Dynamic Function eXchange Project Flow

2022.2

Abstract

This lab introduces the Dynamic Function eXchange (DFX) project flow in the Vivado™ Design Suite.

This lab should take approximately 60 minutes.

CloudShare Users Only

You are provided with three attempts to access a lab, and the time allotted to complete each lab is twice the time expected to complete the lab. Once the timer starts, you cannot pause the timer. Each lab attempt will reset the previous attempt—that is, your work from a previous attempt is not saved.

Objectives

After completing this lab, you will be able to:

- Run through the DFX flow in the project environment
- Perform all steps (from synthesis to bitstream generation) in the GUI
- Create RMs, multiple configurations, and runs using the Dynamic Function eXchange Wizard
- Program the FPGA with full and partial bitstreams to evaluate the design

Introduction

The sample design used throughout this lab is *led shift_count*. This design is very small, which helps to minimize data size and allows you to run the lab quickly with minimal hardware requirements.

The design has two reconfigurable partitions (RPs) (**shift** and **count**). Each reconfigurable partition (RP) has two reconfigurable modules (RM), resulting in four total configurations (**shift_right**, **shift_left**, **count_up**, and **count_down**) for this design.

However, two of the four would be sufficient to consider all cases—one configuration with **count_up** and **shift_left** RM variations and the other configuration with the **count_down** and **shift_right** RM variations.

The DFX flow is now supported in both the Vivado Design Suite project-mode and non-project batch mode. This lab focuses on the DFX project flow.

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 where the lab files are located. This variable comes configured in the CloudShare/CustEd_VM environments.

Some tools can use this environment variable directly (that is, `$TRAINING_PATH` is 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.

Both the Vivado Design Suite and the Vitis platform offer a Tcl environment that is used in many labs. When the 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`.

General Flow

Step 1: Creating and Setting Up a Project for DFX → Step 2: Creating Multiple Configurat-ions → Step 3: Implement-ing the Configurat-ions → Step 4: Program-ming the Device and Evaluating

Creating a Project and Setting Up for DFX Step 1

The first step in this lab is to create a project and set up the design for Dynamic Function eXchange.

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.

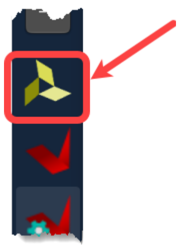


Figure 2-1: Launching the Vivado Design Suite from the Taskbar

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

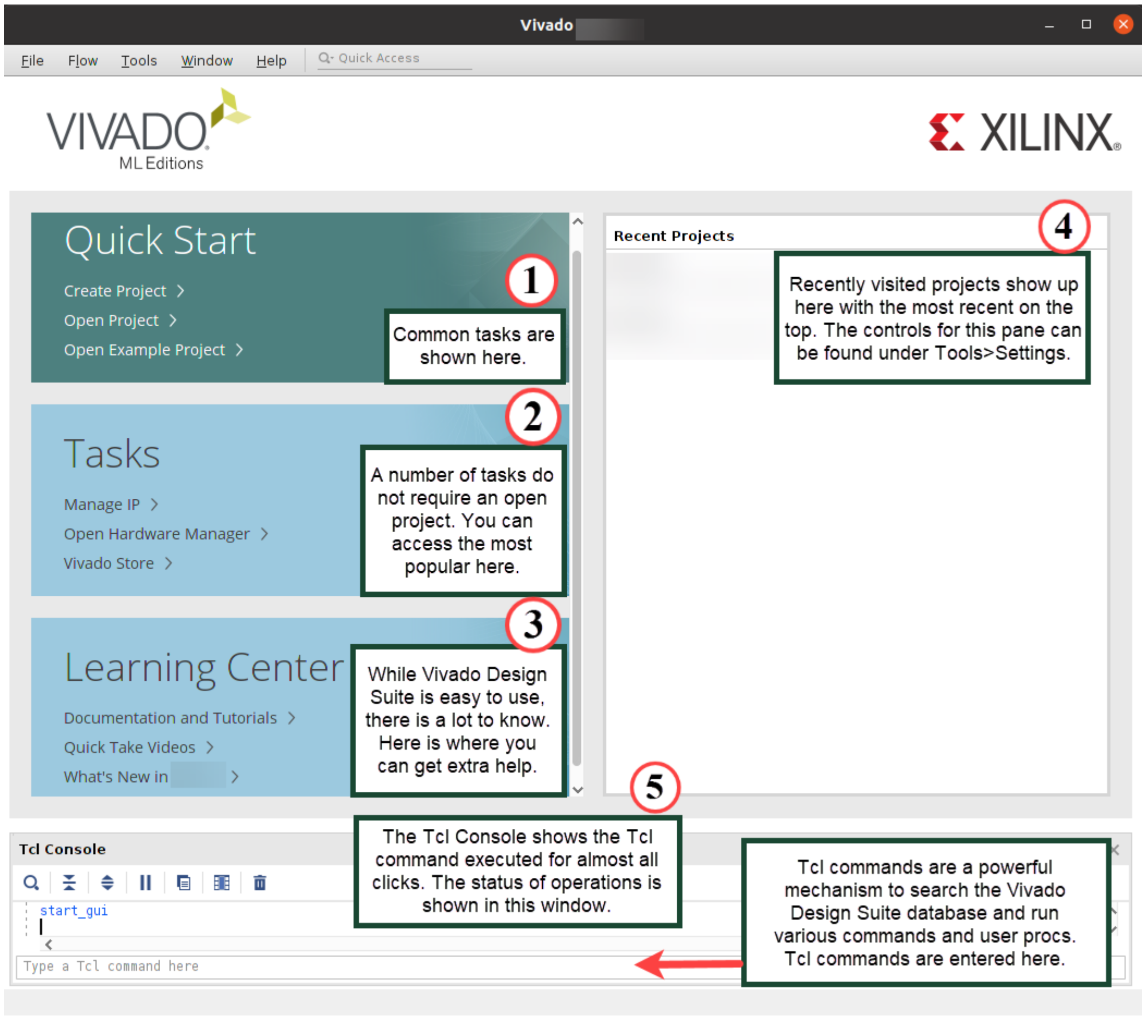
Alternatively, from the Linux terminal window (<**Ctrl** + **Alt** + **T**>), enter the following:

```
source /opt/amd/Vivado/2022.2/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 to the 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.



Common tasks are shown here.

A number of tasks do not require an open project. You can access the most popular here.

While Vivado Design Suite is easy to use, there is a lot to know. Here is where you can get extra help.

Recently visited projects show up here with the most recent on the top. The controls for this pane can be found under Tools>Settings.

The Tcl Console shows the Tcl command executed for almost all clicks. The status of operations is shown in this window.

Tcl commands are a powerful mechanism to search the Vivado Design Suite database and run various commands and user procs. Tcl commands are entered here.

```
start_gui
```

Type a Tcl command here

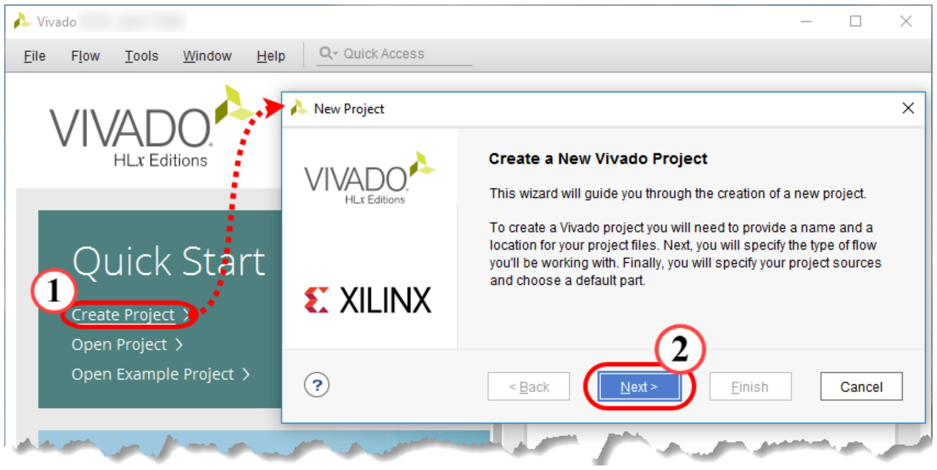
Figure 2-2: Vivado Design Suite Welcome Screen

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

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.

1-2. Create a new Vivado Design Suite project.

1-2-1. Click **Create Project** (1).



Create a New Vivado Project

This wizard will guide you through the creation of a new project.

To create a Vivado project you will need to provide a name and a location for your project files. Next, you will specify the type of flow you'll be working with. Finally, you will specify your project sources and choose a default part.

Figure 2-3: Creating a New Vivado Design Suite Project

This will launch the New Project Wizard.

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

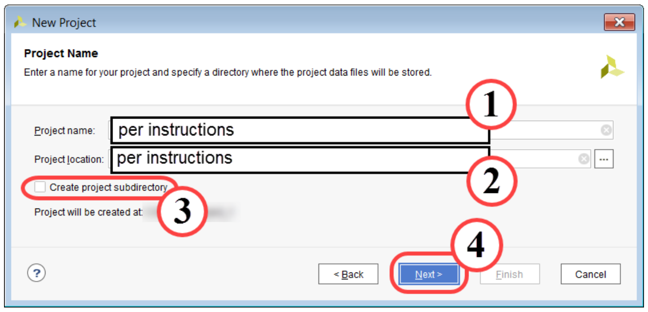
1-3. You will now encounter a series of dialog boxes asking you to enter different pieces of information describing the project.

1-3-1. Enter **DFX_proj** in the Project name field (1).

Enter **`$TRAINING_PATH/DFX_projflow/lab/ZCU104`** in the Project location field (2).

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

1-3-2. Select the **Create Project Subdirectory** option to create a level of hierarchy for the lab (3).



Project Name

Enter a name for your project and specify a directory where the project data files will be stored.

Project name: per instructions

Project location: per instructions

Create project subdirectory

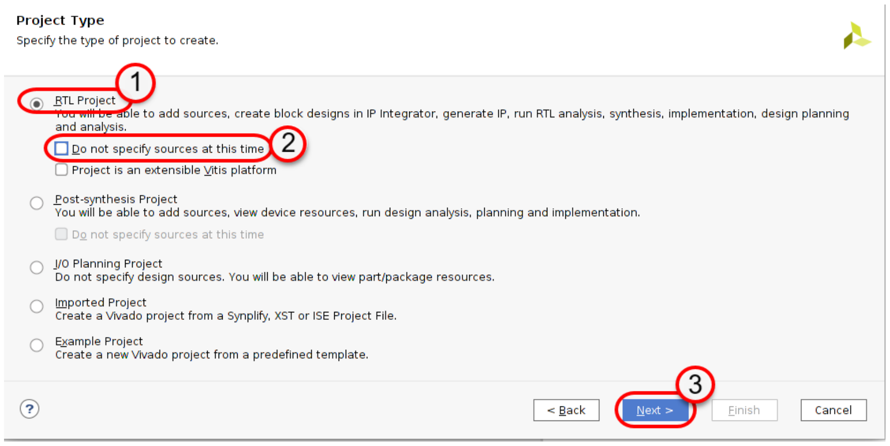
Project will be created at:

Figure 2-4: Entering the Project Name and Location

1-3-3. Click **Next** to accept the selections and advance to selecting a type of project (4).

1-3-4. Select **RTL Project** since this project will be based on source code (rather than a netlist) (1).

1-3-5. Deselect the **Do not specify sources at this time** option since you will be adding RTL files in the next instruction (2).


Project Type
Specify the type of project to create.

● RTL Project
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.
☐ Do not specify sources at this time
☐ Project is an extensible Vitis platform

○ Post-synthesis Project
You will be able to add sources, view device resources, run design analysis, planning and implementation.
☐ Do not specify sources at this time

○ I/O Planning Project
Do not specify design sources. You will be able to view part/package resources.

○ Imported Project
Create a Vivado project from a Synplify, XST or ISE Project File.

○ Example Project
Create a new Vivado project from a predefined template.

Figure 2-5: Setting the Project Type to RTL

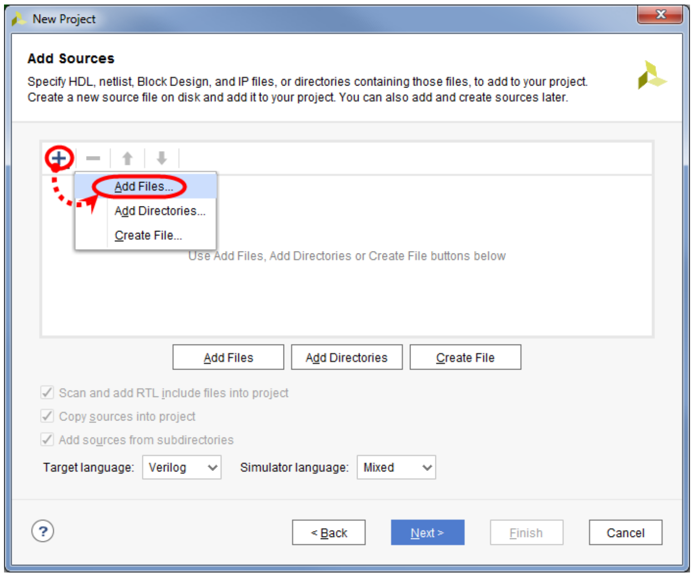
1-3-6. Click **Next** to accept the selection and advance to the adding sources stage (3).

1-4. 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.

1-4-1. Click the **Plus** icon (+) to begin adding objects to the project.

1-4-2. Select **Add Files** from the pop-up menu to begin adding your source files to the project.


New Project

Add Sources
Specify HDL, netlist, Block Design, and IP files, or directories containing those files, to add to your project. Create a new source file on disk and add it to your project. You can also add and create sources later.

Use Add Files, Add Directories or Create File buttons below

✓ Scan and add RTL include files into project
✓ Copy sources into project
✓ Add sources from subdirectories

Target language: Verilog   Simulator language: Mixed

Figure 2-6: 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.

1-4-3. Browse to the `$TRAINING_PATH/DFX_projflow/lab/ZCU104/Sources/hdl` directory (1).

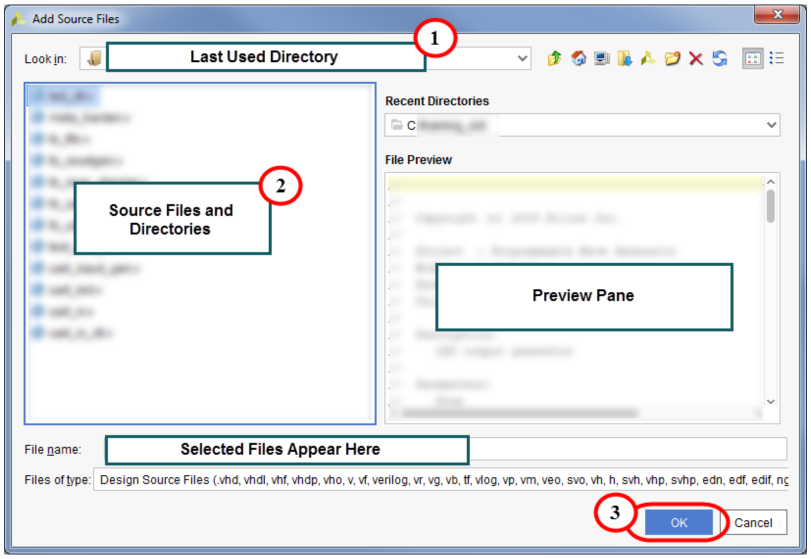

Add Source Files
Look in:
Last Used Directory
1
Recent Directories
File Preview
2
Source Files and Directories
Preview Pane
File name:
Selected Files Appear Here
Files of type:
Design Source Files (.vhd, vhdl, vhf, vhdp, vho, v, vf, verilog, vr, vg, vb, tf, vlog, vp, vm, veo, svo, vh, h, svh, vhp, svhp, edn, edf, edif, ng
3
OK
Cancel

Figure 2-7: Adding Source Files

1-4-4. Select or multi-select **top.v, clocks.xci** from the `top` directory (2).

1-4-5. Click **OK** to accept the selected files and add them as sources to the project (3).

1-4-6. Similarly add **count_up.v** from the `count_up` directory and **shift_left.v** from the `shift_left` directory.

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

1-4-7. Make sure the target language is set to **Verilog** and confirm that the **Copy sources into project** option is selected and click **Next**.

1-5. Add a constraints file to the Vivado Design Suite project.

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 `$TRAINING_PATH/DFX_projflow/lab/ZCU104/Sources/xdc` directory.

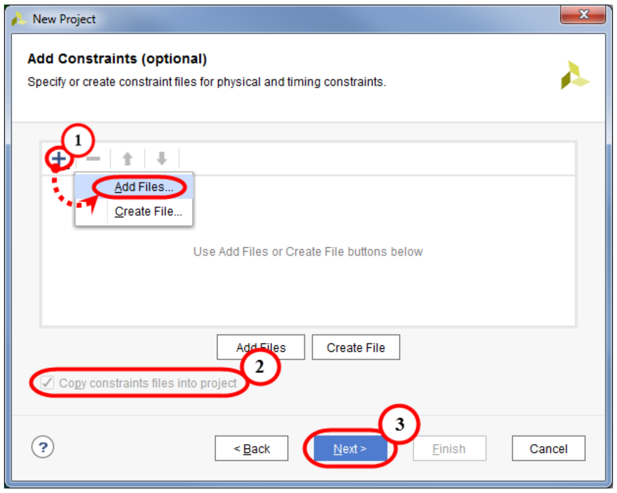

New Project
Add Constraints (optional)
Specify or create constraint files for physical and timing constraints.
1
Add Files...
Create File...
Use Add Files or Create File buttons below
Add Files
Create File
2
Copy constraints files into project
3
< Back
Next >
Finish
Cancel

Figure 2-8: Adding Constraints

1-5-4. Select **top_io.xdc**.

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).

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.

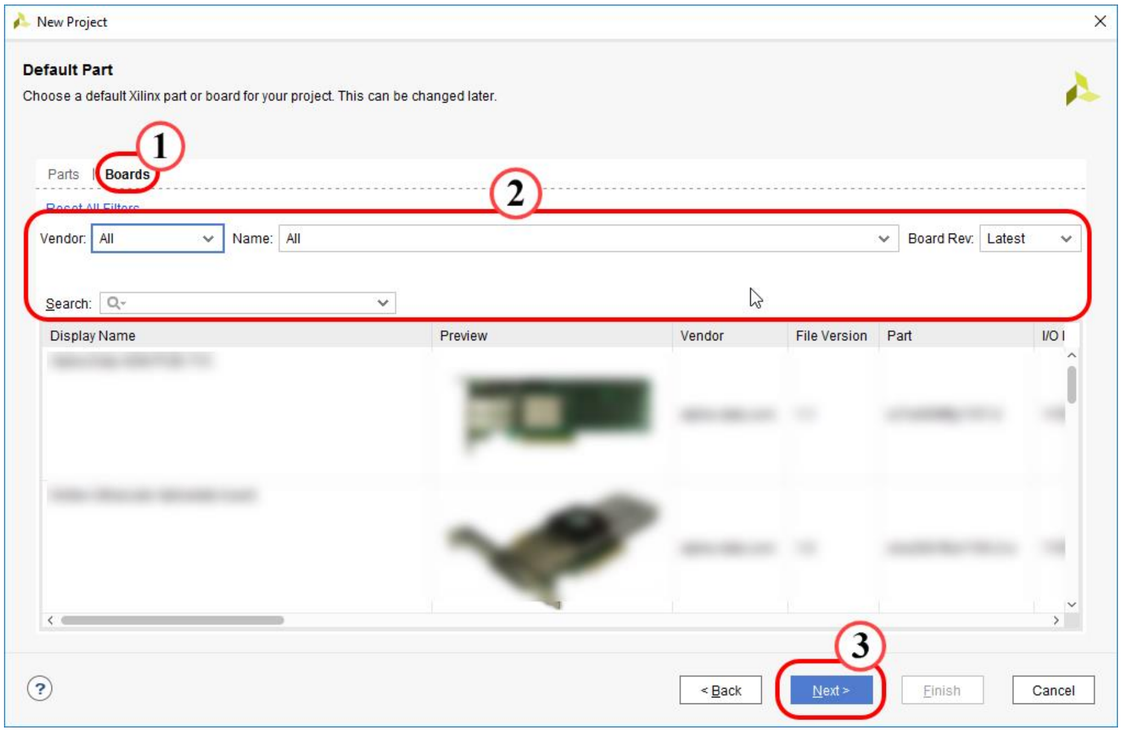
1-6-1. Select **Boards** from the Select area (1).

1-6-2. Select **All** 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 **Zynq UltraScale ZCU104 Evaluation Platform** from the board list.

Alternatively, you can select the board directly from the list at any time while in this dialog box.


New Project

Default Part

Choose a default Xilinx part or board for your project. This can be changed later.

Vendor: All Name: All Board Rev: Latest

Search:

Figure 2-9: 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.

Note: Dynamic Function eXchange is now included with the purchase of the Vivado Design Suite System and Design Edition licenses.

1-7. Examine the design sources.

1-7-1. Select the **Hierarchy** tab of the Sources window.

1-7-2. Expand **Design Sources** > **top**.

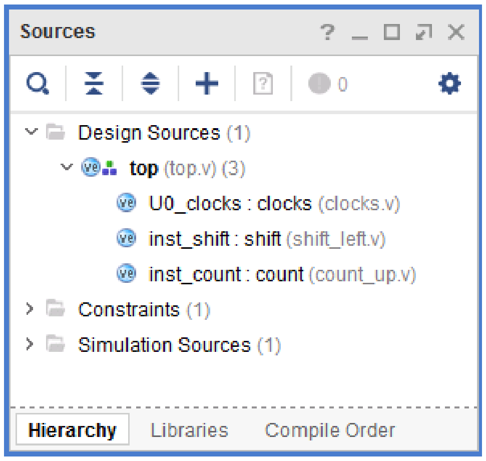


Figure 2-10: Design Sources

top.v is the top-level design file. inst_shift and inst_count are going to be the reconfiguration partition of this design.

1-8. Set up the design for Dynamic Function eXchange and define the reconfiguration partition.

1-8-1. Select **Tools** > **Enable Dynamic Function eXchange**.

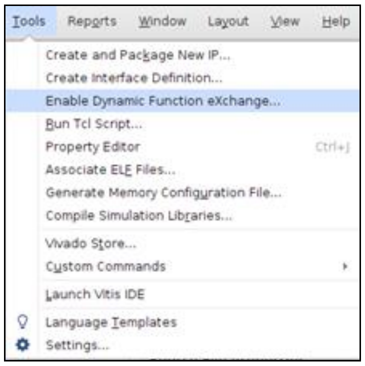


Figure 2-11: Enabling Dynamic Function eXchange

After creating the project in the standard method, the created project has to be set up for Dynamic Function eXchange. The Dynamic Function eXchange option sets the project to DFX.

1-8-2. Click **Convert** in the Enable Dynamic Function eXchange dialog box.

1-8-3. Right-click **inst_shift** in the Hierarchy tab of the Sources window and select **Create Partition Definition**.

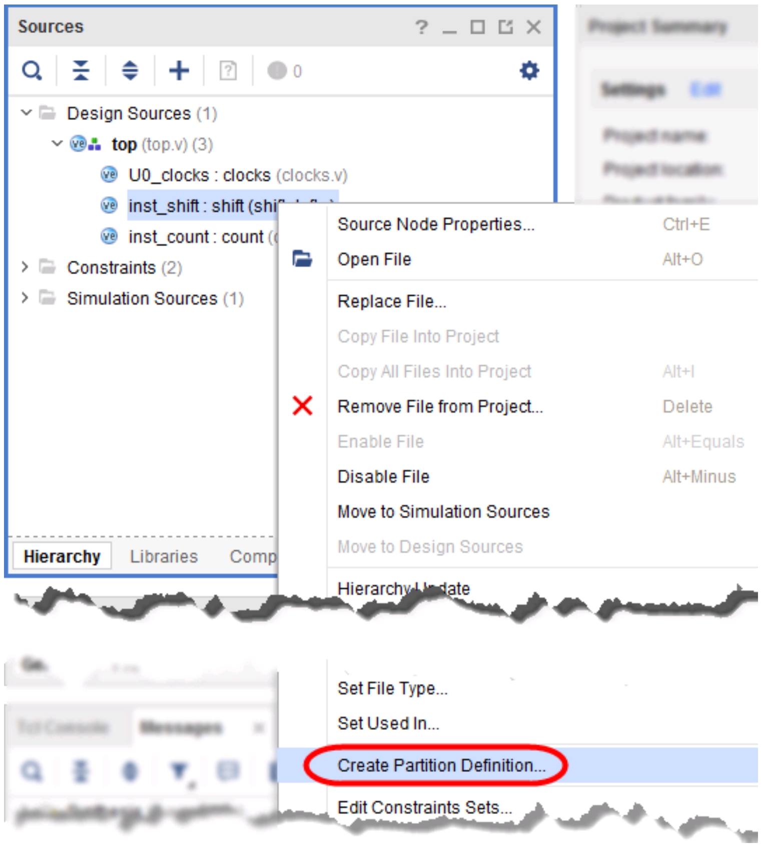
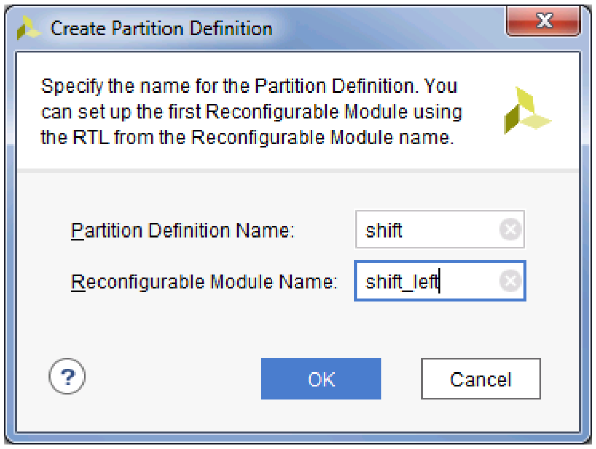


Figure 2-12: Creating a Partition Definition

The Create Partition Definition dialog box opens. Appropriate partition definition and RM names are to be specified.

1-8-4. Define the Partition Definition Name as **shift** and the Reconfigurable Module Name as **shift_left** and click **OK**.


Create Partition Definition

Specify the name for the Partition Definition. You can set up the first Reconfigurable Module using the RTL from the Reconfigurable Module name.

Partition Definition Name: shift

Reconfigurable Module Name: shift_left

Figure 2-13: Naming the Partition Defintition

1-8-5. Repeat the same partition definition step for the **inst_count** module with the Partition Definition Name as **count** and the Reconfigurable Module Name as **count_up** and click **OK**.

The Create Partition Definition option defines inst_shift and inst_count as RPs of the design. The inst_shift and inst_count instances have now changed with yellow diamonds indicating a reconfiguration partition. Out-of-context synthesis is run on these two instances.

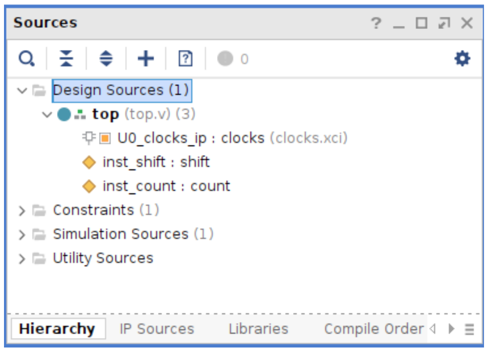


Figure 2-14: Reconfiguration Partitions

Creating Multiple Configurations for the Design Step 2

Here you will add the other Reconfiguration Modules to the design and create multiple configurations using the Dynamic Function eXchange Wizard.

The first configuration will be created with count_up and shift_left reconfigurable module variations. The second configuration will be created with count_down and shift_right RM variations.

2-1. Open the Dynamic Function eXchange Wizard and create the configuration runs.

2-1-1. Select **Dynamic Function eXchange Wizard** from the Flow Navigator.

The wizard appears in the main window of the Vivado IDE.

2-1-2. Click **Next** to edit or modify the RMs for this design in the Edit Reconfigurable Modules dialog box.

2-1-3. Click the **Plus** icon (+) in the Edit Reconfigurable Modules dialog box to begin adding the required RMs to the project.

2-1-4. Click the **Plus** icon again in the Add Reconfigurable Module dialog box and select **Add Files**.

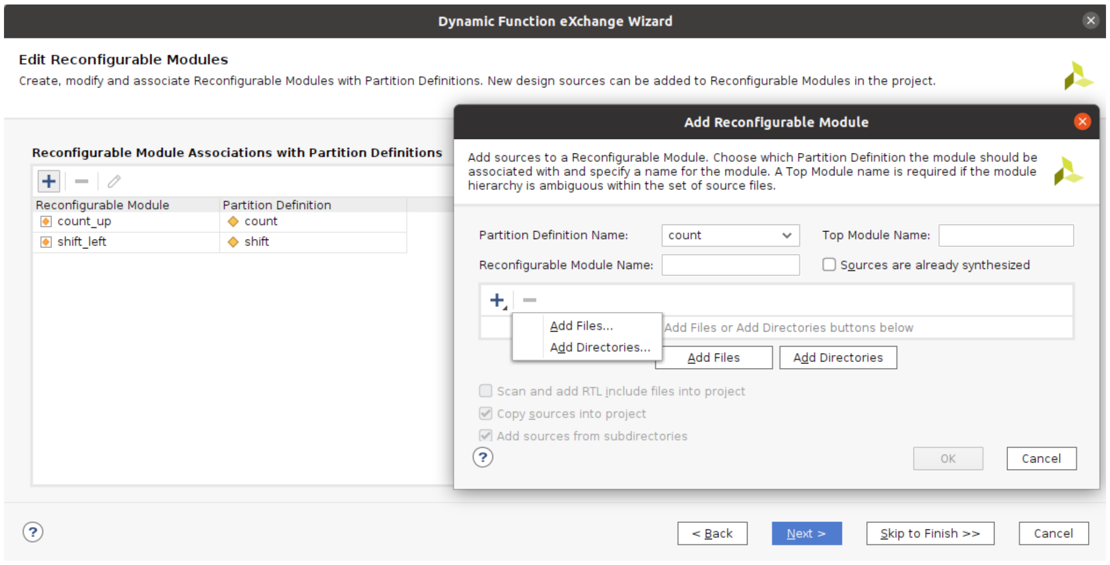

Dynamic Function eXchange Wizard
Edit Reconfigurable Modules
Create, modify and associate Reconfigurable Modules with Partition Definitions. New design sources can be added to Reconfigurable Modules in the project.
Reconfigurable Module Associations with Partition Definitions
Reconfigurable Module
Partition Definition
count_up
count
shift_left
shift
Add Reconfigurable Module
Add sources to a Reconfigurable Module. Choose which Partition Definition the module should be associated with and specify a name for the module. A Top Module name is required if the module hierarchy is ambiguous within the set of source files.
Partition Definition Name:
count
Top Module Name:
Reconfigurable Module Name:
Sources are already synthesized
Add Files...
Add Directories...
Add Files or Add Directories buttons below
Add Files
Add Directories
Scan and add RTL include files into project
Copy sources into project
Add sources from subdirectories
OK
Cancel
< Back
Next >
Skip to Finish >>
Cancel

Figure 2-15: Adding a Reconfigurable Module

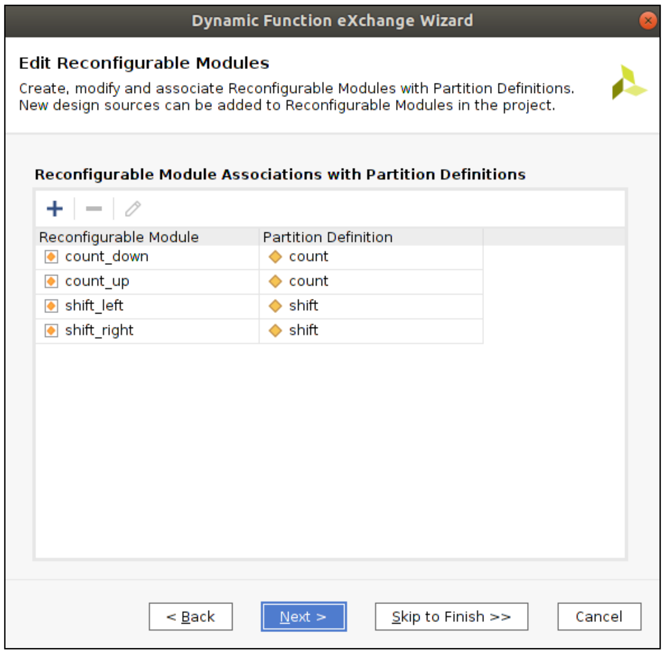
2-1-5. Browse to `$TRAINING_PATH/DFX_projflow/lab/ZCU104/Sources/hdl/count_down` and add **count_down.v**.

2-1-6. Set the Partition Definition Name to be *count* and the Reconfigurable Module Name to be *count_down*, leave the Top Module field empty and the *Sources are already synthesized* option unchecked.

2-1-7. Click **OK** to add the new RM.

2-1-8. Repeat the above steps to add another RM **shift_right** by adding the `shift_right.v` file located in `Sources/hdl/shift_right`, with the Partition Definition Name to be *shift* and the Reconfigurable Module Name to be *shift_right*.

2-1-9. Click **OK** to add the new RM to the design.



Dynamic Function eXchange Wizard

Edit Reconfigurable Modules

Create, modify and associate Reconfigurable Modules with Partition Definitions.
New design sources can be added to Reconfigurable Modules in the project.

Reconfigurable Module Associations with Partition Definitions

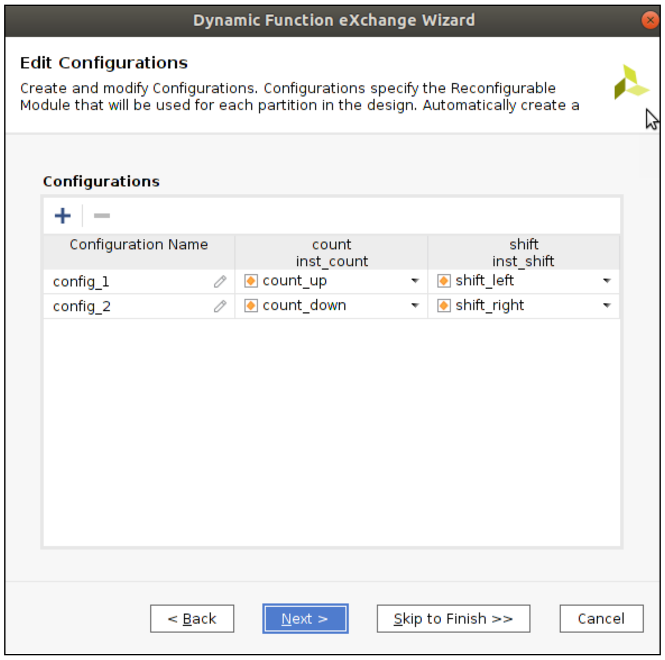
Reconfigurable Module	Partition Definition
count_down	count
count_up	count
shift_left	shift
shift_right	shift

Figure 2-16: Editing the Reconfigurable Modules

2-1-10. Click **Next** to proceed to edit the configurations.

2-1-11. Select the **automatically create configurations** link in the dialog box to let the wizard create the configurations.

The Dynamic function eXchange Wizard creates the two configuration runs: config_1 with count_up and shift_left variations and config_2 with count_down and shift_right variations.



Dynamic Function eXchange Wizard

Edit Configurations

Create and modify Configurations. Configurations specify the Reconfigurable Module that will be used for each partition in the design. Automatically create a

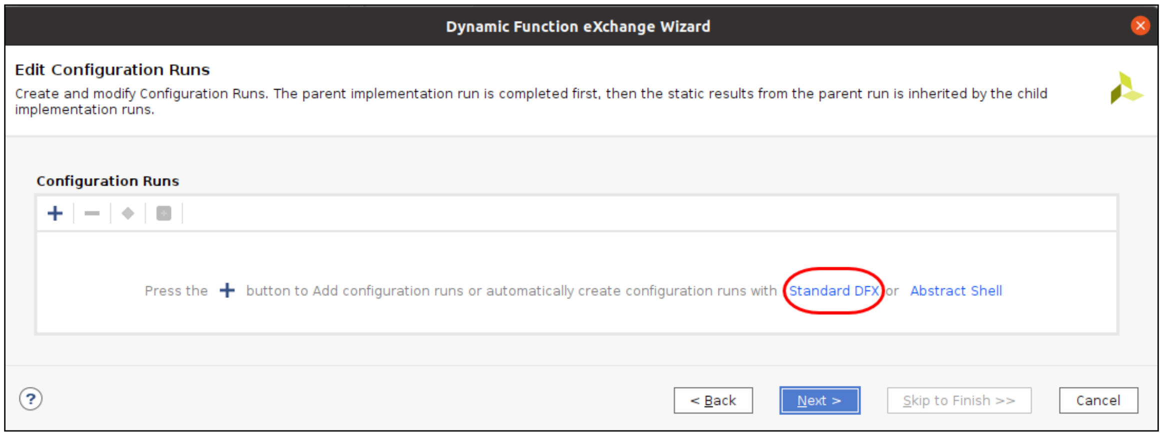
Configurations

Configuration Name	count inst_count	shift inst_shift
config_1	count_up	shift_left
config_2	count_down	shift_right

Figure 2-17: Generated Configurations

2-1-12. Click **Next** again to proceed to edit the configuration runs.

2-1-13. Click the **Standard DFX** link in the dialog box to let the wizard automatically create the configuration runs.



Dynamic Function eXchange Wizard

Edit Configuration Runs

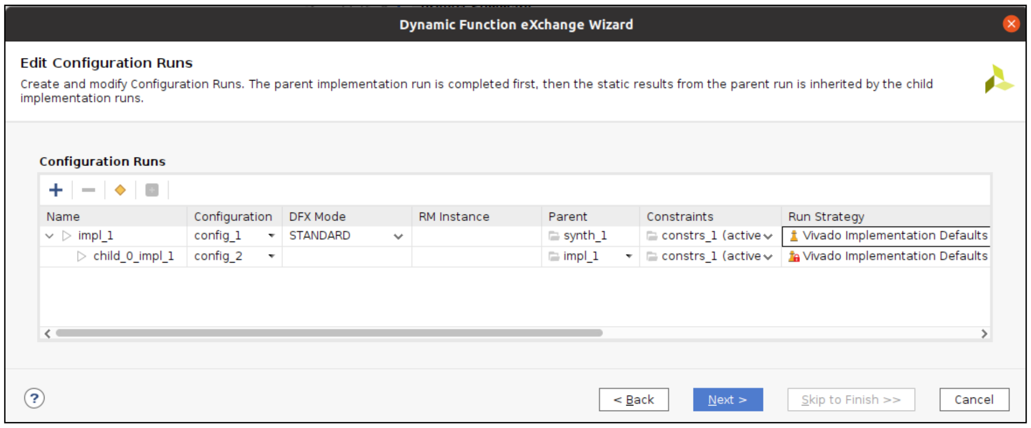
Create and modify Configuration Runs. The parent implementation run is completed first, then the static results from the parent run is inherited by the child implementation runs.

Configuration Runs

Press the + button to Add configuration runs or automatically create configuration runs with Standard DFX or Abstract Shell

Figure 2-18: Creating an Automatic Configuration Run with the Standard DFX Flow

Two minimum implementation runs are created: impl_1 and child_0_impl_1. The first (impl_1) is the parent run for configuration1, and child_0_impl_1 is the child run for configuration2.



Dynamic Function eXchange Wizard

Edit Configuration Runs

Create and modify Configuration Runs. The parent implementation run is completed first, then the static results from the parent run is inherited by the child implementation runs.

Configuration Runs

Name	Configuration	DFX Mode	RM Instance	Parent	Constraints	Run Strategy
impl_1	config_1	STANDARD		synth_1	constrs_1 (active	Vivado Implementation Defaults
child_0_impl_1	config_2			impl_1	constrs_1 (active	Vivado Implementation Defaults

Figure 2-19: Generated Configuration Runs

You can also create other two configurations with count_up, shift_right and count_down, and shift_left RM variations.

2-1-14. Click **Next** to view the Dynamic Function eXchange summary and **Finish** to complete the RM and configuration setup.

2-1-15. Review the two configuration runs (impl_1, child_0_impl_1) and OOC runs for RMs in the Design Runs window at the bottom of the Vivado IDE.

Implementing the Configurations Step 3

In this step, you will perform the synthesis and implementation process. You will also run a DFX configuration analysis report on the RMs and floorplan the design.

3-1. Add the Pblock constraints to the design.

3-1-1. Click **Add Sources** in the Project Manager.

3-1-2. Select **Add or create constraints** and click **Next**.

3-1-3. Click the Plus (+) icon to open the pop-up menu and select **Add Files** to open the Add Source Files dialog box, which allows you to browse to the desired directory.

3-1-4. Browse to the `$TRAINING_PATH/DFX_projflow/lab/ZCU104/Sources/xdc` directory if it is not open already.

3-1-5. Select the **pblocks.xdc** file and click **OK** in the Add Constraint Files dialog box.

3-1-6. Select **Finish** to add the file(s) to the project.

3-2. Synthesize the design.

3-2-1. Click **Run Synthesis** in the Flow Navigator under Synthesis and click **OK** to launch the run.

This process synthesizes all four RMs in OOC mode and then synthesizes the top level.

3-2-2. Verify that no errors are present.

If errors occurred, correct them and resynthesize. This process will take 2-3 minutes to complete.

3-2-3. Select **Open Synthesized Design** in the Synthesis Completed dialog box and click **OK**.

3-3. Run the Configuration Analysis report.

This report compares the set of reconfigurable modules.

3-3-1. Change the directory to the RM location by entering the following command in the Tcl Console:

```
cd $::env(TRAINING_PATH)/DFX_projflow/lab/ZCU104/DFX_proj/
DFX_proj.runs
```

3-3-2. Enter the following command in the Tcl Console to compare the *shift_left* and *shift_right* RMs of the shift partition:

```
report_pr_configuration_analysis -cells inst_shift -dcps
{./shift_left_synth_1/shift.dcp ./shift_right_synth_1/shift.dcp}
```

Currently this report is only available via a Tcl command.

3-3-3. Review the resource usage, clock usage of the two RMs *shift_left* and *shift_right* from the analysis report in the Tcl Console.

3-4. Floorplan the design.

3-4-1. Select the **inst_count** instance from the Netlist window.

3-4-2. Select the **Cell Properties** window > **General** tab and click **pblock_inst_count** to view the Pblock for the count partition in the Device view.

This Pblock is drawn from the constraint file *pblocks.xdc*. Similarly, view the Pblock for the shift partition.

3-4-3. Close the synthesized design.

3-5. Implement the design.

3-5-1. Click **Run Implementation** in the Flow Navigator under Implementation and click **OK** to launch the run.

Launching implementation runs the implementation process for the first configuration (impl_1) and then for the second configuration (child_0_impl_1).

This writes an OOC mode checkpoint for the routed RMs and locks the placement and routing of the static-only design.

First the implementation run for the parent configuration is completed and then the implementation run for second configuration (child_o_impl_1) will start.

3-5-2. Click **Generate Bitstreams** in the Implementation Completed dialog box and click **OK**.

[Linux users]:

Here, there will be two `top.bit` files that are generated:

- One under the `$TRAINING_PATH/DFX_projflow/ZCU104/DFX_proj/DFX_proj.runs/impl_1` directory, which corresponds to the first configuration (count_up, shift_left)
- And another under the `$TRAINING_PATH/DFX_projflow/ZCU104/DFX_proj.runs/child_0_impl_1` directory, which corresponds to the second configuration (count_down, shift_right)

3-5-3. Click **Cancel** once done.

Programming the Device and Evaluating the Design Step 4

This final step is the actual downloading of both the full bitstream and the partial bitstreams. Because this design has two relatively independent sections, you will notice the non-reconfiguring design continuing to run while the new partial bitstream is loaded into the other RP.

Set up and connect the ZCU104 evaluation board. Verify that this has been done properly before turning on the power.

For CloudShare users, if you have the evaluation board locally available, copy the project directory to the local path and proceed with the steps below. You can also review the "How to Perform Board Labs on Your Local Machine" document available in the Overview tab of the CloudShare environment.

Otherwise, you can just review (without performing) the connecting the board, programming the device, and verifying the design on the hardware sections shown below.

4-1. Connect the board to your machine.

4-1-1. Connect a USB cable from a USB port on your computer to the USB UART connector on the evaluation board.

4-1-2. Ensure that the power cord is plugged in and turn on the evaluation board.

4-1-3. Make sure that the board settings are correct.

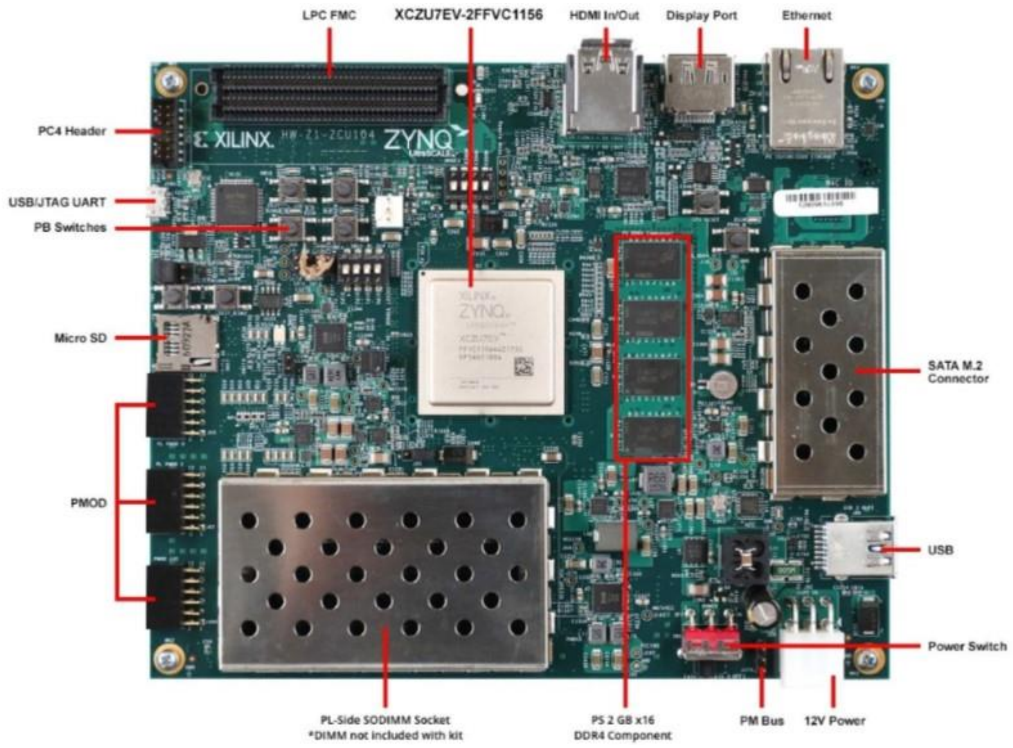


Figure 2-20: ZCU104 Evaluation Board

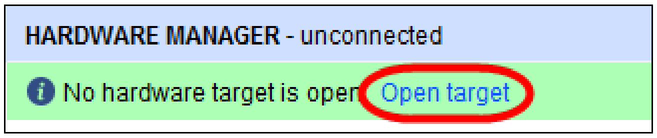
The hardware manager is the portion of the Vivado Design Suite that enables the monitoring of cores that were added to a design.

4-2. Open the hardware manager.

4-2-1. 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.



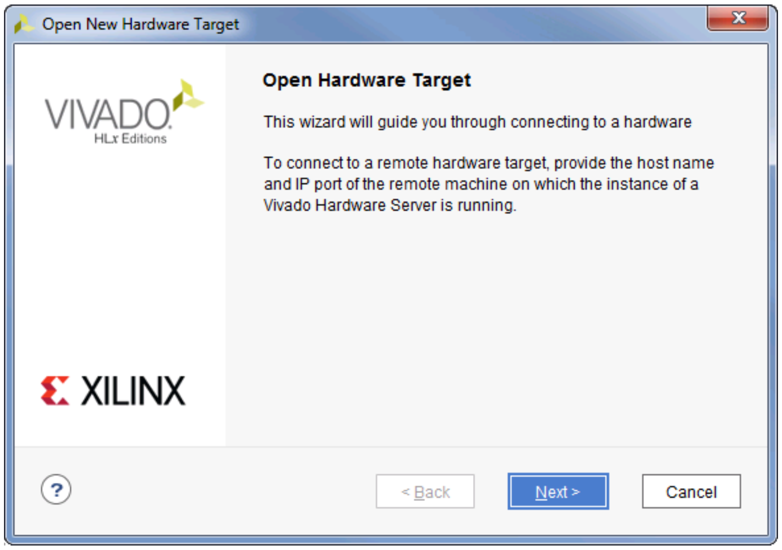
HARDWARE MANAGER - unconnected

No hardware target is open Open target

Figure 2-21: Opening a Hardware Target

4-3. Connect the target through the New Target Wizard to guide you through the process.

4-3-1. Click **Open target** > **Open New Target**.



Open New Hardware Target

Open Hardware Target

This wizard will guide you through connecting to a hardware

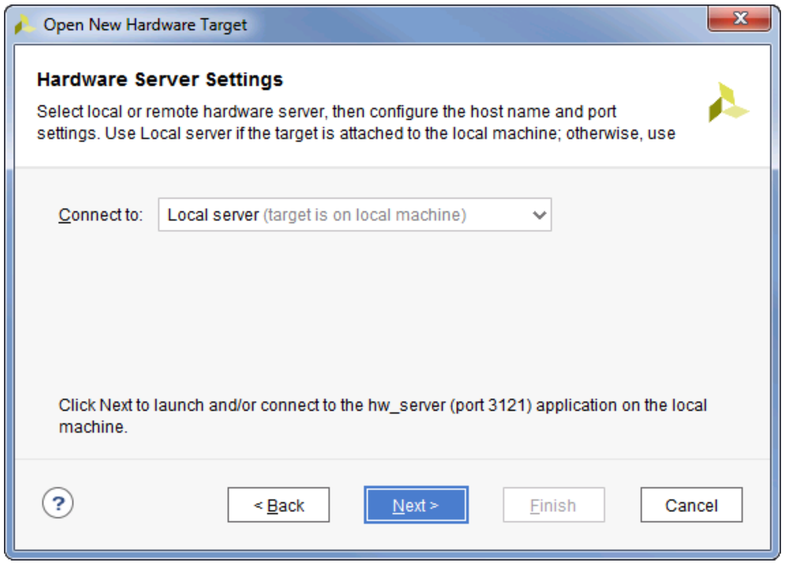
To connect to a remote hardware target, provide the host name and IP port of the remote machine on which the instance of a Vivado Hardware Server is running.

Figure 2-22: Open Hardware Target Dialog Box

4-3-2. Click **Next** to set the hardware server settings.

4-3-3. Enter a name for the server.

Typically, this is left at its default value.


Open New Hardware Target

Hardware Server Settings

Select local or remote hardware server, then configure the host name and port settings. Use Local server if the target is attached to the local machine; otherwise, use

Connect to: Local server (target is on local machine)

Click Next to launch and/or connect to the hw_server (port 3121) application on the local machine.

Figure 2-23: Setting the Server Name for the New Hardware Target

4-3-4. Click **Next** to select the hardware target.

4-3-5. 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.

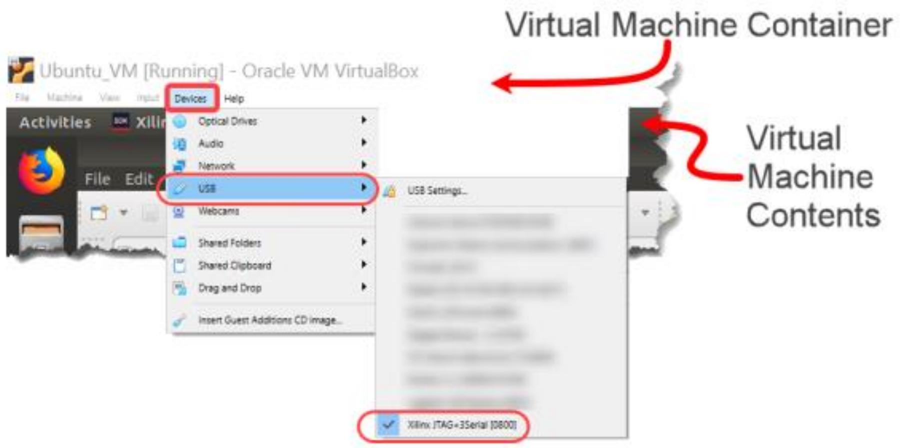


Figure 2-24: Selecting the Hardware Target

4-3-6. Leave the frequency at its default value.

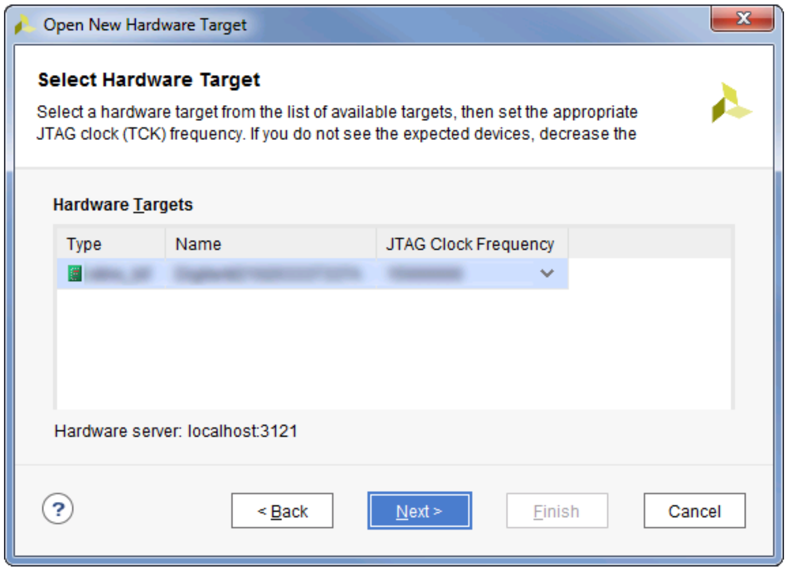
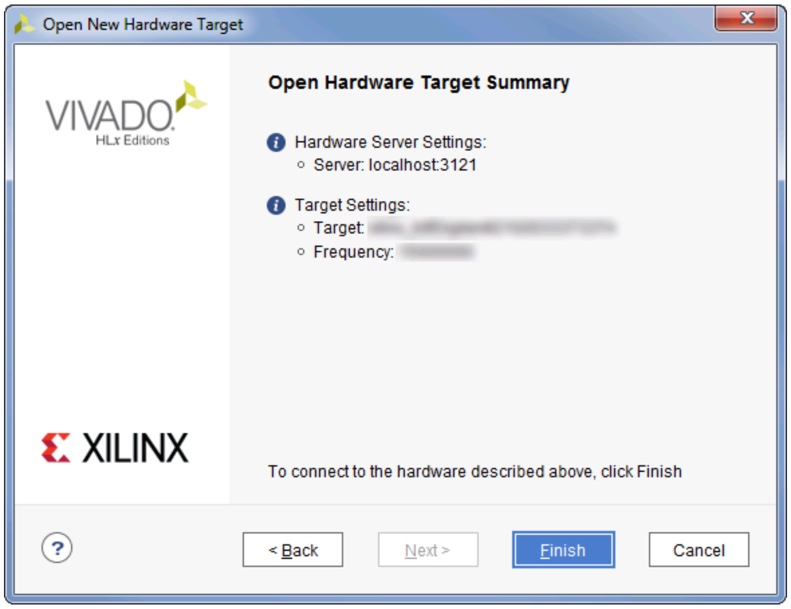

Open New Hardware Target
Select Hardware Target
Select a hardware target from the list of available targets, then set the appropriate JTAG clock (TCK) frequency. If you do not see the expected devices, decrease the
Hardware Targets
Type
Name
JTAG Clock Frequency
Hardware server: localhost:3121
?
< Back
Next >
Finish
Cancel

Figure 2-25: Selecting the Hardware Target

4-3-7. Click **Next** to view the hardware target summary.

A summary of the connection is displayed.



Open Hardware Target Summary

- Hardware Server Settings:
  - Server: localhost:3121
- Target Settings:
  - Target:
  - Frequency:

To connect to the hardware described above, click Finish

Figure 2-26: Summary of the Open Hardware Target Settings

4-3-8. Click **Finish** to connect to the new hardware target.

4-4. Program the device connected from the hardware session.

4-4-1. Right-click **xczu7_0** and select **Program Device**.

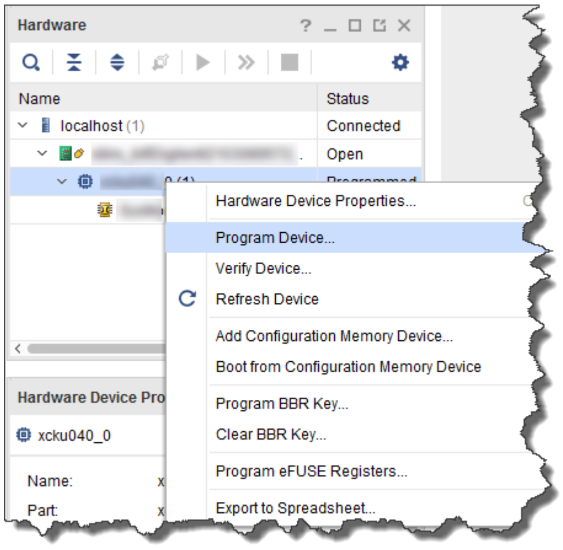


Figure 2-27: Programming the Device

The Program Device dialog box opens.

4-4-2. Browse to the `DFX_proj.runs/impl_1` directory and select **`top.bit`**, which corresponds to first configuration (count_up, shift_left).

4-4-3. Click **OK** to select the bitstream.

4-4-4. Leave the debug probes file field empty as there are no debug cores in the design.

4-4-5. Click **Program** in the Program Device dialog box to program the device with the selected bitstream.

When programming the device is complete, the status will show as Programmed.

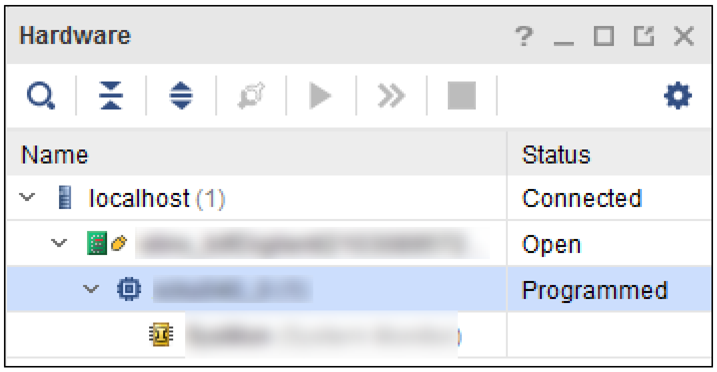


Figure 2-28: Programming Status of the FPGA

A progress bar appears and, when complete, the dialog box closes.

Notice that two GPIO LEDs are performing up counter operation on LEDs 2-3 and other two LEDs are performing shift left operation on the LEDs 0-1. Note the time taken to configure full device.

4-5. Program the top bitstream for the child configuration (Config_2).

4-5-1. Right-click **xczu7_0** and select **Program Device**.

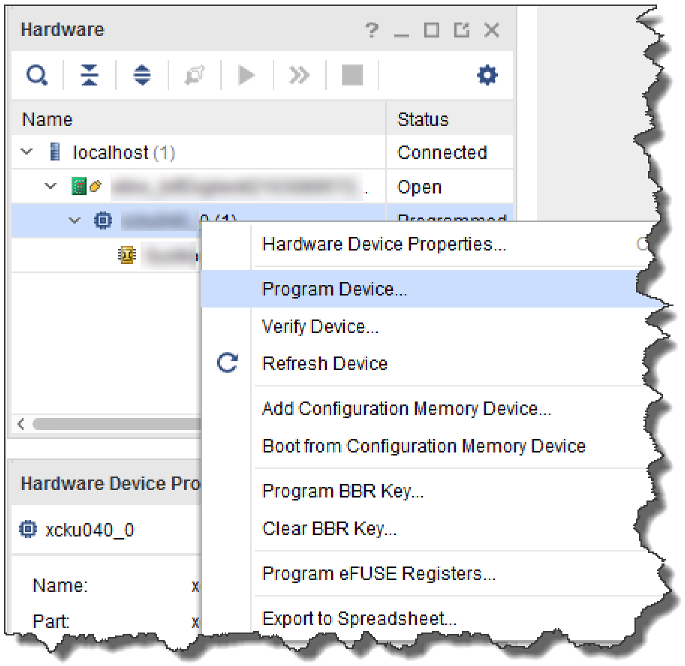


Figure 2-29: Programming the Device in the Hardware Session

The Program Device dialog box opens.

4-5-2. Select the **`top.bit`** file in the Bitstream file field from the `$TRAINING_PATH/DFX_projflow/lab/ZCU104/DFX_projflow.runs/child_0_impl_1` directory.

The `top.bit` file will be used to program the device and is also required to debug the DFX design, which is selected automatically.

4-5-3. Click **Program** in the Program Device dialog box to program the device with the selected bitstream.

When programming the FPGA is complete, the status will show as Programmed.

Notice that two GPIO LEDs are performing down counter operation on LEDs 2-3 and other two LEDs are performing shift right operation on the LEDs 0-1. Note the time taken to configure the device.

4-6. Power off the board.

4-7. Close the Vivado Design Suite project.

If you do not recall how to perform this task, refer to the "Closing the Vivado Design Suite Project" topic under Vivado Design Suite Operations in the *Lab Reference Guide*.

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/DFX_projflow` directory.

4-8. [Optional] [Only for local VMs—not for CloudShare] Clean up the file system.

Using the GUI:

4-8-1. Using the graphical browser (Windows: press the <**Windows**> key + <**E**>; Linux: press <**Ctrl** + **N**>), navigate to `$TRAINING_PATH/DFX_projflow`.

4-8-2. Select **`DFX_projflow`**.

4-8-3. Press <**Delete**>.

-- OR --

Using the command line:

4-8-4. Open a terminal window (Windows: press the <**Windows**> key + <**R**>, then enter **cmd**; Linux: press <**Ctrl** + **Alt** + **T**>).

4-8-5. Enter the following command to delete the contents of the workspace:

[Windows users]: **`rd /s /q $TRAINING_PATH/DFX_projflow`**

[Linux users]: **`rm -rf $TRAINING_PATH/DFX_projflow`**

Summary

After completing this lab, you should be familiar with the steps in the DFX project flow. This design has two relatively independent sections. You observed the non-reconfiguring design continuing to run while the new partial bitstream was loaded into the other RP.

You also used the Dynamic Function eXchange Wizard to add the RMs to the design and created multiple configurations through implementation. You also implemented the configuration runs, understanding the relationship between the parent and child implementation runs.