

Adding Custom IP to the System

Introduction

This lab guides you through the process of creating and adding a custom peripheral to a processor system by using the Vivado IP Packager. You will create an AXI4Lite interface peripheral.

Objectives

After completing this lab, you will be able to:

- Use the IP Packager feature of Vivado to create a custom peripheral
- Modify the functionality of the IP
- Add the custom peripheral to your design
- Add pin location constraints
- Add block memory to the system

Procedure

This lab is separated into steps that consist of general overview statements that provide information on the detailed instructions that follow. Follow these detailed instructions to progress through the lab.

This lab comprises 4 primary steps: You will use a peripheral template to create a peripheral, Package the IP using IP Packager, import, add and connect the IP in the design, add the Block RAM (BRAM) Memory and generate the bitstream.

Design Description

You will extend the Lab 2 hardware design by creating and adding an AXI peripheral (refer to LED_IP in **Figure 1**) to the system, and connecting it to the LEDs on the ZedBoard. You will use the IP Packager to generate the custom IP. Next, you will connect the peripheral to the system and add pin location constraints to connect the LED display controller peripheral to the on-board LED display. Finally, you will add BRAM Controller and BRAM before generating the bitstream.

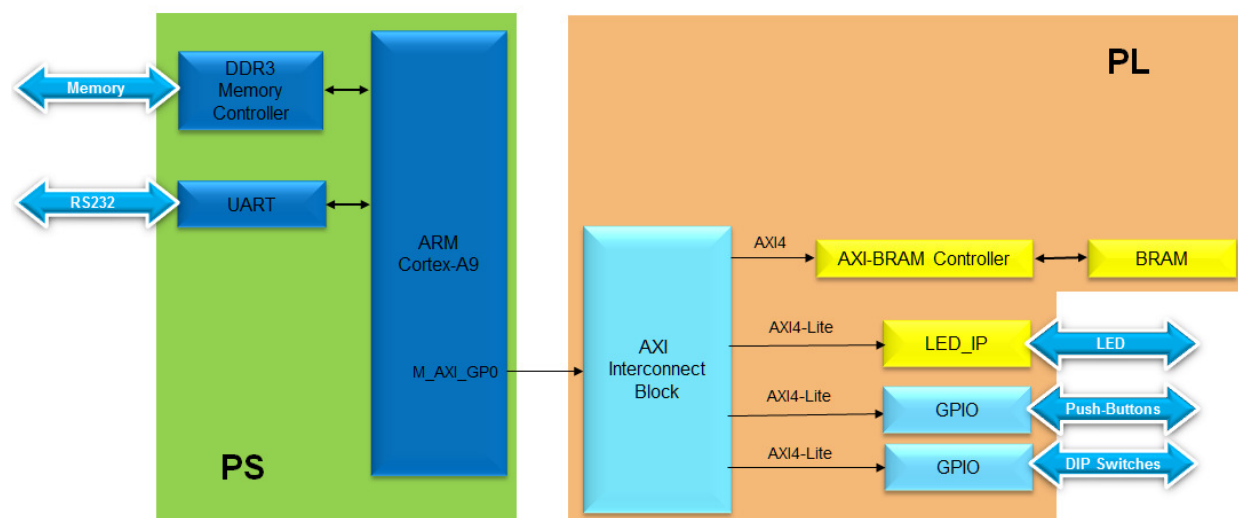
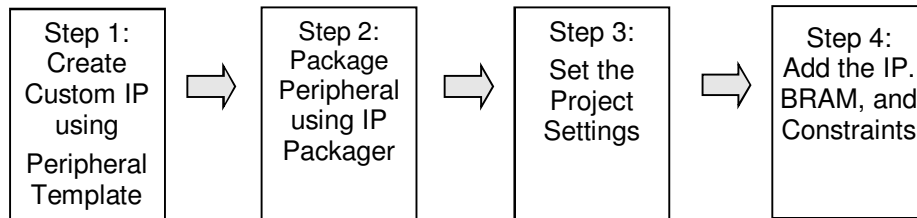


Figure 1 Design Updated from Previous Lab

General Flow for this Lab



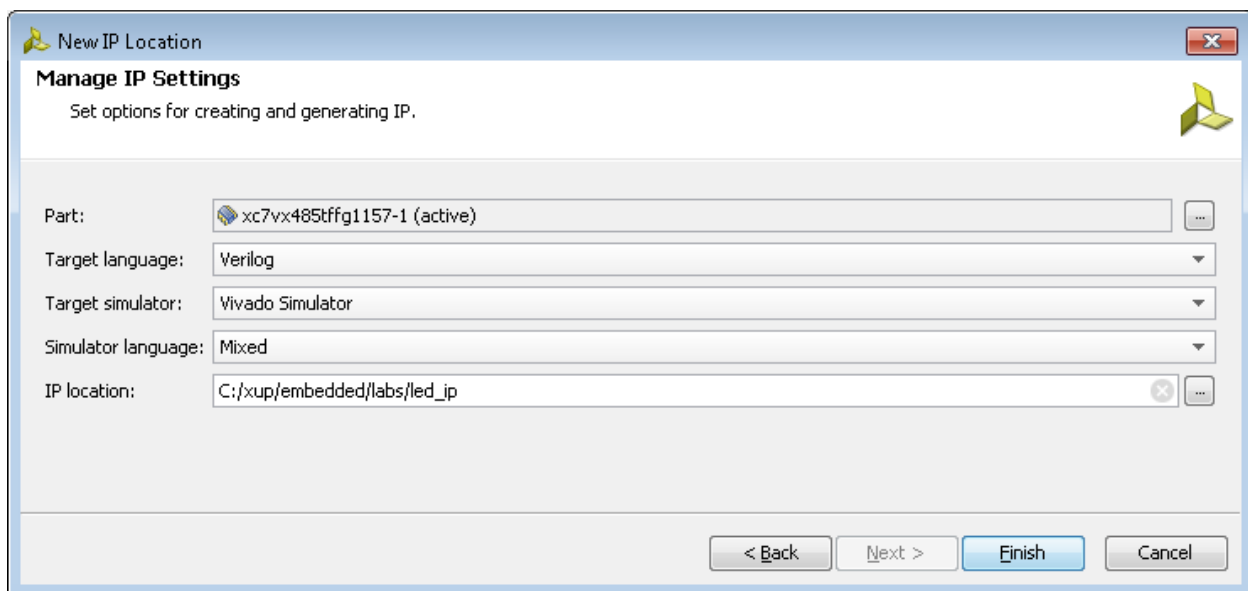
Create a Custom IP using the Create and Package IP Wizard Step 1

1-1. Use the provided `axi_lite` slave peripheral template and the custom IP source code to create a custom IP.

1-1-1. Open Vivado by selecting **Start > All Programs > Xilinx Design Tools > Vivado 2013.3 > Vivado 2013.3**

1-1-2. Click **Manage IP** and select *New IP Location* and click **Next** in the *New IP Location* window

1-1-3. Select **Verilog** as the *Target Language*, **Mixed** as the *Simulator language*, and for *IP location*, type `C:\xup\embedded\labs\led_ip` and click **Finish** (leave other settings as defaults)



A Virtex 7 part is chosen for this project, but later compatibility for other devices will be added to the packaged IP later.

1-1-4. Click **OK** to create the `led_ip` directory.

1-2. Run the Create and Package IP Wizard

1-2-1. Select *Tools > Create and Package IP*

1-2-2. In the window, click **Next**

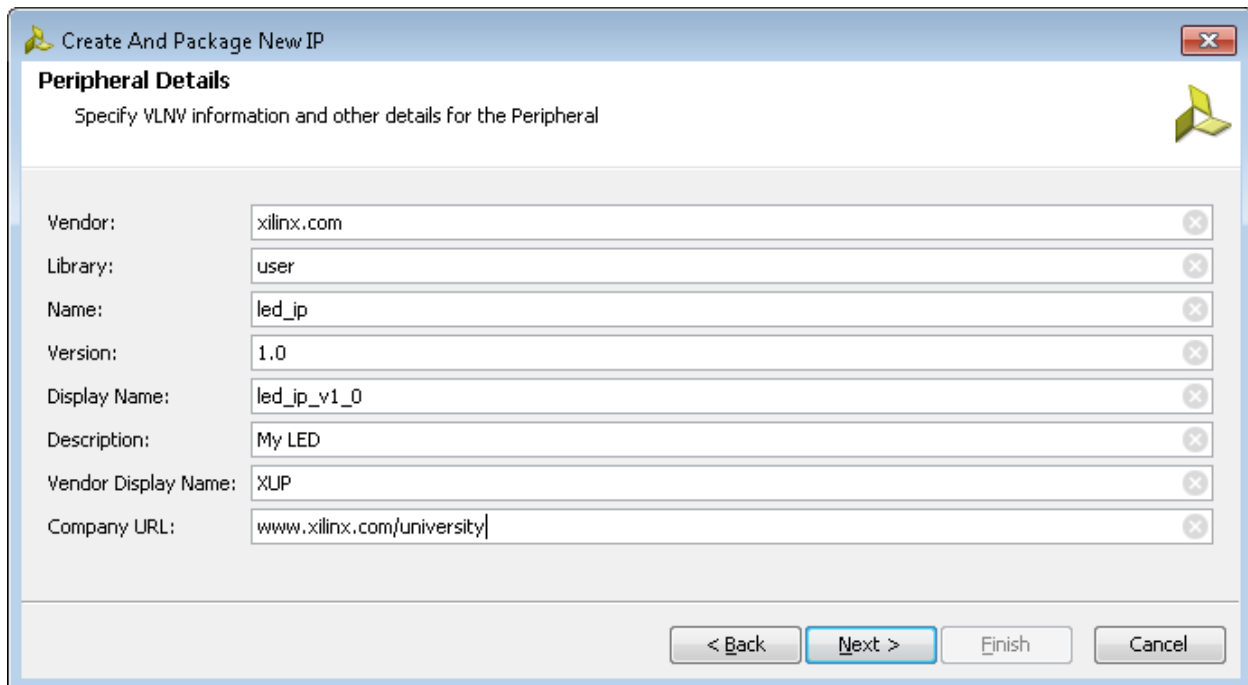
1-2-3. Select *Create New AXI4 Peripheral*, specify the *IP Definition location* as *C:/xup/embedded/labs/led_ip* and click **Next**

1-2-4. Fill in the details for the IP

Name: **led_ip**

Display Name: **led_ip_v1_0**

(Fill in a description, Vendor Name, and URL)



Create And Package New IP

Peripheral Details
Specify VLNW information and other details for the Peripheral

Vendor: xilinx.com

Library: user

Name: led_ip

Version: 1.0

Display Name: led_ip_v1_0

Description: My LED

Vendor Display Name: XUP

Company URL: www.xilinx.com/university

< Back **Next >** Finish Cancel

1-2-5. Click **Next**

1-2-6. Change the Name of the interface to **S_AXI**

1-2-7. Leave the other settings as default and click **Next** (Lite interface, Slave mode, Data Width 32, Registers 4)

Create And Package New IP

Add Interface
Add AXI4 interfaces supported by your peripheral

Add Interface

S_AXI X

Name: S_AXI

Interface Type: Lite

Interface Mode: Slave

Data Width(bits): 32

Number of Registers: 4 [4..512]

Interface types

- ☒ Lite : Simpler, non-burst control register style interface
- ☐ Full : Burst Capable, high-throughput memory mapped interface
- ☐ Stream : Burst Capable, high-throughput streaming interface

Note: For more details on AXI4 specification [click here](#)

< Back Next > Finish Cancel

1-2-8. Select *Generate Drivers* and click **Next**

1-2-9. Select *Add IP to Catalog* and open IP in editing session selected and click **Finish**

1-2-10. In the sources panel, double-click the **led_ip_v1_0.v** file and scroll down to line 15 where last port is defined.

1-2-11. Add the line:

```
output wire [7:0] LED,
```

(Notice the extra comma needed at the end of line)

```
14 // Users to add ports here
15 output wire [7:0] LED,
16 // User ports ends
```

1-2-12. Insert the following at line ~48:

```
. LED(LED),
```

```

43 // Instantiation of Axi Bus Interface S_AXI
44 led_ip_v1_0_S_AXI # (
45     .C_S_AXI_DATA_WIDTH(C_S_AXI_DATA_WIDTH),
46     .C_S_AXI_ADDR_WIDTH(C_S_AXI_ADDR_WIDTH)
47 ) led_ip_v1_0_S_AXI_inst (
48     .LED(LED),
49     .S_AXI_ACLK(s_axi_aclk),

```

1-2-13. Save the file by selecting **File > Save File**

1-2-14. Expand *led_ip_v1_0* in the sources view if necessary, and open **led_ip_v1_0_S_AXI.v**

1-2-15. Add the LED port to this file too, at line 15

```

14     // Users to add ports here
15     output wire [7:0] LED,
16     // User ports ends

```

1-2-16. Scroll down to ~line 397 and insert the following code to instantiate the user logic for the LED IP

(This code can be typed directly, or copied from the *user_logic_instantiation.txt* file in the lab3 source folder.)

```

396     // Add user logic here
397     lab3_user_logic U1(
398     .S_AXI_ACLK(S_AXI_ACLK),
399     .slv_reg_wren(slv_reg_wren),
400     .axi_awaddr(axi_awaddr[C_S_AXI_ADDR_WIDTH-1:ADDR_LSB]),
401     .S_AXI_WDATA(S_AXI_WDATA),
402     .S_AXI_ARESETN(S_AXI_ARESETN),
403     .LED(LED)
404     );
405     // User logic ends

```

Check all the signals that are being connected and where they originate.

1-2-17. Save the file by selecting **File > Save File**

1-2-18. Click on the *Add Sources* in the Flow Navigator pane, select *Add or Create Design Sources*, browse to *c:\xup\embedded\sources\lab3*, select the **lab3_user_logic.v** file and click **OK**, and then click **Finish** to add the file.

Check the contents of this file to understand the logic that is being implemented. Notice the formed hierarchy.

1-2-19. Click on the **Package IP – led_ip** tab

- 1-2-20. For the IP to appear in the IP catalog in particular categories, the IP must be configured to be part of those categories. To change which categories the IP will appear in the IP catalog click the browse box on the *Categories* line. This opens the Choose IP Categories window
- 1-2-21. For the exercise purpose, uncheck the **AXI Peripheral** box and check the **Basic Elements** and click **OK**.

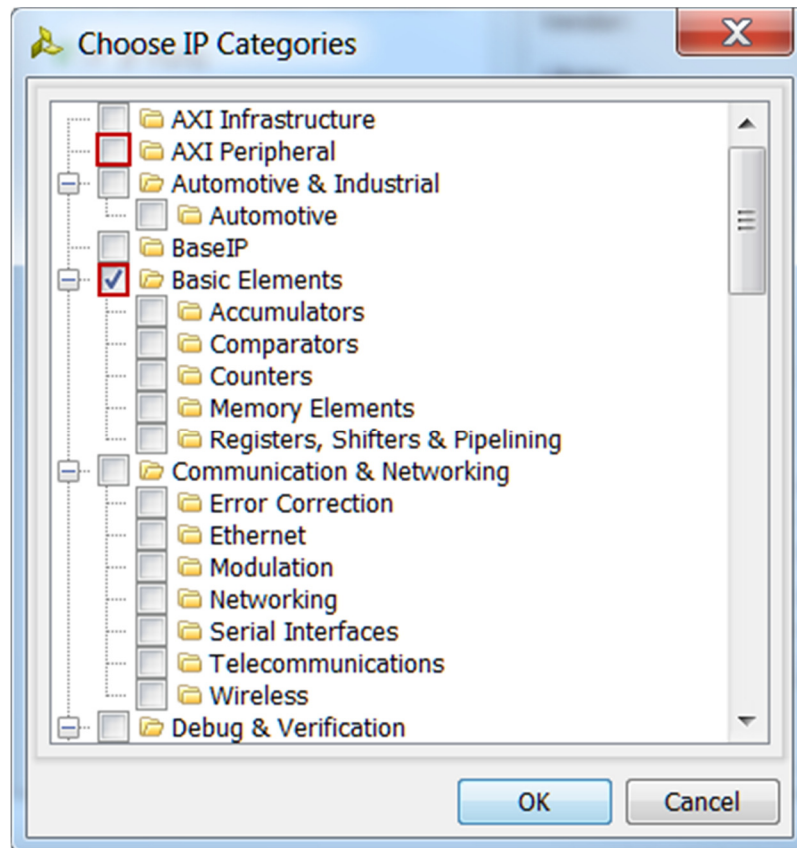
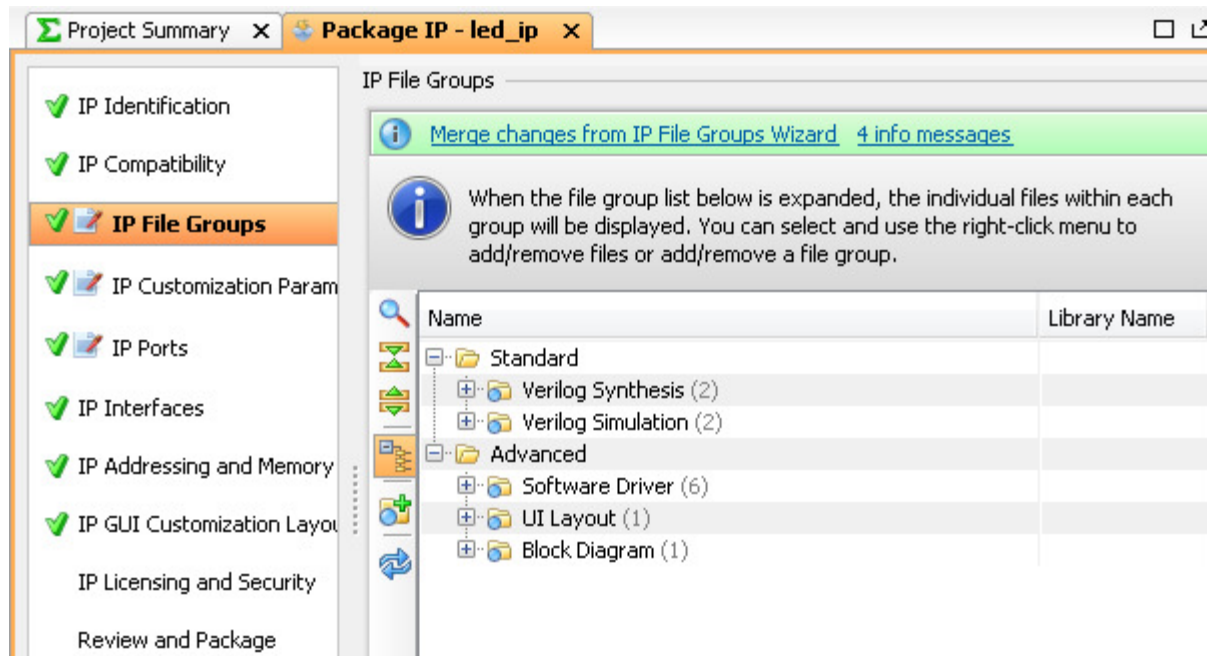


Figure 2 Specify the category for IP Packager IP

- 1-2-22. Select **IP Compatibility**. This shows the different Xilinx FPGA Families that the IP supports. The value is inherited from the device selected for the project.
- 1-2-23. Right click in the *Family Support table* and select **Add Family...** from the menu.
- 1-2-24. Select the **Zynq** family as we will be using this IP on the ZedBoard, and click **OK**.
- 1-2-25. You can also customize the address space and add memory address space using the **IP Addressing and Memory** category. We won't make any changes.
- 1-2-26. Click on **IP File Groups** and click *Merge changes from IP File Groups Wizard*



This is to update the IP Packager with the changes that were made to the IP and the *lab3_user_logic.v* file that was added to the project.

1-2-27. Do the same for IP Customization Parameters (*Merge changes from IP Customization Parameters Wizard*)

Notice that the *IP Ports* view now shows the user created *LED* port

✔ IP Customization Parameters

✔ IP Ports

✔ IP Interfaces

2-1-1. Click **Run Synthesis** and **Save** if prompted. (This is to check the design synthesizes correctly before adding it to the main design.

2-1-2. Check the **Messages** tab for any errors and correct if necessary before moving to the next step

When Synthesis completes successfully, click **Cancel**.

2-1-3. Select *Review and Package*, and click **Re-Package IP**

2-1-4. In the Vivado window click **File > Close Project**

Modify the Project Settings

Step 3

3-1. Open the previous project, or use the lab2 project from the labsolution directory, and save the project as lab3. Set Project Settings to point to the created IP repository.

3-1-1. Start the Vivado if necessary and open either the lab2 project you created in the previous lab or the lab2 project in the labsolution directory

3-1-2. Select **File > Save Project As ...** to open the *Save Project As* dialog box. Enter **lab3** as the project name. Make sure that the *Create Project Subdirectory* option is checked, the project directory path is `c:\xup\embedded\labs\` and click **OK**.

This will create the lab3 directory and save the project and associated directory with lab3 name.

3-1-3. Click **Project Settings** in the *Flow Navigator* pane.

3-1-4. Select **IP** in the left pane of the *Project Settings* form.

3-1-5. Click on the **Add Repository...** button, browse to `c:\xup\embedded\labs\led_ip` and click **Select**.

The led_ip_v1_0 IP will appear the **IP in the Selected Repository** window.

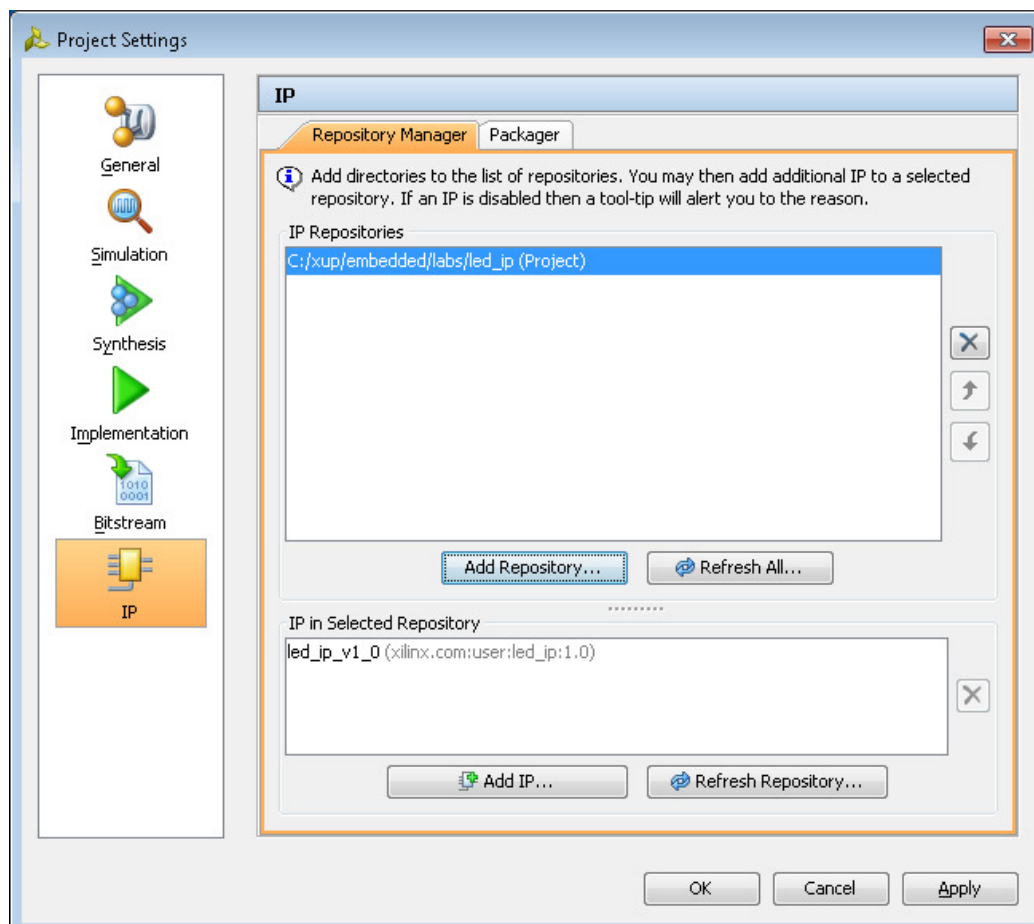


Figure 3 Specify IP Repository

3-1-6. Click **OK**.

Add the Custom IP, BRAM, and the Constraints

Step 4

4-1. Add led_ip to the design and connect to the AXI4Lite interconnect in the System Assembly View. Make internal and external port connections. Establish the LED port as external FPGA pins.

4-1-1. Click **Open Block Design** under **IP Integrator** in the Flow Navigator pane, and select **system.bd** to open IP Integrator

4-1-2. Click the Add IP icon  and search for **led_ip_v1_0** in the catalog by typing "led" in the search field.

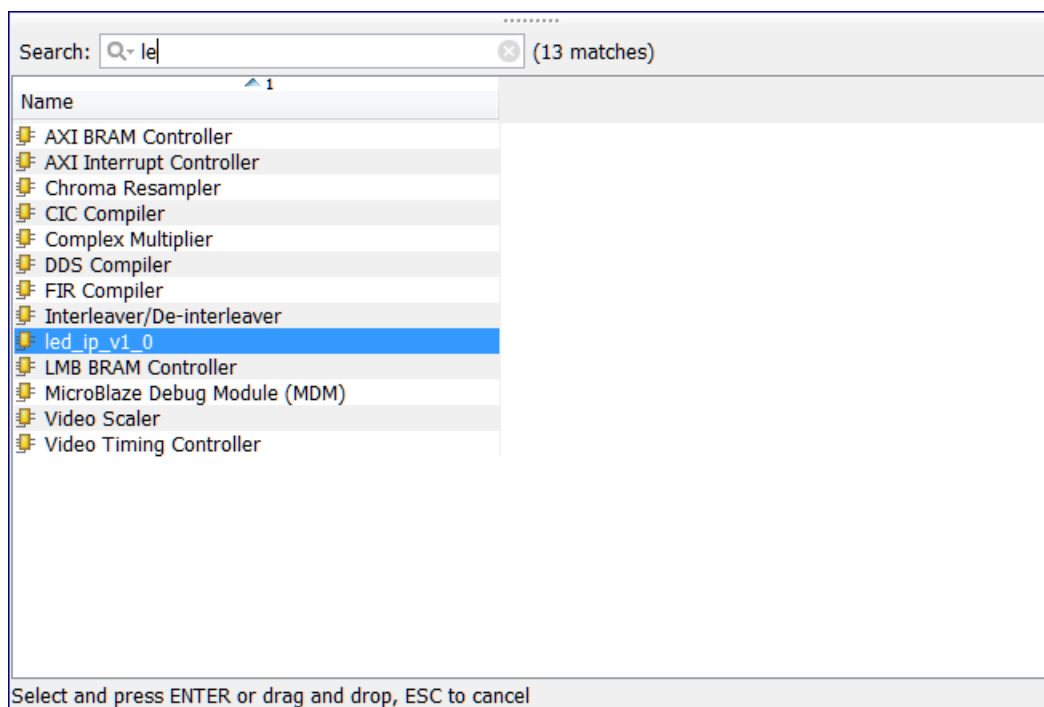


Figure 4 Searching for led_ip in the IP Catalog

4-1-3. Double-click **led_ip_v1_0** to add the core to the design

4-1-4. Select the IP in the block diagram and change the instance name to **led_ip** in the properties view

4-1-5. Click on **Run Connection Automation**, select **/led_ip/S_AXI** and click **OK** to automatically make the connection from the AXI Interconnect to the IP.

4-1-6. Click the regenerate button () to redraw the diagram.

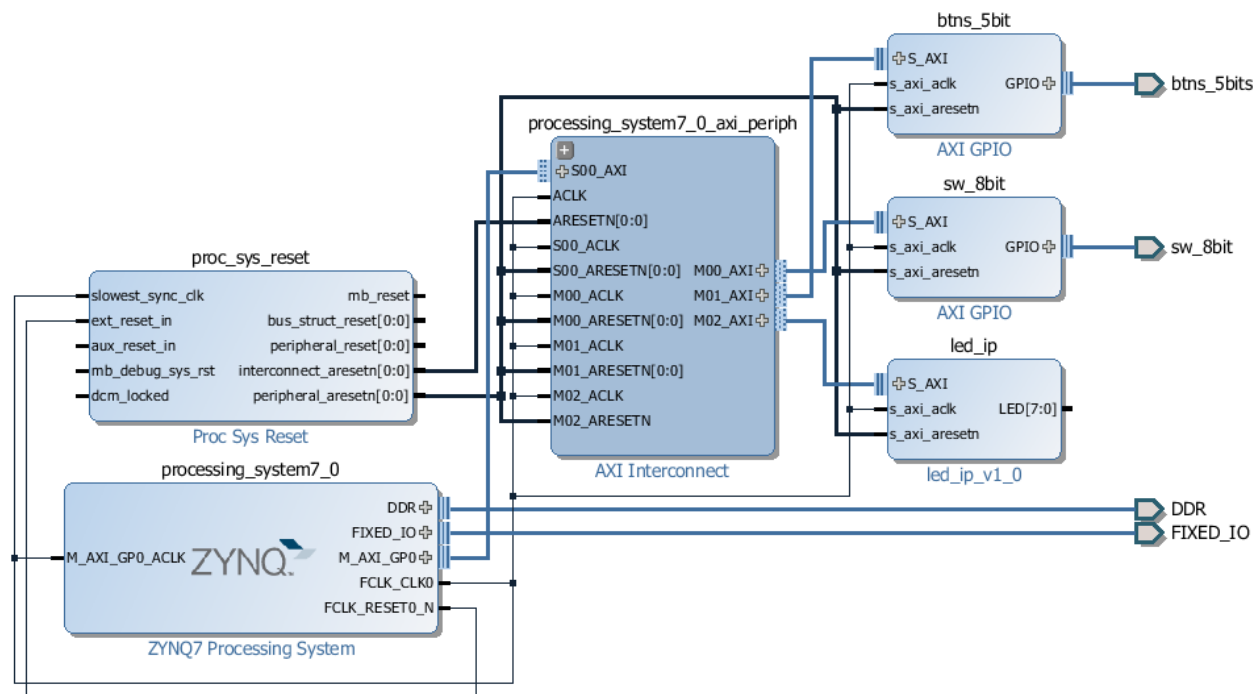


Figure 5 LED IP Block added and connected

4-1-7. Select the **LED[7:0]** port on the led_ip instance (by clicking on its pin), right-click and select **Make External**.

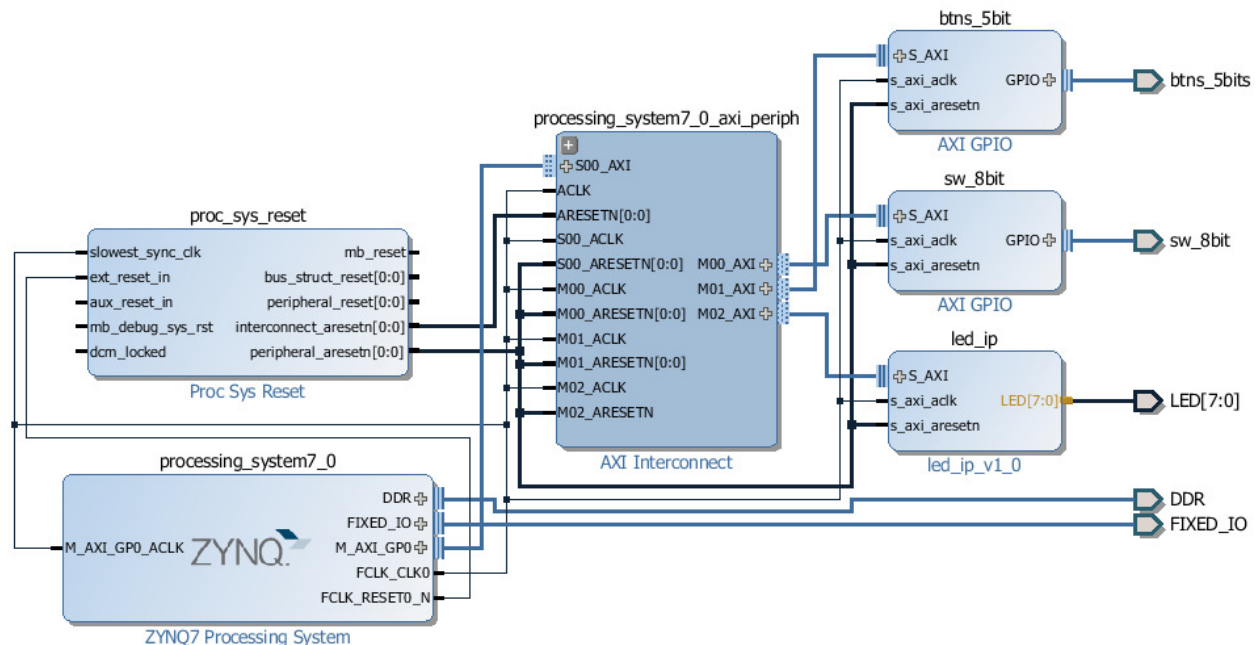


Figure 6 LED external port added and connected

4-1-8. Select the **Address Editor** tab and verify that an address has been assigned to led_ip.

Cell	Interface Pin	Base Name	Offset Address	Range	High Address
processing_system7_0					
Data (32 address bits : 4G)					
sw_8bit	S_AXI	Reg	0x41200000	64K	0x4120FFFF
btns_5bit	S_AXI	Reg	0x41210000	64K	0x4121FFFF
led_ip_0	S_AXI	S_AXI reg	0x43C00000	4K	0x43C00FFF

Figure 7 Address assigned for led_ip

4-1-9. Select **Tools > Validate Design** to run the design rules checker. There should not be any violations.

4-2. Update the top-level wrapper and add the provided lab3_system.xdc constraints file.

4-2-1. In the *sources* view, right-click on the block diagram file, **system.bd**, and select **Create HDL Wrapper** to update the HDL wrapper file, and when prompted, select *Let Vivado Manage Wrapper and auto-update* and click **OK**

The system_wrapper.vhd file will be updated to include the new IP and ports. Double-click on the system-wrapper content to verify that LED port has been added.

4-2-2. Click **Add Sources** in the *Flow Navigator* pane, select **Add or Create Constraints**, and click **Next**.


4-2-3. Click the **Add Files** button, browse to the **c:\xup\embedded\sources\lab3** folder, select **lab3_system.xdc**

4-2-4. Click **Finish** to add the file.

4-2-5. Expand Constraints folder in the *Sources* pane, and double-click **lab3_system.xdc** file entry to see its content. This file contains the pin locations and IO standards for the LEDs on the Zedboard. This information can usually be found in the manufacturer's datasheet for the board.

This part of the lab, creating and adding custom IP is now complete. The next lab will require Block RAM memory which will be added now.

4-3. Add BRAM to the design

4-3-1. In the Block Diagram, click the Add IP icon  and search for BRAM and add one instance of the **AXI BRAM Controller**

4-3-2. Run connection automation, on **axi_bram_ctrl_0/S_AXI** and click **OK** when prompted to connect it to the **M_AXI_GP0 Master**.

4-3-3. Double click on the block to customize it and change the number of BRAM interfaces to 1 and click **OK**.

Notice that the AXI Protocol being used is AXI4 instead of AXI4Lite since BRAM can provide higher bandwidth and the controller can support burst transactions.

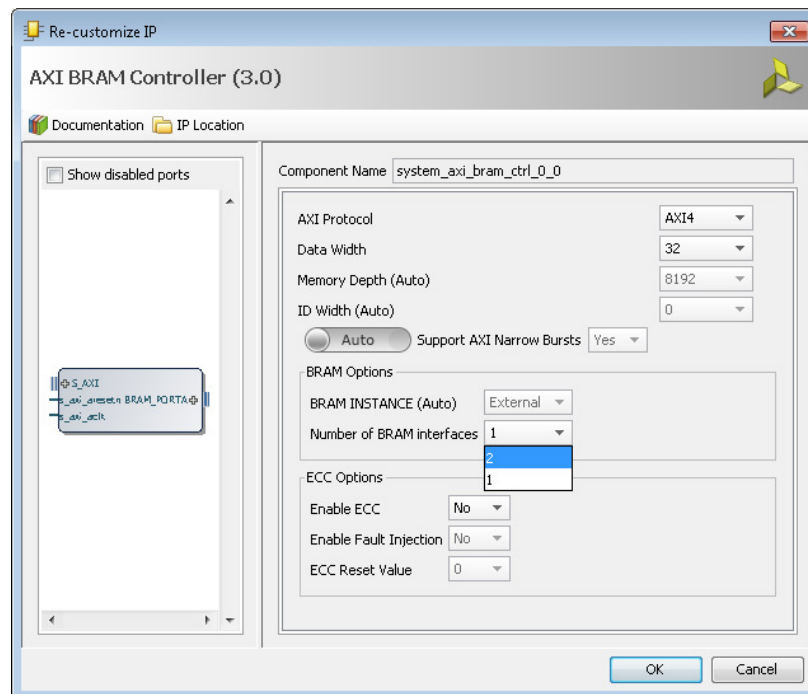


Figure 8 Customize BRAM controller

- 4-3-4. Run *Connection Automation* to add and connect a **Block Memory Generator** (This could be added manually)
- 4-3-5. Validate the design to ensure there are no errors, and click the regenerate button (🔄) to redraw the diagram.

The design should look similar to the figure below.

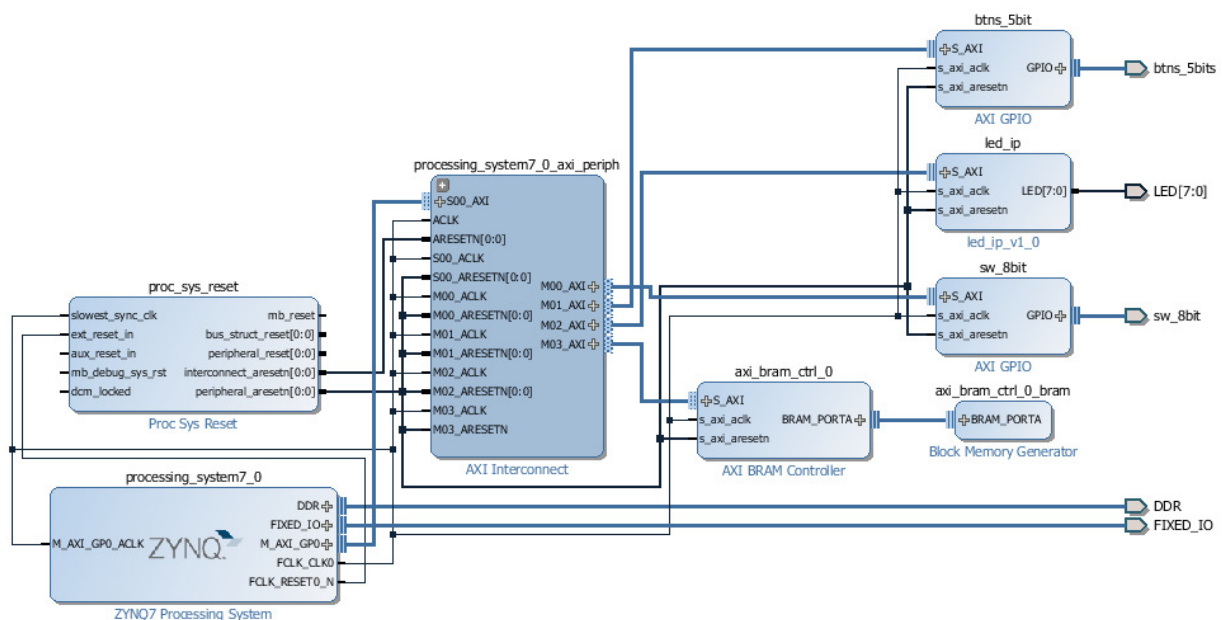


Figure 9 Completed Block Diagram

4-3-6. In the Address editor, increase the Range of the **axi_bram_ctrl_0** to 8K

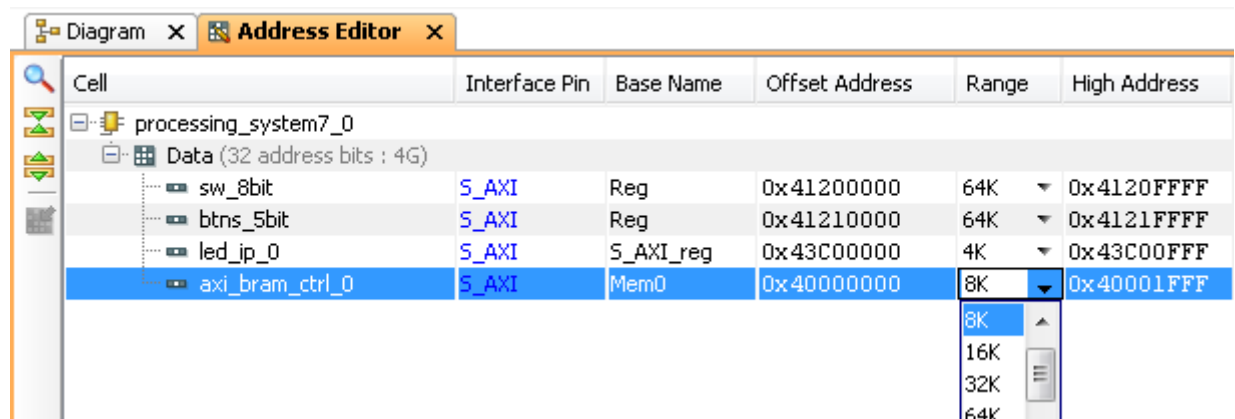


Figure 12 Assigning address range

4-3-7. In the sources view, right click on the **system.bd** block diagram file and select **Create HDL wrapper** (Click **OK** when prompted)

4-3-8. Press **F6** to validate the design one last time.

4-3-9. If there are no errors, click on the **Generate Bitstream** in the Flow Navigator to run the synthesis, implementation, and bitstream generation processes. (Click **Save** if prompted.)

4-3-10. Click **Yes** to run the synthesis process again as the design has changed.

4-3-11. When the build completes, click **OK** if prompted to open the Implemented design.

Conclusion

Vivado IP packager was used to import a custom IP block into the IP library. The IP block was then added to the system. Connection automation was run where available to speed up the design of the system by allowing Vivado to automatically make connections between IP. Pin location constraints were added to the design. Finally, an additional BRAM was added to the design, and the design was built (bitstream generated) in preparation for the next lab.