

Introduction to Zynq Hardware

Lab 7

Adding Custom IP to Vivado IP Catalog



November 2013
Version 03

Lab 7 Overview

Xilinx offers a diverse IP catalog which makes it very easy to connect many of the common interfaces to your design. But the real value of Zynq is combining custom IP to the ARM dual core processing system. This lab will provide step-by-step instructions on how to create custom IP, add it to the IP catalog and then connect it into our design.

Lab 7 Objectives

When you have completed Lab 7, you will know how to do the following:

- Create a new IP project
- Customize the IP
- Add it to the Vivado IP Catalog
- Add this custom IP to your project
- Add an interrupt to the Zynq PS and connect to the custom IP
- Test the custom IP with a custom software application

Experiment 1: Create New IP Project

This experiment shows how to create a new IP project, as well as define the AXI interface and register settings for the IP.

Experiment 1 General Instruction:

Create a new AXI peripheral.

Experiment 1 Step-by-Step Instructions:

1. <Optional> If you did not complete Lab 6 or wish to start with a clean copy, delete the `ZynqDesign`, `IP` and `SDK_Workspace` folders in the `ZynqHW/2013_3` folder. Then unzip `Solutions\ZynqHW_Lab6_Solution.zip` to the `2013_3` folder. If you have 7-zip installed, you can do this by right-clicking and dragging `ZynqHW_Lab6_Solution.zip` to the `2013_3` folder. Select **7-Zip → Extract Here**.
2. Open **Vivado** with the existing project.

3. Select **Tools** → **Create and Package IP**.

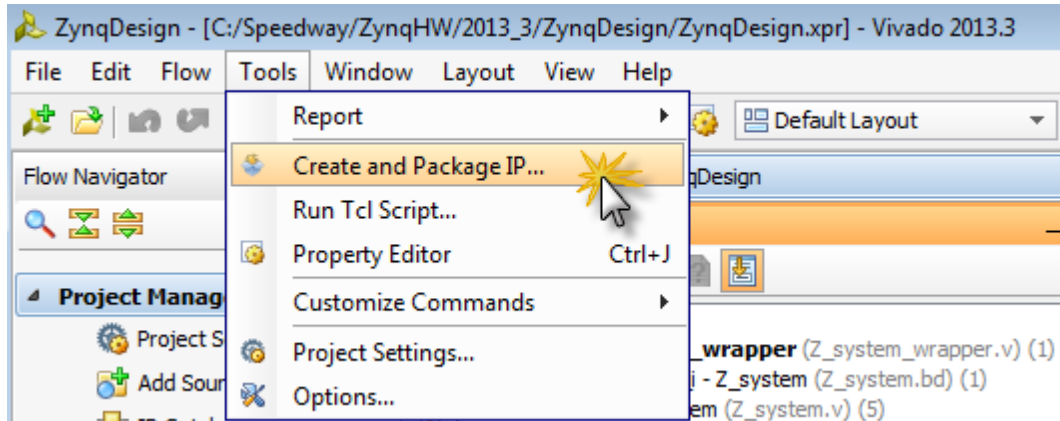


Figure 1 - Create New IP

4. Click **Next >** on Welcome Screen to Create and Package IP.
5. We will be **creating a new AXI peripheral**, check the corresponding box. Additionally, we want to create this IP in a IP repository:
C:/Speedway/ZynqHW/2013_3/IP

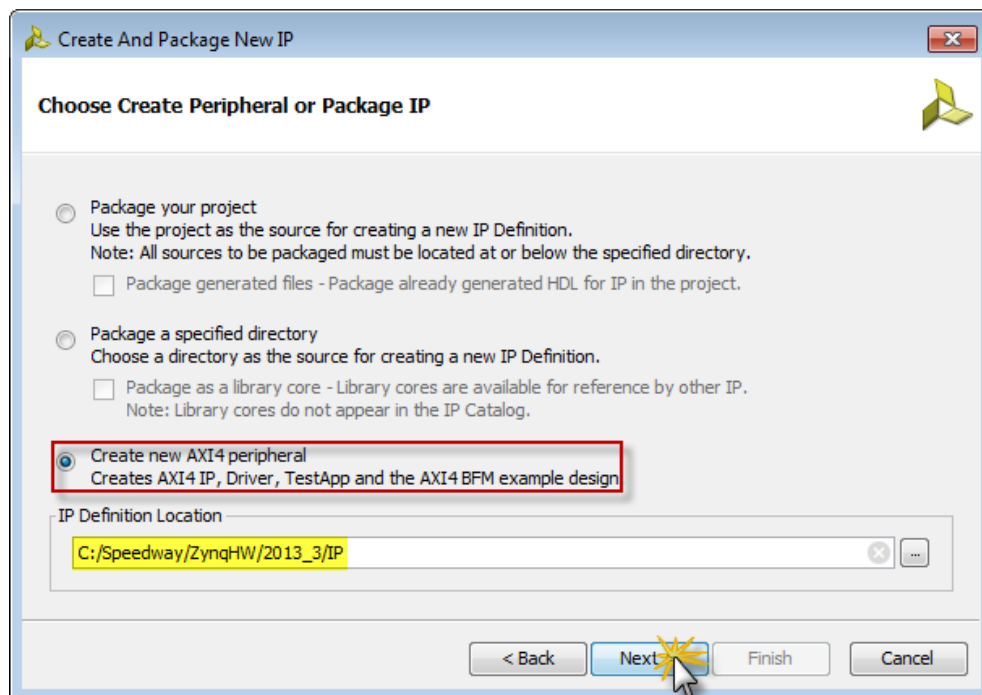
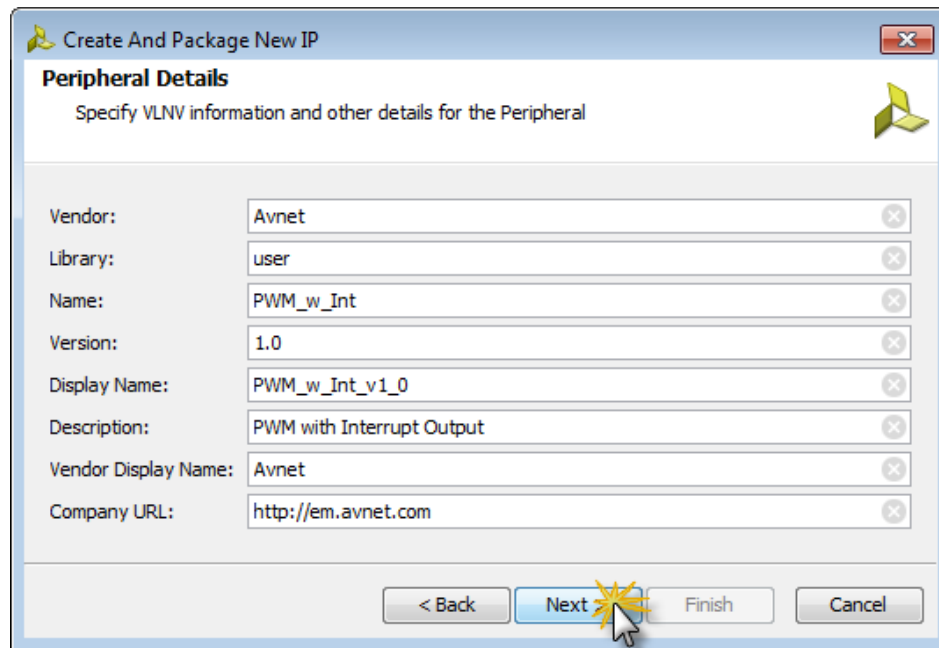


Figure 2 - Create a New AXI Peripheral

6. Click **Next >**.

7. Enter the following fields as shown here. The only critical fields are **Library**, as we will create this in our USER library, and **Name** and **Display Name**. When done, click **Next >**.



Create And Package New IP

Peripheral Details
Specify VLN information and other details for the Peripheral

Vendor: Avnet

Library: user

Name: PWM_w_Int

Version: 1.0

Display Name: PWM_w_Int_v1_0

Description: PWM with Interrupt Output

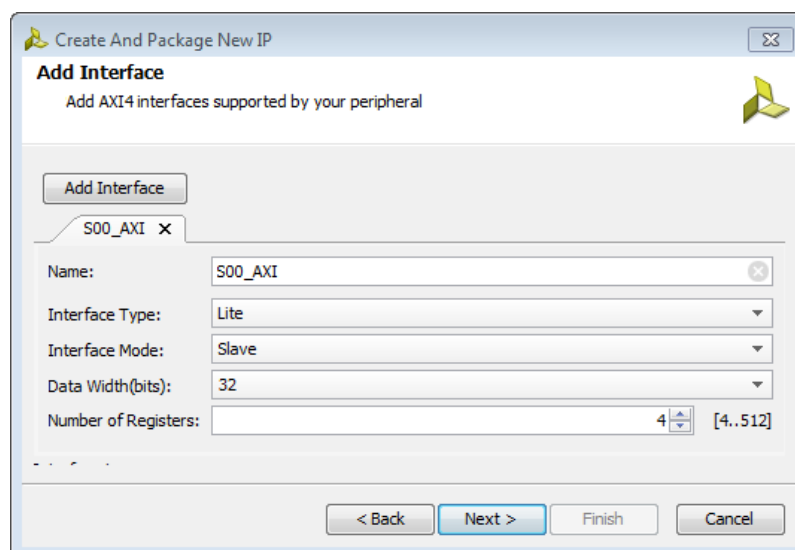
Vendor Display Name: Avnet

Company URL: http://em.avnet.com

< Back Next > Finish Cancel

Figure 3 - IP Details

8. This peripheral is simple and does not require much bandwidth, thus an AXI-Lite interface is acceptable. This will also be a slave device with the PS as the Master. A 32-bit interface is good and we only really need 1 register, but four is the lowest setting. Thus the default parameters as acceptable. Click **Next >**.



Create And Package New IP

Add Interface
Add AXI4 interfaces supported by your peripheral

Add Interface

S00_AXI x

Name: S00_AXI

Interface Type: Lite

Interface Mode: Slave

Data Width(bits): 32

Number of Registers: 4 [4..512]

< Back Next > Finish Cancel

Figure 4 - IP Interface Settings

9. Check the box for **Generate Drivers** as this will make our IP show up in the peripheral list in our hardware definition for SDK. Click **Next** >.

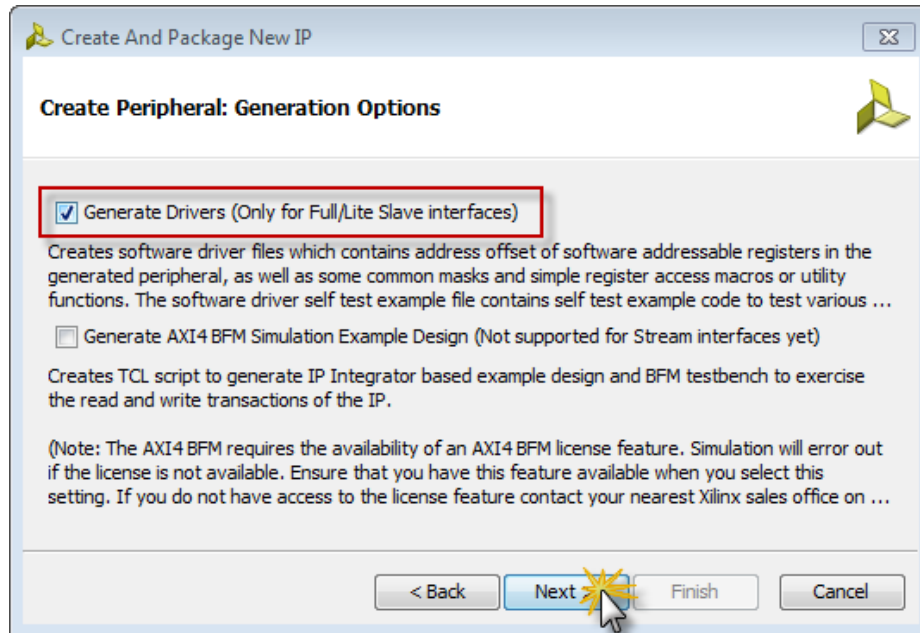


Figure 5 - IP Generation Options

10. Check the **Add IP to the catalog and open IP in editing session**. Verify the *Output Products* and *IP Repository path*. If OK, click **Finish**.

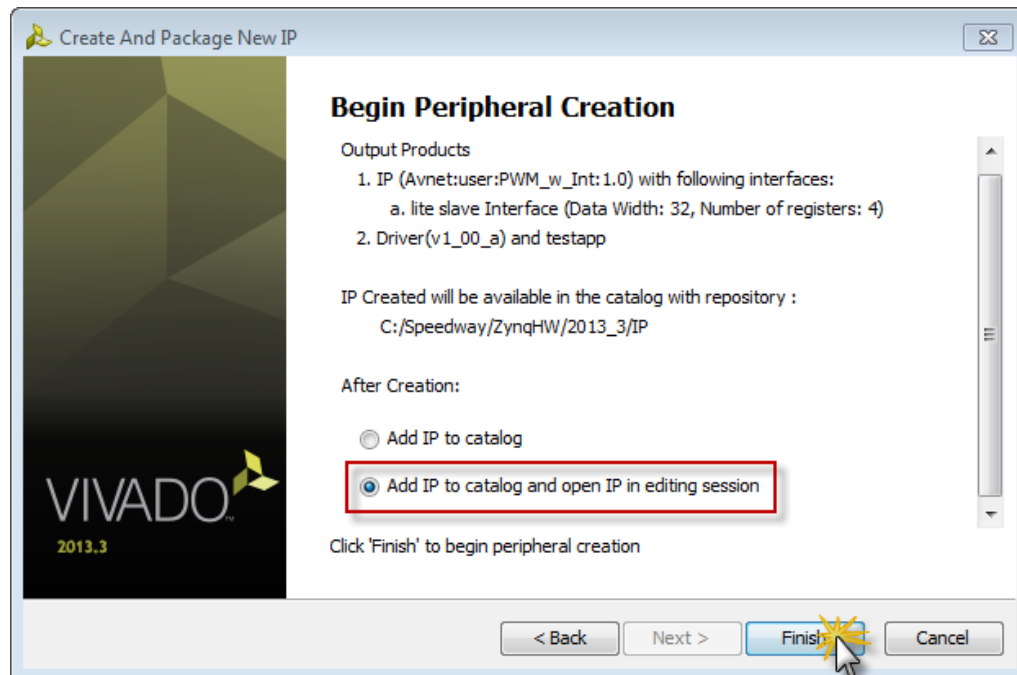


Figure 6 - IP Creation Summary

This will open a new Vivado Project when completed. In this new project we will edit our IP. Take note of the Root Directory and Project Name. This is important for when you want to edit this IP at a later date. The **edit_<IP Name>.xpr** project is the project you will open to edit this IP. In this case, our IP project is named, **edit_PWM_w_Int_v1_0.xpr**.

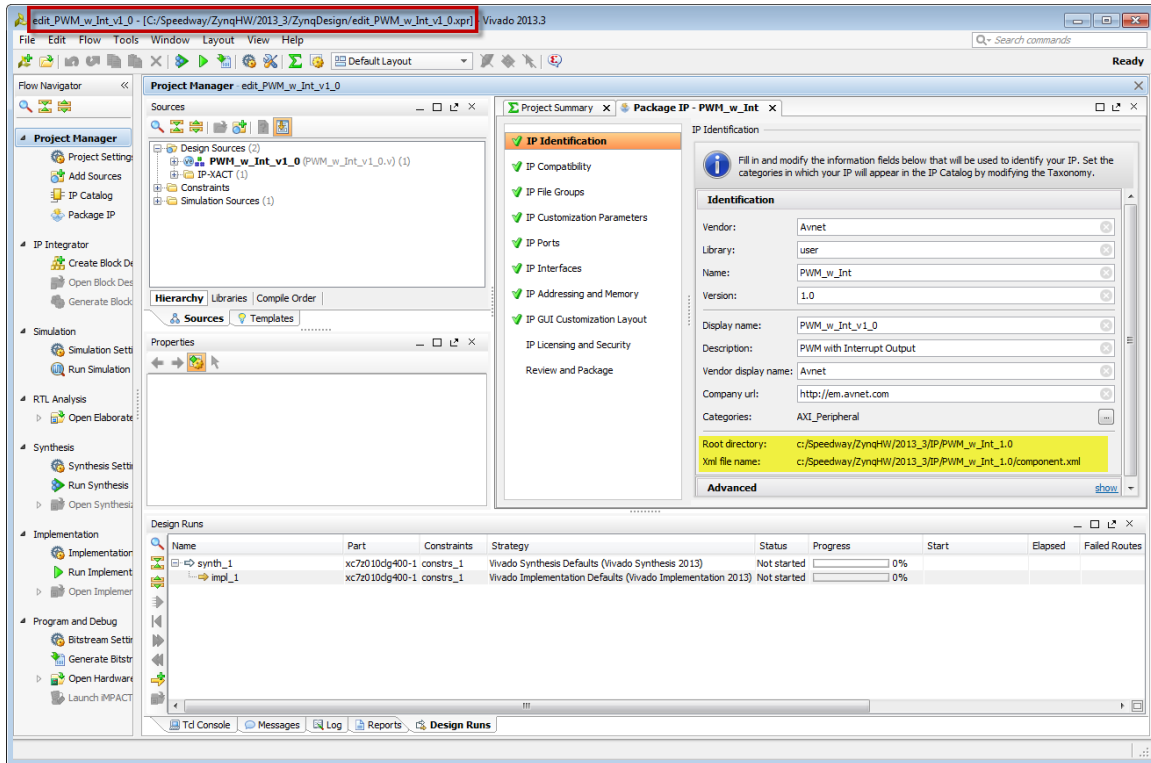


Figure 7 - New IP Project

Questions:

Answer the following questions:

- Where is the IP project located?

Experiment 2: Customize the New IP Project

This experiment shows how to bring in custom user logic and instantiate it into the new IP project. Additionally, we'll customize the IP project adding new port as well as parameter inputs.

Experiment 2 General Instruction:

Import user logic HDL, PWM_Controller_Int.v, and instantiate it into the project. Connect to slv_reg0 from the AXI interface, add PWM outputs to top-level.

Experiment 2 Step-by-Step Instructions:

1. Expand the Design Sources in the Sources window. Notice, when a file is selected in the sources window, its properties show up in the source file properties window. This is helpful in determining where your HDL files exist on your PC.

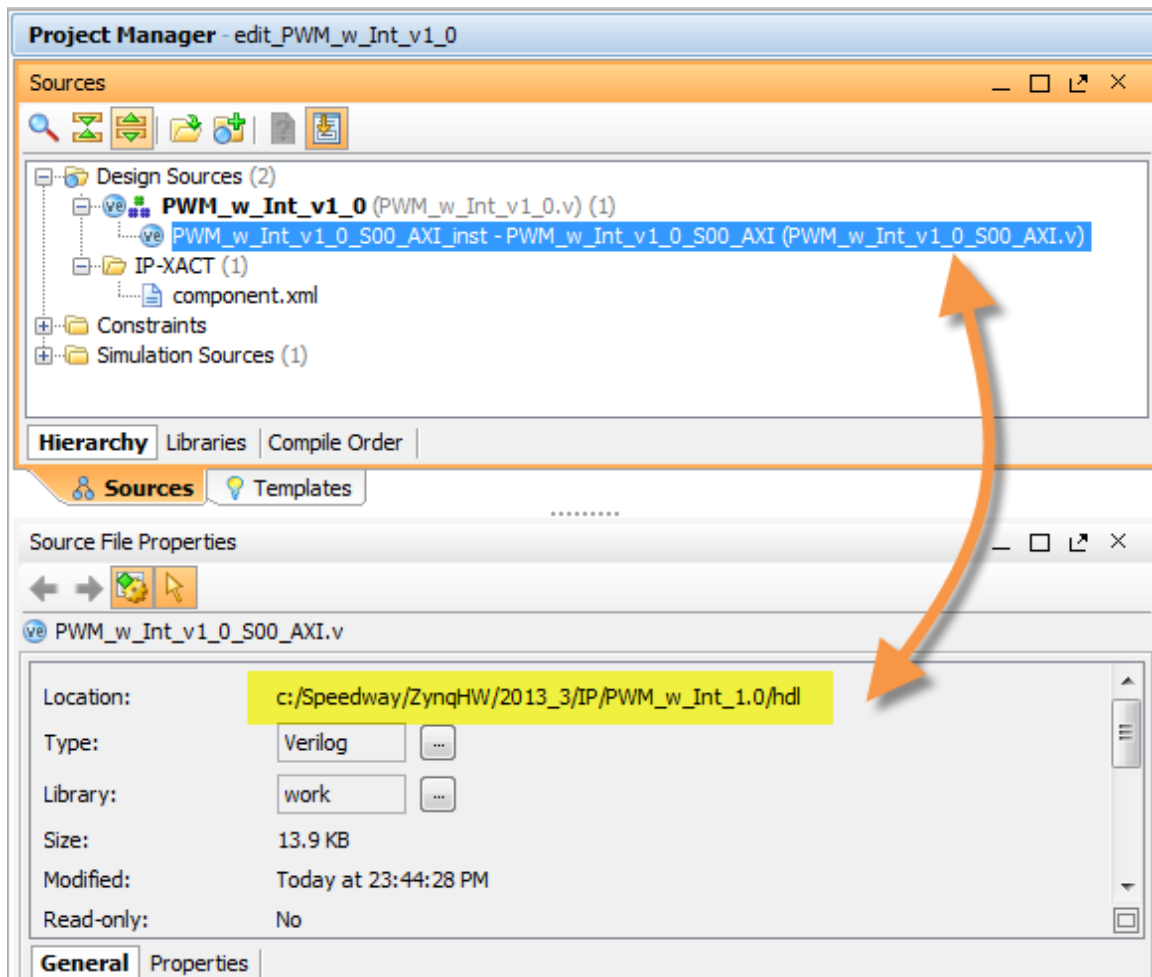


Figure 8 - Design Sources and Properties

2. There are two HDL files. One, **PWM_w_Int_v1_0.v**, is the top-level wrapper file. **Open** this file by double-clicking on it. It contains the AXI interface HDL code, **PWM_w_Int_v1_0_S00_AXI.v** around line 44. And a place holder exists just below this to add user logic, which is where we will add our custom IP HDL.

```
// Instantiation of Axi Bus Interface S00_AXI
PWM_w_Int_v1_0_S00_AXI # (
    .C_S_AXI_DATA_WIDTH(C_S00_AXI_DATA_WIDTH),
    .C_S_AXI_ADDR_WIDTH(C_S00_AXI_ADDR_WIDTH)
) PWM_w_Int_v1_0_S00_AXI_inst (
    .S_AXI_ACLK(s00_axi_aclk),
    .S_AXI_ARESETN(s00_axi_aresetn),
    .S_AXI_AWADDR(s00_axi_awaddr),
        .
        .
        .
    .S_AXI_ARREADY(s00_axi_arready),
    .S_AXI_RDATA(s00_axi_rdata),
    .S_AXI_RRESP(s00_axi_rresp),
    .S_AXI_RVALID(s00_axi_rvalid),
    .S_AXI_RREADY(s00_axi_rready)
);

// Add user logic here

// User logic ends

endmodule
```


3. To save time, a simple HDL core has been created. But, before we import this HDL code. We need to copy it into our HDL directory in our IP project. Using windows explorer, navigate to:

C:\Speedway\ZynqHW\2013_3\Support_documents

Copy **PWM_Controller_Int.v** and paste it into:

C:\Speedway\ZynqHW\2013_3\IP\PWM_w_Int_1.0\hdl

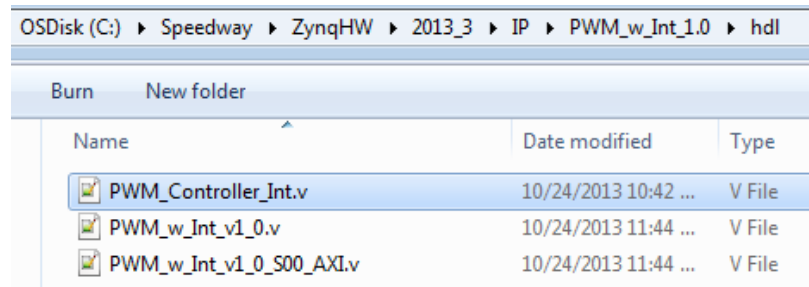


Figure 9 - Copy HDL Files into IP HDL Directory

4. Click **Add Sources** from the Flow Navigator window, and then select **Add or Create Design Source**. When done, click **Next >**.

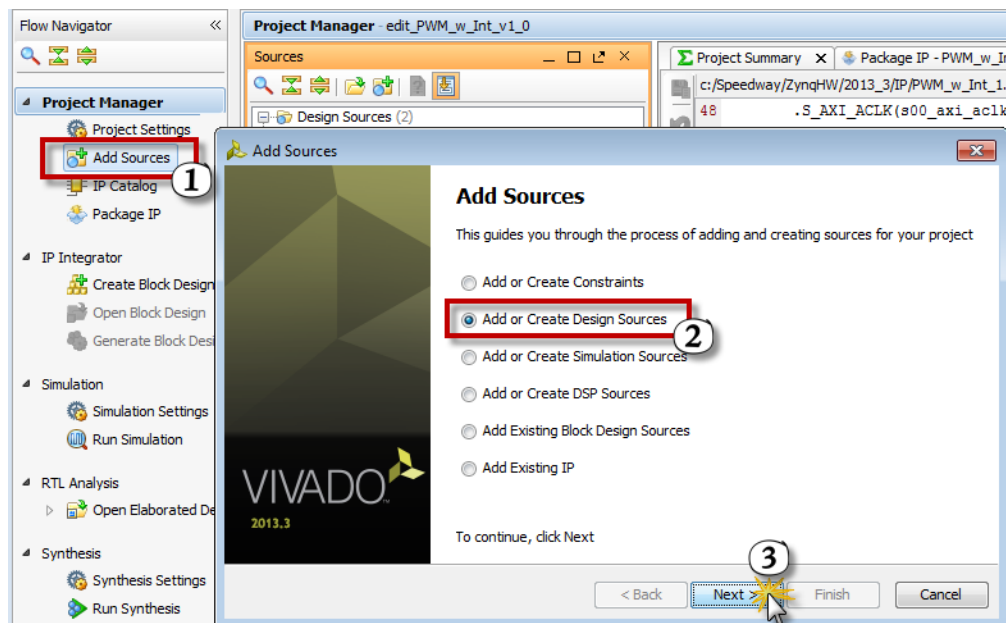


Figure 10 - Add Design Sources

5. In the Add Source Files window, Click **Add Files...**

6. Change the directory to:

C:\Speedway\ZynqHW\2013_3\IP\PWM_w_Int_1.0\hdl

Double-click **PWM_Controller_Int.v** to add it to the project.

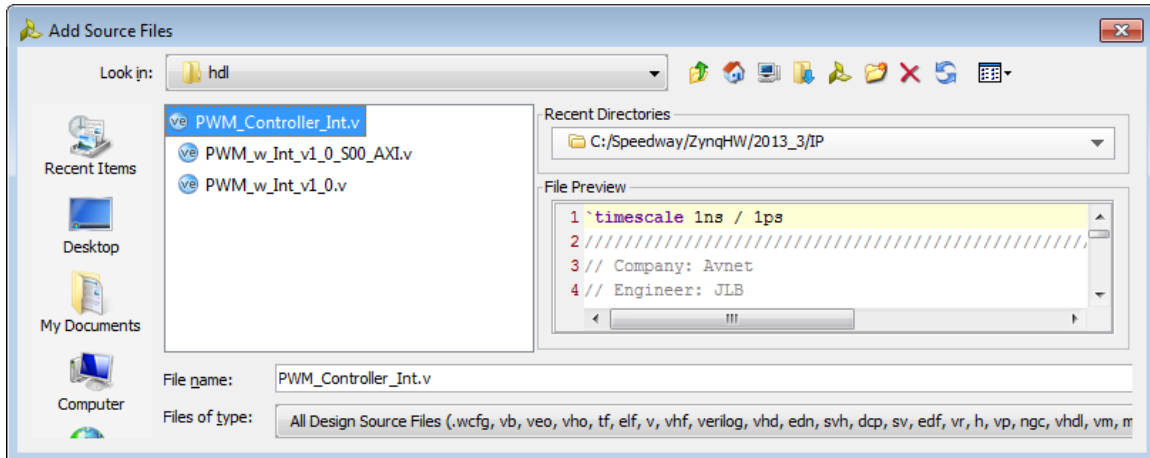


Figure 11 - Add Source Files

7. Verify the file. Also verify *copy sources into project* is not checked. Click **Finish**.

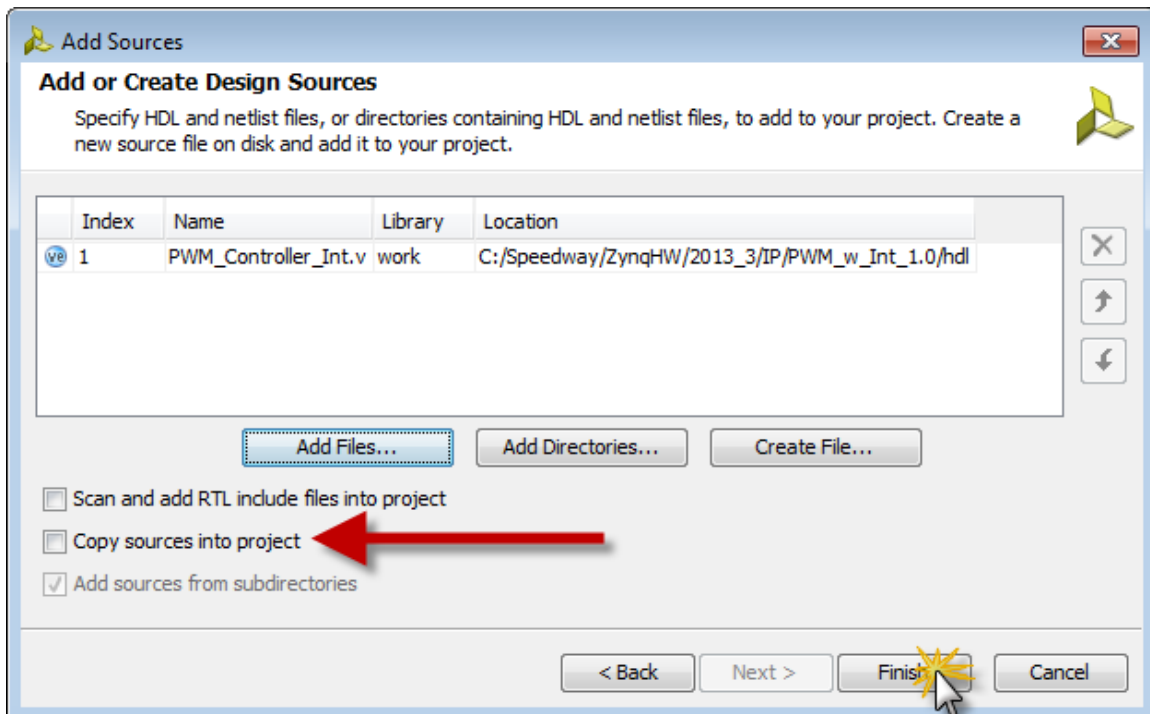


Figure 12 - Add Files to Project

8. The file will now show up in our Sources window. But not under our top-level file. This is expected as we have not instantiated this code in the top-level file yet.

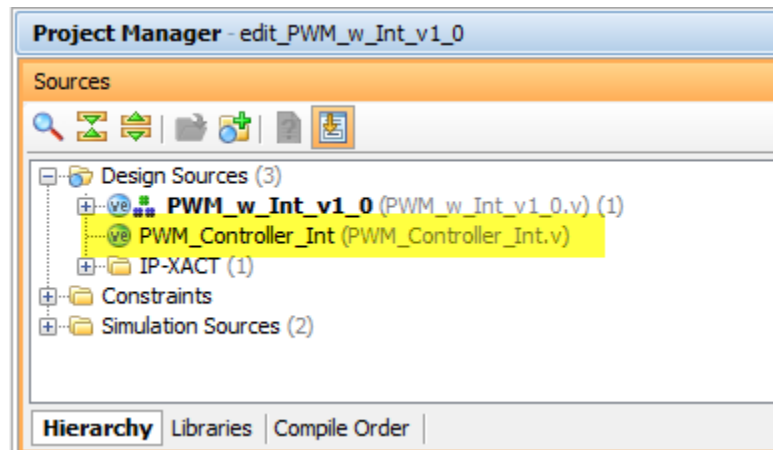


Figure 13 - New Source in Project

9. Open PWM_Controller_Int by double-clicking it. Look over the file and then instantiate it by entering the following code into PWM_w_Int_v1_0.v in the user logic section:

```
// Add user logic here
PWM_Controller_Int #(
    .period( )
) PWM_inst (
    .Clk( ),
    .DutyCycle( ),
    .Reset(),
    .PWM_out( ),
    .Interrupt( ),
    .count( )
);
// User logic ends

endmodule
```

10. Click **Save**, or Control-S.

11. Once entered, this HDL file will now be incorporated into the top level.

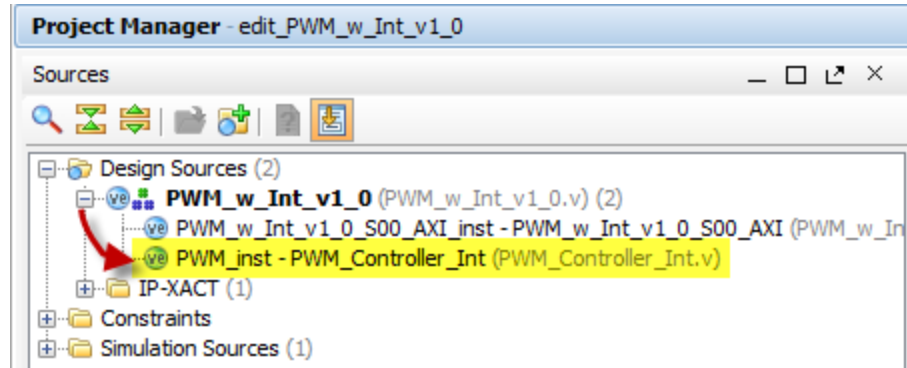


Figure 14 - User Logic now in Project Hierarchy

The next steps guide you through the methodology in connecting the User Logic HDL into the IP Core. If you are comfortable with this process, you can copy the final HDL files from *ZynqHW_Lab7_Solution.zip* in the *Solutions* folder into your IP/PWM_w_Int1.0/hdl folder. If you choose this, skip to step 18, else continue.

12. Now we need to connect the signals of our user logic, PWM_Controller_Int. Start by connecting the easy signals. Connect the clock and reset. Connect:

s00_axi_aclk → Clk
s00_axi_aresetn → Reset

Continue editing PWM_w_Int_v1_0.v so the code looks like this:

```
PWM_Controller_Int #(
    .period( )
) PWM_inst (
    .Clk( s00_axi_aclk ),
    .DutyCycle(),
    .Reset( s00_axi_aresetn ),
    .PWM_out(),
    .Interrupt(),
    .count()
);
```

Next we'll connect the outputs. These signals need to be defined in the top-level port list. This IP will bring out four outputs;

- **LEDs:** The actual PWM output to be connected to the LEDs
- **Interrupt_out:** an Interrupt that indicates an invalid PWM setting
- **PWM_Counter:** (for debug) the free running PWM counter
- **DutyCycle:** (for debug) the value passed down from the PS via the AXI interface to control the duty cycle of the PWM.
- Also add a configurable parameter, **PWM_PERIOD**, that sets the counter depth of the PWM counter.

13. Add these ports at the top of the code as well as the instantiation of our user logic HDL:

```
module PWM_Control_w_INT_v1_0 #
(
    // Users to add parameters here
    parameter integer PWM_PERIOD = 20,
    // User parameters ends

    // Users to add ports here
    output wire Interrupt_out,
    output wire [7:0] LEDs,
    output wire [PWM_PERIOD-1:0] PWM_Counter,
    output wire [31:0] DutyCycle,
    // User ports ends

    PWM_Controller_Int #(
        .period( PWM_PERIOD )
    ) PWM_inst (
        .Clk( s00_axi_aclk ),
        .DutyCycle( DutyCycle ),
        .Reset( s00_axi_aresetn ),
        .PWM_out( LEDs ),
        .Interrupt( Interrupt_out ),
        .count( PWM_Counter )
    );
);
```

That fully connects our IP; however we have not connected to the source of DutyCycle. Again, this comes from the processor. Conveniently, when we went through the Create IP Peripheral wizard, Vivado created an AXI proxy for us.

14. Open **PWM_w_Int_v1_0_S00_AXI.v**. Look over the file, scroll down to 199. Here you will see the definitions for the four registers created by Vivado though this AXI proxy. We only need one of these registers, `slv_reg0`.
15. Make `slv_reg0` an output of this module. At line 14, add this output under User Ports:

```
// Users to add ports here
output [31:0] slv_reg0,
// User ports ends
```

16. **Save** the file, CTRL-S, and return to the top-level file, **PWM_w_Int_v1_0.v**.

Since we've added a new port from slave AXI interface, we need to connect it in the top-level file. Again, this register contains the DutyCycle setting passed down from the processor.

17. Add the port to the slave AXI interface, near line 73. Don't forget to add a comma on the line above the slave reg 0 declaration (see below). **Save** when done.

```
// Instantiation of Axi Bus Interface S00_AXI
PWM_w_Int_v1_0_S00_AXI # (
    .C_S_AXI_DATA_WIDTH(C_S00_AXI_DATA_WIDTH),
    .C_S_AXI_ADDR_WIDTH(C_S00_AXI_ADDR_WIDTH)
) PWM_w_Int_v1_0_S00_AXI_inst (
    .S_AXI_ACLK(s00_axi_aclk),
    .S_AXI_ARESETN(s00_axi_aresetn),
    .S_AXI_AWADDR(s00_axi_awaddr),
    .S_AXI_AWPROT(s00_axi_awprot),
    .S_AXI_AWVALID(s00_axi_awvalid),
    .S_AXI_AWREADY(s00_axi_awready),
    |
    |
    .S_AXI_RDATA(s00_axi_rdata),
    .S_AXI_RRESP(s00_axi_rresp),
    .S_AXI_RVALID(s00_axi_rvalid),
    .S_AXI_RREADY(s00_axi_rready), // ← Add comma here
    .slv_reg0( DutyCycle )
);
```

18. That completes all connections. Select the top-level HDL file, PWM_w_Int_v1_0.v, then click **Run Synthesis** to validate the design.

19. When synthesis completes, the design is good. Do not run implementation. **Cancel, or close**, the *Synthesis Complete* dialog box.

Experiment 3: Package IP Project

This experiment shows how to package the IP for the IP Catalog.

Experiment 3 General Instruction:

Package the IP.

Experiment 3 Step-by-Step Instructions:

1. Click **Package IP** in the Flow Navigator Window.

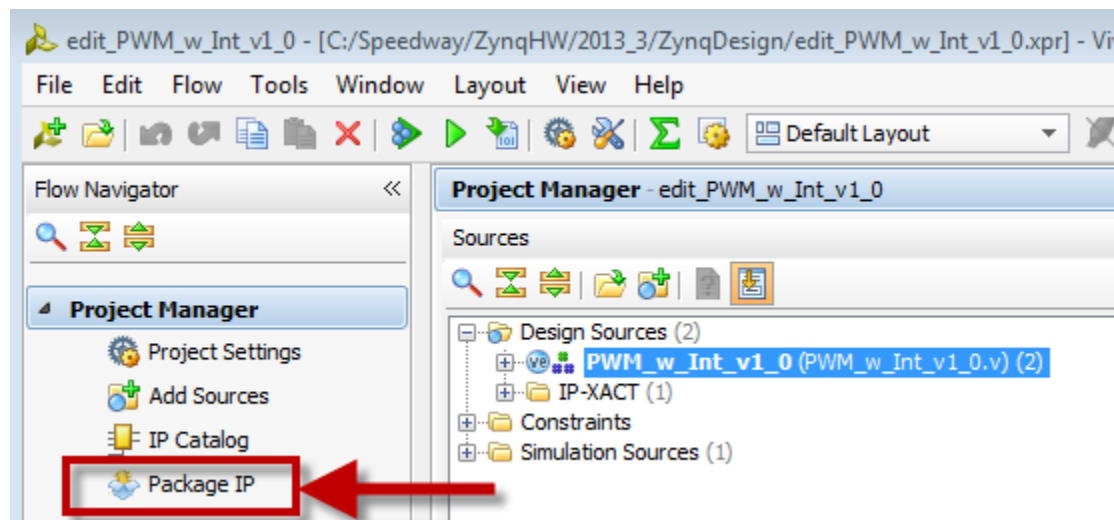
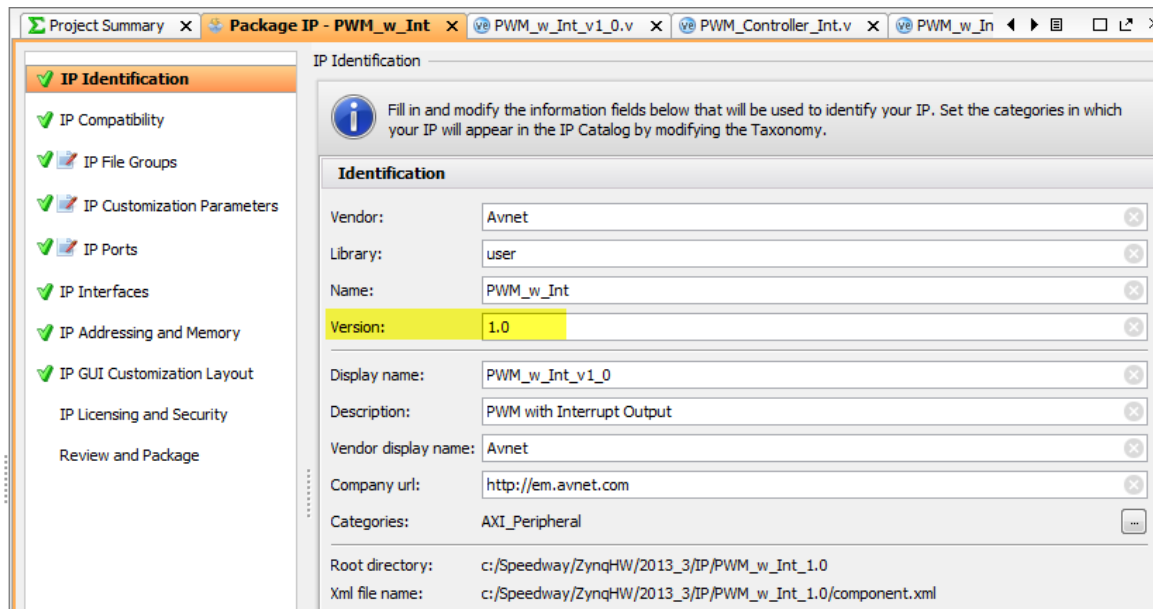


Figure 15 - Package IP

2. This GUI will step-by-step build our IP for the IP Catalog. In the first window, IP Identification, it has many of the parameters we entered when we created this project. **Verify** these parameters. Note, the version of the IP is in this window. If updates or edits are made to this IP, the version number can be updated here.



IP Identification

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: Avnet

Library: user

Name: PWM_w_Int

Version: 1.0

Display name: PWM_w_Int_v1_0

Description: PWM with Interrupt Output

Vendor display name: Avnet

Company url: http://em.avnet.com

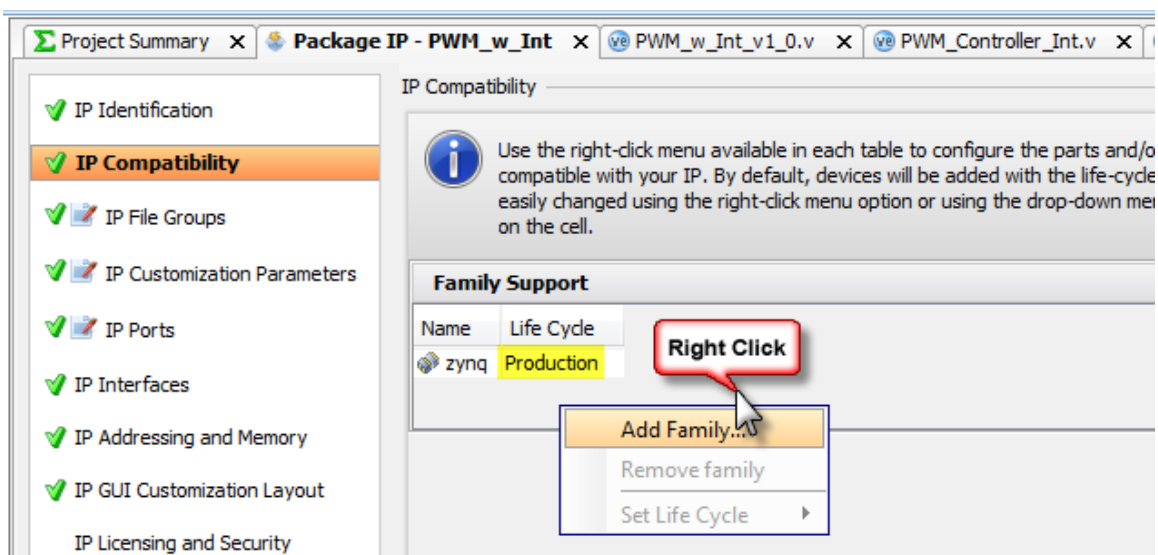
Categories: AXI_Peripheral

Root directory: c:/Speedway/ZynqHW/2013_3/IP/PWM_w_Int_1.0

Xml file name: c:/Speedway/ZynqHW/2013_3/IP/PWM_w_Int_1.0/component.xml

Figure 16 - IP Identification

3. Switch to the next setting, *IP Compatibility*. Here you can set the compatible device families. Zynq is included as that was the target part when we created this IP. More families can be added by right-clicking in the box. Also, change Life Cycle to **Production** as Zynq is in full production now.



IP Compatibility

Use the right-click menu available in each table to configure the parts and/o compatible with your IP. By default, devices will be added with the life-cycle easily changed using the right-click menu option or using the drop-down menu on the cell.

Family Support

Name	Life Cycle
zynq	Production

Right Click

Add Family...

Remove family

Set Life Cycle

Figure 17 - IP Compatibility

- Next is *IP File Groups*, notice at the top of this window a notification is highlighted in green. This indicates that Vivado has detected our changes we made in the HDL. Click **Merge changes from IP File Groups Wizard**.

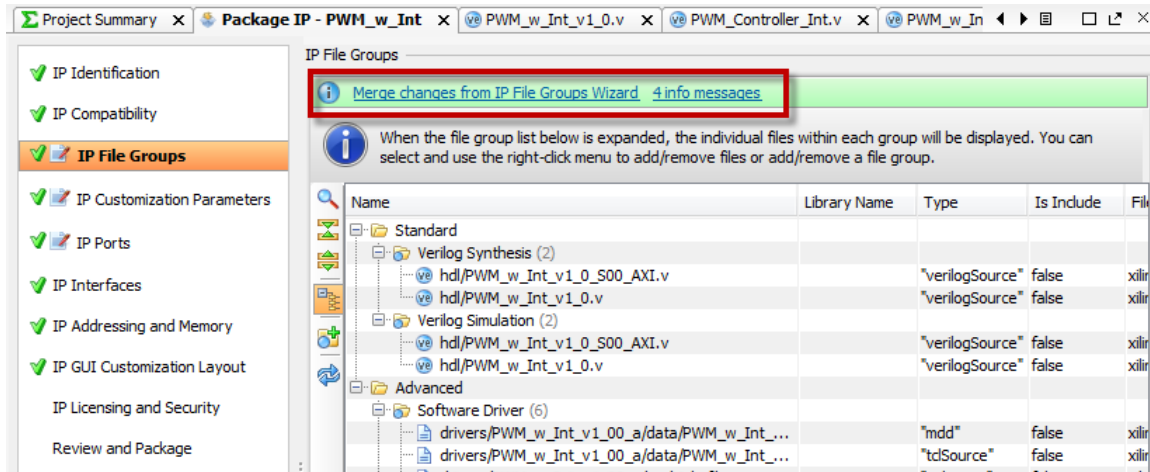


Figure 18 - IP File Groups

- Once the changes have merged, expand all files. Notice it's added our PWM_Controller_Int user logic HDL. However it's made it an absolute path. That is not ideal. To mitigate this, **Remove** this file from the *Verilog Synthesis* list as well as the *Verilog Simulation* list.

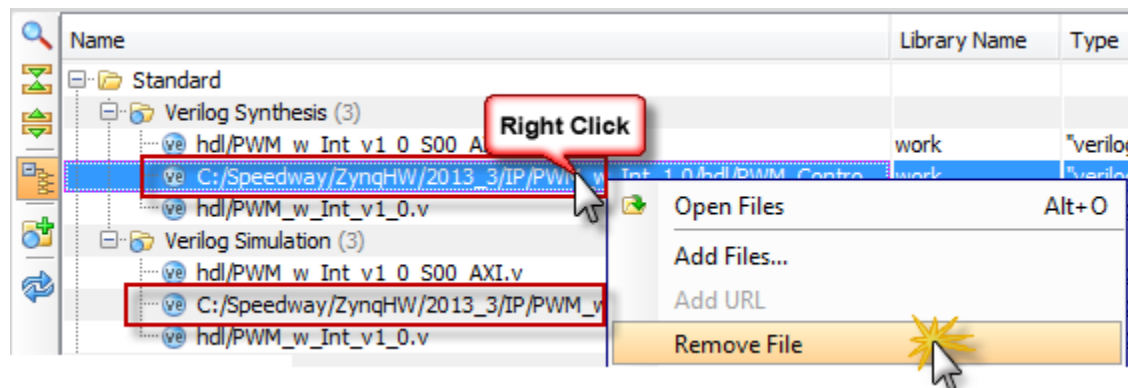


Figure 19 - Remove Files to Eliminate Absolute Paths

6. Now select **Verilog Synthesis**, right-click and select **Add Files**.

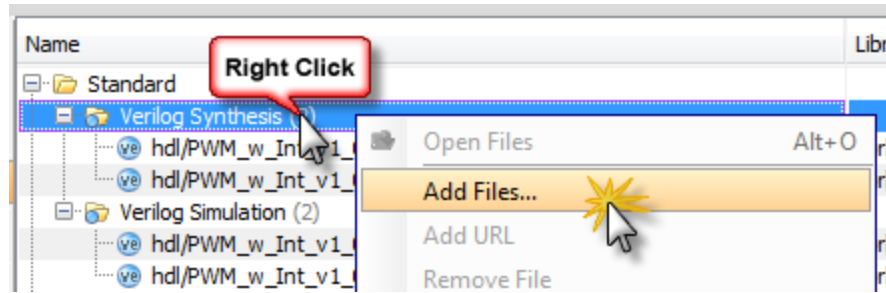


Figure 20 - Add Files

7. When the Add IP Files window opens, click **Add Files...** Vivado should immediately open the HDL directory. If not, navigate to:

C:\Speedway\ZynqHW\2013_3\IP\PWM_w_Int_1.0\hdl

Select **PWM_Controller_Int.v**, verify copy sources into project is unchecked. Click **OK**.

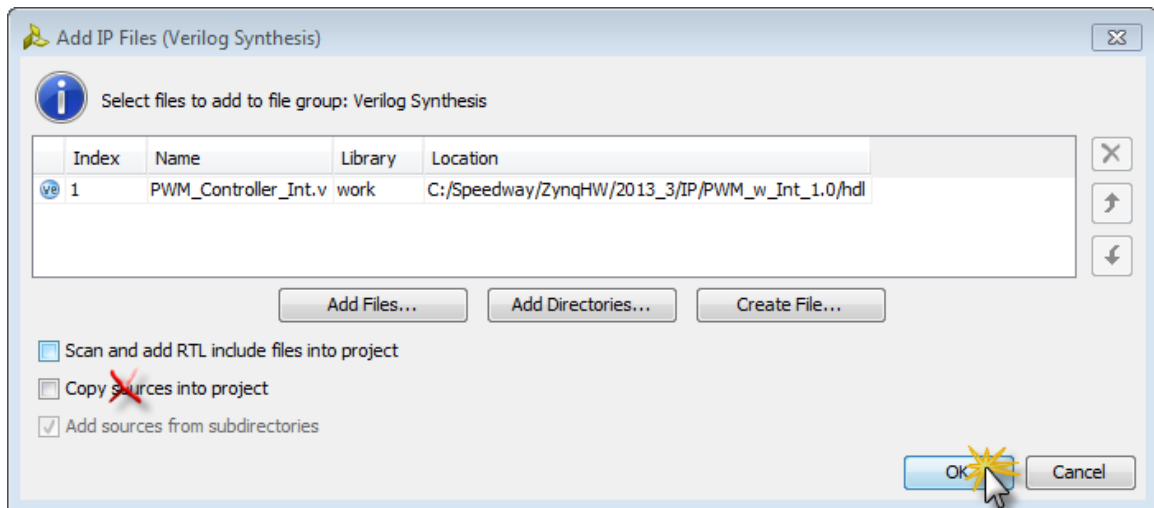


Figure 21 - Add HDL Files to IP File Groups

8. Now the source appears as a relative path. **Repeat** this process for *Verilog Simulation*. Note: Make sure when you add the Simulation files that Vivado is looking for Verilog HDL files.

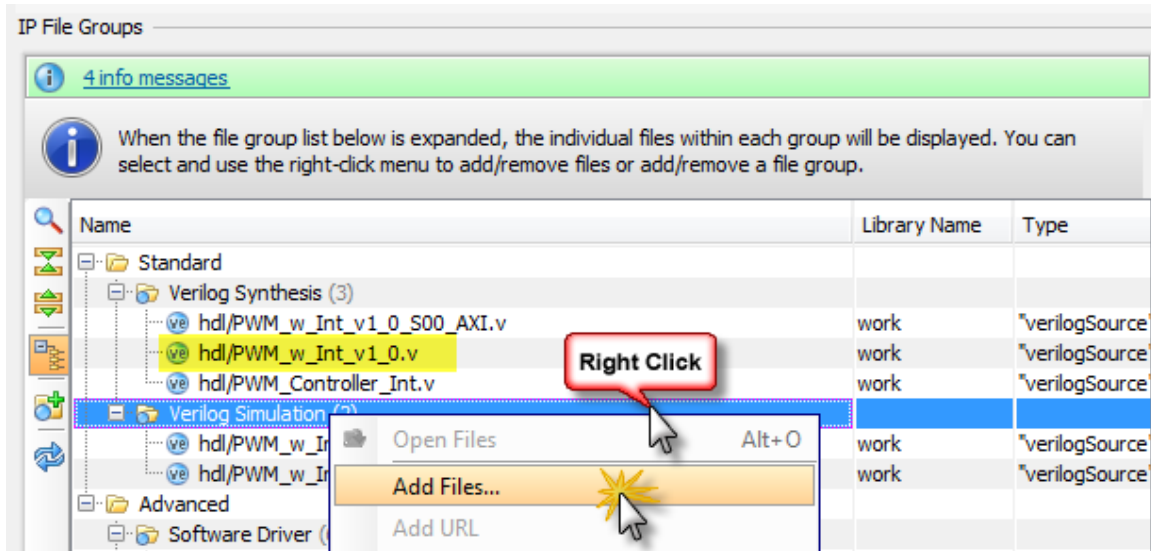


Figure 22 - All Verilog Synthesis files are relative now. Repeat for Simulation

Both files are relative now:

Name	Library Name	Type
Standard		
Verilog Synthesis (3)		
hdl/PWM_w_Int_v1_0_S00_AXI.v	work	"verilogSource"
hdl/PWM_w_Int_v1_0.v	work	"verilogSource"
hdl/PWM_Controller_Int.v	work	"verilogSource"
Verilog Simulation (3)		
hdl/PWM_w_Int_v1_0_S00_AXI.v	work	"verilogSource"
hdl/PWM_w_Int_v1_0.v	work	"verilogSource"
hdl/PWM_Controller_Int.v	work	"verilogSource"

Figure 23 - All files are relative now

- Switch to the *IP Customization Parameters* page. Again, changes are detected and must be merged. Click **Merge Changes**. Next, click on **Parameter Import Dialog**.

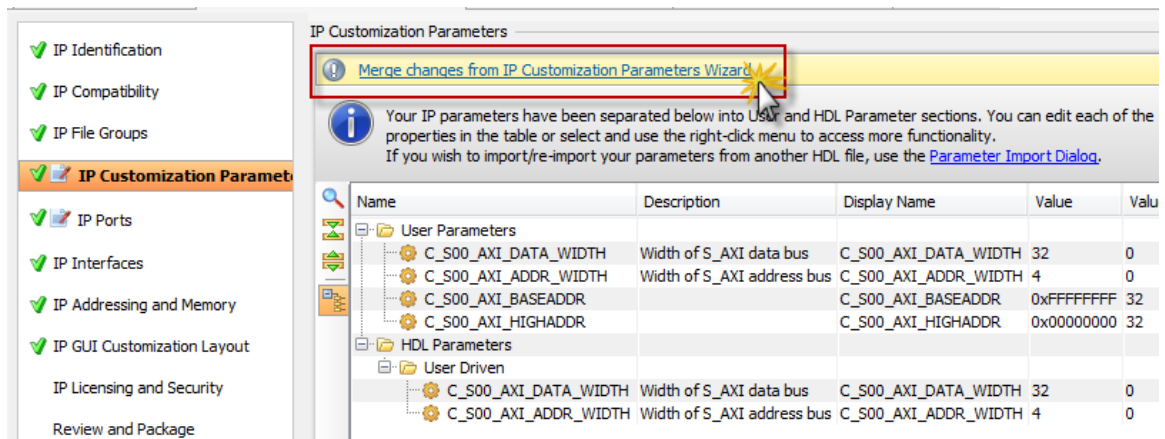


Figure 24 - IP Customization Parameters

- IMPORTANT:** A dialog box opens asking for the top-level HDL file, Choose **PWM_w_Int_v1_0.v** in the HDL directory:

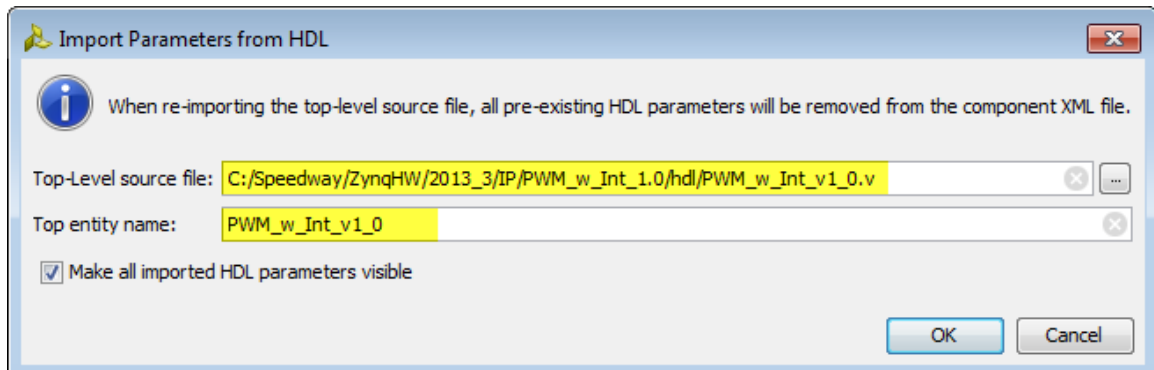


Figure 25 - Choose Top-level Source File

Now our new *PWM_PERIOD* parameter will appear in the customization window when users double-click the IP in their block designs. Again, this will allow them to change the counter depth of the PWM.

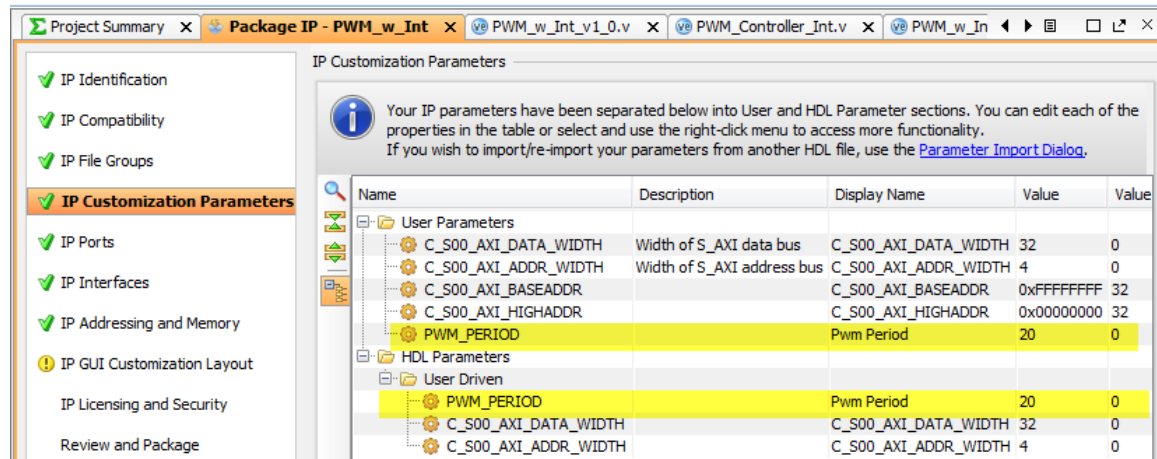


Figure 26 - User Parameters Added

- Next, select the *IP Ports* tab and repeat the above process. Select *Port Import Dialog* and select the **top-level HDL file**, same as step 10, also **important** to choose the right file, **PWM_w_Int_v1_0.v**.

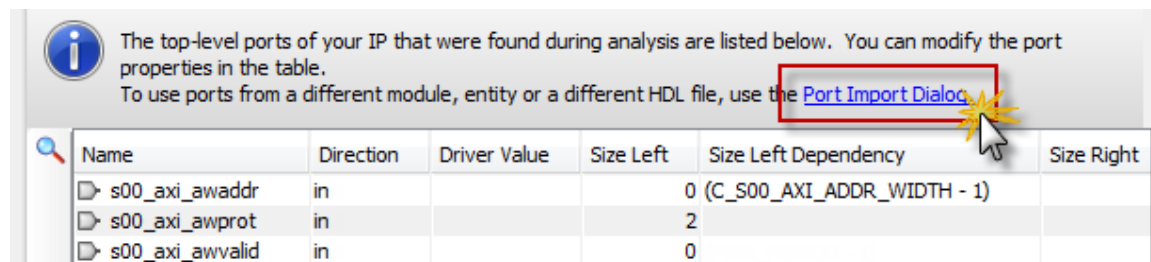


Figure 27 - Import Ports

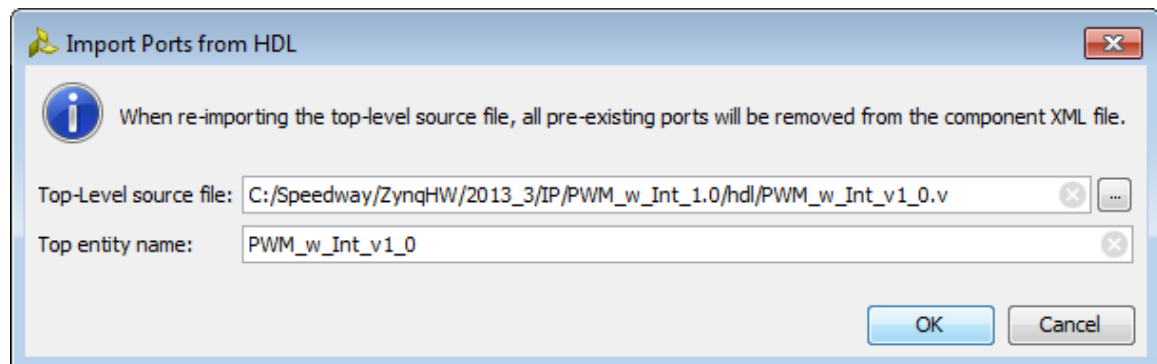


Figure 28 - Select Top-level HDL

When done, all of our user ports should be on the port list as shown below:

IP Ports

The top-level ports of your IP that were found during analysis are listed below. You can modify the port properties in the table.
To use ports from a different module, entity or a different HDL file, use the [Port Import Dialog](#).

Name	Direction	Driver Value	Size Left	Size Left Dependency	Size Right
Interrupt_out	out		0		
LEDs	out		7		
PWM_Counter	out		19 (PWM_PERIOD - 1)		
DutyCycle	out		31		
s00_axi_ack	in		0		
s00_axi_aresetn	in		0		
s00_axi_awaddr	in		3 (C_S00_AXI_ADDR_WIDTH - 1)		
s00_axi_awprot	in		2		
s00_axi_awvalid	in		0		
s00_axi_awready	out		0		

Figure 29 - Add User Ports

12. The next page, *IP Interfaces*, does not require any changes. Our interface is an AXI Slave.

IP Interfaces

Use the right-click menu in the table to create and modify interfaces option will help you map the IP physical ports to interface logical por

Name	Abstraction Type Vlnv	Interface Mod
S00_AXI	xilinx.com:interface:aximm_rtl:1.0	slave
S00_AXI_RST	xilinx.com:signal:reset_rtl:1.0	slave
S00_AXI_CLK	xilinx.com:signal:clock_rtl:1.0	slave

Figure 30 - IP Interface

13. Move to **IP Addressing and Memory**. Run the wizard hyperlinked at the top of the screen:

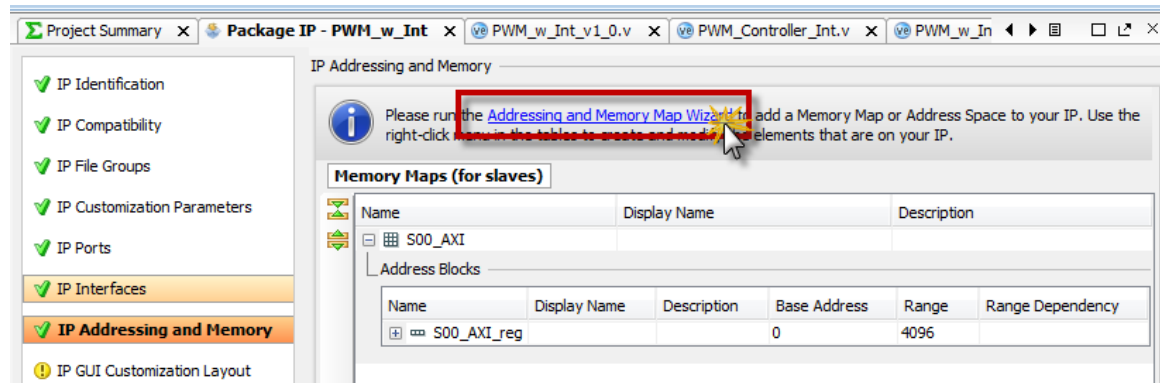


Figure 31 - Run Addressing and Memory Map Wizard

14. Click **Next >** to run the Wizard.

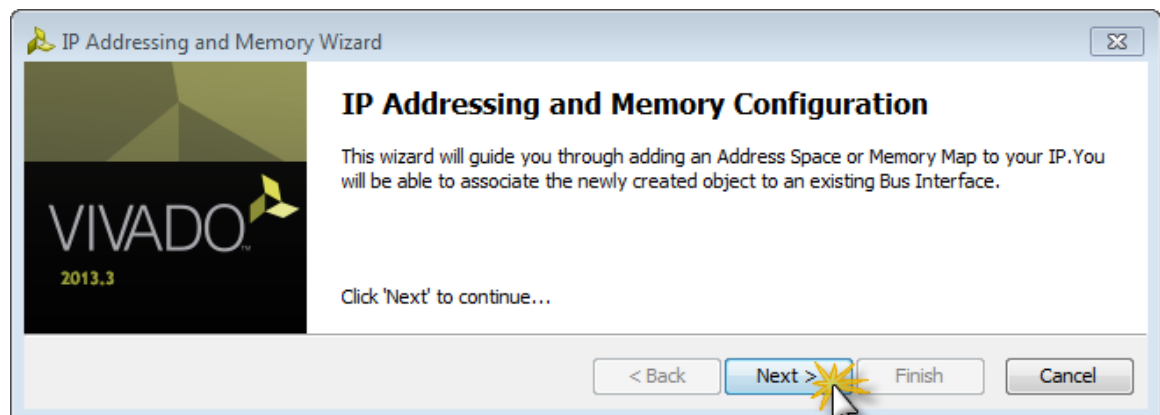


Figure 32 - Run Addressing and Memory Configuration Wizard

15. There is no memory or BRAM in this IP so the screen is blank, click **Cancel**.

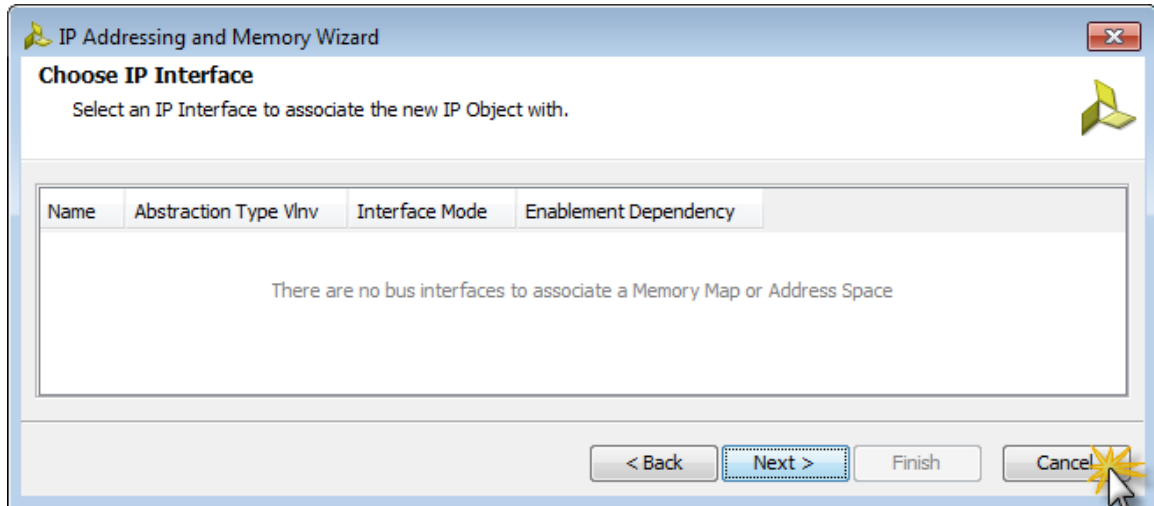


Figure 33 - Memory Detection, No Memory in IP

The purpose of this was to show you, that if your IP had internal memory, it would be detected here.

16. Next move to the *IP GUI Customization Layout*. Run the Wizard.

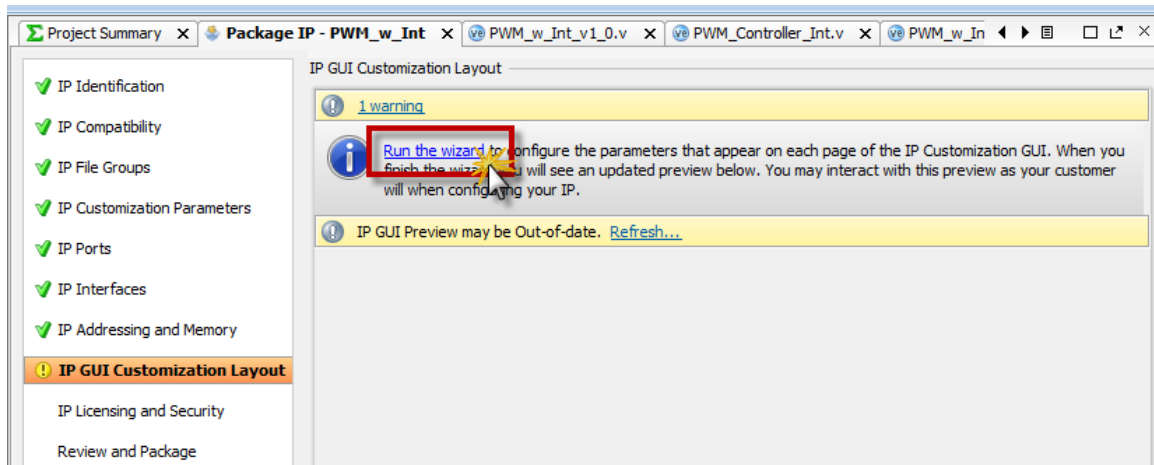


Figure 34 - IP GUI Customization Layout Wizard

17. As we only have one parameter select **Jump straight to the finish page**. Click **Next >**.



Figure 35 - IP GUI Customization Wizard

18. Click **Finish** to Regenerate IP GUI Files.

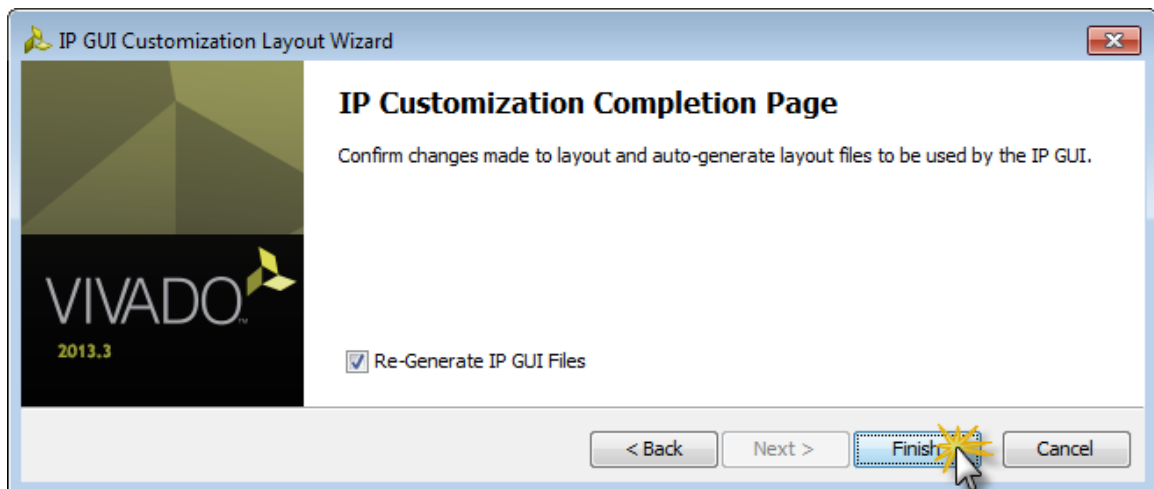


Figure 36 - Finish GUI Customization

19. Click **Refresh** to update the GUI Preview.

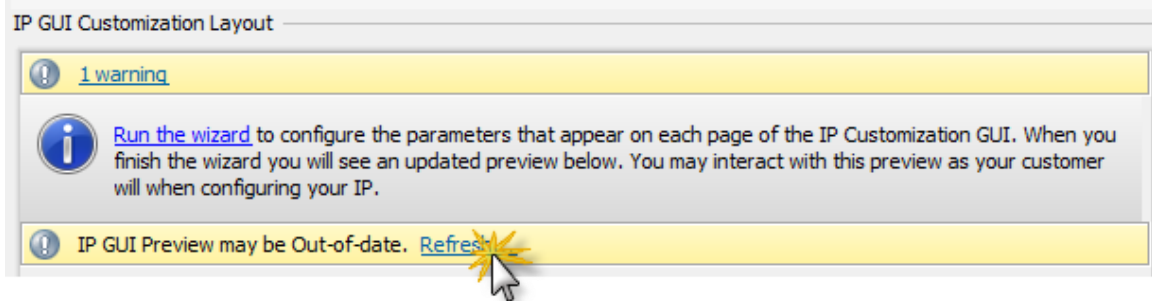


Figure 37 - GUI Preview needs to be refreshed

Ignore the warning. This window now displays what your IP will look like when added from the IP catalog.

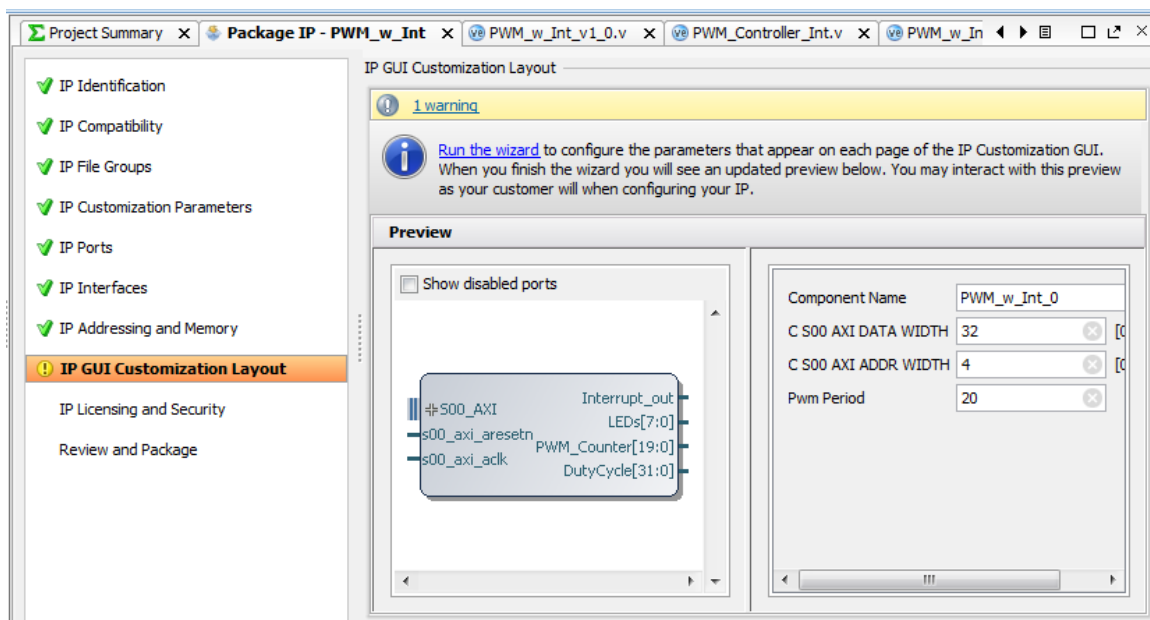


Figure 38 - IP GUI Preview

20. We do not have any licensing features of our IP so IP Licensing and Security can be skipped over. Select **Review and Package** to complete the IP Packaging.

21. Review the settings and click **Re-Package IP**.

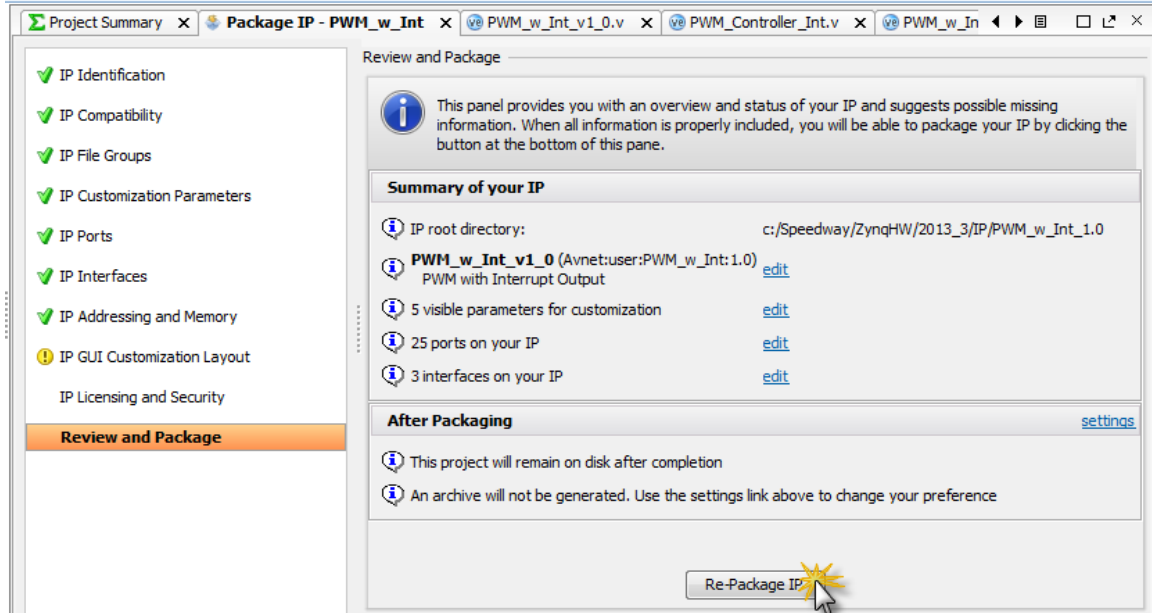


Figure 39 - Re-Package IP

22. When done, Vivado may close. That is OK. If not, you can close this Vivado session.

23. Using Windows Explorer, navigate to our IP repository. Notice our IP project is now in the repository:

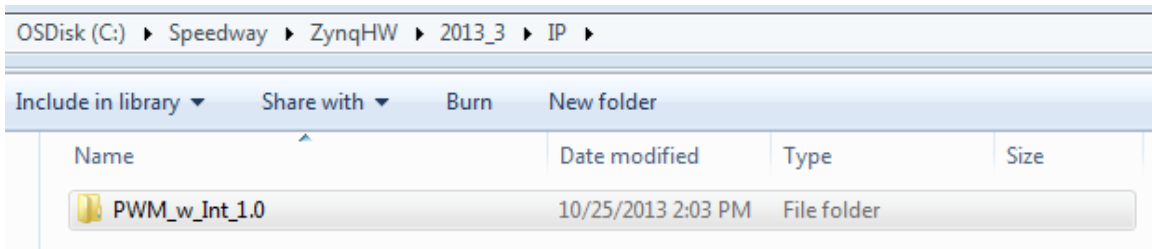
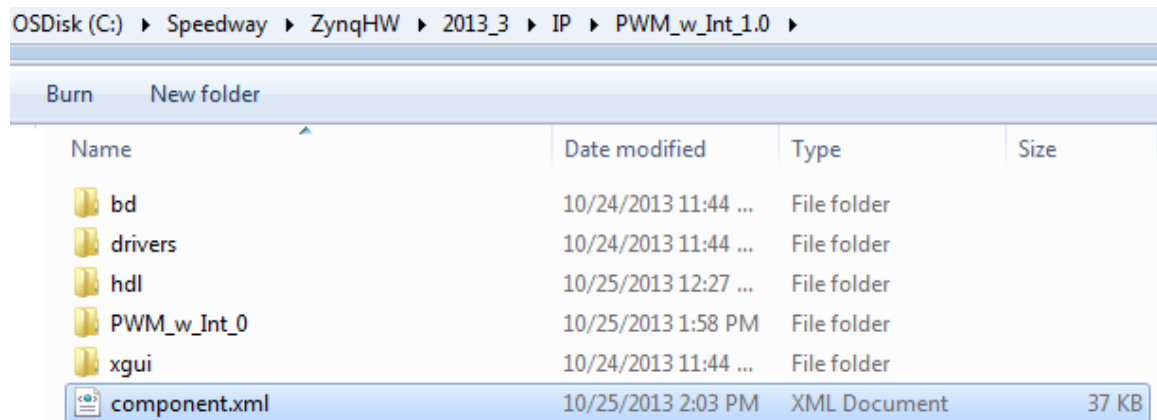


Figure 40 - IP Repository

24. Open the **PWM_w_Int_1.0** folder. Inside are directories that include the HDL files as well as for drivers. Lastly, the **component.xml** file is our package IP that is read in by Vivado.



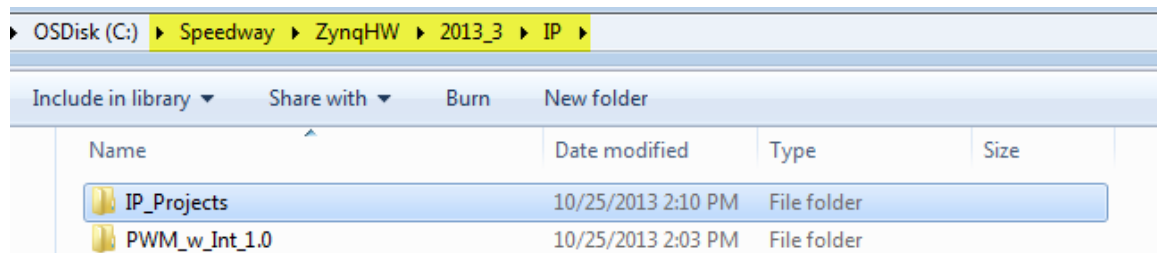
OSDisk (C:) > Speedway > ZynqHW > 2013_3 > IP > PWM_w_Int_1.0 >

Name	Date modified	Type	Size
bd	10/24/2013 11:44 ...	File folder	
drivers	10/24/2013 11:44 ...	File folder	
hdl	10/25/2013 12:27 ...	File folder	
PWM_w_Int_0	10/25/2013 1:58 PM	File folder	
xgui	10/24/2013 11:44 ...	File folder	
component.xml	10/25/2013 2:03 PM	XML Document	37 KB

Figure 41 - IP File Structure and Directories

One piece of housekeeping is recommended. One thing you may have noticed is our edit_IP Vivado Project is not in these directories. That is because we created this IP project from our main Vivado project. Thus it was created in the same directory as our main project. Thus it's wise to move it from there and into an IP_Projects directory in our IP Repository.

25. In the IP Repository, create a folder called **IP_Projects**:



OSDisk (C:) > Speedway > ZynqHW > 2013_3 > IP >

Name	Date modified	Type	Size
IP_Projects	10/25/2013 2:10 PM	File folder	
PWM_w_Int_1.0	10/25/2013 2:03 PM	File folder	

Figure 42 - Create IP Projects Folder

26. Navigate to the **ZynqDesign** Project directory and move (cut) all **edit_PWM*** files and folders to the IP Projects directory.

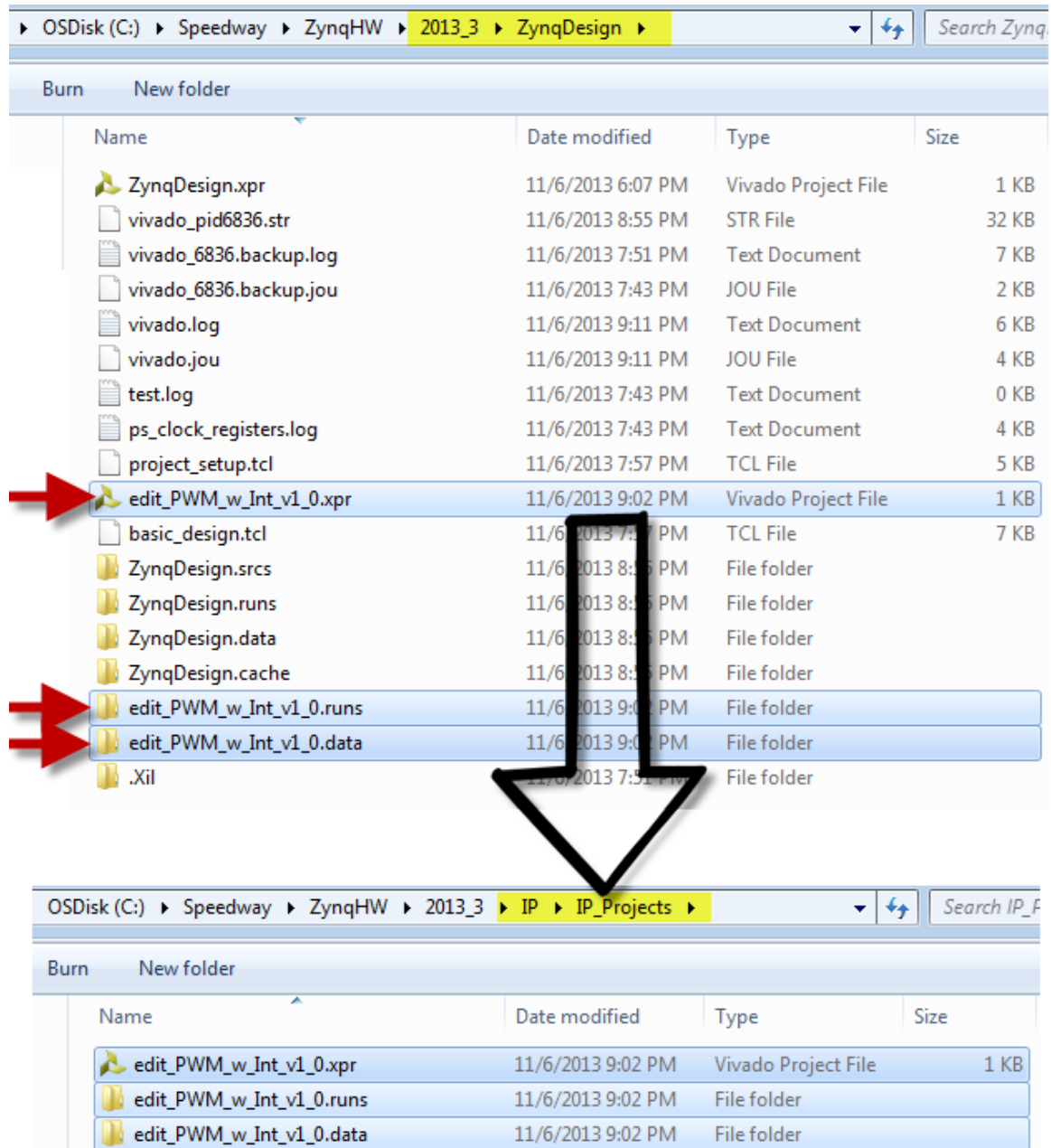


Figure 43 - Move IP Project to IP_Projects folder

That's it, that completes our creating our IP. Next, we'll add it to our design!

Experiment 4: Add IP to Project

This experiment shows how to add the new user IP Repository to our Vivado project. Once added, we'll drop it into our block design and connect it to our AXI Interconnect Block. Additionally, we'll hook up the IP outputs to an Integrated Logic Analyzer core to display the outputs.

Experiment 4 General Instruction:

Import New IP into project and connect it to the design.

Experiment 4 Step-by-Step Instructions:

1. Return to our ZynqDesign Vivado Project.
2. Click on **Project Settings** under Flow Navigator.

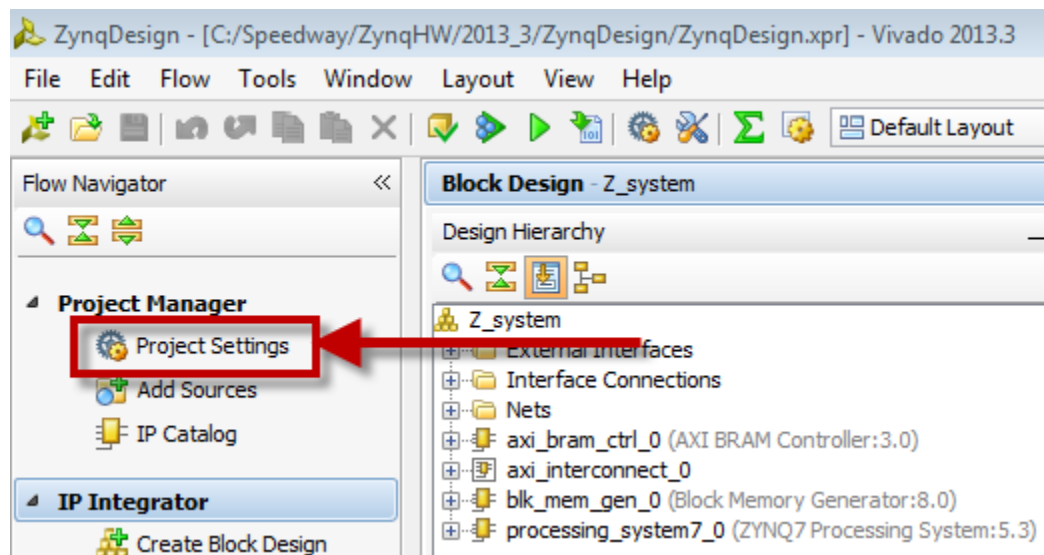


Figure 44 - Open Project Settings

3. Click on **IP**. Since we created this IP from this project it's been automatically added to the project. However, it has the IP repository set to our specific IP. Instead, **Remove** this and reset it to our base IP library:

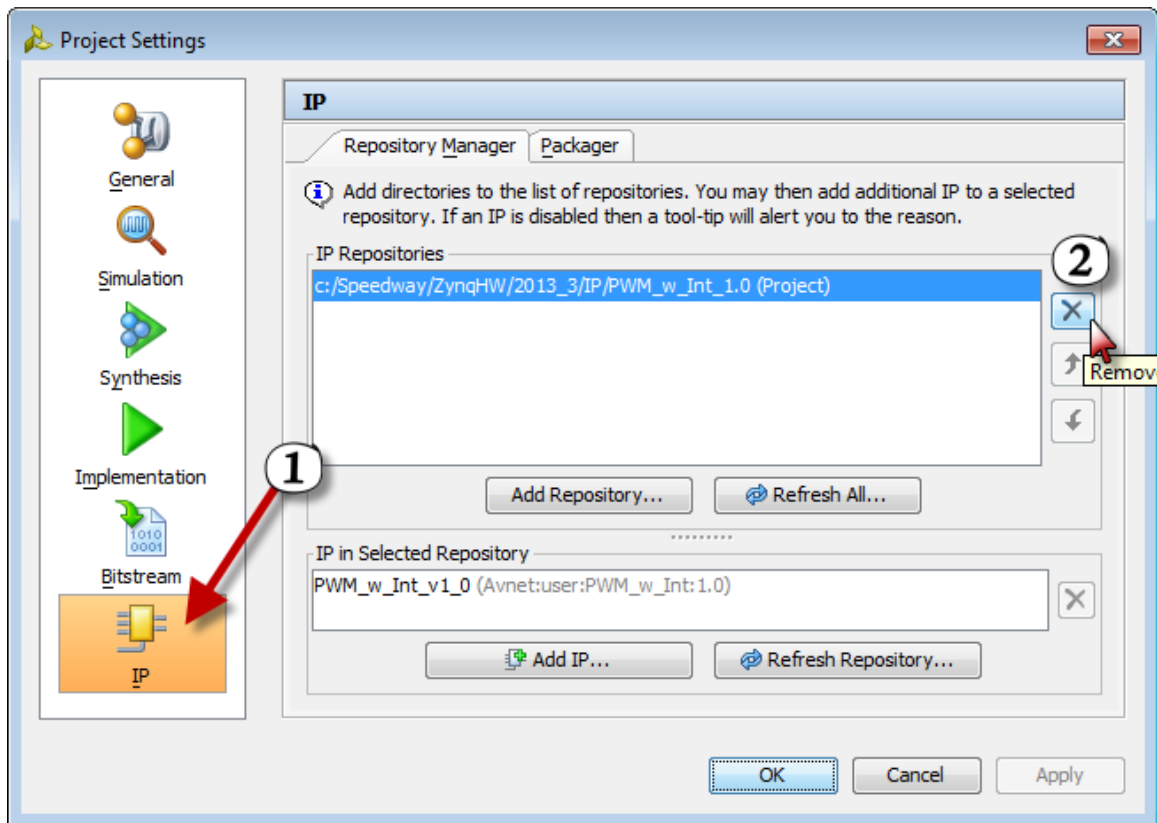


Figure 45 - Remove Current IP Repository

4. Click **Add Repository**, then select our IP directory:
C:\Speedway\ZynqHW\2013_3\IP
5. Select **Refresh All...**, and the following should be displayed. Any time IP is added to the IP Repository, or if the IP has changed, the IP repository here should be refreshed.

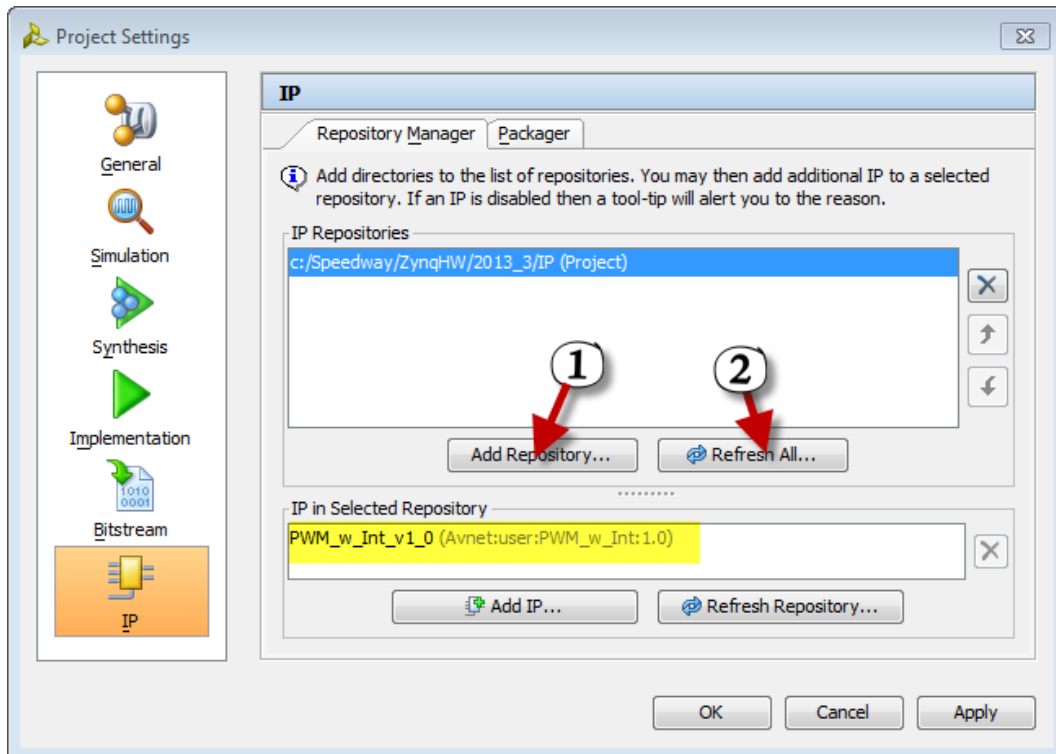


Figure 46 - Refreshed IP Repository

6. Hit **Apply**, and then **OK** to exit Project Settings.
7. Open the block design, **Z_system.bd**.
8. Select **Add IP**.
9. Type **PWM** in the IP search field:

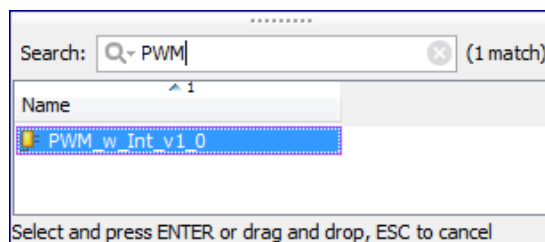


Figure 47 - IP Catalog

10. Double-click **PWM_w_Int_0** to add it to the design.
11. Click the **Regenerate Layout** button from the vertical shortcut bar. This is what should be displayed:

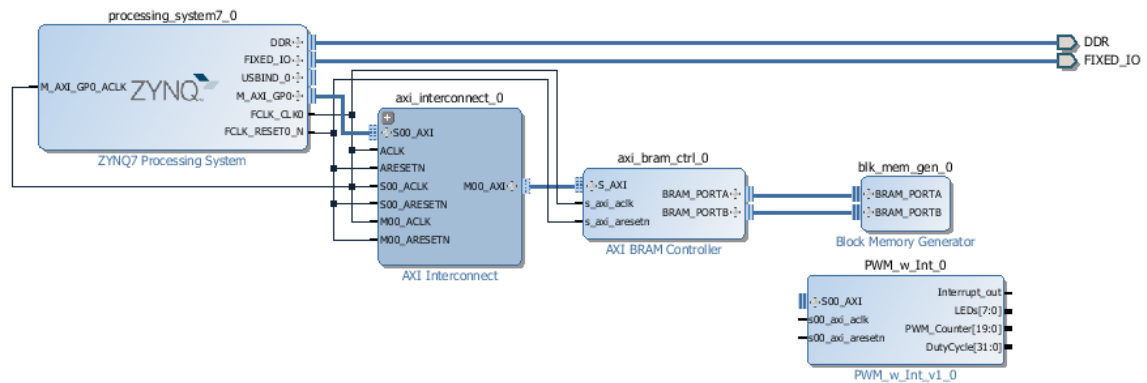


Figure 48 - IP Core Inserted

As we asked in an earlier lab, how would we connect additional IP? The answer, add more AXI Master Interfaces to the AXI Interconnect core.

12. Double-click the **AXI_Interconnect_0** to edit this IP.
13. Change the number of *Master Interfaces* to **2**. Click **OK**.

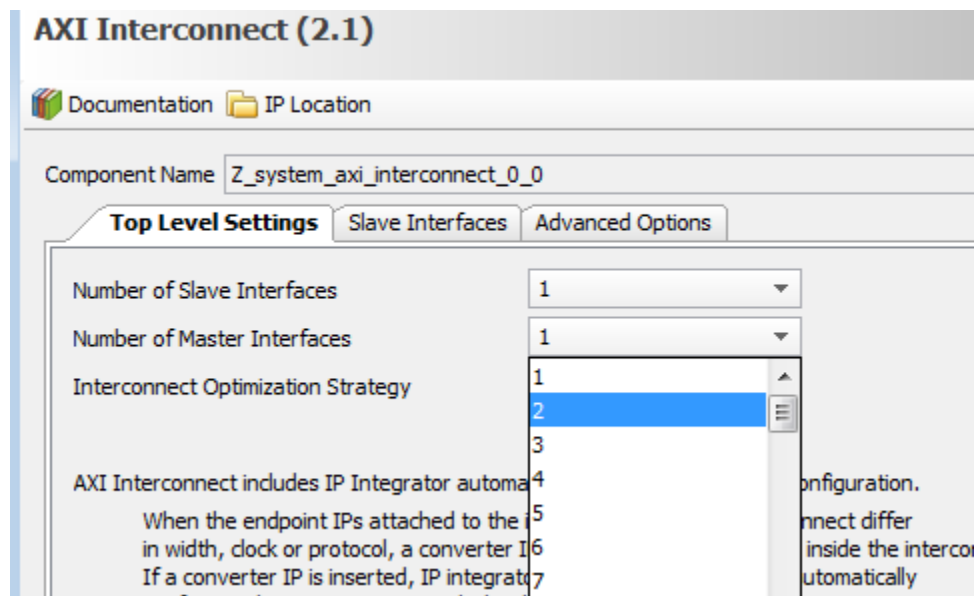


Figure 49 - Customize AXI Interconnect

14. **Connect** this new Master AXI Interface to the slave interface of our IP core. Also connect the clocks and reset to the other clocks and resets. Then hit **Regenerate Layout**.

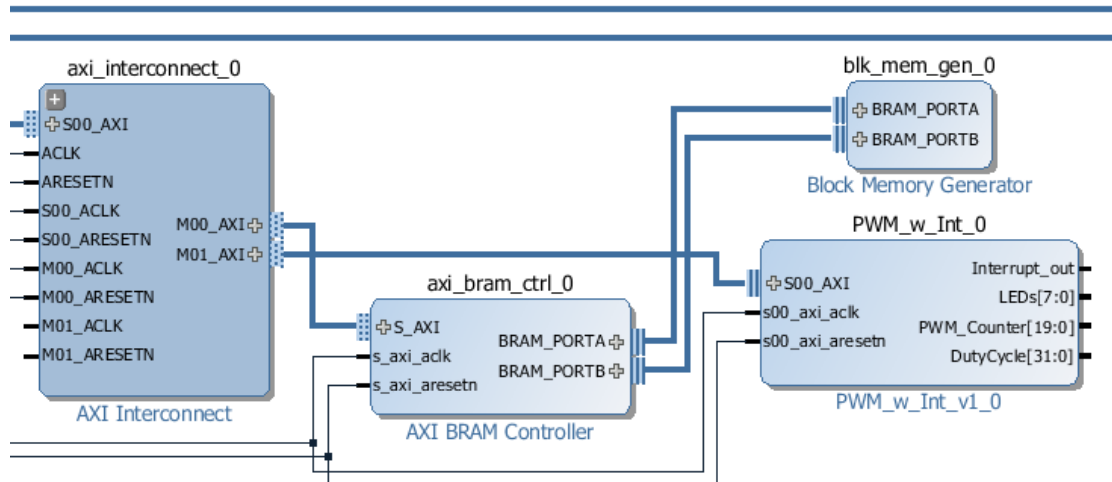


Figure 50 - Connected IP Inputs

15. The new AXI Interface allows a separate clock and reset for the new Master. Connect these to our system clock and reset.

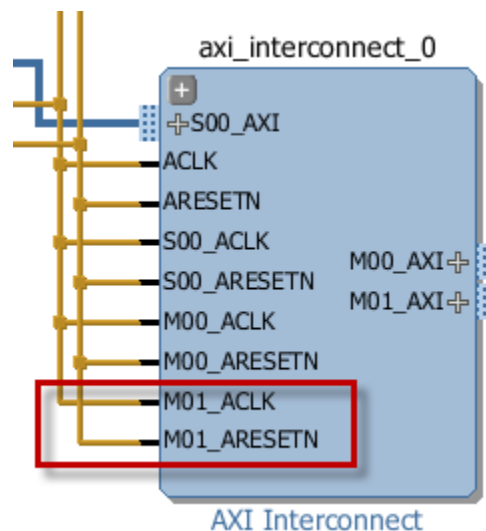


Figure 51 - Connect new clock and reset ports

16. The PWM_w_Int IP has an interrupt output, but the Zynq PS has not been enabled to accept PL interrupts yet. **Double-click** the Zynq PS.

17. Select **Interrupts** from the Page Navigator.
18. Check the box for **Fabric Interrupts** then **expand** it.
19. Expand **PL-PS Interrupt Ports** and check the box next to **IRQ_F2P[15:0]**. F2P is Fabric to PS. When done, click **OK**.

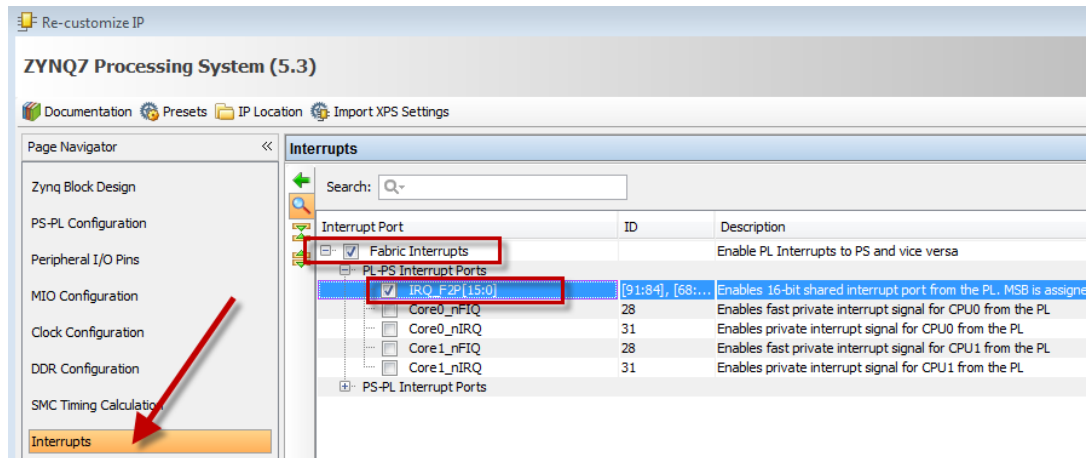


Figure 52 - Add PL-PS Interrupts

20. Connect **Interrupt_Out** from the PWM IP to **IRQ_F2P[0:0]** on the Zynq PS.

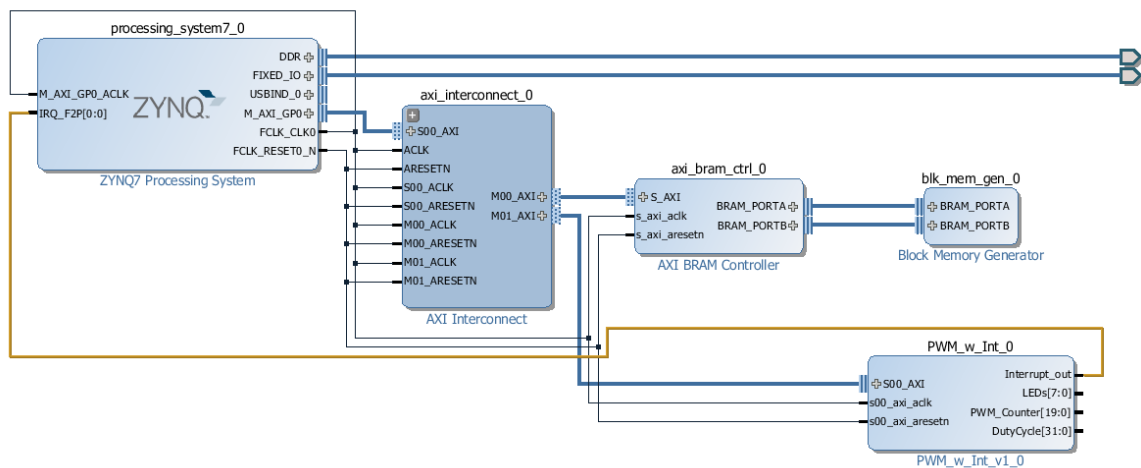


Figure 53 - Connected Interrupt

Finally, we are left with the PWM outputs. MicroZed users do not have access to PL I/O, so instead of using I/O, we'll use an Integrated Logic Analyzer core (ILA).

21. Access the **IP catalog** again, search for **ILA**:

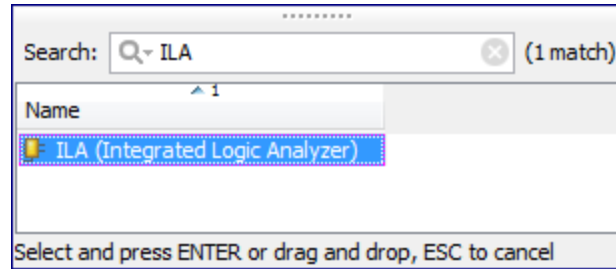


Figure 54 - ILA IP Core

22. **Double-click** this IP to add it to the design.

23. **Double-click** the ILA IP to customize it.

24. Set the number of *Probes* to **4**.

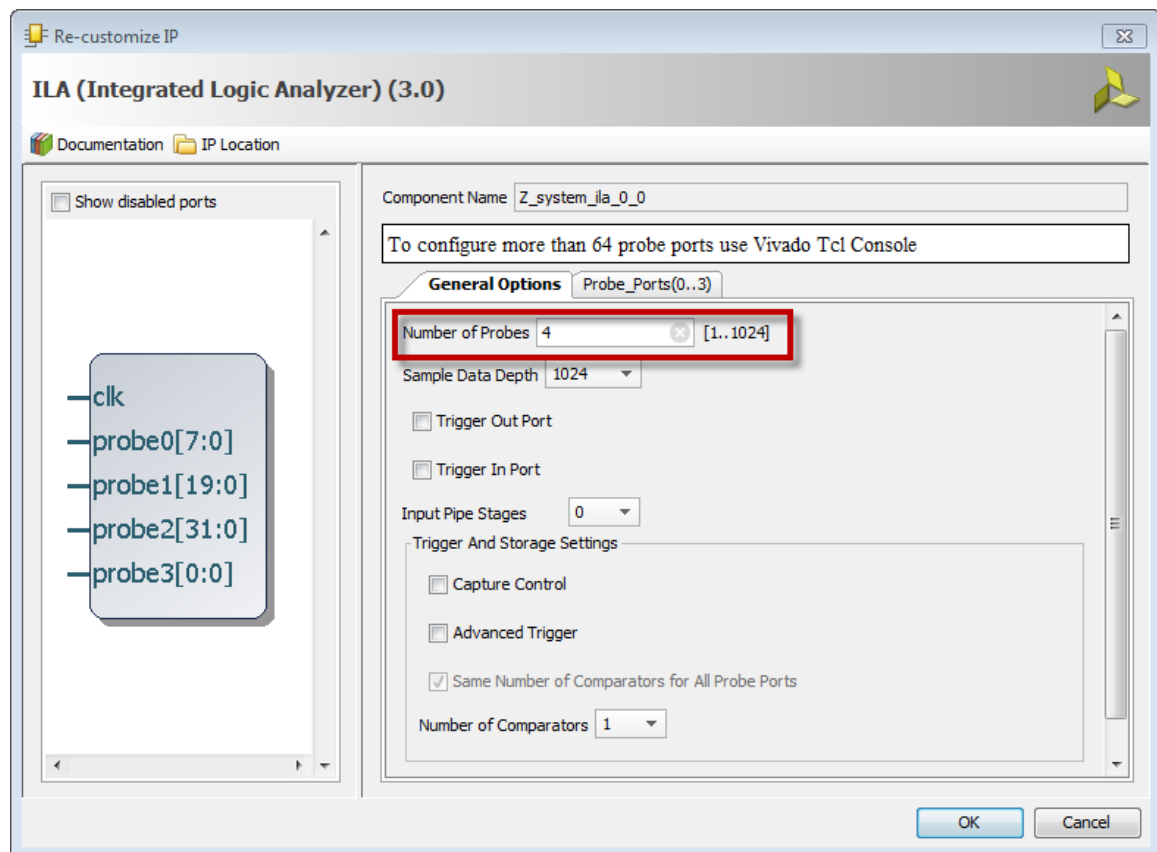


Figure 55 - Customize ILA IP

25. Click the **Probe_Ports** Tab.

26. Change the Probes to match the following:

- a. PROBE0 = 8
- b. PROBE1 = 20
- c. PROBE2 = 32
- d. PROBE3 = 1

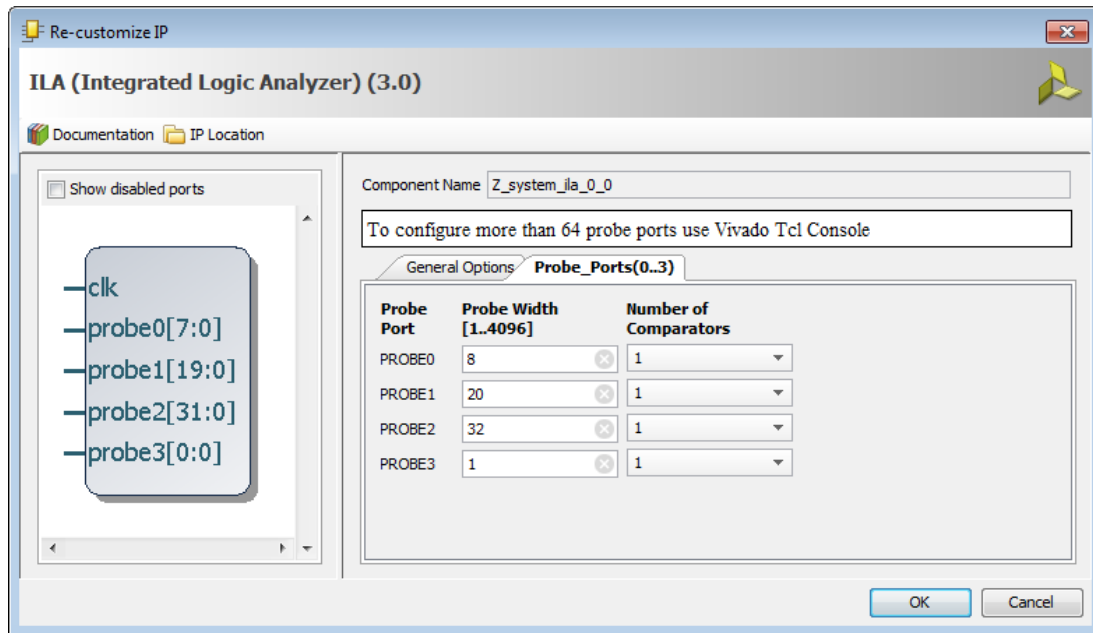


Figure 56 - ILA Probes Configuration

27. Click **OK**.

28. Connect **LEDs** to **probe0**, **PWM_Counter** to **probe1**, **DutyCycle** to **probe2** and **Interrupt_out** to **probe3**. Connect the clk to the other clocks in the design.

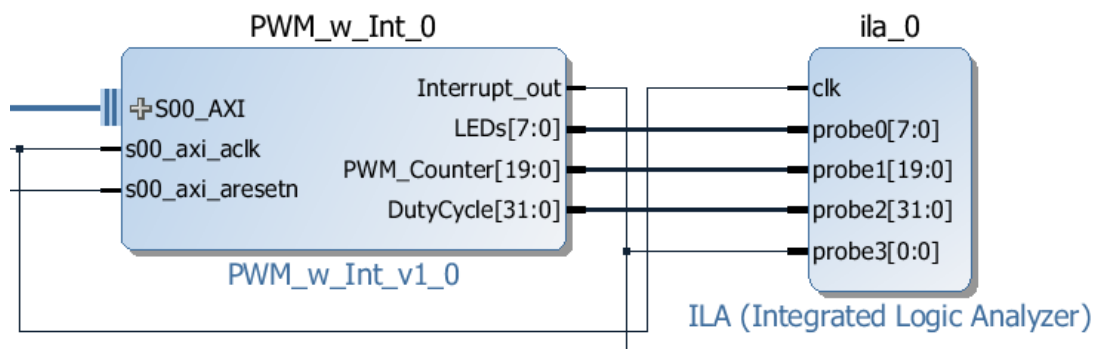


Figure 57 - Connect ILA and PWM

29. Now that we've added new IP, we must update our addressing. Click the **Address Editor** Tab in the block design. Click **Auto Assign Address**.

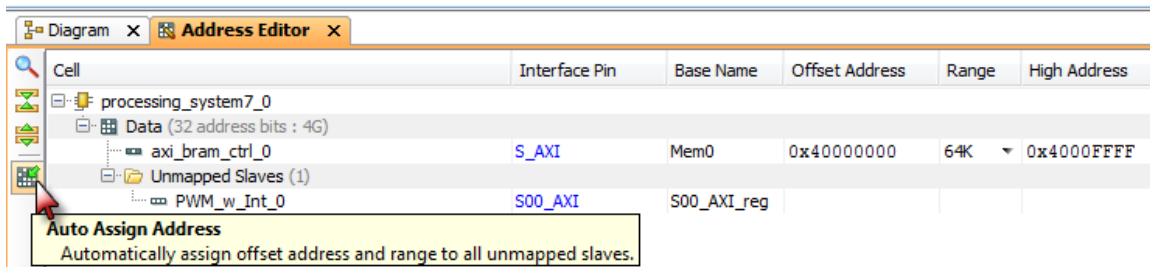


Figure 58 - Assign Address to new IP Core

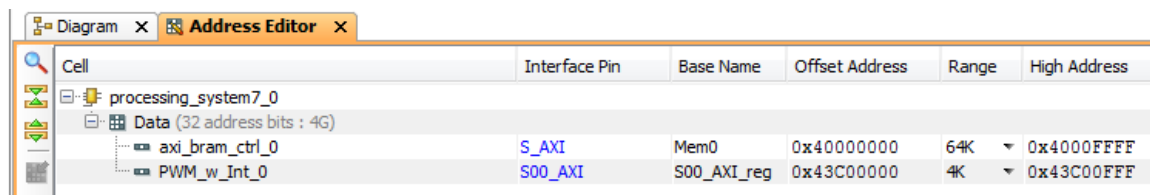


Figure 59 - Assigned IP Addressing

That completes connecting our custom IP to our design. We've created completely new IP, packaged it in the IP Packager, imported it into our design and connected it to the Zynq Processing System. In the next lab we exercise this IP through a debug interface and then add a software application to interact with it.

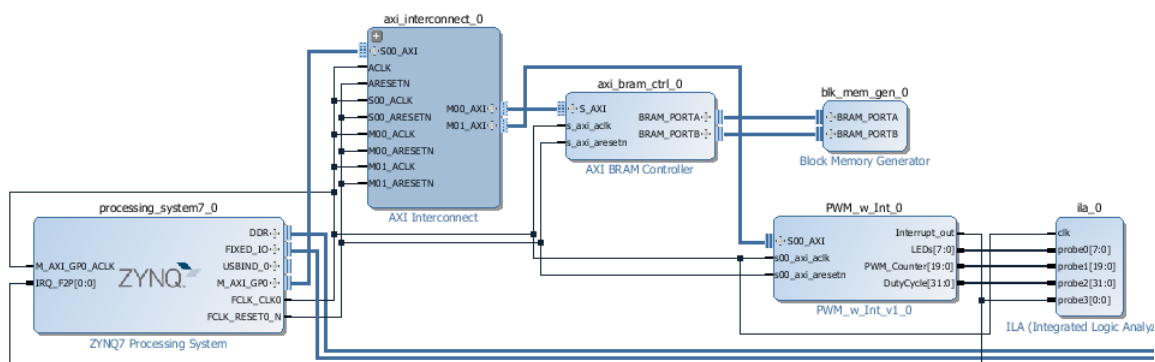


Figure 60 - Fully Connected Design

Experiment 5: Add LogiCORE™ IP JTAG-AXI core

Before we continue, we'll add one more component that we'll use in the next lab. This experiment shows how to add a JTAG-AXI core. We'll explain this IP in the next section, but it's easier to add it now as we will generate a bitstream at the end of this lab.

Experiment 5 General Instruction:

Import New IP into project and connect it to the design.

Experiment 5 Step-by-Step Instructions:

1. In the block design, add the **JTAG to AXI Master** Core, search for **JTAG**.

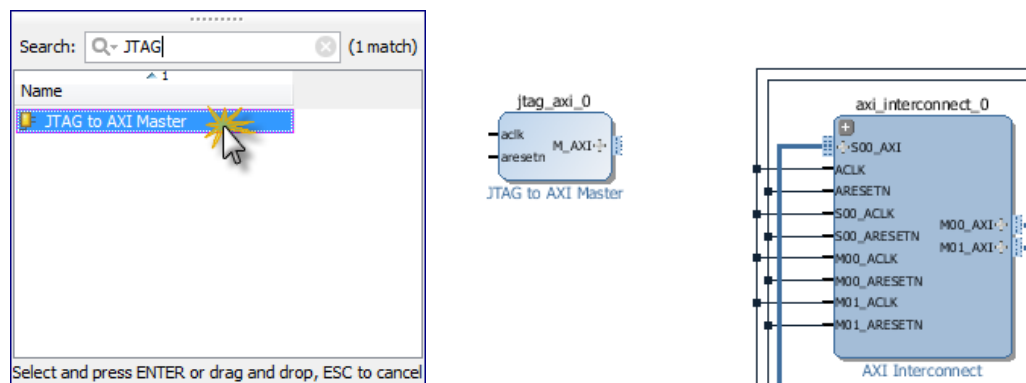


Figure 61 - Add JTAG to AXI Master

The JTAG to AXI Master needs a slave interface to connect into. Note this is a Master, thus it will generate commands and/or data to be sent into a slave. By connecting this directly into the AXI Interconnect, the JTAG to AXI Master will be able to interact with all peripherals connected to that interconnect block.

2. Customize the **AXI Interconnect Block** by double-clicking on it.

3. Add a second slave connection **(2)** to the AXI Interconnect block. Click **OK**.

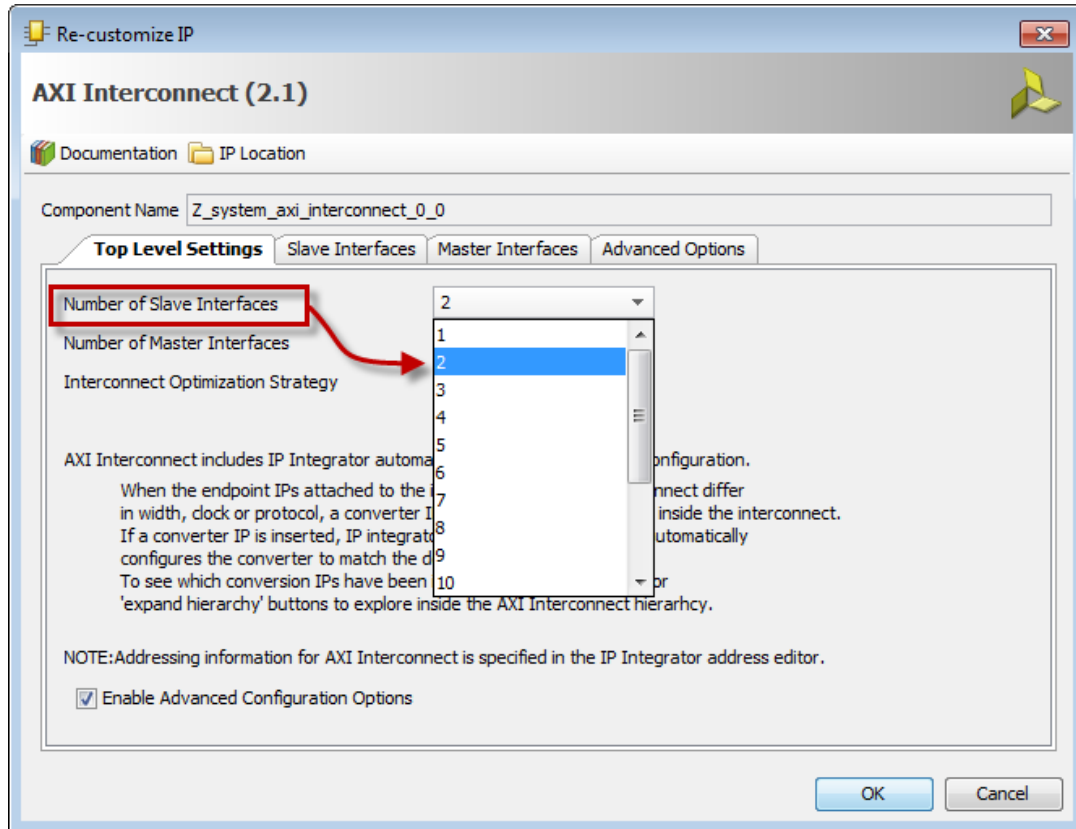


Figure 62 - Add second Slave Interface

4. Connect the JTAG_AXI core to the AXI Interconnect. Connect the clock and reset for both the JTAG_AXI core and the new Slave AXI interface clock and reset for the AXI Interconnect.

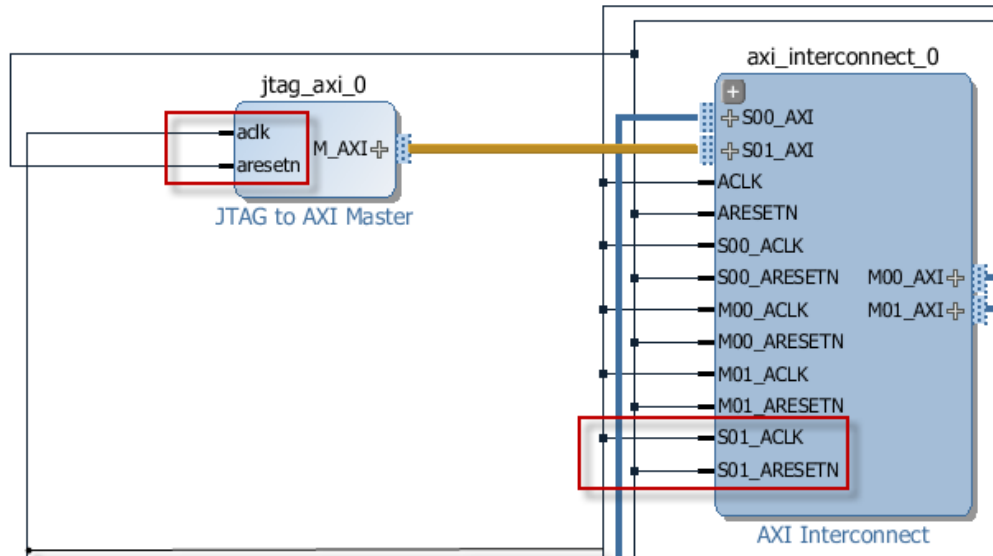


Figure 63 - Connect JTAG_AXI core

The final block design should appear as follows:

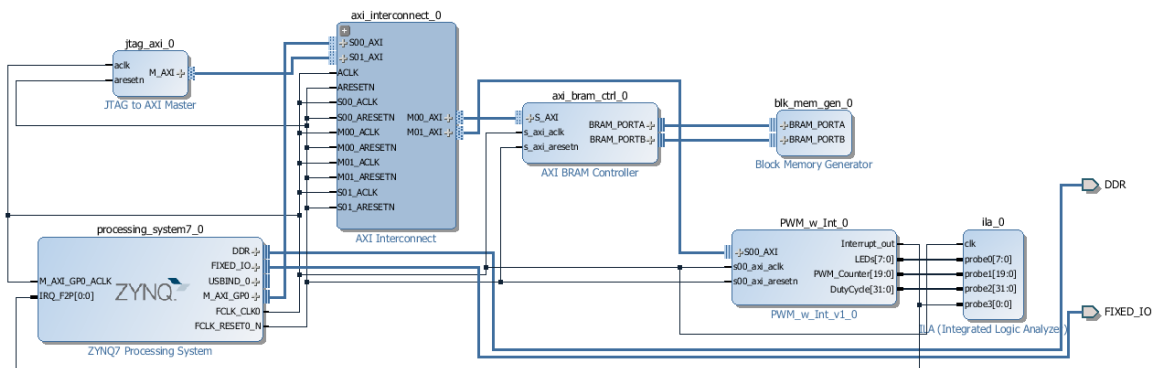


Figure 64 - Final Block Design

5. **Validate** the block design.
6. If OK, **Save** the block design.
7. In the Flow Navigator pane, click **Generate Bitstream**. This will take a few minutes depending on your PC. Let it build, we'll continue with the lecture.

Questions:

Answer the following questions:

- *How many AXI Masters and Slaves can be added to the AXI Interconnect?*

- *How many interrupts can be generated from fabric resources?*

Exploring Further

If you have more time and would like to investigate more...

- ZedBoard users, or MicroZed users who also have an I/O Carrier Card, make the LED's external in your block design and assign them to PL I/O corresponding to the onboard LED's.

This concludes Lab 7.

Revision History

Date	Version	Revision
06 Nov 13	02	Initial Draft
19 Nov 13	03	Pilot Updates

Resources

www.microzed.org

www.zedboard.org

www.xilinx.com/zyng

www.xilinx.com/sdk

www.xilinx.com/vivado

Answers

Experiment 1

- *Where is the IP project located?*

In the same directory as our current project:

C:/Speedway/ZynqHW/2013_3/ZynqDesign

Experiment 5

- *How many AXI Masters and Slaves can be added to the AXI Interconnect?*

16 and 16.

- *How many interrupts can be generated from fabric resources?*

8 per M_AXI_GP = 16

Plus nFIQ & nIRQ per core = 4

20 Total