AMD

together we advance_

AMD ADAPTIVE COMPUTING CUSTOMER TRAINING

Lab Reference Guide

lab-reference-guide-2024.1-wkb-rev1-prelim

Lab Reference Guide 2024.1

AMD
together we advance_

Table of Contents

Lab Reference Guide

Introduction

This *Lab Reference Guide* is a collection of *how to* topics for commonly performed tasks in AMD device and embedded development.

The guide is divided into the following sections:

- Vivado Design Suite Operations

- Vivado Analyzer Operations

- Vitis Model Composer Operations

- Vitis Unified IDE Operations

- Vitis Unified IDE HLS Component Operations

- QEMU Operations

- PetaLinux Operations

- Board, OS, COM, and IP Address Tasks

Each section may contain sub-sections.

Understanding the Lab Environment

The labs and demos provided in this course are designed to run on a Linux® platform.

One environment variable is required: `TRAINING_PATH`, which points to the location of the lab files. This variable comes configured in the CloudShare/CustEd_VM environments.

Some tools can use this environment variable directly (that is, `$TRAINING_PATH` is recognized and automatically expanded), and some tools require manual expansion (`/home/amd/training` for the CloudShare/CustEd_VM environments). The lab instructions describe what to do for each tool. Other environments require the definition of this variable for the scripts to work properly.

The Vivado™ Design Suite and the Vitis™ Unified IDE offer a Tcl environment used in many labs. When either tool is launched, it starts with a clean Tcl environment with none of the procs or variables remaining from any previous launch of the tools.

If you sourced a Tcl script or manually set any Tcl variables and you closed the tool, when you reopen the tool, you will need to re-source the Tcl script and set any variables that the lab requires. This is also true of terminal windows—any variable settings will be cleared when a new terminal opens.

Nomenclature

Formal nomenclature is used to explain how different arguments are used. The following are some of the more commonly used symbols:

Symbol	Description	Example	Explanation
<text**>**	Indicates a field	`cd <dir>`	`<dir>` represents the name of the directory. The < and > symbols are NOT entered. If the directory to change to is `XYZ`, then you would enter **cd XYZ** into the environment.
[text**]**	Indicates an optional argument	`ls [	more]`	This could be interpreted as `ls <Enter>` or `ls	more <Enter>`. The first instance lists the files in the current Linux directory, and the second lists the files in the current Linux directory, but additionally runs the output through the `more` tool, which paginates the output. Here, the pipe symbol (`|`) is a Linux operator.
**	**	Indicates choices	`cmd <ZCU104	VCK190>`	The `cmd` command takes a single argument, which could be `ZCU104` OR `VCK190`. You would enter either `cmd ZCU104` or `cmd VCK190`.

Vivado Design Suite Operations

In This Section

AMD
together we advance_

Creating and Opening Operations

In This Section

Launching the Vivado Design Suite

Here are two ways to open the Vivado Design Suite.

1-1.   Open the Vivado Design Suite.

1-1-1. Click the **Vivado** icon () from the taskbar.
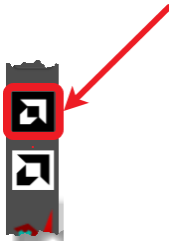
Figure 1: Launching the Vivado Design Suite from the Taskbar

Note: It takes a few moments to launch. The order of the icons in your environment may be different.

Alternatively, open the Linux terminal window (<**Ctrl + Alt + T**>) and enter the following:

```
source /opt/amd/Vivado/2024.1/settings64.sh; vivado
```

Note: This installation path is valid for the CustEd VM and CloudShare environments. Use the proper path for your environment.

The tool opens with a Welcome window. From here you can create a new project, open an existing project, enter Tcl commands, and access documentation and examples.

1-1-2. [Optional] Maximize the window as there is a lot of information to see.
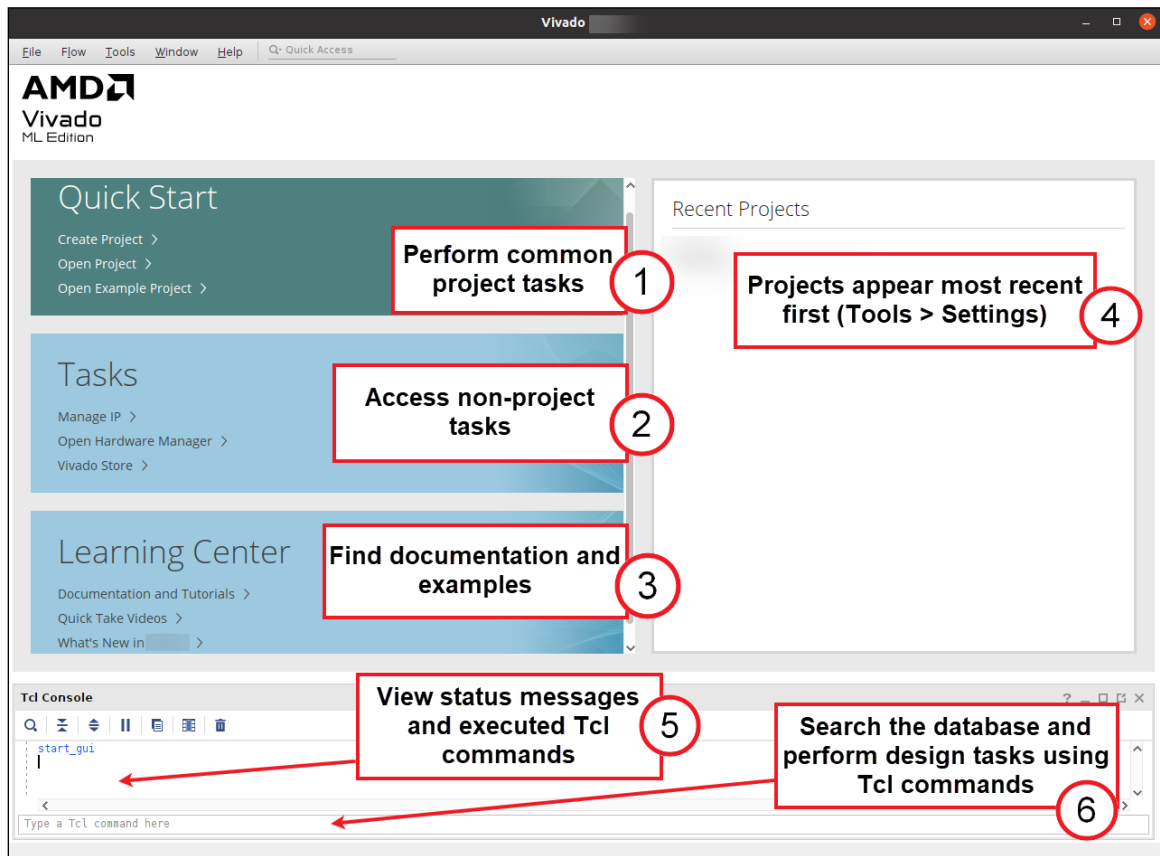
Figure 2: Vivado Design Suite Welcome Screen

Hint: If the Tcl Console is not visible, double-click the **Tcl** tab to make it visible.

Launching the Vivado Design Suite Using a Linux Terminal

1-1. **[Linux users]: If your system has not been configured with a shortcut on the desktop or taskbar, then proceed with setting up and launching the Vivado Design Suite environment.**

1-1-1. Press <**Ctrl** + **Alt** + **T**> to open a new terminal window.

1-1-2. Enter the following command to set up the Vivado Design Suite environment appropriately:

```
source /opt/amd/Vivado/2024.1/settings64.sh; vivado
```

Note: This installation path is valid for the CustEd VM and CloudShare environments. Use the proper path for your environment.

AMD
together we advance_

Launching the Vivado Design Suite Tcl Shell

1-1.   Launch the Vivado Design Suite Tcl shell.

1-1-1. Open a Linux terminal window (press <**Ctrl** + **Alt** + **T**>) and enter the following commands:

[host]$ source /opt/amd/Vivado/2022.2/settings64.sh

[host]$ vivado -mode tcl

Note: The installation path (`/opt/amd`) is valid for the CustEd VM and CloudShare environments. Modify the path as necessary for your environment.

This opens the Tcl interactive shell in the terminal itself.
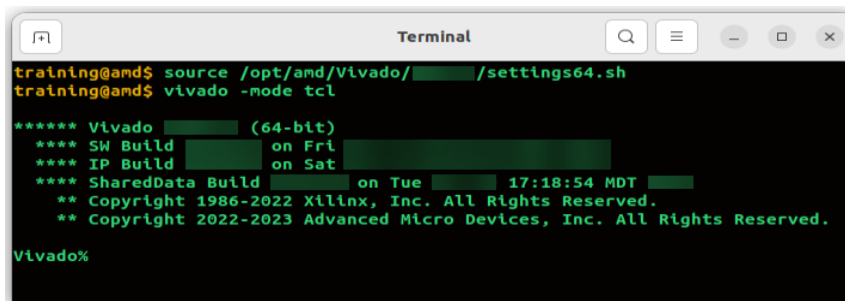
Figure 3: Vivado Tcl Interactive Shell in Terminal

Creating a New Vivado Design Suite Project with Sources

Projects begin with the creation of a new project. The project contains sources, settings, graphics, IP, and other elements that are used to build a final bitstream.

1-1.   Create a new Vivado Design Suite project.

1-1-1. Click **Create New Project** (1).
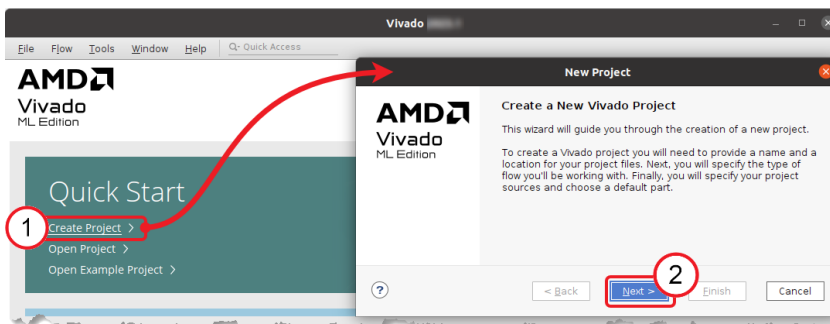
Figure 4: Creating a New Vivado Design Suite Project

This will launch the New Project Wizard.

1-1-2. Click **Next** to begin entering the specifics for this project (2).

1-2. You will now encounter a series of dialog boxes asking you to enter different pieces of information describing the project.

1-2-1. Enter **your project name** in the Project name field (1).

1-2-2. Enter the following location in the Project location field (2):

```
project
```

Important: The Vivado Design Suite is capable of expanding variables when running under both the Linux and Windows environments. The available environment variables (regardless of the OS) must be predicated with the '$' symbol.

Alternatively, you can use the browse feature to navigate to where you want the project to reside.

1-2-3. Uncheck the **Create Project Subdirectory** option if it is selected (3).

Leaving this checked will create an unnecessary level of hierarchy for the lab.
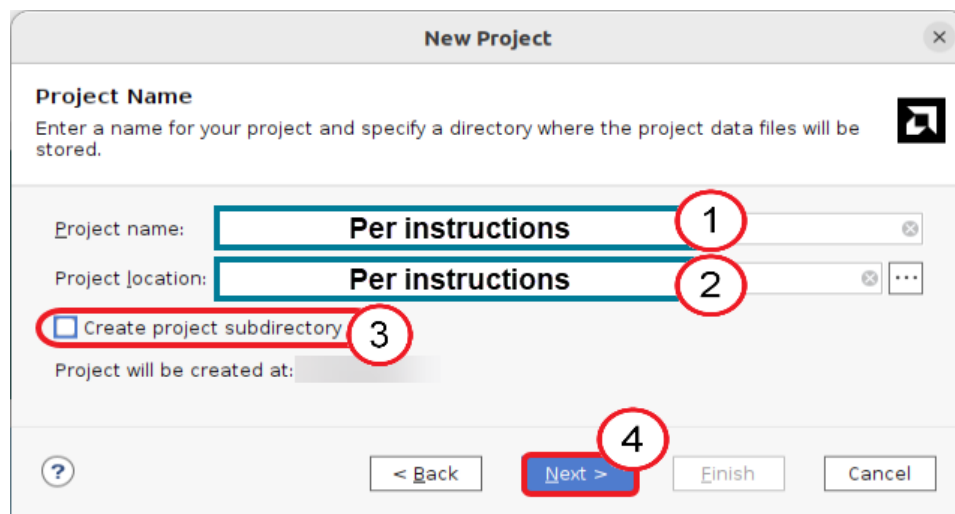
Figure 5: Entering the Project Name and Location

1-2-4. Click **Next** to advance to the next dialog box (4).

Here you will specify your project type as either an RTL project or a post-synthesis project. An RTL project enables you to add or create new HDL files and synthesize them, whereas the post-synthesis project requires pre-synthesized files. When an empty design is created, an RTL project is used.

AMD
together we advance_

1-2-5. Select **RTL Project** since this project will be based on source code (rather than a netlist) (1).

1-2-6. Since you will be adding RTL files in the next instruction, deselect the **Do not specify sources at this time** option (2).
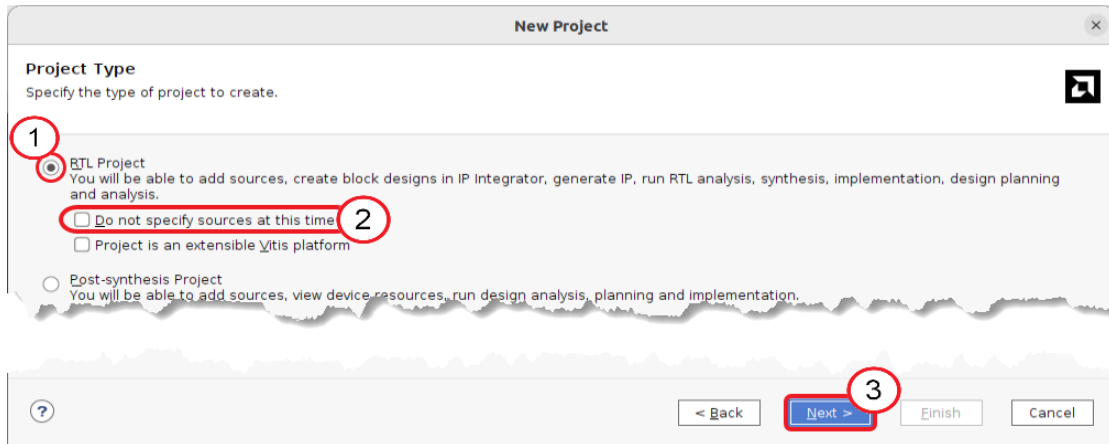
Figure 6: Setting the Project Type to RTL

1-2-7. Click **Next** to accept the selection and advance to the adding sources stage (3).

1-3. **Now that the project name, type, and location have been entered, the wizard invites you to add source files to the project. You can add entire directories where the sources are located, add specific files, or create new sources.**

Begin by adding the sources to your project.

1-3-1. Click the **Plus** icon (➕) to begin adding objects to the project.

1-3-2. Select **Add Files** from the pop-up menu to begin adding your source files to the project.
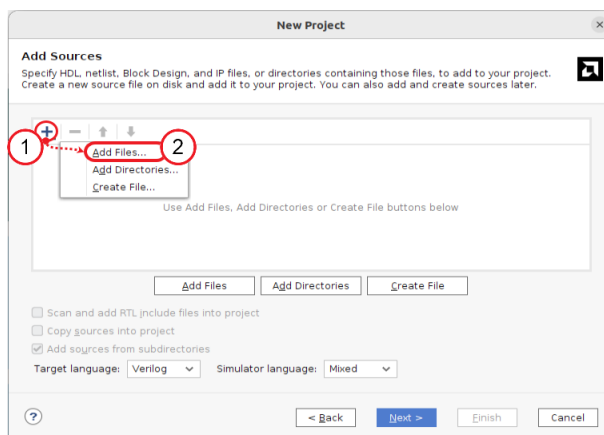
Figure 7: Adding Sources to the Project

After you add all the necessary files, the remainder of this dialog box will be addressed.

The Add Source Files dialog box opens.

AMD
together we advance_

1-3-3. Browse to the **project** directory (1).
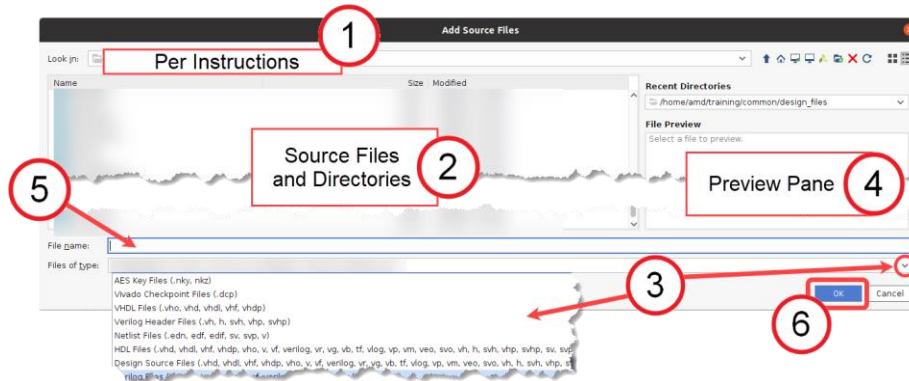
Figure 8: Adding Source Files

1-3-4. Multi-select **your HDL sources** from the list of source files and directories (2).

Note: There are equivalent Verilog and VHDL sources. Here you can choose either. Verilog files have the extension `.v`, whereas VHDL files end with `.vhd`.

Hint: To use multi-select, press <**Ctrl**> and select each file.

Hint: Since this directory contains both VHDL and Verilog sources, you can use the File Type drop-down list and select VHDL files or Verilog files to filter what files are presented (3).

If you select a file, its contents are shown in the preview pane (4).

As you select files, they appear in the File Name field (5).

1-3-5. Click **OK** to accept the selected files and add them as sources to the project (6).

The Add Source Files dialog box closes and brings you back to the Add Sources dialog box.

AMD
together we advance_

1-3-6. Ensure that the **Scan and Add RTL Include Files into Project** and **Copy Sources into Project** options are selected (1).

1-3-7. Select the target language using the drop-down list (2).

1-3-8. Click **Next** (3).
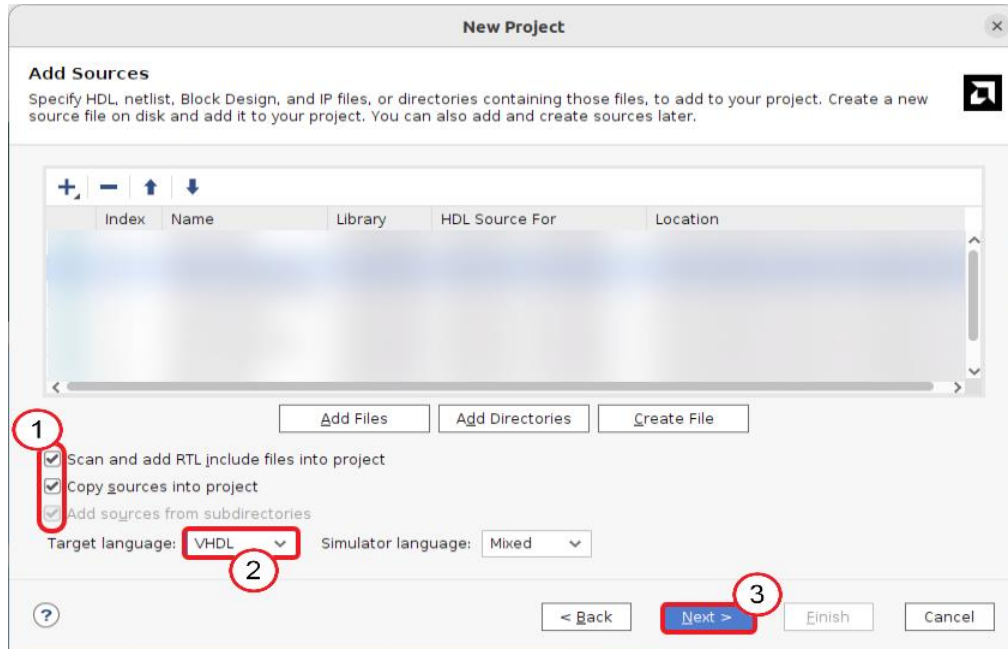
Figure 9: Setting Options in the Add Sources Window

1-4. Add any existing IP modules to the Vivado Design Suite project.

If you have existing IP modules, you will add them here.

1-4-1. Click the **Plus** icon (➕) again to select the IP files.

1-4-2. Select **Add Files** to open the Add Source Files dialog box.

Since all Add Source Files dialog boxes are the same, you can refer to the previous Add Source Files dialog box illustration.

1-4-3. Select or multi-select **your IP modules**.

1-4-4. Click **OK** to accept the selected files and add them as sources to the project.

If you have additional files located in other directories, you can repeat this instruction for each directory.

1-4-5. Confirm that the **Scan and Add RTL Include Files into Project** option is selected (used for Verilog, no effect for VHDL) in the Add Sources dialog box.

This will automatically pull in any include files used by Verilog sources.

1-4-6. Confirm that the **Copy Sources into Project** option is selected.

This will make a local copy of the source in the project space. Selecting this option enables you to make changes to the local copy without affecting the original source file.

1-4-7. Select **your preferred language** from the Target Language drop-down list.

This choice only affects which language is used for the generation of templates and wrappers. You can add files in any language regardless of which target language is selected. If you do not generate or use any templates or wrappers, this step is irrelevant, and you can select any language.

1-4-8. Click **Next** to complete the adding of RTL sources and advance to adding any constraint files.

1-5. **Add any existing constraint files to the Vivado Design Suite project.**

If you have existing constraints, you will add them here.

1-5-1. Click the **Plus** icon (➕) to select the type of object you want to import (1).

1-5-2. Select **Add Files** to open the Add Source Files dialog box.

The Add Source Files dialog box opens.

Since all Add Source Files dialog boxes are the same, you can refer to the previous Add Source Files dialog box illustration.

1-5-3. Browse to the **project** directory.
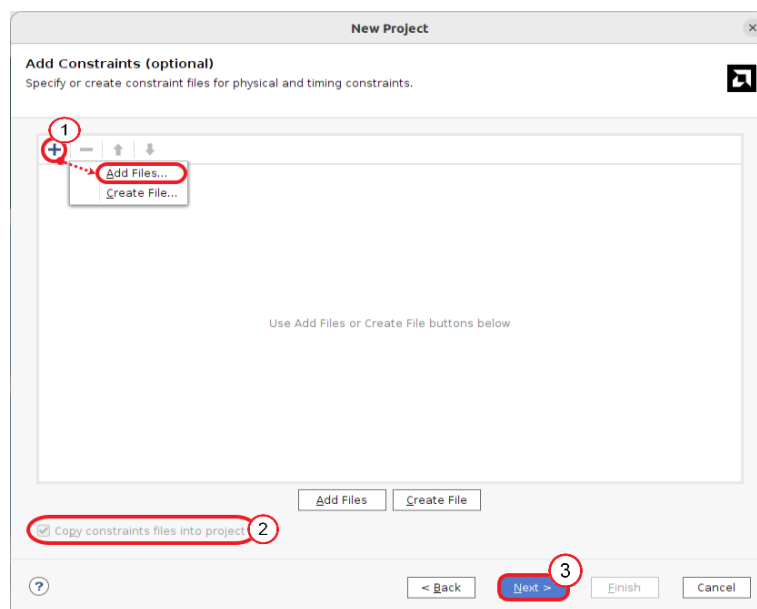
Figure 10: Adding Constraints

1-5-4. Select or multi-select **your constraints file**.

1-5-5. Click **OK** to accept the selected files and add them as sources to the project.

If you have additional files located in other directories, you can repeat this instruction for each directory.

1-5-6. Confirm that the **Copy constraints files into Project** option is selected (2).

This will make a local copy of the source in the project space. Selecting this option enables you to make changes to the local copy without affecting the original source file.

1-5-7. Click **Next** to advance to selecting a target device (3).

AMD
together we advance_

1-**6.** **Select the target part by first filtering by board and then by family. If you are not using a supported board, you will need to filter by part.**

1-6-1. Select **Boards** from the Select area (1).

1-6-2. Select **the vendor of your board** from the Vendor drop-down list in the Filter area (2).

This filters the available boards to those that are populated with any member of the selected library.

1-6-3. Select the **board** board from the list.

Alternatively, you can select the board directly from the list at any time while in this dialog box.
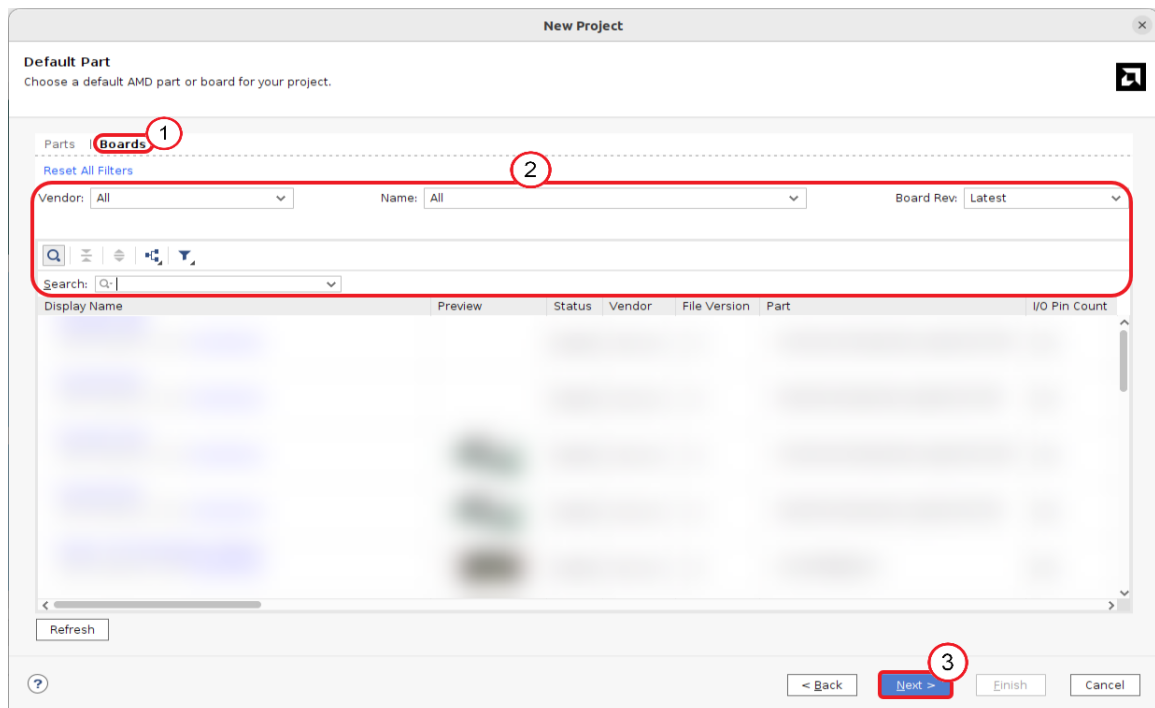
Figure 11: Selecting the Board for the Project

1-6-4. Click **Next** to advance to the summary (3).

A summary of your project is displayed. If you want to change any of the information that you entered, you can do so now by clicking **Back** until you reach the correct dialog box and making the correction, or you can create the project now and edit the project properties, add or remove files, etc. later.

1-6-5. Click **Finish**.

Your project is constructed, and you are presented with the Vivado Design Suite main workspace environment.

AMD
together we advance_
www.amd.com
13

Creating a Blank Vivado Design Suite Project

"Create Project" is the starting point for all designs. Projects contain sources, settings, graphics, IP, and other elements that are used to build a final bitstream and analyze a design. The Create New Project Wizard in the Vivado Design Suite allows you to specify HDL and other project resource files that will be included in the project.
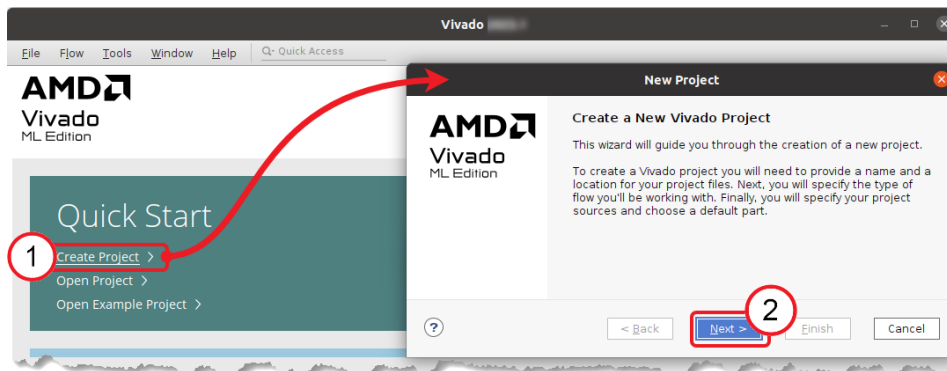
1-1.   Create a new, blank Vivado Design Suite project.

1-1-1. Click **Create Project** to begin the process (1).

Figure 12: Creating a New Vivado Design Suite Project

This will launch the New Project Wizard.

1-1-2. Click **Next** to exit the introductory dialog box and begin entering project-specific information (2).

1-2.   Describe the various aspects of the project.

1-2-1. Enter **your project name** in the Project name field (1).

1-2-2. Enter the following location in the Project location field (2):

Enter project location here

Important: The Vivado Design Suite is capable of expanding variables when running under both the Linux and Windows environments. The available environment variables (regardless of the OS) must be predicated with the '$' symbol.

Alternatively, you can use the browse feature to navigate to where you want the project to reside.

1-2-3. Uncheck the **Create Project Subdirectory** option if it is selected (3).

Leaving this checked will create an unnecessary level of hierarchy for the lab.
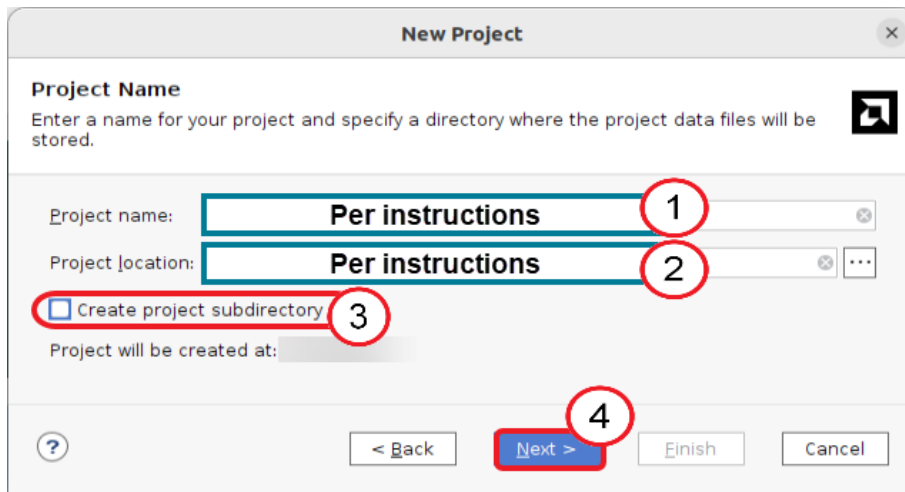
Figure 13: Entering the Project Name and Location

1-2-4. Click **Next** to advance to the next dialog box (4).

Here you will specify your project type as either an RTL project or a post-synthesis project. An RTL project enables you to add or create new HDL files and synthesize them, whereas the post-synthesis project requires pre-synthesized files. When an empty design is created, an RTL project is used.

1-2-5. Select **RTL Project** (1).

1-2-6. Select **Do not specify sources at this time** to instruct the Vivado tool to create a blank project (2).

While existing sources could be entered at this time, you will enter them later so that you can move through this portion of the project creation process more quickly.
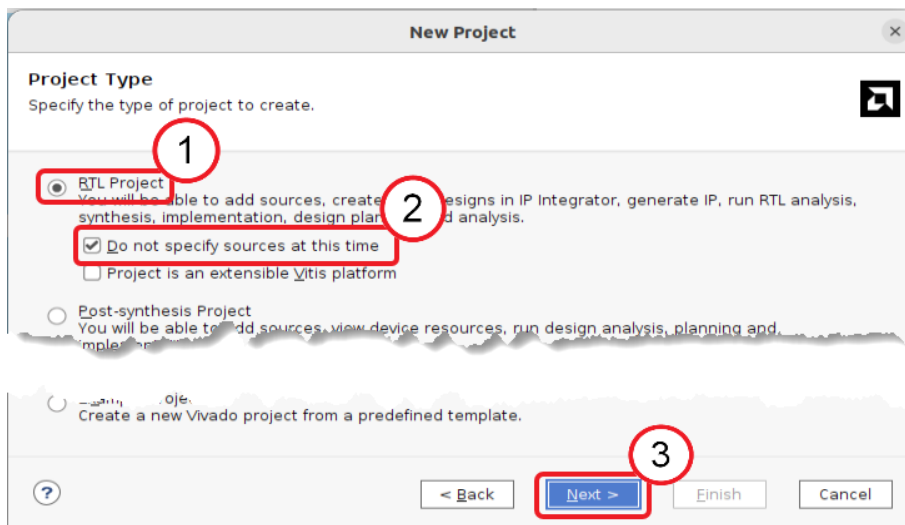
Figure 14: Specifying Project Options

1-2-7. Click **Next** to advance to the target device/platform selection (3).

1-3. **Select the target part by first filtering by the board name. If you are not targeting a supported board, you will need to filter by the part.**

1-3-1. Click **Boards** from the *Default part* area to filter by the board type rather than by the specific part (1).

1-3-2. Select **the vendor of your board** from the Vendor drop-down list in the Filter area (2).

This limits the number of boards seen to those manufactured by the specified vendor.

1-3-3. Select the **board** board from the list.

If you accidentally click the hyperlink, a web page will open for that board. You can close the browser page.

If the board you want to use is not immediately visible, click the **Refresh** button to update the board catalog.

Note: While the web page contains important information and resources for the board, these details are not needed to complete this lab.
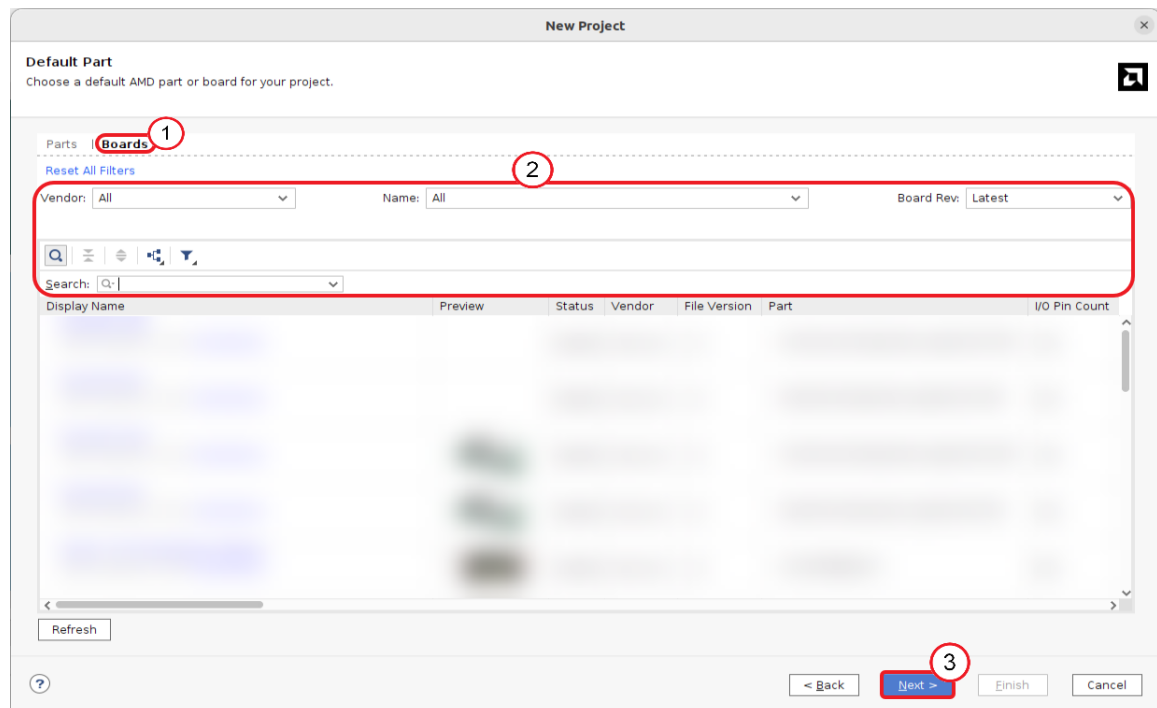
Figure 15: Selecting the Board for the Project

1-3-4. Click **Next** to advance to the summary (3).

A summary of your project is displayed. If you want to change any of the information that you entered, you can do so now by clicking **Back** until you reach the correct dialog box. After the project is created, the project properties can still be edited.

1-3-5. Click **Finish** to accept these settings and build the project.

Your project is constructed and leaves you in the operational portion of the Vivado Design Suite GUI.

AMD
together we advance_

Opening a Vivado Design Suite Project

1-1. **Open the existing Vivado Design Suite project** *your project name.*

1-1-1. Click **Open Project** from the Quick Start section (1).

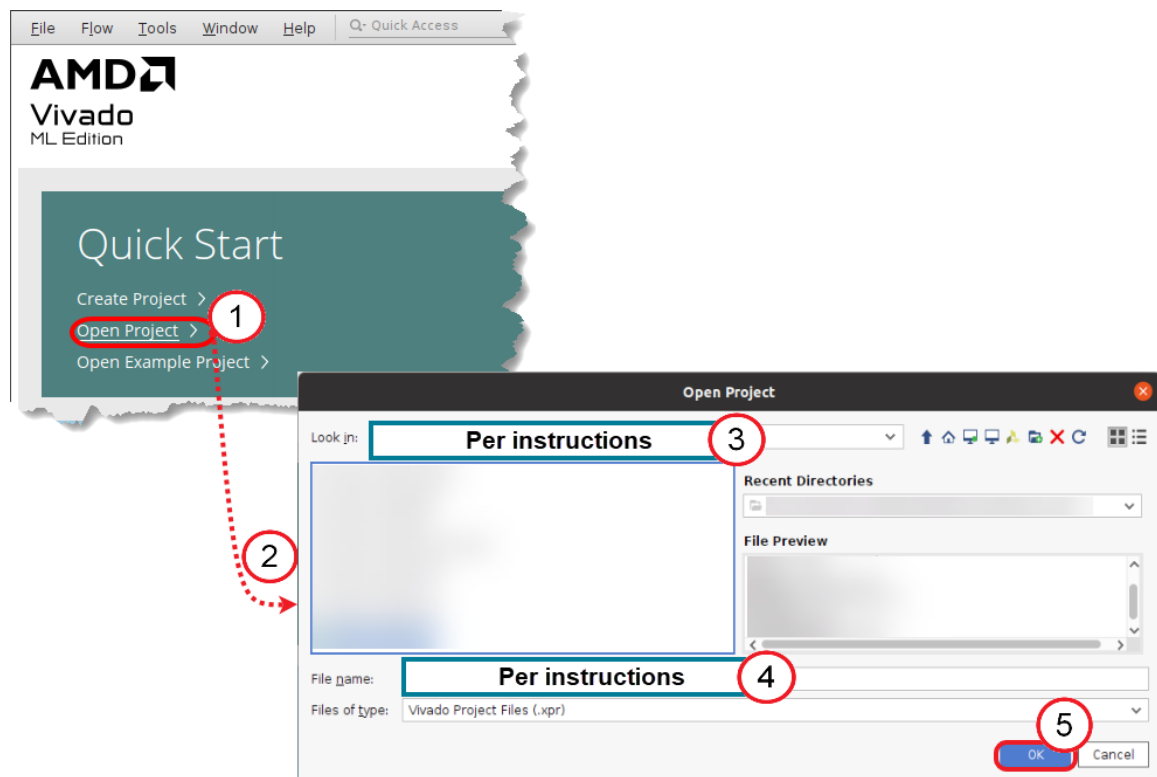The Open Project dialog box opens (2).

1-1-2. Browse to the following directory in the Look in field (3):

```
project
```

Note: The drop-down arrow shows the directory hierarchy.

1-1-3. Select **your project name** from the files displayed (4).

Figure 16: Opening an Existing Project

1-1-4. Click **OK** to open the selected project (5).

The project now opens in the Vivado Design Suite.

Opening Source Code in the Editor

1-1. Open *the desired source file* in the editor.

1-1-1. Under the Sources tab, locate **the desired source file**.

> **Note:** This process is valid for any of the sub-tabs under Sources. Typically the Hierarchy and Libraries sub-tabs are the most commonly used as these organize the user files in the most convenient fashion.
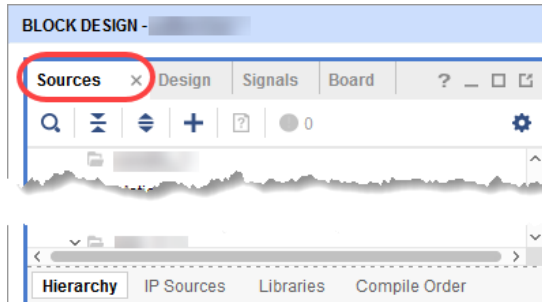
Figure 17: Sources Tab

1-1-2. Double-click **the desired source file** to open it in the editor.

> **Note:** You can also also right-click and select **Open File** to open the file in the editor.

Creating an HDL Source File

1-1. Create a new HDL source file called *your HDL sources*.

1-1-1. Select **Add Sources** in the Flow Navigator under Project Manager.
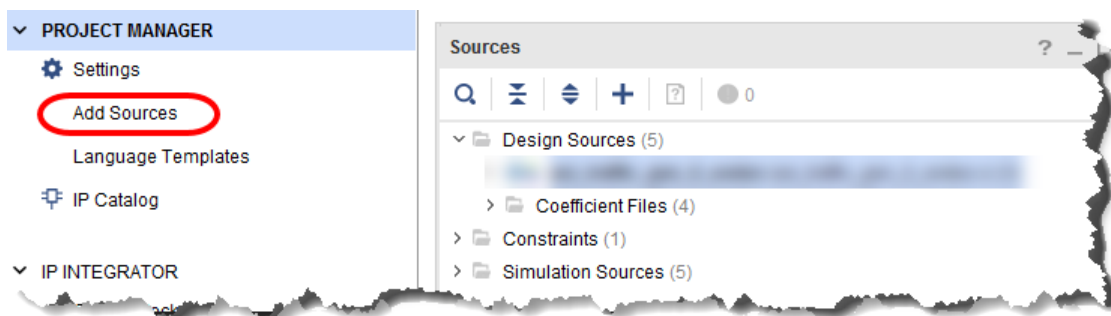
Figure 18: Selecting Add Sources

AMD
together we advance_

1-1-2. Select **Add or create design sources** (1).
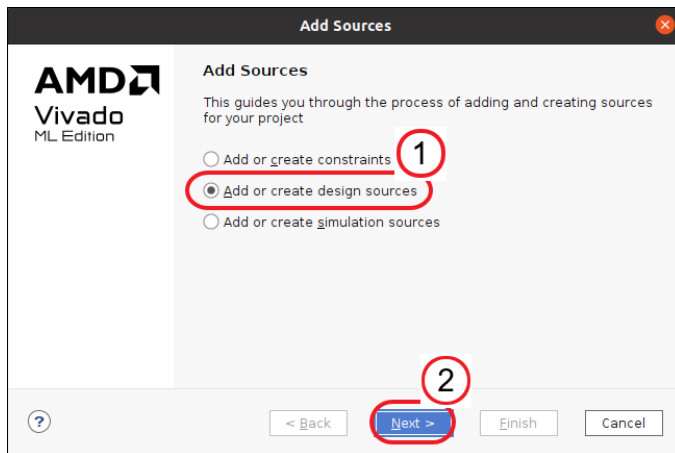
Figure 19: Selecting Add or Create Design Sources

1-1-3. Click **Next** (2).

The Add or Create Design Sources dialog box opens.

1-1-4. Click the **Plus** (+) icon and select **Create File**.
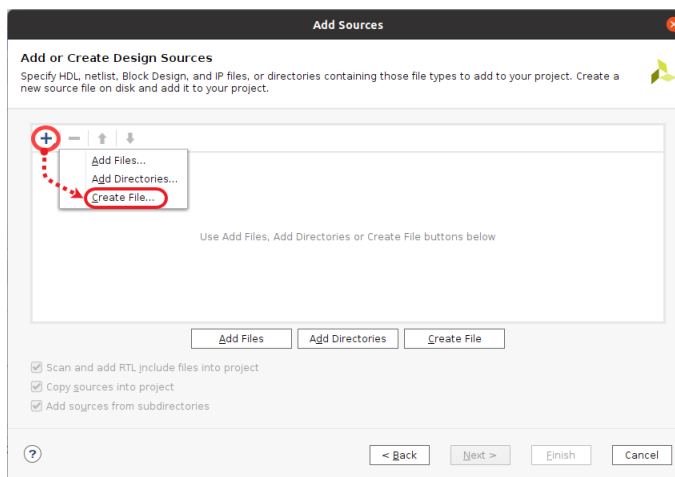
Figure 20: Selecting Create File

The Create Source File dialog box opens.

1-1-5. Select **your preferred language** from the File type drop-down list.

1-1-6. Enter **your HDL sources** as the file name.
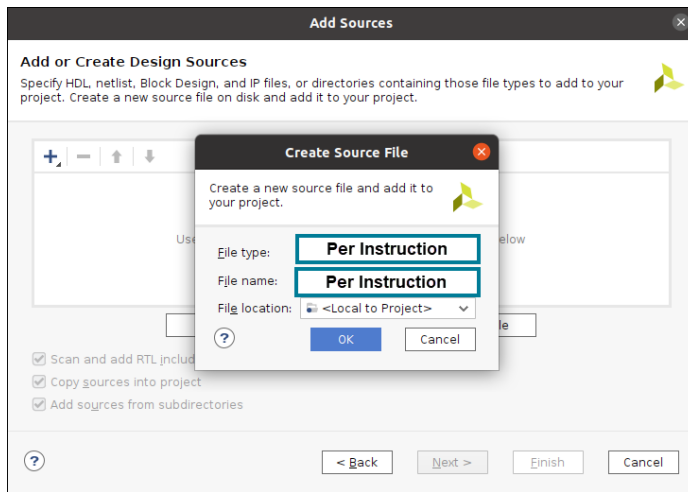
Figure 21: Entering File Name and Type

1-1-7. Click **OK** in the Create Source File dialog box.

1-1-8. Click **Finish** to add the new source file(s).

The Define Module dialog box opens.

Opening a File in the Editor

This process is used to look at any type of text-based file. It is suggested that if the file you want to open is part of the project, then you merely double-click the filename when in the Hierarchical view or the Library view.

1-1. **Open `your file` in the built-in editor.**

1-1-1. Select **File** > **Text Editor** > **Open File**.
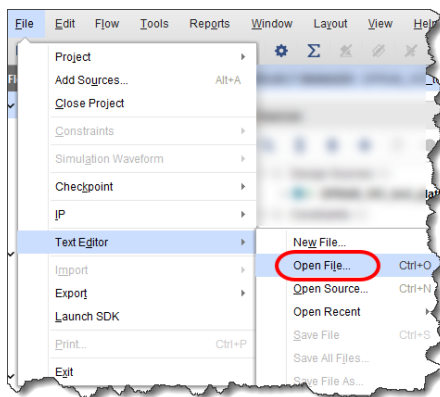
Figure 22: Opening a Text File from the File Menu

A file browser window opens.

AMD
together we advance_

1-1-2. Navigate to `the location of the text file to open`.

1-1-3. Select **your file**.

1-1-4. Click **OK**.

The text file opens in the workspace area.

Importing IP

IP can be imported from a number of tools provided that they adhere to the IP-XACT standard. Once created, the Vivado Design Suite must be made aware of the presence of the IP. This is typically performed from an open project.

The instructions below describe the process of importing a single IP; however, the steps can be repeated as necessary to collect different IPs from various locations.

1-1. **Import your IP cores into the open project.**

1-1-1. Using the Flow Navigator, select **Project Manager** > **Settings** (1).

The Project Settings Dialog box opens with the General tab open.

1-1-2. Expand **IP** (2) and click **Repository** to access the IP settings (3).
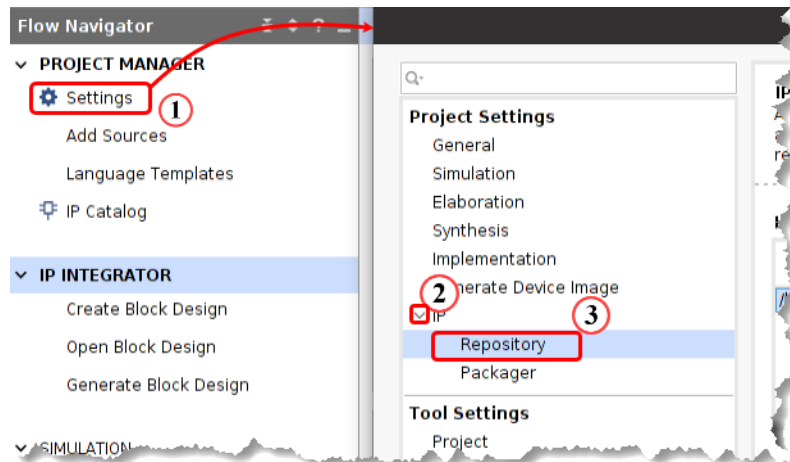
Figure 23: Accessing the Project's IP Settings

1-1-3. Click the **Plus** icon (➕) to open the file browser to point to the IP repository in which your IP is located (1).

1-1-4. Browse to *where your IP is located* (2).

1-1-5. Click **Select** (3).

> **IMPORTANT:** If the IP repository that you have selected has the IP in zip format, then you will need to continue as described below. If, however, your IP has been expanded (for example, if you are revisiting this step or adding another piece of IP), then it will automatically appear in the IP in Selected Repository field. You can simply click **OK** to complete the repository inclusion process.

> The selected IP repository is scanned for all IP that is listed in the Select IP To Add To Repository dialog box.
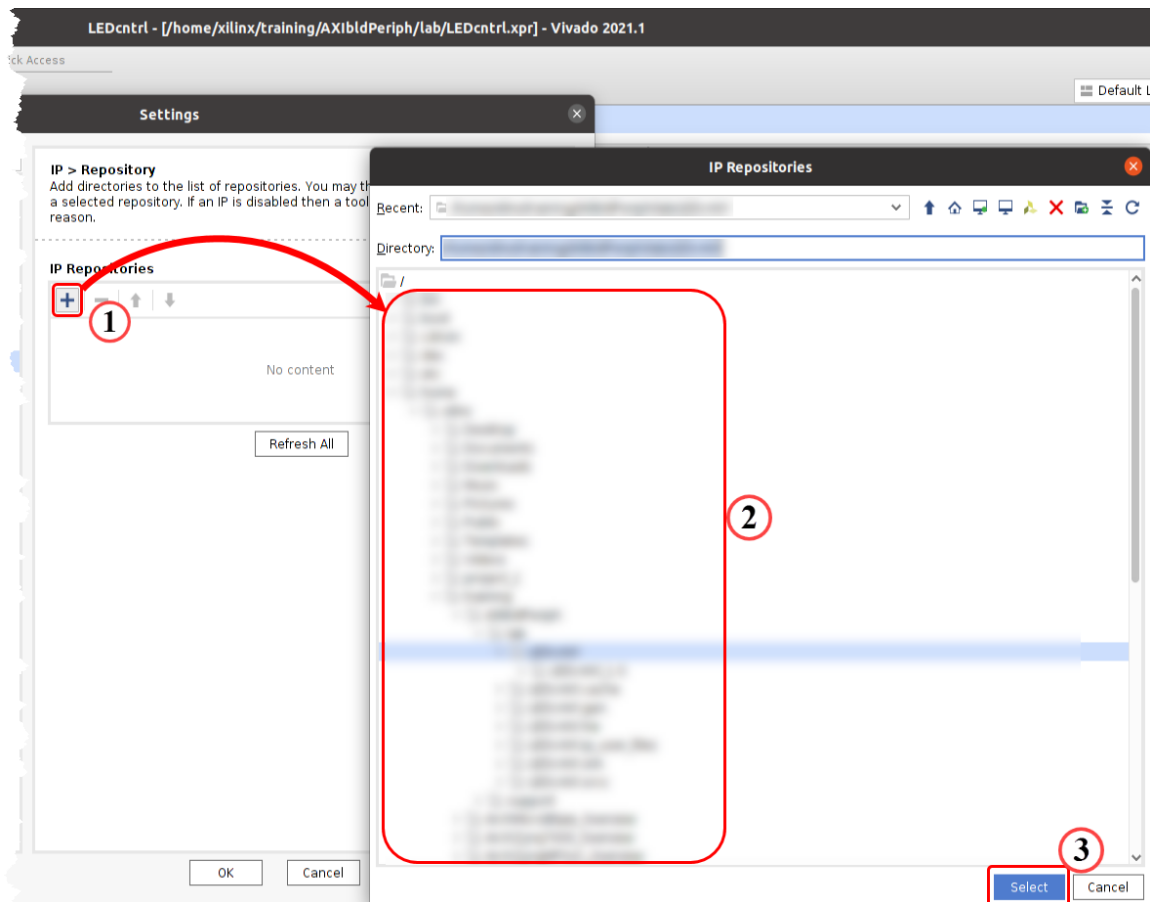
Figure 24: Adding the IP Repositories and the Specific IP

A window opens, showing the number of IPs being added to the project.

1-1-6. Click **OK** to close the Add Repository dialog box.

1-1-7. Click **OK** again to exit the project settings.

AMD
together we advance_

Closing the Vivado Design Suite Project

1-1. Close the project.

1-1-1. Select **File** > **Project** > **Close** to close the project.

The Close Project dialog box opens.
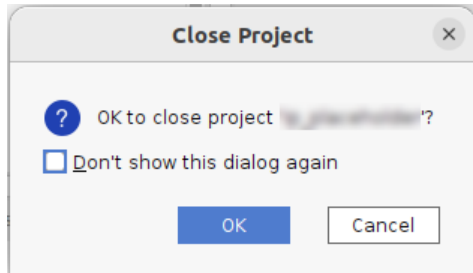
Figure 25: Close Project Dialog Box

1-1-2. Click **OK**.

Closing the Vivado Design Suite

1-1. Close the Vivado Design Suite.

1-1-1. Select **File** > **Exit**.

The Exit Vivado dialog box opens.
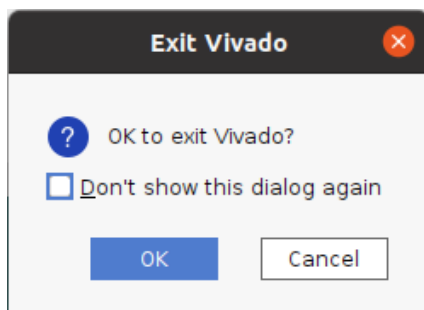
Figure 26: Exit Vivado Dialog Box

1-1-2. If you are asked to save the project or a portion of the project, select whichever elements of the project you want to save, then click **Save** to save the selected elements; otherwise, click **Don't Save**.

1-1-3. Click **OK** when you are asked to exit the Vivado Design Suite.

Note: You can choose to select the *Don't show this dialog again* option to avoid being asked for confirmation when exiting the Vivado Design Suite.

Vivado Add Sources Operations

In This Section

Adding an HDL Source File

HDL source files can be added to the design at any time.

1-1. Add an HDL source file to the design.

1-1-1. Select **Add Sources** under the Flow Navigator tab in the Project Manager.
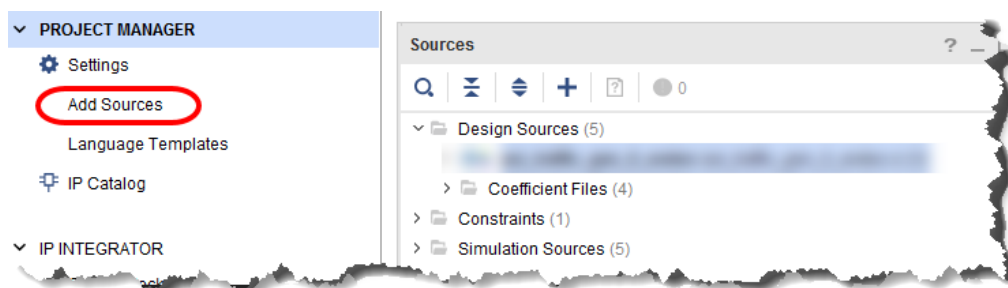
Figure 27: Selecting Add Sources

The Add Sources dialog box opens, allowing you to add HDL source files to the project.

1-1-2. Select **Add or create design sources** (1).
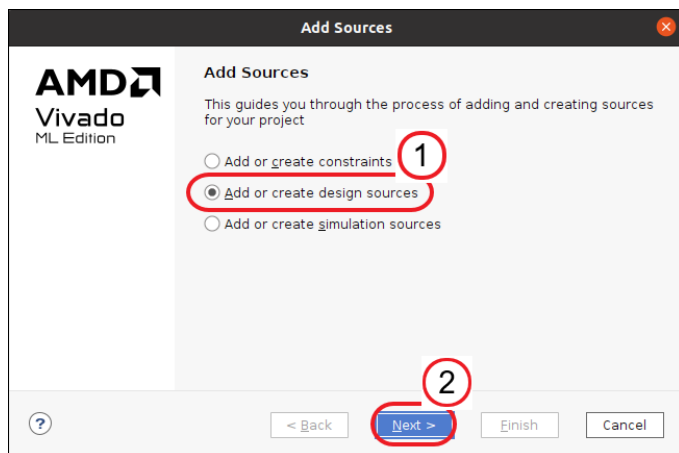
Figure 28: Selecting Add or Create Design Sources

1-1-3. Click **Next** to begin selecting source files (2).

AMD
together we advance_

The Add or Create Design Sources dialog box opens and prompts you to add existing HDL source files or to create new HDL sources files.

1-1-4. Click the **Plus** (✚) icon to open the context menu (1).

1-1-5. Select **Add Files** to begin adding the source files to the project (2).

1-1-6. Browse to the following directory if it is not open already:

`your source directory`

1-1-7. Select **your HDL sources**.

1-1-8. Double-click the source file name in the Add Source Files dialog box to select the file(s) or click **OK** (3).

1-1-9. Ensure that the **Copy sources into project** option is selected (when building IP this will be listed as **Copy sources into IP Directory**) (4).
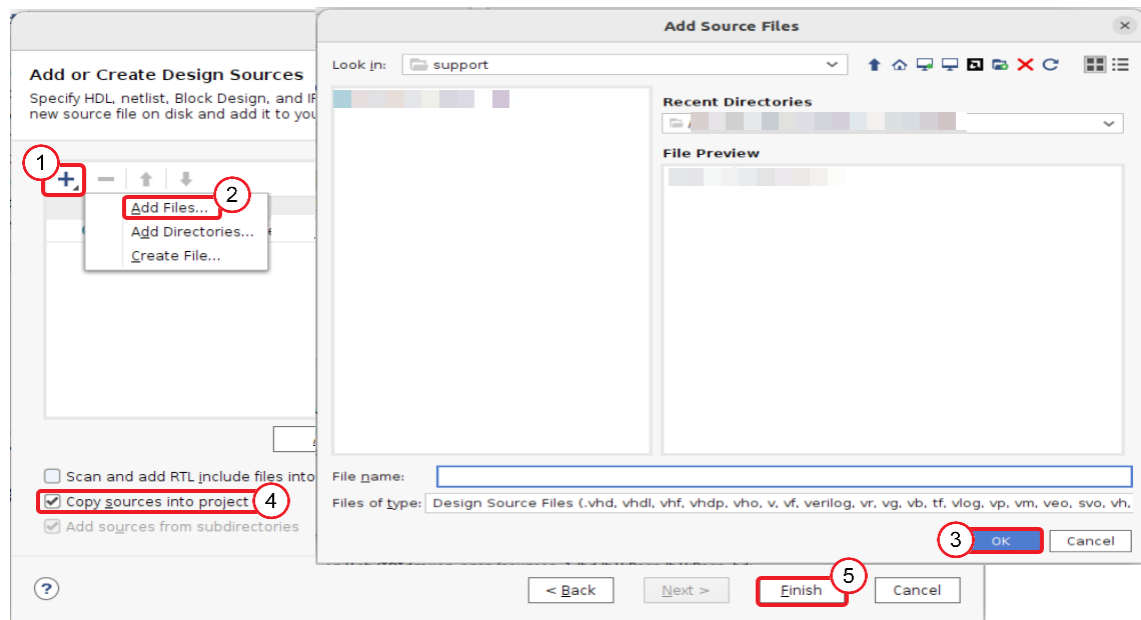
Figure 29: Adding Files

1-1-10. Click **Finish** in the Add or Create Design Sources dialog box to add the HDL sources to the project (5).

Adding a Simulation Source File

HDL simulation files can be added to the design at any time.

1-1.   Add simulation files to the design.

1-1-1. Select **Add Sources** under Project Manager in the Flow Navigator.
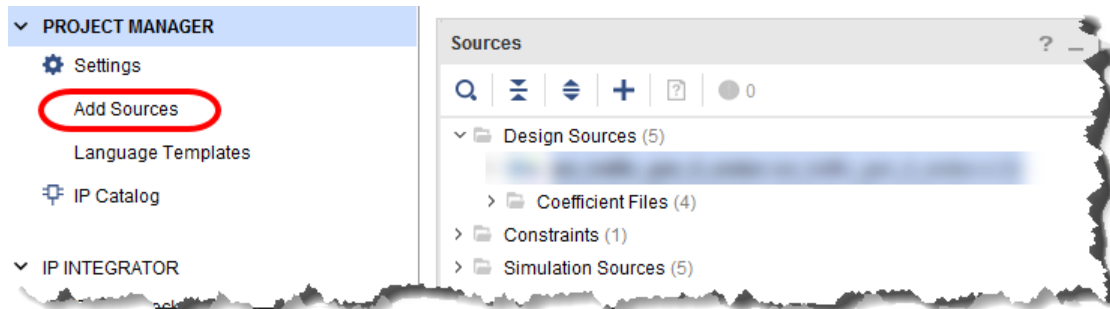
Figure 30: Selecting Add Sources

1-1-2. Select **Add or create simulation sources**.
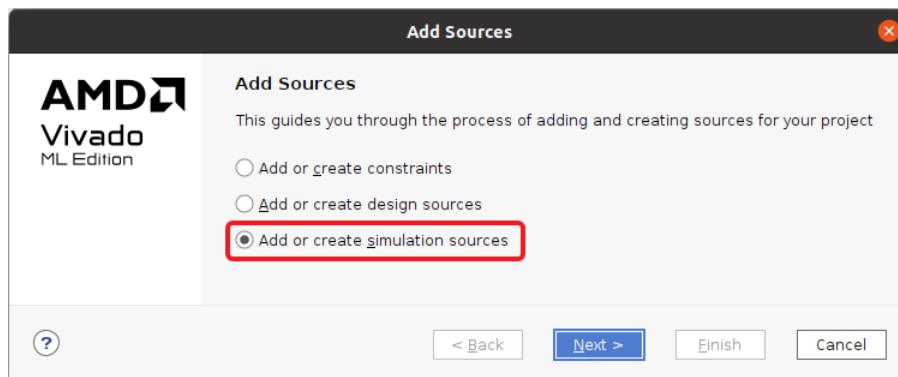
Figure 31: Add Sources Dialog Box

1-1-3. Click **Next**.

1-1-4. Click the **Plus** (✚) icon to open the pop-up menu.

1-1-5. Select **Add Files** to open the Add Source Files dialog box which allows you to browse to the desired directory.

1-1-6. Browse to the following directory if it is not open already:

    $TRAINING_PATH/<the topic cluster name>/support

1-1-7. Select **your HDL sources**.

1-1-8. Click **OK** in the Add Source Files dialog box.

1-1-9. Ensure that the **Copy sources into project** option is selected.

1-1-10. Click **Finish** to add the file(s) to the project.

AMD
together we advance_

Adding a Constraint File

Constraint files can be added to the design at any time. Here's how to add an existing constraint file.

1-1. **Add the constraint file to the Vivado Design Suite project.**

1-1-1. From the Flow Navigator, expand **Project Manager** (1).

1-1-2. Click **Add Sources** to open the wizard (2).

1-1-3. Select **Add or create constraints** (3).
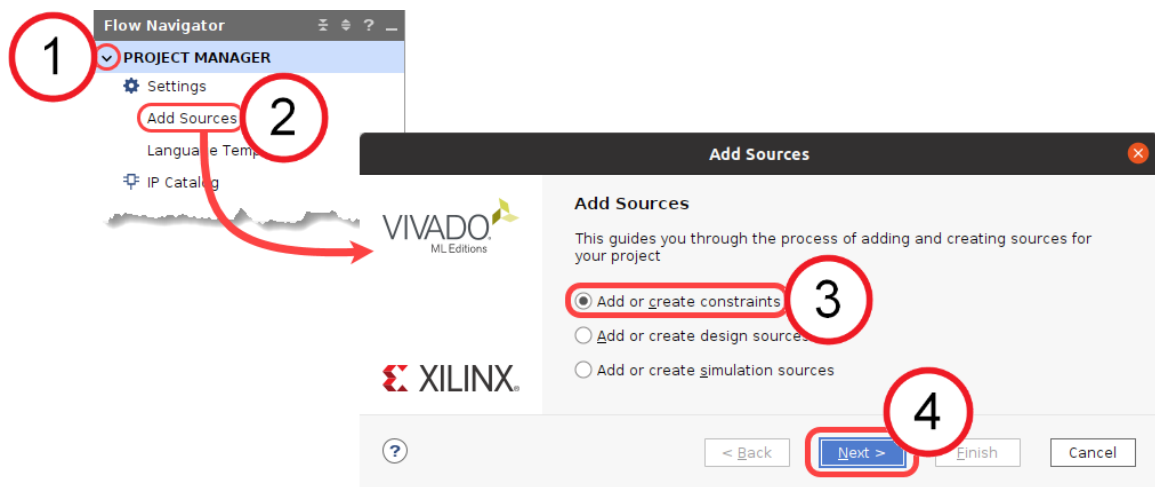
Figure 32: Accessing the Add Source for Constraints

1-1-4. Click **Next** to advance to the next screen (4).

1-1-5. Click the **Plus** icon (+) to select whether you want to import or create a constraint file (1).
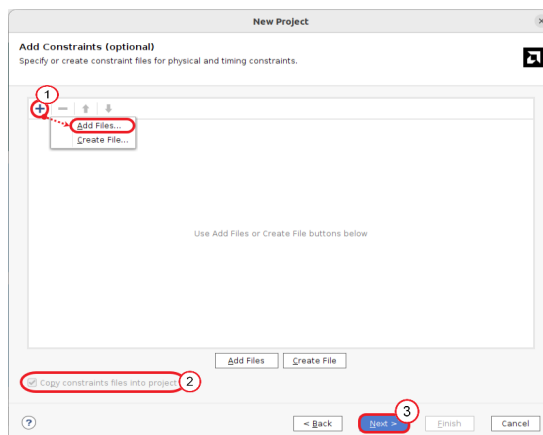
Figure 33: Adding Constraints

1-1-6. Click **Add Files** to open the Add Constraint Files dialog box.

The Add Constraint Files dialog box opens.

1-1-7. Browse to the `location of your files` directory.

> **[KCU105 users]:** Select **your constraints file**.
>
> **[VCK190 users]:** Select **VCK190_uart_led_Project_Flow.xdc**.

1-1-8. Click **OK** to accept the selected files and add them as sources to the project.

> If you have additional files located in other directories, you can repeat this instruction for each directory.

1-1-9. Confirm that the **Copy constraints files into project** option is selected (2).

> This will make a local copy of the source in the project space. Selecting this option enables you to make changes to the local copy without affecting the original source file.

1-1-10. Click **Finish** in the Add or Create Design Sources dialog box to add the sources to the project.

Creating a SystemVerilog Simulation Source File

1-1. **Create a new SystemVerilog file called** *your HDL sources.*

1-1-1. Select **Add Sources** in the Flow Navigator, under Project Manager.
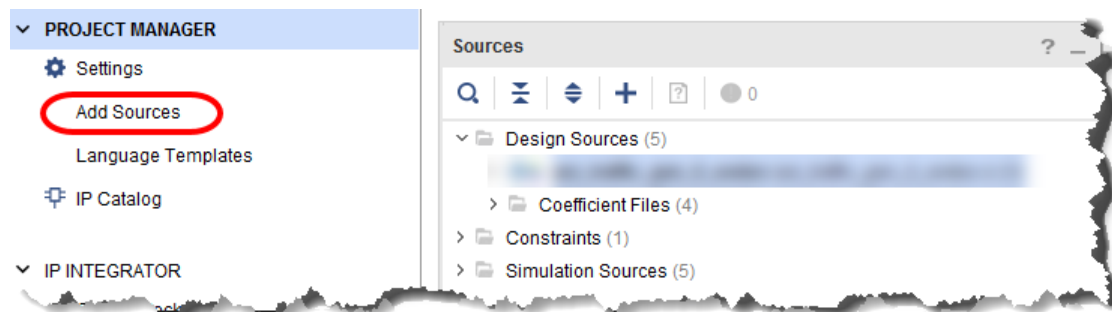
Figure 34: Selecting Add Sources

1-1-2. Select **Add or create simulation sources**.
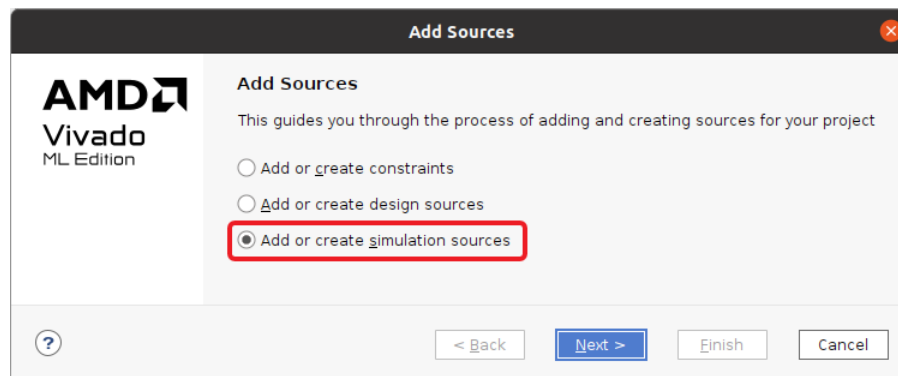
Figure 35: Add Sources Dialog Box

AMD⤢
together we advance_

Note that selecting **Add or create simulation sources** will designate the file for simulation only. When creating a design source file, you would select **Add or create design sources**, which will designate the file for both synthesis and simulation.

1-1-3. Click **Next**.

The Add or Create Simulation Sources dialog box opens.

1-1-4. Click the **Plus** (✚) icon and select **Create File**.
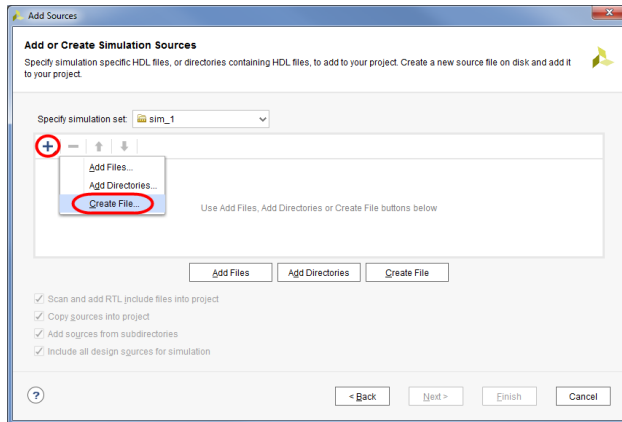
Figure 36: Selecting Create File

The Create Source File dialog box opens.

1-1-5. Enter **your HDL sources** as the file name.

1-1-6. Select **your preferred language** from the File type drop-down list.
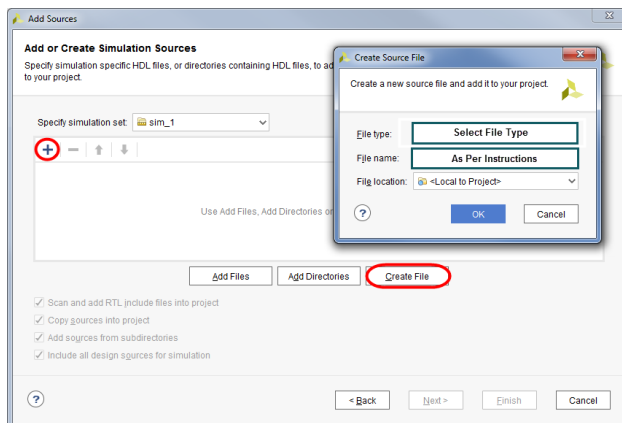
Figure 37: Entering Testbench Filename and Type

1-1-7. Click **OK** in the Create Source File dialog box.

1-1-8. Click **Finish**.

The Define Module dialog box opens.

1-1-9. Click **OK** to create the module without ports, since this module does not require them.

1-1-10. Click **Yes** to confirm that there the module definition has not been changed.

Adding Existing Constraints

1-1. **Add *your constraints file* to the design.**

1-1-1. Click **Add Sources** under Project Manager in the Flow Navigator.

1-1-2. Select **Add or create constraints** (1).
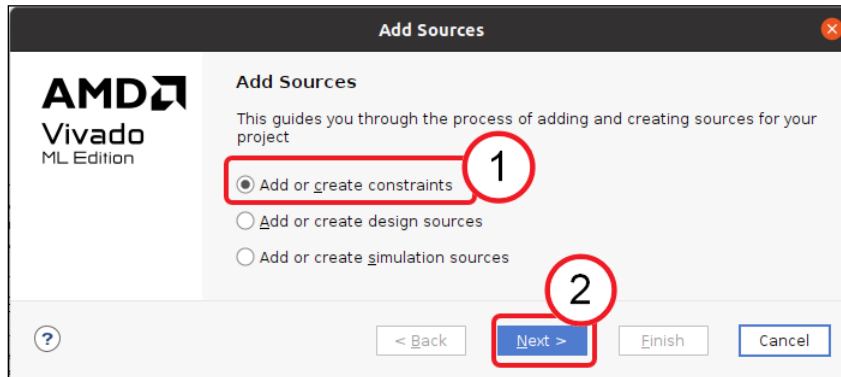
Figure 38: Adding a Constraint File to a Vivado Design Suite Project

1-1-3. Click **Next** to advance to the next dialog box (2).

1-1-4. Click the blue **Plus** icon (➕).

1-1-5. Select **Add Files** to open the Add Constraint Files dialog box.

1-1-6. Browse to the `$TRAINING_PATH/CustEdIP` directory.

> **Remember:** You can always open a terminal window and enter **`echo $TRAINING_PATH`** to see the value of this variable.

1-1-7. Select **your constraints file** as the desired constraint file (1).
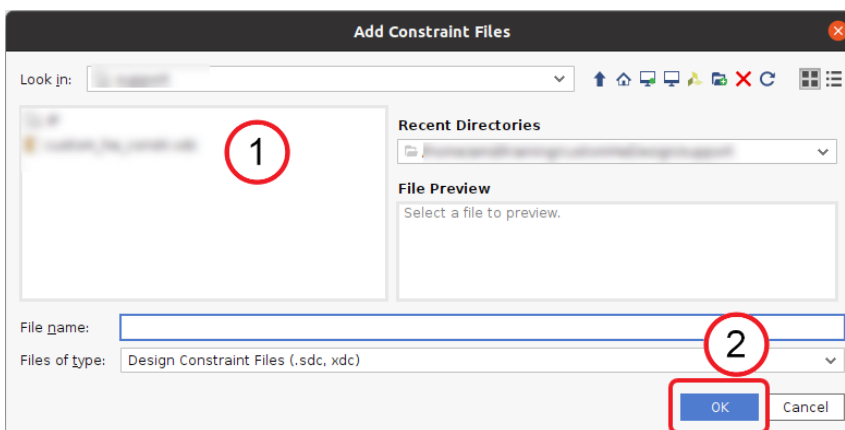
Figure 39: Selecting One or More Constraint Files

1-1-8. Click **OK** to select the file (2).

1-1-9. Click **Finish** to add the constraint file to the project.

AMD
together we advance_

Embedded Operations

In This Section

Exporting Only the Hardware Description

Exporting only the hardware description rather than the completed bitstream is beneficial during the hardware development cycle in that the software engineers can begin writing code for the platform before needing the bitstream.

This engages the software developers earlier in the cycle and provides additional time for the hardware engineers to make final tweaks to the hardware portion of the design and ensure that the design meets timing.

1-1. Export only the hardware description.

1-1-1. Select **File** (1) > **Export** (2) > **Export Hardware** (3).
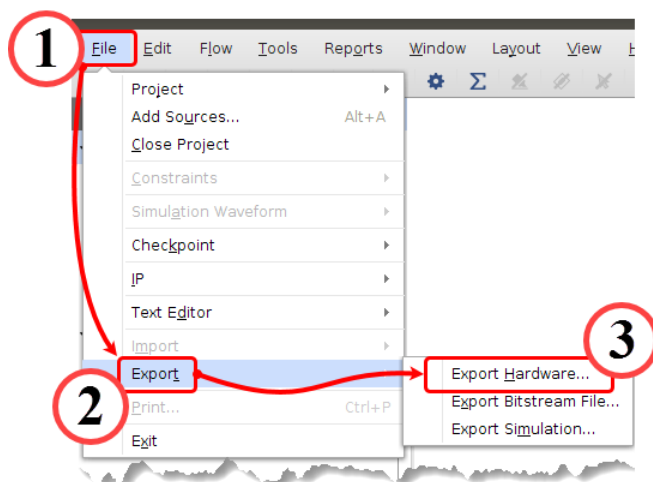
Figure 40: Exporting a Design

The Export Hardware dialog box opens.

1-1-2. When the Export Hardware Platform welcome page opens, click **Next**.

1-1-3. Select **the type of output** (1).

If your design is a PS-only design, select **Pre-synthesis**.

Otherwise, ensure that your design is completely implemented with a generated bitstream and select **Include bitstream**.
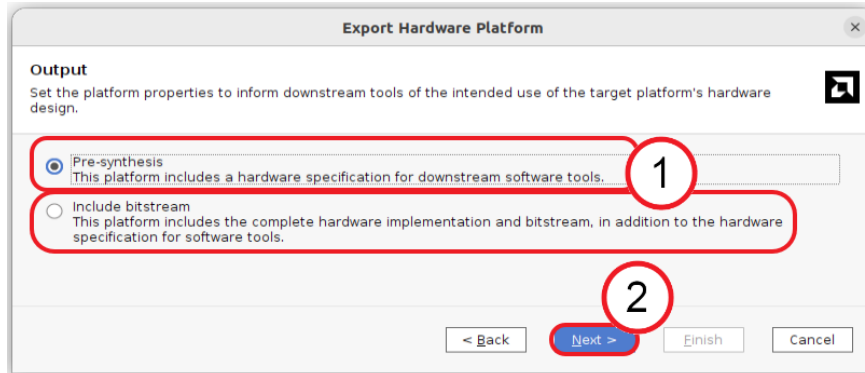
Figure 41: Selecting What to Export

1-1-4. Click **Next** to advance to the next dialog box (2).

1-1-5. Select the XSA file name, which is the name of the output file (1).

This is automatically populated; however, you can select a different name.

1-1-6. Populate the Export to field with *the location of the files to be exported from the Vivado Design Suite* as the export location (2).

This option enables you to specify a location for the exported hardware definition. Often it is beneficial to choose a directory separate from the hardware design files.

With the default *<Local to Project>* option, the Vivado Design Suite will export the hardware definition to a sub-directory of the current Vivado Design Suite project directory.
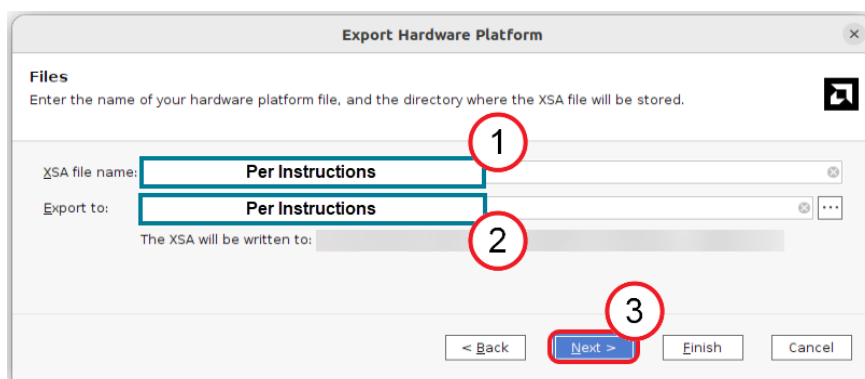
Figure 42: Identifying the Export File Name and Location

1-1-7. Click **Next** to advance to the next dialog box (3).

1-1-8. Click **Finish** to proceed with the export.

This will create a file containing a description of the hardware and may contain a bitstream as well at the selected export path.

AMD
together we advance_

Exporting the Hardware Description and Bitstream for Software Development

1-1. **Export the design for the software development tools.**

The Vivado Design Suite enables you to export a description of the embedded system design in the form of an XSA file. This XSA can contain the design's bitstream.

1-1-1. Select **File** (1) > **Export** (2) > **Export Hardware** (3).
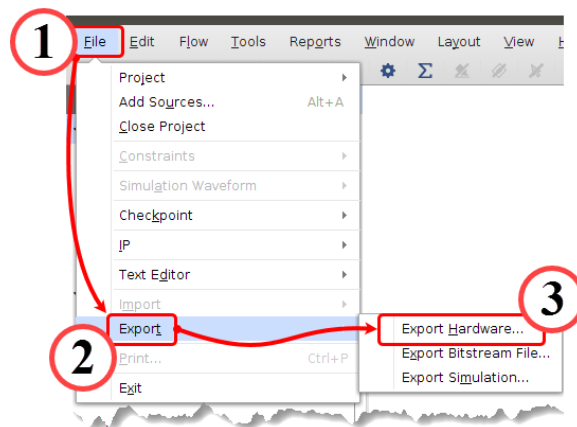
Figure 43: Exporting a Design

The Export Hardware Platform dialog box opens.

1-1-2. Click **Next** to advance past the explanation panel.

1-1-3. Select **the type of output** (1).

If your design is a PS-only design, select **Pre-synthesis**.

Otherwise, to ensure that your design is completely implemented and a bitstream is created, select **Include bitstream** (for Zynq 7000 and Zynq UltraScale+ devices) or **Include device image** (for Versal devices).
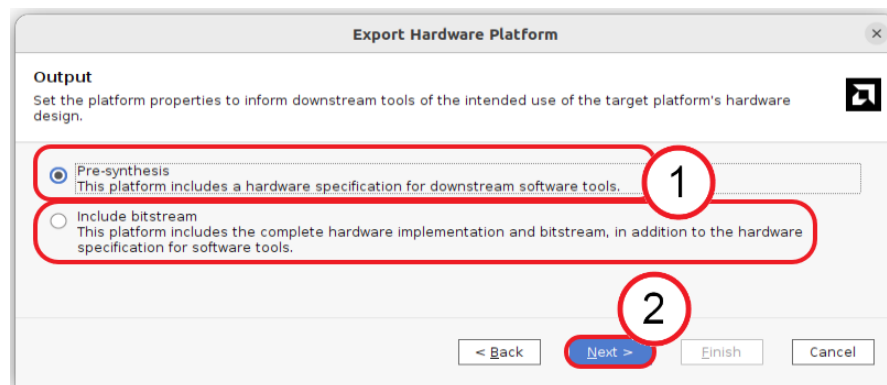
Figure 44: Selecting What to Export

1-1-4. Click **Next** to advance to the next dialog box (2).

1-1-5. Enter **your XSA file name** as the XSA file name, which is the name of the output file (1).

Usually, this is automatically populated; however, you can change this name.

1-1-6. Leave the *Export to* field with its auto-populated path, unless you want to select a different path (2).

This option enables you to specify a location for the exported hardware definition. Often, it is beneficial to choose a directory separate from the hardware design files.

With the default *<Local to Project>* option, the Vivado Design Suite will export the hardware definition to a sub-directory of the current Vivado Design Suite project directory.
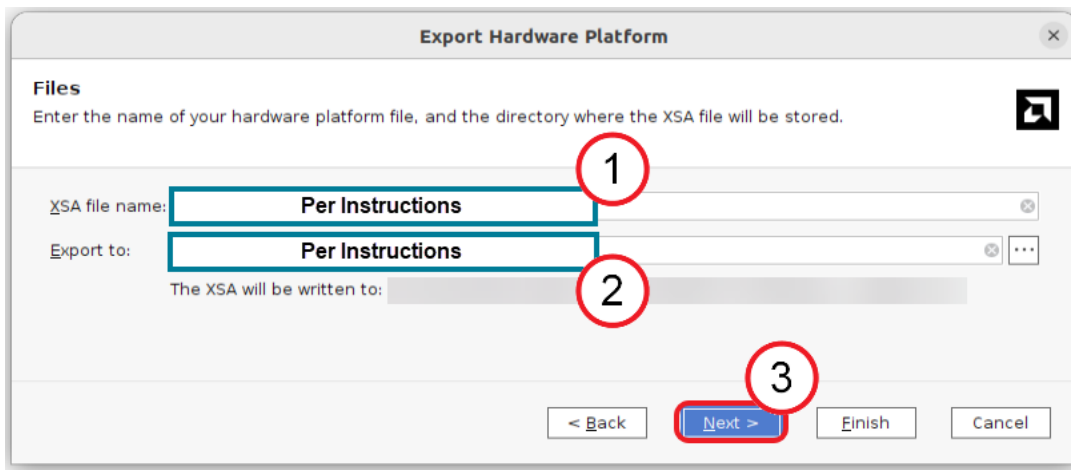
Figure 45: Identifying the Export File Name and Location

1-1-7. Click **Next** to advance to the next dialog box (3).

1-1-8. Click **Finish** to proceed with the export.

This will create a file containing a description of the hardware and may contain a bitstream as well at the selected export path.

AMD
together we advance_

Vivado IP Integrator Operations

In This Section

Creating an IP Integrator Block Design

The Vivado IP integrator is a graphical tool that assists you in "stitching" together various pieces of IP. This tool can be used for both embedded and non-embedded designs.

1-1.   Create a Vivado IP integrator block diagram.

1-1-1. Expand **IP INTEGRATOR** in the Flow Navigator if necessary (1).

1-1-2. Click **Create Block Design** to start creating a new IP subsystem (2).
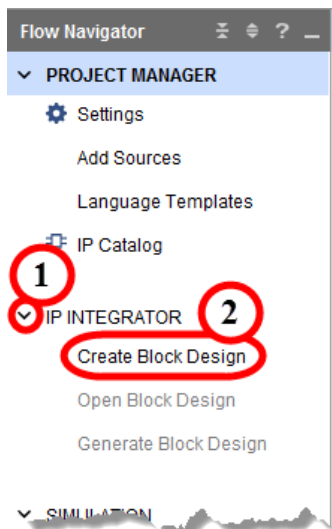
Figure 46: Launching the IP Integrator

1-1-3. Name the design **name of your block design** when the Create Block Design dialog box opens (1).
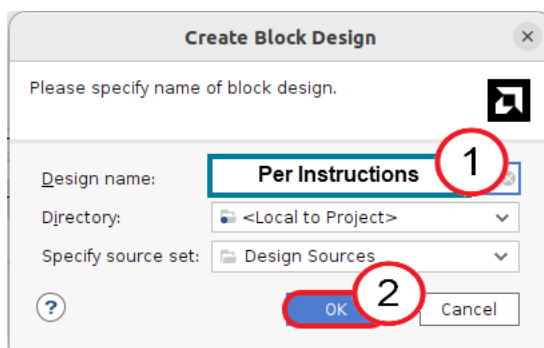
Figure 47: Creating an IP Integrator Block Design

1-1-4. Click **OK** to open a new, blank IP integrator canvas (2).

AMD
together we advance_

The IP integrator workspace opens with a note in the canvas area inviting you to begin adding IP.
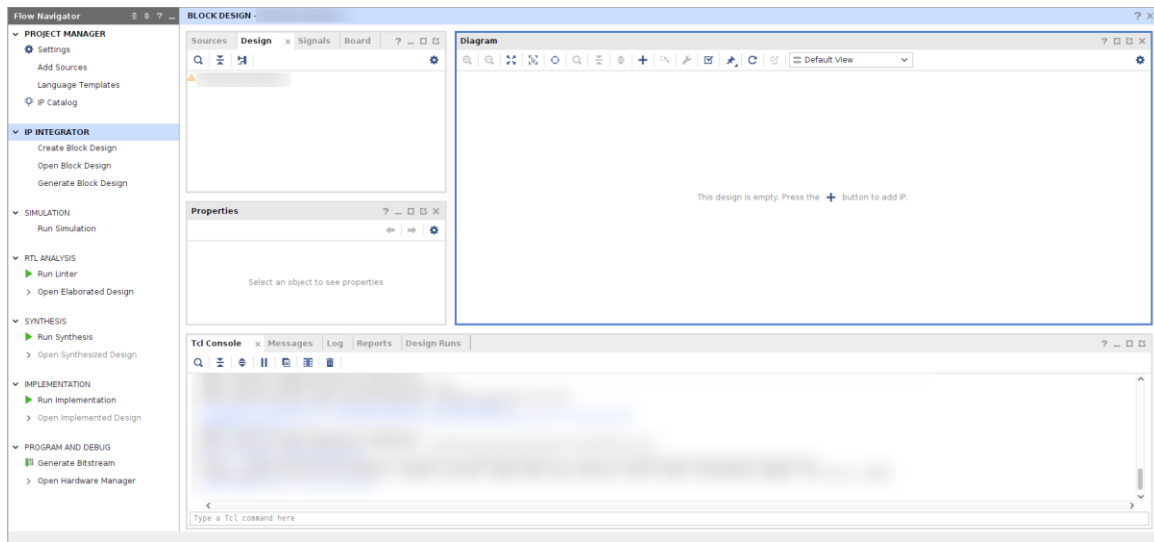
Figure 48: Initial View of the IP Integrator Tool

Closing a Block Design

1-1.   Close the block design.

1-1-1. [Optional] If you want to preserve the block design, press <**Ctrl + S**>.

1-1-2. Select **File** > **Close Block Design**.

Alternatively, you can click the "X" in the upper right-hand corner of the Block Design view.
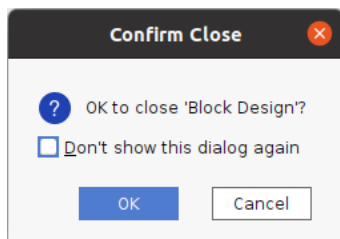
Figure 49: Confirming Block Design Closure

Note: You can select **Don't show this dialog again** if you do not want to confirm closure in the future.

1-1-3. Click **OK**.

Opening a Block Design

1-1. **Open the *name of your block design* block design.**

1-1-1. Access the **Sources** > **Hierarchy** view.

1-1-2. Locate the block design in the listing of the hierarchical sources.

If it is not readily viewable, click the **Expand all** icon (⬍).

1-1-3. Double-click the block design entry.

The block design is indicated with a block design icon (⛁).
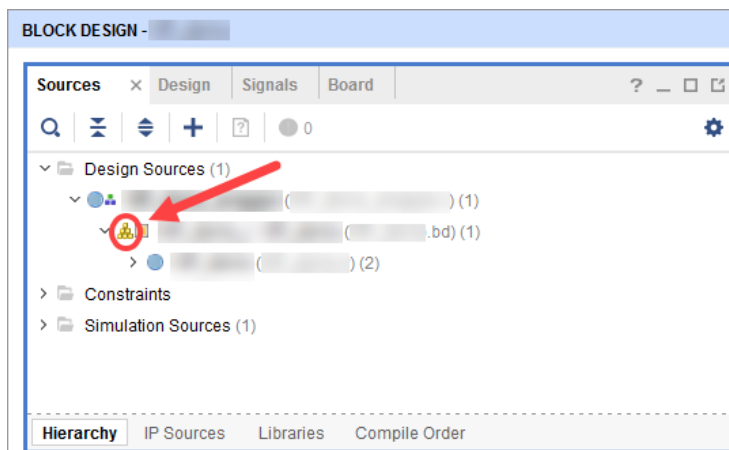
Figure 50: Opening an Existing Block Design

Zoom Operations in the IPI Block Diagram Editor

There are two methods to perform zoom operations in the block diagram editor.

o   Method 1: Use the mouse gestures to zoom in for closer examination.

AMD
together we advance_
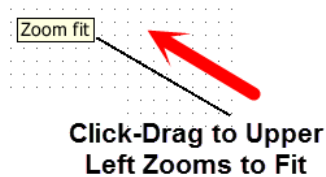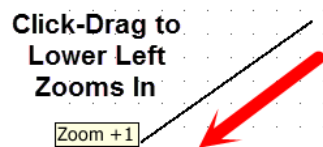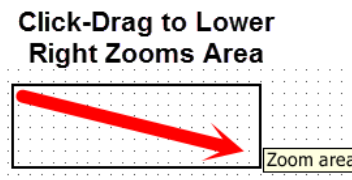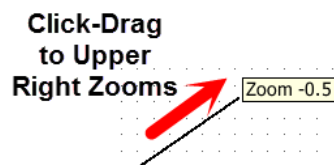
o   Method 2: Use the horizontal toolbar to the top of the block design canvas.
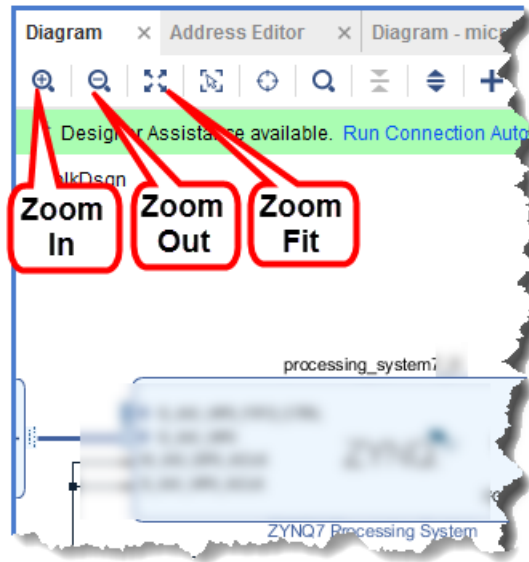
Figure 51: Block Diagram Zoom Toolbar

Adding a Processor to the Block Design

1-1.   Add your processor to the IP integrator canvas.

1-1-1. Open the IP catalog in one of two ways:

o   Click the **Add IP** icon on the left border of the workspace.

Figure 52: Opening the IP Catalog from the Add IP Icon

-- OR --

o   Right-click any background space in the workspace and select **Add IP**.
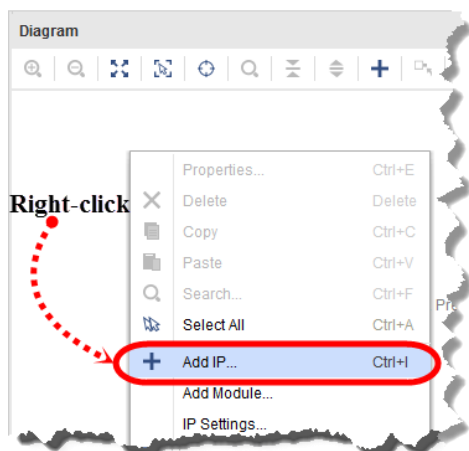
Figure 53: Opening the IP Catalog from Right-Clicking the Workspace

Important Note: Using Window > IP Catalog will add the new IP to the top level of hierarchy of the design—it will NOT add the IP to the diagram! You can, however, float the Window > IP Catalog and drag-and-drop from the catalog onto the canvas of the block design.

Once the IP catalog opens, you can search for the processor block.
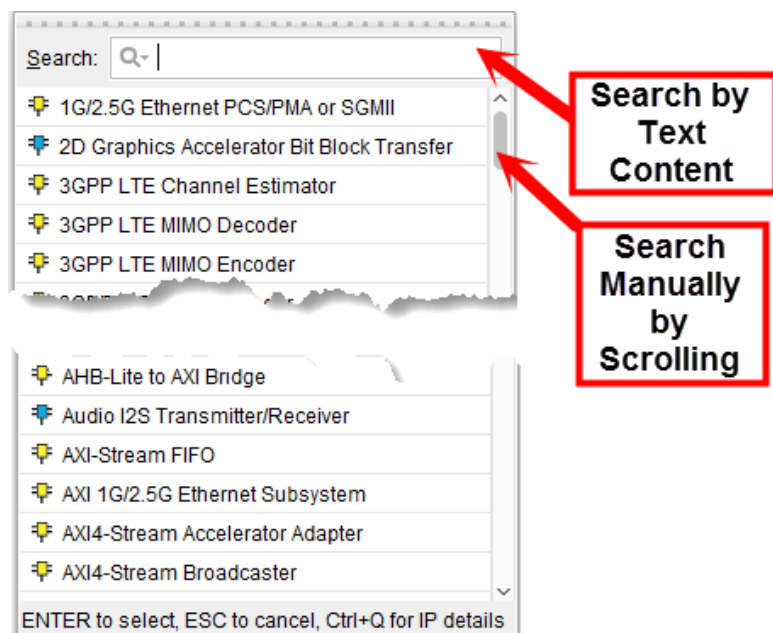
Figure 54: Two Mechanisms to Search for IP

AMD
together we advance_

1-1-2. Enter part of the processor name into the Search field to narrow the search parameters (1).

For example, **zynq** will show the PS for the selected Zynq family or **microblaze** for the MicroBlaze processor. Note that the PS IP will only appear when a family containing this IP or a board with this part has been selected for the design.
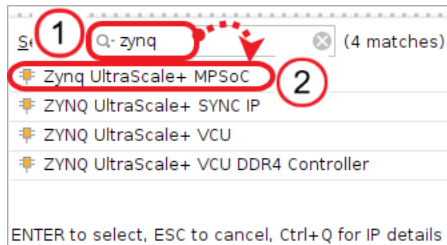
Figure 55: Searching for the Zynq UltraScale+ MPSoC Block

1-1-3. Double-click **your processor** to add its IP block to the design (2).
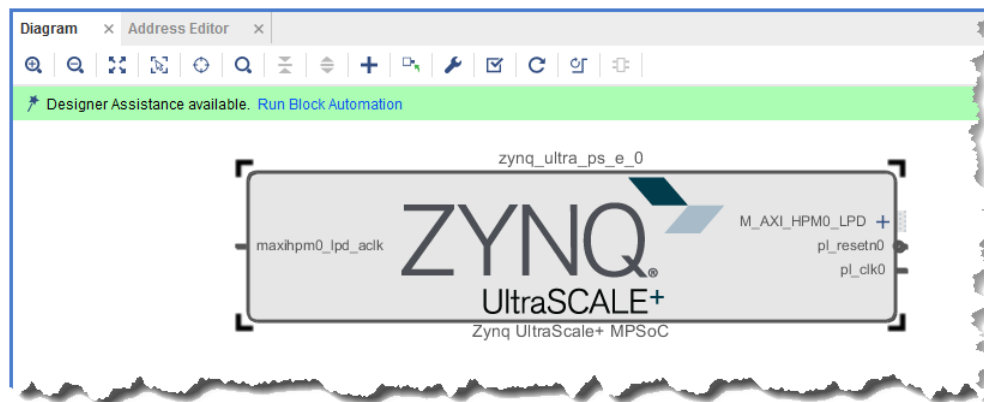
Figure 56: Zynq UltraScale+ MPSoC Block Symbol

Adding a Zynq Family Processor Block

1-1. **Add a Zynq PS processor block to the design. While the specifics of the processing systems found in the Zynq 7000 SoC, Zynq UltraScale+ MPSoC/RFSoC, and Versal devices differ, the process of instantiating them is the same.**

There are several ways to open the IP catalog. Select one of the following procedures to add IP to the canvas.

1-1-1. Use one of the following methods to open the IP catalog:

- Click the **Add IP** icon (**+**) in either the middle of the canvas or the toolbar.

This icon can be found in the middle of the canvas only when you are starting with a blank page.
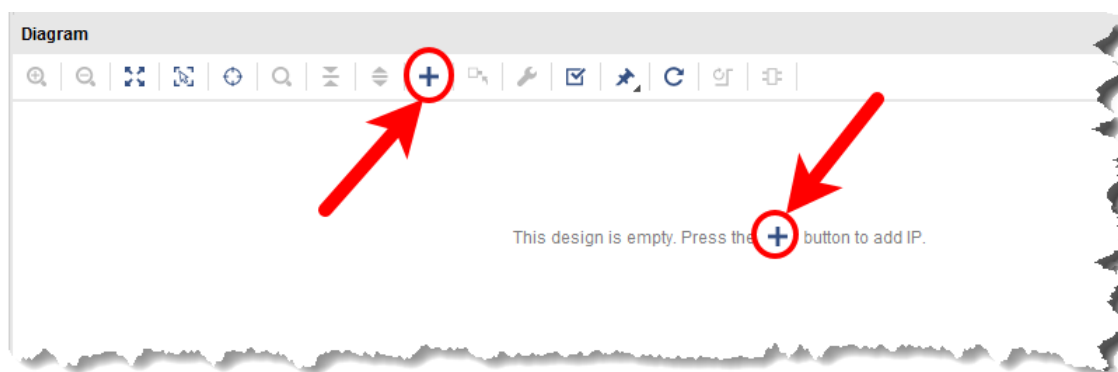
Figure 57: Opening the IP Catalog from the Initial Information Bar Display

-- OR --

- Right-click any background space in the workspace and select **Add IP**.
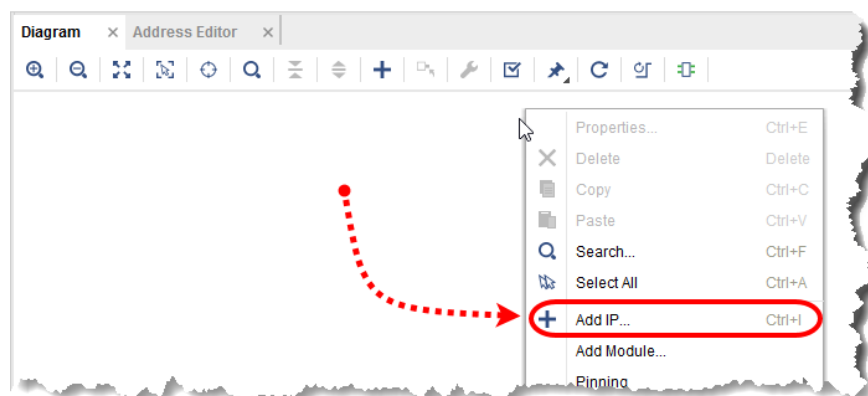
Figure 58: Opening the IP Catalog by Right-Clicking the Workspace

-- OR --

- Press <**Ctrl + I**> to access the IP catalog.

AMD
together we advance_

Note: Although the convention is to show the capital letter, the key combination uses a lower case 'i'.

Once the IP catalog opens, you can search for the Zynq SoC processor block.
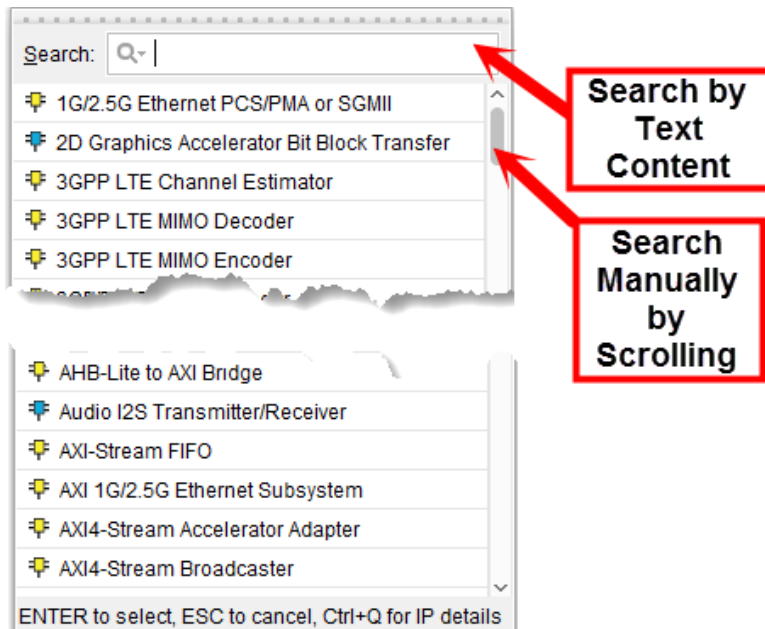
Figure 59: Two Mechanisms to Search for IP

1-1-2. Enter **zynq** into the Search field to narrow the search parameters (1).

The proper IP will appear depending on the board or device that you selected. The PS of the Zynq 7000 device will appear as ZYNQ7 while the MPSoC/RFSoCs will show as Zynq UltraScale+ MPSoC, and so forth. Only the IP supporting the device you selected will appear in the IP list.

Note that the processing system for the Versal devices is referred to as "Control, Interfaces & Processing System" and will NOT appear under a search for "Zynq" as it is not in the Zynq family of devices.
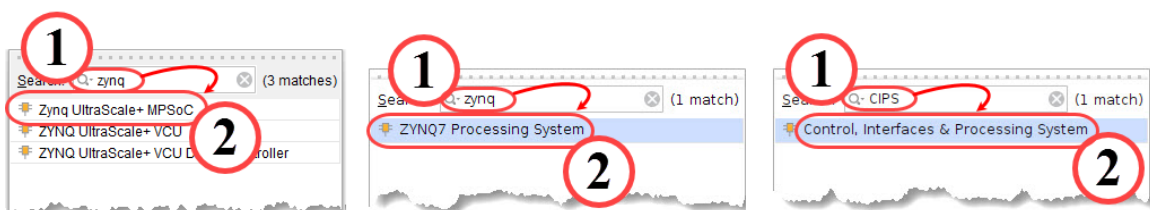
Figure 60: Selecting Multiple Architectures

This is the IP for the entire processing system (PS).

1-1-3. Double-click the IP name to add the processing system block to the design (2).

Figure 61: Zynq UltraScale+ MPSoC/RFSoC PS Block Symbol
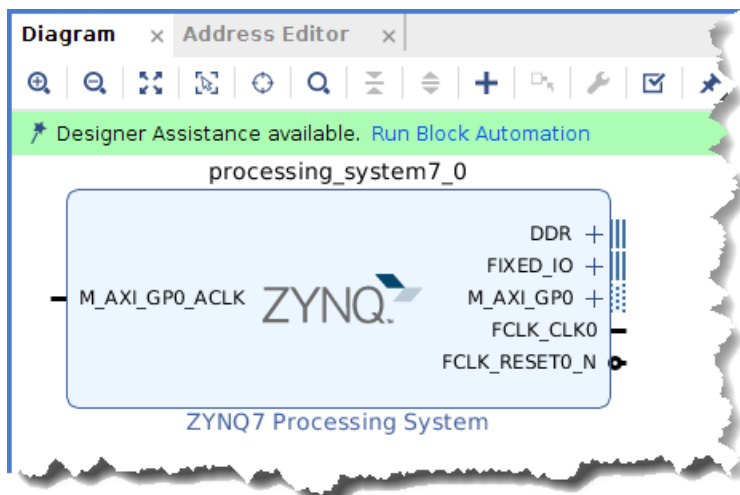
Figure 62: Zynq 7000 SoC PS Block Symbol
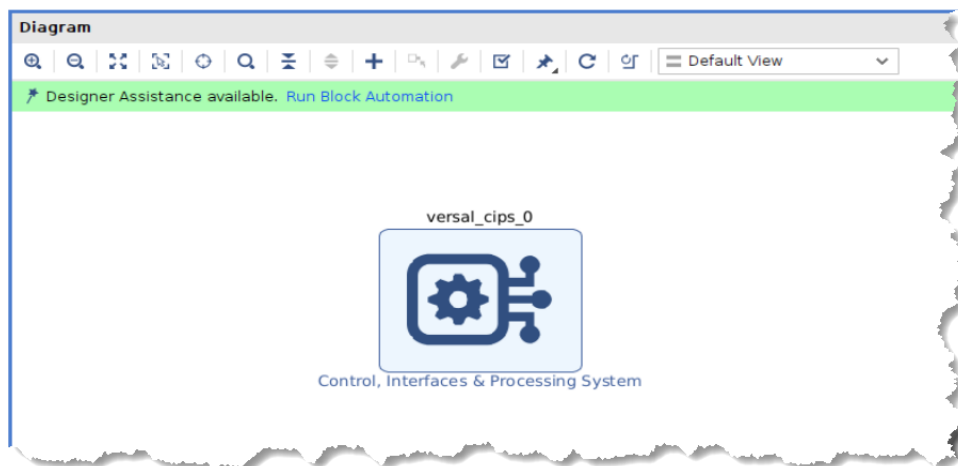
Figure 63: Versal Adaptive SoC CIPS Block Symbol

AMD
together we advance_

Adding a Processing System Block

1-1.   Add a processing system block to the design.

The processing system is available for multiple devices, including the Zynq 7000 SoC, MPSoCs, RFSoCs, and other families. Only the processing system for the specific family can be added to the design. That is, you cannot add a Zynq 7000 SoC processing system to a Versal part.

There are several ways to open the IP catalog. Since this is a blank canvas, you are invited to click the add IP icon in the middle of the canvas.

1-1-1. Click the **Add IP** icon (**+**) to open the IP catalog.
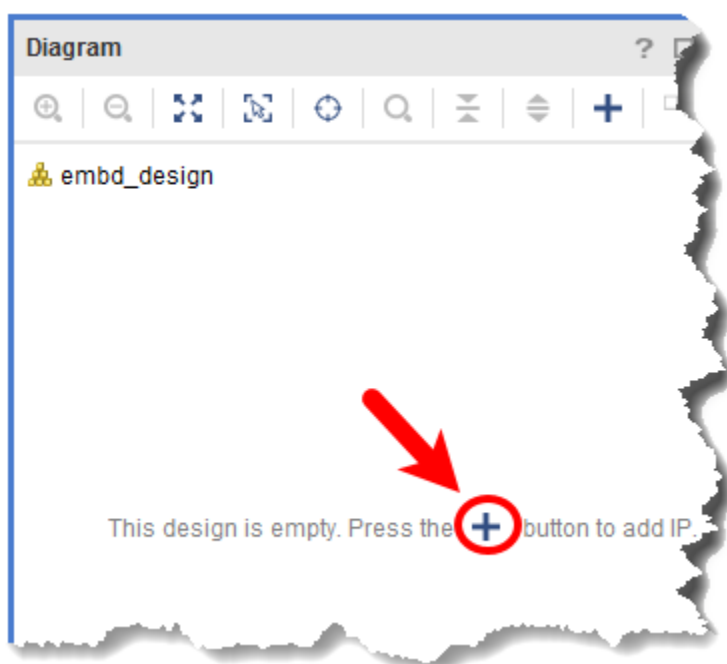
Figure 64: Opening the IP Catalog from the Initial Information Bar Display

Regardless of whether the design already has IP placed, you can always open the IP catalog in one of several ways:

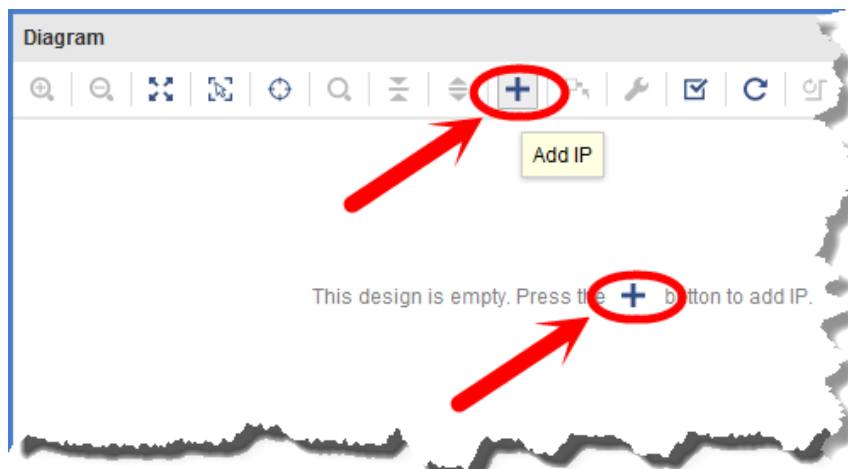o   Click the **Add IP** icon (**+**) in the toolbar of the workspace.
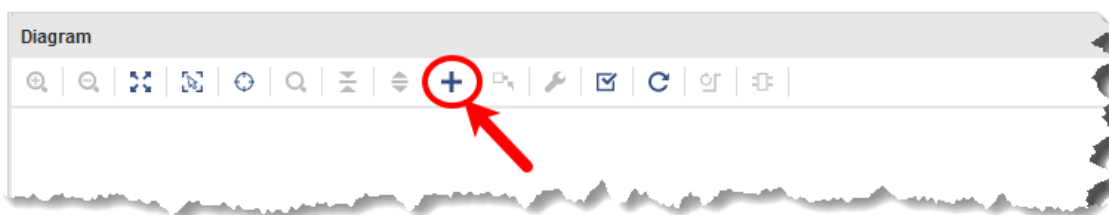
**Figure 65: Opening the IP Catalog from the Add IP Icon**

-- OR --

o   Right-click any background space in the workspace and select **Add IP**.
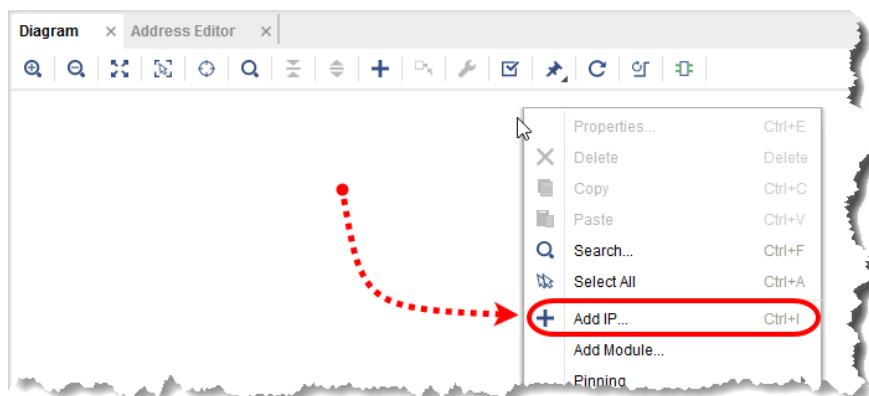
Figure 66: Opening the IP Catalog by Right-Clicking the Workspace

-- OR --

o   Press <**Ctrl + I**> to access the IP catalog.

Note: Using Windows > IP Catalog will add the new IP to the top level of hierarchy of the design — it will NOT add the IP to the diagram! You can, however, float the Windows > IP Catalog and drag-and-drop from the catalog onto the canvas of the block design.

Once the IP catalog opens, you can search for the processing system block.
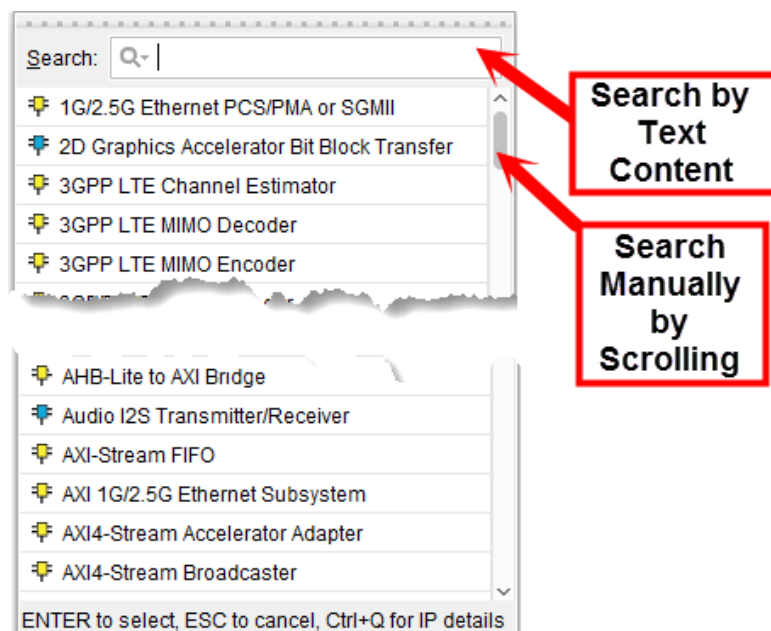

Figure 67: Two Mechanisms to Search for IP

www.amd.com

AMD

together we advance_

1-1-2. Enter **zynq** into the Search field to narrow the search parameters (1).

The processing system for the selected family appears. The exception to this is that an MPSoC PS is used by both the MPSoC and RFSoC parts, so if an RFSoC device is targeted, a Zynq UltraScale+ MPSoCs PS is used.
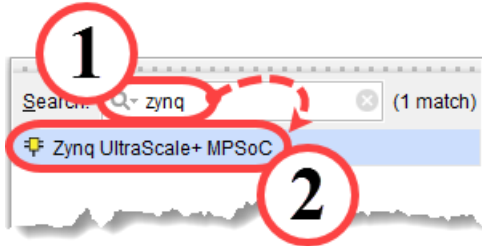
Figure 68: Selecting the Zynq UltraScale+ MPSoC IP

This is the IP for the entire processing system.

1-1-3. Double-click the IP entry to add the processing system block to the design (2).
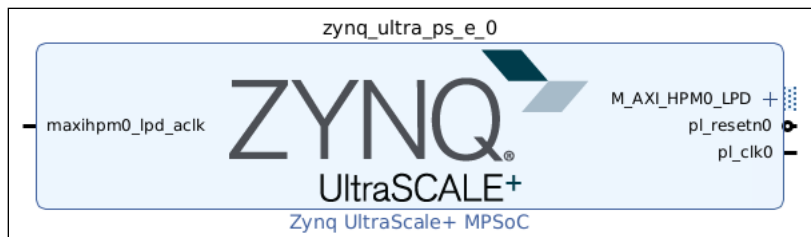
Figure 69: Zynq UltraScale+ MPSoC Block in the Block Diagram

Running Block Automation for the PS

Many IP blocks are supported by Designer Assistance. Designer Assistance automates many of the commonly used basic connections and/or configurations.

1-1. **Use Designer Assistance to make preliminary connections to the IP block.**

1-1-1. If the block design is not already open, select the **Diagram** tab in the Block Design pane to view the block design.
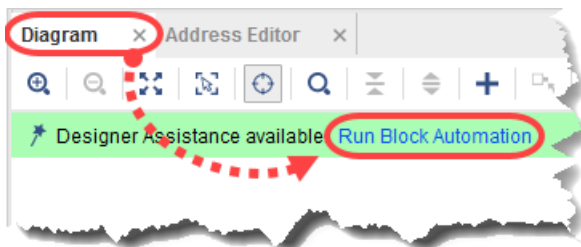
Figure 70: Selecting the Diagram Tab

1-1-2. Click **Run Block Automation** to launch Designer Assistance.

The Run Block Automation dialog box opens.

If multiple IPs have not yet been configured with the Block Automation, they will appear in the left-hand pane. This allows bulk selecting IP.

1-1-3. Select only the name of the PS block from the left-hand pane (1).

This will bring up the specific options for the PS in the right-hand pane.

You now have an opportunity to apply a board preset. If you apply the board preset, this will undo any modifications that you made to the PS using the Re-customization Wizard.

Select this option ONLY if you have not made any changes to the PS's configuration.

1-1-4. Select or deselect the **Apply Board Preset** option (2).

Warning: Having this selection enabled will overwrite any work done using the Re-customization Wizard.
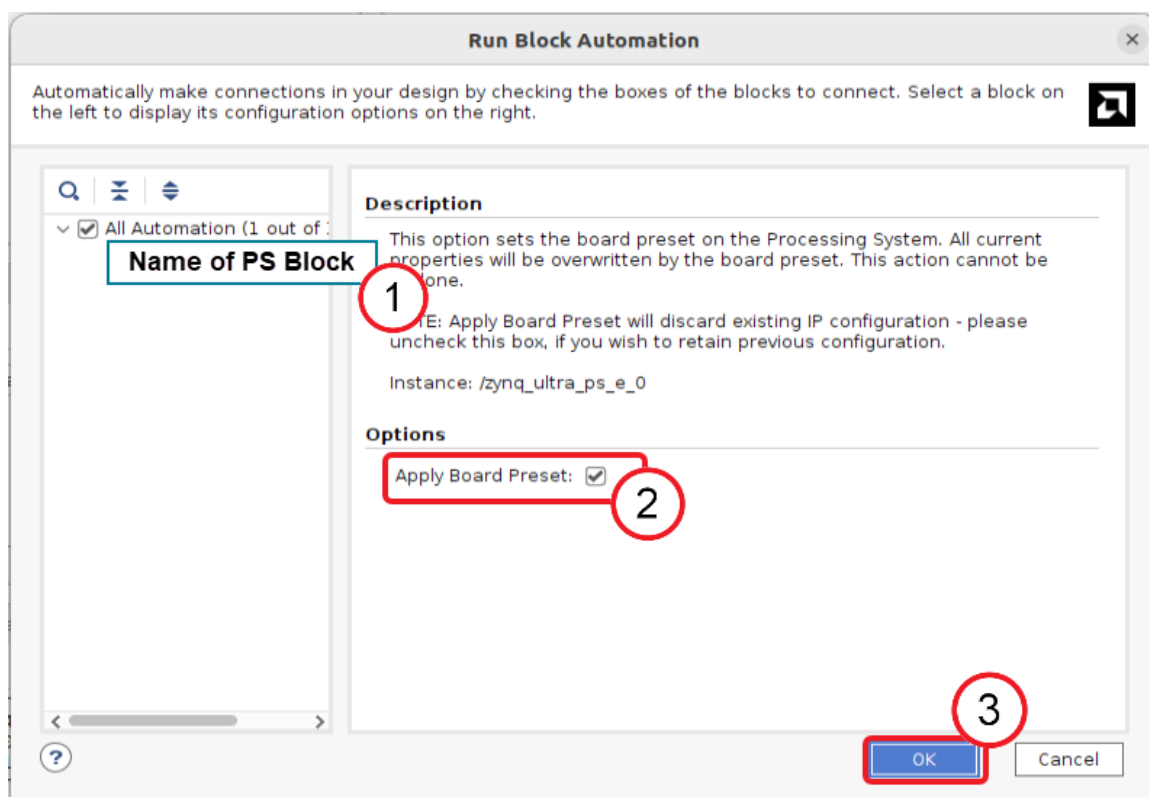
Figure 71: Running the Designer Automation Assistance on the PS

The dialog box lists the connections that will be created: the FIXED_IO (MIO connections) and the DDR interface connections.

1-1-5. Click **OK** to run the block automation on the PS (3).

AMD
together we advance_

Customizing the Processing System IP

The processing system (PS) contains a large number of customizable features. The following instructions will illustrate how to access various sections of the PS. Because the Re-customization Wizard differs among the different processing systems, explicit directions regarding what to turn on or off is omitted. This step is one of exploration rather than achieving a specific outcome.

While the Re-customization Wizard for the Versal devices is significantly different, it fundamentally performs the same types of tasks: identifying which peripherals are used, how they are configured, which AXI connections are available to the PL, etc.

The figures and instructions focus on the Zynq family of PS, and the details of the Versal device's PS are not discussed here; however, notes regarding the Versal device are included where appropriate.

1-1. Access the Re-customization dialog box.

1-1-1. Double-click the PS block.

The specific name of this block varies with the type of device. Some variation of "Zynq" will bring up the PS for the Zynq 7000 devices as well as the MP/RFSoC devices. The Versal device has its processing system IP as "CIPS" for Control, Interfaces & Processing System.

Users with projects targeting a ZCU104 board or any other MPSoC or RFSoC device will see a ZYNQ UltraScale+ block. Zed/ZC702 users will see a Zynq entry specific to the part or board selected. Versal adaptive SoC users (VCK190) will see the CIPS module.

-- OR --

Right-click the PS/CIPS block and select **Customize Block**.
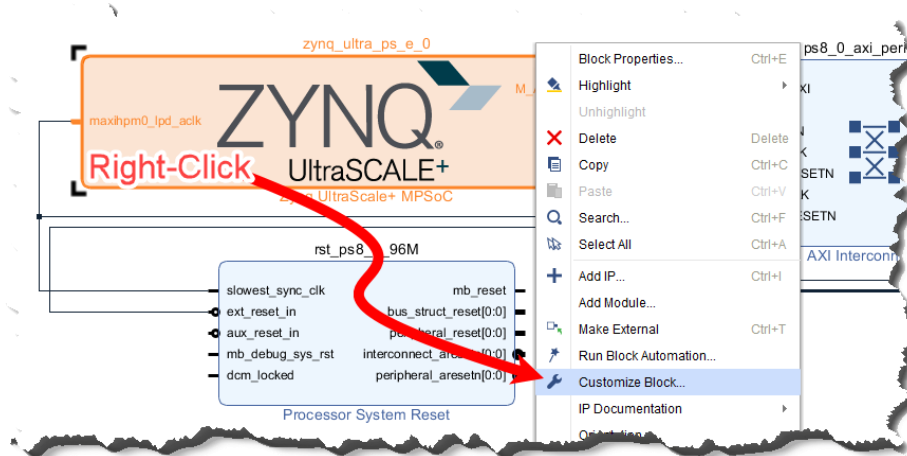
Figure 72: Opening the Re-customization Wizard for the PS (MP/RFSoC Example Shown)

The Re-customization dialog box opens to reveal the PS's block design.

1-2. **[Zynq 7000 SoC users]: Import board-specific settings.**

> **If you are using a supported evaluation board, you will probably want to load the pre-defined settings as it contains many of the parameters specific to that board (such as DDR memory timing parameters, enabled peripherals supported by that board, etc.) A similar capability is available for the MPSoC/RFSoC devices and is explained elsewhere.**
>
> **You can also create a .bd file for your custom board if you want and import those settings.**
>
> **The Zynq UltraScale+ MPSoC presets behave differently than the Zynq 7000 device presets. The Zynq 7000 device presets contain settings for various evaluation boards while Zynq UltraScale+ MPSoC presets enable you to save and recall your own specific settings. The equivalent functionality can be found for the MP/RFSoC devices using Block Automation.**

1-2-1. Click **Presets** to access the pre-defined values associated with one of the supported boards that supports the Zynq 7000 SoC (such as the ZC702 or ZedBoard).

> **Note:** If you are building a board of your own, it is possible to create your own preset for that board. Presets are very convenient as they can describe the DDR settings as well as enable the peripherals and their pin mappings for the specific board. This saves a tremendous amount of effort for the designer.
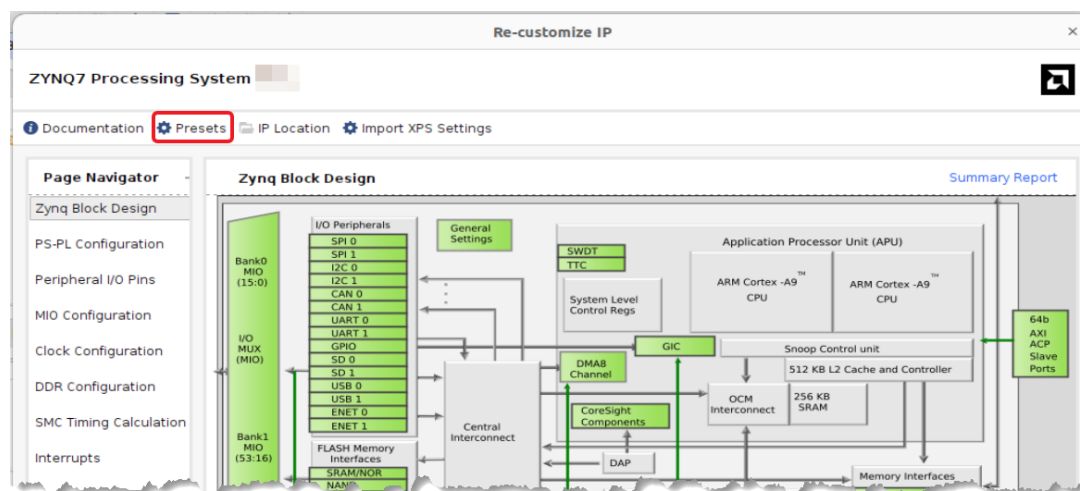
Figure 73: Accessing the Presets (Zynq PS)

> If you have a Tcl script that describes the configuration of the PS, you can load it by selecting *Presets > Apply Configuration.*

1-2-2. Select **your board** for the preset.

> This will configure the PS for the settings specific to that board, including DDR timing parameters, peripherals, etc.
>
> **Note:** The Versal device contains a multitude of presets and individual groups that can be uniquely preset for specific purposes. These settings can be saved as a system-level preset and recalled at will.

1-2-3. Skip the next instruction as it duplicates this instruction, but for Versal device users.

AMD
together we advance_

The next instruction provides you with general guidance as to how to customize various aspects of the PS. At the end of this instruction, you will be provided with specific guidance regarding how you are to customize the PS.

1-3.   [Versal adaptive SoC users]: Explore the Re-customization Wizard.

1-3-1. When the wizard opens, click **Next** to move past the "Presets" page.

1-3-2. Click the **PS PMC** block to open a configuration window for the processing system and platform management controller (1).

This block more closely aligns with the Zynq families in that it provides access to the configuration information for the peripherals, AXI interconnects, clocking, etc.

New in the Versal family's architectures is the network on chip (NoC), which facilitates rapid data movement among the major blocks in the system and, unlike the other processing systems which contains their own DDR memory controllers, the CIPS block requires the NoC to gain access to up to four independent or coordinated DDR memory controllers.

1-3-3. Click each element in the list to see the options available for that sub-section (or click **Next** which will proceed through the list).

1-3-4. Click **Cancel** to return to the previous page without making any changes (2).

1-3-5. Click the **CPM** block to access the CPM (PCIe block and CPM interface) configuration settings (3).

1-3-6. Explore the various modes available for the PCIe blocks.

1-3-7. Click **Cancel** to return to the previous page without making any changes (4).
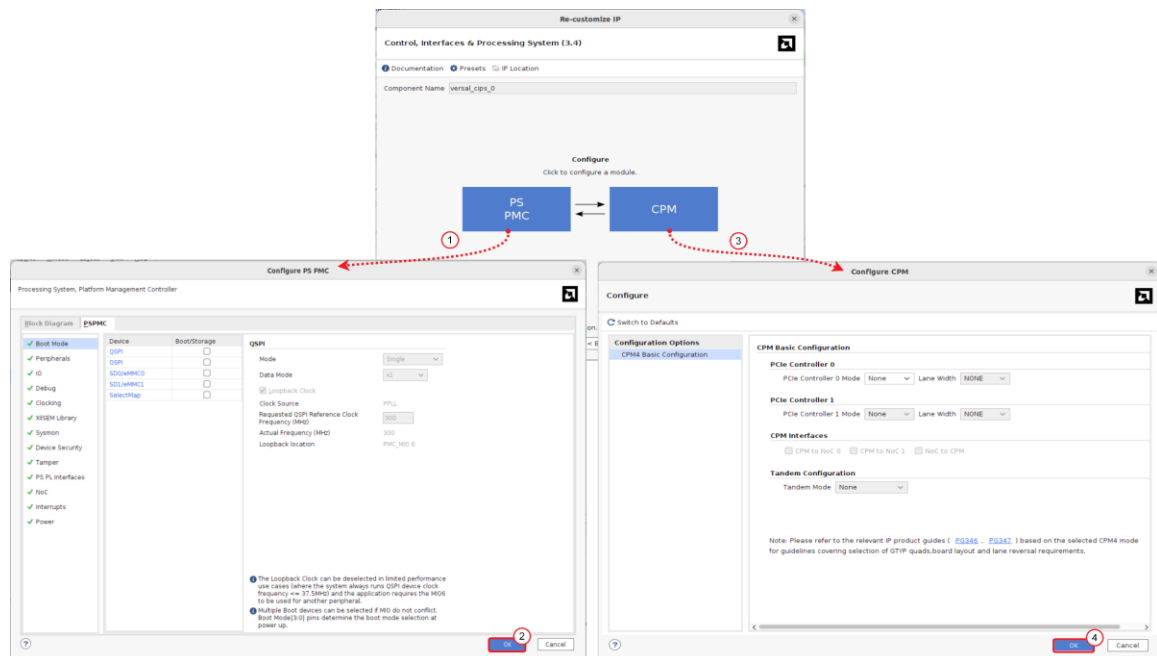
Figure 74: CIPS Re-customization Wizard - Page 2

1-3-8. Click **Cancel** to exit the wizard.

1-3-9. Enter the following in the Tcl command line to run the Tcl proc to configure the Versal device CIPS:

```
versalCIPSconfigure
```

This proc configures the CIPS IP so that the following instructions can be performed.

1-4. [All Zynq device users]: Select the page or specific block to re-customize.

1-4-1. Click the topic in the Page Navigator or any green (active) box from the diagram.

Along the left are the navigational tabs that you will use to access different groups of parameters.

Note: Advanced mode is available for the MPSoC/RFSoC families, but not for the Zynq 7000 family. The figure below represents the MPSoC devices' Re-customization main dialog box. The Zynq 7000 family is similar, but less complex.
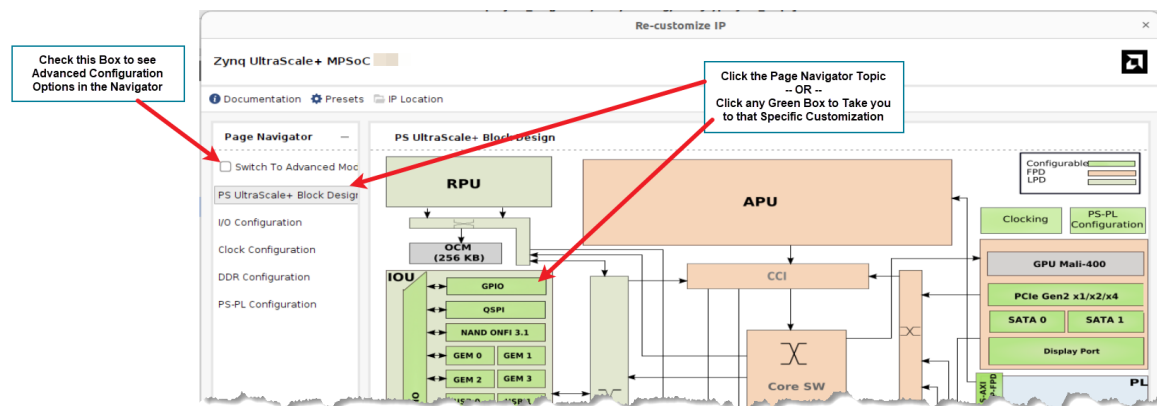
Figure 75: Navigating the Zynq PS Recustomization Dialog Box

- o PS-PL Configuration: This page contains all the signals that cross the PS-PL boundary. For the Zynq 7000 devices, these signals are broken down further into:

  - General: Contains default baud rates for the UARTs, FTM trace buffer settings, PS-PL cross triggering enables, enables for the clock triggers, and reset.

  - DMA Controller: Contains enables for the four peripheral request interfaces.

  - GP Master AXI Interface: Enables/disables access to the GP Master AXI Interfaces to the PL.

  - GP Slave AXI Interface: Enables/disables access to the GP Slave AXI Interfaces from the PL.

  - HP Slave AXI Interface: Enables/disables access to the HP Slave AXI Interfaces from the PL and sets the port width.

  - ACP Slave AXI Interface: Enables/disables access to the ACP Slave AXI Interface from the PL.

- o Peripheral I/O Pins (for Zynq 7000 SoC only)

- o I/O Configuration (MIO Configuration for Zynq 7000 SoC)

- o Clock Configuration

AMD
together we advance_

o DDR Configuration

o SMC Timing Calculation (for Zynq 7000 SoC only)

o Interrupts (Zynq UltraScale+ has this setting under PS-PL Configuration)

o PS-PL Configuration

Customizing the Zynq UltraScale+ MPSoC PS

The Zynq UltraScale+ MPSoC PS contains many customizable features. The following instructions will illustrate how to access various sections of the Zynq UltraScale+ MPSoC PS, not necessarily indicating which settings to configure.

1-1. Access the Re-customization dialog box.

1-1-1. Double-click the **Zynq UltraScale+** icon.

-- OR --

Right-click the **Zynq UltraScale+** icon and select **Customize Block**.
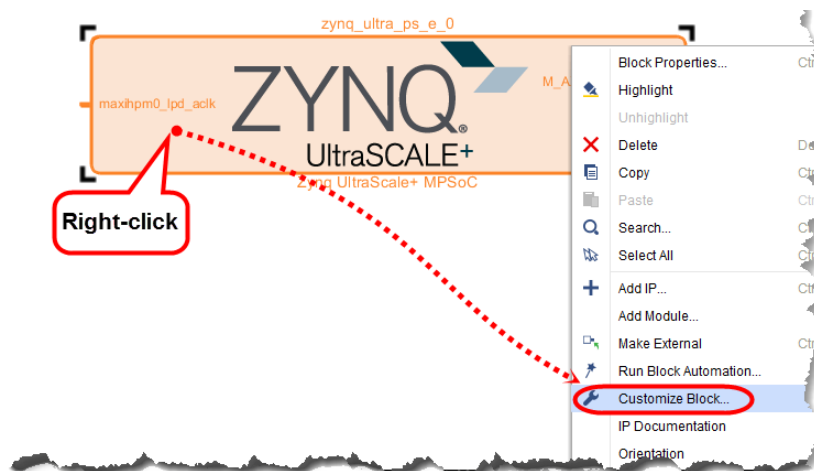
Figure 76: Customizing the Zynq UltraScale+ MPSoC

The Re-customization dialog box opens to reveal the Zynq block design.

1-2. Select the page or specific block to re-customize.

1-2-1. Click the topic in the Page Navigator or any green (active) box from the Zynq UltraScale+ MPSoC block diagram.

Along the left are the navigational tabs that you will use to access different groups of parameters.
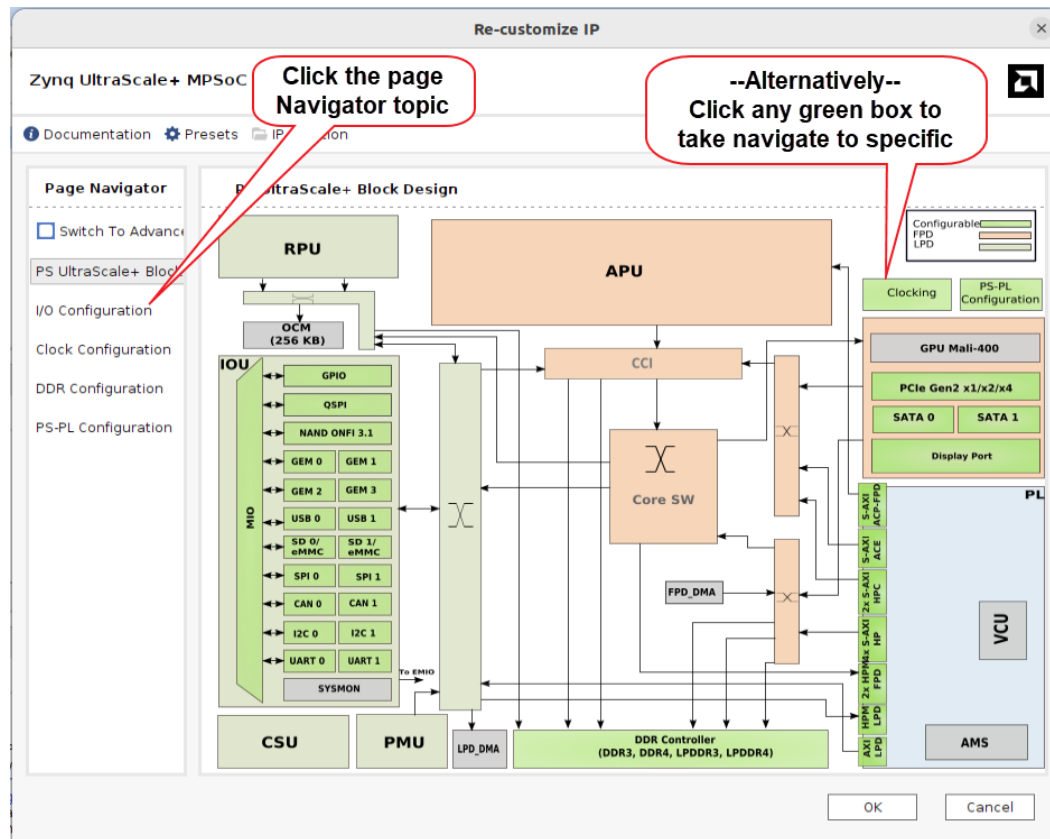
Figure 77: Re-customize IP Dialog Box

- o PS-PL Configuration: This page contains all the signals that cross the PS-PL boundary. These signals are broken down further into:

    - General: Contains interrupts from PS to PL and PL to PS, Address Fragmentation parameters and others such as the RTC, DMA, and Live Audio and Video settings.

    - PS-PL Interfaces: Contains enables for all the available AXI masters and slaves including the ACP and ACE slave AXI interfaces.

    - Debug: PS - PL cross-trigger inputs and outputs and general-purpose inputs and outputs.

- o I/O Configuration

- o Clock Configuration

- o DDR Configuration

AMD
together we advance_

Selecting a Zynq7 PS Preset Template

1-1. Configure the ZYNQ7 processing system using the Preset template.

1-1-1. Double-click the **ZYNQ7 Processing System** IP block to open the Re-customize IP dialog box.

1-1-2. Click **Presets** in the Re-customize IP dialog box.

This preset contains settings appropriate for your board, such as peripheral pin locations, DDR memory parameters, etc.

1-1-3. Select **the preset for your board**.
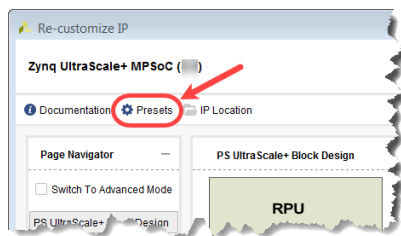
Figure 78: Selecting a Preset Template

1-1-4. Click **OK** to load the values associated with this preset.

Customizing the MicroBlaze Processor

1-1. Configure the MicroBlaze processor system with a Typical configuration.

The Typical configuration is one of several customizable configurations. It includes an MDM interface for debugging, instruction and data caches for use with DDR memory, and exception/interrupt handling. This configuration provides a balance between frequency, area, and overall performance.

1-1-1. Double-click the **MicroBlaze** IP block or right-click the **MicroBlaze** IP block and select **Customize IP**.
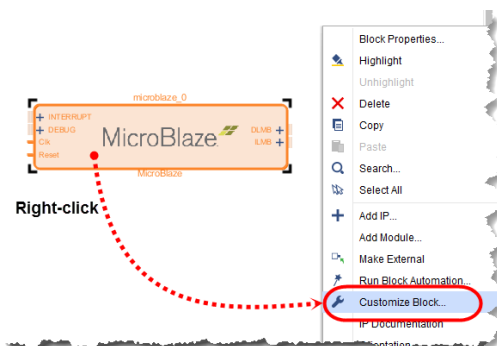
Figure 79: Starting Customization on the MicroBlaze Processor

The Re-customization IP dialog box opens.

1-1-2. Select **Typical** from the Predefined Configurations > Select Configuration drop-down list.
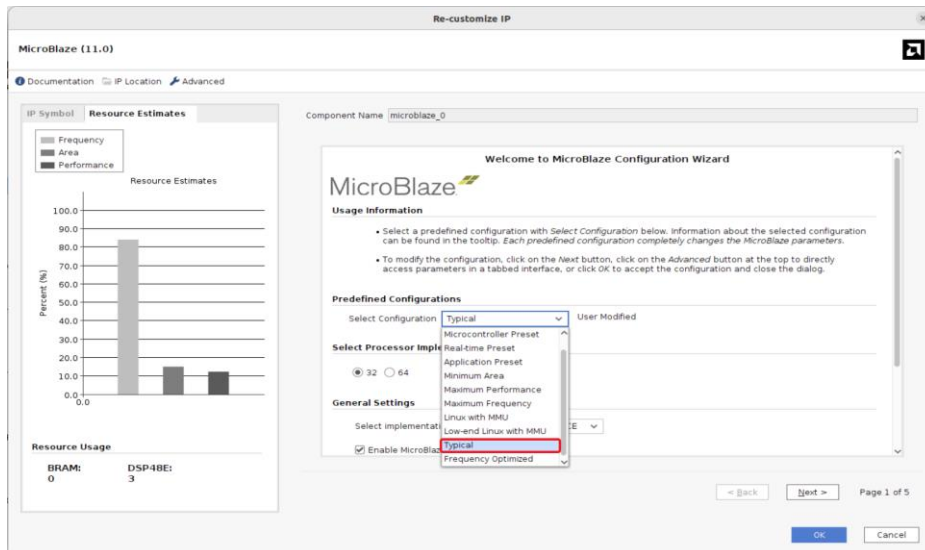
Figure 80: Selecting the Typical Configuration for the MicroBlaze Processor

Notice that the Enable MicroBlaze Debug Module, Use Instruction and Data Caches, and Enable Exceptions options are automatically selected as part of the Typical configuration. Cache memory is good to have when dealing with slower memories such as DDR and Flash.

1-1-3. Review the default settings for the Typical configuration.

The MicroBlaze processor can be built as either a 32-bit or 64-bit implementation.

1-1-4. Select the 32-bit Implementation.

1-1-5. Leave the General Settings pull-down option at **PERFORMANCE**.

While the predefined configurations and general settings (implementation optimization) choices define the general configuration and options of the MicroBlaze processor, other options can be selected and overloaded into the default settings made by these configurations. Generally, these options address supporting hardware for the core processor's capabilities and includes debugging capabilities through the MDM, caches, memory management units, etc.

1-1-6. If selected, deselect the **Use Instruction and Data Caches** option.

1-1-7. Ensure that the **Enable MicroBlaze Debug Module Interface** and **Enable Exceptions** options are selected.

This is done to illustrate how *preconfigured* settings can be overridden by the user. Cache is most useful when applied in conjunction with DDR memory. If the MicroBlaze processor is running entirely out of block RAM, the caches may actually degrade the performance of the system.

Note that there are several pages where modifications to this configuration can be made. You can explore these settings as time permits.

1-1-8. Click **OK** to save this configuration.

AMD
together we advance_

Adding IP to an IP Integrator Block Design

1-1.   Open the IP catalog.

1-1-1. The IP catalog can be opened in several ways:

- If the design is empty, add IP by clicking either of the '+' icons.
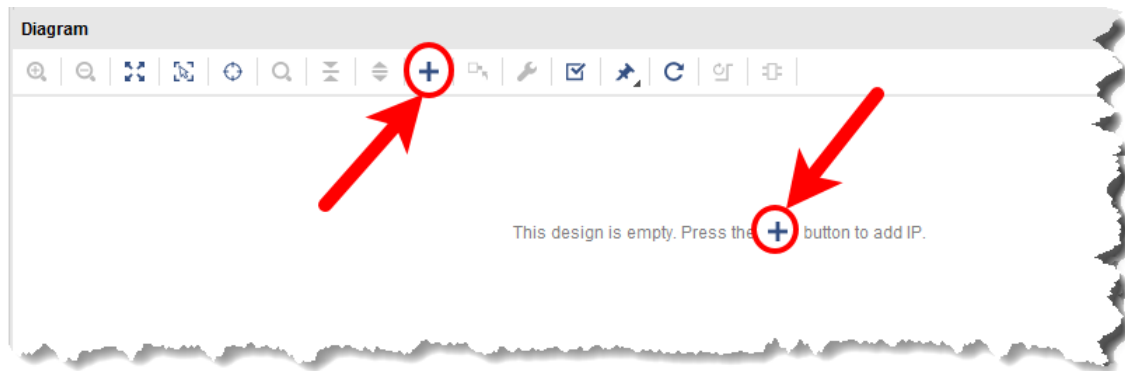
Figure 81: Opening the IP Catalog from the Initial Information Bar Display

- Right-click any background space in the workspace and select **Add IP**.
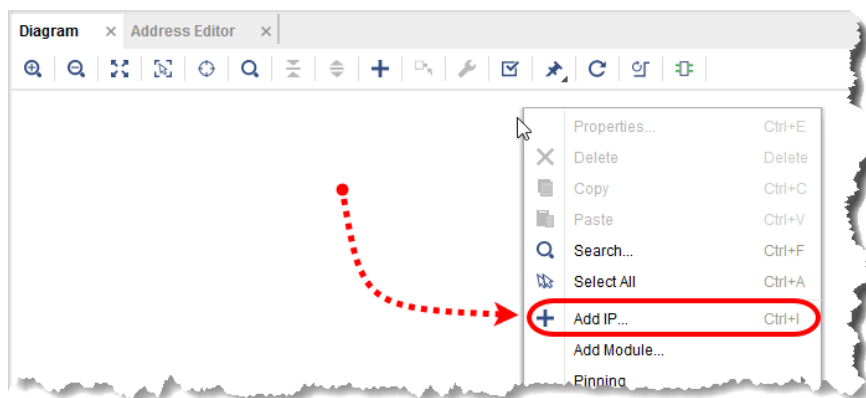
Figure 82: Opening the IP Catalog by Right-Clicking the Workspace

- Press <**Ctrl + I**> to open the IP catalog.

Important Note: Using Windows > IP Catalog will add the new IP to the top level of the design's hierarchy—it will NOT add the IP to the diagram! It is, however, possible to float this window and drag-and-drop IP into the Block Design canvas.

1-2. **Once the IP catalog opens, search for *the desired piece of IP* and add it to the canvas.**

1-2-1. Type all or part of the IP name into the Search field.

1-2-2. Locate **the desired piece of IP**.

> **Hint:** You can scroll with the mouse or use the scroll bar on the right side of the list. The more of the IP name that you provide, the fewer items that you will need to manually scroll through.
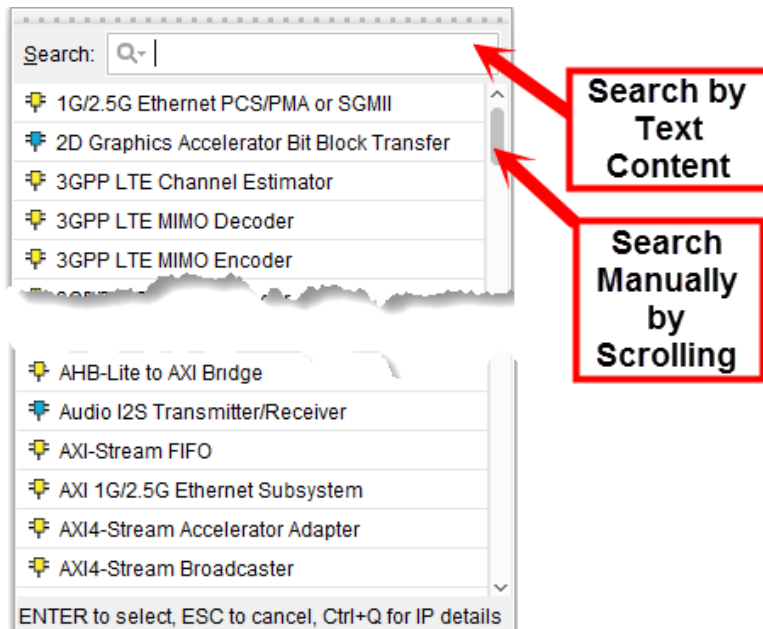
Figure 83: Two Mechanisms to Search for IP

1-2-3. Double-click the name of the IP to add it to the design (if you use this method, the IP catalog will close after the IP has been added to the design).

or

Drag-and-drop the IP into the workspace.

The IP catalog remains open once the IP has been added to the design. This is a convenient way to quickly add multiple pieces of IP. Pressing the **<Esc>** key or clicking an open space in the canvas will close the IP catalog.

AMD
together we advance_

Customizing IP

Almost every piece of IP is customizable. This provides a great deal of flexibility to the designer to create a system that is tailored to specific needs.

1-1. Open the Re-customization Wizard for *the desired piece of IP*.

1-1-1. Double-click the IP block.

Alternatively, you can right-click the IP block and select **Customize Block** from the context menu.
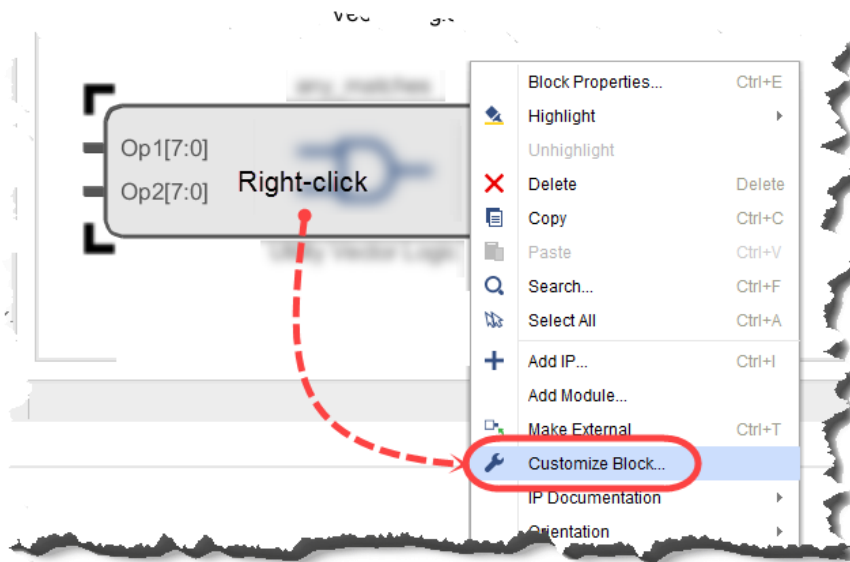
Figure 84: Opening the Recustomization Wizard for a Piece of IP

The Re-customization Wizard for that piece of IP opens.

Making Manual Connections Between Two Objects in the IP Integrator

Making connections is a simple process in the IP integrator tool. Hover over a port or interface on a block with your mouse (the cursor changes to a pencil icon from the arrow icon). You can then click-drag the connection to a legal port or interface. As the connecting wire is stretched to its destination, an assistant runs in the background and places a green check mark next to all the ports or interfaces that can be connected when the cursor approaches a legitimate connection point.

There are two mechanisms for making these connections: Using the Run Connection Automation feature, which is available for most IP, and manual connections. The latter is addressed here.

1-1.   Connect two IP blocks using the manual connection technique.

A "net" is another name for the wire, signal, or connection between two or more pins.

1-1-1. Click **a port or interface** to begin the connection process.
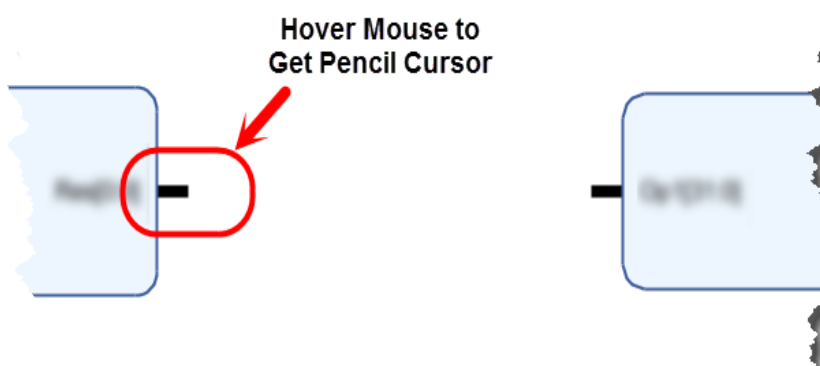
Figure 85: Hovering the Mouse Over a Port to Get the Pencil Icon

The port or interface is highlighted, showing that it has been selected and the cursor changes to a pencil, indicating that it is ready to draw/connect a wire.

1-1-2. Drag the pencil to **another port or interface**.

AMD
together we advance_

As the cursor approaches a valid port or interface, a green arrow appears next to legal connections points.

Figure 86: Finding Legal Connection Points

1-1-3. Release the mouse button to make the connection.

Figure 87: Finishing the Connection

Interfaces are connected in exactly the same way, except that interfaces are comprised of a bundle of signals rather than a single wire. The tools are designed to automatically make the proper connections between two compatible interfaces.

Note: If you want to exit this drawing mode, press <**Esc**>.

Renaming an IP Block

Renaming an IP block often helps the designer keep track of what that particular piece of IP *does* rather than what it *is*.

1-1.   Rename the IP block *with the new name for this piece of IP*.

1-1-1. Click the desired IP block in the Block Design window to open its Block Properties dialog box (1).

The current name of the selected IP block appears in the Block Properties window when the General tab (located towards the bottom left of the window) is selected.

1-1-2. Enter the new name (**with the new name for this piece of IP**) for this specific piece of IP into the Name field (2).
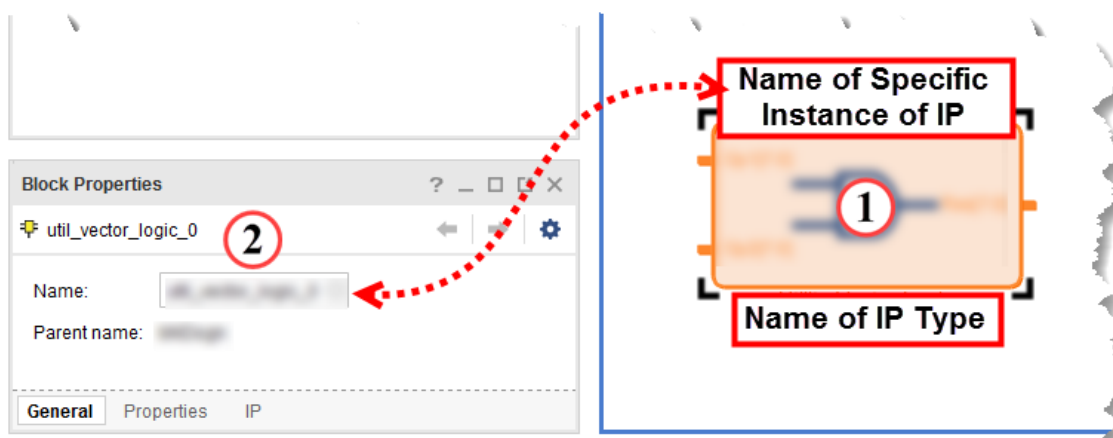
Figure 88: Renaming an IP Block

1-1-3. Press <**Enter**> to save the change.

AMD
together we advance_

Adding a Port to an IP Integrator Block Diagram

Ports are constructs that allow signals to pass between hierarchical levels. Ports can be combined together to form interfaces. Here you will learn how to create a single port. Ports can take on a variety of special cases, all of which can be selected through the *type* parameter.

Port Type	Meaning
Clock	Clock signals: placed on clock networks; enter the frequency in the Frequency field
Reset	Reset network
Interrupt	For embedded systems: ports marked as interrupt are entered into the interrupt select table for the processor(s)
Data	General-purpose data signals
Clock Enable	Clock enable signals: used with clocks
Other	Undefined signals: cannot be connected to ports other than those marked as "other"

1-1. Create a port.

1-1-1. Right-click an unoccupied section of the Vivado IP integrator canvas (1).

1-1-2. Select **Create Port** (2).
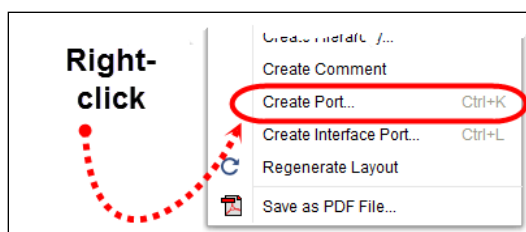
Figure 89: Creating a New Port

The Create Port dialog box opens.

1-1-3. Enter **the name of the port** in the Port name field (1).

1-1-4. Select the port direction as **input, output, or inout/bidirectional** (2).

1-1-5. Select the type to be **according to the table below** (3)**.**

1-1-6. Ensure that the **Create vector** option is selected and if this port is more than one signal wide, select the Create vector option and indicate the range of the vector (4).
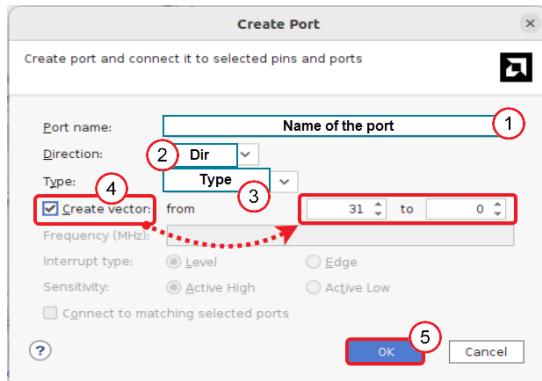
Figure 90: Creating a Port

1-1-7. Click **OK** (5).

Making a Pin or Interface External

There are several mechanisms for making a pin or interface external. The easiest way is described here.

1-1. **Make *your ports and/or interfaces* external to the block design.**

1-1-1. Locate the IP block containing the pin or interface you want to make external.

Right-click the pin or interface, making sure that just the pin of interest is highlighted.

1-1-2. Select **Make External**.
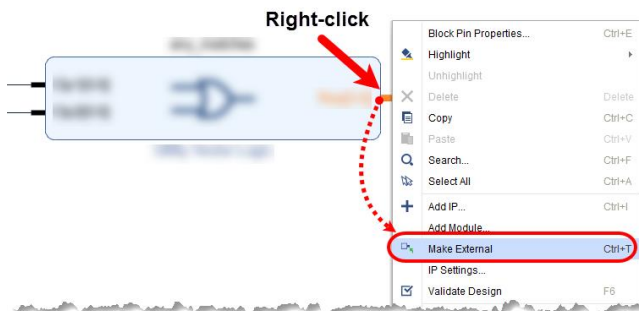
Figure 91: Making a Port or Interface External

A port is created, and a net is generated to connect it to the pin or interface that you specified.

AMD
together we advance_

Running Connection Automation

1-1. **Use Connection Automation to make connections to *the desired piece of IP*.**

1-1-1. Click **Run Connection Automation** in the Design tab information bar.
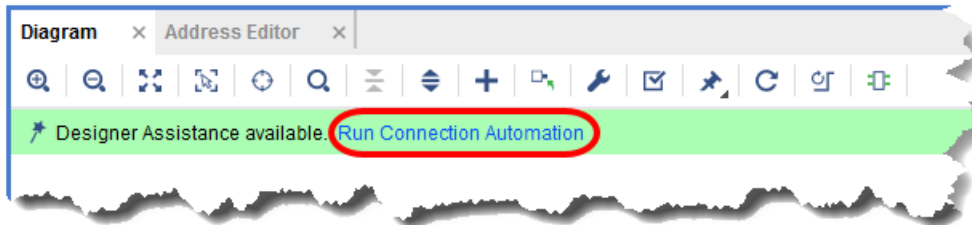
Figure 92: Running Connection Automation

This opens a Run Connection Automation dialog box listing all the IP blocks currently in the design that can have Designer Assistance run on it. IPs are listed in a hierarchy on the left where each child node of an IP node represents an interface that is eligible for automation. Any options associated with an automation will be shown on the right whenever an interface node is selected on the left.

1-1-2. Click the **Expand All** icon (⇕) above the list of IPs in the left-hand panel so that you have a full view of all the IP available for connection automation.

1-1-3. Check **your interface ports**, on which Connection Automation will be run.

Notice how options appear on the right after the interface node is selected. In some cases, you want to leave all automation options at their defaults but, in other cases, it can be useful to customize the options here. For example, several automations allow you to select which clock or AXI master will be automatically connected to the IP.
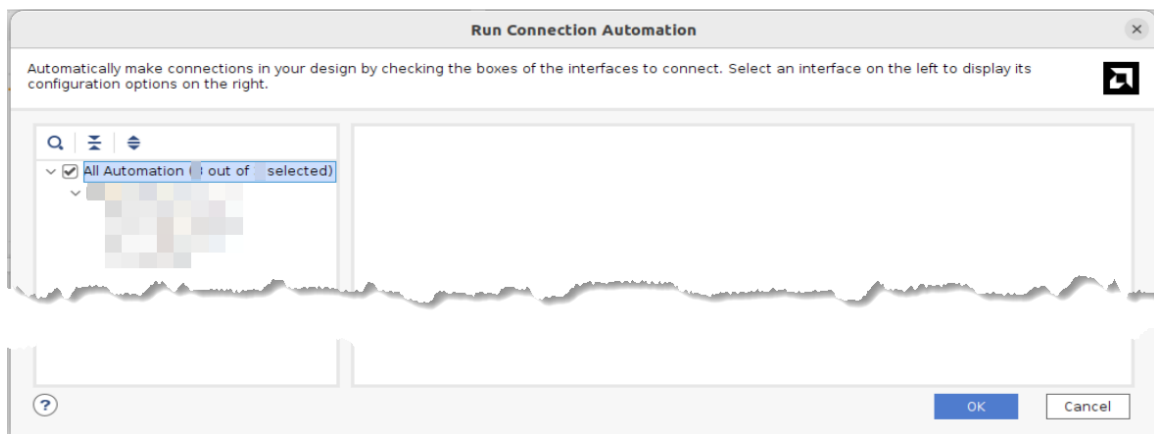
Figure 93: Generic Run Connection Automation Wizard

1-1-4. Click **OK** to run the selected automation.

Connection Automation is fully capable of performing multiple connections—here a single connection was illustrated to show the basics of the capability.

Running Block Automation

1-1.   Use Block Automation to automate the configuration of recently instantiated IP.

1-1-1. Click **Run Block Automation** from the Designer Assistance information bar.
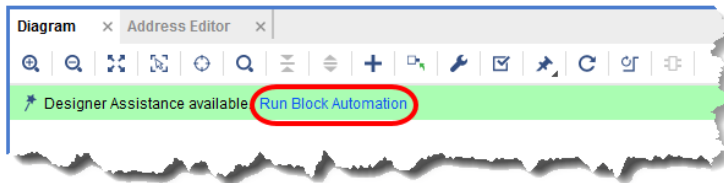
Figure 94: Designer Assistance - Block Automation (Run Connection Automation May be Absent)

This opens a Run Block Automation dialog box listing all the IP currently in the design eligible for block automation. IPs are listed in a hierarchy on the left and any options associated with a particular automation will be shown in the right pane whenever an IP instance is selected in the left pane.

1-1-2. Click the **Expand All** icon (⬍) to ensure that the list of available automations is fully visible (1).

1-1-3. Select either the top-level **All Automation** check box or individual blocks as listed below.

Unless otherwise indicated within the block list, maintain default automation options for all blocks.

o   Blocks: *the desired piece of IP*

1-1-4. Deselect **Apply Board Preset** when using Zynq devices, as this will undo any work that you have done in the processor's Re-customization Wizard (2).
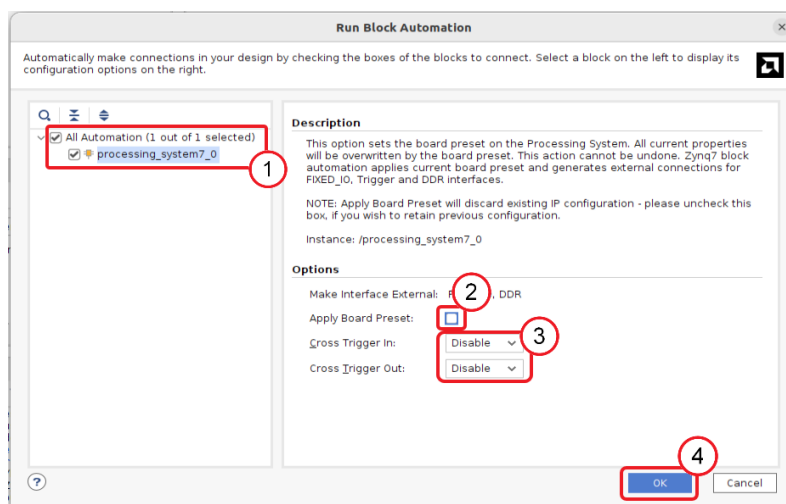
Figure 95: Run Block Automation Dialog Box

1-1-5. Click **OK** to run the selected automation (3).

AMD
together we advance_

Locating Objects in the Board Design

Some designs can become quite large and complex. When "Regenerate Layout" is run, how can you find various IP blocks, nets, or ports/interfaces? This instruction reviews the use of the design hierarchy in locating objects.

1-1. Locate the object.

1-1-1. Locate the **Design** tab next to the Sources tab.

This is where all the elements in a design are located. Elements are organized into external interfaces, interface connections, ports, nets, hierarchical blocks, and IP.
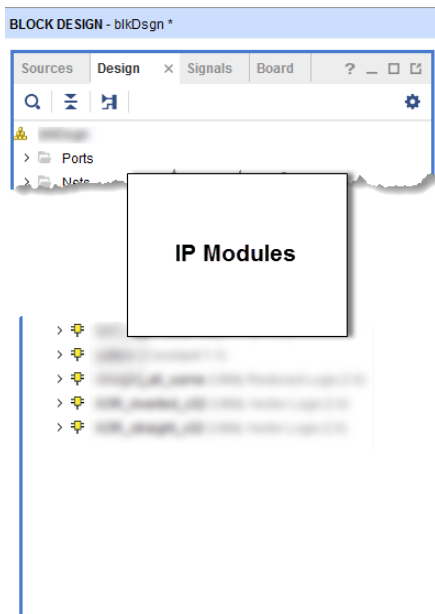
Figure 96: Locating the Design Hierarchy View

1-1-2. Expand the branch corresponding to the type of element you are looking for (hierarchy block, port, etc.)

If you are looking for a specific port or interface on an IP block, you can expand that IP block. Hierarchy blocks contain their own hierarchy of elements.
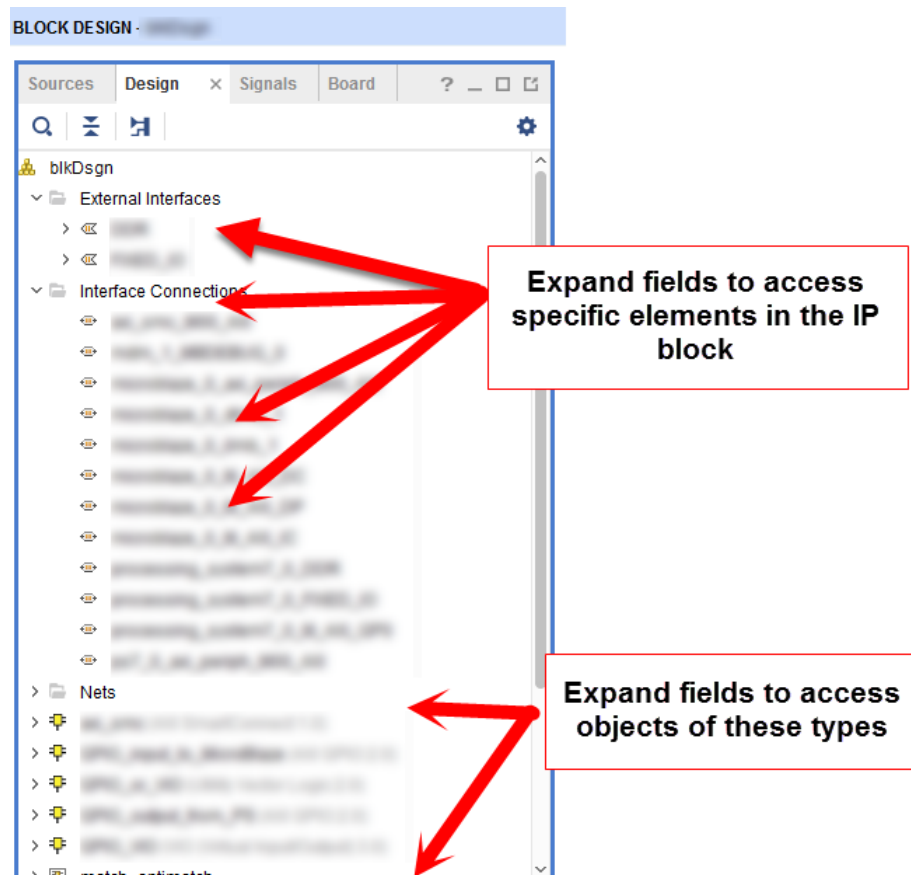
Figure 97: Expanding Branches in the Design Hierarchy Tree

1-1-3. Click the specific element or elements that you want to locate.

The Schematic view will highlight (select) that item.

Mapping Peripheral Addresses

Ensuring that each peripheral has its own memory location is a key aspect of getting the hardware and software to work together.

1-1.   Ensure that all peripherals have an address assigned to them.

1-1-1. Select the **Address Editor** tab (1).

1-1-2. Click the **Expand** icon (⬍) to expand all the entries in the window (2).

1-1-3. Search for any field containing the text "Unmapped Slaves" (3).
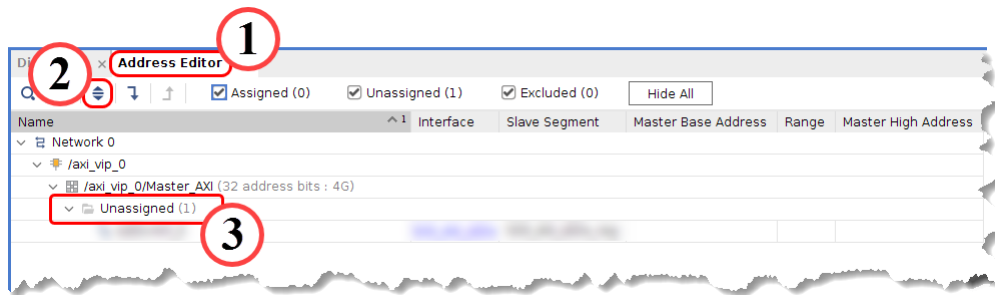
Figure 98: Verifying Peripheral Addresses

This field will appear *only once for each AXI tree.*

If there are unmapped devices, continue with the next instruction; otherwise skip the next instruction.

1-2.   Assign addresses for any peripheral that do not yet have an address assigned to them.

1-2-1. Expand the tree until you see the unmapped devices (1).

1-2-2. Right-click the peripheral that you want to assign an address to or select **Unassigned** to assign addresses to all unassigned elements (2).

1-2-3. Select **Assign Address** to assign addresses peripheral by peripheral.

-- OR --

Select **Assign All** to assign addresses to all peripherals that do not have an address assigned to them (3).
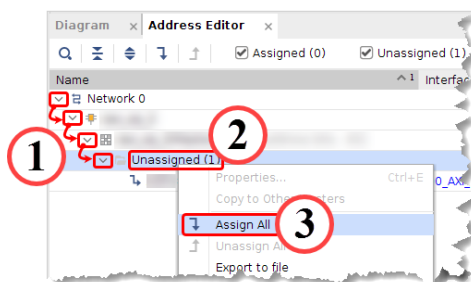
Figure 99: Opening the Dialog Box for Assigning an Address

Generating Output Products

Once a block design is complete, its output products can be generated. The synthesis task automatically runs this phase; however, other operations require the output products, such as exporting a block design.

1-1. **Generate the output products for the items you want.**

1-1-1. Select **the items you want** to generate the output products.

It may be necessary to expand the hierarchy in the Sources window to locate the object(s) of interest.

1-1-2. Right-click the items and select **Generate Output Products** from the context menu.
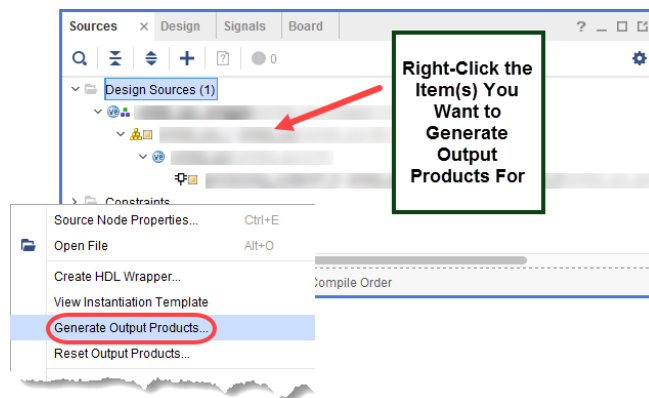
Figure 100: Selecting the Items to Generate Output Products For

The Generate Output Products dialog box opens, listing which output products will be generated.

How the design will be synthesized is up to the user. *Global* means that all the IP in the design (along with the user code) is synthesized as a whole. *Out of context per IP* will synthesize and generate a netlist for each piece of IP. *Out of context per Block Design* will synthesize only the contents of the selected block design.

1-1-3. Select the **Global** synthesis option so that the complete design—the block design, its IP, and the user code (including the wrapper)—will be generated as a single module (1).

If you are changing the settings from out of context to global, click **Apply**.

Note: Out of context is helpful when developing large designs with many blocks. Since each block is synthesized independently, if that block is not changed, then it is not resynthesized, which can offer significant time savings. Contrast this with the Global synthesis option in which all aspects of the design will be resynthesized if any changes are detected.

AMD
together we advance_
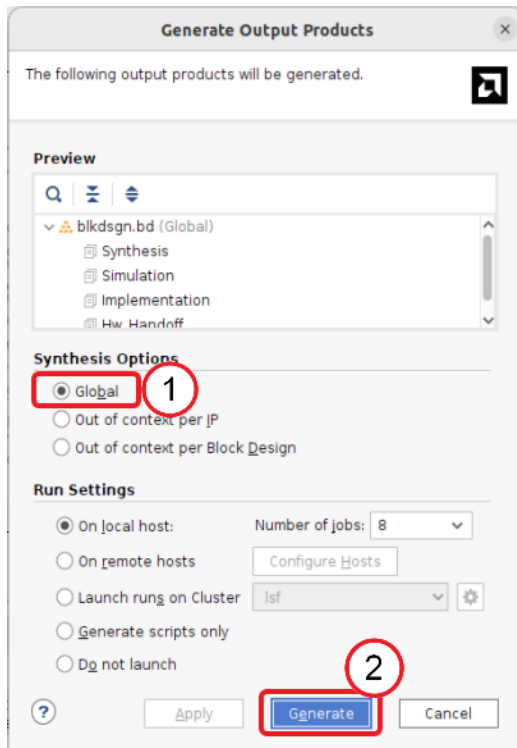
Figure 101: Managing the Output Products

The Run Settings specify the number of processors on your development machine that you would like to run the task on. Generally, this is set to the maximum number of processors.

1-1-4. Click **Generate** to start the generation process (2).

When generation completes, a success dialog box opens.
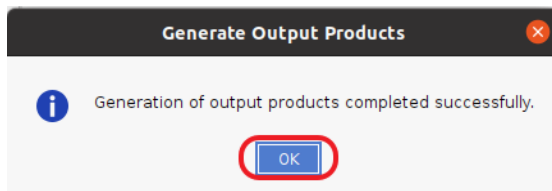
Figure 102: Successful Output Products Generation

1-1-5. Click **OK** to close the Generate Output Products dialog box.

Creating a Hierarchical Block in a Block Design

Hierarchy blocks are useful for grouping similar logic together to simplify cluttered block designs.

1-1. **Create and name a hierarchy block.**

There are two processes involved in creating a hierarchy block. One is the creation of a hierarchy block; the other is the addition of IP to the block.

These two tasks can be performed in either order:

- **Select the IP to be included in the hierarchy block and create a hierarchy block around that IP**

- **Create an empty hierarchy block and add IP to it.**

These processes are not mutually exclusive because once a hierarchy block is created, additional IP can be added to it regardless of how the hierarchy block was initially created.

The following describes how to create an empty hierarchy block.

1-1-1. Right-click in the block diagram.

1-1-2. Select **Create Hierarchy** from the context menu.

The Create Hierarchy dialog box allows you to specify the name for the hierarchy.

1-1-3. Enter **the name of your hierarchy** in the Cell name field.
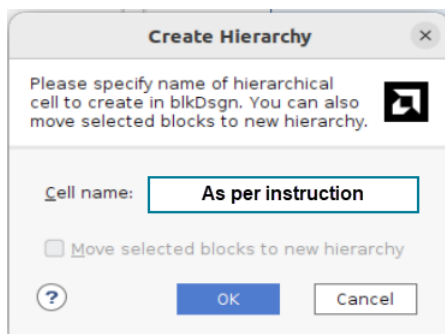
Figure 103: Creating a Hierarchy

1-1-4. Click **OK** to create an empty hierarchy.

Note that if you were to select the IP from the canvas and/or the Design view, then follow the above tasks—you will create the hierarchy block with the IP already included in it.

AMD
together we advance_

Adding IP to a Hierarchy Block

IP can be added to a hierarchy block before or after the hierarchy block is created.

1-1. Add *your IP modules* IP blocks to the hierarchal block.

1-1-1. Select **your IP modules** IP blocks.

This can be done graphically by using <**Ctrl + click**> to select each piece of IP from the canvas as well as selecting the IP from the Design tab.

Note that when you select the IP blocks in the block diagram, they will also be highlighted in the Design window and vice versa.

1-1-2. Drag-and-drop the selected IP into the name of your hierarchy.

This is performed by pressing the <**Ctrl**> key and clicking (but not releasing) one of the IP blocks. Drag the selected item into the name of your hierarchy by moving the cursor of the hierarchy block and then releasing the mouse button.

When moving the IP, you will notice the "plus-sign-in-a-diamond" appear in the IP's outline. This indicates that it will be added to the hierarchy block that it is being dragged over.
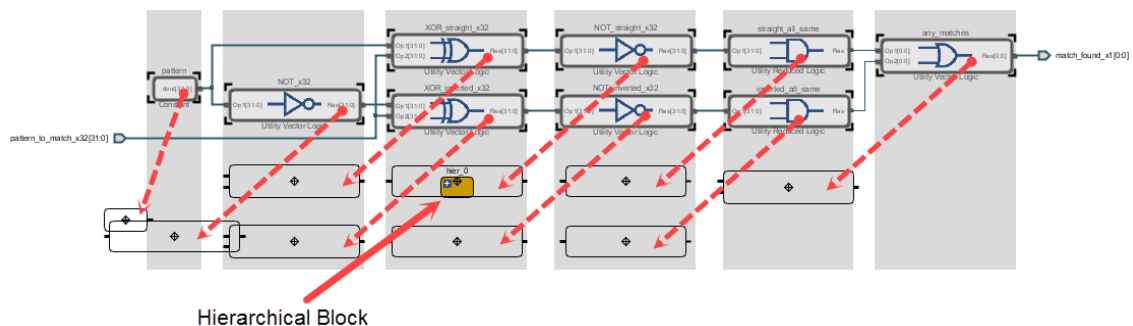
Figure 104: Example of Dragging IP into a Hierarchical Block

Alternatively, before creating the hierarchical block, you can select the IP from the canvas or from the Design tab and then right-click the canvas to open the context menu and select **Create Hierarchy**. The hierarchical block will be created with the selected IPs added to it.

Expanding the Contents of a Hierarchical Block

Sometimes it is easier to understand the diagram when the contents are displayed in another tab, but sometimes you want to see the context of how the contents of a hierarchical block interact with the current level of hierarchy.

1-1. **Expand the name of your hierarchy.**

1-1-1. Locate **the name of your hierarchy**.

This can be done by visually searching through the canvas, or using the alphabetical list of items in the Design tab.

1-1-2. Click the '**+**' sign in the upper-left corner of the hierarchy block.

Figure 105: Expanding a Hierarchy Block

Note: The hierarchy block can be collapsed by clicking the '-' button in the upper-left corner.

AMD
together we advance_

Opening a Hierarchical Block in a New Tab

Once a hierarchy exists, there are times when you might want to see what is happening inside the block. Here's how to access the contents of a hierarchy block.

1-1.   Open the name of your hierarchy in a new tab.

As with many other features of the tool, there are several ways to gain entry into the hierarchy block. First, you must identify which block you want to enter.

1-1-1. Identify the block that you want to enter.

This can be done in one of two ways:

o   Right-click the hierarchy block and select **Open in new tab** or press <**F4**>.
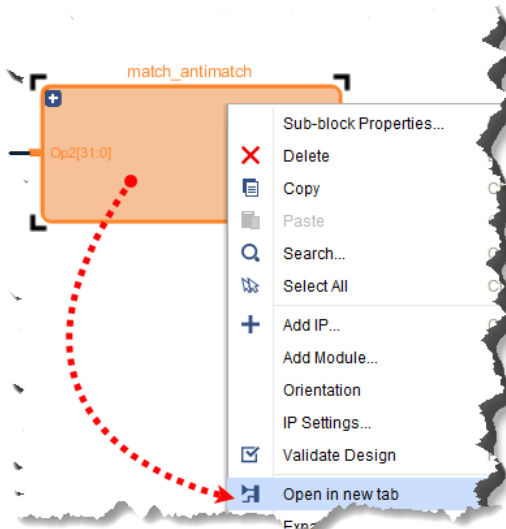
Figure 106: Opening the Hierarchy Block in a New Tab

o   Alternatively, you can use the hierarchy browser, which is part of the Sources tab. When the block design is expanded, the individual pieces of IP are shown. Similarly, the Design tab divides the canvas into ports, nets, IP, and hierarchies. Each hierarchy is further sub-divided into ports, nets, IP, and other hierarchies.

1-1-2. From the Design tab (next to the Sources tab), double-click the **match_antimatch** entry to both expand the ports, nets, and IP within this hierarchy as well as open a new tab with only the contents of the hierarchy present.

1-1-3. Select **the name of your hierarchy** from the list of hierarchies to open the hierarchy in a new tab.

Note: If the block design hierarchy list is not as shown in the figure, click the **Settings** symbol in the extreme top-right corner. Go to the General section and select the **Show Hierarchy** option to enable this capability. The hierarchy can be opened from this location as well as the Design tab.
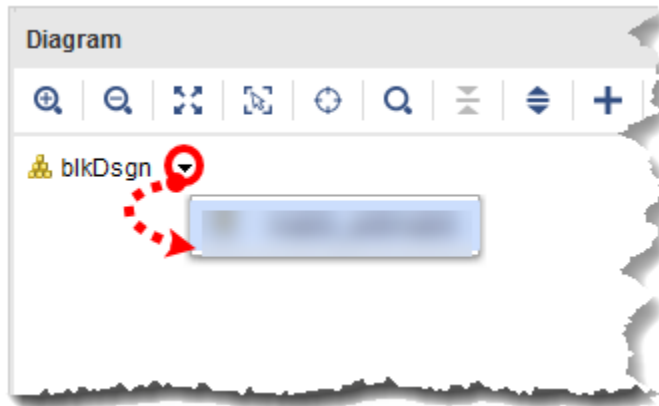
Figure 107: Accessing the Hierarchical Blocks in a Block Diagram

A new tab with the name Diagram - the name of your hierarchy opens.

Regenerating the Layout

The canvas can be some disorganized after manual edits. The IP integrator tool can automatically reconfigure the layout to be ascetically pleasing.

1-1. Regenerate the layout.

1-1-1. Right-click anywhere on the canvas and select **Regenerate Layout**.

Alternatively, you can click the **Regenerate Layout** icon.
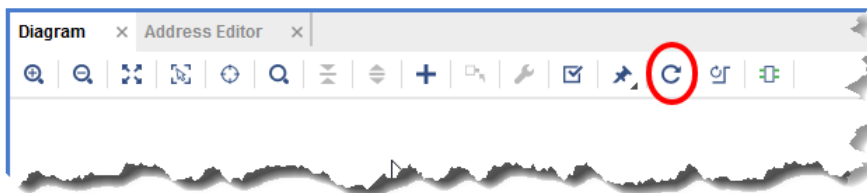
Figure 108: Regenerating the Layout

AMD
together we advance_

Running a Design Rule Check/Design Validation

Each time an IP is connected, a suite of partial design rule checks automatically runs. Design validation is a more comprehensive (but not exhaustive) block design-wide verification and must be run manually.

1-1.   Run a manual design validation.

1-1-1. Select **Tools** > **Validate Design** or press **<F6>**.

Figure 109: Validating the Design

When design validation is complete, results are reported via the Messages tab at the bottom of the Vivado IDE.

Because many messages may be displayed, you have the option of filtering what to view.

If no issues are found, then a dialog box will appear that reports that validation succeeded.

1-1-2. Click **OK** to close the dialog box.

If issues were located:

1-1-3. Select the **Messages** tab to view the items discovered by the validation process.

1-1-4. Filter the results by selecting or deselecting the types of messages you want to see: Error, Infos, or Status messages.

The display is automatically updated.

Hyperlinks are provided to help you quickly locate the issue.
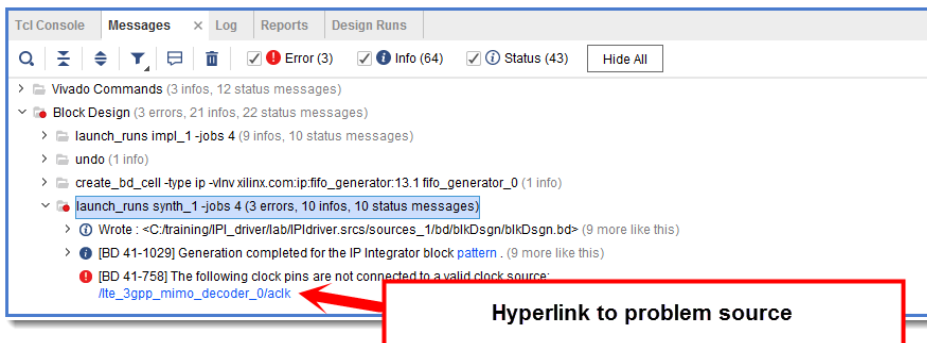
Figure 110: Viewing the Results of the Design Validation

1-1-5. Correct any issues and rerun validation until no more issues are found.

Validating the Block Design

1-1. **Validate the design to catch any connection and address map errors.**

1-1-1. Select **Tools** > **Validate Design** or press **<F6>.**

Any issues are displayed in the Console window.

1-1-2. Click **OK** to close the window.

Saving the Block Design

1-1. **Save the block design.**

1-1-1. Select **File** > **Save Block Design** to save the block design.

Alternatively, you can press **<Ctrl + S>**.
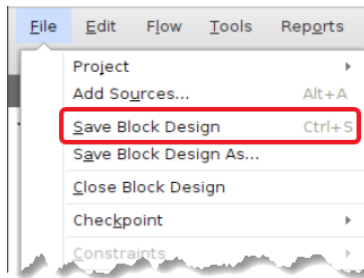
Figure 111: Saving the Block Design

AMD
together we advance_

Generating a Wrapper for a Block Design

While a block design can be directly converted into a structural netlist, block designs are most frequently used as structural modules within a larger design.

Typically, once the design is complete, an HDL wrapper is created that instantiates the design and provides a platform for adding other logic. This wrapper can be automatically generated (and optionally terminated) by the tools, or it can be added manually by the user.

1-1. Create a wrapper for the block design.

1-1-1. Ensure that the Sources tab is selected; if it is not, select it to open the Sources view (1).

While the block diagram can be found under any of the tabs found at the bottom of the pane (Hierarchy, IP Sources, Libraries, or Compile Order), it is shown here from the Hierarchy tab as this is a common practice.

1-1-2. Using the Sources > Hierarchy tab, right-click the block design under the Design Sources node (2).

Notice the icon (🟡) which indicates that this entry is a block design.

This will open a pop-up window to actions that can be taken for this entry.

1-1-3. Select **Create HDL Wrapper** to begin the wrapper creation process (3).
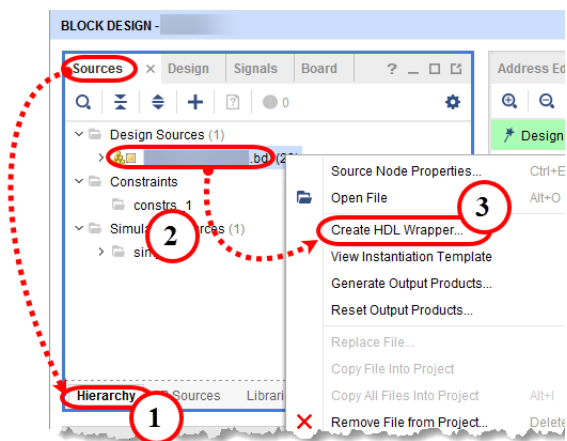
Figure 112: Selecting Create HDL Wrapper

The Create HDL Wrapper dialog box opens, showing a list of options relating to how the wrapper should be handled once generated.

Two choices are presented:

o Generate a wrapper that you can manually edit, which is useful when you have additional logic to add to the top-level wrapper, or

o Generate a wrapper that is managed by the Vivado Design Suite, which is ideal for designs that are fully described by one or more block designs.

1-1-4. Select the option which best matches your needs (typically this is the default option of letting the Vivado Design Suite manage the wrapper).
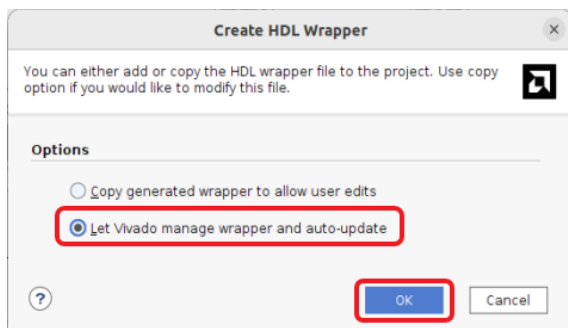
Figure 113: Generating the Wrapper and Copying the Generated File into the Project

This will produce a wrapper file in the target language. You can double-click it to open it in a new editor window. The project hierarchy will automatically be adjusted to reflect the addition of this new file.

Note: Many simulations require Verilog. If this is necessary for your simulation tool, select **Project Manager** > **Settings** > **General** > **Target Language** to cause the tools to generate all sources in Verilog, including the wrapper.

1-1-5. Click **OK** to create the wrapper.

Vivado Elaborated Design Operations

In This Section

Opening the Elaborated Design

1-1. Open the elaborated design.

1-1-1. Click **Open Elaborated Design** under RTL Analysis in the Flow Navigator to elaborate the design.

The elaborated RTL design enables various analysis views, including RTL Netlist, Schematic, and Graphical Hierarchy. These views have a cross-selection feature that allows you to debug and optimize the RTL.

The Elaborate Design dialog box opens.

1-1-2. Click **OK** in the dialog box to open the elaborated design.

AMD
together we advance_

Creating a Schematic on an Elaborated Design

1-1.   Create a Schematic on an Elaborated Design.

1-1-1. In the Flow Navigator, under RTL Analysis > Elaborated Design, click **Schematic**.

The RTL Schematic opens in the main workspace area.

Running UltraFast DRC Checks on the Elaborated Design

1-1.   Run UltraFast™ design methodology DRC checks on the elaborated design.

1-1-1. Click **Report Methodology** under RTL Analysis > Open Elaborated Design in the Flow Navigator.

The Report Methodology dialog box opens.

1-1-2. Click **OK** to run the design methodology checks and find errors or problems in the current design.

Running DRC Checks on the Elaborated Design

1-1.   Run DRC checks on the elaborated design.

1-1-1. Click **Report DRC** under RTL Analysis > Open Elaborated Design in the Flow Navigator.

The Report DRC dialog box opens.

1-1-2. Select **default** in the Vivado Rule Decks section.

1-1-3. Observe the categories in the **Rules** section.

1-1-4. Click **OK** to generate the DRC report.

Closing the Elaborated Design

1-1. Close the elaborated design.

1-1-1. Select **File** > **Close Elaborated Design** to close the elaborated design.

The Confirm Close dialog box opens.
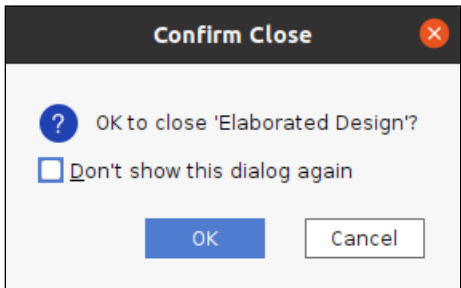
Figure 114: Confirm Close Dialog Box

1-1-2. Click **OK**.

Vivado Simulation Operations

In This Section

Running Behavioral Simulation

1-1. Run behavioral simulation using the Vivado simulator.

1-1-1. Select **Simulation** > **Run Simulation** > **Run Behavioral Simulation** under Simulation from the Flow Navigator.

The simulation window opens, and the simulation runs for the length of time specified in the Simulation Settings (1 us default).

You can alternatively start the Vivado simulator by entering `launch_simulation` in the Tcl command line.

AMD
together we advance_

Running Timing Simulation

1-1. Run behavioral simulation using the Vivado simulator.

1-1-1. Select **Run Simulation** > **Run Post-Implementation Timing Simulation** from the Flow Navigator, under Simulation.

The simulation window opens, and the simulation runs for the length of time specified in the Simulation Settings (1 us default).

Accessing Simulation Settings

1-1. Open the simulation settings.

1-1-1. Select **Settings** in the Flow Navigator, under Project Manager and go to **Simulation**.

The simulation settings open. There are five sub-tabs: Compilation, Elaboration, Simulation, Netlist, and Advanced.
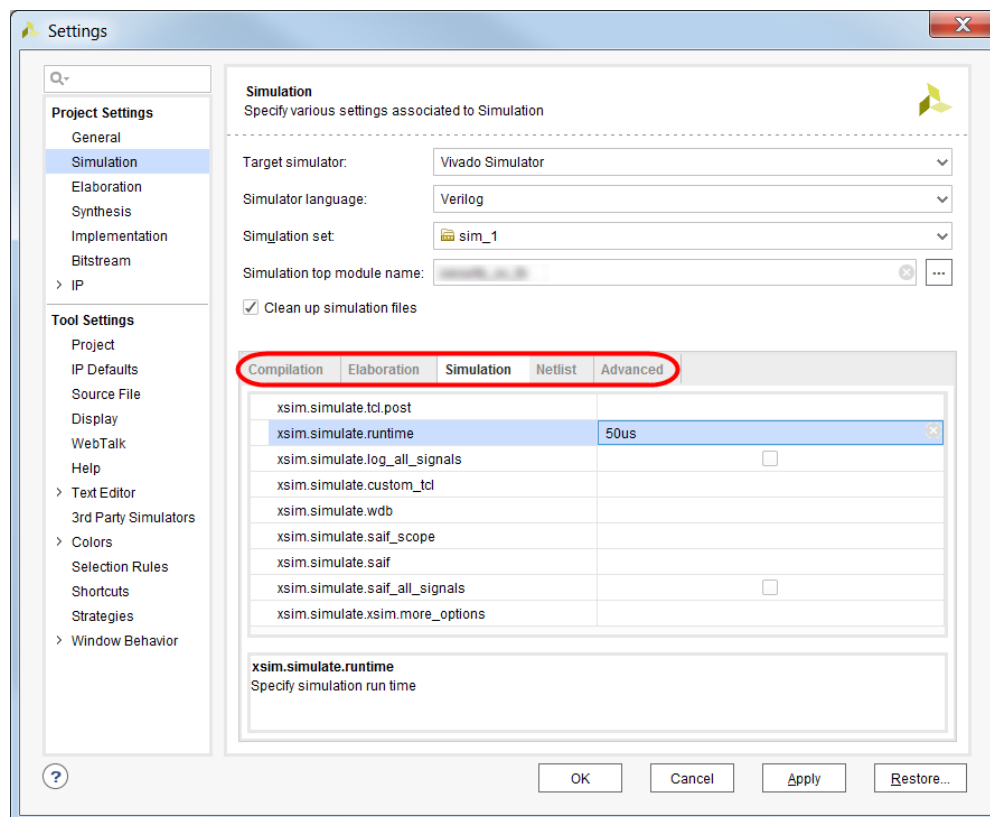
Figure 115: Vivado Simulator Settings

Closing Vivado Simulation

1-1. Close the simulation.

1-1-1. Select **Simulation** in the Flow Navigator, then right-click and select **Close Simulation**.

1-1-2. Click **OK** to close the simulation.

Vivado Synthesis Operations

In This Section

Running Synthesis

1-1. Run synthesis.

1-1-1. Expand **Synthesis** in the Flow Navigator (1).

1-1-2. Click **Run Synthesis** (2).
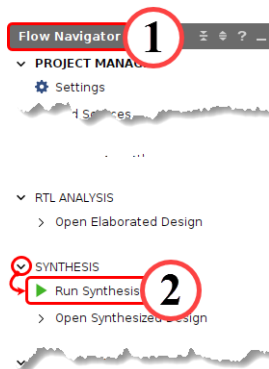
Figure 116: Running Synthesis

Alternatively, you can also press <**F11**>.

The Launch Runs dialog box opens.

AMD
together we advance_

1-1-3. Ensure that the **Launch runs on local host** option is selected (1).

1-1-4. Select the largest values from the *Number of jobs* drop-down list (2).

This allocates the number of processor cores recruited to run synthesis. Generally, the more cores, the faster synthesis completes.
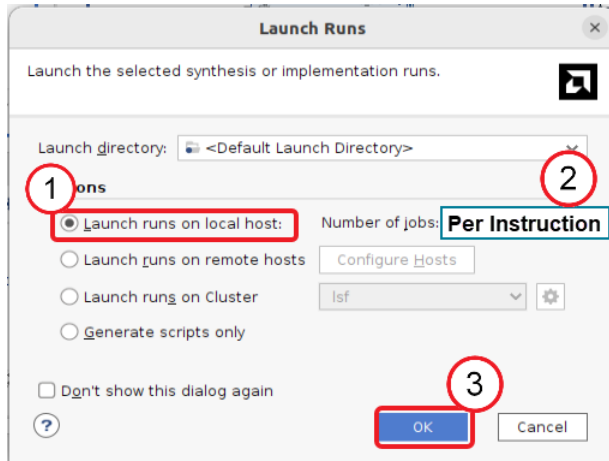
Figure 117: Setting the Launch Run Configuration

1-1-5. Click **OK** to launch the runs (3).

1-1-6. Click **Save** if you are asked to save your files.

Once synthesis completes, you are asked what task you want to perform next: run implementation, open the synthesized design, view reports, or none of the above.

1-1-7. Select **whatever process you want to perform next or Cancel to exit the dialog box** (1).

1-1-8. [Optional] If you are familiar with accessing these various capabilities, you can disable this dialog box from appearing again by selecting **Don't show this dialog again** (2).
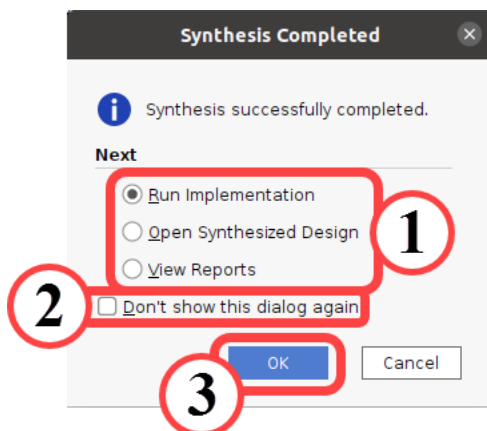
Figure 118: Selecting Post-Synthesis Options

1-1-9. Click **OK** to take the action you just selected (3) or click **Cancel** to simply close the dialog box.

Opening the Synthesized Design

1-1.   Open the synthesized design.

1-1-1. Click **Open Synthesized Design** under Synthesis in the Flow Navigator.

Alternatively, you can select **Flow** > **Open Synthesized Design**.
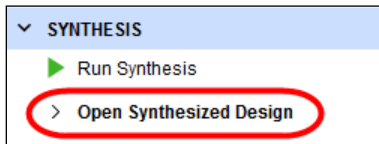
Figure 119: Opening the Synthesized Design

Generating a Utilization Report

1-1.   Generate a Utilization report.

1-1-1. Click **Report Utilization** under Synthesized Design in the Flow Navigator.

Alternatively, you can select **Reports** > **Report Utilization**.

Generating a Clocks Networks Report

1-1.   Generate a Clocks Networks report.

1-1-1. Click **Report Clock Networks** in the Flow Navigator under Synthesized Design.

Alternatively, you can select **Reports** > **Timing** > **Report Clock Networks**.

The Report Clock Networks dialog box opens.

1-1-2. Click **OK**.

Running a DRC Report on the Synthesized Design

1-1.   Run a DRC report on the synthesized design to check for any violations.

1-1-1. Click **Report DRC** under Synthesis > Synthesized Design in the Flow Navigator.

The Report DRC dialog box opens.

1-1-2. Select **default** in the Rule Decks section of the Report DRC dialog box.

1-1-3. Click **OK** to start the default DRC checks on the synthesized design.

AMD
together we advance_

Opening the Timing Constraints Window after Synthesis

1-1. Open the Timing Constraints window.

1-1-1. Select **Window** > **Timing Constraints**.

This window can also be opened by clicking **Edit Timing Constraints** under Synthesized Design in the Flow Navigator.

The Timing Constraints window opens in the main workspace area.

Generating a Timing Summary Report on the Synthesized Design

The timing summary command is accessible in the Vivado Design Suite GUI once the synthesized design is opened in the Vivado IDE. If the synthesized design is not opened in the GUI and you do not remember how to open the design, refer to the "Opening the Synthesized Design" section in the *Lab Reference Guide*.

1-1. Generate a Timing Summary report.

1-1-1. Click **Report Timing Summary** under **Synthesis** > **Open Synthesized Design** in the Flow Navigator.

The Report Timing Summary dialog box opens.
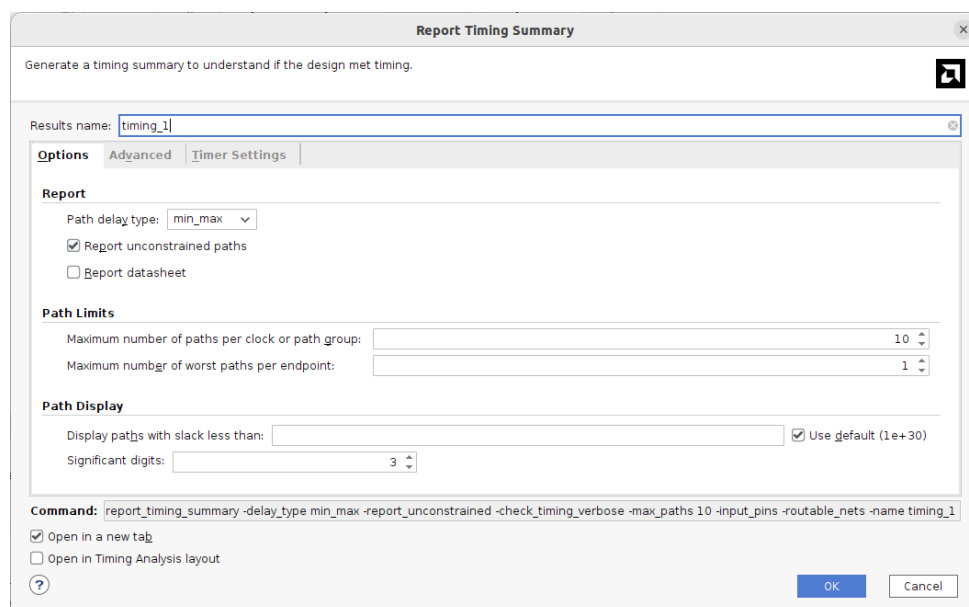
Figure 120: Report Timing Summary Dialog Box

1-1-2. Click **OK** to generate the Timing Summary report using the default settings.

The Design Timing Summary window opens at the bottom in the Timing tab.

Running Simultaneous Switching Noise (SSN) Analysis

Simultaneous switching noise (SSN) analysis can be performed to help identify potential signal integrity issues.

1-1. Run SSN on the design.

1-1-1. From the Flow Navigator, click **Report Noise** under Synthesis > Synthesized Design.

1-1-2. Click **OK** in the Report Noise dialog box.

The Noise Report window opens at the bottom in the Noise tab.

Generating a Clock Interaction Report on the Synthesized Design

The clock interaction command is accessible in the Vivado Design Suite GUI once the synthesized design is opened in the Vivado IDE. If the synthesized design is not opened in the GUI and you do not remember how to open the synthesized design, refer to the "Opening the Synthesized Design" section in the *Lab Reference Guide*.

1-1. Generate a Clock Interaction Report.

1-1-1. In the Flow Navigator, under Synthesis > Synthesized Design, select **Report Clock Interaction**.

Alternatively, you can select **Tools** > **Timing** > **Report Clock Interaction**.

The Report Clock Interaction dialog box opens.

1-1-2. Click **OK** with the default settings.

The Clock Interaction Report opens in GUI mode.

Closing the Synthesized Design

1-1. Close the synthesized design.

1-1-1. Select **File** > **Close Synthesized Design**.

The Confirm Close dialog box may open.
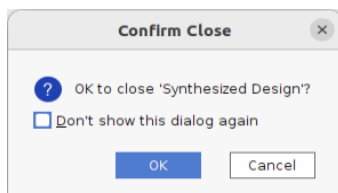
Figure 121: Confirm Close Dialog Box

AMD
together we advance_

1-1-2. Click **OK** to close the synthesized design if the Confirm Close dialog box opens.

Alternatively, you can also close the synthesized design by clicking the **X** in the Synthesized Design status bar at the top or entering the Tcl command `close_design` in the Tcl Console.

Vivado Implementation Operations

In This Section

Implementing a Design

1-1. Implement the design.

1-1-1. From the Flow Navigator, under Implementation, click **Run Implementation** to run all operations necessary to generate the bitstream.

This includes synthesis if the netlist has not already been generated and the implementation processes if not already performed.
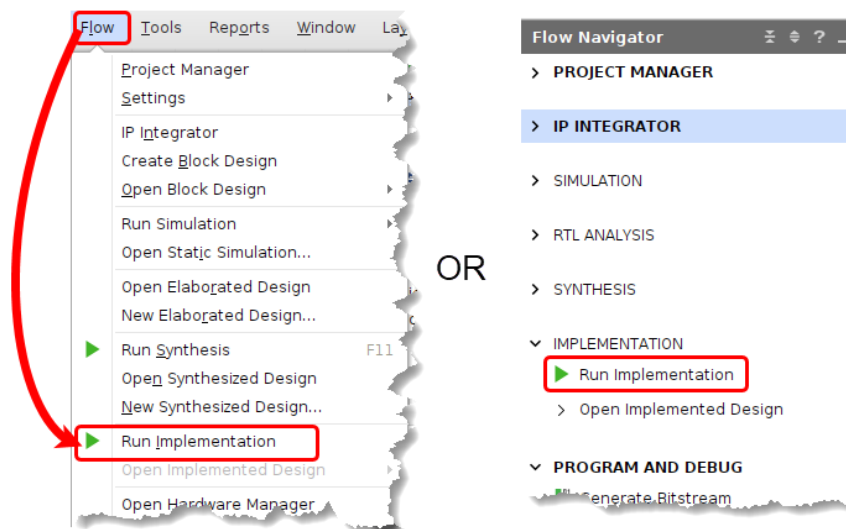
Figure 122: Two Ways to Run Implementation

The Launch Runs dialog box opens. Here is where you can specify where the synthesis or implementation activity will occur and how many resources are assigned to the task.

1-1-2. Select **whatever process you wish to perform next or cancel to exit the dialog** (1).

For the *Launch runs on local host* option, the number of jobs represents the number of processing cores that can be assigned to this task, which is platform specific.
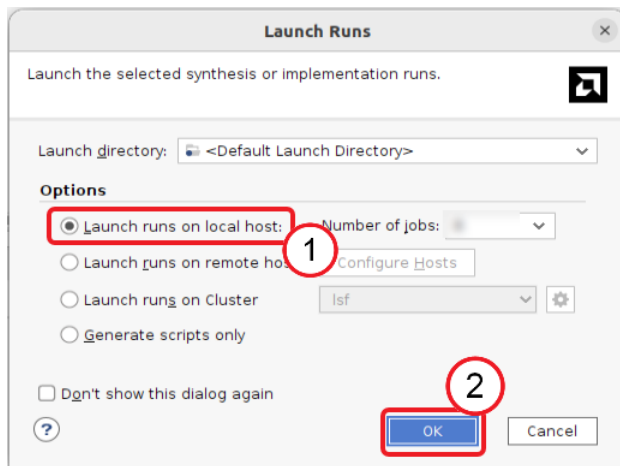
Figure 123: Selecting Run Options and Launching the Task

1-1-3. Click **OK** to begin the implementation process (2).

1-1-4. Click **OK** if any critical warning messages dialog boxes appear.

You will need to determine if the critical warnings need to be addressed immediately or if they can be ignored.

When implementation is complete, the Implementation Complete dialog box opens.

When the design finishes the implementation, the Implementation Completed dialog box opens. You can view reports to review what was created or open the hardware manager to program the device or view the information from the Vivado analyzer tool. You can also simply cancel to perform neither of these tasks. These tasks, and others, are still available through the Flow Navigator.

1-1-5. From the dialog box, select **to your desired settings** (1).
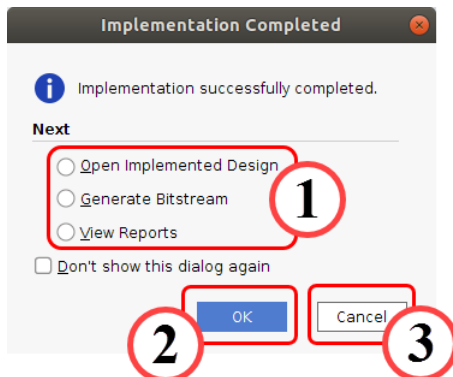
Figure 124: Selecting which Activity to Perform after Implementation

1-1-6. Click **OK** to open the selected item or **Cancel** to perform none of the listed tasks (2, 3).

AMD
together we advance_

Generating a Utilization Report on the Implemented Design

1-1. Generate a Utilization report on the implemented design.

1-1-1. Click **Report Utilization** under Open Implemented Design in the Flow Navigator.

Alternatively, you can select **Reports** > **Report Utilization**.

The Report Utilization dialog box opens.
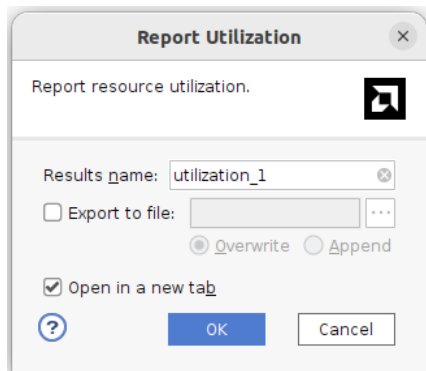
Figure 125: Report Utilization Dialog Box

1-1-2. Click **OK** using the default settings.

The Design Utilization Summary window opens at the bottom in the Utilization tab.

Note that the device resources are listed in a hierarchical format for each RTL design module.

Opening the Implemented Design

1-1. Open the implemented design.

1-1-1. Click **Open Implemented Design** under Implementation in the Flow Navigator.

Alternatively, you can select **Flow** > **Open Implemented Design**.
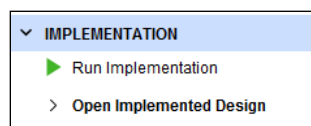
Figure 126: Opening the Implemented Design

Note that the Timing Summary report is automatically generated in read only mode when the implemented design is opened.

Generating a Clock Interaction Report on the Implemented Design

The clock interaction command is accessible in the Vivado Design Suite GUI once the implemented design is opened in the Vivado IDE. If the implemented design is not opened in the GUI and you do not remember how to open the implemented design, refer to "Opening the Implemented Design" section in the *Lab Reference Guide*.

1-1. Generate a Clock Interaction Report.

1-1-1. In the Flow Navigator, under Implementation > Implemented Design, select **Report Clock Interaction**.

Alternatively, you select **Tools** > **Timing** > **Report Clock Interaction**.

The Report Clock Interaction dialog box opens

1-1-2. Click **OK** with the default settings.

The Clock Interaction Report opens in GUI mode.

Closing the Implemented Design

1-1. Close the implemented design.

1-1-1. Select **File** > **Close Implemented Design** to close the implemented design.

The Confirm Close dialog box opens.
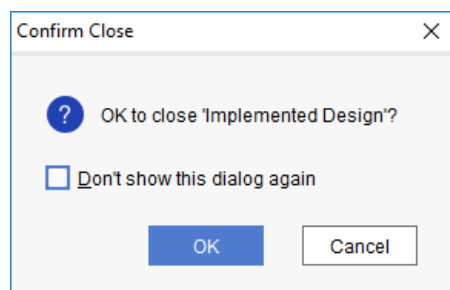
Figure 127: Confirm Close Dialog Box

1-1-2. Click **OK** to close the implemented design.

Alternatively, you can also close the implemented design by selecting **Flow** > **Close Implemented Design** or clicking the **X** in the Implemented Design status bar at the top.

AMD
together we advance_

Generating Clock Networks

1-1. Generate a Clocks Networks report.

1-1-1. In the Flow Navigator, under Implemented Design, click **Report Clock Networks**.

Alternatively, you can select **Tools** > **Report** > **Report Clock Networks**.

Generating the Bitstream

For users that have a few hours, it is time to generate the bitstream. If you are running in class or on CloudShare, only review this and the next step.

> **Note:** Do not perform this step as generating the bitstream will take approximately 3-4 hours depending on your system configuration. The instructions below are provided for review on what the necessary actions are.
>
> For CloudShare users, if you are going to perform this step, you should do so in your local environment, not in CloudShare. This is because the memory available in CloudShare is 12 GB, and 16 GB is the minimum requirement. Generating the bitstream will still take approximately 3-4 hours.

1-1. Generate the bitstream.

1-1-1. Locate the **Generate Bitstream** entry under Program and Debug in the Flow Navigator.
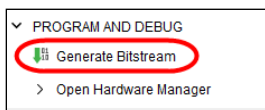
Figure 128: Generate Bitstream in the Flow Navigator

1-1-2. Click **Generate Bitstream** to start the bitstream generation.

If neither synthesis nor implementation has been run, you will see a "No Implementation Results available" message.

1-1-3. Click **Yes** to continue the process as both synthesis and implementation will be run during the generation of the bitstream.

1-1-4. When the Launch Runs window opens, ensure that the **Launch runs on local host** option is selected.

1-1-5. Set the number of jobs to the highest number available.

This specifies how many CPU cores are allocated to this task.

1-1-6. Click **OK** to begin the bitstream generation process.

Track the progress using the status indicator in the upper-right corner of the workspace as well as in the design runs console. Successful completion of the bitstream is indicated with a message in the Console tab.

Errors are reported through pop-up messages.

Vivado Tcl Operations

In This Section

Running a Tcl Script from within the Vivado Design Suite

The Vivado Design Suite offers both GUI and scripted control. Tcl provides scripted control. These Tcl commands can be entered directly into the tool one at a time (or block copied), or an entire Tcl script can be loaded and executed (sourced).

1-1. Run a Tcl script.

1-1-1. Select the **Tcl Console** tab to view the console.

> **Hint:** If the Tcl Console is minimized, clicking the tab will partially expand the view.

1-1-2. Locate the Tcl command line entry.

The command line entry can be found either on the Welcome page before a project is opened, or once a project has been opened.

From the Welcome screen:
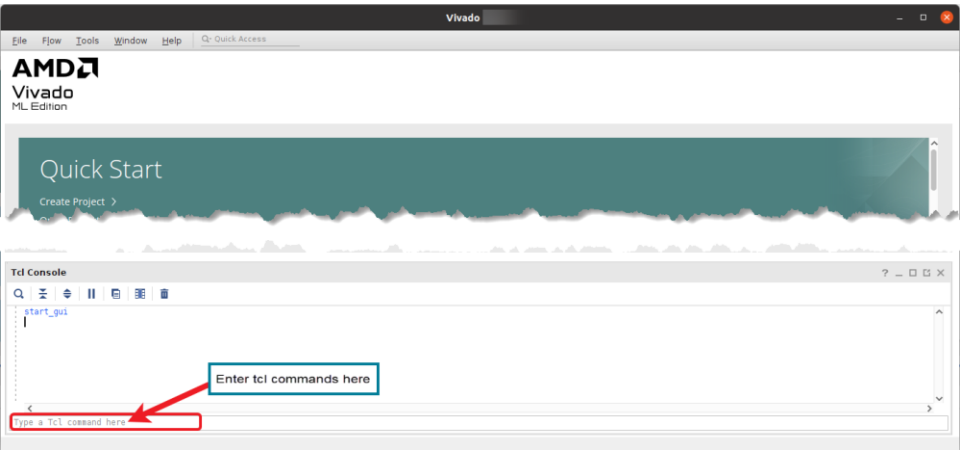
Figure 129: Accessing the Tcl Console from the Getting Started Page

You can always click the Tcl tab to maximize/restore it.

AMD
together we advance_

From the Tcl Console tab in an open project:
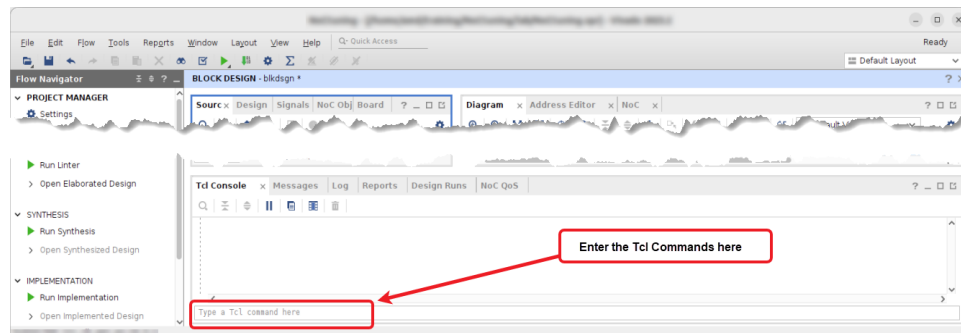
Figure 130: Entering Commands into the Tcl Console from an Open Project

Initially, the directory location begins within the tool installation directory.

1-1-3. Enter the following command to change the current working directory to where the Tcl script is located:

```
cd $::env(TRAINING_PATH)/<the topic cluster name>/support
```

Note: The Tcl proc `env` reaches out to the operating system and returns the value of the environment variable given by `TRAINING_PATH`. The $ indicates that the value of that variable should be returned and the two colons (::) indicate from which namespace the information should be pulled. Because there is nothing in front of the ::, the global or base layer of the namespace is to be used.

1-1-4. Verify that you are now where you want to be by entering the following into the Tcl command line:

```
pwd
```

This should show that you are in the `support` directory under the `<the topic cluster name>` directory in the `training` directory.

Note that the `training` directory can be anywhere in the system. What is essential is that you are currently working from the `support` directory under `<the topic cluster name>`.

1-1-5. Run the following Tcl command to run the helper Tcl script for this lab:

```
source your Tcl script.tcl
```

The Tcl script is run as though you typed each command included in the Tcl script into the Tcl command line. You can follow the execution of the script and monitor for any errors or warnings in the Tcl Console.

Clearing the Tcl Console

1-1. **Clear the Tcl Console window.**

This helps to make it clear how each command behaves.

1-1-1. From the Tcl Console, click the **Trash Can** icon in the Tcl Console toolbar.
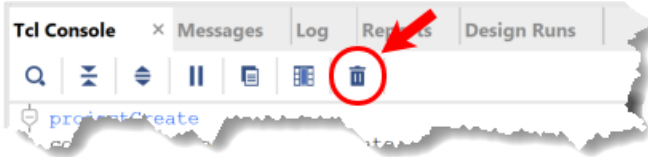
Figure 131: Clearing the Tcl Console

A dialog box opens, asking you to confirm the clearing of the Tcl console.

1-1-2. Click **Yes** to clear the Tcl Console.

Generating a Timing Summary Report with the Tcl Interface

The timing summary command is accessible in the Vivado Design Suite GUI once the synthesized design or implemented design is opened in the Vivado IDE. If the synthesized/implemented design is not opened in the GUI and you do not remember how to open the design, refer to the "Opening the Synthesized Design" or "Opening the Implemented Design" section in the *Lab Reference Guide*.

1-1. **Generate a timing report with the Tcl interface.**

1-1-1. In the Tcl Console at the bottom of the GUI, enter the following command:

```
report_timing_summary -delay_type max -report_unconstrained -
check_timing_verbose -max_paths 10 -input_pins
```

This displays the 10 worst paths per clock or per group and reports the unconstrained paths as well.

1-1-2. If you want to save the timing results to a text file, enter the following command:

```
report_timing_summary -delay_type max -report_unconstrained -
check_timing_verbose -max_paths 10 -input_pins -file
C:/<file_location>/<file_name>.txt
```

AMD
together we advance_

1-1-3. If you want to see the timing results in the GUI use the `-name` option:

```
report_timing_summary -delay_type max -report_unconstrained -
check_timing_verbose -max_paths 10 -input_pins -name timing_1 -
file C:/<file location>/<file_name>.txt -name timing_1
```

- o `-delay_type` – Values are min, max and   min_max (sets the type of analysis to be run. For synthesized designs, only max delay analysis).
- o `-report_unconstrained` – Reports the unconstrained paths in the report.
- o `-check_timing_verbose` – Reports the missing timing constraints in the report.
- o `-max_paths` – Number of worst paths to be displayed in the timing report.
- o `-input_pins` – Shows the inputs pins the timing report.
- o `-file` – Writes the output timing results to a file to the specified location.
- o `-name` – To see the results in GUI mode.

Using the llength Command to Obtain the Number of Paths Between Two Domains

The `llength` Tcl command can be used to return the length of the list (elements).

For example:

`set Vivado [list 1 2 3]` --> returns the 1 2 3

`length $Vivado` --> returns 3

1-1. **Use the `llength` command to obtain the number of paths that exists between two domains.**

1-1-1. In the Tcl Console, generate the list of timing paths that exist between two domains with `get_timing_paths`:

```
get_timing_paths -from [get_clocks <clock_name>] -to [get_clocks
<clock_name>] -max_paths 100
```

This command will display the timing paths that exist between the two clock domains.

1-1-2. Use the `llength` command to return the number of paths that exist between the two clock domains:

```
llength [get_timing_paths -from [get_clocks <clock_name>] -to
[get_clocks <clock_name>] -max_paths 100]
```

Using the join Command to Join All Timing Paths Between Two Domains

The `join` Tcl command can be used to join two strings or characters with a new line or another character.

For example:

```
set Vivado [1 2 3]
```

  returns --> 1 2 3

```
join $Vivado  \n
```

returns   -->   1

                      2

                      3

1-1.   Use the `join` command to join the timing paths between two domains with a new line.

1-1-1. In the Tcl Console, generate the list of timing paths that exist between two domains with `get_timing_paths`:

```
get_timing_paths -from [get_clocks <clock_name>] -to [get_clocks
<clock_name>] -max_paths 100
```

This command will display the timing paths that exist between the two clock domains.

1-1-2. Join the timing paths with a new line.

```
join [get_timing_paths -from [get_clocks <clock_name>] -to
[get_clocks <clock_name>] -max_paths 100]  \n
```

Note that `\n` adds a new line for each timing path.

AMD
together we advance_

Building a Custom Timing Report with the Tcl Interface

The `report_timing` command is accessible in the Vivado Design Suite GUI once the synthesized design or implemented design is opened in the Vivado IDE. If the synthesized/implemented design is not opened in the GUI and you do not remember how to open the design, refer to the "Opening the Synthesized Design" or "Opening the Implemented Design" section in the *Lab Reference Guide*.

The `report_timing` command displays specific timing paths only.

1-1.   Generate a custom timing report with Tcl interface.

1-1-1. In the Tcl Console at the bottom of the GUI, enter the following command:

```
report_timing -from [startup points] -to [end points] -delay_type
min_max -max_paths 10 -sort_by group -input_pins -name timing_1
```

- o   `-from` – Startup points such as sequential components, F/F pins, etc.

- o   `-to` – End points such as data inputs of sequential components, etc.

- o   `-delay_type` – Values are min, max and   min_max (sets the type of analysis to be run. For synthesized designs, only max delay analysis).

- o   `-report_unconstrained` – Reports the unconstrained paths in the report.

- o   `-check_timing_verbose` – Reports the missing timing constraints in the report.

- o   `-max_paths` – Number of worst paths to be displayed in the timing report.

- o   `-input_pins` – Shows the inputs pins the timing report.

- o   `-file` – Writes the output timing results to a file to the specified location.

- o   `-name` – To see the results in GUI mode.

For more information about the options that can be used with `report_timing`, type `report_timing -help` in the Tcl Console.

Vivado Hardware Session Operations

In This Section

Implementing a Design and Generating a Bitstream

1-1. **Generate the implemented design and bitstream.**

The Vivado Design Suite runs the equivalent of a "make" file that understands dependencies. If synthesis or implementation has not been run, and bitstream generation is requested, then the tools are smart enough to synthesize whatever modules need to be synthesized and implement the design before generating the bitstream.

1-1-1. Click **Generate Bitstream** from the Flow Navigator under Program and Debug to run all operations necessary to generate the bitstream.
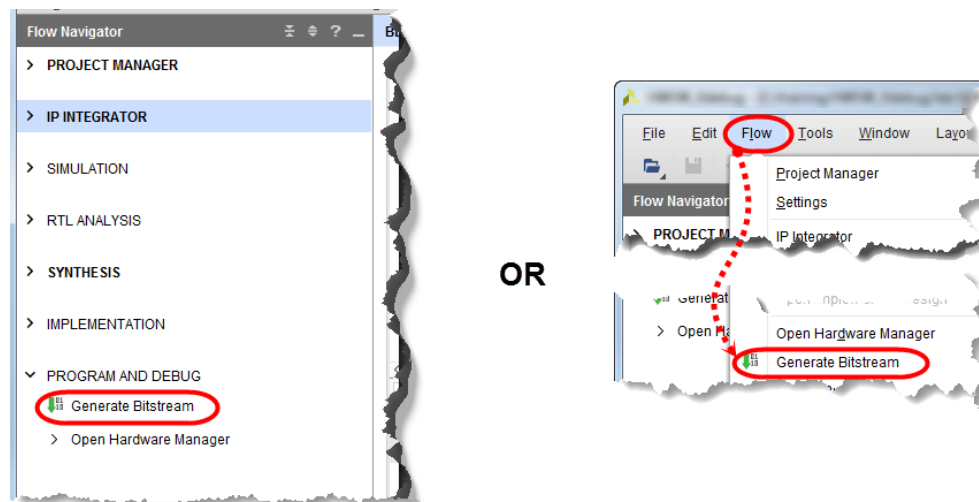
Figure 132: Launching the Generate Bitstream Process

1-1-2. If you are presented with a warning that there aren't any implementation results available, click **Yes**.
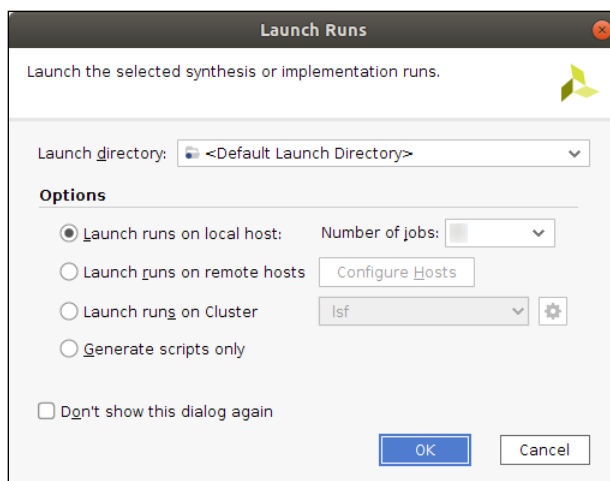
Figure 133: Launch Runs Dialog Box

The default settings are sufficient for this run.

AMD
together we advance_

1-1-3. Click **OK** to proceed with synthesis, implementation, and bitstream generation.

1-1-4. Address any critical warning messages dialog boxes that appear and continue the bitstream generation operation.

You will need to determine if the critical warnings need to be addressed immediately or if they can be ignored.

When implementation is complete, the Implementation Complete dialog box opens.

The status indicator in the upper-right corner of the workspace will indicate when bitstream generation is complete as well as in the Design Runs tab in the bottom panel.

When generation is complete, the Bitstream Generation Completed dialog box opens.
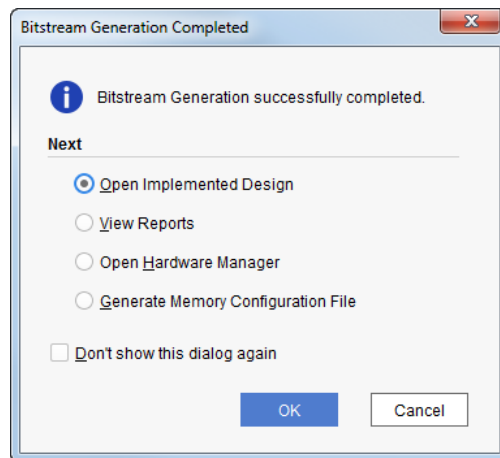
Figure 134: Bitstream Generation Completed Dialog Box

You can view reports to review what was created or open the hardware manager to program the device or view the information from the Vivado analyzer tool.

You can also simply cancel to perform neither of these tasks. These tasks, and others, are still available through the Flow Navigator.

1-1-5. Click **Cancel** to perform none of these tasks and end the bitstream generation process.

OR

If you would like to quickly jump to one of these frequently used next steps, select the option of your choice and click **OK**.

Opening the Hardware Manager

The hardware manager is the portion of the Vivado Design Suite that enables the monitoring of cores that were added to a design.

1-1.   Open the hardware manager.

1-1-1. Click **Open Hardware Manager** in the Bitstream Completed dialog box and click **OK**.

The Hardware Manager window opens.

The hardware needs to be connected and the information bar invites you to open an existing or a new target.
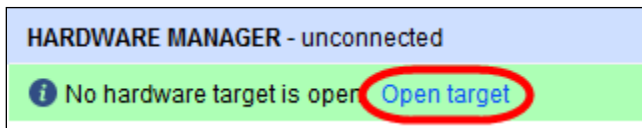
Figure 135: Opening a Hardware Target

1-2.   Connect the target through the New Target Wizard to guide you through the process.

1-2-1. Click **Open target** > **Open New Target**.
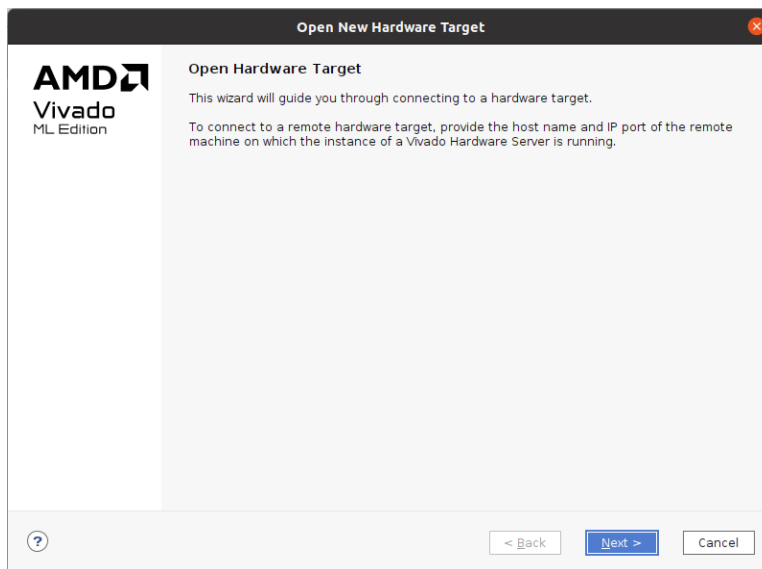
Figure 136: Open Hardware Target Dialog Box

1-2-2. Click **Next** to set the hardware server settings.

1-2-3. Enter a name for the server.

AMD
together we advance_

Typically, this is left at its default value.
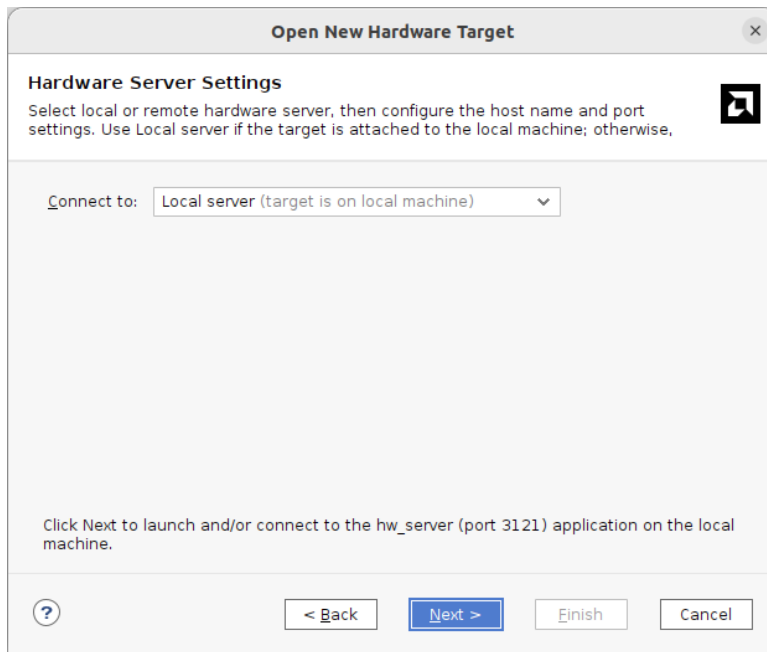
Figure 137: Setting the Server Name for the New Hardware Target

1-2-4. Click **Next** to select the hardware target.

1-2-5. Verify the hardware target.

This becomes important when there are multiple targets connected to the PC.

You can change the frequency of the JTAG cable if you are experiencing communications problems.

[Linux users]: If you receive an error saying that no active target is found, check the USB connections by selecting **Devices** > **USB** in the VirtualBox toolbar at the top.

Figure 138: Selecting the Hardware Target

1-2-6. Leave the frequency at its default value.
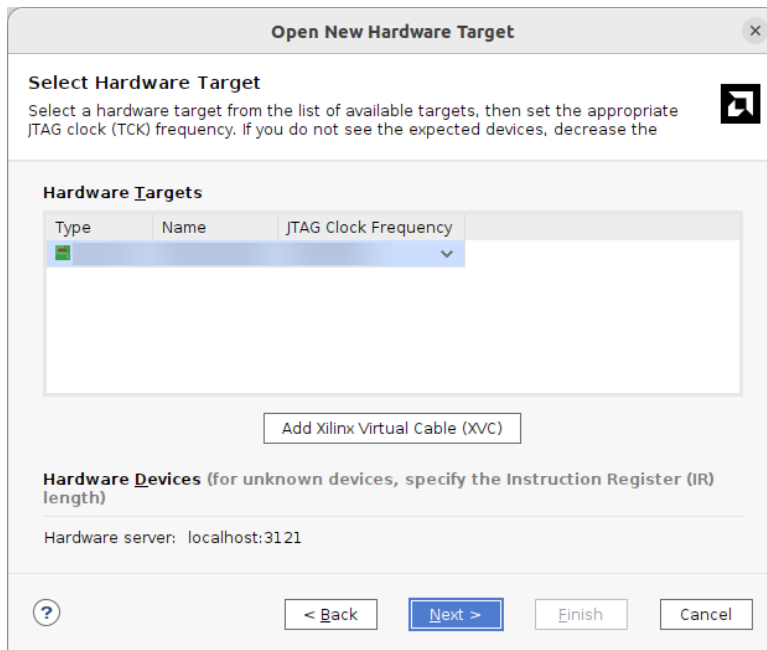
Figure 139: Selecting the Hardware Target

1-2-7. Click **Next** to view the hardware target summary.

A summary of the connection is displayed.
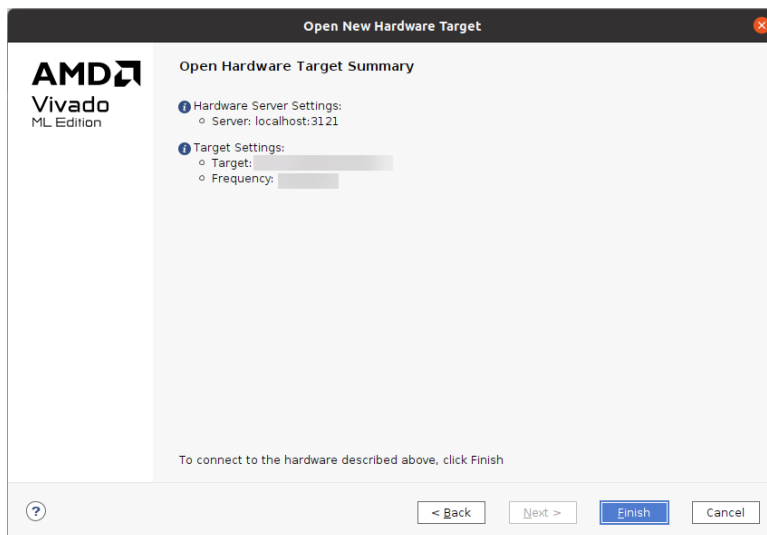
Figure 140: Summary of the Open Hardware Target Settings

1-2-8. Click **Finish** to connect to the new hardware target.

AMD
together we advance_

Launching and Configuring the Terminal Emulator

Maintaining consistency in the instructions is an important aspect of the lab experience. The labs will use general and vague commands such as "Open the terminal emulator". Depending on the system that you are using, you must know which terminal emulator to configure and launch. Here are the instructions for opening a terminal emulator for both the Linux and Windows environments.

Generally, the software labs will use the serial terminal built into the Vitis platform; however, there are some labs that will require you to communicate with the board without using the Vitis environment. These situations require the use of a third-party serial port emulator.

We have tested Tera Term for the Windows environment and GTKTerm for Linux. The GTKTerm tool is pre-installed with the Customer Training VM. Both terminal emulators listed here run independently of the tools. If you have another terminal emulator that you prefer, you can certainly use it; however, you are responsible for figuring out how to configure it.

Linux

GTKTerm is a simple GTK+ terminal used to communicate with the serial port. Other terminal emulators may be used; however, the installation and configuration instruction provided here are for GTKTerm.

GTKTerm is already installed in the VM provided by the Customer Training team; however native Linux users may need to install GTKTerm.

1-1.   [Native Linux users] Acquire and install the GTKTerm software from the command line.

1-1-1.   Press <**Ctrl** + **Alt** + **T**> to open a new terminal.

1-1-2.   Download and install GTKTerm:

    [host] $ sudo apt install gtkterm

1-1-3.   When prompted for the password, enter the super user password.

The tools will install in about a minute or less.

GTK Term is an interface that only supports serial port/UART communications. The more general Terminal tab supports other formats, such as SSH and Telnet.

Since serial port communications occur over the USB, which is an enumerated interface, the board must be powered on with the cable connected for the port to be recognized.

If you are using the VM, ensure that the proper USB port is allowed to cross the host/VM threshold by selecting **Devices** > **USB** > **Xilinx <board selection> [0800]** in the VirtualBox Manager for Zynq UltraScale+ MPSoC/Versal adaptive SoC boards or selecting **Devices** > **USB** > **Digilent Adept USB Device [0700]** *AND* **Silicon Labs CP2103 USB to UART Bridge Controller [0100]** for Zynq 7000 SoC boards.

1-2.   Launch GTKTerm and set the port configuration.

1-2-1. Click the **GTKTerm** icon (⬚) from the quick launch toolbar.

Alternatively, GTKTerm can be launched from a Linux terminal window (<**Ctrl + Alt + T**>) and entering:

```
[host] $ sudo gtkterm
```

Note: While the application will run as a regular user, you must be a super user to access the ports.

When the GTKTerm window opens, perform the following.

1-2-2. Select **Configuration** > **Port** to open the Configuration dialog box.

1-2-3. Identify the port associated with your board and set the port as **/dev/ttyUSBx** (where x could be 0, 1, 2, 3, etc.)

1-2-4. Set the baud rate to **115200**.

1-2-5. Leave the rest of the settings at their default.

Note: You can open multiple instances of `/dev/USBx` if you are unable to determine which port your UART is connected to.

Figure 141: Opening GtkTerm and Selecting the Port Configuration

[Optional]: You can save these settings so that you do not have to reconfigure GKTTerm each time you open it by selecting **Configuration** > **Save configuration**.

AMD
together we advance_

If you save the configuration as "default", this configuration will open when GTKTerm starts. Otherwise, you can save the configuration with another name; however, you will then need to load the configuration each time you start GTKTerm.

1-2-6. Click **OK** to save the settings and leave the terminal open.

1-3. Enable auto CR/LF mode.

GTKTerm is a terminal emulator that supports automatic translation of CR (carriage return) characters to LF (line feed) characters and vice versa.

1-3-1. Select **Configuration** > **CR LF auto** to enable auto CR/LF mode.

Figure 142: Enabling CR/LF auto in GTKTerm

Note: This step is optional, but it will ensure that the serial messages appear aligned on the terminal.

Windows

Tera Term is a popular public domain terminal emulation program. It is capable of operating as a serial port terminal or as a telnet client.

1-4. Download and install Tera Term.

1-4-1. Acquire and install the Tera Term software from any legitimate site. Recommended sites:

- o ttssh2.osdn.jp/index.html.en (Tera Term home page with documentation)
- o osdn.net/projects/ttssh2
- o en.sourceforge.jp/projects/ttssh2
- o logmett.com

1-4-2. Install per instructions.

There are two types of downloads: a traditional zip install, and a self-installing version, which is recommended.

[Optional]

Certain drivers like installing their com port numbers using high numbered serial ports. Tera Term does not accept these port numbers by default, so you will need to override the Tera Term settings:

1-4-3. Open **TERATERM.INI** (found in the install path for Tera Term) with an ASCII text editor.

1-4-4. Set to MaxComPort=256.

1-4-5. Save and close the INI file.

1-4-6. Use the **Setup** > **Save Setup** option to save the setup.

1-5. Launch the Tera Term terminal program.

1-5-1. Double-click the **Tera Term** icon on the Windows desktop to launch Tera Term.

Alternatively, you can select **Start** > **All Programs** > **Tera Term** > **Tera Term**.

1-5-2. Select **Serial** as the connection (1).

1-5-3. Click the **Port** drop-down list to view the available COM ports (2).

Note: If your port is not listed, exit Tera Term, power cycle your board and restart this step.

1-5-4. Select the COM **#** (3).

Hint: MPSoC and RFSoC devices display three COM ports. Their specific numbers may vary based on the USB enumeration process; however, the first two COM ports connect to the UARTs in the PS and the third connects to PL pins.
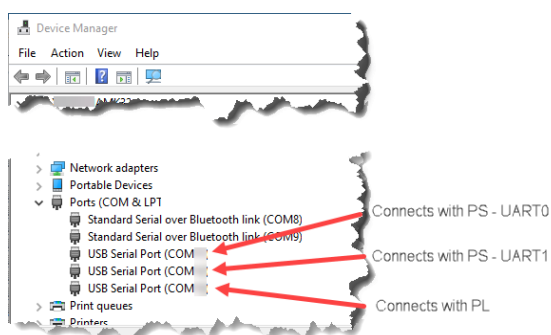
Figure 143: Locating and Identifying COM Ports from the Windows Device Manager
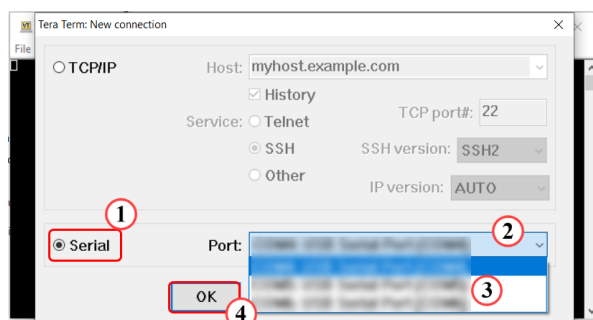
Figure 144: Selecting the COM Port

Note: The COM port setting is specific to the computer being used and may need to be different than shown. Use the COM port **#** that was discovered in the previous step.

1-5-5. Click **OK** (4).

AMD
together we advance_

The terminal console window opens.

1-5-6. Select **Setup** > **Serial Port**.
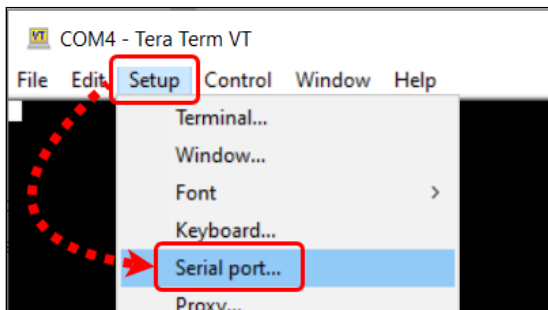
Figure 145: Opening the Tera Term Serial Port Setup Window

The Tera Term Serial Port Setup dialog box opens.

1-5-7. Confirm that the proper serial port has been selected (1).

1-5-8. Set the baud rate to **115200** (2).
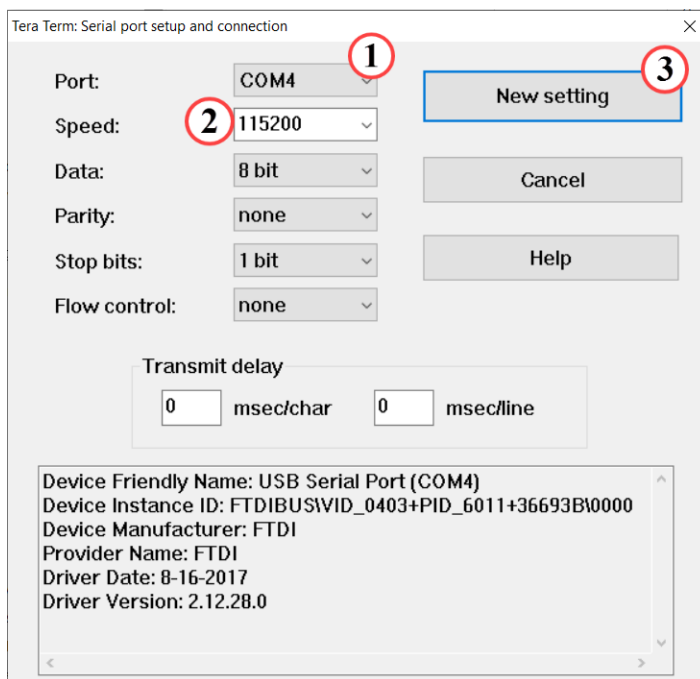
Figure 146: Setting the Parameters for the Serial Port

Note: The COM port setting is specific to the computer being used and may need to be different than shown. Use the COM port **#** that was discovered in the previous step.

1-5-9. Click **OK** (3).

Tera Term is now configured to receive and transmit serial information to/from the evaluation board.

Vivado Analyzer Operations

This sub-section covers many of the common activities associated with configuring IP cores and collecting data.

In This Section

Informative Information

The following information is for background understanding of the materials discussed in the Instructions section.

Overview of the Vivado Analyzer Hardware Session

The following is a quick review of the Hardware Session screen usage.
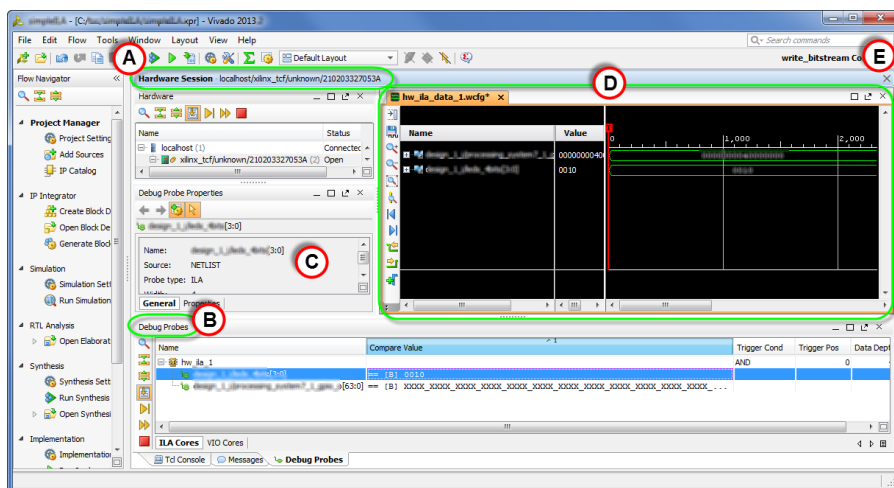
Figure 147: Vivado Analyzer - Hardware Session Screen Usage

- o A: Name of the active function (hardware session) and its TCF connection – This shows that you have successfully opened the hardware session.
- o B: Debug Probes – This section lists each ILA and VIO and the various probes within each.
- o C: Debug Probe Properties – Provides relevant information about the selected probe (the type of device it is attached to, its width, etc.)
- o D: Waveform Window – Each tab supports a different ILA or VIO.
- o E: Exit Hardware Session – Click the 'X' to exit the hardware session and access other Vivado Design Suite capabilities.

AMD
together we advance_

Understanding Trigger Conditions

Trigger conditions describe the desired state of various trigger-capable signals. When these settings are found, the ILA will trigger and capture data.

The Vivado analyzer recognizes five different signal conditions: Rising, Falling, 1, 0, and X (don't care). These signals conditions can be logically joined to describe more complex situations.

First, consider a single probe example:
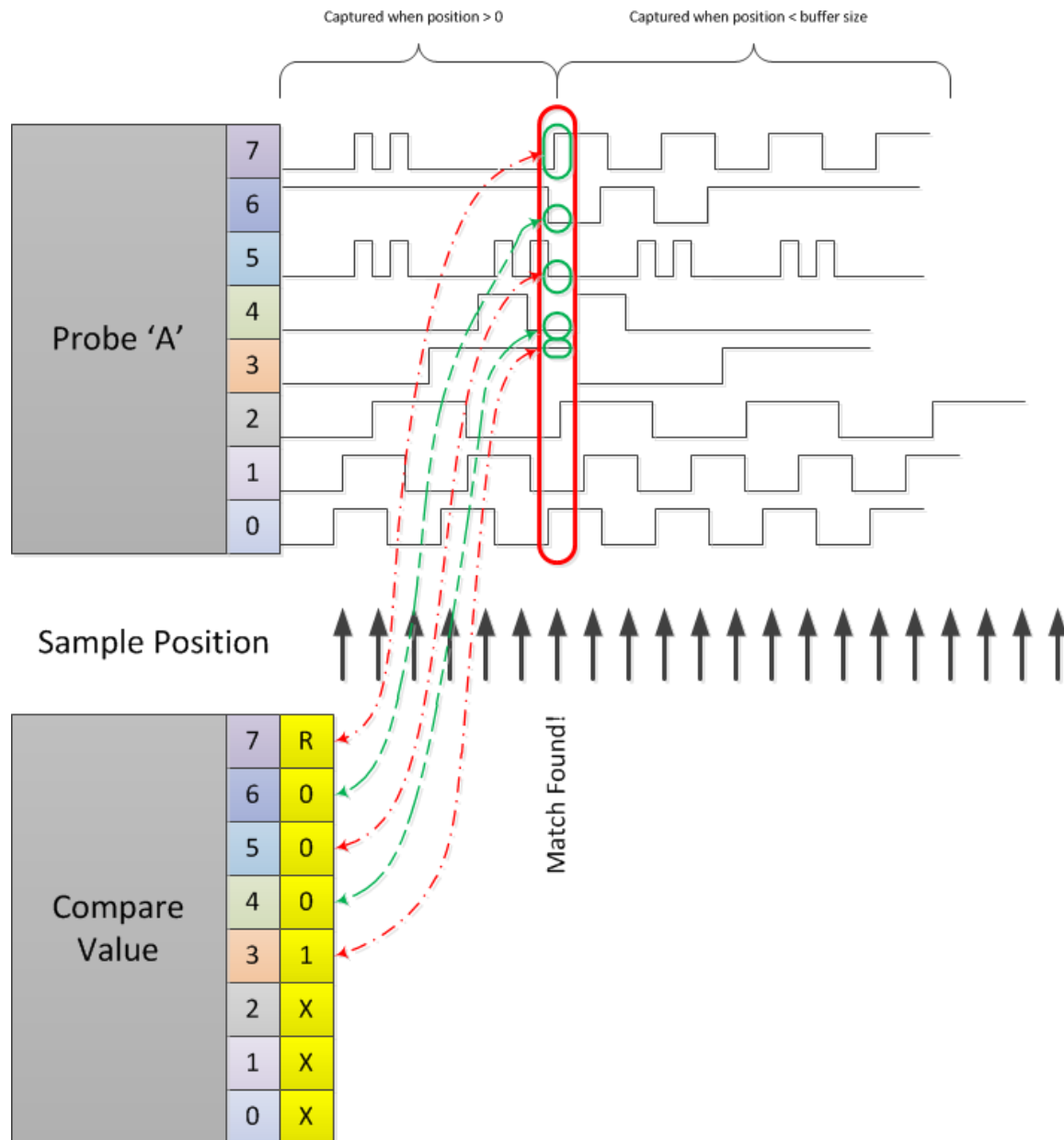
Figure 148: An Individual Probe's Compare Value

Each position or signal within a given probe corresponds to one of five possible match conditions (R,F,0,1,X) as specified in the Compare Value setting. The first time each signal matches its specified "compare value", the combination becomes "true". When all of the signals simultaneously match all of the "compare values", the probe condition becomes true. If there is only one probe, then the trigger conditions are met and data is captured.

If there are multiple probes within the ILA/VIO, then the results of the probes can be ANDed or ORed together to cause a trigger. For example, if it does not matter which probe triggers in order to capture data, you would set the ILA trigger condition to "OR". If both probes must be simultaneously "true", then set the trigger condition to "AND", which will then cause the trigger.

Instructions

The following sections contain only the detailed instructions for the specified task.

In This Section

AMD
together we advance_

Opening the Vivado Analyzer Hardware Manager

The Vivado logic analyzer requires a connection to the target hardware board. The actual connection is typically made using a USB-to-JTAG intelligent cable. Many of our evaluation boards provide support for the intelligent cable as a module component on the board.

When you open a hardware session, a cable server program is launched that identifies the cable type and JTAG components on the board. A TCP/IP port is opened for a connection that may be to a logic analyzer or other application.

1-1.   Open the hardware manager.

1-1-1. Expand **PROGRAM AND DEBUG** from the Flow Navigator.

1-1-2. Double-click **Open Hardware Manager**.
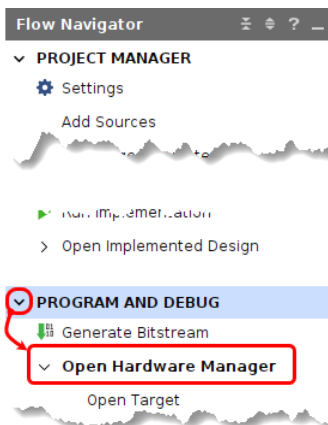
Figure 149: Opening the Hardware Manager from an Open Project

1-1-3. Click **Open target** > **Open New Target** from the Hardware Session window to open a wizard that will facilitate making a connection to the USB platform JTAG download cable.
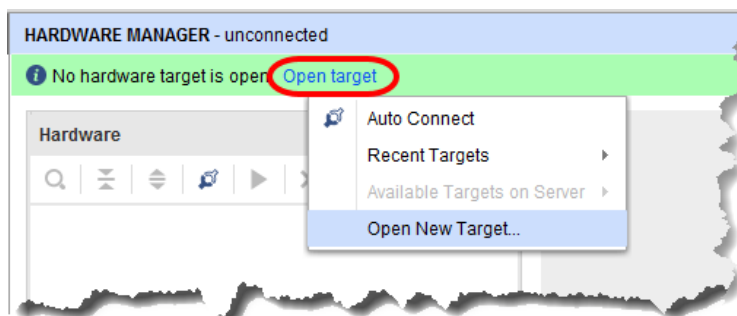
Figure 150: Selecting Open New Hardware Target

1-1-4. Click **Next** to bypass the Welcome dialog box.
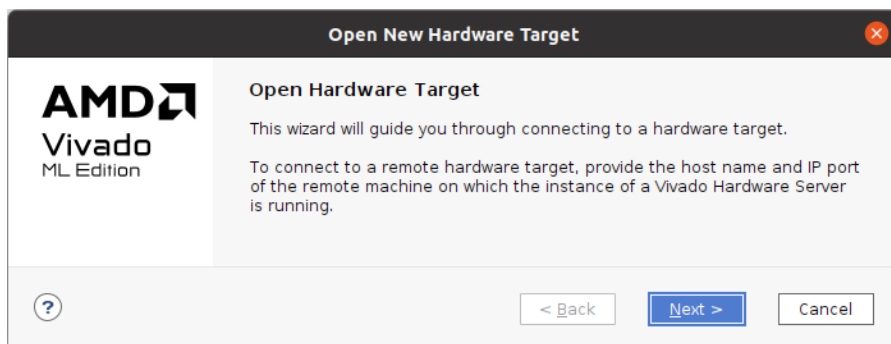
Figure 151: Opening a New Hardware Target

1-1-5. Use the default local server name since your board is connected to the machine that you are running the Vivado Design Suite on (1).

If this is not the case, you will need to select the connection to the machine that is attached to the board.
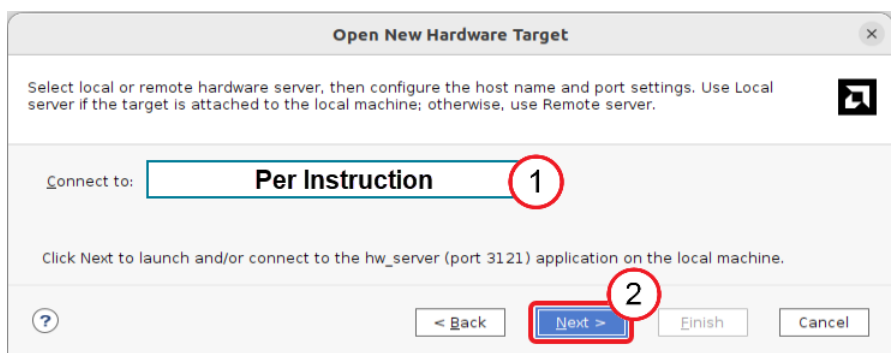
Figure 152: Using the Default Server Name

1-1-6. Click **Next** to scan the JTAG chain and view the nodes within the chain (2).

You should now see the xilinx_tcf hardware target and the hardware devices present in the JTAG chain. The JTAG clock speed is also shown.
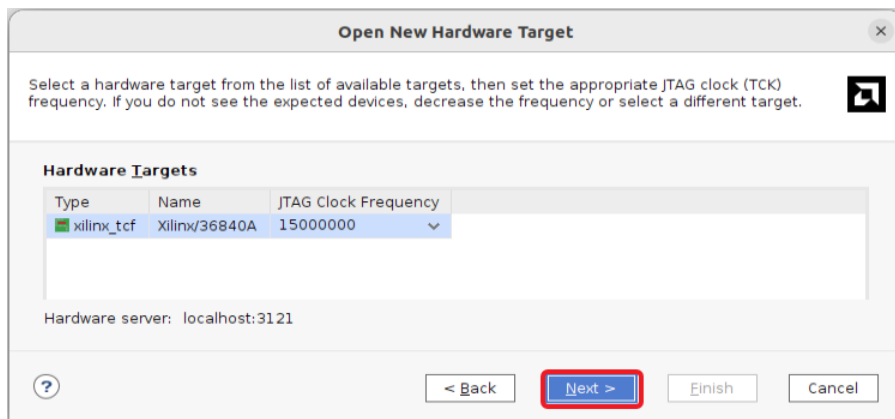
Figure 153: Viewing the Hardware Target and Devices

AMD
together we advance_

1-1-7. Click **Next** to advance to the summary.
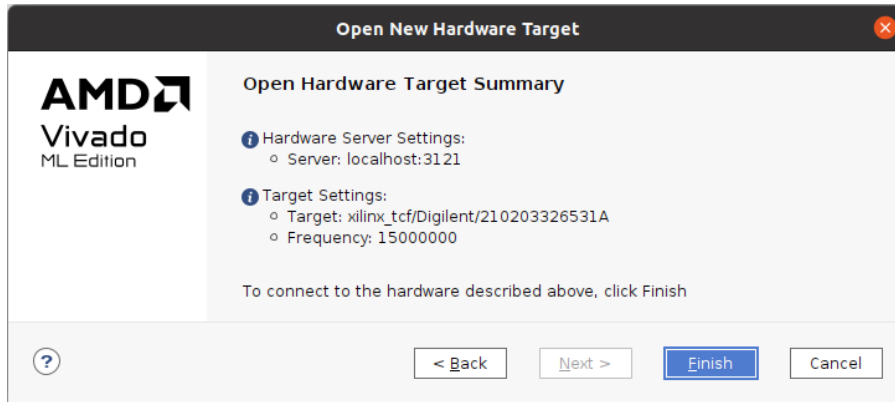
Figure 154: Summary Dialog Box

1-1-8. Click **Finish** to connect to the target.

Adding Probes (Signals) to the Waveform Viewer

When the hardware session opens, one or more probes (inputs into an ILA or VIO) are present.

1-1. Add a probe.

1-1-1. [Optional] Select one or more probes to add to the waveform in the Design Probes window.

1-1-2. Right-click the probe to add to the waveform from the Design Probes window.

  o Select **Add Probes to Waveform** to add the selected probes to the waveform or

  o Select **Add All Probes to Waveform** to add all the probes
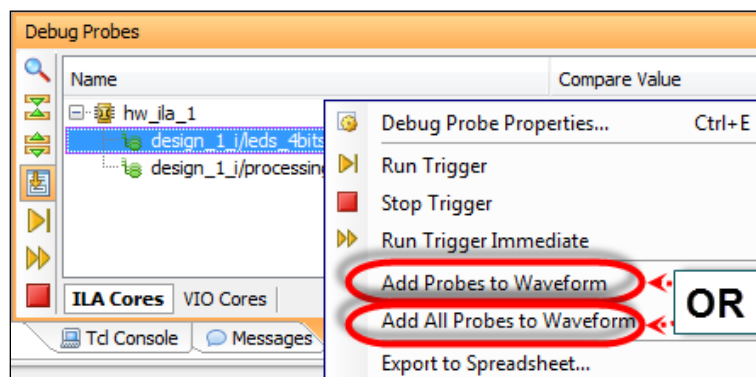
Figure 155: Adding Selected (or All) Probes to the Waveform

The probes will be added after any existing probes in the waveform.

Note that any given probe may be added more than once. This may be desirable as each probe entry in the waveform can have a different radix or color.

Removing Probes (Signals) from the Waveform Viewer

When the hardware session opens, one or more probes (inputs into an ILA or VIO) are present.

1-1. Remove one or more probes that you no longer want.

1-1-1. Select one or more probes from the Waveform window.

1-1-2. Press the **Del** key or right-click and select **Delete**.
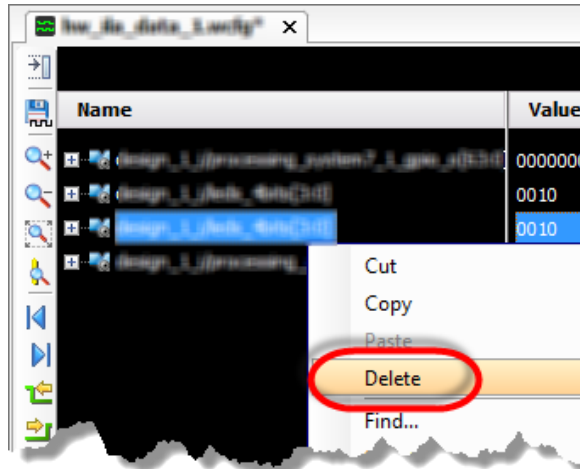
Figure 156: Deleting a Probe from the Waveform Viewer

AMD
together we advance_

Arming the Vivado Analyzer Core

The Vivado analyzer cores are individually armed. Arming a core simply means that the core is made ready to collect incoming signals.

There are two basic modes of arming: Run Trigger and Run Trigger Immediate. Run Trigger Immediate ignores the trigger conditions and simply captures data from the time that you click Run Trigger Immediate until the core's memory fills. Run Trigger immediately captures data when the specified trigger condition is met.

Cores must be armed one at a time; therefore, it is good design methodology to use the Trigger In and Trigger Out ports of the ILA and arm the cores from the last core backwards to the first core. In this fashion, the sequence cannot be completed until the first ILA is armed. This prevents "earlier" ILAs from arming and having their trigger conditions met before you can arm the subsequent ILAs.
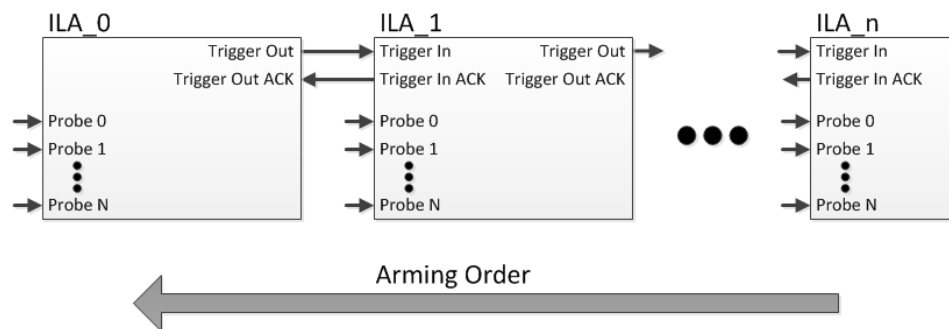
Figure 157: Connections and Arming Order for Multiple ILAs

1-1.   Arm the ILA core.

1-1-1. Click the **Run Trigger** icon (▶).

Triggering the ILA Immediately

Sometimes it is beneficial to see the activity on an ILA or VIO without having to set or modify the trigger conditions.

If trigger conditions have not yet been set (all comparisons are set to 'X' - don't care), this is equivalent to Run Trigger Immediate.

1-1.   Capture whatever data is currently presented to the specified analyzer core (trigger the ILA/VIO immediately).

1-1-1. Click the **Run Trigger Immediate** icon (»).

After a (typically) short delay, depending on the sample rate and size of the capture buffer, data will be uploaded from the device and displayed in the Waveform viewer.

Vitis Model Composer Operations

This section contains instructions for commonly performed tasks using Vitis Model Composer.

In This Section

Launching Vitis Model Composer

1-1.   Launch Vitis Model Composer.

1-1-1. Press <**Ctrl + Alt + T**> to open a new terminal window.

1-1-2. Enter the following command to source Vitis Model Composer:

```
[host]$ source /opt/amd/Model_Composer/2024.1/settings64.sh
```

Note: The installation path (`/opt/amd`) is valid for the Customer Training VM and CloudShare environments. Modify the path as necessary for your environment.

1-1-3. Enter the following command to set the installed MATLAB software path:

```
[host]$ export PATH=$PATH:<Path_to_your_Matlab_installation>/bin
```

For the Customer Training VM and CloudShare environments, the MATLAB software is installed in the `/home/amd/matlab` path. Enter the following command:

```
[host]$ export PATH=$PATH:/home/amd/matlab/bin
```

1-1-4. Enter the following command to launch Vitis Model Composer:

```
[host]$ model_composer
```

Simulating a Model in the Simulink Software

1-1.   Simulate the model.

1-1-1. Click the **Run** icon in the toolbar from the SIMULATION tab in the Simulink software.

AMD
together we advance_

Vitis Unified IDE Operations

Instructions

In This Section

Launching the Vitis Unified IDE and Setting a Workspace

1-1. Launch the Vitis Unified IDE.

1-1-1. Click the **Vitis** icon ( ) from the taskbar or desktop to launch the tool.
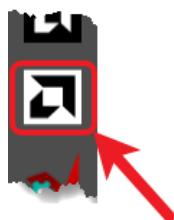
Figure 158: Launching the Vitis Unified IDE from the Taskbar

Note: The Vivado (hardware) tool has a similar icon but with inverted colors. The CloudShare and CustEd VM environments display both icons in the toolbar with the Vivado tool icon visible just above the Vitis Unified IDE icon. Make certain that you select the correct icon.

Alternatively, from a Linux terminal window (**<Ctrl + Alt + T>**), enter the following:

```
[host] $ source /opt/amd/Vitis/2024.1/settings64.sh; vitis
```

Note: This installation path is valid for the CloudShare and CustEd VM environments. Use the proper path for your environment.

It is important to note that not all tools included in the Vitis suite of tools are supported under Windows.

1-1-2. If not already maximized, click the Maximize icon in the upper-right corner of the tool window to see all the views.

1-2.   Set the workspace.

Best practices recommend opening a workspace regardless of the flow you are following as it makes it easier to find the tool-generated logs and scripts.

Alternatively, you can immediately jump into one of the development flows, where you will eventually be asked to specify a workspace. This causes the log and script files to be placed in two different directories.

1-2-1. From the Vitis Components window, click the **Open Workspace** link.
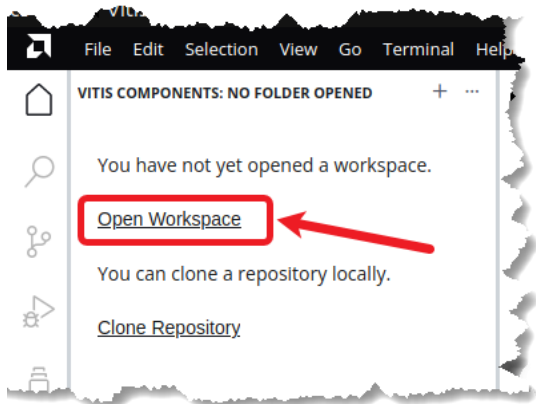
Figure 159: Opening the Workspace in the Vitis Unified IDE

1-2-2. Browse to the following location:

```
$TRAINING_PATH/<the topic cluster name>/lab
```

1-2-3. Click **Ok** to open the new workspace.

The tool relaunches using the new workspace.

1-2-4. Close the **Welcome** tab if it appears.

This tab opens various types of components, enables access to documentation and examples, quickly switches workspaces, etc. If you want to reopen the Welcome tab, select **Help** > **Welcome**.

AMD
together we advance_

Opening a Vitis Unified IDE Workspace from the Welcome Window

1-1. **Set the workspace.**

Best practices recommend opening a workspace regardless of the flow you are following as it makes it easier to find the tool-generated logs and scripts.

Alternatively, you can immediately jump into one of the development flows, where you will eventually be asked to specify a workspace. This causes the log and script files to be placed in two different directories.

1-1-1. From the Vitis Components window, click the **Open Workspace** link.
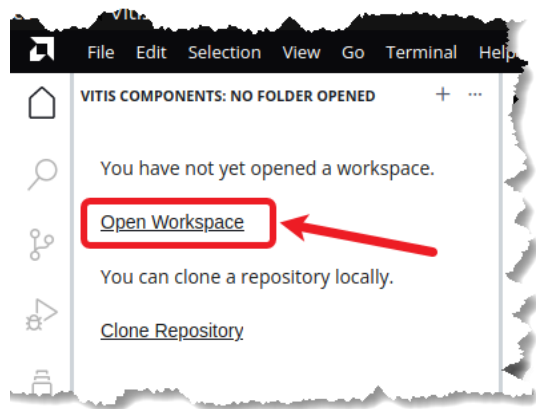
Figure 160: Opening the Workspace in the Vitis Unified IDE

1-1-2. Browse to the following location:

```
$TRAINING_PATH/<the topic cluster name>/lab
```

1-1-3. Click **Ok** to open the new workspace.

The tool relaunches using the new workspace.

1-1-4. Close the **Welcome** tab if it appears.

This tab opens various types of components, enables access to documentation and examples, quickly switches workspaces, etc. If you want to reopen the Welcome tab, select **Help** > **Welcome**.

Launching the Python Command Line Interface

The Vitis Unified IDE supports the use of a Python command line interface that allows scripting to automate complex or repetitive processes, reducing human error and/or tedium.

The *Vitis Unified Software Platform Documentation: Embedded Software Development User Guide* (UG1400), "Vitis Python CLI" chapter describes the available commands and provides more examples.

Access the Python CLI in the Vitis Unified IDE via a terminal window to enter Python commands directly into the environment, allowing you to configure Vitis tool projects without the use of the GUI.

1-1. Open a terminal and set up the CLI.

1-1-1. Select **Terminal** > **New Terminal** from the Vitis Unified IDE menu bar to launch a new instance of the Vitis terminal.

1-1-2. Enter the following command to launch the Python CLI in interactive mode:

```
[host]$ vitis -i
```

Creating an Application Component from Example Templates

Once the platform component is built, the next step is to create an application.

The Create Application Component Wizard provides a quick way to create an application via the example templates available in the Vitis Unified IDE. Based on the template that is selected, appropriate options are populated, and you choose the right options to create the application component.

1-1. Create an application component.

1-1-1. Navigate to the toolbar and click **File** (1).

1-1-2. Hover over **New Component** and select **From Examples** from the options (2).
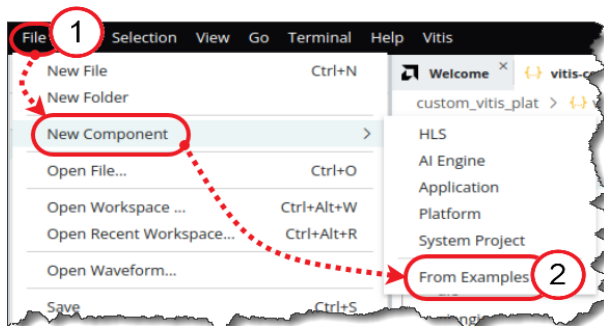
Figure 161: Creating an Application Component from Examples

AMD
together we advance_

The Examples window becomes visible.

1-1-3. Search for the **application template** example from the Examples window (1).

1-1-4. Select **application template** from System Design Examples > Vitis Accel Examples Repository > Installed Examples Repository (2).

1-1-5. Click **Create Application from Template** to create the application component (3).
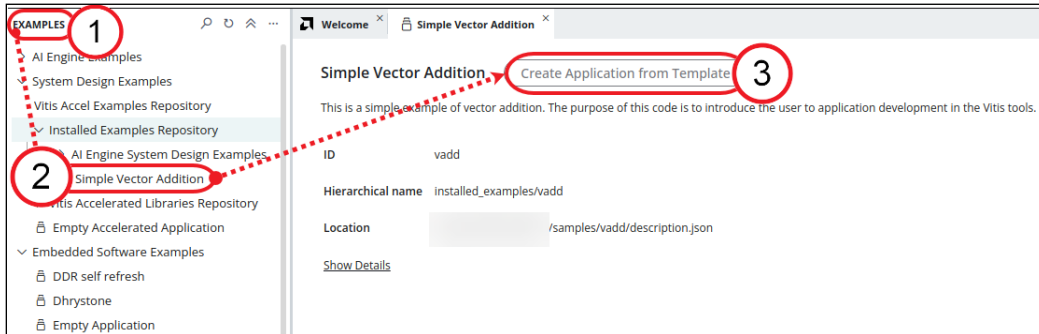
Figure 162: Selecting the Example Template and Creating an Application Component

The Create System Project dialog box appears.

1-1-6. Enter **your system project name** in the system project name field (1).

1-1-7. Click **Browse** in the System project location field to set the appropriate location (2).

1-1-8. Browse to the following directory:

```
$TRAINING_PATH/<the topic cluster name>/lab
```

1-1-9. Click **OK**.
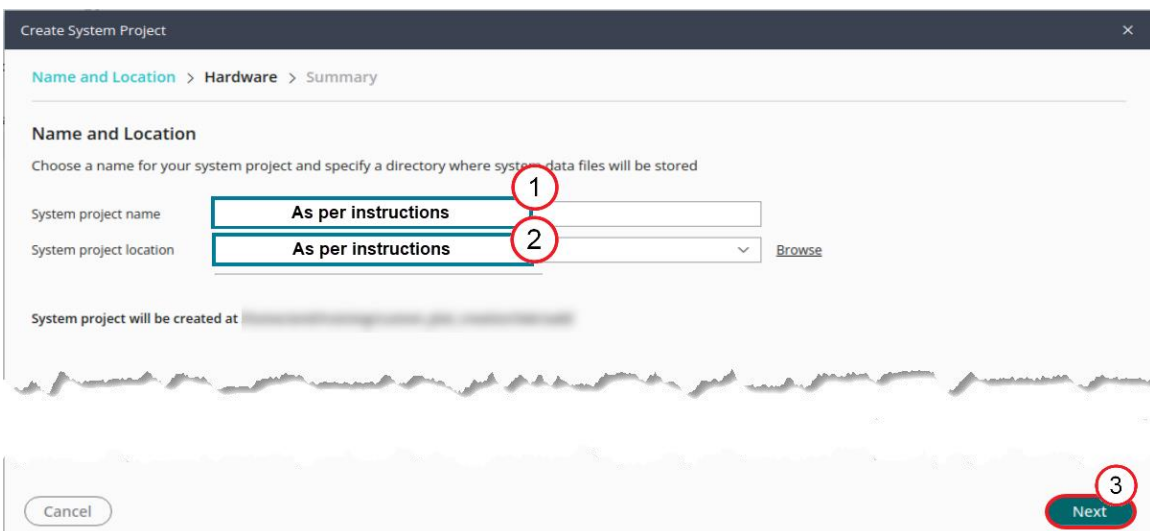
Figure 163: Creating the System Project and Selecting the Location

1-1-10. Click **Next** (3).

1-1-11. Select **your platform project name** from the list.

1-1-12. Click **Next**.

1-1-13. Select the following paths (1):

- o Kernel image:

    `the path to the kernel image`

- o Root FS:

    `the path to the file system root`

- o Sysroot path:

    `the path to the root of the system`

1-1-14. Select the **Update Workspace Preference** option (2).

Figure 164: Selecting the Embedded Components Paths

1-1-15. Click **Next** (3).

1-1-16. Review all the options and click **Finish**.

AMD
together we advance_

Creating a Platform Component

A platform component contains a thorough description of a hardware design, including:

- o Which processors are available
- o The PL-based peripherals and the enabled PS peripherals
- o A full system memory map

Based on this description, software, such as the board support package (BSP), and applications can be tailored to the hardware.

Normally, you would use your hardware description file, which is typically generated by the Vivado Design Suite; however, there are situations when you just want to quickly test your code using the standard peripheral set for the supported device. Choose from the provided collection of predefined hardware templates for various boards.

The following instructions illustrate how to include custom hardware.

1-1. Create a platform component.

1-1-1. Navigate to the toolbar and click **File** (1).

1-1-2. Hover over or click **New Component** to open its context menu (2).

1-1-3. Select **Platform** from the options (3).
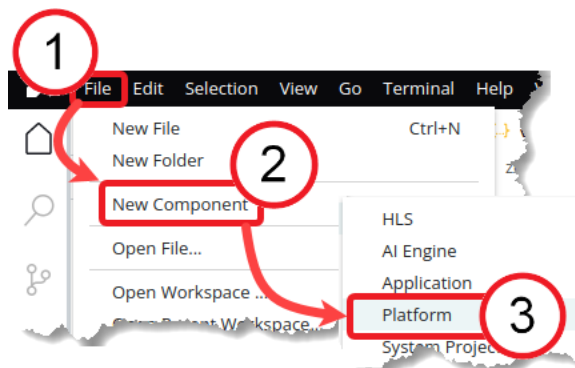
Figure 165: Opening the Create Platform Component Wizard

The Create Platform Component dialog box appears.

Note: There are other ways to access this wizard; however, this is the easiest.

1-1-4. Enter **your platform project name** in the Component name field (1).

> **Hint:** Make certain that there aren't any spaces in this field.

> This sets the name of the platform component. The associated directories and files are populated within the current workspace.

> **Note:** The Component location field is automatically set to the workspace location. The location can be changed if you want to place this component somewhere else.
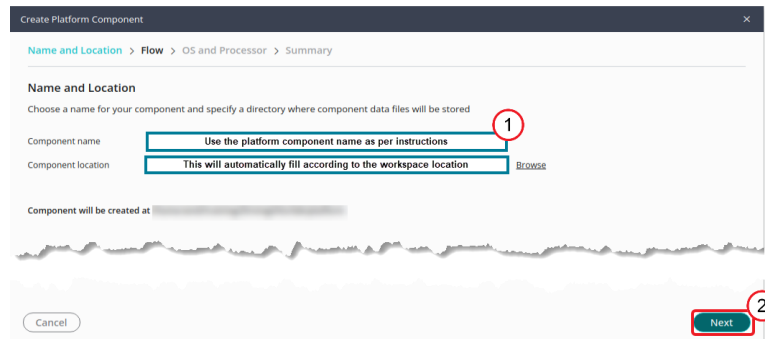
Figure 166: Entering the Platform Component Name and Location

1-1-5. Click **Next** to advance to the next page of the wizard (2).

1-1-6. Select the **Hardware Design** option so that you can import an embedded design from one of the supplied boards or one of your own (1).

1-1-7. Click **Browse** next to the Hardware Design (XSA) field (2).

1-1-8. Browse to the following directory:

```
the location of the files to be exported from the Vivado Design
Suite
```

1-1-9. Double-click the **hardware platform description name.xsa** file.

> Remember that the Vitis Unified IDE does not automatically expand the environment variables. You must type out or navigate to the path. For more details about environment variables, refer to the Introduction section of this lab.
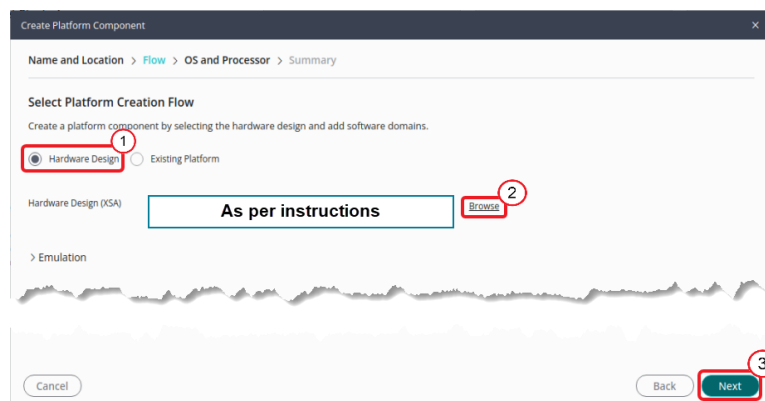
Figure 167: Specifying the XSA File

1-1-10. Click **Next** to select the operating system and processor (3).

AMD
together we advance_

1-1-11. If not already selected, select **the operating system** from the Operating system drop-down list (1).

1-1-12. If not automatically populated, select **your processor** from the Processor drop-down list (2).

Depending on the processor selection, additional options may appear, enabling you to select the processor's instruction set architecture or provide options for boot artifacts. Unless there is a specific reason to do so, leave this at its default setting.

Note: Zynq UltraScale+ MPSoC-based boards will allow you to select the architecture, whereas this option is not available for Versal devices.

1-1-13. Click the down arrow to select the appropriate boot components as follows (3):

Note: The following platforms are listed here for completeness. Follow the guidance for the board or device you selected.

[Versal users]: This field is not available for this target—continue with the next task.

[MPSoC users]: Select **Generate PMU Firmware**, which generates the firmware for the PMU present in Zynq UltraScale+ MPSoCs (4).

Select **Target processor to create FSBL**, which generates an FSBL that runs on the specified processor. Under most circumstances, it doesn't matter which processor is chosen (5).

[Zynq 7000 users]: Select **Generate boot artifacts**. Since there is only one type of processor in this device's PS, the choice to select a processor type is unavailable.

Note: The boot process is handled differently in Versal devices; hence the absence of this field when devices are selected.

Figure 168: Specifying the OS and Target Processor for the Platform

1-1-14. Click **Next** to proceed to the Summary section (6).

1-1-15. Verify the information for the platform component and then click **Finish**.

Note: It takes a few minutes to process. The status is usually indicated in the bottom-left corner, and progress messages appear in the Output window.

Along with the platform component, the wizard creates a domain within the platform component based on the selected operating system and processor.

Creating a Bare-Metal Application Component for an Embedded Flow

Using the Create Application Component Wizard is a quick way to create an application after a platform component is created.

Based on the dialog box choices, the appropriate toolchain is selected for pre-processing, compiling, assembling, and linking.

1-1.   Create an application component.

1-1-1. Navigate the toolbar and click **File** (1).

1-1-2. Hover over **New Component** and select **From Examples** from the options (2).
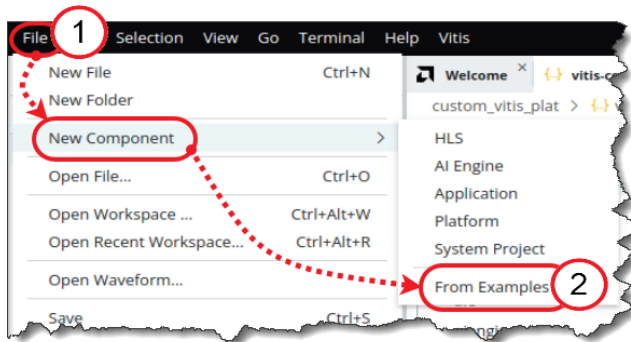
Figure 169: Creating an Application Component from Examples

The Examples window becomes visible.

1-1-3. Select **application template** under Embedded Software Examples (1).

1-1-4. Click **Create Application Component from Template** to create the application component (2).

Alternatively, you can click the **'+'** symbol that appears when hovering over the template name (3).
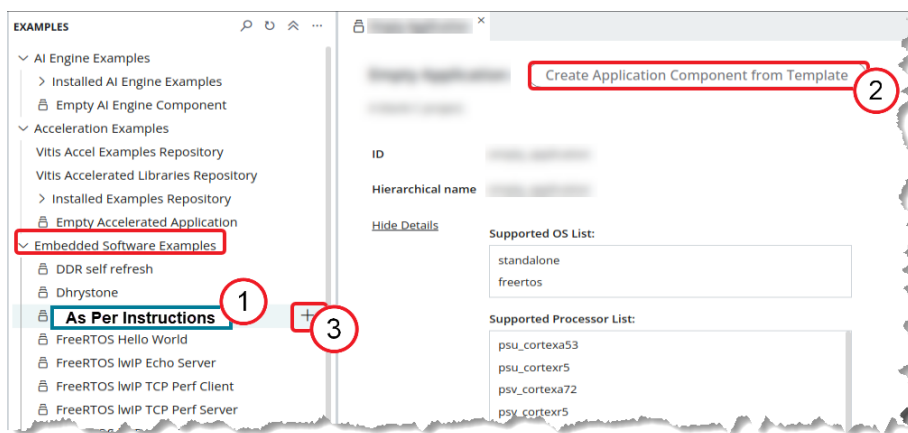
Figure 170: Selecting an Example Template

The Create Application Component - application template dialog box appears.

AMD
together we advance_

1-1-5. Enter **your application project name** in the Component name field (1).

Hint: Make certain that there aren't any spaces in this field.

This sets the name of the application component. While the "app" suffix is not required, using it simplifies identifying the component type

Figure 171: Naming the Application

1-1-6. Click **Next** to continue with the process (2).

You will now associate the application with a platform and domain.

1-1-7. Select **your platform project name** from the list (1).

Note: The platform information view will be populated with values from the selected platform.

Figure 172: Selecting the Platform Component

1-1-8. Click **Next** to continue to the domain selection (2).

You can now associate this application with an existing domain or create a new domain.

1-1-9. If not already selected, select **your domain name** to link to the application component (1).
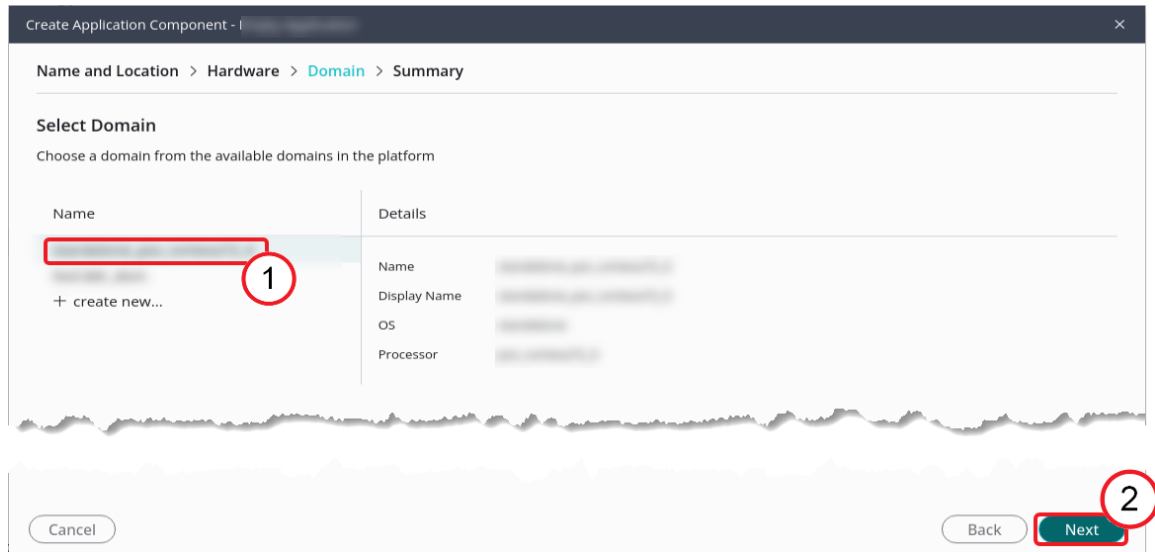
Figure 173: Associating the Application with a Domain

1-1-10. Click **Next** to continue to view the summary (2).

1-1-11. Click **Finish** to build the application component.

> **Note:** This assembles all the pieces but does not compile the application.

© Copyright 2024 Advanced Micro Devices, Inc.

Importing Sources to an Application

Add *your files* source files to the application component.

1-1.   Import the sources.

1-1-1. If not already done, expand **your application project name** > **Sources** so that the src folder is visible in the Vitis Components window (1).

1-1-2. Right-click the **src** directory as this is where you want to place the resource files (2).

The tools require certain files to be in specific locations. The tools look in the src directory for source files and linker scripts. Other types of files can be imported into other directories.

1-1-3. Hover over **Import** and select **Files** to select the necessary files (3).
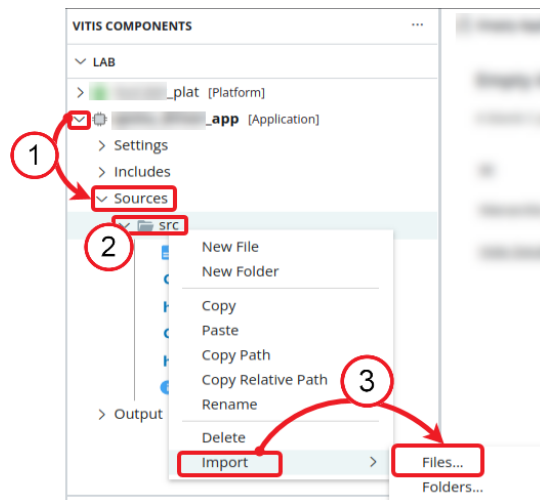
Figure 174: Importing Sources

1-1-4. Browse to the following directory:

```
the directory where your files are located
```

1-1-5. Press <**Ctrl**> and click **your files** to select the file(s).

1-1-6. Click **Open** to add all the files to the src directory.

Opening a Source File in the Editor

1-1. Open *the desired source file* in the editor.

1-1-1. Locate **the desired source file** in the Vitis Components window.

> **Note:** Source files are typically located in the application's src folder and can be found under your application project name > Sources > src (1).

1-1-2. Click **the desired source file** to open it in the editor window (2).
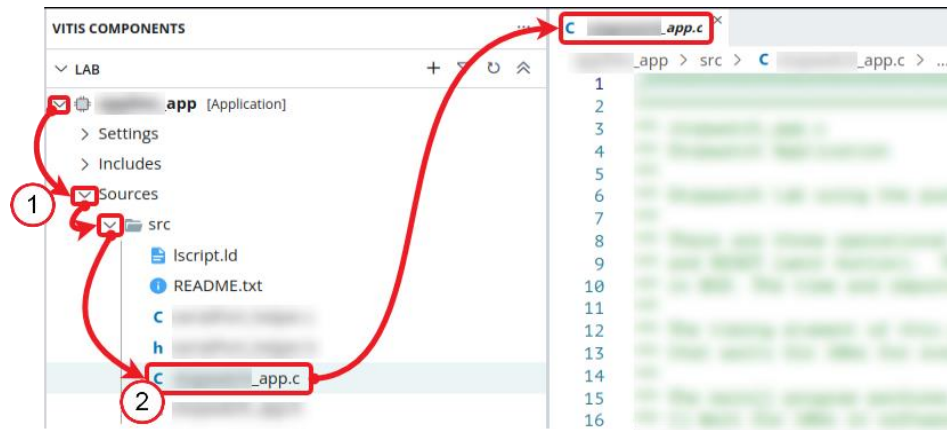
Figure 175: Opening a Source File

Finding Text in a Source File

1-1. Find *the text that you are looking for*.

1-1-1. Select **View** > **Search** or press <**Ctrl + F**>.

> The Search view or Find/Replace dialog box opens based on how it was invoked.

> When invoked from a view window or from selecting View from the toolbar, the Search view is opened.

> **Note:** The Search view can also be invoked by clicking the magnifying glass icon in a view window.

Figure 176: Search View

AMD
together we advance_

When the keyboard shortcut is used, the Find/Replace dialog box appears on the top-right side in the editor.

Figure 177: Search/Replace Dialog Box

Note: To view the Replace field, click the arrow to the left side of the Search or Find field to expand the view or dialog box.

1-1-2. Enter **the text that you are looking for** in the Search or Find field.

1-1-3. Press <**Enter**> to find the next occurrence of *the text that you are looking for*.

1-1-4. Continue pressing <**Enter**> or clicking the down arrow until you locate the specific instance that you are looking for.

If you are looking for text within a specific region of the code, you would first highlight the region to perform the search in and then launch the find/replace function using the keyboard shortcut as described in this set of instructions.

1-1-5. Click the **X** icon to close the Find/Replace dialog box.

Setting Compiler Optimization Options

Compiler options communicate the designer's intentions to the compiler. These options directly affect how the object file is constructed.

1-1. **Set the baseline application's optimization to your desired level of optimization.**

The Optimization settings enable you to select the level of optimization needed to meet your timing requirements for this application. Note that the optimization should be set to zero (-O0) for debugging; otherwise, the one-to-one correspondence between the source code and the executing code is lost.

Remember that the levels of optimization listed are collections of specific optimization flags. Refer to the *GNU Compiler Manual* for a listing of which flags are enabled for each optimization level.

If the predefined optimization settings do not suit your needs, you can add your optimization flags to the *Other optimization flags* text field. Remember that optimizations can change the way your code operates.

1-1-1. Expand **your application project name** > **Settings** in the Vitis Components window (1).

1-1-2. Double-click **UserConfig.cmake** to open the compiler settings for this application component (2).

1-1-3. Click **Optimization** under Compiler Settings (3).

1-1-4. Click the drop-down list in the **Optimization level** setting (4).

1-1-5. If the optimization is not already set to your desired level of optimization, then select **your desired level of optimization** from the *Optimization level* drop-down list (5).

If you need to add any additional flags, place them in the *Other optimization flags* field (6).
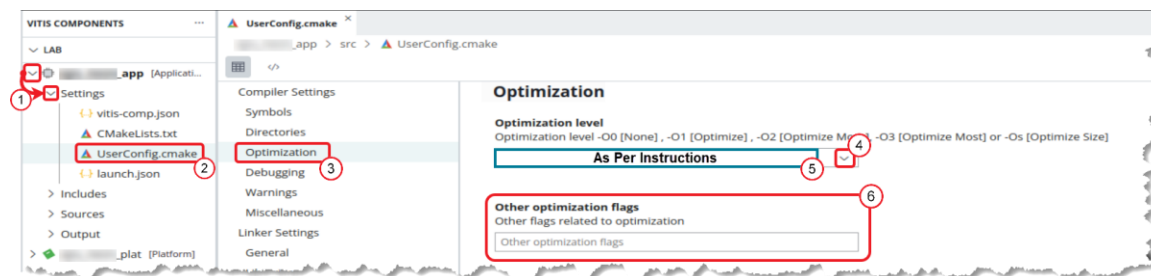
Figure 178: Updating the Application Component Compiler Settings

AMD
together we advance_

Adding Symbols to Application Settings

Adding a symbol to an application is equivalent to adding a #DEFINE SYMBOL_NAME statement that is visible to the entire project. It is common practice to use symbols to conditionally compile code by guarding code with conditional pre-processor statements (e.g., #ifdef SYMBOL_NAME ... #endif).

Remember that #ifdef only tests to see if a preprocessor symbol has been defined or not, and #if can test against specific values (assigning a value to a symbol can be done like this: NEW_SYMBOL=123).

1-1.   Add symbol(s) to the compiler settings.

1-1-1. Expand **your application project name > Settings** in the Vitis Components window (1).

1-1-2. Click **UserConfig.cmake** to open the compiler settings for this application (2).

1-1-3. Click **Add Item** under Defined Symbols (-D) to add the first symbol (3).
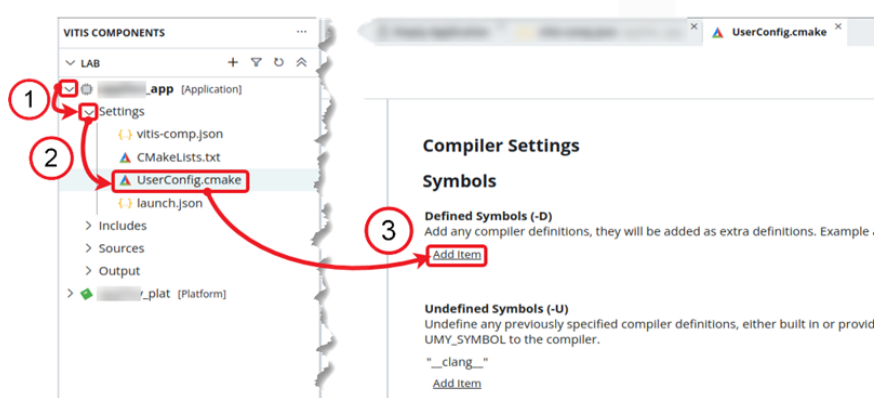
Figure 179: Accessing the UserConfig Settings for the Application

1-1-4. Enter the symbol name in the field that opened after the previous task (1).

Note: Symbols can only be added one at a time.
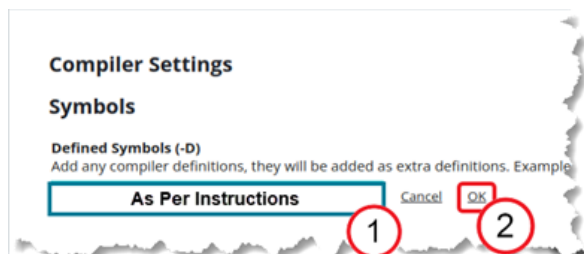
Figure 180: Defining a New Compiler Symbol

1-1-5. Click **OK** to add each new symbol individually (2).

Note: Follow tasks 3 through 5 to add the symbol(s) in this list: **application project symbol names as required**.

1-1-6. Once all the symbol(s) are added, close the **UserConfig.cmake** tab.

Building the Components

1-1. Confirm the build setting and build the components.

1-1-1. From the Flow window, click the arrow for the Component drop-down list (1).

Hint: You may need to click the '>' symbol next to Flow to expand the Flow window.
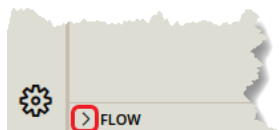
Figure 181: Expanding the Flow Window

1-1-2. Select **your application project name** (2).
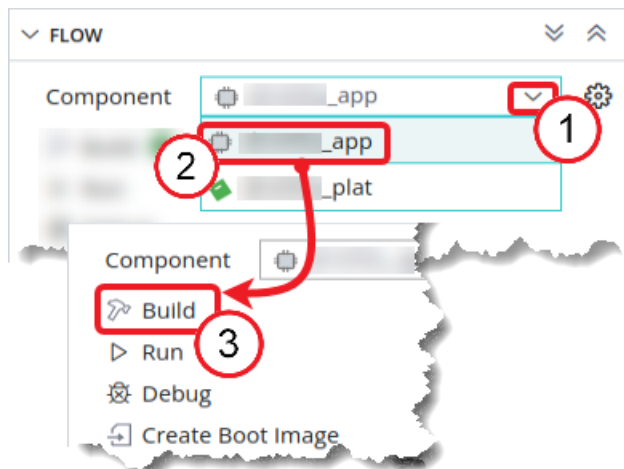
Figure 182: Building the Application Component

1-1-3. Click **Build** (3).

This will build all components with the settings specified in the build configuration (if any).

Note: Click **OK** if prompted with the Platform Build message.

Any issues are shown in the Problems or Output tabs.

AMD
together we advance_

Note: Monitor the progress of the build via the percentage values displayed in the Output tab.
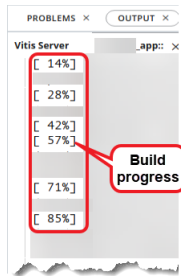
Figure 183: Build Progress Indicator

When the compilation completes successfully, a size summary of the build is displayed.
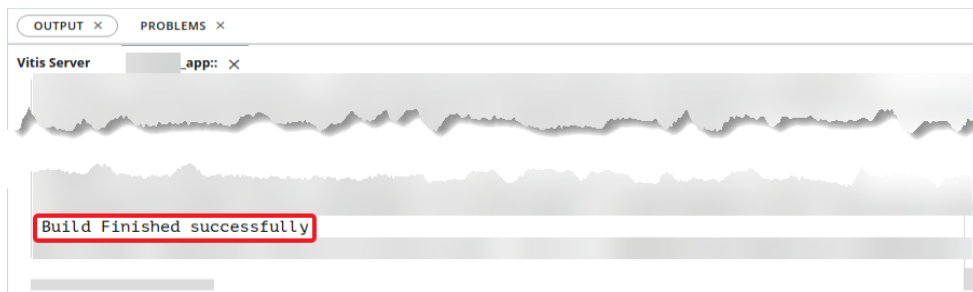
Figure 184: Successful Build Console Message

Running with a Default Configuration

Configurations are a powerful ally in setting up a run for specific sets of criteria. The default configuration settings are adequate for most basic programs and do not require clicking through multiple dialog boxes just to run the program.

1-1. **Run *your application project name* with the default configuration settings.**

1-1-1. Select **your application project name** from the Component drop-down list from the Flow view (1).

1-1-2. Click **Run** to begin running the application on the hardware (2).
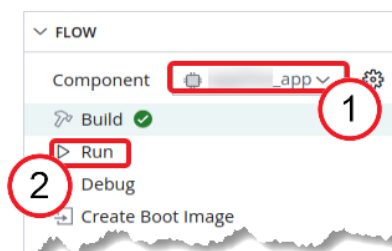
Figure 185: Launching the Application on Hardware Using the Default Configuration Settings

Running an Application on Hardware

1-1. Run the application on the board.

1-1-1. Switch back to the Vitis Unified IDE.

1-1-2. Click **Run** from the Flow view.
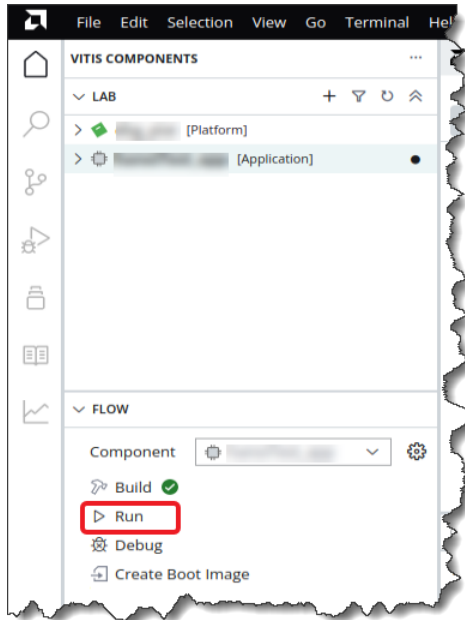
Figure 186: Running the Application on Hardware

Note: This will take a few seconds. During this time, the tool runs several commands to initialize the hardware. First, the hardware is reset. Once reset is successful, the device is programmed with the application.

Launching QEMU for the Embedded Software Flow

> **Note:** The current version of the Vitis Unified IDE does not have direct support for QEMU in the embedded software development flow; hence, as a workaround, you will run QEMU externally to validate the application. Here you will copy the necessary files into the project and run several commands to set up QEMU.

1-1. **Launch a terminal and copy the files needed to launch and run QEMU.**

1-1-1. From the Vitis Unified IDE menu bar, select **Terminal** > **New Terminal** to launch the built-in terminal.

1-1-2. Select the **Terminal** tab to enter commands and view the status of executed commands.

1-1-3. Enter the following command to confirm that the terminal is pointing to
`$TRAINING_PATH/<the topic cluster name>/lab`:

`[host] $ pwd`

1-1-4. Copy the QEMU-related files using the following commands:

`[host] $ cp $TRAINING_PATH/CustEdIP/makefile .`

`[host] $ cp $TRAINING_PATH/CustEdIP/xsct_script.tcl .`

Note: The dot at the end of the command means "copy to the current directory" and MUST be included in these commands.

[MPSoC users]: Also enter the following command:

`[host] $ cp $TRAINING_PATH/CustEdIP/pmu_rom_qemu_sha3.elf .`

This set of commands copies one makefile, one QEMU ELF file, and one Tcl file into the Vitis tool workspace. The makefile sources the Tcl file procs and builds the files to support QEMU and the application executable.

Note: This is a patch. Ideally, the Vitis Unified IDE will open QEMU natively; however, for 2024.1, this minor workaround must be performed.

You will now run the makefile that you just copied to build all the QEMU DTS and scripts necessary to run QEMU from the terminal. Once these files are generated, you will then build and run the application on QEMU and observe the results in the console.

1-2. **Run the makefile to generate the QEMU scripts and then run the application.**

1-2-1. Run the `build_qemu_dts` command to generate the necessary device tree files:

`[host] $ make build_qemu_dts`

With the device tree files generated, the application now needs the executables to be created based on the processor type and design.

1-2-2. Generate the application executables:

[host] $ make generate_apps APP=your application project name PROC=<processor name> XSA=<xsa file name>

where:

[MPSoC users]: `PROC = psu_cortexa53_0`

[Versal users]: `PROC = versal_cips_0_pspmc_0_psv_cortexa72_0`

`<xsa file name>:` If the XSA file that was used to create this project is in the lab directory, then only the file name is required, not the full path. If not, the full path of the file must be provided.

Once the application executables are generated, view the contents of the `a72.txt` or `a53.txt` file to make sure it contains the right path for the application ELF file.

1-2-3. Open the **a53.txt** or **a72.txt** file:

[MPSoC users]:

[host] $ gedit a53.txt

[Versal users]:

[host] $ gedit a72.txt

1-2-4. Verify that the content of the text file is as follows:

```
-device loader,file=your application project name/build/your
application project name.elf,
cpu-num=0
```

Note: If the content of the text file is not the same as the above, replace the text with the above line and save and close the file.

1-2-5. Close the editor by clicking the 'X' in the upper-left corner of the tool.

1-2-6. Generate the QEMU scripts:

[host] $ make generate_qemu_scripts XSA=<xsa file name>

The generated scripts will run the application in the QEMU instance.

A message in the terminal shows that the scripts are generated and are ready to run:

Figure 187: Terminal Message When Generating the QEMU Scripts

1-2-7. Run the `run_qemu` command to ensure that QEMU is ready and to run the application:

[host] $ make run_qemu

AMD
together we advance_

Launching QEMU Using a Script in the Vitis Unified IDE

> **Note:** The current version of the Vitis Unified IDE does not have direct support for QEMU in the embedded software development flow; hence, as a workaround, you will run QEMU by using a script to validate the application.

1-1. Execute the application from a Linux terminal.

1-1-1. From the Vitis Unified IDE menu bar, select **Terminal** > **New Terminal** to launch the built-in terminal.

1-1-2. Select the **Terminal** tab to enter commands and view the status of executed commands.

1-1-3. Enter the following command to navigate to the `support` directory:

```
[host] $ cd $TRAINING_PATH/<the topic cluster name>/support
```

1-1-4. Run the script that launches QEMU with the necessary command line arguments:

```
[host] $ ./<the topic cluster name>_QemuRun.sh your processor
```

It will take a few moments to get QEMU configured and running. The output will appear in the terminal window.

Note: If the script fails to run, you may need to set its permissions. Enter `chmod 777 <the topic cluster name>_QemuRun.sh` and then rerun the script.

Creating and Running the Default Debug Configuration

There are many options as to how to set up a Debug configuration. Often, the default options are enough when you just want to quickly run your application in debug mode.

1-1. **Launch the XSDB console and determine the TCF port number.**
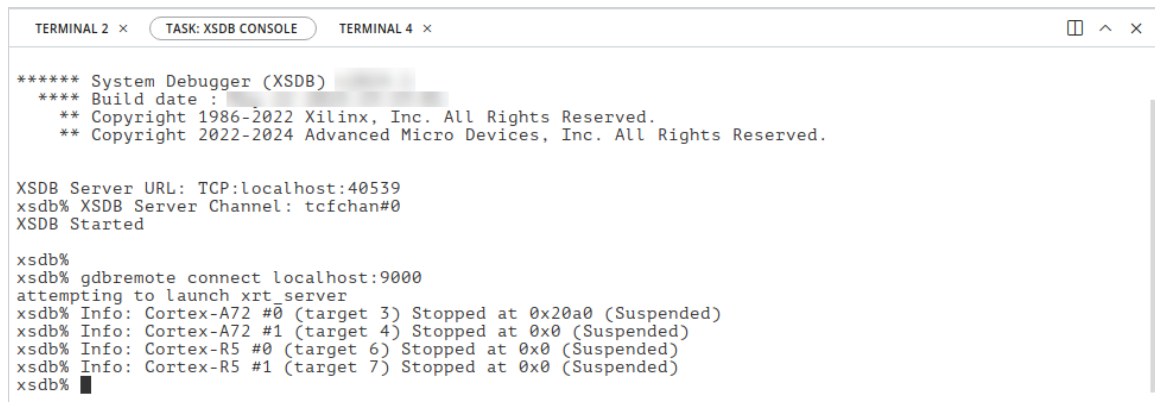
1-1-1. Select **Vitis** > **XSDB Console** to launch the XSDB console.

1-1-2. After the console launches, enter the following command to set up a TCF channel:

Note: You may have to press **<Enter>** in order to type in the XSDB console.

xsdb% gdbremote connect localhost:9000

Note: If `connection refused` messages appear after running the above command, replace 9000 with 9001 and try again.

Figure 188: Setting Up a TCF Connection

1-1-3. Enter the following command to display the currently running TCF channel:

xsdb% conn –list

Figure 189: List of Running TCF Connections

Note down the port number—it will be needed for the instructions that follow.

1-2. **Create a new target connection for the Hardware Server.**

1-2-1. Select **Vitis** > **Target Connections** to create a new target connection.

AMD
together we advance_

1-2-2. Click the **+** icon for a new target connection.
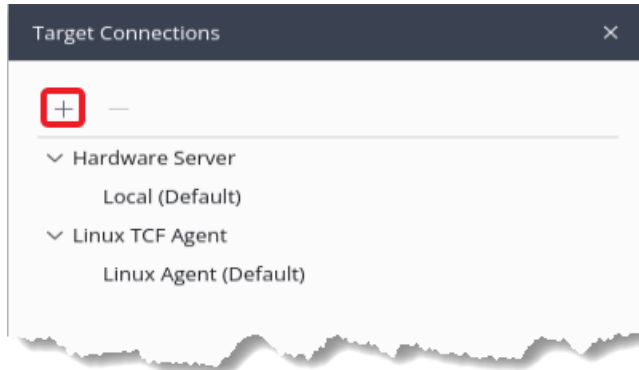
Figure 190: Creating a New Target Connection

The Target Connection Details dialog box opens.

1-2-3. Enter **qemu_debug** in the Name field (1).

1-2-4. Ensure that **Hardware Server** is selected as the type (2).

1-2-5. Enter **127.0.0.1** as the host IP (3).

1-2-6. For the port number, enter the value that you noted down in the previous instructions (4).
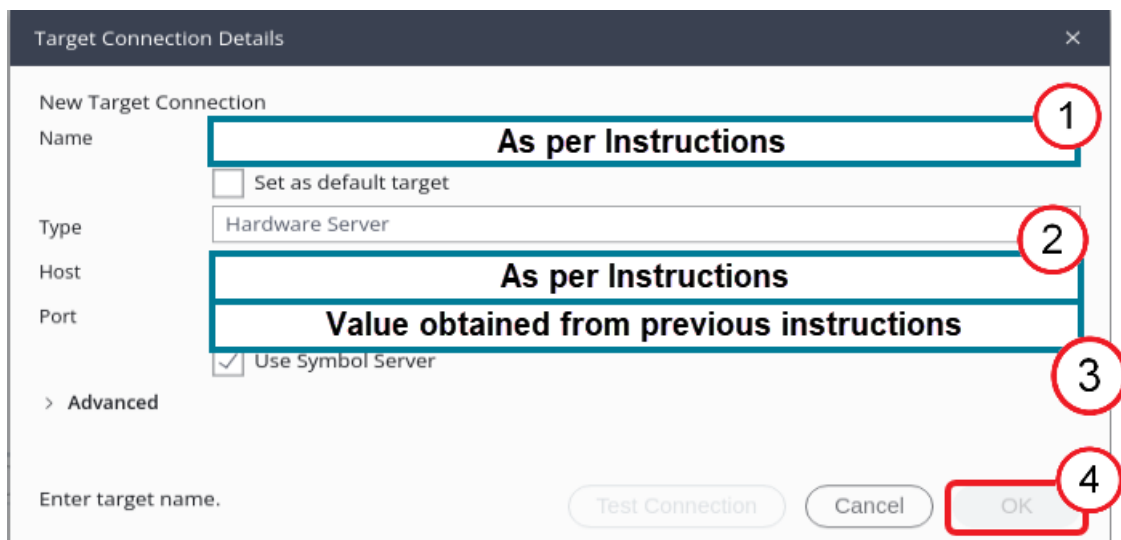
Figure 191: Configuring the Target Connection

1-2-7. Click **Test Connection** to verify whether the connection is established.

1-2-8. Click **OK** twice to exit the Target Connection Details dialog box (4).

1-2-9. Close the Target Connections window.

1-3. Create a new launch configuration.

1-3-1. Right-click in the empty space of the Explorer pane.

> **Note:** The Explorer pane can be opened by selecting **View** > **Explorer**.

1-3-2. Select **Edit Launch Configurations**.

1-3-3. Click **New Launch Configuration** to launch the Create Launch Configuration Wizard (1).

1-3-4. Select **Attach to Running Target** as the target setup mode (2).
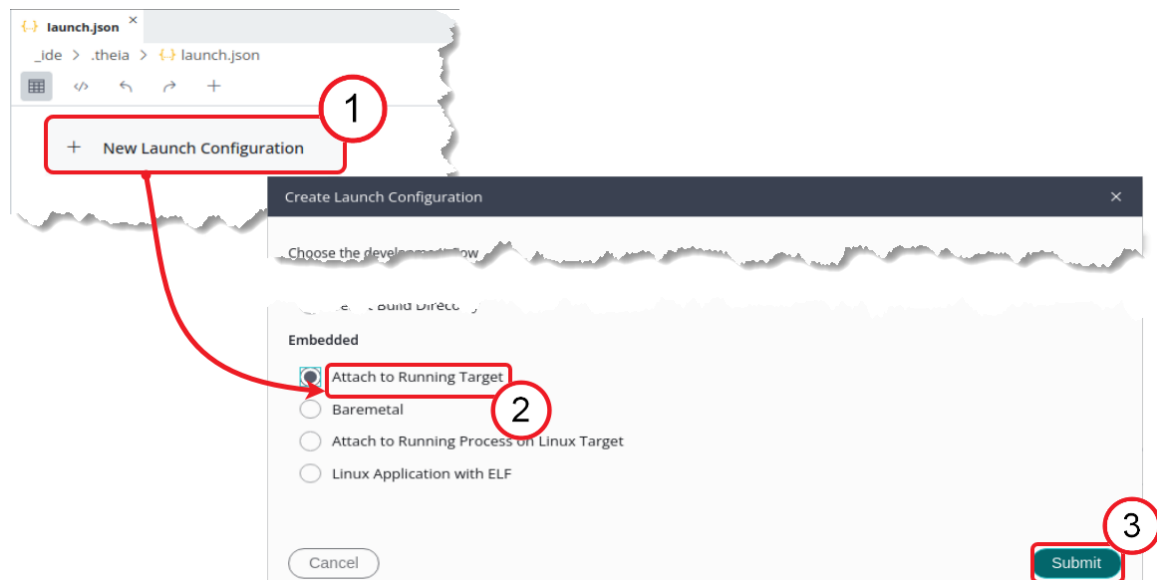
Figure 192: Creating a New Launch Configuration

1-3-5. Click **Submit** to close the Create Launch Configuration dialog box (3).

A launch configuration with default settings is generated.

1-3-6. Ensure that the Target Connection field points to the target connection created in the previous instructions (4).

1-3-7. Click the **Debug** icon to begin a debugging session (5).
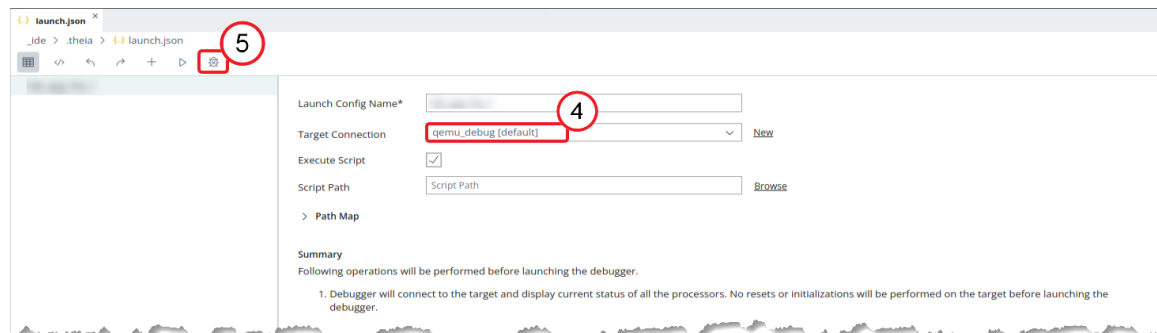
Figure 193: Newly Generated Launch Configuration

AMD
together we advance_

1-4. Add and download a symbol file.

The debugger has launched, but for it to proceed, a symbol file needs to be added. A symbol file is the application ELF file created by the tool.

1-4-1. Expand the **Debug** window (1).

The Debug window lists the processors available in the target board you are emulating.

1-4-2. Right-click **your processor** (2).

1-4-3. Select **Manage Symbol Files** (3).
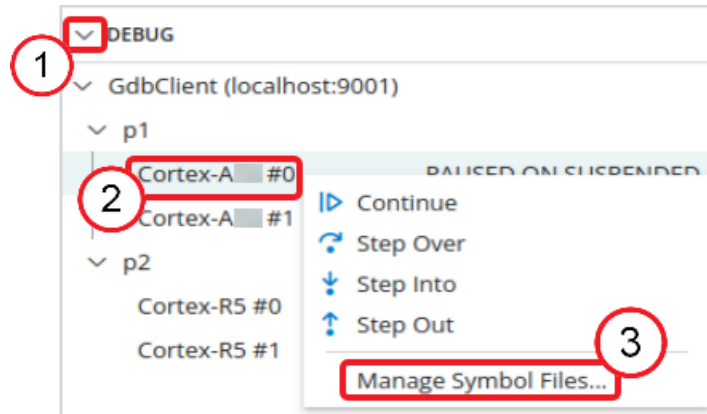
Figure 194: Adding a Symbol File to the Debugger

1-4-4. Click the '**+**' symbol.

1-4-5. Navigate to the following directory:

```
location of the ELF file
```

1-4-6. Double-click **name of the ELF file**.

1-4-7. Click **OK** to close the Symbol Files dialog box.

1-4-8. In the XSDB console, enter the following command to download the ELF file into it:

xsdb% dow location of the ELF file/name of the ELF file

Note: $TRAINING_PATH may need to be expanded.

The debugger is now configured and launched in the main source file. The following message appears on the XSDB console:
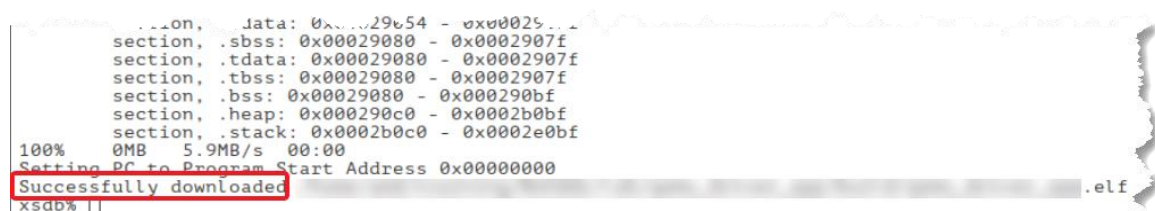
Figure 195: Successful Download Message

Switching Views from the Toolbar

1-1. **The Vitis toolbar on the left of the IDE is a collection of views that assist you with a specific task. There are several predefined views, including the Vitis Components view for developing applications and a Debug view for debugging.**

1-1-1. Locate the icon for the view that you would like to switch to in the toolbar on the left side of the IDE.

Figure 196: Different Views in the Toolbar

1-1-2. Click the desired icon.

Known Issue - ZCU104 - xparameters

Known issue: For applications using the serialPort_helper.c file as part of the source files (observed in labs using ZCU104):

Sometimes, because of the way the xparameters.h file is generated, the block of code that contains the definitions of UART base addresses may not be pointing to the right constant definitions in the xparameters file. This can be observed when you're unable to give keyboard inputs in the Terminal.

If this happens, you will need to:

- Manually locate the xparameters.h file located in the following location:

```
$TRAINING_PATH/<the topic cluster name>/lab/your platform project
name/psu_cortexa53_0/
standalone_psu_cortexa53_0/bsp/include
```

- Search for **/* Definitions for peripheral UART0 */** (1).

- Locate the constant definition that holds the base address ending with **BASEADDR** (2).

- Copy the definition.

AMD
together we advance_

- Replace the current value of UART_BASE_ADDRESS with the correct base address definition you just identified.
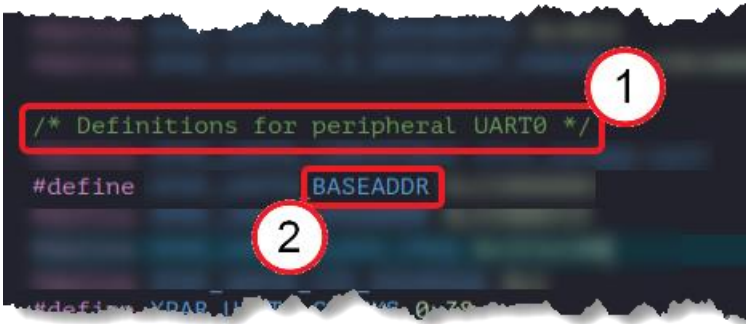
Figure 197: Known Issue – serialPort_helper Definitions

Debugging

In This Section

Debug Controls

The Debug toolbar icons provide easy access to the various debugger features. Take a moment to familiarize yourself with the toolbar location, icons, and their functions. Note that some of the more popular icons have function keys associated with them.
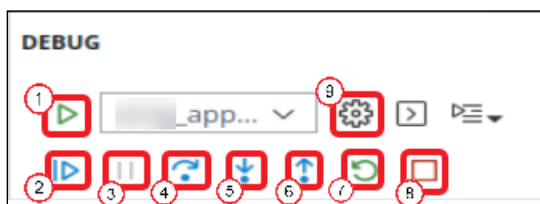
Figure 197: Debug view options

- o Start Debugger: Launches the debugging process (1).
- o Continue: Allows program execution to continue from a current breakpoint or paused state (2).
- o Pause: Enables pausing the execution of the program—usually to inspect variables or to stop the program from running further (3).

o   Step Over: Executes the current line of code and moves to the next line but does not enter into function calls (4).

o   Step Into: Moves the debugger to the next line of code, entering function calls to debug them step by step (5).

o   Step Out: Allows the debugger to execute until it exits the current function, then stops at the line immediately after the function call (6).

o   Restart: Restarts the debugger by placing it in the beginning of the application (7).

o   Stop: Stops the debugging process and disconnects the debugger from the target (8).

o   Debugger settings: Opens the debugger configuration settings (9).

Running the Debugger on Hardware

There are many options as to how to set up a Debug configuration. Often, the default options are enough when you just want to quickly run your application in debug mode.

1-1.   Run the debugger from the Flow view.

1-1-1.   Select **your application project name** from the Component drop-down list from the Flow view (1).

Ensure that the application has been built by verifying that there is a green check mark next to Build.
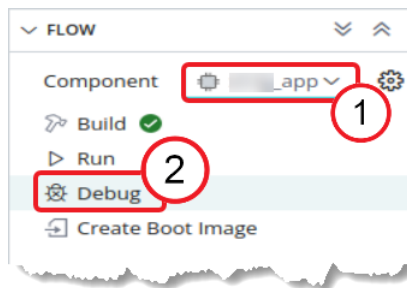
Figure 198: Running the Debugger

1-1-2.   Click **Debug** (2).

The Vitis Unified IDE will start programming the board.

This might take a few minutes to finish. The Debug view will automatically open.

AMD
together we advance_

Breakpoints

In This Section

Opening the Breakpoints View

1-1.   Open the Breakpoints view.

1-1-1. Select the **BREAKPOINTS** view in the Debug window to see all of the automatically generated breakpoints as well as any breakpoints that you have created.

Tip: You can reopen the view by selecting Debug.
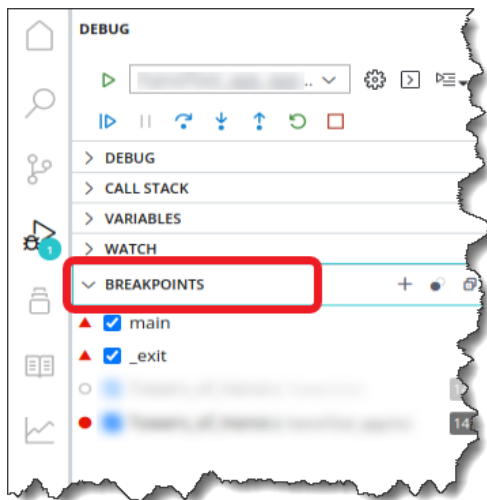
Figure 199: Locating the Breakpoints View

Running to a Breakpoint

1-1.   Run to the next breakpoint.

1-1-1. Click the **Continue** icon (⏵) to continue operation.

The application will run until an unconditional breakpoint is met or a conditional breakpoint whose condition is satisfied is met.

Closing the Vitis IDE

1-1. Close the Vitis Unified IDE.

1-1-1. Select **File** > **Close Window** to close the tool.

Vitis Unified IDE HLS Component Operations

This section contains instructions for commonly performed tasks using the Vitis High-Level Synthesis (HLS) tool.

In This Section

Creating an HLS Component

1-1. Create the HLS component for *your HLS project name.*

1-1-1. Navigate to the toolbar and click **File** (1).

1-1-2. Hover over **New Component** (2).
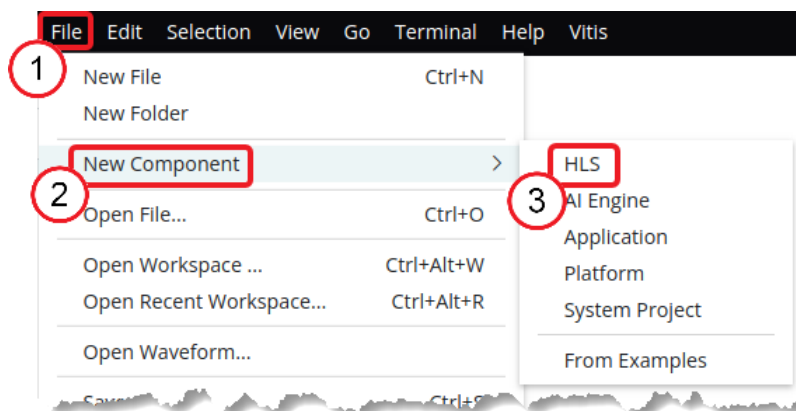
Figure 200: Creating a New HLS Component

1-1-3. Select **HLS** from the options to create the HLS component (3).

The Create HLS Component dialog box appears.

AMD
together we advance_

1-1-4. Enter **your HLS project name** in the Component name field (1).

This sets the name of the HLS component, and the associated directories and files are then populated within the current workspace.

Note that the Component location field is automatically set to the workspace location (2).

Figure 201: Choosing the Component Name and Location

1-1-5. Click **Next** to advance to the Configuration File page (3).

1-1-6. Select **Empty File** (1).

The remaining fields can be left at their default settings.

Note that the new configuration name will be set to hls_config.

Figure 202: Selecting the Configuration File

1-1-7. Click **Next** to advance to the Source Files page (2).

1-2. Add the source and testbench files to the HLS component.

1-2-1. Click the **Add Files** icon in the Design Files section (1).

1-2-2. Browse to the following directory:

`the directory where your files are located`

1-2-3. Select **your HLS source files** and click **Open**.

1-2-4. Click **Browse** next to the Top Function field (2).

1-2-5. Select the **name of the top function** kernel function (3) and click **OK** (4).

1-2-6. Click the **Add Files** icon in the Test Bench Files section (5).

1-2-7. Browse to the following directory:

`the directory where your files are located`

1-2-8. Select **your HLS testbench files** and click **Open**.
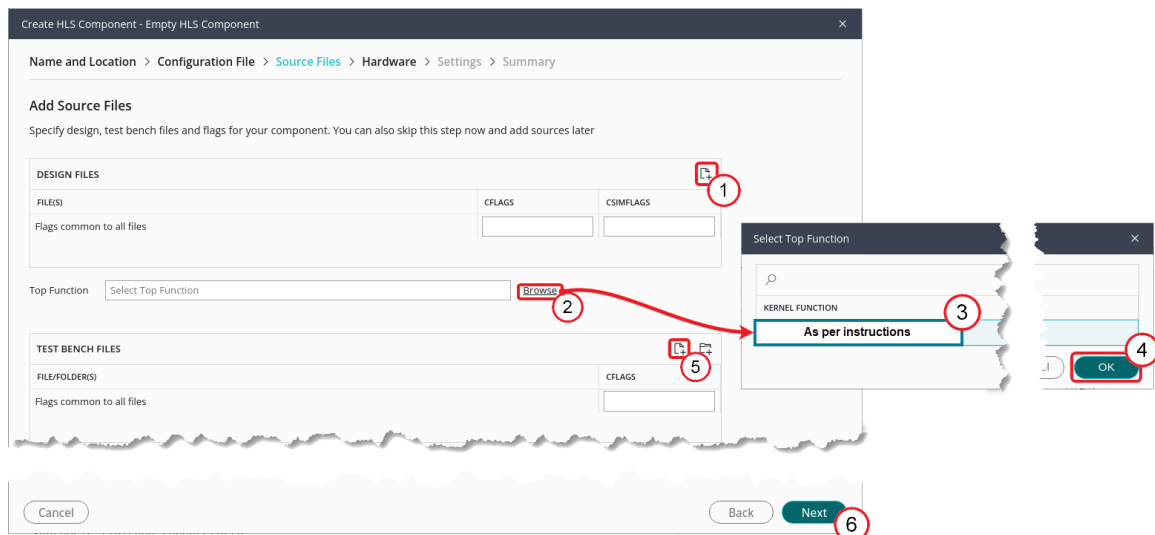
Figure 203: Adding the Source and Test Bench Files

1-2-9. Click **Next** to advance to the Part selection page (6).

AMD
together we advance_

1-3. Specify the part number and settings for the HLS component.

1-3-1. Ensure that the **Part** option is selected (1).

1-3-2. Select **the part number** from the list of parts (2).

Figure 204: Adding the Part Number to the HLS Component

1-3-3. Click **Next** to advance to the Settings page (3).

1-3-4. Enter **your clock period** for the clock setting (1).

1-3-5. Select **Vivado IP Flow Target** from the flow_target drop-down list (2).

Note: By default, the package.output.format option is selected as **Generate Vivado IP and .zip archive**.

Figure 205: Specifying the HLS Settings

1-3-6. Click **Next** to advance to the Summary page (3).

1-3-7. Review the HLS component summary and click **Finish**.

Simulating an HLS Component

1-1. Simulate the design.

1-1-1. Go to the **Flow** view (1).

The Flow view displays C Simulation, C Synthesis, C/RTL Cosimulation, Package, and Implementation as the primary steps of the HLS component workflow.

1-1-2. Select the **active HLS component** to simulate from the Component drop-down list (2).

Note: You can have multiple components in a system project. These components will be displayed here.

Figure 206: Running C Simulation

1-1-3. Click **Run** under C Simulation (3).

You can view the simulation log in the Output tab.

AMD
together we advance_

Synthesizing an HLS Component

1-1.   Synthesize the design.

1-1-1. Go to the **Flow** view (1).

The Flow view displays C Simulation, C Synthesis, C/RTL Cosimulation, Package, and Implementation as the primary steps of the HLS component workflow.

1-1-2. Select the **active HLS component** to synthesize from the Component drop-down list (2).

Note: You can have multiple components in a system project. These components will be displayed here.

Figure 207: Running C Synthesis

1-1-3. Click **Run** under C synthesis (3).

You can view the synthesis log in the Output tab.

Implementing an HLS Component

1-1.   Implement the design.

1-1-1. Go to the **Flow** view (1).

The Flow view displays C Simulation, C Synthesis, C/RTL Cosimulation, Package, and Implementation as the primary steps of the HLS component workflow.

1-1-2. Select the **active HLS component** to simulate from the Component drop-down list (2).

Note: You can have multiple components in a system project. These components will be displayed here.
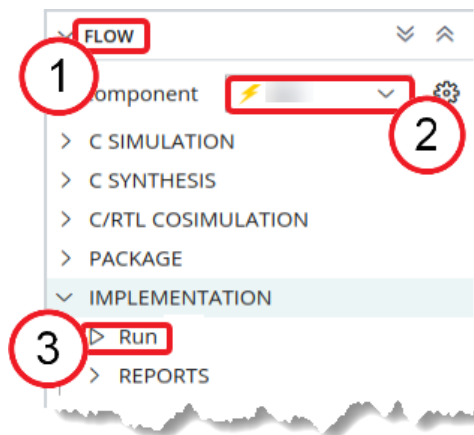
Figure 208: Running Implementation

1-1-3. Click **Run** under Implementation (3).

You can view the implementation log in the Output tab.

AMD
together we advance_

Running C/RTL Cosimulation with Trace Level

1-1.   Set the trace level for C/RTL cosimulation.

1-1-1. Expand **your HLS project name** > **Settings** in the Vitis Components window (1).

1-1-2. Click **hls_config.cfg** to open the configuration settings.

1-1-3. Scroll down and click **C/RTL Cosimulation** (2).

1-1-4. Scroll on the right side and locate **cosim.trace_level**.

1-1-5. Click the down arrow to open the list of trace levels.

1-1-6. Select **all** from the list of trace levels (3).
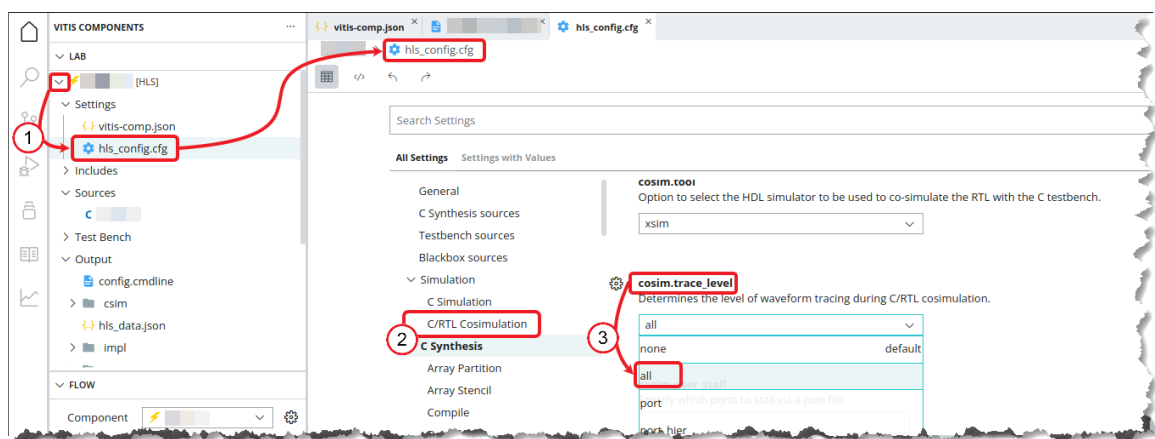
Figure 209: Configuration Settings for C-RTL Cosimulation

The trace_level option specifies the level of trace file output written to the *sim/Verilog* or *sim/VHDL* directory of the current solution when the simulation executes.

1-2. Run C/RTL cosimulation for *your HLS project name.*

1-2-1. Go to the **Flow** view (1).

The Flow view displays C Simulation, C Synthesis, C/RTL Cosimulation, Package, and Implementation as the primary steps of the HLS component workflow.

1-2-2. Select the **active HLS component** to simulate from the Component drop-down list (2).

Note: You can have multiple components in a system project. These components will be displayed here.
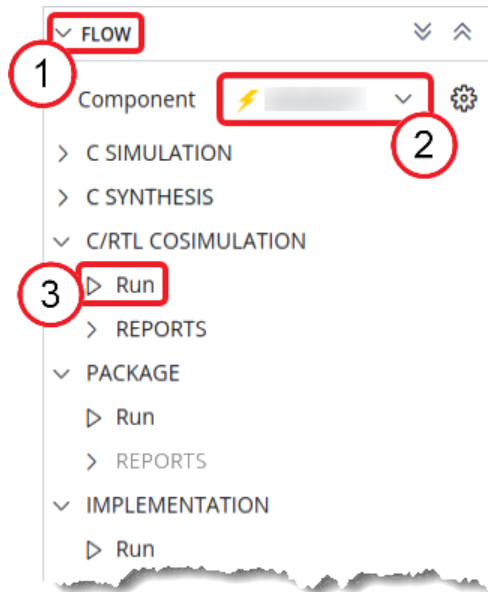
Figure 210: Running C/RTL Cosimulation

1-2-3. Click **Run** under C/RTL Cosimulation (3).

You can view the simulation log in the Output tab.

AMD
together we advance_

Setting the Trace Level for C/RTL Cosimulation

1-1.   Set the trace level for C/RTL cosimulation.

1-1-1. Expand **your HLS project name** > **Settings** in the Vitis Components window (1).

1-1-2. Click **hls_config.cfg** to open the configuration settings.

1-1-3. Scroll down and click **C/RTL Cosimulation** (2).

1-1-4. Scroll on the right side and locate **cosim.trace_level**.

1-1-5. Click the down arrow to open the list of trace levels.

1-1-6. Select **all** from the list of trace levels (3).
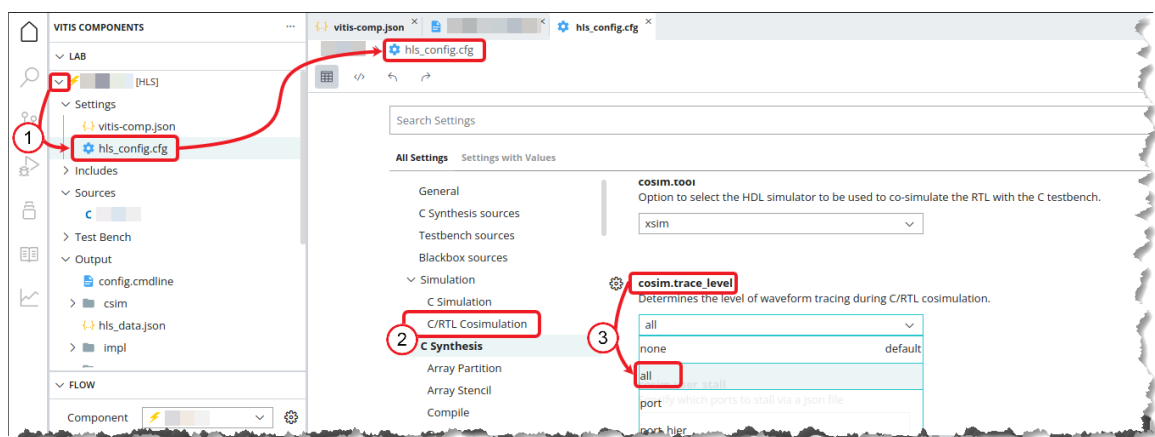
Figure 211: Configuration Settings for C-RTL Cosimulation

The trace_level option specifies the level of trace file output written to the *sim/Verilog* or *sim/VHDL* directory of the current solution when the simulation executes.

Running C/RTL Cosimulation

1-1.   Run C/RTL cosimulation for *your HLS project name*.

1-1-1. Go to the **Flow** view (1).

The Flow view displays C Simulation, C Synthesis, C/RTL Cosimulation, and Implementation as the primary steps of the HLS component workflow.

1-1-2. Select the **active HLS component** to simulate from the Component drop-down list (2).

Note: You can have multiple components in a system project. These components will be displayed here.

1-1-3. Click **Run** under C/RTL Cosimulation (3).

You can view the simulation log in the Output tab.

AMD
together we advance_
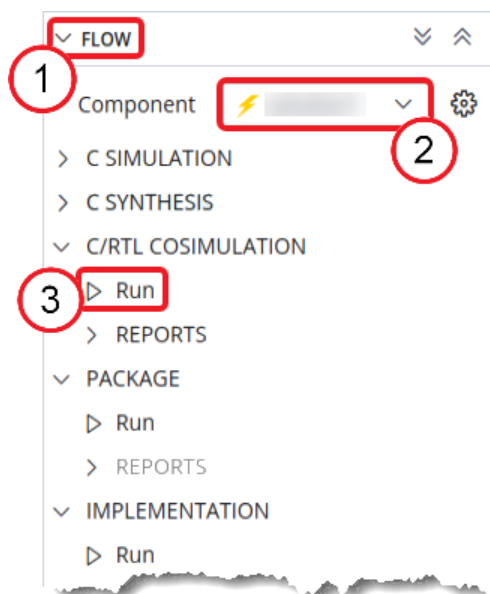
Packaging RTL as IP

1-1. Package the RTL as an IP core.
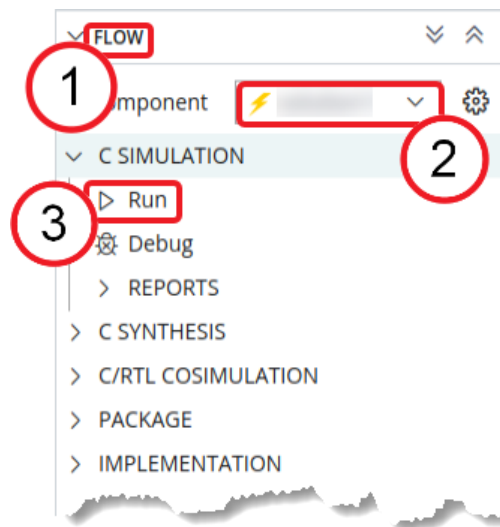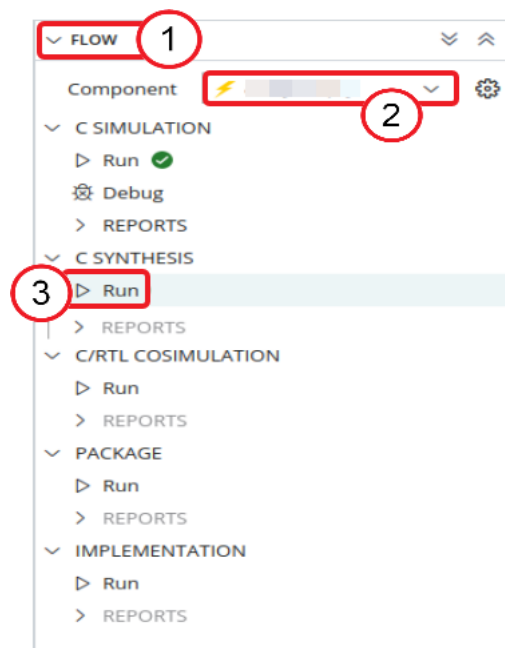
1-1-1. Go to the **Flow** view (1).

The Flow view displays C Simulation, C Synthesis, C/RTL Cosimulation, Package, and Implementation as the primary steps of the HLS component workflow.

1-1-2. Select the **active HLS component** to package from the Component drop-down list (2).

Note: You can have multiple components in a system project. These components will be displayed here.
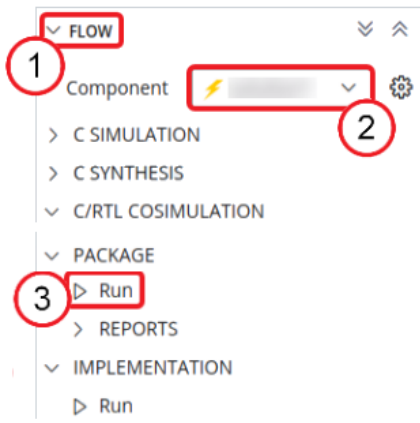
1-1-3. Click **Run** under Package (3).

You can view the log in the Output tab.

QEMU Operations

In This Section

Terminating QEMU in the Embedded Software Flow

1-1. Terminate the program.

1-1-1. Press <**Ctrl + A**> followed by <**X**> to terminate the current QEMU session.

PetaLinux Operations

In This Section

Creating a New PetaLinux Project (Using Any BSP with Project Name)

1-1. Use the `petalinux-create` command to create a new embedded Linux for the chosen platform.

1-1-1. Change to the lab directory:

`[host] $ cd $TRAINING_PATH/<the topic cluster name>/lab`

This is done so that when the `petalinux-create` command is run, the project is created at this location.

1-1-2. Create a new PetaLinux project:

`[host] $ petalinux-create -t project -s your BSP name -n your project name`

Note: After the command is executed, information about the created project will be displayed.

Note: The PetaLinux project directory is `$TRAINING_PATH/<the topic cluster name>/lab/your project name` and this is where all work for this project must be performed.

1-1-3. Change the directory to the PetaLinux project:

`[host] $ cd your project name`

AMD
together we advance_

Creating a New Application

1-1. **Create a new application.**

The PetaLinux tools allow you to create user applications using predefined C, C++, and autoconf templates. The autoconf template is used for GNU autoconfig.

These templates include application source code and makefiles so that you can easily configure and compile applications for the target and install them into the root file system.

1-1-1. Change the directory to the PetaLinux project:

```
[host] $ cd $TRAINING_PATH/<the topic cluster name>/lab/your
project name
```

1-1-2. Enter the following command to create a new user application inside the PetaLinux project:

```
[host] $ petalinux-create -t apps --name your application
--template c
```

Creating a New Application (Enable App)

1-1. **Create a new application and enable it in the root file system.**

The PetaLinux tools allow you to create user applications using predefined C, C++, and autoconf templates. The autoconf template is used for GNU autoconfig.

These templates include application source code and makefiles so that you can easily configure and compile applications for the target and install them into the root file system.

1-1-1. Make sure that you are in the PetaLinux project directory as follows:

```
$TRAINING_PATH/<the topic cluster name>/lab/the name of the
PetaLinux project
```

1-1-2. Enter the following command to create a new user application inside a PetaLinux project:

```
[host] $ petalinux-create -t apps --name your application --
enable
```

The `--enable` option automatically enables it in the project's root file system. To create a C++ application template, pass the `--template c++` option.

Creating a New Application (Enable App) Using Any BSP

1-1. **Create a new application and enable it in the root file system.**

The PetaLinux tools support multiple predefined templates, enabling you to more easily create a project tailored to your needs. Here you will create a user application from the default _C_ template.

Other templates exist to facilitate other types of projects. For more details, refer to UG1144: _PetaLinux Tools Documentation: Reference Guide._ All templates include application source code and makefiles for easy configuration and compilation.

1-1-1. Make sure that you are in the PetaLinux project directory as follows:

```
$TRAINING_PATH/<the topic cluster name>/lab/the name of the
PetaLinux project
```

1-1-2. Enter the following command to create a new user application inside a PetaLinux project:

`[host] $ petalinux-create -t apps --name the desired source file --enable`

Note: After execution the following information will be printed.

```
INFO: Create apps: the desired source file
```

```
INFO: New apps successfully created in $TRAINING_PATH/<the topic
cluster name>/lab/your project name/project-spec/meta-
user/recipes-apps/your application project name
```

```
INFO: Enabling created component...
```

```
INFO: Sourcing build environment
```

```
INFO: Silentconfig rootfs
```

```
INFO: your application project name has been enabled
```

Note: The new application that you have created can be found in the `<project-root>/project-spec/meta-user/recipes-apps/your application project name` directory, where `<project-root>` is `$TRAINING_PATH/<the topic cluster name>/lab/your project name`.

The `--enable` option automatically enables the just created application in the project's root file system. To create a C++ application template, pass `--template c++` as another parameter to the just entered command.

AMD
together we advance_

Selecting the New Application to be Included in the Build Process

1-1. **Select the new application to be included in the build process. The application is not enabled by default.**

1-1-1. Ensure that your current working directory (project directory) is the following:

$TRAINING_PATH/<the topic cluster name>/lab/your project name

1-1-2. Enter the following command to launch the rootfs configuration menu:

[host] $ petalinux-config -c rootfs

Note: Make sure that the terminal window is at least 80 columns wide. If your terminal is not at least 80 columns wide, you will see "Error: Failed is config linux/rootfs! Your display is too small to run menuconfig."

The Linux/rootfs configuration menu opens.

1-1-3. Press the **Down Arrow** key ⬇ to scroll down the menu to **user packages**.
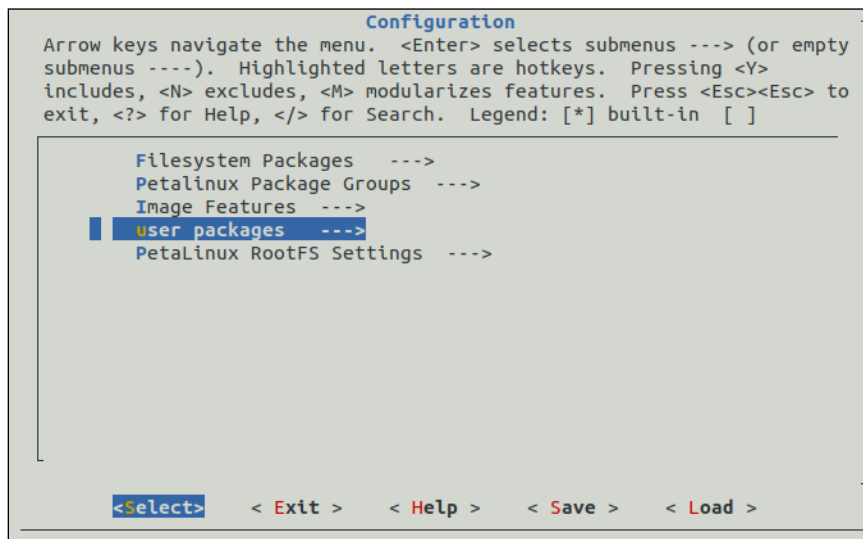
Figure 212: Linux/rootfs Configuration Menu

1-1-4. Press <**Enter**> to go into the user packages submenu.

The new application **your application** is listed in the menu.

1-1-5. Press the <**Space Bar**> to select the application **your application**.

Selecting the application will be included in the build process.

1-1-6. Select **Exit** and save the configuration.

Building the Linux Image

1-1.   Build the Linux image.

This process is time consuming, typically taking 60 minutes or more depending on your system configuration. If you have time, you can build the image yourself with the process provided here, or you can skip this instruction and proceed with the next step in the lab.

Extremely important! Before running the `petalinux-build` command, make sure that you are connected to the internet.

1-1-1. Make sure that you are in the root of the PetaLinux project directory:

```
[host] $ pwd
```

If you are not at the proper location, you can navigate there by entering:

```
cd $TRAINING_PATH/<the topic cluster name>/lab/your project name
```

1-1-2. Enter the following command to build the image:

```
[host] $ petalinux-build
```
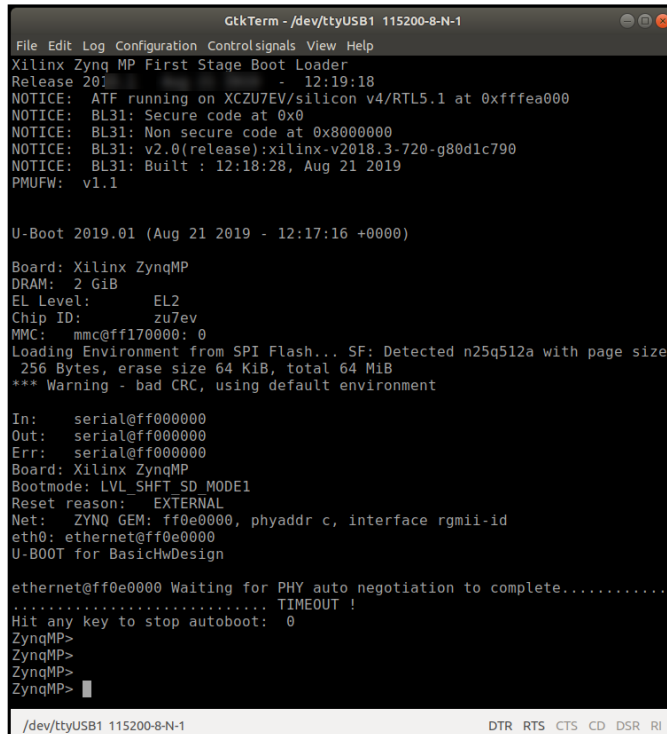
Running `petalinux-build` in the project directory will build the system image, including the selected user application `your project name`.

The full compilation log (`build.log`) is stored in the build subdirectory of the PetaLinux project. The Linux software images and the device tree are generated in the `<plnx_project_root>/images/linux` subdirectory of the PetaLinux project.

AMD
together we advance_

Booting the Linux Image on the Board (Running Netboot)

1-1. Boot the new Linux image on the board.

1-1-1. Press any key to stop auto-boot when you see messages similar to the following in the GtkTerm window:

Figure 213: Stopping the Autoboot (ZedBoard)

If the system has booted, reset the board (BTN7) again and stop auto-boot.

Note: If you have finished setting the IP address and server IP address and saved once, you can directly run the command `run netboot`.

1-1-2. Set the IP address for the target board so that it can communicate to the host and load the new image:

`Zynq> setenv ipaddr 192.168.1.10`

1-1-3. Set the TFTP server IP to the host IP:

`Zynq> setenv serverip 192.168.1.1`

1-1-4. Save the environment to the SPI Flash:

`Zynq> saveenv`

1-1-5. Download and boot the new image using `TFTP`:

`Zynq> run netboot`

This command will download the `image.ub` file from `/tftpboot` on the host to the main memory of the ARM Cortex MPCore system and boot the system with the image.

Running the Application in QEMU

1-1. Run the application in QEMU.

1-1-1. Enter the following command to boot the newly built PetaLinux image through QEMU:

```
# petalinux-boot --qemu --kernel
```

1-1-2. After the system boots, log into the system by entering `root` as the both the login name and password.

Configuring Network Settings

1-1. Launch VirtualBox.

1-1-1. Select **Start Menu** > **All Programs** > **Oracle VM VirtualBox** > **Oracle VM VirtualBox** to start the VirtualBox Manager program.

1-1-2. Select the **name of the virtual machine** virtual machine titled from the left pane (1).

1-1-3. Double-click the virtual machine (1) or click the green right arrow icon to launch the selected virtual machine (2).
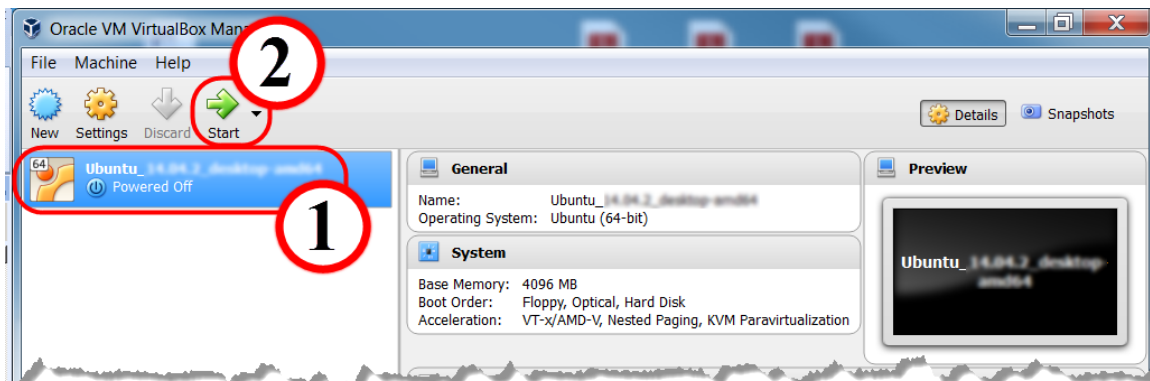
Figure 214: VirtualBox Manager Window

The name of the virtual machine virtual machine will now launch in a new window.

1-1-4. Click the **Maximize** (🗖) icon in the window title bar to make it full screen if the virtual machine window is not maximized.

AMD
together we advance_

1-2. Disable the static IP settings.

1-2-1. Press <**Ctrl + Alt + T**> to open a new terminal window.

1-2-2. Enter the following command to open the interface file:

```
gedit /etc/network/interfaces
```

1-2-3. Comment all the lines in the interfaces file except the two lines as shown below:

```
#interfaces(5) file used by ifup(8) and ifdown(8)
auto lo
iface lo inet loopback
#auto eth0
#iface eth0 inet static
#address 192.168.1.1
#netmask 255.255.255.0
#broadcast 192.168.1.255
#gateway 192.168.1.
```

1-2-4. Press <**Ctrl + S**> to save the changes.

1-2-5. Click <**X**> in the upper-left corner of the gedit window to exit the editor.

1-2-6. Enter the following command in the terminal to shut down the OS:

```
sudo poweroff
```

1-2-7. Enter **xilinx** as password when asked to enter a password.

1-3. Change the VirtualBox network settings.

1-3-1. Select **Network** as shown in the figure below.
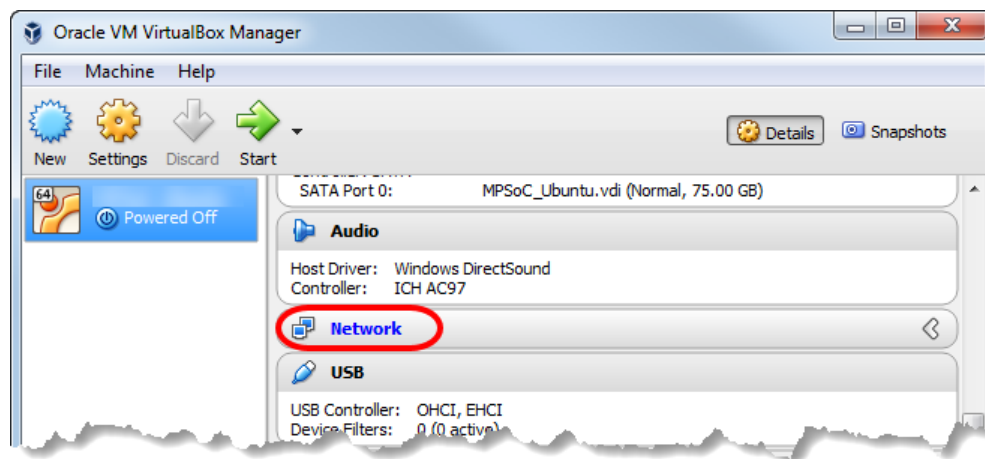
Figure 215: Selecting the Network Tab

1-3-2. Select the **Enable Network Adapter** option to enable the network adapter.
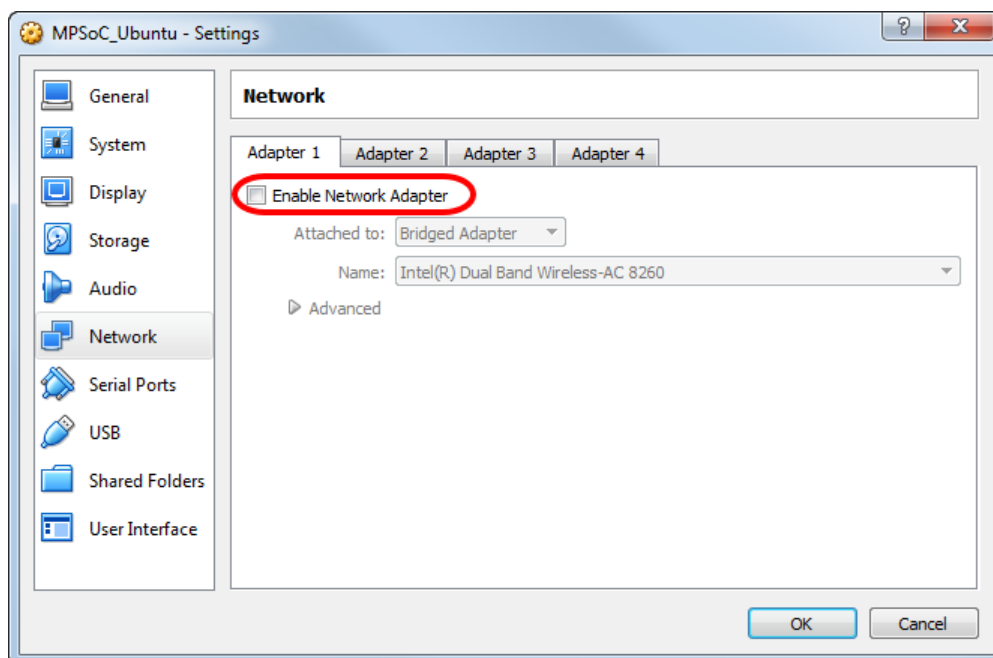
Figure 216: Enabling the Network Adapter

1-3-3. Select the appropriate name option from the drop-down list.

This will vary from system to system. The figures shown here are for illustration purposes only.
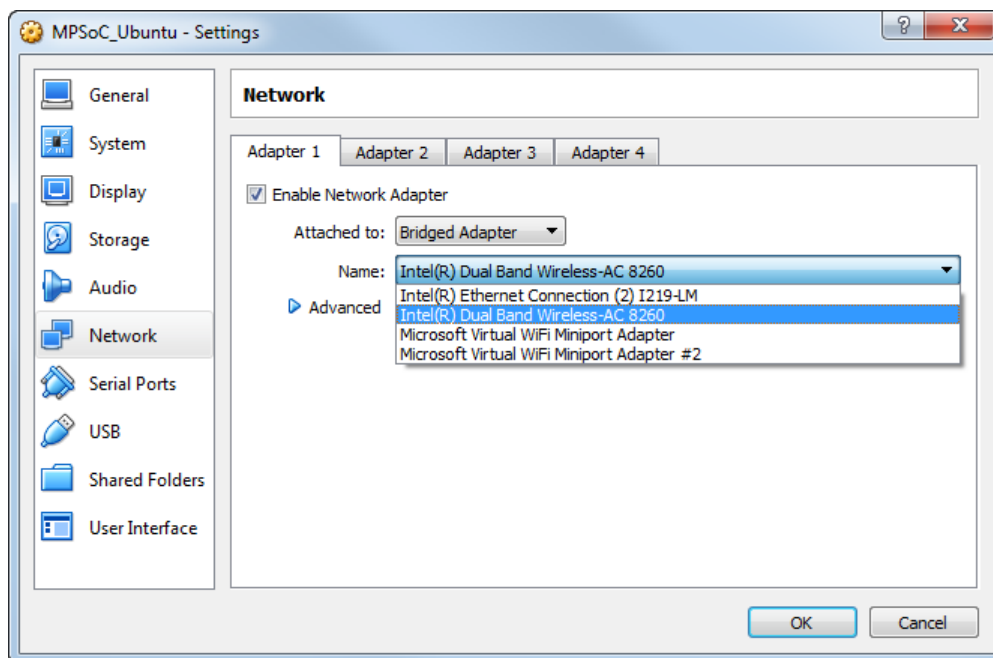
Figure 217: Selecting the Appropriate Network Connection

AMD
together we advance_

This example shows that **Intel(R) Dual Band Wireless-AC 8260** (or the wired network **Intel(R) Ethernet Connection (2) I2I9-LM**) is selected because it is the name of our network connection on the Windows machine running at the backend as shown below.
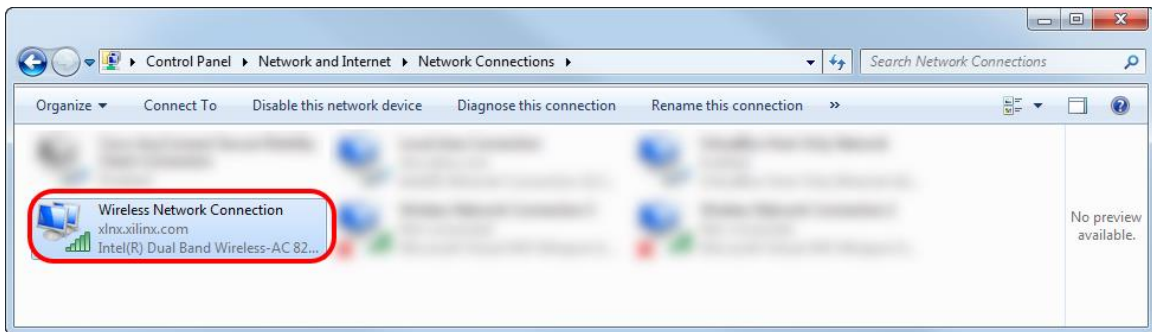
Figure 218: Network Connection on Windows Machine

You will have to check the network connection of your Windows machine and then select the appropriate name in the VirtualBox.

1-3-4. Select the **Advanced** option in the Adapter tab in the VirtualBox to show the extra configuration parameters.

1-3-5. Edit the configuration as shown below.
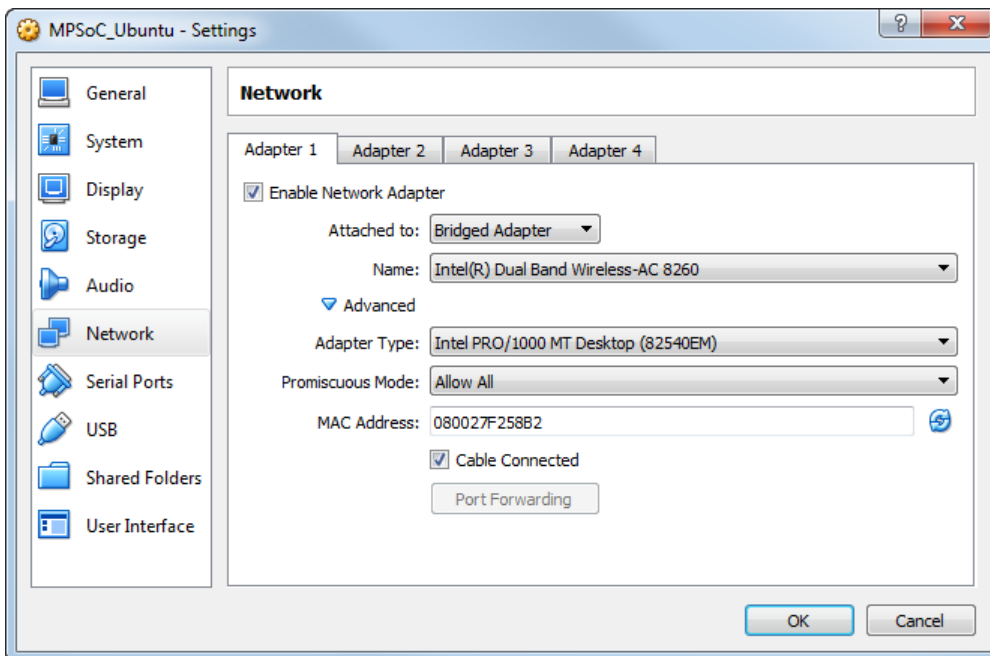
Figure 219: Configuring Advanced Options

Note: The network connection name may be different in your environment.

1-3-6. Click **OK** to bring the changes into effect.

Adding Ethernet Options

1-1. Add the Ethernet information in the device tree.

1-1-1. Change to the device tree folder:

[host] $ cd <project-root>/project-spec/meta-user/recipes-bsp/device-tree/files

Note: <project-root> here is the $<the topic cluster name>/lab/your project name directory.

1-1-2. Open the system-user.dtsi file:

[host] $ gedit system-user.dtsi

1-1-3. Add the following lines at the end of the file:

Note: You will find the completed system-user.dtsi in the $<the topic cluster name>/support directory.

```
&gem3 {
      status = "okay";
      local-mac-address = [00 0a 35 00 02 90];
      phy-mode = "rgmii-id";
      phy-handle = <&phy0>;
      phy0: phy@c {
            reg = <0xc>;
            ti,rx-internal-delay = <0x8>;
            ti,tx-internal-delay = <0xa>;
            ti,fifo-depth = <0x1>;
      };
};

&qspi {
      status = "okay";
      flash@0 {
            compatible = "m25p80"; /* 32MB */
            #address-cells = <1>;
            #size-cells = <1>;
            reg = <0x0>;
            spi-tx-bus-width = <1>;
            spi-rx-bus-width = <4>; /* FIXME also DUAL
configuration possible */
            spi-max-frequency = <108000000>; /* Based on DC1 spec
*/
            partition@qspi-fsbl-uboot { /* for testing purpose */
                  label = "qspi-fsbl-uboot";
                  reg = <0x0 0x100000>;
            };
            partition@qspi-linux { /* for testing purpose */
                  label = "qspi-linux";
                  reg = <0x100000 0x40000>;
            };
            partition@qspi-device-tree { /* for testing purpose */
```

AMD
together we advance_

```
                label = "qspi-device-tree";
                reg = <0x00140000 0x2000000>;
        };
        partition@qspi-rootfs { /* for testing purpose */
                label = "qspi-rootfs";
                reg = <0x02140000 0x80000>;
        };
    };
};
```

1-1-4. Save and close the file.

Board, OS, COM, and IP Address Tasks

In This Section

Configuring the Terminal

The Terminal tab is used with a serial source (UART), SSH, or Telnet session. The following instructions will show you how to configure this tab to work with a serial UART.

1-1.   Open the Terminal tab.

1-1-1. If the Terminal tab is not visible, select **Windows** > **Show View** > **Other** > **Terminal** > **Terminal** to open it or create a new Terminal tab.

This will open a Terminal tab in one of the panels. The tab can be moved to any other panel by click-dragging it to the new panel.

Note: More than one terminal can be enabled at a time.

1-1-2. Select the **Terminal** tab to access the terminal icons (1).

1-1-3. Click the **Settings** icon (▣) (2).
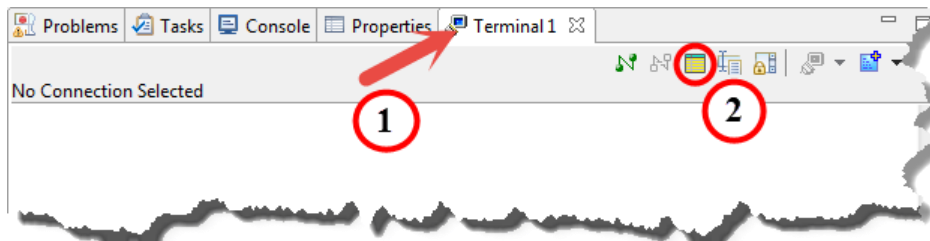
Figure 220: Accessing the Terminal Settings

Alternatively, you can also click the **Connect** icon (▣) to open the Terminal Settings dialog box. If the terminal was previously configured, this action will not open the Terminal Settings dialog box, rather it will attempt to open the communication channel with the previously selected settings.

Changing the settings requires that you close the channel and use the Settings icon.

1-2.   Configure the settings as shown in the following figure.

1-2-1. Select the connection type as **Serial**.

1-2-2. Select the proper port for the COM **#**.

Your board must be on and connected; otherwise, the USB bridge is not enumerated, and your COM port will not appear.

AMD
together we advance_

1-2-3. Set the baud rate to **115200**.
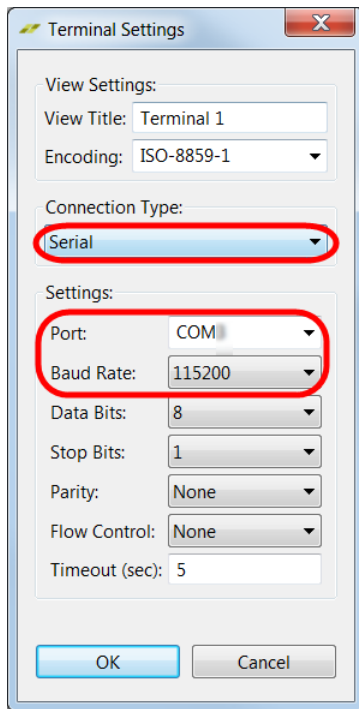
Figure 221: Configuring the Terminal Settings

1-2-4. Click **OK**.

The terminal session will be connected to the associated COM port on the PC.

Setting Up the ZCU104 Board

1-1. **Bring up the ZCU104 board.**

The figure below of the ZCU104 evaluation board enumerates some of its more popular features. Take a moment to familiarize yourself with the PCB location of these features. Before the board can be used, there are cables and switch settings that need to be in place and set (or just verified) to ensure proper board operation.

The steps that follow will reference items at the *numbered* PCB locations. Additional information can be found in the *ZCU104 Evaluation Board User Guide* (UG1267).
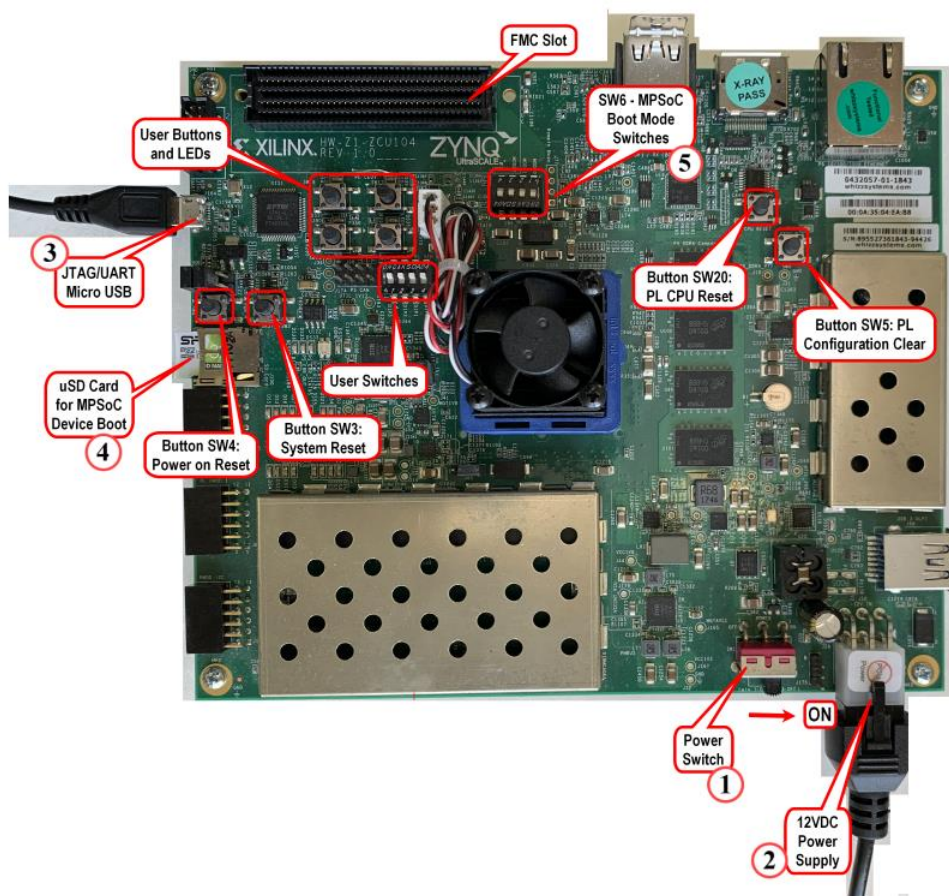
Figure 222: ZCU104 Overview

1-1-1. Ensure that the board is powered off by verifying that the power slide switch is in the position furthest from the power connector (1).

The figure above shows the switch in the "ON" position.

1-1-2. Ensure that the power connector is plugged in (2).

1-1-3. Connect the micro-USB JTAG/UART cable between the board and the host (3).

AMD
together we advance_

This USB cable from the host connects to a four-channel USB-to-UART bridge chip on the board:

- o Channel 1: JTAG serial device pins, recognized by the AMD JTAG driver.
- o Channel 2: PS UART0, recognized by host software as COM port COMx and is the most frequently used port.
- o Channel 3: PS UART1, recognized by host software as COM port COMx+1.
- o Channel 4: PL device pins, recognized by host software as COM port COMx+2.
  - ▪ TXD: Pin A20
  - ▪ RXD: Pin C19
  - ▪ RTS: Pin C18
  - ▪ CTS: Pin A19

1-1-4. If the boot mode is an SD card, insert the microSD card containing the Zynq UltraScale+ MPSoC device boot image (e.g., BOOT.bin) into the uSD card slot on the top side of the board (4).

1-1-5. Locate SW6 on the board (5).

This switch configures the MPSoC device's boot mode, which should be set to **the desired boot mode**.

1-1-6. Set SW6 as shown below to ensure that the board is configured to boot from **the desired boot mode**.

Note: Settings are also shown below to illustrate different boot mode settings. Make sure you select **the desired boot mode**.

Boot Mode	Mode Pins [3:0]	Mode SW6 [4:1]
JTAG	0000/0x0	ON,ON,ON,ON
QSPI32	0010/0x2[1]	ON,ON,OFF,ON
SD1	1110/0xE	OFF,OFF,OFF,ON

Figure 223: ZCU104 Boot Mode Switch Settings Table

Figure 224: ZCU104 Boot Mode Switch Settings

1-1-7. Slide the power switch to the "**ON**" position to power on the board.

You are now ready to configure the board.

Setting Up the VCK190 Board

1-1. **Bring up the VCK190 board.**

The figure below of the VCK190 evaluation board enumerates some of its more popular features. Take a moment to familiarize yourself with the PCB location of these features. Before the board can be used, there are cables and switch settings that need to be in place and set (or just verified) to ensure proper board operation. The steps that follow will reference items at the *numbered* PCB locations.

The VCK190 is more complex than most other boards as it incorporates a platform system controller using a Zynq UltraScale+ MPSoC along with Versal adaptive SoC technology. Additional information can be found in the *VCK190 Evaluation Board User Guide* (UG1366).

Figure 225: VCK190 Overview

AMD
together we advance_

1-1-1. Ensure that the board is powered off by verifying that the power slide switch is in the position closest to the power connector (1).

The figure above shows the switch in the "ON" position.

1-1-2. Ensure that the power connector is plugged in (2).
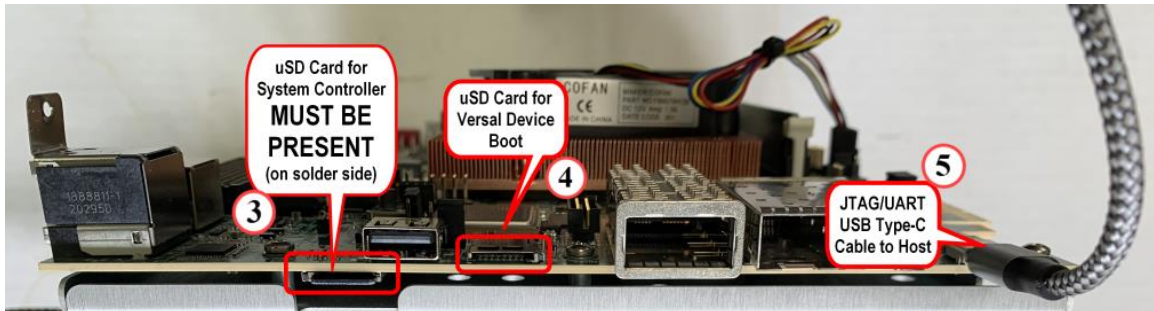
Figure 226: VCK190 uSD Cards and USB C Connector Locations

1-1-3. If not already present, insert the programmed system controller microSD card into the microSD card slot on the bottom side of the board (3).

Note: This is NOT the SD card that is used to boot the Versal device—instead, it boots an UltraScale+ device that is used for overall board management.

1-1-4. If the boot mode is an SD card, insert the microSD card containing the Versal device boot image (e.g., BOOT.pdi) into the uSD card slot on the top side of the board (4).

1-1-5. Connect the USB-C JTAG/UART cable between the board and the host (5).

This USB cable from the host connects to a four-channel USB-to-UART bridge chip on the board:

o   Channel 1: JTAG serial device pins, recognized by the AMD JTAG driver.

o   Channel 2: PS UART0, recognized by host software as COM port COMx and is the most frequently used port.

o   Channel 3: Versal PL device pins, recognized by host software as COM port COMx+1.

  ▪   TXD: Pin K33

  ▪   RXD: Pin L33

o   Channel 4: System Controller PS UART0, recognized by host software as COM port COMx+2.

Caution: SW11 (located in the 11 o'clock position relative to the Versal device in the figure below) controls the mode settings for the system controller. The switch settings MUST NOT be changed from the SD card boot setting (1=ON; 2,3,4=OFF); otherwise, the system controller will not boot, blocking the Versal device from booting.
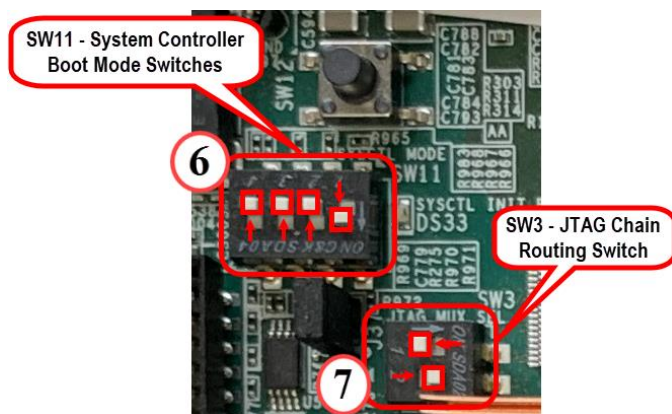
Figure 227: VCK190 SW11 and SW3 Settings

1-1-6. Verify that SW11 (which controls the boot mode for the system controller) is set as shown (6).

1-1-7. Verify that SW3 (which controls how the JTAG chain is routed) is set so that the switch marked "1" is off, and the switch marked "2" is on, as shown (7).

1-1-8. Locate SW1 on the board.

This switch configures the Versal device's boot mode, which should be set to **the desired boot mode**.

Set SW1 as shown below to ensure that the board is configured to boot from **the desired boot mode**.

Note: Settings are also shown below to illustrate different boot mode settings. Make sure you select the desired boot mode.

Boot Mode	Mode Pins [3:0]	Mode SW1 [4:1]
JTAG	0000 / 0x0	ON,ON,ON,ON
QSPI32	0010 / 0x2	ON,ON,OFF,ON
SD1	1110 / 0xE	OFF,OFF,OFF,ON
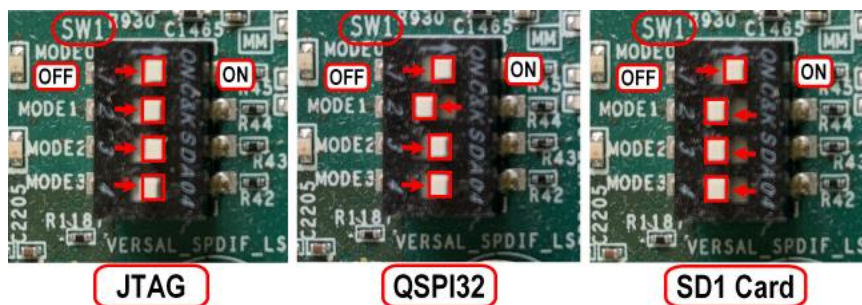
Figure 228: VCK190 Boot Mode Switch Settings

1-1-9. Slide the power switch to the "**ON**" position to power on the board.

You are now ready to configure the board.

AMD
together we advance_

Setting Up the ZC702 Board

1-1. Bring up the ZC702 board.

The figure below of the ZC702 evaluation board enumerates some of its more popular features. Take a moment to familiarize yourself with the PCB location of these features. Before the board can be used, there are cables and switch and jumper settings that need to be in place and set (or just verified) to ensure proper board operation.

The steps that follow will reference items at the *numbered* PCB locations. Additional information can be found in the *ZC702 Evaluation Board for the Zynq 7000 XC7020 SoC User Guide* (UG850).
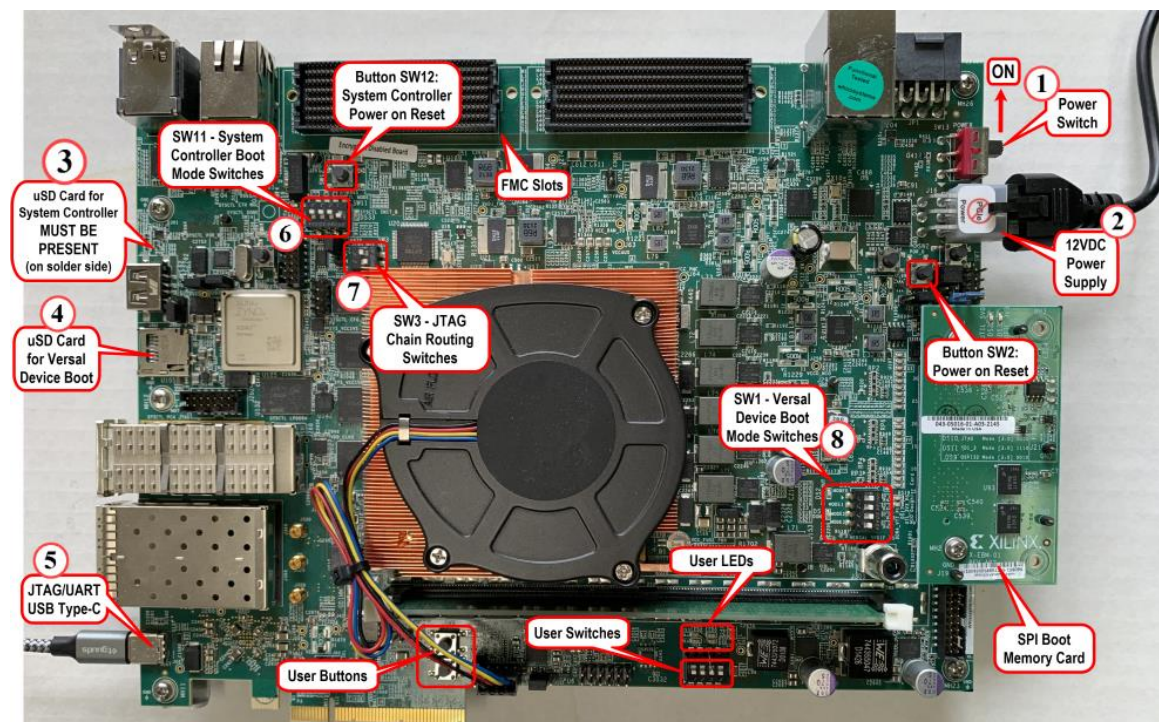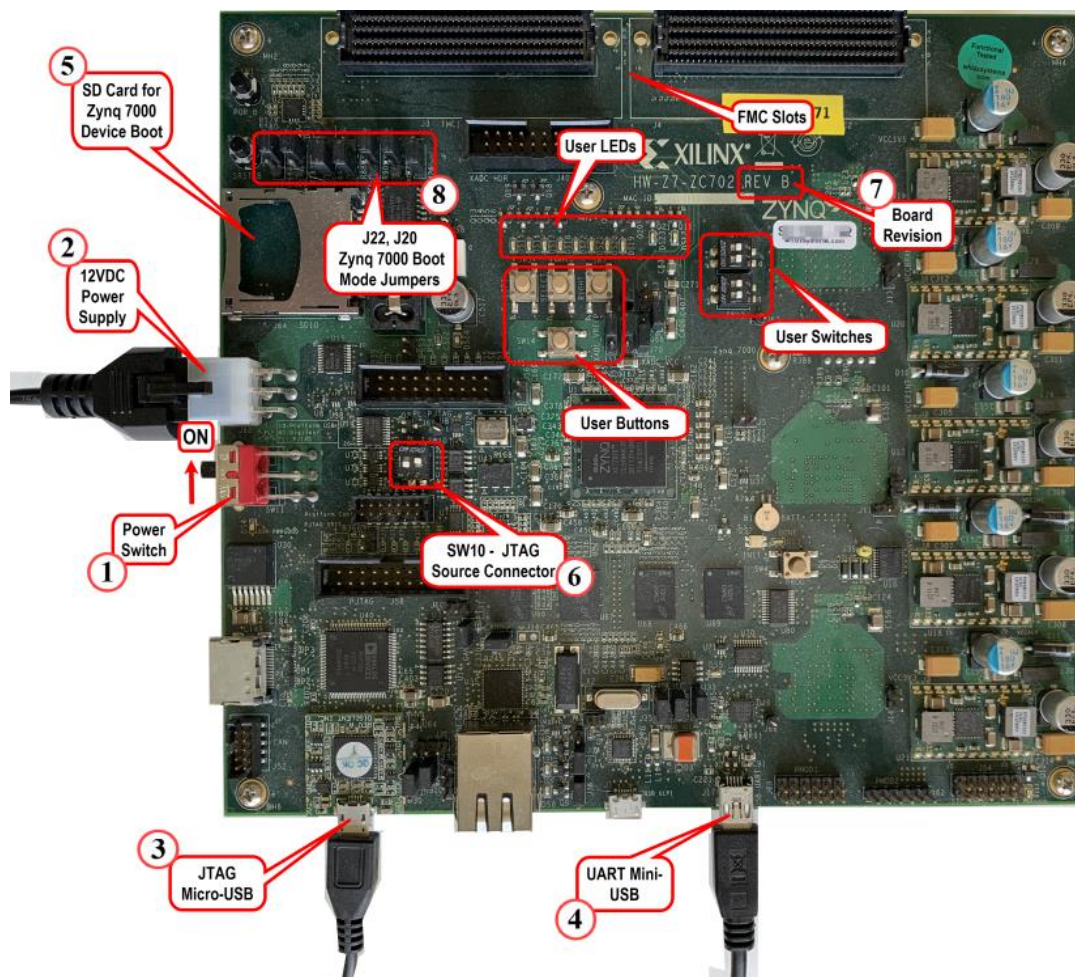

Figure 229: ZC702 Overview

1-1-1. Ensure that the board is powered off by verifying that the power slide switch is in the position furthest from the power connector (1).

The figure above shows the switch in the "ON" position.

1-1-2. Ensure that the power connector is plugged in (2).

1-1-3. Connect the micro-USB JTAG cable between the board and the host (3).

This connection implements JTAG communication to boot, download software object code, and configure the PL of the Zynq 7000 SoC.

1-1-4. Connect the mini-USB UART cable between the board and the host (4).

This connection uses a USB to UART bridge chip to implement host COMx port communication to PS UART1 in the Zynq 7000 SoC.

1-1-5. If the boot mode is an SD card, insert the SD card containing the Zynq 7000 SoC device boot image (e.g., BOOT.bin) into the SD card slot located on the top side of the board (5).

1-1-6. Locate the two-position dip switch, SW10 (6).

SW10 selects the Zynq 7000 SoC device JTAG source, which, for this lab, will be the onboard USB to JTAG daughter card that the micro USB cable is plugged into (other connector sources exist).

1-1-7. Ensure that SW10.1 is OFF and SW10.2 is ON (6).
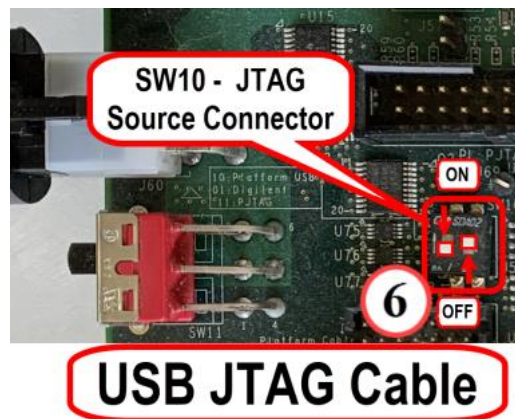
Figure 230: ZC702 SW10 JTAG Source Settings

Note: Earlier ZC702 boards with a *letter* `REV` designation use shunt jumpers to configure the boot mode. Later versions with a *number* `REV` designation replace five of the configuration shunt with a dip switch. Both `REV` configurations are shown below.

1-1-8. Locate and note the board REV, either *letter* or *number* (7).

1-1-9. Locate the board configuration jumpers (8).

If the REV is a *letter*, then the boot configuration uses jumper shunts.

These jumpers (`REV` <*letter*>) or switches (`REV` <*number*>) configure the Zynq 7000 SoC device's boot mode, which should be set to **the desired boot mode**.

1-1-10. Set the shunt jumpers as shown below to ensure that the board is configured to boot from **the desired boot mode**.

AMD
together we advance_

Note: Settings are also shown below to illustrate different boot mode settings. Make sure you select **the desired boot mode**.

ZC702 REV <letter> Jumpers	J26	J25	J22	J20	J21
ZC702 REV 1.<x> Switch SW16	SW16.5	SW16.4	SW16.3	SW16.2	SW16.1
JTAG mode[1]	0	0	0	0	0
Independent JTAG mode	1	0	0	0	0
Quad SPI mode	0	0	0	1	0
SD mode	0	0	1	1	0
MIO configuration pin	MIO2	MIO3	MIO4	MIO5	MIO6

These two jumpers/switches set the boot mode

Figure 231: ZC702 - Boot Mode Settings Table

Jumpers J27, J28, J21, J20, and J26 (shown in **yellow**) are in the same position for all of the boot configurations.

Jumpers J27 and J28 are selected to enable SW1 and SW2 buttons to respectively initiate PS power on and system reset operations. Jumpers J21, J20, and J26 determine the QSPI memory configuration.

Jumpers J22 and J25 (shown in **red**) select the configuration mode: **JTAG** (cable), **QSPI** (on-board SPI memory), or **SD card** (using the PS SD0 controller).
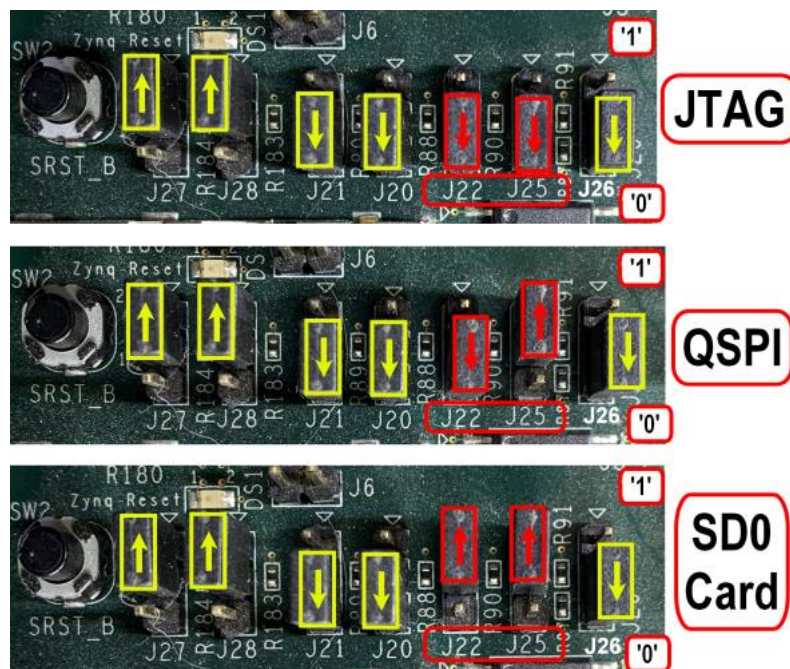
Figure 232: ZC702 Boot Mode Jumper Settings

If the ZC702 board REV is a **number**, then boot configuration uses a five-position dip switch designated SW16.

Jumpers J27 and J28 (shown in **yellow**) are in the same position for all of the boot configurations.

Jumpers J21, J20, J22, J25, and J26 are not populated as they are implemented via SW16.

1-1-11. Set SW16 as shown below to ensure that the board is configured to boot from **the desired boot mode**.

SW16.1, SW16.2, and SW16.5 (shown in **green**) replace jumpers J21, J20, and J26 and are set in the same position, '0', for all boot configuration modes.

SW16.3 and SW16.4 replace jumpers J22 and J25 and set the boot source mode.
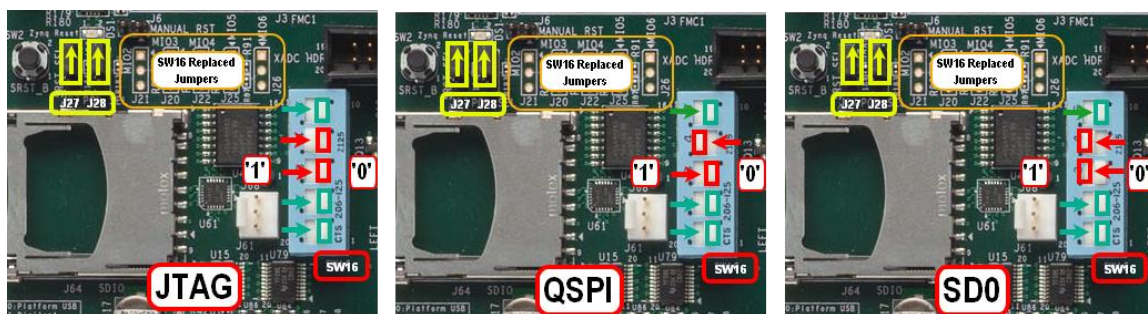
Figure 233: ZC702 Boot Mode SW16 Switch Settings

1-1-12. Slide the power switch to the "ON" position to power on the board.

AMD
together we advance_

Setting Up the ZedBoard

1-1. Bring up the ZedBoard.

The figure below of the ZedBoard enumerates some of its more popular features. Take a moment to familiarize yourself with the PCB location of these features. Before the board can be used, there are cables and switch and jumper settings that need to be in place and set (or just verified) to ensure proper board operation.

The steps that follow will reference items at the *numbered* PCB locations. Additional information can be found in the *ZedBoard HW User Guide* available from the Digilent website (the board manufacturer).
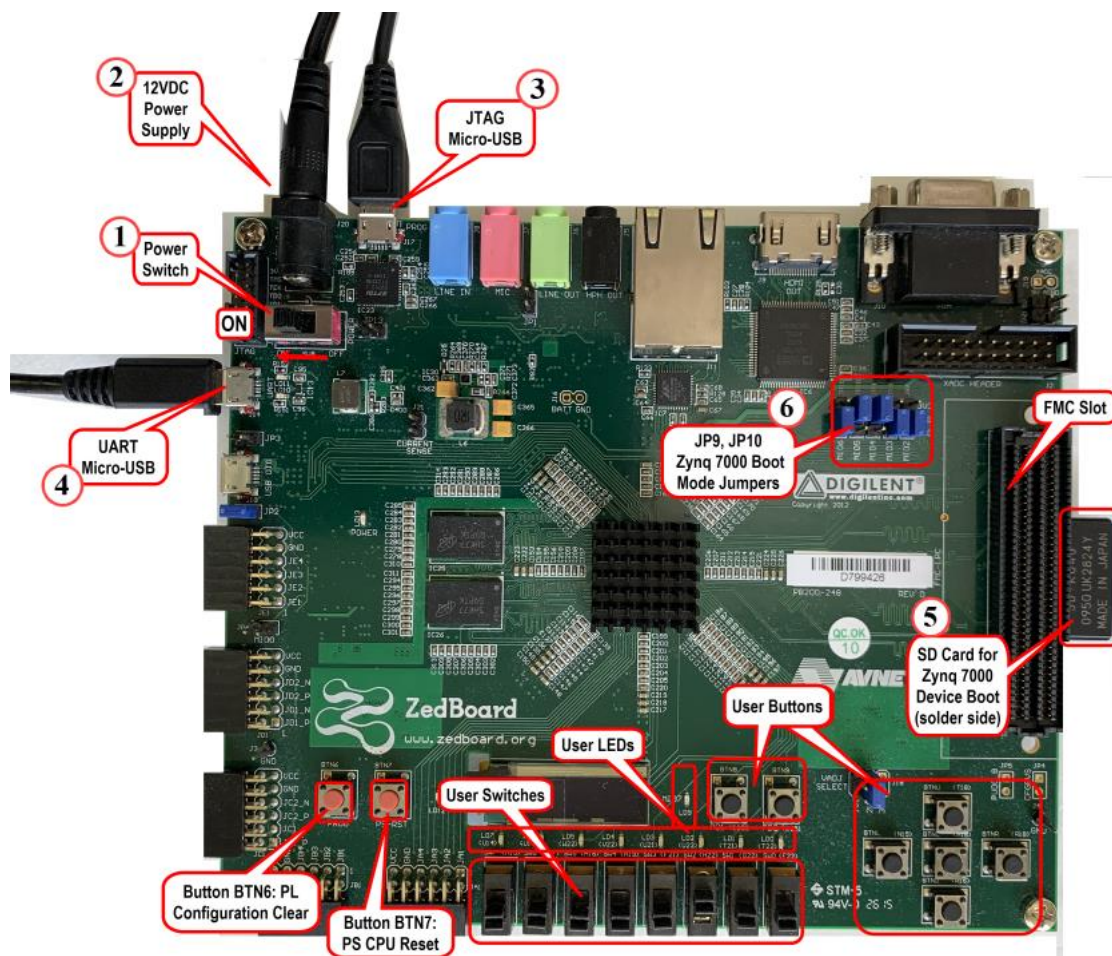
Figure 234: ZedBoard Overview

1-1-1. Ensure that the board is powered off by verifying that the power slide switch is in the position furthest from the board edge (1).

The figure above shows the switch in the "ON" position.

1-1-2. Ensure that the power connector is plugged in (2).

1-1-3. Connect the micro-USB JTAG cable between the board and the host (3).

This connection implements JTAG communication to boot, download software object code, and configure the PL of the Zynq 7000 SoC.

1-1-4. Connect the micro-USB UART cable between the board and the host (4).

This connection uses a USB-to-UART bridge chip to implement host COMx port communication to PS UART1 in the Zynq 7000 SoC.

1-1-5. If the boot mode is an SD card, insert the SD card containing the Zynq 7000 SoC device boot image (e.g., BOOT.bin) into the SD card slot located on the bottom (solder) side of the board (5).

1-1-6. Locate the board configuration jumpers (6).

These jumpers configure the Zynq 7000 SoC device's boot mode, which should be set to **the desired boot mode**.

1-1-7. Set the shunt jumpers as shown below to ensure that the board is configured to boot from **the desired boot mode**.

Note: Settings are also shown below to illustrate different boot mode settings. Make sure you select **the desired boot mode**.

Jumpers JP11, JP8, and JP7 (shown in **yellow**) are in the same position for all of the boot configurations. Jumper JP11 enables the PS internal PLL. Jumpers JP8 and JP7 determine the QSPI memory configuration.

Jumpers JP10 and JP9 (shown in **red**) select the configuration mode: **JTAG** (cable), **QSPI** (on-board SPI memory), or **SD card** (using the PS SD0 controller).

These two jumpers set the boot mode	ZedBoard Configuration Mode Jumpers				
	MIO[6]	MIO[5]	MIO[4]	MIO3[]	MIO[2]
AMD TRM	Boot_Mode[4]	Boot_Mode[0]	Boot_Mode[2]	Boot_Mode[1]	Boot_Mode[3]
Jumper	JP11	JP10	JP9	JP8	JP7
JTAG	0	0	0	0	0
Quad SPI	0	1	0	0	0
SD Card	0	1	1	0	0

Figure 235: ZedBoard Boot Mode Settings Table
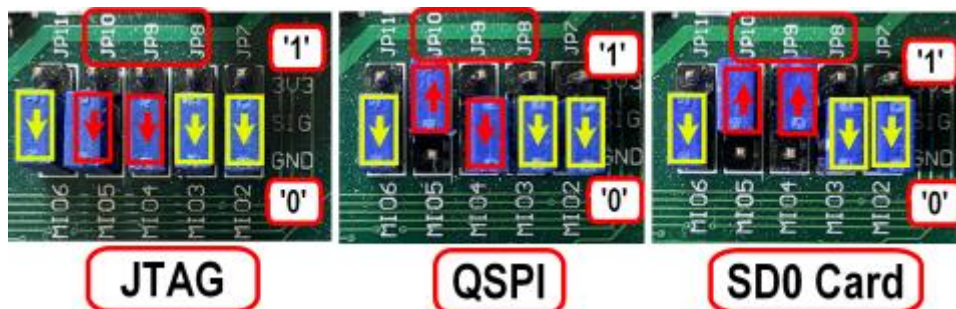
Figure 236: ZedBoard Boot Mode Jumper Settings

1-1-8. Slide the power switch to the "ON" position to power on the board.

AMD
together we advance_

Determining the COM Port Assigned by the USB Driver

A serial UART terminal emulator program provides a simple and straightforward way to collect and transmit serial information using the RS-232 protocols. Most modern PCs lack the traditional DB-9 connectors for RS-232 communication and instead use USB ports.

From the PC's perspective there is still a COMx port being used; however, the actual COM port number is not known until USB enumeration. The PC is capable of supporting multiple COM ports, so it is important to know which connection is the one to use when communicating with the development board.

When using a PC COM port with communications software (terminal emulator) on the PC such as the Vitis Serial Terminal tab, Tera Term, or other software, it is necessary to identify the COM port number associated with serial connection to the evaluation board. This is done by identifying the USB COM port driver and which COM port is being assigned.

The MPSoC boards support three USB ports. Two are connected to the UARTs in the PS and one is available to the PL, assuming that there is a UART implemented in the PL.

1-1. **Determine which COM port is connected to the USB serial port from the board.**

1-1-1. Make sure that the development board is powered on and the serial UART device USB cable is in place.

This ensures that the USB-to-serial bridge will be enumerated by the PC host.

The Device Manager in Windows lists all of the hardware items in the system. While there are a number of routes to opening the device manager, the easiest is as follows.

1-1-2. Click **Start** (1).

1-1-3. Enter the following into the search bar (2):

```
device manager
```
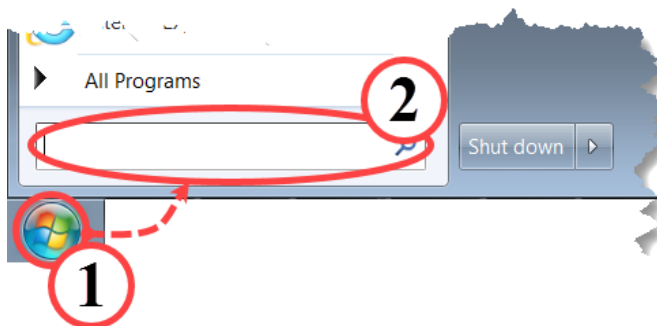

Figure 237: Accessing the Windows Application Search

Note: You may be asked to confirm opening the Device Manager. If so, click **Yes**.

1-1-4. Expand **Ports (COM & LPT)** so that you can see all of the enumerated ports.

Various boards use different USB bridge chips. The name of the driver varies correspondingly.

o **ZCU102/4 users:** Locate **Silicon Labs Quad CP210x USB to UART Bridge: Interface 0 (COM#)**.

▪ Note that there will be a group of (at least) four COM ports listed. This is because this board has a quad USB to UART bridge. Interface 0 and 1 are attached to stdout 0 and 1, respectively. Also note the COM port number assigned to *Silicon Labs Quad CP210x USB to UART Bridge: Interface 0 (COM#)"*as this is the COM port number you will need when establishing communications with the board, specifically the output from the PMU.

Note: The **#** indicates the port number for this serial connection.

Example:
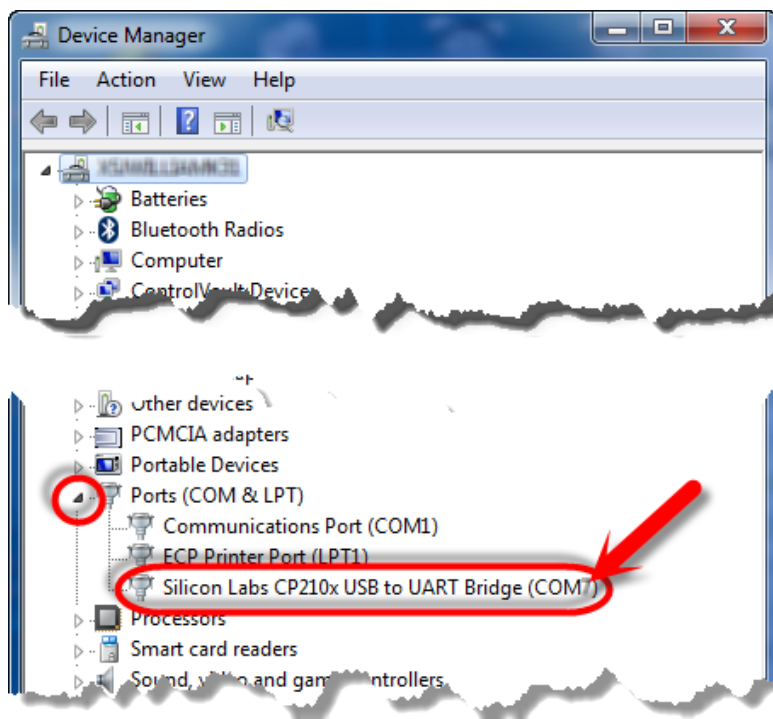
Figure 238: Locating the Silicon Labs Com Port Driver

1-1-5. Note the COM# corresponding to the USB bridge chip.

This is the port number you will use when connecting a terminal to the board.

1-1-6. Close the Device Manager by clicking the red **'X'** in the upper-right corner of the panel.

AMD
together we advance_

Opening a Terminal – Linux Ubuntu

Sometimes it is easier to enter operating system commands via a Linux terminal rather than a GUI. A terminal can completely configure and control a Linux system without the need for a graphical desktop.

Even when there is a graphical desktop available for use, it may sometimes be necessary or more convenient to use the terminal window to launch programs and configure settings.

1-1. **Open a Linux terminal window.**

1-1-1. Click the Linux terminal window icon in the taskbar.
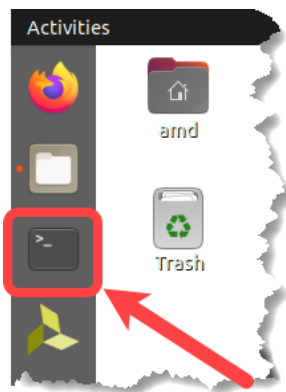
Figure 239: Opening the Linux Terminal Window from the Taskbar

Alternatively, you can press <**Ctrl** + **Alt** + **T**>.

The terminal window will open, leaving you in your home directory.

Shutting Down Ubuntu Linux

1-1. [Windows users]: Shut down Ubuntu Linux.

1-1-1. Click the **System Settings** icon in the upper-right corner of the Ubuntu desktop (1).

1-1-2. Select **Shut Down** (2).

The Shut Down dialog box opens.

1-1-3. Click **Shut Down** (3).
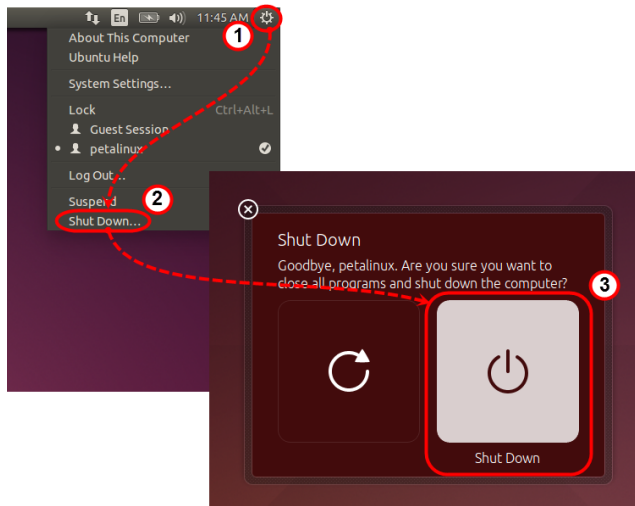
Figure 240: Shutting Down

Running a Virtual Machine Using VirtualBox

For CloudShare users, you can skip the launching VirtualBox instructions below and proceed with the launching the Vivado Design Suite instructions.

1-1. Launch the VirtualBox Manager application.

Running Linux under Windows is easily achieved using a virtual machine that forms an isolated container for another OS. The Oracle VirtualBox application was selected as the framework for hosting virtual machines as it works well and is available without the hassles of licensing. The Customer Training team provides a VM that is properly configured to run the labs and demos.

1-1-1. Select **Start Menu** > **Oracle VM VirtualBox** > **Oracle VM VirtualBox** to launch the VirtualBox Manager program.

Alternatively, if a desktop or taskbar icon is available, you can simply click it.

When the VirtualBox Manager opens, the left-hand pane will show all of the available virtual machines.

AMD
together we advance_

1-1-2. Double-click the virtual machine (1).

Alternatively, you can single-click the virtual machine of choice, then click the green right arrow icon to launch the selected virtual machine (2).
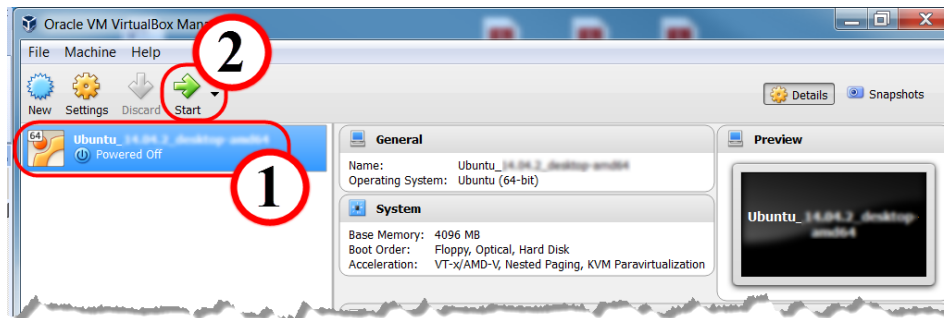
Figure 241: VirtualBox Manager Window

The virtual machine will now launch in a new window. It takes a few moments to boot. If any warning messages appears, such as anything regarding the mouse or keyboard, click the "x" on the line with the error and continue.

1-1-3. Click the **Maximize** (▣) icon in the window title bar to expand the screen to its full size, if not already maximized.

Cleaning Up the VirtualBox File System

Some systems (particularly VMs) may be memory constrained. Removing the workspace frees a portion of the disk space, allowing other labs to be performed.

You can delete the directory containing the lab you just ran by using the graphical interface or the command-line interface. You can choose either mechanism. Both processes will recursively delete all the files in the `$TRAINING_PATH/<the topic cluster name>` directory.

1-1. **[Optional] [Only for local VMs—not for CloudShare] Clean up the file system.**

Using the GUI:

1-1-1. Navigate to `$TRAINING_PATH/<the topic cluster name>`.

1-1-2. Select **<the topic cluster name>**.

1-1-3. Press <**Delete**>.

-- OR --

[Linux users]: Using the command line:

1-1-4. Press <**Ctrl** + **Alt** + **T**> to open a terminal window.

1-1-5. Enter the following command to delete the contents of the workspace:

`[host]$ `**`rm -rf $TRAINING_PATH/<the topic cluster name>`**

Setting the Static Host IP Address Using Windows 10

These instructions illustrate how to change to or set up a *static* IP address on the host computer. Follow the reverse of these instructions to change back to a DHCP-received address.

For labs that require an Ethernet connection between the host computer, the hardwired Ethernet settings on your host machine must be set up properly to work with the hardware evaluation board.

Typically, the PC's Ethernet port is set to automatically request an IP address from the network DHCP client server. Because the hardwire Ethernet connection will attach directly to the hardware evaluation board, there will not be a DHCP server to supply the needed IP address for the host computer.

It is therefore necessary to reconfigure the TCP/IP client (host computer) with its own *static* IP address. After you are finished with this lab, you can revert this step and set TCP/IP properties to obtain an IP address automatically.

Most of our software applications use an IP address of 192.168.1.11. The host computer is required to have an IP address of 192.168.1.**num**, where **num** is any number between 2 and 255 except 11 (which is reserved for the hardware).

These instructions describe how to perform this task on a Windows 10 machine.

1-1. Set the static host IP address.

1-1-1. If wireless is on, it is suggested that you **turn if off** with the switch on your computer or disable in settings.

How to perform this will vary with different PC wireless hardware.

While this step should not be necessary, there has been reports that the wired Ethernet port does not work with static address (192.168.1.**num**) using the same base subnet address of the wireless adapter.

1-1-2. Right-click the networking icon on the bottom-right corner of the taskbar (1).

1-1-3. Select **Open Network and Sharing Center** (2).
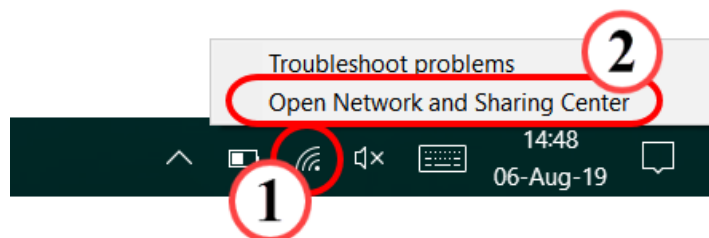
Figure 242: Selecting Open Network and Sharing Center

AMD
together we advance_

1-1-4. From the Network and Sharing Center left pane, select **Change Adapter Settings**.
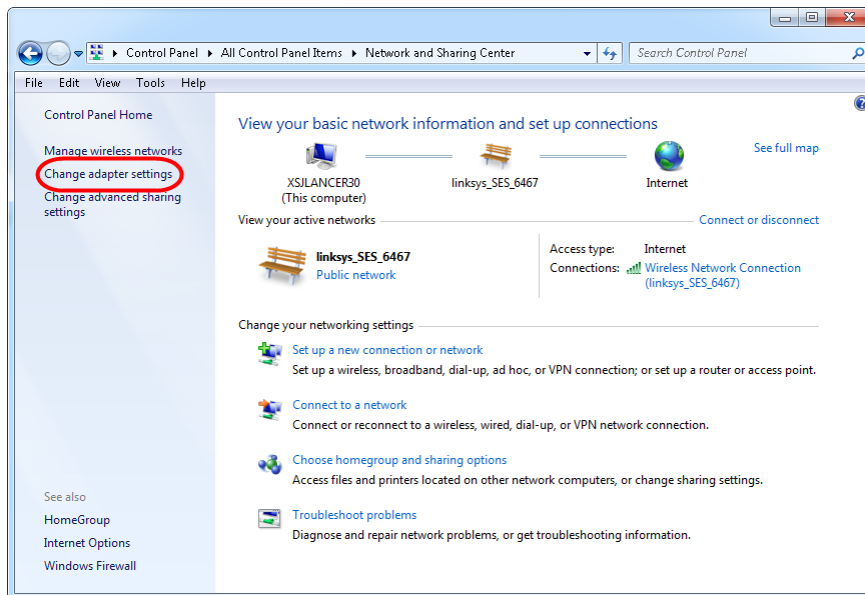
Figure 243: Control Panel Networt and Sharing Center

1-1-5. Select the Ethernet adapter to be used.

1-1-6. Right-click and select **Properties**.
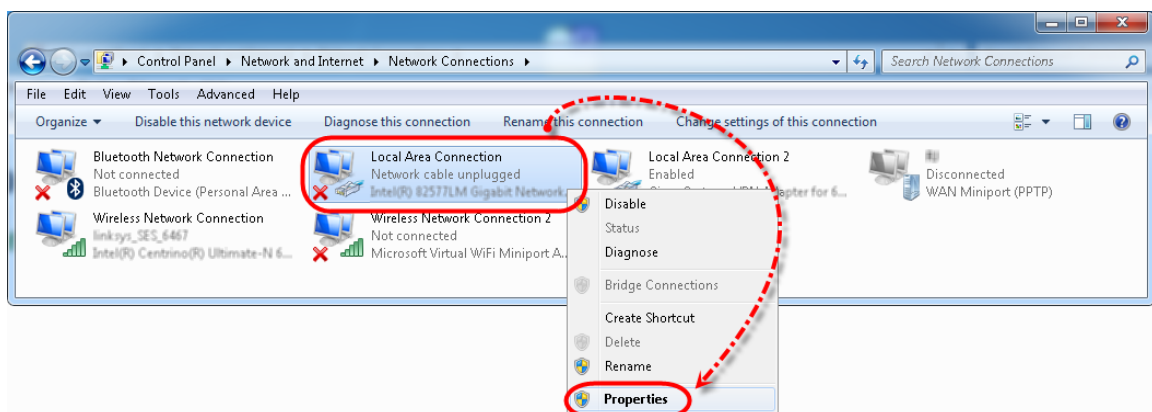
Figure 244: Network Connections

1-1-7. Click **Yes** to make changes (if necessary).

1-1-8. Deselect **Internet Protocol Version 6 (TCP/IPv6)** (1).

1-1-9. Select **Internet Protocol Version 4 (TCP/IPv4)** (2).

1-1-10. Click **Properties** (2).
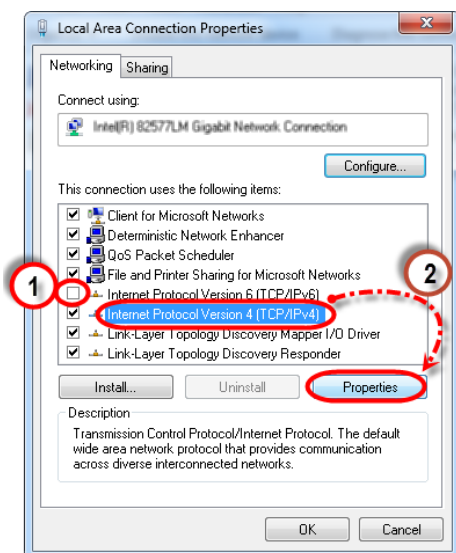
Figure 245: Local Area Connection Properties

1-1-11. Select **Use the following IP address** (1).

1-1-12. Enter **192.168.1.10** in the IP address field (2).

This value is fairly arbitrary, but it *cannot* be the same as the IP address designated for the board.

The Subnet mask field should fill in with **255.255.255.0** automatically after leaving the IP address field (3).
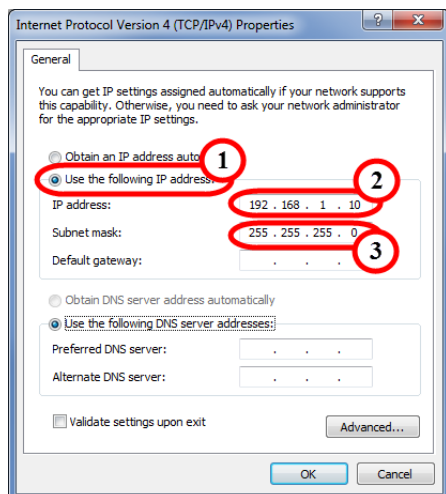
Figure 246: Set TCP-IPv4 Properties for Static Address

1-1-13. Click **OK**.

1-1-14. Click **Close** to close the Control Panel.

AMD
together we advance_

1-2. To return to a DHCP-obtained address, follow the same steps to revert the settings.

1-2-1. Select the **Internet Protocol Version 6 (TCP/IPv6)** option.

1-2-2. Select **Internet Protocol Version 4 (TCP/IPv4)**

1-2-3. Click **Properties**.

1-2-4. Select **Obtain an IP address automatically**.
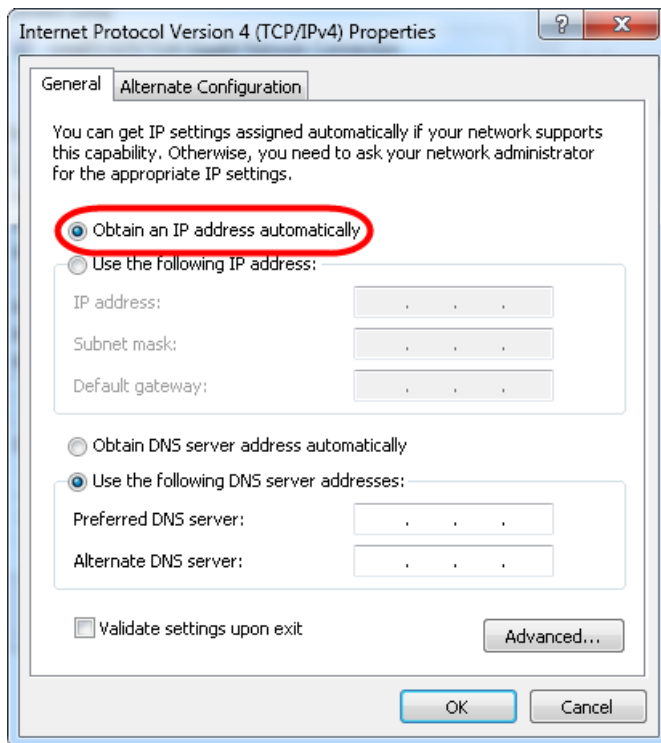
Figure 247: Set TCP-IPv4 Properties for DHCP Obtained Address

1-2-5. Click **OK**.

1-2-6. Click **Close**.

Setting the Static Host IP Address Using Linux

These instructions illustrate how to change to or set up a *static* IP address on the host computer. Follow the reverse of these instructions to change back to a DHCP-received address.

For labs that require an Ethernet connection between the laptop host computer, the hardwired Ethernet settings on your host machine must be set up properly to work with the hardware evaluation board. Typically, the PC's Ethernet port is set to automatically request an IP address from the network DHCP client server. Because the hardwire Ethernet connection will attach directly to the hardware evaluation board, there will not be a DHCP server to supply the needed IP address for the host computer. It is therefore necessary to reconfigure the TCP/IP client (laptop) with its own *static* IP address. After you are finished with this lab, you can revert this step and set TCP/IP properties to obtain an IP address automatically.

Most AMD software applications use an IP address of 192.168.1.11. The host computer is required to have an IP address of 192.168.1.**num**, where **num** is any number between 2 and 255 except 11 (which is reserved for the AMD hardware).

These instructions describe how to perform this task on a Linux machine.

1-1. Set the static host IP address.

1-1-1. Press <**Ctrl** + **Alt** + **T**> to open a new terminal.

1-1-2. Enter the following command to set the static IP:

```
[host] $ ifconfig eth0 192.168.1.10
```

This value is fairly arbitrary, but it *cannot* be the same as the IP address designated for the board.

1-1-3. Exit the terminal:

```
[host] $ exit
```

1-2. To return to a DHCP-obtained address, follow the same steps to revert the settings.

1-2-1. Reboot the Linux machine or VM to revert the static IP settings that you set in the previous step.

AMD
together we advance_

Verifying Board Jumper/Switch Settings Configured to Boot from the SD Card

The board evaluation hardware platform has jumpers (or switches) to select the boot mode for the processor.

1-**1.** **Verify that the board hardware platform is properly connected and the jumpers (or switches) are configured to boot from the SD card.**

1-1-1. Confirm that the power to the evaluation board is OFF.

1-1-2. Ensure that the connections between the host PC and the hardware evaluation board are correct (ask for assistance from the instructor if necessary).

The connections include power, serial bridge USB, Ethernet, and USB Platform download cables.

1-1-3. Make sure that the SD card with a Linux image is inserted into the board card slot.

1-1-4. Verify the jumper settings on the board are set up to boot from the SD card.

Refer to the "Jumper/Switch Settings your board - Boot from SD Card " topic under the Hardware Requirements - <Name of Board> Hardware Setup section in the *Lab Reference Guide* or ask your instructor for assistance.

Powering Up the Hardware Platform

1-**1.** **Apply power to (turn on) the board hardware platform.**

1-1-1. Make sure that AC power is connected to the power brick.

1-1-2. Ensure that the power brick is connected to the board.

1-1-3. Confirm the serial port USB connection between the board and the PC.

1-1-4. Confirm the USB-JTAG connection between the board and the PC.

1-1-5. Slide the power switch to the on position.

Some LEDs on the board will illuminate when the board is powered.

Preparing an SD Card

1-1. Prepare an SD card.

1-1-1. Insert an SD card into the PC's SD card slot.

1-1-2. **[Linux users]:** Make sure that you copy the SD card files from Linux to the *C:\training* directory by entering the following command:

```
[host] $ cp –rf <path_to_sd_card_files> /media/sf_training
```

1-1-3. Browse to the SD card drive using Windows Explorer.

Optional: You may want to erase or reformat the SD card at this time, which may prevent any unwanted interaction with other files that may be on the card.

1-1-4. Open a second Windows Explorer window to browse to *the location of the files you want to copy to the SD card*, which will be copied to the SD card.

Be aware that many SD card images are delivered as compressed files. You will need to ensure that these files are properly decompressed unless otherwise indicated.

1-1-5. Drag-and-drop all the files from the source directory to the SD card directory.

This will automatically copy the files onto the SD card.

Note: The files should go into the root of the SD card unless there are specific instructions to the contrary.

1-1-6. Close both Windows Explorer windows.

1-1-7. Remove the SD card from the PC card slot.

1-1-8. Turn off power to the hardware platform.

Note: The boot selection switches or jumpers must be properly set to boot from the SD card. See the appropriate section in the *Lab Setup Guide*.

1-1-9. Insert the SD card into its slot on the hardware platform.

AMD
together we advance_

Booting Linux

You must have an SD card with a properly constructed Linux boot image. This can be open-source Linux, PetaLinux, or a third-party image.

1-1.   Set the jumpers/switches on your board to boot from the SD card.

If you do not recall how to perform this task, refer to the "Setting the Jumpers on the <Name of Board> Board" section under Board, OS, COM, and IP Address Tasks in the *Lab Reference Guide*.

1-2.   Insert an SD card with a properly constructed Linux boot image.

1-3.   Apply power to the board.

1-3-1. Open a terminal window if you want to observe the Linux boot process.

This process typically takes several minutes to complete. The reason for applying power prior to setting up a terminal window is that the PC must first "see" the UART (via USB). This can only happen when the board is powered up. If you want to see all of the Linux boot-up messages, you will need to recycle power on the board.

Launching and Configuring the Terminal Emulator

Maintaining consistency in the instructions is an important aspect of the lab experience. The labs will use general and vague commands such as "Open the terminal emulator". Depending on the system that you are using, you must know which terminal emulator to configure and launch. Here are the instructions for opening a terminal emulator for both the Linux and Windows environments.

Generally, the software labs will use the serial terminal built into the Vitis platform; however, there are some labs that will require you to communicate with the board without using the Vitis environment. These situations require the use of a third-party serial port emulator.

We have tested Tera Term for the Windows environment and GTKTerm for Linux. The GTKTerm tool is pre-installed with the Customer Training VM. Both terminal emulators listed here run independently of the tools. If you have another terminal emulator that you prefer, you can certainly use it; however, you are responsible for figuring out how to configure it.

Linux

GTKTerm is a simple GTK+ terminal used to communicate with the serial port. Other terminal emulators may be used; however, the installation and configuration instruction provided here are for GTKTerm.

GTKTerm is already installed in the VM provided by the Customer Training team; however native Linux users may need to install GTKTerm.

1-1.   [Native Linux users] Acquire and install the GTKTerm software from the command line.

1-1-1. Press <**Ctrl** + **Alt** + **T**> to open a new terminal.

1-1-2. Download and install GTKTerm:

```
[host] $ sudo apt install gtkterm
```

1-1-3. When prompted for the password, enter the super user password.

The tools will install in about a minute or less.

GTK Term is an interface that only supports serial port/UART communications. The more general Terminal tab supports other formats, such as SSH and Telnet.

Since serial port communications occur over the USB, which is an enumerated interface, the board must be powered on with the cable connected for the port to be recognized.

If you are using the VM, ensure that the proper USB port is allowed to cross the host/VM threshold by selecting **Devices** > **USB** > **Xilinx <board selection> [0800]** in the VirtualBox Manager for Zynq UltraScale+ MPSoC/Versal adaptive SoC boards or selecting **Devices** > **USB** > **Digilent Adept USB Device [0700]** *AND* **Silicon Labs CP2103 USB to UART Bridge Controller [0100]** for Zynq 7000 SoC boards.

1-2.   Launch GTKTerm and set the port configuration.

1-2-1. Click the **GTKTerm** icon ( ) from the quick launch toolbar.

Alternatively, GTKTerm can be launched from a Linux terminal window (<**Ctrl** + **Alt** + **T**>) and entering:

```
[host] $ sudo gtkterm
```

Note: While the application will run as a regular user, you must be a super user to access the ports.

When the GTKTerm window opens, perform the following.

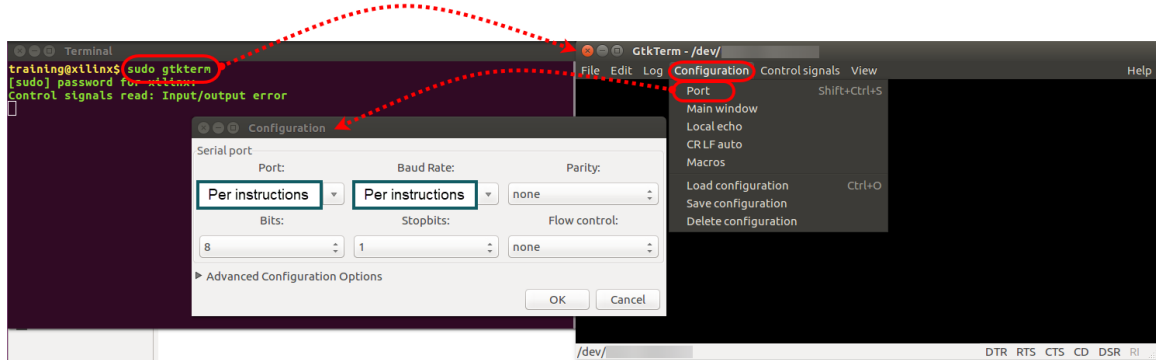**1-2-2.** Select **Configuration** > **Port** to open the Configuration dialog box.

1-2-3. Identify the port associated with your board and set the port as **/dev/ttyUSBx** (where x could be 0, 1, 2, 3, etc.)

AMD
together we advance_

1-2-4. Set the baud rate to **115200**.

1-2-5. Leave the rest of the settings at their default.

Note: You can open multiple instances of `/dev/USBx` if you are unable to determine which port your UART is connected to.

Figure 248: Opening GtkTerm and Selecting the Port Configuration

[Optional]: You can save these settings so that you do not have to reconfigure GKTTerm each time you open it by selecting **Configuration** > **Save configuration**.

If you save the configuration as "default", this configuration will open when GTKTerm starts. Otherwise, you can save the configuration with another name; however, you will then need to load the configuration each time you start GTKTerm.

1-2-6. Click **OK** to save the settings and leave the terminal open.

1-3. **Enable auto CR/LF mode.**

GTKTerm is a terminal emulator that supports automatic translation of CR (carriage return) characters to LF (line feed) characters and vice versa.

1-3-1. Select **Configuration** > **CR LF auto** to enable auto CR/LF mode.
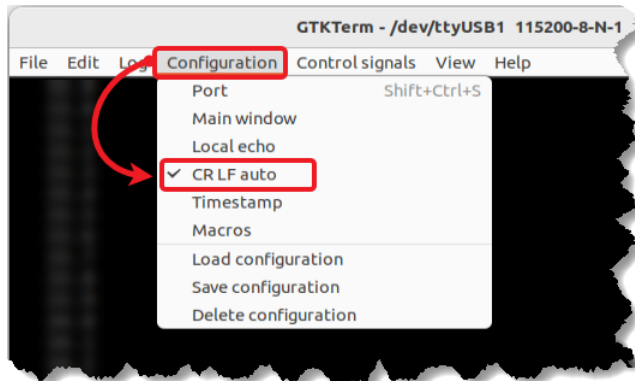
Figure 249: Enabling CR/LF auto in GTKTerm

Note: This step is optional, but it will ensure that the serial messages appear aligned on the terminal.

Windows

Tera Term is a popular public domain terminal emulation program. It is capable of operating as a serial port terminal or as a telnet client.

1-4. Download and install Tera Term.

1-4-1. Acquire and install the Tera Term software from any legitimate site. Recommended sites:

- o ttssh2.osdn.jp/index.html.en (Tera Term home page with documentation)
- o osdn.net/projects/ttssh2
- o en.sourceforge.jp/projects/ttssh2
- o logmett.com

1-4-2. Install per instructions.

There are two types of downloads: a traditional zip install, and a self-installing version, which is recommended.

[Optional]

Certain drivers like installing their com port numbers using high numbered serial ports. Tera Term does not accept these port numbers by default, so you will need to override the Tera Term settings:

1-4-3. Open **TERATERM.INI** (found in the install path for Tera Term) with an ASCII text editor.

1-4-4. Set to MaxComPort=256.

1-4-5. Save and close the INI file.

1-4-6. Use the **Setup** > **Save Setup** option to save the setup.

1-5. Launch the Tera Term terminal program.

1-5-1. Double-click the **Tera Term** icon on the Windows desktop to launch Tera Term.

Alternatively, you can select **Start** > **All Programs** > **Tera Term** > **Tera Term**.
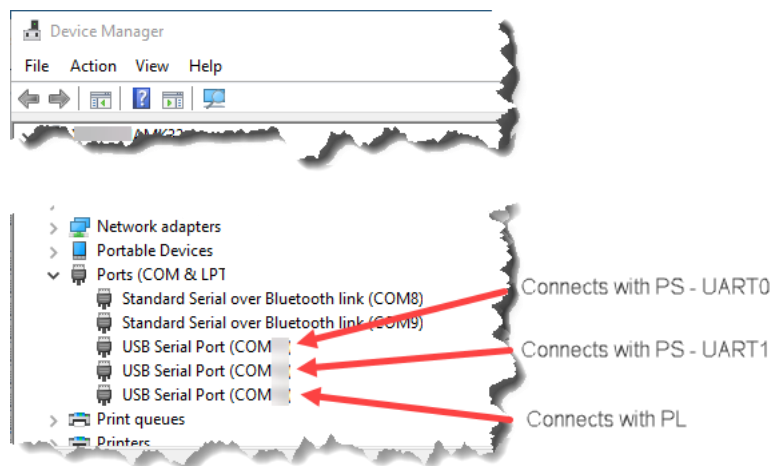
1-5-2. Select **Serial** as the connection (1).

1-5-3. Click the **Port** drop-down list to view the available COM ports (2).

Note: If your port is not listed, exit Tera Term, power cycle your board and restart this step.

1-5-4. Select the COM **#** (3).

AMD
together we advance_

Hint: MPSoC and RFSoC devices display three COM ports. Their specific numbers may vary based on the USB enumeration process; however, the first two COM ports connect to the UARTs in the PS and the third connects to PL pins.

Figure 250: Locating and Identifying COM Ports from the Windows Device Manager
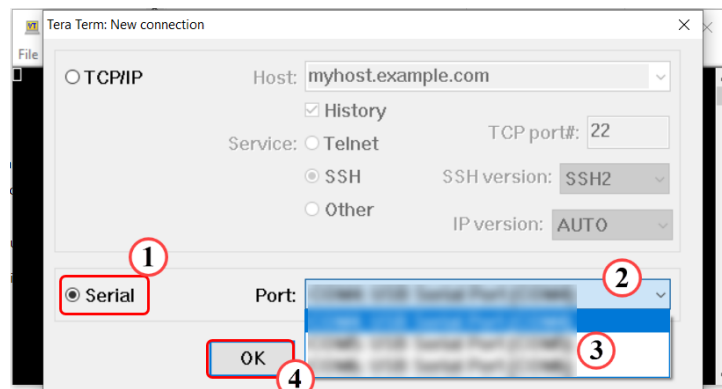
Figure 251: Selecting the COM Port

Note: The COM port setting is specific to the computer being used and may need to be different than shown. Use the COM port **#** that was discovered in the previous step.

1-5-5. Click **OK** (4).

The terminal console window opens.

1-5-6. Select **Setup** > **Serial Port**.

Figure 252: Opening the Tera Term Serial Port Setup Window

The Tera Term Serial Port Setup dialog box opens.

1-5-7. Confirm that the proper serial port has been selected (1).

1-5-8. Set the baud rate to **115200** (2).

Figure 253: Setting the Parameters for the Serial Port
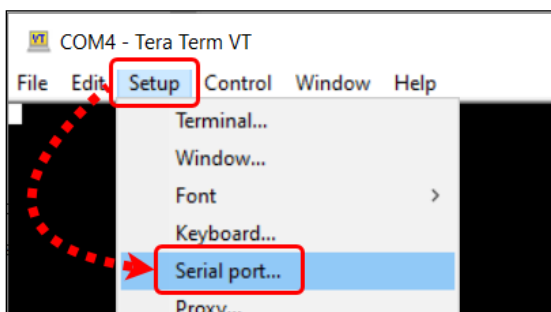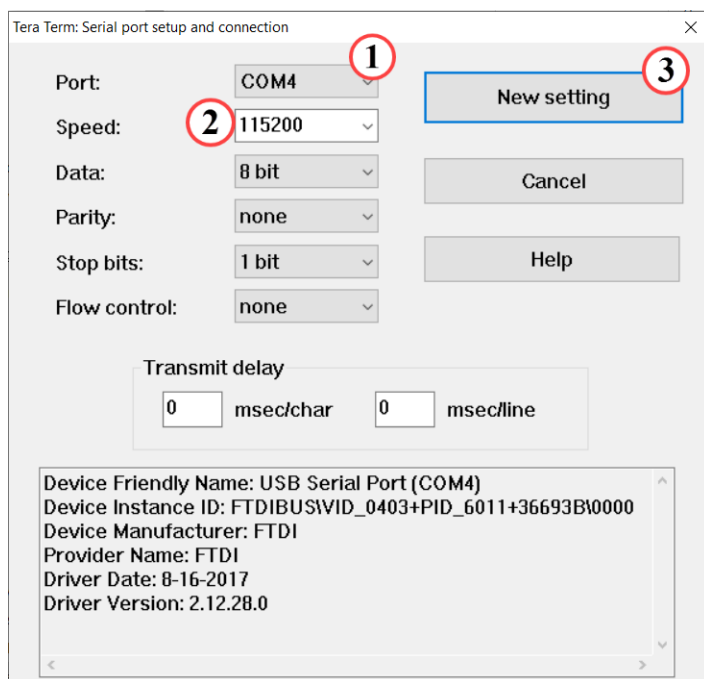
Note: The COM port setting is specific to the computer being used and may need to be different than shown. Use the COM port **#** that was discovered in the previous step.

1-5-9. Click **OK** (3).

Tera Term is now configured to receive and transmit serial information to/from the evaluation board.

AMD
together we advance_

Setting the Jumpers on the ZC702 Board

The jumpers connected to the Zynq device are read by the Stage 0 bootloader and are used to determine where the Zynq PS boots from.

1-1.   Set the jumpers on the ZC702 board based on your booting requirements.

- **Boot from SD card**
- **JTAG communications with PC (no boot)**
- **Boot from QSPI**

The three following topics cover these cases.

Jumper/Switch Settings ZC702 - Boot from SD Card

SD Card Boot:

| J27 | J28 | J21 | J20 | J22 | J25 | J26 |

Figure 254: SD Card Boot Settings (ZC702 Board)

Jumper/Switch Settings ZC702 - Boot from JTAG

JTAG Mode:

J27	J28	J21	J20	J22	J25	J26
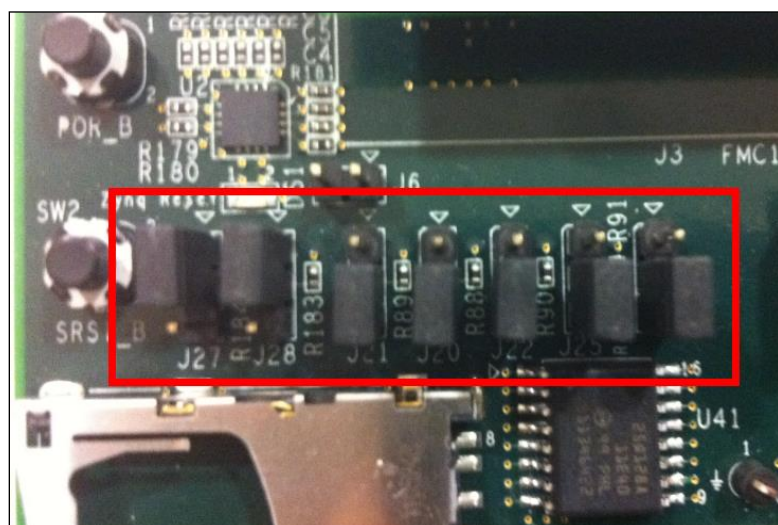
Figure 255: JTAG Boot Settings (ZC702 Board)

It is worth noting that if the SD card settings are used, but no SD card is present in the socket, then the boot mode will fall back to JTAG.

The ZC702 board has two JTAG cable options (most ATPs use option 1):

1. Digilent on-board platform cable USB interface using USB A-mini B cable: Set SW10 (i.e. JTAG Select dip switches) on Zynq board to "01"
2. Platform USB: Set SW10 (i.e. JTAG Select dip switches) on Zynq board to "10".

Jumper/Switch Settings ZC702 - Boot from QSPI

Flash Boot:

J27	J28	J21	J20	J22	J25	J26
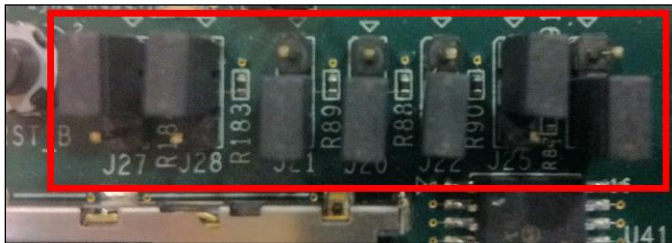
Figure 256: Flash Boot Settings (ZC702 Board)

Setting the Jumpers on the ZC706 Board

1-1. **Set the jumpers on the ZC706 board based on your booting requirements.**

- **Boot from SD card**
- **JTAG communications with PC (no boot)**
- **Boot from QSPI**

The three following topics cover these cases.

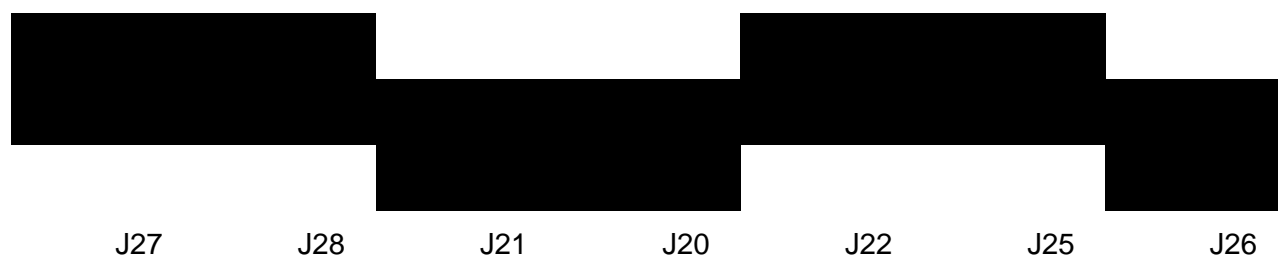# Jumper/Switch Settings ZC706 - Boot from SD Card

SD Card Boot:

SW11.1	SW11.2	SW11.3	SW11.4	SW11.5

Figure 257: SD Card Boot Settings (ZC706 Board)

AMD
together we advance_

Jumper/Switch Settings ZC706 - Boot from JTAG

JTAG Mode:

SW11.1	SW11.2	SW11.3	SW11.4	SW11.5

Figure 258: JTAG Boot Settings (ZC706 Board)

It is worth noting that if the SD card settings are used, but no SD card is present in the socket, then the boot mode will fall back to JTAG.

The ZC706 board has a Digilent interface option, and you can use a USB A-mini B cable.

To use this cable instead of Platform USB, make sure that the jumper settings are in JTAG mode as mentioned above.

- Set SW4 (i.e., the JTAG select dip switches) on the board to "01", which is to select the Digilent interface. To select Platform USB, SW4 should be "10".

Jumper/Switch Settings ZC706 - Boot from QSPI

Flash Boot:

SW11.1	SW11.2	SW11.3	SW11.4	SW11.5

Figure 259: Flash Boot Settings (ZC706 Board)

AMD
together we advance_

Setting the Jumpers on the ZCU102 Board

1-1. Set the jumpers on the ZCU102 board based on your booting requirements.

- **Boot from SD card**
- **JTAG communications with PC (no boot)**

   The following topics cover these cases.

Jumper/Switch Settings ZCU102 - Boot from SD Card

SD Card Mode:

Up		■		■
Down	■		■	
	SW6.1	SW6.2	SW6.3	SW6.4

Figure 260: SD Card Boot Settings (ZCU102 Board)

Jumper/Switch Settings ZCU102 - Boot from JTAG

JTAG Mode:

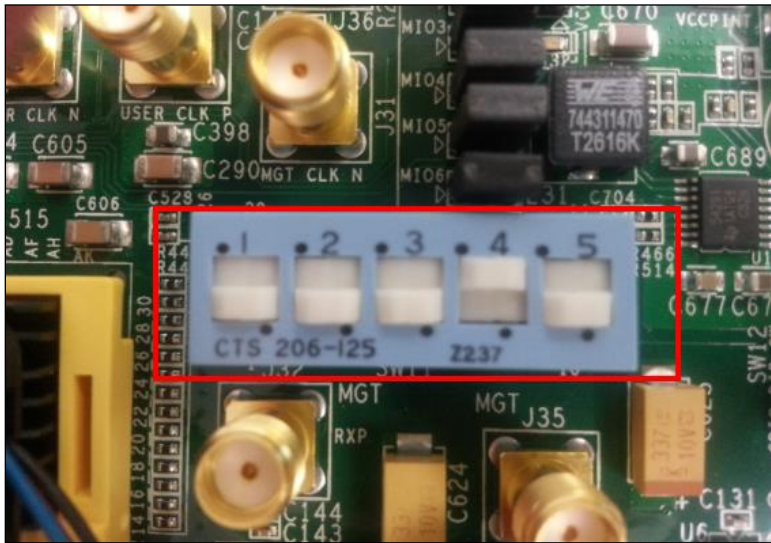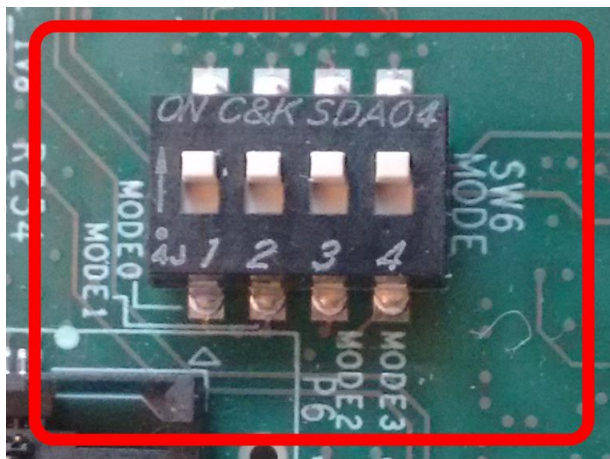|				
Up	███	███	███	███
Down				
	SW6.1	SW6.2	SW6.3	SW6.4

Figure 261: JTAG Boot Settings (ZCU102 Board)

Setting the Jumpers on the ZCU104 Board

1-1. Set the jumpers on the ZCU104 board based on your booting requirements.

- **Boot from SD card**
- **JTAG communications with PC (no boot)**

The following topics cover these cases.

AMD
together we advance_

Jumper/Switch Settings ZCU104 - Boot from SD Card

SD Card Mode:

Up	███████			
Down		██████████████████████		
	SW6.1	SW6.2	SW6.3	SW6.4

Figure 262: SD Card Boot Settings

Jumper/Switch Settings ZCU104 - Boot from JTAG

JTAG Mode:

Up	████████████████████████████████			
Down				
	SW6.1	SW6.2	SW6.3	SW6.4

Figure 263: JTAG Boot Settings

Jumper/Switch Settings ZCU104 - Boot from QSPI

Flash Boot:

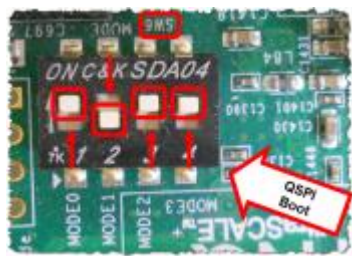|				
Up	⬛		⬛	⬛
Down		⬛		
	SW6.1	SW6.2	SW6.3	SW6.4

Figure 264: QSPI Boot Settings

Setting the Jumpers on the ZedBoard

1-1. Set the jumpers on the ZedBoard based on your booting requirements.
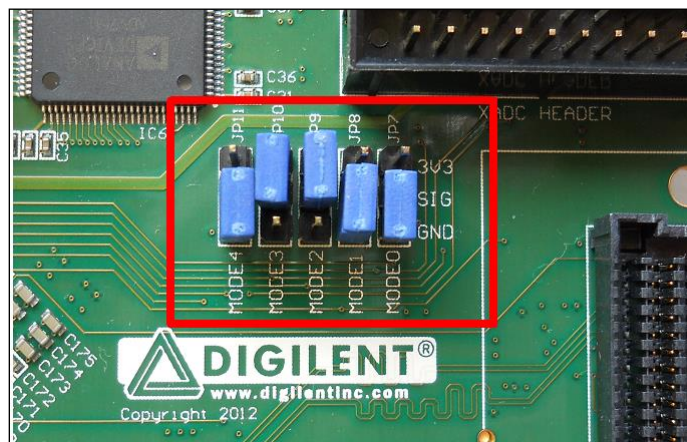
- **Boot from SD card**

AMD
together we advance_

Jumper Settings ZedBoard - Boot from SD Card

SD Card Boot:

JP8/JP11 - Mode 4	JP7/JP10 - Mode 3	JP6/JP9 - Mode 2	JP5/JP8 - Mode 1	JP4/JP7 - Mode 0	
	▪	▪			ON/3.3V
▪	▪	▪	▪	▪	
▪			▪	▪	Off/Gnd

Jumper - JP6 (shortened)

VADJ SELECT - 3V3

**Figure 265: SD Card Boot Settings (ZedBoard)**

Setting Up the Hardware - KC705

Set up and connect the hardware evaluation board or verify that this has properly been done before turning on the power.

1-1. Connect the board to your machine as shown below.

1-1-1. Connect a USB cable from a USB port on your computer to the **USB UART** connector on the evaluation board.

1-1-2. Similarly, connect the USB cable to the **Digilent USB JTAG** interface.

1-1-3. Ensure that the power cord is plugged in and turn on the evaluation board.

1-1-4. Make sure that the board settings are proper.

1-1-5. Ensure that all DIP switches (SW11) are in the off position.
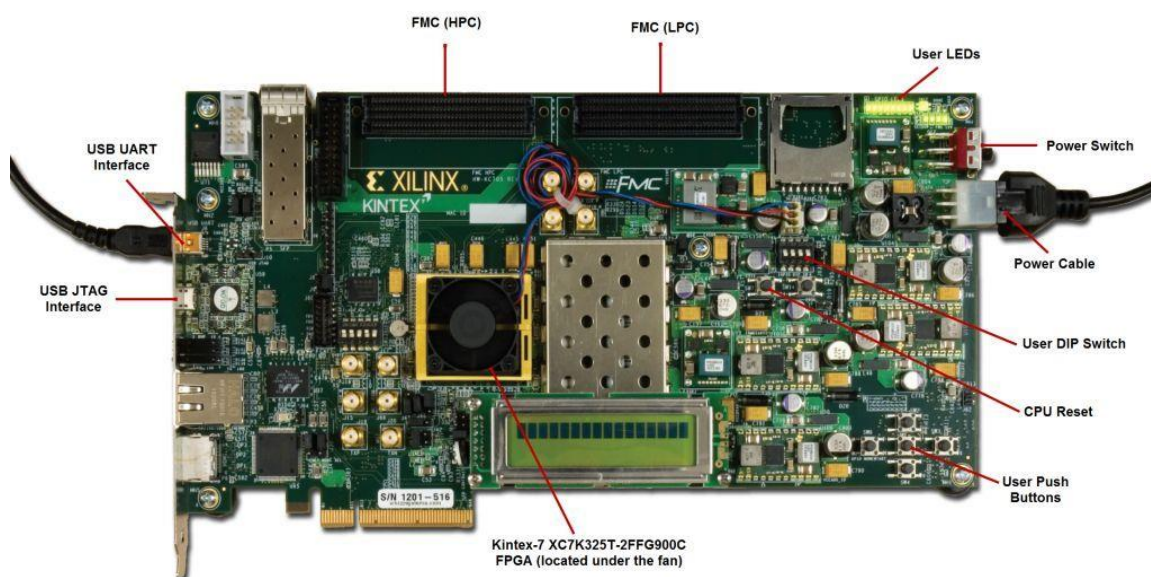
**Figure 266: KC705 Evaluation Board**

You may be prompted to install drivers when the board is first connected. Do not allow the driver installation to search the Web but allow it to search for the drivers on your computer.

AMD
together we advance_

Default Switch Settings

1-1. Default DIP switch for SW11 user GPIO settings.

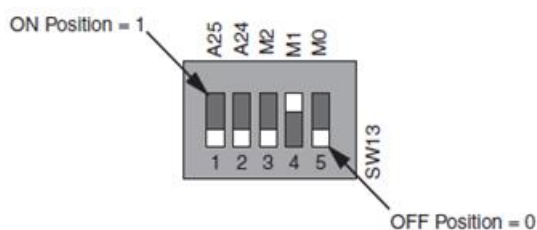**1-1-1.** Set all the switch positions to the default settings.
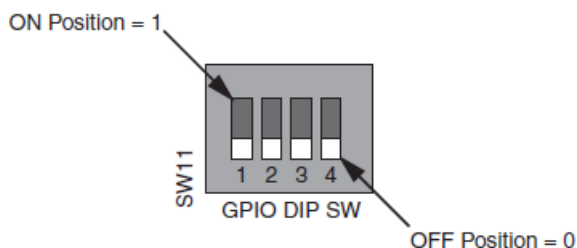
SW11 Default Switch Settings

Position	Function	Default
1	GPIO_DIP_SW3	Off
2	GPIO_DIP_SW2	Off
3	GPIO_DIP_SW1	Off
4	GPIO_DIP_SW0	Off

Figure 267: KC705: SW11 Default Switch Settings

1-2. Default DIP switch SW 13 mode and Flash address settings.

1-2-1. Set the switch positions to the default settings.

SW13 Default Switch Settings

Position	Function		Default
1	FLASH_A25	A25	Off
2	FLASH_A24	A24	Off
3	FPGA_M2	M0	Off
4	FPGA_M1	M1	On
5	FPGA_M0	M3	Off

Figure 268: KC705: SW13 Default Switch Settings

Setting Up the Hardware - KCU105

Set up and connect the hardware evaluation board or verify that this has been done properly before turning on the power.

1-1.   Connect the board to your machine as shown below.

1-1-1. Connect a USB cable from a USB port on your computer to the USB UART connector on the evaluation board.

1-1-2. Similarly, connect the USB cable to the USB JTAG connector on the evaluation board.

1-1-3. Ensure that the power cord is plugged in and turn on the evaluation board.
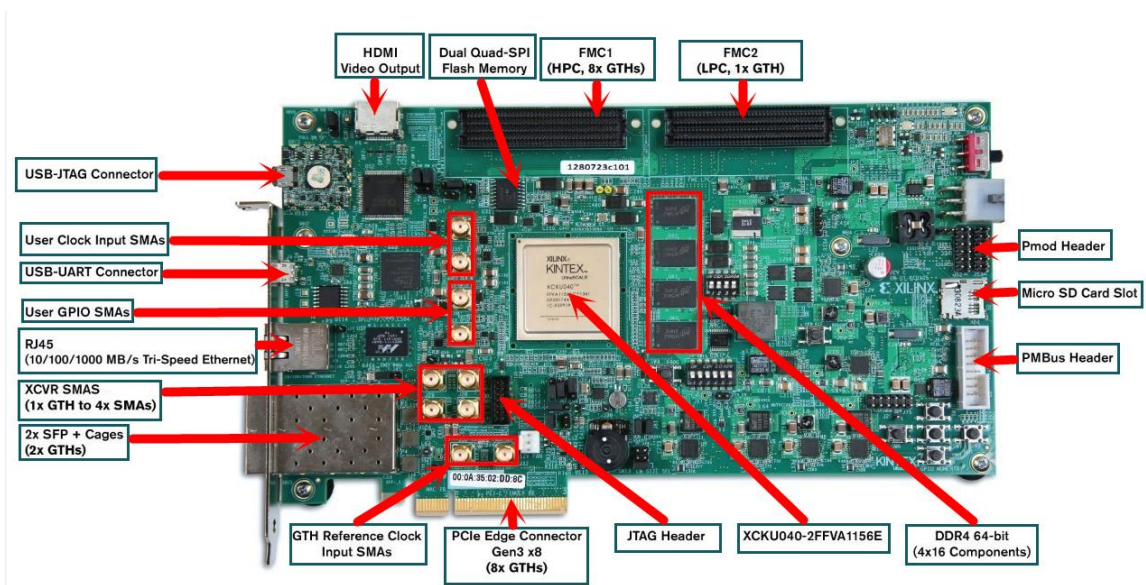
1-1-4. Make sure that the board settings are correct.

Figure 269: KCU105 Evaluation Board

AMD
together we advance_