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

Lab #4 Part 1 credit is earned when Lab #4 Part 2 is completed and demonstrated.
Lab 3 is created after the led IP is created. Lab 3 is part of Lab 4 Part 1.

After completing this lab, you will be able to:

- Use the IP Packager feature of Vivado to create
- 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, and add the Block RAM (BRAM) Memory.

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 Zynq board you are using. 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.

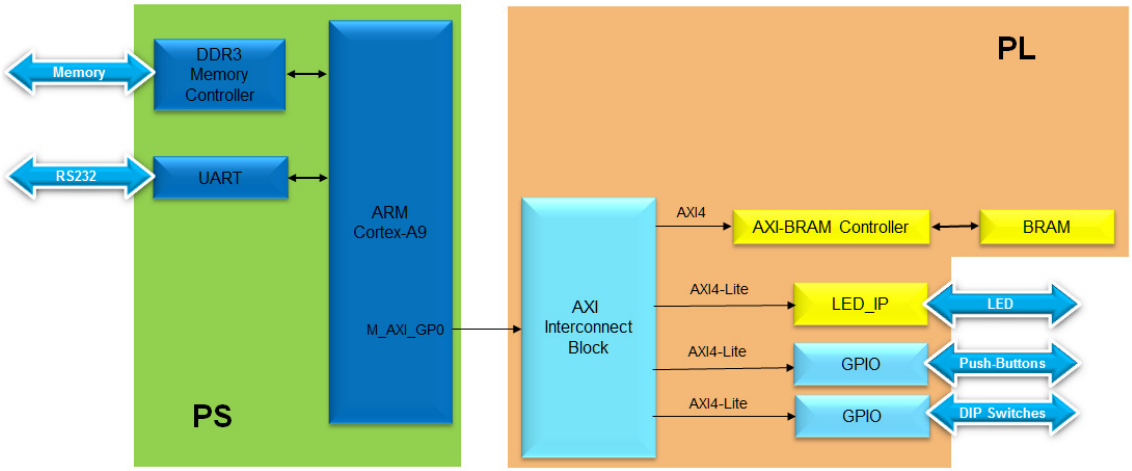
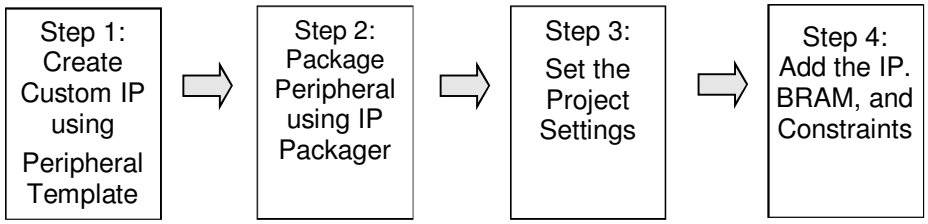


Figure 1. Design Updated from Previous Lab

General Flow for this Lab



In the instructions below;
{***sources***} refers to: C:\xup\embedded\2015_2_zynq_sources
{***labs***} refers to : C:\xup\embedded\2015_2_zynq_labs
{***labsolutions***} for the ZedBoard refers to: C:\xup\embedded\2015_2_zedboard_labsolution
or for the Zybo refers to: C:\xup\embedded\2015_2_zybo_labsolution

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 2016.2 > Vivado 2016.2**

1-1-2. Click **Manage IP** and select *New IP Location* and click **Next t**o select device as Zybo Z7-10 board.

1-1-3. Select **Verilog** as the *Target Language*, **Mixed** as the *Simulator language*, and for *IP location*, type *{labs}\led_ip* and click **Finish** (leave other settings as defaults and click **OK** if prompted to create the directory)

Part Number is Zybo Z7-10 or xc7z010iclg400-1L

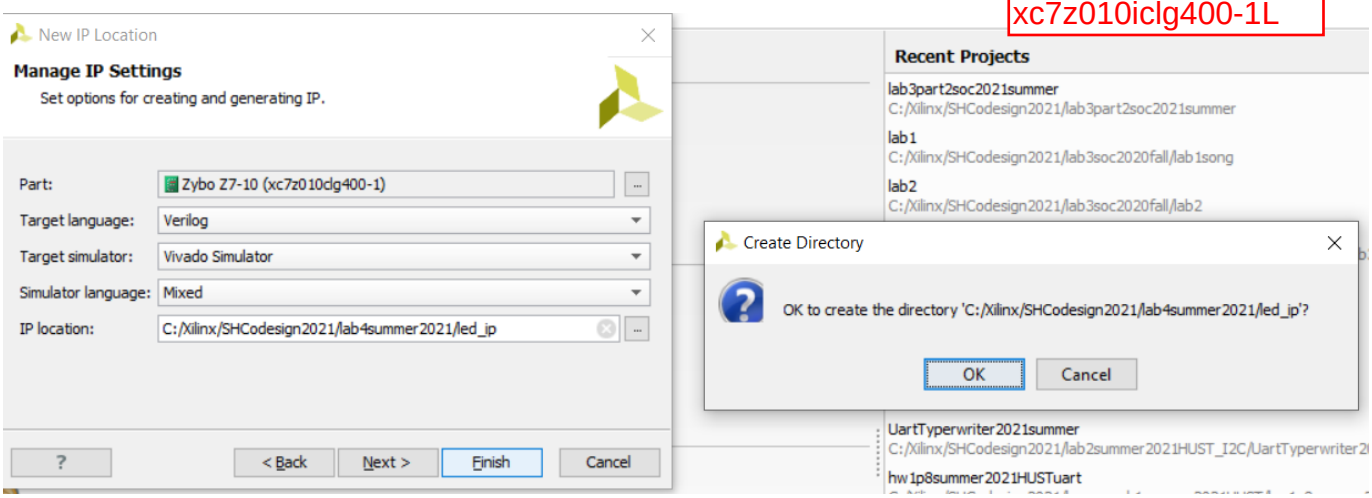
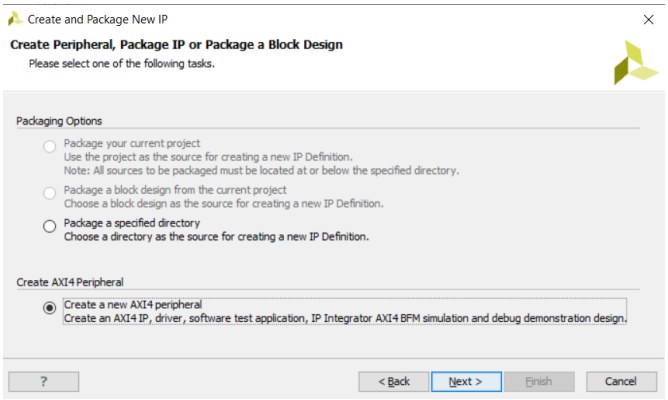


Figure 2. Device Selection and New IP Location form

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

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 a new AXI4 peripheral*, 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)

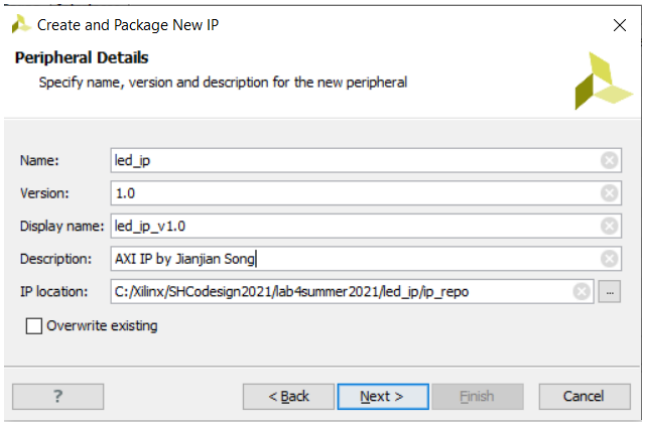


Figure 3. Updating Peripheral Details form

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)

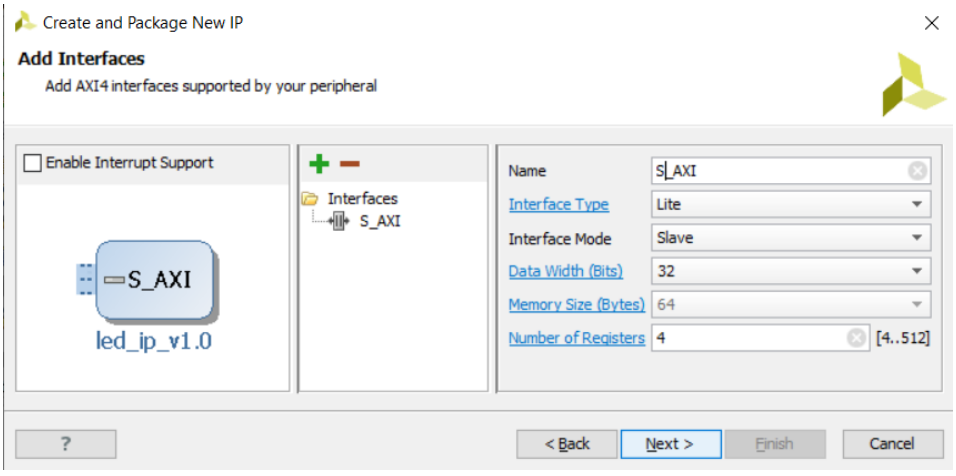
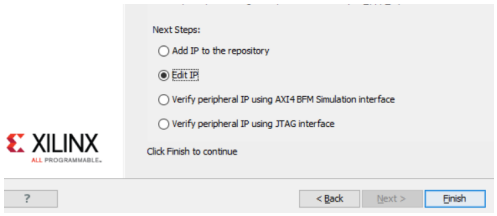


Figure 4. Naming the AXI interface

1-2-8. Select *Edit IP* and click **Finish** (a new Vivado Project will open)



1-3. Create an interface to the LEDs

1-3-1. In the sources panel, double-click the **led_ip_v1_0.v** file.

This file contains the HDL code for the interface(s) selected above. The top level file contains a module which implements the AXI interfacing logic, and an example design to write to and read from the number of registers specified above. This template can be used as a basis for creating custom IP. A new parameterized output port to the LEDs will be created at the top level of the design, and the AXI write data in the sub-module will be connected back up to the external LED port.

Scroll down to line 7 where a user *parameters* space is provided.

1-3-2. Add the line:

```
parameter integer LED_WIDTH	= 8,
```

1-3-3. Go to line 18 and add the line:

```
output wire [LED_WIDTH-1:0]	LED,
```

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

```
 4    module led_ip_v1_0 #
 5    (
 6        // Users to add parameters here
 7        parameter integer LED_WIDTH             = 8,
 8        // User parameters ends
 9        // Do not modify the parameters beyond this line
10
11
12        // Parameters of Axi Slave Bus Interface S_AXI
13        parameter integer C_S_AXI_DATA_WIDTH    = 32,
14        parameter integer C_S_AXI_ADDR_WIDTH    = 4
15    )
16    (
17        // Users to add ports here
18        output wire [LED_WIDTH-1:0]      LED,
19        // User ports ends
```

Figure 5. Adding users parameter, and port definition

1-3-4. Insert the following at line ~48:

```
.LED_WIDTH(LED_WIDTH),
```

1-3-5. Insert the following at line ~52 (notice a comma is needed at the end of each statement):

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

```
46 // Instantiation of Axi Bus Interface S_AXI
47     led_ip_v1_0_S_AXI # (
48         .LED_WIDTH(LED_WIDTH),
49         .C_S_AXI_DATA_WIDTH(C_S_AXI_DATA_WIDTH),
50         .C_S_AXI_ADDR_WIDTH(C_S_AXI_ADDR_WIDTH)
51     ) led_ip_v1_0_S_AXI_inst (
52         .LED(LED),
53         .S_AXI_ACLK(s_axi_aclk),
54         .S_AXI_ARESETN(s_axi_aresetn),
```

Figure 6. Adding port connection with a lower-level module

1-3-6. Save the file by selecting **File > Save File**

1-3-7. Expand *led_ip_v1_0* in the sources view if necessary, and open **led_ip_v1_0_S_AXI.v**

1-3-8. Add the LED parameter and port to this file too, at lines 7 and 18

```
4  module led_ip_v1_0_S_AXI #
5  (
6      // Users to add parameters here,
7      parameter integer LED_WIDTH              = 8,
8      // User parameters ends
9      // Do not modify the parameters beyond this line
10
11     // Width of S_AXI data bus
12     parameter integer C_S_AXI_DATA_WIDTH     = 32,
13     // Width of S_AXI address bus
14     parameter integer C_S_AXI_ADDR_WIDTH     = 4
15 )
16 (
17     // Users to add ports here
18     output wire [LED_WIDTH-1:0]      LED,
19     // User ports ends
```

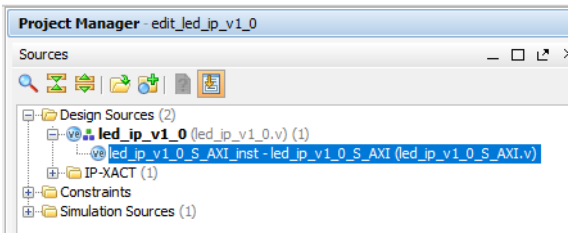


Figure 7. Declaring users port in the lower-level module for the Zybo

1-3-9. Scroll down to ~line 400, where there is a comment line that says "Add usre logic here. 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.)

```
384          // output the read dada
385          if (slv_reg_rden)
386            begin
387              axi_rdata <= reg_data_out;     // register read data
388            end
389        end
390    end
391
392    // Add user logic here
393
394    // User logic ends
395
396    endmodule
397
```

```
391
392     // Add user logic here
393     lab3_user_logic  # (
394        .LED_WIDTH(LED_WIDTH)
395     )
396     U1(
397        .S_AXI_ACLK(S_AXI_ACLK),
398        .slv_reg_wren(slv_reg_wren),
399        .axi_awaddr(axi_awaddr[C_S_AXI_ADDR_WIDTH-1:ADDR_LSB]),
400        .S_AXI_WDATA(S_AXI_WDATA),
401        .S_AXI_ARESETN(S_AXI_ARESETN),
402        .LED(LED)
403     );
404     // User logic ends
```

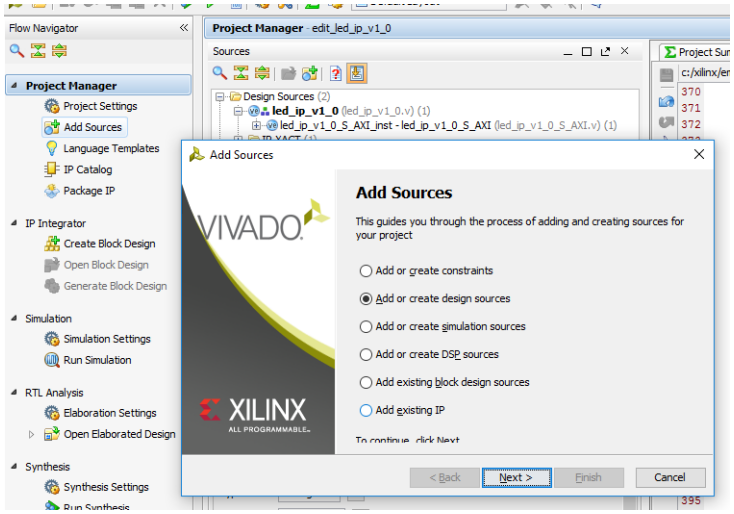
```
lab3_user_logic  # (
   .LED_WIDTH(LED_WIDTH)
)
U1(
   .S_AXI_ACLK(S_AXI_ACLK),
   .slv_reg_wren(slv_reg_wren),

.axi_awaddr(axi_awaddr[C_S_AXI_ADDR_WIDTH-1:ADDR_LSB]),
   .S_AXI_WDATA(S_AXI_WDATA),
   .S_AXI_ARESETN(S_AXI_ARESETN),
   .LED(LED)
);
```

Figure 8. Instantiating lower-level user module

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

1-3-10. Save the file by selecting **File > Save File**

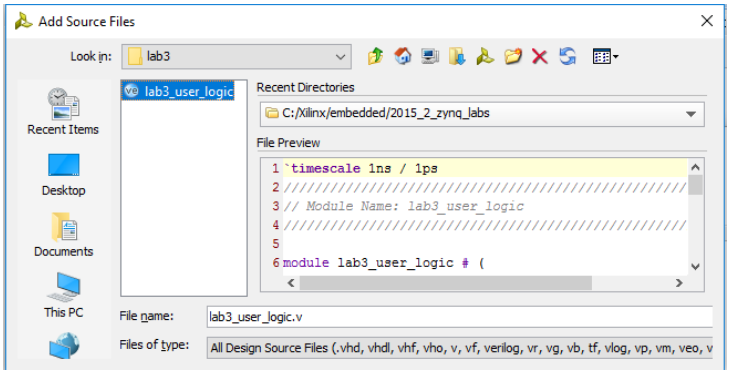
1-3-11. Click on the *Add Sources* in the Flow Navigator pane, select *Add or Create Design Sources*, click



Next, then click the **Green Plus** then **Add Files…**, browse to **{sources}\lab3**, select the **lab3_user_logic.v** file and click **OK.** Check **"Copy Sources into IP Directory** and then click **Finish** to add the file. This file is added and can be see under led_ip_v1_0_S_AXI.v of Design Sources folder.

Make sure all sources are copied to the project directory to avoid source location confusion in the future.

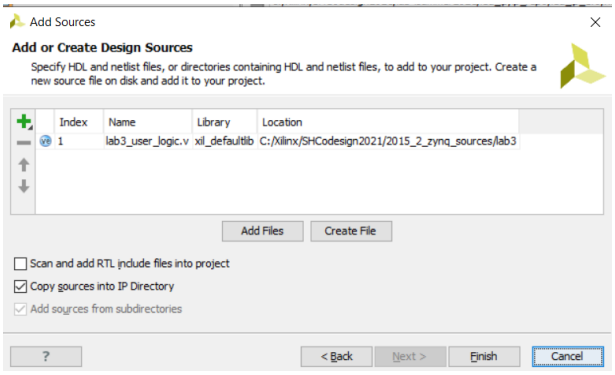
Check the contents of this file to understand the logic that is being implemented. Notice the formed hierarchy.



1-3-12. Click **Run Synthesis** and *Save* if prompted. (This is to check the design synthesizes correctly before packaging the IP. If this was your own design, you would simulate it and verify functionality before proceeding)

1-3-13. **<u>Check the *Messages* tab for any errors and correct if necessary before moving to the next step</u>**

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



1-4. Package the IP

1-4-1. Click on the **Package IP – led_ip** tab under Flow Navigator panel.

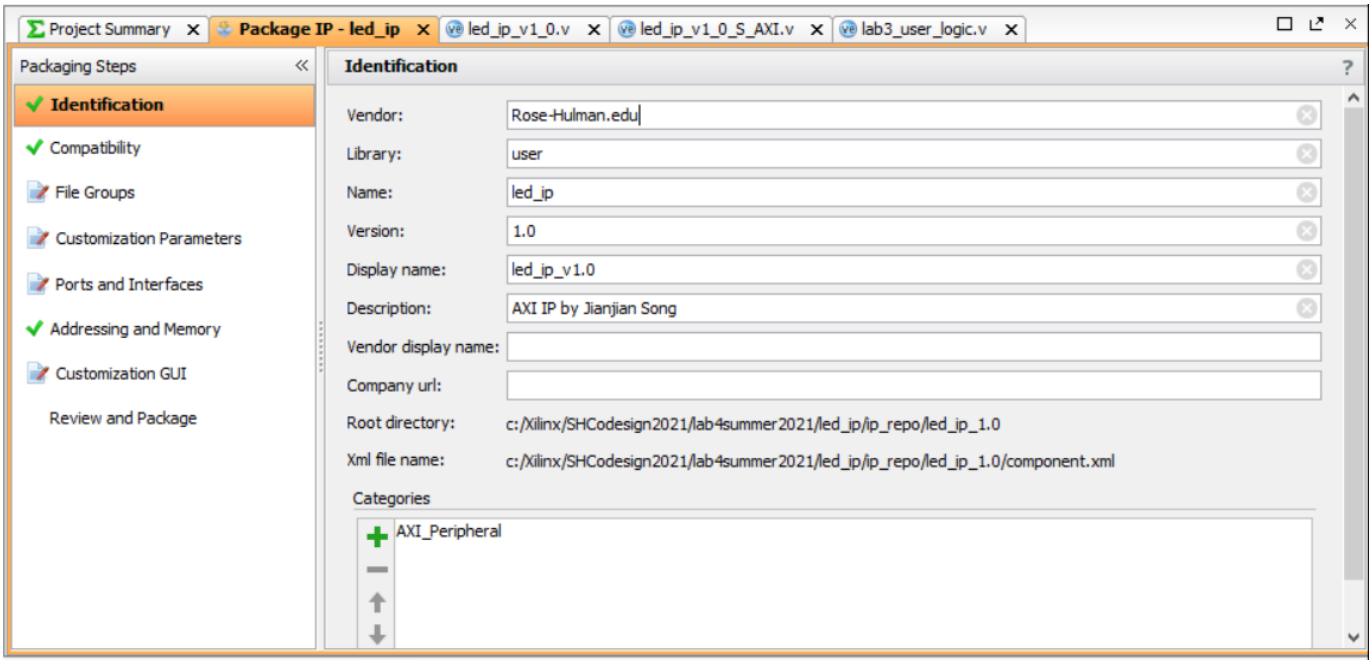


Figure 9. Package IP

1-4-2. 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 **Green Plus** in the *Categories* section on the bottom. This opens the Choose IP Categories window.

1-4-3. For the purpose of this exercise, uncheck the **AXI Peripheral** box and check the **Basic Elements** and click **OK**.

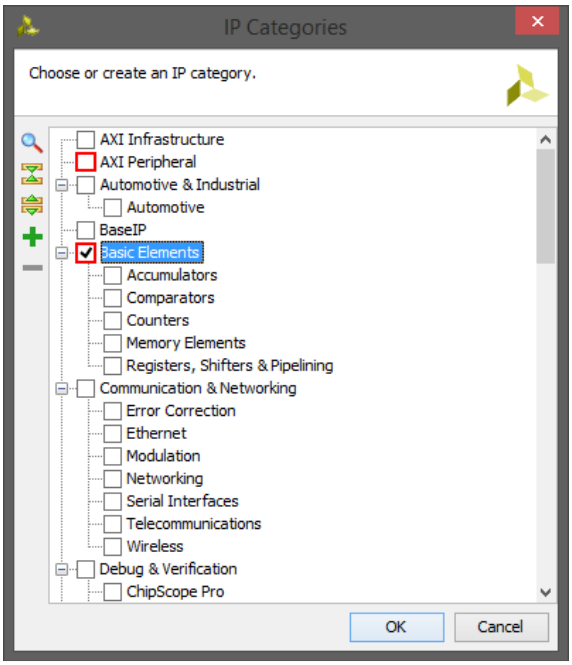
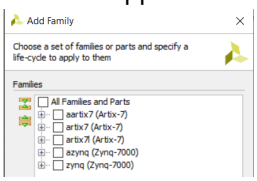


Figure 10. Specify the category for IP Packager IP

1-4-4. Select **Compatibility**. This shows the different Xilinx FPGA Families that the IP supports. The value is inherited from the device selected for the project.



1-4-5. Click the **Green Plus** then **Add Family Explicitly…** from the menu.

1-4-6. Select the **Zynq** family as we will be using this IP on the Zybo and Zedboard, and click **OK**.

1-4-7. 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.

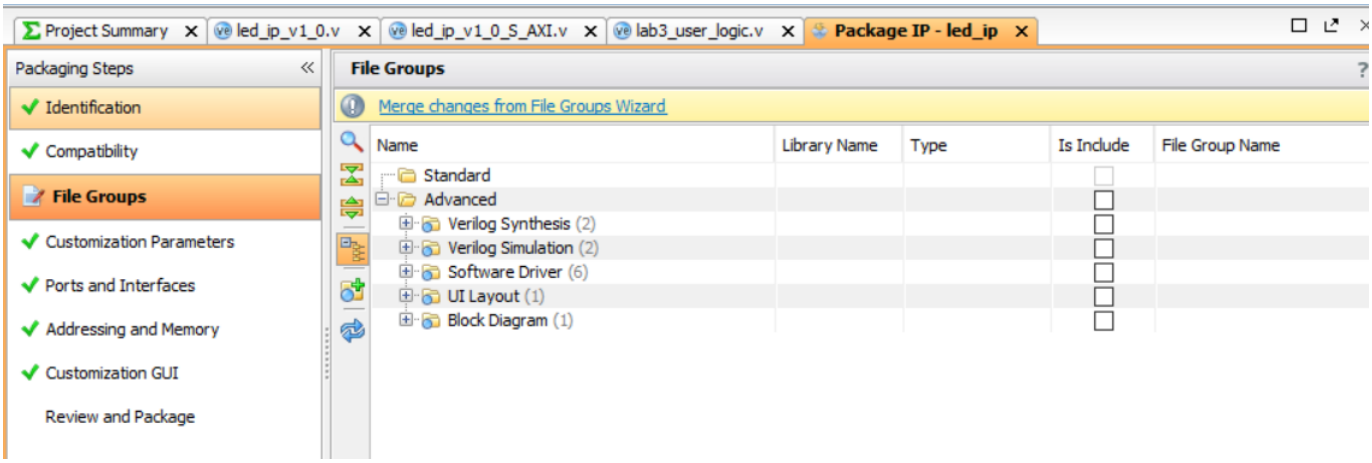
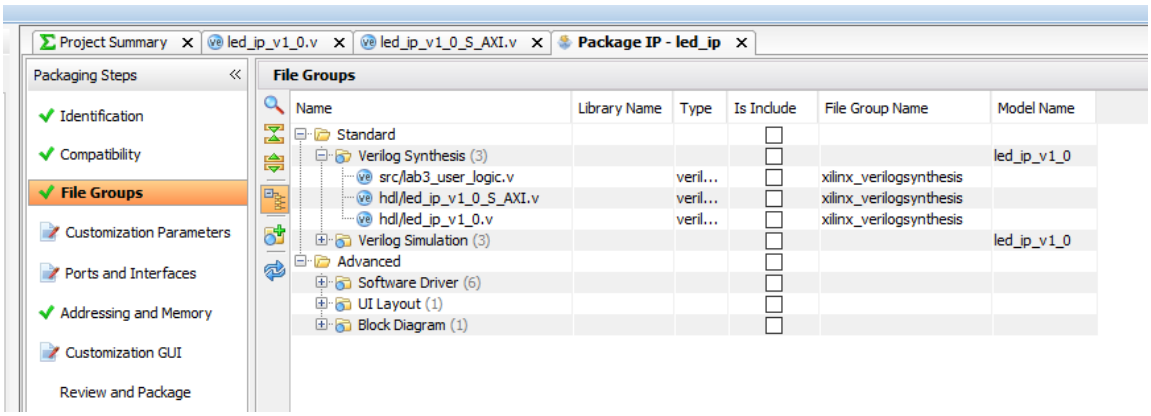


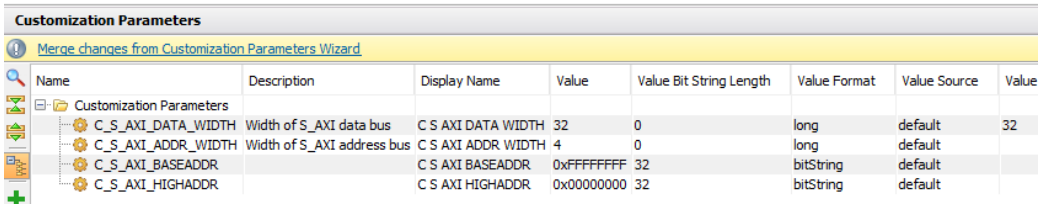
Figure 11. Updating the file group

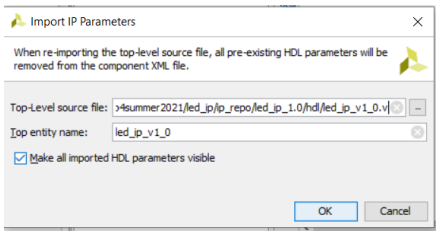
This is to update the IP Packager with the changes that were made to the IP and the *lab3_user_logic_*.v* file that was added to the project.

1-4-9. Expand *Verilog Synthesis* under File Groups panel and notice **lab3_user_logic.v** has been included



1-4-10. Click on **Customization Parameters** and again *Merge changes if neede from Customization Parameters Wizard* Notice that the *Ports and Interfaces* view now shows the user created *LED* port. you may also need to import IP parameters if you do not see LED_WIDTH. Right click to Import IP port to see LED port.





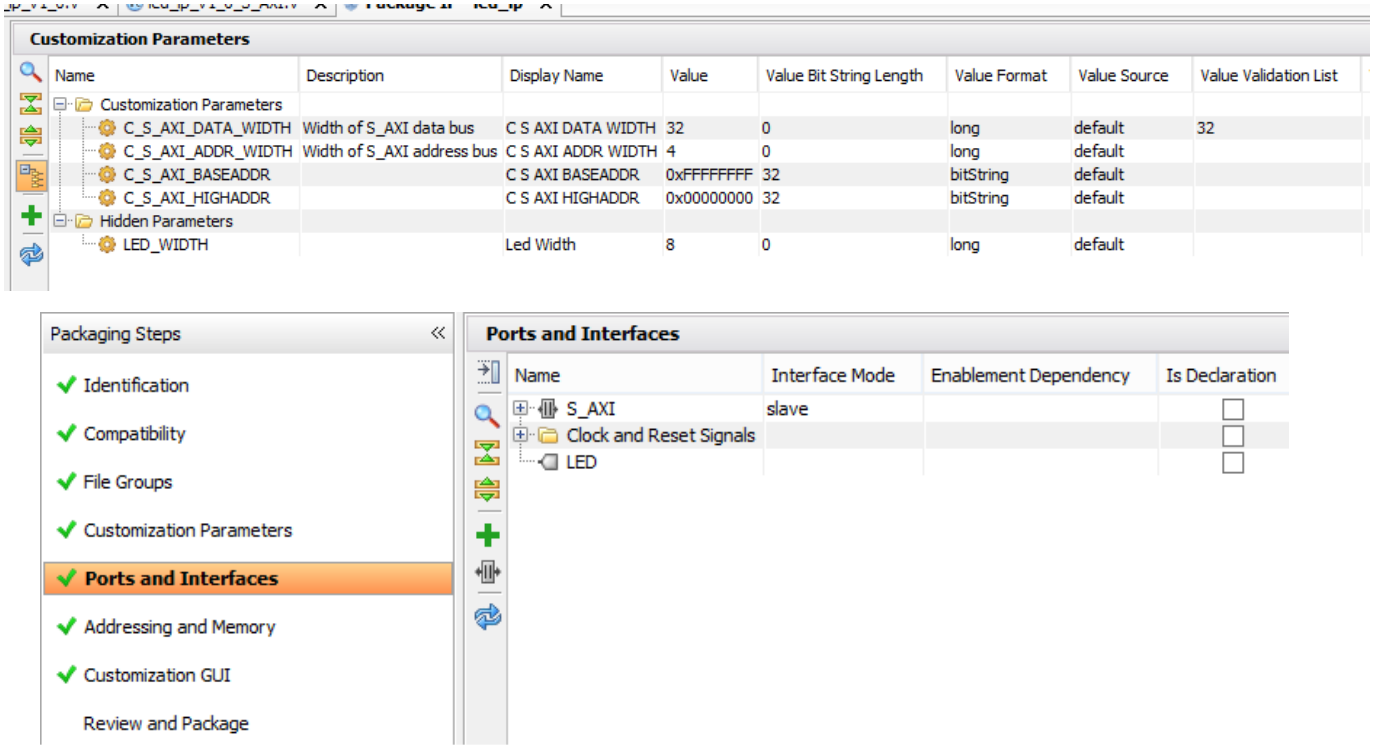


Figure 12. User parameter and port

1-4-11. If you do not see LED port, select **Customization Parameters**, expand *Hidden Parameters*, right-click on **LED_WIDTH**, and select I**mport IP Parameters**… and click OK.

1-4-12. Select **Customization GUI** and notice that the *Led Width* is visible.

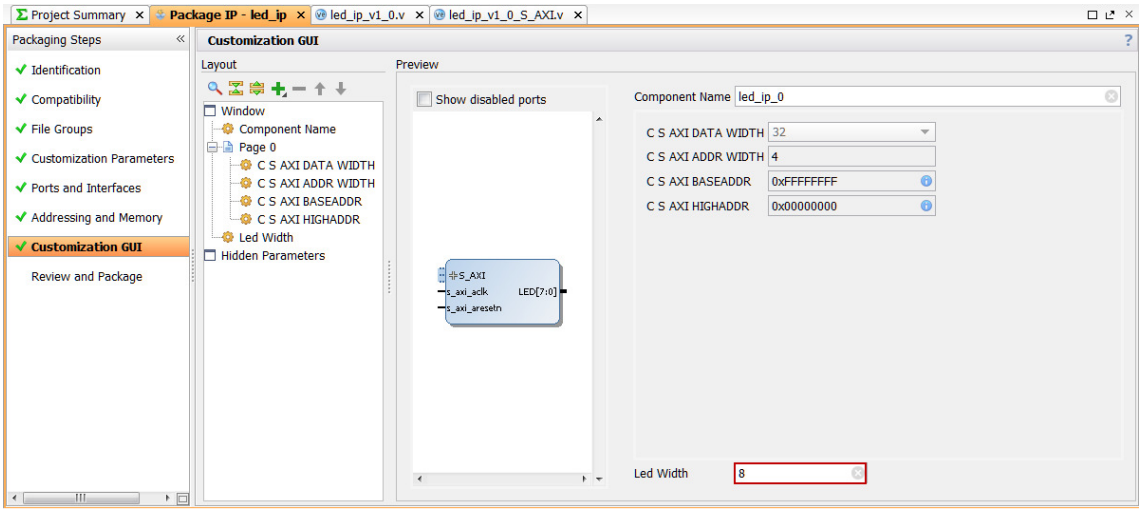
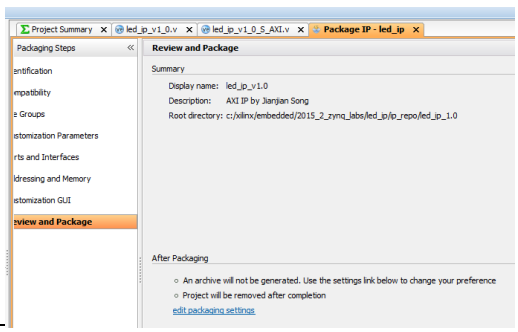


Figure 13. User customization parameter



1-4-13. Select *Review and Package*, and notice the path where the IP will be created.

1-4-14. Click **Re-Package IP**. Click **Yes** and the project will close when complete.

1-4-15. In the original Vivado window click **File > Close Project**

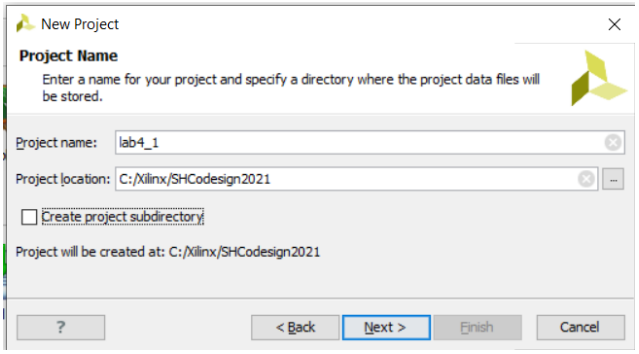
Modify the Project Settings Step 3

2-1. You can use the previous project or create a new project in your class project directory save the project as lab4_1. Set Project Settings to point to the created IP repository.

It is easier to create a new project and name it to be lab4_1, which is part of our Lab #4 Part 1. Make sure to choose the new Zybo board support files.

2-1-1. 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

2-1-2. Select **File > Save Project As …** to open the *Save Project As* dialog box. Enter **lab4_1** as the project name. Make sure that the *Create Project Subdirectory* option is checked, the project directory path is {labs}\ and click **OK**.



This will create the lab3_1 directory and save the project and associated directory with lab3_1 name.

2-1-3. Click **Project Settings** in the *Flow Navigator* pane.

2-1-4. Select **IP** in the left pane of the *Project Settings* form.

2-1-5. Click on the **Green Plus** button, browse to **{labs}\led_ip** and click **Select**.

The led_ip_v1.0 IP will appear the **IP in the Selected Repository** window.

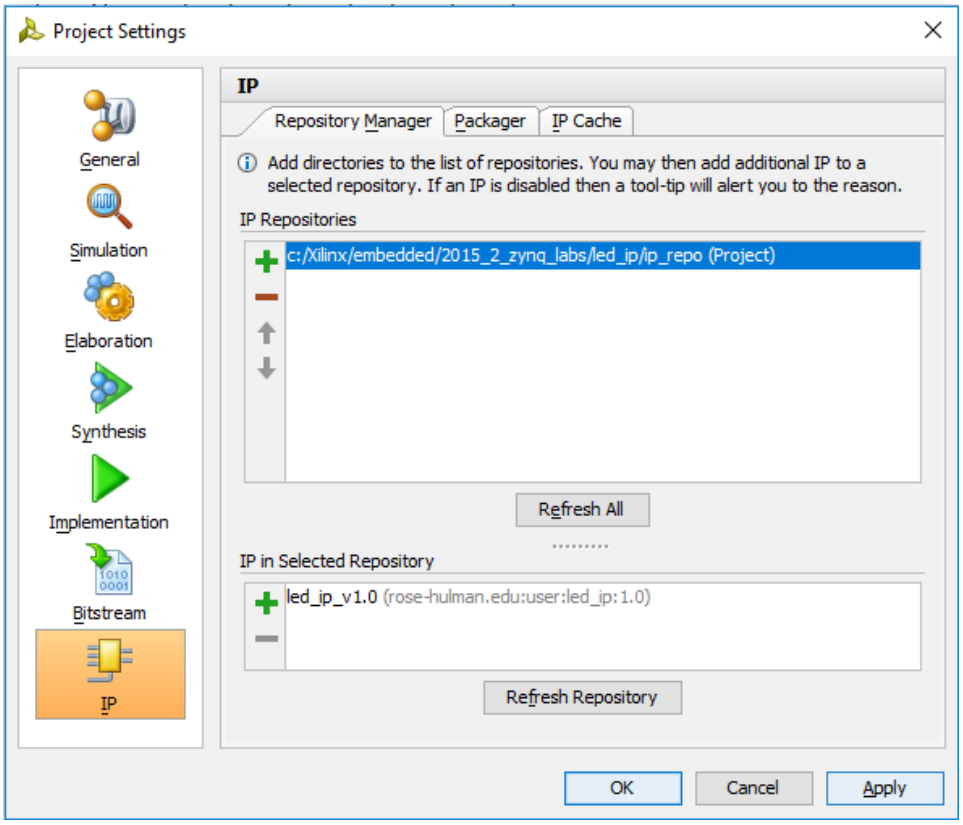


Figure 14. Specify IP Repository

2-1-6. Click **OK**.

Add the Custom IP, BRAM, and the Constraints Step 4

3-1. Add led_ip to the design and connect to the AXI4Lite interconnect in the IPI. Make internal and external port connections. Establish the LED port as external FPGA pins.

3-1-1. Click **Open Block Design** under **IP Integrator** in the Flow Navigator pane

3-1-2. Click the Add IP icon and search for **led_ip_v1.0** in the catalog by typing "led" in the search field.

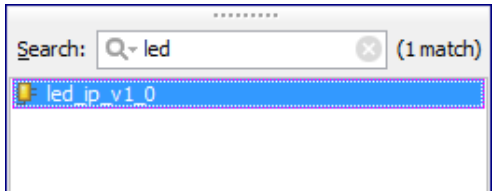
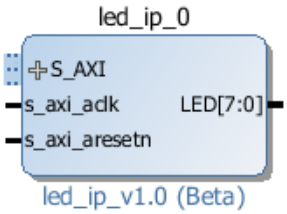


Figure 15. Searching for led_ip in the IP Catalog

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



3-1-4. Double click on the led_ip_0 in the block diagram and change the instance name to **led_ip** in the properties view.

3-1-5. Double click the block to open the configuration properties

3-1-6. For the ZedBoard, leave the *Led Width* set to 8, or for the Zybo, set the width to 4.

3-1-7. Click **OK**.

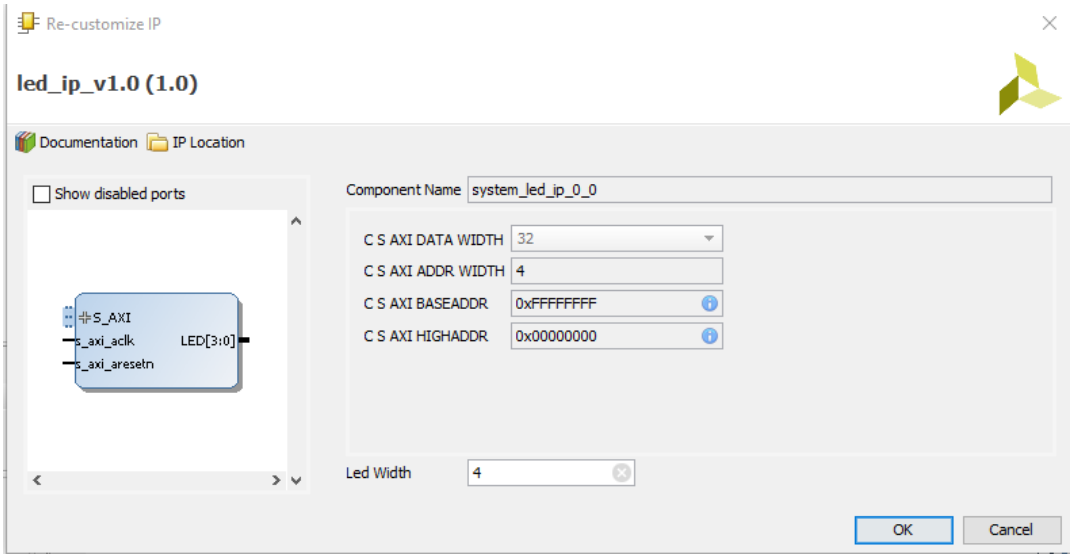


Figure 16. Configure the LED IP LED_WIDTH

3-1-8. 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.

Click the regenerate button ( ) to redraw the diagram.

3-1-9. Select the *LED* port on the led_ip instance (by clicking on its pin), right-click and select **Make External**.

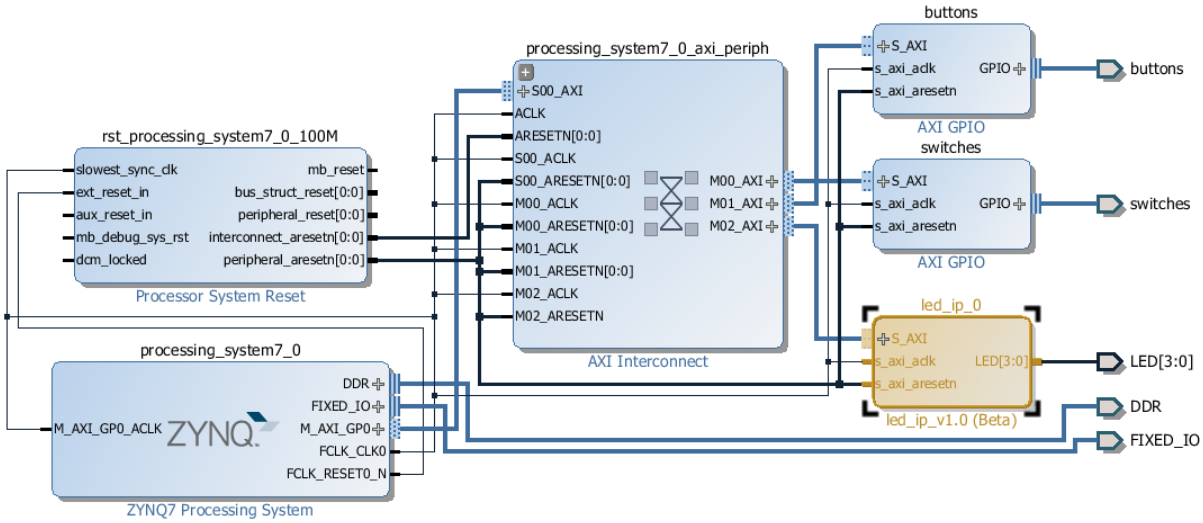


Figure 17. LED external port added and connected

3-1-10. Select the **Address Editor** tab and verify that an address has been assigned to *led_ip*.

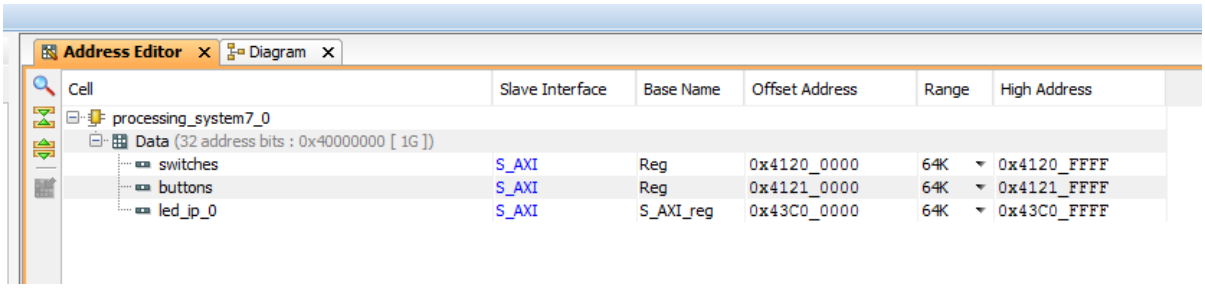
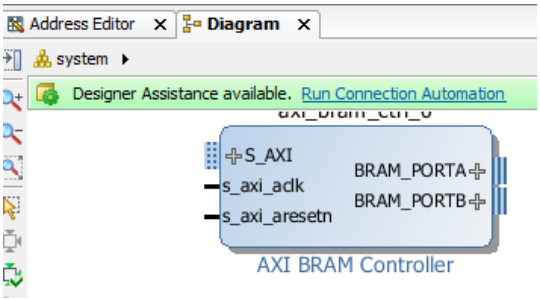


Figure 18. Address assigned for led_ip

3-2. Add BRAM to the design

3-2-1. In the Block Diagram, click the Add IP icon and search for BRAM and add one instance of **AXI_BRAM CONTROOLER**



3-2-3. 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.

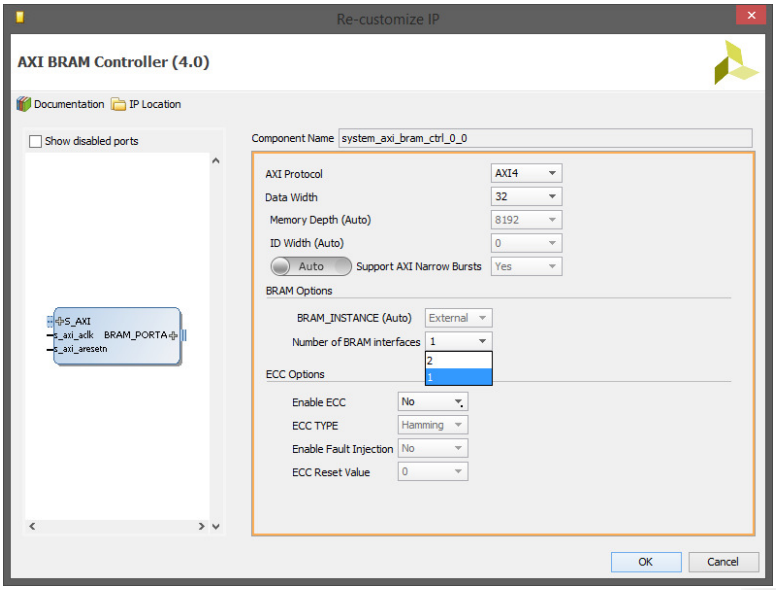
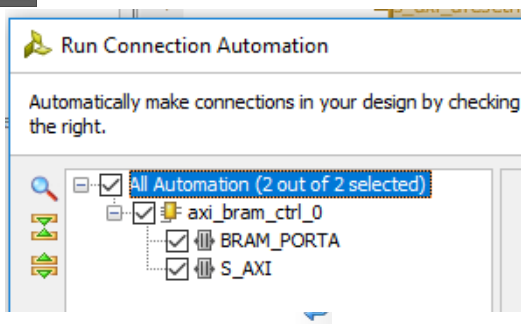


Figure 19. Customize BRAM controller

3-2-4. Click on *Run Connection Automation* to add and connect a **Block Memory Generator by**

selecting ALL, axi_bram_ctrl_0/BRAM_PORTA and click OK (This could be added manually)



3-2-5. Validate the design to ensure there are no errors (F6), and click the regenerate button ( ) to redraw the diagram.

The design should look similar to the figure below.

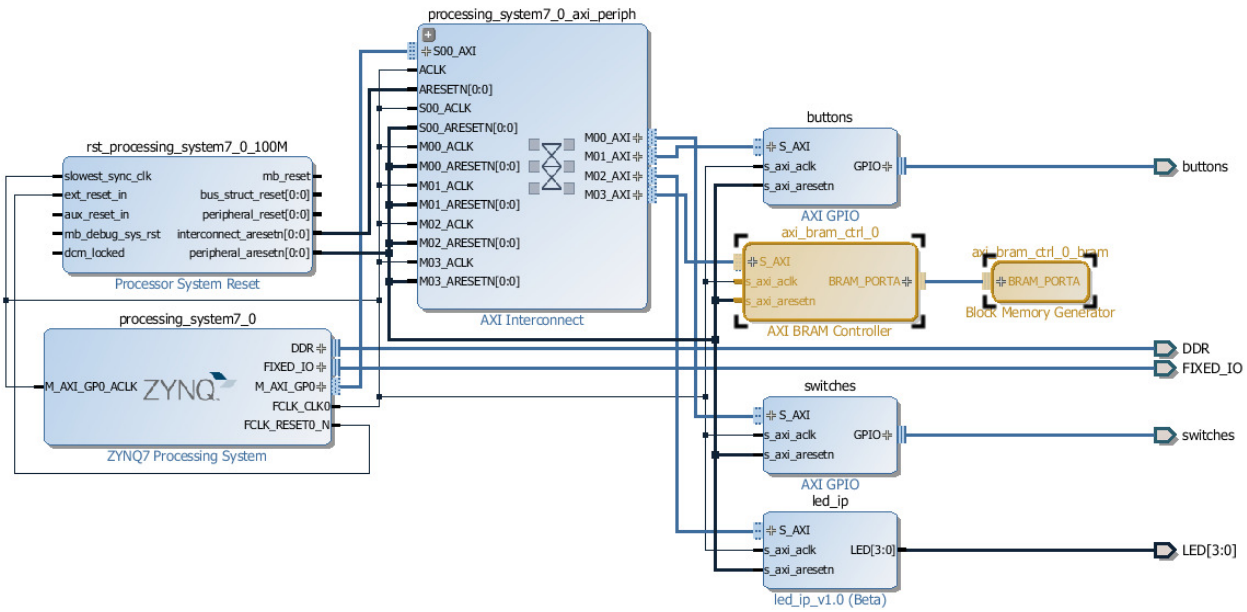


Figure 20. Completed Block Diagram

3-2-6. In the Address editor, notice the Range of the **axi_bram_ctrl_0** is **8K**. We will leave it at that.

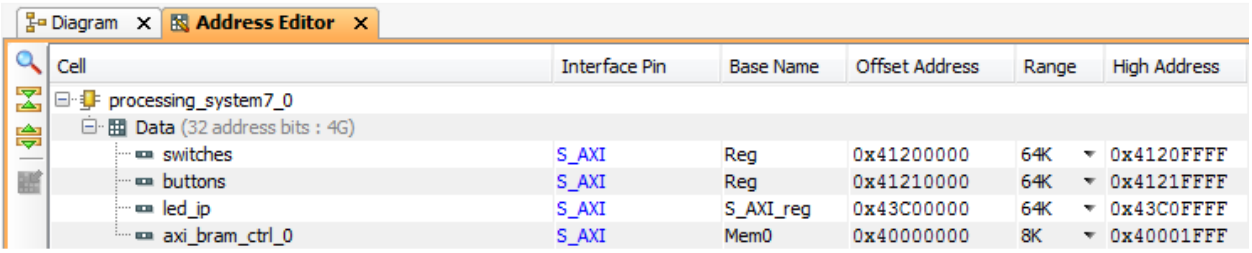


Figure 21. Adjusting memory size

3-2-7. Press **F6** to validate the design one last time.

3-3. Add the provided lab3_*.xdc constraints file.

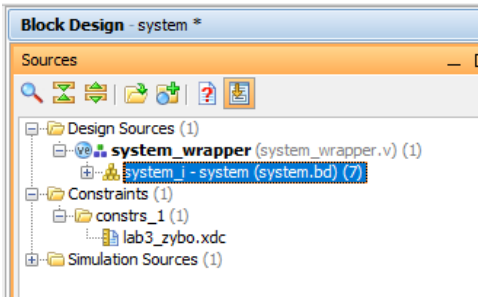
3-3-1. Click **Add Sources** in the *Flow Navigator* pane, select **Add or Create Constraints**, and click **Next**.

3-3-2. Click the **Green Plus** button, and then **Add Files…**, browse to the {**sources**}**lab3** folder, select **lab3_zed.xdc** for the ZedBoard, or **lab3_Zybo.xdc** for the Zybo

3-3-3. Click **Finish** to add the file.

3-3-4. Expand Constraints folder in the *Sources* pane, and double click the **lab3_*.xdc** file entry to see its content. This file contains the pin locations and IO standards for the LEDs on the Zynq board. This information can usually be found in the manufacturer's datasheet for the board.

3-3-5. Right click on *system.bd* under **system_wrapper** and select *Generate output products*



3-3-6. Click on **Generate Bitstream** and click **Yes** if prompted to save the Block Diagram, and click **Yes** again if prompted to launch synthesis and implementation. Click **Cancel** when 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. An additional BRAM was added to the design. Finally, pin location constraints were added to the design.