

# 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, 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.

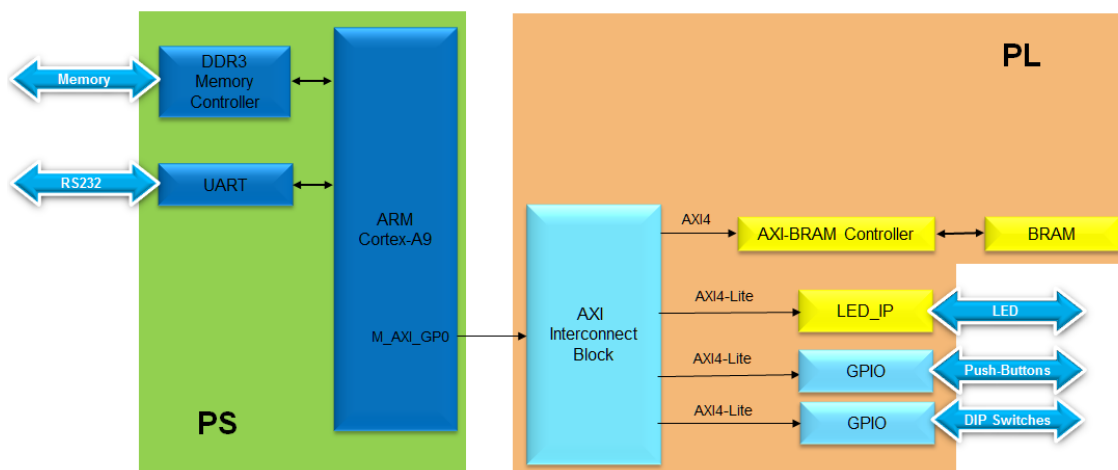
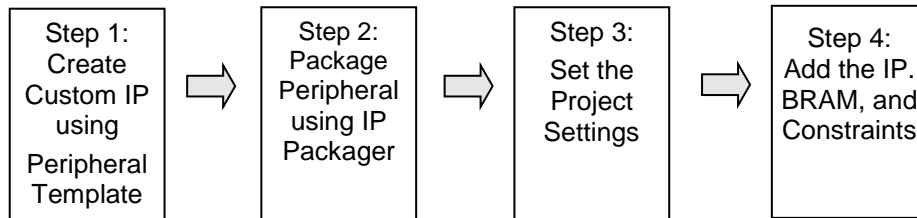


Figure 1. Design Updated from Previous Lab

## General Flow for this Lab



In the instructions below;

{**sources**} refers to: C:\Xilinx\_trn\Zynq\_base\lab\_sources

{**labs**} refers to : C:\Xilinx\_trn\Zynq\_base

## 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 WEB 2018.3 > Vivado 2018.3**

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

1-1-3. In the window appeared select

- Part: **ZedBoard Zynq Evaluation and Development Kit (xc7z020clg484-1)**
- Target Language: **Verilog**,
- Target Simulator: Vivado Simulator,
- Simulator language : **Mixed**

**New IP Location**

**Manage IP Settings**  
Set options for creating and generating IP.

Part: ZedBoard Zynq Evaluation and Development Kit (xc7z020clg484-1) ...

Target language: Verilog ▾

Target simulator: Vivado Simulator ▾

Simulator language: Mixed ▾

IP location: C:/Xilinx\_trn/Zynq\_base/led\_ip X ...

? < Back Next > Finish Cancel

**Figure 2. New IP Location form**

The **xc7z020clg484-1** part is chosen for this project, but later compatibility for other devices will be added to the packaged IP.

**1-1-4.** Click **Finish** (click **OK** if prompted to create the directory)

## **1-2. Run the Create and Package IP Wizard**

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

Create and Package New IP ×

### Peripheral Details

Specify name, version and description for the new peripheral

Name:

Version:

Display name:

Description:

IP location:  ⋮

☐ Overwrite existing

?
< Back
Next >
Finish
Cancel

Figure 3. Updating Peripheral Details form

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

1-2-6. Change the Name of the interface to **S\_AXI**

Create and Package New IP ×

### Add Interfaces

Add AXI4 interfaces supported by your peripheral

☐ Enable Interrupt Support

Interfaces

- S\_AXI

led\_ip\_v1.0

Name:

Interface Type:

Interface Mode:

Data Width (Bits):

Memory Size (Bytes):

Number of Registers:  [4..512]

?
< Back
Next >
Finish
Cancel

Figure 4. Naming the AXI interface

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

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.
- is the top level file containing a module (**led\_ip\_v1\_0\_S\_AXI**), which implements the AXI interfacing logic, and is 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.

You need to create:

- at the top level of the design - a new parameterized output port for LEDs
- at the module (**led\_ip\_v1\_0\_S\_AXI**) - a new parameterized output port for LEDs

1-3-2. In the file **led\_ip\_v1\_0.v** scroll down to line 7 where a user *parameters* space is provided

1-3-3. Add the line (It is a parameter **LED\_WIDTH** definition. The parameter defines width of a bus for the LED):

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

(Notice the comma needed at the end the line)

1-3-4. Go to line 18 and add the line (It is a declaration of an output port for the top level file):

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

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

```

c:/Xilinx_trn/Zynq_base/led_ip/ip_repo/led_ip_1.0/hdl/led_ip_v1_0.v
1
2  `timescale 1 ns / 1 ps
3
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
20     // Do not modify the ports beyond this line

```

Figure 5. Adding users parameter, and port definition

- 1-3-5. Go to line 47 and insert the following line between line 47 and 48 (It is a parameter **LED\_WIDTH** definition for the low level module **led\_ip\_v1\_0\_S\_AXI**):

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

- 1-3-6. Go to line 51 and insert the following line between line 51 and 52:

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

```

c:/Xilinx_trn/Zynq_base/led_ip/ip_repo/led_ip_1.0/hdl/led_ip_v1_0.v
45 );
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),
55     .S_AXI_AWADDR(s_axi_awaddr),
56     .S_AXI_AWPROT(s_axi_awprot),
57     .S_AXI_AWVALID(s_axi_awvalid),
58     .S_AXI_AWREADY(s_axi_awready),
59     .S_AXI_WDATA(s_axi_wdata),
60     .S_AXI_WSTRB(s_axi_wstrb),

```

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

1-3-7. Save the file by clicking the tool Save File



1-3-8. Expand `led_ip_v1_0` in the sources view, if necessary, and open `led_ip_v1_0_S_AXI.v` – it is a file with a description of the low level module ( `led_ip_v1_0_S_AXI` ).

1-3-9. Add the LED parameter at line 7

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

1-3-10. Add the LED port at line 18

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

The screenshot shows the code editor for `led_ip_v1_0_S_AXI.v`. The file path is `c:/Xilinx_trn/Zynq_base/led_ip/ip_repo/led_ip_1.0/hdl/led_ip_v1_0_S_AXI.v`. The code is as follows:

```

1
2  `timescale 1 ns / 1 ps
3
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
20         // Do not modify the ports beyond this line

```

Green arrows point to line 7 (the `LED_WIDTH` parameter) and line 18 (the `LED` output port).

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

1-3-11. Scroll down to ~line 400 and insert the following code to instantiate the user logic for the LED IP

(This code can be typed directly, or copied from the  
**C:\Xilinx\_trn\Zynq\_base\lab\_sources\lab3\user\_logic\_instantiation.txt** file)

c:/Xilinx\_trn/Zynq\_base/led\_ip/ip\_repo/led\_ip\_1.0/hdl/led\_ip\_v1\_0\_S\_AXI.v

```

399
400     // Add user logic here
401     lab3_user_logic # (
402         .LED_WIDTH(LED_WIDTH)
403     )
404     U1(
405         .S_AXI_ACLK(S_AXI_ACLK),
406         .slv_reg_wren(slv_reg_wren),
407         .axi_awaddr(axi_awaddr[C_S_AXI_ADDR_WIDTH-1:ADDR_LSB]),
408         .S_AXI_WDATA(S_AXI_WDATA),
409         .S_AXI_ARESETN(S_AXI_ARESETN),
410         .LED(LED)
411     );
412     // User logic ends
413
414     endmodule

```

**Figure 8. Instantiating lower-level user module**

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

**1-3-12.** Save the file.

**1-3-13.** Click on the *Add Sources* in the Flow Navigator pane.

**1-3-14.** Select *Add or Create Design Sources*.

**1-3-15.** Click Next, then click **Add Files...**,

**1-3-16.** Browse to **C:\Xilinx\_trn\Zynq\_base\lab\_sources\lab3**

**1-3-17.** 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.

**1-3-18.** Click **Run Synthesis** and click **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).

**1-3-19.** When Synthesis completes successfully, click **Cancel**.

**1-3-20.** **Check the Messages tab for any errors and correct if necessary before moving to the next step**



## 1-4. Package the IP

### 1-4-1. Select **Package IP – led\_ip** tab

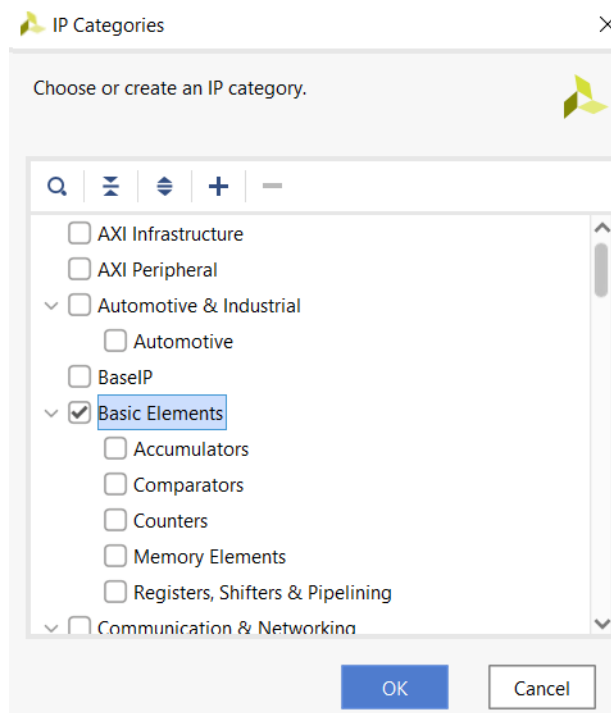
The screenshot shows the 'Package IP - led\_ip' tab selected in the Vivado IDE. The 'Identification' section is active, displaying the following fields:

Field	Value
Vendor:	xilinx.com
Library:	user
Name:	led_ip
Version:	1.0
Display name:	led_ip_v1.0
Description:	My new AXI IP
Vendor display name:	
Company url:	
Root directory:	c:/Xilinx_trn/Zynq_base/led_ip/ip_repo/led_ip_1.0
Xml file name:	c:/Xilinx_trn/Zynq_base/led_ip/ip_repo/led_ip_1.0/component.xml

The 'Categories' section shows a list with 'AXI\_Peripheral' and a plus button to add more categories.

**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 **Plus** in the *Categories* section. This opens the 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 **Plus** then **Add Family Explicitly...** from the menu.
- 1-4-6. Select the **Zynq** family and Life-cycle as **Production** 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.
- 1-4-8. Click on **File Groups** and click *Merge changes from File Groups Wizard*

**File Groups**

Q [Icons] + C

Name	Library Name	Type	Is Include	Used In Constant	File Group Name	Model Name
Standard			<input type="checkbox"/>	<input type="checkbox"/>		
Advanced			<input type="checkbox"/>	<input type="checkbox"/>		
Verilog Synthesis (3)			<input type="checkbox"/>	<input type="checkbox"/>		led_ip_v1_0
src/lab3_user_logic.v		verilogSource	<input type="checkbox"/>	<input type="checkbox"/>	xilinx_verilogsynthesis	
hdl/led_ip_v1_0_S_AXI.v		verilogSource	<input type="checkbox"/>	<input type="checkbox"/>	xilinx_verilogsynthesis	
hdl/led_ip_v1_0.v		verilogSource	<input type="checkbox"/>	<input type="checkbox"/>	xilinx_verilogsynthesis	
Verilog Simulation (3)			<input type="checkbox"/>	<input type="checkbox"/>		led_ip_v1_0
src/lab3_user_logic.v		verilogSource	<input type="checkbox"/>	<input type="checkbox"/>	xilinx_verilogbehavioralsimulation	
hdl/led_ip_v1_0_S_AXI.v		verilogSource	<input type="checkbox"/>	<input type="checkbox"/>	xilinx_verilogbehavioralsimulation	
hdl/led_ip_v1_0.v		verilogSource	<input type="checkbox"/>	<input type="checkbox"/>	xilinx_verilogbehavioralsimulation	
UI Layout (1)			<input type="checkbox"/>	<input type="checkbox"/>		
xgui/led_ip_v1_0.tcl		tclSource	<input type="checkbox"/>	<input type="checkbox"/>	xilinx_xpgui	
Software Driver (6)			<input type="checkbox"/>	<input type="checkbox"/>		
drivers/led_ip_v1_0/src/Makef...		driver_src	<input type="checkbox"/>	<input type="checkbox"/>	xilinx_softwaredriver	
drivers/led_ip_v1_0/src/led_ip...		cSource driver_src	<input type="checkbox"/>	<input type="checkbox"/>	xilinx_softwaredriver	
drivers/led_ip_v1_0/src/led_ip.h		cSource driver_src	<input type="checkbox"/>	<input type="checkbox"/>	xilinx_softwaredriver	
drivers/led_ip_v1_0/src/led_ip.c		cSource driver_src	<input type="checkbox"/>	<input type="checkbox"/>	xilinx_softwaredriver	
drivers/led_ip_v1_0/data/led_i...		tclSource driver_...	<input type="checkbox"/>	<input type="checkbox"/>	xilinx_softwaredriver	
drivers/led_ip_v1_0/data/led_i...		mdd driver_mdd	<input type="checkbox"/>	<input type="checkbox"/>	xilinx_softwaredriver	
Block Diagram (1)			<input type="checkbox"/>	<input type="checkbox"/>		
bd/bd.tcl		tclSource	<input type="checkbox"/>	<input type="checkbox"/>	bd_tcl	

**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* and notice **lab3\_user\_logic.v** has been included

**1-4-10.** Click on **Customization Parameters** and again *Merge changes from Customization Parameters Wizard*

**Packaging Steps**

- ✓ Identification
- ✓ Compatibility
- ✓ File Groups
- ✓ Customization Parameters
- ✓ Ports and Interfaces
- ✓ Addressing and Memory
- ✓ Customization GUI

**Customization Parameters**

Name	Description	Display Name	Value
Customization Parameters			
C_S_AXI_DATA_WIDTH	Width of S_AXI data bus	C S AXI DATA WIDTH	32
C_S_AXI_ADDR_WIDTH	Width of S_AXI address bus	C S AXI ADDR WIDTH	4
C_S_AXI_BASEADDR		C S AXI BASEADDR	0xFFFFFFFF
C_S_AXI_HIGHADDR		C S AXI HIGHADDR	0x00000000
Hidden Parameters			
LED_WIDTH		Led Width	8

**Packaging Steps**

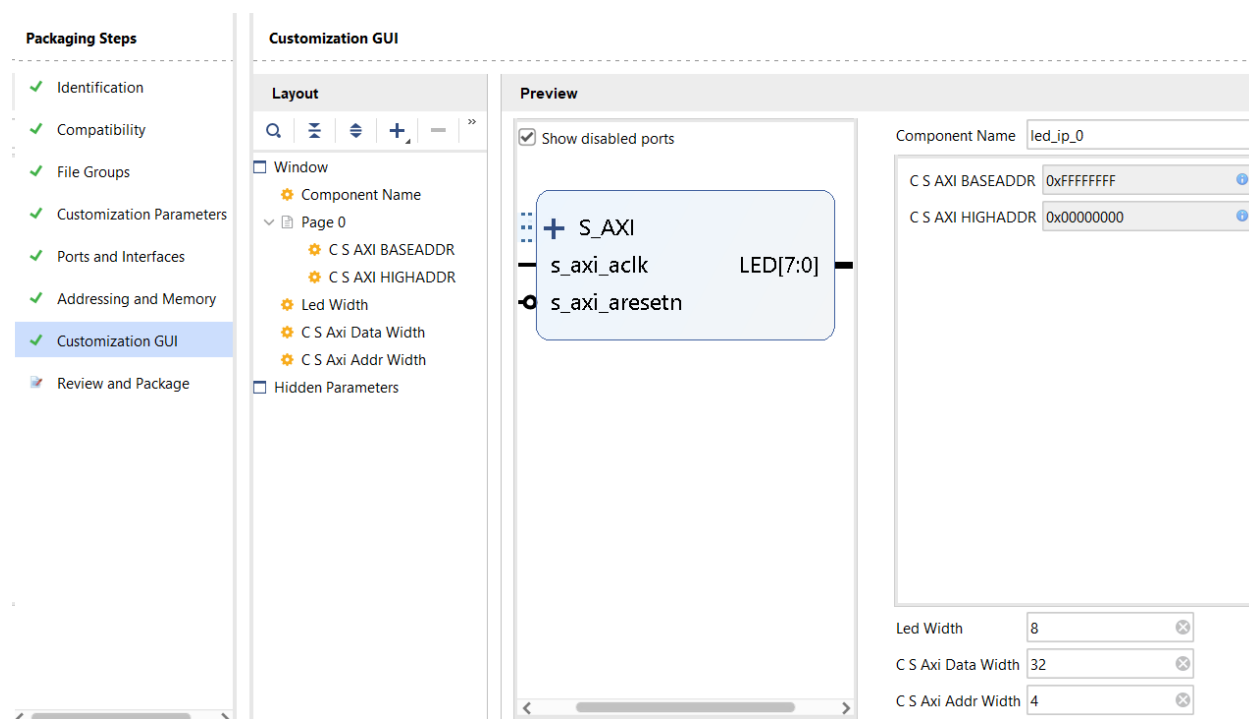
- ✓ Identification
- ✓ Compatibility
- ✓ File Groups
- ✓ Customization Parameters
- ✓ Ports and Interfaces
- ✓ Addressing and Memory
- ✓ Customization GUI

**Ports and Interfaces**

Name	Interface Mode	Enablement Dependency	Is Declaration	Access Handle	Access Type	Direction
S_AXI	slave		<input type="checkbox"/>			
Clock and Reset Signals			<input type="checkbox"/>			
LED			<input type="checkbox"/>		ref	out

Figure 12. User parameter and port

- 1-4-11. Click on **Ports and Interfaces**. Notice that the Ports and Interfaces view now shows the user created LED port
- 1-4-12. Select **Customization Parameters** again, expand *Hidden Parameters*, right-click on **LED\_WIDTH**, and select *Import IP Parameters...* and click **OK**.
- 1-4-13. Select **Customization GUI** and notice that the *Led Width* is visible.



**Figure 13. User customization parameter**

**1-4-14.** Select *Review and Package*, and notice the path (c:/Xilinx\_trn/Zynq\_base/led\_ip/ip\_repo/led\_ip\_1.0) where the IP will be created.

**1-4-15.** Click **Re-Package IP**. Click **Yes** and the project will close when complete.

**1-4-16.** In the original Vivado window click **File > Close Project**

## Modify the Project Settings

## Step 2

### 2-1. Open the previous (lab2) project and save the project as lab3. Set Project Settings to point to the created IP repository.

2-1-1. Start the Vivado 2018.3 if necessary and open the lab2 project you created in the previous lab.

2-1-2. Select **File > Project > Save As ...** to open the *Save Project As* dialog box.

2-1-3. Enter **lab3** as the project name.

2-1-4. Make sure that the *Create Project Subdirectory* option is checked and the *Project location* is **C:\Xilinx\_trn\Zynq\_base** and click **OK**.

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

2-1-5. Click **Settings** in the *Flow Navigator* pane.

2-1-6. In the left pane of the *Project Settings* form select **IP=>Repository**.

2-1-7. Click on the **Plus** button, browse to **C:\Xilinx\_trn\Zynq\_base\led\_ip** and click **Select**. Then click **OK**.

The led\_ip\_v1.0 IP will appear the **IP in the Selected Repository** window.

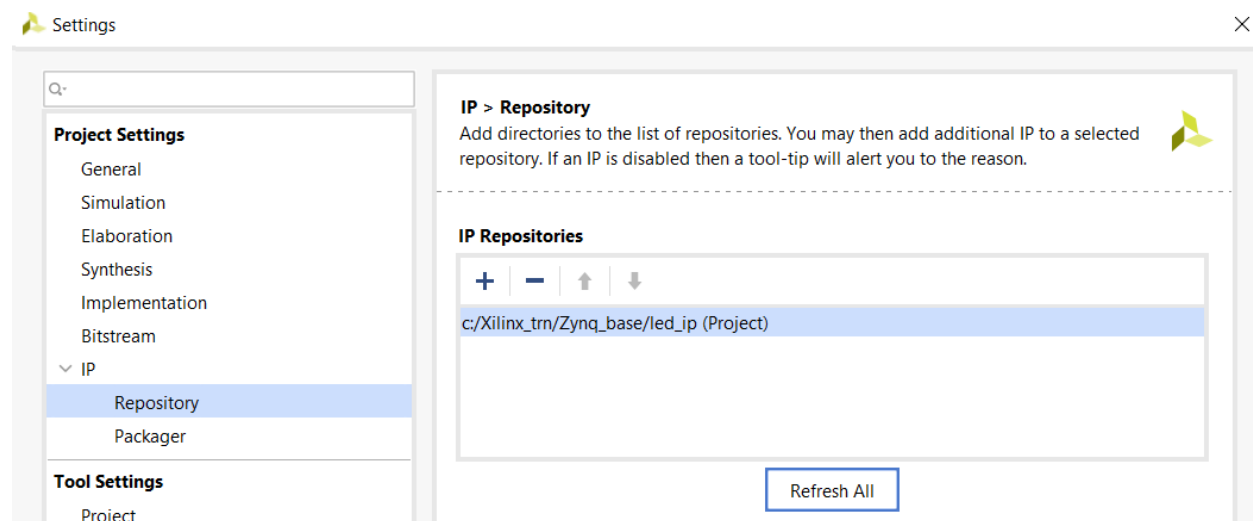


Figure 14. Specify IP Repository


2-1-8. Click **OK**.

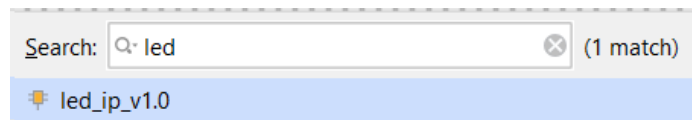
## Add the Custom IP, BRAM, and the Constraints

## Step 3

**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.



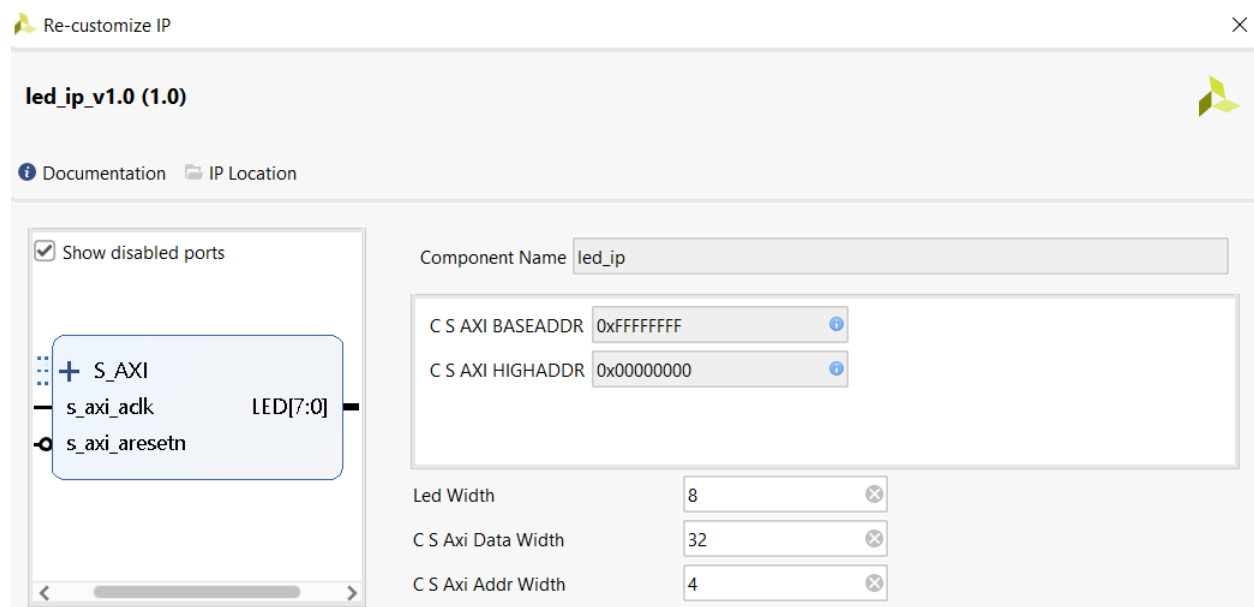
**Figure 15. Searching for led\_ip in the IP Catalog**

**3-1-3.** Double-click **led\_ip\_v1\_0** to add the core to the design.

**3-1-4.** Select the IP 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.** You should have the window like the figure below.



**Figure 16. Configure the LED IP**

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

- 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.
- 3-1-9. Select the **LED** port on the **led\_ip** instance (by clicking on its pin), right-click and select **Make External**.
- 3-1-10. Select external port and in External Port Properties plane rename external port from LED\_0 to **LED**.
- 3-1-11. Click the regenerate button (↺) to redraw the diagram.

The design should look similar to the figure below.

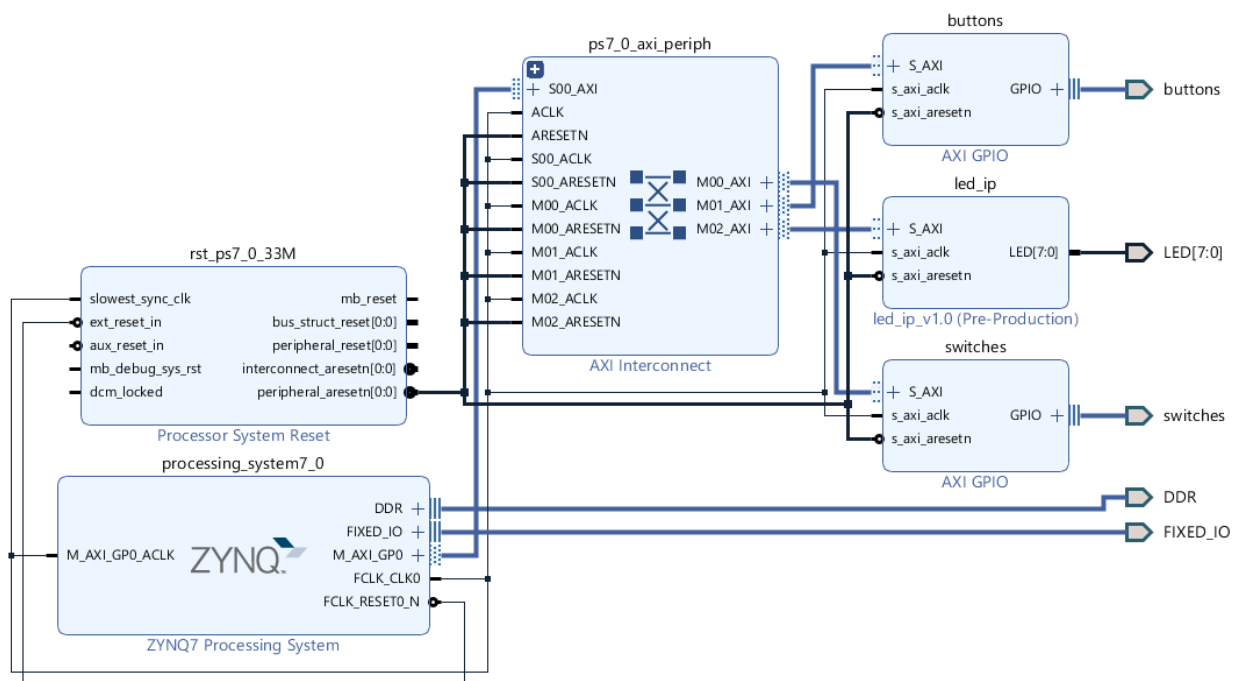


Figure 17. LED external port added and connected

- 3-1-12. Select the **Address Editor** tab and verify that an address has been assigned to **led\_ip**.


Diagram x Address Editor x

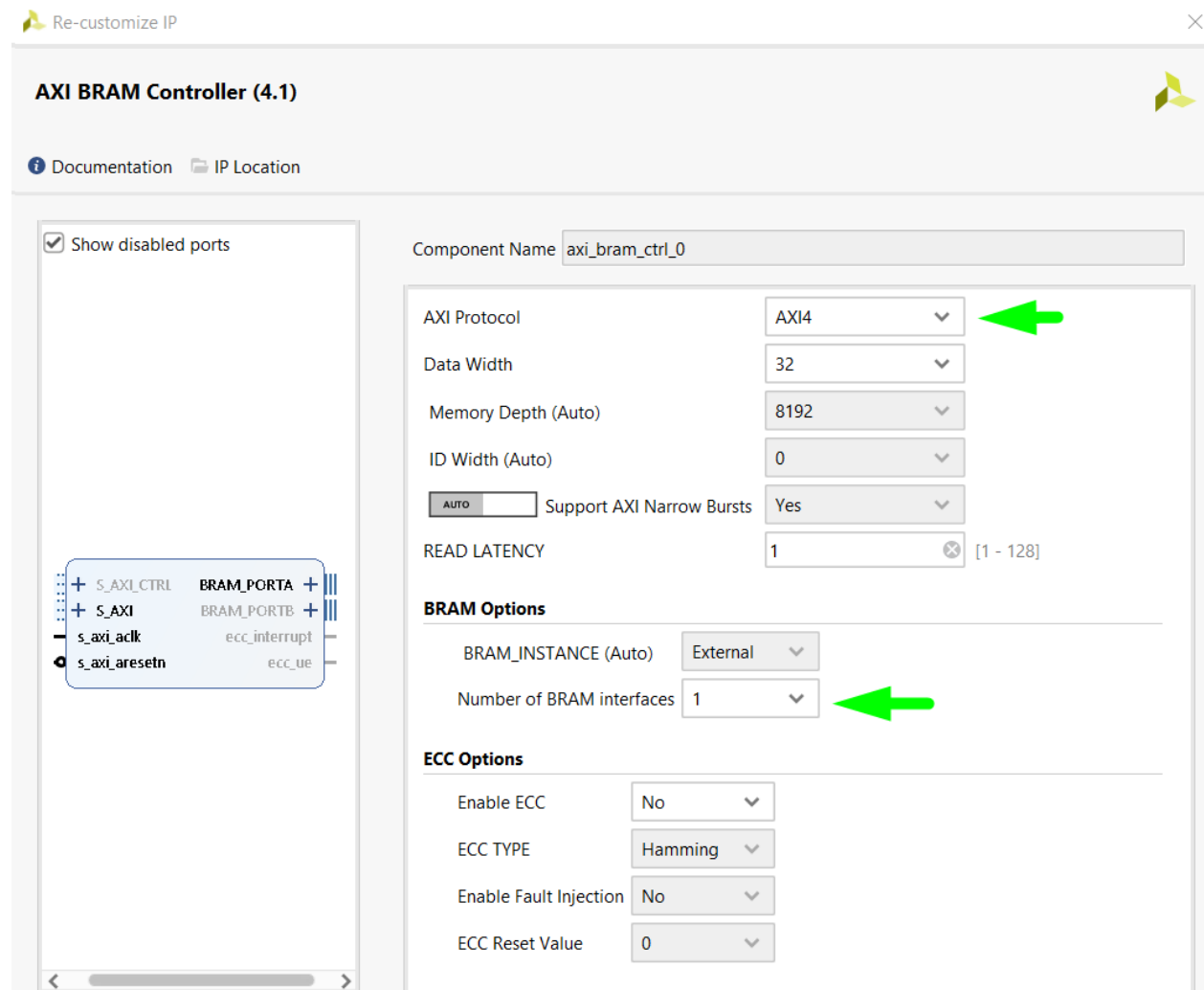
Cell	Slave Interface	Base Name	Offset Address	Range	High Address
▼  processing_system7_0					
▼  Data (32 address bits : 0x40000000 [ 1G ])					
buttons	S_AXI	Reg	0x4121_0000	64K ▼	0x4121_FFFF
switches	S_AXI	Reg	0x4120_0000	64K ▼	0x4120_FFFF
led_ip	S_AXI	S_AXI_reg	0x43C0_0000	64K ▼	0x43C0_FFFF

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  then search for BRAM and add one instance of the **AXI BRAM Controller**
- 3-2-2. Run *Connection Automation* on **axi\_bram\_ctrl\_0/S\_AXI**.
- 3-2-3. Double click on the block to customize it.



**Figure 19. Customize BRAM controller**

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.

- 3-2-4. Change the number of BRAM interfaces to 1 and click **OK**.
- 3-2-5. Click on *Run Connection Automation* to add and connect a **Block Memory Generator** by selecting **axi\_bram\_ctrl\_0/BRAM\_PORTA** and click **OK** (This could be added manually.)
- 3-2-6. 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.

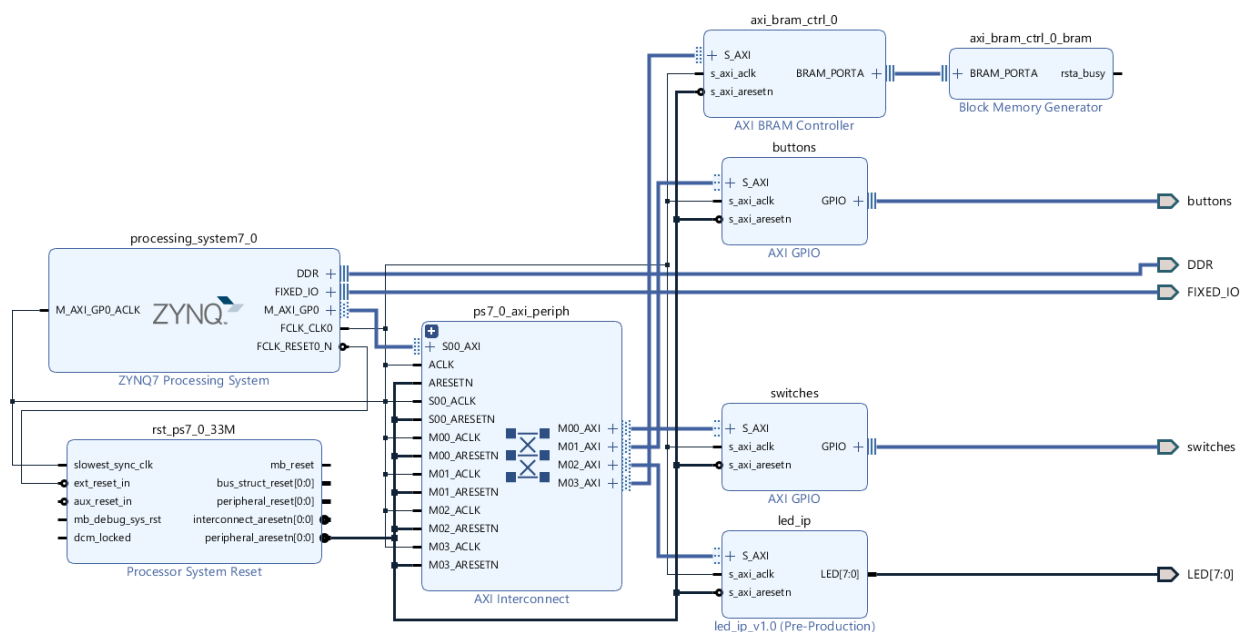


Figure 20. Completed Block Diagram

3-2-7. In the Address editor, notice the Range of the `axi_bram_ctrl_0` is 8K.

Address Editor

Diagram

Cell	Slave Interface	Base Name	Offset Address	Range	High Address
▼  processing_system7_0					
▼  Data (32 address bits : 0x40000000 [ 1G ])					
buttons	S_AXI	Reg	0x4121_0000	64K ▼	0x4121_FFFF
switches	S_AXI	Reg	0x4120_0000	64K ▼	0x4120_FFFF
led_ip	S_AXI	S_AXI_reg	0x43C0_0000	64K ▼	0x43C0_FFFF
axi_bra...	S_AXI	Mem0	0x4000_0000	8K ▼	0x4000_1FFF

Figure 21. Adjusting memory size

3-2-8. Save block diagram.

3-2-9. Press **F6** to validate the design one last time.

### 3-3. Add the provided lab3\_\*.xdc constraints file.

The file defines pins locations and properties for the LED[7:0] external port.

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

3-3-2. Click the **Add Files...**, browse to the **C:\Xilinx\_trn\Zynq\_base\lab\_sources\lab3** folder, select **lab3\_zed.xdc** and click **OK**.

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

3-3-4. Expand Constraints folder in the *Sources* pane, and double click the **lab3\_zed.xdc** file entry to see its content.

This file contains the pin locations and IO standards for the LEDs on the Zed board. This information can usually be found in the manufacturer's datasheet for the board.

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

3-3-6. Click on Run Synthesis, the click OK and click OK one more time.

3-3-7. Open Design Runs tab to see a status of the current run.

3-3-8. In the window appeared select Run Implementation and click OK. Click OK one more time.

3-3-9. In the window appeared select Generate Bitstream and click OK. Click OK one more time.

3-3-10. In the window appeared click **Cancel**.

3-3-11. Close Vivado.

## 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.