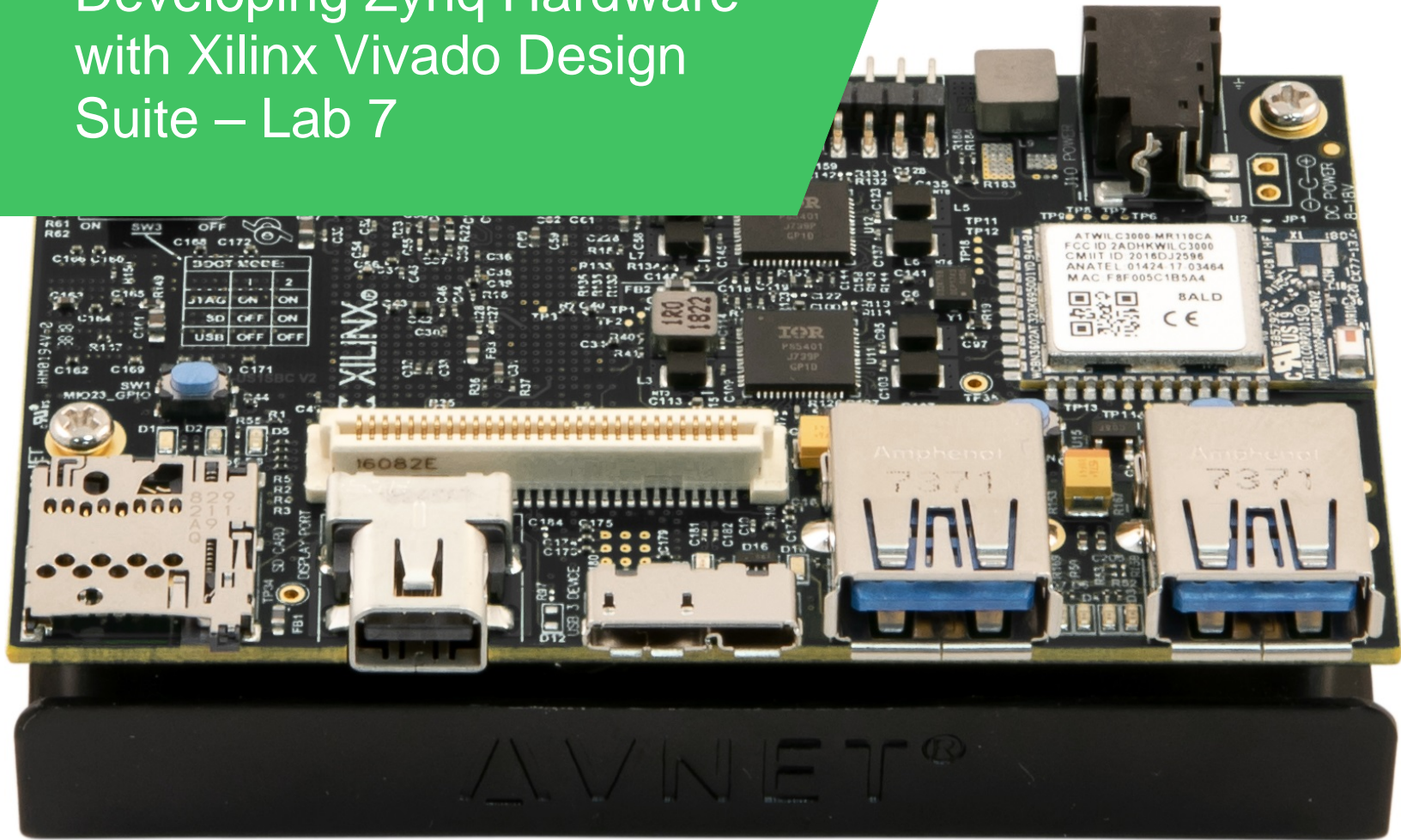


Avnet Technical Training Course

Developing Zynq Hardware with Xilinx Vivado Design Suite – Lab 7



Tools:	2019.1
Training Version:	v14
Date:	July 2019

© 2019 Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

Lab 7 Overview

Xilinx offers a diverse IP catalog which makes it very easy to connect many of the common interfaces to your design. 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

© 2019 Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

Experiment 1: Create A 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 5 or wish to start with a clean copy, delete the `ZynqDesign` and `SDK_Workspace` folders in the `ZynqHW/2019_1` folder. Then unzip **Solutions_Minized/ZynqHW_Lab5_Solution.zip** to the `2019_1` folder. If you have Archive Manager installed, you can do this by right-clicking and selecting Archive Manager then extracting in to the `2019_1` folder.
2. Open **Vivado** with the existing project.
3. Select **Tools → Create and Package IP New IP....**

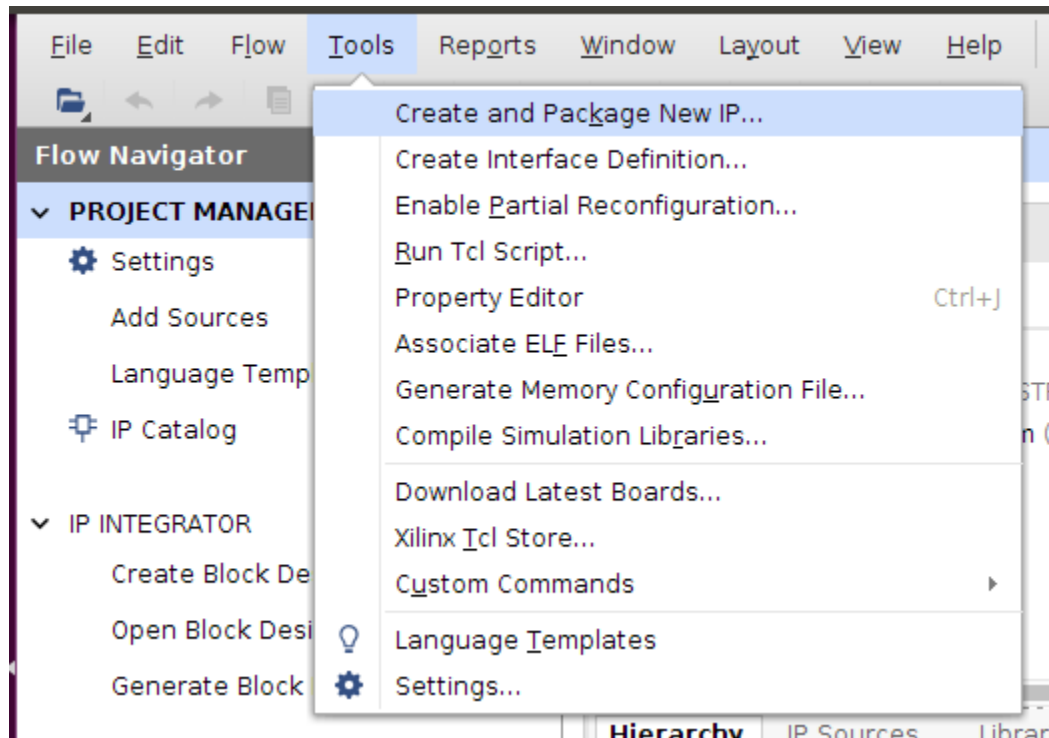


Figure 1 - Create New IP

4. Click **Next >** on Welcome Screen to Create and Package IP.

© 2019 Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

5. We will be **creating a new AXI peripheral**, check the corresponding box and hit **Next >**.

Create and Package New IP

Create Peripheral, Package IP or Package a Block Design
Please select one of the following tasks.

Packaging Options

- ☐ Package your current project
Use the project as the source for creating a new IP Definition.
- ☐ Package a block design from the current project
Choose a block design as the source for creating a new IP Definition.
Select a block design:
- ☐ Package a specified directory
Choose a directory as the source for creating a new IP Definition.

Create AXI4 Peripheral

- ☒ Create a new AXI4 peripheral
Create an AXI4 IP, driver, software test application, IP Integrator AXI4 VIP simulation and debug demonstration design.

Figure 2 - Create a New AXI Peripheral

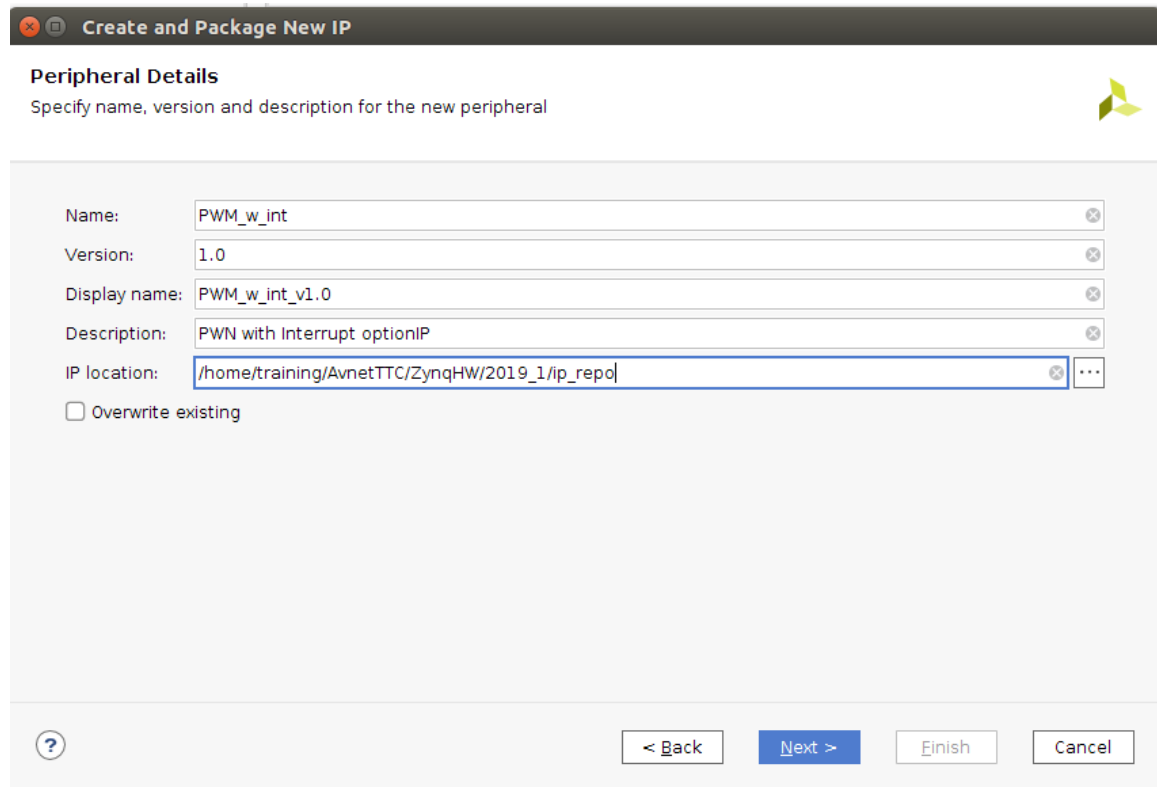
© 2019 Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

*****Very Important*** Please use the same naming conventions outlined in step 6.**

6. 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 >**.

Name:	PWM_w_Int
Version:	1.0
Display name:	PWM_w_Int_v1.0
Description:	PWM with Interrupt option
IP location:	/home/training/AvnetTTC/ZynqHW/2017_4/ip_repo



The screenshot shows a window titled "Create and Package New IP" with a tab labeled "Peripheral Details". Below the tab is the instruction "Specify name, version and description for the new peripheral". The form contains the following fields:

- Name: PWM_w_int
- Version: 1.0
- Display name: PWM_w_int_v1.0
- Description: PWN with Interrupt optionIP
- IP location: /home/training/AvnetTTC/ZynqHW/2019_1/ip_repo

There is an "Overwrite existing" checkbox which is currently unchecked. At the bottom of the window are four buttons: a help button (question mark in a circle), "< Back", "Next >" (highlighted in blue), "Finish", and "Cancel".

Figure 3 - IP Details

© 2019 Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

7. 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 are acceptable. Click **Next >**.

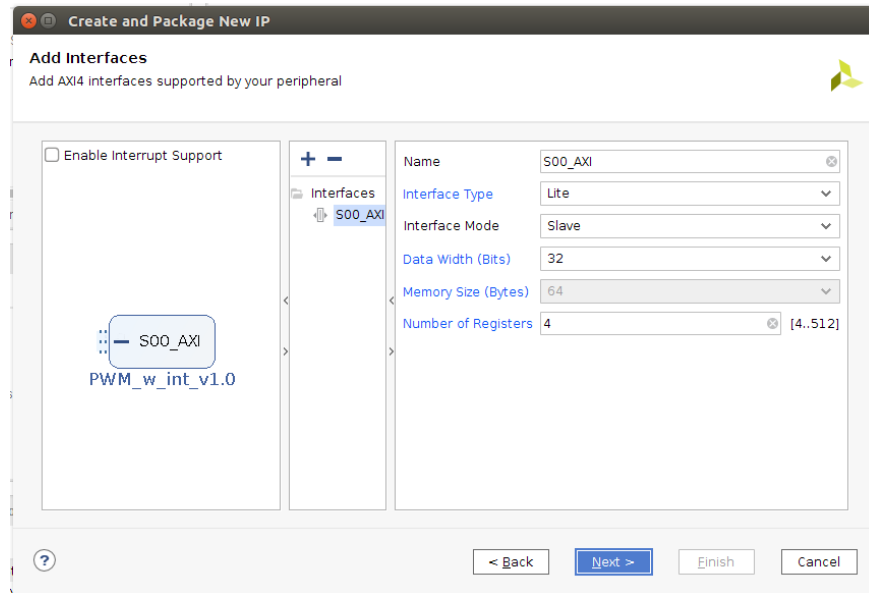


Figure 4 - IP Interface Settings

© 2019 Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

8. Select the **Add IP to the repository** radio button. Verify the *Catalog* path. If OK, click **Finish**.

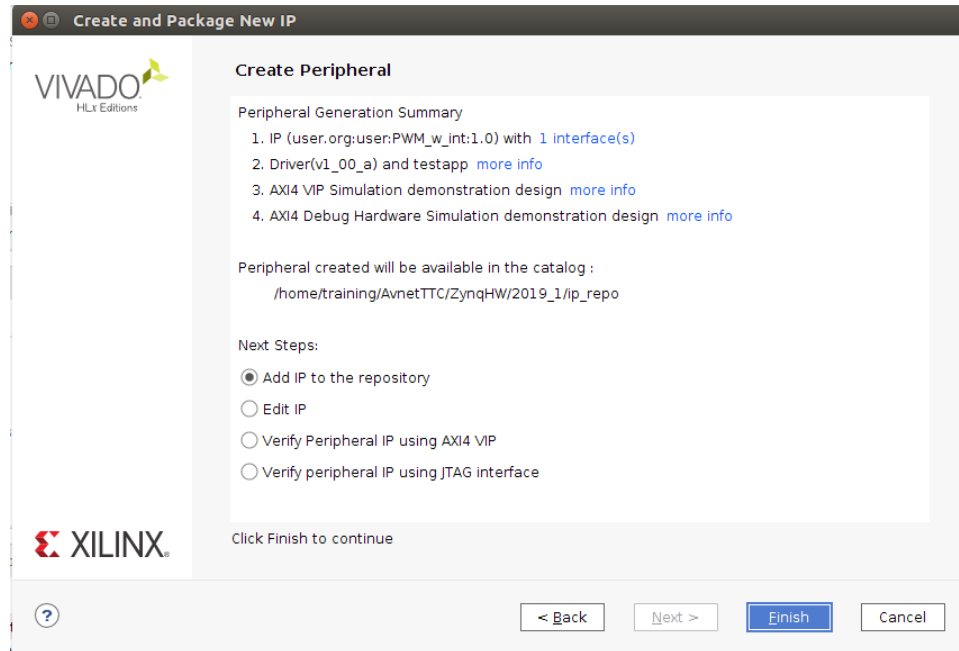


Figure 5 - IP Creation Summary

Questions:

Answer the following questions:

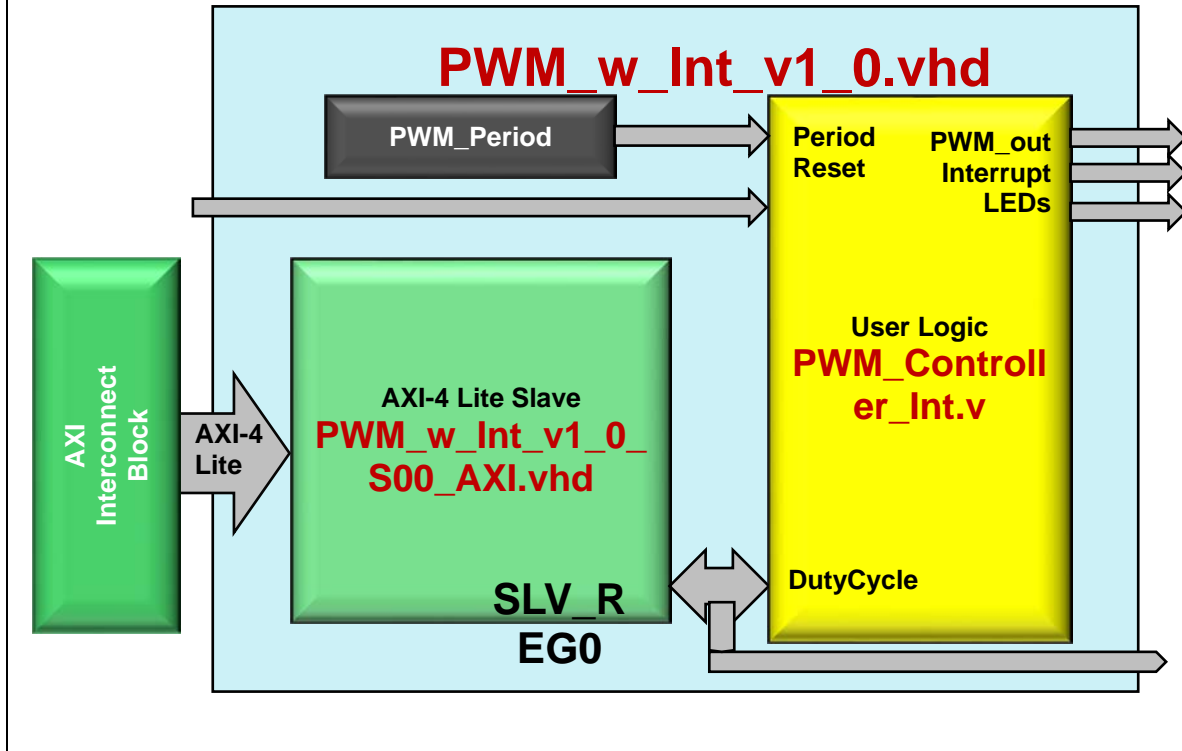
- *Where is the IP project located?*

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 a 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. In Vivado, select **Window** → **IP Catalog**.
2. Select the IP in the IP catalog by searching for "PWM" in the search window. Right click and select **Edit in IP Packager**.

© 2019 Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

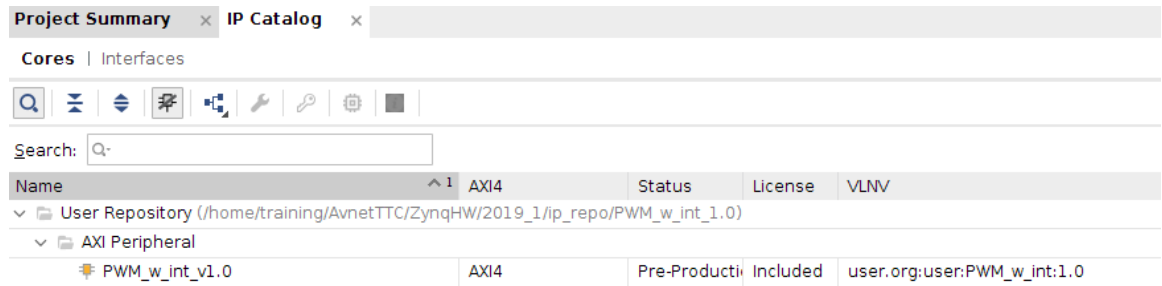


Figure 6 – Edit IP Project

3. Accept the project defaults and click **OK**.

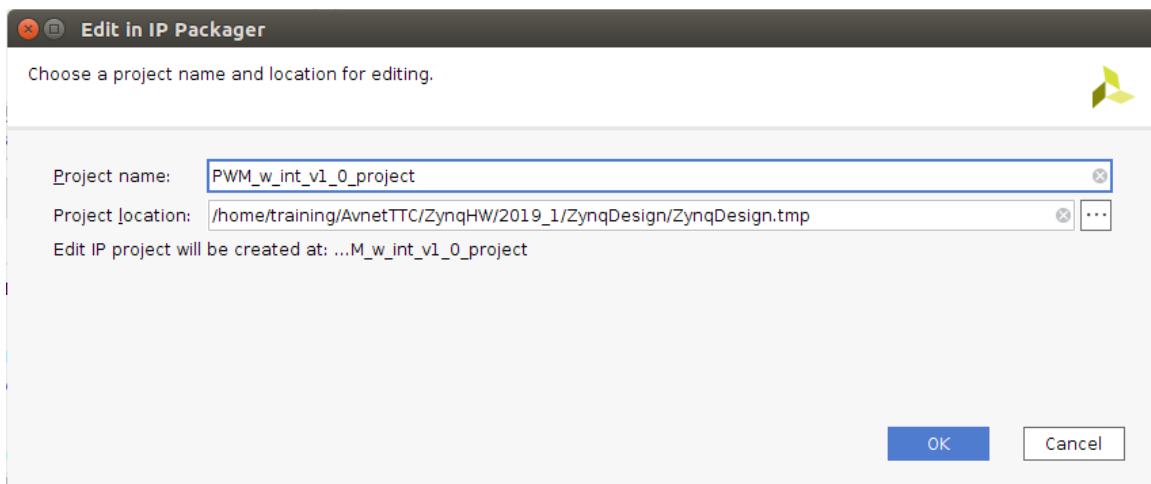


Figure 7 – IP Project Name and Location

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 **<IP Name>_project.xpr** project is the project you will open to edit this IP. In this case, our IP project is named, **PWM_w_Int_v1_0_project.xpr**.

4. 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.

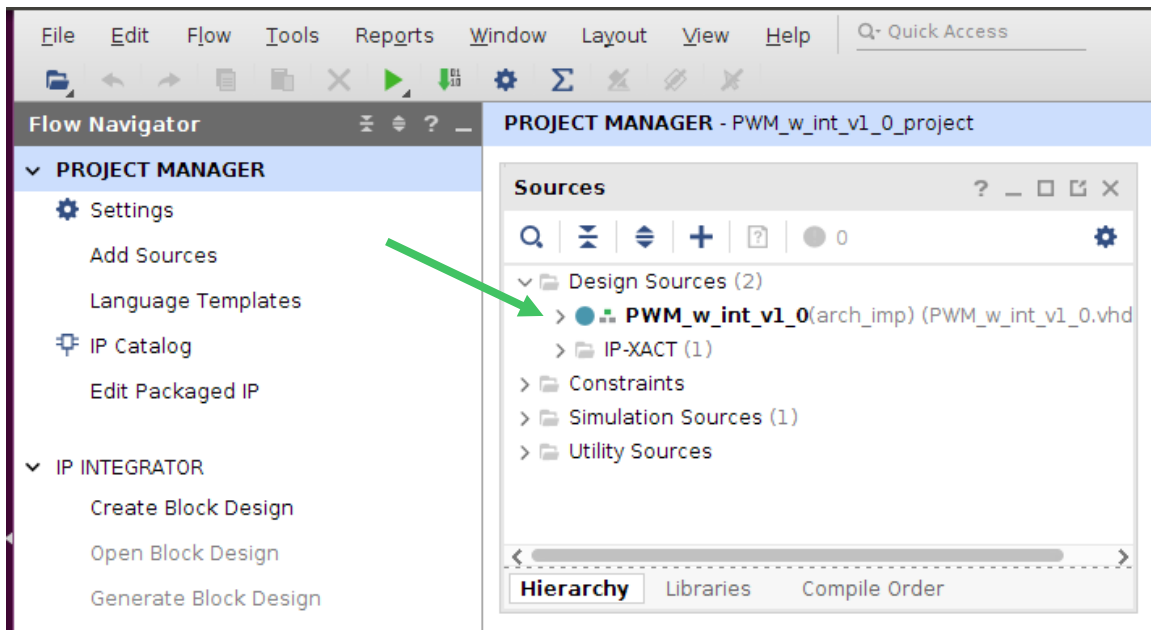


Figure 8 - Design Sources and Properties

© 2019 Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

- There are two VHDL files. One, **PWM_w_Int_v1_0.vhd**, is the top-level wrapper file. **Open** this file by double-clicking on it. It instantiates the AXI interface, **PWM_w_Int_v1_0_S00_AXI** around line 84, which is defined by the 2nd HDL file **PWM_w_Int_v1_0_S00_AXI.vhd**. A place holder exists just below this to add user logic, which is where we will add our custom IP HDL.

```

84  -- Instantiation of Axi Bus Interface S00_AXI
85  PWM_w_Int_v1_0_S00_AXI_inst : PWM_w_Int_v1_0_S00_AXI
86      generic map (
87          C_S_AXI_DATA_WIDTH  => C_S00_AXI_DATA_WIDTH,
88          C_S_AXI_ADDR_WIDTH  => C_S00_AXI_ADDR_WIDTH
89      )
90      port map (
91          S_AXI_ACLK  => s00_axi_aclk,
92          S_AXI_ARESETN  => s00_axi_aresetn,
93          S_AXI_AWADDR  => s00_axi_awaddr,
94          S_AXI_AWPROT  => s00_axi_awprot,
95          S_AXI_AWVALID  => s00_axi_awvalid,
96          S_AXI_AWREADY  => s00_axi_awready,
97          S_AXI_WDATA  => s00_axi_wdata,
98          S_AXI_WSTRB  => s00_axi_wstrb,
99          S_AXI_WVALID  => s00_axi_wvalid,
100         S_AXI_WREADY  => s00_axi_wready,
101         S_AXI_BRESP  => s00_axi_bresp,
102         S_AXI_BVALID  => s00_axi_bvalid,
103         S_AXI_BREADY  => s00_axi_bready,
104         S_AXI_ARADDR  => s00_axi_araddr,
105         S_AXI_ARPROT  => s00_axi_arprot,
106         S_AXI_ARVALID  => s00_axi_arvalid,
107         S_AXI_ARREADY  => s00_axi_arready,
108         S_AXI_RDATA  => s00_axi_rdata,
109         S_AXI_RRESP  => s00_axi_rresp,
110         S_AXI_RVALID  => s00_axi_rvalid,
111         S_AXI_RREADY  => s00_axi_rready
112     );
113
114     -- Add user logic here
115
116     -- User logic ends

```

Figure 9 – User Logic Location

- Now we want to copy **PWM_Controller_Int.v** from the

/home/training/AvnetTTC/ZynqHW/2017_4/Support_documents/Lab7

directory and paste it into

/home/training/AvnetTTC/ZynqHW/2017_4/ip_repo/PWM_w_Int_1.0vhd

© 2019 Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

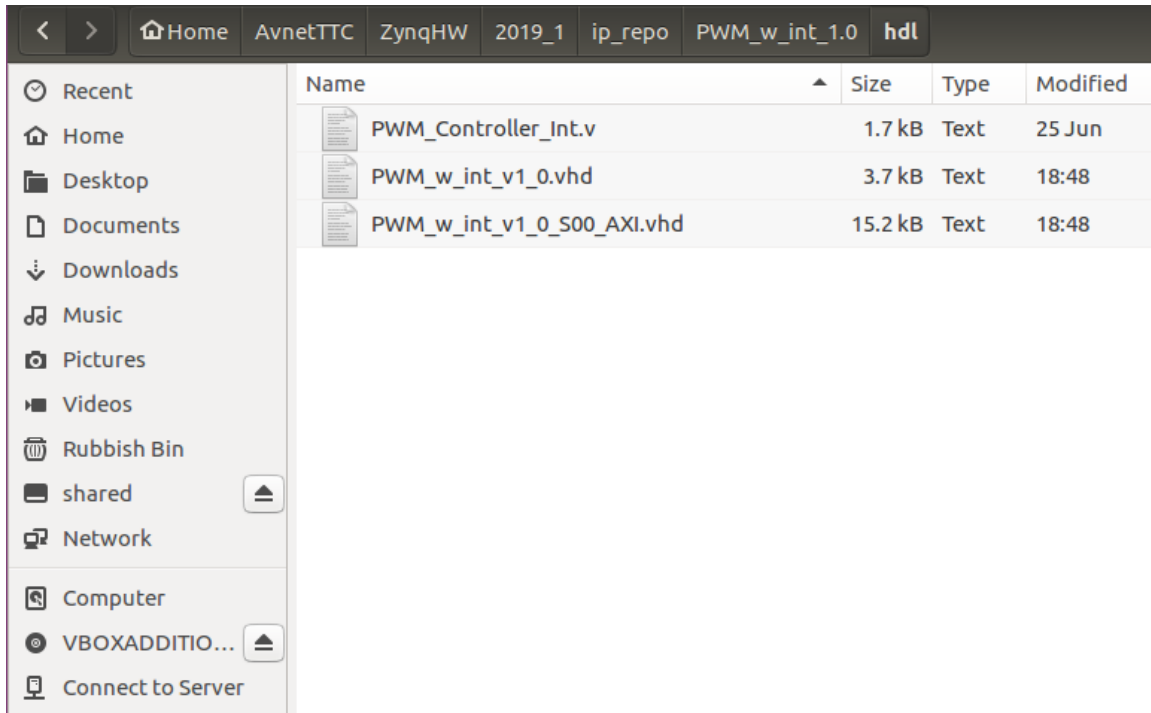


Figure 10 - Copy HDL Files into IP HDL Directory

- Click **Add Sources** from the Flow Navigator window, and then select **Add or create design source**. When done, click **Next >**.

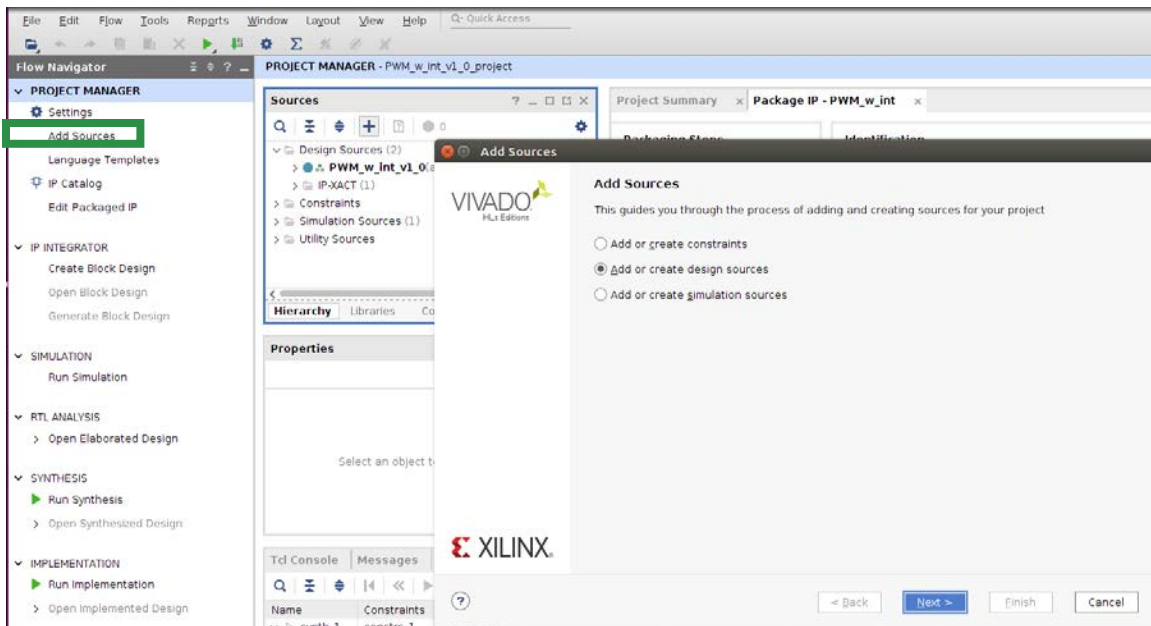


Figure 11 - Add Design Sources

© 2019 Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

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

9. Change the directory to:

/home/training/AvnetTTC/ZynqHW/2019_1/ip_repo/PWM_w_Int_1.0/hdl

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

10. Verify *copy sources into IP Directory* is not checked. Click **Finish**.

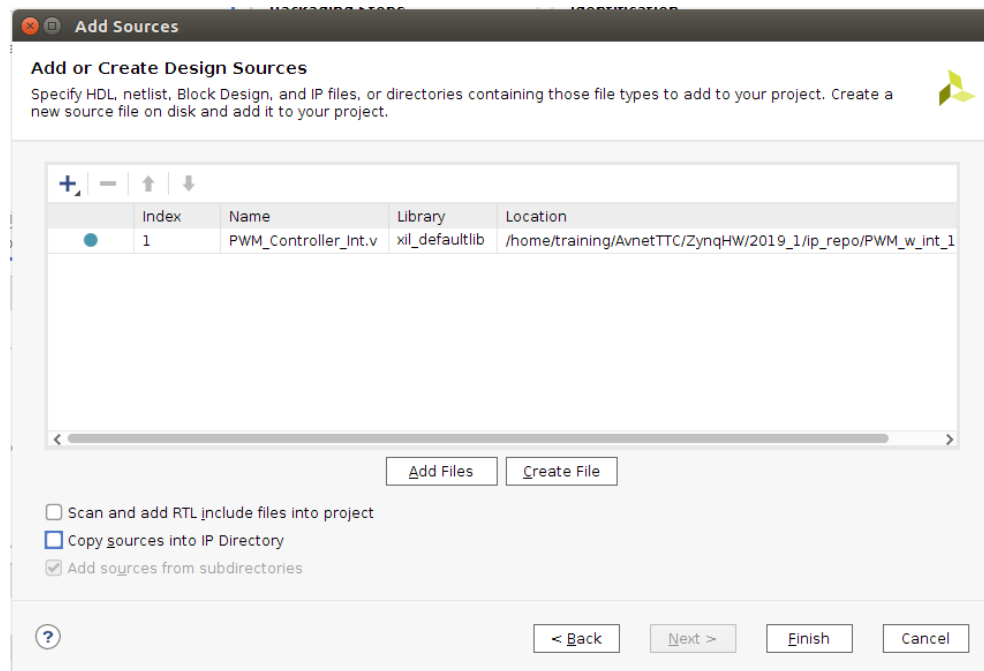


Figure 12 - Add Files to Project

*****Note***** In the directory specified below, there are the three files that require modification in the following steps leading up to Step 22. If you encounter an issue writing the HDL code while synthesizing the design, feel free to copy and paste the file text into your Vivado project.

/home/training/ZynqHW/2019_1/Support_documents/PWM_Controller_Int_Modified

11. 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.

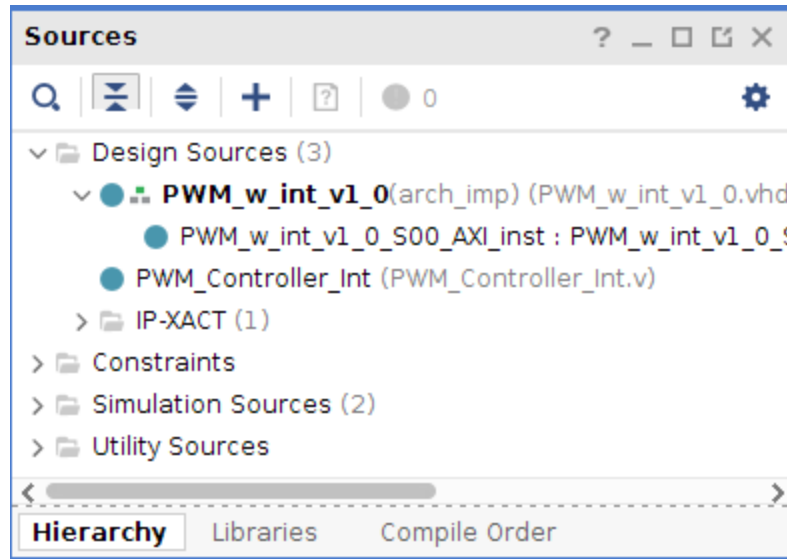


Figure 13 - New Source in Project

12. Open PWM_Controller_Int by double-clicking it. Look over the file and then instantiate it by double clicking on PWM_w_Int_v1_0.vhd. Then declare the component **PWM_Controller_Int** in the component declaration section of **PWM_w_Int_v1_0.vhd** around line 52.

```

51  -- component declaration
52  component PWM_Controller_Int is
53      generic (
54          period : integer := 20
55      );
56      port (
57          Clk : in std_logic;
58          DutyCycle : in std_logic_vector(31 downto 0);
59          Reset : in std_logic;
60          PWM_out : out std_logic_vector(0 downto 0);
61          Interrupt : out std_logic;
62          count : out std_logic_vector(period-1 downto 0)
63      );
64  end component PWM_Controller_Int;

```

Figure 14 – Add Component Declaration

13. Scroll to bottom of **PWM_w_Int_v1_0.vhd** and enter the following in the user logic section (around line 129) :

```

128      -- Add user logic here
129      PWM_Controller_Int_Inst : PWM_Controller_Int
130      generic map (
131          period =>
132      )
133      port map (
134          Clk =>,
135          DutyCycle =>,
136          Reset =>,
137          PWM_out =>,
138          Interrupt =>,
139          count =>
140      );
141      -- User logic ends

```

Figure 15 – Add User Logic

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

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

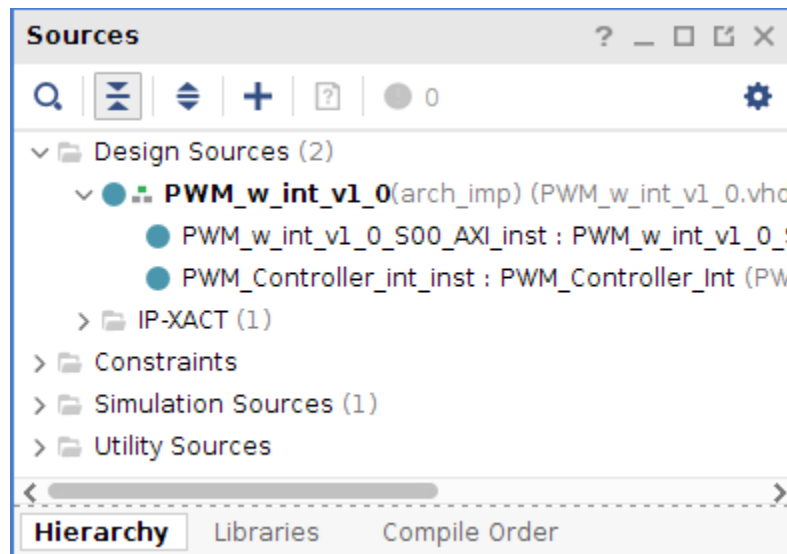


Figure 16 - User Logic now in Project Hierarchy

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 21, else continue.

© 2019 Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

15. Now we need to connect the signals of our user logic, PWM_Controller_Int. Start by connecting the obvious signals in the PWM_Controller_Int instantiation. The clock and reset have the following associations between the Controller signal and the Slave AXI.

s00_axi_aclk to Clk
s00_axi_aresetn to Reset

Edit **PWM_w_Int_v1_0.vhd** so the code looks like this (around line 134):

```
128      -- Add user logic here
129  PWM_Controller_Int_Inst : PWM_Controller_Int
130      generic map (
131          period =>
132          )
133      port map (
134          Clk => s00_axi_aclk,
135          DutyCycle =>,
136          Reset => s00_axi_aresetn,
137          PWM_out =>,
138          Interrupt =>,
139          count =>
140      );
141      -- User logic ends
```

Figure 17 – Add in Clock and Reset

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.

16. Add these ports at the top of the code as well as the instantiation of our user logic HDL(around line 8):


```

5 entity PWM_w_Int_v1_0 is
6     generic (
7         -- Users to add parameters here
8         PWM_PERIOD : integer :=20;
9         -- User parameters ends
10        -- Do not modify the parameters beyond this line
11
12
13        -- Parameters of Axi Slave Bus Interface S00_AXI
14        C_S00_AXI_DATA_WIDTH    : integer    := 32;
15        C_S00_AXI_ADDR_WIDTH    : integer    := 4
16    );
17    port (
18        -- Users to add ports here
19        LED : out std_logic_vector(0 downto 0);
20        Interrupt_Out : out std_logic;
21        PWM_Counter : out std_logic_vector(PWM_PERIOD-1 downto 0);
22        DutyCycle : out std_logic_vector(31 downto 0);
23        -- User ports ends
24        -- Do not modify the ports beyond this line

```

Figure 18 – Add Outputs

(Add Signal around line 99)

```

98
99 signal DutyCycle_int : std_logic_vector(31 downto 0);
100
101 begin
102
103 DutyCycle <= DutyCycle_int;
104

```

Figure 19 – Add Signals

(Complete user logic around line 138)

```

135      -- Add user logic here
136  PWM_Controller_Int_Inst : PWM_Controller_Int
137      generic map (
138          period => PWM_PERIOD
139      )
140      port map (
141          Clk => s00_axi_aclk,
142          DutyCycle => DutyCycle_int,
143          Reset => s00_axi_aresetn,
144          PWM_out => LED ,
145          Interrupt => Interrupt_Out,
146          count => PWM_Counter
147      );
148      -- User logic ends

```

Figure 20 - Add User Logic

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.

17. Open **PWM_w_Int_v1_0_S00_AXI.vhd**. Look over the file, scroll down to line 100 and 202. Here you will see a commented explanation and the definitions for the four registers created by Vivado through this AXI proxy. We only need one of these registers, slv_reg0.

18. Make slv_reg0 a module output. Around line 19, add an output (**slave_reg0** for example) under User Ports. Then connect it to slv_reg0 (around line 123) :

```

17      port (
18          -- Users to add ports here
19          slave_reg0: out std_logic_vector(C_S_AXI_DATA_WIDTH-1 downto 0);
20      -- User ports ends
21      -- Do not modify the ports beyond this line

```

Figure 21 - Add Slave Module Output

```

122      -- I/O Connections assignments
123      slave_reg0 <= slv_reg0;
124      S_AXI_AWREADY <= axi_awready;
125      S_AXI_WREADY <= axi_wready;
126      S_AXI_BRESP <= axi_bresp;
127      S_AXI_BVALID <= axi_bvalid;
128      S_AXI_ARREADY <= axi_arready;
129      S_AXI_RDATA <= axi_rdata;
130      S_AXI_RRESP <= axi_rresp;
131      S_AXI_RVALID <= axi_rvalid;

```

Figure 22 – Connect Slave Register

19. **Save** the file, CTRL-S, and return to the top-level file, **PWM_w_Int_v1_0.vhd**.

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.

20. Add the slave_reg0 port to the slave AXI interface component declaration (around line 75).

```

69      component PWM_w_Int_v1_0_S00_AXI is
70      generic (
71          C_S_AXI_DATA_WIDTH : integer := 32;
72          C_S_AXI_ADDR_WIDTH : integer := 4;
73      );
74      port (
75          slave_reg0: out std_logic_vector(C_S_AXI_DATA_WIDTH-1 downto 0);
76          S_AXI_ACLK : in std_logic;
77          S_AXI_ARESETN : in std_logic;
78          S_AXI_AWADDR : in std_logic_vector(C_S_AXI_ADDR_WIDTH-1 downto 0);

```

Figure 23 - Add Files to Project

Connect it to **DutyCycle_int** signal, Around line 113. **Save** when done.

```

107      PWM_w_Int_v1_0_S00_AXI_inst : PWM_w_Int_v1_0_S00_AXI
108      generic map (
109          C_S_AXI_DATA_WIDTH => C_S00_AXI_DATA_WIDTH,
110          C_S_AXI_ADDR_WIDTH => C_S00_AXI_ADDR_WIDTH
111      )
112      port map (
113          slave_reg0 => DutyCycle_int,
114          S_AXI_ACLK => s00_axi_aclk,

```

Figure 24 - Add Files to Project

© 2019 Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

21. That completes all connections. Select the top-level HDL file, PWM_w_Int_v1_0.vhd, then click **Run Synthesis** to validate the design. Click yes to save is prompted then OK on the launch runs window.
22. Do not run implementation. Open the synthesized design from the *Synthesis Completed* dialog box.

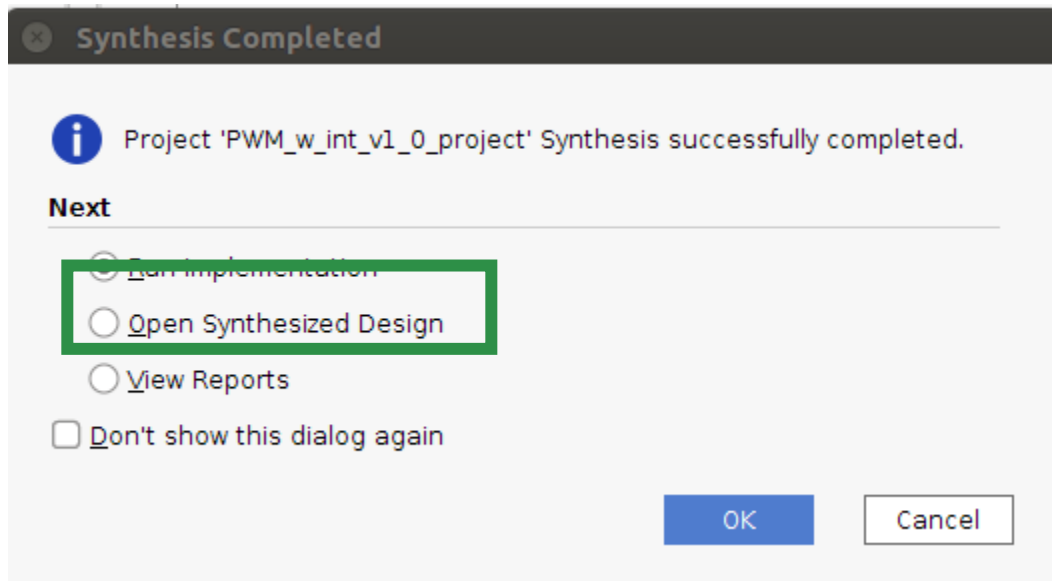


Figure 25 – Synthesis Completed

23. Once the synthesized design opens, open the schematic view by clicking on **Schematic** under *Synthesis* in the *Flow Navigator*. Check that all the expected I/O are present. If it looks correct the design should be good. In a real world design further verification would be in order, but for now we can package the IP.

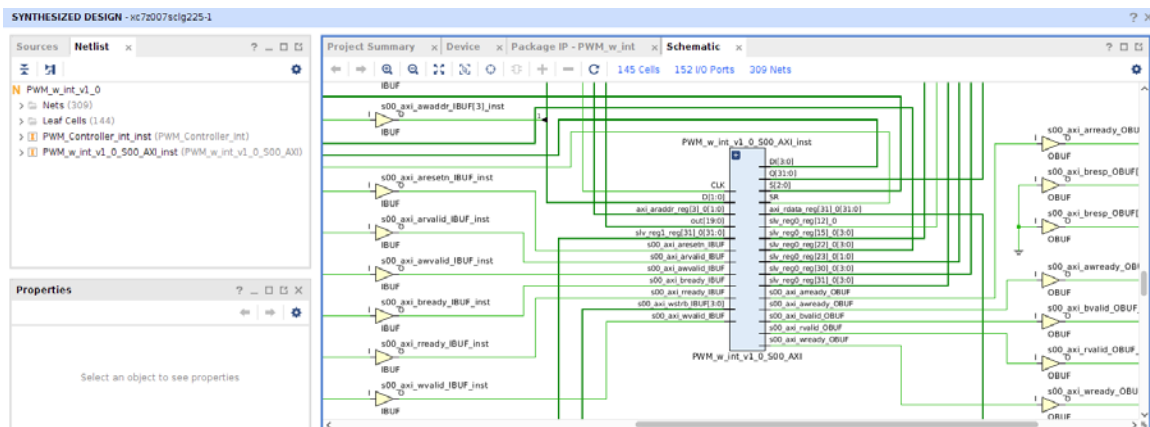


Figure 26 – Synthesis View

© 2019 Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

Experiment 3: Package the 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 **Edit Packaged IP** in the Flow Navigator Window.

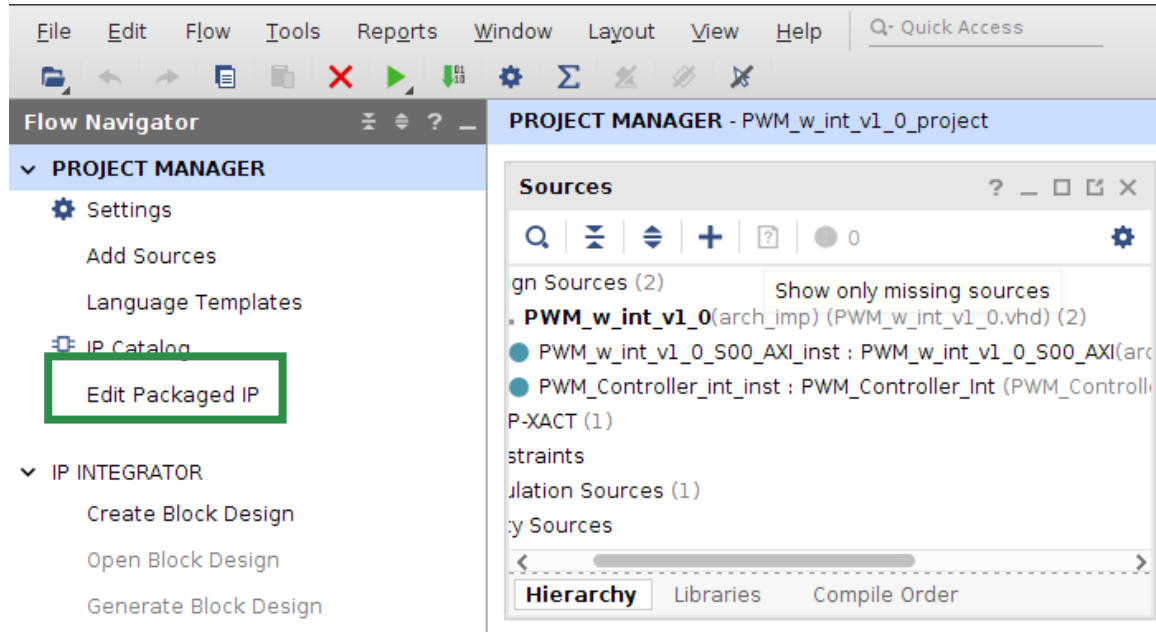


Figure 27 - Package IP

2. This GUI will build our IP step-by-step 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.

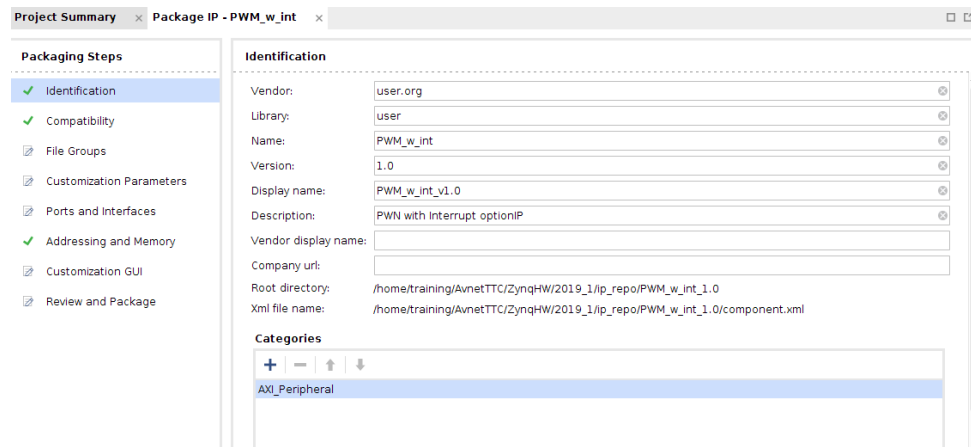


Figure 28 - IP Identification

3. Switch to the next setting, *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. Change the *Life Cycle* to **Production** by **selecting** it in the field shown in figure 28.

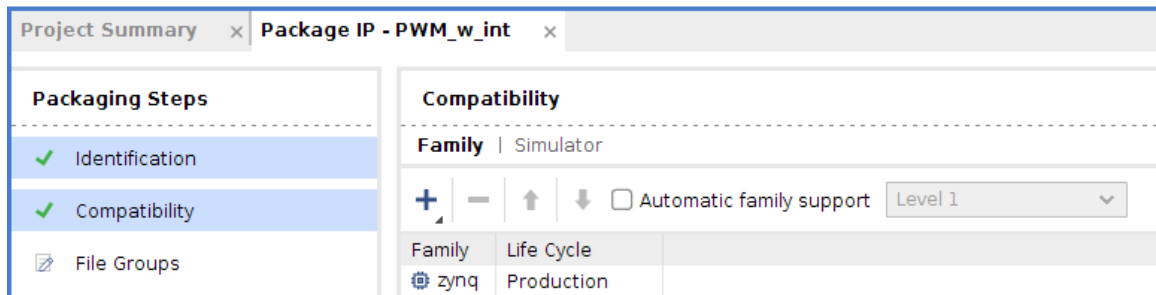


Figure 29 - IP Compatibility

© 2019 Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

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

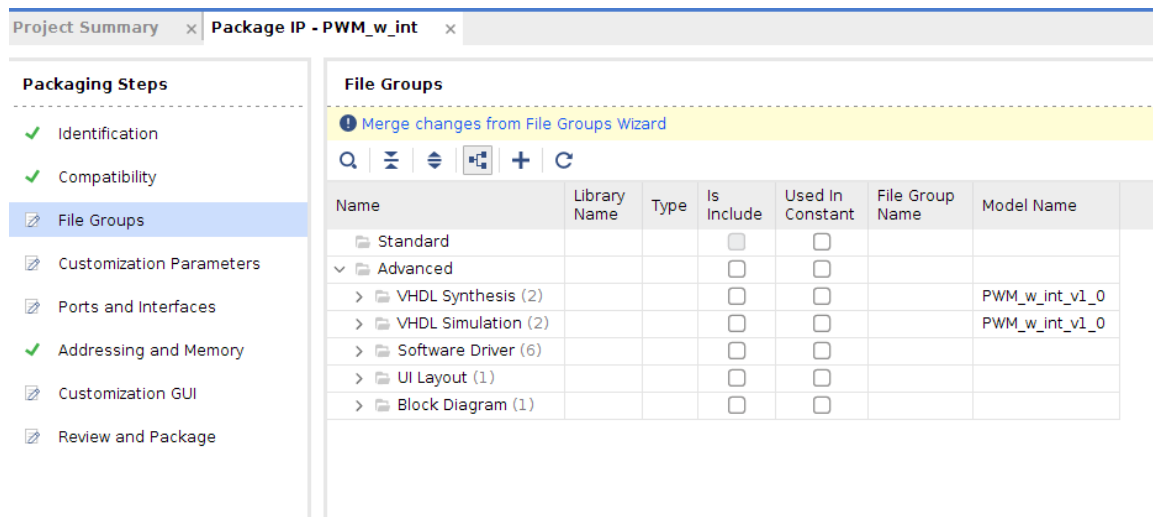


Figure 30 - IP File Groups

- Once the changes have merged, expand all files. Notice it's added our **PWM_Controller_Int** user logic HDL. However, this Verilog module is within VHDL groups, hence the critical warnings.

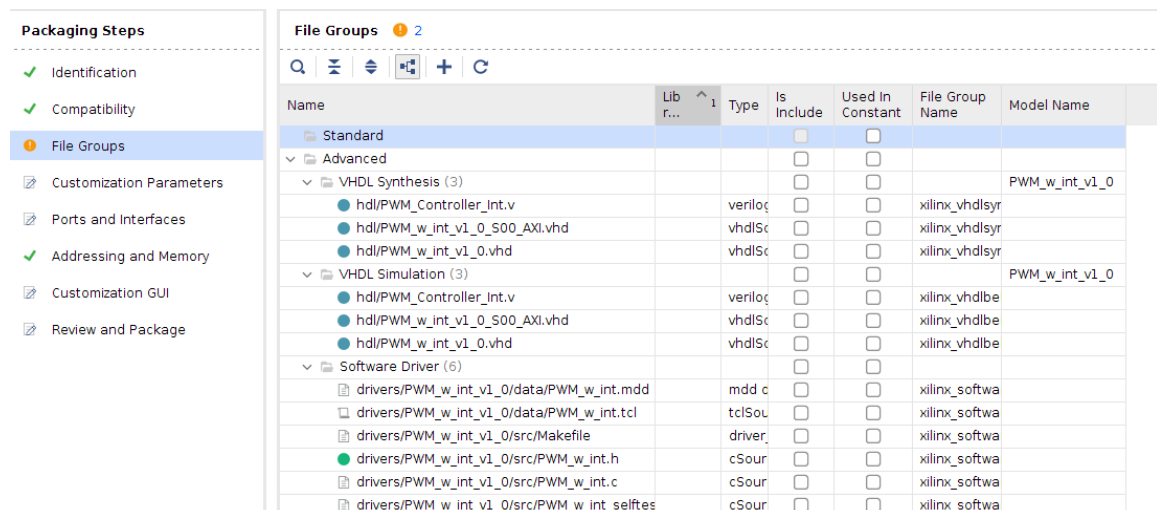


Figure 31 – File Groups

- Select **Standard**, right-click and **Add File Group...** To use both VHDL and Verilog modules, select **Synthesis** and **Simulation** groups and click **OK**. Use the <CTRL> key to select both synthesis and simulation.

© 2019 Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

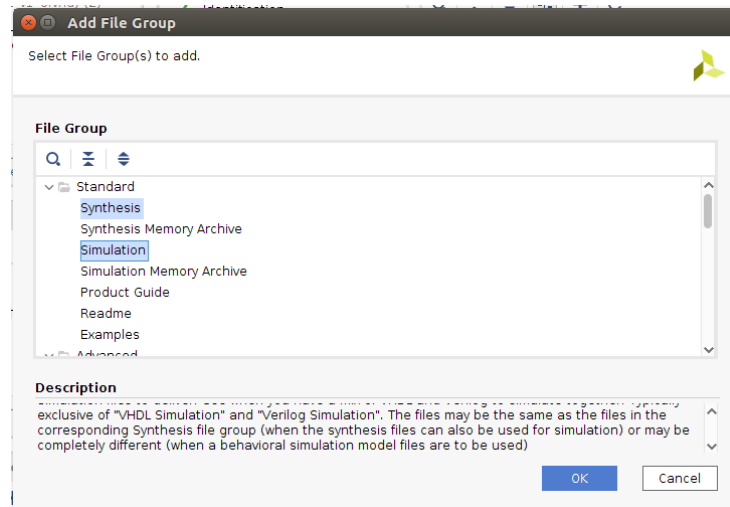


Figure 32 - Add File Group

7. Now select **Synthesis**, right-click and select **Add Files**.
8. When the Add IP Files window opens, click **Add Files**, Vivado should immediately open the HDL directory. If not, navigate to:

`/home/training/AvnetTTC/ZynqHW/2019_1/ip_repo/PWM_w_Int_1.0/hdl`

Change 'Files of type' option to **All Files**. Select **PWM_Controller_Int.v**, **PWM_w_Int_v1_0_S00_AXI.vhd** and **PWM_w_Int_v1_0.vhd**. Click **OK**. Make sure to not copy sources into project.

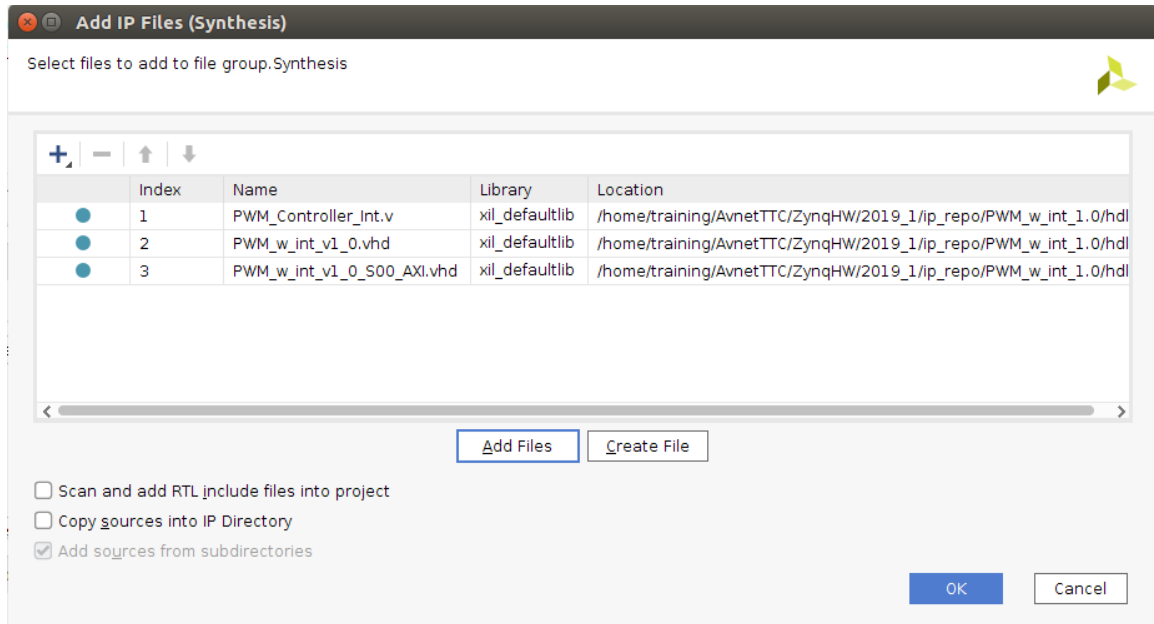


Figure 33 - Add HDL Files to IP File Groups

9. **Repeat** this process for *Simulation* group.

10. The IP Packager will give **errors**.

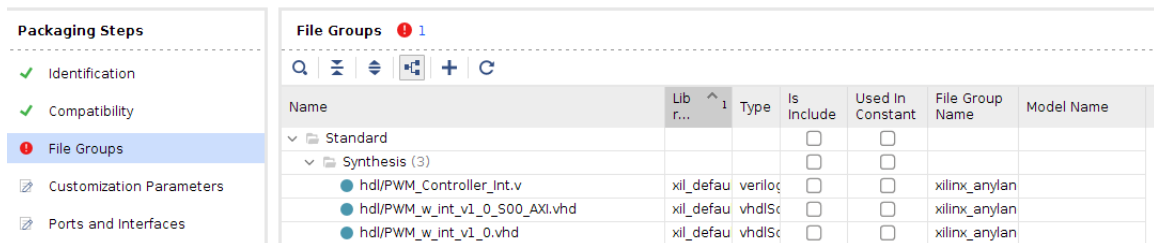


Figure 34 – Two Errors and Warnings

11. Specify **PWM_w_Int_v1_0** in **Model Name** for **Synthesis** and **Simulation** groups under the **Standard** folder. The IP File Groups should appear as follows now, and the 2 errors will disappear :

Packaging Steps		File Groups 2					
✓ Identification							
✓ Compatibility							
⚠ File Groups							
Customization Parameters							
Ports and Interfaces							
✓ Addressing and Memory							
Customization GUI							
Review and Package							

Name	Library Name	Type	Is Include	Used In Constant	File Group Name	Model Name
Standard			<input type="checkbox"/>	<input type="checkbox"/>		
▼ Synthesis (3)			<input type="checkbox"/>	<input type="checkbox"/>		PWM_w_Int_v1_0
C:/Speedway/Zynq-HW/2017_4/Ip_repo/PWM_w_In...	xil_de...	vhdlSource	<input type="checkbox"/>	<input type="checkbox"/>	xilinx_anylanguagesynthesis	
C:/Speedway/Zynq-HW/2017_4/Ip_repo/PWM_w_In...	xil_de...	verilogSource	<input type="checkbox"/>	<input type="checkbox"/>	xilinx_anylanguagesynthesis	
C:/Speedway/Zynq-HW/2017_4/Ip_repo/PWM_w_In...	xil_de...	vhdlSource	<input type="checkbox"/>	<input type="checkbox"/>	xilinx_anylanguagesynthesis	
▼ Simulation (3)			<input type="checkbox"/>	<input type="checkbox"/>		PWM_w_Int_v1_0
C:/Speedway/Zynq-HW/2017_4/Ip_repo/PWM_w_In...	xil_de...	vhdlSource	<input type="checkbox"/>	<input type="checkbox"/>	xilinx_anylanguagebehavioral...	
C:/Speedway/Zynq-HW/2017_4/Ip_repo/PWM_w_In...	xil_de...	verilogSource	<input type="checkbox"/>	<input type="checkbox"/>	xilinx_anylanguagebehavioral...	
C:/Speedway/Zynq-HW/2017_4/Ip_repo/PWM_w_In...	xil_de...	vhdlSource	<input type="checkbox"/>	<input type="checkbox"/>	xilinx_anylanguagebehavioral...	
Advanced			<input type="checkbox"/>	<input type="checkbox"/>		
▼ VHDL Synthesis (3)			<input type="checkbox"/>	<input type="checkbox"/>		PWM_w_Int_v1_0

Figure 35 – Errors Disappear

12. Switch to the *Customization Parameters* page. Again, changes are detected and must be merged. Click **Merge changes...**

Packaging Steps		Customization Parameters					
✓ Identification		Merge changes from Customization Parameters Wizard					
✓ Compatibility							
⚠ File Groups							
Customization Parameters							
Ports and Interfaces							
✓ Addressing and Memory							
Customization GUI							
Review and Package							

Name	Description	Display Name	Value	Value Bit String Length	Value Format	Value Source
▼ Customization Parameters						
C_S00_AXI_DATA_WIDTH	Width of S_AXI data bus	C S00 AXI DATA WIDTH	32	0	long	default
C_S00_AXI_ADDR_WIDTH	Width of S_AXI address bus	C S00 AXI ADDR WIDTH	4	0	long	default
C_S00_AXI_BASEADDR		C S00 AXI BASEADDR	0xFFFFFFFF	32	bitString	default
C_S00_AXI_HIGHADDR		C S00 AXI HIGHADDR	0x00000000	32	bitString	default

Figure 36 - IP Customization Parameters

Our **PWM_PERIOD** parameter appears in the **Hidden Parameters** folder.

Packaging Steps		Customization Parameters					
✓ Identification							
✓ Compatibility							
⚠ File Groups							
Customization Parameters							
Ports and Interfaces							
✓ Addressing and Memory							
Customization GUI							
Review and Package							

Name	Description	Display Name	Value	Value Bit String Length	Value Format
▼ Customization Parameters					
C_S00_AXI_DATA_WIDTH	Width of S_AXI data bus	C S00 AXI DATA WIDTH	32	0	long
C_S00_AXI_ADDR_WIDTH	Width of S_AXI address bus	C S00 AXI ADDR WIDTH	4	0	long
C_S00_AXI_BASEADDR		C S00 AXI BASEADDR	0xFFFFFFFF	32	bitString
C_S00_AXI_HIGHADDR		C S00 AXI HIGHADDR	0x00000000	32	bitString
▼ Hidden Parameters					
PWM_PERIOD		Pwm Period	20	0	long

Figure 37 - User Parameters Added

© 2019 Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

13. Proceed back to File Groups and remove **VHDL Synthesis** and **VHDL Simulation** groups.

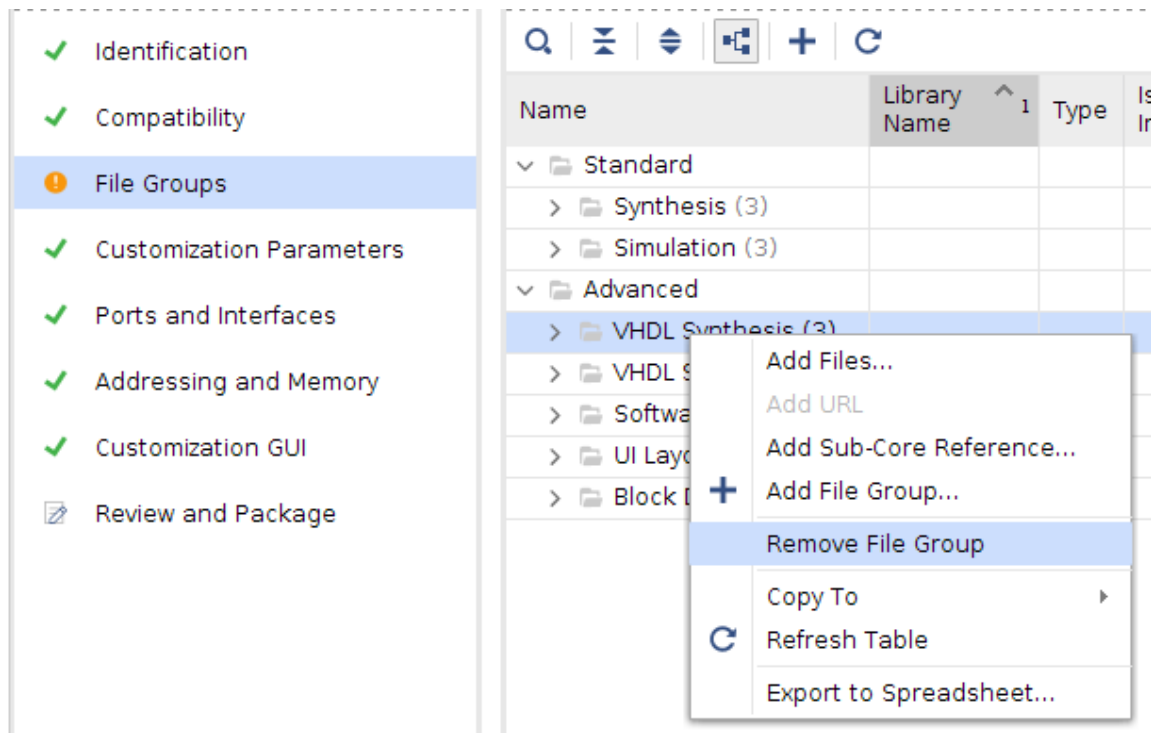


Figure 38 – Remove VHDL Synthesis and Simulation Groups

© 2019 Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

14. Return to Customization Parameters and double click on PWM_PERIOD and check the **Visible in Customization GUI**, then click on **OK**.

Edit IP Parameter

Use the options below to customize how the parameter will appear in the Customization GUI for users of the IP.

Name: PWM_PERIOD

☒ Visible in Customization GUI

☒ Show Name

Display Name: Pwm Period

Tooltip:

Format: long

Editable: Yes

Dependency: No

☐ Specify Range

Type: List of values

Press the button to add a value

Show As: Not Applicable

Layout: Not Applicable

Default Value: 20

OK Cancel

Figure 39 – Edit IP Parameter Window

© 2019 Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

This will allow it to be viewed in the Customization Parameters folder and not labeled as a Hidden parameter.

✓ Identification

✓ Compatibility

✓ File Groups

✓ Customization Parameters

✓ Ports and Interfaces

✓ Addressing and Memory

Customization Parameters

Q

≡

⚙

+

↺

Name	Description	Display Name	Value	Value Bit String Length
Customization Parameters				
C_S00_AXI_DATA_WIDTH	Width of S_AXI data bus	C S00 AXI DATA WIDTH	32	0
C_S00_AXI_ADDR_WIDTH	Width of S_AXI address bus	C S00 AXI ADDR WIDTH	4	0
C_S00_AXI_BASEADDR		C S00 AXI BASEADDR	0xFFFFFFFF	32
C_S00_AXI_HIGHADDR		C S00 AXI HIGHADDR	0x00000000	32
PWM_PERIOD		Pwm Period	20	0

Figure 40 - User Parameters Added to Customization Folder

15. Next, select the *Ports and Interfaces* tab.

All of our user ports should be on the port list as shown below:

Packaging Steps

✓ Identification

✓ Compatibility

✓ File Groups

✓ Customization Parameters

✓ Ports and Interfaces

✓ Addressing and Memory

✓ Customization GUI

📝 Review and Package

Ports and Interfaces

🔍

⌵

⚙

+

🔌

↺

Name	Interface Mode	Enablement Dependency	Is Declaration	Access Handle	Access Type	Direction
> 🔌 S00_AXI	slave		<input type="checkbox"/>			
> 📁 Clock and Reset Signals			<input type="checkbox"/>			
🔌 led			<input type="checkbox"/>		ref	out
🔌 Interrupt_out			<input type="checkbox"/>		ref	out
🔌 PWM_Counter			<input type="checkbox"/>		ref	out
🔌 DutyCycle			<input type="checkbox"/>		ref	out

Figure 41 - User IO Ports

© 2019 Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

16. Right click on **Interrupt_Out** → **Auto Infer Single Bit Interface** → **Interrupt** to define the interrupt

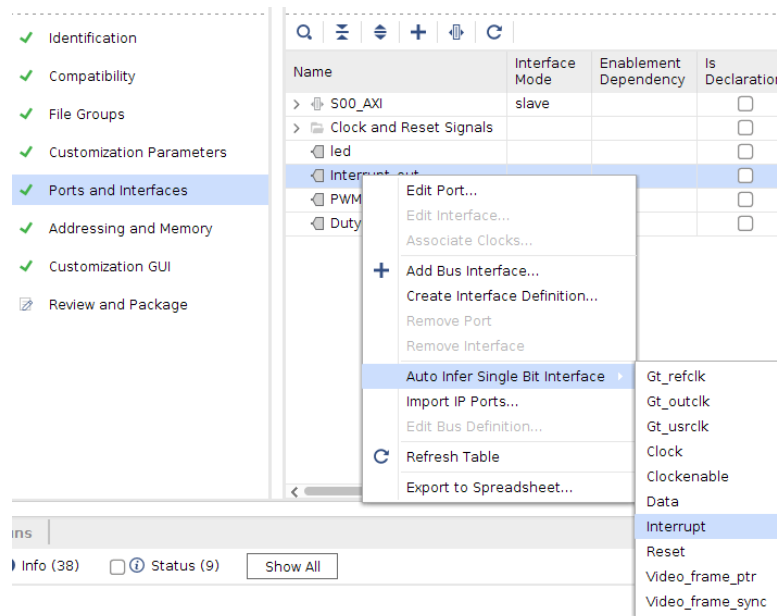


Figure 42 – Defining Interrupt

17. Move to **Addressing and Memory**.

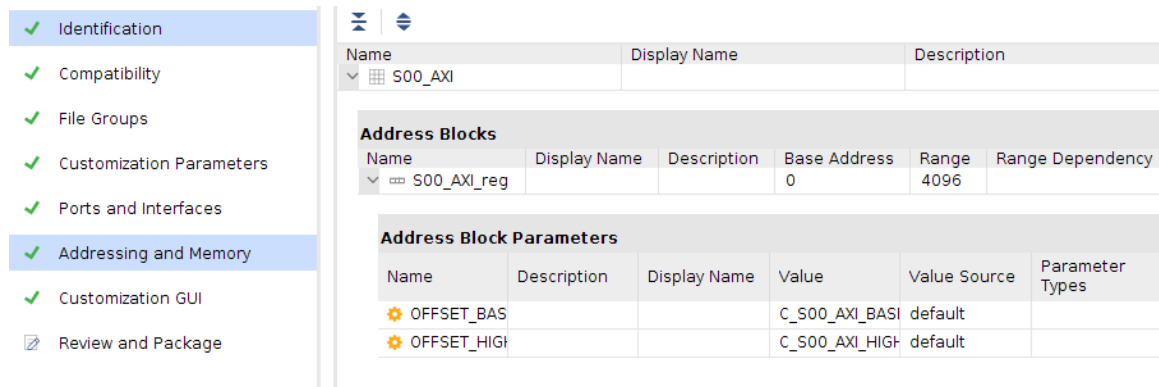


Figure 43 - Addressing and Memory Map

© 2019 Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

There is no memory or BRAM in this IP. This screen lists the memory map for the Slave AXI interface.

18. Next, select the *Customization GUI* tab.

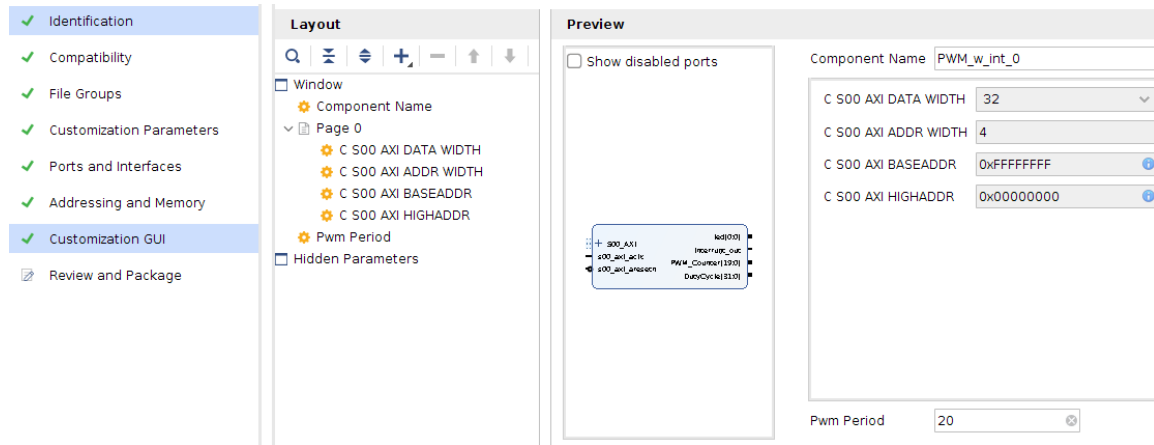


Figure 44 - IP GUI Preview

19. elect **Review and Package** to complete the IP Packaging.

© 2019 Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

20. Review the settings and click the **Re-Package IP** button. If you were to edit this IP package later on and re-package the edits you make, you will then see a button labeled **Re-Package IP** here instead.

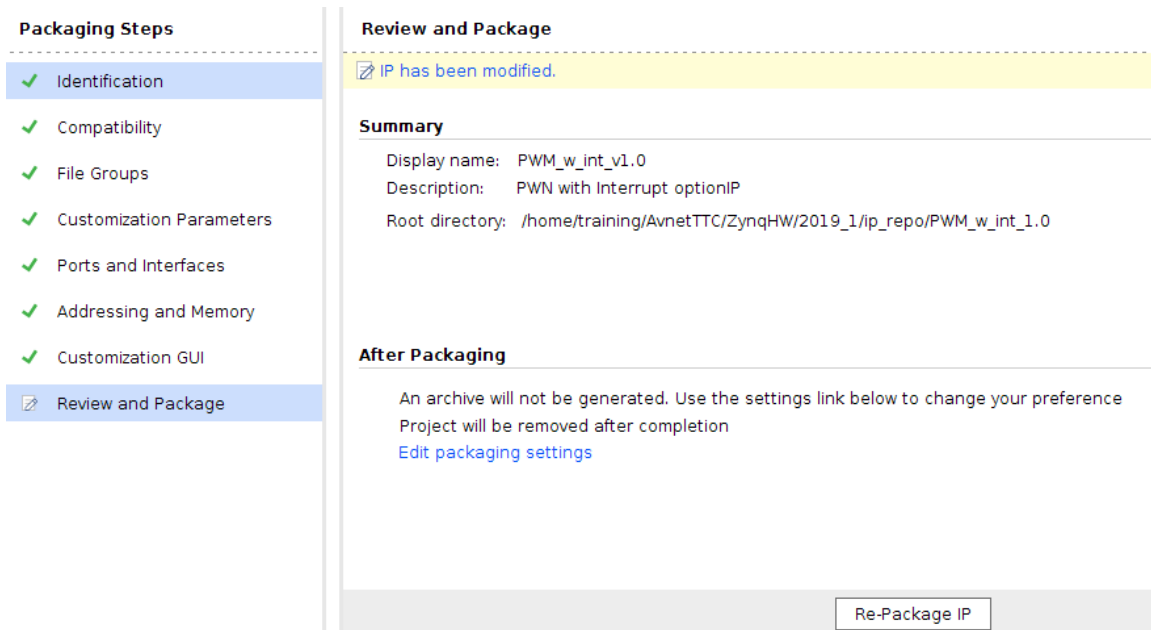


Figure 45 - Package IP

21. When finished, it will ask to close the IP project. Click Yes.
22. Using a file explorer, navigate to our IP repository. Notice our IP project is now in the repository:

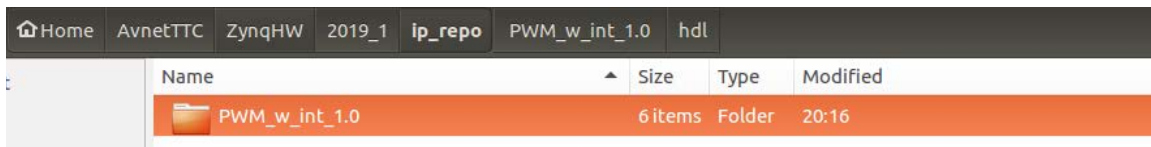
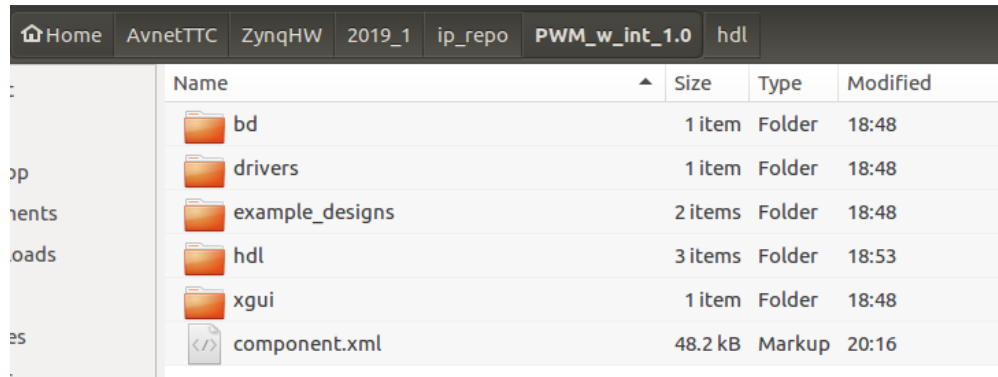


Figure 46 - IP Repository

© 2019 Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

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



Name	Size	Type	Modified
bd	1 item	Folder	18:48
drivers	1 item	Folder	18:48
example_designs	2 items	Folder	18:48
hdl	3 items	Folder	18:53
xgui	1 item	Folder	18:48
component.xml	48.2 kB	Markup	20:16

Figure 47 - IP File Structure and Directories

That's it, which completes our creating of 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 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 the project and connect it to the design.

Experiment 4 Step-by-Step Instructions:

1. Return to our **ZynqDesign** Vivado Project.
2. Open the block design.

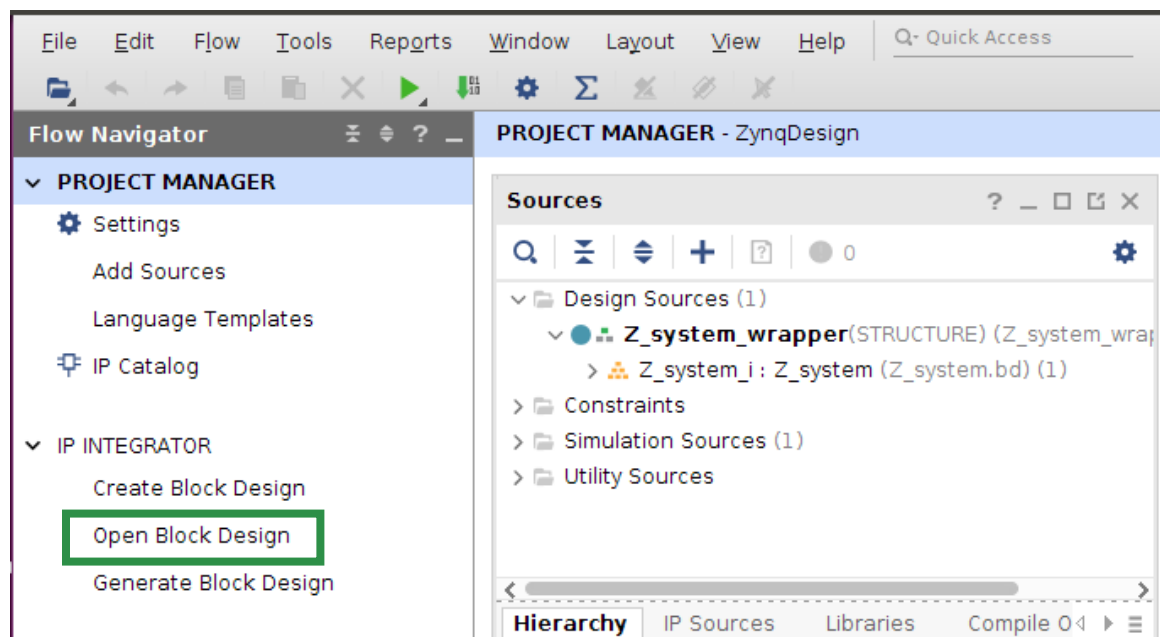




Figure 48 - Open Block Design

3. Select the *Add IP* icon  or right-click inside the *Diagram Window* and select *Add IP*.
4. Enter "PWM" and Double-click **PWM_w_Int_v1.0** to add it to the design.

© 2019 Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

- Click the **Regenerate Layout** button from the vertical shortcut bar . This is what should be displayed:

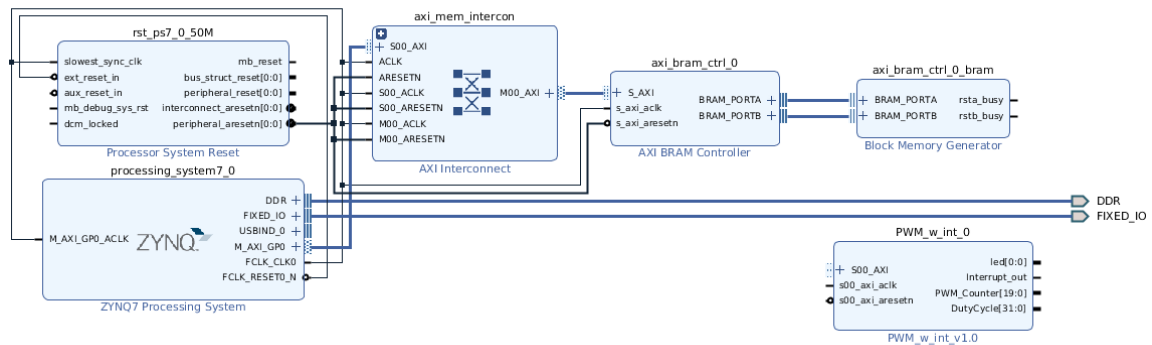


Figure 49 - IP Core Inserted

As we asked in an earlier lab, how would we connect additional AXI slave IP? One answer, add more AXI Master Interfaces to the AXI Interconnect core. A second possibility, which we will not do now, is to enable and use the second general purpose AXI Master port from the ZYNQ7 Processing System. However, the tool is going to take care of this for us!

- Click **Run Connection Automation**.
- Make sure **All Automation** is checked, then click **OK**.

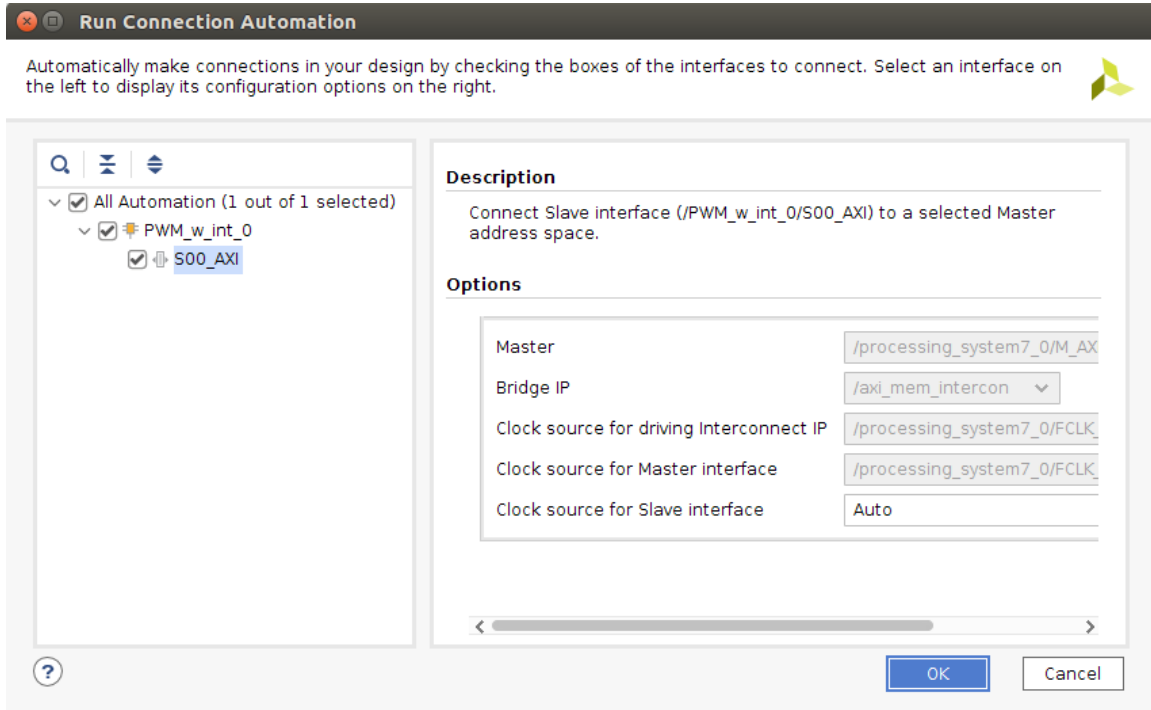


Figure 50 – Run Connection Automation for PWM

The automation properly adjusts the AXI Interconnect block to have a 2nd Master. Then the AXI, clock, and reset are connected, as shown below.

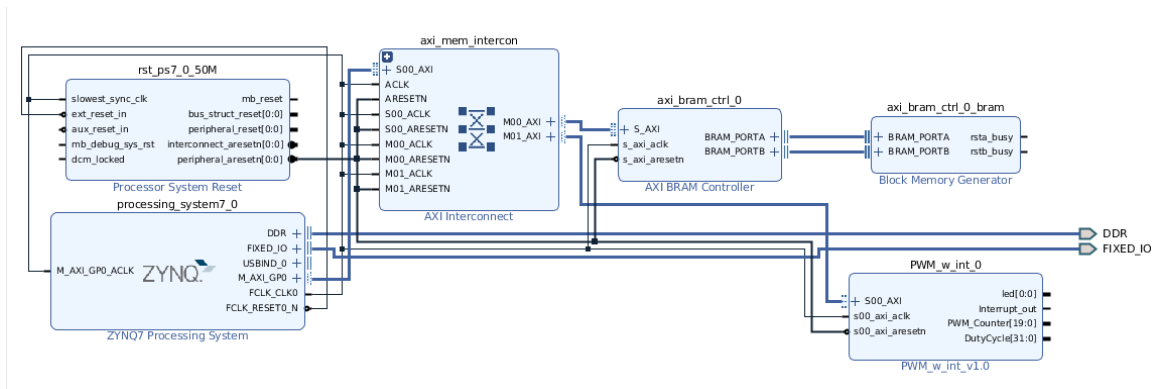


Figure 51 – PWM Controller AXI, clk, and reset Connected

8. The PWM_w_Int IP has an interrupt output, but the ZYNQ7 Processing System IP block has not been enabled to accept PL interrupts yet. **Double-click** the ZYNQ7 Processing System IP block.
9. Select **Interrupts** from the Page Navigator.

© 2019 Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

10. Check the box for **Fabric Interrupts** then **expand** it.
11. Expand **PL-PS Interrupt Ports** and check the box next to **IRQ_F2P[15:0]**. F2P is Fabric to PS. When done, click **OK**.

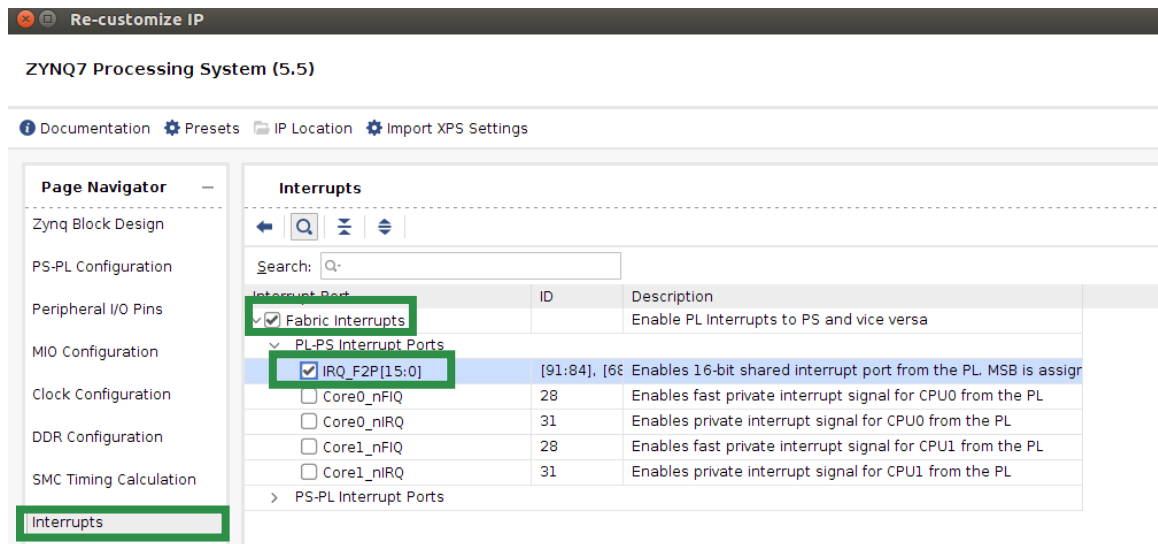


Figure 52 - Add PL-PS Interrupts

12. Connect **Interrupt_Out** from the PWM_w_Int_v1.0 IP block to **IRQ_F2P[0:0]** on the ZYNQ7 Processing System IP block by clicking on either pin creating a line then clicking on the other pin.

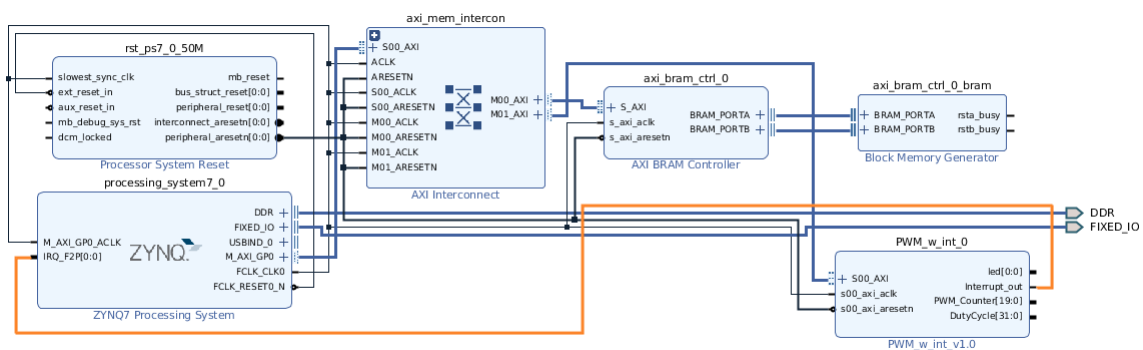


Figure 53 - Connected Interrupt

Finally, we are left with the PWM outputs.

13. Access the **Add IP** again, search for **ILA**:

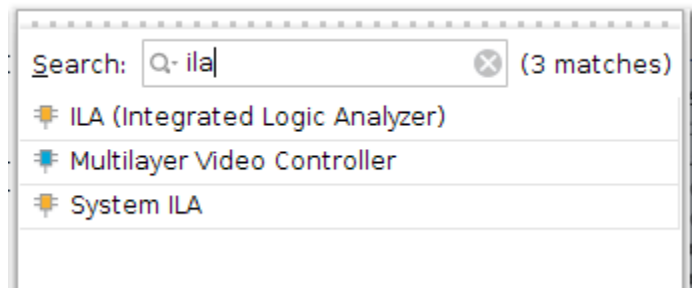


Figure 54 - ILA IP Core

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

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

16. Set the *Monitor Type* to “Native” and set the number of *Probes* to **4**. Then click the **Probe_Ports(0...7)** Tab.

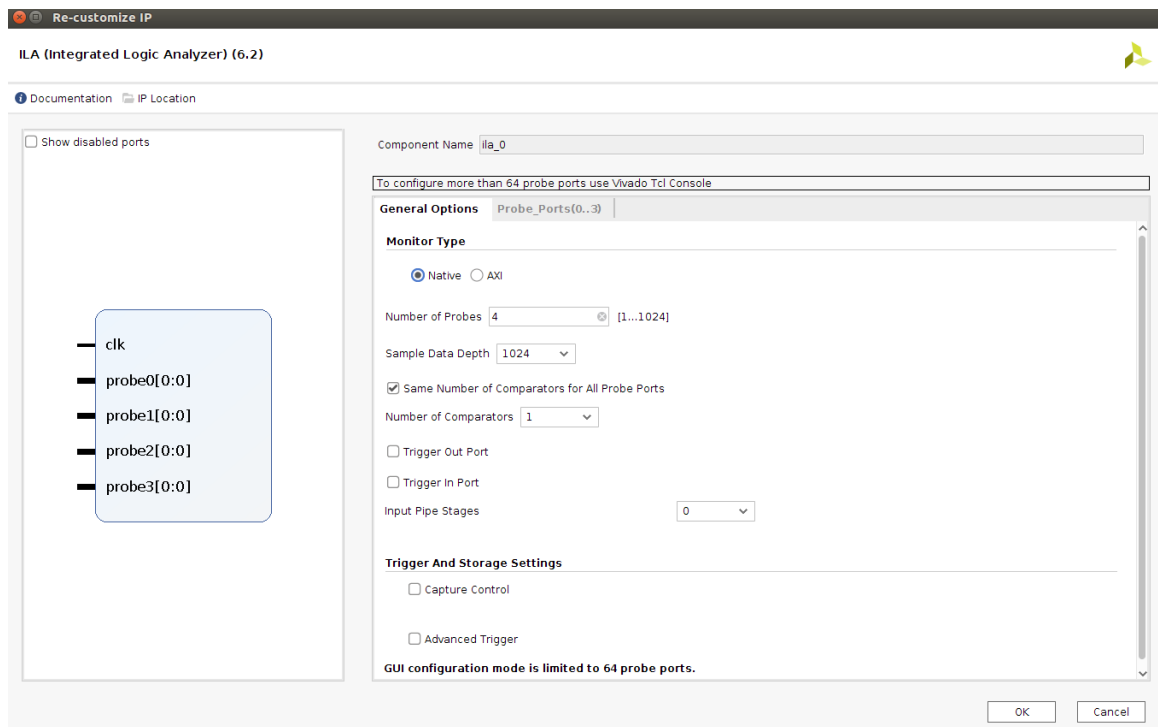


Figure 55 - Customize ILA IP

© 2019 Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

17. Change the Probes Widths to match the following:

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

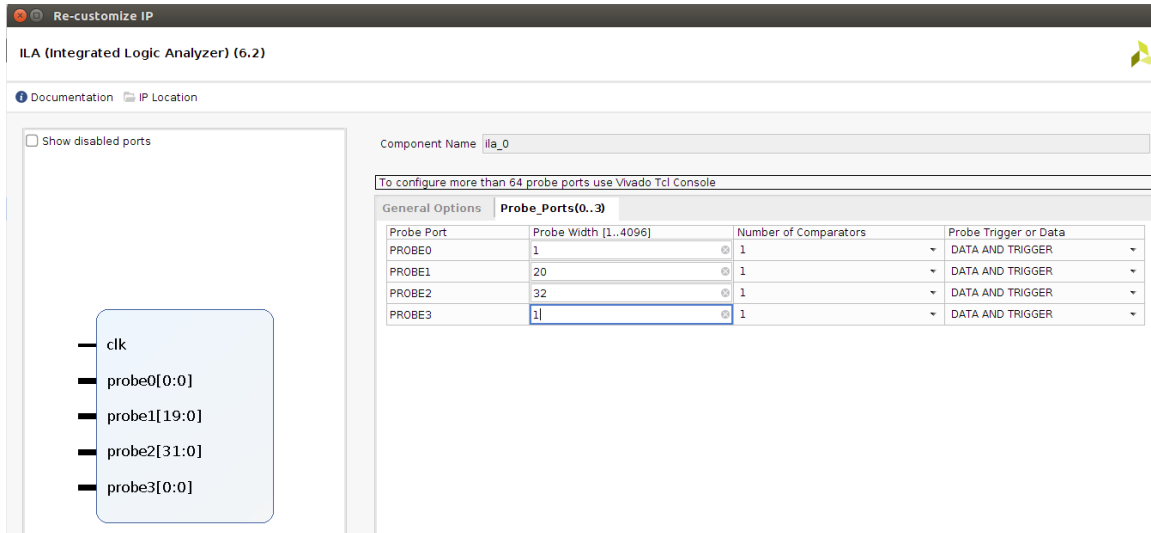


Figure 56 - ILA Probes Configuration

18. Click **OK**.

19. Connect **LED** to **probe0**, **PWM_Counter** to **probe1**, **DutyCycle** to **probe2** and **Interrupt_out** to **probe3**. Connect the clk to **FCLK_CLK0**.

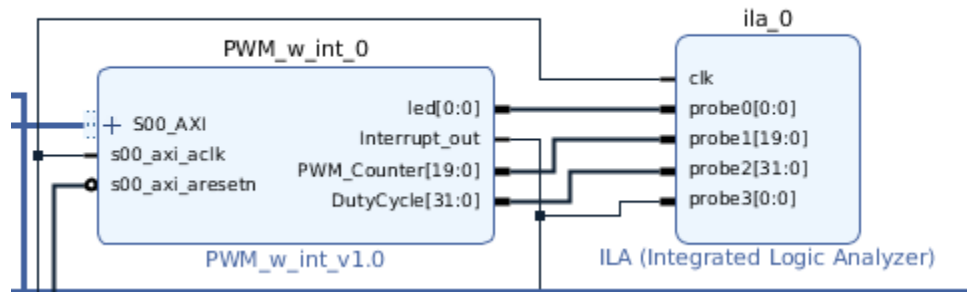


Figure 57 - Connect ILA and PWM

20. 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** if the address is not already assigned.

Diagram x Address Editor x					
<div> <div>Q</div> <div>≡</div> <div>⬇</div> <div>⬆</div> </div>					
Cell	Slave Interface	Base Name	Offset Address	Range	High Address
<div> <div>▼</div> <div>⚡</div> <div>processing_system7_0</div> </div>					
<div> <div>▼</div> <div>🗃</div> <div>Data (32 address bits : 0x40000000 [1G])</div> </div>					
<div> <div>▢</div> <div>axi_bram_ctrl_0</div> </div>	S_AXI	Mem0	0x4000_0000	8K ▼	0x4000_1FFF
<div> <div>▢</div> <div>PWM_w_int_0</div> </div>	S00_AXI	S00_AXI_reg	0x43C0_0000	64K ▼	0x43C0_FFFF

Figure 58 - 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.

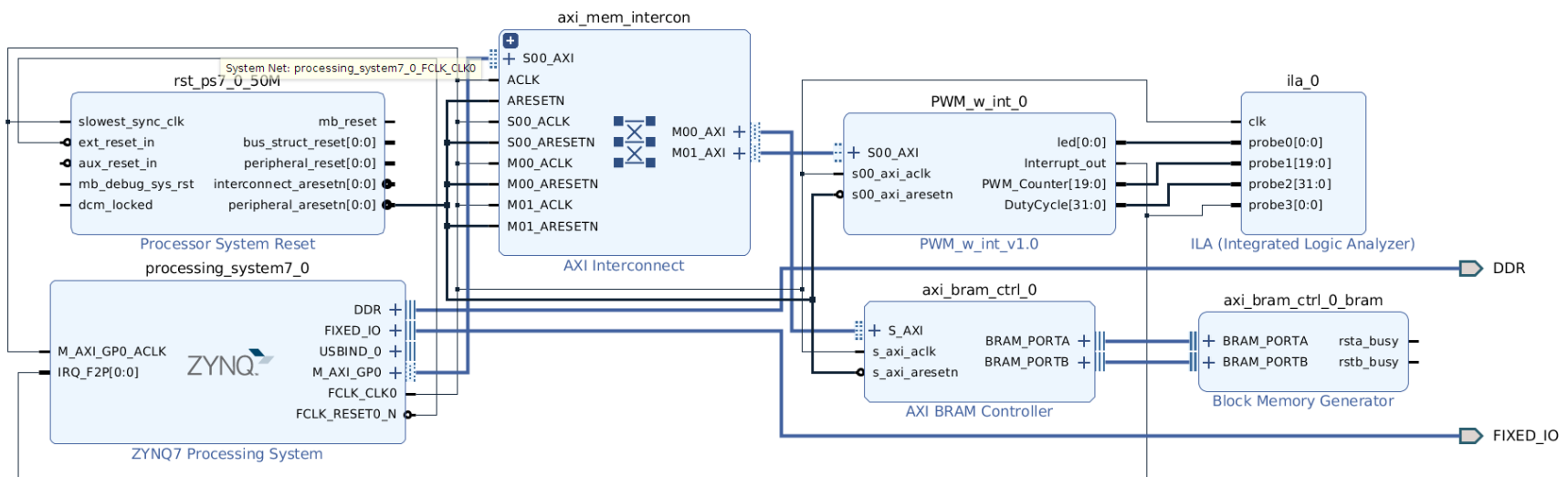


Figure 59 - Fully Connected Design

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:

Add JTAG to AXI Master IP into the project and connect it to the design.

Experiment 5 Step-by-Step Instructions:

1. In the block design, select **Add IP** then add the **JTAG to AXI Master** Core (search for JTAG).

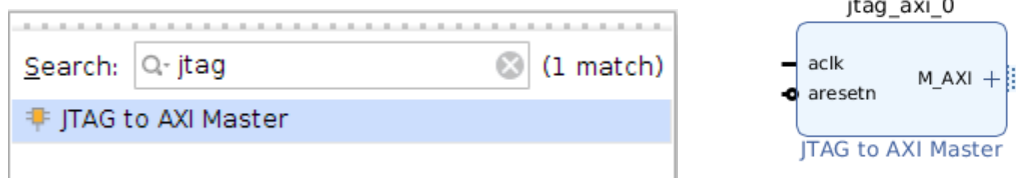


Figure 60 - Add JTAG to AXI Master

The JTAG to AXI Master needs a slave interface to connect into. Note this IP is an AXI Master, thus it will initiate AXI transactions with an AXI 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. Click **Run Connection Automation**. Make sure **All Automation** is selected, then click **OK**.

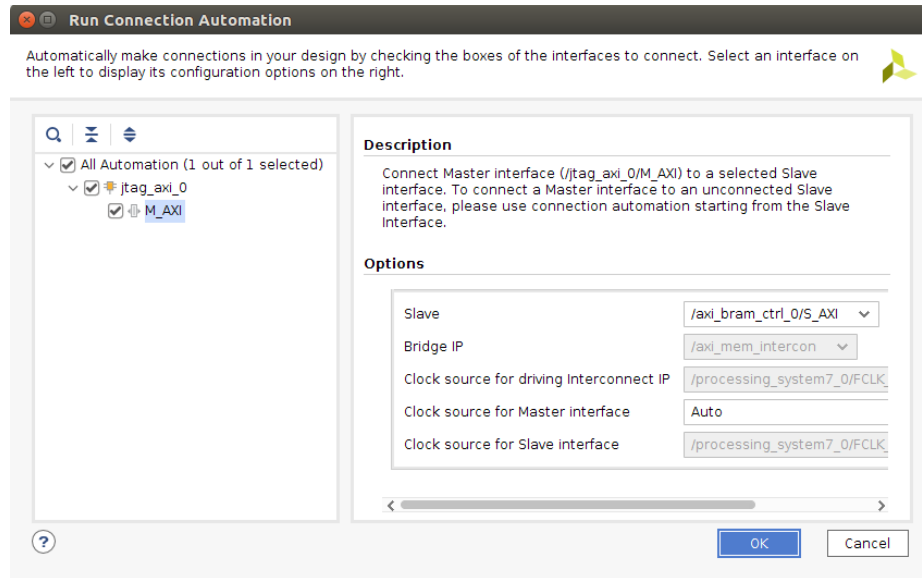


Figure 61 – Run Connection Automation on JTAG AXI

3. The JTAG to AXI Master is now correctly setup.

Experiment 6: Add LED Constraints

Experiment 6 General Instruction:

Add Led constraints to PWM_w_Int IP.

1. **Open** your Block Design.
2. **Right Click** on the **LED[0:0]** pin located on the PWM_w_Int_v1.0 IP. Then **Select Create Port**.

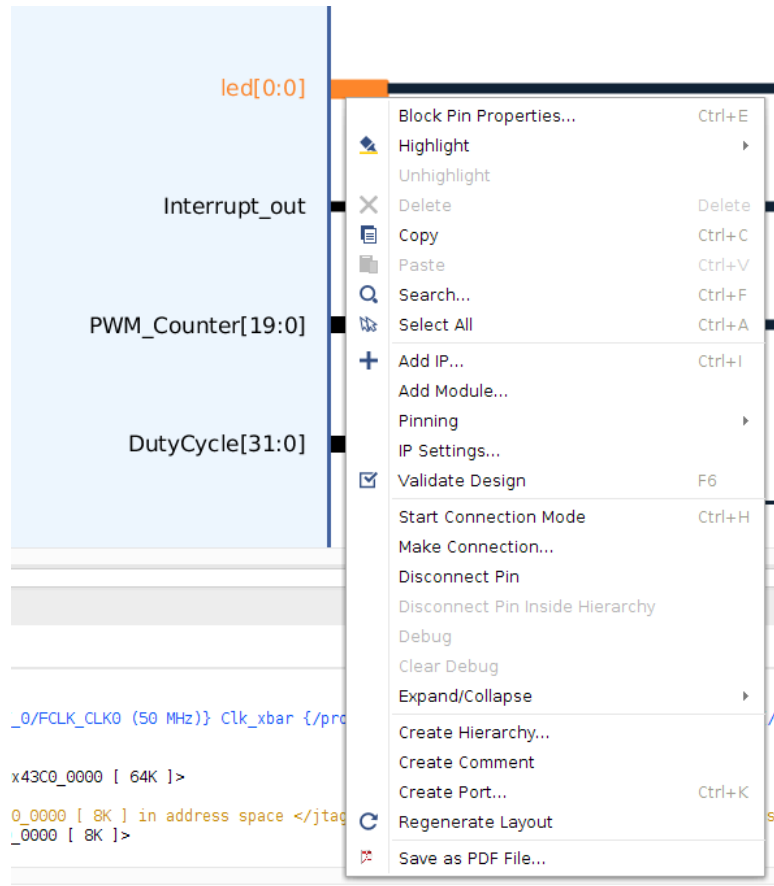


Figure 62 -- Create Port

3. You will now be prompted to customize your port. Change the Port name to **PL_LED_R** and then **set the Vector from [0 : 0]**. Select **OK**.

© 2019 Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

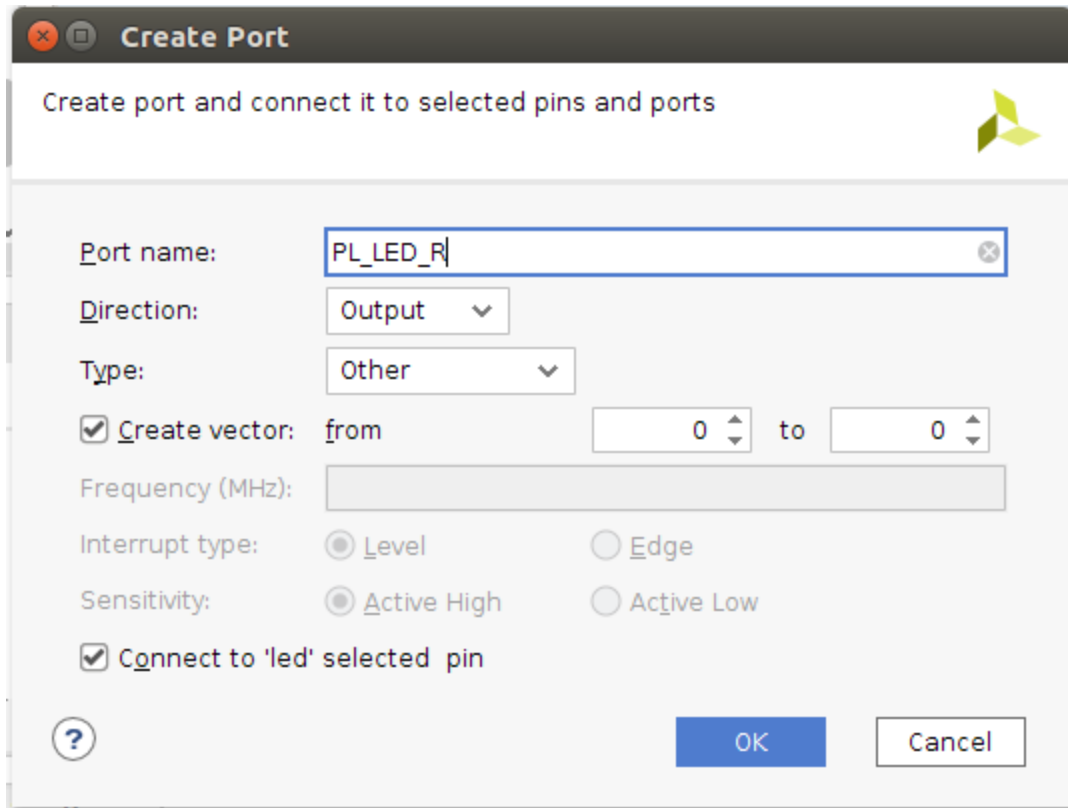


Figure 63 -- Port Configurations

4. Now that the Port has been generated and properly connect to the LED output.

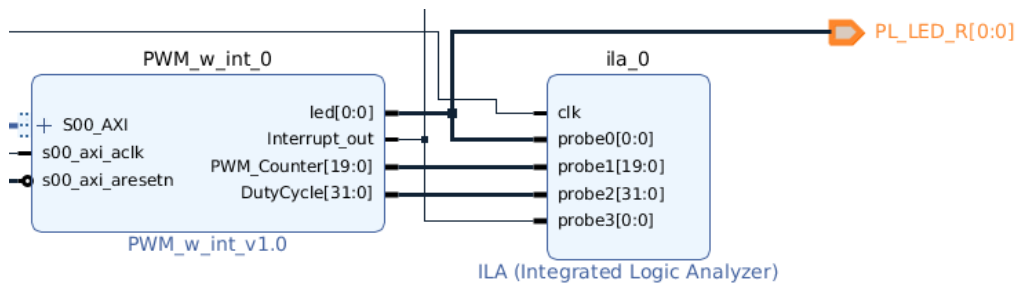


Figure 64 -- Connecting PL_LED_R Port

5. Now that we have the PL output port created, we must now add in constraints. Under Project Manager select **Add Sources**.

© 2019 Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

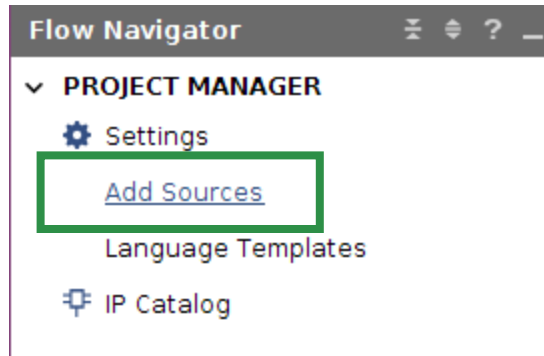


Figure 65 -- Add Sources

6. Make sure **Add or create constraints** is selected then click **Next>**.

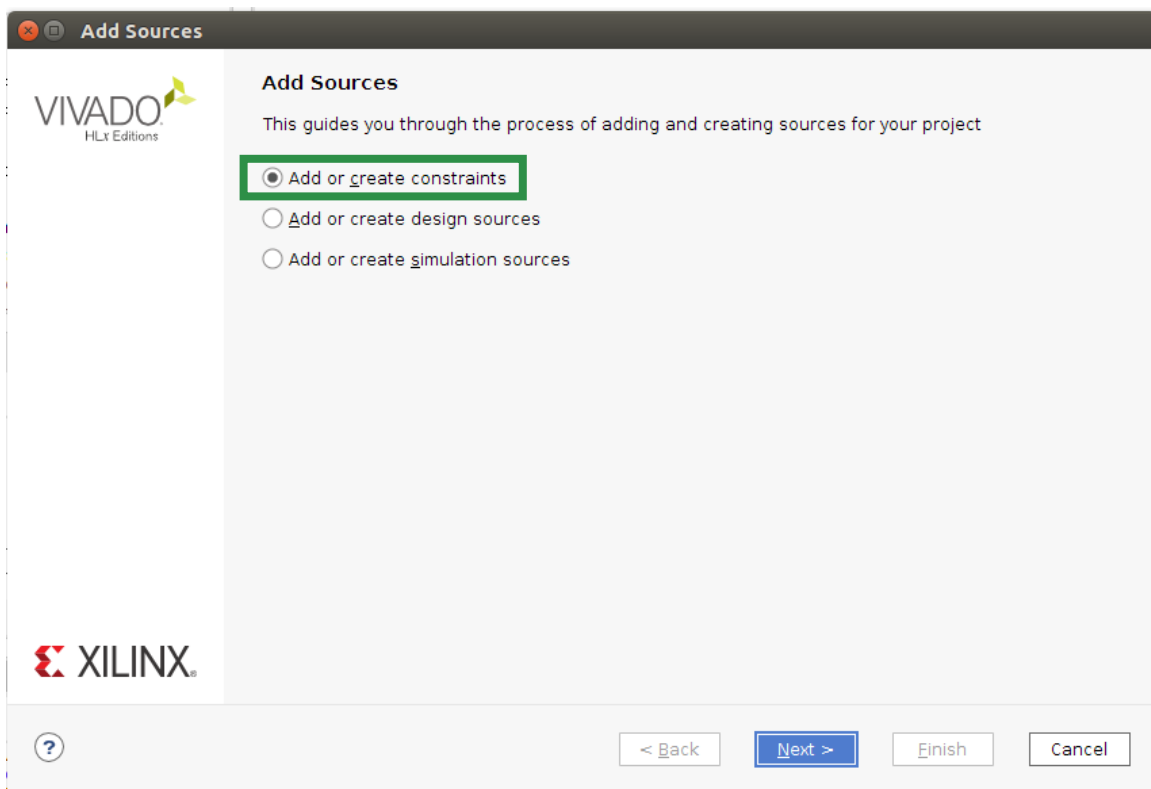


Figure 66 -- Add Constraints

7. Select Add Files and browse to **/home/training/AvnetTTC/ZynqHW/2019_1/Support_documents/Lab7** location and select the MiniZed_Speedway.xdc and then select **OK**.
8. You will now be back at the Add Sources page, make sure **copy constraints files into project is selected**, then **select Finish**.

© 2019 Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

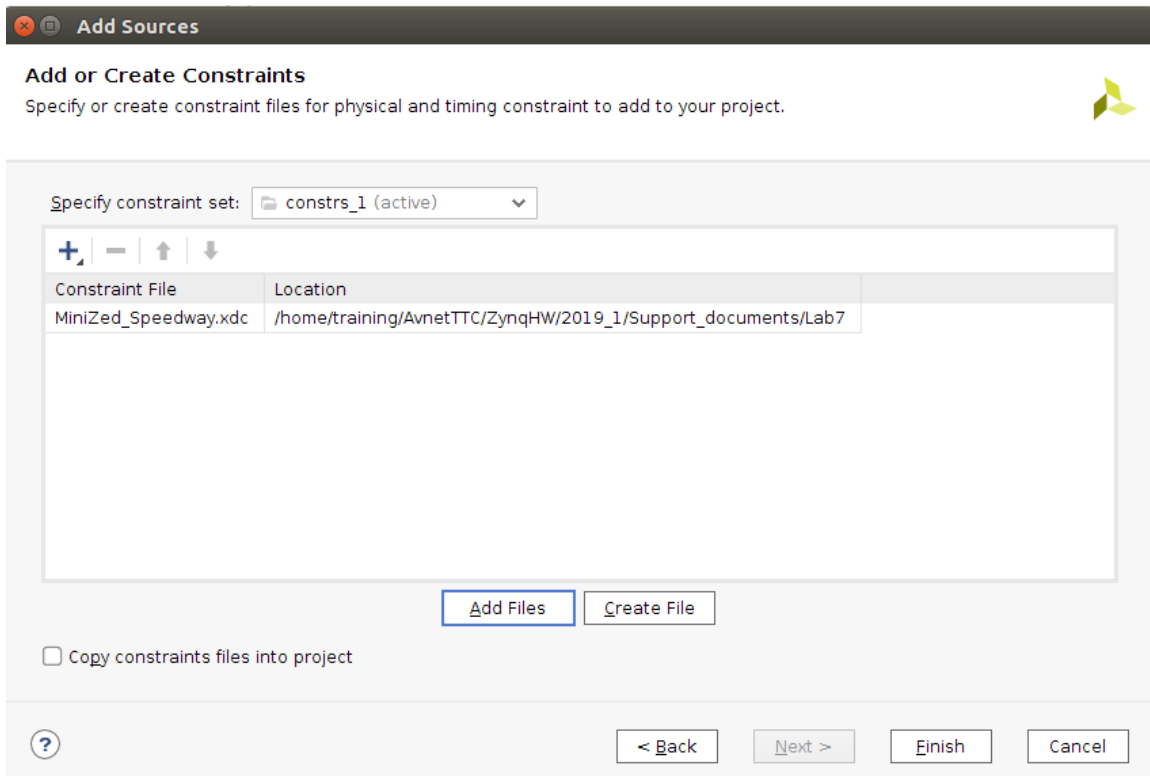


Figure 67 -- Add Constraints

9. **Validate** the block design. If OK, **Save** the block design.
10. Right-click on Z_system_i in the Sources tab and **Reset Output Products** then **Generate Output Products**.
11. In the Flow Navigator pane, click **Generate Bitstream**. This will take a few minutes depending on your PC. Also allow synthesis and implementation to run when prompted to.
12. Open implemented design when bitstream is successfully written.

© 2019 Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

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?*

This concludes Lab 7.

© 2019 Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

Revision History

Date	Version	Revision
6 Nov 13	02	Initial Draft
19 Nov 13	03	Pilot updates
6 Nov 14	04	Updated for Vivado 2014.3
5 Jan 15	05	Updated for Vivado 2014.4
06 Mar 15	06	Finalize for Vivado 2014.4. Update dma_test.c to use valid xil_types and also for BRAM xparameters.h bug in 2014.4
16 Mar 15	07	Minor edits for release
Oct 15	08	Updated to Vivado 2015.2
July 2016	09	Updated to Vivado 2016.2
May 2017	10	Updated to Vivado 2017.1
June 2017	11	Updated to Vivado 2017.1 for MiniZed + Rebranding
Nov 2017	12	Minor Update
Jan 2018	13	Updated to Vivado 2017.4
July 2019	14	Updated to Vivado 2019.1

Resources

www.minized.org

www.microzed.org

www.picozed.org

www.zedboard.org

www.xilinx.com/zynq

www.xilinx.com/sdk

www.xilinx.com/vivado

© 2019 Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

Answers

Experiment 1

- *Where is the IP project located?*

Just above the directory of our current project:

C:\Speedway\ZynqHW2017_4lip_repo

Experiment 5

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

16 slaves and 16 masters (Note: can support 64 masters when only 1 slave is enabled)

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

8 per M_AXI_GP = 16

Plus nFIQ & nIRQ per core = 4

20 Total