

# PMOD Tutorial

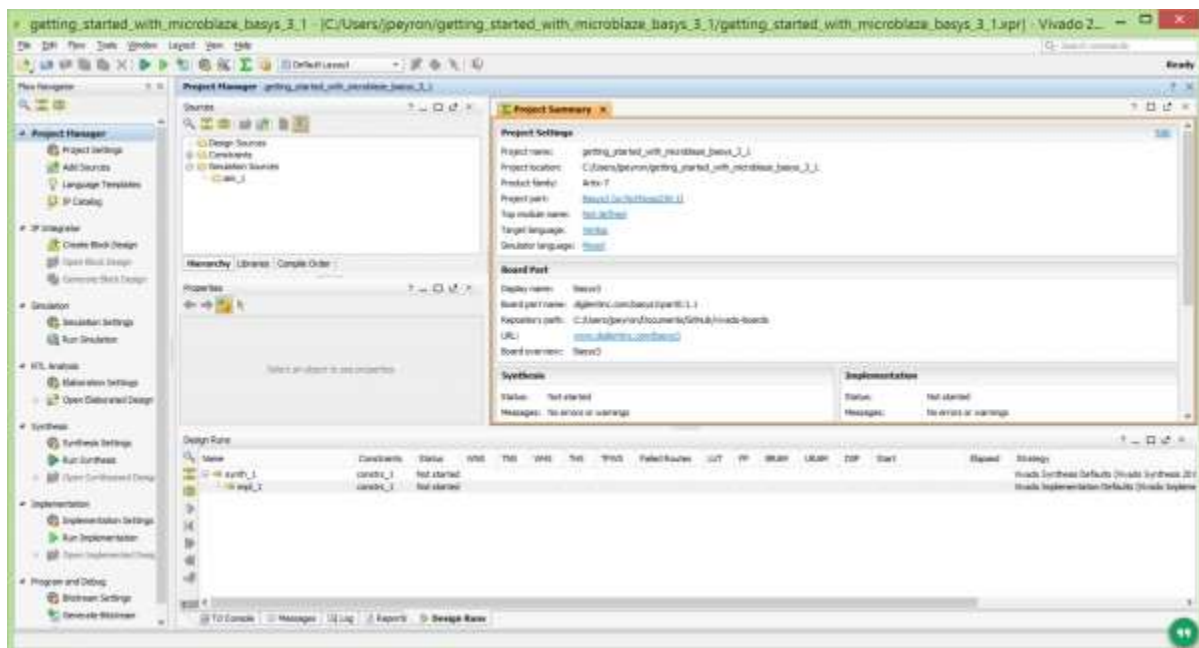
## 1. Overview

Digilent provides several IPs that are designed to make implementing and using a Pmod on an FPGA as straightforward as possible. This guide will describe how to use a Pmod IP core in Vivado Microblaze design.

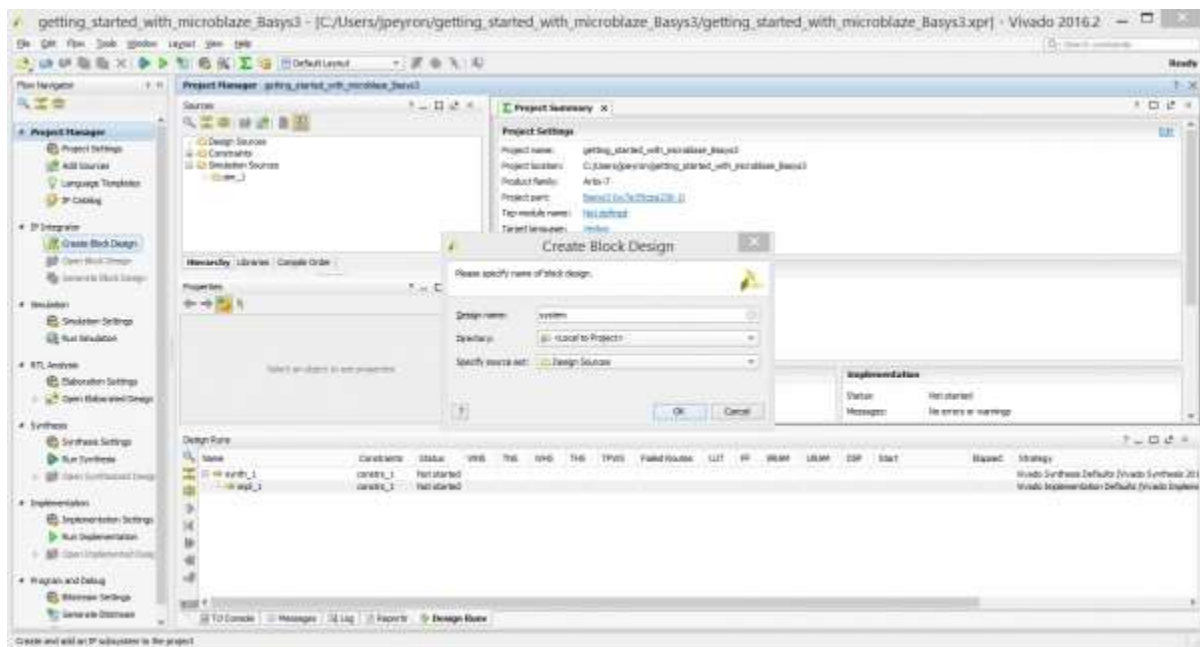
At the end of this tutorial you will have a Vivado design and demo for your FPGA platform that uses a Digilent Pmod IP core.


## 2. Create a New Microblaze Block Design

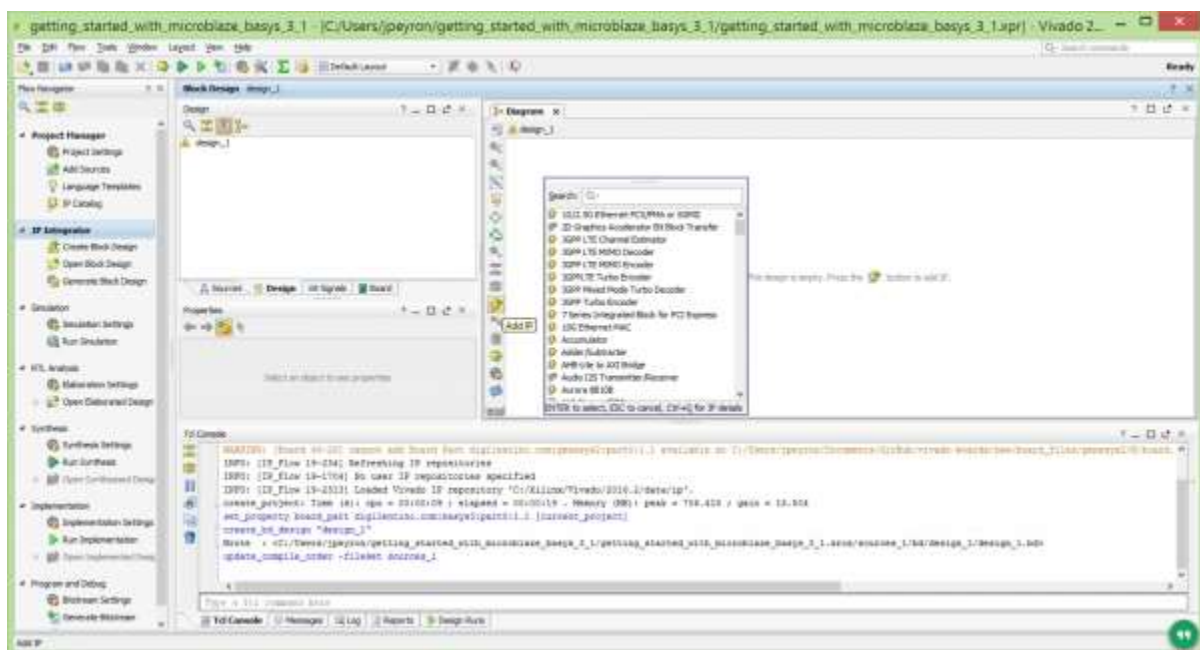
2.1) This is the main project window where you can create a IP based block design or add RTL based design sources. The flow navigator panel on the left provides multiple options on how to create a hardware design, perform simulation, run synthesis and implementation and generate a bit file. You can also program the board directly from Vivado with the generated bit file for an RTL project using the Hardware Manager. For our design, we will use the IP Integrator to create a new block design.



2.2) On the left you should see the Flow Navigator. Select **Create Block Design** under the IP Integrator. You can leave the default design\_1 and click OK.

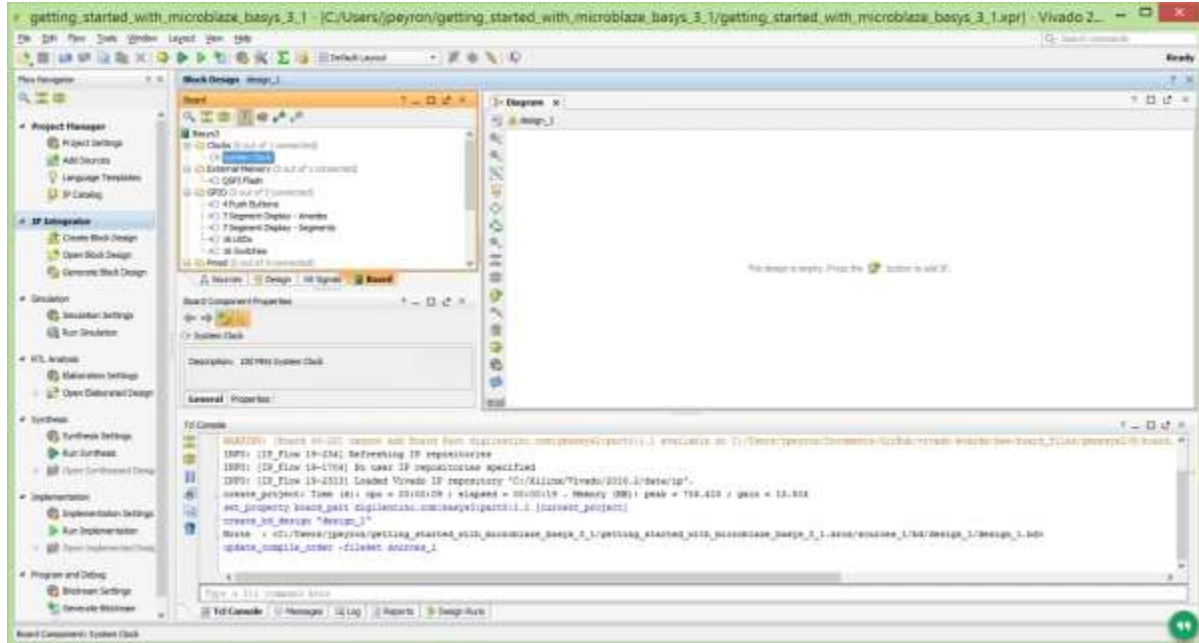


2.3) An empty design workspace is created where you can add IP blocks. Add an IP core by clicking on the  **Add IP** icon. This should open a catalog of pre-built IP blocks from Xilinx IP repository.



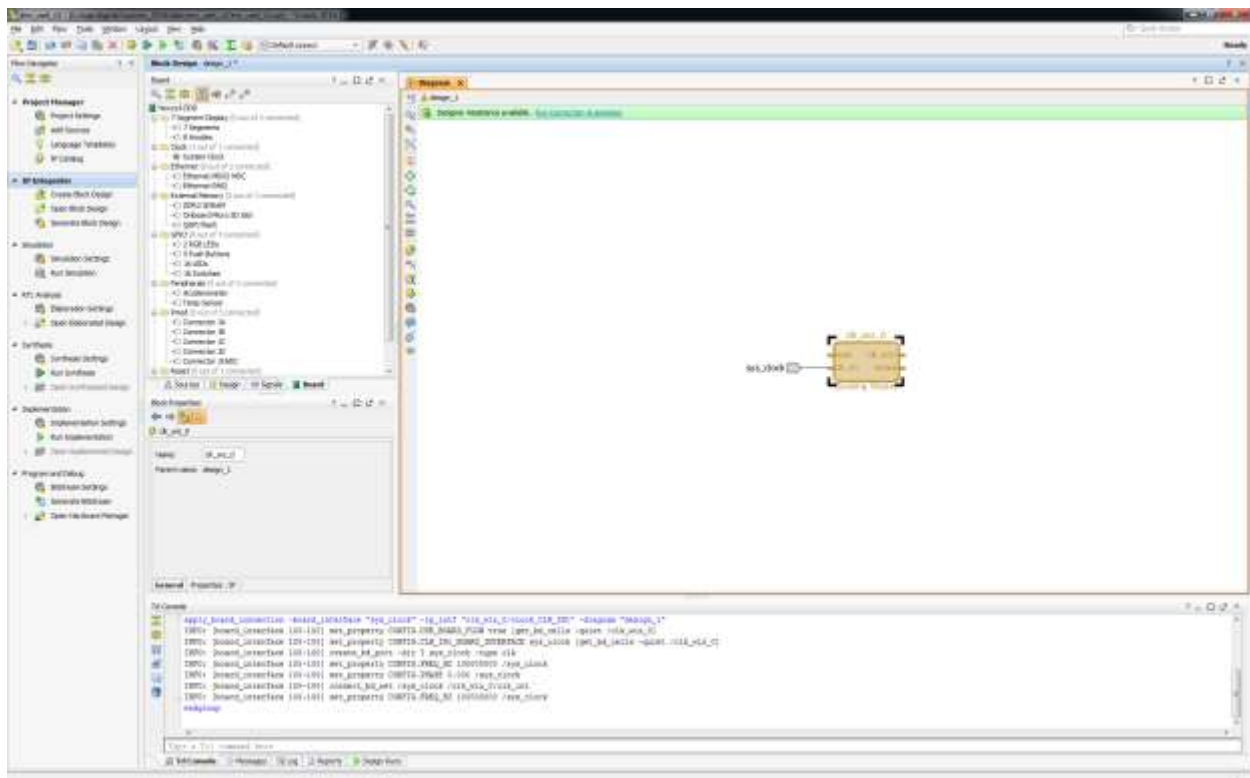
3. Adding the clock

### 3.1) Click the **Board** tab

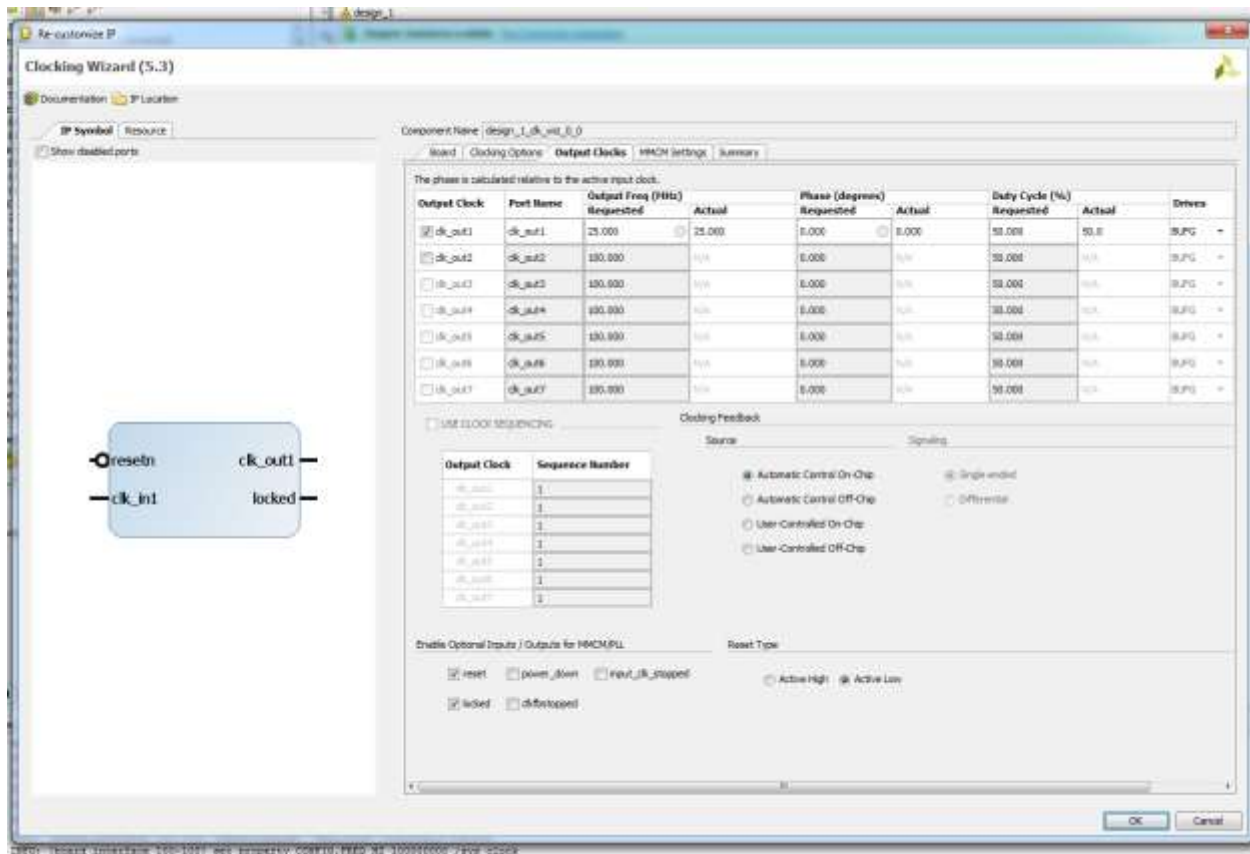


This list contains all of the components defined in the board file you installed before. These are already configured to work with several Vivado IPs.


3.2) Click and drag the **System Clock** component onto the empty block design. Vivado will automatically connect this system clock to a new Clocking Wizard block.

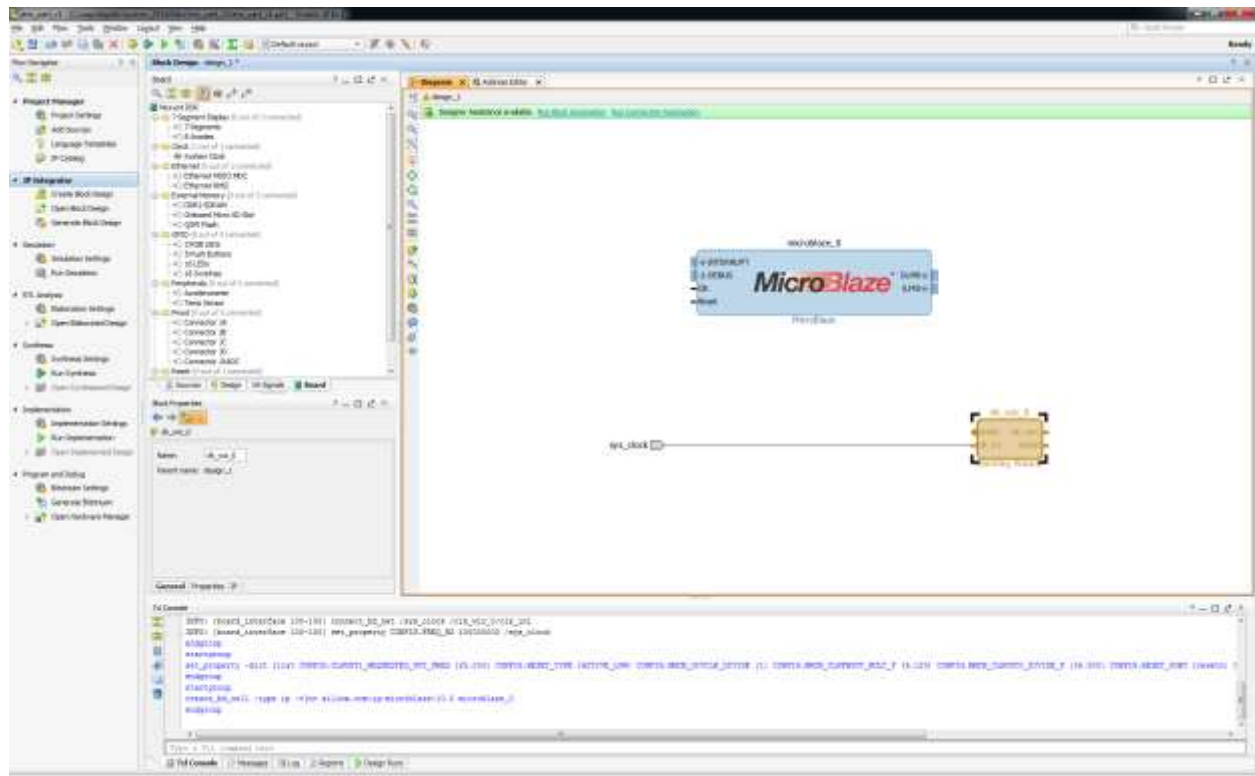


3.3) Double click the **Clocking Wizard** block to customize it. Click on the **Output Clocks** tab. Make sure **clk\_out1** is “25.000” and the **Reset Type** is **Active Low** and click OK.

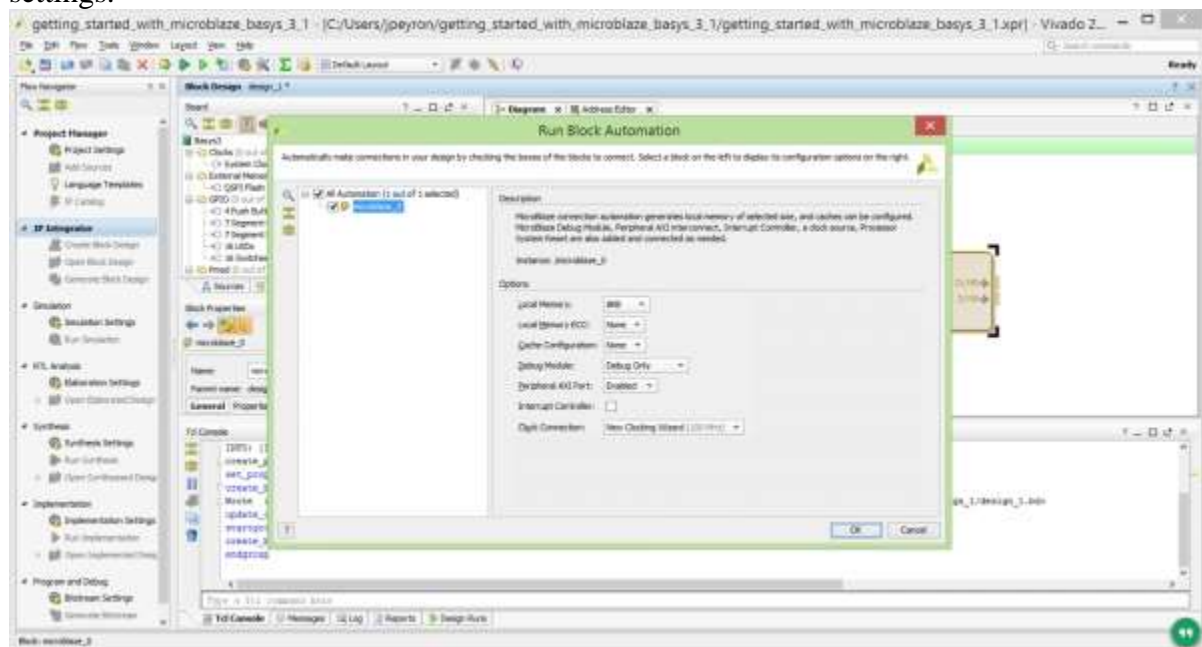


## 4. Adding Microblaze IP and Customization

4.1) Add an IP core by clicking on the  **Add IP** icon. Search for “Microblaze” and double click on it to add the IP block to your design. This is the Xilinx Microblaze IP block. When a new IP block is added the user can customize the block properties by either clicking on the **Run Block Automation** message prompt or by double clicking on the block itself.



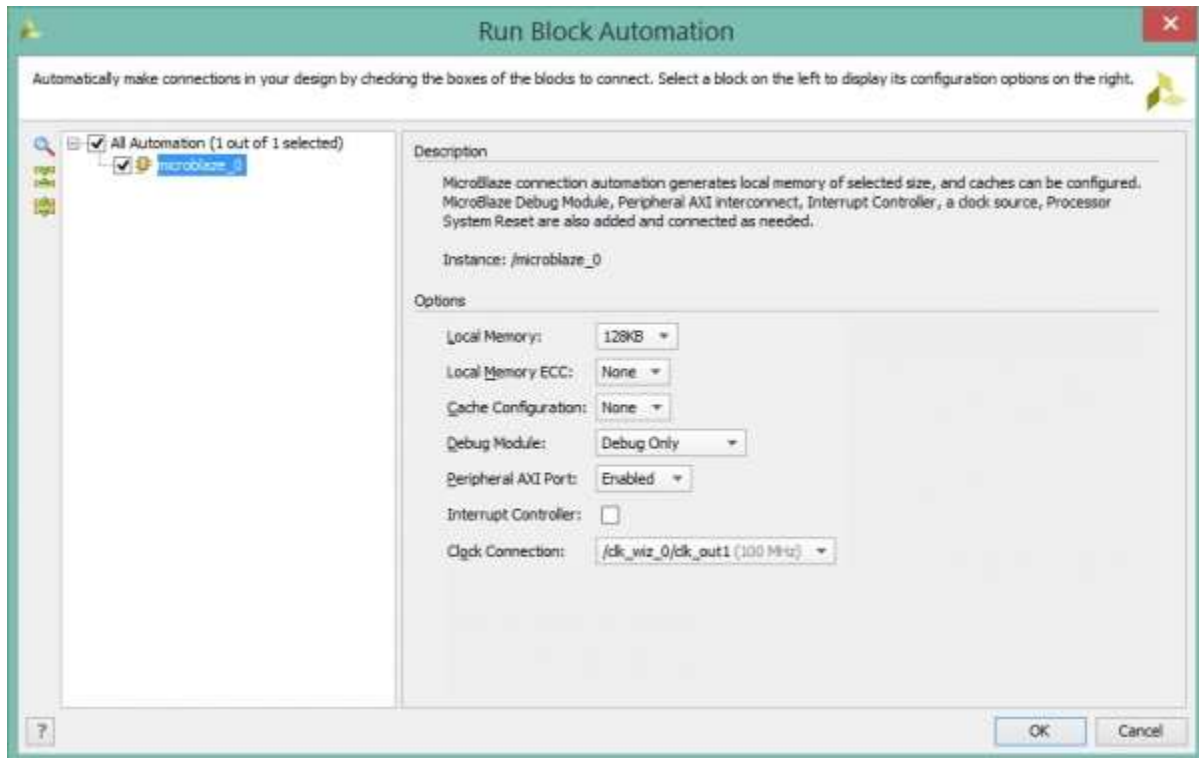
4.2) Select **Run Block Automation** and a customization assistant window will open with default settings.



4.3) Change default settings in the block options as shown below and click **OK**. This will customize the block with our new user settings.

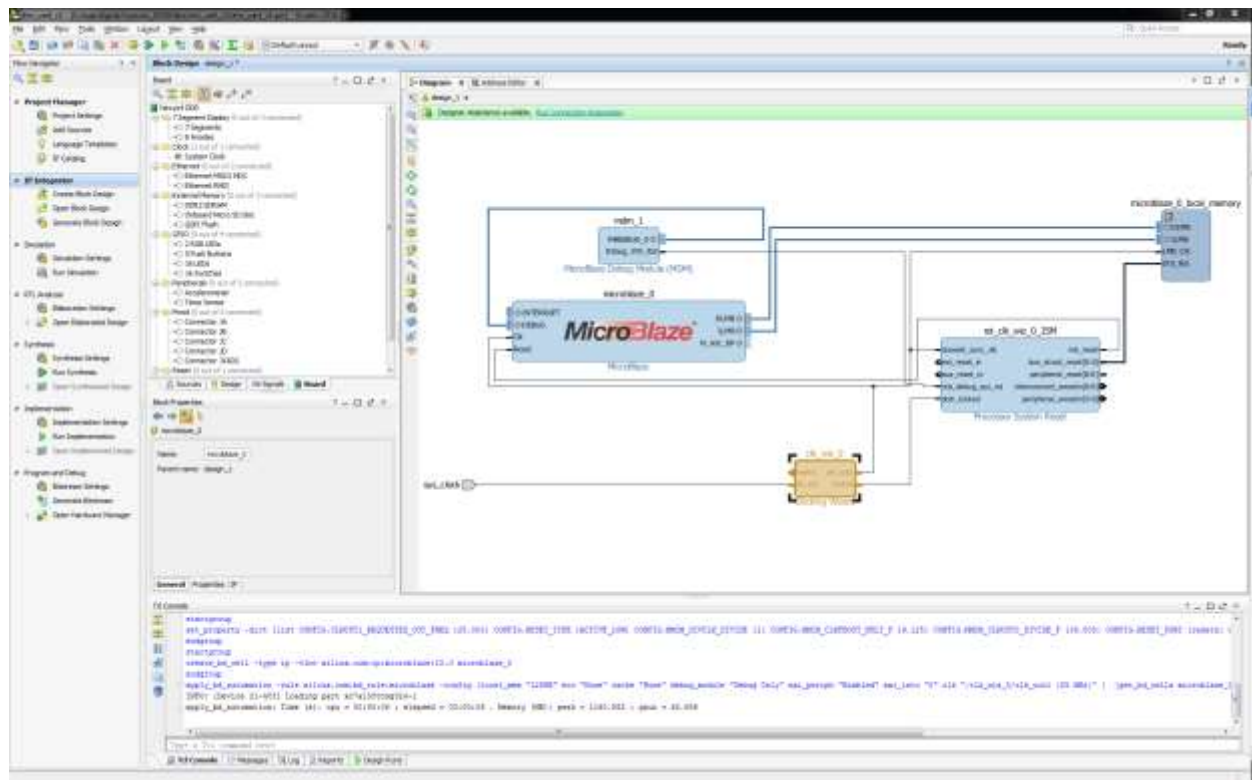
Local Memory: 128KB

Local Memory ECC: None  
Cache Configuration: None  
Debug Module: Debug only  
Peripheral AXI Port: Enabled  
Interrupt Controller: unchecked  
Clock Connection: /clk\_wiz0/clk\_out1(25 MHZ)



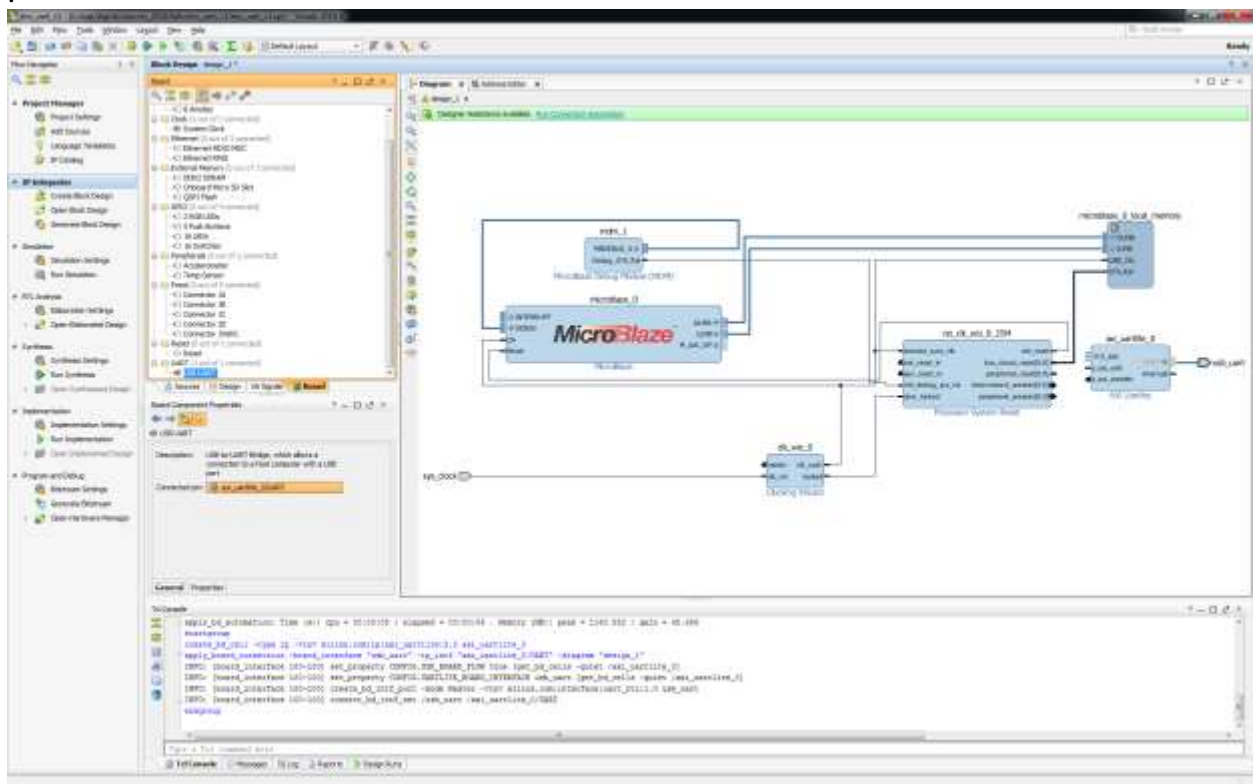
4.4) Running the block automation will auto-generate a set of additional IP blocks which will be added to our hardware design automatically based on the options selected in the previous step. **Do not click on Run Connection Automation yet.**



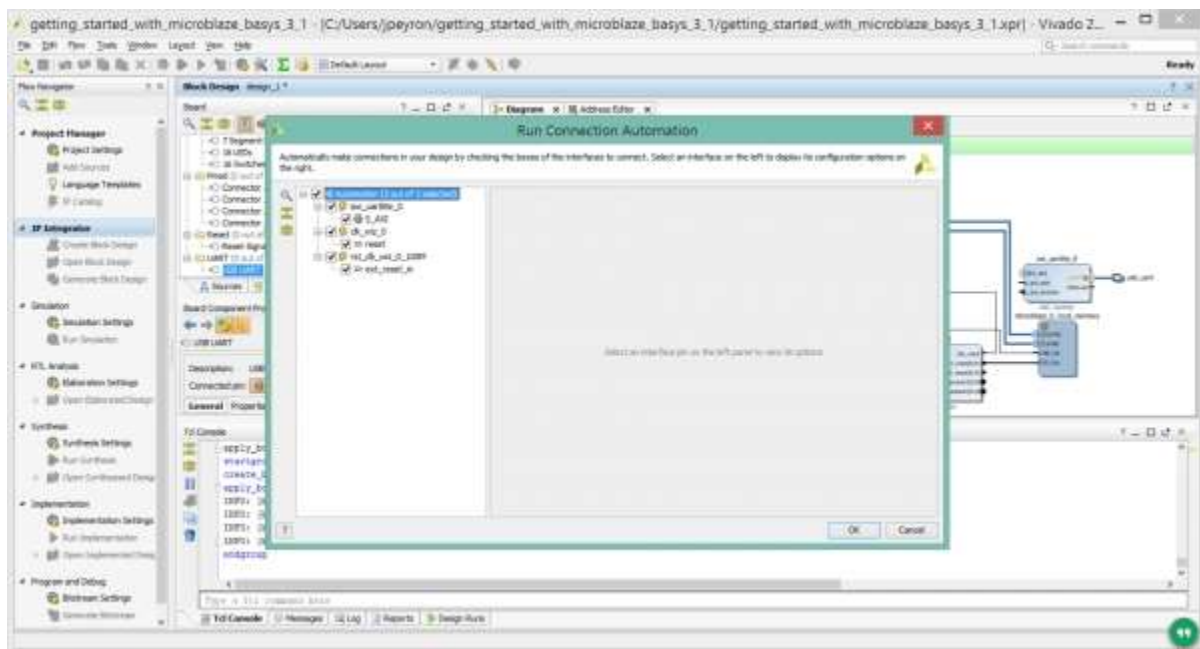



## 5. Adding Peripheral Components

5.1) Go into the **Boards** tab again and find the **USB UART** component. **Click and drag** this onto the block design to add the Uartlite block to your design.



5.2) Click **Run Connection Automation** in the green banner. Check the **All Automation** box and then click **OK**.




5.3) Select  **Validate Design**. This will check for design and connection errors.

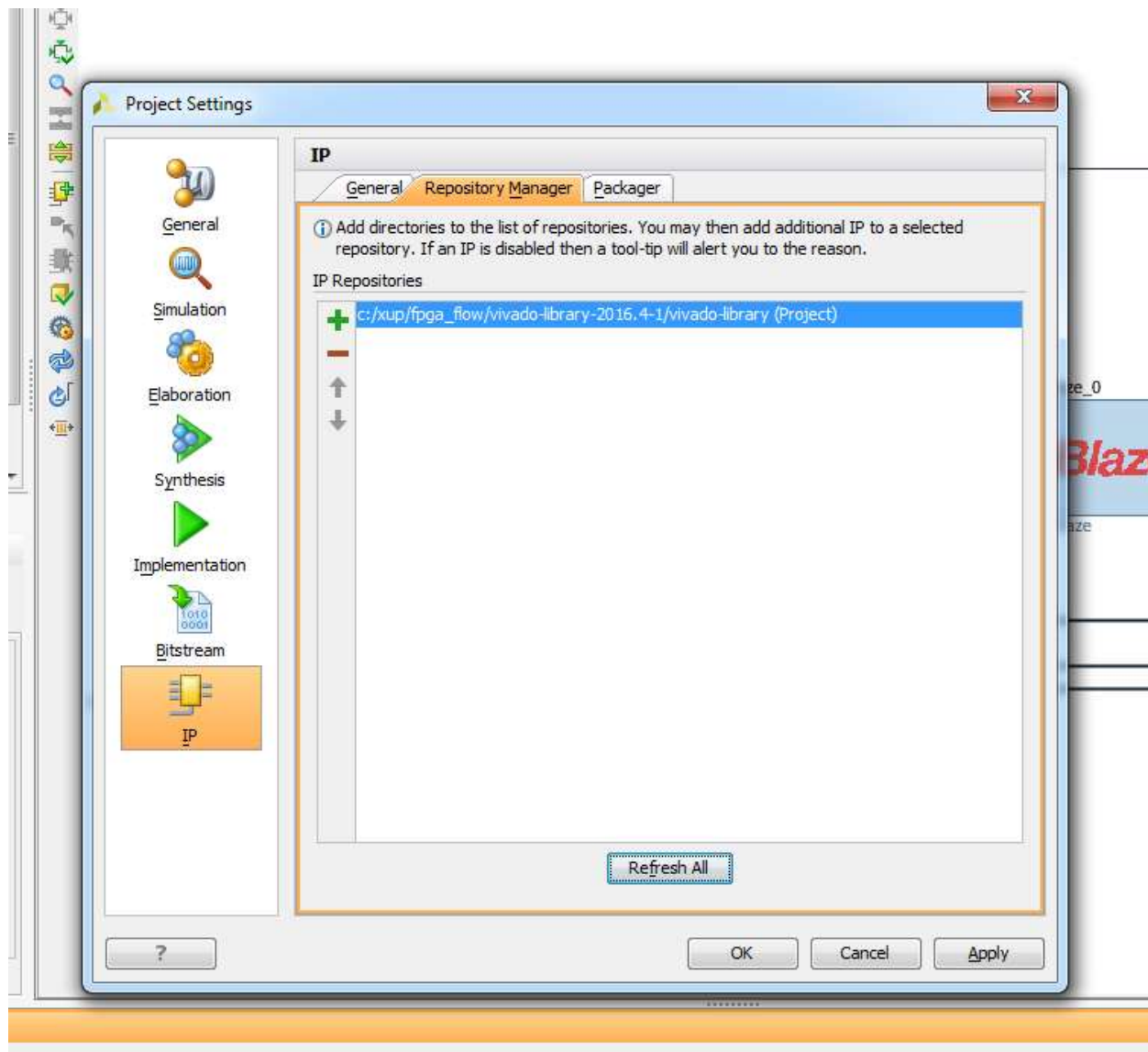




6.2) Click **Project Settings** under Project Manager.

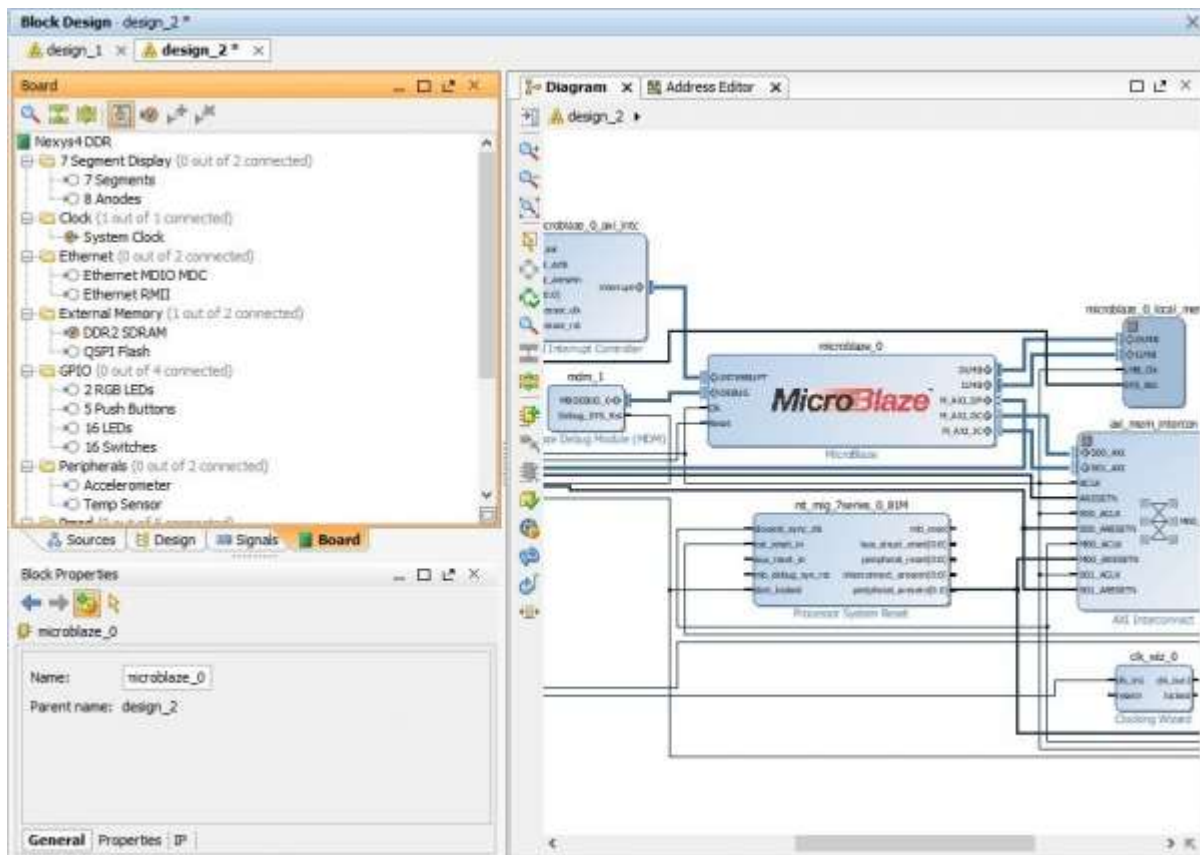


6.3) Click **IP** then open the **Repository Manager** tab. Click the  **Add** button and select the **vivado-library** folder from where the ZIP archive was extracted to. Click **OK**.



## 7. Add the Pmod to Your Block Design

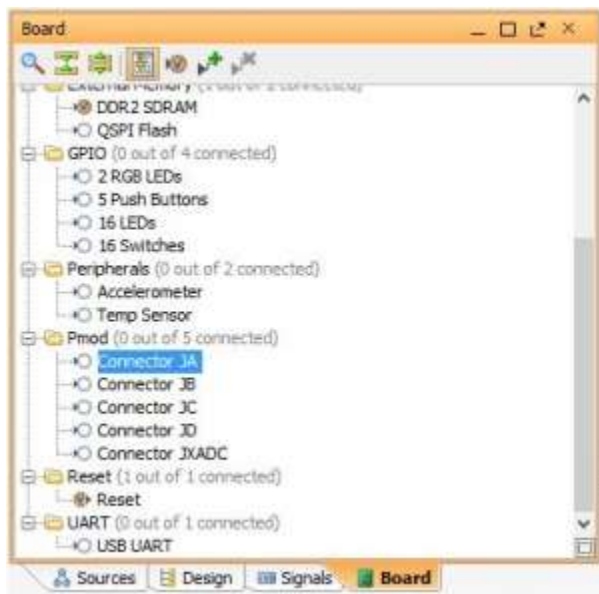
7.1) Click the **Board** tab (Highlighted in orange below)



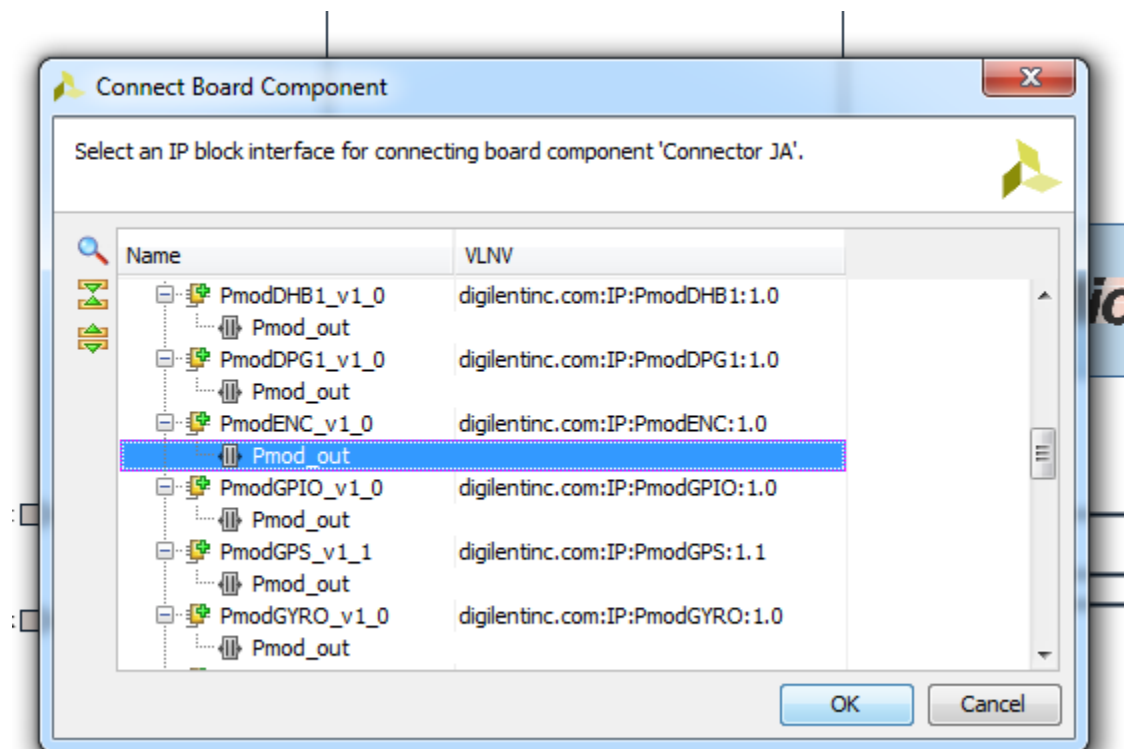
## Info

This list contains all of the components defined in the board file for your platform. You can use it to easily insert an IP block that will work with a piece of hardware found on your platform, for example an Ethernet port or general purpose LED. Several of these should have already been selected when you created your initial design in step 1.1.

7.2) Scroll down to the Pmod section of the board components. Double click the connector that you want to set up (we'll use the JA connector for this tutorial).



7.3) Select the Pmod IP for your specific Pmod and click **OK**. The photo below shows the IP core for PmodEnc (i.e. rotary encoder module) being selected.



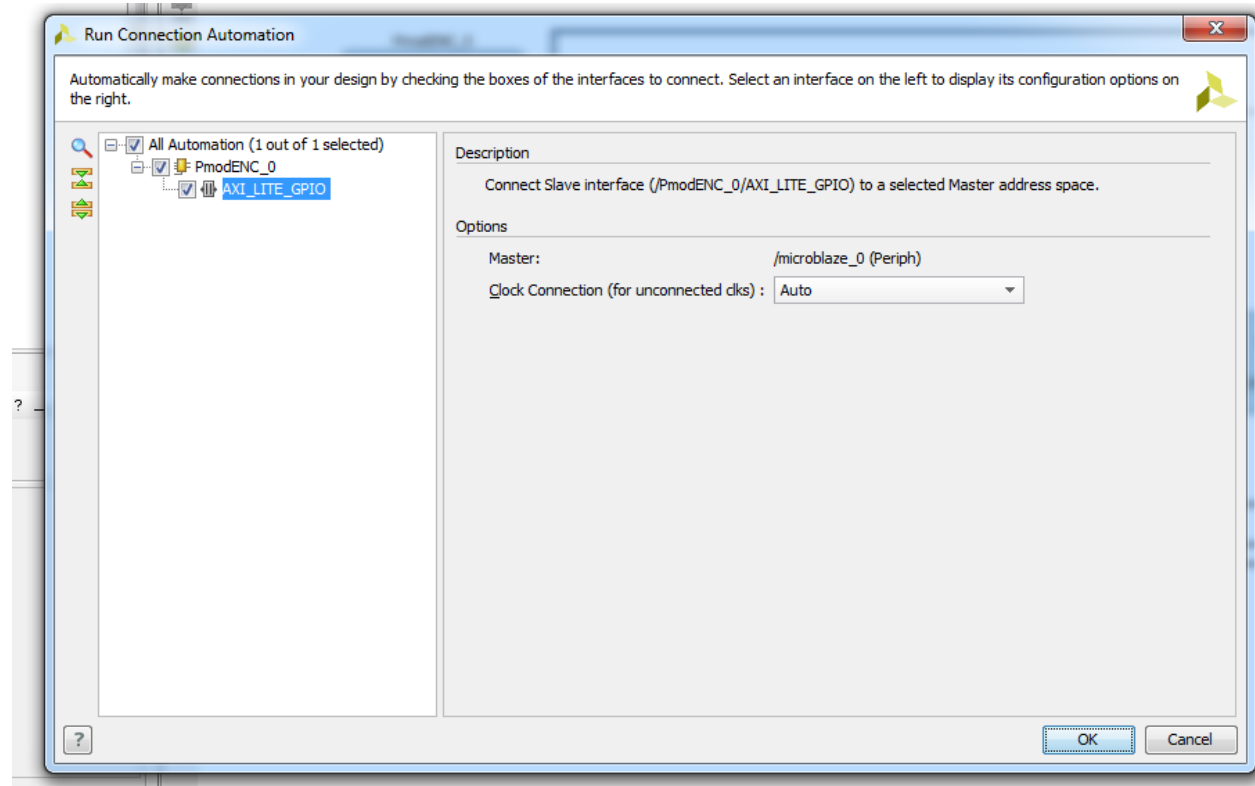
### Tip

Several of the simpler GPIO Pmods can be used with the PmodGPIO IP Core. To see if your Pmod is supported with this IP core consult the Pmod compatibility table found on the digilent web site at

<https://reference.digilentinc.com/learn/programmable-logic/tutorials/pmod-ips/start>

## 8. Run Connection Automation

8.1) Click **Run Connection Automation** then check the box next to the name of your Pmod IP core and click **OK**.



## 9. Connect Reference Clocks

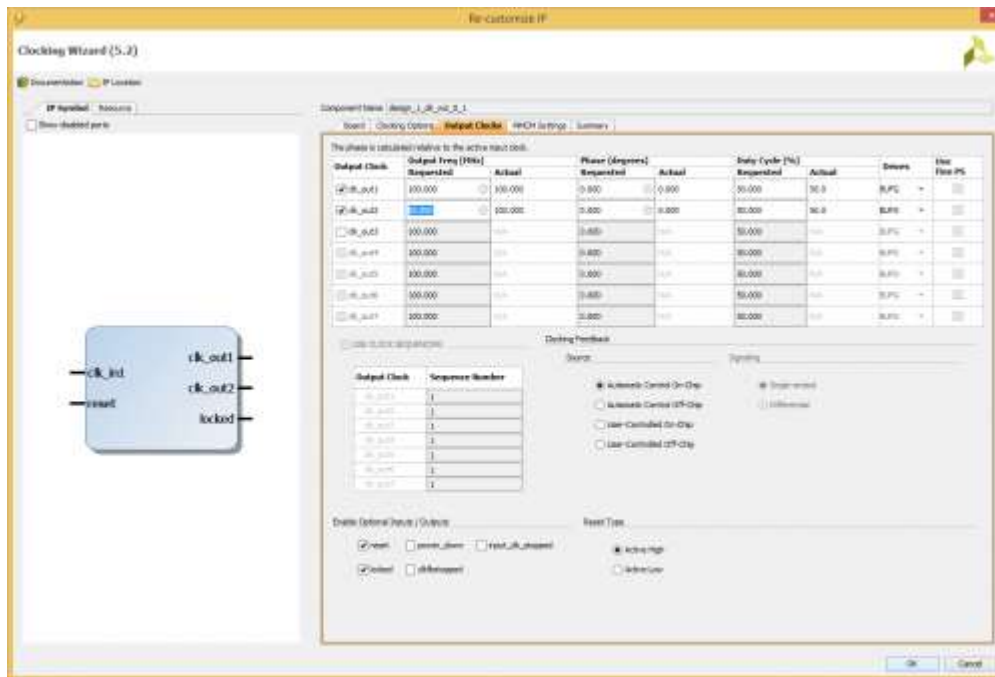
### Important

Some Pmod IP cores require a reference clock to function properly. To see if your Pmod requires a reference clock consult the Pmod compatibility table found at <https://reference.digilentinc.com/learn/programmable-logic/tutorials/pmod-ips/start>.

If your Pmod does not require a reference clock (Note: the rotary encoder we're using for this tutorial does not require a reference clock) then skip to [10](#).

9.1) Double click the clocking wizard IP block to re-customize it. In the customization dialog, select the Output Clocks tab. Check the next clock that is not already checked to activate it. Set the requested output frequency to the frequency required for your Pmod. This frequency can be found in the Pmod compatibility chart. Click **OK**.

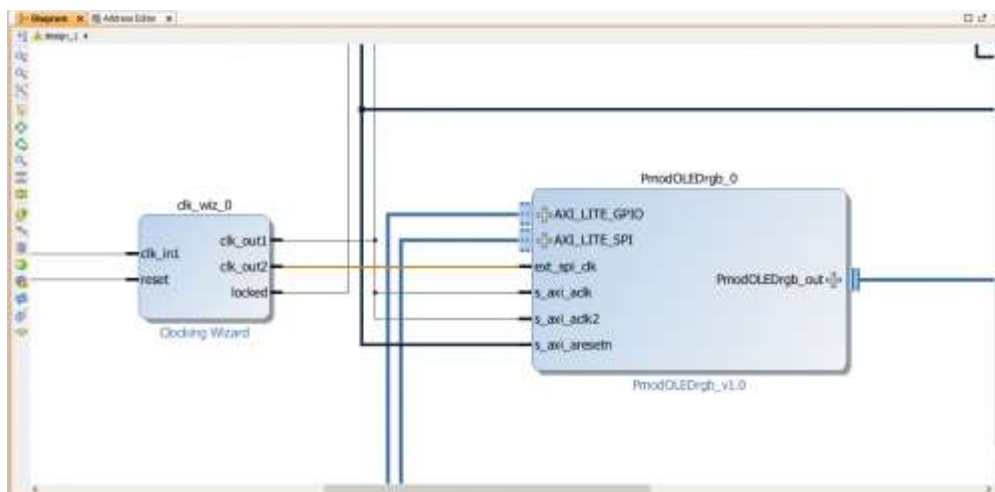




## Tip

If one of the other clocks is already being used and has a frequency that matches the frequency needed for your Pmod then you may use that clock. It is possible to connect a single clock to multiple destinations.

9.2) Connect this new clock to the clock input on your Pmod IP Core. The name of the clock input for your Pmod IP core can be found in the Pmod compatibility chart.

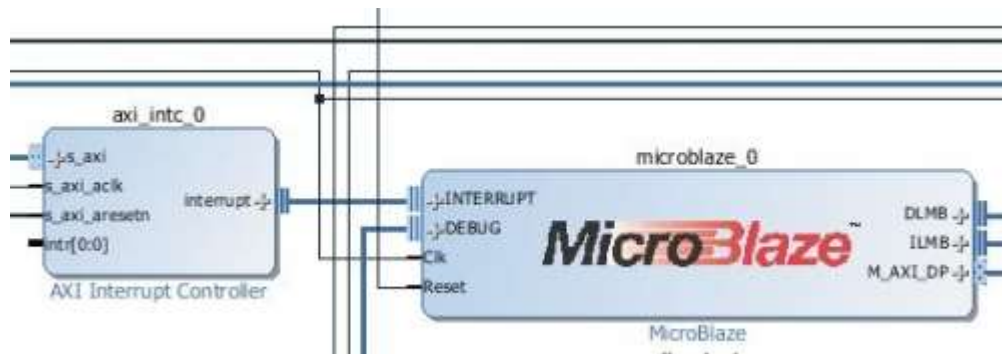


## 10. Connect Interrupts

### Important

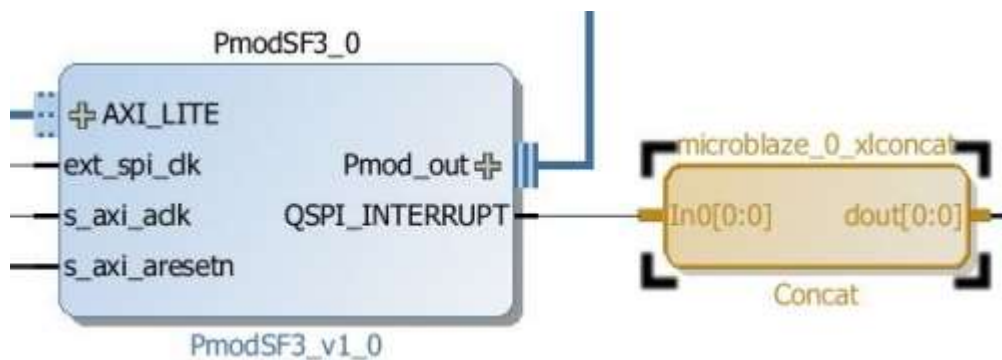
Some Pmod IP cores require an interrupt to function properly. To see if your Pmod requires an interrupt consult the Pmod compatibility table (note: the rotary encoder does not require an interrupt). If your Pmod does not require an interrupt then skip to [11](#).

10.1) If your MicroBlaze block doesn't have an AXI Interrupt Controller connected to its INTERRUPT port, add an AXI Interrupt controller to your block design. Manually connect the interrupt port of the axi\_intc\_0 to the INTERRUPT port of your MicroBlaze block. Use **Connection Automation** to connect its AXI interface to the rest of the system.




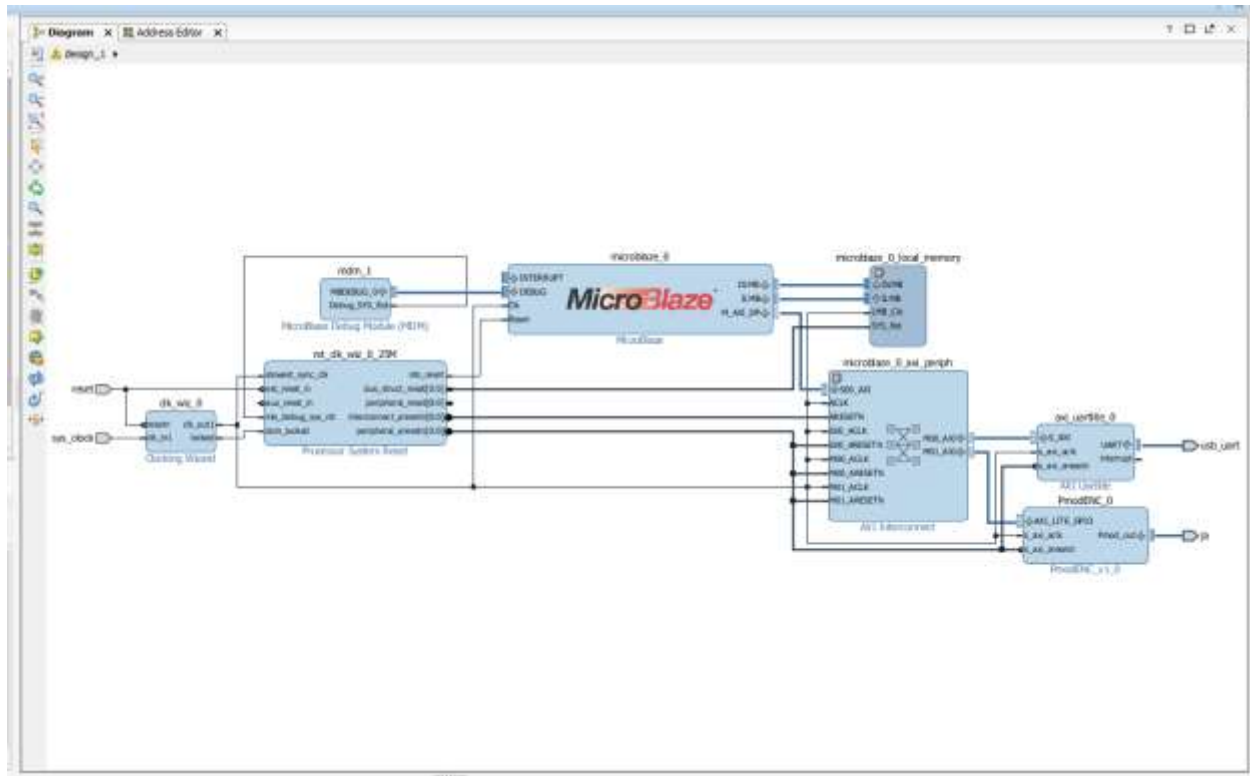
10.2) Add a Concat IP core to the block design. Re-customize the concat block to make sure that the number of inputs matches the number of interrupts you need to connect to your Microblaze Processor - probably only one. Click **Ok**.


10.3) Connect your Pmod IP core's interrupt port to the input port to the concat block, and the concat's output port to the *intr* port of your AXI Interrupt Controller.



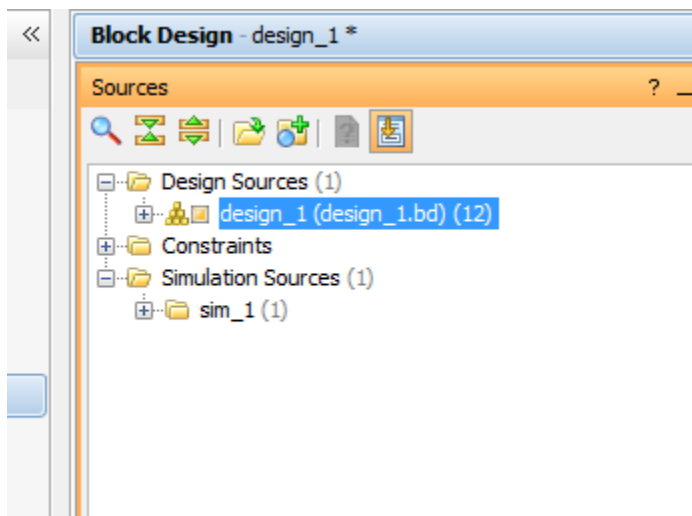
## 11. Validate the Design

11.1) Click the  **Regenerate Layout** button to rearrange your block design.

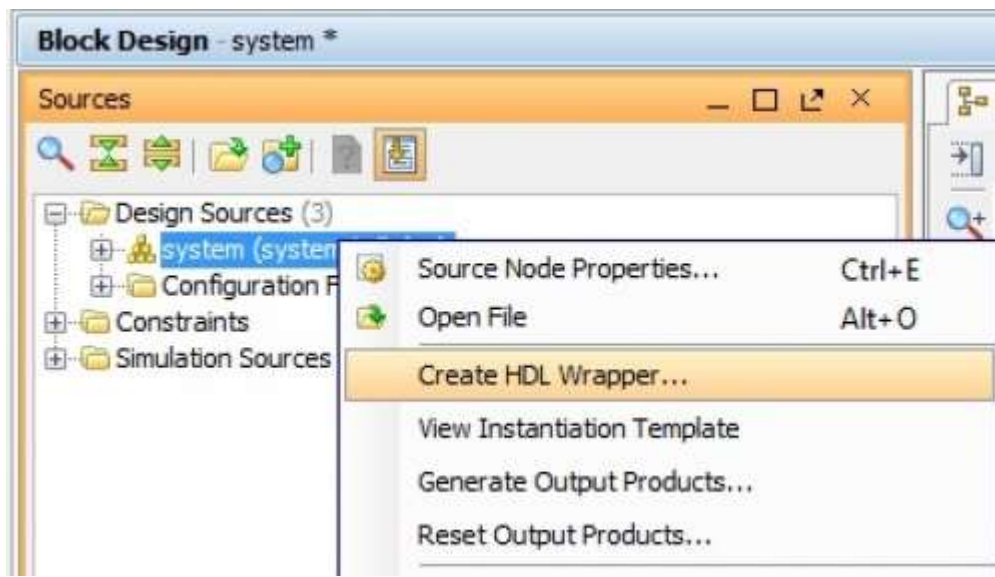


11.2) Select  **Validate Design**. This will check for design and connection errors.

11.3) After the design validation step we will proceed with creating a HDL System Wrapper. Click on the **Sources** tab and find your block design.




11.4) Right click on your block design and click **Create HDL Wrapper**. Let Vivado manage the wrapper and automatically update it and click **OK**.



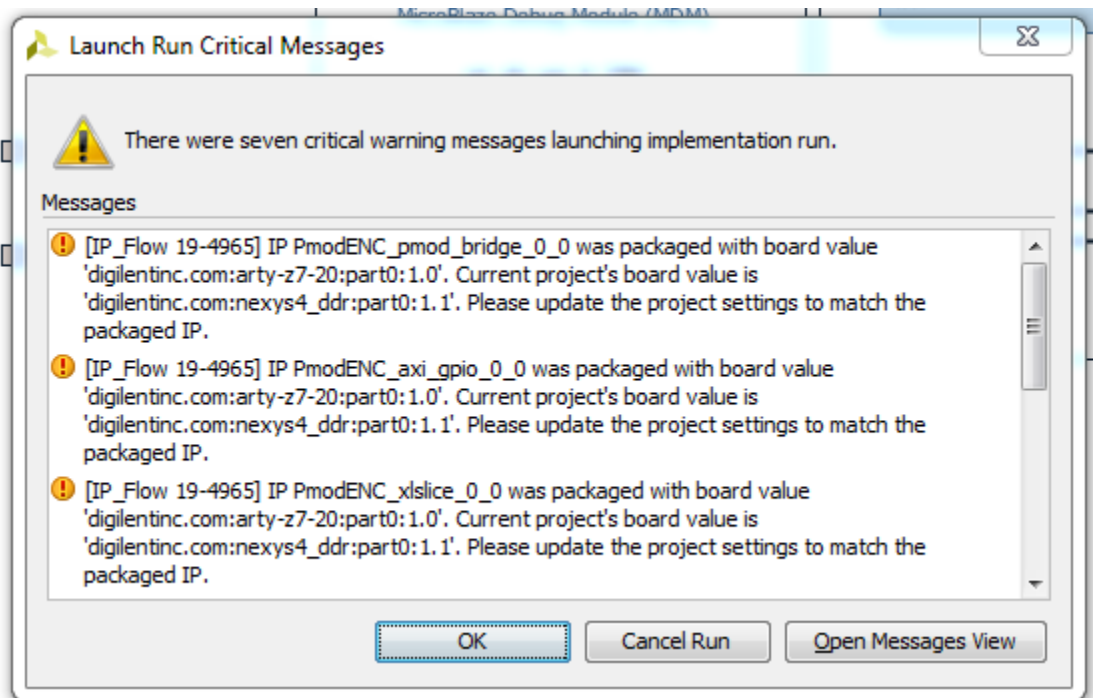
This will create a top module in verilog and will allow you to generate a bitstream.

## 12. Generate the Bit File

12.1) In the top toolbar in Vivado, click  **Generate Bitstream**. This can also be found in the *Flow Navigator* panel on the left, under *Program and Debug*. If you haven't already saved your design, you will get a prompt to save the block design.

12.2) The bit file generation will begin. The tool will run *Synthesis* and *Implementation*. After both have been successfully completed, the bit file will be created. You will find a status bar of Synthesis and Implementation running on the top right corner of the project window.

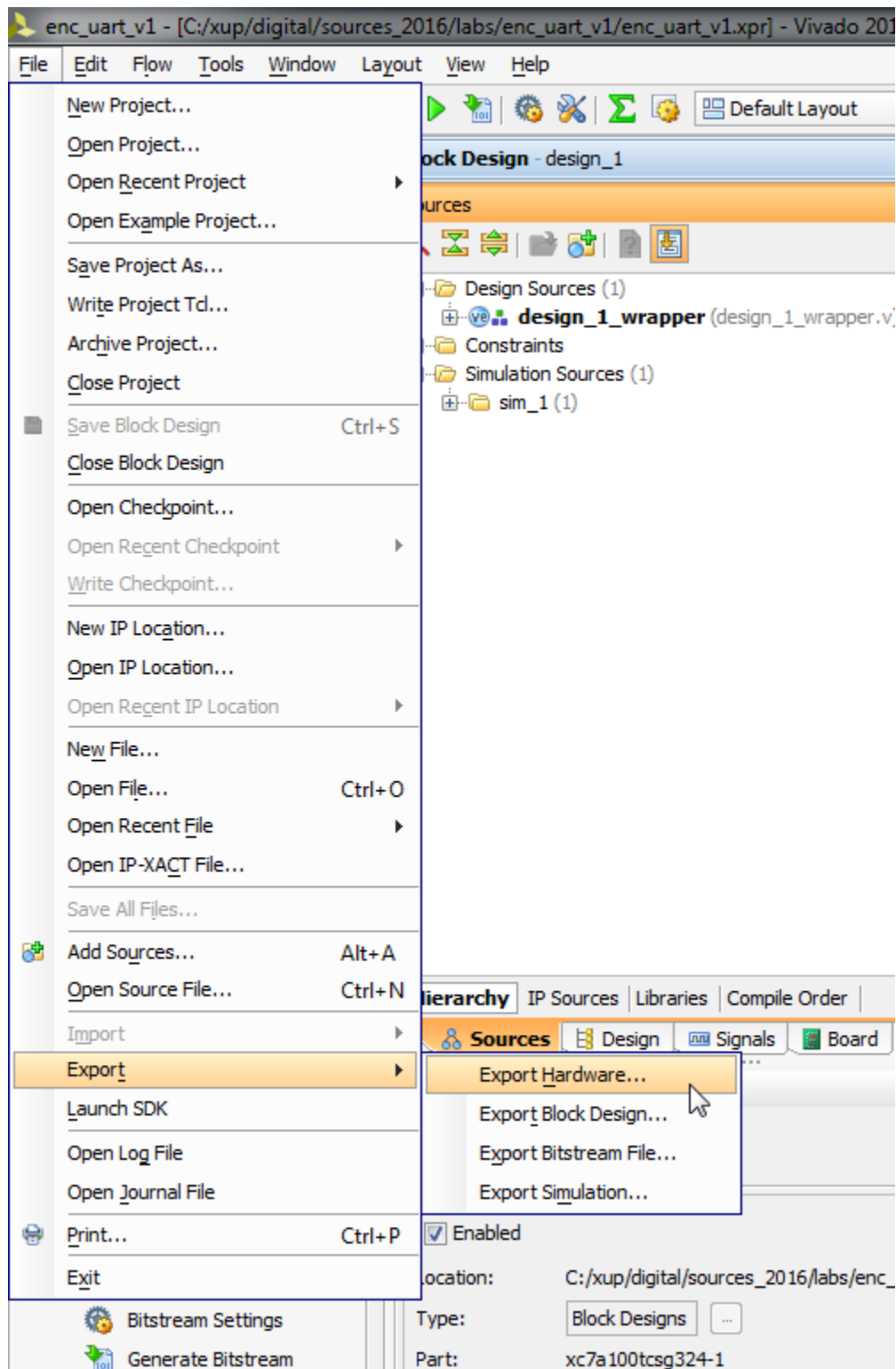
This process can take anywhere from **5 to 60 minutes** depending on your computer and target board. Note: you may get several critical warning about the board value (i.e. the pmod modules we're originally compiled with a different board). You can ignore theses and just click OK.



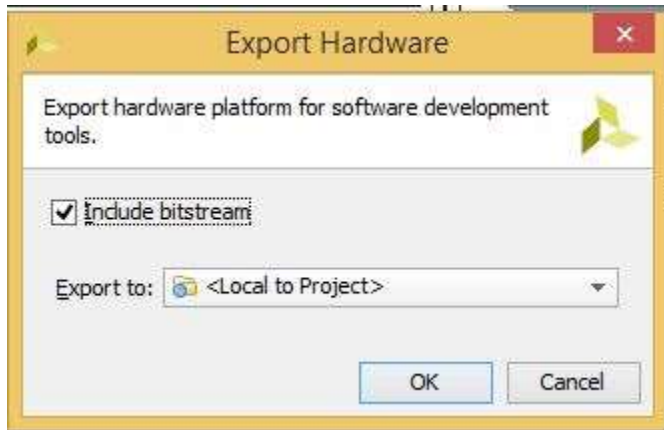
12.3) After the bitstream has been generated, a message prompt will pop-up on the screen. You don't have to open the Implemented Design for this tutorial. Just click **Cancel**.

## 13. Export the Hardware Design to SDK

13.1) On the main toolbar, click **File** and select **Export -> Export Hardware**. Check the box to **Include Bitstream** and click **OK**. This will export the hardware design with system wrapper for the Software Development Tool - Xilinx SDK.

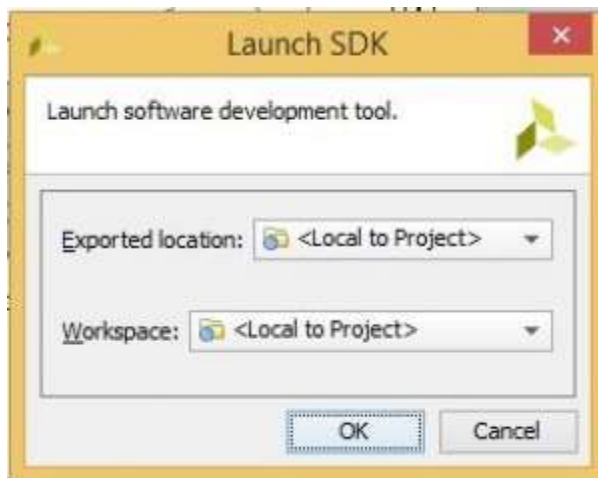






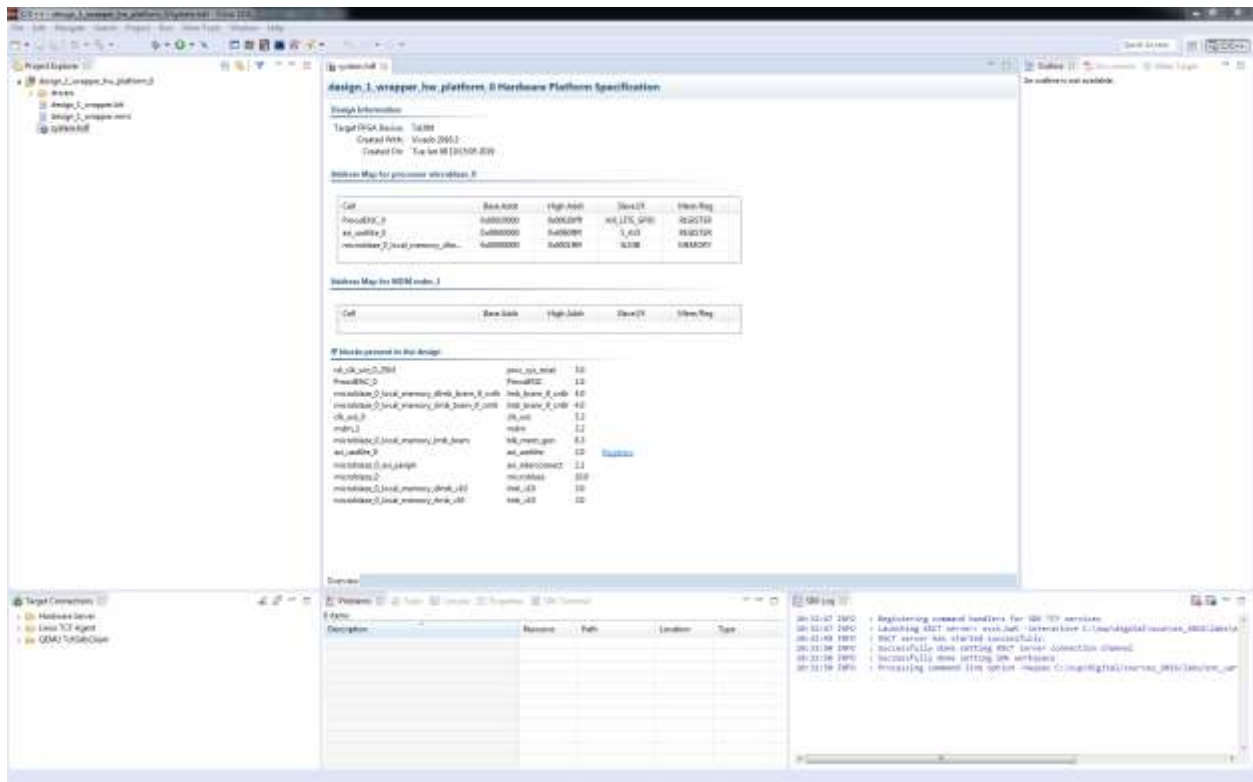
A new file directory will be created in your project directory under **project\_name.sdk** similar to the Vivado hardware design project name. A *.hdf* is also created. This step essentially creates a new SDK Workspace.

13.2) On the main toolbar, click **File** and then **Launch SDK**. Leave both of the dropdown menus as their default *Local to Project* and click **OK**. This will open Xilinx SDK and import your hardware.



## 14. Tour Xilinx SDK

The HW design specification and included IP blocks are displayed in the *system.hdf* file. Xilinx SDK is independent of Vivado, i.e. from this point, you can create your SW project in C/C++ on top of the exported HW design. If necessary, you can also launch SDK directly with the *.SDK* folder created in the main Vivado Project directory as the workspace. From this point, if you need to go back to Vivado and make changes to the HW design, then it is recommended to close the SDK window and make the required HW design edits in Vivado. After this you must follow the sequence of saving design, allowing Vivado to regenerate your HDL wrapper, and generating a new bit file. This new bit file must then be exported to SDK.



Within the *Project Explorer* tab on the left, you can see your hardware platform. **system** is the name of your block design created in Vivado. This hardware platform has all the HW design definitions, IP interfaces that have been added, external output signal information and local memory address information.

The drivers for the Pmod IP device can be found in the appropriate folder in the hardware platform, under **/drivers**. If you want to edit these drivers, use the versions found in your board support package under **libsrc**. If you do modify the drivers, keep in mind that any changes to your hardware will overwrite these changes, as well as any use of **Regenerate BSP sources**.

## 15. Create a New Application Project in SDK

15.1) Click the  **New** dropdown arrow and select **Application Project**.

**SDK New Project**

**Application Project**  
Create a managed make application project.

Project name:

☒ Use default location

Location:

Choose file system:

OS Platform:

**Target Hardware**

Hardware Platform:

Processor:

**Target Software**

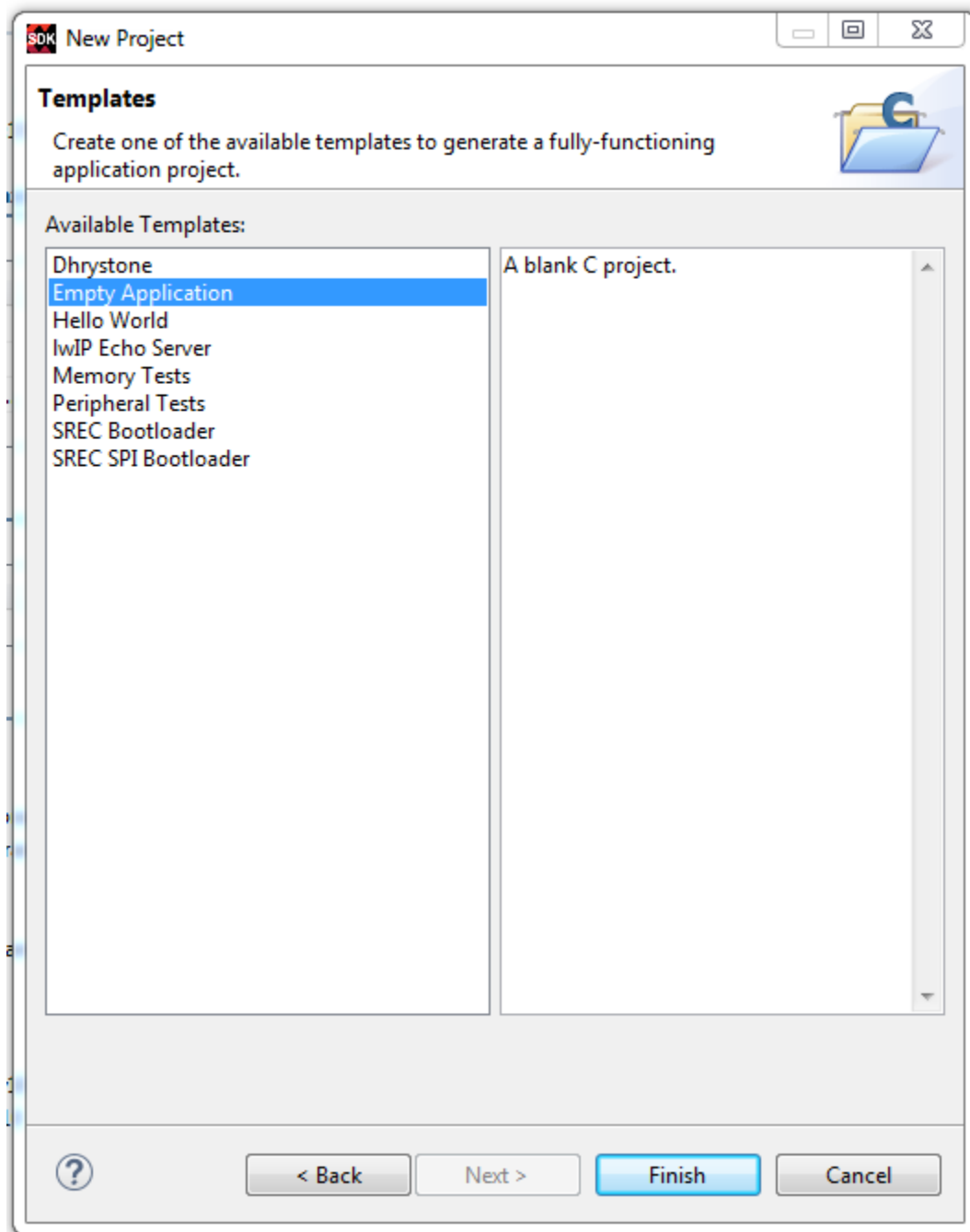
Language: ☒ C ☐ C++

Compiler:

Board Support Package: ☒ Create New  ☐ Use existing

Give your project a name that has no empty spaces and click **Next**.

15.2) Select **Empty Application** from the list of templates and click **Finish**.



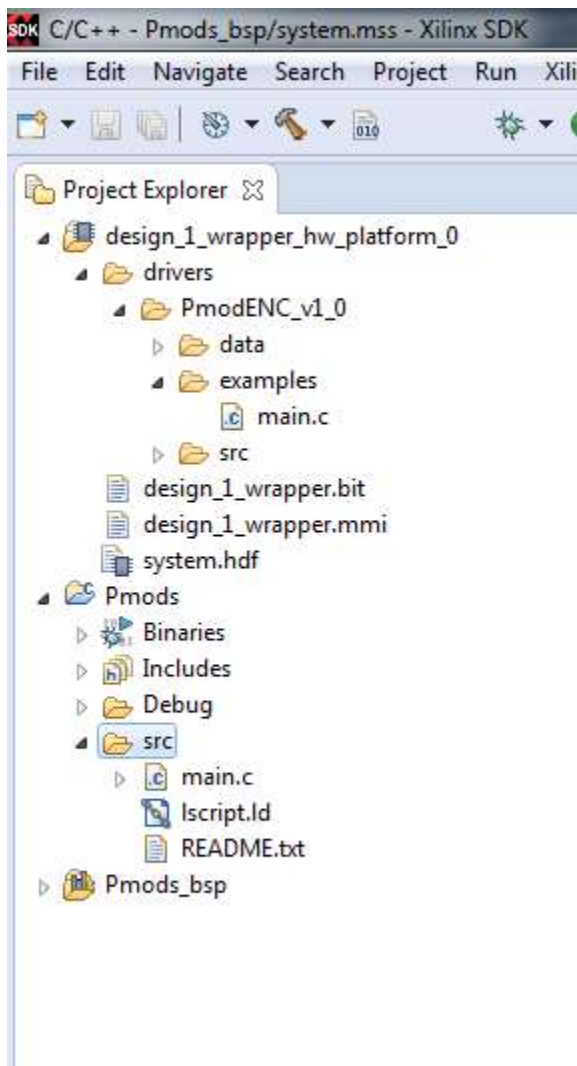
You will see two new folders in the **Project Explorer** panel.

- Your application project which contains all the binaries, .C and .H (Header) files
- The board support package for your project, which contains driver source files that your project may include


Our main working source folder also contains an important file shown here which is the “lscript.ld”. This is a Xilinx auto generated linker script file and includes information about memory addresses for different IP components of your block design, as well as the sizes of other memory regions.

## 16. Import the Example Project

16.1) Expand the **design\_1\_wrapper\_platform\_0** folder. Within the **drivers** folder, you will find the list of Pmods that you are using in your design. Expand the **Pmod.../examples** folder and copy all of the files found here into the **project\_name/src** folder.

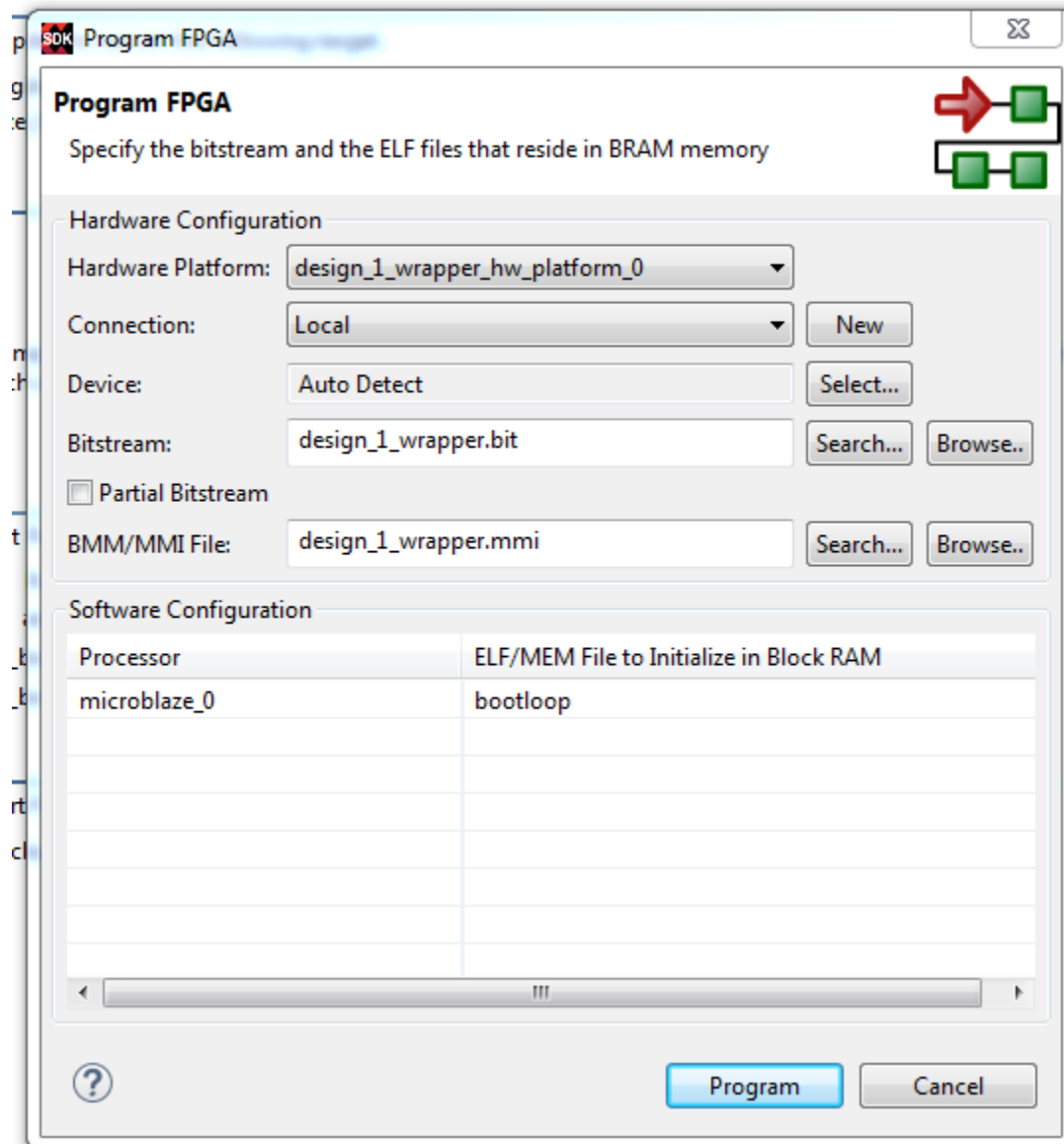


## 17. Program the FPGA with the Bit File

17.1) Make sure that your board is turned on and connected to the host PC with micro USB cables for UART and programming. On some boards, you will only need to connect a single PROG/UART port while on others you will need to connect your PC to two different ports typically named UART and PROG or JTAG (note: for the nexys4\_dds you only need to connect the PC to one port). Make sure the Pmod rotary encoder module is connected to the **TOP** row of the Pmod JA connector of the nexys4\_dds board. On the top toolbar, click the  **Program FPGA** button. Some boards will also

require that they be connected to a separate power source (note: the nexys4\_dds does not require a separate power source).

17.2) Click **Program** to program your FPGA with your hardware design. The green Done LED on the board should turn on when programming is completed.



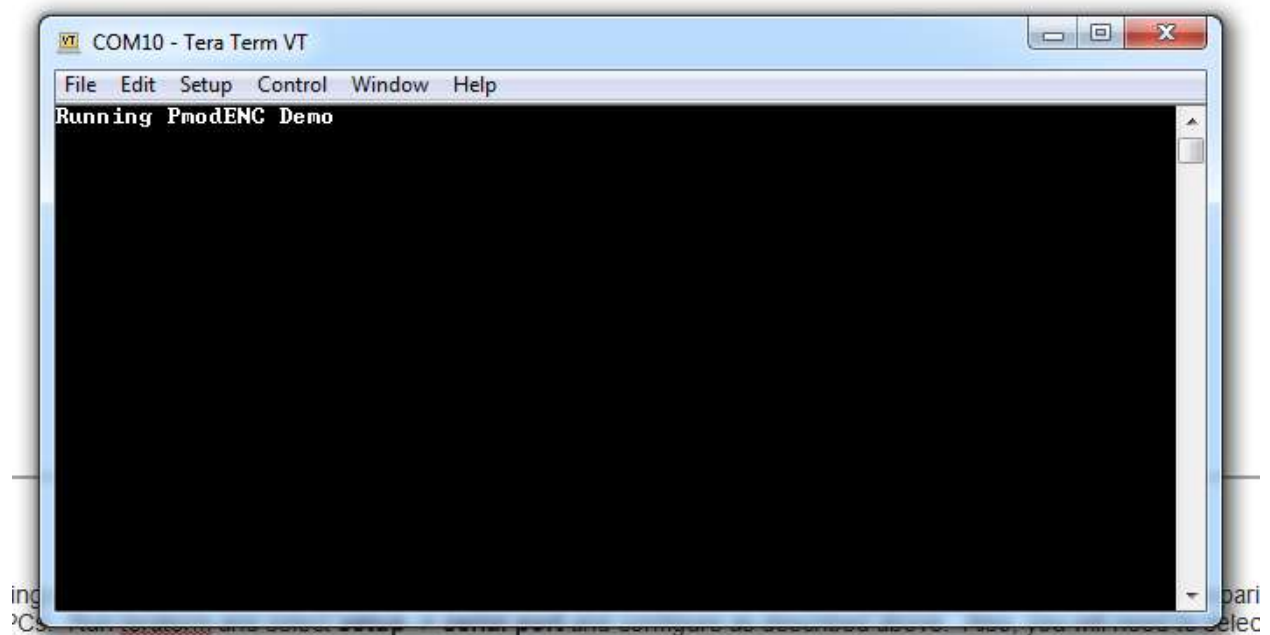
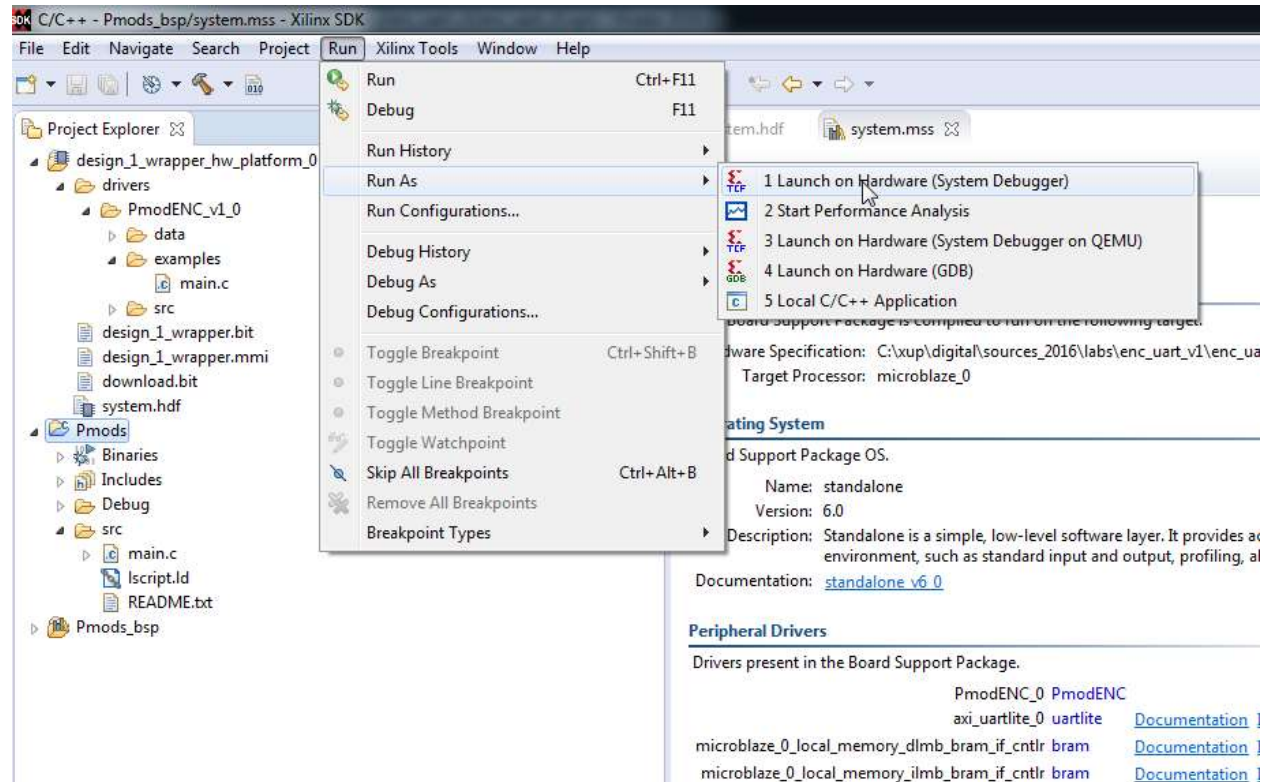
## 18. Program the Microblaze Processor

18.1) The majority of demos require that you use a serial terminal program on your PC to read messages printed by the demo. Settings for the terminal will vary depending on your board, but typically you will need to use a baud rate of 9600 for a microblaze design, 8 bit data, no parity bit, and one stop bit. (note: this depends on the configuration of Uartlite IP in Vivado). To interface to the PC serial terminal we will use the teraterm application installed on the lab PCs. Run teraterm and select **setup -> serial port** and configure as described above. Also, you will need to select the

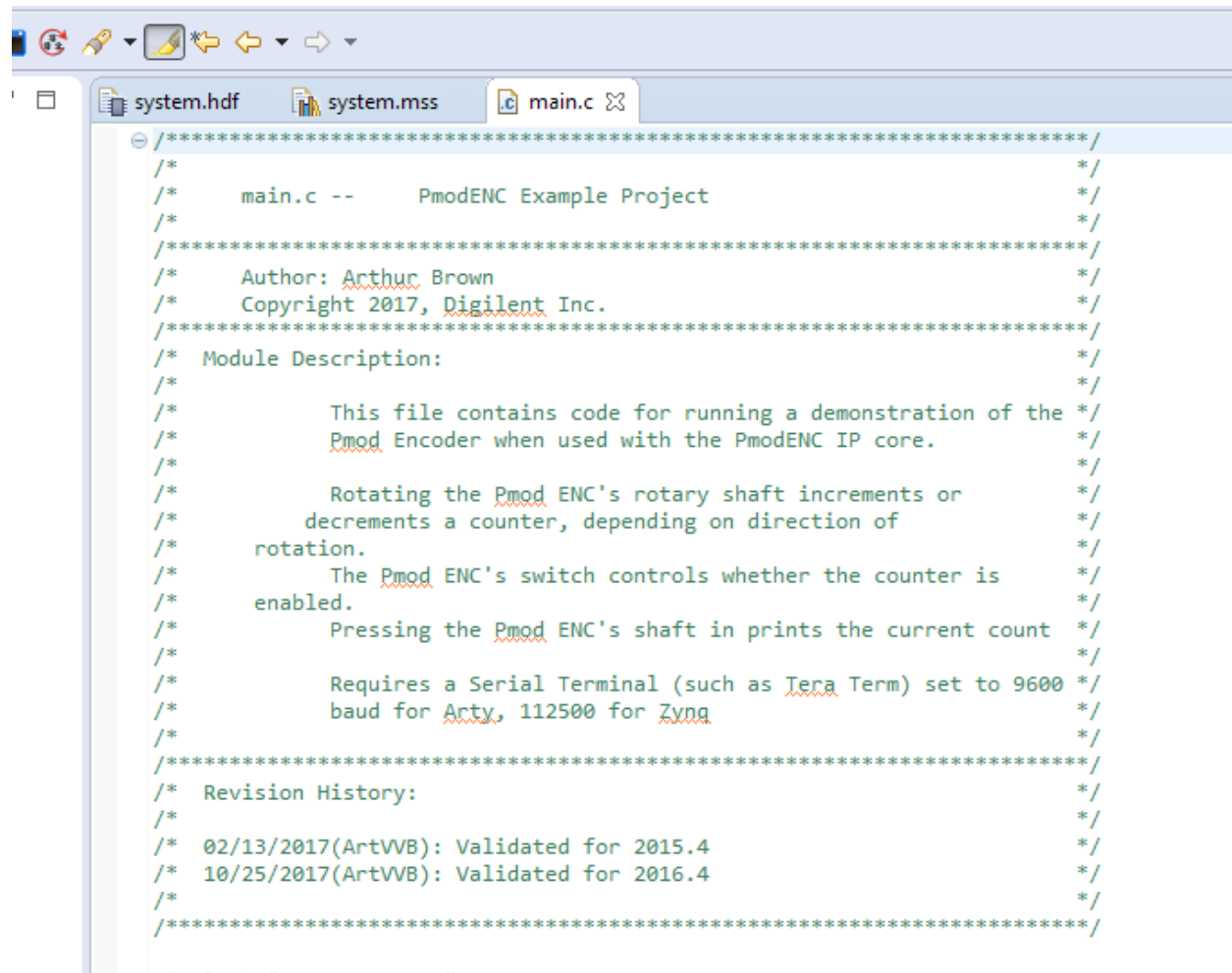


correct COM port for teraterm which can be found using the microsoft windows design manager to find out with COM port is connected to the nexys4\_ddr board.

18.2) Select your application project and click **Run -> Run As -> Launch on Hardware (System Debugger)**. You should see the “Running PmodENC demo” printed in your teraterm window.

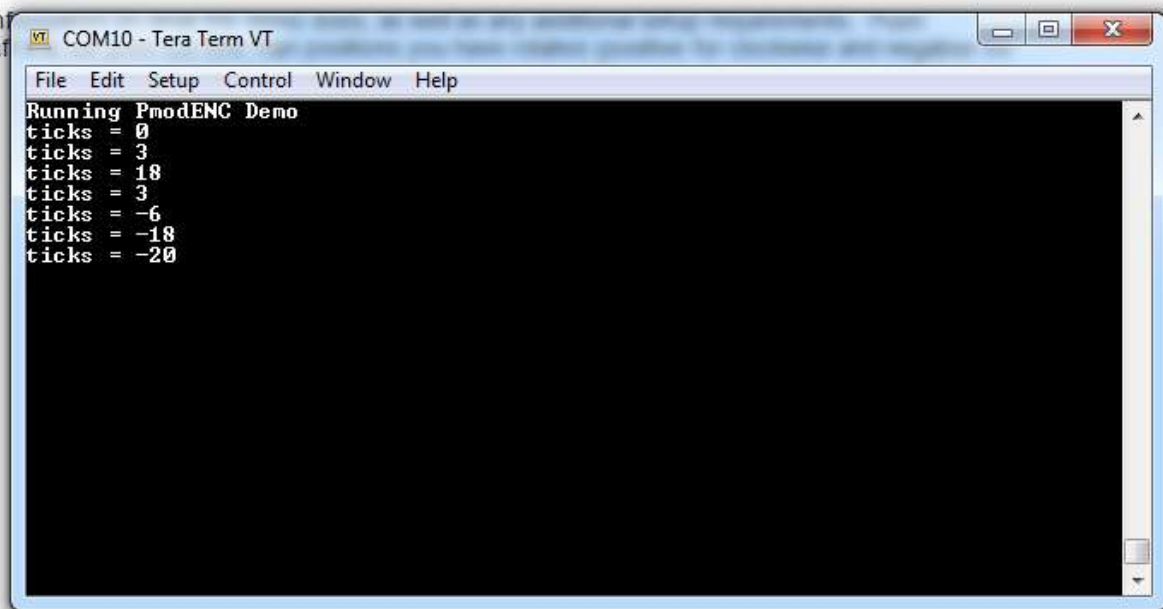


18.3) Xilinx SDK will run the program starting in main.c on your Microblaze/Zynq processor. Review the comment header at the top of the example main file to get more information on what the demo does, as well as any additional setup requirements. Push The rotary encoder switch button to the enable position (toward the “SW1” printed on the Pmod board). Rotate the rotary shaft. When you press down on the rotary shaft it will print out how many positions (i.e. ticks) you have rotated (positive for clockwise and negative for counter clockwise).



```
system.hdf system.mss main.c
/* **** */
/*
/*      main.c --      PmodENC Example Project
/*
/* **** */
/*      Author: Arthur Brown
/*      Copyright 2017, Digilent Inc.
/* **** */
/*      Module Description:
/*
/*      This file contains code for running a demonstration of the
/*      Pmod Encoder when used with the PmodENC IP core.
/*
/*      Rotating the Pmod ENC's rotary shaft increments or
/*      decrements a counter, depending on direction of
/*      rotation.
/*      The Pmod ENC's switch controls whether the counter is
/*      enabled.
/*      Pressing the Pmod ENC's shaft in prints the current count
/*
/*      Requires a Serial Terminal (such as Tera Term) set to 9600
/*      baud for Arty, 112500 for Zynq
/*
/* **** */
/*      Revision History:
/*
/*      02/13/2017(ArtVVB): Validated for 2015.4
/*      10/25/2017(ArtVVB): Validated for 2016.4
/*
/* **** */
```

re in  
r shat



A screenshot of a Tera Term VT window titled "COM10 - Tera Term VT". The window has a menu bar with "File", "Edit", "Setup", "Control", "Window", and "Help". The main display area is black with white text. The text shows the output of a "PmodENC Demo", starting with "Running PmodENC Demo" followed by several lines of "ticks" values: 0, 3, 18, 3, -6, -18, and -20. The window has standard Windows-style window controls (minimize, maximize, close) in the top right corner.

```
COM10 - Tera Term VT
File Edit Setup Control Window Help
Running PmodENC Demo
ticks = 0
ticks = 3
ticks = 18
ticks = 3
ticks = -6
ticks = -18
ticks = -20
```

## 19. Celebrate!

You just created your own Pmod IP core design.

### 19.1. **Demonstrate the working circuit to the TA.**