

# Vivado Design Suite User Guide

## *Designing with IP*

UG896 (v2014.4) November 19, 2014

## Revision History

Date	Revision	Changes
11/19/2014	2014.4	<p>Added information about regenerating targets for LogicCORE IP when a new license is acquired.</p> <p>Added note after <a href="#">Example Tcl Script for Non Project Mode</a>, page 63.</p>
10/01/2014	2014.3	<p>Added tip that explains color coding for text boxes.</p> <p>Added New Group by option in IP Catalog Features.</p> <p>Added content to Using the IP Catalog. Changed Filtering IP.</p> <p>Added Resolving Duplicate IP. New Alliance Partner section - Partner Alliance IP. Added references to ug1118 and ug1119 to Using the Repository Manager. Added new Using IP Project Settings. Changed Using the Packager Settings.</p> <p>Added Recommendation regarding hexadecimal to decimal conversion. Added information on MIG in Creating a Memory Interface Generator Customization.</p> <p>Added note on NGC format no longer supported.</p> <p>Changed content in Constraint File Processing Order.</p> <p>Revised Determining Clocking Constraints and Interpreting Clocking Messages, to reflect new in_context.xdc rules and the deprecation of dont_buffer.xdc.</p> <p>Added content to Debugging IP.</p> <p>New Test Bench section Using a Test Bench for IP.</p> <p>New locked and is managed Icons. New description of Locked.</p> <p>Changed Simulating IP, page 66.</p> <p>Added references to ug1118 and ug1119 in Chapter 7, Working with Revision Control. Added directories and descriptions to Appendix A, IP Files and Directory Structure.</p> <p>Changed list of IP that support Board Flow in Appendix B, Using the IP Board Flow.</p> <p>Added Appendix C, Editing IP, on how to unlock an IP to edit and to create a DCP. Added new Figures: Changed the IP-Centric Design flow figure. Added Existing IP figure. New FIFO Generator figure. Added Standards and Third-Party Documentation, and Training Resources.</p>
06/01/2014	2014.2	<p>Added Creating and Packaging IP Chapter.</p> <p>Removed the Run <code>-upgrade_core</code> option when opening IP Packager option from the Project Settings menu.</p>
04/01/2014	2014.1	<p>Added a section in IP Basics on copying an IP.</p> <p>Documented how IP require top- level clocks.</p> <p>Reorganized the Tcl Commands in the Tcl Command chapter to go by flow.</p> <p>Documented the dont_touch/dont_buffer/in_context XDC files in the IP Constraints section of IP Basics.</p> <p>Added detail on working with IP and third-party simulators.</p> <p>Enhanced description of set of possible IP output products, directory paths, and important files common to all IP.</p>

# Table of Contents

Revision History .....	2
<b>Chapter 1: IP-Centric Design Flow</b>	
Introduction .....	5
IP Terminology .....	7
IP Catalog .....	8
IP Packager .....	8
<b>Chapter 2: IP Basics</b>	
Introduction .....	9
IP Catalog Features .....	10
Using the IP Catalog .....	11
Synthesis Options for IP .....	14
Generating Output Products .....	19
Using IP Project Settings .....	21
Creating an IP Customization .....	24
Creating a Memory Interface Generator Customization .....	26
Re-Customizing Existing IP .....	27
Manually Generating Output Products .....	27
Instantiating an IP .....	28
Adding Existing IP to a Project .....	30
Upgrading IP .....	32
Copying an IP .....	36
Understanding IP States Within a Project .....	37
Understanding IP Constraints .....	41
Using Fee-Based Licensed IP .....	46
Debugging IP .....	48
Using a Test Bench for IP .....	51
<b>Chapter 3: Using Manage IP Projects</b>	
Introduction .....	54
Managed IP Features .....	54
Using the Manage IP Flow .....	55

## Chapter 4: Using IP Example Designs

Introduction .....	59
Opening an Example Design .....	59

## Chapter 5: Using Xilinx IP with Third-Party Synthesis Tools

Introduction .....	61
Synthesis Flow .....	61

## Chapter 6: Simulating IP

Introduction .....	64
Delivering IP Simulation Models .....	64
Simulating IP .....	66
Tcl Commands for IP Simulation .....	67
Using Cadence Incisive Enterprise Simulator and Synopsys VCS MX Simulator .....	68

## Chapter 7: Working with Revision Control

Introduction .....	69
Using Revision Control .....	69

## Chapter 8: Tcl Commands for Common IP Operations

Introduction .....	72
Using IP Tcl Commands In Design Flows .....	72
Tcl Commands for Common IP Operations .....	73
Example IP Flow Commands .....	75

## Appendix A: IP Files and Directory Structure

Introduction .....	80
IP-Generated Directories and Files .....	80

## Appendix B: Using the IP Board Flow

Using the IP Board Flow .....	82
-------------------------------	----

## Appendix C: Editing IP

Introduction .....	86
Modifying IP Sources .....	87

## Appendix D: Additional Resources and Legal Notices

Xilinx Resources .....	89
Standards and Third-Party Documentation .....	90
Training Resources .....	91
Please Read: Important Legal Notices .....	91

# IP-Centric Design Flow

---

## Introduction

The Xilinx® Vivado® Integrated Design Environment (IDE) provides an IP-centric design flow that lets you add IP modules to your design from various design sources. Central to the environment is an extensible IP Catalog that contains Xilinx-delivered *Plug-and-Play* IP. The IP Catalog can be extended by adding the following:

- Modules from System Generator for DSP designs (MATLAB® from Simulink® algorithms)
- Vivado High-Level Synthesis (HLS) designs (C/C++ algorithms)
- Third-party IP
- Designs packaged as IP using the Vivado IP packager

[Figure 1-1, page 6](#) illustrates the IP-centric design flow.

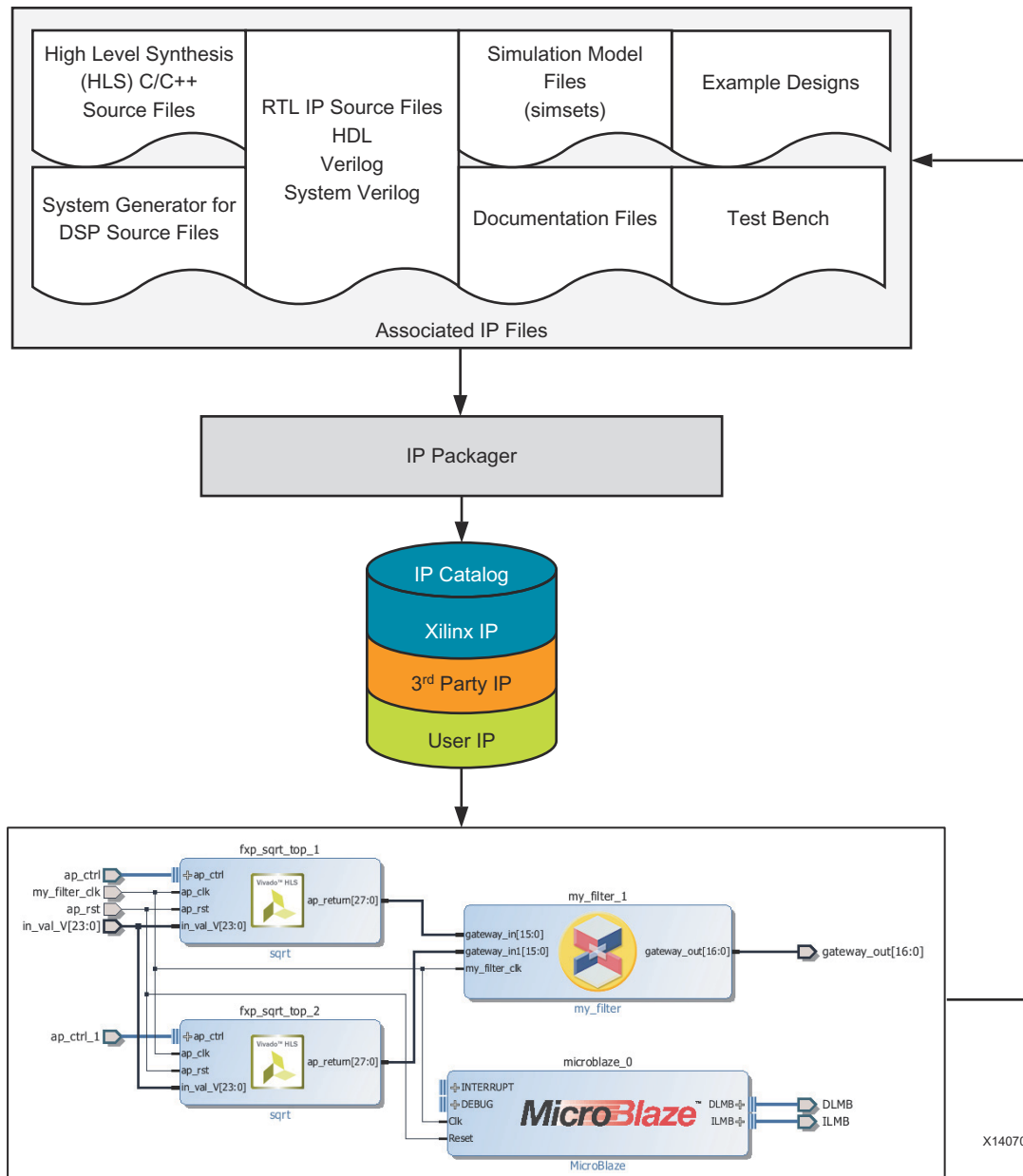


Figure 1-1: IP-Centric Design Flow

**Note:** In some cases, third-party providers offer IP as synthesized EDIF netlists. You can load these files into a Vivado IDE design using the **Add Sources** command.

The available methods to work with IP in a design are as follows:

- Use the Managed IP Flow to customize IP and generate output products, including a synthesized design checkpoint (DCP).
- Use IP in either Project or Non-Project modes by referencing the created Xilinx Core Instance (XCI) file, which is a recommended method for large projects with many team members.
- Access the IP Catalog in a project to create and add IP to design. Store the IP either inside the project or save it externally to the project, which is the recommended method for projects with small team sizes.
- Create and customize IP and generate output products in a Non-Project script flow, including generation of a DCP.

The *Vivado Design Suite Tutorial: Designing with IP* (UG939) [Ref 16] provides instruction on how to use Xilinx IP in Vivado.



**VIDEO:** See the following Quick Take Videos for more information: [Customizing and Instantiating IP](#), [Vivado IP Constraints Overview](#), [Configuring and Managing Reusable IP in Vivado](#), and [Managing Vivado IP Version Upgrades](#).



**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](#) and [Embedded Systems Software Design](#).

## IP Terminology

The Vivado IDE uses the following terminology to describe IP, where it is stored, and how it is represented

- **IP Definition:** The description of the IP-XACT characteristics for IP.
- **IP Customization:** Customizing an IP from an IP definition, resulting in an XCI file.
- **IP Location:** A directory that contains one or more customized IP.
- **IP Repository:** A unified view of a collection of IP definitions.
- **Output Products:** Generated files produced for an IP customization. They can include HDL, constraints, and simulation targets. The XCI file stores the configuration and constraint options that are user-specified during customization. During generation, these customizations are used to produce the files that are used during synthesis and simulation.

---

## IP Catalog

The IP Catalog allows for the exploration of Xilinx *Plug-and-Play* IP, as well as other IP-XACT compliant Intellectual Property. This can include designs that you package as IP. See [Chapter 2, IP Basics](#) for more information.

---

## IP Packager

The Vivado IP packager lets you create plug-and-play IP and add that IP to the extensible Vivado IP Catalog. The IP packager wizard, is based on the IEEE Standard for IP-XACT (IEEE Std 1685), *Standard Structure for Packaging, Integrating, and Reusing IP within Tool Flows* [Ref 1].

After you have assembled your Vivado IDE user design, the IP packager lets you turn your design into a reusable IP module that you can then add to the Vivado IP Catalog, and that others can use for design work. You can use packaged IP within a Project Mode-based IP flow.

You can locate the IP within the project or use a remote location (recommended).

See the *Vivado Design Suite: Creating and Packaging Custom IP* (UG1118) [Ref 1] for more information about using the packaging feature.

The *Vivado Design Suite Tutorial: Creating and Packaging Custom IP* (UG1119) [Ref 2] provides labs with design solutions included that show you how to use the packaging feature.



# IP Basics

---

## Introduction

This chapter describes the basic features of the IP catalog, how to create and instantiate IP, and how to generate output products for use within a design.

Working with Xilinx® IP consists of first creating a customization of the IP. You can create an IP customization in various ways using the Vivado® Integrated Design Environment (IDE), as follows:

- Directly in a project using the IP Catalog
- Using the Manage IP project flow
- Using a Tcl script to create IP customization

After creating a customization, you can either automatically generate output products, or defer until later.

- In a Project Mode flow, if the output products are not present, they are generated automatically prior to synthesis or simulation.
- In a Non-Project Mode flow, you must generate the output products prior to synthesis or simulation.

To use a customization in a design, instantiate the IP in the HDL code. One of the IP output products is an instantiation template based upon the target language set in the project settings.



---

**TIP:** When entering or modifying data in a text box, if a value is used and editable, the text is black and the background is white. If a value is used but not editable, the text is black and the background is gray. If a value is unused or not applicable, the text is gray, including the label that precedes or follows it.

---

---

## IP Catalog Features

The key features of the Vivado IP Catalog include:

- Consistent, easy access to Xilinx IP, including building blocks, wizards, connectivity, DSP, embedded, AXI infrastructure, and video IP from a single common repository regardless of the end application being developed.
- Support for multiple physical locations, including shared network drives, allowing users or organizations to leverage a consistent IP deployment environment for third-party or internally-developed IP.
- Access to the Xilinx-delivered IP, which is rigorously tested prior to inclusion in the IP Catalog.
- Access to IP customization and generation using the Vivado IDE or an automated script-based flow using Tcl.
- On-demand delivery of IP output products such as instantiation templates and simulation models (HDL, C, or MATLAB® software).
- IP example designs that provide capability to evaluate IP directly as an instantiated source in a Vivado Design Suite project.
- Access to version history details as recorded in the Change Log.
- A <major#.minor#.Rev#> numbering scheme unifies the IP version numbers. For information, see the Xilinx IP Versioning page available from the Xilinx website: [www.xilinx.com/ipcenter/vivado\\_ip\\_versioning.htm](http://www.xilinx.com/ipcenter/vivado_ip_versioning.htm).
- Catalog filter options that let you filter by Supported Output Products, Supported Interfaces, Licensing, Provider, and Status.
- Group IP by Taxonomy or Repository.

For information on IP that supports the Vivado Design Suite, see [www.xilinx.com/support/index.html/content/xilinx/en/supportNav/ip\\_documentation.html](http://www.xilinx.com/support/index.html/content/xilinx/en/supportNav/ip_documentation.html).

For information on specific IP, see the [www.xilinx.com/ipcenter](http://www.xilinx.com/ipcenter) or look at the IP Catalog.

See [www.xilinx.com/ipcenter/axi4.htm](http://www.xilinx.com/ipcenter/axi4.htm) for information on AXI IP.

## Using the IP Catalog

Figure 2-1 shows the Vivado IP Catalog, which lists the available IP.

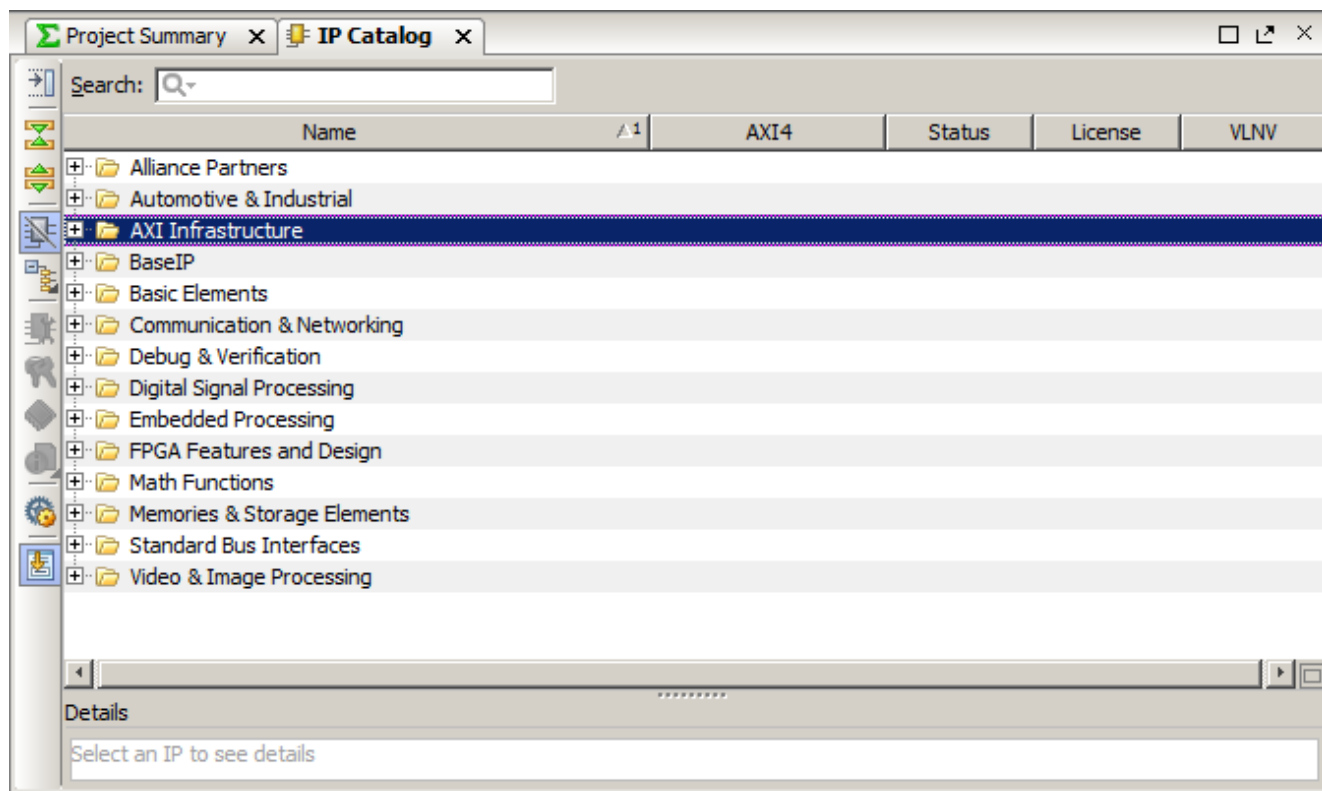



Figure 2-1: Xilinx IP Catalog

## Filtering IP

The IP catalog contains categories of IP that you can filter and search.

Figure 2-2 shows the IP Catalog Filter options that let you filter the IP catalog by categories. Select the **Filter** button  in the top left corner to open the filter options.

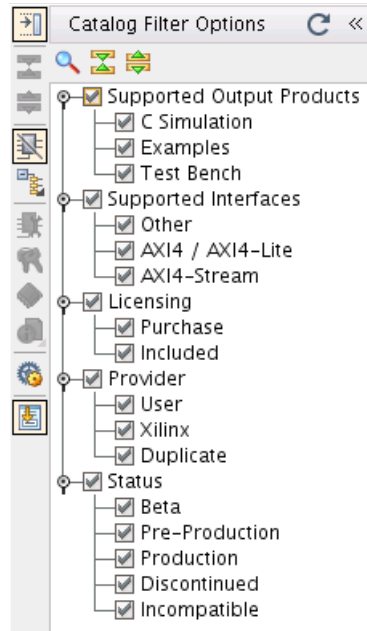



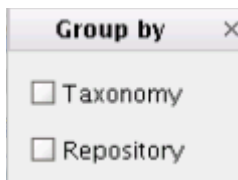








Figure 2-2: IP Catalog Filter

Uncheck the filter options to filter out IP that meets unwanted criteria.

IP Catalog options give you the ability to expand, collapse, and zoom the IP Catalog content.  These icons are consistent with other Vivado IDE.

Other filter options are:

Option	Description
	Hide Incompatible IP
	Group by Category. Right-click this icon and the <b>Group by</b> selection option opens, where you can choose <b>Taxonomy</b> or <b>Repository</b> : 

Option	Description
	Customize IP
	License Status
	Show Compatible Families
	View Product Guide, Change Log, Product Webpage, and Answer Records
	Settings for IP Catalog, IP generation, and Repository Manager and IP Packager
	Auto-scroll to Selected Items



**IMPORTANT:** It is possible to create duplicate IP. The Vivado tools issue a warning. See [Resolving Duplicate IP, page 13](#) for the possible resolutions.

## Resolving Duplicate IP

If you enter an IP that is a duplicate of an existing IP, the Repository Manager issues a warning banner.

- Review the duplicated IP in the IP Catalog.
- Review the directories in the description.
- If necessary, remove the IP or remove the repository that contains the IP.

## Partner Alliance IP

The Vivado IP Catalog includes IP available for purchase from select Alliance Partners. These IP are signified by the blue color disk on the icon, as shown in Figure 2-3.

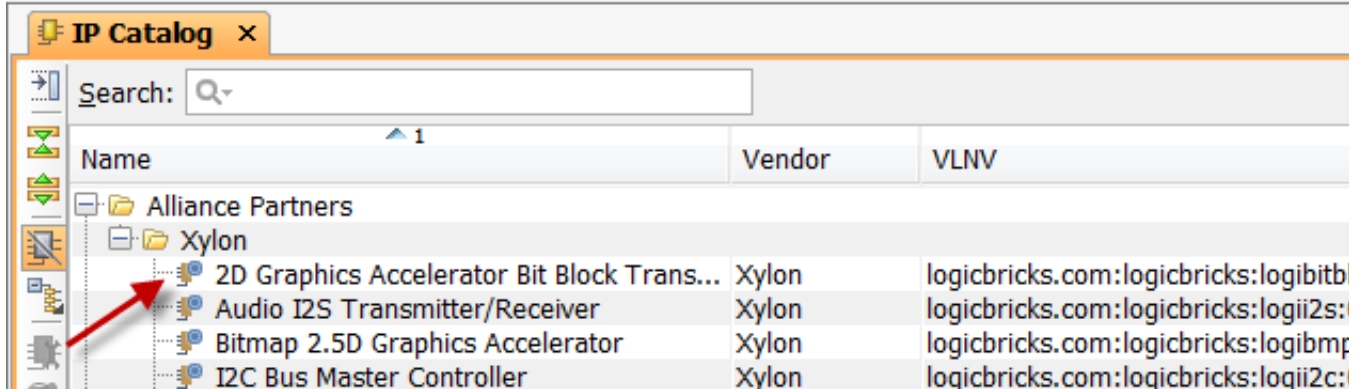


Figure 2-3: Alliance Partner IP

When you select an Alliance Partner IP, a dialog box opens that provides you with a link to where you can purchase the IP, as shown in Figure 2-4. The option to Customize IP is greyed-out until you purchase and install the IP.

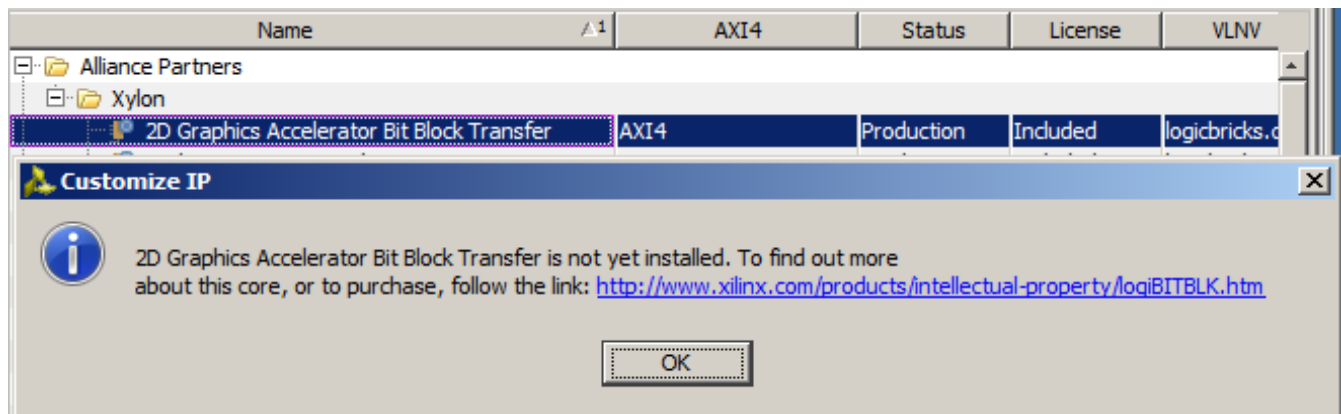


Figure 2-4: Partner Alliance IP Dialog Box

## Synthesis Options for IP

When generating the output products for an IP, the default behavior is to produce an out-of-context (OOC) synthesized design checkpoint (DCP). Alternatively, you can choose to synthesize the IP along with the top level user logic, which is called *global synthesis*.

Figure 2-5 shows the two flows possible for synthesizing IP:

- The default OOC flow
- Synthesizing the IP HDL along with the top-level user HDL (global synthesis)

In either flow, Vivado IDE generates HDL and XDC files for the IP, and uses those files during synthesis and during implementation.

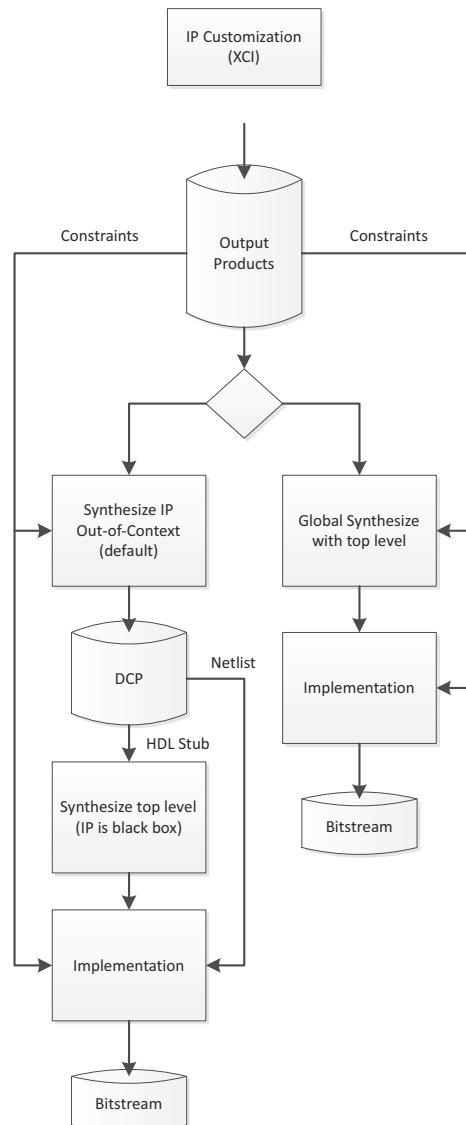


Figure 2-5: **Out-Of-Context (OOC) and Global Synthesis Flow**

## Out-of-Context Flow

In the OOC flow, the IP is synthesized alone and an OOC DCP is produced. A special OOC flow-only XDC output product file (the `_ooc.xdc`) is used when synthesizing the IP, which provides default input clock definitions. The produced DCP is a container file, and includes a netlist as well as constraints.

When synthesizing the entire design, an HDL stub module is provided in the DCP which causes a black box to be inferred for the IP.

Also, during synthesis of the entire design, the DCP provides an XDC file which defines any clocks an IP might output (the `_in_context.xdc`).

During implementation, the netlist from the IP DCPs are linked with the netlist produced when synthesizing the top-level design files, and the Vivado IDE resolves the IP black boxes. The IP XDC output products that were generated for use during implementation are applied along with any user constraints.

The OOC flow is the default flow because of two main benefits:

- It improves synthesis run times because you synthesize the IP only once.
- It produces either a `_functsim.v` or a `_functsim.vhdl` structural simulation model. You can use these files during simulation if you use a single language simulator and the IP does not deliver behavioral HDL in that language.

## Implementing IP OOC

By default, the Vivado IDE creates a synthesized design checkpoint (DCP) file automatically during the customization process for most Vivado Design Suite IP.

When performing synthesis of the top-level design, IP with an associated DCP file is a black box because it is being synthesized OOC.

Use this procedure to perform manual analysis of the IP standalone after implementation:

1. In the Design Runs window, right-click the IP design run, and select **Launch Runs** to launch the implementation run as shown in [Figure 2-6](#).

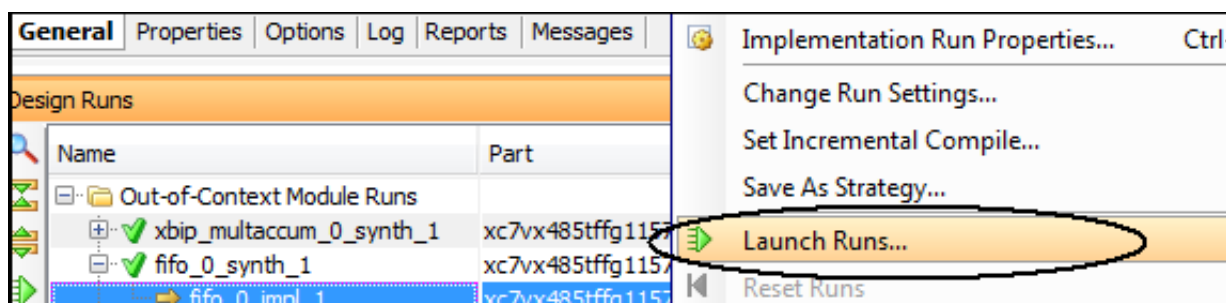


Figure 2-6: Implementing an IP Standalone



- After implementation completes, right-click the IP design run, and select **Open Implemented Design** to open the design for analysis, as shown in Figure 2-7.

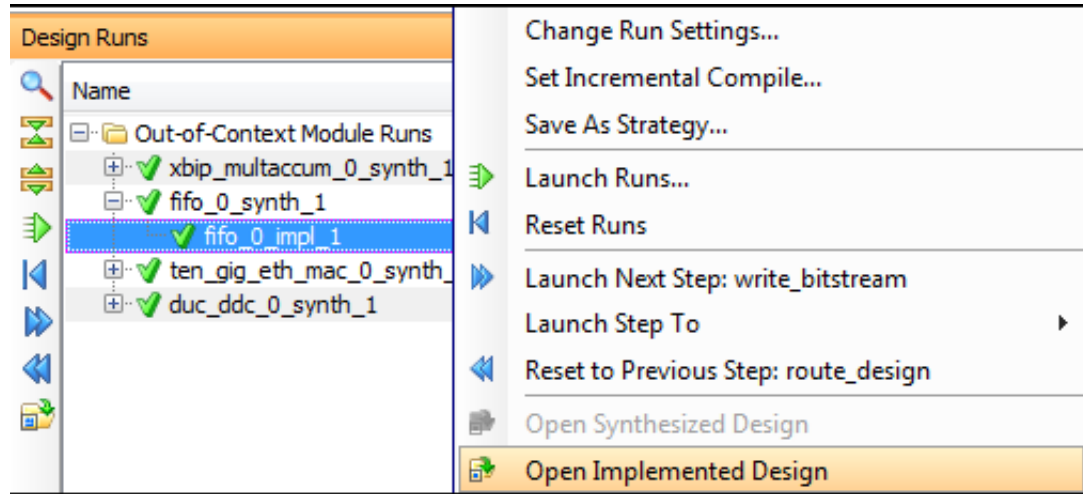


Figure 2-7: Opening an Implemented IP

When performing timing analysis, the results are not accurate because the clocks are not yet routed and ideal clocks are used. This is most obvious when performing hold analysis, because the router cannot fix hold violations.

Some IP include the `HD.CLK_SRC` property in the `<IP_Name>_ooc.xdc` file, which improves the accuracy of post-implementation timing analysis. This property provides a location for a clock buffer, and the SLEW timer models, more accurately.



**IMPORTANT:** The implemented IP is for analysis only and is not used during synthesis or implementation of the top-level design. For information on using an implemented version of the IP, see the Vivado Design Suite User Guide: Hierarchical Design (UG905) [Ref 13].

## Out-Of-Context Settings

When working with Vivado Design Suite IP, you can disable the generation of a DCP and instead synthesize the IP RTL with the top-level RTL.



**RECOMMENDED:** Use the OOC flow when generating IP. The OOC flow speeds up run time for the complete project, and lets you avoid re-synthesizing IP when doing project runs.

In the Generate Output Products dialog box, click **Out-of-Context Settings**.

In the Out-of-Context Settings dialog box (Figure 2-8, page 18), you can:

- Uncheck the check box. This prevents the IP files from being automatically generated out-of-context to the design for synthesis.
- Specify the number of DCP generation jobs to run at one time.

By default, one job is specified, and the design runs launch sequentially. A higher number specifies the maximum number of design runs that can be running in parallel.

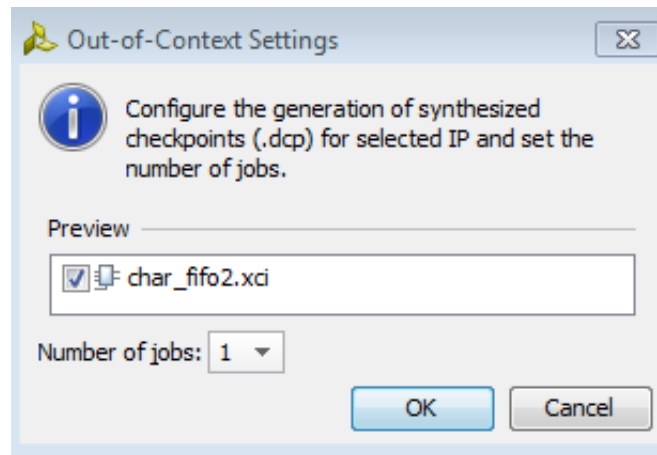


Figure 2-8: Out of Context Dialog Box

After changing these settings, you can either generate the output products or delay output product generation. DCP generation preferences are preserved whether or not you generate output products now.

For a scripted flow, you can disable the automatic generation of the DCP in either Project Mode or Non-Project Mode by setting the `GENERATE_SYNTH_CHECKPOINT` property to `FALSE` for the XCI file, as shown in the following Tcl command:

#### Tcl Command to Disable OOC Options on IP

```
set_property GENERATE_SYNTH_CHECKPOINT FALSE \
  [get_files <IP_Name>.xci]
```

## Global Synthesis Flow

In the global synthesis flow, the IP is synthesized along with user HDL. Any changes made to user HDL result in the IP being re-synthesized as well.

During implementation, the IP XDC output products that were generated for use during implementation are applied along with any user constraints.



**RECOMMENDED:** Always reference the IP using the XCI file. It is not recommended to read just the IP DCP file, either in a Project Mode or Non-Project Mode flow. While the DCP does contain constraints, it does not provide other output products that an IP could deliver and that could be needed, such as ELF, COE, and Tcl scripts.

## Generating Output Products

After IP customization is complete, the Generate Output Products dialog box opens, as shown in [Figure 2-9](#).

Output products delivered by the IP are listed in the Preview area, and you can do any of the following:

- To generate the listed output products, click **Generate**. By default, the Vivado IDE creates an XCI and a DCP for the IP when you generate the output products, as well as a change log, a behavioral simulation model, and an instantiation template.
- To delay generation of output products, click **Skip**. This lets you select multiple IP customizations and generate all output products at one time, including launching parallel synthesis runs for IP DCP files.

**Note:** When working in Project Mode, output products are automatically generated as needed prior to synthesis of the top-level design. This includes any specified DCP files.

- The Vivado IDE generates the synthesized IP DCPs out-of-context. See the [Synthesis Options for IP](#), page 14.

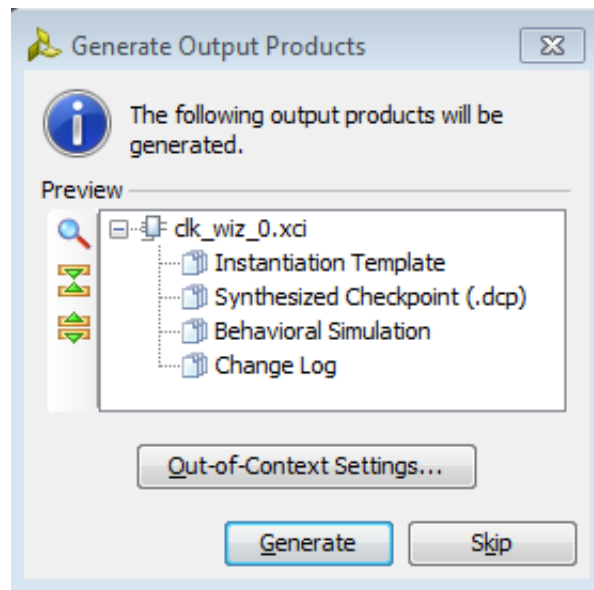
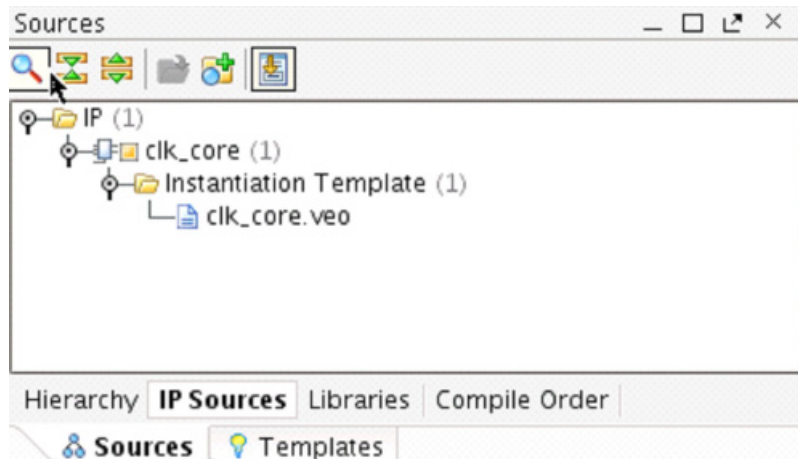


Figure 2-9: Generated Output Products

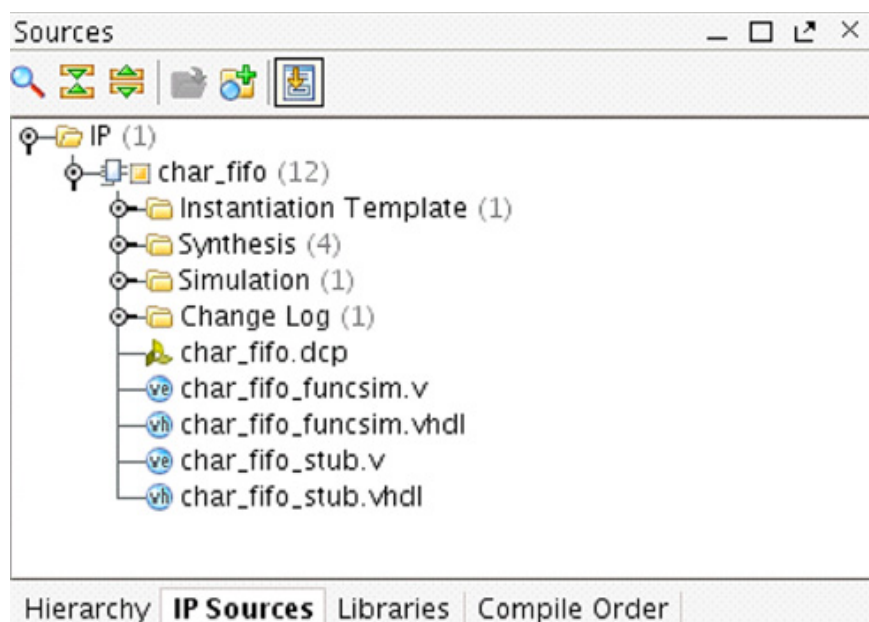
## Examining Generated Output Products

The IP Sources window shows the generated output products for all IP in the project. If you skipped generation of the output products an instantiation template is the only generated product (besides the XCI and BOM files, which are not displayed) as shown in [Figure 2-10](#), page 20.



**Figure 2-10: IP Customization with Generation of Output Products Skipped**

As shown in (Figure 2-11), when the output products are generated, the Sources window lists unencrypted files.



**Figure 2-11: IP Customization with Output Products Generated, Including OOC DCP**

These files include: an instantiation template, synthesis and simulation targets, XDC constraints, a change log, and other products.

By default, the Vivado IDE creates an OOC DCP along with structural simulation netlists (`_funcsim.v/_funcsim.vhdl`), and creates stub files (`*_stub.v/*_.vhdl`) for use with third-party synthesis tools to infer a black box for the IP.

After generating the synthesis output products, the Vivado IDE creates and launches a design run to produce the OOC DCP.

While the synthesis run is processing, the OOC related files are shown as missing (Figure 2-12).

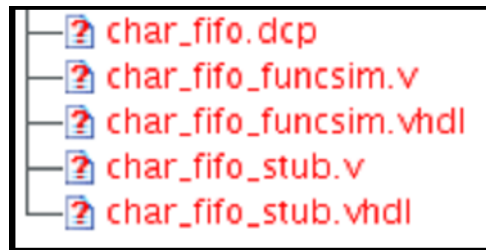


Figure 2-12: OOC Flow Output Products Pending Creation

If you elected to not generate the OOC DCP, the Vivado IDE does not create `_funcsim`, and stub files.

## Using IP Project Settings

See the following documents for more information about the Repository Manager and Packager features:

- *Vivado Design Suite User Guide: Creating and Packaging Custom IP* (UG1118) [Ref 1]
- *Vivado Design Suite Tutorial: Creating and Packaging Custom IP* (UG1119) [Ref 2]

When working with IP in a Manage IP project or in an RTL project, you can configure IP-specific project settings using the IP category in the Project Settings dialog box. The following tabs are available:

- **Repository Manager:** Adds IP repositories and specifies the IP to include in the IP catalog.
- **Packager:** Sets the default behavior used by the IP packager when packaging IP.

**Note:** The IP Project Settings and the Vivado IP Catalog are only available when working with an RTL project or when using Manage IP from the Getting Started page. When using Manage IP, a subset of the IP settings is available unless a project is created.

## Using the Repository Manager

To use the Repository Manager, shown in the following figure, do the following:

1. Select **Tools > Project Settings**.
2. In the left pane of the Project Settings dialog box, click **IP**.
3. Click the **Repository Manager** tab, and do the following:

- a. In the IP Repositories section, click **Add Repository** to specify the directories that contain packaged IP to add to the IP repositories list. The Repository Manager hierarchically searches within the user repository paths for IP definitions.

You can either use your packaged IP or acquire it from a third-party supplier.

- b. In the **IP > Selected Repository** section, click **Add IP** to specify the IP to include in the IP Catalog.

**Note:** The IP catalog shows the included IP, and you can create a customization of the IP for use in a design.

- c. To update the contents of the IP Catalog with the IP within each repository, click **Apply**.

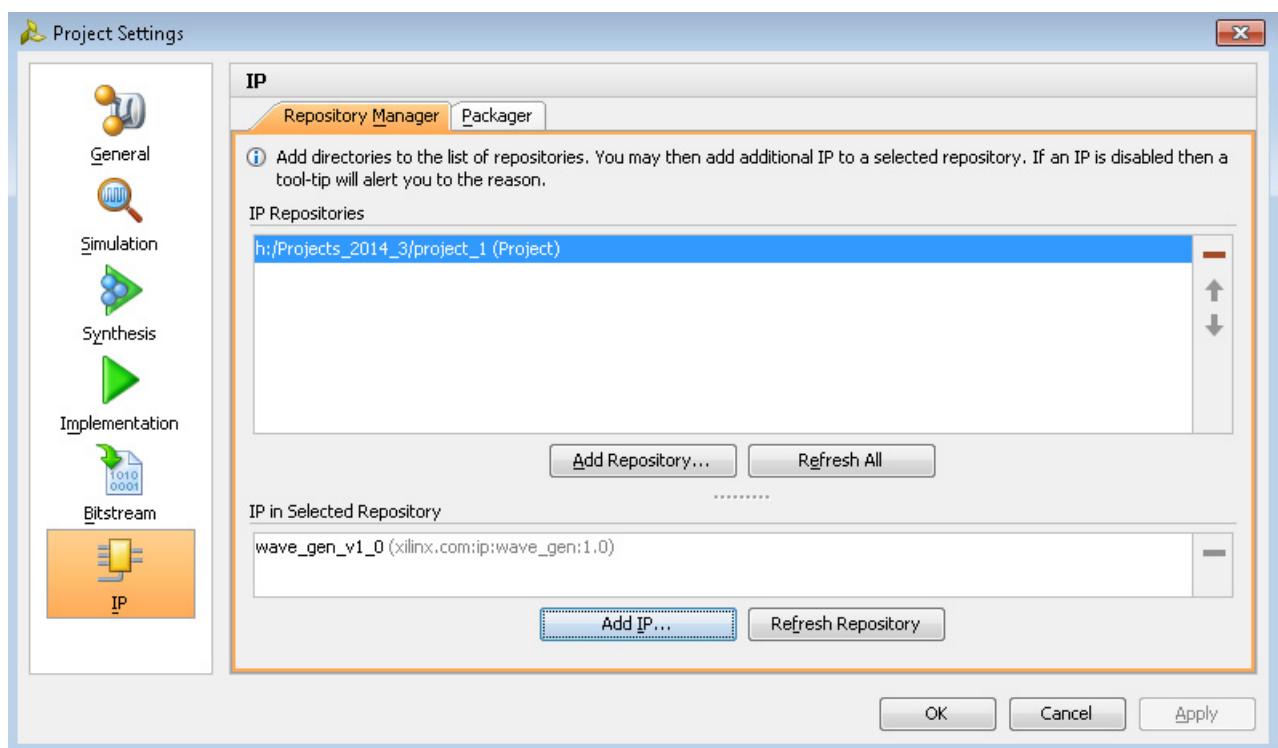


Figure 2-13: IP Project Settings: Repository Manager

## Using the Packager Settings

To set the Packager options:

1. Click the **Packager** tab in the following figure:

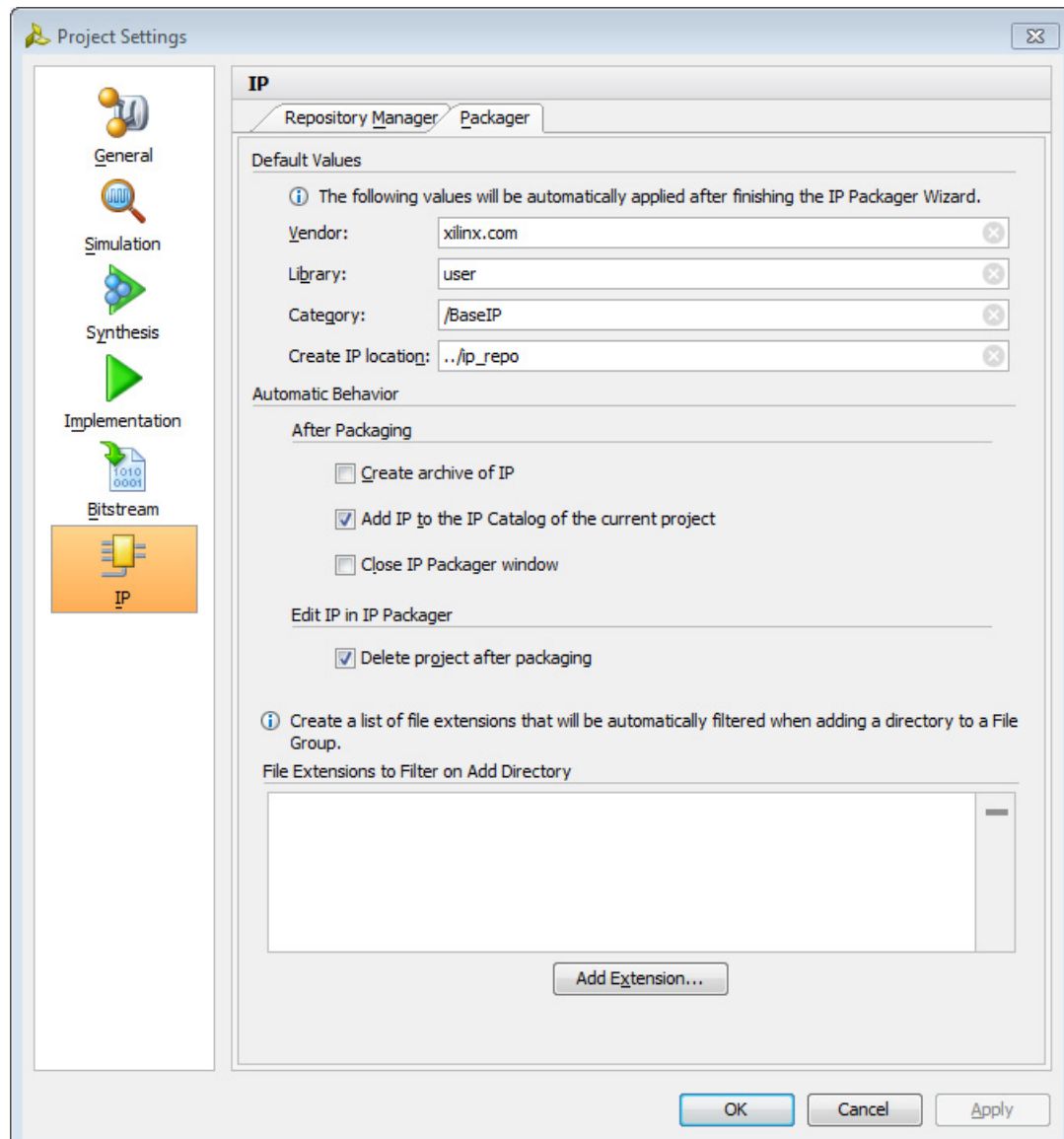


Figure 2-14: IP Project Settings—Packager Tab

2. Fill out the following information:

- In **Default Values**, set the following options:
  - **Vendor:** Sets the vendor name to use when packaging a new IP. This is, for example, the top domain name of a company.
  - **Library:** Sets the associated library for the IP. This category, along with the Vendor are used in conjunction with the IP name to create a unique identifier.
  - **Taxonomy:** Specifies the categories in the IP Catalog in which to place the IP. For example, /BaseIP.

**Note:** If necessary, you can change the default values for packaging IP during the IP packaging process.

3. In **Automatic Behavior**, check or uncheck the options you want:
  - **After Packaging:**
    - **Close IP Packager window:** Closes the Package IP window automatically when IP packaging is complete.
    - **Add IP root directory to current project's IP repository paths:** Adds the current IP to the IP repository.
    - **Create archive of IP:** After packaging an IP, automatically create an archive (ZIP format) of the IP.
  - **Edit IP in IP Packager:**
    - **Delete project after packaging:** Removes the iterative editing project after the IP is re-packaged.
    - In **Filtered Extensions**, add extensions (for example .txt) to automatically filter when selecting a directory to include in a **File Group** when packaging an IP.

---

## Creating an IP Customization

You can create an IP customization using the IP Catalog. The steps are:

1. Locate an **IP Definition** that matches your design requirements.
2. Customize that IP for your design or for the general use of a design team, thus creating an *IP Customization*.
3. Save the IP customization inside a project directory (default), or into a centralized directory.
4. Select an IP from the IP Catalog and customize the IP for use in your design.
  - a. From the **IP Catalog**, select the IP.
  - b. Double-click the selected IP, or select the **Customize IP** command from the toolbar.

The Customize IP dialog box shows the various parameters available for you to customize the IP. This interface varies, depending on the IP you select, and can include one or more tabs in which to enter values.

The Customize IP dialog box includes the following:

- An IP symbol
- Various tabs for setting configuration options for the specific IP



The IP symbol supports zoom, re-size, and auto-fit options that are consistent with the schematic viewer canvas in Vivado IDE. [Figure 2-15, page 25](#) shows the Customize IP interface for the FIFO Generator IP.

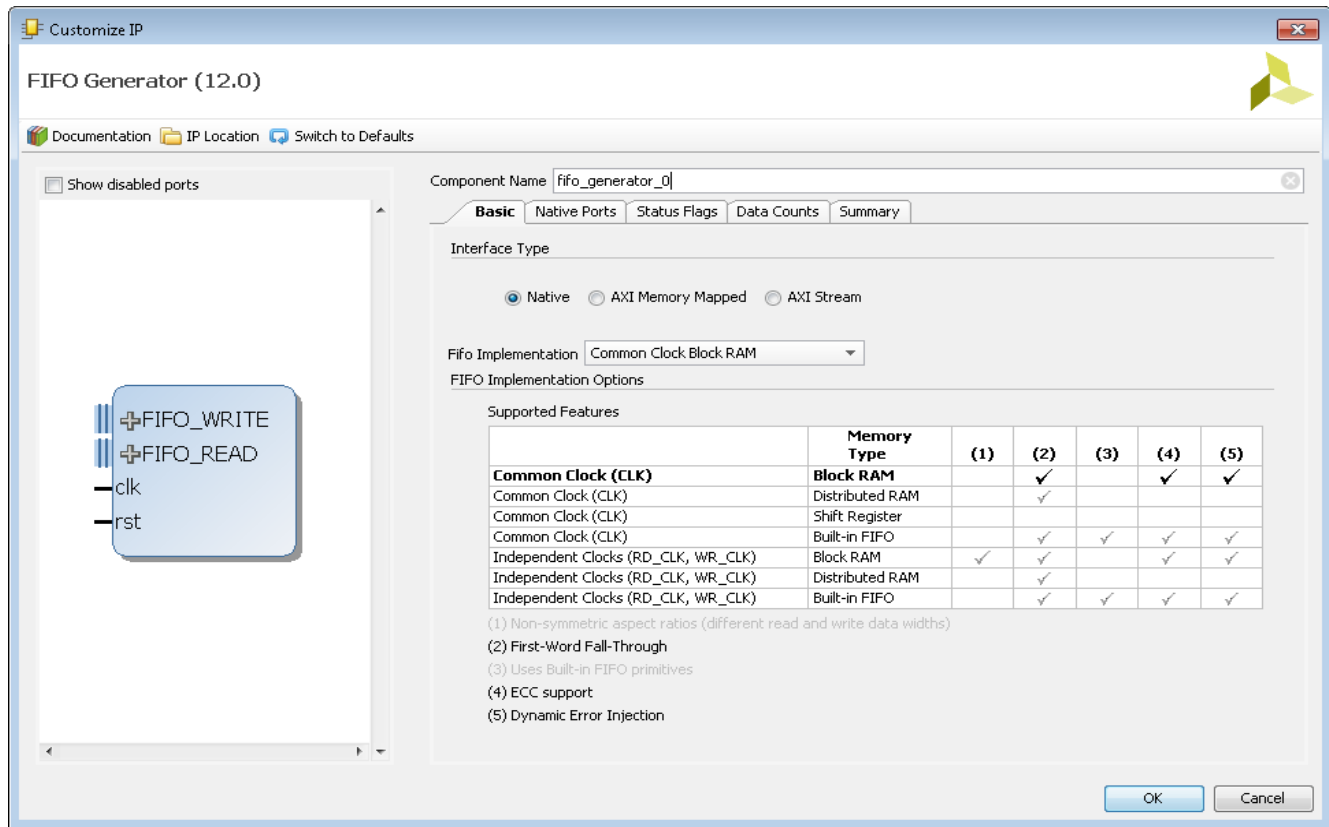


Figure 2-15: FIFO Generator Dialog Box

5. In the IP dialog box:
  - Click **Documentation > Product Guide** to open the product guide for the selected IP.
  - Click **IP Location** to specify the location on disk to store the IP. You can only change this location when using the IP Catalog in an RTL project. This is *not an option* in a Manage IP project.
  - Click **Switch to Defaults** to display a window asking if you want to reset all configuration options back to their default starting point.
6. Customize the IP as needed for your design, and click **OK**.



**RECOMMENDED:** When specifying a numerical value, use hexadecimal to speed processing.



**VIDEO:** See the [Vivado Design Suite QuickTake Video: Customizing and Instantiating IP](#) for a demonstration of the IP Customization and Instantiation process.

### Tcl Command Example for Creating an IP Customization

You can also create IP customizations using the `create_ip` Tcl command. For example:

```
create_ip -name fifo_generator -version 12.0 -vendor xilinx.com -library ip
-module_name fifo_gen
```

You must specify either `-vlnv` or all of `-vendor`, `-library`, `-name`, and `-version`.

**Note:** Executing the `create_ip` Tcl command creates the IP customization file (XCI), which is the configuration for the IP, as well as the instantiation template and BOM (XML) file, but does not create any other output products.

The default configuration for the IP is set with the `create_ip` command.

### Tcl Command Example for Setting IP Properties

To configure with different settings, use the `set_property` command. For example:

```
set_property CONFIG.Input_Data_Width 12 [get_ips fifo_gen]
```

### Tcl Command Example for Reporting IP Properties

To get a list of properties available for an IP use the `report_property` command. For example:

```
report_property [get_ips fifo_gen]
```

Configuration properties start with `CONFIG`.

---

## Creating a Memory Interface Generator Customization

The Memory Interface Generator (MIG) creates memory controllers for Xilinx devices and IP. MIG creates complete customized Verilog or VHDL RTL source code, pinout and design constraints for the selected FPGA, and script files for implementation and simulation.

The pages of the wizard take you through the MIG options. See the *Zynq-7000 SoC and 7 Series FPGAs Memory Interface Solutions* (UG586) [Ref 30] for more information.

For UltraScale™ MIG, Vivado launches a pin planning project, and lets you set the appropriate pins for that device. See the *Vivado Design Suite User Guide: I/O and Clock Planning* (UG899) [Ref 22].



**VIDEO:** See the Quick Take Video: [Creating a Memory Interface Design Using Vivado MIG](#) for instructions on how to use the MIG wizard to create a MIG design.

---

Also, visit the Xilinx [MIG Solution Center](#) for more information about MIG. The Designing with IP Design Hub in Document Navigator provides videos and links to MIG documentation.

## Re-Customizing Existing IP

You can re-customize existing IP in either an RTL project or in a Manage IP project.

1. To open the IP customization dialog box for an IP, you can either:
  - Double-click the IP.
  - In the **IP Sources** view, right-click the IP, and select **Re-customize IP** from the menu as shown in [Figure 2-16](#).

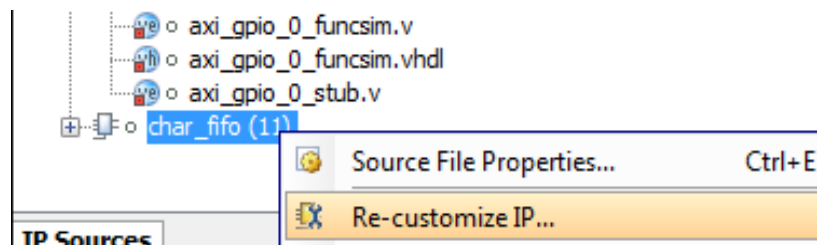


Figure 2-16: Re-customize IP Option

2. Change IP parameterization options, and click **OK**.

The Generate Output Products dialog box opens.

Alternatively, you can view the current settings, and click **Cancel** to keep the settings intact.

When you do make changes to the IP configuration and generate the output products, the existing products are reset and, if enabled, the design run resets and launches again.



**RECOMMENDED:** Always generate the output products for IP, including the synthesized DCP.

## Manually Generating Output Products

At any point you can manually generate output products as follows:

1. In the IP Sources or the Hierarchy view, select the IP.
2. Right-click and select **Generate Output Products**.

The default flow when generating output products is to create and launch an Out-Of-Context (OOC) synthesis design run for the IP. This results in the inference of a black box for the IP when synthesizing the rest of the design.



**IMPORTANT:** *The implementation stage resolves black boxes by extracting the netlists from the DCP of the IP.*

**Note:** If you do not want to automatically create an OOC run during synthesis, uncheck the **Generate Synthesized Checkpoint** check box and press **Generate**, as described in [Re-Customizing Existing IP](#), page 27.



**RECOMMENDED:** *OOC is the recommended flow. Using the OOC reduces the run time on synthesizing your design: you do not need to synthesize the IP every time you run synthesis during development.*

## Instantiating an IP

After you create an IP Customization, you can then instantiate that IP in a design. An instantiation template is created after IP customization, regardless of whether you generated output products. The instantiation template is available in the IP Sources tab of the Sources window in the Instantiation Template folder as shown in [Figure 2-17](#).

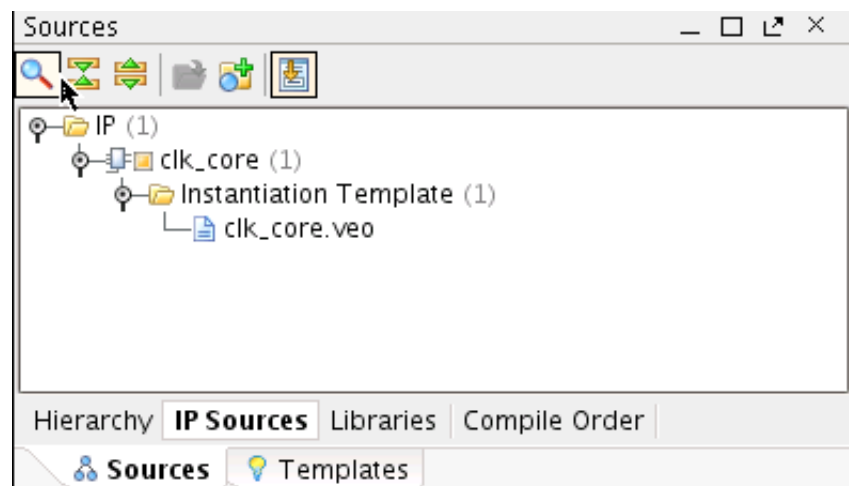


Figure 2-17: Location of Instantiation Template

Depending on the set project target language, either a VHO or VEO file is present, which contains the instantiation template that you can copy and paste into your RTL design.

[Figure 2-18](#), page 29 shows the instantiation template for a Clocking Wizard IP customization.

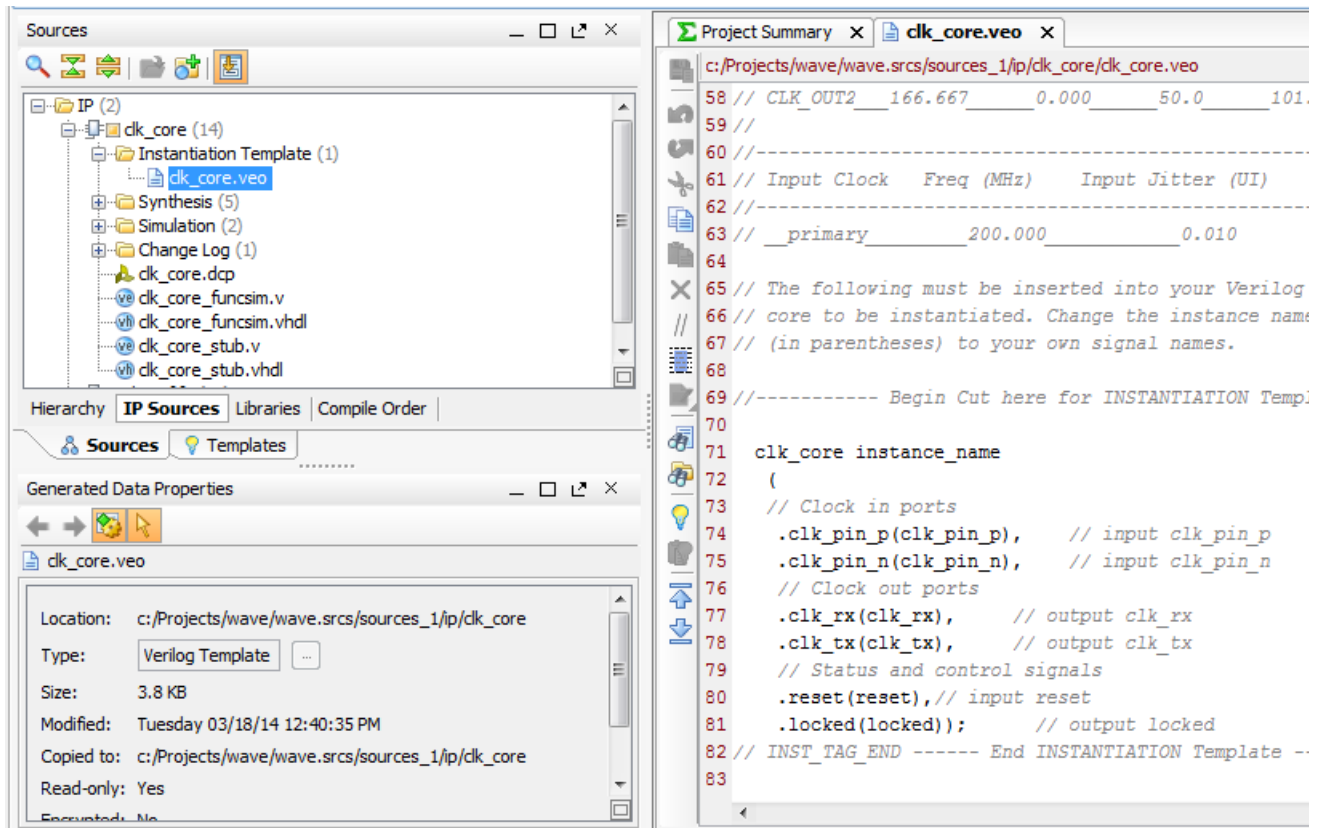


Figure 2-18: Instantiation Template

To use the instantiation template in your design:

1. Open either the VEO or VHO template file for the IP customization by double-clicking the file in the Sources view, or by selecting the file using the **Open Files** option.
2. Highlight the Instantiation Template between the comments as indicated and copy the section.
3. Open HDL file in which you want to instantiate the IP.
4. Paste the copied template to the location of your choice.
5. Edit the HDL to integrate the template into your design as needed; for example, change the port connections.

After the IP is instantiated in a design, the IP files show in the Hierarchy tab Sources window at the location in the design where it is instantiated, as shown in [Figure 2-19, page 30](#), which shows two IP instantiated in a design.

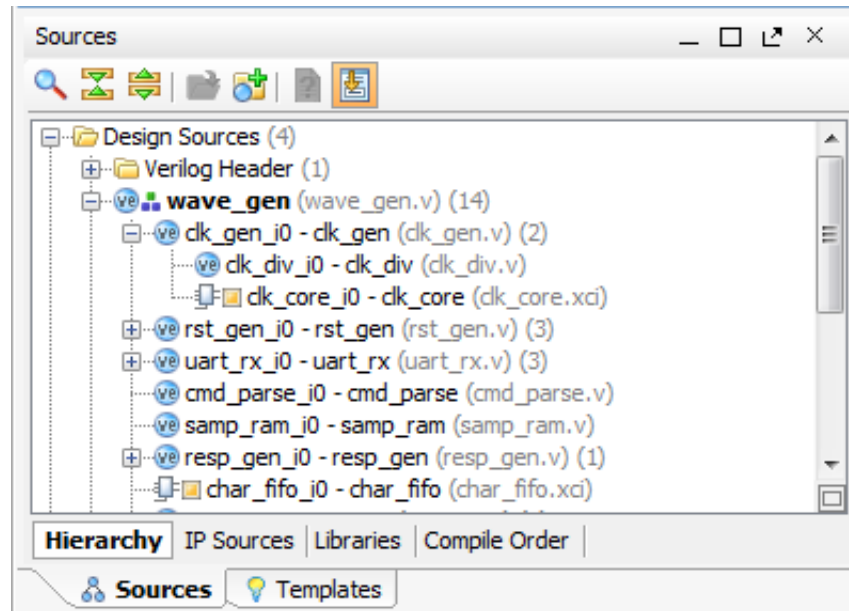



Figure 2-19: IP Instantiated in a Design

With the IP customization properly instantiated into your design, you are ready to synthesize the IP along with the rest of your design, either as a black box if the OOC flow is used or together, if you are using global synthesis. See [Synthesis Options for IP, page 14](#) for more details.

When you expand the IP hierarchy by right-clicking the IP, you could see the Encrypted IP source icon . This source cannot be viewed.

## Adding Existing IP to a Project

You can add previously created CORE Generator™ IP (<IP\_Name>.xco files) or Vivado IP (<IP\_Name>.xci files) by using the **Add Existing IP** option in the Add Sources dialog box, as shown in [Figure 2-20, page 31](#).



Figure 2-20: Add Sources Dialog Box

You can either reference the IP and any generated output products from the location specified or copy the IP and any generated output products into your project.

See [Chapter 3, Using Manage IP Projects](#) for details on creating IP using the Managed IP flow.

After you click **Next**, the Add Existing IP wizard opens, as shown in the following figure.

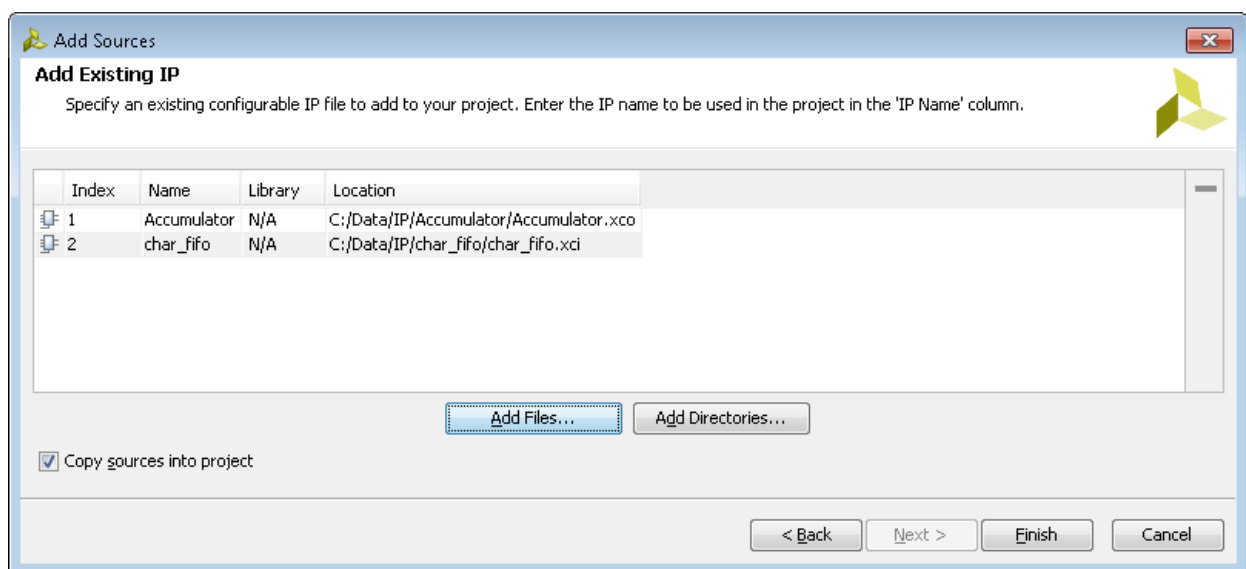


Figure 2-21: Add Existing IP

The added IP and any output products are shown in the IP Sources tab of the Sources view, as well as with other source files in the Hierarchy, Libraries, and Compile Order views.

You can select the IP in the IP Sources and view the properties for it in the Source File Property window.




---

**IMPORTANT:** *NGC format files are not supported in the Vivado Design Suite for UltraScale devices. It is recommended that you regenerate the IP using the Vivado Design Suite IP customization tools with native output products. Alternatively, you can use the `NGC2EDIF` command to migrate the NGC file to EDIF format for importing; however, Xilinx recommends using native Vivado IP rather than XST-generated NGC format files going forward.*

---




---

**IMPORTANT:** *The Vivado Design Suite does not support UCF constraints.*

---

## Tcl Command Examples

The following are commonly used IP Tcl command examples.

### Tcl Command Example to Add Existing IP

You can add existing IP using the following Tcl Command:

```
import_ip
```

### Tcl Command Examples to Import IP

The following are examples of `import_ip`:

```
import_ip -files C:/vivado_ip/aurora_8b10b_v7_1.xci -name aurora_1234_v7_1
import_ip -files C:/vivado_ip/blk_mem_gen_v6_1.xci
```

### Tcl Command Example to Read IP

To remotely access an IP, use:

```
read_ip <IP_Name>
```

This command does not copy IP into the project.

### Tcl Command Example to Add Files

```
add_files <File_Name>
```

## Upgrading IP

Future versions of, or patches for, the Vivado Design Suite might have newer versions of IP that are used in a project. Only one version of an IP is delivered in each release of the Vivado Design Suite.



Prior to moving to a new Vivado Design Suite release, do one or more of the following:

- Generate all the output products for the IP, including the DCPs. This lets you use the old version of the IP in the new release of the Vivado Design Suite, if needed.



**IMPORTANT:** *It is especially important for IP that have a major revision change between Vivado Design Suite releases because these IP typically require RTL changes.*

**Note:** You cannot generate output products, including DCPs, for IP that is not the current version.

- If you are using Manage IP projects, copy the entire Manage IP project location as a backup.
- Archive design projects that contain IP.



**RECOMMENDED:** *When migrating the project to the newer version of the Vivado Design Suite, some IP might become locked because they are not current. It is recommended that the IP be upgraded.*

Before upgrading IP, view the change log for information on the changes.

When an upgrade is available for an IP, you can upgrade the IP using either of the following methods:

- In the IP Sources window or the Hierarchy window, right-click the IP, and select **Upgrade IP** as shown in the following figure.
- In the IP Status window (Figure 2-24, page 35) that opens when you run the **Report IP Status** command, click the **Upgrade** option next to the IP.

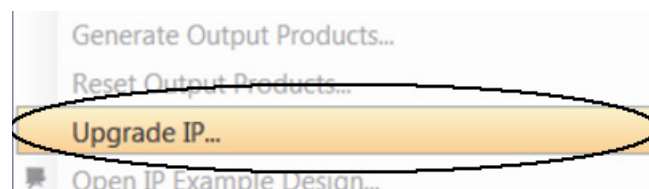


Figure 2-22: Upgrade IP Option

**Note:** When upgrading IP that is in a board design, see the *Vivado Design Suite User Guide: Designing IP Subsystems Using IP Integrator* (UG994) [Ref 3] for upgrade instructions.



**VIDEO:** See the [Vivado Design Suite QuickTake Video: Managing Vivado IP Version Upgrade](#) for a demonstration of upgrading IP.

Upgraded IP creates an update log automatically. These logs are available from the /IP Update Log folder in the Design Sources window as shown in the following figure.

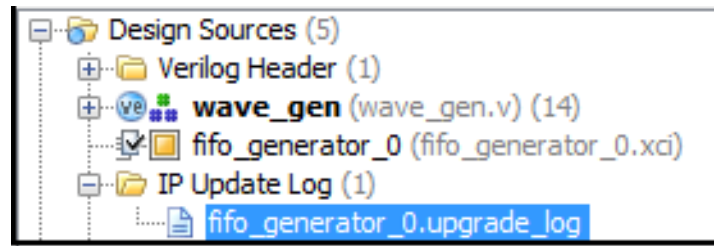


Figure 2-23: IP Update Log Folder

This log contains information about the IP upgrade, such as:

- If the upgrade is successful
- If user intervention is required

**Note:** If the upgraded IP requires user intervention, the log lists the issues encountered, such as parameters that no longer exist.

- The original version and revision of the IP
- The upgraded version and revision of the IP
- Additional information as appropriate:
  - A description of changes made by the upgrade script.

Examples of such information are: "Renamed parameter DATA\_W to C\_DATA\_WIDTH", and "Set parameter BUFFER\_LENGTH to value 32").

- Warnings issued during customization of the IP, such as ports added, changed, and removed during an upgrade.
- Warnings associated with the upgrade; either general issues relating to the upgraded version, or specific issues related to the current parameterization.
- Any warnings issued during customization of the IP.



**CAUTION!** When upgrading an IP, all previously generated output products are removed, including the DCPs and any associated design runs.

## Viewing the IP Status Report

You can view the status report of IP in a project using the **Tools > Report > Report IP Status** pull-down menu. A new tab titled **IP Status** displays the report results. Multiple runs cause additional reports to display in the IP Status tab (Figure 2-24, page 35).

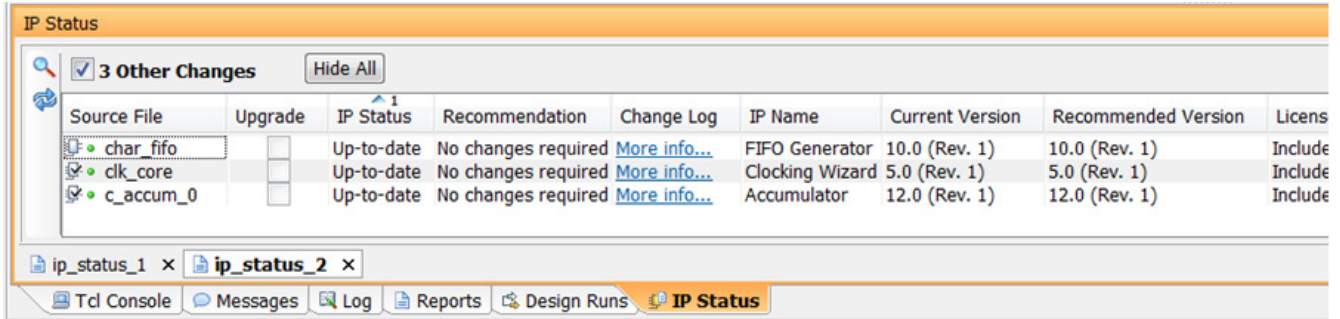


Figure 2-24: Report IP Status View in GUI

When you bring in older projects that contain IP, the Vivado IDE prompts you to open the **IP Status Report** to review the status of the IP in the design.

The change log provides information about changes to the IP for each release.

To view the change log, either:

- Right-click the IP sources in the Report IP Status view, and select **IP Documentation > View Change Log**,
- On the Report IP Status view, click the **More Info** link in the **Change Log** column.

The pop-up menu has only the latest changes. Use the **More info** link to see all the information, as shown in the following figure.

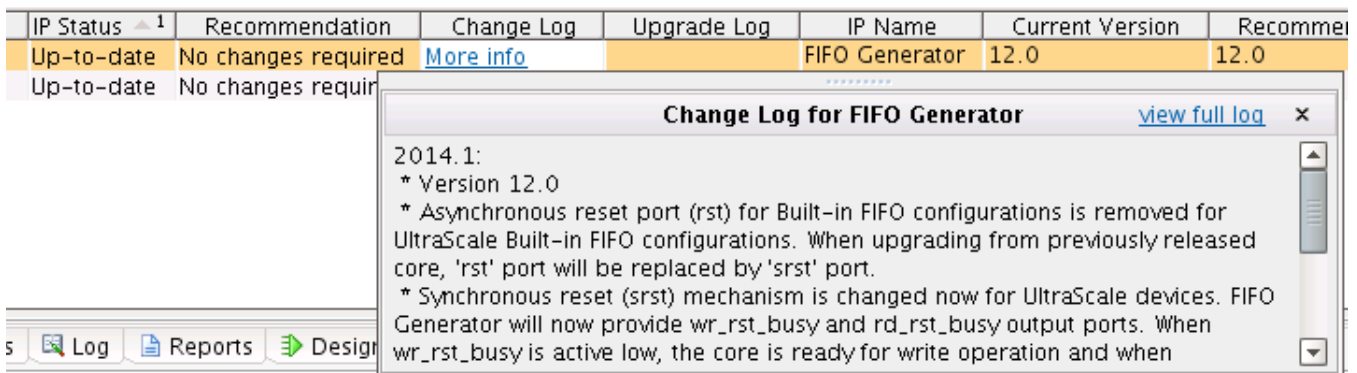


Figure 2-25: Change Log from More info Link

### Reporting IP Status using a Tcl Command

You can also generate this information using the following Tcl command:

```
report_ip_status
```

Optionally, use the following command to create a text file:

```
-file <file_name>
```

## Copying an IP

You can copy an existing IP customization to use as a starting point for a new IP. This is useful when you already have customized an IP and need only make small or simple customization changes and do not want to start from the default customization.

To copy an IP:

1. In the IP Sources tab, select the IP, right-click and select **Copy IP**, as shown in the following figure.

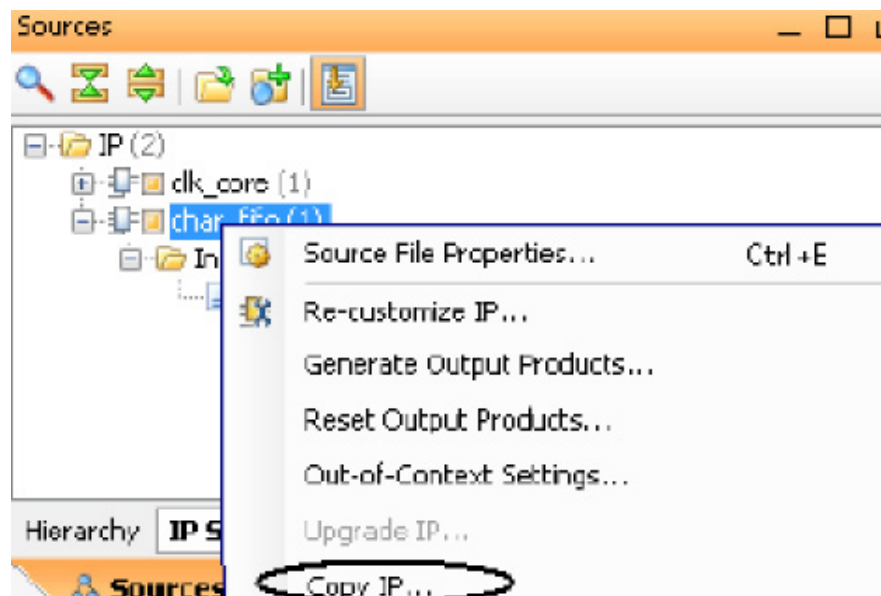


Figure 2-26: Select IP to Copy

2. Provide a destination name and location for the copy, as shown in the following figure.

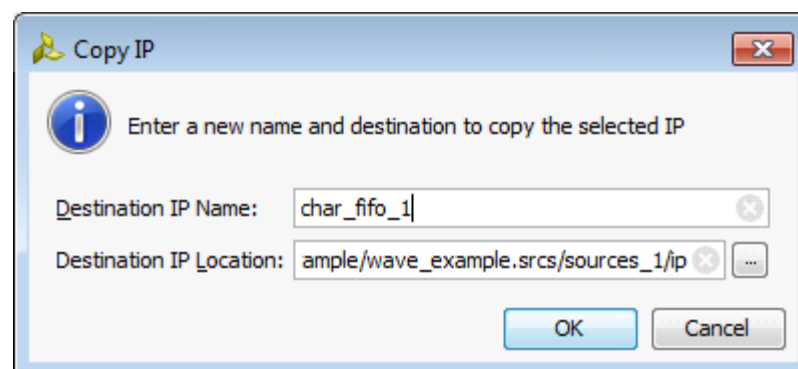


Figure 2-27: Copy IP Dialog Box

You can save the copied IP into a project directory structure, which by default is located at `<project_name>.src/sources_1/ip/`.

When working with a **Manage IP** project the default location is the same location as the `/manage_ip_project` directory.

You can now make changes to the copied IP customization by either double-clicking the IP, or right-clicking and selecting **Re-customize IP** from the IP Sources tab. The copied IP customization window opens with the customization settings from the original IP. You can now make edits.



**CAUTION!** *It is possible to have two versions of the same IP that reference different subcore versions; an older version that is locked and another, newer version. In such a case, a synthesis tool could produce errors or logic bugs because files exist with the same module names but with different contents.*

### Tcl Command Example for Copying IP

You can use the `copy_ip` command to create a copy of an IP customization: For example:

```
copy_ip -name newFIFO [get_ips char_fifo]
```

This command creates a copy of the `char_fifo` IP, names it `newFIFO`, and adds it to the project. Because no directory was specified using the `-dir` option, the IP is created inside the project directory structure.

## Understanding IP States Within a Project

There are several states in which an IP can display within in a project, depending if it is the current version in the catalog and if you have generated output products.



**IMPORTANT:** *For imported IP cores with versions that are not accessible from the Vivado IP catalog, re-customizing, re-setting, and re-generating the IP is not enabled.*

When you add existing IP (either in the XCI or XCO form), if present, the output products (such as HDL files) are also added. [Table 2-1](#) shows the buttons that represent the states of the Vivado IDE IP:

**Table 2-1: IP States in a Project**





Button	Description
	IP in an RTL project to be synthesized out-of-context (OOC).
	Customized IP which is in the IP Catalog that is to be synthesized with the project (global synthesis).

Table 2-1: IP States in a Project (Cont'd)

Button	Description
	Unmanaged IP. The user has changed the <code>IS_MANAGED</code> property of the IP to be <code>false</code> and has taken responsibility for the management of the IP. The IP becomes locked also. The purpose would be for the user to make modifications to unencrypted HDL sources or constraints.
	Locked IP that have output products can be used in the flow, but cannot be recustomized or regenerated. Locked IP with no output products are not usable in the flow. To use this IP, either provide the original output products or upgrade to the latest version. See <a href="#">Determining Why IP is Locked, page 38</a> for more information.



**IMPORTANT:** When working with Xilinx-delivered patches, you might notice IP locking due to changes in the IP definitions from the patch.

## Determining Why IP is Locked

IP becomes locked for several reasons. The Vivado IDE provides an IP status report that provides the reason and a recommendation. [Table 2-2, page 39](#) lists the locked IP messages and recommendations. See [Viewing the IP Status Report, page 34](#).

Table 2-2: IP Locked Reasons and Recommendations

Brief Reason	Verbose Reason	Brief Recommendation	Verbose Recommendation
IP file read-only	IP <IP_Name> has a read-only file <IPXMLFILE>. IP is write-protected. IP <IP_Name> has a read-only file <IPXCIFILE> with restricted functionality. Commands to change the configuration of this IP are disallowed.	Check file and project permissions	Review your project and file system permissions causing the IP to be read-only.
Shared output directory	IP <IP_Name> shares a common output directory with other IP. Xilinx recommends that you place each IP in its own directory.	Move IP	<ul style="list-style-type: none"> <li>Manually remove the IP from the project with the <code>remove_files</code> command (see the <i>Vivado Design Suite Tcl Command Reference Guide</i> (UG835) [Ref 4].</li> <li>Import it back into the project with a unique directory <code>import_IP</code> command (see <a href="#">Tcl Command Examples to Import IP, page 32</a>).</li> </ul>
IP definition not found	IP definition <CURRENT_IPDEF> for IP <IP_Name> (customized with software release <SWVERSION>) was not found in the IP Catalog.	Add IP definition to catalog.	Consult the IP Catalog for the replacement IP.
IP major version change	IP definition <CURRENT_IPDEF> for IP <IP_Name> (customized with software release <SWVERSION>) has a newer major version in the IP Catalog.	Upgrade IP.	Target IP definition <TARGET_IPDEF> requires a major version change. Review the impact on the design before upgrading the IP
IP minor version change	IP definition <CURRENT_IPDEF> for IP <IP_Name> (customized with software release <SWVERSION>) has a newer minor version in the IP Catalog.		Target IP definition <TARGET_IPDEF> requires a minor version change. Review the change log before upgrading the IP.
IP revision change	IP definition <CURRENT_IPDEF> for IP <IP_Name> (customized with software release <SW_VERSION>) has a different revision in the IP Catalog.		Target IP definition <TARGET_IPDEF> requires a revision change. Review the change log before upgrading the IP.
Incompatible IP data detected	The IP Data in the repository is not compatible with the current instance (despite having identical Version and Revision). This typically occurs if you are using IP that is currently under development. You cannot able to view the customization or generate outputs until it is updated.	Upgrade the IP.	Upgrade the IP.

Table 2-2: IP Locked Reasons and Recommendations (Cont'd)

Brief Reason	Verbose Reason	Brief Recommendation	Verbose Recommendation
IP unsupported part	IP <IP_Name> does not support the current project part <CURRENT_PART>. However part differences can result in undefined behavior.	Unsupported Upgrade IP.	Target IP definition <TARGET_IPDEF> does not support the current project part <CURRENT_PART>. Select a supported project part before upgrading the IP
IP license not found	IP <IP_Name> requires one or more mandatory licenses but no valid licenses were found. However license checkpoints could prevent the use of this IP in some tool flows.	Unlicensed Upgrade IP	Target IP definition <TARGET_IPDEF> requires a valid license. Obtain a valid license before upgrading the IP.
		Check IP license	IP <IP_Name> requires a valid license. Obtain a valid license or review your licensing environment.
IP board change <sup>(1)</sup>	This IP has board specific outputs. Current project board <CURRENT_BOARD> and the board <ORIGINAL_BOARD> used to customize the IP <IP_Name> do not match.	Retarget IP	Change the project part or re-target this IP using the upgrade flow to the current project part or board.
IP part change	Current project part <CURRENT_PART> and the part <ORIGINAL_PART> used to customize the IP <IP_Name> do not match		Change the project part or re-target this IP using the upgrade flow to the current project part or board.
IP contains locked subcore	IP <IP_Name> contains one or more locked subcore.	Upgrade parent IP	Upgrade the parent IP <PARENTNAME>

1. When an is locked due to a part or board change and is upgraded, you need to review the ports. Some IP have port differences based upon the part selected. For example, debug ports names and functions change when you update from 7 series to UltraScale for the QSGMII IP. You must make RTL changes to avoid encountering errors during synthesis and or implementation. See the appropriate IP product guide for more details.



## Understanding IP Constraints

The Vivado IDE manages both user-defined XDC timing and physical constraints for the entire design, as well as for Xilinx IP. It handles the association and the unification of constraints for Xilinx IP instantiated multiple times within a project.

Most IP in the IP catalog deliver IP-specific XDC constraints based on user customization. The constraints delivered by the IP are optimized using the default synthesis settings.



**RECOMMENDED:** *Do not change these settings for any of the IP design runs because you could encounter issues with applying constraints. To take ownership of constraining an IP, disable the XDC file(s) that are delivered with an IP. If you must change the synthesis settings for an IP, select the IP run from the **Out-of-Context Module Runs** in the Design Runs tab, then make changes in the Options tab of the **Synthesis Run Properties**. You can also script this action using the Tcl command:*

```
set_property <synthesis_option> <value [get_runs <IP_Name>_synth_1].
```

### Tcl Command Example for Changing Synthesis Run Properties

```
set_property STEPS.SYNTH.DESIGN.ARGS.FSM_EXTRACTION sequential /  
[get_runs my_IP_synth_1]
```

During design synthesis and implementation, the Vivado IDE processes the IP-delivered XDC constraints before processing the user-defined constraints, or after, depending on the constraint file.

### Tcl Command Example to Report Constraint Compile Order

```
report_compile_order -constraints
```

This command provides the procession order of synthesis HDL files as well as constraints used for synthesis and implementation of user logic, and provides a breakdown of the constraints used for each IP synthesis run used for the generation of the IP DCP.

## Constraint File Processing Order

Vivado IP can generate multiple XDC constraints files. By default, IP constraints are processed before user constraints because of the following possibilities:

- The IP might produce a clock that must be available to the end-user constraints.
- If the IP delivers physical constraints, the end-user can override them if necessary.

Some constraints that an IP delivers could have a dependency on a clock object that comes from either the end-user or another IP. These constraints are provided in a separate XDC file and are processed after the end-user constraints.

Typically, an IP delivers a core XDC file that can contain clock creation commands as well as commands without external clock dependencies. The customization file name is `<IP_Name>.xdc`, and is referred to as the *core* XDC file. By default, the Vivado IDE processes the core XDC files before any user constraints.

Some IP can also include another XDC file that contains clock-dependent commands. Because top-level clock can come from other constraints, or from other IP with a dependency, you must place any command that need those clocks to be defined first in the `<IP_Name>_clocks.xdc`. By default, the Vivado IDE processes the `<IP_Name>_clocks.xdc` file after user constraints and other IP core XDC files.

Most IP deliver an OOC XDC file as well, (`<IP_Name>_OOC.xdc`). This file contains default top-level definitions for input clocks to the IP. This file is only used in the DCP creation when using the recommended default flow (IP synthesized OOC to the top-level design). When the Vivado Design Suite synthesizes the IP OOC of the top-level design, clocks that are created by the end-user or other IP are not available; consequently, this file is necessary to provide the clock definitions for synthesizing the IP.

The `<IP_Name>_OOC.xdc` is not needed during implementation of the user logic with the IP, because all the netlists are linked together before constraints are applied. At that point a user-created clock or an IP-created clock is available to any IP that requires a clock.

Some IP can deliver additional XDC files. This might be because they deliver constraints that are to be used only during synthesis or only during implementation. For a list of possible XDC files that an IP can deliver, see [Appendix A, IP Files and Directory Structure](#).

Some IP support a board flow, [Appendix B, Using the IP Board Flow](#). When creating a project, if you select a development board during part selection, that board is available during the customization of the IP that lets you specify which connections on the board to which to connect the IP. This produces an `<IP_Name_board>.xdc` file which contains `PACKAGE_PIN`, `IOSTANDARD`, and other physical constraints.

For more detailed information on XDC constraints, including specifics about XDC constraints for IP, see *Vivado Design Suite User Guide: Using Constraints* (UG903) [\[Ref 5\]](#).

After some constraints are processed for a project, those constraints can become project *Properties*. For more information regarding properties, see the *Vivado Design Suite Properties Reference Guide* (UG912) [\[Ref 6\]](#).

The following subsections briefly describe some of the constraint files that the Vivado IDE creates when processing IP.



**VIDEO:** See the [Vivado Design Suite QuickTake Video: IP Constraints Overview](#), for a demonstration of how the following constraints are used during IP flow.

### **dont\_touch.xdc**

The Vivado Design Suite uses the `dont_touch.xdc` to set `DONT_TOUCH` properties on the IP top-level during synthesis of the IP. This prevents interface ports from being removed.

You can see this constraint file being processed in the synthesis log file, either for the IP when it is synthesized using OOC (the default flow), or in the global synthesis log file when synthesizing the IP with the end-user RTL.

### **dont\_buffer.xdc**

**Note:** The `dont_buffer.xdc` file is deprecated in 2014.3. IP generated prior to 2014.3 still have a `dont_buffer.xdc` file generated; the `in_context.xdc` has taken over the functionality.

The Vivado Design Suite uses the `dont_buffer.xdc` file when synthesizing the end-user logic when IP is present that uses the OOC (default) flow.

The IP is treated as a black box during top-level synthesis, and the `dont_buffer.xdc` file sets the `BUFFER_TYPE` property to `NONE` for all the interface pins of the IP.

Setting the property prevents an additional I/O buffer from being inserted during top-level synthesis.

Later, if an I/O buffer is necessary, any required I/O buffer is inserted.

### **in\_context.xdc**

IP is treated as a black box during top-level synthesis. During the creation of the IP DCP, an `<IP_Name>_in_context.xdc` file is created and stored in the IP DCP file, in the following conditions:

- The IP output any clocks
- The IP has an instance of I/O buffers

If present, the `<IP_Name>_in_context.xdc` file is processed before the end-user constraints. This file is not necessary during implementation because the IP are no longer a black box.

If clocks are created, they are placed on the boundary pins of the IP black box cell.

If I/O buffers are present, the `IO_BUFFER_TYPE` property is set to `NONE` for the interface pin with an I/O buffer. Setting the property prevents an additional I/O buffer from being inserted during top-level synthesis.

## Determining Clocking Constraints and Interpreting Clocking Messages

Vivado can contain hierarchical constraints, top-level user constraints, and constraints that are delivered by an IP. These constraints can have dependencies which must be met to work correctly. One such constraint is clock creation.

- Some IP create clocks that might be needed by other IP or the end-user top-level.
- Some IP require a clock to exist to function correctly and not produce critical warnings.

If the necessary clock constraint is not being provided, then the IP at the top-level of the design issues a CRITICAL WARNING as described in [Examples of Critical Warnings and Warnings on Clocking, page 45](#).

### *Tcl Example of an IP Clock Dependency*

The following is an example of a constraint that an IP could deliver that has a dependency on a top-level clock which enters the IP on the port `ref_clk`:

```
set_max_delay -from [get_cells data_reg] -to [get_cells synchro_stage0_reg]
-datapath_only [get_property PERIOD [get_clocks -of_objects [get_ports
ref_clk]]]
```

As explained in the *Vivado Design Suite Properties Reference Guide* (UG912) [\[Ref 6\]](#), the `get_ports` command is converted into a `get_pins` command on the IP cell instance. Depending upon how the IP is connected in a design, the clock could come either from a user-supplied clock or from another IP.

- In the case of another IP supplying the clock, the clock is provided and no critical warnings are produced.
- If the clock is provided in the end-user logic, a critical warning is produced if no clock object is created (using the `create_clock` or `create_generated_clock` command).

### *Tcl Command Examples for Clocking*

One way to find clocks in your design that are not being properly generated is to use:

```
report_clock_networks
```

This command produces a clock module for the design, including constrained and unconstrained clocks. You can use the report to determine if the clock module connected to your IP is missing a clock definition.

Another useful command is:

```
report_clocks
```

This command returns a table showing all the clocks in a design, including propagated clocks, generated and auto-generated clocks, virtual clocks, and inverted clocks in the current synthesized or implemented design.

As well as the command:

```
report_compile_order -constraints
```

This command shows which XDC files the design is using for synthesis and implementation and in what order they are processed. If an IP XDC that is creating a clock comes after an IP XDC that needs the clock, this clarifies the relationship.

Often you can resolve issues of a missing clock coming from an IP by adding a constraint to your top-level XDC timing constraints file.

This could be the case when working with an XPS design where there are no XDC files present for some of IP that could be creating a clock, such as a serial transceiver.

For more information about working with the designs with clocking requirements, see the *Vivado Design Suite User Guide: Using Constraints* (UG903) [\[Ref 6\]](#).

## Examples of Critical Warnings and Warnings on Clocking

The following are examples of the errors for a design that fails to find the clock constraint needed by a piece of IP.

```
CRITICAL WARNING: [Vivado 12-259] No clocks specified, please specify clocks
using -clock, -fall_clock, -rise_clock options [C:/Design/v_tc.xdc:1]INFO:
[Vivado 12-1399]
```

```
There are no top level ports directly connected to pins of cell
'system/v_tc', returning the pins matched for query '[get_ports s_axi_aclk]'
of cell 'system/v_tc'.
```

```
[C:/Design/v_tc.xdc:1]Resolution: The get_ports call is being converted to a
get_pins call as there is no direct connection to a top level port. This
could be due to the insertion of IO Buffers between the top level terminal
and cell pin. If the goal is to apply constraints that will migrate to top
level ports it is required that IO Buffers manually be instanced.
```

```
CRITICAL WARNING: [Vivado 12-1387] No valid object(s) found for
set_max_delay constraint with option 'from'.
```

```
[C:/Design/v_tc.xdc:1]Resolution:
```

```
Check if the specified object(s) exists in the current design. If it does,
ensure that the correct design hierarchy was specified for the object.
```

```
WARNING: [Vivado 12-584] No ports matched 's_axi_aclk'.
[C:/Design/v_tc.xdc:2]
```

```
WARNING: [Vivado 12-1008] No clocks found for command 'get_clocks -of  
[get_ports s_axi_aclk]'. [C:/Design/v_tc.xdc:2]INFO: [Vivado 12-626] No  
clocks found. Please use 'create_clock' or 'create_generated_clock' command  
to create clocks. [C:/Design/v_tc.xdc:2]
```

---

## Using Fee-Based Licensed IP

The Vivado IP catalog displays either **Included** or **Purchase** under the License column in the IP catalog. The following definitions apply to IP offered by Xilinx:

- **Included:** The Xilinx [End User License Agreement](#) applies to Xilinx LogiCORE™ IP cores that are licensed within the Xilinx Vivado Design Suite software tools at no additional charge.
- **Purchase:** The [Core License Agreement](#) applies to fee-based Xilinx LogiCORE IP, and the [Core Evaluation License Agreement](#) applies to the evaluation of fee-based Xilinx LogiCORE IP.

For more information on how to obtain IP licenses, see the Xilinx Licensing site at [http://www.xilinx.com/ipcenter/ip\\_license/ip\\_licensing.htm](http://www.xilinx.com/ipcenter/ip_license/ip_licensing.htm).

For fee-based IP, the **OK** button on the Customize IP dialog box is disabled until an evaluation or a paid license is found as shown in the [Figure 2-28, page 47](#).

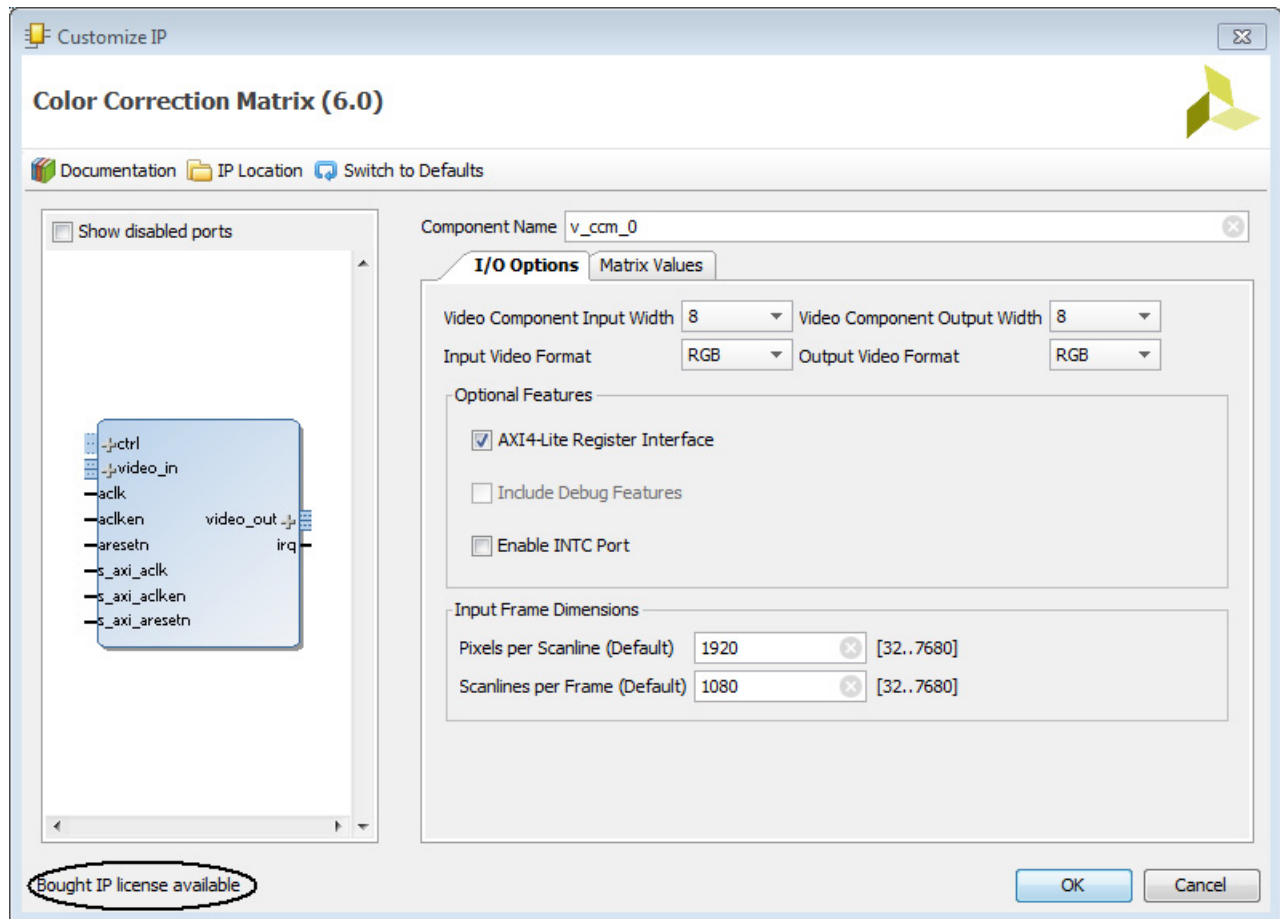


Figure 2-28: Fee-Based IP “OK” Button Enabled after License Found



**IMPORTANT:** During implementation, the Vivado tools compare both the current License status (level) and the license level of the generated IP core netlist in a design, then passes the lowest level forward; consequently, you must regenerate the LogiCORE IP core for the core netlist to receive the current license status.

The Tcl commands required to reset the and generate the target are as follows:

```
reset_target all
[get_files /project_1/project_1.srscs/sources_1/ip/<core_name>.xci]
generate_target all
[get_files project_1/project_1.srscs/sources_1/ip/<core_name>.xci]
```

## Debugging IP

The Vivado IDE includes features let you perform in-system programming and debugging of the post-implemented design in a device. The benefits for debugging your design in-system include debugging your timing-accurate, post-implemented design in the actual system environment at system speeds. The debug cores include:

- Vivado Logic Analyzer (see [ILA](#), page 49)
- Vivado Virtual I/O Analyzer (see [VIO](#), page 50)
- IBERT Serial Analyzer (see [IBERT](#), page 50)
- JTAG to AXI (see [JTAG-to-AXI](#), page 50)

You can use the Vivado Hardware Manager to test and verify the IP capabilities when attached to a Xilinx board with a JTAG connection.

See the following documents for more information:

- *Vivado Design Suite Tutorial: Programming and Debugging* (UG936) [\[Ref 7\]](#)
- *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [\[Ref 8\]](#)
- *UltraFast Design Methodology Guide for the Vivado Design Suite* (UG949) [\[Ref 9\]](#)

## Using Debug Flows

The Vivado tools provide several methods to add debug probes in your design. You need to determine which flow suits the requirements of your design. The available debug flows are:

- **HDL instantiation probing flow:** This flow involves explicitly attaching signals in the HDL source to an ILA debug core instance.
  - Advantage: Provides the ability to probe at the HDL design level.
  - Disadvantages:
    - You must add and remove debug nets and IP from your design manually, which requires that you modify your HDL source
    - Easy to make mistakes when generating, instantiating, and connecting debug cores.
- **Netlist insertion probing flow** (Recommended -only available with ILA 2.1 or later): this flow involves explicitly attaching signals in the HDL source to an ILA debug core instance:
  - Use the `MARK_DEBUG` attribute to mark signals for debug in the source RTL code.
  - Use the `MARK_DEBUG` right-click menu option to select nets for debugging in the synthesized design netlist.



The second step uses the Set up Debug wizard to guide you through the netlist insertion probing flow.

- Advantages:
  - Most flexible with good predictability.
  - Allows probing at different design levels (HDL, synthesized design, system design).
  - Does not require HDL source modification.
- Disadvantages: None
- **Tcl-based netlist insertion probing flow:** Use the `set_property` Tcl command to set the `MARK_DEBUG` property on debug nets then use netlist insertion probing Tcl commands to create debug cores and connect them to debug nets.
  - Advantages:
    - Fully automatic netlist insertion.
    - Turn debugging on or off by modifying the Tcl commands.
    - No required HDL source modification.
  - Disadvantages: Requires knowledge of Tcl commands.

## Marking Debug on Nets

Using `MARK_DEBUG` on nets of interest and adding the debug cores into the design is a two-step process:

1. Identify the nets for debug by right-clicking on nets and setting the `MARK_DEBUG` attribute or use the properties window and/or Tcl commands to set this property.
2. After synthesis, use the Set up Debug wizard. You can access the wizard from the Flow Navigator Synthesis tab or when in the design select **Tools > Set up Debug**.

## Vivado Debug Cores

The following sections describe the IP that assist with debugging customized Vivado IP.

### ILA

The integrated logic analyzer (ILA) also called *Vivado logic analyzer*, lets you perform in-system debugging of post-implemented designs on an FPGA. Use this feature when you need to monitor signals in a design.

You can also use this feature to trigger on hardware events and capture data at system speeds. You can instantiate the ILA core in your RTL code or insert the core, post-synthesis, in the Vivado design flow. See the *LogiCORE IP Integrated Logic Analyzer Product Guide* (PG172) [Ref 25].

## VIO

The virtual input/output (VIO) debug feature, also called the *Vivado serial I/O analyzer* can both monitor and drive internal FPGA signals in real time.

In the absence of physical access to the target hardware, you can use this debug feature to drive and monitor signals that are present on the real hardware.




---

**IMPORTANT:** *This debug core must be instantiated in the RTL code; consequently, you need to know what nets to drive.*

---

The IP Catalog lists this core under the Debug category. See the *LogiCORE IP Virtual Input/Output Product Guide* (PG159) [Ref 29].

## IBERT

The integrated bit error ratio tester (IBERT) serial analyzer enables in-system serial I/O validation and debug. This allows you to measure and optimize your high-speed serial I/O links in your FPGA-based system.




---

**RECOMMENDED:** *Use the IBERT Serial Analyzer when you are interested in addressing a range of in-system debug and validation problems from simple clocking and connectivity issues to complex margin analysis and channel optimization issues. Use the Vivado IBERT serial analyzer design when you are interested in measuring the quality of a signal after a receiver equalization is applied to the received signal. This ensures that you are measuring at the optimal point in the TX-to-RX channel and thereby real and accurate data.*

---

You can access this design by selecting configuring, and generating the IBERT core from the IP catalog and selecting the **Open Example Design** feature of this core.

See details on the this IP in the following documents:

- *LogiCORE IP IBERT for 7 Series GTX Transceivers* (PG132) [Ref 26]
- *LogiCORE IP IBERT for 7 Series GTP Transceivers* (PG133) [Ref 27]
- *LogiCORE IP IBERT for 7 Series GTH Transceivers* (PG152) [Ref 28]

## JTAG-to-AXI

- The JTAG-to-AXI debug feature generates AXI transactions that interact with various AXI4 and AXI4-Lite slave cores in a system that is running in hardware.




---

**IMPORTANT:** *Use this core to generate AXI transactions and debug and to drive AXI signals internal to an FPGA at run time. You can use this core in IP designs without processors as well. The IP Catalog lists the core under the **Debug** category.*

---

## Debugging with Simulation

Simulating customized IP lets you see if you are achieving the results you expect. See the following for more information about simulation options:

- [Chapter 6, Simulating IP](#)
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [\[Ref 14\]](#)

## Using a Test Bench for IP

The IP delivered in the IP Catalog delivers IP with output products that can perform behavioral simulation.

Many IP also deliver a test bench for simulating the IP standalone. If an IP delivers a test bench you will see it listed as an output product in the Generate Output Product dialog box, as shown in the following figure.

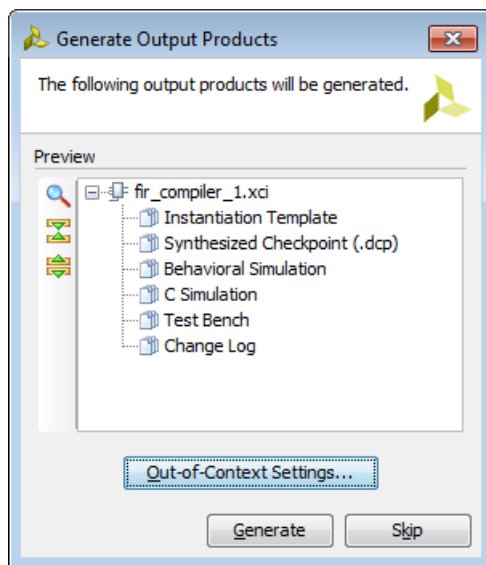



Figure 2-29: Test Bench Output Product

To use the test bench provided by the IP, do the following:

1. Find the IP in the hierarchy in the Simulation Sources section and expand the IP hierarchy. You do this by pressing the:
  - Circle (Linux)
  - Plus sign  (Windows) next to the IP in the `sim_1` set.

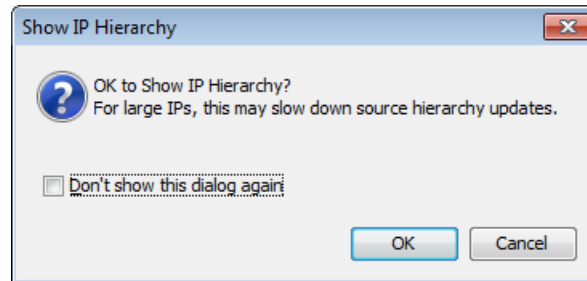


Figure 2-30: Expanding the IP hierarchy

The Show IP Hierarchy dialog displays as shown in the following figure.

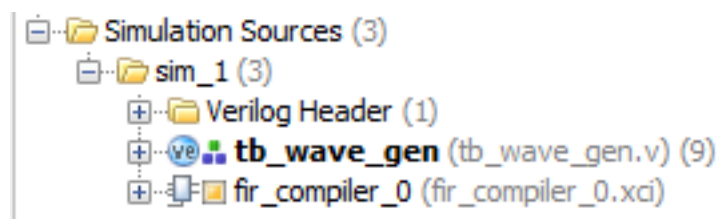


Figure 2-31: Show IP Hierarchy Dialog

2. Click **OK** to expand the hierarchy.
3. Find and select the IP test bench, named `tb_<IP_Name>`.
4. Right-click and select **Set as Top**.
5. You could get an Invalid Top Module dialog box. If so, select the **Ignore and continue with invalid top module** and click **OK**, as shown in the following figure.

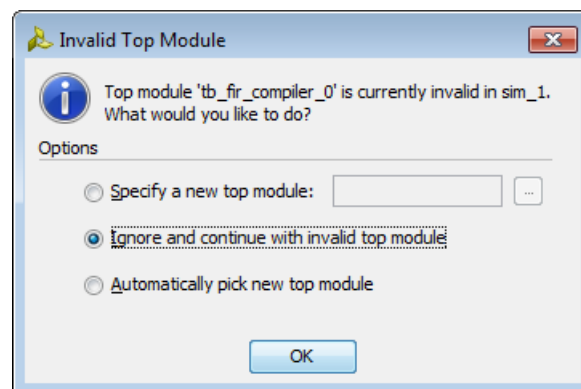


Figure 2-32: Invalid Top Module

Setting the IP Test Bench as the top module can result in the IP synthesis run being set out-of-date because a project level property changed.

If this happens you can force the run up-to-date by right-clicking and selecting **Force Up-to-Date**, as shown in the following figure. If you do not do this, the next time you launch top-level synthesis the IP will resynthesize.

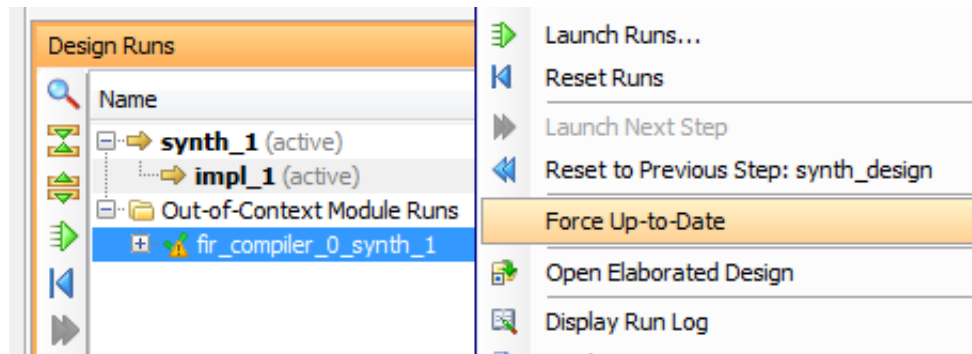


Figure 2-33: Force IP Synthesis Run Up-to-Date

In the Flow Navigator select Run Simulation or launch simulation in the Tcl Console.

**Note:** You can look at the current simulation settings in the **Project Settings** in the Simulation section. Here you see the simulation top-module name, which should match the IP test bench that you set. You can change the simulator setting here, which affects the behavior of the **Run Simulation** button.

# Using Manage IP Projects

---

## Introduction

The Vivado® Integrated Design Environment (IDE) provides mechanisms for:

- Exploring IP in the IP Catalog
- Customizing IP
- Managing centralized location of customized IP.

The Vivado IDE can create a special project, referred to as a *Manage IP* project, which facilitates the management of IP customizations. From this project type, you can view the IP catalog, customize IP, and generate output products. The IP customization (XCI) and generated output products are stored in separate directories located outside of the Manage IP project. The Manage IP project manages the IP design runs for the generation of the synthesized design checkpoint (DCP) files.



**RECOMMENDED:** *When working in teams, or if the design uses many Xilinx® IP, create and maintain your customized IP in a location outside of the Vivado project structure. This method makes revision control more straightforward and allows for ease of sharing customized IP with others. This is also the recommended methodology for working with IP in a non-project, script-based flow.*

---

---

## Managed IP Features

When you use the Manage IP flow in the Vivado IDE, the following features are available:

- Simple IP project interface
- Direct access to the Xilinx IP Catalog
- Ability to customize multiple IP
- Separate, unique directories for each IP instantiation with all related IP files
- Option to generate or skip generation of the Design CheckPoint (DCP) file. A DCP file consists of both a netlist and constraints for the IP, and creating this file is the default flow.

## Using the Manage IP Flow

1. Invoke the Vivado IDE, and from the **Getting Started** page, select **Manage IP**, as shown in the following figure.

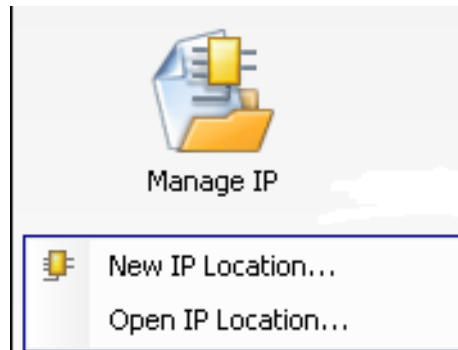


Figure 3-1: Invoking the Manage IP Flow

2. Open a new or an existing IP location.
  - **New IP Location:** Opens a new IP project at the location specified for exploring the IP catalog and customizing IP, including generation of output products.
  - **Open IP Location:** Lets you navigate to a location from which to open an IP.
  - **Recent Projects:** Lists recent locations from which to open an IP Project. If the specified location already contains an IP Project, then a window opens asking if you would like to open the existing project.

When you select the **New IP Location** option, the **Create a New Customized IP Location** menu informs you that a wizard will guide you through creating and managing a new customized IP location, as shown in [Figure 3-2, page 56](#).

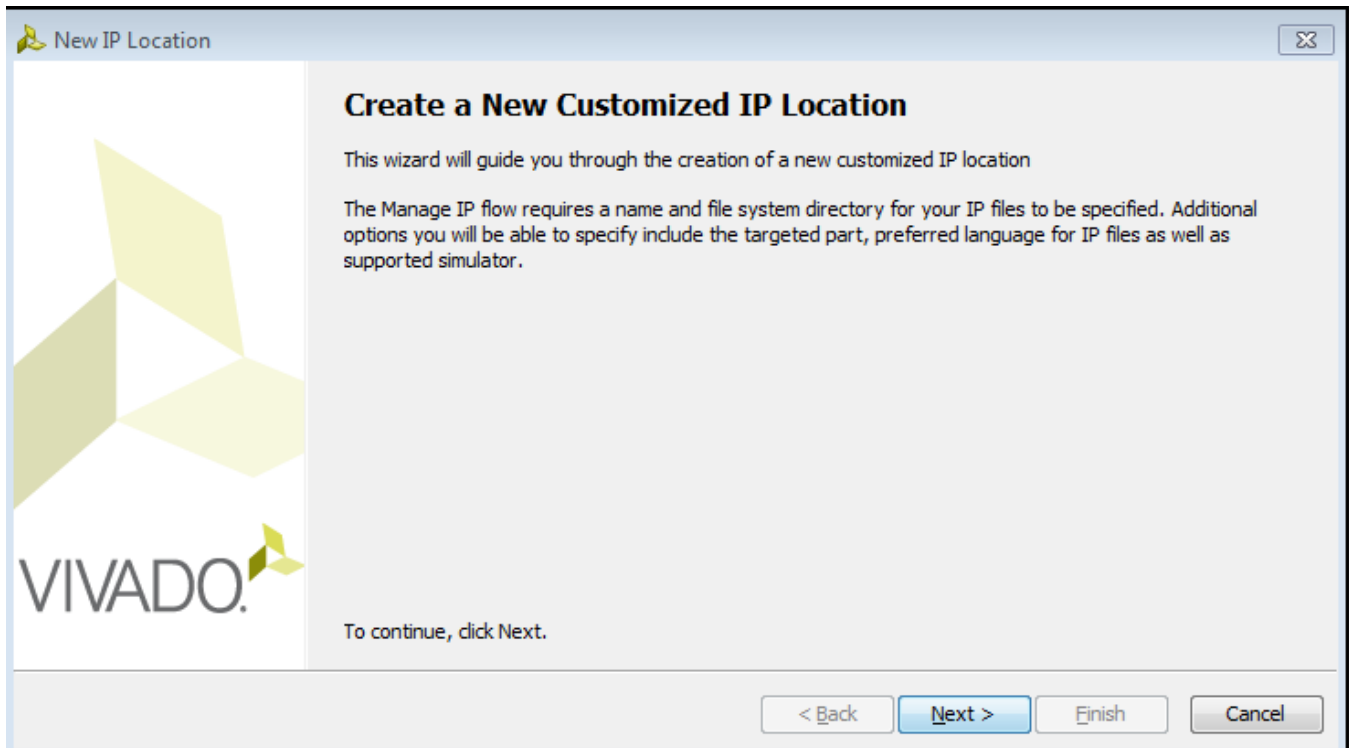


Figure 3-2: Create a New Customized IP Location

If the location specified to create a new manage IP project already contains a project, a dialog box opens (see the following figure).

You can either choose to open the existing project or cancel to go back to the **New IP location** and specify a different location.

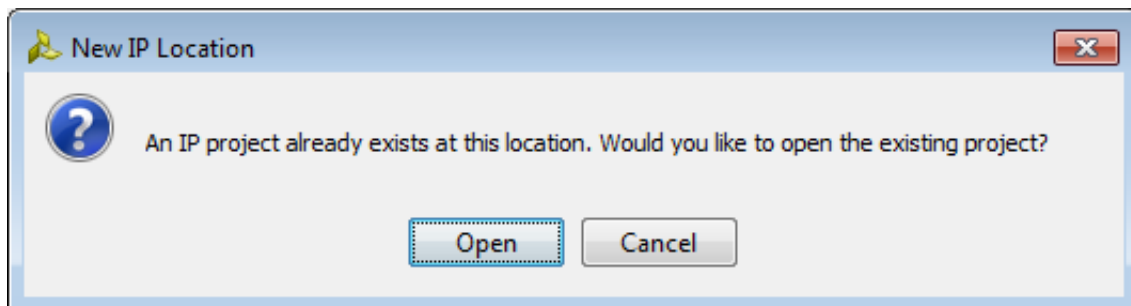


Figure 3-3: New IP Location Dialog Box

3. Select **Next**.

The Manage IP Settings dialog box opens (Figure 3-4, page 57).



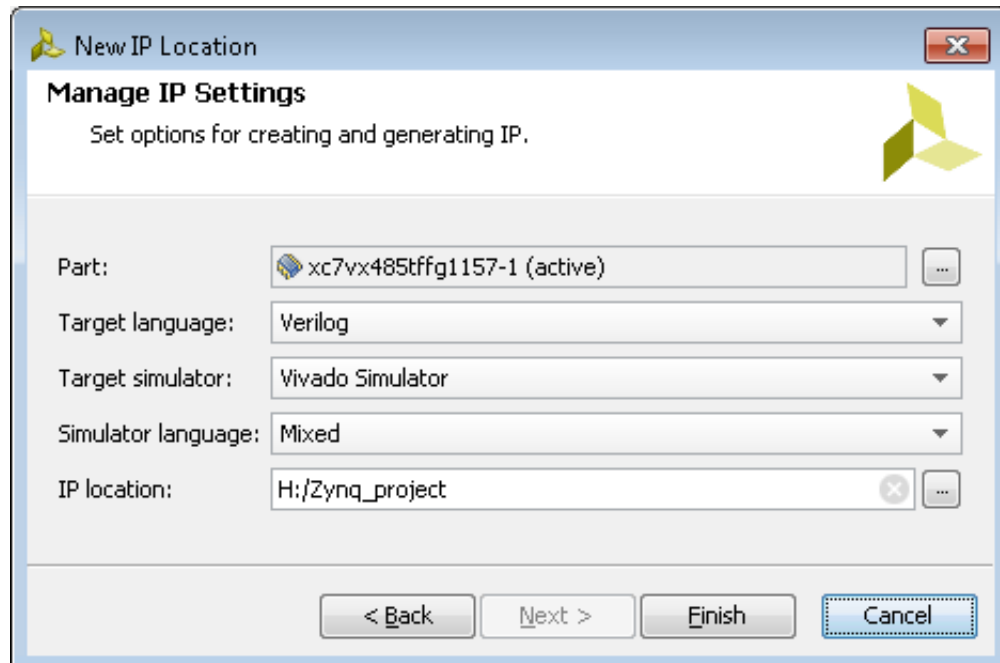


Figure 3-4: Manage IP Settings Dialog Box

## Managing IP Settings

To manage the IP settings:

1. Enter the following:
  - **Part:** Select the active part.
  - **Target language:** Set the target language to VHDL or Verilog depending on your design top.
  - **Simulator language:** Options are **VHDL**, **Verilog SV**, or **Mixed** depending on the license that you have available for the simulator. For the Vivado simulator, the default is **Mixed**.
  - **IP location:** The location where the Vivado IDE creates the `managed_ip_project` directory.
2. Click **Finish**.

The Manage IP window opens, and you can now select and customize IP.

You have access to the full IP catalog, including IP Product Guides, Change Logs, Product web pages, and Answer Records.

After you customize an IP, the Sources and Properties windows display, providing information about the IP created in the project.

The following figure shows the Manage IP Project window where you customize and manage multiple IP.

Each IP customization has a directory created under the specified manage IP location. This directory contains the XCI file and any generated output products.

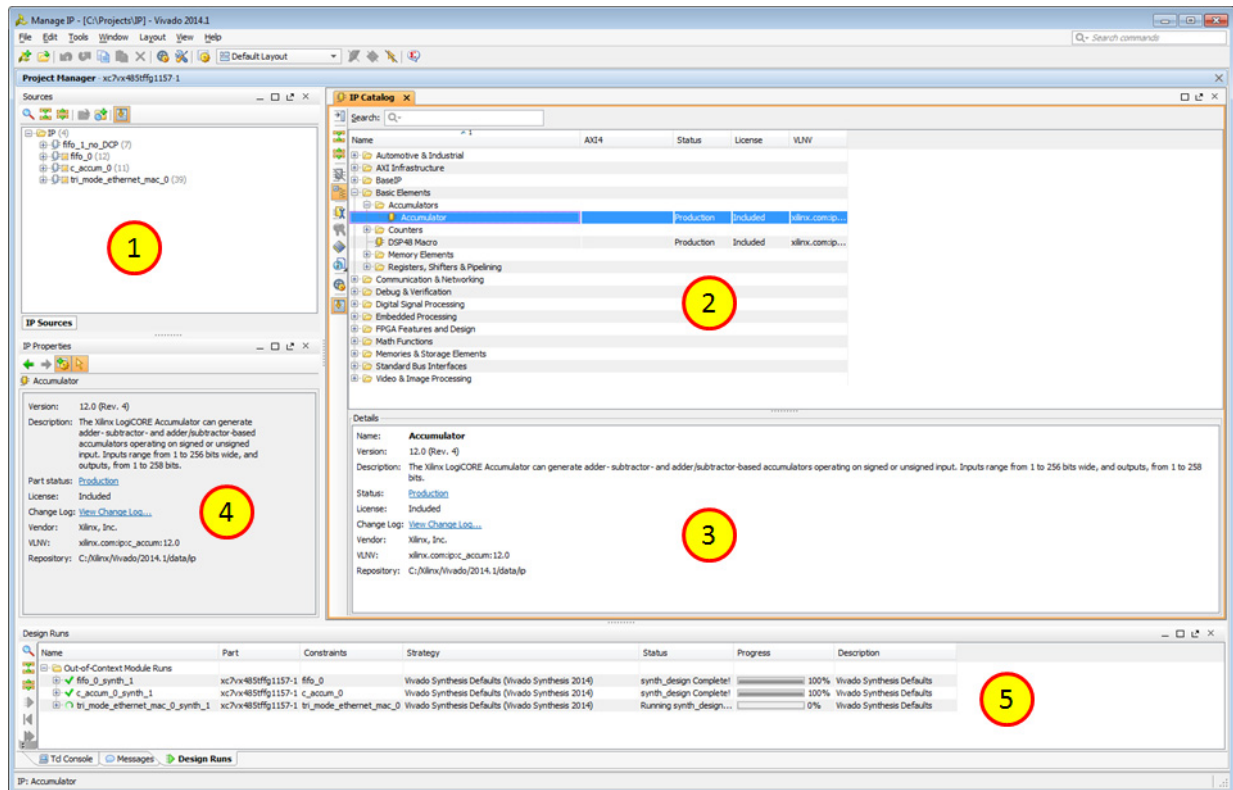


Figure 3-5: Manage IP Project Window Containing Three IP

The main sections to an IP Project window are as follows:

1. **IP Sources:** Lists the IP that are customized for the project, where you can view the output products and manage the generation of additional output products.
2. **IP Catalog:** Lets you explore the IP catalog and create customized IP to add to the IP Project.
3. **Details:** Displays the details of the selected IP.
4. **IP Properties:** Displays the properties and general detail information for the selected IP.
5. **Design Runs:** If you generate a synthesis design checkpoint (DCP) output product, a run for the IP shows in this view.

See [Appendix A, IP Files and Directory Structure](#) for a list of possible files and directories that the Vivado IDE can create for IP, depending upon the requirements.

# Using IP Example Designs

## Introduction

Many Xilinx® IP deliver an example design project. The example design project consists of top-level logic and constraints that interact with the created IP customization. These example designs typically come with an example test bench that helps simulate the design.

## Opening an Example Design

To open an example design project for an IP, select the IP customization in the IP Sources tab, then right-click and select **Open IP Example Design** from the context menu.

This opens the **Open IP Example Design** window for you to specify the location. The project is called <IP\_Name>\_ex.

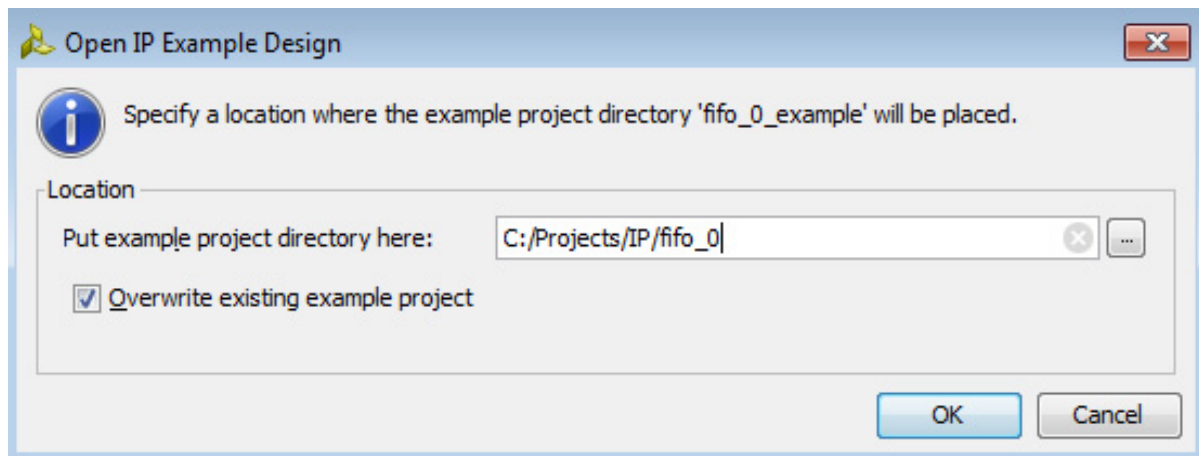


Figure 4-1: Open IP Example Design Dialog Box

A new session of the Vivado IDE opens that shows the example design in the Design Sources window (Figure 4-2, page 60).

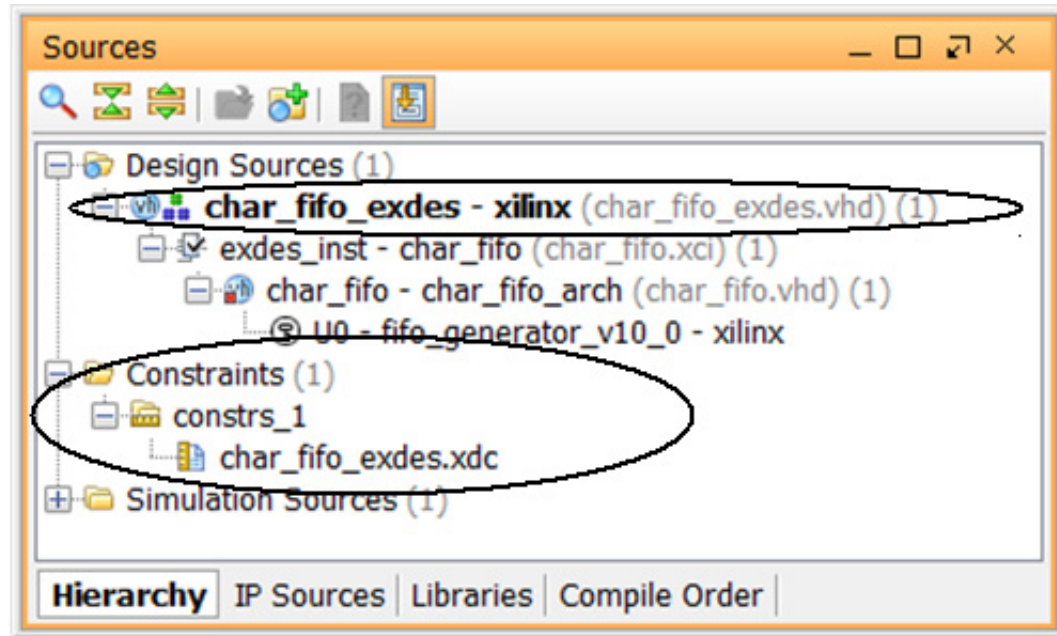


Figure 4-2: IP Example Design Instance with Constraint File

The IP is instantiated in the example design with an example XDC constraint file to enable further evaluation of the IP.

## Tcl Command to Open a Project

Alternatively, you can use the following Tcl command to open a project:

```
open_example_project [get_ips <IP_Name>]
```

You can use the Tcl help to find other available options.

# Using Xilinx IP with Third-Party Synthesis Tools

---

## Introduction

Xilinx® supports netlists created by third-party synthesis tools for user logic. When using Xilinx IP the only supported synthesis tool is the Vivado® synthesis tool. The Vivado IDE infers a black box during logic synthesis for Xilinx IP. A file is provided to infer the black box as described in the following sections. The Vivado IDE resolves the black boxes during implementation.

Synthesis of Xilinx IP is supported with the Vivado synthesis tool only, including the IP core and any example design files an IP might deliver.



**IMPORTANT:** *Xilinx encrypts IP HDL files with the IEEE Recommended Practice for Encryption and Management of Electronic Design Intellectual Property (IP) (IEEE Std P1735) encryption: consequently those files are readable only when using the Vivado IDE for synthesis. You can use a third-party synthesis tool for the end-user logic and generate a netlist that Vivado implementation can use. Support is enabled for third-party simulation tools to perform behavioral simulations using the encrypted RTL.*

---

---

## Synthesis Flow

When using a Synopsys® Synplify Pro or a Mentor® Graphics Precision netlist for synthesis with a design that has Xilinx IP, the recommended flow is:

1. Use the **Manage IP** flow in to create and customize IP.
2. Generate output products for the IP, including the synthesis design checkpoint (DCP) for each IP.

When you generate the DCP file, two additional files are created to infer the black box when used with the third-party synthesis tool: `<IP_Name>_stub.v` and `<IP_Name>_stub.vhdl`.

3. Add the Verilog stub file or the VHDL stub file to the project with the third-party synthesis tool.

The Verilog or VHDL stub file infers a black box for the Xilinx IP, and prevents the synthesis tool from adding I/O buffers. The `<IP_Name>_stub.v`, and the `<IP_Name>_stub.vhdl` contain synthesis directives that instruct the third-party synthesis tool to not infer I/O buffers for the IP if the IP connects to top-level ports. Change these directives as required for the third-party synthesis tool being used.

4. Generate a netlist with the third-party synthesis tool.
5. Create a Vivado netlist project. See the *Vivado Design Suite User Guide: System-Level Design Entry* (UG895) [Ref 15] for more information about netlist projects.
6. Add the following into the Vivado netlist project:
  - The netlist from the third-party synthesis tool
  - User-level top-level constraints
  - The XCI files for the IP (one XCI file per IP)

The netlist in the IP DCP as well as the XDC output products are used automatically during implementation when using the XCI file for the IP.

When you use the **Add Sources** button to add design sources, select **Add Files** and change the **Files of type** to **All Files** so that the XCI files are listed.

7. Implement the design in the Vivado IDE.

Vivado implementation adds any required I/O buffers if they are not already present in the DCP of the IP.




---

**RECOMMENDED:** Use the IP XCI file when referencing Xilinx IP in either a Project Mode or Non-Project Mode and not the DCP file directly. While the DCP does contain constraints, they are resolved Out-Of-Context of the end-user constraints. Using the XCI results in the XDC output product for the IP being applied after all the netlists are combined (end-user and IP). Additionally, any Tcl script in the IP XDC is then evaluated in context of the end-user constraints and netlist.

---

## Using a Third-Party Synthesis as a Black Box Flow in Project Mode

In Project Mode, to synthesize top-level logic separately with a third-party synthesis tool, create an EDIF, then include that EDIF into Vivado, along with the IP-level DCP files, and run implementation.

### Example Tcl Script for Project Mode

```
# Create a project on disk
create_project <name> -part <part>

# configure as a netlist project
set_property design_mode "GateLvl" [current_fileset]

# Add in the netlist from third-party synthesis tool
add_files top.edif

# Add in XCI files for the IP
add_files {ip1.xci ip2.xci ip3.xci}

# Add in top level constraints: this might include XDC files from the third-party
# synthesis tool
add_files top.xdc
# Launch implementation
launch_run impl_1 -to write_bitstream
```

## Using a Black Box Flow in Non Project Mode

To synthesize top-level logic separately with a third-party synthesis tool, create an EDIF, then include that EDIF back into Vivado, along with the IP-level DCP files, and run implementation.

### Example Tcl Script for Non Project Mode

```
# Create an in memory project and set part
create_project -in_memory -part <part>

# Read the netlist from third-party synthesis tool
read_edif top.edif

# Read in the IP XCIs
read_ip ip1.xci
read_ip ip2.xci

# read in top level constraints
read_xdc top.xdc
# Implement the design
link_design -top <top>
opt_design
place_design
phys_opt_design
route_design
write_bitstream -file <name>
```

**Note:** Ensure when reading in the IPs that you are reading the XCI file from the location where the output products of the IP were previously generated or alternatively, read in the XCI file and then generate the IP using the `synth_ip` command.

# Simulating IP

---

## Introduction

Design simulation is an important and necessary step in the design flow to verify the functionality and performance of the design. To enable this, IP in the Vivado® IP catalog delivers a simulation model that you can include in the simulation of the overall design.

The simulation model delivered for the IP can be one of the following:

- Custom behavioral simulation model
- Plain text or encrypted synthesizable RTL sources used for simulation
- Structural simulation model

**Note:** Some IP (for example, the FIR Compiler IP) deliver IP-level test benches that you can directly use to simulate the IP.

Third-party simulators that are typically used for simulating Xilinx devices are integrated as options in the Vivado® Integrated Design Environment (IDE).

You can use the `launch_simulation` Tcl command after selecting the preferred simulator and language.

Vivado IDE automatically compiles the necessary libraries and generates the simulation.

See the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 14] for more information.

---

## Delivering IP Simulation Models

Most Xilinx® IP deliver RTL sources for a single language only, effectively disabling simulation for *language locked* simulators if you do not have licensing for the appropriate language.

To simulate your design and include IP, Vivado ensures the availability of an appropriate simulation model for the IP using the project property **Simulator language** setting. The



values are **Verilog**, **VHDL**, and **Mixed**. Set this property Manage IP, Project-based, and Non-project based flows.

Some IP deliver simulation files for VHDL and some for Verilog. When the simulator language is set to **Mixed** it is possible that the same module for both languages can be sent to the simulator by different IP.

The Vivado simulator is a mixed language simulator and can handle simulation models in both VHDL and Verilog. If you are using other simulators and have license for a single language only, change the Simulator language to match your license.

If the IP does not deliver a behavioral model or does not match the chosen and licensed simulator language, the Vivado tools automatically generate a structural simulation model (`_funcsim.v` or `_funcsim.vhdl`) to enable simulation.



**RECOMMENDED:** When you generate IP output products, enable the synthesized design checkpoint (.DCP) option to ensure that the Vivado IDE can deliver a structural simulation netlist for the IP. For more information, see [Re-Customizing Existing IP in Chapter 2](#).

## Tcl Command to Set Simulator Language

To set the simulator language property use:

```
set_property simulator_language <language_option> [current_project]
```

[Table 6-1](#) illustrates how the `simulator_language` property controls the delivery of the IP simulation model.

**Table 6-1: Simulator Language Property**

Simulation Model	Language	Simulation Model
IP delivers VHDL and Verilog behavioral models	Mixed	Behavioral model (target_language)
	Verilog	Verilog behavioral model
	VHDL	VHDL behavioral model
IP delivers Verilog behavioral model only	Mixed	Verilog behavioral model
	Verilog	Verilog behavioral model
	VHDL	VHDL simulation netlist generated from the IP DCP
IP delivers VHDL behavioral model only	Mixed	VHDL behavioral model
	Verilog	Verilog simulation netlist generated from the IP DCP
	VHDL	VHDL behavioral model
IP deliver no behavioral models	Mixed/Verilog/VHDL	Netlist generated from DCP (target_language)

**Note:** Where available, a *Behavioral Simulation* always takes precedence over a *Structural Simulation*. Vivado does not offer a choice of simulation model.

**Note:** The setting for the project property `target_language` is used to deliver simulation models when the IP supports both; either language could be used for simulation, as shown in the following example:

### Tcl Command to Set Properties on the Target Language

```
set_property target_language <language_option> [current_project]
```

## Simulating IP

Simulators that are typically used to simulate FPGA designs are integrated in the Vivado IDE when you are in Project Mode. [Figure 6-1](#) shows the available options.

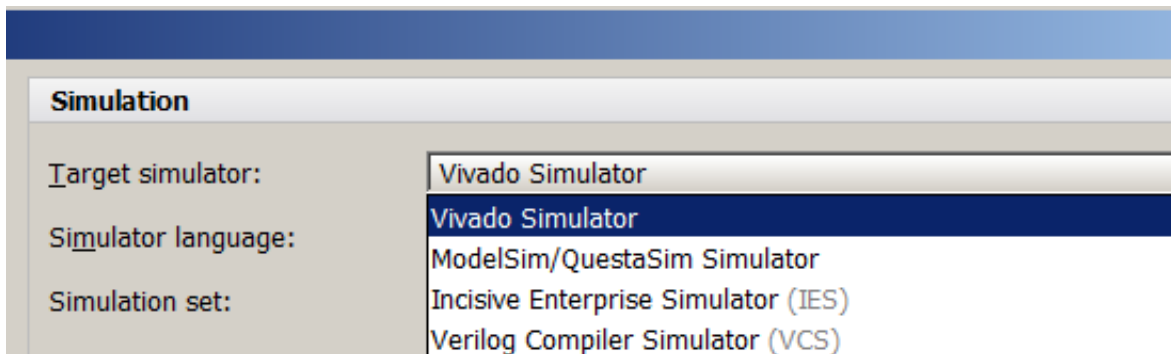


Figure 6-1: Simulation Project Settings

To simulate your design instantiating Xilinx IP:

1. In **Project Settings > Simulation**, set the **Target simulator** to the simulator of your choice.
2. From the **Flow Navigator**, select **Run Behavioral Simulation** to start the simulation.

Alternatively, you can run simulations outside of the Vivado tools environment, using a Tcl command to generate scripts for the target simulator.

### Tcl Command to Create Simulation Scripts

You can produce a run script using the `launch_simulation` command:

```
launch_simulation -scripts_only -mode behavioral
```

This produces a script for use with the target simulator.

**Note:** To launch third-party simulator tools, the specified simulator must be installed, and appear in your `$PATH` to be properly invoked when launching simulation.

## Tcl Command to Change Target Simulator

To change the target simulator, set the target simulator property for the project:

```
set_property -target_simulator <target_simulator> [current_project]
```

Where:

<target\_simulator> is one of the following options: ModelSim, VCS, IES, XSim

## Tcl Command to Compile Simulation Libraries

To support the use of QuestaSim/ModelSim you must compile the Xilinx simulation libraries for use with the target simulator:

```
compile_simlib
```

After the libraries are compiled, the simulator references these compiled libraries using the `modelsim.ini` file. The `modelsim.ini` file is the default initialization file and contains control variables that specify reference library paths, optimization, compiler and simulator settings. The `modelsim.ini` is located as follows:

- The path specified by the `-directory` argument at the time `compile_simlib` is run.
- The path defined by `$MODELSIM` environment variable.
- The path defined by `$MGC_WD` environment variable.
- The simulation run directory of the project.

**Note:** If the `modelsim.ini` file is not found at any of these locations a warning message is returned by the simulator. The command returns the transcript of the simulator.

For more details on using QuestaSim/ModelSim, see *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 14].

---

## Tcl Commands for IP Simulation

If you want to manage your own scripts for simulating IP, you can use the `get_files` command to get all the files that an IP delivers for simulation.

```
get_files -compile_order sources -used_in simulation \
-of_objects [get_files <IP name>.xci]
```

Where:

- `compile_order` supports RTL sources only.
- `used_in` lets you specify files used in Vivado simulation or synthesis.
- `of_objects` takes the IP XCI file object from which all the files related to the IP can then be filtered.

For VHDL, you must specify the library associated with each file also. When you use a Tcl script, you can determine this information quickly.

The following script prints out each simulation file, and its file path, as well as the associated library:

```
# Get the list of files required for simulation
set ip_files [get_files -compile_order sources -used_in simulation \
-of_objects [get_files <IP_Name>.xci]
# For each of these files, get the library information
foreach file $ip_files {
puts "[get_property LIBRARY $file] $file"
}
```

The `LIBRARY` specifies the default library for the project. All files without an explicit library specification are compiled in the `xil_defaultlib`, which is created by the Vivado IDE. You can select a library name, or specify a new library name by typing in the **Library** text field.

---

## Using Cadence Incisive Enterprise Simulator and Synopsys VCS MX Simulator

Xilinx encrypts the Vivado IDE IP using industry standard IEEE P1735 encryption; you can use simulators that support this encryption to do behavioral simulation. Behavioral simulation requires a list of simulation files and the libraries to which they belong.

The Vivado IDE can generate scripts for Cadence® Incisive Enterprise Simulator, (`ies`) and the Synopsys® VCS MX simulator (`vcs_mx`). The generated scripts contains simulator commands for compiling, elaborating, and simulating the design. The script commands retrieve the simulation compile order of specified objects, and export this information in a text file with the compiler commands and default options for the target simulator.

The specified object can be either a simulation file set or an IP.

# Working with Revision Control

---

## Introduction

The Vivado® Design Suite is designed to work with any revision control system.

For IP, there are strategies to consider, and each has a trade-off on either run time or the number of files that must be managed. This chapter discusses these strategies in detail.

For more information on how to use Vivado Design Suite with version control systems, see the following:

- *Vivado Design Suite User Guide: Creating and Packaging Custom IP* (UG1118) [\[Ref 1\]](#)
  - *Vivado Design Suite Tutorial: Creating and Packaging Custom IP* (UG1119) [\[Ref 2\]](#)
- 

## Using Revision Control

When working with revision control systems and Xilinx IP there are a few possible options:

- Full Revision Control
- Partial Revision Control

### Full Revision Control

Place the entire IP directory into revision control, including all output products as well as the synthesized design checkpoint (DCP).



**RECOMMENDED:** *This is the recommended option because it gives the flexibility to decide when and if to upgrade the IP at a future point. It also provides better run time as the IP does not have to be regenerated every time it is used.*

---

The Vivado IDE only supports one version of an IP in the IP catalog. If you upgrade the tool and the IP is no longer current, it is still usable. The IP is locked and you are not able to re-customize or generate output products; however, if all the output products are present, the IP can be used.

## Partial Revision Control

Though it is not recommended, you can put a subset of the files associated with an IP into revision control. The following are some options and their limitations:

- Only the IP XCI file. This is the absolute minimum of what could be placed in revision control. If the IP is current, from the XCI all required output products can be generated, including the DCP.
  - **Limitation:** If the IP is not current your only option is to upgrade.
- The IP XCI file plus a subset of the output products which could include a structural simulation model and, or, the synthesis Design Checkpoint (DCP). This way the customization of the IP is retained and IP can be upgraded to the latest version if necessary. If you do not want to upgrade the IP, you can continue to use the DCP. The structural simulation model is used for simulating the IP.
  - **Limitation:** The XCI file does not track the output product generation; this is done by the BOM XML file. You must use the DCP or any other output products directly, and not through the XCI. The XCI would be used only to track the IP upgrade status.
  - **Limitation:** If constraints in the DCP are applied standalone, any Tcl queries would be resolved out-of-context of the entire design. When the IP is fully generated, only the netlist is used from the DCP; the constraints are applied from the XDC output product files.

### Tcl Script to Get All IP Files

To get a list of all the output products generated for an IP, use the following Tcl command:

```
get_files -all -of_objects [get_ips <IP_Name>]
```

This commands lists all files generated for the IP including:

- BOM XML synthesis and simulation targets
- Created DCPs
- Third-party synthesis stubs
- Structural simulation models

### Tcl Script for Synthesis-Only Files

To get a list of the files used for synthesis only, use the following Tcl command:

```
get_files -used_in synthesis -compile_order sources -of_objects [get_ips <IP_Name>]
```




---

**IMPORTANT:** Without the BOM XML file the `-used_in *` commands are not associated with the IP XCI.

---

### ***Tcl Script for Simulation-Only Files***

To get a list of the files used for simulation you can use the following Tcl command:

```
get_files -used_in simulation -compile_order sources -of_objects [get_ips <IP_Name>]
```

### ***Tcl Command to Archive a Project***

To create an archive of the entire project, including any IP, use the following command:

```
archive_project
```

This command consolidates all the project files, bringing external ones into the archive project to ensure a complete archival. The original project is not modified.

## Tcl Commands for Common IP Operations

### Introduction

This chapter covers the Tcl commands to use for common IP operations.

For more information about using Tcl and Tcl scripting, see the following:

- *Vivado Design Suite User Guide: Using Tcl Scripting* (UG894) [\[Ref 19\]](#)
- *Vivado Design Suite Tcl Command Reference Guide* (UG835) [\[Ref 11\]](#)

For a step-by-step tutorial that shows how to use Tcl in the Vivado tool, see the *Vivado Design Suite Tutorial: Design Flows Overview* (UG888) [\[Ref 18\]](#).

### Using IP Tcl Commands In Design Flows

Generally, the IP Tcl commands used for working are consistent between the Project Mode flow and the Non-Project Mode flow with a few exceptions related to setting the part to be used for IP creation and synthesis. The following table lists the Tcl command in the order that you would use them in a design.

**Table 8-1: IP Tcl Commands in Order of Design Use**

Action	Project Mode Command	Non-Project Mode Command
Set part for IP creation	N/A. Part is a project setting.	<pre>create_project -in_memory / -part &lt;part_name&gt;</pre> <p><b>Note:</b> Creates project in memory, not on disk. Commands that have a part option, for example, <b>synth_design</b> then use the specified part.</p>
Create an IP Customization	<pre>create_ip &lt;IP_Name&gt;</pre>	<pre>create_ip &lt;IP_Name&gt;</pre>
Configure IP Customization	<pre>set_property \   CONFIG.Input_Data_Width 8 \   [get_ips &lt;IP_Name&gt;]</pre>	<pre>set_property \   CONFIG.Input_Data_Width 8 \   [get_ips &lt;IP_Name&gt;]</pre>
Generate output products	<pre>generate_target \   [get_ips &lt;IP_Name&gt;]</pre> <p><b>Note:</b> Optionally, you can specify the target(s) you want to generate.</p>	<pre>generate_target \   [get_ips &lt;IP_Name&gt;]</pre> <p><b>Note:</b> Optionally, you can specify the target(s) you want to generate.</p>



Table 8-1: IP Tcl Commands in Order of Design Use (Cont'd)

Action	Project Mode Command	Non-Project Mode Command
Synthesize IP to create OOC DCP	<pre>create_ip_run \   [get_ips &lt;IP_Name&gt;] launch_runs &lt;IP_Name&gt;_synth_1</pre>	<pre>synth_ip [get_ips &lt;IP_Name&gt;]</pre>
Read an IP	<p>Copy an IP into a project along with any output products:</p> <pre>import_ip &lt;ip_name&gt;.xci</pre> <p>Add the IP to a project along with any output products and reference from the specified location. Use either of the following:</p> <pre>add_files &lt;ip_name&gt;.xci read_ip &lt;ip_name&gt;.xci</pre> <p><b>Note:</b> <code>import_ip</code> allows you to rename the IP using the <code>-name</code> option.</p>	<p>Read the IP as well as any generated output products. Use either of the following:</p> <pre>add_files &lt;ip_name&gt;.xci read_ip &lt;ip_name&gt;.xci</pre> <p><b>Note:</b> Unlike in the Project Flow, the output products are not generated automatically. You must generate them using the <code>generate_target</code> command.</p> <p>If you use the <code>synth_ip</code> command to produce a DCP for the IP, it is not necessary to generate the output targets first; those targets are generated automatically.</p>
File queries	<pre>get_files -of_objects \   [get_ips &lt;IP_Name&gt;]</pre>	<pre>get_files -of_objects \   [get_ips &lt;IP_Name&gt;]</pre>
Simulation	See <a href="#">Chapter 6, Simulating IP</a> .	See <a href="#">Chapter 6, Simulating IP</a> .

## Tcl Commands for Common IP Operations

Within the Vivado IDE, the Vivado IP Catalog can be accessed from the Vivado IDE and the Tcl design environment.

To accommodate end-users that prefer batch scripting mode, every IP Catalog action such as IP creation, re-customization, output product generation, and so forth, which is performed in the Vivado IDE echoes an equivalent Tcl command into the `vivado.log` file; consequently, anything that you can do in the Vivado IDE you can script also.

The Vivado IP catalog provides direct access to IP parameter customization from the integrated Vivado IDE Tcl Console so you can set individual IP parameters directly from the Tcl Console.

The following are examples of common IP operations:

Create a customization of the accumulator IP:

- **Tcl Command:**

```
create_ip -name c_accum -vendor xilinx.com -library ip \
-module_name c_accum_0
```

To change customization parameter such as input and output widths:

- **Tcl Command:**

```
set_property -dict [list CONFIG.Input_Width {10}
CONFIG.Output_Width {10}] [get_ips c_accum_0]
```

To generate selective output products:

- **Tcl Command:**

```
generate_target {synthesis instantiation_template simulation} \
[get_ips c_accum_0]
```

To reset any output products generated:

- **Tcl Command:**

```
reset_target all [get_ips c_accum_0]
```

You can use a Tcl script to list the user configuration parameters that are available for an IP.

Use either the `list_property` or `report_property` command and reference the created IP.

The difference between these commands is:

- `list_property` returns a list of objects which can be processed with a Tcl script as a list.
- `report_property` returns a text report giving the current value for each parameter, its type, and other parameters.

To get a list of all properties which apply to an IP:

- **Tcl Command:**

```
list_property [get_ips fifo_generator_0]
```

To get an alphabetized list of just the customization parameters you can augment this further:

- **Tcl Command:**

```
lsearch -all -inline [ list_property [ get_ips fifo_generator_0 ] ] CONFIG.*
```

Create a report listing all the properties for an IP, including the configuration parameters:

- **Tcl Command:**

```
report_property [get_ips fifo_generator_0]
```

For more information on the supported IP Tcl commands type, `help -category IPFlow` in the Tcl Console as shown in [Figure 8-1, page 75](#).

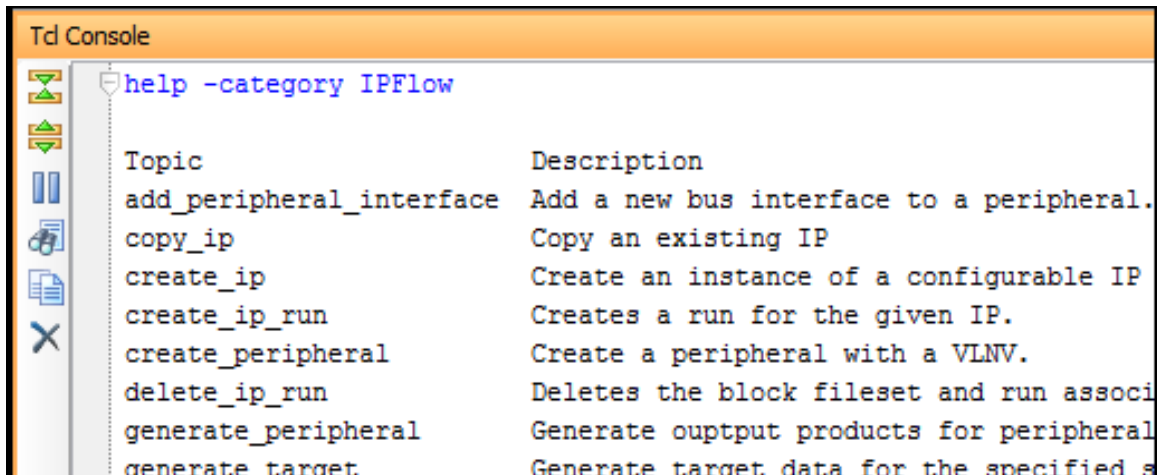


Figure 8-1: Getting Help on IP Tcl Commands

**Note:** The Vivado Design Suite Tutorial: Designing with IP (UG939) [Ref 15] contains labs that cover scripting of both Project Mode and Non-Project Mode flows with IP. They include examples of generating output products as well as selectively upgrading IP.

## Example IP Flow Commands

This section provides Tcl script examples for some common operations.

### Commands to Create IP

The `create_ip` command is used to create IP customizations.



**RECOMMENDED:** Perform this operation as described in [Using the Manage IP Flow, page 55](#). When you create IP with the Manage IP flow, you can subsequently use that IP in Project and Non-Project mode.

The following script shows how to create a manage IP project, create and customize an IP, and generate a DCP:

```
# Create a Manage IP project
create_project <managed_ip_project> ./managed_ip_project -part <part> -ip

# Set the simulator language (Mixed, VHDL, Verilog)
set_property simulator_language Mixed [current_project]

# Target language for instantiation template and wrapper (Verilog, VHDL)
set_property target_language Verilog [current_project]

# Create an IP customization
create_ip -name c_accum -vendor xilinx.com -library ip -module_name c_accum_0

# configure the parameters for the IP customization
```

```
set_property -dict {CONFIG.Input_Width 10 CONFIG.Output_Width 10} [get_ips
c_accum_0]

# Create a synthesis design run for the IP
create_ip_run [get_ips c_accum_0]

# Launch the synthesis run for the IP
# Because this is a project, the output products are generated automatically
launch_run c_accum_0_synth_1
```




---

**IMPORTANT:** *Xilinx recommends project-based flows. Project-based flows can run in either the Vivado IDE or using the Tcl commands.*

---

## Tcl Command to Remove a Design Run

```
delete_ip_run
```

Use this command to remove the design run for an IP. If you reset the output products for an IP in the Vivado IDE, two Tcl commands are issued: `reset_target` and `delete_ip_run`.

## Querying IP Customization Files

### *Tcl Script for Getting Files for Source Control*

This example script shows how to get all files for a given IP customization. You can use this script to generate a list of files for use with a source control system.

```
# Create a project in memory, no project directory
# created on disk
create_project -in_memory -part <part>

# read an IP customization
read_ip <ip_name>.xci

# Generate all the output products
generate_target all [get_ips <ip_name>]

# Create a DCP for the IP
synth_ip [get_ips <ip_name>]

# Query all the files for this IP
get_files -all -of_objects [get_files <ip_name>.xci]
```

**Note:** See [Table 8-1, page 72](#) for a more detailed explanation on these Tcl commands and when to use them.

## Querying an Ordered Source List

When creating custom scripts, you can use one of the following commands:

- **Tcl Command for IP Only: Synthesis**

```
get_files -compile_order sources -used_in synthesis \
-of_objects [get_files <ip_name>.xci]
```

- **Tcl Command for IP Only: Simulation**

```
get_files -compile_order sources -used_in simulation \
-of_objects [get_files <ip_name>.xci]
```

- **Tcl Command for Top-Level Design: Including IP For Synthesis**

```
get_files -compile_order sources -used_in synthesis
```

- **Tcl Command for Top-Level Design: Including IP For Simulation**

```
get_files -compile_order sources -used_in simulation
```

## Scripting Examples

### Implementing an IP Example Design

Create a project to run implementation on an IP example design.

```
# Create a project
create_project <name> <dir> -part <part>

# Create an IP customization and a DCP
# This will also generate all the output products
create_ip ...
create_ip_run [get_ips <ip>.xci]
launch_runs <ip>_synth_1
wait_on_run <ip>_synth_1

# Open the example design for the IP
# This will use the IP DCP generated
open_example_project -force -dir "." -in_process [get_ips <ip>]
launch_runs synth_1
wait_on_run synth_1
launch_runs impl_1
wait_on_run impl_1 -to write_bitstream
open_run impl_1

# produce some reports
report_timing_summary ...
report_utilization ...
```

## Non-Project Synthesis

Synthesize and implement design in a non-project flow with one IP which has an OOC DCP generated and one IP being synthesized along with user logic.

When reading an IP XCI file, all output products present, including an OOC DCP, are used, there is no need to generate them.

If the output products have not been generated for the IP, you must generate the output products (or create a DCP using the `synth_ip` command which generates the output products also).

If you elect to use global synthesis for an IP (see the [Synthesis Options for IP in Chapter 2](#)) then you must disable checkpoint support and generate the output products. This Tcl script provides a template for this.

```
#create an in memory project to provide the part to use for IP creation and for
running synthesis

create_project -in_memory -part <part>

# read in sources
read_verilog top.v

# Read in an existing IP customization
# or create an IP from scratch
# create_ip ... or read_ip ip1.xci

# Generate a DCP for the IP
# will generate output products if needed
synth_ip [get_ips ip1]

# Read in an existing IP customization
# or create an IP from scratch
# create_ip ... or read_ip ip2.xci

# Set IP to use global synthesis (no DCP generated)
set_property generate_synth_checkpoint false [get_files ip2.xci]

# Need to generate output products for IP
generate_target all [get_ips ip2]

# synthesis the complete design
synth_design -top top

# run implementation
opt_design
place_design
route_design

# write the bitstream
write_bitstream -file top
```

## ***Simulating an IP Example Design***

Create a project to run simulation on an IP example design.

```
#create the project
create_project <name> <dir> -part <part>

# create IP and a synthesis run
create_ip ...
create_ip_run [get_ips <ip_name>]

#launch runs
launch_runs <ip>_synth_1
wait_on_run <ip>_synth_1

#open the example project
open_example_project -force -dir "." -in_process [get_ips <ip>]

#launch simulation
<launch_simulation> | <target_simulator>
```

## ***Synthesizing and Simulating an IP***

If an IP does not deliver an example design, but does deliver a test bench, you can perform simulation of just the IP.

```
#create the project
create_project <name> <dir> -part <part>

# create_ip ... or add_files ip.xci

# create an IP design run
create_ip_run [get_ips <ip_name>]

#launch IP synthesis run
launch_run <ip>_synth_1
wait_on_run <ip>_synth_1

# Setting up simulation test bench
set_property top <tb> [current_fileset -simset]

# Launch simulation
<launch_simulation> | <target_simulator>
```

## IP Files and Directory Structure

### Introduction

When customizing an IP using the IP Catalog, either directly in a project or using the Managed IP Flow, the Vivado® Integrated Development Environment (IDE) creates a unique directory for each IP.

After creating an IP customization, a unique directory is made which contains the Xilinx® Core Instance (XCI) file, instantiation template, BOM file, and any generated output products. In this IP directory, there are several additional directories. There is no common structure for the organization of the files that each IP delivers, but there are some common files that are created for each IP.

### IP-Generated Directories and Files

The following table lists the IP-generated target directories and files, which are also known as output products.



**RECOMMENDED:** Xilinx recommends that you use Tcl commands to access the list of related files rather than view the file and directory structure. For example, you can use the `get_files` Tcl commands, which are shown in [Querying IP Customization Files in Chapter 8](#). For more information, see the Vivado Design Suite Tcl Command Reference Guide (UG835) [\[Ref 4\]](#).

Table A-1: IP Output Products

Directory Name, File Name, or File Type	Description
doc	Contains the <code>&lt;Core_Name&gt;_changelog.txt</code> file that provides information about changes to the IP for each release.
sim	Contains the simulation sources files for IP. This directory is not present for all IP.
synth	Contains synthesizable source files for IP. This directory is not present for IP that does not support synthesis, such as simulation-only bus functional model (BFM) IP.



Table A-1: IP Output Products (Cont'd)

Directory Name, File Name, or File Type	Description
<IP_Name>.xci	Contains the IP customization information. You can generate the output products from this file. If an upgrade path exists for the IP in the Catalog, you can upgrade from this file to the latest version.
<IP_Name>.veo vho	Verilog (VEO) or VHDL (VHO) instantiation template. You would use one of these files to instantiate the IP inside your design.
<IP_Name>.xml	IP Bill of Material (BOM) file that keeps track of the current state of the IP, including generated files, computed parameters, and interface information.
<IP_Name>.dcp*	Synthesized Design Checkpoint file contains a post-synthesis netlist and processed XDC constraints. Xilinx recommends that you do not directly reference the IP DCP file; instead use the XCI file, which will bring in the DCP when needed.
<IP_Name>_stub.[v vhdl]*	Module (Verilog) and component (VHDL) for use with third-party synthesis tools to infer a black box for the IP.
<IP_Name>_funcsim.[v vhdl]*	Post synthesis structural simulation netlist files
<IP_Name>.xdc	Timing and/or physical constraints. These files are not present for all IP, and their location varies by IP.
<IP_Name>_in_context.xdc	See <a href="#">Determining Clocking Constraints and Interpreting Clocking Messages, page 44</a> for more information.
dont_buffer.xdc	Deprecated file. Functionality is included in <IP_Name>_in_context.xdc
<IP_Name>_clocks.xdc	Constraints with a clock dependency. These files are not present for all IP, and their location varies by IP.
<IP_Name>_board.xdc	Constraints used in a board flow. These files are not present for all IP, and their location varies by IP.
<IP_Name>_ooc.xdc	Default clock definitions used when synthesizing the IP out-of-context.
Encrypted HDL for the IP	Files used for synthesizing and simulating the IP. These files are not present for all IP, and their location varies by IP.

\* The DCP, \_stub, and \_funcsim files are created only when using the Out-of-Context flow for synthesis (default). See [Synthesis Options for IP in Chapter 2](#) for more details.

**Note:** Although example design are not output products, they are commonly generated for IP. The example design files are only available when the example design is opened with one of the following:

- open\_example\_project Tcl command
- Vivado IDE with the **Open IP Example Design** menu command.

For more information, see [Chapter 4, Using IP Example Designs](#).

## Using the IP Board Flow

### Using the IP Board Flow

The IP Board Flow feature is supported by some IP and gives you the ability to select board interfaces while customizing an IP. When you use this feature, the creation of physical constraints for the IP is automated by delivering additional XDC constraints in a special file.

As shown in the following figure, when creating a new project, you can select a board as the default part.

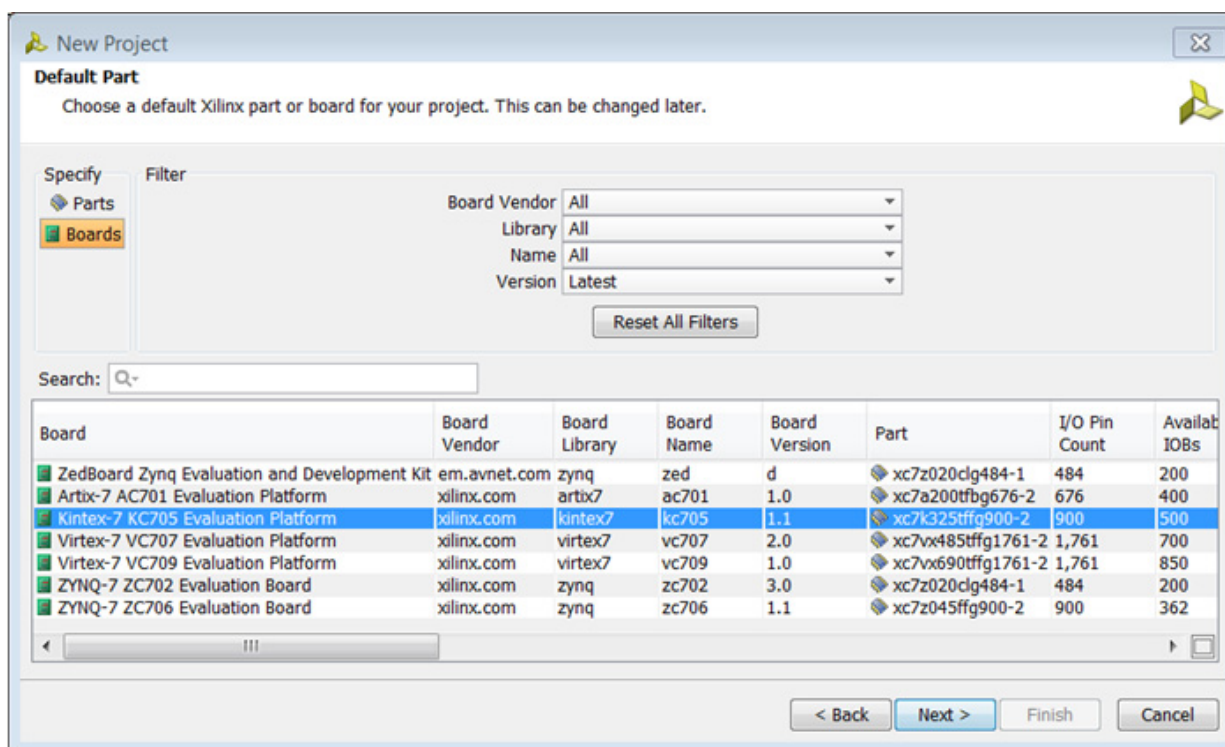


Figure B-1: Selecting a Board as the Default Part

Selecting one of the listed boards results in IP that supports the board flow to have a new tab visible during customization as shown in the following figure.

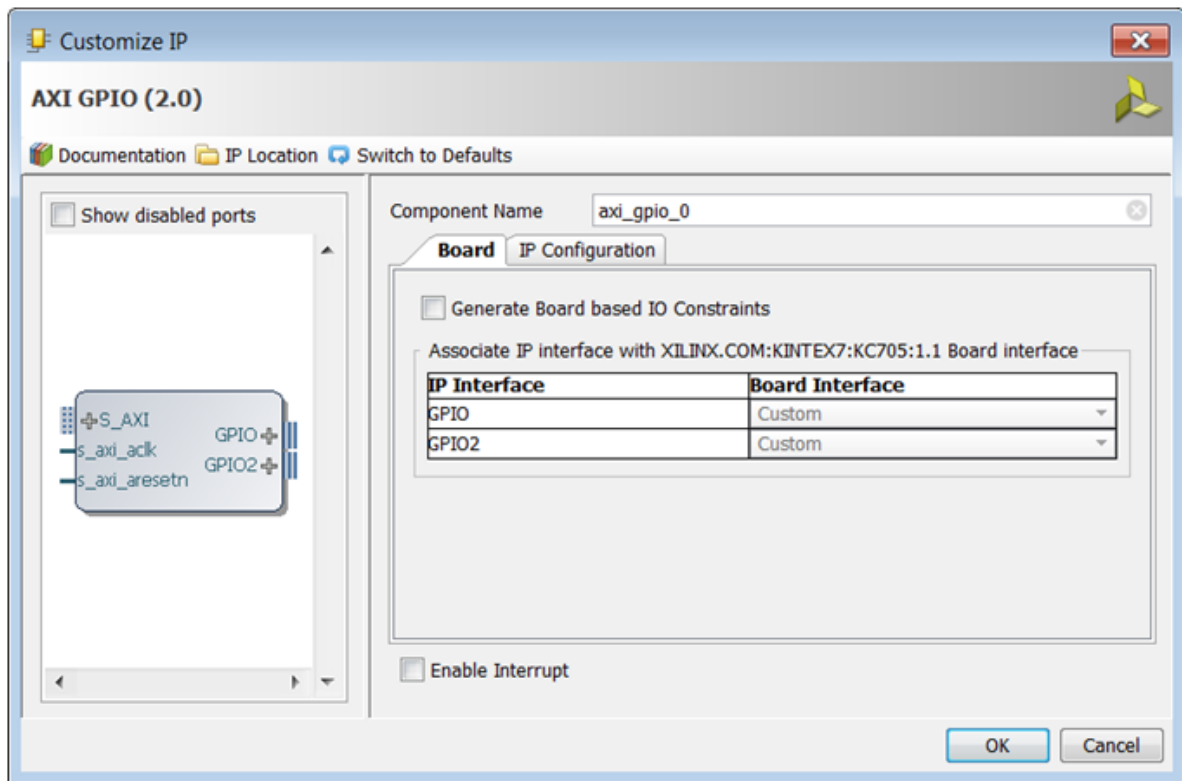


Figure B-2: Board Type Visible in Supported IP Customization

The following figure shows that when you check the **Generate Board based IO Constraints** check box, you can associate the IP interface to the available board interface.

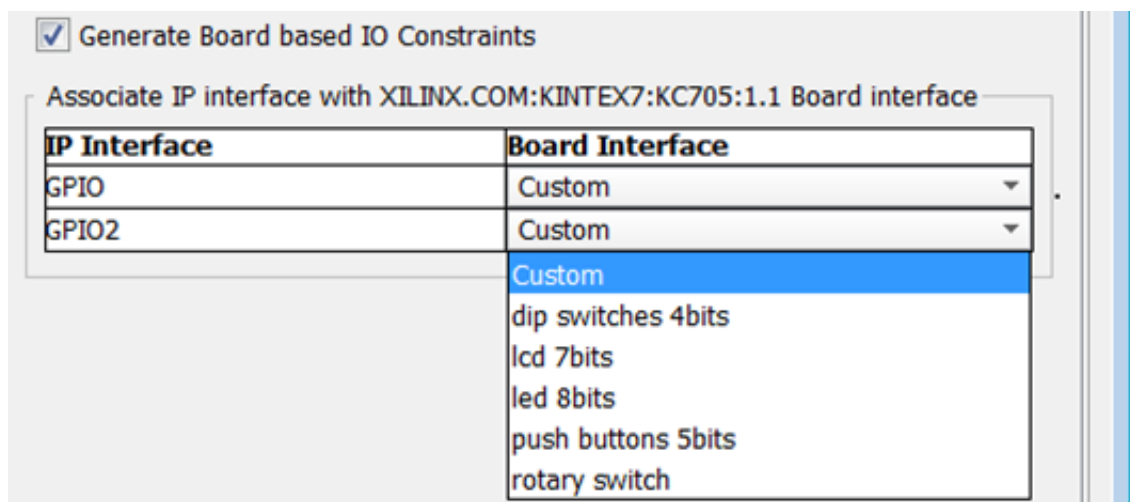


Figure B-3: Associating the IP Interface with the Board Interface

When the Vivado IDE generates the IP output products in the IP Sources view, you see the <IP\_Name>\_board.xdc file listed.

This file contains physical constraints assigning ports of the IP to the package pins that connect to the related board connector or device such as a USB port, LED, button, or switch.

The following figure shows the XDC constraints created for the GPIO IP when you connect the GPIO interface to the board LCD interface and the connect the GPIO2 interface to the board push buttons.

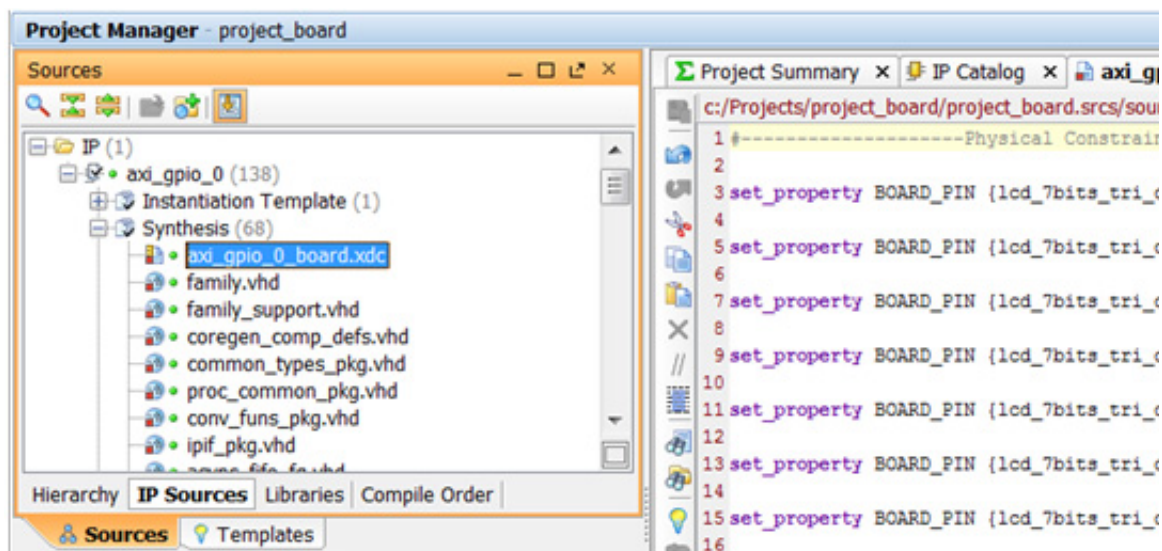


Figure B-4: IP Board XCD File

## IP that Supports Board Flow

The IP that support the board flow include the following:

- axi\_emc\_v3\_0
- axi\_ethernet\_buffer\_v2\_0
- axi\_ethernetlite\_v3\_0
- axi\_ethernet\_v6\_2
- axi\_gpio\_v2\_0
- axi\_iic\_v2\_0
- axi\_pcie3\_v1\_0
- axi\_quad\_spi\_v3\_2
- axi\_uart16550\_v2\_0
- axi\_uartlite\_v2\_0
- clk\_wiz\_v5\_1

- gig\_ethernet\_pcs\_pma\_v14\_3
- iomodule\_v3\_0
- microblaze\_mcs\_v2\_2
- mig\_7series\_v2\_2
- mii\_to\_rmii\_v2\_0
- proc\_sys\_reset\_v5\_0
- tri\_mode\_ethernet\_mac\_v8\_3

When generating IP for use in the Vivado IP integrator, you must be familiar with the parameters and the valid values for key bus interfaces. These interfaces include:

- AXI4-Lite
- AXI4 memory mapped protocol
- AXI4-Stream
- clock
- interrupt
- reset

See the *Vivado AXI Reference Guide* (UG1037) [Ref 21], and the IP-specific product guides for more information on interface naming conventions.

# Editing IP

---

## Introduction

At times, you might need to edit unencrypted source files that an IP delivers, including XDC files, HDL files, or ports of an IP. This should only be done if absolutely necessary.



---

**RECOMMENDED:** *If you determine you must modify any of the IP sources, follow the guidelines provided in this appendix and do not directly modify the sources on disk. Directly making modifications can result in your changes being removed because the IP could become reset or regenerated during the flow.*

---

Xilinx® IP delivered in the IP Catalog have an `IS_MANAGED` property that affects the ability to edit the IP in the Vivado IDE. IP is one of the following:

- `IS_MANAGED` property is user-changeable.
- `IS_MANAGED` property is read-only. This cannot be changed by the user.

The IP that have the `IS_MANAGED` property as read-only are subsystem IP and are more complex.

One example is the UltraScale® Ten Gigabit Ethernet PCS/PMA IP that contains dynamic Transceiver Wizard IP references. The following section provides information on editing IP that is not a subsystem IP.



---

**IMPORTANT:** *It is important that you mark the IP as being under user management or locking it before making edits. Making the IP user managed also indicates that the IP was locked by a user and that changes to the IP have been made. Revisit the changes before upgrading the IP at a future time because the changes are lost when you put the IP back into tool control. It is important that you correctly setup the IP and generate the output products prior to marking it for editing. After the IP is under user management, you can no longer customize the IP or generate output products.*

---

The following sections apply to both a Managed IP project and are applicable for an RTL project with IP present, either stored within the project directory structure or saved to an external location.

## Modifying IP Sources

To prepare an IP for editing:

1. If you have not customized the IP, do so, and generate all output products, including the DCP.

If you do not want to use the default OOC flow for the IP you must disable the DCP creation by going to the out-of-context settings.



**RECOMMENDED:** *Xilinx highly recommends that you use the default flow.*



2. After you generate the output products (including the DCP, if applicable) are generated, set the `IS_MANAGED` property to `false` on the XCI file for the IP using the following Tcl command:

```
set_property IS_MANAGED false [get_files <IP_NAME>.xci]
```

If it is a complex subsystem IP, the following error message displays:

```
ERROR: [IP_Flow 19-3666] The is_managed property cannot be directly modified for hierarchical IP.
```

3. Upon receipt of this error, read the [Editing Subsystem IP, page 88](#), and follow those steps.

Setting the `IS_MANAGED` property to `false` causes the property `IS_LOCKED` to become `TRUE`. The IP icon in the IP Sources window changes from  to , showing the IP is not managed by Vivado and is instead user-managed.

In the output window of the **Report IP Status** command you see that the IP is under user management, and you can modify non-encrypted HDL files and XDC files.

4. Complete the required edits.
5. Recreate the DCP using the modified files as follows:
  - a. In the Design Runs tab, right-click the IP and select **Launch Runs**.

A dialog box opens and asks you if you want to launch the selected run.

- b. Click **OK**.

Another dialog box opens and informs you that the run must be reset and all the files will be deleted for the run.

**Note:** This is not referring to the IP source files, only the run-related files and output.

After the run is complete, you can use the IP as before.




---

**RECOMMENDED:** By referencing the XCI file (which is recommended) you have access to the IP source files for simulation, and the DCP for synthesis of the top-level file as well as for implementation.

---

## Editing Subsystem IP

Some complex subsystem IP do not allow changes to the `IS_MANAGED` property. This condition is applicable for IP from 7 series® and UltraScale™ FPGA families. Whether or not the subsystem IP allows the `IS_MANAGED` property to be changed depends on particular customization options of the specific IP.





---

**CAUTION!** Editing the RTL files of such IP has risks. It is possible to make a change that invalidates the connectivity to the sub-cores. Making changes to these IP HDL sources should be carefully considered.

---

1. If edits are required for these IP, manually lock the IP before making edits, using the following Tcl command:

```
set_property IS_LOCKED TRUE [get_files ten_gig_eth_pcs_pma_0.xci]
```

After you lock the IP you see in the IP Source tab that the icon now has a lock by the IP symbol .

Locking the IP prevents the Vivado tools from modifying the IP output products or resetting the IP. It also provides a visual queue that the IP has been user-modified.

Unlike with IP marked as being user-managed using the `IS_MANAGED` property, locking the IP does not enabling editing the IP when using the Vivado text editor.

2. You must either:
  - Change to another editor using **Tools > Options > General** in the text editor section.
  - Edit the files directly on disk using your text editor of choice.
3. After you modify the files, use [step 5, page 87](#) from [Modifying IP Sources, page 87](#) to recreate the DCP and re-launch a design run.



# Additional Resources and Legal Notices

---

## Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see [Xilinx Support](#).

## Solution Centers

See the [Xilinx Solution Centers](#) for support on devices, software tools, and intellectual property at all stages of the design cycle. Topics include design assistance, advisories, and troubleshooting tips.

See the Xilinx [MIG Solution Center](#) for information regarding the Memory Interface Generator.

## Vivado Design Suite Documentation

The following documents are cited within this guide:

1. *Vivado Design Suite User Guide: Creating and Packaging Custom IP* ([UG1118](#))
2. *Vivado Design Suite Tutorial: Creating and Packaging Custom IP* ([UG1119](#))
3. *Vivado Design Suite User Guide: Designing IP Subsystems Using IP Integrator* ([UG994](#))
4. *Vivado Design Suite Tcl Command Reference Guide* ([UG835](#))
5. *Vivado Design Suite User Guide: Using Constraints* ([UG903](#))
6. *Vivado Design Suite Properties Reference Guide* ([UG912](#))
7. *Vivado Design Suite Tutorial: Programming and Debugging* ([UG936](#))
8. *Vivado Design Suite User Guide: Programming and Debugging* ([UG908](#))
9. *UltraFast Design Methodology Guide for the Vivado Design Suite* ([UG949](#))
10. *Vivado Design Suite User Guide: Getting Started* ([UG910](#))
11. *Vivado Design Suite User Guide: Release Notes, Installation, and Licensing* ([UG973](#))
12. *Vivado Design Suite User Guide: Design Flows Overview* ([UG892](#))

13. *Vivado Design Suite User Guide: Hierarchical Design* ([UG905](#))
14. *Vivado Design Suite User Guide: Logic Simulation* ([UG900](#))
15. *Vivado Design Suite User Guide: System-Level Design Entry* ([UG895](#))
16. *Vivado Design Suite Tutorial: Designing with IP Tutorial* ([UG939](#))
17. *Vivado Design Suite Tutorial: Logic Simulation* ([UG937](#))
18. *Vivado Design Suite Tutorial: Design Flows Overview* ([UG888](#))
19. *Vivado Design Suite User Guide: Using Tcl Scripting* ([UG894](#))
20. *Vivado Design Suite User Guide: Using the Vivado IDE* ([UG893](#))
21. *Vivado AXI Reference Guide* ([UG1037](#))
22. *Vivado Design Suite User Guide: I/O and Clock Planning* ([UG899](#))
23. [Vivado Design Suite QuickTake Video Tutorials](#)
24. [Vivado Design Suite Documentation](#)

## Xilinx IP Documentation

25. *LogiCORE IP Integrated Logic Analyzer Product Guide* ([PG172](#))
26. *LogiCORE IP IBERT for 7 Series GTX Transceivers* ([PG132](#))
27. *LogiCORE IP IBERT for 7 Series GTP Transceivers* ([PG133](#))
28. *LogiCORE IP IBERT for 7 Series GTH Transceivers* ([PG152](#))
29. *LogiCORE IP Virtual Input/Output Product Guide* ([PG 159](#))
30. *Zynq-7000 SoC and 7 Series FPGAs Memory Interface Solutions* ([UG586](#))

---

## Standards and Third-Party Documentation

1. IP-XACT (IEEE Std 1685), *Standard Structure for Packaging, Integrating, and Reusing IP within Tool Flows*

---

## Training Resources

Xilinx provides a variety of training courses and QuickTake videos to help you learn more about the concepts presented in this document. Use these links to explore related training resources:

1. [Essentials of FPGA Design](#)
  2. [Embedded Systems Software Design](#)
- 

## 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>.

AMBA, AMBA Designer, ARM, ARM1176JZ-S, CoreSight, Cortex, and PrimeCell are trademarks of ARM in the EU and other countries.

MATLAB and Simulink are registered trademarks of The MathWorks, Inc.

PCI, PCI Express, PCIe, and PCI-X are trademarks of PCI-SIG.

© Copyright 2012-2014 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, UltraScale, 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.