Vivado Design Suite Tutorial

Creating and Packaging Custom IP

UG1119 (v 2014.3) October 15, 2014

XILINX
ALL PROGRAMMABLE™

Revision History

The following table shows the revision history for this document.

Date	Version	Changes
October 15	2014.3	Initial Release.

Table of Contents

Tutorial Introduction

This tutorial takes you through the required steps to create and package a custom IP in the Vivado® Design Suite IP packager tool.

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 following figure, 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.

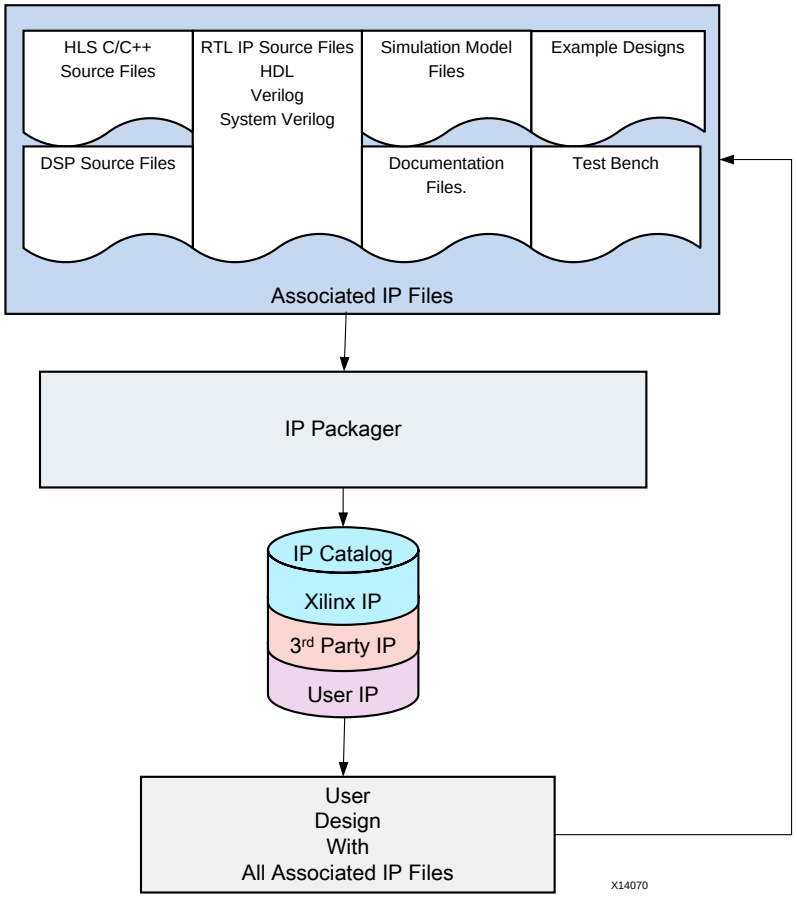

Figure 1: Vivado Design Suite IP Design Flow

The Vivado IP packager tool is a unique design reuse feature based on the IP-XACT standard. The IP packager tool 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.

VIDEO: *You can also learn more about the creating and using IP cores in Vivado Design Suite by viewing the quick take videos:* Configuring and Managing Custom IP *and* Customizing and Instantiating IP.

TRAINING*: Xilinx provides training courses that can help you learn more about the concepts presented in this document. Use these links to explore related courses:*

Essentials of FPGA Design

Embedded Systems Software

Hardware Requirements

This tutorial requires that the 2014.3 Vivado Design Suite software release or later is installed. The following partial list describes the operating systems that the Vivado Design Suite supports on x86 and x86-64 processor architectures:

Microsoft Windows Support:

- Windows 8.1 Professional (32-bit and 64-bit), English/Japanese
- Windows 7 and 7 SP1 Professional (32-bit and 64-bit), English/Japanese

Linux Support:

- Red Hat Enterprise Workstation 6.4 and 6.5 (32-bit and 64-bit)
- SUSE Linux Enterprise 11 (32-bit and 64-bit)
    - Cent OS 6.4 and 6.5 (64-bit)

See the *Vivado Design Suite User Guide: Release Notes, Installation, and Licensing* (UG973) for a complete list and description of the system and software requirements.

Software Requirements

This tutorial requires that you have installed:

Vivado Design Suite version 2014.3.

Tutorial Design Description

The small sample design used in this tutorial has a set of RTL design sources consisting of Verilog files, along with a PDF that describes how to add a document file to your IP.

Locating Tutorial Design Files

1. Download the zip file from the Xilinx website: https://secure.xilinx.com/webreg/clickthrough.do?cid=370138
2. Extract the zip file contents into any write-accessible location.

Introduction

In this lab, you define a new custom IP from an existing Vivado project, using the Create and Package IP wizard.

You start with an existing design project in the Vivado IDE, define identification information for the new IP, add documentation to support its use, and add the IP to the IP Catalog.

After packaging, you verify the new IP through synthesis in a separate design project.

The lab project contains Verilog source files for a simple UART interface.

Step 1: Open the Vivado Project

1. Launch Vivado.

On Linux:

   a. Change to the directory where the lab materials are stored: `cd <Extract_Dir>/lab_1`

   b. Launch the Vivado IDE: **`vivado`**

On Windows:

Launch the Vivado Design Suite IDE:

`Start > All Programs > Xilinx Design Tools > Vivado 2014.3 > Vivado 2014.3`[1]

Or, click the **Vivado 2014.3** desktop icon to start the Vivado IDE.

The Vivado IDE Getting Started page displays with links to open or create projects, and to view documentation. For either Windows or Linux, continue the lab from this point.

2. Click **Open Project**, and browse to: `<Extract_Dir>/lab_1/my_simple_uart`

3. Select the `my_simple_uart.xpr` project and click **OK**.

   The design loads, and you see the Vivado IDE in the default layout view, with the Project Summary information as shown in the figure below.

[1] Your Vivado Design Suite installation might have a different name on the Start menu.

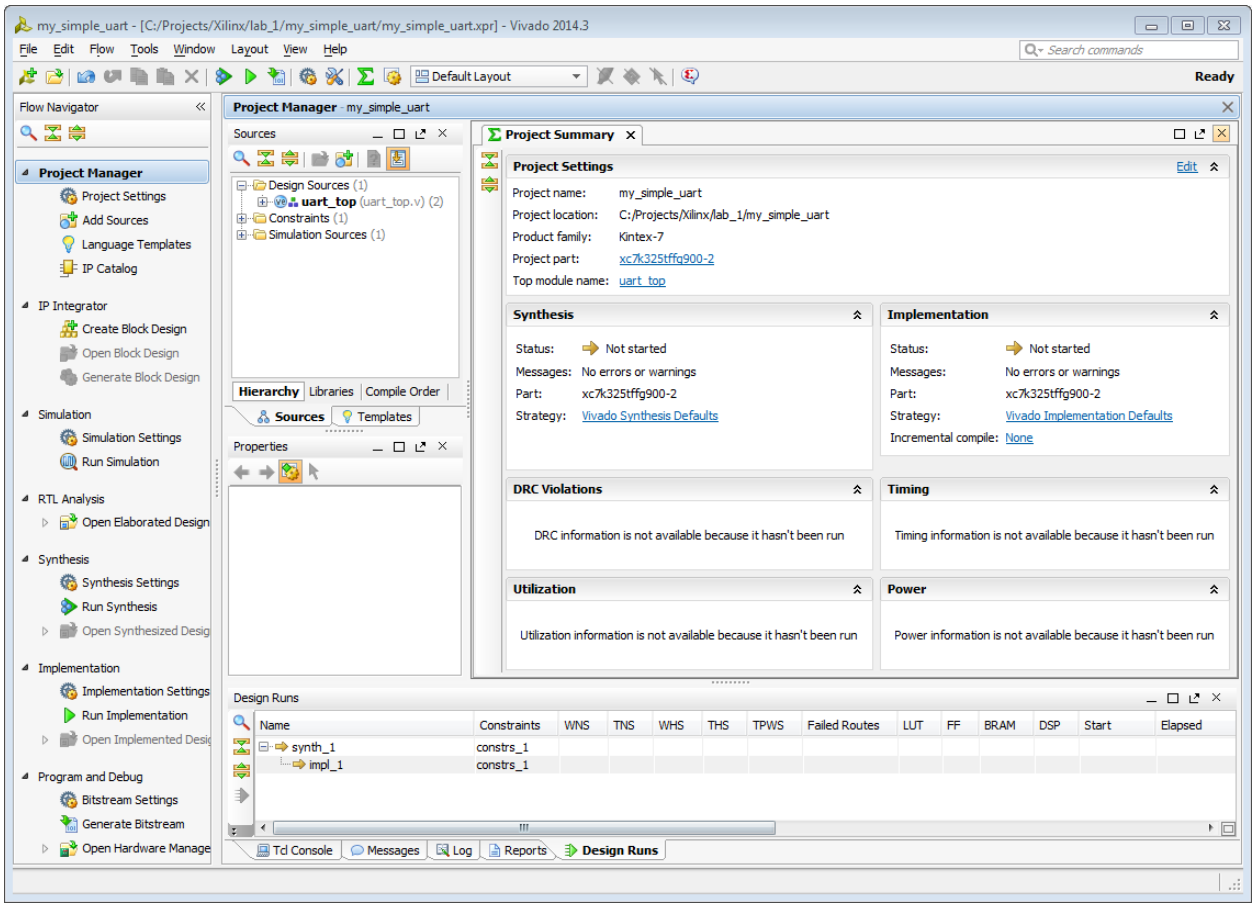

Figure 2: Project Default View Layout

Step 2: Preparing Design Constraints

The existing design includes timing constraints defined in an XDC file (`uart_top.xdc`). These constraints were defined for the UART design as a standalone design. However, when packaged as an IP, the design inherits some of the needed constraints from the parent design. In this case, you must modify the XDC file to separate constraints the IP requires when used in the context of a parent design, and the constraints the IP requires when used out-of-context (OOC) in a standalone capacity. This requires splitting the current XDC file.

You should prepare the design constraints prior to packaging the design for inclusion in the IP catalog; however, you can also perform these steps after packaging the IP.

IMPORTANT: *A synthesized design checkpoint (DCP) is created as part of the default Out-of-Context (OOC) design flow for IP packaging and use.*

To ensure that the packaged IP functions properly in the default Out-of-Context (OOC) design flow, the IP packaging must include a standalone XDC file to define all external clocking information for the IP. Vivado

synthesis uses the standalone XDC file in the Out-of-Context synthesis run to constrain the IP to the recommended clock frequency.

When used in the context of a top-level design, the parent XDC file provides the clock constraints and the standalone OOC XDC file is not needed.

For more information on the Out-Of-Context (OOC) design flow, and the use of the DCP file, see the *Vivado Design Suite User Guide: Designing with IP* (UG896).

TIP: *Depending on the function and use of the packaged IP, the design constraints may also have to be adjusted to ensure proper scoping. For more information, refer to Constraints Scoping in the Vivado Design Suite User Guide: Using Constraints (UG903).*

Analyze the Current Constraints Files

1. Open the target XDC file (`uart_top.xdc`) listed under the Constraints folder in the Hierarchy pane of the Sources window.

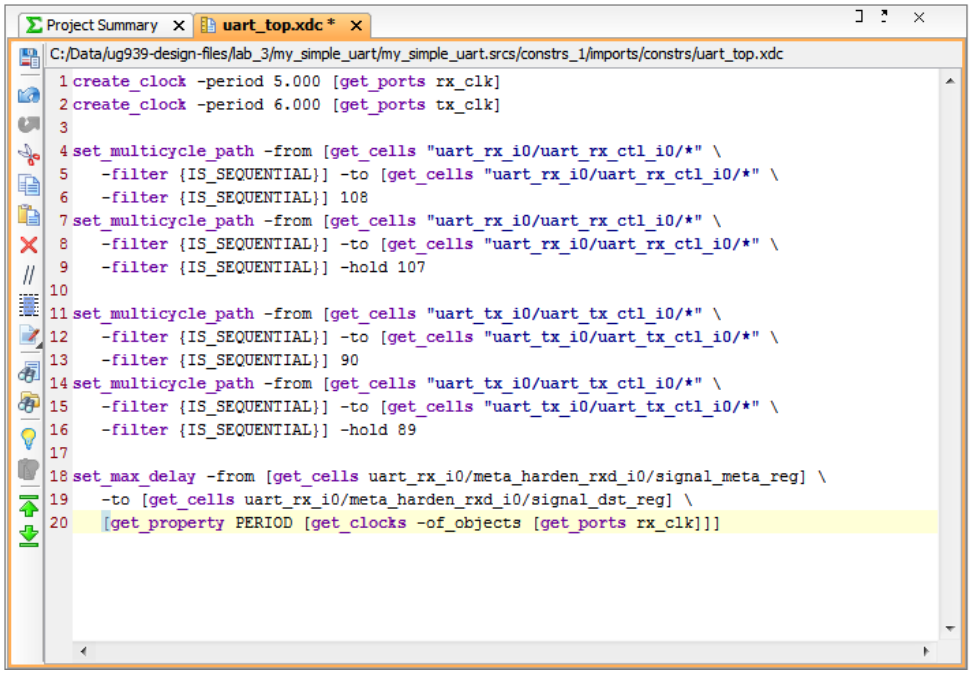

Figure 3: File Contents of uart_top.xdc

There are two items to take note of in the XDC file, as seen in Figure 2, above.

`create_clock` constraints (Lines 1 and 2)

`set_max_delay` constraint relying on the clock object period value (line 18).

Note: *The line numbers referenced in Figure 2 might differ from the line numbers in your XDC file because the constraints have been edited for easier viewing in this tutorial.*

2. Examine all `create_clock` constraints prior to packaging the new IP definition.

   If the created clock is internal to the IP (`GT`), or if the IP contains an input buffer (`IBUF`), the `create_clock` constraint should stay in the IP XDC file because it is needed to define local clocks. Clocks that are not internal, or local, to the IP should be moved from the IP XDC file to an OOC XDC file, because they are provided by the parent design.

For this example, you move the `create_clock` constraints on line 1 and 2 from the design XDC file to an OOC XDC file. When a user instantiates the IP you are packaging, from the IP catalog into a design, the IP inherits the clock definitions from the parent design.

The `set_max_delay` constraint is also noteworthy in that it has a dependency on the PERIOD property of defined clocks, (`get_clocks -of_objects`). This dependency is affected by the order of processing of the constraints of the IP and top-level design.

By default, when IP customizations are instantiated into a design, the Vivado IDE processes the XDC files of an IP before the XDC files of the top-level design. This is known as EARLY processing, and is defined by the PROCESSING_ORDER property on the XDC file.

The XDC files of the top-level design are marked for NORMAL processing by default. This means that the processing of XDC files for IP constraints happens before the top-level design constraints created by the user. However, in the case of the `set_max_delay` constraint, the dependency on the clock PERIOD will cause errors in processing the IP constraints early and defining the clock later. To resolve this issue, you will mark the XDC files of the UART IP for LATE processing.

TIP: *Xilinx delivered IP with "_clock" appended to the XDC filename are all marked for LATE processing.*

Create an Out-Of-Context (OOC) XDC file

1. In the Flow Navigator, or from the File menu, select **Add Sources**, or select the **Add Sources** button.

   The Add Sources dialog box opens.

2. Select **Add or Create Constraints**, click **Next**.
3. In the Add or Create Constraints pane, click the **Create File** button.
4. In the Create Constraints File dialog box, fill in the constraints file information with the following, as shown in the figure below.
   - File type: **XDC**
   - File name: `uart_top_ooc.xdc`
   - File location: `<Local to Project>`
5. Click **OK**.

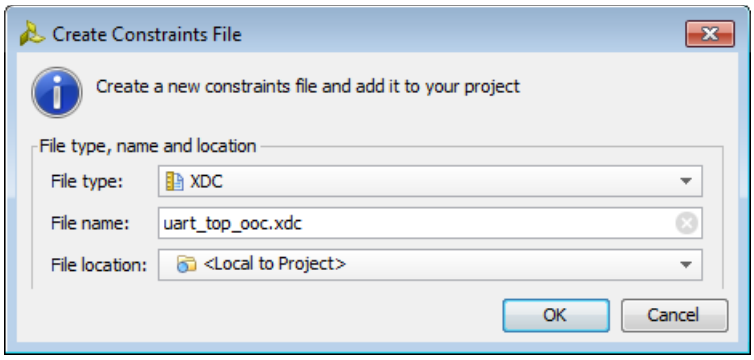

Figure 4: Create Constraints File Dialog Box

TIP: *For Xilinx delivered IPs, the Out-of-Context XDC file has "_ooc" appended to the filename. However, the* `USED_IN` *property of the file determines if it is an OOC XDC file, not the filename.*

6. Click **Finish** to complete the Add Sources dialog box.

   The new XDC file is created in the project and is displayed under the Constraints section in the Hierarchy pane of the Sources window.

   You now move the `create_clock` constraints from the XDC file of the original design (`uart_top.xdc`) into the OOC XDC file (`uart_top_ooc.xdc`).

7. In the Sources window, open the new OOC XDC file (`uart_top_ooc.xdc`) by double-clicking the file. The file is empty.

8. Cut and paste the `create_clock` constraints, from lines 1 and 2 of the IP XDC file (`uart_top.xdc`) into the empty OOC XDC file.

   The OOC XDC file should now contain only those two `create_clock` constraints.

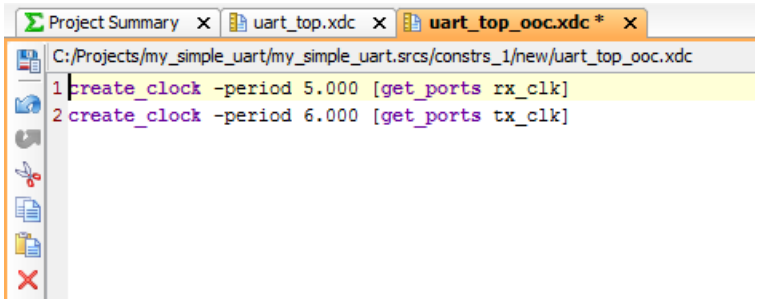

Figure 5: OOC XDC

9. Select the **Save File** button, , to save the updated contents of the OOC XDC file.

10. Check to be sure that the `create_clock` commands are removed from the IP XDC file (`uart_top.xdc`), and save the file.

As previously mentioned, the `create_clock` constraints are no longer needed because the clocks are defined by the parent design. The IP XDC file should now only contain the constraints as shown in the following figure. The OOC XDC file defines the clocks needed for standalone processing.

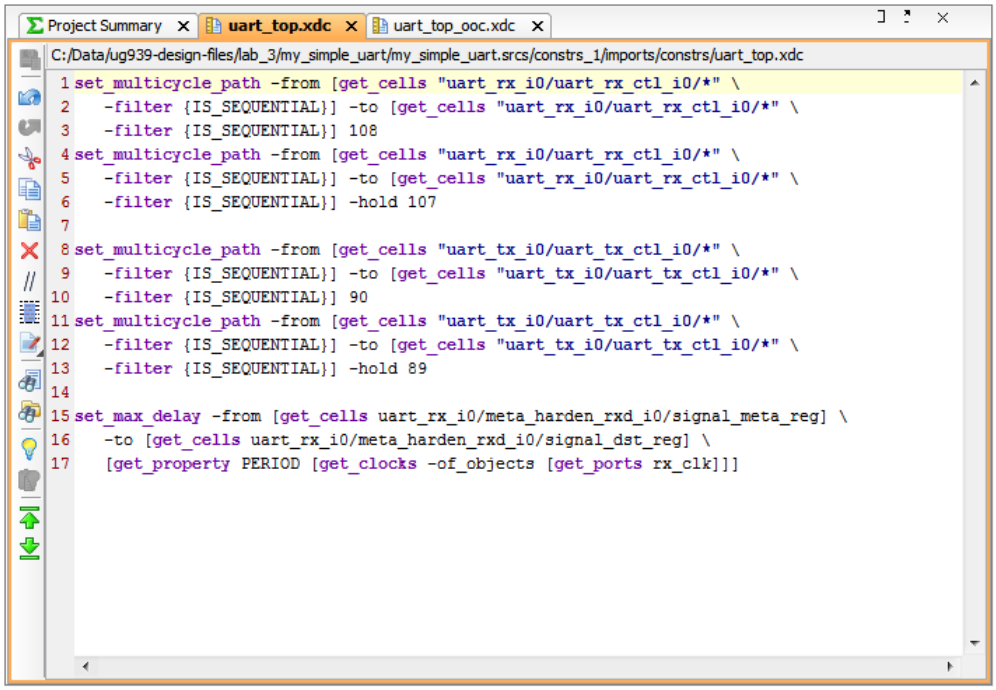

Figure 6: Updated uart_top.xdc

11. Close the two open XDC files.

    With the OOC and IP XDC files defined, you must set the `USED_IN` and `PROCESSING_ORDER` properties on the XDC files so that the Vivado Design Suite correctly processes the constraint files for the IP.

12. In the Hierarchy pane of the Sources window, select the OOC XDC file (`uart_top_ooc.xdc`) listed under the Constraints section.
13. Right-click the file, and select **Source File Properties** from the right-click menu.
14. From the Source File Properties window, scroll down and select the **USED_IN** property value to open the **Make Selection** dialog.
15. Select **out_of_context** in the unused values and select the **Move right** button,  , to add the value to the `USED_IN` property.

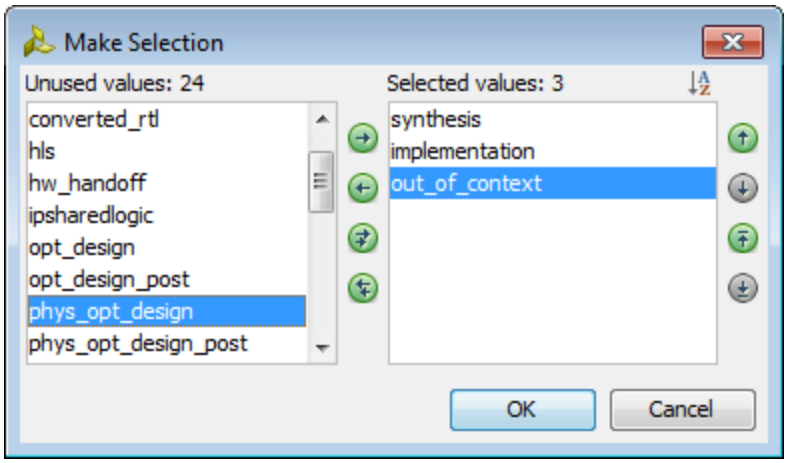

Figure 7: Make Selection dialog

16. *Optional*: You can optionally adjust the `USED_IN` property in the Tcl console. To set the `USED_IN` property of the OOC XDC file to include the "out_of_context" using the following Tcl command:

```
set_property USED_IN {synthesis implementation out_of_context} \
[get_files uart_top_ooc.xdc]
```

When the `USED_IN` property includes the `out_of_context` setting, the XDC file is only used for synthesis or implementation in Out-of-Context runs (`-mode out_of_context`).

IMPORTANT: *The* `USED_IN` *property for an OOC XDC file should be* `{synthesis implementation out_of_context}`. *If it is just* `out_of_context`, *it is not used during synthesis or implementation.*

Setting the Processing Order for the IP XDC

1. In the Hierarchy pane of the Sources window, select the IP XDC file (`uart_top.xdc`) listed under the Constraints section.
2. Right-click the file, and select **Source File Properties** from the right-click menu.
3. From the Source File Properties window, scroll down and change the **PROCESSING_ORDER** property value to **LATE**, as shown in the figure below.

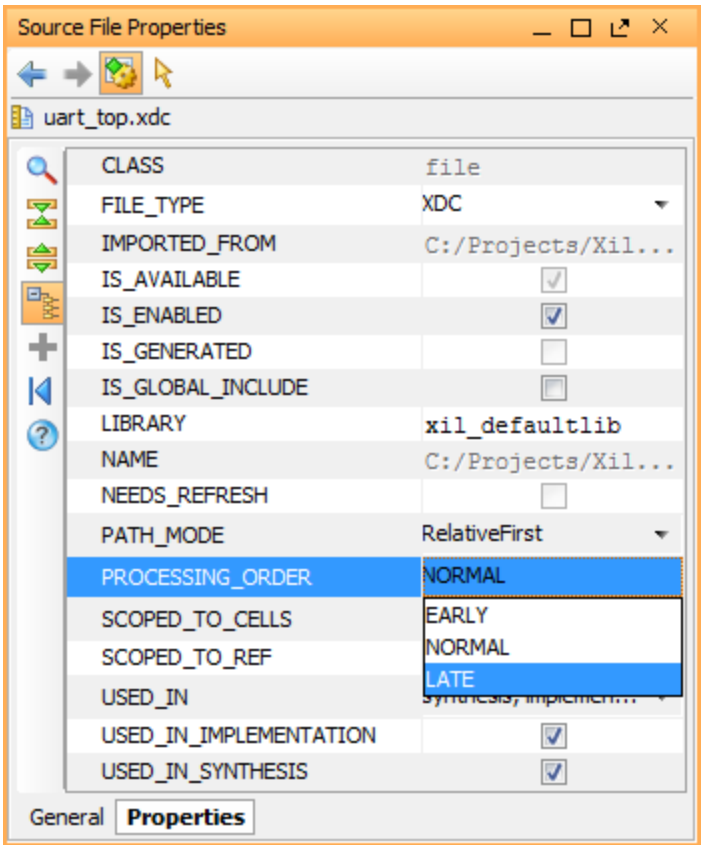

Figure 8: Source File Properties

The property value can also be changed in the Tcl Console with the following Tcl command:

```
set_property PROCESSING_ORDER LATE [get_files uart_top.xdc]
```

After completing the above steps, the XDC files are correctly prepared for packaging and the Out-Of-Context (OOC) design flow.

Step 3: Package the IP

After setting up the design and supporting constraint files, the next step is to create and package the new IP Definition, and add it to the IP Catalog.

1. From the Tools menu, select the **Create and Package IP** command to open the Create and Package IP Wizard.

   The Welcome window opens for the Create And Package New IP dialog box.

2. Click **Next**.

   The Choose Create Peripheral or Package IP dialog box opens, as shown in the following figure.

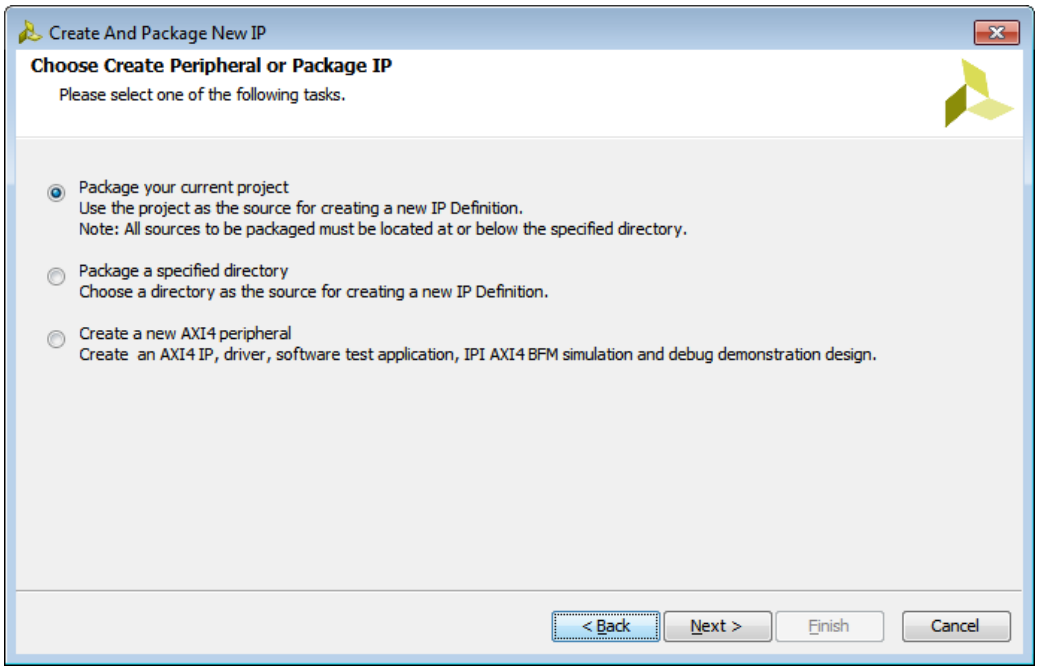

Figure 9: Choose Create Peripheral or Package IP Window

3. Select the **Package your current project** option to use the current project as the source for creating the new IP Definition.
4. Select **Next**.

   The Package Your Current Project dialog box opens, as shown in the following figure.

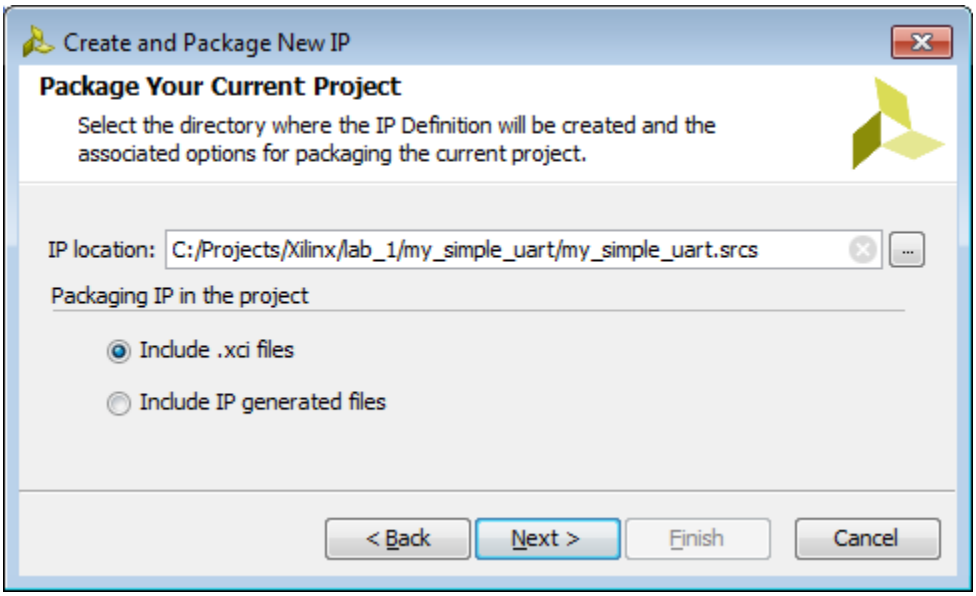

Figure 10: Package Current Project

5. Click **Next** to accept the defaults.

   The New IP Creation dialog box, as shown in the following figure, opens to summarize the information the wizard will automatically gather from the project.

Figure 11: Begin IP Creation

6. Click **Finish**.

   After the wizard has been completed, the Vivado IDE initially packages the current project as an IP for inclusion in the IP repository, and the Package IP dialog box appears to report success.

7. Click **OK**.

The Package IP window opens and displays the basic IP package in a staging area for editing and repackaging, as seen in the following figure.

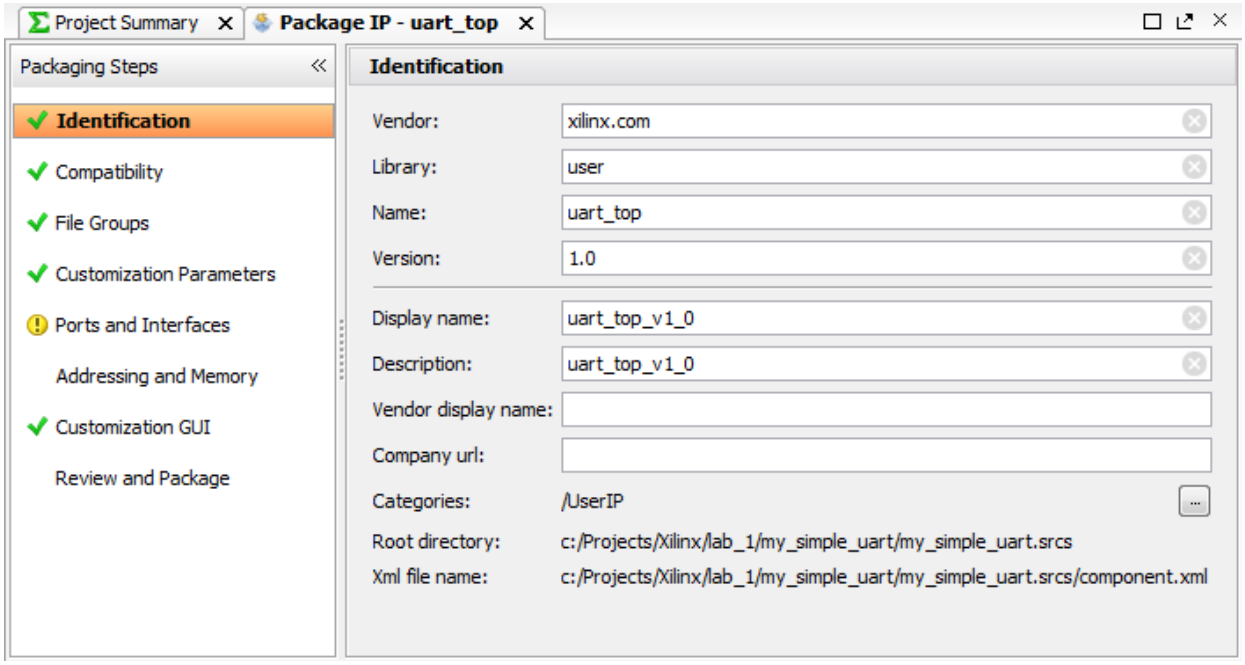

Figure 12:Editing the Default IP Definition

Modify the IP Definition

The Package IP window shows the current IP identification information, including Vendor, Library, Name, and Version (VLNV) attributes of the newly packaged IP.

1. In the Package IP window, select the **Identification pane** in the left side panel, and fill in the right side with the following information:
   - **Vendor:** `my_company`
   - **Name:** `my_simple_uart`
   - **Display name:** `My Simple UART`
   - **Description:** `My simple example UART interface`
   - **Vendor display name:** `My Company`
   - **Company url:** `http://www.my_company_name.com`
2. For the **Categories** option, select the browse button, ..., to open the **Choose IP Categories** dialog box, as shown in following figure.

   The Choose IP Categories dialog box lets you select various appropriate categories to help classify the new IP definition. When the IP definition is added to the IP Catalog, the IP is listed under the specified categories.

3. Select the **Serial Interfaces** box under **Communications & Networking** because the IP is a UART interface.
4. Click **OK**.

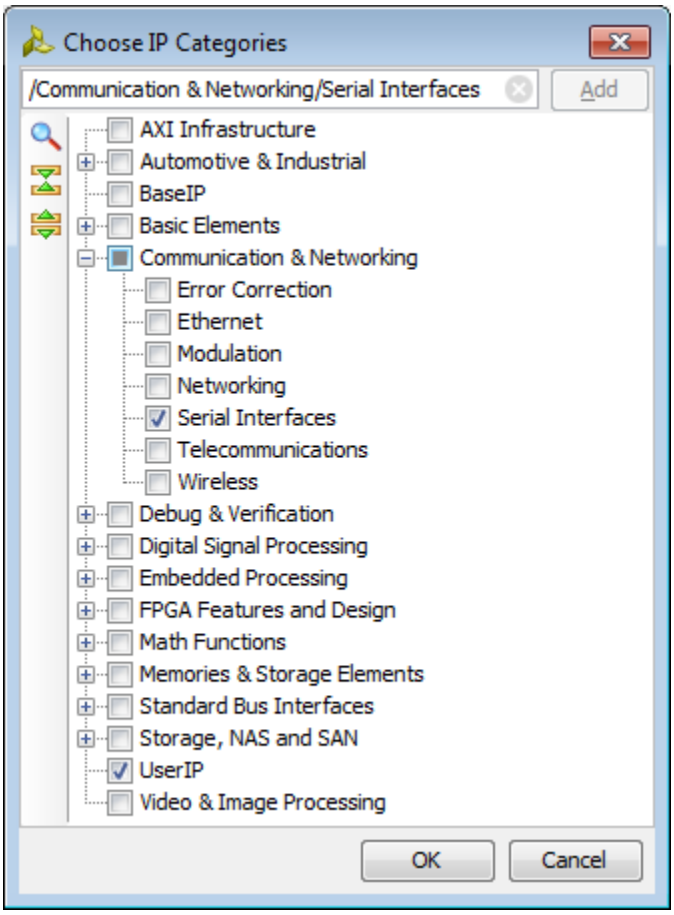

Figure 13: Choose IP Categories

Add Product Guide to the IP

1. On the left side of the Package IP window, select the **File Groups** item to display the File Groups panel on the right side.

   The File Groups panel provides a listing of the files that to be packaged as part of the IP.

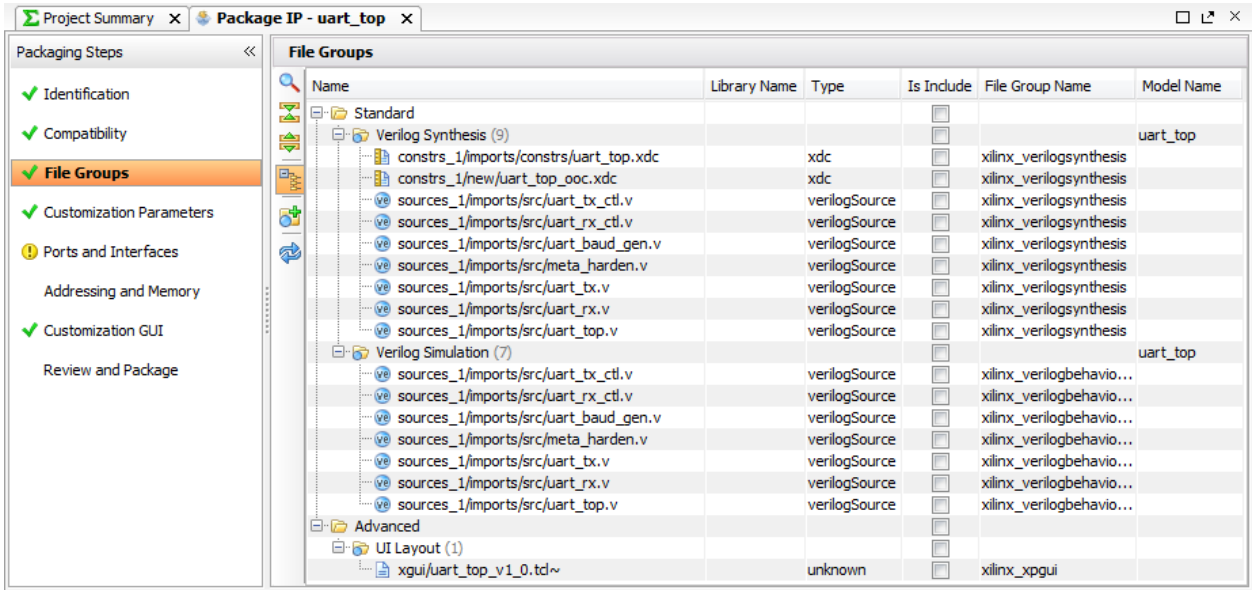

Figure 14: File Groups

2. Open the Messages window, and review the IP Packager messages as seen in the figure below.

   The IP Packager messages inform you of the state of the IP. The File Groups Wizard message indicates that the IP definition does not include any documentation.

   The Customization Parameters Wizard informs you that specific parameters of the IP do not have range values.

   As `INFO` messages, these are quick checks of the IP definition that do not prevent you from moving forward if you choose. However, in the next step you add the product guide to the IP definition.

   The Ports and Interfaces wizard has two warnings related to the inferred single-bit clock interfaces inferred by the IP Packager for missing `ASSOCIATED_BUSIF` parameters. These parameters are required for AXI interfaces in IPI, but can be ignored for this exercise.

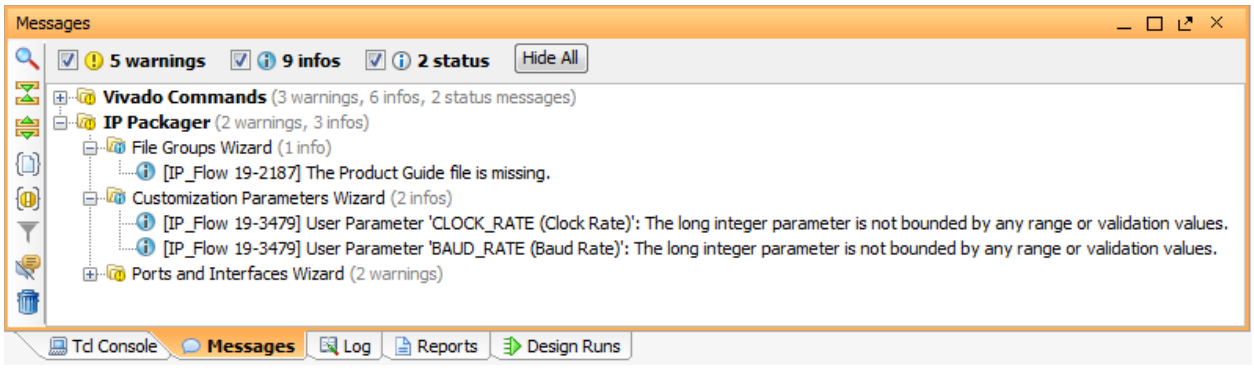

Figure 15: IP Packager Messages

3. In the Package IP window, right-click in the File Groups panel, and select **Add File Group**.

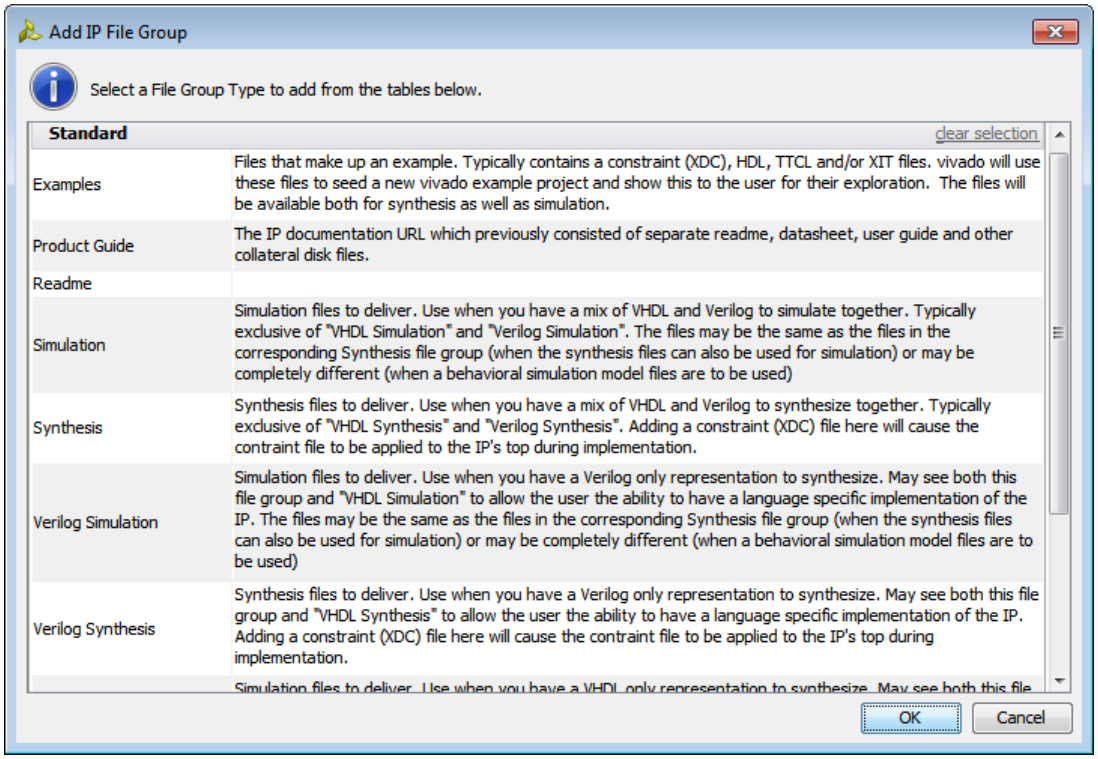

Figure 16: Add IP File Group – Product Guide

4. In the Add IP File Group dialog box, select **Product Guide** from the Standard File Groups section, as shown in in the previous figure.
5. Click **OK**.

   The IP File Groups pane now updates with the Product Guide group in the list. There is a 0 next to the Product Guide name as there are 0 files added to the newly created group.

Note: *A critical warning opens when the Product Guide file group is added, noting that the file group is empty.*

6. Right-click the **Product Guide** file group, and select **Add Files**.
7. In the opened Add IP Files (Product Guide) dialog box, click **Add Files**.
8. Browse to `<Extract_Dir>/lab_1/my_simple_uart/docs`, and select **All Files** in the **Files of type**: entry line.
9. Select `my_simple_uart_product_guide.pdf`, and click **OK.**
10. In the Add IP Files (Product Guide) dialog box, ensure that **Copy sources into project** is selected.

    The option ensures that the file is imported in the project sources directory to ensure the file is remotely referenced by the IP Packager.

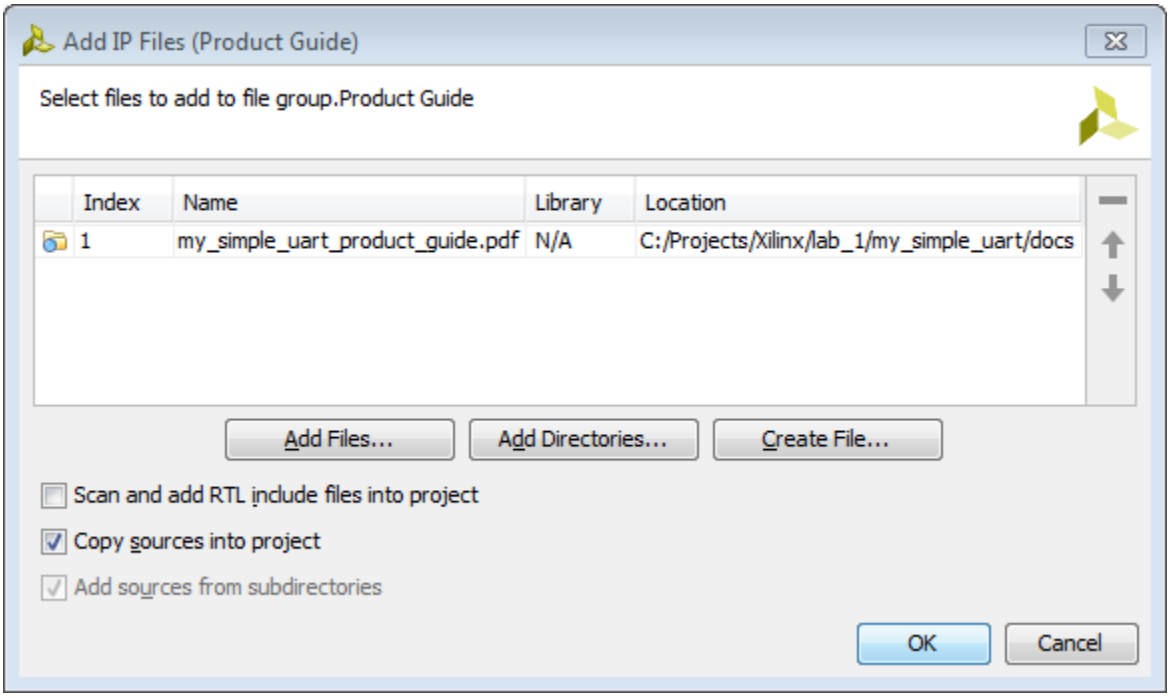

Figure 17: Add Product Guide

11. Click **OK**.

    The PDF file of the Product Guide is added to the files defined as part of the IP, and the Critical Warning is resolved.

Review and Package the IP

The custom IP was initially packaged at the end of the Create and Package IP wizard, but since changes were made in the Package IP window, the custom IP will have to be repackaged for the changes to take effect.

1. On the left side of the Package IP window, select the **Review and Package** panel.

   The Review and Package panel provides a summary of the IP being packaged, as shown in the following figure.

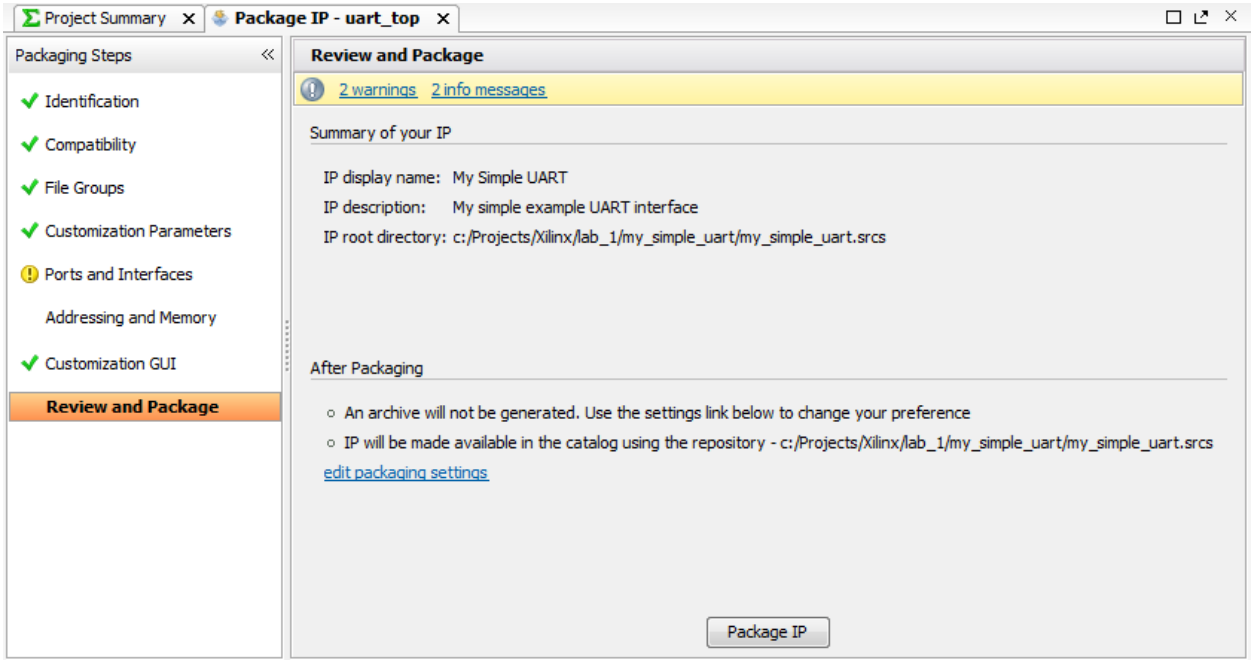

Figure 18: Review and Package IP

   With default settings of the current project, Vivado will not generate an archive for this IP after packaging. This is reflected in the **After Packaging** section of the Review and Package panel of the Package IP window.

   Make a note of the location of the IP repository in the After Packaging section. This will be needed to validate the custom IP in the next step.

2. In the Package IP window, **click Package IP**, to package the current project, add it to the IP Catalog.
3. After the packaging process completes, close the Vivado project.

Step 4: Validate the New IP

With the new custom IP definition packaged and added to the IP Catalog, you can validate that the IP works as expected when added to designs. To validate the IP, add a new customization of the UART IP to a project, and synthesize the design.

1. From the Vivado IDE Getting Started page, select **Manage IP > New IP Location** to create a new project.

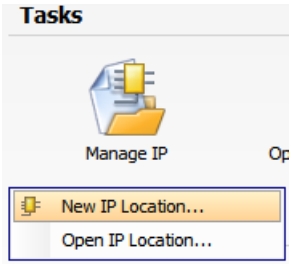

Figure 19: New Manage IP Project

TIP: *You can use either an RTL project or a Manage IP project to validate IP.*

2. Click **Next** at the New IP Location dialog box that opens.

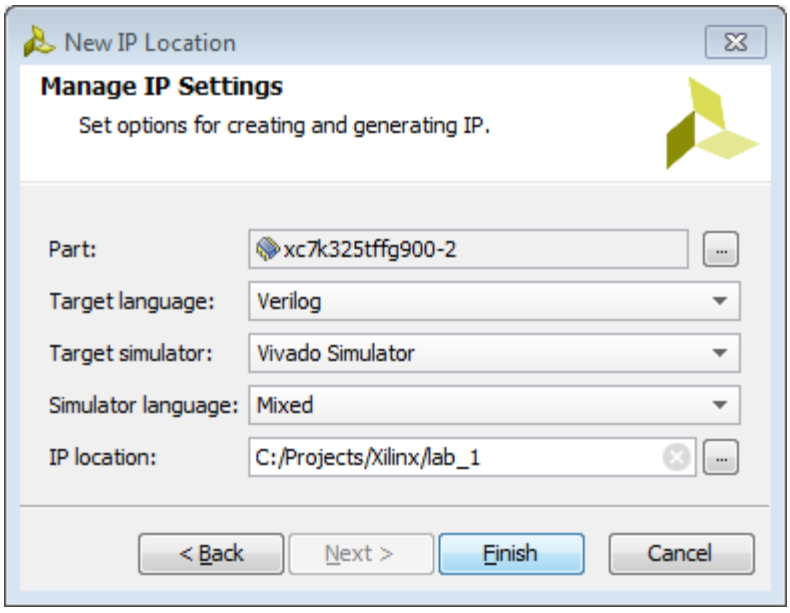

Figure 20: Manage IP Settings

3. In the Manage IP Settings dialog box, set the following options as they appear in the previous figure.
   - Part: **xc7k325tffg900-2**
   - Target language: **Verilog**
   - Target Simulator: **Vivado Simulator**
   - Simulator Language: **Mixed**
   - IP Location: **<Extract_Dir>/lab_1**
4. Click **Finish** to create the Manage IP project.

   A new Manage IP project opens in the Vivado IDE. The IP Catalog opens automatically in a Manage IP project; however, the IP Catalog does not contain the repository used to package the custom UART IP.

   You now add the IP repository to the IP Catalog at this time.
5. In the IP Catalog window, right-click and select **IP Settings**.

   The **Tools > Project Settings > IP** dialog box opens.
6. In the Repository Manager tab, click the **Add Repository** button to open the IP Repositories Dialog Box.
7. In the IP Repositories dialog box, **browse** to and **select** the following location:

   `<Extract_Dir>/lab_1/my_simple_uart/my_simple_uart.srcs`
8. Click **Select** to add the selected repository.

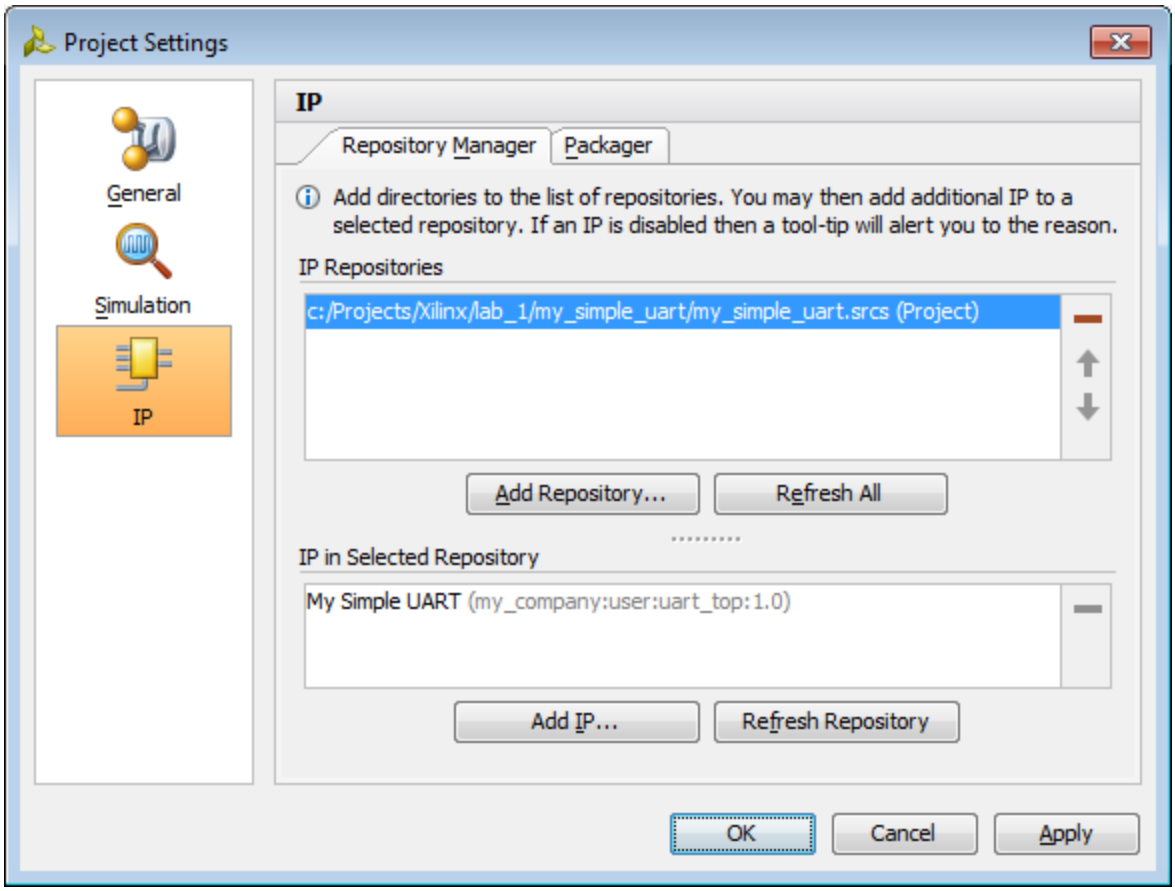

Figure 21: Manage IP Repository

As seen in the previous figure, the added location displays in the **IP Repositories** section, and any packaged IP found in the repositories is displayed under the **IP in Selected Repository**. The **My Simple UART** IP definition, which you packaged in Step #3, is listed.

9. Press **OK** to add the IP repository to the IP Catalog and close the dialog box.

TIP: *To define a custom IP repository for use across multiple design projects you can use the **Tools** > **Options** command in the Vivado IDE to set the Default IP Repository Search Paths under the General options. The default IP repository search path is stored in the* `vivado.ini` *file, and added to new projects using the* `IP_REPO_PATHS` *property for the current_fileset:*

```
set_property IP_REPO_PATHS {…} [current_fileset]
```

See the Vivado Properties Reference Guide (UG912) for more information.

---

10. In the search field at the top of the **IP Catalog**, type **UART**.

    The My Simple UART is reported under the UserIP and Serial Interfaces categories that it was previously assigned to during packaging.

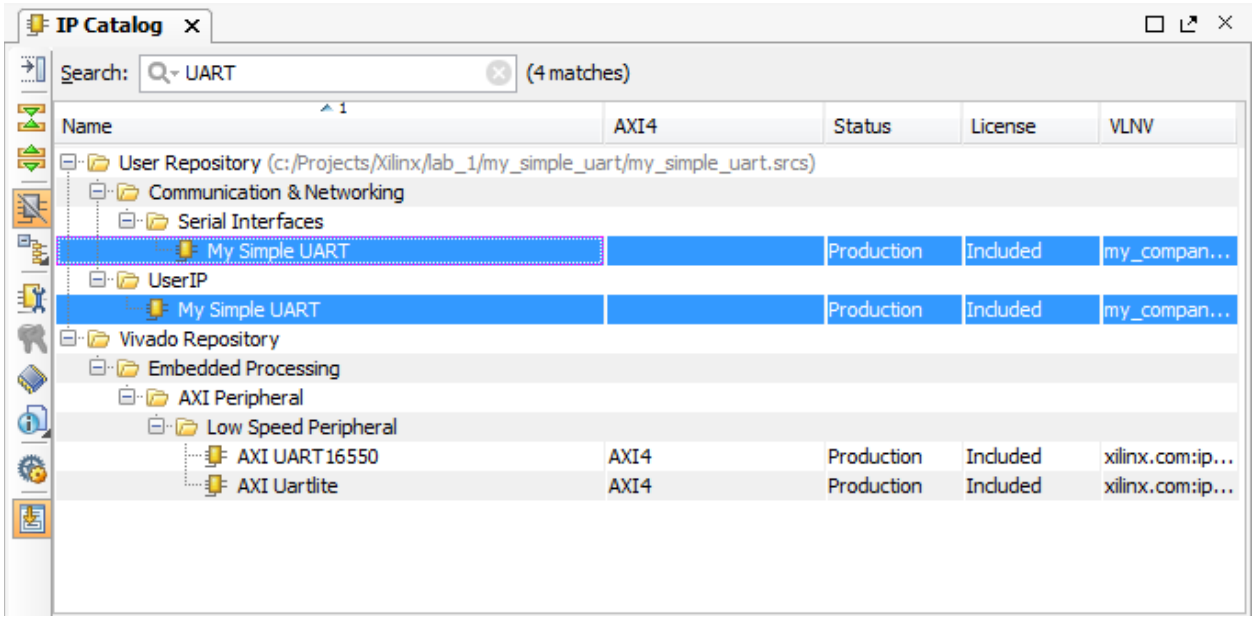

Figure 22: Search IP Catalog for UART

11. Select the **My Simple UART** by clicking it under either the UserIP or Serial Interfaces category.

    Examine the Details pane of the IP Catalog window, as shown in the following figure. Notice the details match the information provided when you packaged the IP.

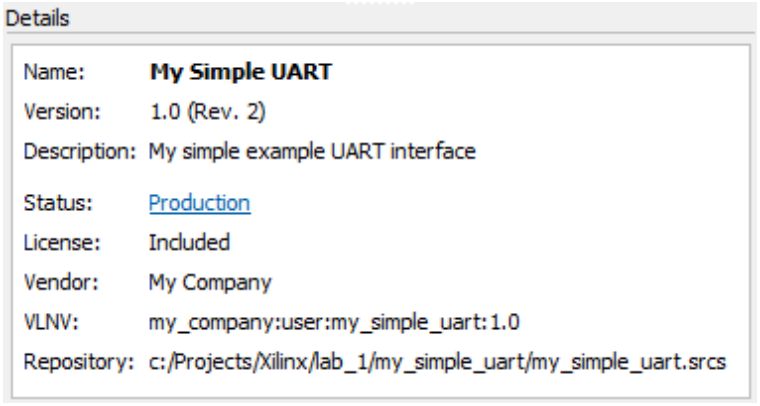

Figure 23: My Simple UART - Details

12. Double click **My Simple UART** in the IP Catalog to open the Customize IP dialog box, as shown in the following figure.

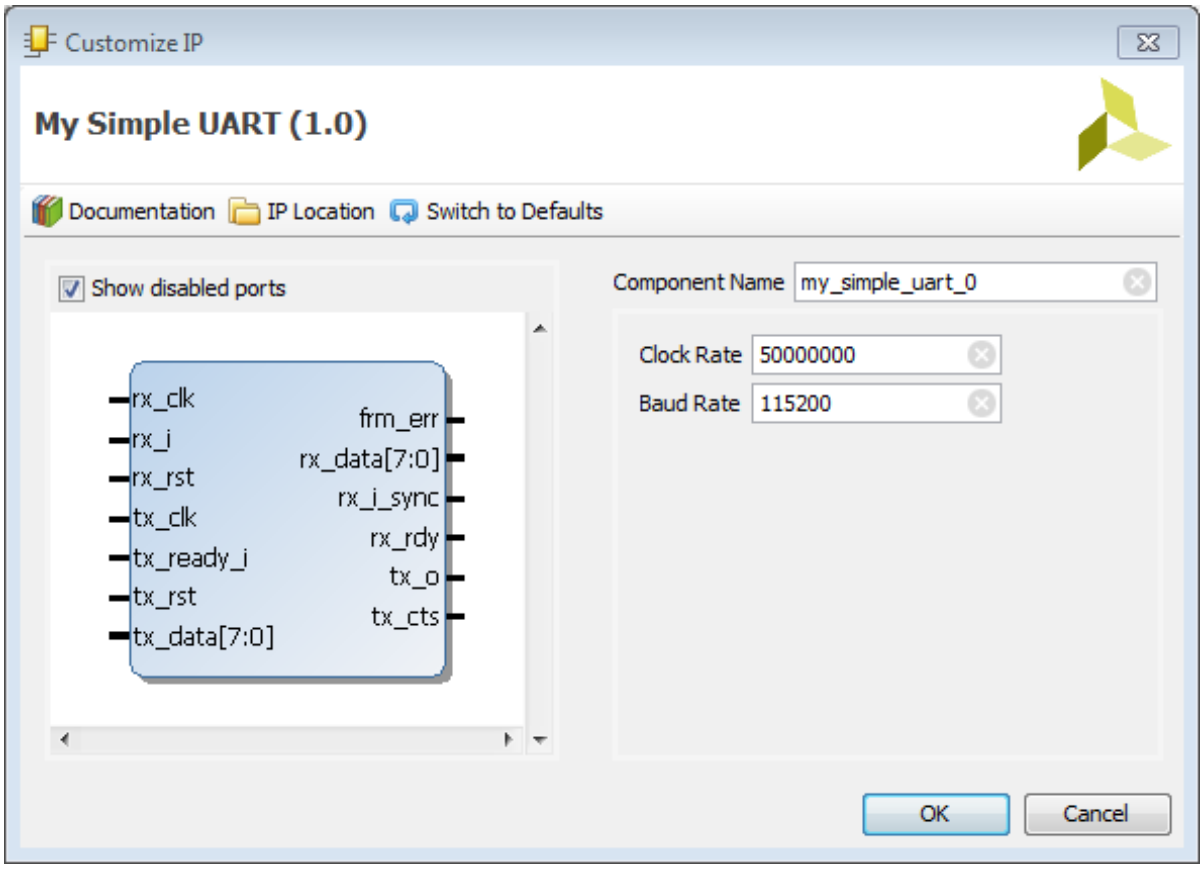

Figure 24: Customize IP – My Simple UART

13. *Optionally*: In the **Customize IP** dialog box, click **Documentation** and open the **Product Guide**.
14. Click **OK**, accepting the default Component Name and other options.

    The customized IP is added to the current project, and is shown in the IP Sources window. In addition, the Generate Output Products dialog box opens, as shown in the following figure.

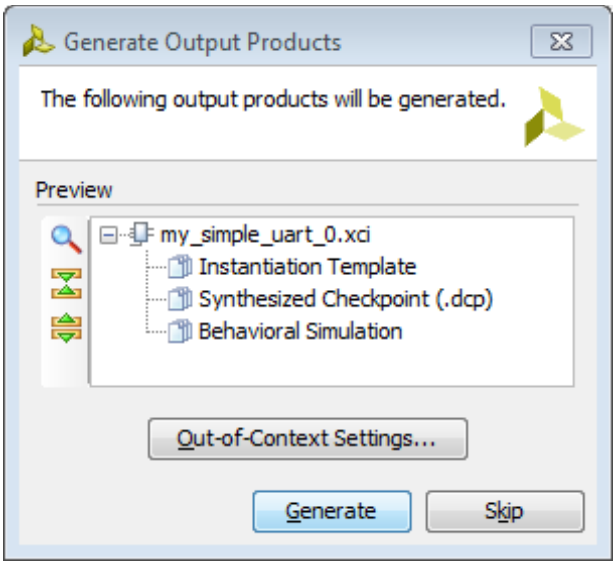

Figure 25: Generate Output Products

15. Click **Generate**.

    This generates the various files required for this IP in the current Manage IP project, and launches an Out-of-Context synthesis run for the IP to create a DCP.

    Recall this OOC synthesis run uses the OOC XDC file that defines the needed clocks for the standalone IP.

    The Generate Output Products dialog reappears to report the output products were generated successfully.

16. Click **OK**.

17. Examine the IP Sources window and the various design and simulation source files that are added to the project.
18. In the Design Runs window, shown in the following figure, verify that the Out-Of-Context synthesis run was successful.

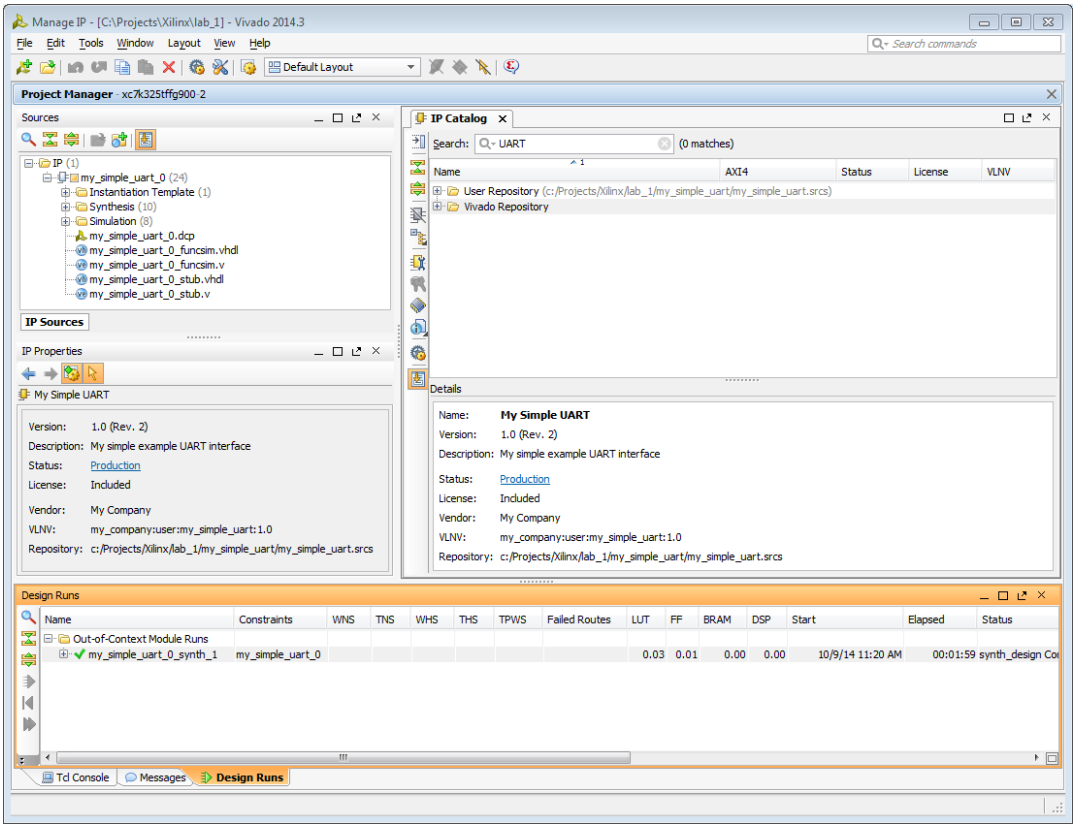

Figure 26: Validate IP in Managed IP Project

Conclusion

In this Lab, you did the following:

Used the Create and Package IP Wizard to create a custom IP definition for the tutorial project, `my_simple_uart`.

Setup the XDC files to support the processing order requirements as well as Out-Of-Context synthesis.

Validated the packaged IP by creating a Managed IP project, and then adding the new IP repository to the IP Catalog.

Created a customization of the IP, and generated a DCP of the IP to validate that the IP definition was complete and included all the necessary files to support using the IP in other designs.

Please Read: Important Legal Notices

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 Xilinx’s limited warranty, please refer to Xilinx’s Terms of Sale which can be viewed at http://www.xilinx.com/legal.htm#tos; 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 such critical applications, please refer to Xilinx’s Terms of Sale which can be viewed at http://www.xilinx.com/legal.htm#tos.