Third-Party Simulation Tool Usage Guidelines and Tips

Application Note

FPGA-AN-02084-1.1

July 2024

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults, and all associated risk is the responsibility entirely of the Buyer. The information provided herein is for informational purposes only and may contain technical inaccuracies or omissions, and may be otherwise rendered inaccurate for many reasons, and Lattice assumes no obligation to update or otherwise correct or revise this information. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. LATTICE PRODUCTS AND SERVICES ARE NOT DESIGNED, MANUFACTURED, OR TESTED FOR USE IN LIFE OR SAFETY CRITICAL SYSTEMS, HAZARDOUS ENVIRONMENTS, OR ANY OTHER ENVIRONMENTS REQUIRING FAIL-SAFE PERFORMANCE, INCLUDING ANY APPLICATION IN WHICH THE FAILURE OF THE PRODUCT OR SERVICE COULD LEAD TO DEATH, PERSONAL INJURY, SEVERE PROPERTY DAMAGE OR ENVIRONMENTAL HARM (COLLECTIVELY, "HIGH-RISK USES"). FURTHER, BUYER MUST TAKE PRUDENT STEPS TO PROTECT AGAINST PRODUCT AND SERVICE FAILURES, INCLUDING PROVIDING APPROPRIATE REDUNDANCIES, FAIL-SAFE FEATURES, AND/OR SHUT-DOWN MECHANISMS. LATTICE EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS OF THE PRODUCTS OR SERVICES FOR HIGH-RISK USES. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Inclusive Language

This document was created consistent with Lattice Semiconductor’s inclusive language policy. In some cases, the language in underlying tools and other items may not yet have been updated. Please refer to Lattice’s inclusive language FAQ 6878 for a cross reference of terms. Note in some cases such as register names and state names it has been necessary to continue to utilize older terminology for compatibility.

Contents

Figures

Tables

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviation	Definition
DUT	Device Under Test
GSR	Global Set/Reset
IP	Intellectual Property
OEM	Original Equipment Manufacturer
PUR	Power-Up Reset
RTL	Register Transfer Level

1. Introduction

Project simulation is one of the most important steps in the FPGA project development flow. However, one of the first steps is to decide on what simulation tool to use. All of Lattice's flagship software tools – the Lattice Radiant™ software, Lattice Diamond™ software, and Lattice Propel™ Builder software–come packaged with their own original equipment manufacturer (OEM) simulators. However, these OEM simulators have some feature and performance limitations that may not be ideal for everyone.

As an alternative to the included OEM simulators, Lattice also supports several different types of third party simulation tools, which you can use to simulate your own Lattice FPGA projects separately from any OEM simulation tool. These supported third-party non-OEM simulation tools include Mentor Graphics® ModelSim, QuestaSim®, Synopsys® VCS, and Cadence® Xcelium™.

In general, project simulation requires some register transfer level (RTL) that reflects the design a user wants to implement, as well as a testbench which is meant to model realistic stimulus for the device under test (DUT). However, depending on the intellectual property (IP) and primitives used in the DUT, there is some additional overhead that must be considered when using third-party simulation tools. If a design is purely RTL and only consists of Verilog or VHDL code, then out-of-the-box project simulation should be straightforward, and not require much additional consideration. However, it is common for user designs to utilize IP or primitives to model some on-device FPGA function. Since these IP and/or primitives are not purely RTL, a simulation library is required so simulation tools can resolve these references and understand how to model their behavior during simulation. For OEM simulators, Lattice includes precompiled simulation libraries. However, for third-party simulation tools, you are required to compile the relevant Lattice simulation libraries ahead of time before simulating your designs.

Each Lattice FPGA device family has its own simulation library, who's source files come packaged with every Radiant software and Diamond software installation. For the Diamond softwarer, both Verilog and VHDL libraries are supported, while for the Radiant software only Verilog-based simulation libraries are supported. The source files for each device's simulation library can be found within the following directory: *<Lattice tool installation directory>/cae_library/simulation/verilog/<device>*. The *<Lattice tool installation directory>* is the base directory where either the Radiant or Diamond software is installed to, and the *<device>* refers to the target device. Every Lattice tool release comes with pre-compiled versions of these libraries that are compatible with the respective tool's included OEM simulator so no additional work is required if you are using an OEM simulation tool from Lattice. However, for third-party simulators, it is recommended that you compile these libraries rather than reference their source files directly to improve simulation runtime and enable easier debugging of errors later down the line.

Note that all of Lattice's simulation libraries are Verilog-based. Because of this, a mixed-language simulation license is required for your target simulation tool of choice if you want to simulate a VHDL or mixed-language design with a third-party tool.

1.1. Compiling Lattice Simulation Libraries

Depending on your non-OEM simulation tool of choice, you have the option to compile your required Lattice simulation libraries before simulating your design with a third-party simulation tool ahead of time. This is true for Mentor Graphics ModelSim, QuestaSim, Cadence Xcelium, and Synopsys VCS simulators. Take note on the version dependency between different versions of the simulation tool of choice. For example, the pre-compiled simulation libraries with Lattice's OEM version of ModelSim are not directly compatible with other versions of ModelSim simulators, which you may have licensed through a third-party. Because of this, its recommended that you recompile your device libraries of choice ahead of time if you plan to update the version of simulation tool.

To assist in the process of compiling Lattice FPGA simulation libraries for use with third-party simulators, Lattice provides a *cmpl_libs* TCL script that you can use to semi-automatically compile all the relevant simulation files required to simulate a Lattice FPGA project. This script can be invoked directly from the Radiant or Diamond software's integrated TCL consoles or directly from a command-line environment. The purpose of this script is to simplify the process of creating simulation libraries for Lattice's various supported third-party simulation tools so you do not need to create these libraries manually, or compile all the RTL from a device's simulation library each time you run a simulation for your project.

With that said, the syntax for the *cmpl_libs* method of compiling libraries for use with third-party simulation tools is simple, which requires you to only specify up to four settings depending on your simulation tool of choice and target device.

The first and only mandatory setting is *-sim_path*, which is used to point towards the base installation directory of the third-party simulation tool of choice. For example, if Synopsys VCS is your simulator of choice, you must point to the directory containing its binary executable for this option.

Aside from the mandatory setting, this script also contains a few optional settings such as *-sim_vendor*, which is used to specify the target simulation tool of choice. The default setting is *Mentor* if nothing is specified, which will compile the specified simulation libraries for use with either the QuestaSim or ModelSim simulator depending on the binary specified with *-sim_path*. Although this setting is optional, it is crucial that you set this correctly if you are using a non-Mentor simulator, as the commands required to compile a simulation library vary from tool to tool. Additionally, it is important to keep in mind that the supported third-party simulators depend on whether you are using the Radiant or Diamond software. For the Diamond software, only the Mentor simulators are supported as a third-party simulation tool.

Aside from that, there is also a *-device* setting, which is used to specify the target devices to create simulation libraries for. If this option is not specified, then simulation libraries will be created for all the Lattice tool's supported devices. However, there is also the option to specify a device if you do not plan on using multiple device families, and want to reduce the amount of time taken to compile them with *cmpl_libs*. For this option, note that the available devices for selection depends on which Lattice software you use to simulate your design. For example, if you are using the *cmpl_libs* script from the Radiant software, then only devices that are supported within the Radiant software can be specified for this option.

For the Diamond software, there is also an additional *-lang* option, which can be used to select which hardware description language (HDL) your compiled simulation libraries will be based on. The three options you can select for this part of the cmpl_libs command are Verilog and VHDL, as well as "all" which creates simulation libraries for both languages.

Lastly, the *-target_path* setting is another optional setting that specifies where to generate the simulation libraries that are compiled and created with *cmpl_libs*. By default, the current directory that the script is invoked from will be selected if no option is set for *-target_path*. However, you can also specify any other directory using this option. Depending on the *-device* option selected, *cmpl_libs* will generate a few additional folders within this project directory that correspond to the Lattice simulation libraries that are required for project simulation. For example, if your selected device is *ice40up*, then a directory called *ice40up*, *uaplatform*, and *pmi* are generated for whichever *-target_path* you selected. Alternatively, if you select *all* instead of a specific device, then additional directories corresponding to the other supported devices are generated, such as *lifcl*, *lfcpnx*, *lfd2nx*, and so on.

The following lists the syntax for *cmpl_libs:*

Syntax: *cmpl_libs -sim_path* <target simulator base directory>
-sim_vendor <target simulator>
-device <target device>
-lang <Verilog | VHDL | all>
-target_path <location to generate compiled libraries>

The following table contains the descriptions for each of these command options.

Table 1.1. cmpl_libs TCL Script Settings and Options

Setting	Required?	Value(s)	Description
-sim_path	Yes	Directory	Base directory of the chosen third-party simulation tool.
-sim_vendor	No	**Radiant software:** Mentor (default), Cadence, Synopsys, Aldec **Diamond software:** Mentor (default)	Third-party simulation tool of choice. For the Diamond software, only Mentor software is supported.
-device	No	**Radiant software**: ice40up, lifcl, lfcpnx, lfd2nx, lfmxo5-25, lfmxo5-15d, lfmxo5-100t, lfmxo5-	Lattice FPGA to compile simulation libraries for.

Setting	Required?	Value(s)	Description
		55t, lavat, ut24c, ut24cp, all (default) **Diamond software**: sc, scm, ec, xp, ecp, machxo, ecp2, ecp2m, xp2, ecp3, machxo2, machxo3l, machxo3d, machxo3lfp, lptm, lptm2, ecp5u, ecp5um, lfmnx, lifmd, lifmdf, all (default)	
-lang	No	Verilog, VHDL, all (default)	HDL language of choice to compile simulation libraries (Diamond software only).
-target_path	No	Directory	The location to compile the selected device's simulation libraries. The default setting is the directory to which the *cmpl_libs* script is invoked from.

The following table lists the full device names for each cmpl_lib's *-device* option.

Table 1.2. Device Family Names for cmpl_libs -device Setting

Target Tool	Cmpl_lib Device Setting	Device Family Name
Radiant software	ice40up	iCE40 UltraPlus™
	lifcl	CrossLink™-NX
	lfcpnx ut24c ut24cp	CertusPro™-NX
	lfd2nx	Certus™-NX
	lfmxo5-25 lfmxo5-15d lfmxo5-100t lfmxo5-55t	MachXO5™-NX
	lavat	Avant™
Diamond software	xp	LatticeXP™
	xp2	LatticeXP2™
	sc scm	LatticeSC™/M
	ec ecp	LatticeEC™/P
	ecp2 ecp2m	LatticeECP2™/M
	ecp3	LatticeECP3™
	ecp5u ecp5um	LatticeECP5™
	lptm lptm2	Lattice Platform Manager
	machxo2	MachXO2™
	machxo3l	MachXO3L™
	machxo3d	MachXO3D™
	machxo3lfp	MachXO3LF™
	lfmnx	Mach™-NX
	lifmd	CrossLink™
	lifmdf	CrossLinkPlus™

The following lists the methods to compile simulation libraries with the cmpl_libs script:

- Using the Integrated TCL console on the Radiant or Diamond software
- Using the pnmainc application or the Radiant or Diamond software directory
- Using the Command Line

1.1.1. Method 1: Using the Integrated TCL Console

Access the Integrated TCL console located at the bottom page of the Radiant and Diamond software GUI. If the TCL console cannot be located from your perspective, enable it through **View > Show Views > TCL console.** One thing to keep in mind when using this method is the default location compiled libraries are generated to. If a project is not open, the compilation directory defaults to the <Lattice tool installation directory>/bin/nt64 directory. If a project is open, the compilation directory defaults to the directory of the active project unless the *-target_path* option is used. Because of this, Lattice recommends using the *-target_path* option to avoid any potential issues or confusion.

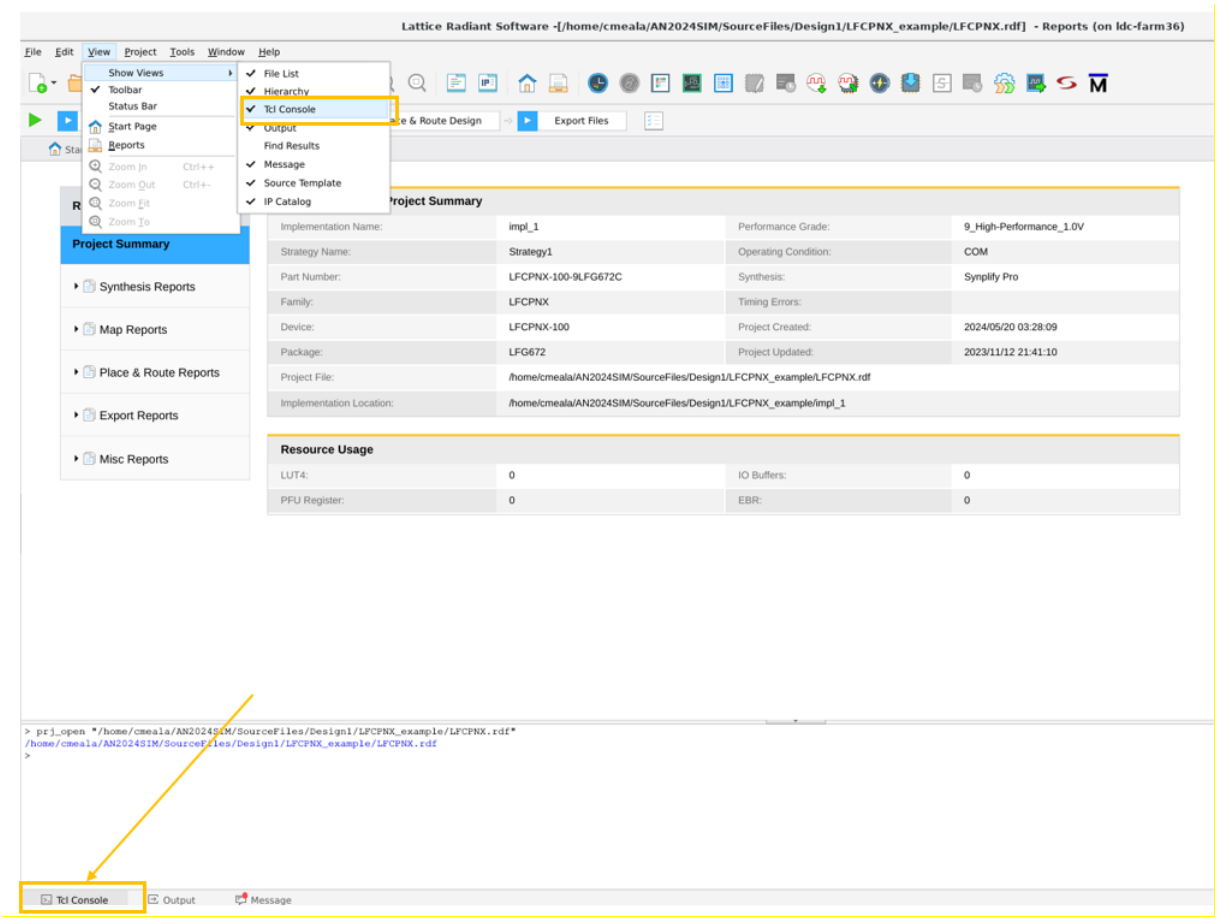

Figure 1.1. Radiant Software Integrated TCL Console

The following are a few examples for compiling simulation libraries using the cmpl_libs script:

1. cmpl_libs: this allows you to verify the Usage/Syntax used for the *cmpl_libs* command.
2. cmpl_libs *-sim_path* /tools/dist/cadence/XCELIUM/XCELIUM20.09.004/Linux/tools.lnx86/bin/ *-sim_vendor* cadence -device lfd2nx *-target_path* /home/cmeala/AN2024SIM/radiant_models/lfd2nx_model/:
    a. *-sim_path* – has been targeted to the 3rd party tool's binary executable.
    b. *-target_path*– is targeted to a folder named lfd2nx_model. This folder is where your logical libraries are located once library compilation is successful.
3. The libraries are successfully generated with the following message:
   Libraries located in <target_path> is updated.

Figure 1.2. TCL Command Examples for Compiling Simulation Libraries

1.1.2. Method 2: Using the Standalone TCL Console

Aside from the integrated TCL console described in Method 1, both the Diamond and Radiant software also support a standalone TCL console, which can also be used to compile your simulation libraries. Ultimately this method works the same as Method 1, with the main difference being how you interact with the Radiant or Diamond software's TCL consoles. In this flow, as its name implies, you use a standalone TCL console rather than the one integrated within the Radiant or Diamond software. After you have opened the TCL console, the following few steps to compile the simulation libraries are exactly the same.

To open the standalone TCL console for the Radiant software, execute the pnmainc or radiantc application in the following directory:

- For Windows operating system, these files are located at <Radiant installation directory>/bin/nt64.
- For Linux operating system, you must use the radiantc file that is located at <Radiant installation directory>/bin/lin64.

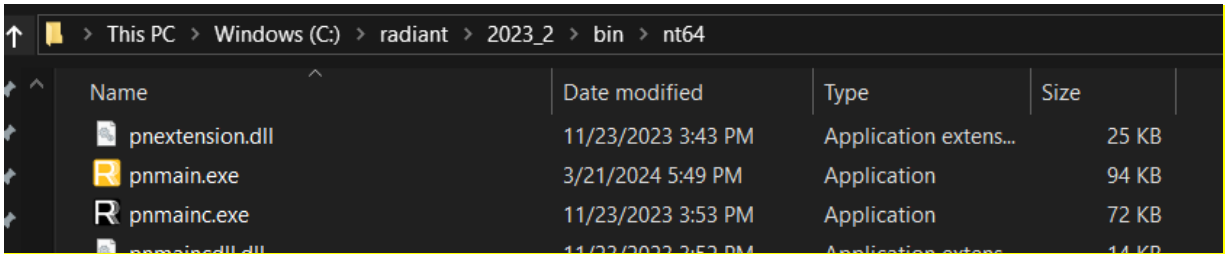

Figure 1.3. Folder Location of pnmainc

To open the standalone TCL console for the Diamond software, execute the pnmainc application in the following directory:

- For Windows, the file is located at <Diamond installation directory>/bin/nt64.
- For Linux, you must use the diamond file located at <Diamond installation directory>/bin/lin64.

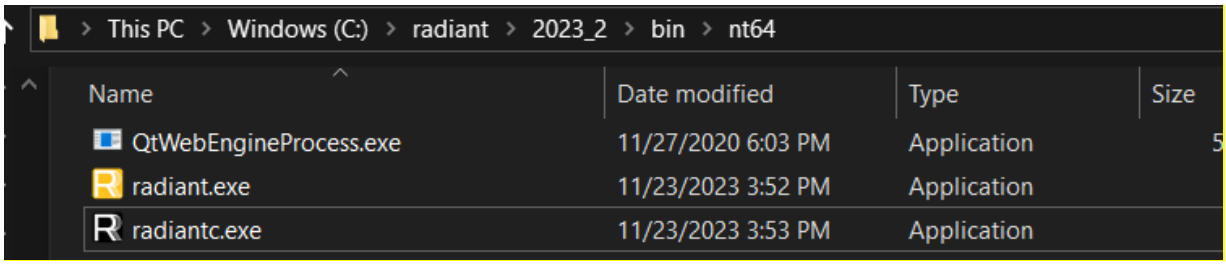

Figure 1.4. Folder Location of the Radiant Software Application

Note: You may use the same commands from the example in the Method 1: Using the Integrated TCL Console section to compile the simulation libraries.

Figure 1.5. TCL Command Examples for Compiling Simulation Libraries Using the radiantc Application

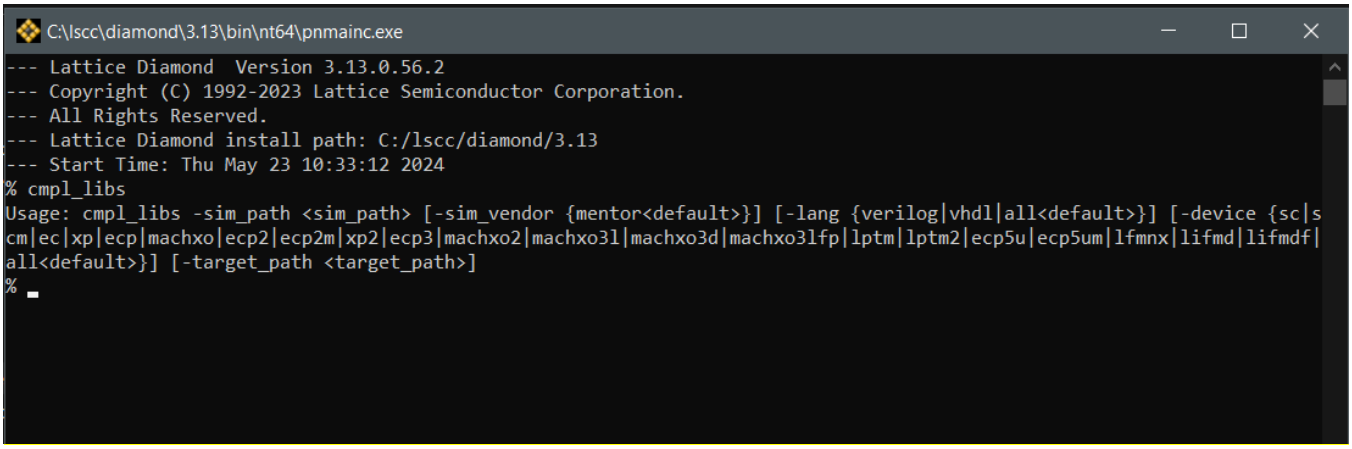

Figure 1.6. TCL Command Examples for Compiling Simulation Libraries Using the pnmainc Application

1.1.3. Method 3: Using Command Line

To compile simulation libraries using the command line, you must set the following few environment variables so your command prompt will be able to recognize Lattice specific commands. The following lists all the environment variables that you must set:

- For Windows operating systems execute the following commands:
  - PATH:
    Set PATH = <SW_Installation_Directory>\bin\nt64\; <SW_Installation_Directory>\ispfpga\bin\nt64
  - FOUNDRY:
    set FOUNDRY= <SW_Installation_Directory>\ispfpga
- For Linux operating system:
  - FOUNDRY
    If you are using CSH: setenv FOUNDRY/<SW_Installation_Directory>/ispfpga
    If you are using BASH: export FOUNDRY=/<SW_Installation_Directory>/ispfpga
  - bindir
    If you are using CSH: setenv bindir/<SW_Installation_Directory>/bin/lin64
    If you are using BASH: export bindir= /<SW_Installation_Directory>/bin/lin64

For more information, refer to the *Setting Up the Batch Environment* section in the Scripting Lattice FPGA Build Flow Application Note (FPGA-AN-02073).

The following figure shows an example of setting up the environment variables on a Windows platform.

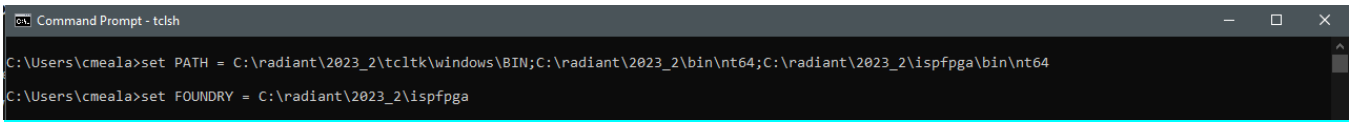

Figure 1.7. An Example of Setting Up the Environment Variables on a Windows Platform

The following figure shows an example of setting up the environment variables on a Linux platform.

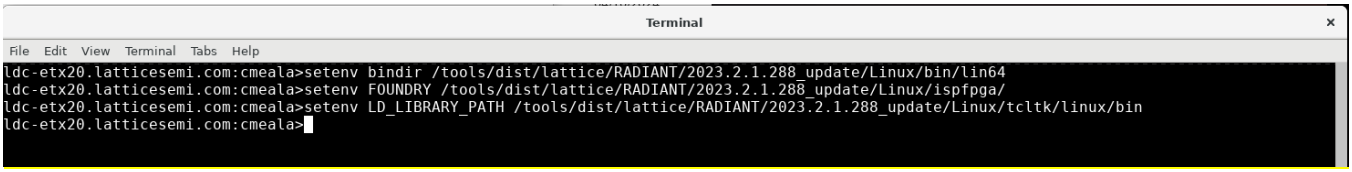

Figure 1.8. An Example of Setting Up the Environment Variables on a Windows Platform

Note: Ensure that you have TCL version 8.5 and above on your machine. If you do not have a TCL installed, or you have an older version, you must add the tcl bin directory on the PATH variable (for Windows), or on the LD_LIBRARY_PATH variable (for Linux), as shown in Figure 1.7 and Figure 1.8.

The following lists the generic syntax for setting the variables:

- Windows: <SW_Installation_Directory>\tcltk\windows\BIN
- Linux: <SW_Installation_Directory>\tcltk\linux\bin.

After the variables are all set, you must invoke the TCL script through the *tclsh* command on the command line to begin utilizing the *cmpl_libs* script on the command line with tcl. To verify that the *tclsh* command is running correctly on your machine, run the *info tclversion* command**.**

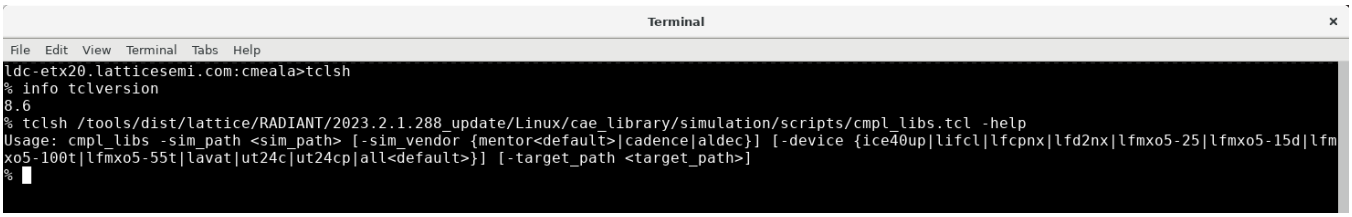

Figure 1.9. Running tclsh from the Command Line or Terminal

In comparison to Method 1 and Method 2, running the *cmpl_libs* script in Method 3 requires you to define the directory of the script file.

The following figure shows an example of libraries that are compiled correctly.

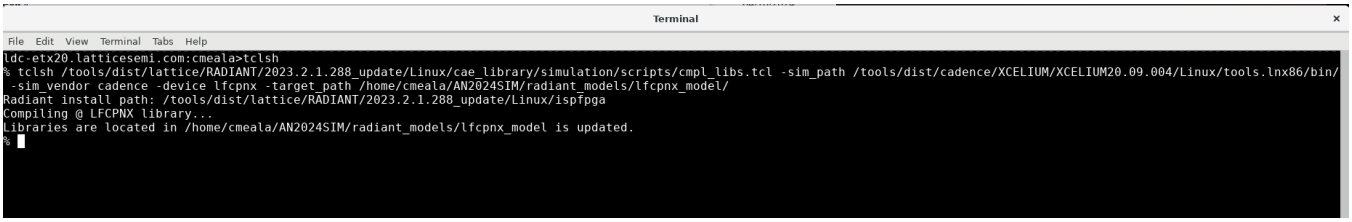

Figure 1.10. Compiling cmpl_libs with tclsh

The following figure shows an example of the content of the compiled libraries.

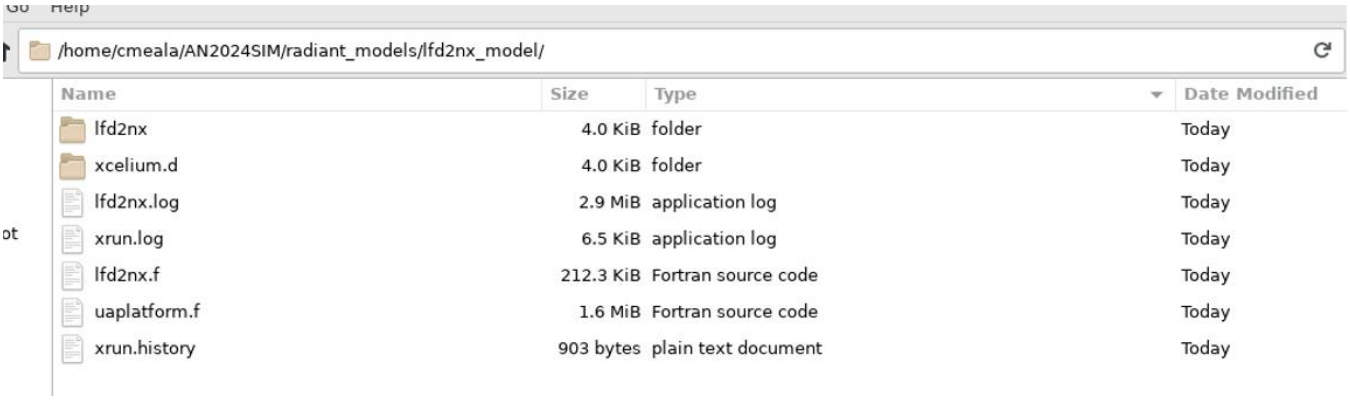

Figure 1.11. Example of cmpl_libs Library Outputs

After using the *cmpl_libs* script to generate the required Lattice simulation libraries for use with the selected third-party simulation tool, the next steps in the project simulation flow depend on the simulator you plan on using. For more information on the next steps in each of these respective simulation flows, refer to the Lattice FPGA Simulation with Third-Party Tools section.

2. Lattice FPGA Simulation with Third-Party Tools

This section describes the various third-party simulation tools that you can use to simulate your Lattice FPGA design.

2.1. Mentor Graphics ModelSim or QuestaSim Simulator

When you simulate your design with a non-OEM version of the QuestaSim or ModelSim simulator, take note on the version dependencies of the simulator you are using. For example, the libraries that are compiled for the OEM version of the ModelSim simulator, which is included in both the Radiant and Diamond software is not directly compatible with a non-OEM version of the ModelSim simulator, even if it is the same release number. With that in mind, the first thing you need to do when working with either the ModelSim or QuestaSim simulator is to compile the Lattice FPGA simulation libraries for the simulation tool version you are using by pointing to the correct installation directory using the *cmpl_libs* script's *-sim_path* option.

```
[Library]
lifcl = C:/Files/lattice_libs/lifcl
pmi = C:/Files/lattice_libs/pmi_work
```

Figure 2.1. Library Mappings in an Active ModelSim.ini file for a Third-Party Version of the ModelSim Software

The next step in the project simulation flow is to update your library mappings for the active simulation tool you are using. To do this, enter some additional lines under the *[Library]* tag of the modelsim.ini for your active simulation tool of choice. The purpose of these library mappings is to map a logical library (e.g., *lifcl*) to the physical directory containing the compiled libraries. After this modification has been completed, you can proceed to use any of these simulation libraries as any other kind of simulation library in the ModelSim simulator. Based on the example in Figure 2.1, the correct usage in a scripted flow would be to call the ModelSim or QuestaSim *VLOG* command with the following options *-L lifcl -L uaplatform -L pmi* so that the ModelSim simulator knows which simulation libraries to use when simulating a design.

2.2. Cadence Xcelium Simulator

For Cadence Xcelium simulator, the process for utilizing the output of the *cmpl_libs* script is straightforward. After the *cmpl_libs* script has finished running, the next step is to invoke Xcelium with the testbench files for simulation followed by the *-reflib* option. The *-reflib* option works similarly to the *-L* option in the ModelSim or QuestaSim simulator. The *-reflib* option specifies the pre-compiled simulation libraries that you must use during simulation to resolve unresolved references to device-specific simulation primitives. After the *xrun* command has finished running, the output is a simulation snapshot that reflects the implemented result of the simulation. This snapshot can be loaded by calling the *xrun* command with the *-R* option to enable further debugging and analysis.

The following list is an example of a script on utilizing the compiled libraries and running the design with the Xcelium simulator, as shown in shown in Figure 2.2:

- Line 1: *-reflib*—location of libraries compiled using the cmpl_libs script.
- Line 2: Indicates the design files to be simulated and added under the work library.
- Line 3: *-y*—Allows you to specify a library directory. If the tool is not able to locate any of the modules within the source code, it will search for those missing modules in any directories defined here.
- Line 4: *-incdir*— Allows you to specify the directory of the include files, if your RTL in line 2 includes any includes.
- Line 5: *-libext*— Indicates the tool extension for modules searched under *-y*. In this example, Xcelium will search all the files with the .v extension in the -y path.
- Line 6: *-sv*—Allows usage of system Verilog constructs.
- Line 7: *-l*—Provides an option for you to generate a log file.

```
Open   sim_script.f   ~/AN2024SIM   Save

1 -reflib /home/cmeala/AN2024SIM/radiant_models/lfcpnx_model/lfcpnx

2 /home/cmeala/AN2024SIM/SourceFiles/cpnx_tcl/count_tf.v
  /home/cmeala/AN2024SIM/SourceFiles/cpnx_tcl/top_counter.v
  /home/cmeala/AN2024SIM/SourceFiles/cpnx_tcl/osc_test/rtl/osc_test.v

3 -y /home/cmeala/AN2024SIM/SourceFiles/cpnx_tcl/

4 -incdir /home/cmeala/AN2024SIM/SourceFiles/cpnx_tcl/


5 -libext .v
6 -sv
-l sim_test.log
7
```

Figure 2.2. Example Script to Simulate a Project Using Xcelium

After the script has been defined as shown in Figure 2.2, run the script by invoking the *xrun* command. The *-f* option allows you to specify the script file you want to load with Xcelium.

```
Terminal
File  Edit  View  Terminal  Tabs  Help
ldc-etx20.latticesemi.com:cmeala>xrun -v_XCELIUM20.09.004 -f /home/cmeala/AN2024SIM/sim_script.f -gui
```

Figure 2.3. Command to Run the Example Script

The following figure shows the result from running the script file.

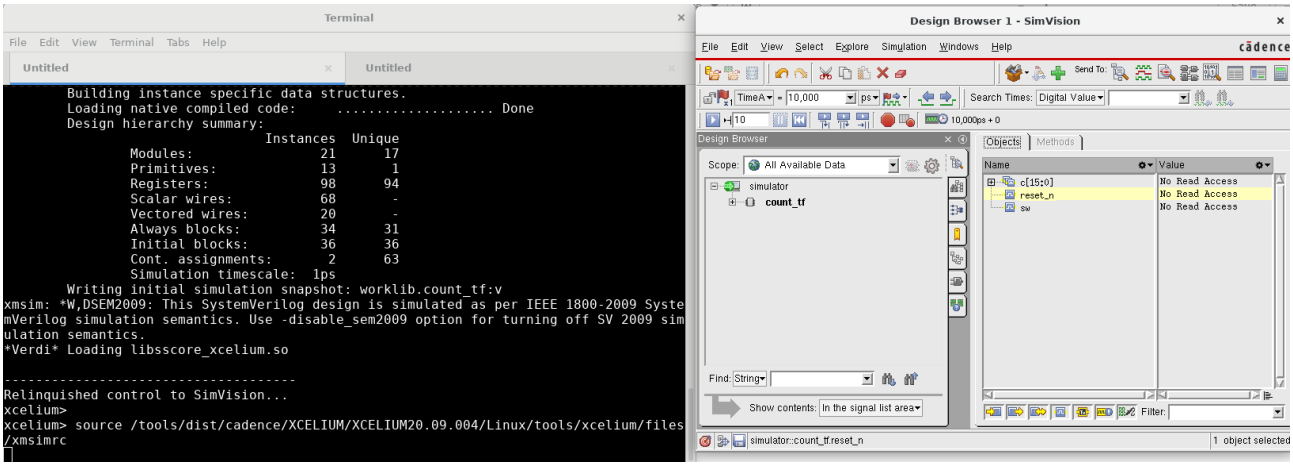

Figure 2.4. Results from Running the Example Script

2.3. Synopsys VCS Simulator

The compilation of Lattice FPGA simulation libraries for use with the Synopsys VCS simulator is currently only supported for Lattice Avant™ device family. To simulate Lattice FPGA projects targeting other devices with the VCS simulator, you must point directly to the source RTL for the associated simulation libraries for that device using other methods, such as *-y* or *-incdir*. If you are targeting an Avant device and have finished compiling the associated device simulation libraries, the next step is to modify the *synopsys_sim.setup* file that is generated to map your logical simulation library to its physical location on your file system. After that has been done, copy the *synopsys_sim.setup* file to the directory that the VCS simulator is run from so that the VCS simulator knows where to find the various libraries that might contain primitive references required for simulation. From that point onward, you can proceed as normal to simulate your Lattice FPGA project.

3. Common Pitfalls

This section describes the common pitfalls that you might encounter when simulating your designs with third-party simulation software.

3.1. Pitfall Example 1: Simulating a Protected Component

You may encounter the following error message or some other reference to a protected component during simulation:

*Hierarchical component lookup failed for '{*Name Protected*}' at '{*Name Protected*}'*

The following describes the error message and how to avoid or resolve the error:

- Certain IP or primitives require the instantiation of global set/reset (GSR) and power-up reset (PUR) to function correctly
- Some of the RTL for these primitives within the associated Lattice simulation library are encrypted, which prevents you from debugging the error further
- Ensure that both the GSR and PUR are instantiated in the top-level testbench for the project being simulated. For more information, refer to the GSR and PUR sections of the Lattice Radiant Software Help and Lattice Diamond Software Help.

3.2. Pitfall Example 2: Incorrect Compilation of Simulation Libraries

You may encounter the following error messages when there's an incorrect compilation of simulation libraries or incorrect library references during simulation:

- *Design instance not found*
- *Could not resolve reference to…*
- *Unresolved modules…*
- *Hierarchical component lookup failed at …*
- *… of design unit … is unresolved in …*

The following describes the error message and how to avoid or resolve the error:

- If *cmpl_libs* was used to compile the simulation libraries for the target third-party simulation tool of choice, check the *<device name>.log* file that was generated for each compiled library being used. A log file is generated for every device library that was compiled, review this log to ensure that all device libraries were created and compiled as expected
- Simulation library contents vary from device-to-device. If you compiled multiple libraries using *cmpl_libs*, ensure that you are pointing to the correct ones when invoking your simulation tool of choice. In general, you will need to point to *pmi*, *uaplatform*, as well as the device library corresponding to the device of choice for your project (e.g., *lfcpnx* for CertusPro™-NX device family).
- Ensure that all required device libraries are linked during simulation (e.g., *lfcpnx*, *pmi*, *uaplatform*). Alternatively, if you are using non-precompiled libraries, ensure that you are using an option which points to the source files for these simulation libraries so your simulation tool of choice can resolve these references.
- Depending on the IP used in a project, there may be some additional simulation-specific files which must be included for simulation. Double check the IP user guide for the IP you are simulating in your project to ensure that no files are omitted when invoking your simulation tool of choice.

If none of the steps above work, try pointing to the physical RTL, which is included with each base Lattice tool installation using options such as *incdir* or *-y*. *-Incdir* and *-y* are options that are supported by some simulation tools, which can be used to resolve missing module references by pointing directly to files or directories containing those missing references. For example, *-incdir /home/lattice/lscc/radiant/2023.1/cae_library/simulation/lfmxo5*.

3.3. Pitfall Example 3: Incorrect Timescale Settings

You may encounter the following error messages or other similar warnings when there's an incorrect definition or setting of the timescale, which may lead to functional failure during simulation:

- *Unnamed generate block….*
- *Timescale is not defined…*

To avoid or resolve this error, ensure that the correct timescale is set in all the associated testbench files in your simulation environment.

References

For more information, refer to the following resources:

- Lattice Radiant Software User Guide
- Scripting Lattice FPGA Build Flow Application Note (FPGA-AN-02073)
- Lattice Radiant Software Help
- Lattice Insights web page for Lattice Semiconductor training courses and learning plans

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

For frequently asked questions, refer to the Lattice Answer Database at www.latticesemi.com/Support/AnswerDatabase.

Revision History

Revision 1.1, June 2024

Section	Change Summary

Introduction	• Updated the Introduction section.
	• Updated the Compiling Lattice Simulation Libraries section.
	• Added the following subsections in the Compiling Lattice Simulation Libraries section:
	  • Method 1: Using the Integrated TCL Console section.
	  • Method 2: Using the Standalone TCL Console section.
	  • Method 3: Using Command Line section.

Lattice FPGA Simulation with Third-Party Tools	• Updated the Cadence Xcelium Simulator section. • Updated the following sentence in the Synopsys VCS Simulator section: *From that point onward, you can proceed as normal to simulate your Lattice FPGA project.*
References	Added a new reference—Scripting Lattice FPGA Build Flow Application Note (FPGA-AN-02073).

Revision 1.0, March 2024

Section	Change Summary
All	Initial release.

www.latticesemi.com