

Vivado Design Suite Tutorial

Designing with IP

UG939 (v 2012.2) September 12, 2012





Notice of Disclaimer

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of the Limited Warranties which can be viewed at <http://www.xilinx.com/warranty.htm>; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in Critical Applications: <http://www.xilinx.com/warranty.htm#critapps>.

©Copyright 2012 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.

Date	Changes
09/12/2012	Initial Xilinx Release.

Table of Contents

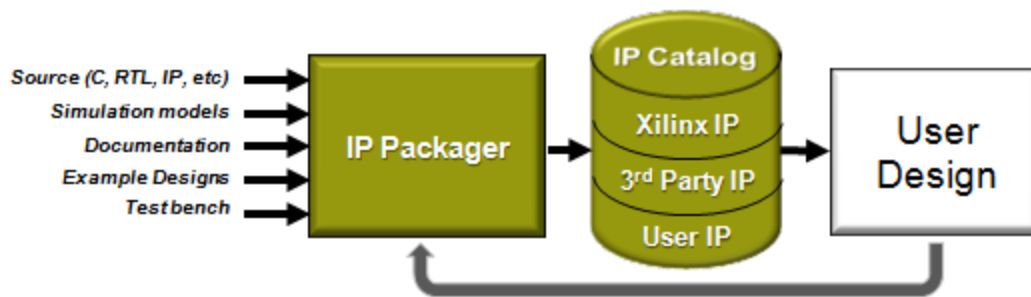
Designing with IP.....	1
Vivado Tutorial: Designing with IP.....	5
Overview.....	5
Software Requirements	6
Hardware Requirements.....	6
Tutorial Design Description	6
Locating Tutorial Design Files	6
Lab 1: Designing with the IP Catalog.....	7
Step 1: Invoke the Vivado IDE.....	7
Step 2: Open the project_wave_gen_ip Project	7
Step 3: Launch the IP Catalog	9
Step 4: Select and Customize the FIFO Generator	9
Step 5: Generate the IP Sources.....	12
Step 6: Instantiate the IP into a Design	13
Conclusion.....	15
LAB 2: Create and Verify a Standalone IP.....	17
Step 1: Invoke the Vivado IDE.....	17
Step 2: Create a New Vivado Project.....	17
Step 3: Select and Customize the FIFO Generator	19
Step 4: Generate the IP Sources.....	22
Step 5: Create an IP Netlist.....	23
Conclusion.....	24
LAB 3: IP Packaging Steps.....	25
Step 1: Open the my_complex_mult Project.....	25
Step 2: Invoke IP Packager	26
Step 3: Add IP Identification Information	27
Step 4: Add a Document to the New IP Package	28
Step 5: Package the IP into a ZIP File	29

Step 6: Add the New IP to the IP Catalog.....	30
Step 7: Verify the New IP in the Customize IP Dialog Box.....	33
Step 8: Verify the New IP through Synthesis and Implementation.....	34
Conclusion.....	34

Vivado Tutorial: Designing with IP

Overview

The Vivado™ Design Suite provides an IP-centric design flow that helps you quickly turn designs and algorithms into reusable IP. As shown in the figure below, the Vivado IP catalog is a unified IP repository that provides the framework for the IP-centric design flow. This catalog consolidates IP from all sources including Xilinx® IP, IP obtained from third parties, and end-user designs targeted for reuse as IP into a single environment.



The Vivado IP packager is a unique design reuse feature based on the IP-XACT standard. IP packager provides any Vivado user the ability to package a design at any stage of the design flow and deploy the core as system-level IP.

In this tutorial, you will do the following:

- Lab 1: Open an existing Vivado IDE design that has a missing IP, create a customized version of the IP from the IP catalog, and instantiate the customized IP into the design.
- Lab 2: Create a new Vivado IDE project, include an IP from the IP catalog as the top-level source, customize and verify the IP, then netlist the IP for inclusion in another design.
- Lab 3: Open the completed Vivado IDE project, use IP packager to package the design as IP, add the new IP to the IP catalog, then verify the new IP through synthesis and implementation.

Software Requirements

This tutorial requires that the 2012.2 Vivado Design Suite software release or later is installed.

Hardware Requirements

The supported Operating Systems include Redhat 5.6 Linux 64, Windows 7 and Windows XP (32 and 64 bit).

Xilinx recommends a minimum of 2 GB of RAM when using the Vivado tool.

Tutorial Design Description

The example design used in Lab 1 consists of a modified version of the Xilinx `wave_gen` design that has a missing IP. In Lab 3, you will package the completed `wave_gen` design as an IP module and add it to the Vivado IP catalog.

Locating Tutorial Design Files

You can find the design files for this tutorial next to the associated document file on the Web

- <www.xilinx.com> -> **Support > Product Support & Documentation > Design Tools > See All Vivado Design Suite Documentation > Vivado Design Suite -2012.2 Tutorials** (http://www.xilinx.com/support/documentation/dt_vivado.htm)

After downloading the file **ug939-design-files.zip**, extract the contents to any write-accessible location on your disk. This tutorial assumes that you are extracting the data files to the C drive.

Note: You will modify the tutorial design data while working through this tutorial. You should use a new copy of the original **ug939-design-files** directory each time you start this tutorial.

Lab 1: Designing with the IP Catalog

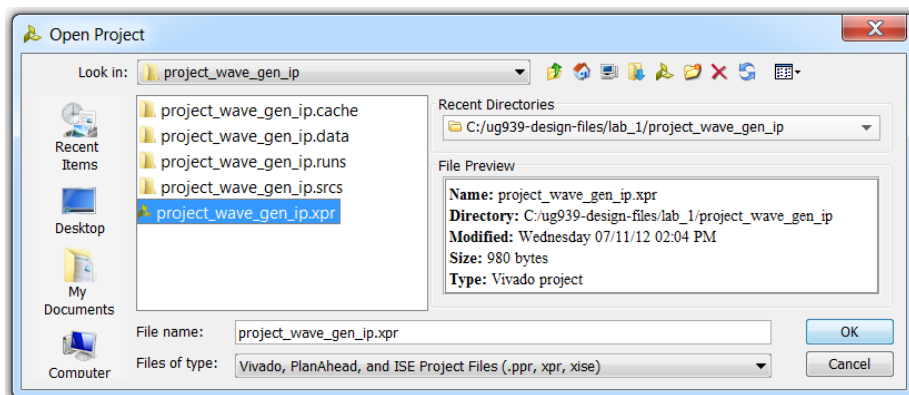
In this exercise, you will learn how to use the IP catalog in the Vivado Integrated Design Environment (IDE). You will be using a version of the Xilinx wave generator sample design; it is lacking a FIFO which you will customize, generate, and instantiate.

Step 1: Invoke the Vivado IDE

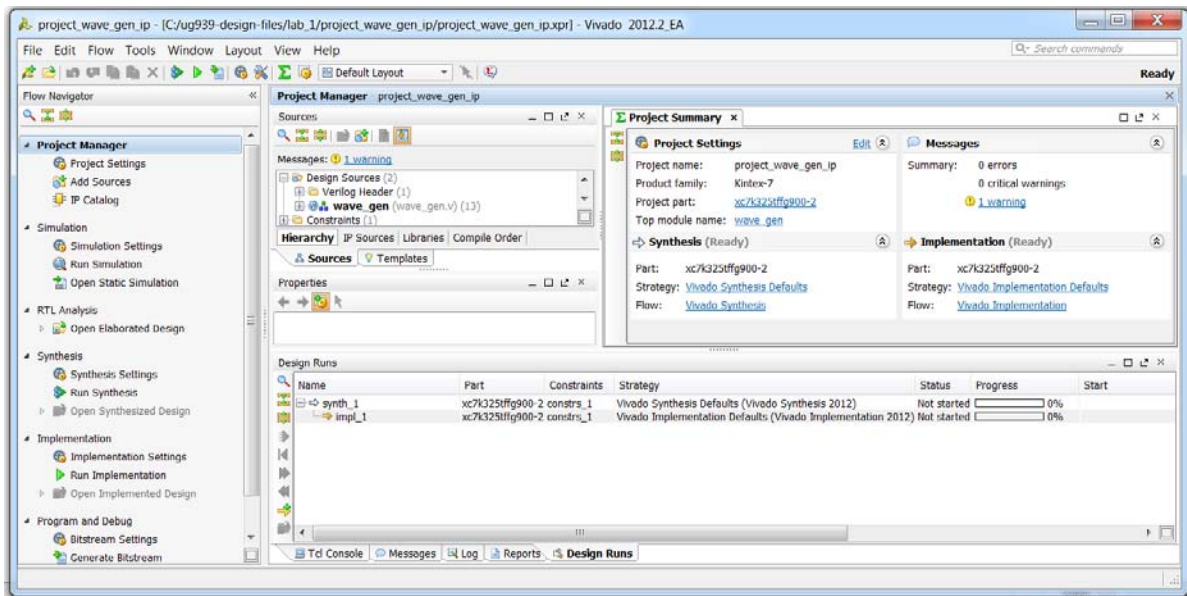
Invoke the Vivado IDE by clicking on the Vivado desktop icon or by typing **vivado** at a terminal command line.

Step 2: Open the project_wave_gen_ip Project

Select **Open Project** from the Getting Started page and browse to the area you extracted the **ug939-design-files**. Navigate to the **lab_1/project_wave_gen_ip** directory, and select **project_wave_gen_ip.xpr** as shown below. Click **OK**.



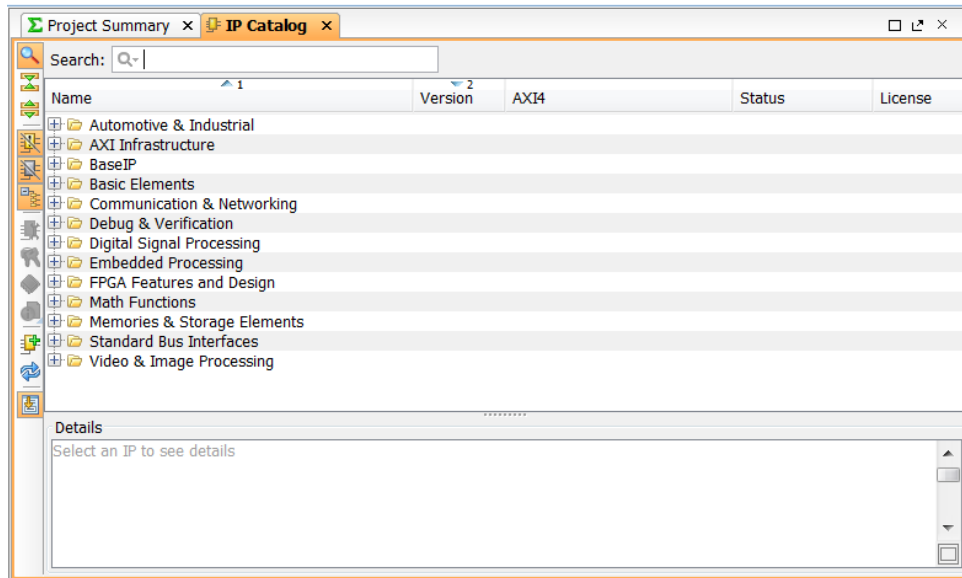
The design will load and you will see the Vivado IDE with the project information as shown below.



Since this is an RTL project, you can do behavioral simulation, elaborate the design, or launch synthesis. The Vivado IDE also offers a one-button flow so you can generate a bitstream, which will automatically launch synthesis and implementation.

Step 3: Launch the IP Catalog

1. Select **IP Catalog** from the Flow Navigator in the Project Manager area. The IP Catalog displays in a new tab as shown below:

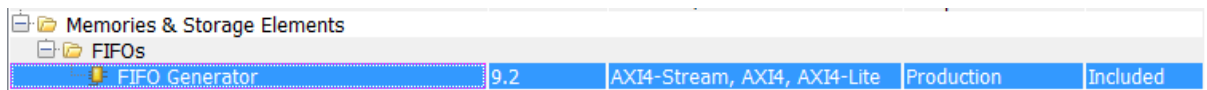


You can work with the IP Catalog in a variety of ways. You can search using keywords in the search box or browse through the catalog in the various categories.

2. Type **FIFO** in the search box and press **Enter**.


Step 4: Select and Customize the FIFO Generator

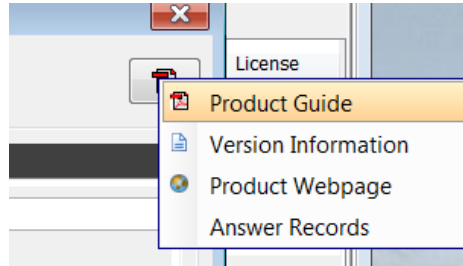
1. From the **Memories & Storage Elements > FIFOs** group select **FIFO Generator**.



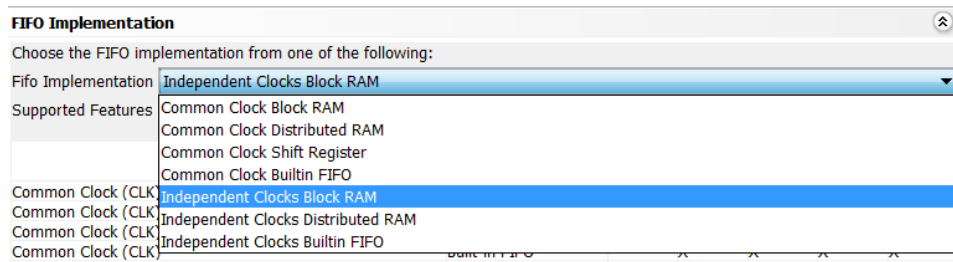
2. To customize this IP, double click on it or right click and select **Customize IP**. The FIFO Generator is displayed in the Customize IP window.

This presentation in the Customize IP window is typical of IP delivered with the Vivado IDE. The device symbol is on the left with only the enabled ports displayed. You can display all ports by checking the **Show Disabled Ports** box. This device symbol view has zoom capabilities using mouse strokes with the left mouse button. This device view changes as you customize the IP.

As shown below, you can right-click on the  button in the upper-right corner, to get more information about the selected IP.

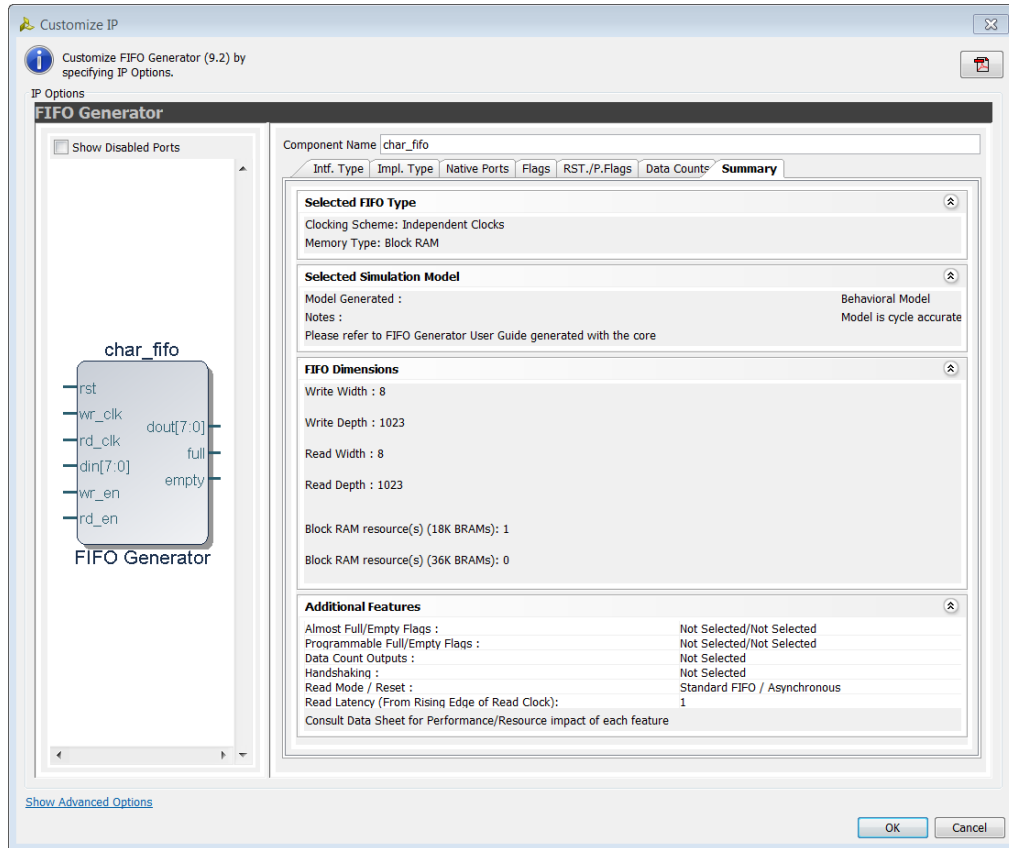


3. The first tab sets the interface type, **Intf. Type**. This design requires the native interface, which is the default. Change the Component Name to **char_fifo** from the default name of `fifo_generator_v9_2_0`.
4. Select the **Impl. Type** tab. This configures the implementation type of the FIFO. In this tab, you set what type of memory you want to use and if you want independent clocks. For this design, you want **Independent Clocks Block RAM**. Select this from the drop down menu as shown below.



5. Select the **Native Ports** tab. Here you can set the Read Mode, Built-in FIFO Options, Data Port Parameters, and Implementation Options. For this design, you need the **Write Width** to be **8** bits. Changing this will automatically change the **Read Width** as you click on the **Read Width** dialog area. Leave everything else with the default settings on this tab.
6. Browse through the **Flags**, **RST./P.Flags**, and **Data Counts** tabs. These tabs configure other options for the FIFO Generator. For this design, leave everything with the default settings.

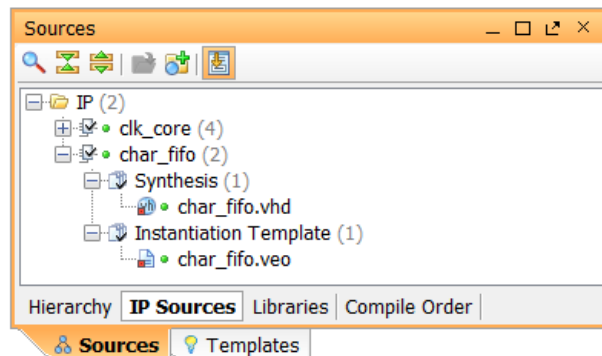
7. Select the **Summary** tab. This displays a summary of all the options selected as well as listing resources used for this configuration. You should see:



Verify that the information is correct. For this configuration you are using a one 18K BRAM.

8. Make any changes required to match the configuration and once you are satisfied click **OK**.
9. The FIFO now appears in the **Sources** view. In the Hierarchy tab, it is at the same hierarchy level as the top-level module (wave_gen). In the Libraries and Compile Order tabs, it appears in the **Unreferenced** folder since an instance of the IP has not yet been added to the project.

In the IP Sources tab, you can see the sources delivered for this IP by default.



This IP delivers a synthesis target and an instantiation template. Since the target language is set to Verilog (the default), the instantiation template is a VEO file.

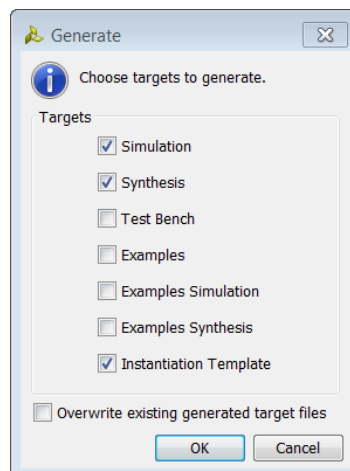
This process can also be accomplished via Tcl scripting. If you look at the Tcl Console, you will see the Tcl commands that are issued for this IP customization.

```
create_ip -name fifo_generator -version 9.2 -vendor xilinx.com -library ip
-module_name char_fifo
set_property -name CONFIG.Component_Name -value {char_fifo} -objects
[get_ips char_fifo]
set_property -name CONFIG.Fifo_Implementation -value
{Independent_Clocks_Block_RAM} -objects [get_ips char_fifo]
set_property -name CONFIG.Input_Data_Width -value {8} -objects [get_ips
char_fifo]
generate_target {instantiation_template synthesis} [get_files C:/ug939-
design-
files/lab_1/project_wave_gen_ip/project_wave_gen_ip.srcs/sources_1/ip/char
_fifo/char_fifo.xci -of_objects [get_filesets sources_1]] -force
```

The `create_ip` command adds the IP to the design and configuration options are all done via the `set_property` command.

Step 5: Generate the IP Sources

1. In the **IP Sources** tab, select **char_fifo**, right click and select **Generate**. This displays the generation targets possible for this IP. Selecting one or more of these options results in additional targets being added to this IP.



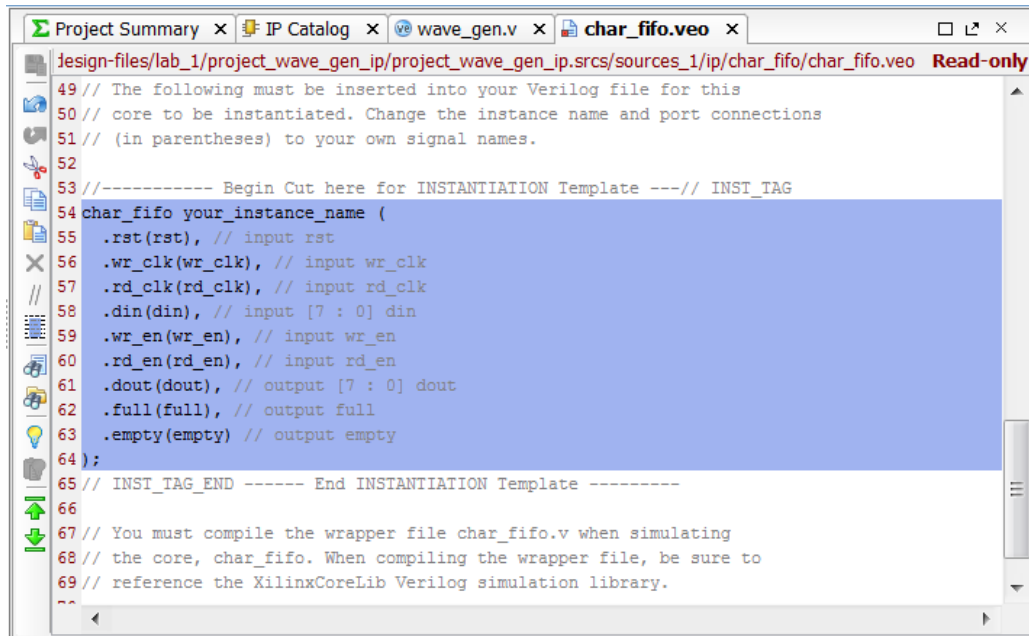
Notice that Synthesis and Instantiation Template are selected by default.

2. Select **Simulation** as well and click **OK**.

You will see the new target in the **IP Sources** tab.

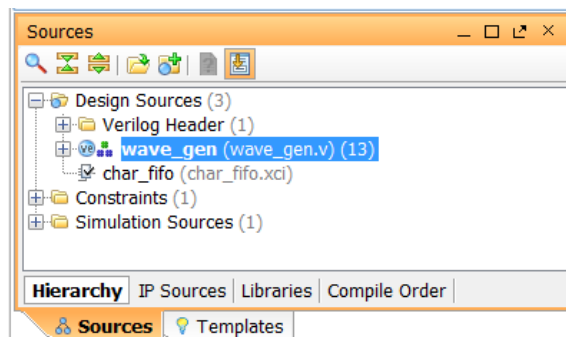
Step 6: Instantiate the IP into a Design

1. You will now instantiate the IP into the design. Expand the **Instantiation Template** and double click on the **char_fifo.veo** file to open the template so you can copy and paste the template into the design. Go down to line 54 and copy the text as shown below:.

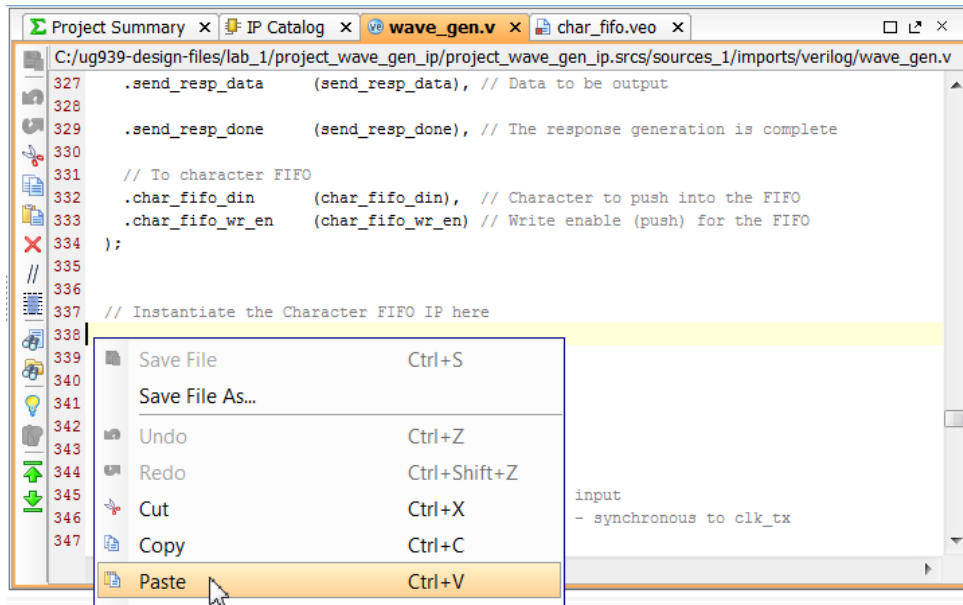


```
design-files/lab_1/project_wave_gen_ip/project_wave_gen_ip.srcs/sources_1/ip/char_fifo/char_fifo.veo Read-only
49 // The following must be inserted into your Verilog file for this
50 // core to be instantiated. Change the instance name and port connections
51 // (in parentheses) to your own signal names.
52
53 //----- Begin Cut here for INSTANTIATION Template ---// INST_TAG
54 char_fifo your_instance_name (
55     .rst(rst), // input rst
56     .wr_clk(wr_clk), // input wr_clk
57     .rd_clk(rd_clk), // input rd_clk
58     .din(din), // input [7 : 0] din
59     .wr_en(wr_en), // input wr_en
60     .rd_en(rd_en), // input rd_en
61     .dout(dout), // output [7 : 0] dout
62     .full(full), // output full
63     .empty(empty) // output empty
64 );
65 // INST_TAG_END ----- End INSTANTIATION Template -----
66
67 // You must compile the wrapper file char_fifo.v when simulating
68 // the core, char_fifo. When compiling the wrapper file, be sure to
69 // reference the XilinxCoreLib Verilog simulation library.
--
```

2. Next, you will copy the template code to the RTL code. From the **Hierarchy** tab of the **Sources** view, double click on **wave_gen.v** to open this file for editing.



- Go down to line **337** and paste the template code into the file..



- Next, you need to change “your_instance_name” to “char_fifo_i0” and change the wire names to match the design as shown in the table below. Make the following edits to connect the ports of the IP to the design correctly.


IP Port	RTL connection
rst	rst_i
wr_clk	clk_rx
rd_clk	clk_tx
din	char_fifo_din
wr_en	char_fifo_wr_en
rd_en	char_fifo_rd_en
dout	char_fifo_dout
full	char_fifo_full
empty	char_fifo_empty

After you make these edits, the code should look like the following:

```

337 // Instantiate the Character FIFO IP here
338 char_fifo char_fifo_i0 (
339     .rst(rst_i), // input rst
340     .wr_clk(clk_rx), // input wr_clk
341     .rd_clk(clk_tx), // input rd_clk
342     .din(char_fifo_din), // input [7 : 0] din
343     .wr_en(char_fifo_wr_en), // input wr_en
344     .rd_en(char_fifo_rd_en), // input rd_en
345     .dout(char_fifo_dout), // output [7 : 0] dout
346     .full(char_fifo_full), // output full
347     .empty(char_fifo_empty) // output empty
348 );

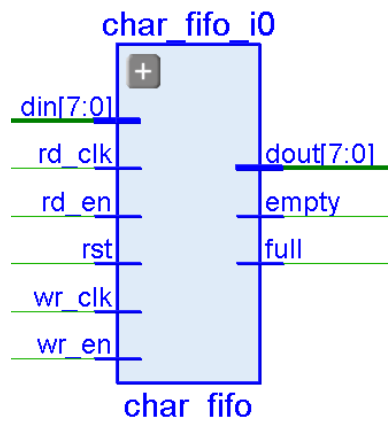
```

- Click on the **Save File** icon () to save the **wave_gen.v** file.

Notice that the Hierarchy, Libraries, and Compile Order tabs are all updated to indicate that the IP has been instanced into the design.

- Elaborate the design to verify that your connections are correct. In the RTL Analysis section of the Flow Navigator select **Open Elaborated Design**.

An RTL Schematic opens and you can zoom in to see the connections of the char_fifo_i0 IP. From the RTL Netlist tab in the Sources view, select **char_fifo_i0**. This highlights the cell in the schematic. Zoom in and you will see the following:



- Congratulations! You have successfully customized the FIFO IP and added it to the design. Exit the Vivado IDE or proceed to simulation or synthesis/implementation.

Conclusion

In this exercise, you learned how to select and customize an IP in the IP catalog, and how to instantiate the customized IP into a design. You can do this interactively within the Vivado IDE or via Tcl scripting.

LAB 2: Create and Verify a Standalone IP

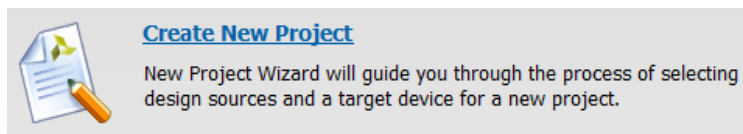
In this exercise, you will create and verify an IP as a standalone source in a Vivado IDE project. This allows you to customize, verify, synthesize, and implement an IP as the top-level design. As will be shown, to use the Vivado IDE IP with third-party tools you will need to create a netlist.

Step 1: Invoke the Vivado IDE

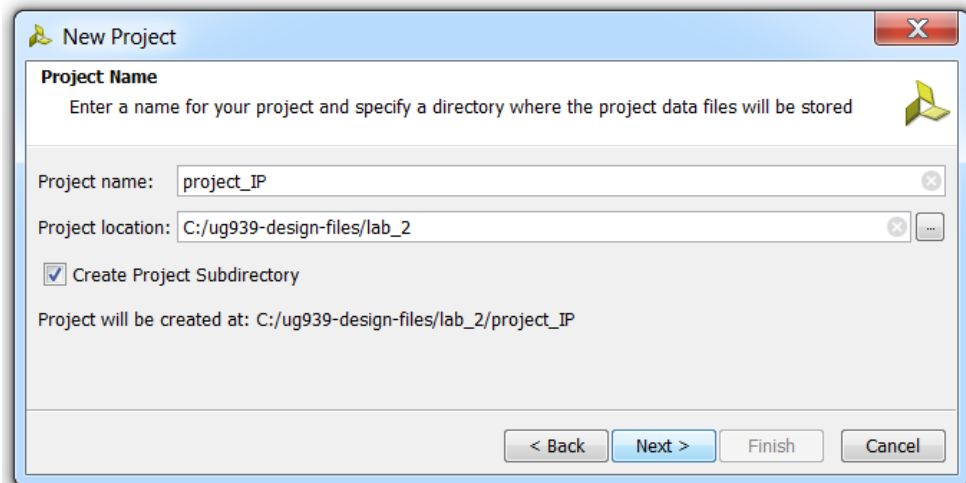
Invoke the Vivado IDE by clicking on the desktop icon or typing **vivado** at a terminal.

Step 2: Create a New Vivado Project

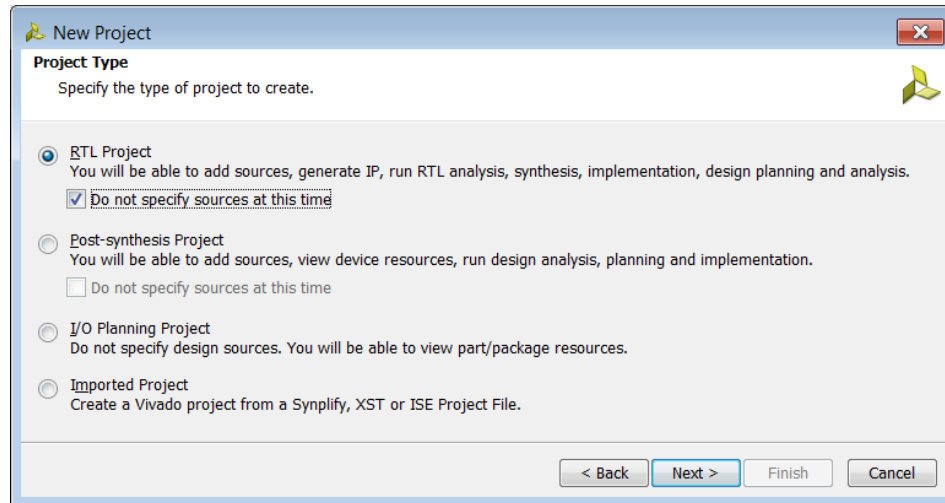
1. On the Getting Started page, click **Create New Project**, and then click **Next**.



2. The New Project window displays. Click **Next** at the introduction screen. At the New Project Name screen enter the Project name **project_IP** and Project location **C:/ug939-design-files/lab2** as shown below. Click **Next**.



- The Project Type window displays. Verify the selection of **RTL Project** and check the box **Do not specify sources at this time**. You will be adding the IP later once the project has been created and opened. Click **Next**.



- The Default Part window displays. Select the default Xilinx part or board for your project. You can change this setting later. You can leave the default 7 series part or select another if known. Click **Next**.
- The New Project Summary window displays. Click **Finish**.



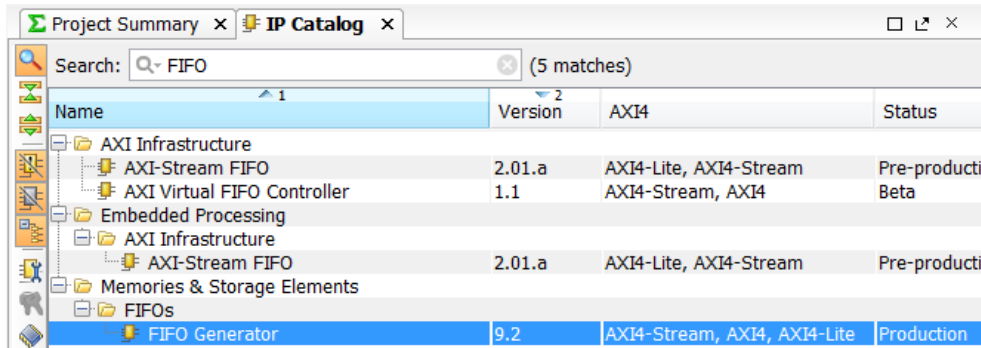
The project will open. You now have an empty project to add an IP as a source using the IP catalog.

Step 3: Select and Customize the FIFO Generator

1. From the Flow Navigator under Project Manager select **IP Catalog**.


The IP catalog opens. You can work with the IP catalog in two ways, either searching with a keyword, or browsing through the categories.

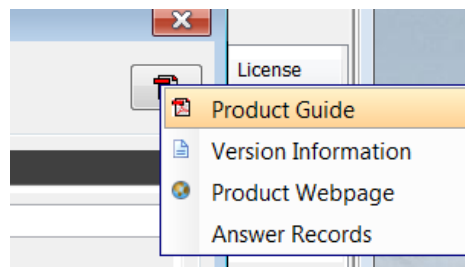
2. Type **FIFO** in the search bar and select **FIFO Generator**.



3. Double click on FIFO Generator to bring up the Customize IP window.

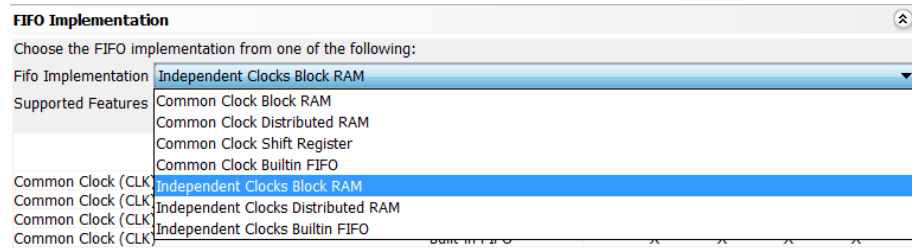
This customization window is typical of IP delivered with the Vivado IDE. The device symbol is on the left with only the enabled ports displayed. You can display all ports by checking the **Show Disabled Ports** box. This device symbol view has zoom capabilities using the mouse with the left button. This device view will change as you customize the IP.

You can get more information about the IP by clicking the  button in the upper-right corner.



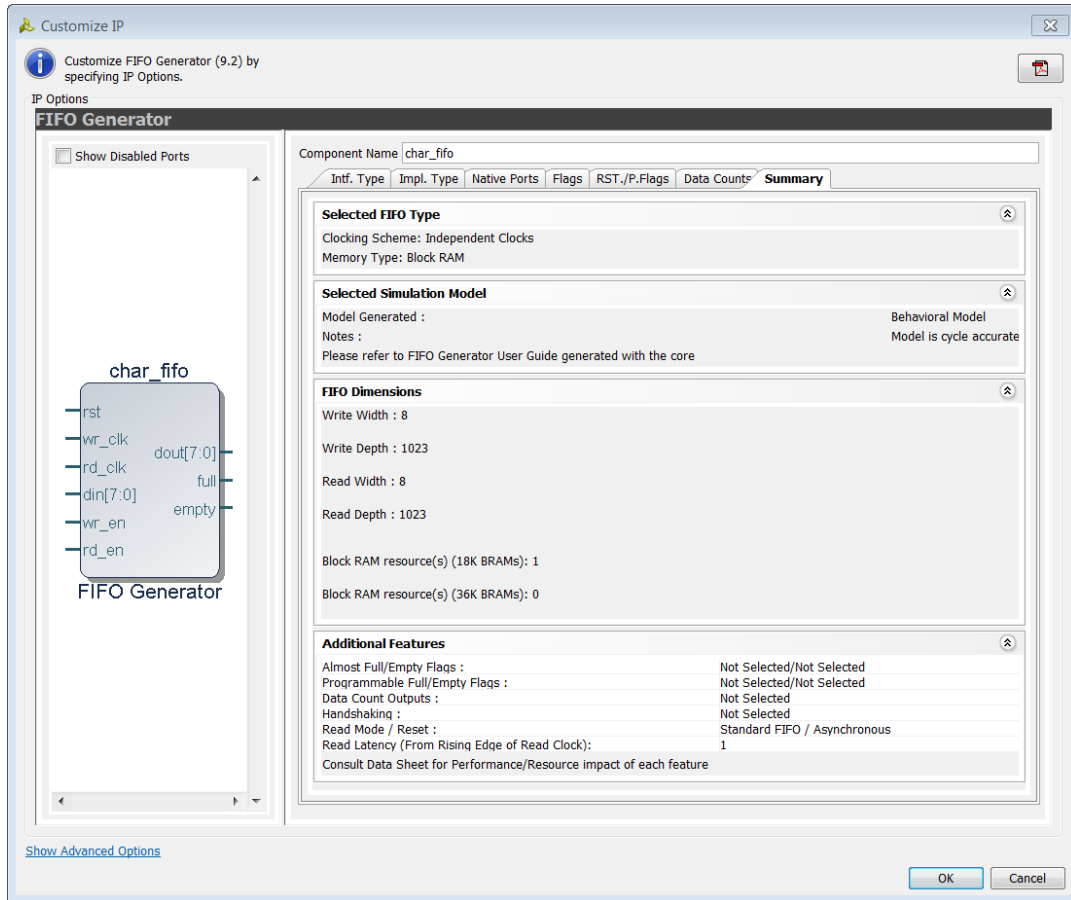
4. The first tab is **Intf. Type** and sets the interface type. This design requires the native interface, which is the default. Change the Component Name to **char_fifo** from the default name of `fifo_generator_v9_2_0`.

5. Select the **Impl. Type** tab. This configures the implementation type of the FIFO. In this tab, you set what type of memory you want to use and if you want independent clocks. For this design, let us assume you want independent clocks using a Block RAM (BRAM). Select this from the drop down menu as shown below.



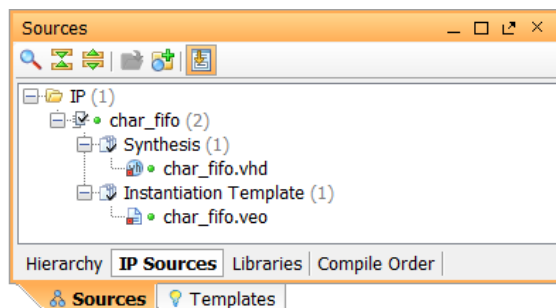
6. Select the **Native Ports** tab. Here you can set the Read Mode, Built-in FIFO Options, Data Port Parameters, and Implementation Options. For this design, you need the **Write Width** to be **8** bits. Changing this will automatically change the **Read Width** as you click on the **Read Width** dialog area. Leave everything else with the default settings on this tab.
7. Browse through the **Flags**, **RST./P.Flags**, and **Data Counts** tabs. These configure other options for the FIFO Generator. For this design, leave everything with the default settings.

8. Select the **Summary** tab. This displays a summary of all the options selected as well as listing resources used for this configuration. You should see:



Verify that the information is correct. For this configuration you will see you are using one 18K BRAM.

9. Make any changes required to match the configuration and once satisfied click **OK** on the bottom-right to add the customized IP to the project.
10. The FIFO now appears in the **Sources** view. In the **Hierarchy** tab, the IP is set as the top-level module. In the **IP Sources** tab, you can see the sources delivered for this IP by default.



This IP delivers a synthesis target and an instantiation template. Since the target language is set to Verilog (the default), the instantiation template is a VEO file.

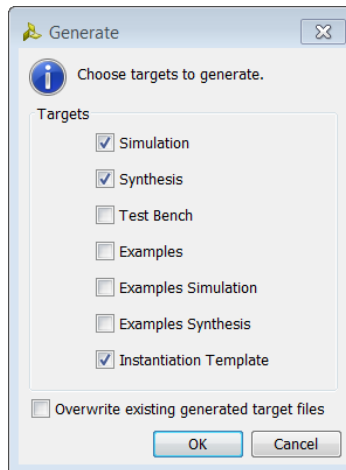
This process can also be accomplished via Tcl scripting. If you look at the Tcl Console you will see the Tcl commands that you issued for this IP customization.

```
create_ip -name fifo_generator -version 9.2 -vendor xilinx.com -
library ip -module_name char_fifo
set_property -name CONFIG.Component_Name -value {char_fifo} -objects
[get_ips char_fifo]
set_property -name CONFIG.Fifo_Implementation -value
{Independent_Clocks_Block_RAM} -objects [get_ips char_fifo]
set_property -name CONFIG.Input_Data_Width -value {8} -objects
[get_ips char_fifo]
generate_target {instantiation_template synthesis} [get_files
C:/ug939-design-
files/lab2/project_IP/project_IP.srscs/sources_1/ip/char_fifo/char_fi
fo.xci -of_objects [get_filesets sources_1]] -force
```

The `create_ip` command adds the IP to the design and configuration options are all done via the `set_property` command.

Step 4: Generate the IP Sources

1. In the **IP Sources** tab select **char_fifo**, right click and select **Generate**. This displays the generation targets possible for this IP, including Simulation, Test Bench, Examples, etc. When you select one or more of these options and click **OK**, additional targets are added to this IP. Check **Simulation** and click **OK**.

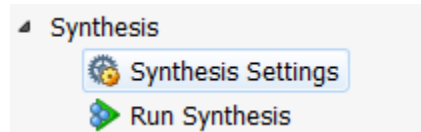


You will see the new target in the **IP Sources** tab.

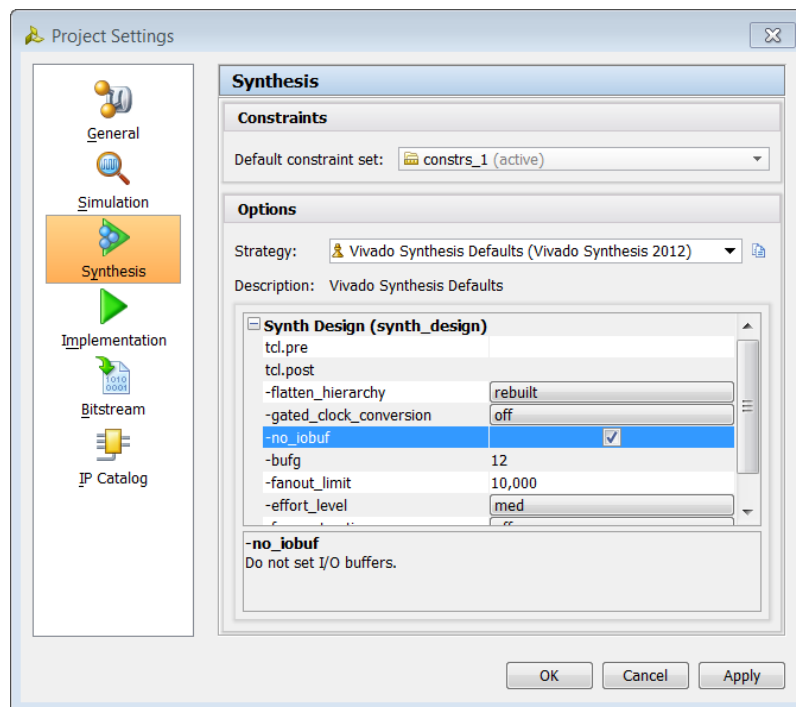
2. If you desire, you can simulate the IP stand-alone by either creating a testbench or using one of the examples that can be generated.

Step 5: Create an IP Netlist

1. A netlist for this IP can now be created. You can use the netlist in another Vivado IDE project, such as the wave generator sample design, or with a third-party tool. You need to configure synthesis to disable the insertion of IO buffers. Select **Synthesis Settings** from the Flow Navigator in the Synthesis section.

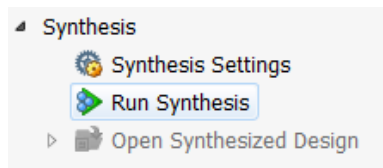


2. In the **Synth Design** options area, check the box **-no_iobuf** as shown below:

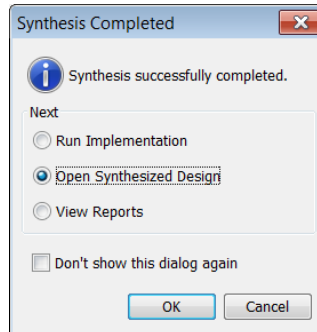


Click **OK** to save the change. Now the netlist you produce will not have IO buffers inserted unless they are explicitly instantiated in the IP.

3. To start synthesis, select **Run Synthesis** from the Flow Navigator.

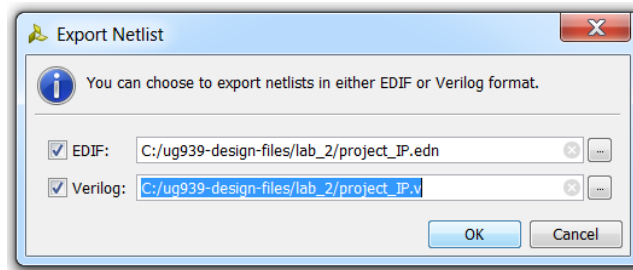


- When synthesis completes, open the synthesized design by selecting **Open Synthesized Design** and clicking **OK** in the **Synthesis Completed** window.



You can now do things like explore the netlist, apply timing constraints and generate reports.

- Create a Structural Verilog netlist called **char_fifo.v** from **File > Export > Export Netlist**



You can also create netlists from the Tcl Console using the following commands:

- `write_verilog`
- `write_edif`
- `write_vhdl`

The VHDL or Verilog produced are structural simulation models that you can use with third- party simulators. This entire process can be scripted and executed in batch or Tcl mode.

- Open the **vivado.jou** file. The Vivado IDE creates this file automatically. You can open this file in the Vivado IDE via **File > Open Journal File**. You can see all the Tcl commands that you used to accomplish what you just did interactively.
- Exit the Vivado IDE.

Conclusion

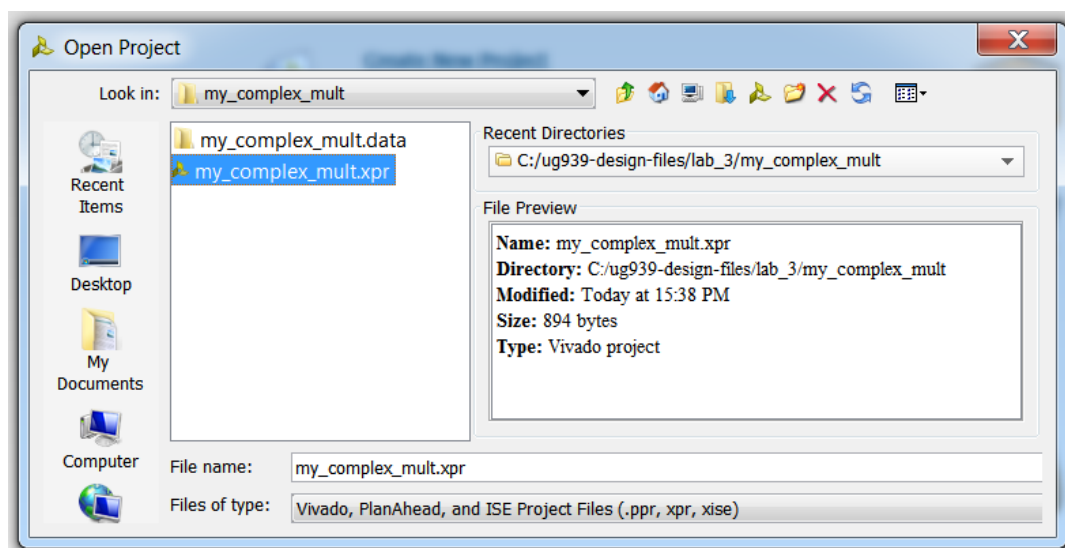
In this lab, you learned how to create a new Vivado IDE project for standalone IP. You customized and validated the IP before you generated a netlist for use in another Vivado IDE project or a third-party tool.

LAB 3: IP Packaging Steps

In this exercise, you will open the **my_complex_mult** project that is located in the in Lab 3 folder, go through the IP packaging steps to create a new IP module, then add the new IP to the Vivado IP catalog. Finally, you will verify the new IP through synthesis and implementation.

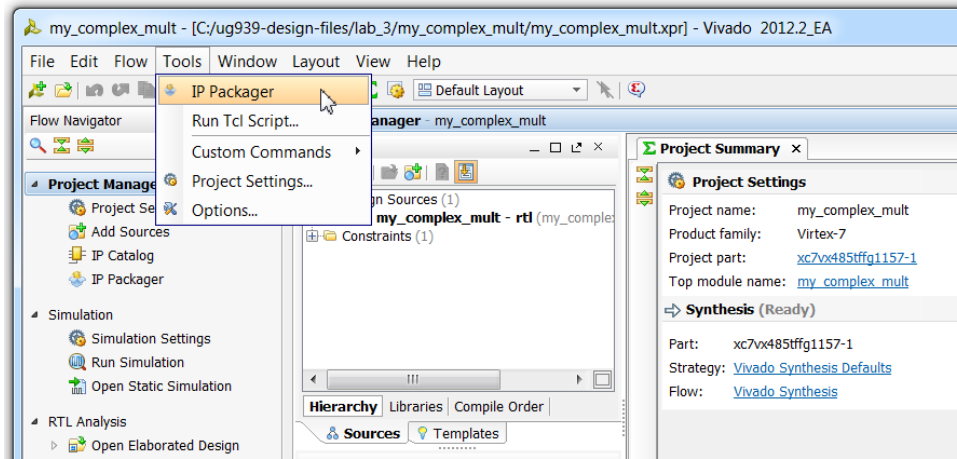
Step 1: Open the my_complex_mult Project

Invoke the Vivado IDE. Click **Open Project**, and navigate to the **my_complex_mult** folder in the lab_3 work area. As shown below, select the project file `my_complex_mult.xpr` and click **OK**.

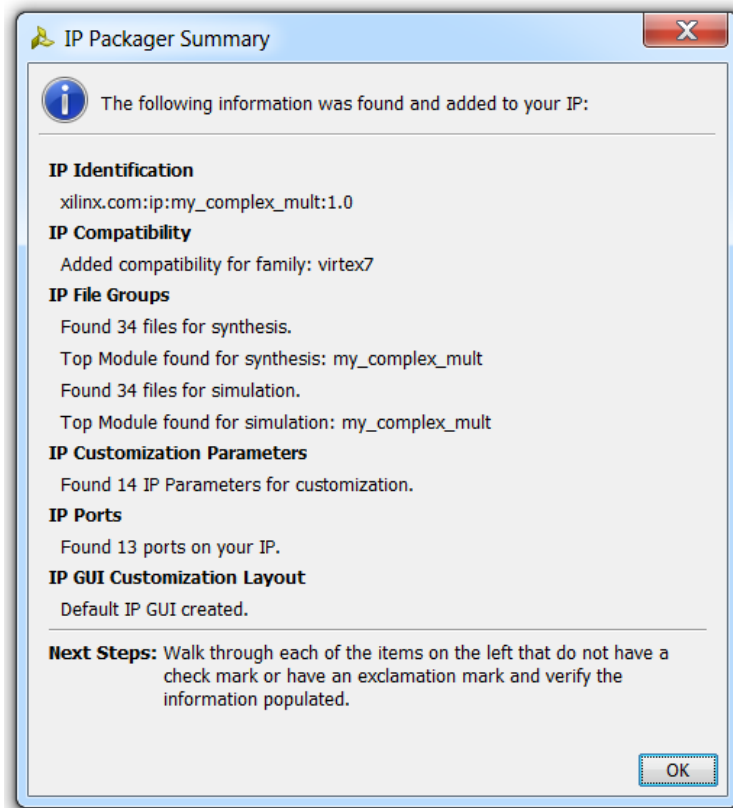


Step 2: Invoke IP Packager

1. As shown below, select the pulldown menu **Tools > IP Packager**.



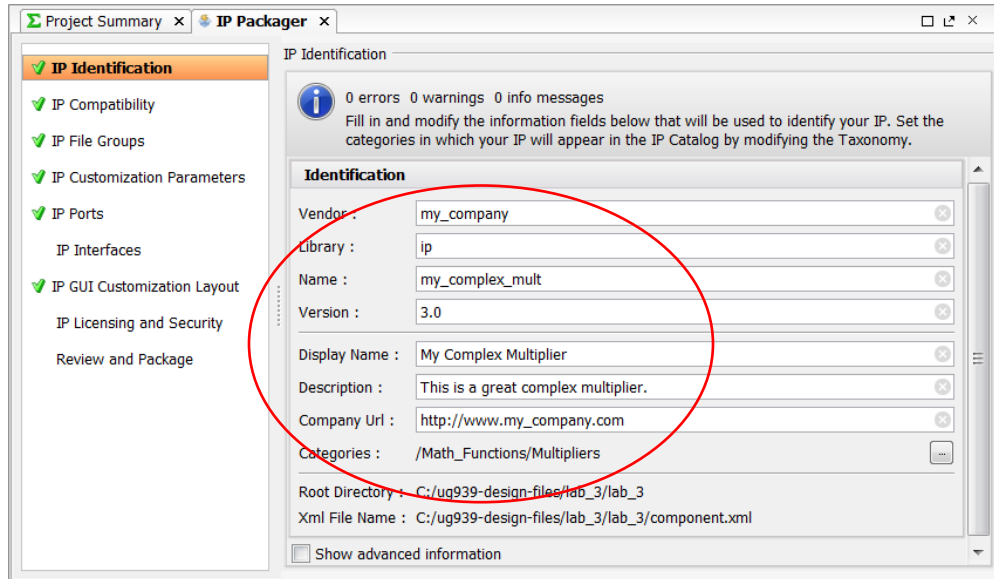
2. Click **Next** and then click **Finish**. The IP packager collects the available information in the `my_complex_mult` project and displays what it finds:



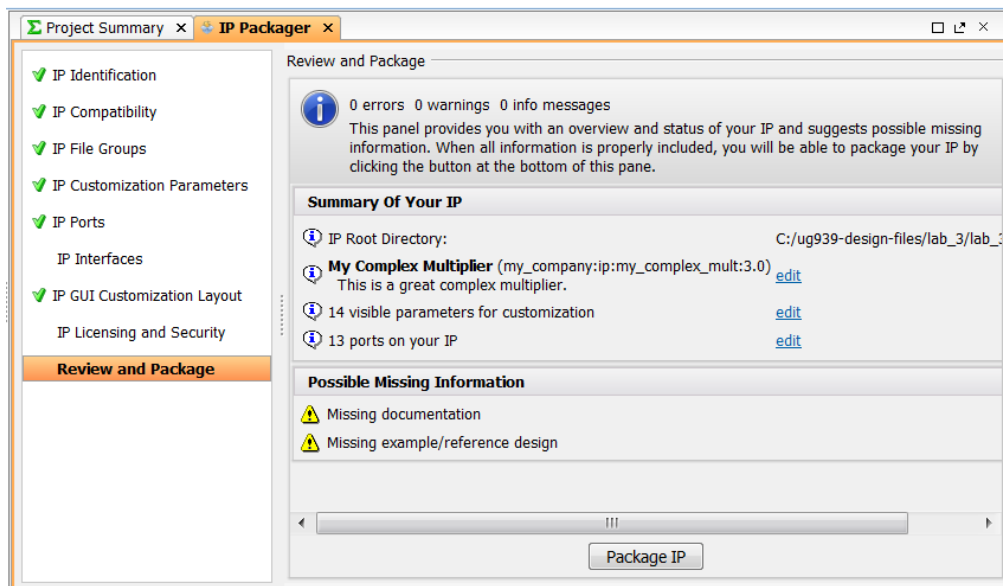
3. Click **OK**.

Step 3: Add IP Identification Information

1. In the IP Identification pane, fill in the empty fields with the information shown in the following figure. Notice that required information is marked with a red error symbol.



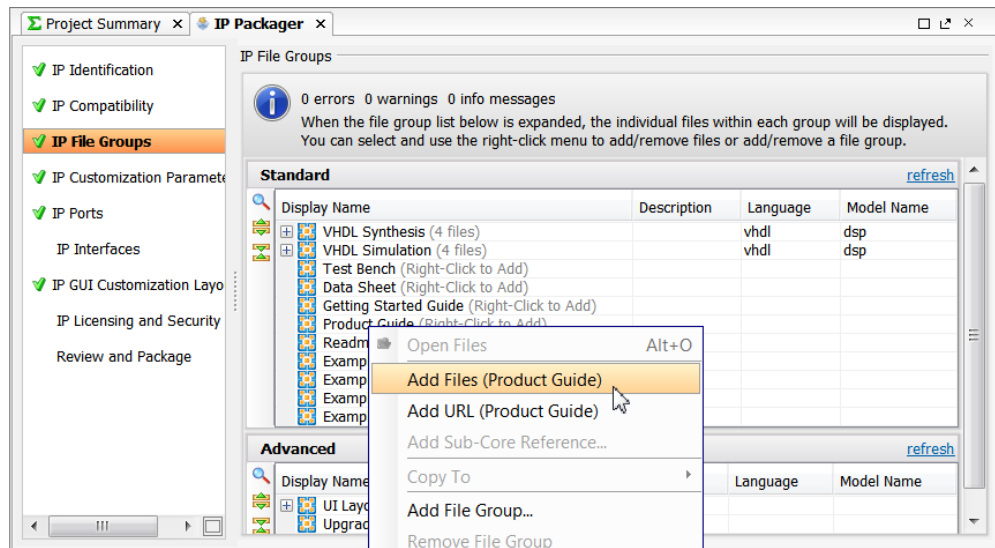
2. As shown below, click **Review and Package**. Notice that possible missing information such as documentation and design examples are identified. You will add a document file in the next step.



Step 4: Add a Document to the New IP Package

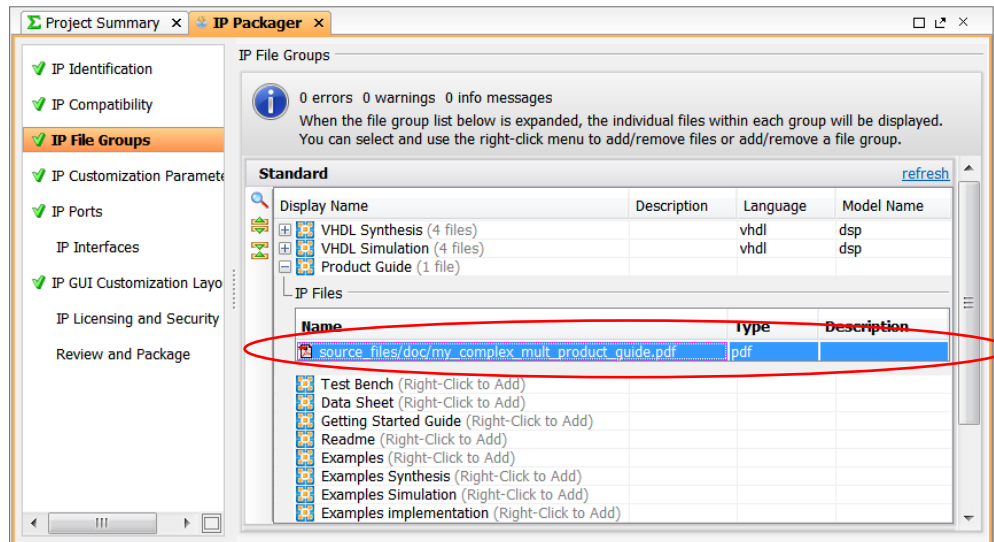
The following procedure explains how to add a non-HDL file to the package.

1. In the IP Packager tab, select **IP File Groups** in the left frame, right-click on the **Product Guide** category in the right frame and select **Add Files (Product Guide)**, as shown in the figure below.



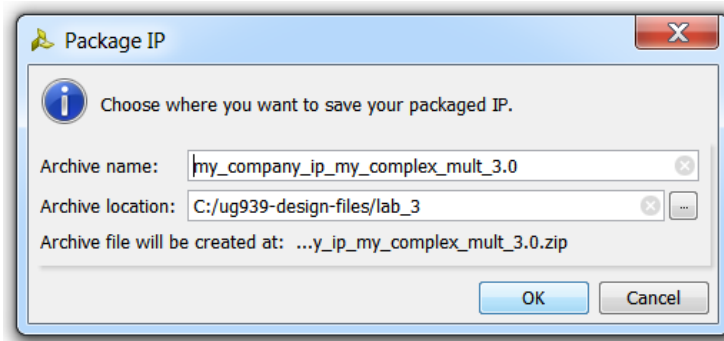
2. Click on the **Add Files...** button in the popup dialog box, navigate to the directory `C:/ug939-design-files/lab_3/source_files/doc` and select **All Files** in the Files of type: entry line. You should now see a documentation file in the popup dialog window.
3. Select the file `my_complex_mult_product_guide.pdf` and click **OK**. Click **OK** again in the Add IP Files (Product Guide) dialog box.

- Expand the Product Guide (1 file) category, as shown in the following figure and see that the document file has been added to the package.



Step 5: Package the IP into a ZIP File

- Click the **Review and Package** button, then the **Package IP** button. In the **Package IP** dialog box, do the following:
 - Verify that the name of the ZIP file is:
my_company_ip_my_complex_mult_3.0
as shown in the following figure.
 - Verify that the Output Directory is set to C:/ug939-design-files/lab_3
 - Click **OK**.



- Check the folder C:/ug939-design-files/lab_3 to make sure that the new ZIP file was added.

3.

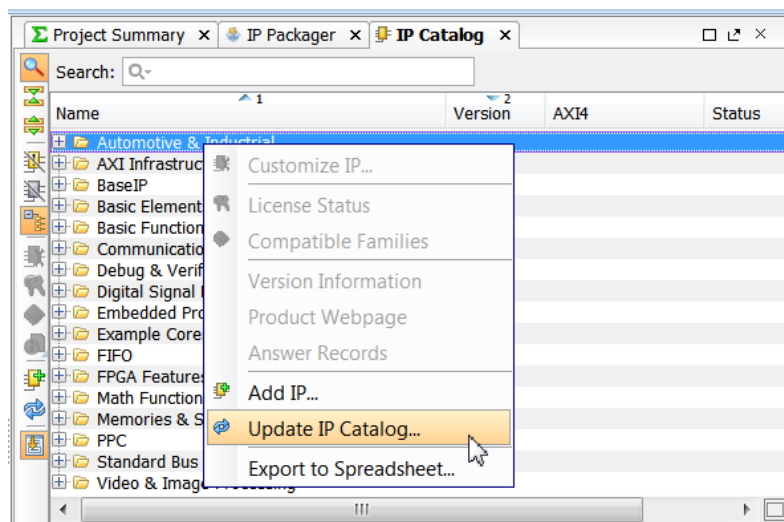
Step 6: Add the New IP to the IP Catalog

1. Unzip the newly-created ZIP file in the `C:\ug939-design-files\lab_3` folder.



IMPORTANT: When adding IP to the IP catalog that is packaged as a ZIP file, you must first unzip the ZIP file before you can add the directory as a user repository.

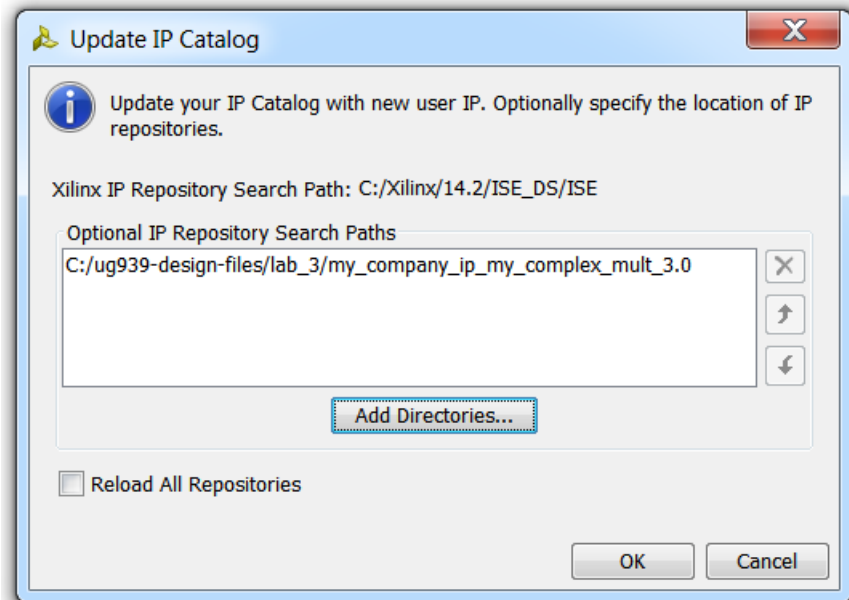
2. In the Project Manager area of the Flow Navigator (left side of the main window), click **IP Catalog**.
4. In the IP catalog window, right-click and select **Update IP Catalog**.



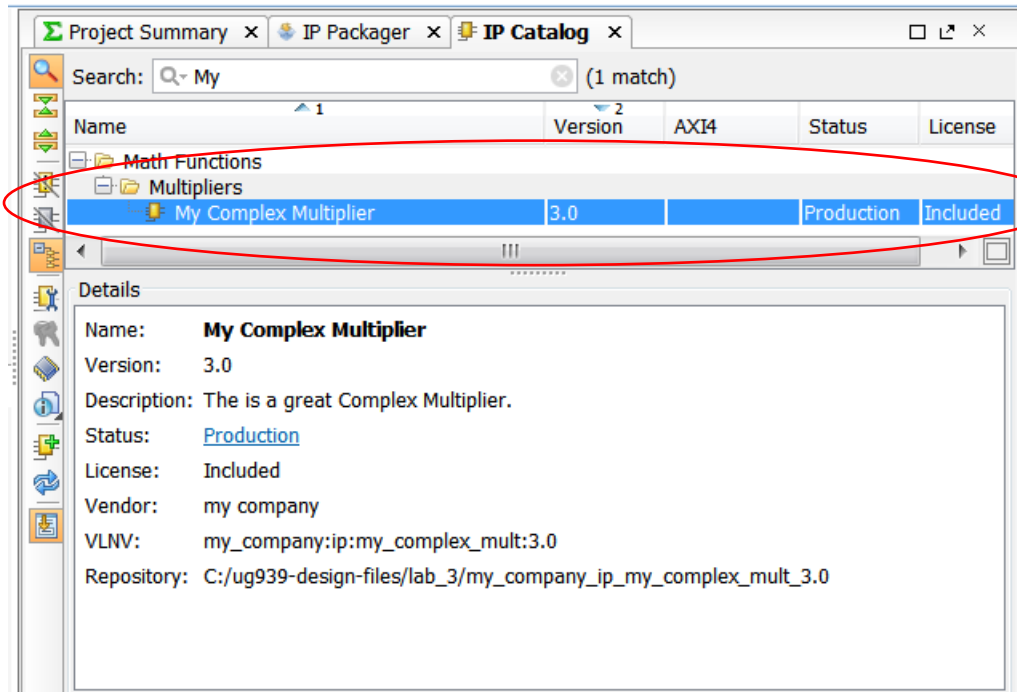
As shown in the figure below, in the Update IP Catalog dialog box, click **Add Directories**, select `C:/ug939-design-files/lab3/Xilinx.com_ip_wave_gen_1.0` as the IP Repository Search Path and click **OK**.



IMPORTANT: This search path leaf must contain the component `.xml` file for the IP

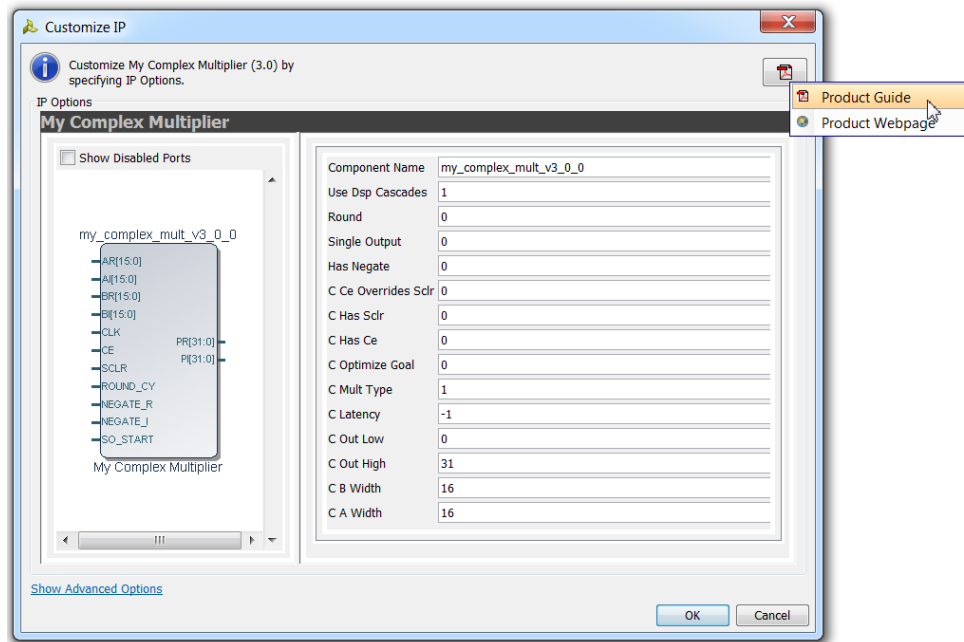


3. In the IP catalog pane, type **My** in the Search field to verify that **My Complex Mult** is added to the IP catalog. Also, verify that the metadata you entered is correctly displayed in the Details window, as shown below.



Step 7: Verify the New IP in the Customize IP Dialog Box

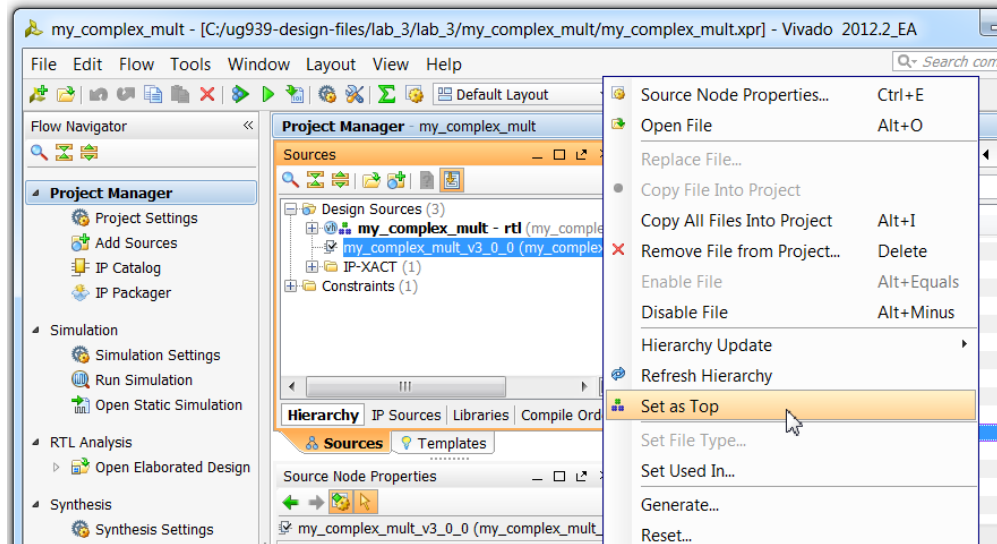
1. Double click on **My Complex Mult** in the IP catalog. Verify that the information presented is correct.



2. As shown above, right-click on the document button in the upper-left corner and select **Product Guide**. Verify that the document you added displays properly. Close the document.
3. Click **OK** on the Customize IP dialog box.

Step 8: Verify the New IP through Synthesis and Implementation

1. As shown below, right click on the **my_complex_mult_v3_0_0** instance in the Design Sources pane and select **Set as Top**.



2. Click **Run Synthesis** in the Flow Navigator pane.
3. After synthesis completes successfully, click **Run Implementation**.
4. Verify that implementation completes successfully.

Conclusion

In this exercise, you invoked IP packager on the **my_complex_mult** project. You packaged the complex multiplier design as an IP module and then added the new IP to the Vivado IP catalog. You then invoked the Customize IP dialog box on the new IP and verified that the information presented was correct. Finally, you added My Complex Multiplier to the Design Sources pane, set the IP as 'Top', and then ran the IP through Synthesis and Implementation.