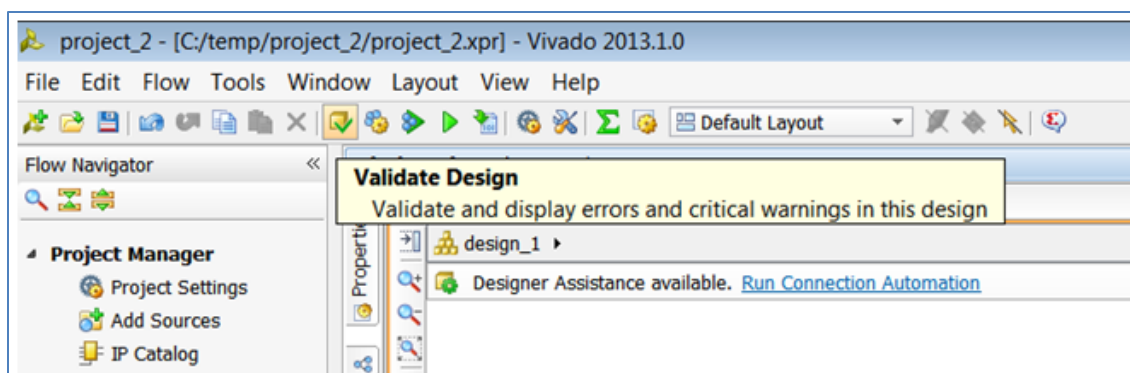


Figure 33: Validating the Design

If the design is free of Warnings and/or Errors, a pop-dialog box such as that shown in the figure below is displayed after running **Validate Design**.

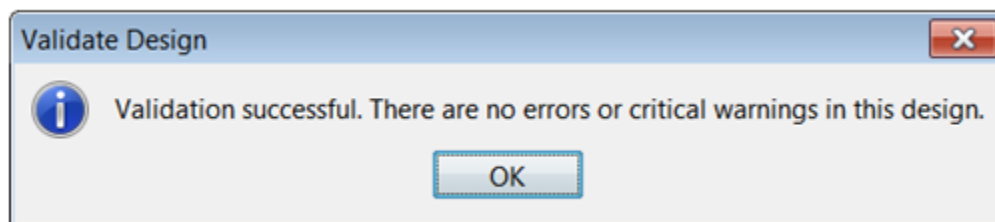


Figure 34: Successful Validation Message

Integrating the Block Diagram into a Top-Level Design

Once the block diagram is complete and the design is validated there are two more steps required to complete the design. First, the output products must be generated. This is when the source files and the appropriate constraints for all the IP will be generated and made available in the Vivado Sources pane. Depending upon the target language selected during project creation, appropriate files will be generated. If the source files for a particular IP cannot be generated in the target language specified, then a message stating so will be displayed in the console. To generate output products, in the Vivado sources pane, you right click on the block diagram, as shown in the following figure, and select **Generate Output Products**.

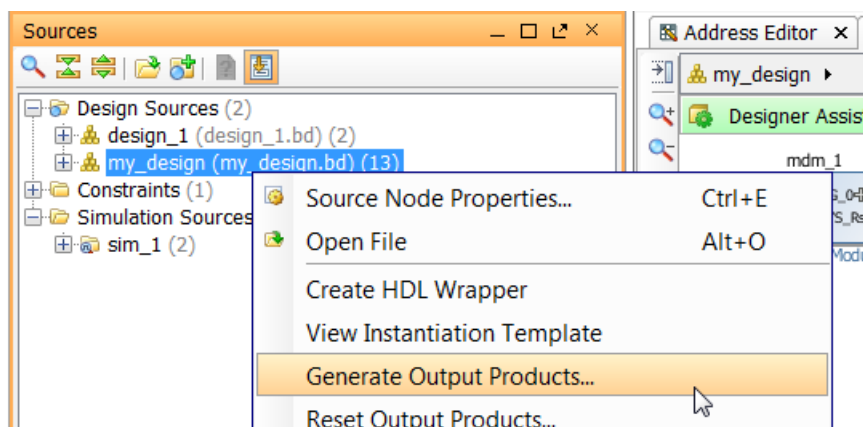


Figure 35: Generating Output Products

An IP integrator block diagram can be integrated into a higher-level design or it can be the highest level in the design hierarchy. To integrate the IP integrator design into a higher-level design, simply instantiate the design in the top-level HDL file.

A higher-level instantiation of the block diagram can also be done by selecting the block diagram in the **Sources** pane in Vivado and selecting **Create HDL Wrapper** (shown below). This will generate a top-level HDL file for the IP integrator sub-system. At this point, you are ready to take the design through elaboration, synthesis and implementation.

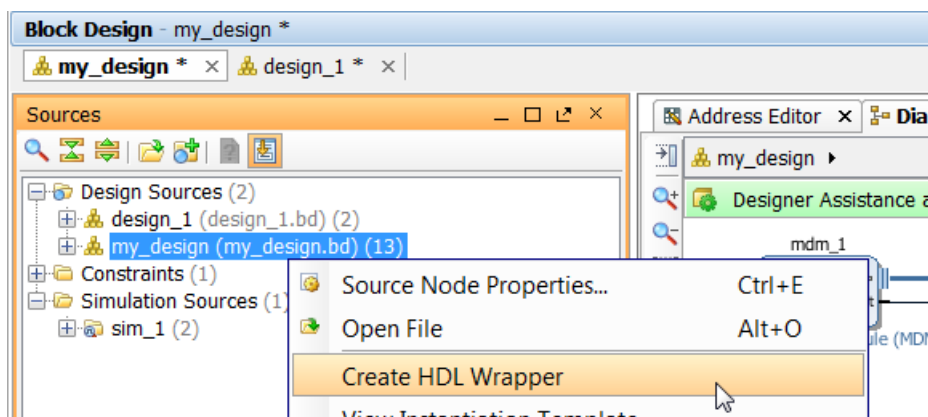


Figure 36: Creating an HDL Wrapper

Exporting Hardware Definition to SDK

If you are working on a design that contains processors such as MicroBlaze or Zynq7, you can export the hardware definition of your project to the Software Development Kit (SDK). This action exports the necessary xml files needed for SDK to understand the IP used in the design and also their memory mapping from the processor's perspective. To export the hardware definition, you need to take the design through implementation and possibly generate the bitstream. Bitstream generation is not required if a Zynq7 based design has nothing in the Processing Logic fabric of the FPGA, which is seldom the case.

To export hardware to SDK, from the menu select **File > Export Hardware for SDK**.

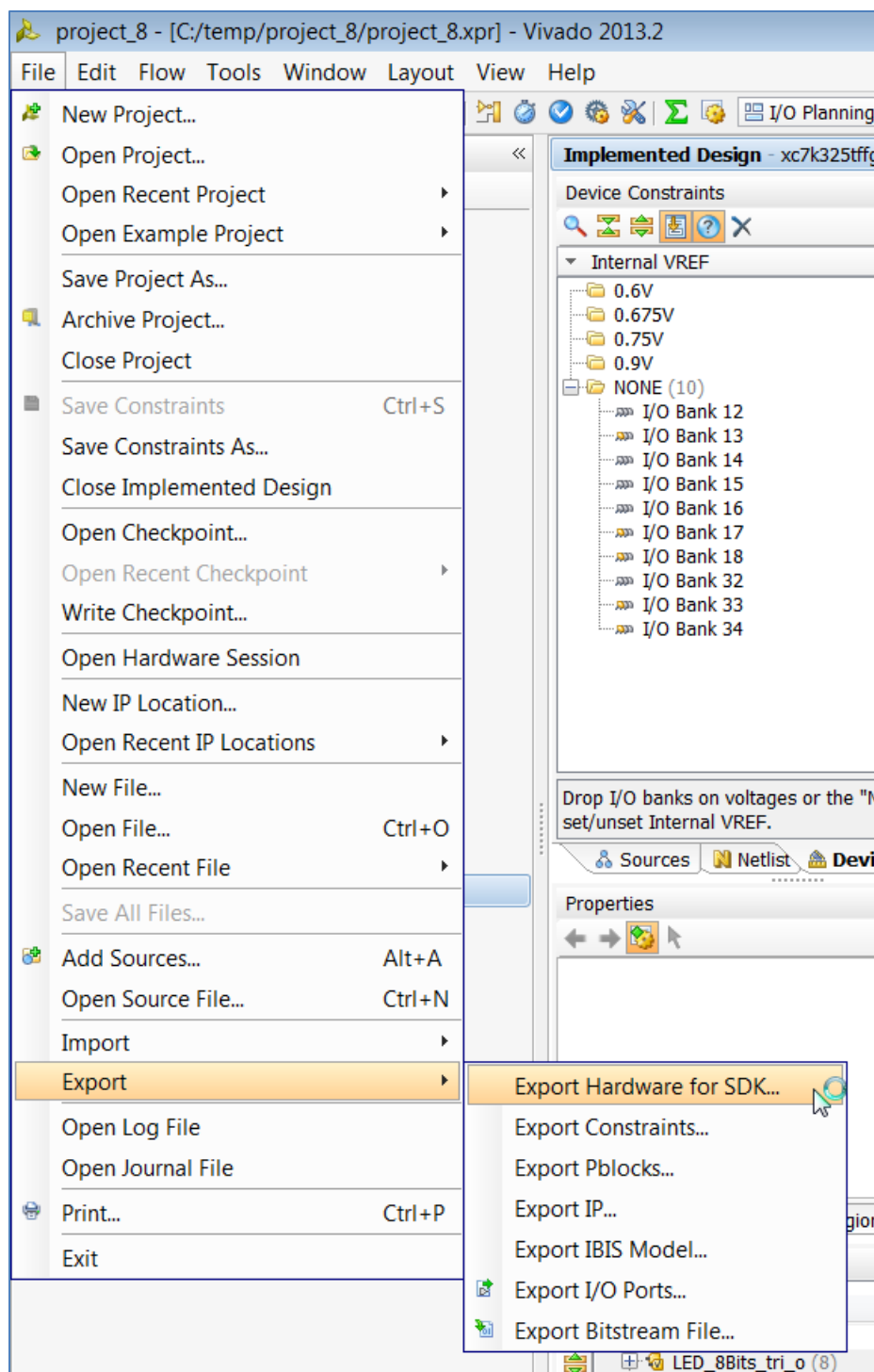


Figure 337: Exporting the Hardware Definition for a Project

The Export Hardware for SDK dialog box opens.

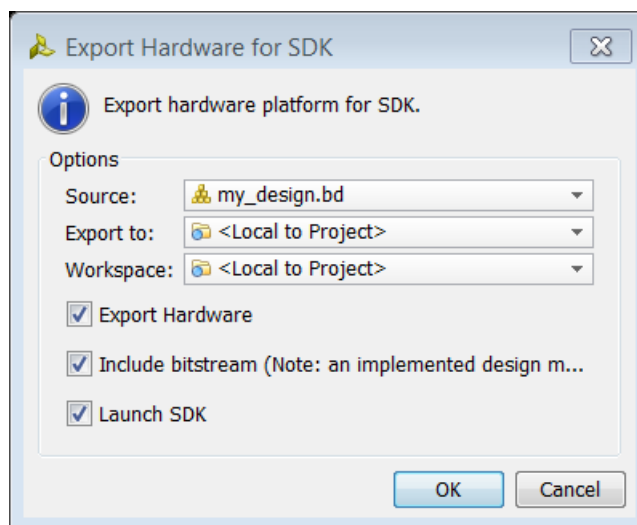


Figure 38: Exporting the Hardware Definition for a Project

There are three options that are presented as can be seen from the above figure. Checking the first option only, just creates the XML files needed for SDK. Checking the second option includes the bitstream as a part of the export and finally, checking the third option also launches SDK from Vivado and opens up in the right “workspace”.

Note: For Export Hardware to work, a block diagram needs to be open. Export Hardware works only on active block diagrams. Additionally, an implemented design must be open in order for the “Include bitstream” option.

In a project based flow, the hardware is exported at the following location:

project_name/project_name.sdk/SDK/SDK_Export/hw

Once SDK launches, custom application project can be created in it using the hardware definitions exported. SDK creates the necessary drivers and board support package for the target hardware.

Packaging the Block Diagram

Often times it happens that you create an IP Integrator design, implement it and test it on a target hardware. If you are satisfied with the functionality, you may want to “package” it and convert it into an IP that can be reused in another design. When you package a design, it gets converted into an IP and is available for you in the IP catalog. You can instantiate that IP as a part of a different design.

To package a block diagram, you right-click on the block diagram in the Sources window in the Vivado IDE and select **Package Block Design**.

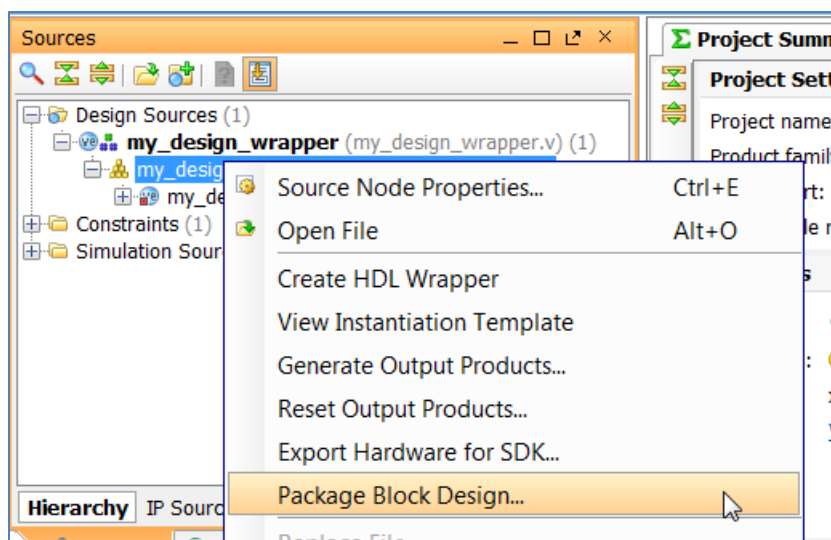


Figure 39: Selecting Package Block Design

The Package Block Design dialog box opens up stating that the block diagram needs to be generated. Click **Yes**.

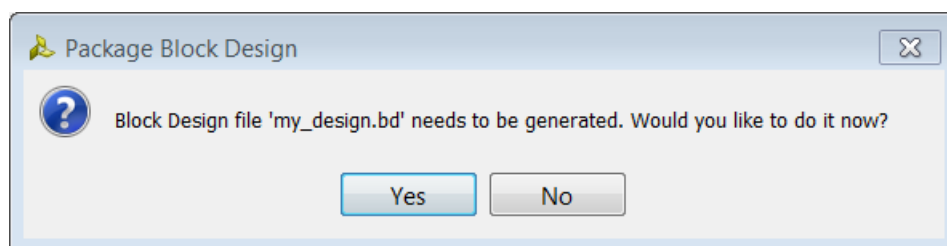


Figure 40: Package Block Design Dialog Box

When you click **Yes** on the above dialog box, the output products for all the IP instantiated in the block diagram are generated and the Manage Output Products dialog box opens.

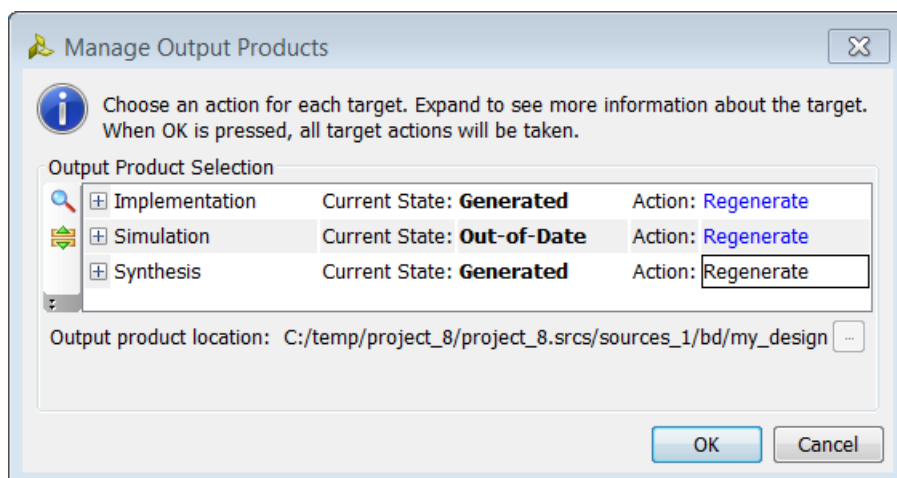


Figure 41: Manage Output Products Dialog Box

If you need to, you can change the **Action** field in the Manage Output Products dialog box to **Regenerate** and click **OK**.

The Package IP window opens up.

Package IP - my_design

IP Identification

0 errors 0 warnings 0 info messages
Fill in and modify the information fields below that will be used to identify your IP. Set the categories in which your IP will appear in the IP Catalog by modifying the Taxonomy.

Identification

Vendor: xilinx.com
Library: ip
Name: my_design
Version: 1.0
Display name: my_design
Description: my_design
Vendor display name:
Company url: http://www.xilinx.com
Categories: /Packaged_BlockDiagram_Designs
Root directory: c:/temp/project_8/project_8.srcs/sources_1/bd/my_design
Xml file name: c:/temp/project_8/project_8.srcs/sources_1/bd/my_design/comp
☐ Show advanced information

Figure 42: The Package IP Window

You select the **Review and Package** menu item in the dialog box and then click on **Package IP**. Once packaged, the IP is available in the IP Integrator catalog as shown below.

Search: my_ (1 match)

Name	Version	AXI4
my_design	1.0	

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

Figure 43: The Package IP Window

The newly packaged design is also available in the Vivado IP Catalog under the category **Packaged BlockDiagram Designs**. This category can be changed and renamed while packaging the block diagram.

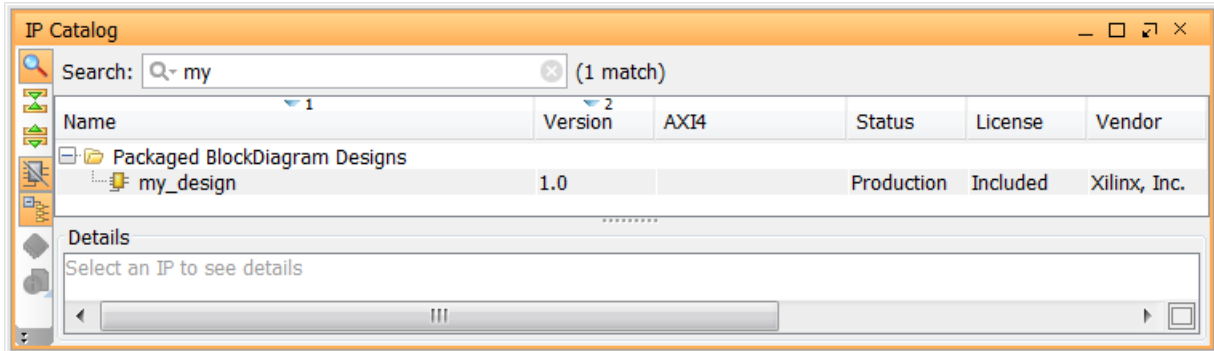


Figure 44: The Packaged BlockDiagram Designs Catagory

Adding and Associating an ELF File to an Embedded Design

In a microprocessor-based design such as a MicroBlaze- or Zynq7- based design, an ELF file generated in the SDK (or in other software development tool) can be imported and associated with a block design in the Vivado IDE. A bitstream can then be programmed along with the ELF file from the Vivado IDE and run on target hardware.

1. To accomplish this task, you add the ELF file by selecting and right-clicking Design Sources in the Sources window and then selecting **Add Sources** as shown below.

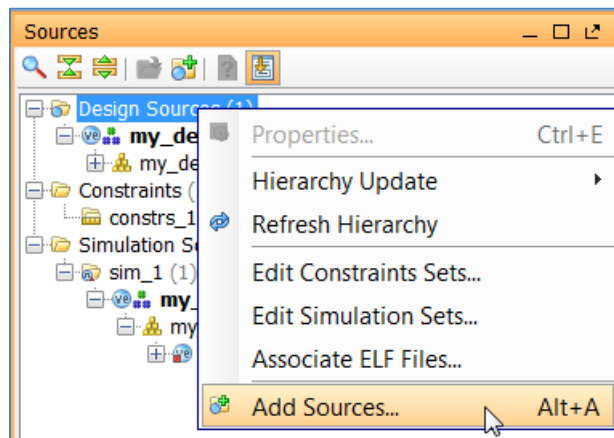


Figure 45: Selecting Add Sources

2. The Add Sources dialog box opens. **Add or Create Design Sources** is selected by default. Click **Next**.
3. In the Add or Create Design Sources dialog box, click on **Add Files**.

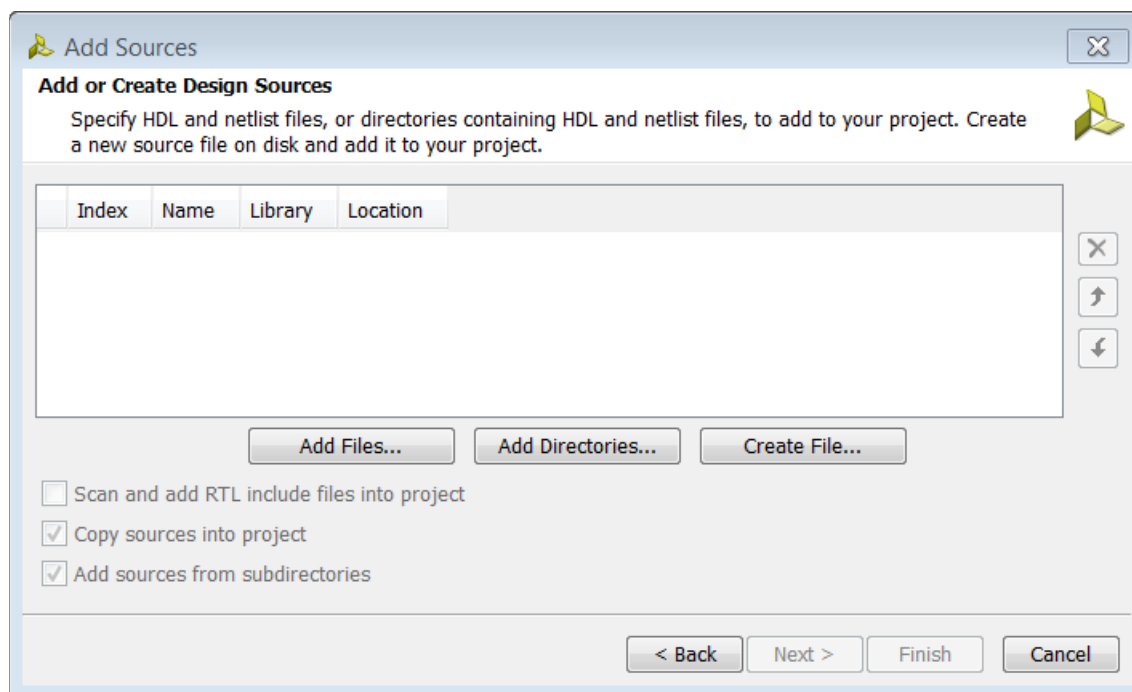


Figure 46: Add Sources Dialog Box

4. The Add Source Files dialog box opens. Navigate to the ELF file, select it and click **OK**.

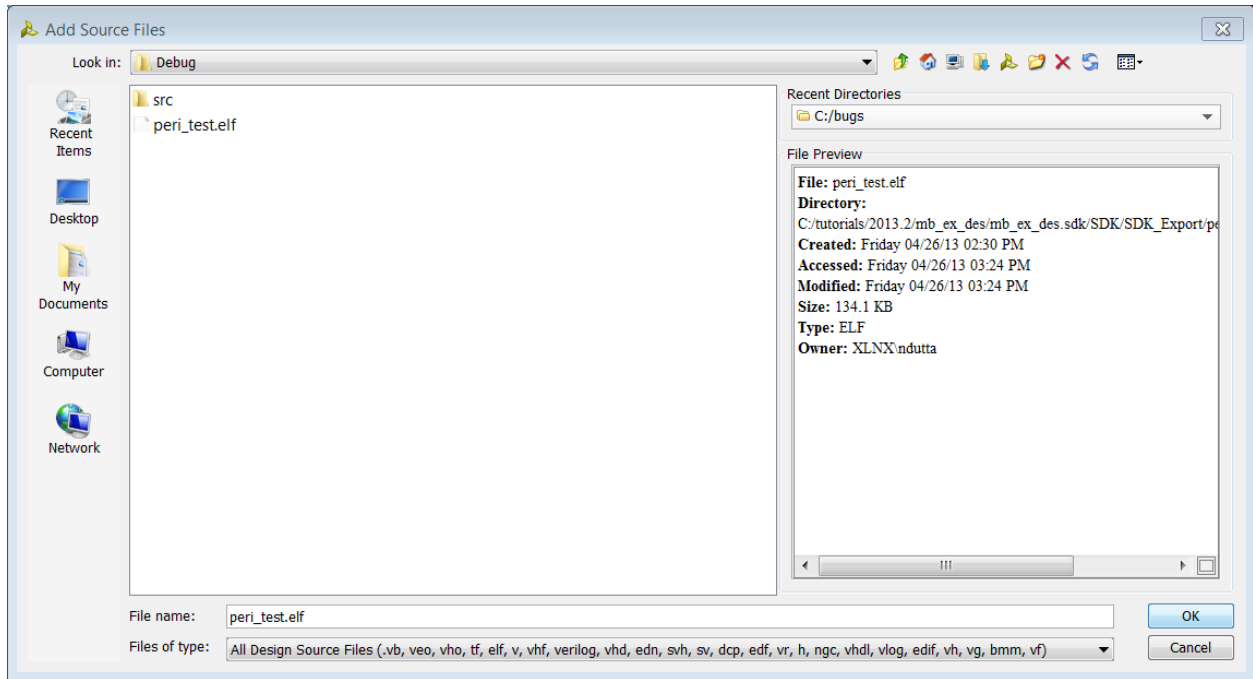


Figure 47: Add Sources Files Dialog Box

5. In the Add or Create Sources dialog box, you can see the ELF file added to the project. Depending on your preference, you can either copy the ELF file into the project by checking **Copy sources into project** or leave that option unchecked if you want to work with the original ELF file. Click **Finish**.
6. Under the Sources window, you will see the ELF file added under the ELF folder.

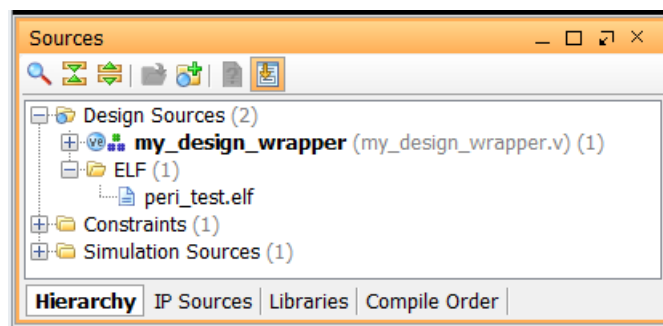


Figure 48: Sources Window with ELF file Displayed

7. Next, you associate that ELF file with the microprocessor design in question. To do this, select and right-click on the Design Sources folder in the Sources window and select **Associate ELF Files**.

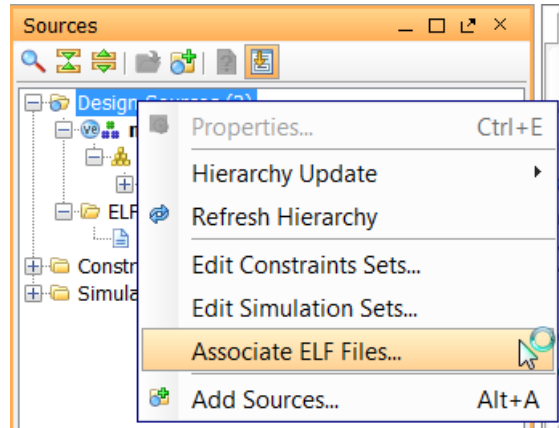


Figure 49: Selecting Associate ELF Files

8. You can add an ELF for synthesis as well as simulation. Click on the appropriate browse icon (Under Design Sources or Simulation Sources) and browse to the newly added ELF file to the design.

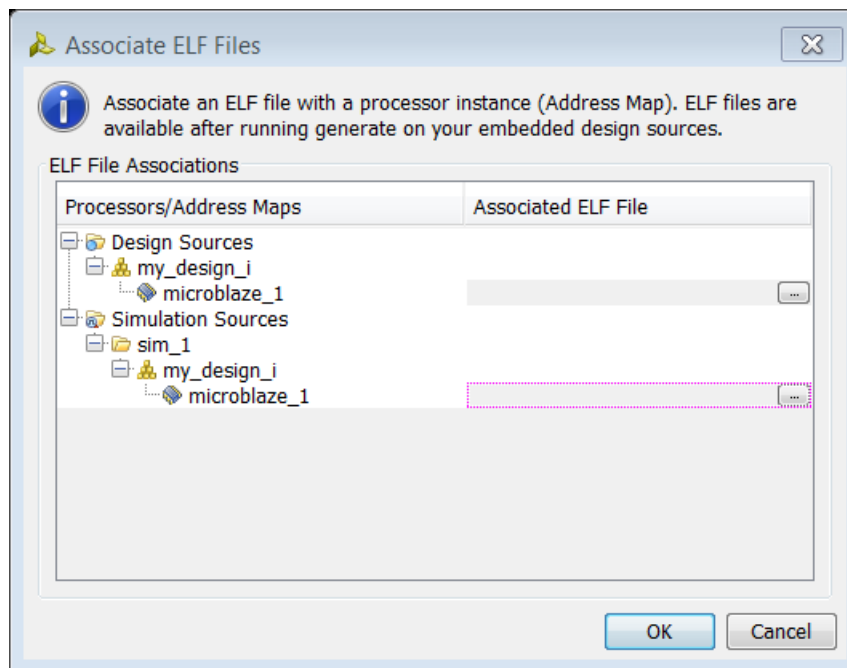


Figure 50: Associating ELF Files with a Microprocessor

The Associate ELF file dialog box opens. Highlight the file, as shown below, and click **OK**.

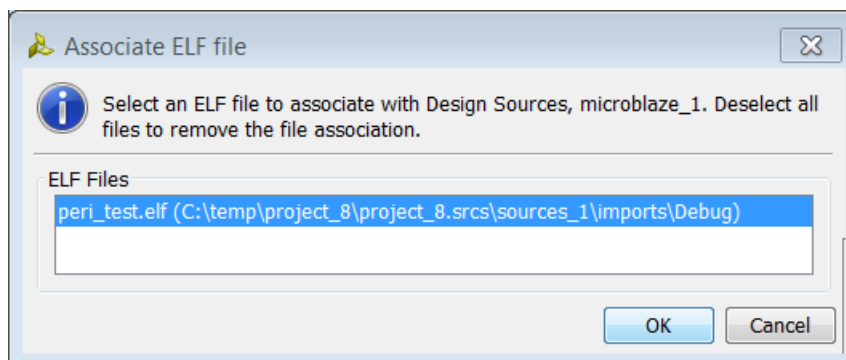


Figure 51: Highlight the ELF File to Associate

Makes sure that the ELF file is populated in the Associated ELF File Column and click **OK**.

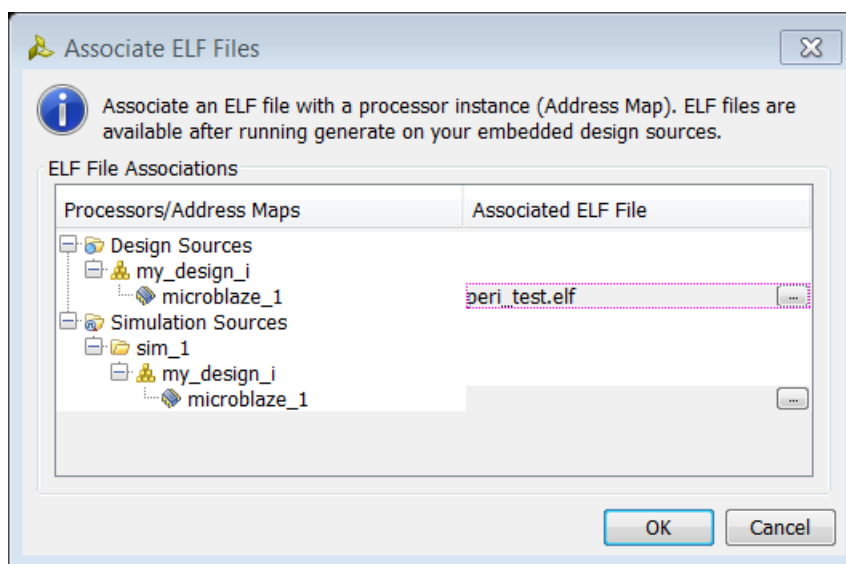


Figure 52: Making Sure the ELF File is Populated

