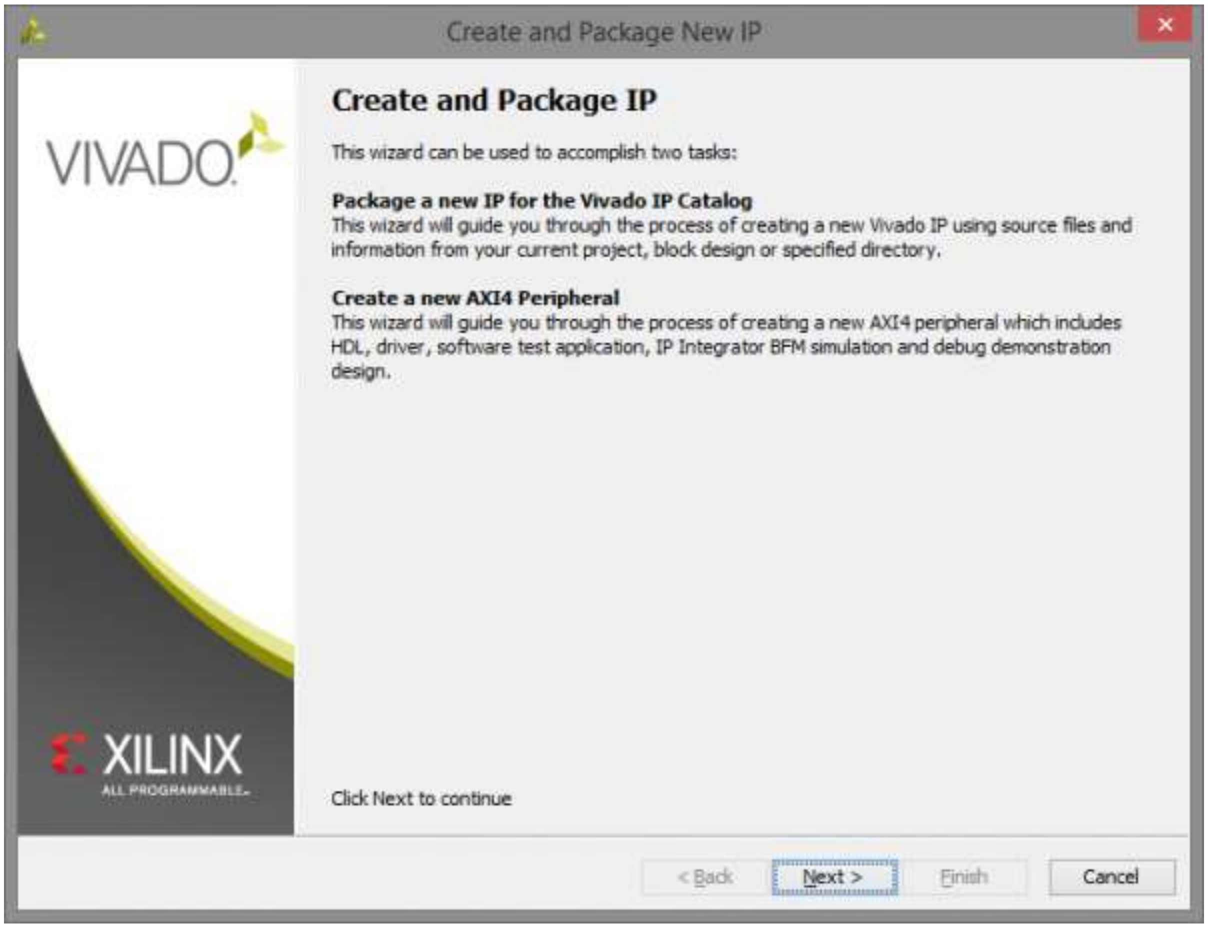
Custom Intellectual Property (IP) Tutorial

This tutorial will show how to build a basic PWM controller to manipulate on board LEDs of the Nexys4DDR board using the processing system of the xilinx microblaze soft core processor. You will also learn how to package the PWM controller as a custom IP that interfaces to the internal AXI bus used by the microblaze processor.   You will be able to change the PWM window size from the IP graphic interface and then control the duty cycle with a program written in C for the processor.

1. Open Vivado and create a new project

Open a new project as shown in previous labs
Go to **Tools→Create and package new IP and click Next**



Create and Package New IP

Create and Package IP

This wizard can be used to accomplish two tasks:

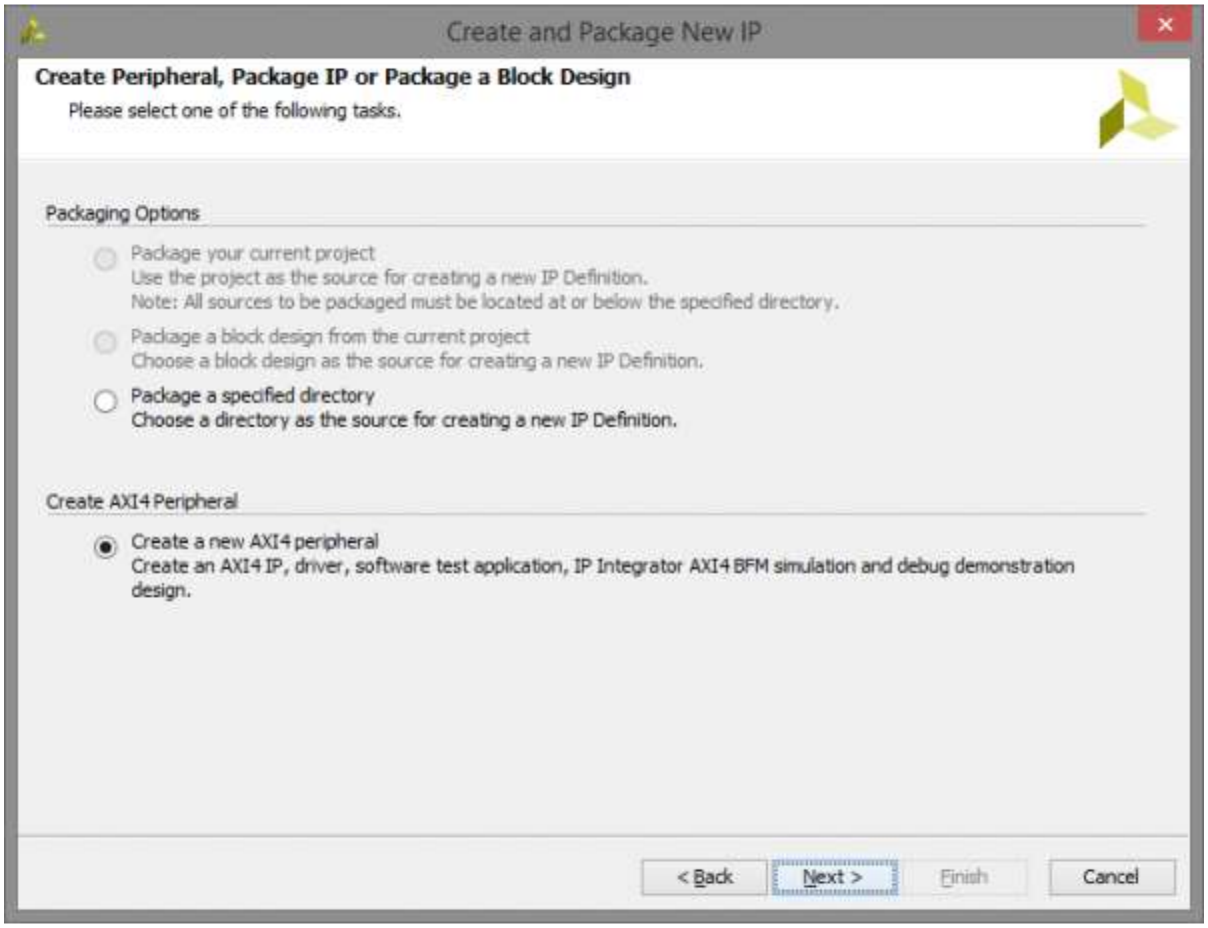
Package a new IP for the Vivado IP Catalog
This wizard will guide you through the process of creating a new Vivado IP using source files and information from your current project, block design or specified directory.

Create a new AXI4 Peripheral
This wizard will guide you through the process of creating a new AXI4 peripheral which includes HDL, driver, software test application, IP Integrator BFM simulation and debug demonstration design.

Click Next to continue

2. Create your custom IP project

2.1) Select **Create a new AXI4 peripheral** and click **Next**



Create and Package New IP

Create Peripheral, Package IP or Package a Block Design

Please select one of the following tasks.

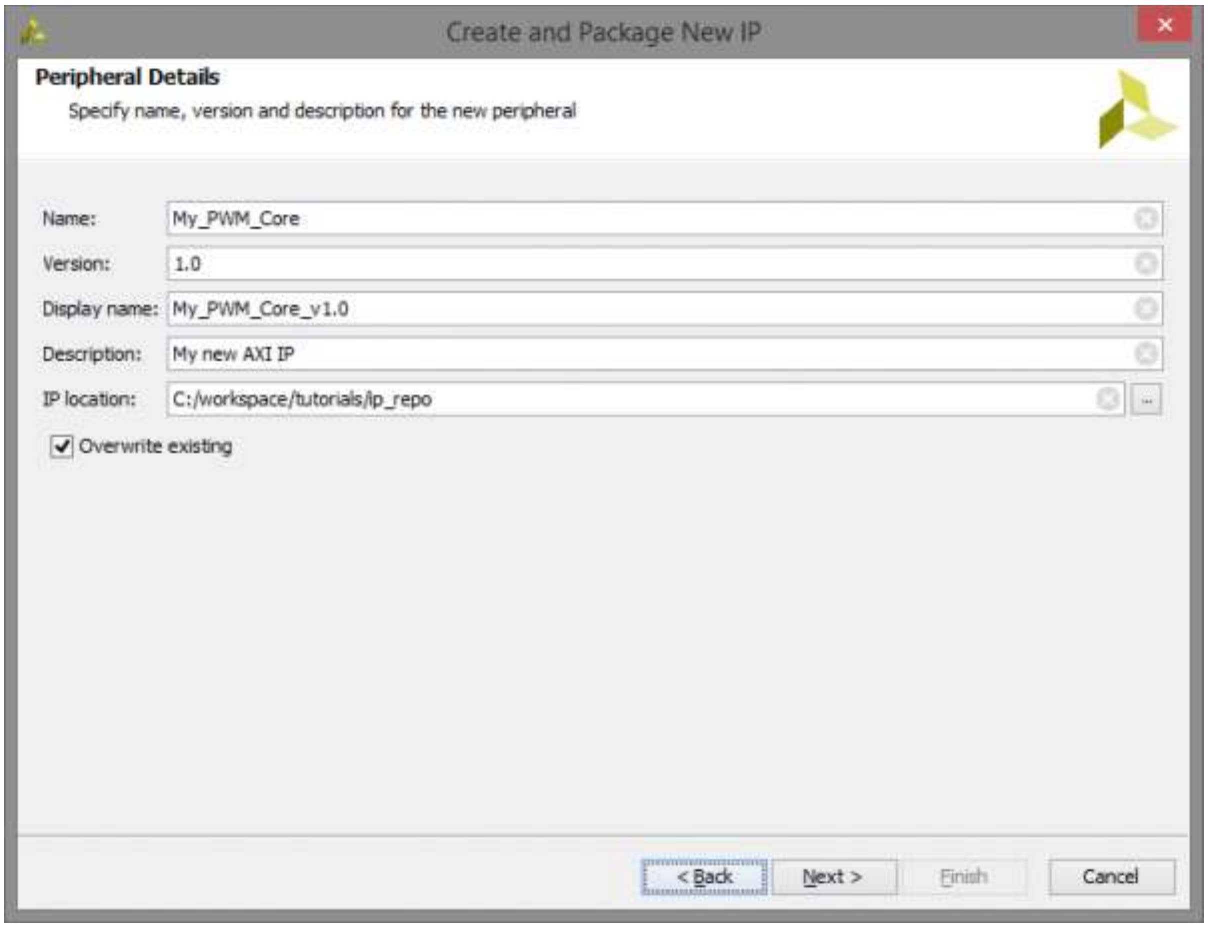
Packaging Options

- ○ Package your current project
  Use the project as the source for creating a new IP Definition.
  Note: All sources to be packaged must be located at or below the specified directory.
- ○ Package a block design from the current project
  Choose a block design as the source for creating a new IP Definition.
- ○ 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 BFM simulation and debug demonstration design.

2.2) Input “My_PWM_Core” in the name field and check Overwrite existing and click **Next**



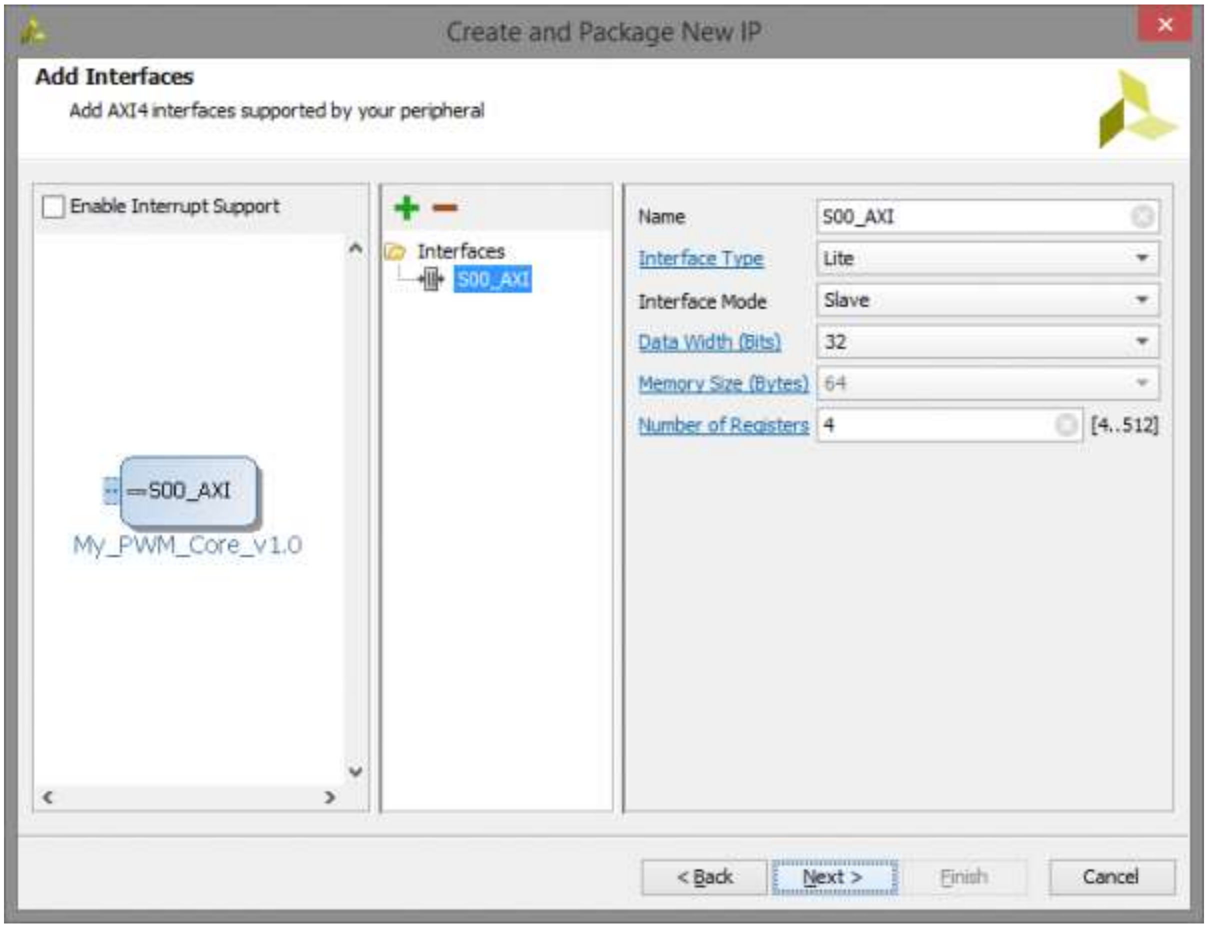
Peripheral Details

Specify name, version and description for the new peripheral

Name:	My_PWM_Core
Version:	1.0
Display name:	My_PWM_Core_v1.0
Description:	My new AXI IP
IP location:	C:/workspace/tutorials/ip_repo

[✓] Overwrite existing

2.3) No changes to the AXI interface are required, so click **Next**


Create and Package New IP

Add Interfaces

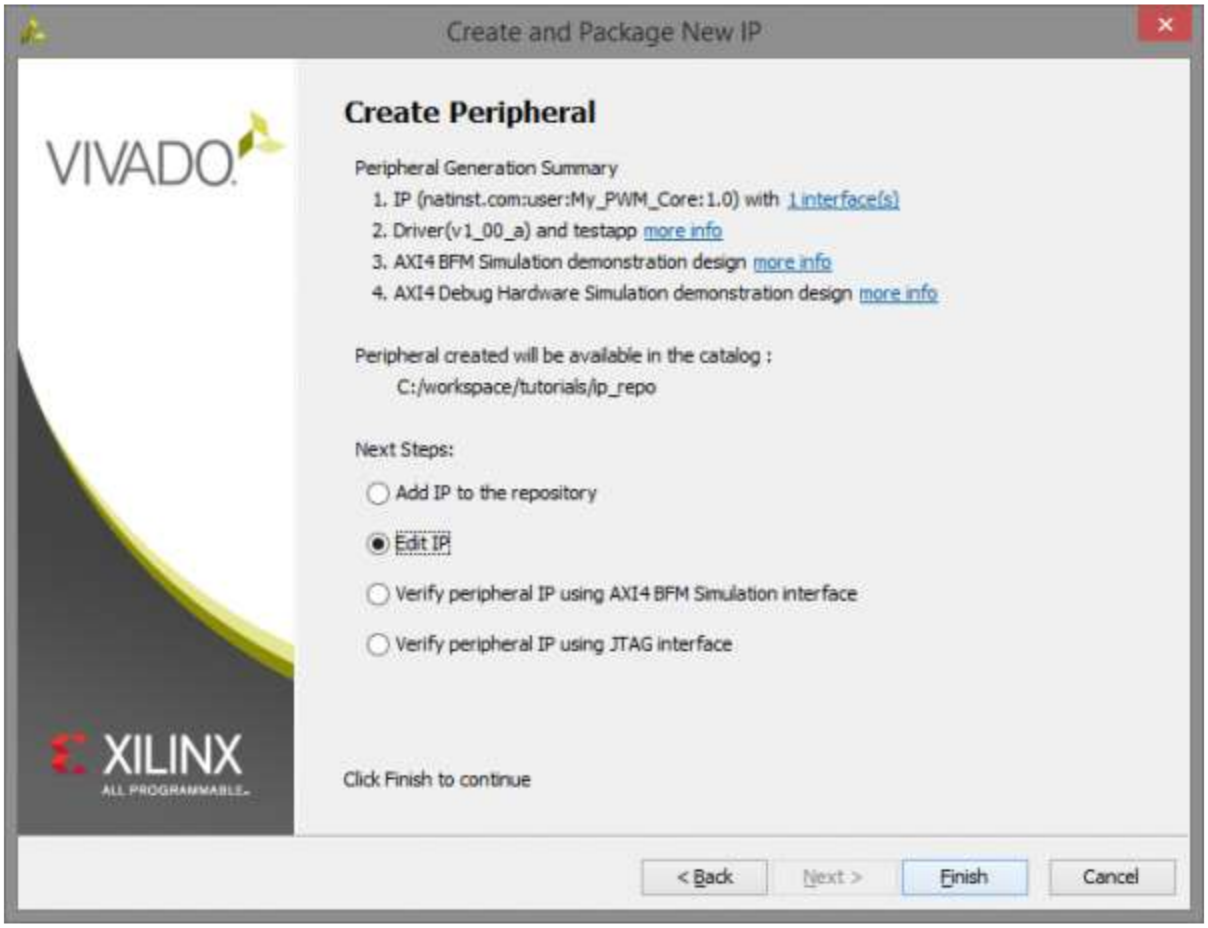
Add AXI4 interfaces supported by your peripheral

Enable Interrupt Support

Interfaces
S00_AXI

Name	S00_AXI
Interface Type	Lite
Interface Mode	Slave
Data Width (Bits)	32
Memory Size (Bytes)	64
Number of Registers	4 [4..512]

2.4) Select **Edit IP** and click **Finish**


Create and Package New IP

Create Peripheral

Peripheral Generation Summary
1. IP (natinst.com:user:My_PWM_Core:1.0) with 1 interface(s)
2. Driver(v1_00_a) and testapp more info
3. AXI4 BFM Simulation demonstration design more info
4. AXI4 Debug Hardware Simulation demonstration design more info

Peripheral created will be available in the catalog :
C:/workspace/tutorials/ip_repo

Next Steps:
- Add IP to the repository
- [✓ Edit IP]
- Verify peripheral IP using AXI4 BFM Simulation interface
- Verify peripheral IP using JTAG interface

Click Finish to continue

3. Designing the IP core

3.1) A new instance of Vivado will open up for the new IP core. Expand the top level file **My_PWM_Core_v1_0**. Then double-click on **My_PWM_Core_v1_0_S00_AXI** to open it in the editor.

4. Modifying My_PWM_Core_v1_0_S00_AXI

4.1)To allow for the user to change the maximum value of the PWM window in the top level, change this:

```
    // Users to add parameters here


    // User parameters ends

    // Do not modify the parameters beyond this line
```

To this:

```
    // Users to add parameters here
```

```
    parameter integer PWM_COUNTER_MAX = 1024,
```

```
    // User parameters ends
```

```
    // Do not modify the parameters beyond this line
```

4.2) To give the PWM signals a port out of the custom IP core, change this:

```
    // Users to add ports here
```

```

```

```
    // User ports ends
```

```
    // Do not modify the ports beyond this line
```

To this:

```
    // Users to add ports here
```

```
    output wire PWM0,
```

```
    output wire PWM1,
```

```
    output wire PWM2,
```

```
    output wire PWM3,
```

```
    // User ports ends
```

```
    // Do not modify the ports beyond this line
```

4.3) The following modification creates a counter that is 16-bits wide with a maximum of (2^16)-1, which will be more than enough for most applications. Scroll to the bottom of the file and change this:

```
// Add user logic here

// User logic ends
```

To this:

```
// Add user logic here
reg [15:0] counter = 0;

//simple counter
always @(posedge S_AXI_ACLK) begin
    if(counter < PWM_COUNTER_MAX-1)
        counter <= counter + 1;
    else
        counter <= 0;
end

//comparator statements that drive the PWM signal
assign PWM0 = slv_reg0 < counter ? 1'b0 : 1'b1;
assign PWM1 = slv_reg1 < counter ? 1'b0 : 1'b1;
```

```
    assign PWM2 = slv_reg2 < counter ? 1'b0 : 1'b1;

    assign PWM3 = slv_reg3 < counter ? 1'b0 : 1'b1;


    // User logic ends
```

Overall this module will write data into the slave registers from the processor. The simple counter will count till the max value and reset forever. Then each of the comparator statements checks if the current counter value is greater than the value stored in the slave registers and sets PWM high if the compare value is less than the current counter.

What we have accomplished:
– Parameterized the PWM window size with PWM_COUNTER_MAX
– Added ports so the higher level file can get the PWM signals
– Added a simple counter that counts from zero to PWM_COUNTER_MAX-1
– Added four asynchronous comparator signals that create the PWM signal

5. Modifying My_PWM_Core_v1_0

5.1) Double-click on **My_PWM_Core_v1_0** to open it in the editor.

The following modifications add the ports for the PWM signals and the parameter up to the top HDL file. This will allow the GUI to change, connect, and modify the IP core.

5.2) Change this:

```
module My_PWM_Core_v1_0 #

(

    // Users to add parameters here


    // User parameters ends

    // Do not modify the parameters beyond this line
```

To this:

```
module My_PWM_Core_v1_0 #
```

```
(

    // Users to add parameters here

    parameter integer PWM_COUNTER_MAX = 128,



    // User parameters ends

    // Do not modify the parameters beyond this line
```

5.3) Then change this:

```
    // Users to add ports here



    // User ports ends
```

To this:

```
    // Users to add ports here

    output wire PWM0,

    output wire PWM1,

    output wire PWM2,

    output wire PWM3,

    // User ports ends
```

5.4) And this:

```
    // Instantiation of Axi Bus Interface S00_AXI
    My_PWM_Core_v1_0_S00_AXI # (

        .C_S_AXI_DATA_WIDTH(C_S00_AXI_DATA_WIDTH),

        .C_S_AXI_ADDR_WIDTH(C_S00_AXI_ADDR_WIDTH)

    ) My_PWM_Core_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_WDATA(s00_axi_wdata),

        .S_AXI_WSTRB(s00_axi_wstrb),

        .S_AXI_WVALID(s00_axi_wvalid),

        .S_AXI_WREADY(s00_axi_wready),

        .S_AXI_BRESP(s00_axi_bresp),

        .S_AXI_BVALID(s00_axi_bvalid),

        .S_AXI_BREADY(s00_axi_bready),

        .S_AXI_ARADDR(s00_axi_araddr),
        .S_AXI_ARPROT(s00_axi_arprot),
        .S_AXI_ARVALID(s00_axi_arvalid),
        .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)
    );
```

To this:

```
    // Instantiation of Axi Bus Interface S00_AXI
    My_PWM_Core_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_COUNTER_MAX(PWM_COUNTER_MAX)
    ) My_PWM_Core_v1_0_S00_AXI_inst (
        .PWM0(PWM0),
        .PWM1(PWM1),
        .PWM2(PWM2),
        .PWM3(PWM3),

        .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_WDATA(s00_axi_wdata),

        .S_AXI_WSTRB(s00_axi_wstrb),

        .S_AXI_WVALID(s00_axi_wvalid),

        .S_AXI_WREADY(s00_axi_wready),

        .S_AXI_BRESP(s00_axi_bresp),

        .S_AXI_BVALID(s00_axi_bvalid),

        .S_AXI_BREADY(s00_axi_bready),

        .S_AXI_ARADDR(s00_axi_araddr),

        .S_AXI_ARPROT(s00_axi_arprot),

        .S_AXI_ARVALID(s00_axi_arvalid),

        .S_AXI_ARREADY(s00_axi_arready),
        .S_AXI_RDATA(s00_axi_rdata),
```

```
        .S_AXI_RRESP(s00_axi_rresp),
```

```
        .S_AXI_RVALID(s00_axi_rvalid),
```

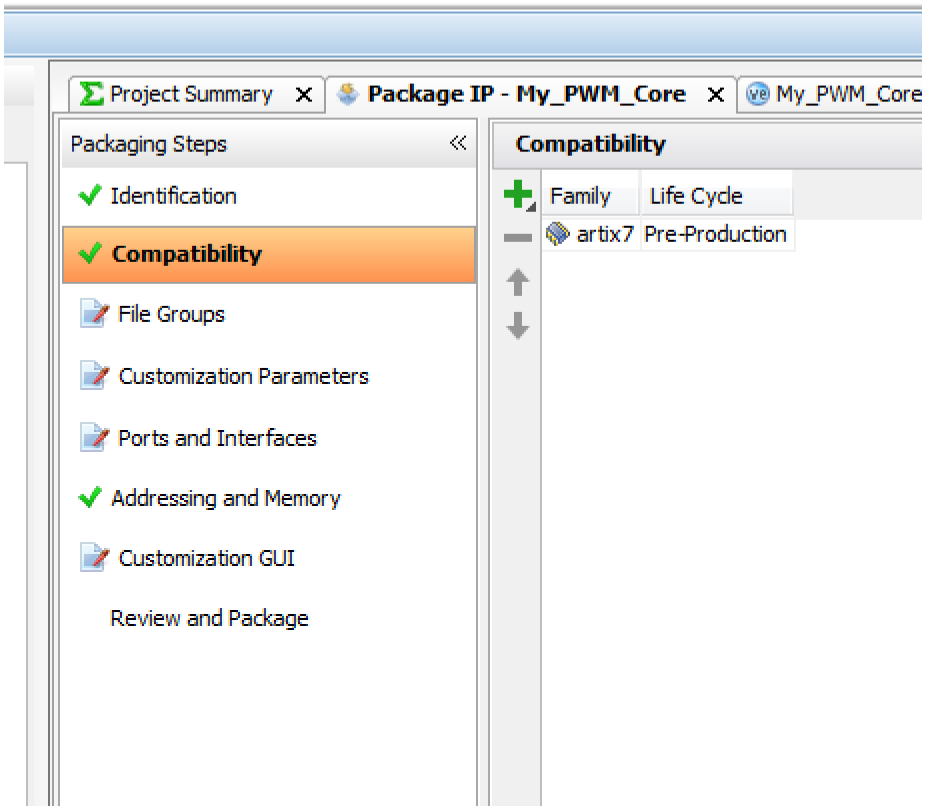
```
        .S_AXI_RREADY(s00_axi_rready)
```

```
    );
```

6. Packaging the IP core

Now that we have written the core, it is time to package up the HDL to create a complete IP package.

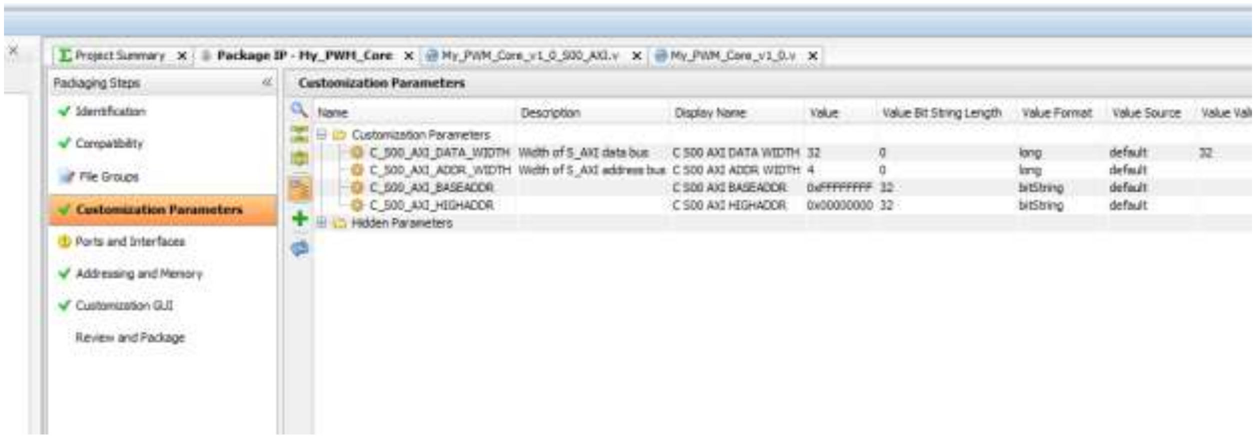
6.1) Now click on **Package IP** in the Flow Navigator and you should see the Package IP tab. Select **Compatibility** and make sure "Artix7" is present. If it is not there, you can add it by clicking the plus button. The Life Cycle does not matter at this point.



Family	Life Cycle
artix7	Pre-Production

6.2) Select **Customization Parameters** and select the line for **Merge Changes from Customization Parameters Wizard**. This will add the PWM_COUNTER_MAX parameter from the top file under the hidden parameters folder. You can ignore the warning sign on the Ports and

Interfaces.

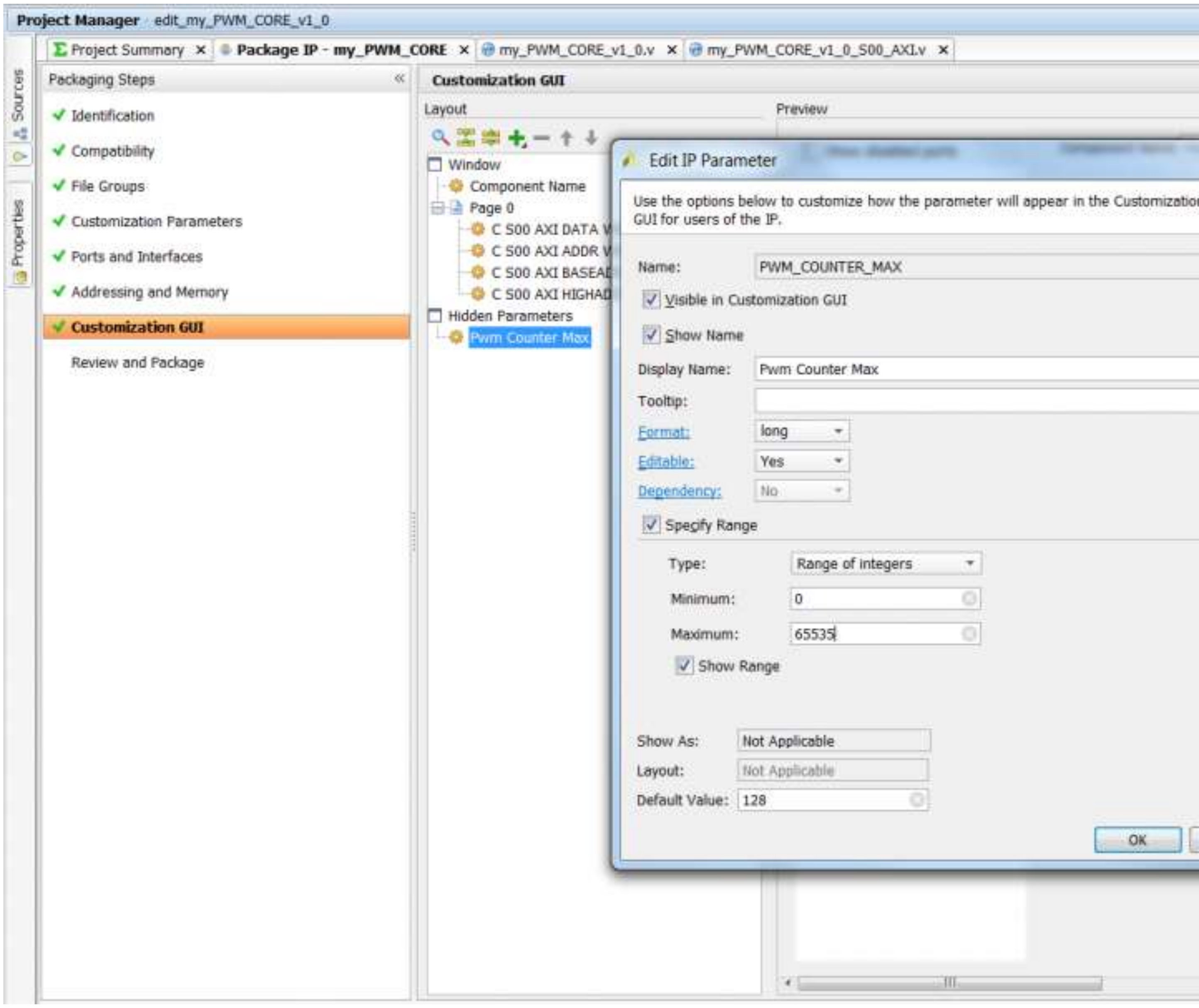


Packaging Steps: Identification, Compatibility, File Groups, Customization Parameters, Ports and Interfaces, Addressing and Memory, Customization GUI, Review and Package

Customization Parameters

Name	Description	Display Name	Value	Value Bit String Length	Value Format	Value Source	Value Val
Customization Parameters							
C_S00_AXI_DATA_WIDTH	Width of S_AXI data bus	C S00 AXI DATA WIDTH	32	0	long	default	32
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	
Hidden Parameters							

6.3) Select **Customization GUI**. This is where we get to change our graphical interface. One problem, there is not a window for our parameter. Right-click **Pwm Counter Max** and select **Edit Parameter…**. Check the box next to **Visible in Customization GUI**. Check the **Specify Range** box. Select **Range of Integers** from the **Type** drop-down menu. Set the max value to (2^16)-1 = 65535 and the min value of 0. Click **Ok**.



Edit IP Parameter

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

Name: PWM_COUNTER_MAX

[✓] Visible in Customization GUI

[✓] Show Name

Display Name: Pwm Counter Max

Tooltip:

Format: long

Editable: Yes

Dependency: No

[✓] Specify Range

Type: Range of integers

Minimum: 0

Maximum: 65535

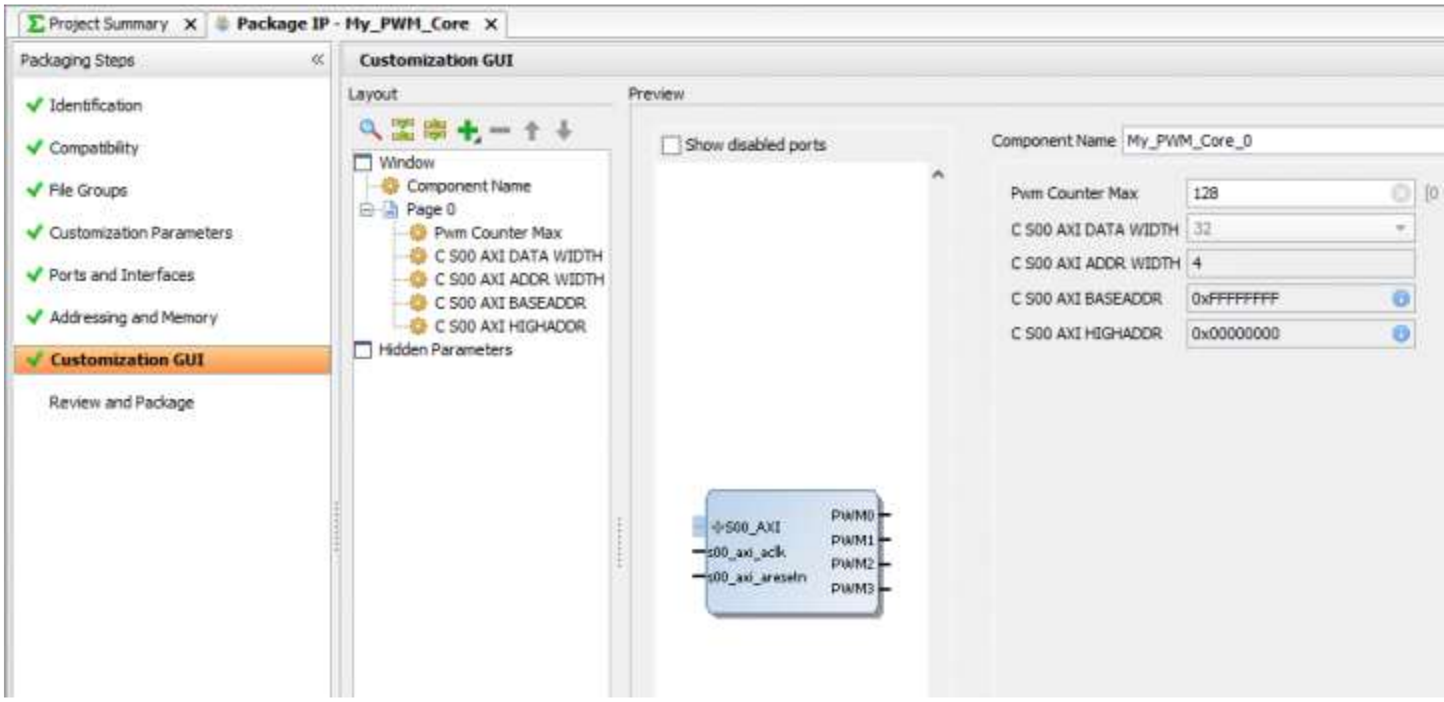
[✓] Show Range

Show As: Not Applicable

Layout: Not Applicable

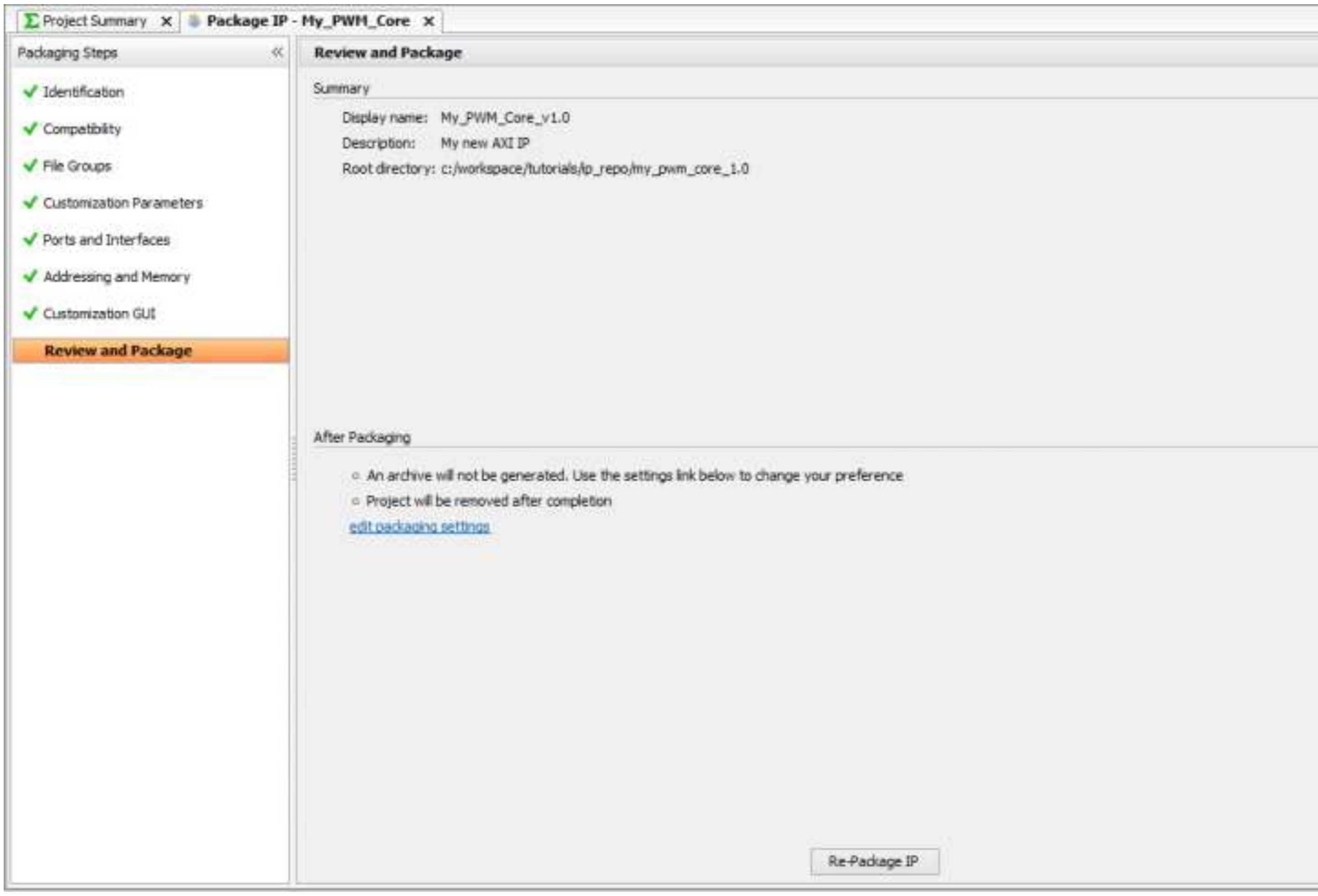
Default Value: 128

6.4) Drag the **Pwm Counter Max (just above the hidden parameters section of the layout)** into **Page 0** to get it in the main page.



Component Name	My_PWM_Core_0
Pwm Counter Max	128
C S00 AXI DATA WIDTH	32
C S00 AXI ADDR WIDTH	4
C S00 AXI BASEADDR	0xFFFFFFFF
C S00 AXI HIGHADDR	0x00000000

6.5) Now the core should be complete so select **Review and Package (you can ignore the 1 warning and 2 info messages)** and click the **Re-package IP** button.



Review and Package

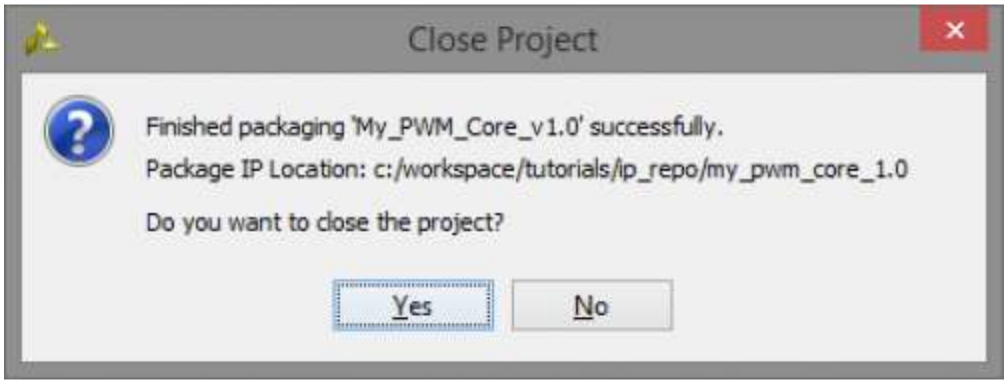
Summary

Display name: My_PWM_Core_v1.0
Description: My new AXI IP
Root directory: c:/workspace/tutorials/ip_repo/my_pwm_core_1.0

After Packaging

- An archive will not be generated. Use the settings link below to change your preference
- Project will be removed after completion

6.6) A popup will ask if you want to close the project, Select **Yes**.



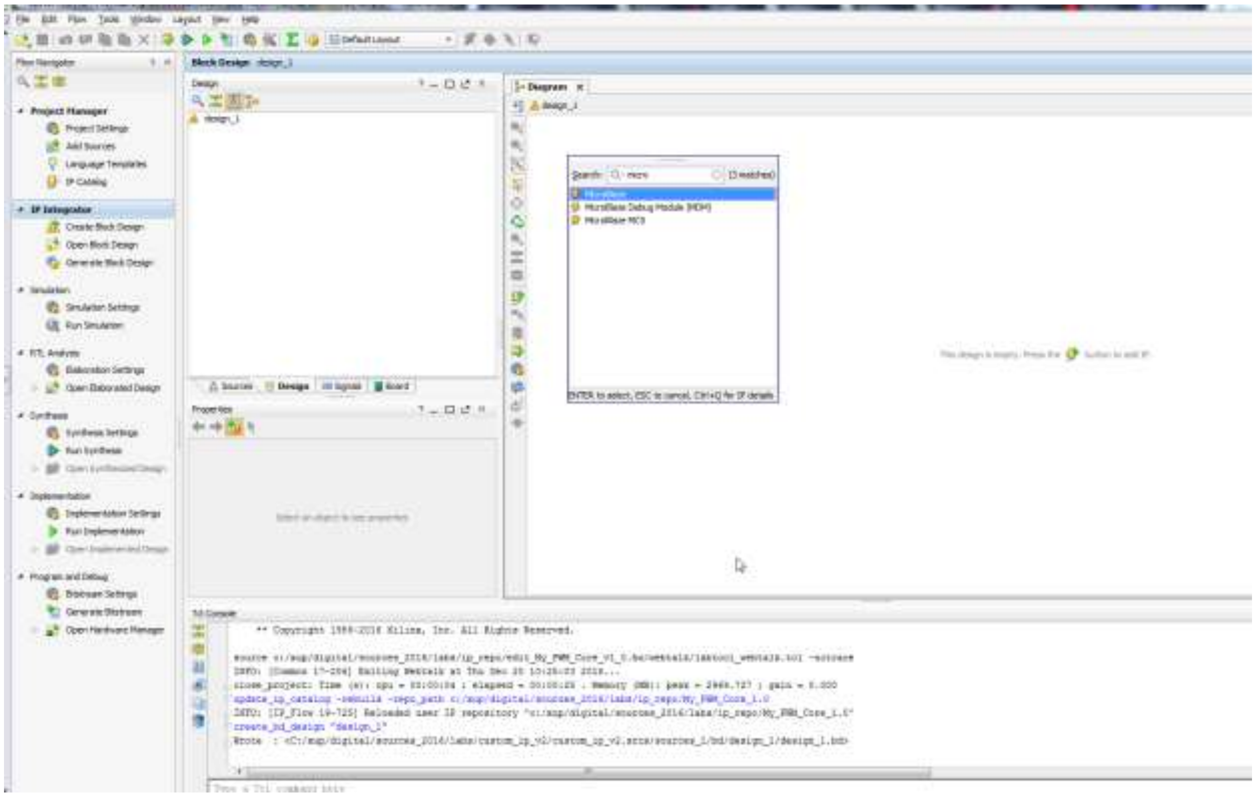
Finished packaging 'My_PWM_Core_v1.0' successfully.
Package IP Location: c:/workspace/tutorials/ip_repo/my_pwm_core_1.0
Do you want to close the project?

---

7. Create Microblaze design

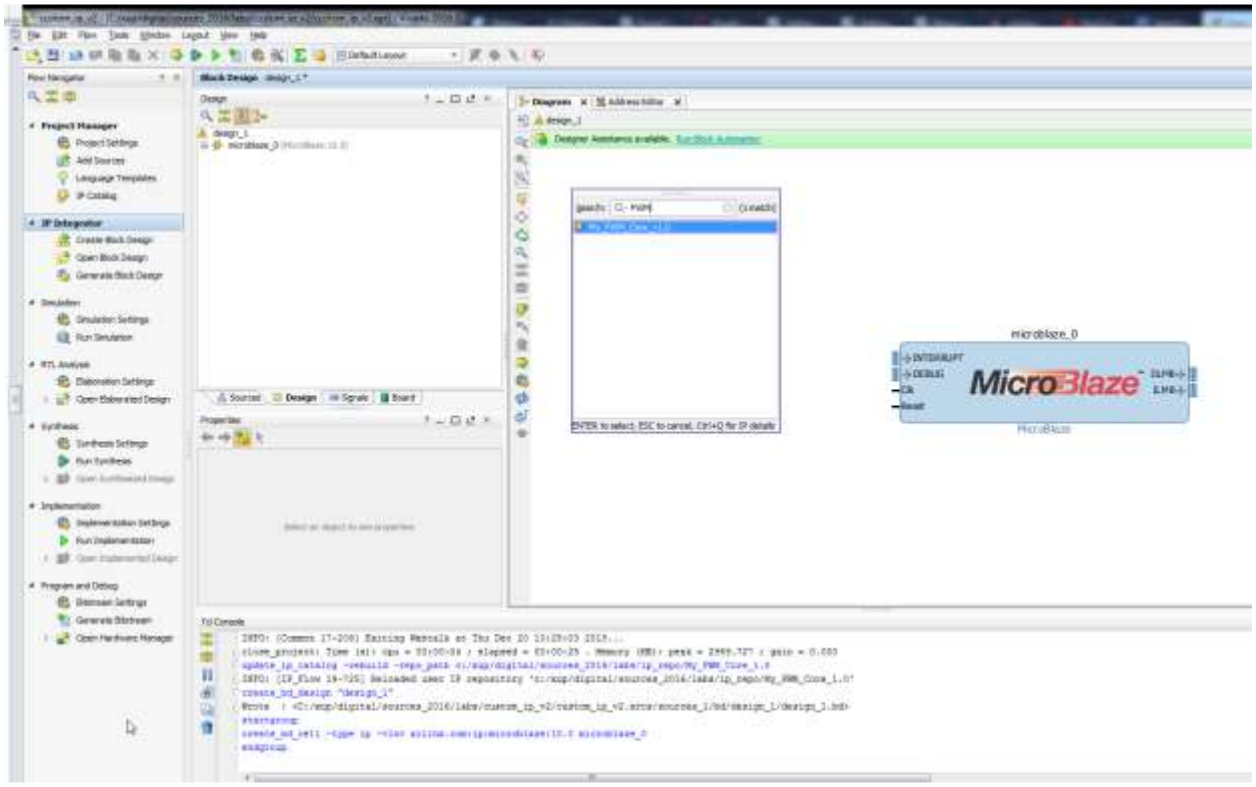
7.1) In the project manager page of the original window select Design Sources and click **Create Block Design**. This adds a block design to the project. You can leave the default design_1 and click OK.

7.2) Use the **Add IP** button to add the Microblaze **Processing System**. Search for microblaze and then hit enter.



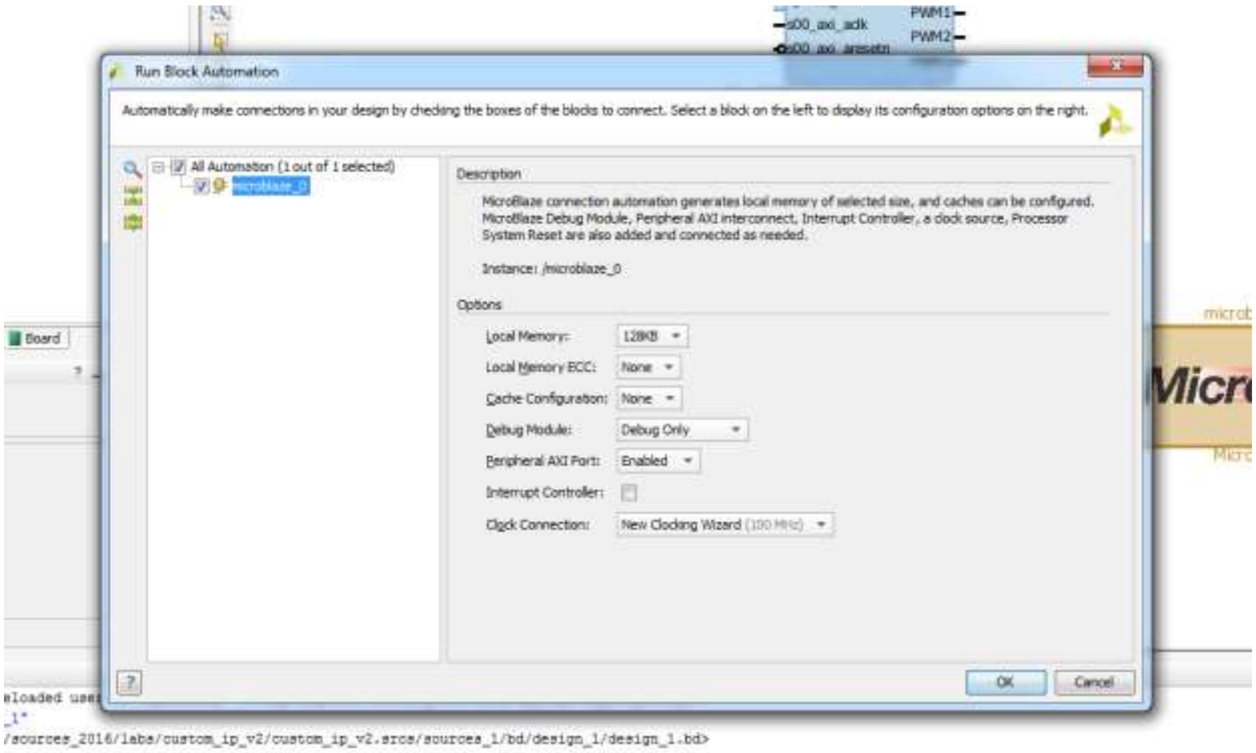
```
    ** Copyright 1986-2016 Xilinx, Inc. All Rights Reserved.

source c:/mop/digital/sources_2016/labs/ip_repo/edit_My_PWM_Core_v1_0.hw/webtalk/labtool_webtalk.tcl -notrace
INFO: [Common 17-206] Exiting Webtalk at Thu Dec 20 10:25:23 2018...
close_project: Time (s): cpu = 00:00:04 ; elapsed = 00:00:25 . Memory (MB): peak = 2966.727 ; gain = 0.000
update_ip_catalog -rebuild -repo_path c:/mop/digital/sources_2016/labs/ip_repo/My_PWM_Core_1.0
INFO: [IP_Flow 19-725] Reloaded user IP repository 'c:/mop/digital/sources_2016/labs/ip_repo/My_PWM_Core_1.0'
create_bd_design "design_1"
Wrote  : <C:/mop/digital/sources_2016/labs/custom_ip_v2/custom_ip_v2.srcs/sources_1/bd/design_1/design_1.bd>
```

7.3) Use the **Add IP** button again to add our **PWM core**.

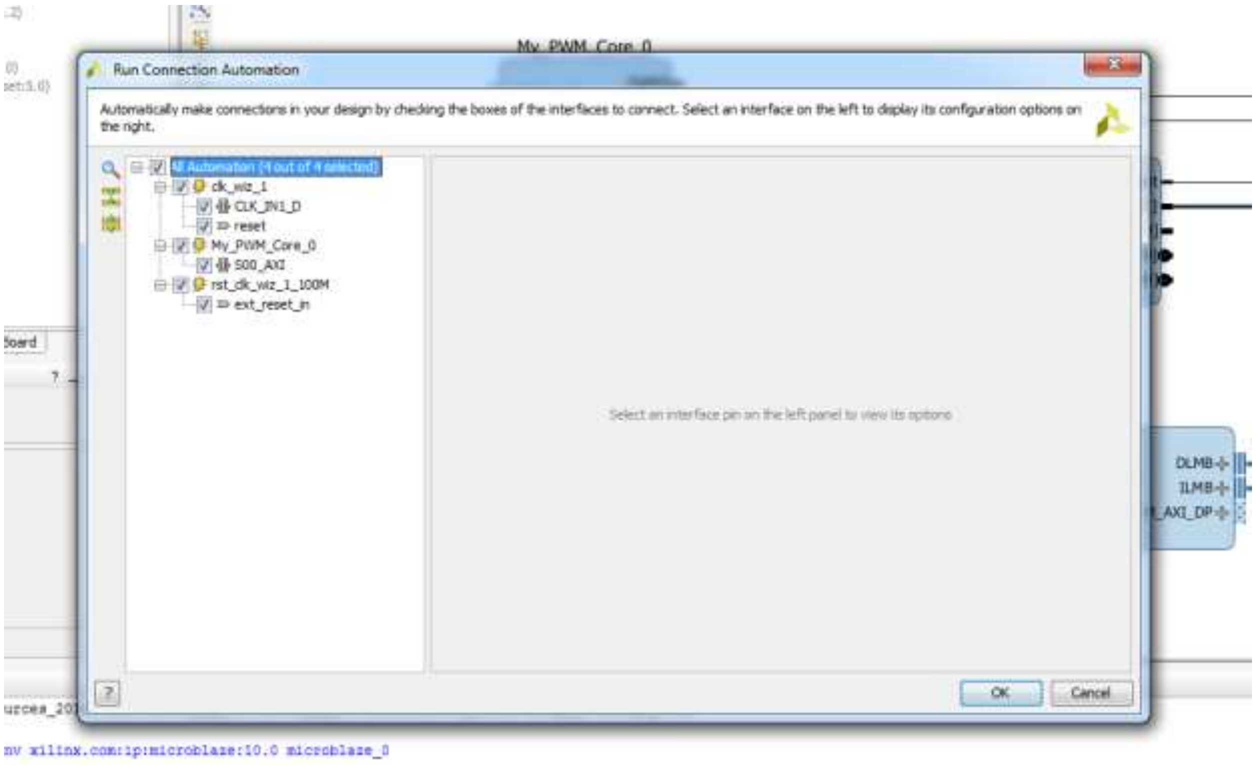


```
INFO: [Common 17-206] Exiting Webtalk at Thu Dec 20 13:23:03 2018...
close_project: Time (s): cpu = 00:00:08 ; elapsed = 00:00:25 . Memory (MB): peak = 2989.727 ; gain = 0.000
update_ip_catalog -rebuild -repo_path c:/mpp/digital/sources_2016/labs/ip_repo/My_PWM_Core_1.0
INFO: [IP_Flow 19-725] Reloaded user IP repository 'c:/mpp/digital/sources_2016/labs/ip_repo/My_PWM_Core_1.0'
create_bd_design "design_1"
Wrote  : <C:/mpp/digital/sources_2016/labs/custom_ip_v2/custom_ip_v2.srcs/sources_1/bd/design_1/design_1.bd>
startgroup
create_bd_cell -type ip -vlnv xilinx.com:ip:microblaze:10.0 microblaze_0
endgroup
```

7.4) **Run Block Automation**. For local memory select 128KB and leave all other option as their defaults and click OK.


Run Block Automation
Automatically make connections in your design by checking the boxes of the blocks to connect. Select a block on the left to display its configuration options on the right.
All Automation (1 out of 1 selected)
microblaze_0
Description
MicroBlaze connection automation generates local memory of selected size, and caches can be configured. MicroBlaze Debug Module, Peripheral AXI interconnect, Interrupt Controller, a clock source, Processor System Reset are also added and connected as needed.
Instance: /microblaze_0
Options
Local Memory: 128KB
Local Memory ECC: None
Cache Configuration: None
Debug Module: Debug Only
Peripheral AXI Port: Enabled
Interrupt Controller:
Clock Connection: New Clocking Wizard (100 MHz)
OK
Cancel

7.5) Run Connection Automation and select All Automation and click OK.



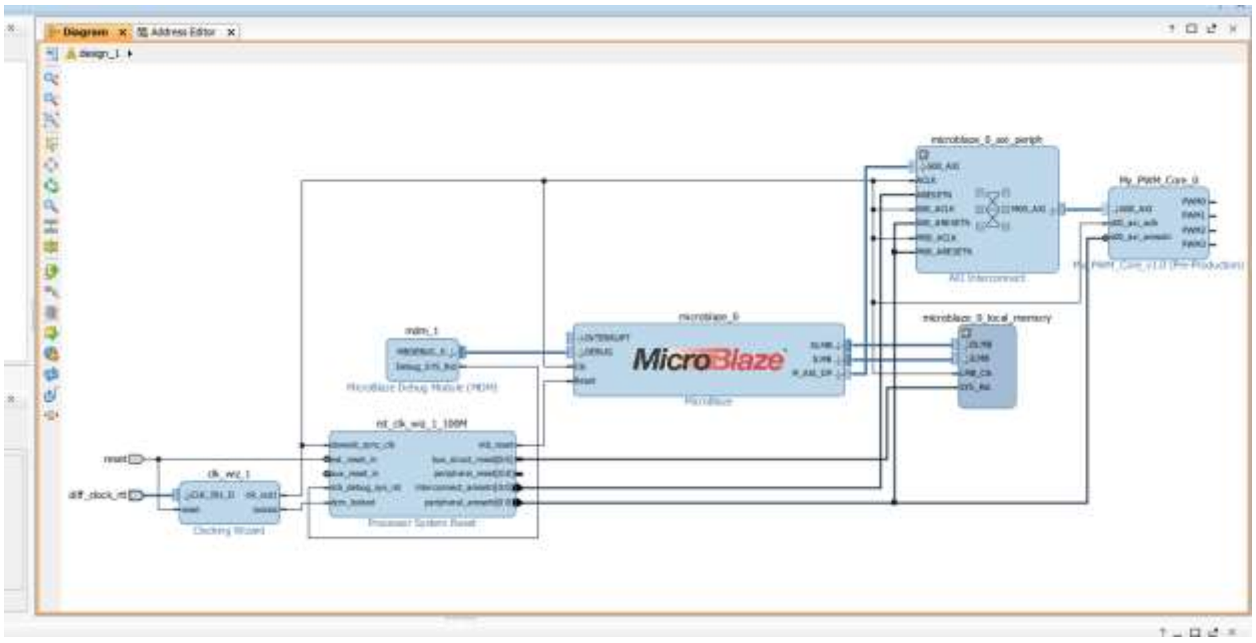
Run Connection Automation

Automatically make connections in your design by checking the boxes of the interfaces to connect. Select an interface on the left to display its configuration options on the right.

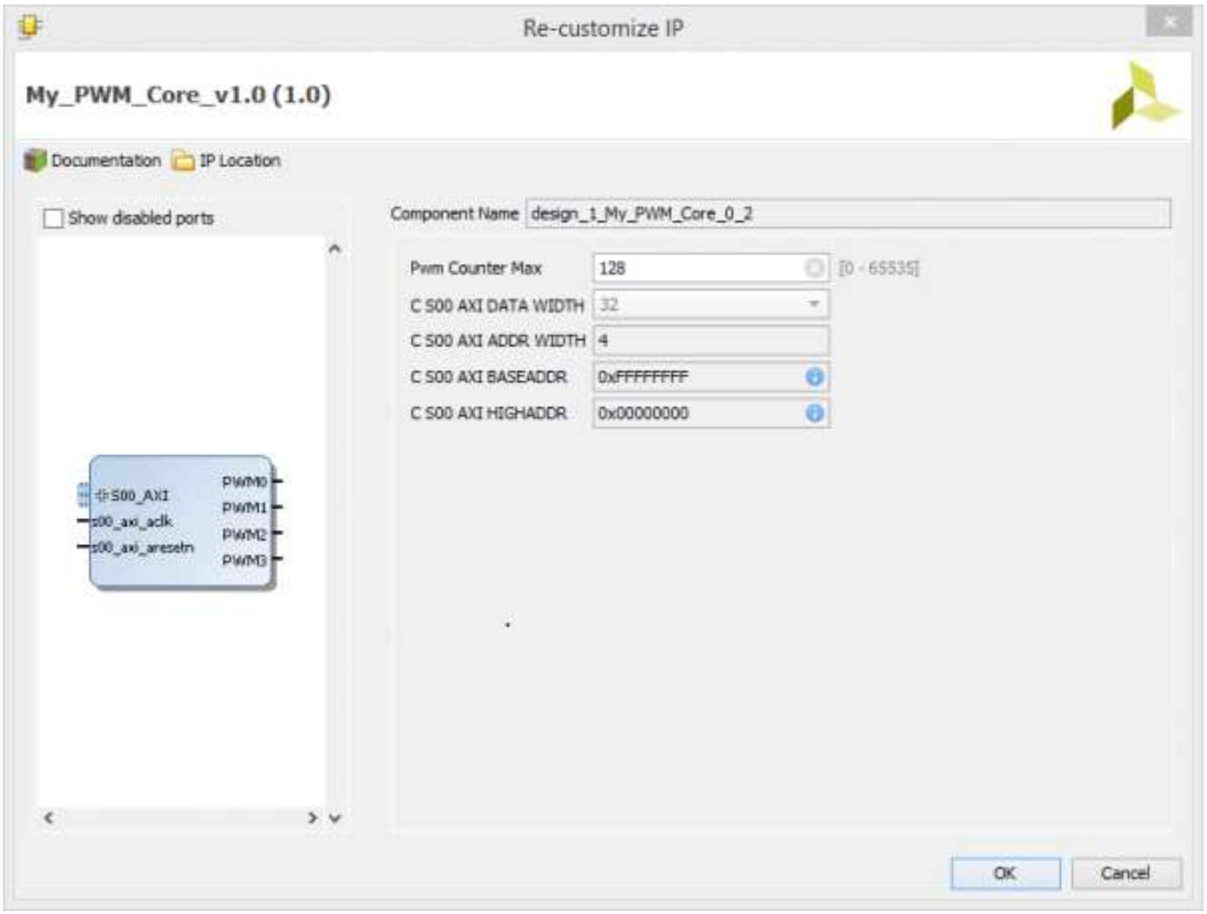
- [✓] All Automation (4 out of 4 selected)
  - [✓] clk_wiz_1
    - [✓] CLK_IN1_D
    - [✓] reset
  - [✓] My_PWM_Core_0
    - [✓] S00_AXI
  - [✓] rst_clk_wiz_1_100M
    - [✓] ext_reset_in

Select an interface pin on the left panel to view its options

Click on regenerate layout . You design should look like this.



7.6) Double-click our **PWM core** to customize the parameter that we made earlier. Set the PWM Counter Max to 1024. and click **OK**.



Re-customize IP

My_PWM_Core_v1.0 (1.0)

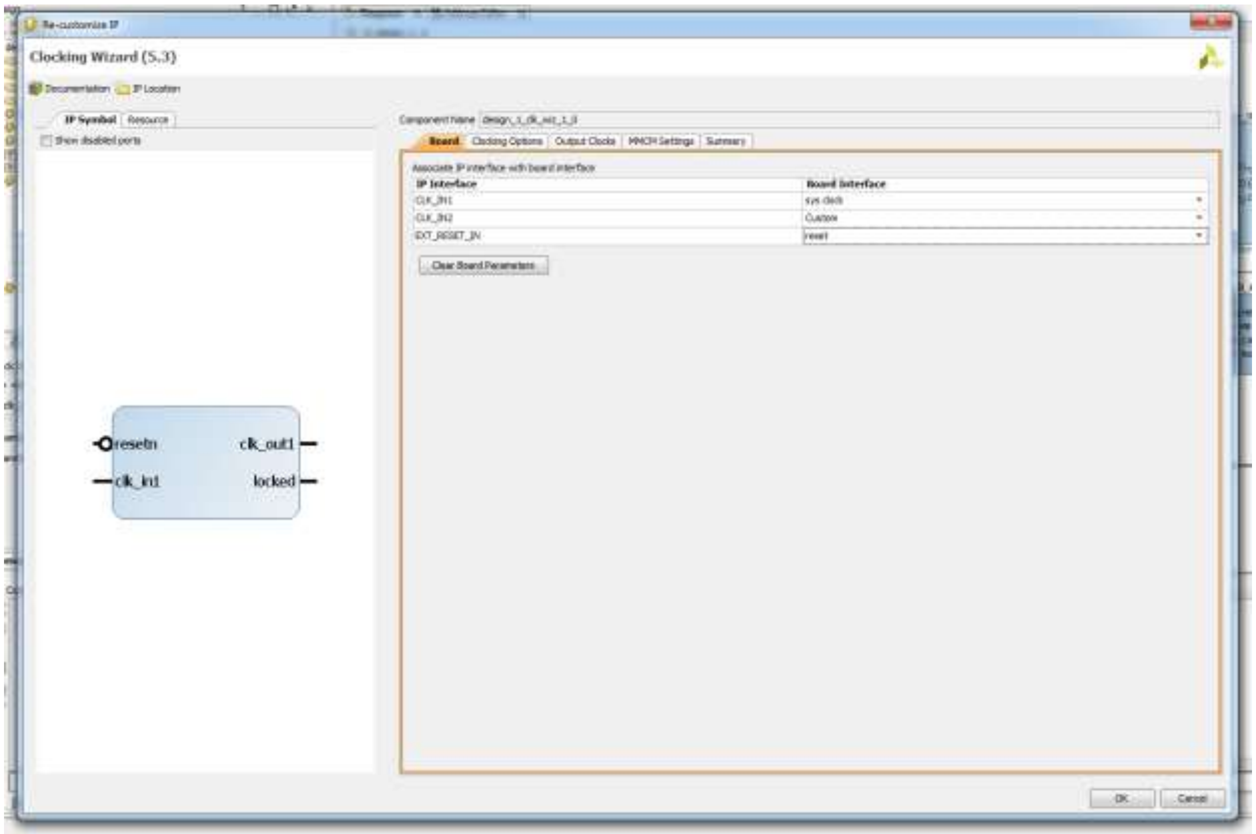
Documentation IP Location

Show disabled ports

Component Name design_1_My_PWM_Core_0_2

Pwm Counter Max	128	[0 - 65535]
C S00 AXI DATA WIDTH	32	
C S00 AXI ADDR WIDTH	4	
C S00 AXI BASEADDR	0xFFFFFFFF	
C S00 AXI HIGHADDR	0x00000000	

Double-click on the clock wizard to customize its paramaters. For the Board tab set the CLK_IN1 to sys_clock and EXT_RESET_IN to reset from the drop down menus.
For the Output Clocks tab set clk_out1 to 25MHz and reset type to **Active Low** and then click on OK and then rerun connection automation and all automation. Select and
delete the unconnected diff_clock_rtl port previously connected to the clk_wiz_1 block.



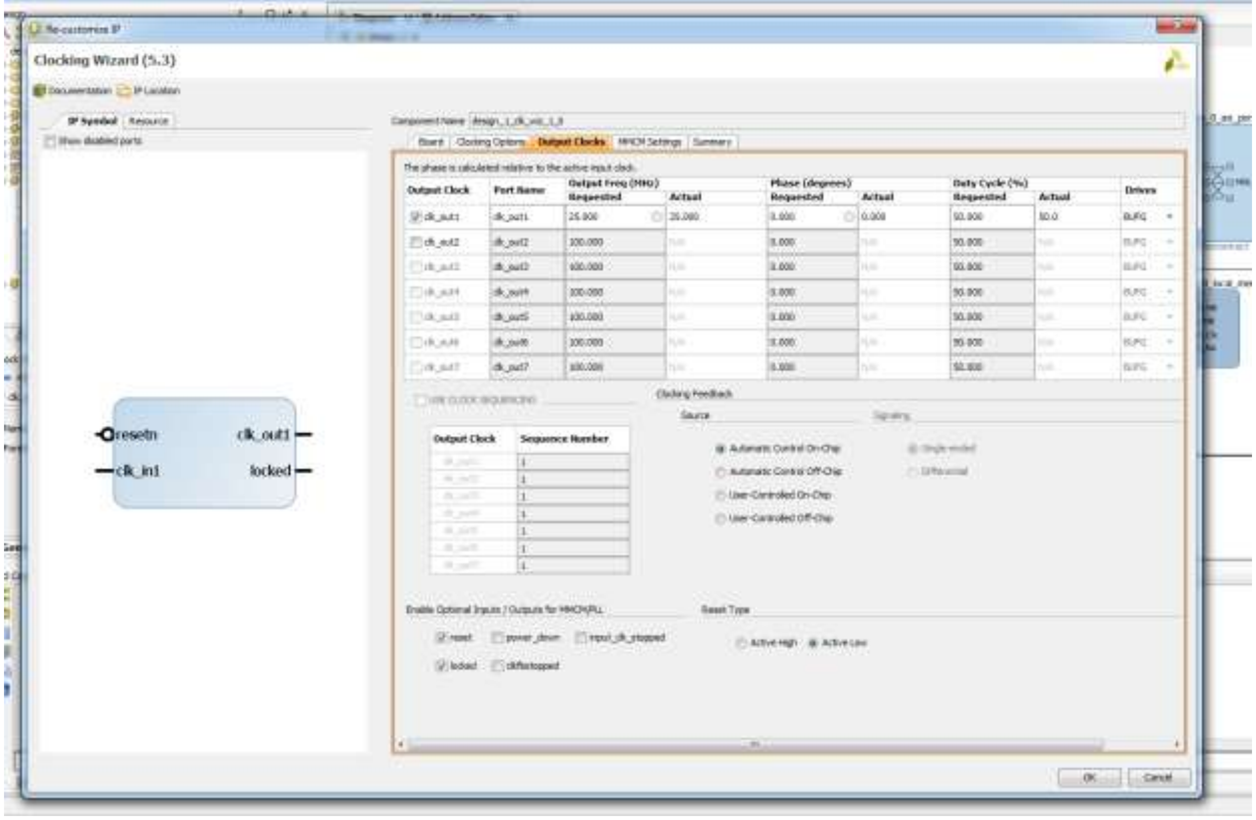
Component Name design_1_clk_wiz_1_0

Board | Clocking Options | Output Clocks | MMCM Settings | Summary

Associate IP interface with board interface

IP Interface	Board Interface
CLK_IN1	sys clock
CLK_IN2	Custom
EXT_RESET_IN	reset

Clear Board Parameters



Component Name design_1_clk_wiz_1_0

Board | Clocking Options | Output Clocks | MMCM Settings | Summary

The phase is calculated relative to the active input clock.

Output Clock	Port Name	Output Freq (MHz)		Phase (degrees)		Duty Cycle (%)		Drives
		Requested	Actual	Requested	Actual	Requested	Actual	
✓ clk_out1	clk_out1	25.000	25.000	0.000	0.000	50.000	50.0	BUFG
clk_out2	clk_out2	100.000	N/A	0.000	N/A	50.000	N/A	BUFG
clk_out3	clk_out3	100.000	N/A	0.000	N/A	50.000	N/A	BUFG
clk_out4	clk_out4	100.000	N/A	0.000	N/A	50.000	N/A	BUFG
clk_out5	clk_out5	100.000	N/A	0.000	N/A	50.000	N/A	BUFG
clk_out6	clk_out6	100.000	N/A	0.000	N/A	50.000	N/A	BUFG
clk_out7	clk_out7	100.000	N/A	0.000	N/A	50.000	N/A	BUFG

USE CLOCK SEQUENCING

Output Clock	Sequence Number
clk_out1	1
clk_out2	1
clk_out3	1
clk_out4	1
clk_out5	1
clk_out6	1
clk_out7	1

Clocking Feedback

Source: [✓ Automatic Control On-Chip] Automatic Control Off-Chip / User-Controlled On-Chip / User-Controlled Off-Chip

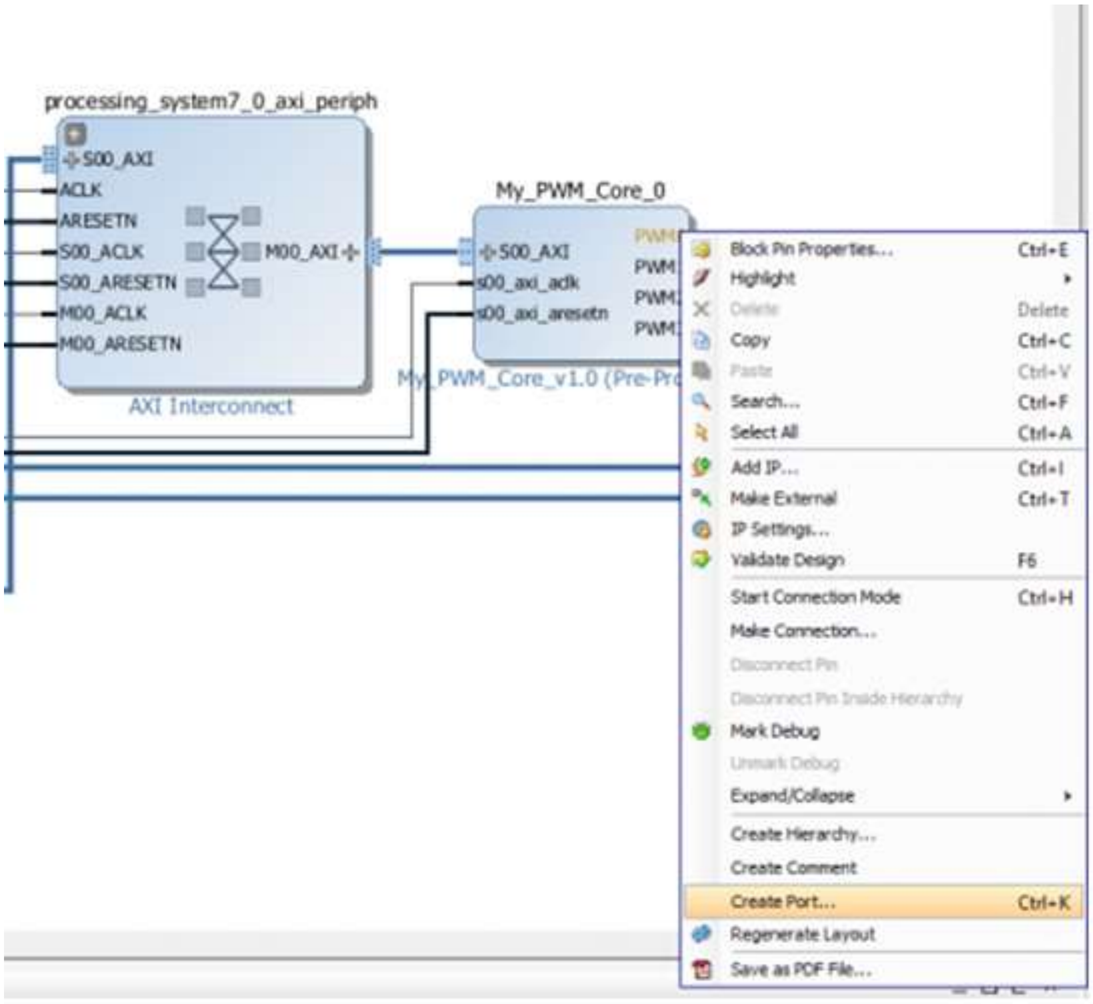
Signaling: [✓ Single-ended] Differential

Enable Optional Inputs / Outputs for MMCM/PLL

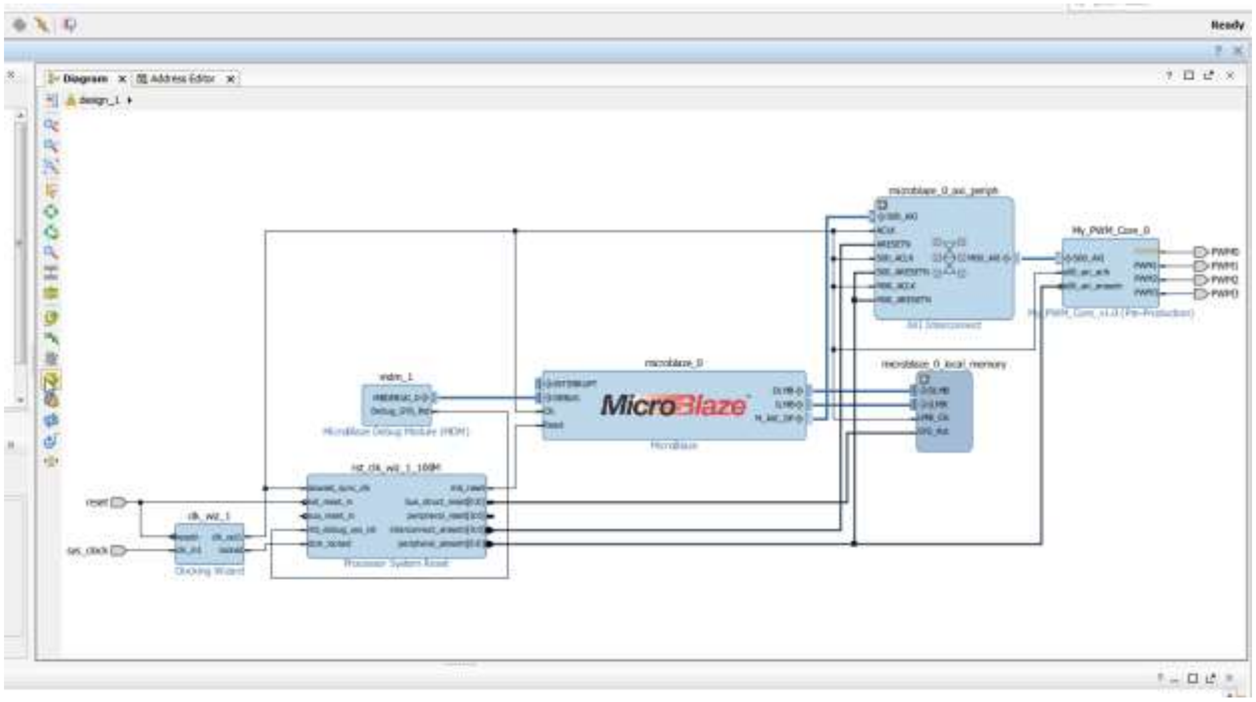
✓ reset power_down input_clk_stopped
✓ locked clkfbstopped

Reset Type: Active High [✓ Active Low]

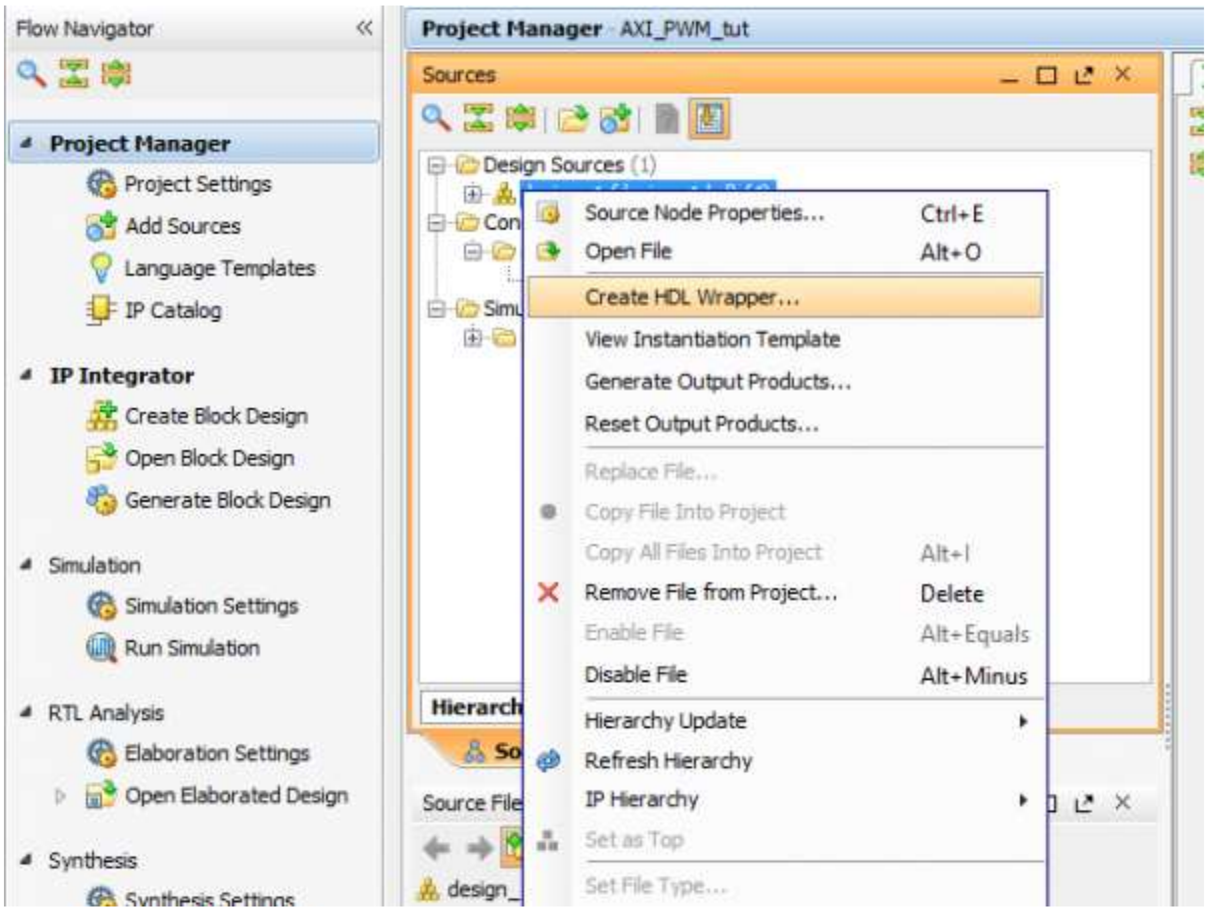
7.7) Right-click each of the PWM signals and select **Create Port…. You can leave the default name for each port.**



7.8) Click the button for **Validate Design**. The design should look as shown below.



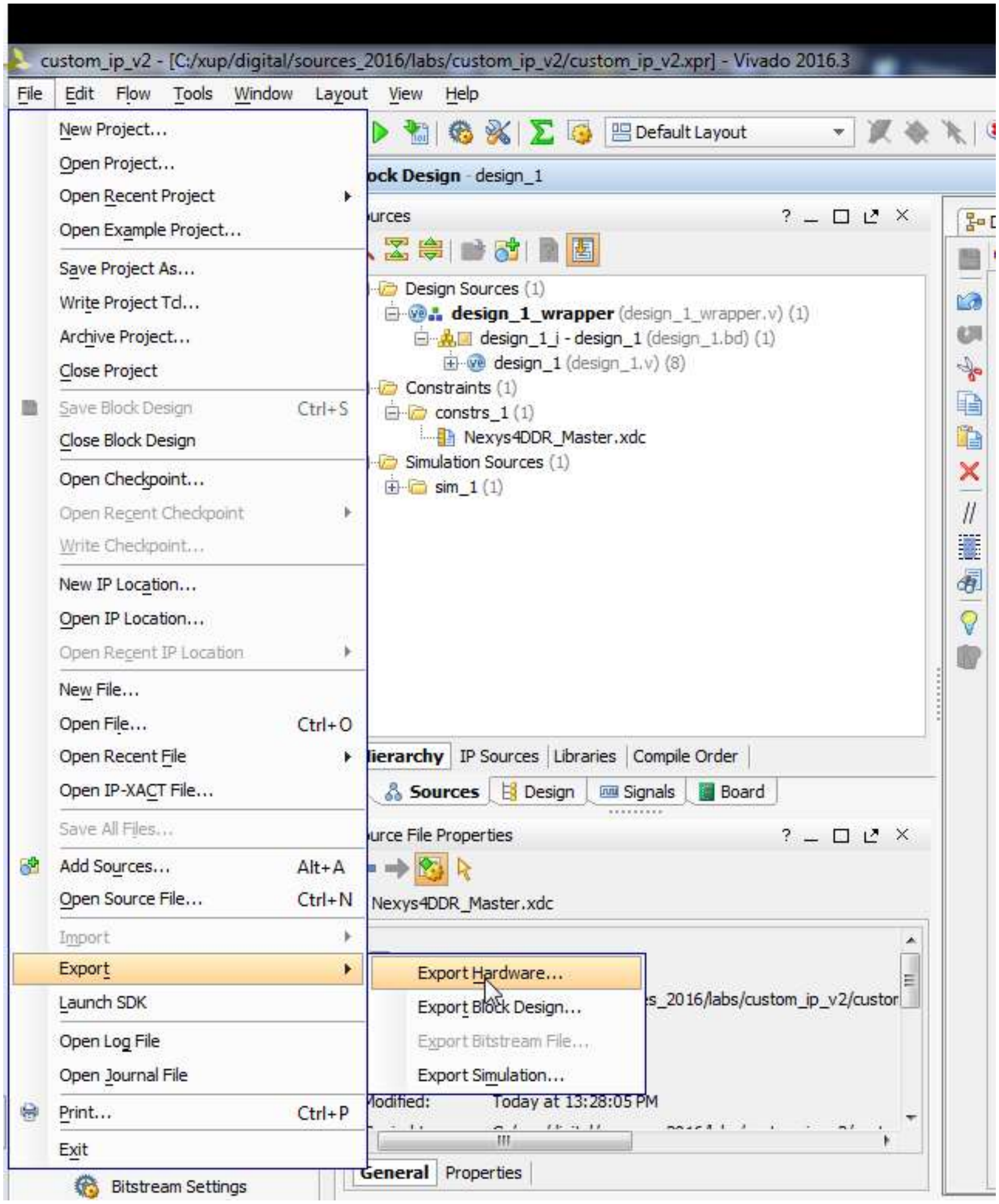
7.9) Click **on the source tab and** right-click the “design_1.bd” file and **Create HDL Wrapper and Let Vivado manage wrapper and auto-update and then click OK.**



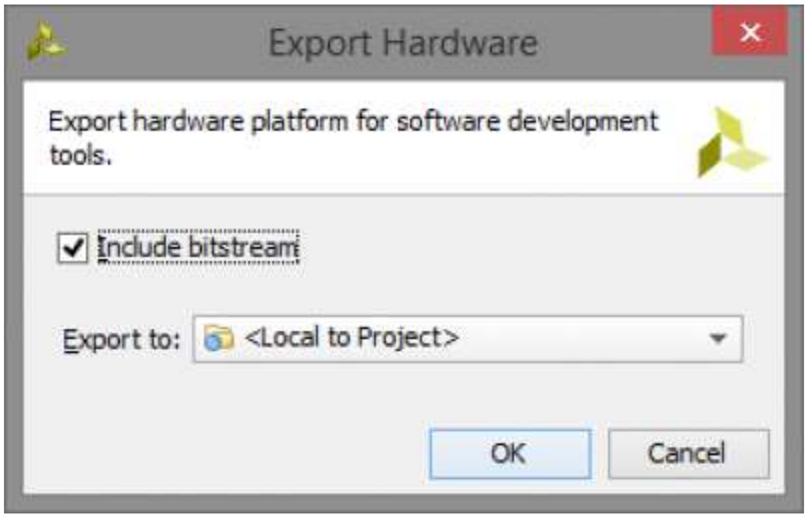
7.10) Double-click **design_1_wrapper.v** to open it in the editor. Take note of the names for the signals. Add the “Nexys4DDR_CustomIP_Master.xdc” constraint file that was provided with the tutorial.

7.11) Click **Generate Bitstream**. Building the bit file can take some time.

7.12) Export the hardware by going to **File→Export→Export Hardware…**



7.13) Check the box **Include bitstream** and click **Ok**

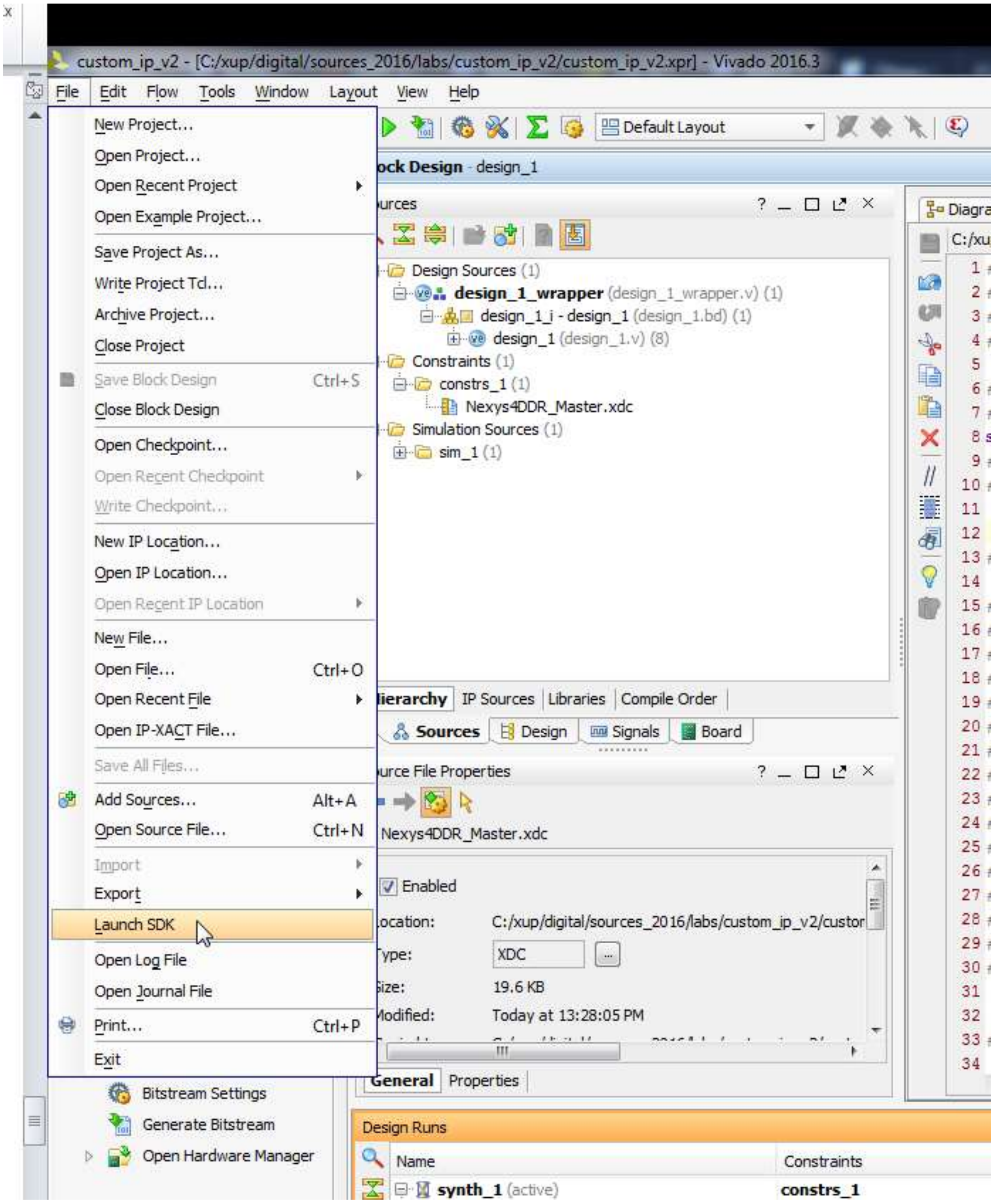

Export Hardware

Export hardware platform for software development tools.

☑ Include bitstream

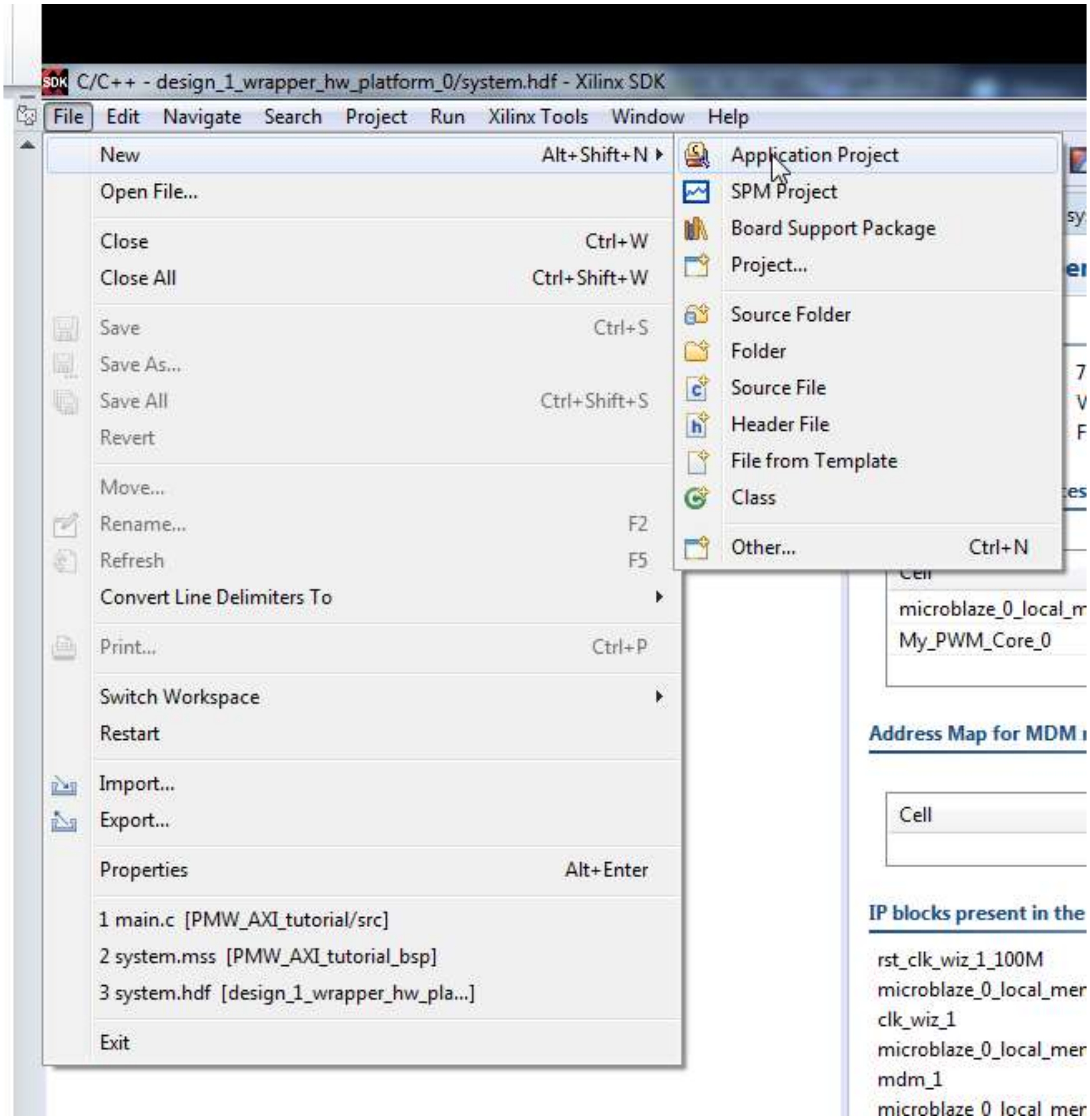
Export to: <Local to Project>

7.14) Select **File→Launch SDK**, and hit **Ok** on the window that pops up.

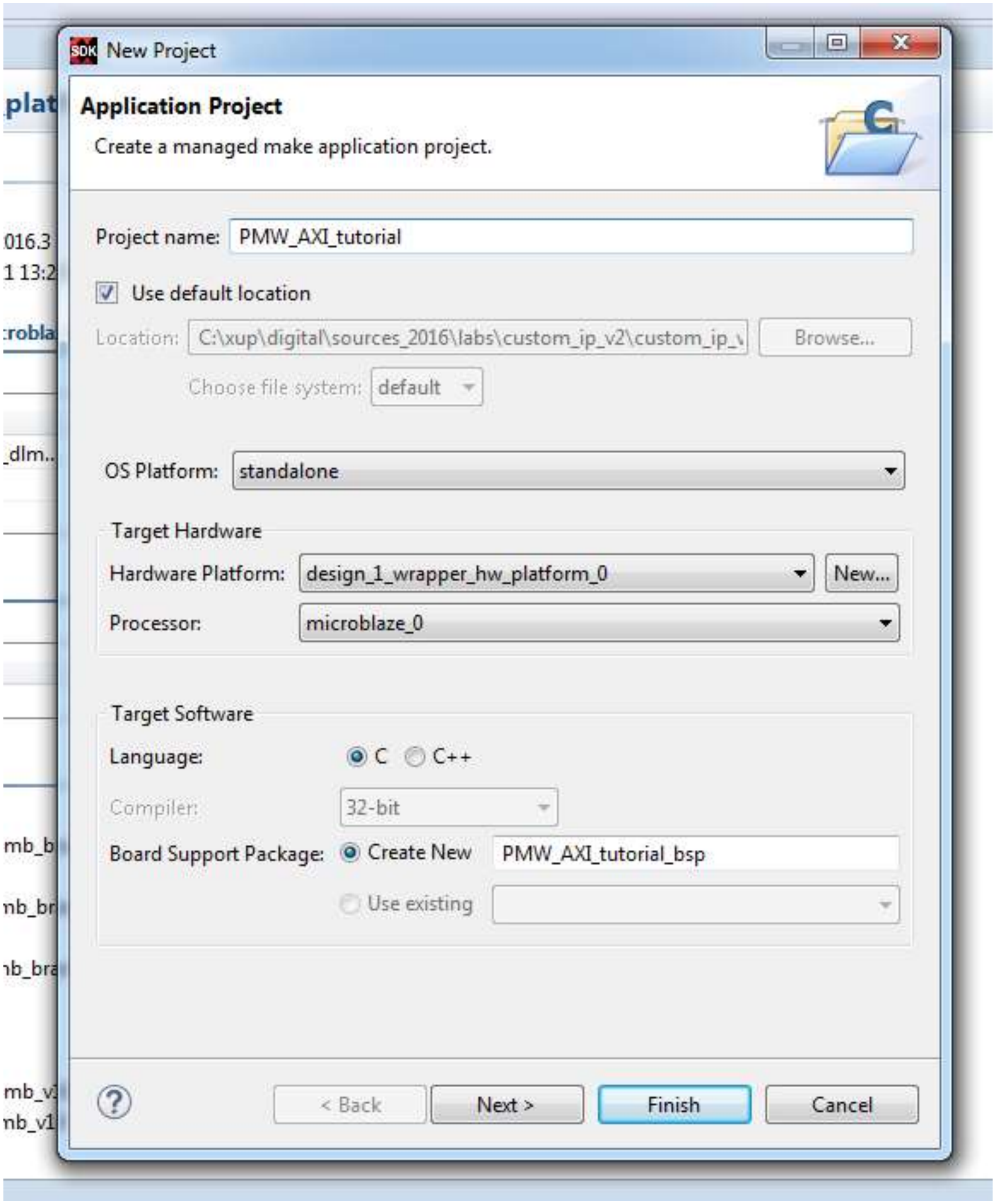


8. Programming in SDK

8.1) Create a new application project File-> New - > Application Project



8.2) Setup the application project window as shown below, click **Next** select Empty Application and then click **Finish**.



New Project

Application Project

Create a managed make application project.

Project name: PMW_AXI_tutorial

[✓] Use default location

Location: C:\xup\digital\sources_2016\labs\custom_ip_v2\custom_ip_\

Choose file system: default

OS Platform: standalone

Target Hardware

Hardware Platform: design_1_wrapper_hw_platform_0

Processor: microblaze_0

Target Software

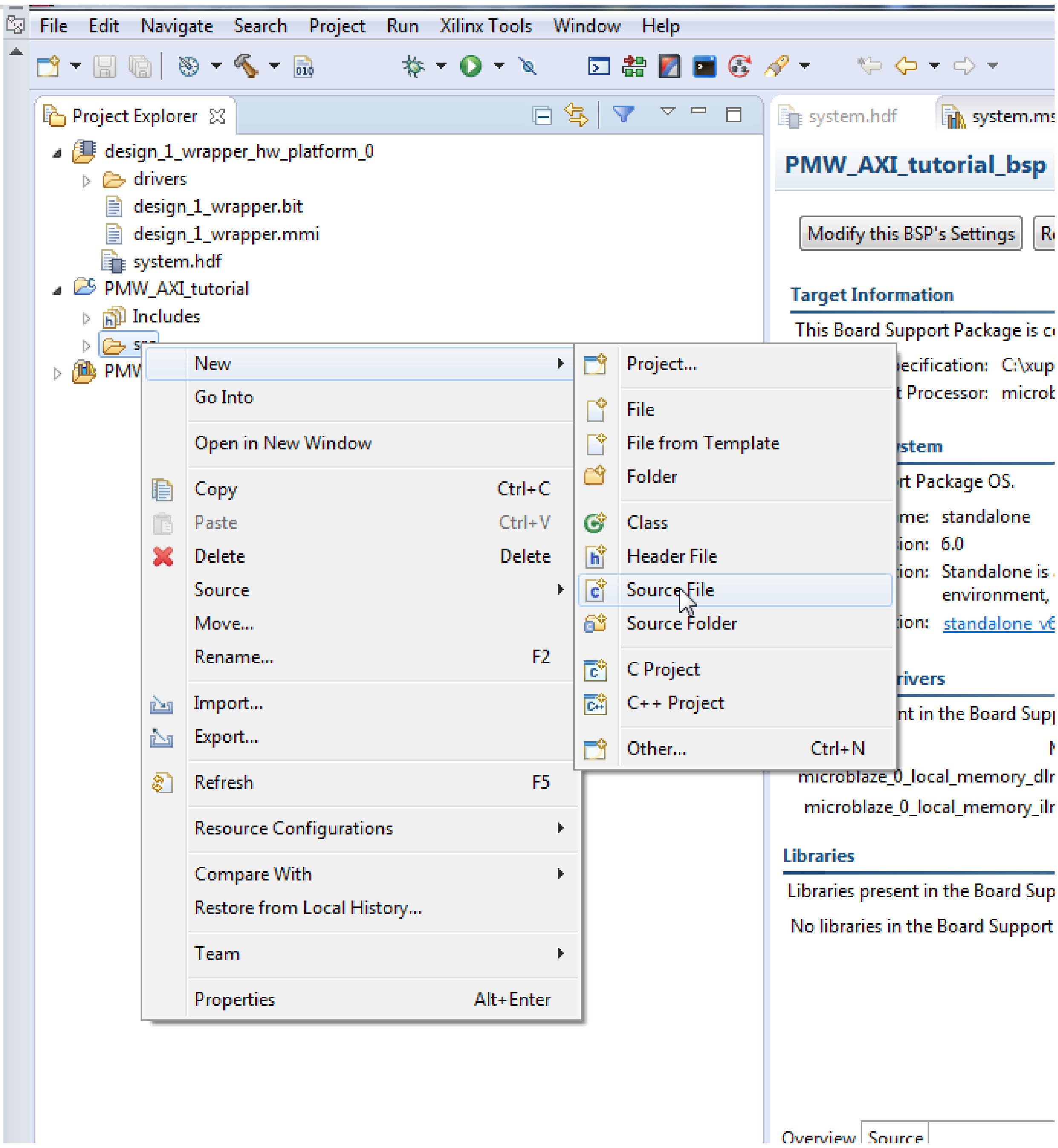
Language: [✓ C] C++

Compiler: 32-bit

Board Support Package: [✓ Create New] PMW_AXI_tutorial_bsp

Use existing

8.3) Expand the **PWM_AXI_tutorial→src** folder. Right-click the **src** folder and **add a new source file**. Create a file named “main.c”.



8.4) Add the lines:

```
#include "xparameters.h"
```

```
#include "xil_io.h"
```

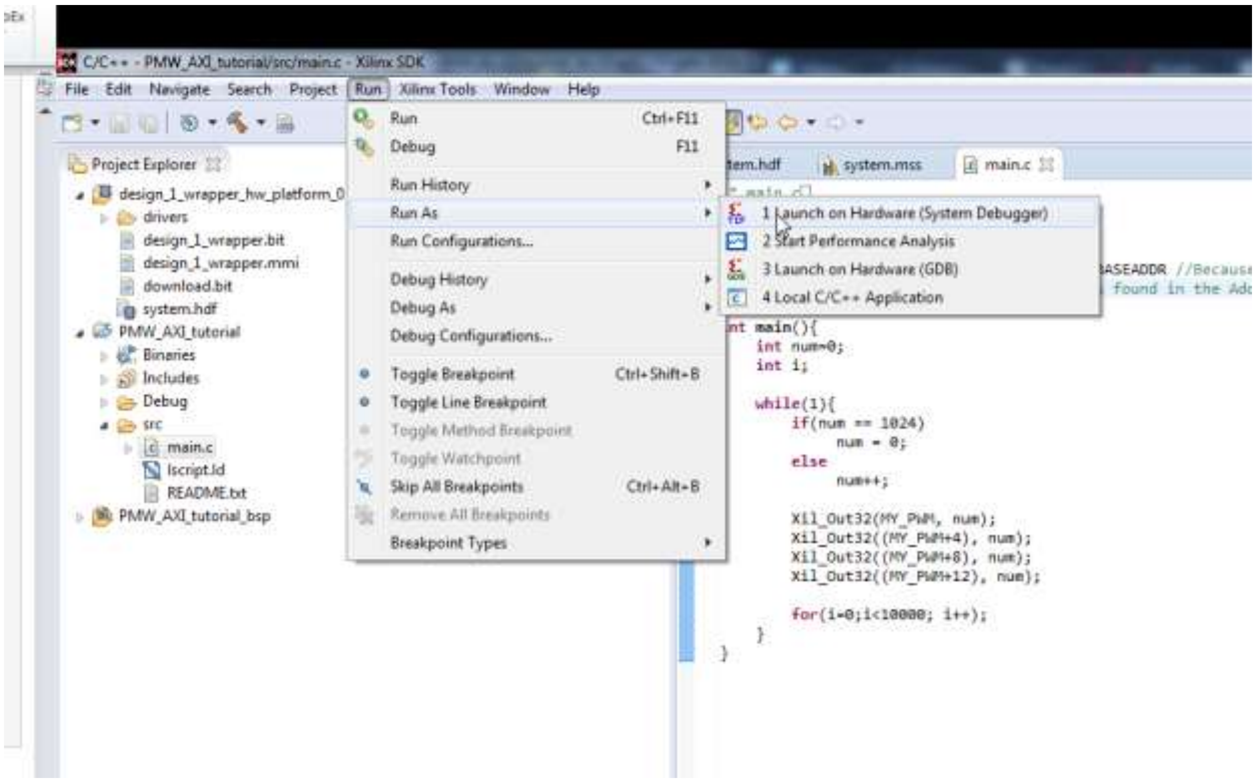
```

```

```
#define MY_PWM XPAR_MY_PWM_CORE_0_S00_AXI_BASEADDR

//#define MY_PWM 0x44A00000 //This value is found in the Address Map tab in Vivado (s
ystem.hdf)


int main(){

    int num=0;

    int i;


    while(1){

        if(num == 1024)

             num = 0;

        else

             num++;


        Xil_Out32(MY_PWM, num);

        Xil_Out32((MY_PWM+4), num);

        Xil_Out32((MY_PWM+8), num);

        Xil_Out32((MY_PWM+12), num);

        for(i=0;i<10000; i++);
```

```
    }
```

```
}
```

8.5) To program the FPGA, Go to **Xilinx Tools→Program FPGA**. To load the SDK application onto the Nexys4DDR, click on main.c and select **Run -> Run As→Launch on Hardware (System Debugger)**.



```
nt main(){
    int num=0;
    int i;

    while(1){
        if(num == 1024)
            num = 0;
        else
            num++;

        Xil_Out32(MY_PWM, num);
        Xil_Out32((MY_PWM+4), num);
        Xil_Out32((MY_PWM+8), num);
        Xil_Out32((MY_PWM+12), num);

        for(i=0;i<10000; i++);
    }
}
```

9. Celebrate!

Now the 4 LEDs on the Nexys4DDR will be pulsing. You just created your own custom IP core.

9.1. Demonstrate the working circuit to the TA.