

# Creating a custom IP block in Vivado

by Jeff Johnson | Aug 4, 2014 | Vivado | 13 comments



8 Votes

## Tutorial Overview

In this tutorial we'll create a custom AXI IP block in Vivado and modify its functionality by integrating custom VHDL code. We'll be using the Zynq SoC and the MicroZed as a hardware platform. For simplicity, our custom IP will be a multiplier which our processor will be able to access through register reads and writes over an AXI bus.

The multiplier takes in two 16 bit unsigned inputs and outputs one 32 bit unsigned output. A single 32 bit write to the IP will contain the two 16 bit inputs, separated by the lower and higher 16 bits. A single 32 bit read from the peripheral will contain the result from the multiplication of the two 16 bit inputs. The design doesn't serve much purpose, but it is a good example of integrating your own code into an AXI IP block.

## Requirements

Before following this tutorial, you will need to do the following:

- Vivado 2014.2
- MicroZed
- Platform Cable USB II (or equivalent JTAG programmer)

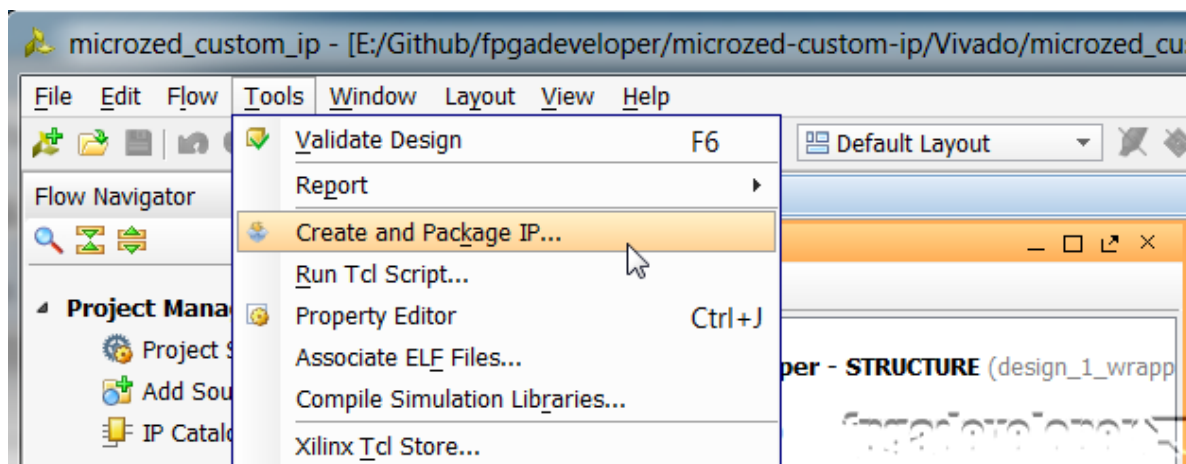
## Start from a base project

You can do this tutorial with any existing Vivado project, but I'll start with the base system project for the MicroZed that you can access here:

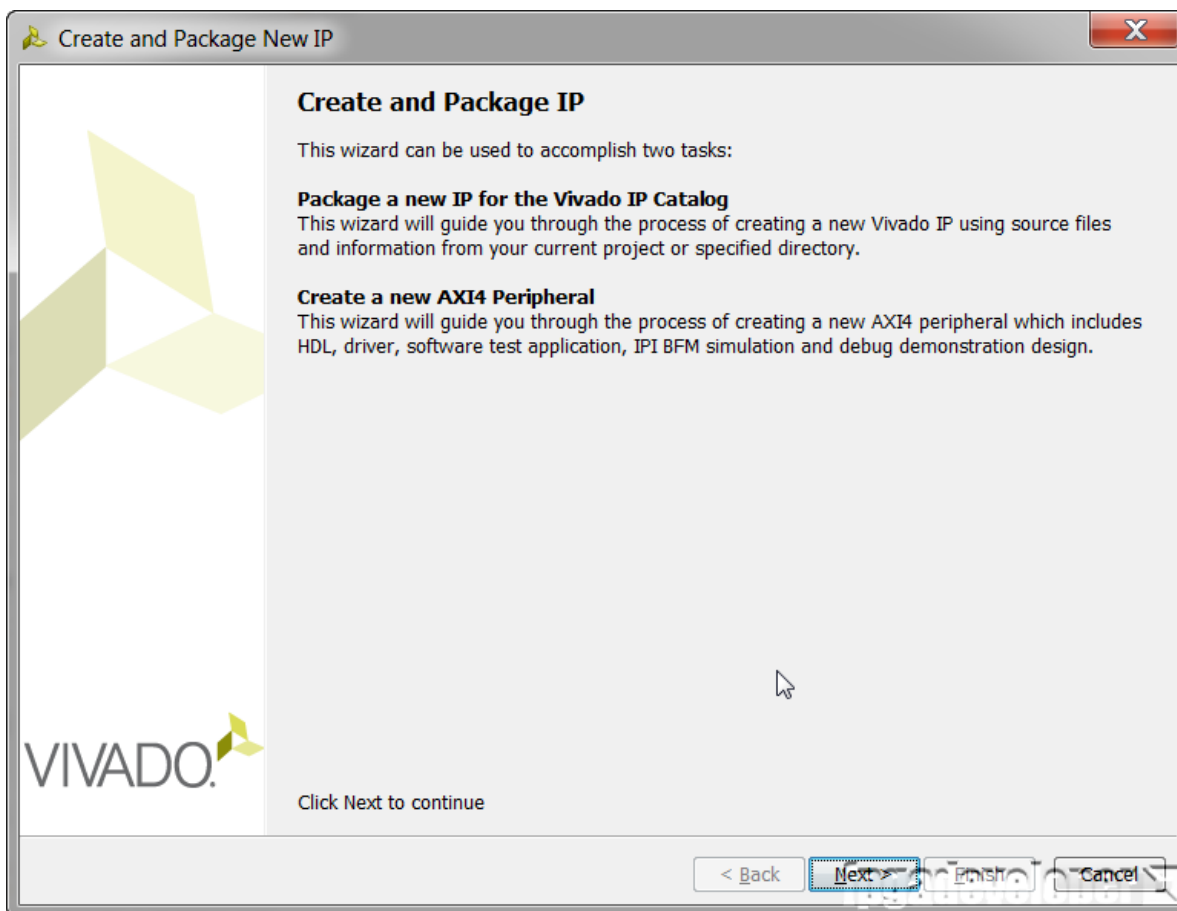
Base system project for the MicroZed

## Create the Custom IP

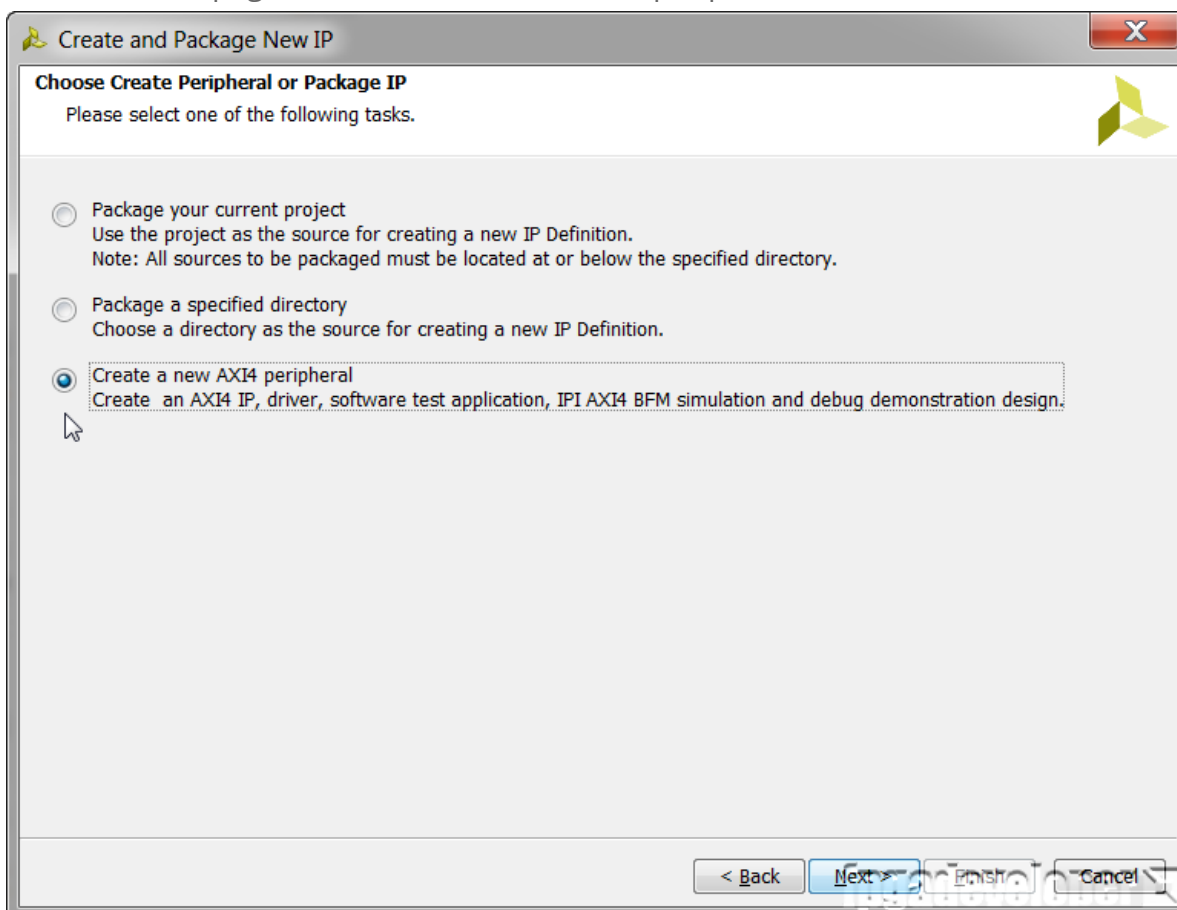
1. With the base Vivado project opened, from the menu select Tools->Create and package IP.



2. The Create and Package IP wizard opens. If you are used to the ISE/EDK tools you can think of this as being similar to the Create/Import Peripheral wizard. Click "Next".

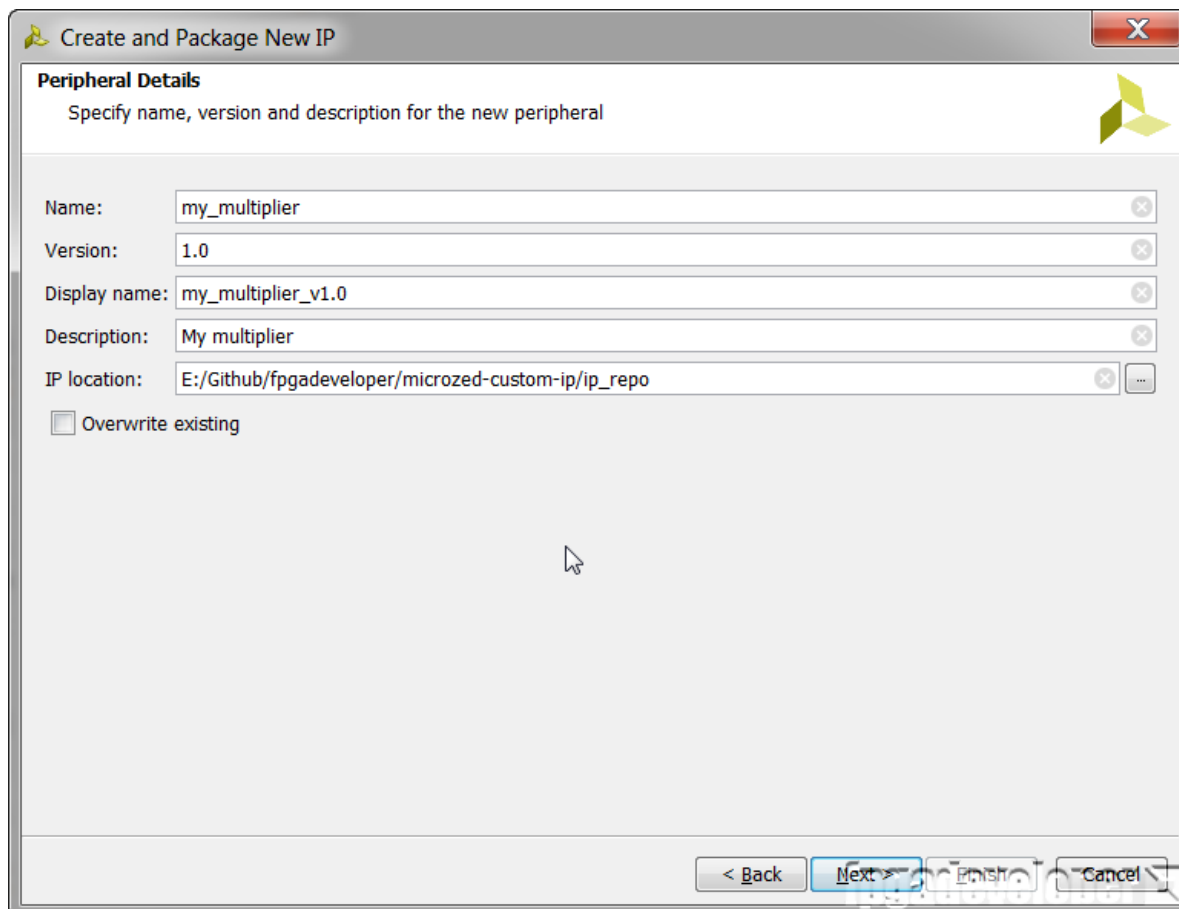


3. On the next page, select "Create a new AXI4 peripheral". Click "Next".



4. Now you can give the peripheral an appropriate name, description and location.

Click "Next".



**Create and Package New IP**

**Peripheral Details**  
Specify name, version and description for the new peripheral

Name:

Version:

Display name:

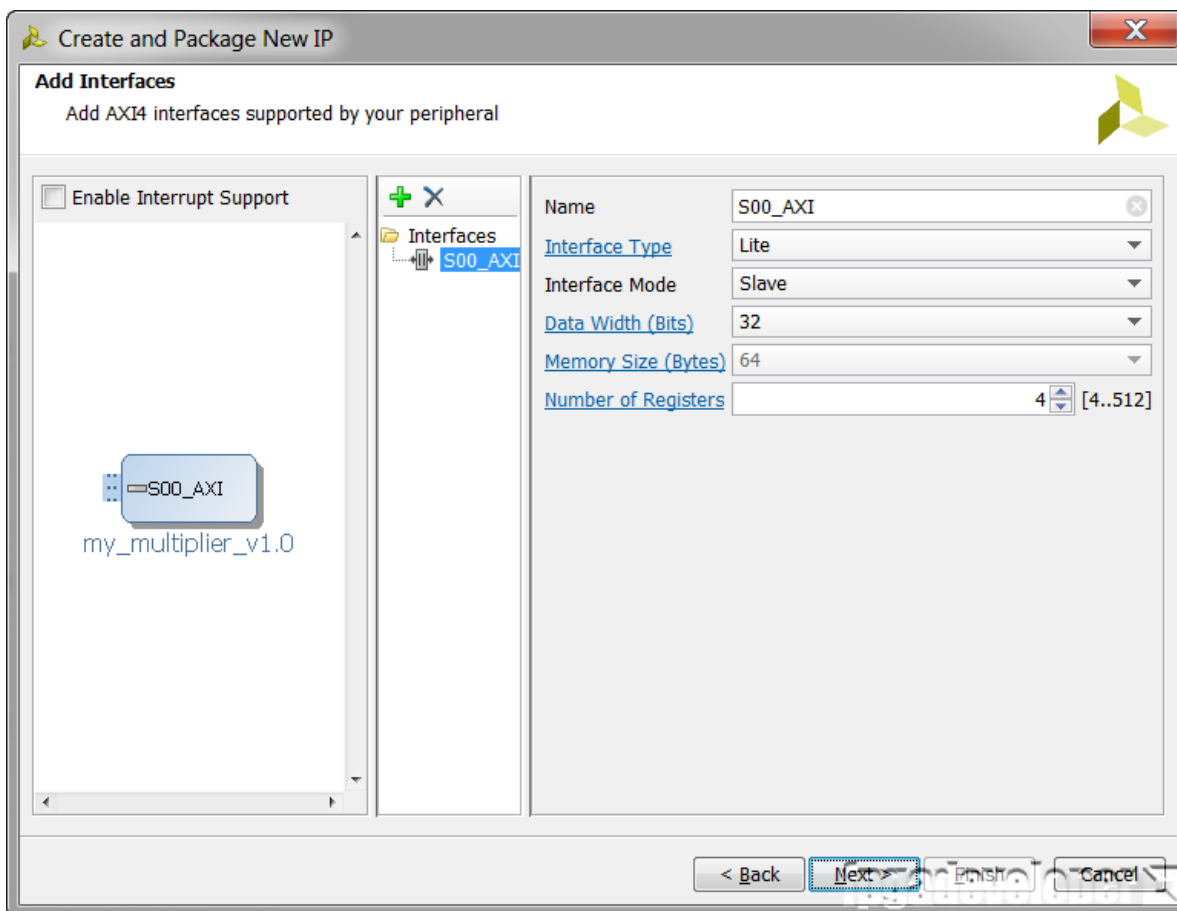
Description:

IP location:

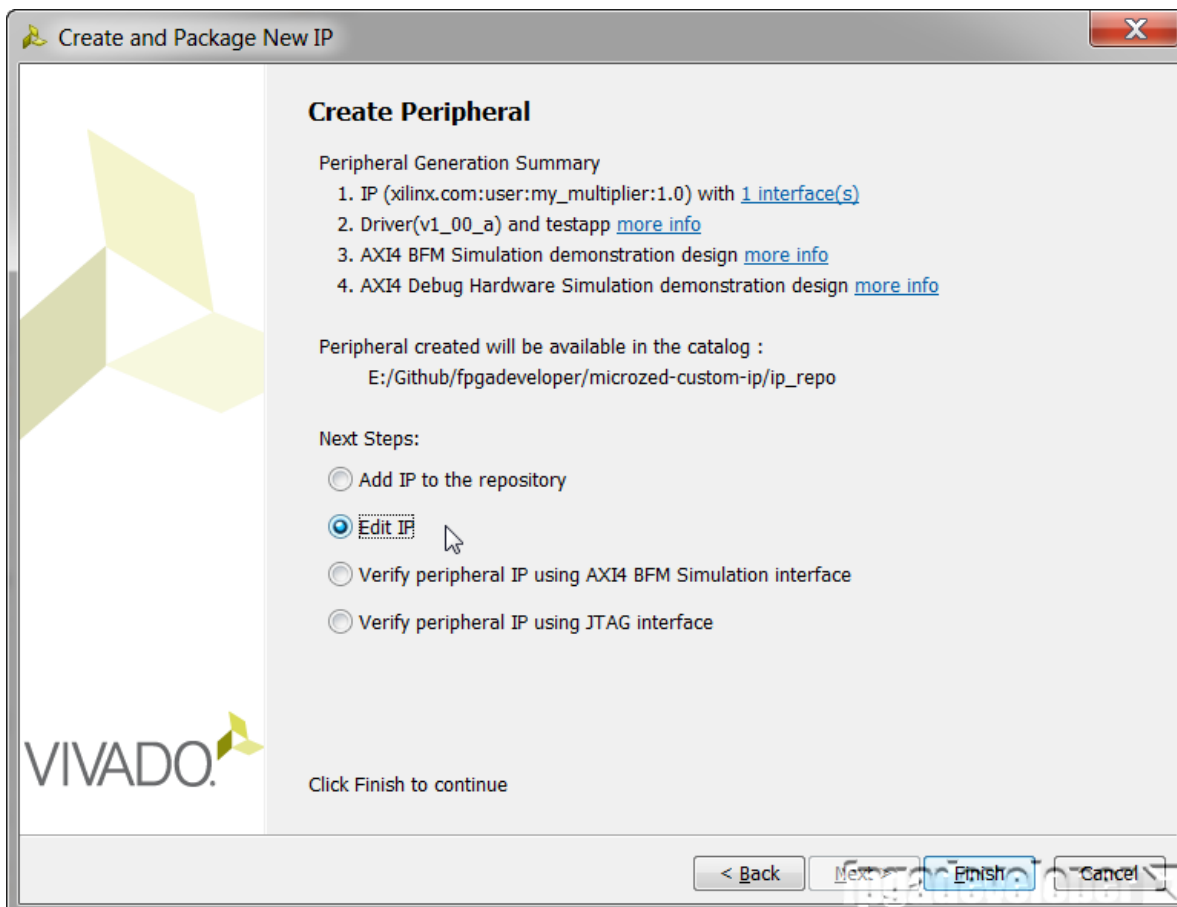
☐ Overwrite existing

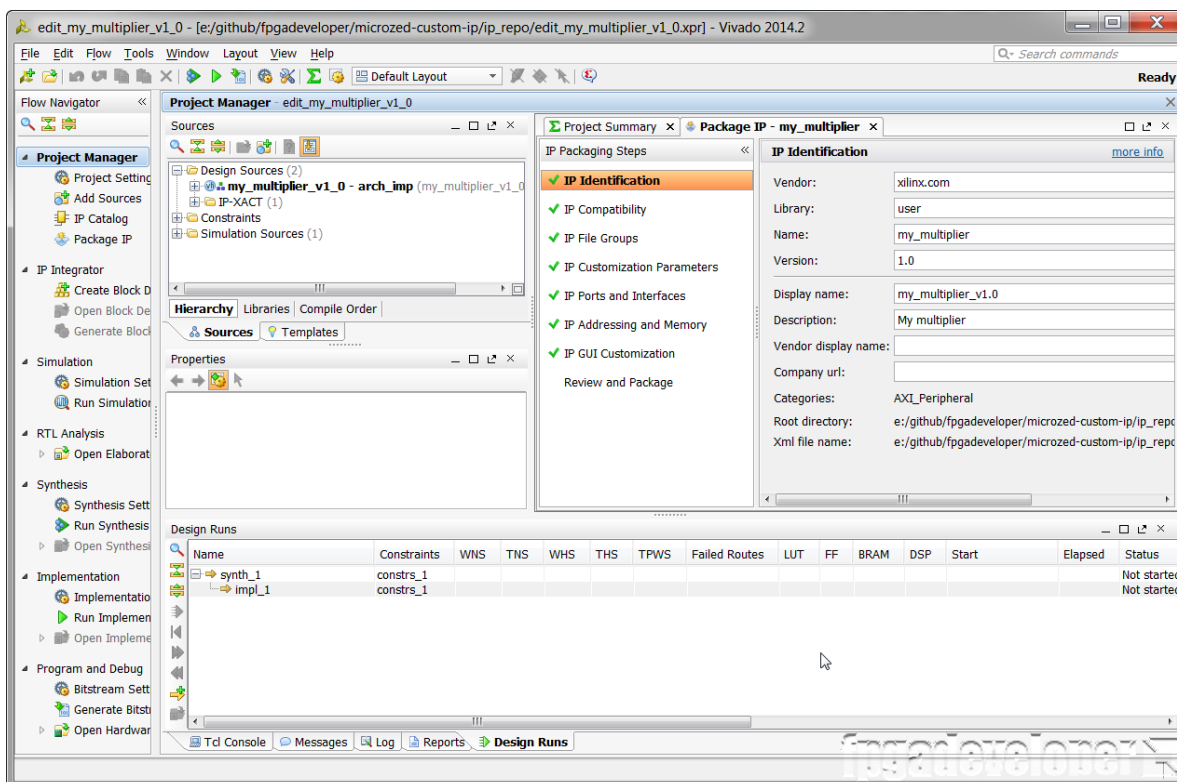
< Back Next > Finish Cancel

5. On the next page we can configure the AXI bus interface. For the multiplier we'll use AXI lite, and it'll be a slave to the PS, so we'll stick with the default values.



6. On the last page, select "Edit IP" and click "Finish". Another Vivado window will open which will allow you to modify the peripheral that we just created.





At this point, the peripheral that has been generated by Vivado is an AXI lite slave that contains 4 x 32 bit read/write registers. We want to add our multiplier code to the IP and modify it so that one of the registers connects to the multiplier inputs and another to the multiplier output.

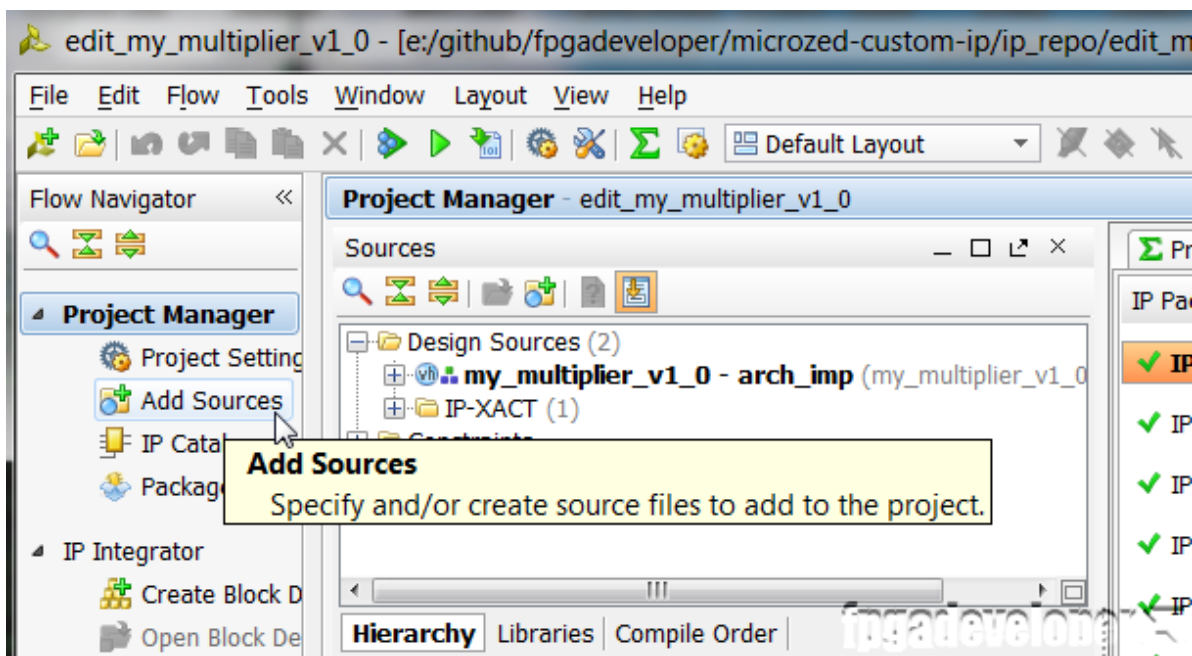
## Add the multiplier code to the peripheral

You can find the multiplier code on Github at the link below. Download the “multiplier.vhd” file and save it somewhere, the location is not important for now.

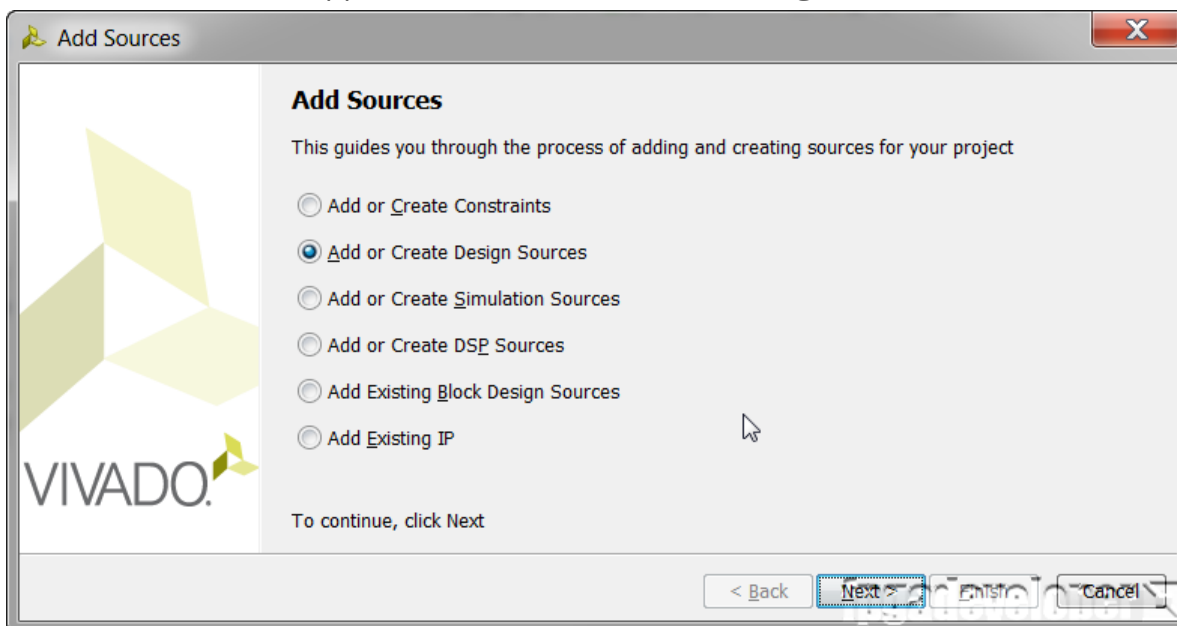
[https://github.com/fpgadeveloper/microzed-custom-ip/blob/master/ip\\_repo/my\\_multiplier\\_1.0/src/multiplier.vhd](https://github.com/fpgadeveloper/microzed-custom-ip/blob/master/ip_repo/my_multiplier_1.0/src/multiplier.vhd)

Note that these steps must be done in the Vivado window that contains the peripheral we just created (not the base project).

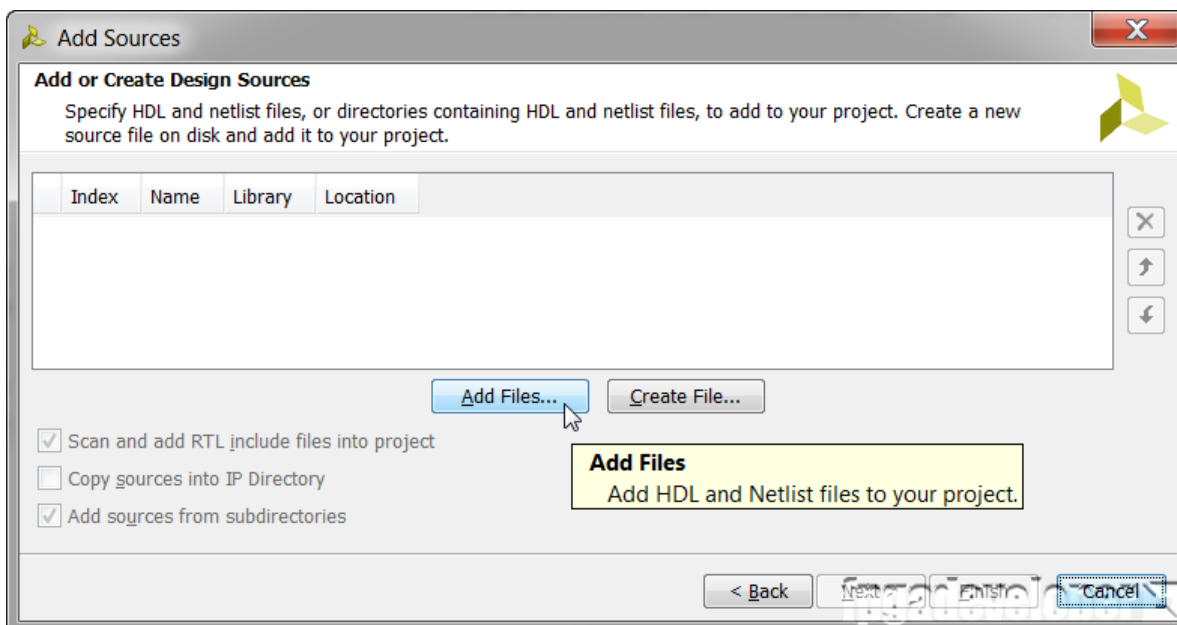
1. From the Flow Navigator, click “Add Sources”.



2. In the window that appears, select "Add or Create Design Sources" and click "Next".

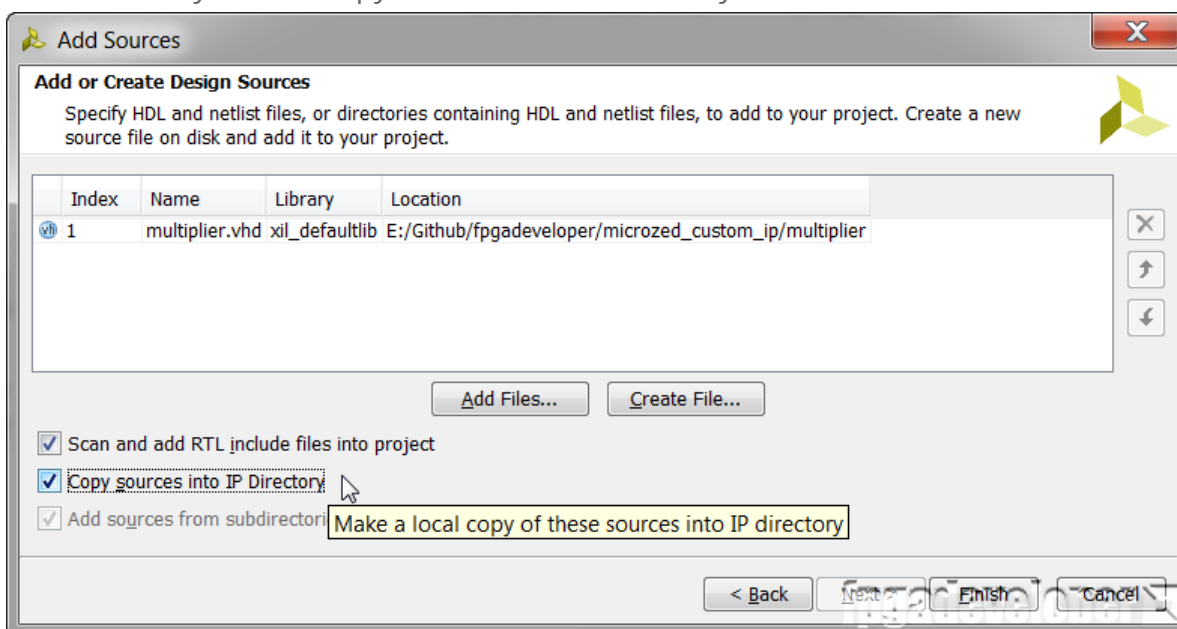


3. On the next window, click "Add Files".



4. Browse to the “multiplier.vhd” file, select it and click “OK”.

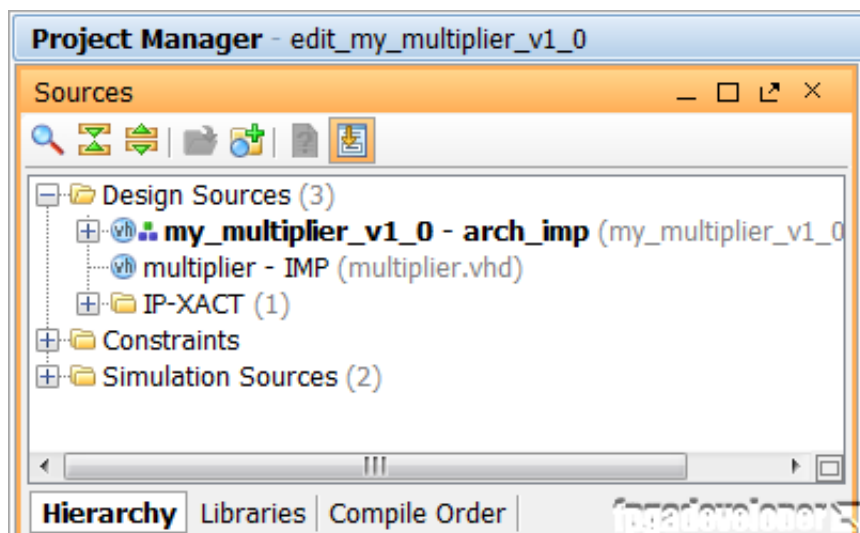
5. Make sure you tick “Copy sources into IP directory” and then click “Finish”.



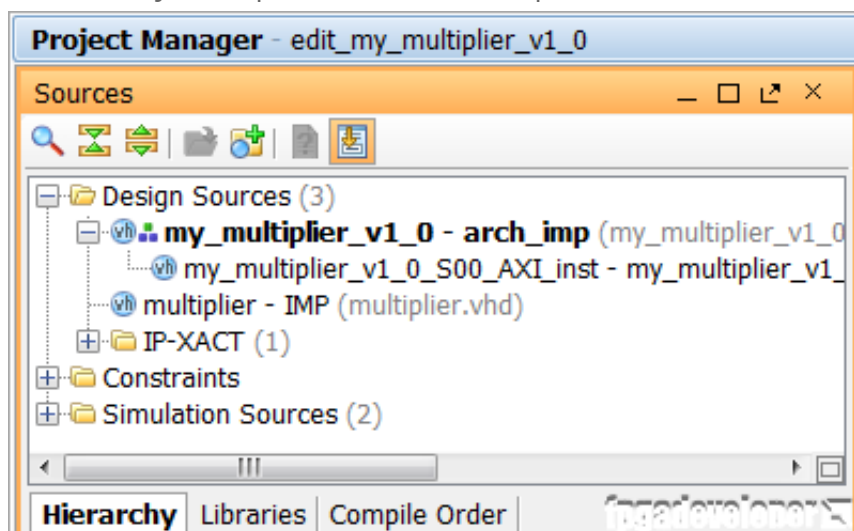
The multiplier code is now added to the peripheral, however we still have to instantiate it and connect it to the registers.

## Modify the Peripheral

At this point, your Project Manager Sources window should look like the following:



1. Open the branch "my\_multiplier\_v1\_0 - arch\_imp".



2. Double click on the "my\_multiplier\_v1\_0\_S00\_AXI\_inst" file to open it.
3. The source file should be open in Vivado. Find the line with the "begin" keyword and add the following code just above it to declare the multiplier and the output signal:

```

1. signal multiplier_out : std_logic_vector(31 downto 0);
2.
3. component multiplier
4. port (
5.   clk: in std_logic;
6.   a: in std_logic_VECTOR(15 downto 0);
7.   b: in std_logic_VECTOR(15 downto 0);
8.   p: out std_logic_VECTOR(31 downto 0));
9. end component;
```

4. Now find the line that says "- Add user logic here" and add the following code

below it to instantiate the multiplier:

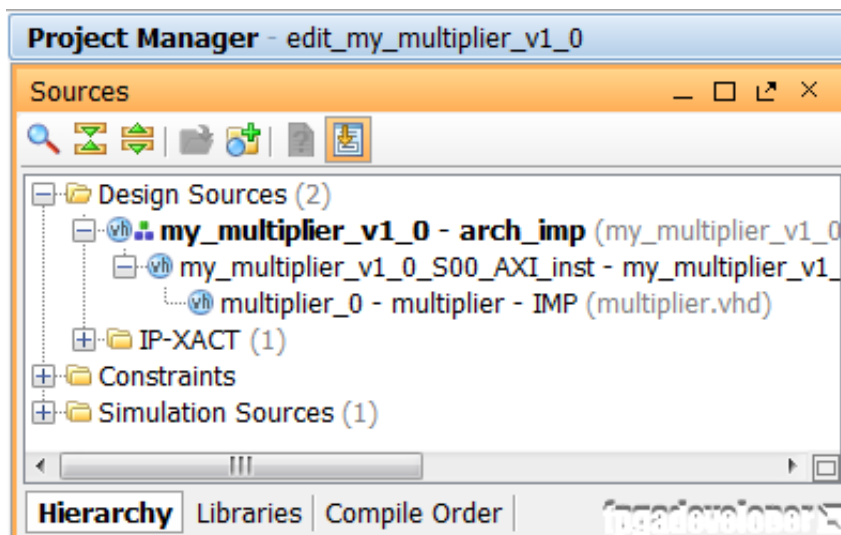
```
1. multiplier_0 : multiplier
2. port map (
3.   clk => S_AXI_ACLK,
4.   a => slv_reg0(31 downto 16),
5.   b => slv_reg0(15 downto 0),
6.   p => multiplier_out);
```

5. Find this line of code “reg\_data\_out <= slv\_reg1;” and replace it with “reg\_data\_out <= multiplier\_out;”.

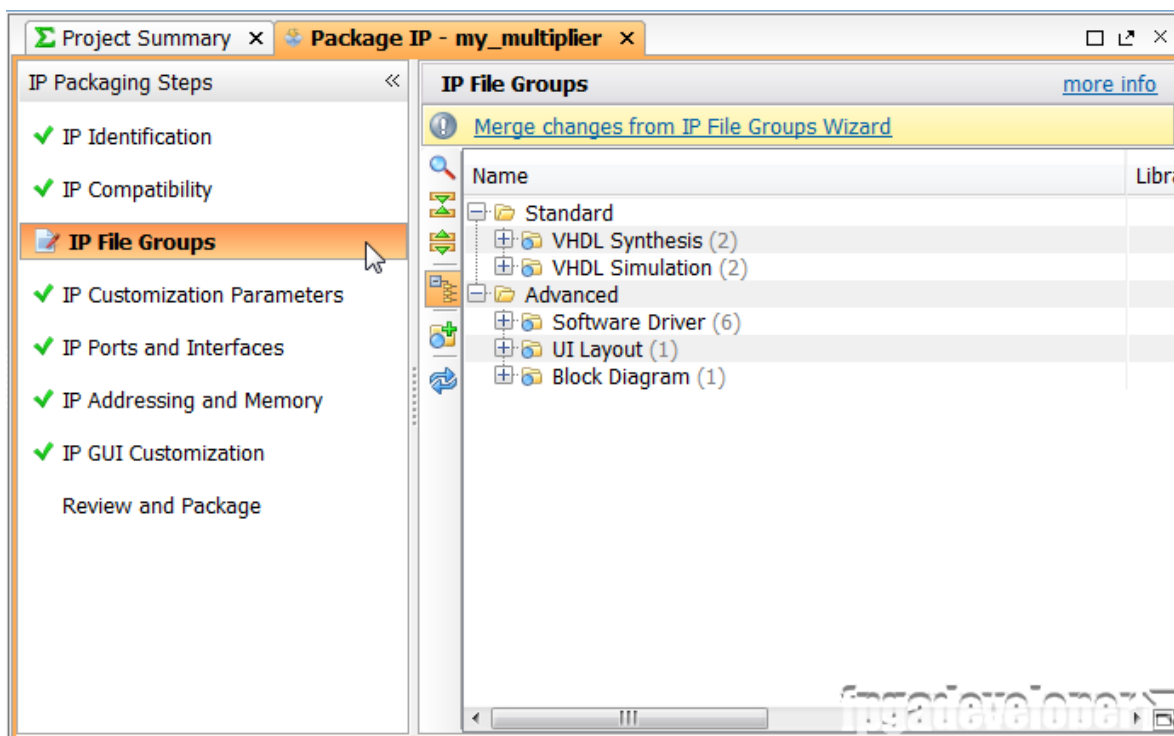
6. In the process statement just a few lines above, replace “slv\_reg1” with “multiplier\_out”.

7. Save the file.

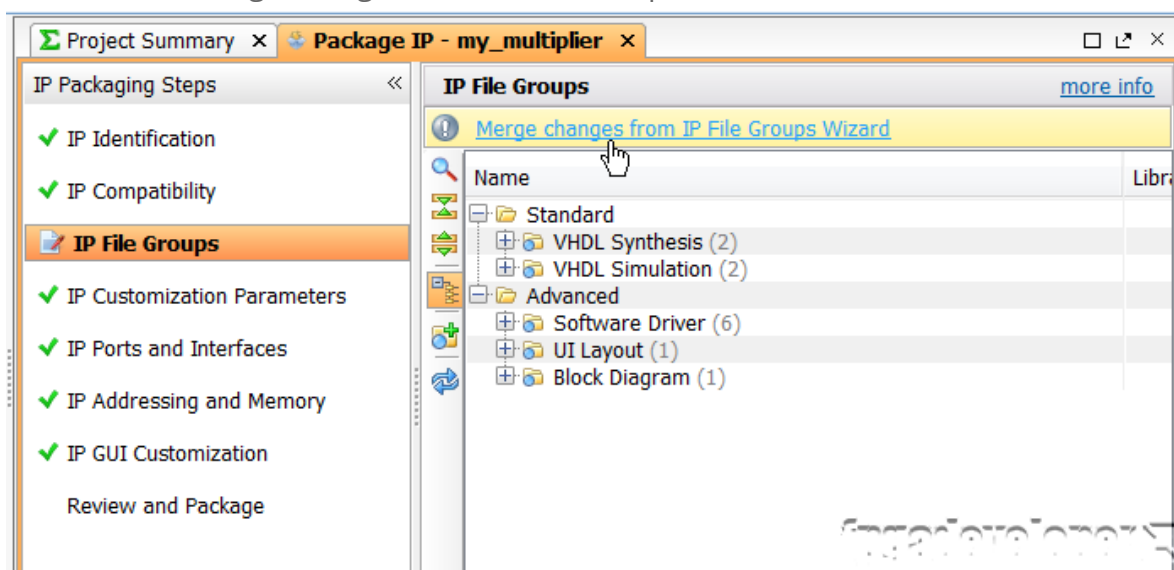
8. You should notice that the “multiplier.vhd” file has been integrated into the hierarchy because we have instantiated it from within the peripheral.



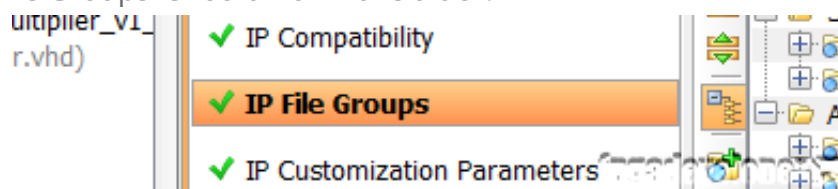
9. Click on “IP File Groups” in the Package IP tab of the Project Manager.



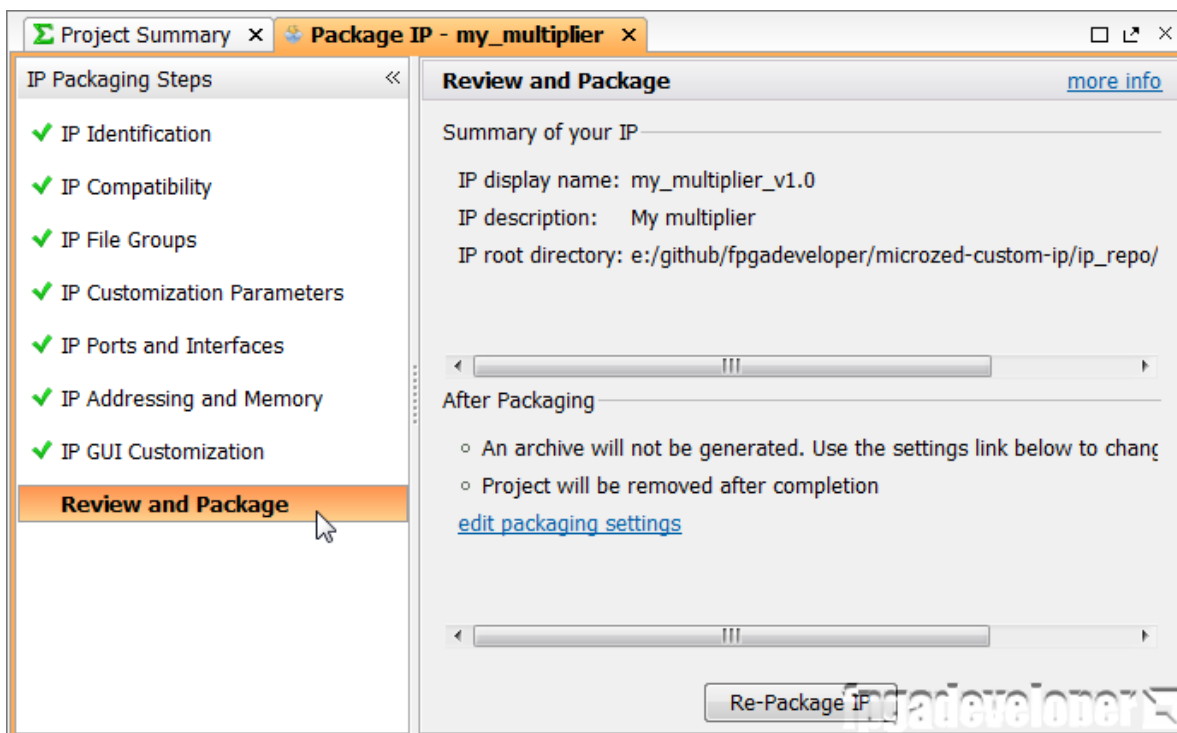
10. Click the “Merge changes from IP File Group Wizard” link.



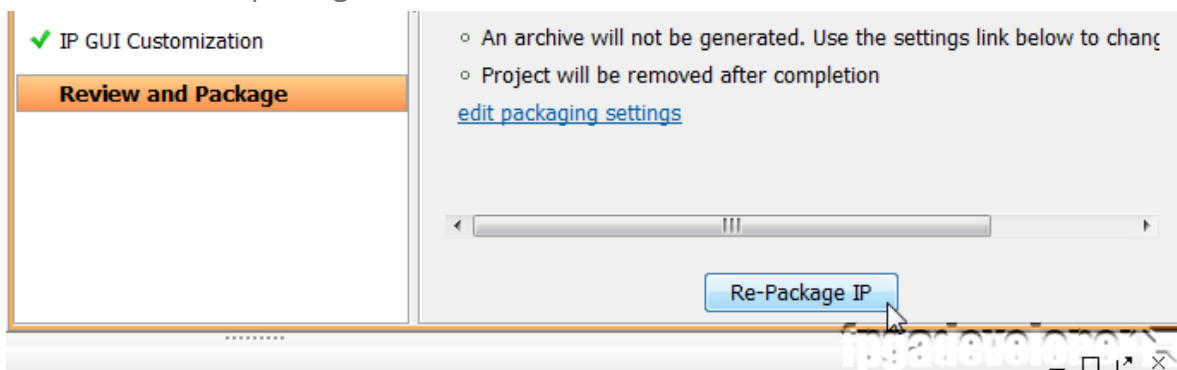
11. The “IP File Groups” should now have a tick.



12. Now click “Review and Package IP”.



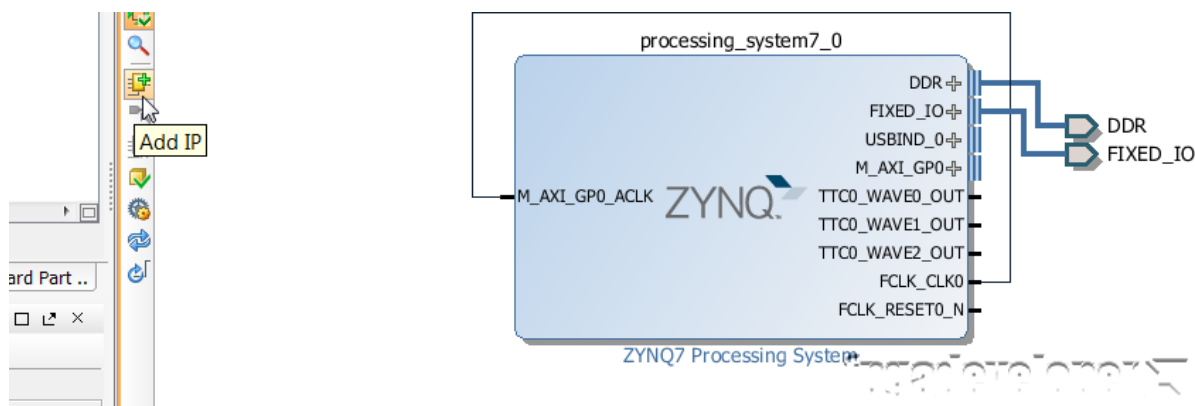
13. Now click "Re-package IP".



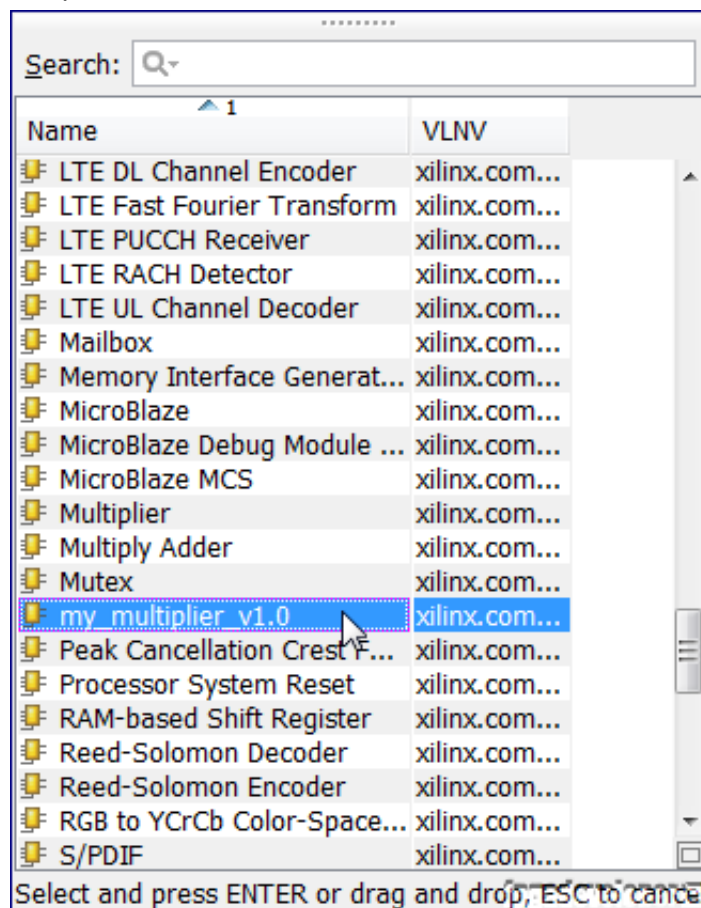
The peripheral will be packaged and the Vivado window for the peripheral should be automatically closed. We should now be able to find our IP in the IP catalog. Now the rest of this tutorial will be done from the original Vivado window.

## Add the IP to the design

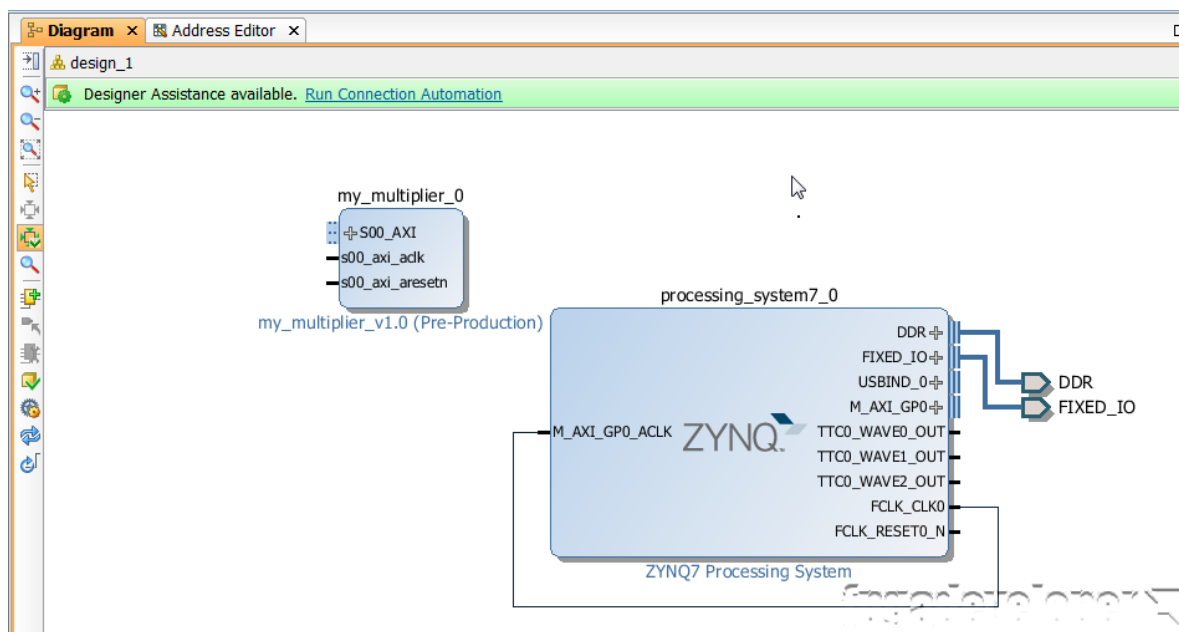
1. Click the "Add IP" icon.



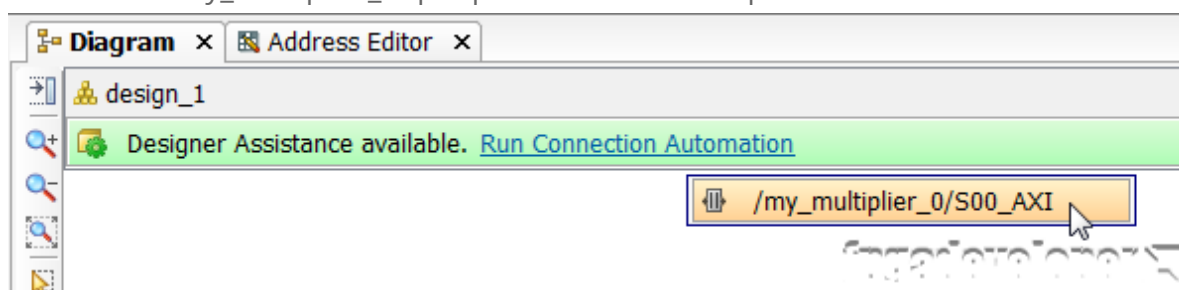
2. Find the “my\_multiplier” IP and double click it.



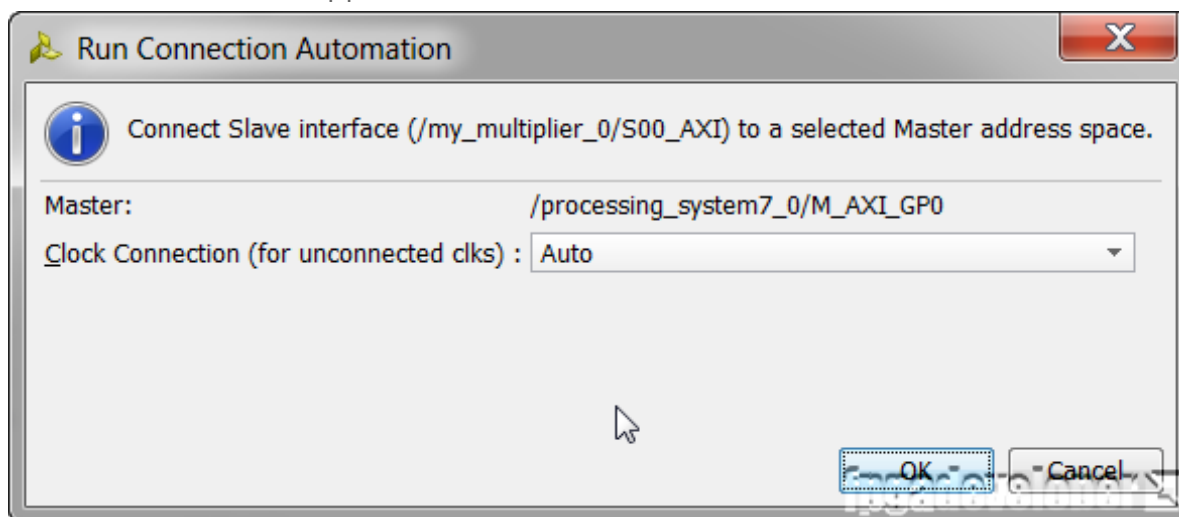
3. The block should appear in the block diagram and you should see the message “Designer Assistance available. Run Connection Automation”. Click the connection automation link.



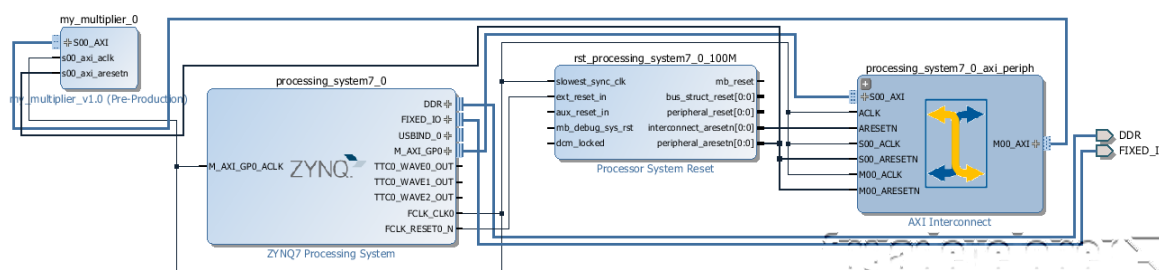
4. Click the “my\_multiplier\_0” peripheral from the drop-down menu.



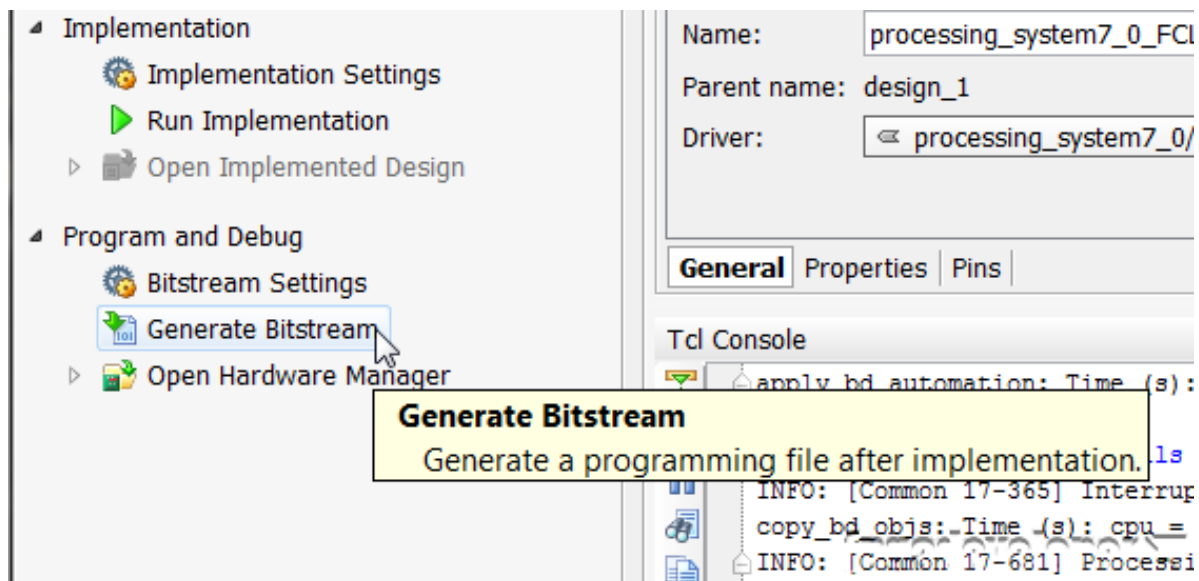
5. In the window that appears, set Clock connection to “Auto” and click “OK”.



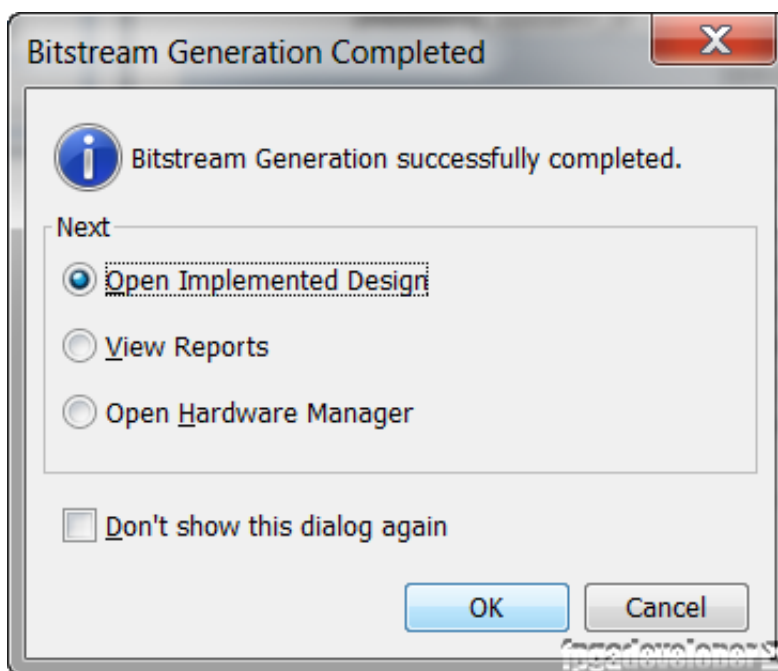
6. The new block diagram should look like this:



7. Generate the bitstream.



8. When the bitstream is generated, select "Open the implemented design" and click "OK".



## ~~Export the hardware design to SDK~~

~~Once the bitstream has been generated, we can export our design to SDK where we can then write code for the PS. The PS is going to write data to our multiplier and read back the result.~~

~~1. In Vivado, from the File menu, select "Export > Export hardware".~~