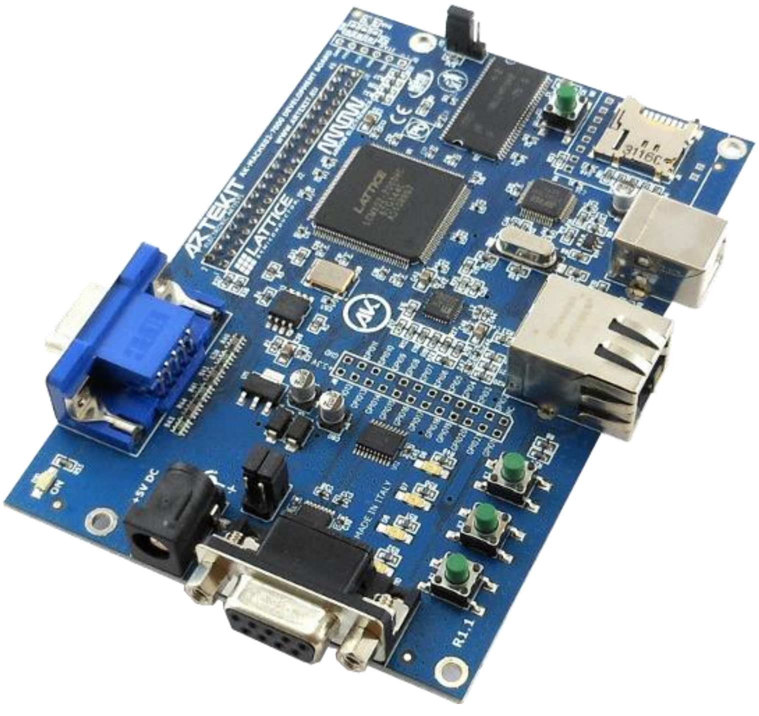
Lattice Diamond Tutorial

For the AK-MACHX02-7000



Contents

About this document

Revision history

The table below displays the revision history for the chapters in this manual.

Chapter	Date	Revision	Changes made
All	August 2013	1.0	First publication

Contact information

For the latest news, upgrades and information about Artekit products, visit the Artekit web site at http://www.artekit.eu
For technical support on this product, visit the support page at http://www.artekit.eu/contact or use the Artekit forum.
For additional information about Artekit products, consult the sources below.

Information type	Resource
Technical support	support@artekit.eu
Literature	www.artekit.eu
Sales	sales@artekit.eu
Products forum	http://www.artekit.eu/forums/

Copyright information

Lattice Diamond Tutorial

The next generation design tool for FPGA design, Lattice Diamond, is designed to address the needs of high-density FPGA designers. This tutorial leads you through all the basic steps of designing and implementing a VHDL design targeted to the MachXO2 device family and AK-MACHXO2-7000 Development Kit board. It shows you how to use several processes, tools, and reports from the Lattice Diamond software to import sources, run design analysis, view design hierarchy and inspect strategy settings.
The tutorial then proceeds to step through the processes of examining the device resources, setting timing and location assignments, and editing preferences to implement the design to the target device.

Learning Objectives

When you have completed this tutorial, you should be able to do the following:

- Create a new Lattice Diamond project
- Check Hardware Description Language (HDL)
- Verify functionality with simulation
- Examine resources
- Run synthesis process
- Set timing and location assignments
- Run place and route
- Examine post place and route results
- Download a programming file to an FPGA

Time to Complete This Tutorial

The time to complete this tutorial is about 60 minutes.

System Requirements

The following software and hardware are required to complete the tutorial:

- Lattice Diamond software
- ARTEKIT AK-MACHXO2-7000 Development Kit

(http://www.artekit.eu/products/devboards/ak-machx02-7000/)

Accessing Online Help

You can find online help information on any tool included in the tutorial at any time by choosing **Help > Lattice Diamond Help**.

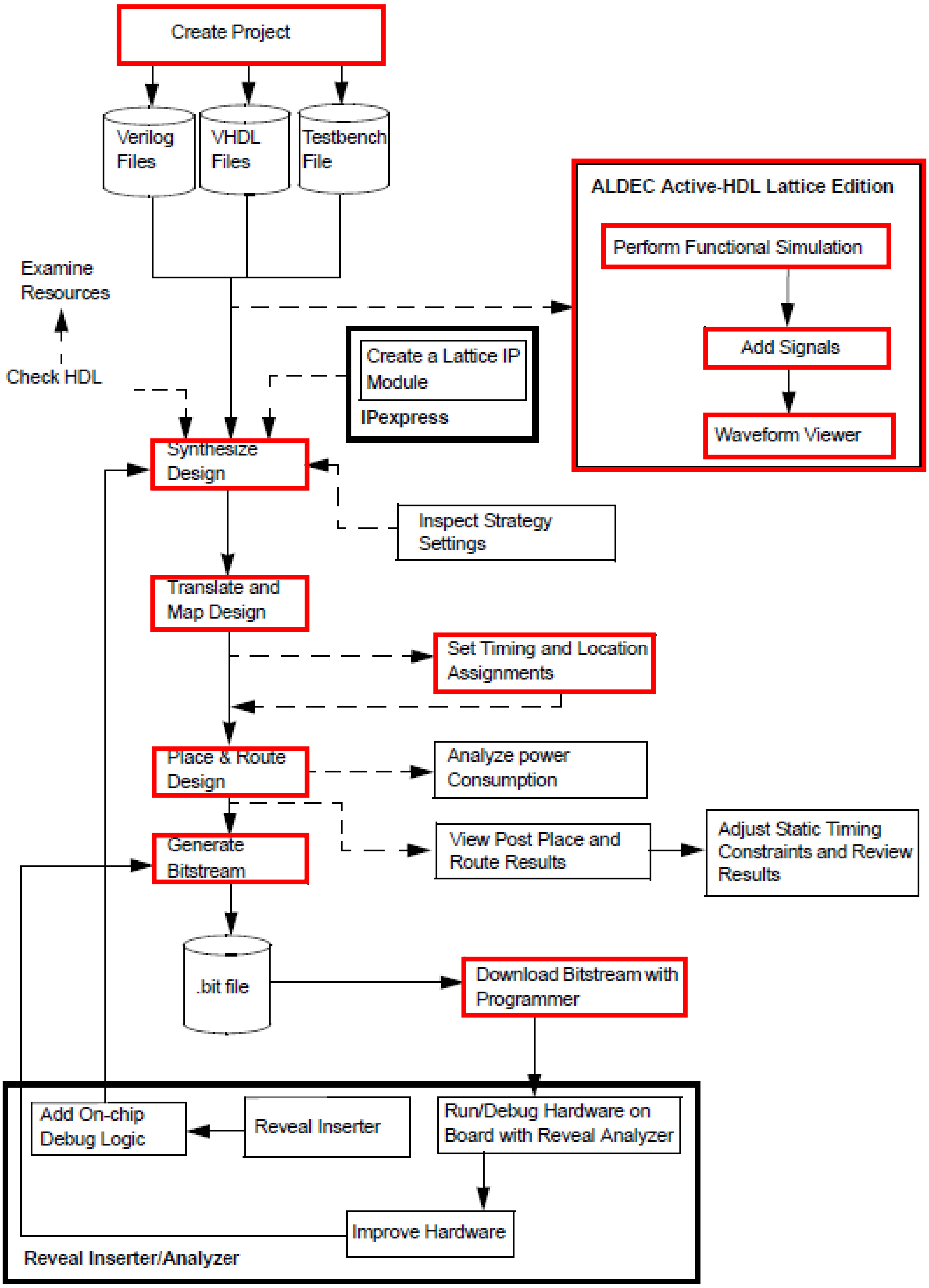
About the Tutorial Design

The design in this tutorial consists of a VHDL file. The design that you create is targeted to MachXO2 device families.

About the Tutorial Data Flow

Figure 1 illustrates the FPGA design data flow through the system. You may find it helpful to refer to this diagram as you move through the tutorial tasks highlighted in red.

Figure 1: Tutorial Data Flow


Create Project
Verilog Files
VHDL Files
Testbench File
ALDEC Active-HDL Lattice Edition
Perform Functional Simulation
Add Signals
Waveform Viewer
Examine Resources
Check HDL
Create a Lattice IP Module
IPexpress
Synthesize Design
Inspect Strategy Settings
Translate and Map Design
Set Timing and Location Assignments
Place & Route Design
Analyze power Consumption
Generate Bitstream
View Post Place and Route Results
Adjust Static Timing Constraints and Review Results
.bit file
Download Bitstream with Programmer
Add On-chip Debug Logic
Reveal Inserter
Run/Debug Hardware on Board with Reveal Analyzer
Improve Hardware
Reveal Inserter/Analyzer

Task 1: Create a New Lattice Diamond Project

Projects are used to manage input files, preferences, and optimization options related to an FPGA implementation. While there are a number of tasks you can perform independent of a project, most designs start with creating a new project.

To create a new project:

1. Do one of the following depending on your operating system:
    - From your Windows desktop, choose Start **> Programs > Lattice Diamond > Lattice Diamond**

Note

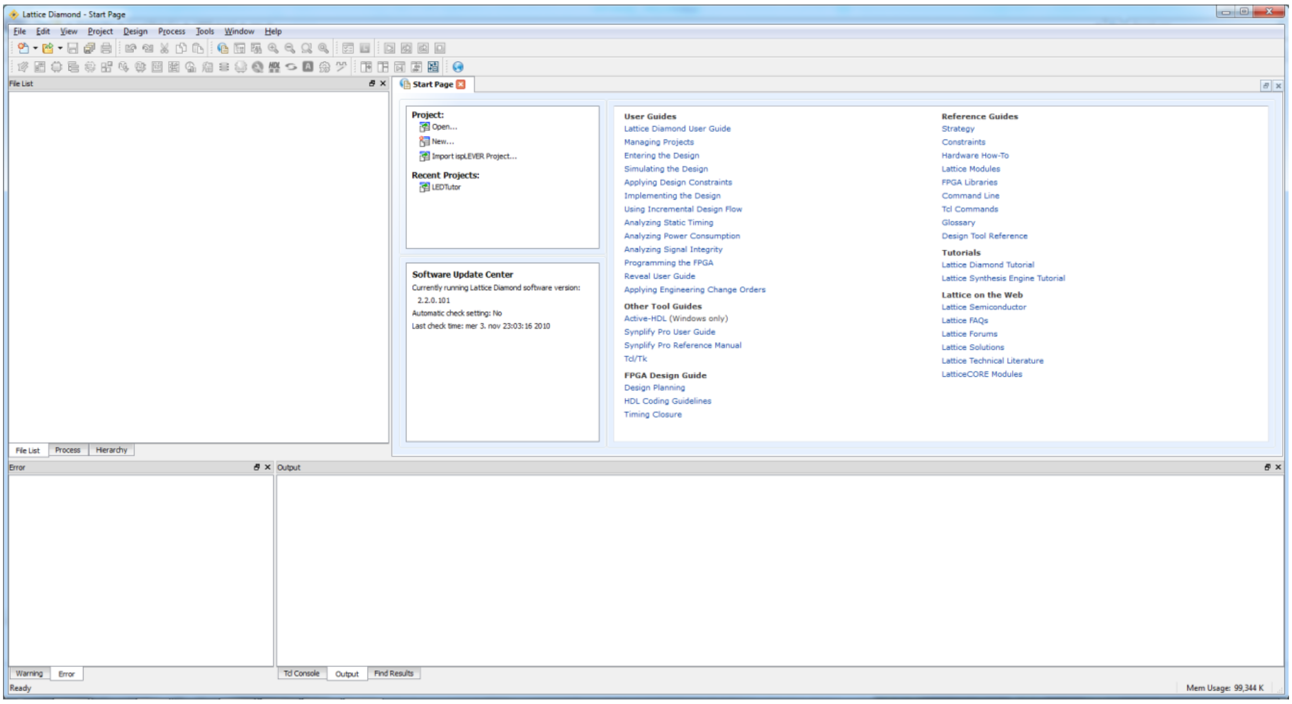
Lattice Diamond is the default Programs folder name when you install the Lattice Diamond software. Change this name accordingly if you have chosen another folder name during installation.

- From your Linux platform shell window or C-shell window, execute:

```
<install_path>/bin/lin/diamond
```

The Lattice Diamond Design Environment appears, as shown in Figure 2.

Figure 2: Diamond Design Environment



Software Update Center
Currently running Lattice Diamond software version: 2.2.0.101
Automatic check setting: No
Last check time: mer 3. nov 23:03:16 2010

Mem Usage: 99,344 K

The initial layout provides the Start Page, which provides a list of common Project actions like Open to open a pre-existing project and New to run the New Project Wizard. Hyperlinks in the right pane of the Start Page provide access to user guides, reference material, and online resources available from www.latticesemi.com.

For almost all questions, the place to start is Lattice Diamond's online Help. It describes the FPGA design flow using Diamond, the libraries of logic design elements, and the details of the Diamond design tools.
The Help also provides easy access to many other information sources. The Help can be accessed from **Help > Lattice Diamond Help**.

2. Open a new project in one of the following ways:

- In the Start page, under **Project**, click **New.**
- From the Diamond main window choose **File > New > Project**.
- Click the down arrow in the icon from the toolbar and then choose **Project**.

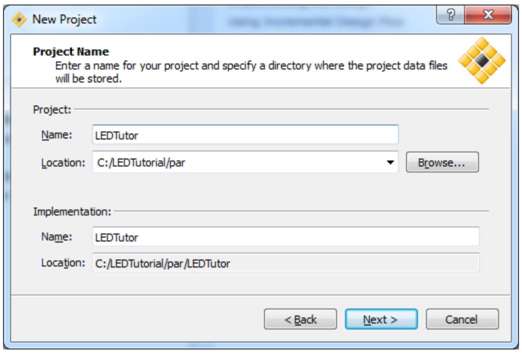
Click **Next**. The New Project dialog box of the Project Wizard opens.

3. Specify Project name: **LEDTutorial**

Note
File names for Diamond projects and project source files must start with a letter (A-Z, a-z) and must contain only alphanumeric characters (A-Z, a-z, 0-9) and underscores (_).

4. Click **Browse**. In the Project Location dialog box, browse to <*path_LEDTutorial*>/par where we are going to store the project's files. Click **Select Folder.**
By default, when you specify the project name, the implementation name is simultaneously specified the same, as shown in Figure 3. For this tutorial, leave the default implementation name as **LEDTutorial**. The directory to store the implementation is automatically displayed in the Location area.

Figure 3: New Project, Project Name Window



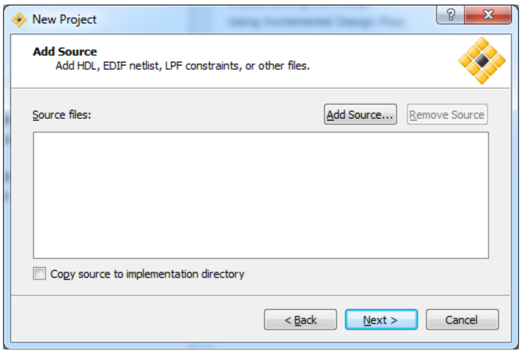
Project Name
Enter a name for your project and specify a directory where the project data files will be stored.

Project:
Name: LEDTutor
Location: C:/LEDTutorial/par

Implementation:
Name: LEDTutor
Location: C:/LEDTutorial/par/LEDTutor

5. Click **Next**. The Add Source dialog box appears as shown in next figure.

Figure 4: New Project, Add Source Window



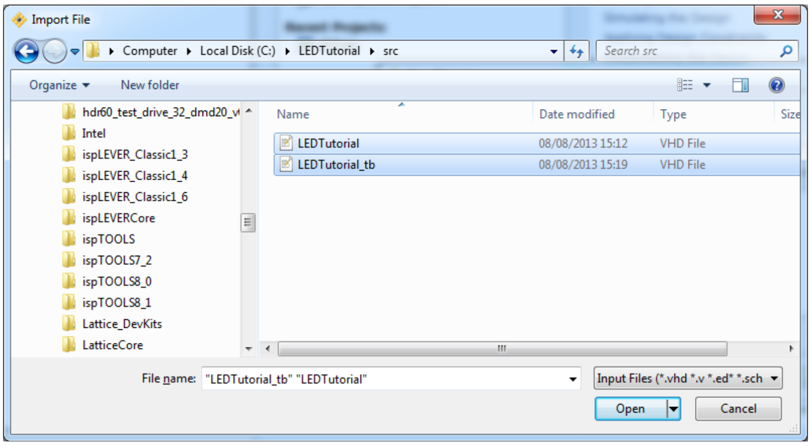
Add Source
Add HDL, EDIF netlist, LPF constraints, or other files.

☐ Copy source to implementation directory

6. Click **Add Source**. The **Import File** dialog box appears.

7. Navigate to the folder containing the source files, which are located in the <*path_LEDTutorial*>/src directory (i.e. C:/LEDTutorial/src).
Select as reported in figure 5 LEDTutorial.vhd and LEDTutorial_tb.vhd in the directory and click **Open**

Figure 5: New Project, Import File Window



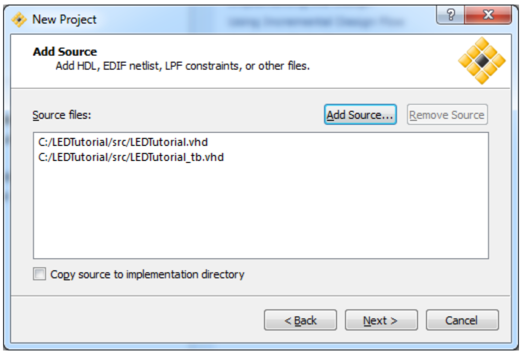
Name	Date modified	Type
LEDTutorial	08/08/2013 15:12	VHD File
LEDTutorial_tb	08/08/2013 15:19	VHD File

File name: "LEDTutorial_tb" "LEDTutorial"

Input Files (*.vhd *.v *.ed* *.sch

The Add Source step of the Wizard appears with the selected source file added.
8. Leave unchecked **Copy source to implementation directory,** as reported in figure **6.**
9. and click **Next**.

Figure 6: New Project, Add Source Window



Source files:

C:/LEDTutorial/src/LEDTutorial.vhd
C:/LEDTutorial/src/LEDTutorial_tb.vhd

☐ Copy source to implementation directory

10. In Add Source window click **Next**. The Device Selector dialog box appears.

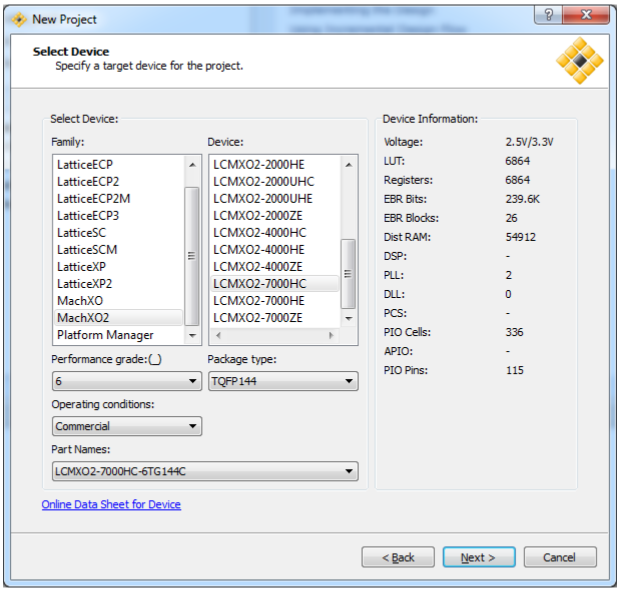
11. Select the following device options:

- Family: **MachXO2**
- Device: **LCMXO2-7000HC**
- Performance Grade: **6**
- Package type: **TQFP144**
- Operating Conditions: **Commercial**

Part Names: **LCMXO2-7000HC-6TG144C**

The dialog box should resemble Figure 7.

Figure 7: New Project Wizard Device Selector Dialog Box



Performance grade: 6

Package type: TQFP144

Operating conditions: Commercial

Part Names: LCMXO2-7000HC-6TG144C

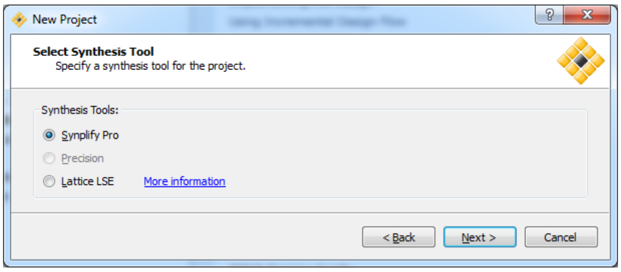
Device Information:

Voltage:	2.5V/3.3V
LUT:	6864
Registers:	6864
EBR Bits:	239.6K
EBR Blocks:	26
Dist RAM:	54912
DSP:	-
PLL:	2
DLL:	0
PCS:	-
PIO Cells:	336
APIO:	-
PIO Pins:	115

12. Click **Next.**

The Select Synthesis Tool dialog box opens: choose **Synplify Pro** as in figure 8.

Figure 8: New Project, Select Synthesis Tool Window

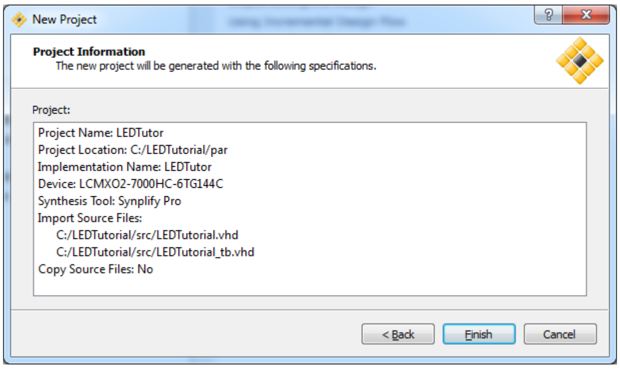


● Synplify Pro
○ Precision
○ Lattice LSE

13. Click **Next.**

The Project Information dialog box appears as in figure 9. The project information includes project name, location, implementation name, device, synthesis tool, and import source.

Figure 9: New Project, Project Information Window



New Project
Project Information
The new project will be generated with the following specifications.
< Back
Finish
Cancel

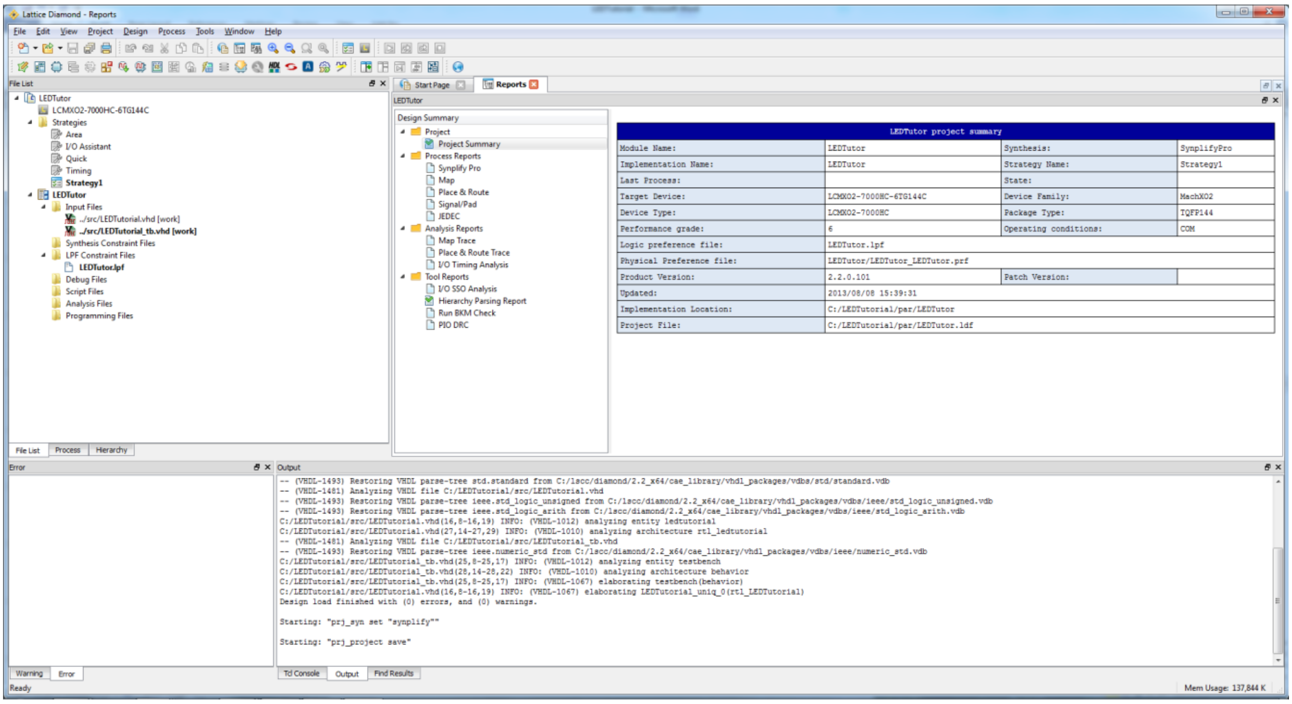
Project:

Project Name: LEDTutor
Project Location: C:/LEDTutorial/par
Implementation Name: LEDTutor
Device: LCMXO2-7000HC-6TG144C
Synthesis Tool: Synplify Pro
Import Source Files:
C:/LEDTutorial/src/LEDTutorial.vhd
C:/LEDTutorial/src/LEDTutorial_tb.vhd
Copy Source Files: No

14. Click **Finish**.

The File List and Process views are populated and the Reports view appears as reported in figure 10. The File List view displays the components of the project. The imported VHDL file appears in the Input Files folder in the File List view. The File List view organizes project files by categories: Strategies, and Implementation including Input Files, Constraint Files, Debug Files, Script Files, and Analysis Files. You may adjust file order by dragging and dropping of the filenames in the list. Properties of each file are accessed by highlighting a file, clicking the right mouse button, and selecting Properties from the pop-up menu.

Figure 10: File List and Reports view



LEDTutor project summary			
Module Name:	LEDTutor	Synthesis:	SynplifyPro
Implementation Name:	LEDTutor	Strategy Name:	Strategy1
Last Process:		State:	
Target Device:	LCMXO2-7000HC-6TG144C	Device Family:	MachXO2
Device Type:	LCMXO2-7000HC	Package Type:	TQFP144
Performance grade:	6	Operating conditions:	COM
Logic preference file:	LEDTutor.lpf		
Physical Preference file:	LEDTutor/LEDTutor_LEDTutor.prf		
Product Version:	2.2.0.101	Patch Version:	
Updated:	2013/08/08 15:39:31		
Implementation Location:	C:/LEDTutorial/par/LEDTutor		
Project File:	C:/LEDTutorial/par/LEDTutor.ldf		

```
-- (VHDL-1493) Restoring VHDL parse-tree std.standard from C:/lscc/diamond/2.2_x64/cae_library/vhdl_packages/vdbs/std/standard.vdb
-- (VHDL-1481) Analyzing VHDL file C:/LEDTutorial/src/LEDTutorial.vhd
-- (VHDL-1493) Restoring VHDL parse-tree ieee.std_logic_unsigned from C:/lscc/diamond/2.2_x64/cae_library/vhdl_packages/vdbs/ieee/std_logic_unsigned.vdb
-- (VHDL-1493) Restoring VHDL parse-tree ieee.std_logic_arith from C:/lscc/diamond/2.2_x64/cae_library/vhdl_packages/vdbs/ieee/std_logic_arith.vdb
C:/LEDTutorial/src/LEDTutorial.vhd(16,8-16,19) INFO: (VHDL-1012) analyzing entity ledtutorial
C:/LEDTutorial/src/LEDTutorial.vhd(27,14-27,29) INFO: (VHDL-1010) analyzing architecture rtl_ledtutorial
-- (VHDL-1481) Analyzing VHDL file C:/LEDTutorial/src/LEDTutorial_tb.vhd
-- (VHDL-1493) Restoring VHDL parse-tree ieee.numeric_std from C:/lscc/diamond/2.2_x64/cae_library/vhdl_packages/vdbs/ieee/numeric_std.vdb
C:/LEDTutorial/src/LEDTutorial_tb.vhd(25,8-25,17) INFO: (VHDL-1012) analyzing entity testbench
C:/LEDTutorial/src/LEDTutorial_tb.vhd(28,14-28,22) INFO: (VHDL-1010) analyzing architecture behavior
C:/LEDTutorial/src/LEDTutorial_tb.vhd(25,8-25,17) INFO: (VHDL-1067) elaborating testbench(behavior)
C:/LEDTutorial/src/LEDTutorial.vhd(16,8-16,19) INFO: (VHDL-1067) elaborating LEDTutorial_uniq_0(rtl_LEDTutorial)
Design load finished with (0) errors, and (0) warnings.

Starting: "prj_syn set "synplify""

Starting: "prj_project save"
```

Note

You can also see Area, I/O Assistant, Quick, and Timing listed in the Strategies folder in the File List view. These are predefined strategies supplied by Lattice Semiconductor. They are designed to solve particular types of design. For details of these predefined strategies, refer to the Diamond online Help.

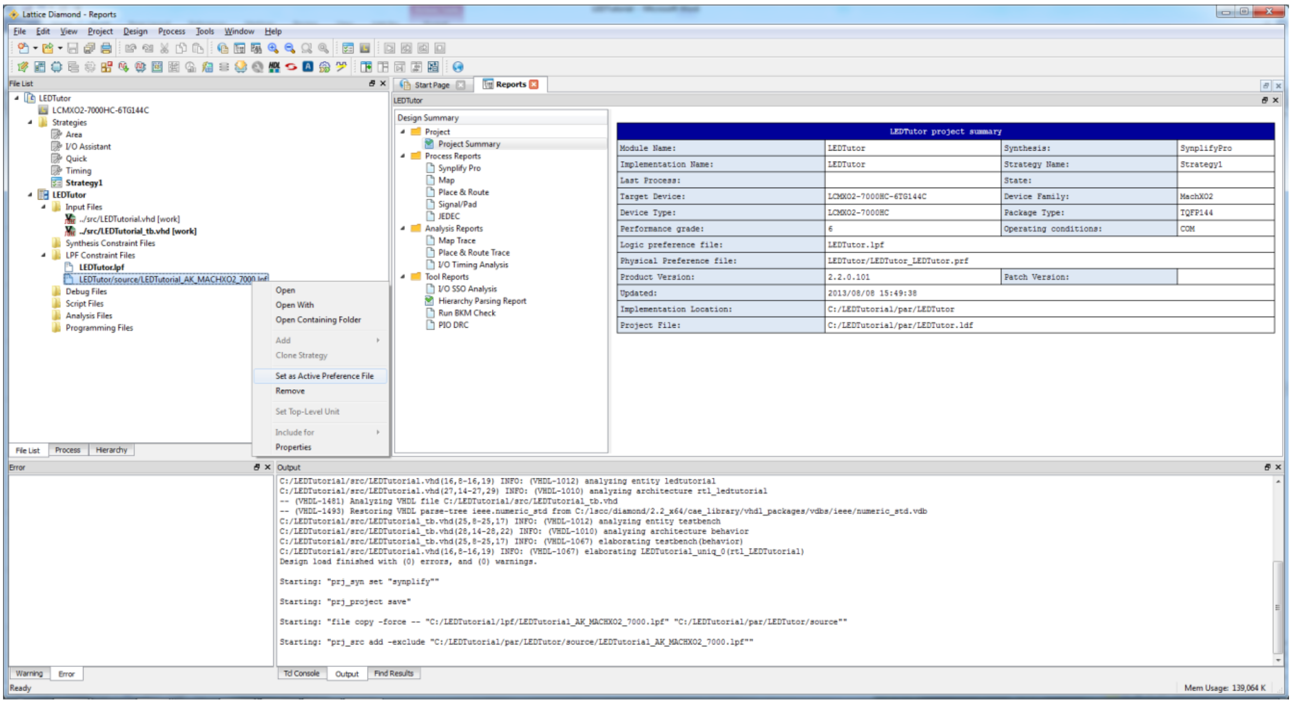
When you create a new project in Diamond, a logical preference file (.lpf) is automatically generated and assigned the same name as the FPGA project, LEDTutorial.lpf in this case. For this tutorial a logical preference file named **LEDTutorial_AK_MACHXO2_7000.lpf** is provided in <*path_LEDTutorial*>/lpf directory (i.e. C:/LEDTutorial/lpf) and contains all the pin assignments needed to program this design project onto the MachXO2 FPGA. All changes that you make to logical constraints will be saved in this file until you create a new logical preference file or add another existing one.

15. In the File List, right-click on **LPF Constraint file**, and select **Add > Existing File**. Add Existing File dialog box appears.

16. Navigate to <*path_LEDTutorial*>/lpf and select the file **LEDTutorial_AK_MACHXO2_7000.lpf**. Choose **Copy file to Implementation's Source directory**, and click **Add**.

17. In the File List, right-click on LEDTutorial_AK_MACHXO2_7000.lpf and choose **Set as Active Preference File**. See figure 11 regarding this step.

Figure 11: File List and Reports view, Setting Active Preference File

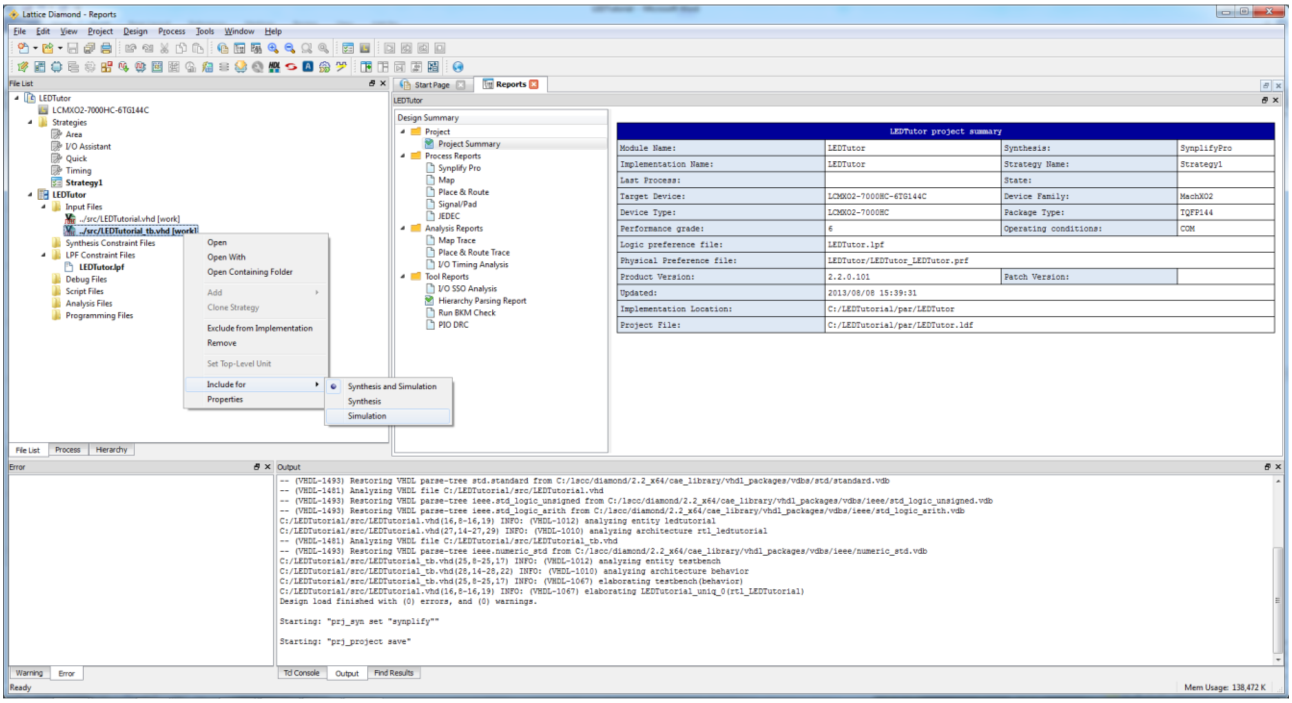


LEDTutor project summary			
Module Name:	LEDTutor	Synthesis:	SynplifyPro
Implementation Name:	LEDTutor	Strategy Name:	Strategy1
Last Process:		State:	
Target Device:	LCMXO2-7000HC-6TG144C	Device Family:	MachXO2
Device Type:	LCMXO2-7000HC	Package Type:	TQFP144
Performance grade:	6	Operating conditions:	COM
Logic preference file:	LEDTutor.lpf		
Physical Preference file:	LEDTutor/LEDTutor_LEDTutor.prf		
Product Version:	2.2.0.101	Patch Version:	
Updated:	2013/08/08 15:49:38		
Implementation Location:	C:/LEDTutorial/par/LEDTutor		
Project File:	C:/LEDTutorial/par/LEDTutor.ldf		

```
C:/LEDTutorial/src/LEDTutorial.vhd(16,8-16,19) INFO: (VHDL-1012) analyzing entity ledtutorial
C:/LEDTutorial/src/LEDTutorial.vhd(27,14-27,29) INFO: (VHDL-1010) analyzing architecture rtl_ledtutorial
-- (VHDL-1481) Analyzing VHDL file C:/LEDTutorial/src/LEDTutorial_tb.vhd
-- (VHDL-1493) Restoring VHDL parse-tree ieee.numeric_std from C:/lscc/diamond/2.2_x64/cae_library/vhdl_packages/vdbs/ieee/numeric_std.vdb
C:/LEDTutorial/src/LEDTutorial_tb.vhd(25,8-25,17) INFO: (VHDL-1012) analyzing entity testbench
C:/LEDTutorial/src/LEDTutorial_tb.vhd(28,14-28,22) INFO: (VHDL-1010) analyzing architecture behavior
C:/LEDTutorial/src/LEDTutorial_tb.vhd(25,8-25,17) INFO: (VHDL-1067) elaborating testbench(behavior)
C:/LEDTutorial/src/LEDTutorial.vhd(16,8-16,19) INFO: (VHDL-1067) elaborating LEDTutorial_uniq_0(rtl_LEDTutorial)
Design load finished with (0) errors, and (0) warnings.

Starting: "prj_syn set "synplify""

Starting: "prj_project save"

Starting: "file copy -force -- "C:/LEDTutorial/lpf/LEDTutorial_AK_MACHXO2_7000.lpf" "C:/LEDTutorial/par/LEDTutor/source""

Starting: "prj_src add -exclude "C:/LEDTutorial/par/LEDTutor/source/LEDTutorial_AK_MACHXO2_7000.lpf""
```

18. As reported in figure 12, in the File List, right-click on LEDTutorial_tb.vhd and choose **Include for > Simulation.**

Figure 12: File List and Reports view, Include Input file for Simulation

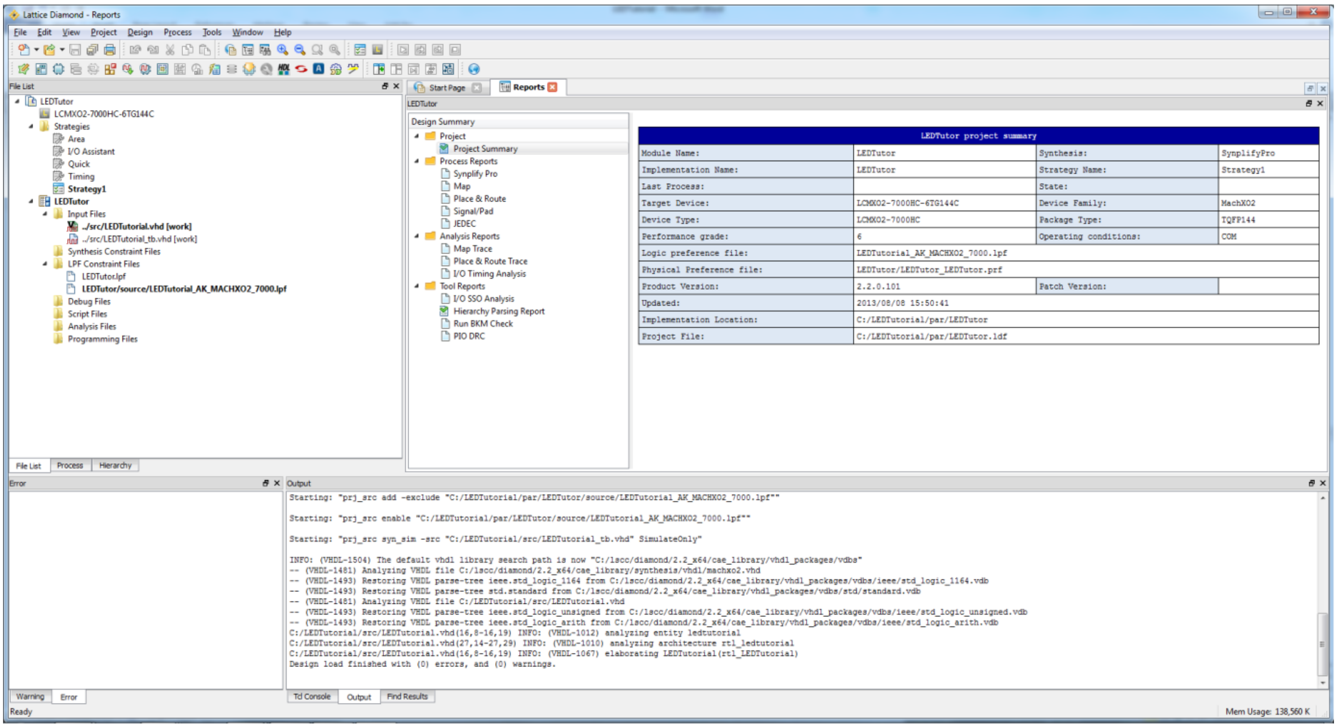


LEDTutor project summary			
Module Name:	LEDTutor	Synthesis:	SynplifyPro
Implementation Name:	LEDTutor	Strategy Name:	Strategy1
Last Process:		State:	
Target Device:	LCMXO2-7000HC-6TG144C	Device Family:	MachXO2
Device Type:	LCMXO2-7000HC	Package Type:	TQFP144
Performance grade:	6	Operating conditions:	COM
Logic preference file:	LEDTutor.lpf		
Physical Preference file:	LEDTutor/LEDTutor_LEDTutor.prf		
Product Version:	2.2.0.101	Patch Version:	
Updated:	2013/08/08 15:39:31		
Implementation Location:	C:/LEDTutorial/par/LEDTutor		
Project File:	C:/LEDTutorial/par/LEDTutor.ldf		

```
-- (VHDL-1493) Restoring VHDL parse-tree std.standard from C:/lscc/diamond/2.2_x64/cae_library/vhdl_packages/vdbs/std/standard.vdb
-- (VHDL-1481) Analyzing VHDL file C:/LEDTutorial/src/LEDTutorial.vhd
-- (VHDL-1493) Restoring VHDL parse-tree ieee.std_logic_unsigned from C:/lscc/diamond/2.2_x64/cae_library/vhdl_packages/vdbs/ieee/std_logic_unsigned.vdb
-- (VHDL-1493) Restoring VHDL parse-tree ieee.std_logic_arith from C:/lscc/diamond/2.2_x64/cae_library/vhdl_packages/vdbs/ieee/std_logic_arith.vdb
C:/LEDTutorial/src/LEDTutorial.vhd(16,8-16,19) INFO: (VHDL-1012) analyzing entity ledtutorial
C:/LEDTutorial/src/LEDTutorial.vhd(27,14-27,29) INFO: (VHDL-1010) analyzing architecture rtl_ledtutorial
-- (VHDL-1481) Analyzing VHDL file C:/LEDTutorial/src/LEDTutorial_tb.vhd
-- (VHDL-1493) Restoring VHDL parse-tree ieee.numeric_std from C:/lscc/diamond/2.2_x64/cae_library/vhdl_packages/vdbs/ieee/numeric_std.vdb
C:/LEDTutorial/src/LEDTutorial_tb.vhd(25,8-25,17) INFO: (VHDL-1012) analyzing entity testbench
C:/LEDTutorial/src/LEDTutorial_tb.vhd(28,14-28,22) INFO: (VHDL-1010) analyzing architecture behavior
C:/LEDTutorial/src/LEDTutorial_tb.vhd(25,8-25,17) INFO: (VHDL-1067) elaborating testbench(behavior)
C:/LEDTutorial/src/LEDTutorial.vhd(16,8-16,19) INFO: (VHDL-1067) elaborating LEDTutorial_uniq_0(rtl_LEDTutorial)
Design load finished with (0) errors, and (0) warnings.

Starting: "prj_syn set "synplify""

Starting: "prj_project save"
```

The Process view, shown in Figure 13, lists all the processes available, such as Synthesize Design, Translate Design, Map Design, Place & Route Design, and Export Files. The Reports view provides a way to examine and print process reports. The Reports view displays reports for the major processes. There are two panes in the Reports view. The left pane lists the reports. The right pane displays the reports. Log messages are displayed in the Output frame of the Diamond main window.

Figure 13: File View and Reports View, Project Summary



LEDTutor project summary			
Module Name:	LEDTutor	Synthesis:	SynplifyPro
Implementation Name:	LEDTutor	Strategy Name:	Strategy1
Last Process:		State:	
Target Device:	LCMXO2-7000HC-6TG144C	Device Family:	MachXO2
Device Type:	LCMXO2-7000HC	Package Type:	TQFP144
Performance grade:	6	Operating conditions:	COM
Logic preference file:	LEDTutorial_AK_MACHXO2_7000.lpf		
Physical Preference file:	LEDTutor/LEDTutor_LEDTutor.prf		
Product Version:	2.2.0.101	Patch Version:	
Updated:	2013/08/08 15:50:41		
Implementation Location:	C:/LEDTutorial/par/LEDTutor		
Project File:	C:/LEDTutorial/par/LEDTutor.ldf		

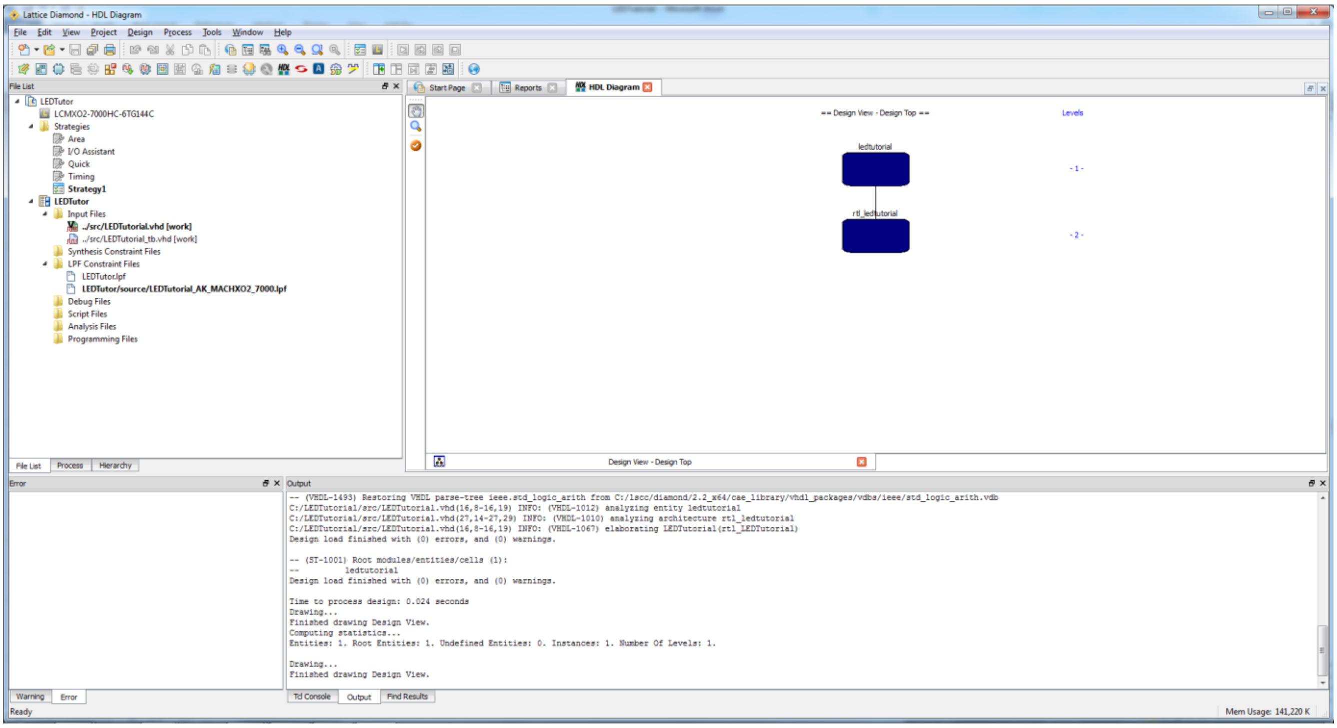
```
Starting: "prj_src add -exclude "C:/LEDTutorial/par/LEDTutor/source/LEDTutorial_AK_MACHXO2_7000.lpf""

Starting: "prj_src enable "C:/LEDTutorial/par/LEDTutor/source/LEDTutorial_AK_MACHXO2_7000.lpf""

Starting: "prj_src syn_sim -src "C:/LEDTutorial/src/LEDTutorial_tb.vhd" SimulateOnly"

INFO: (VHDL-1504) The default vhdl library search path is now "C:/lscc/diamond/2.2_x64/cae_library/vhdl_packages/vdbs"
-- (VHDL-1481) Analyzing VHDL file C:/lscc/diamond/2.2_x64/cae_library/synthesis/vhdl/machxo2.vhd
-- (VHDL-1493) Restoring VHDL parse-tree ieee.std_logic_1164 from C:/lscc/diamond/2.2_x64/cae_library/vhdl_packages/vdbs/ieee/std_logic_1164.vdb
-- (VHDL-1493) Restoring VHDL parse-tree std.standard from C:/lscc/diamond/2.2_x64/cae_library/vhdl_packages/vdbs/std/standard.vdb
-- (VHDL-1481) Analyzing VHDL file C:/LEDTutorial/src/LEDTutorial.vhd
-- (VHDL-1493) Restoring VHDL parse-tree ieee.std_logic_unsigned from C:/lscc/diamond/2.2_x64/cae_library/vhdl_packages/vdbs/ieee/std_logic_unsigned.vdb
-- (VHDL-1493) Restoring VHDL parse-tree ieee.std_logic_arith from C:/lscc/diamond/2.2_x64/cae_library/vhdl_packages/vdbs/ieee/std_logic_arith.vdb
C:/LEDTutorial/src/LEDTutorial.vhd(16,8-16,19) INFO: (VHDL-1012) analyzing entity ledtutorial
C:/LEDTutorial/src/LEDTutorial.vhd(27,14-27,29) INFO: (VHDL-1010) analyzing architecture rtl_ledtutorial
C:/LEDTutorial/src/LEDTutorial.vhd(16,8-16,19) INFO: (VHDL-1067) elaborating LEDTutorial(rtl_LEDTutorial)
Design load finished with (0) errors, and (0) warnings.
```

Task 2: Check Hardware Description Language

Diamond provides hardware description language (HDL) visualization and rule-checks to detect coding style violations that may lead to pre-/post-synthesis simulation mismatches. The Hierarchy view mirrors the portion of the design hierarchy that is displayed in the active graphical views.
To analyze and view the HDL design:

1. Choose T**ools > HDL Diagram**. The Hierarchy view appears as shown in Figure 14.

Figure 14: Hierarchy View



```
-- (VHDL-1493) Restoring VHDL parse-tree ieee.std_logic_arith from C:/lscc/diamond/2.2_x64/cae_library/vhdl_packages/vdbs/ieee/std_logic_arith.vdb
C:/LEDTutorial/src/LEDTutorial.vhd(16,8-16,19) INFO: (VHDL-1012) analyzing entity ledtutorial
C:/LEDTutorial/src/LEDTutorial.vhd(27,14-27,29) INFO: (VHDL-1010) analyzing architecture rtl_ledtutorial
C:/LEDTutorial/src/LEDTutorial.vhd(16,8-16,19) INFO: (VHDL-1067) elaborating LEDTutorial(rtl_LEDTutorial)
Design load finished with (0) errors, and (0) warnings.

-- (ST-1001) Root modules/entities/cells (1):
--          ledtutorial
Design load finished with (0) errors, and (0) warnings.

Time to process design: 0.024 seconds
Drawing...
Finished drawing Design View.
Computing statistics...
Entities: 1. Root Entities: 1. Undefined Entities: 0. Instances: 1. Number Of Levels: 1.

Drawing...
Finished drawing Design View.
```

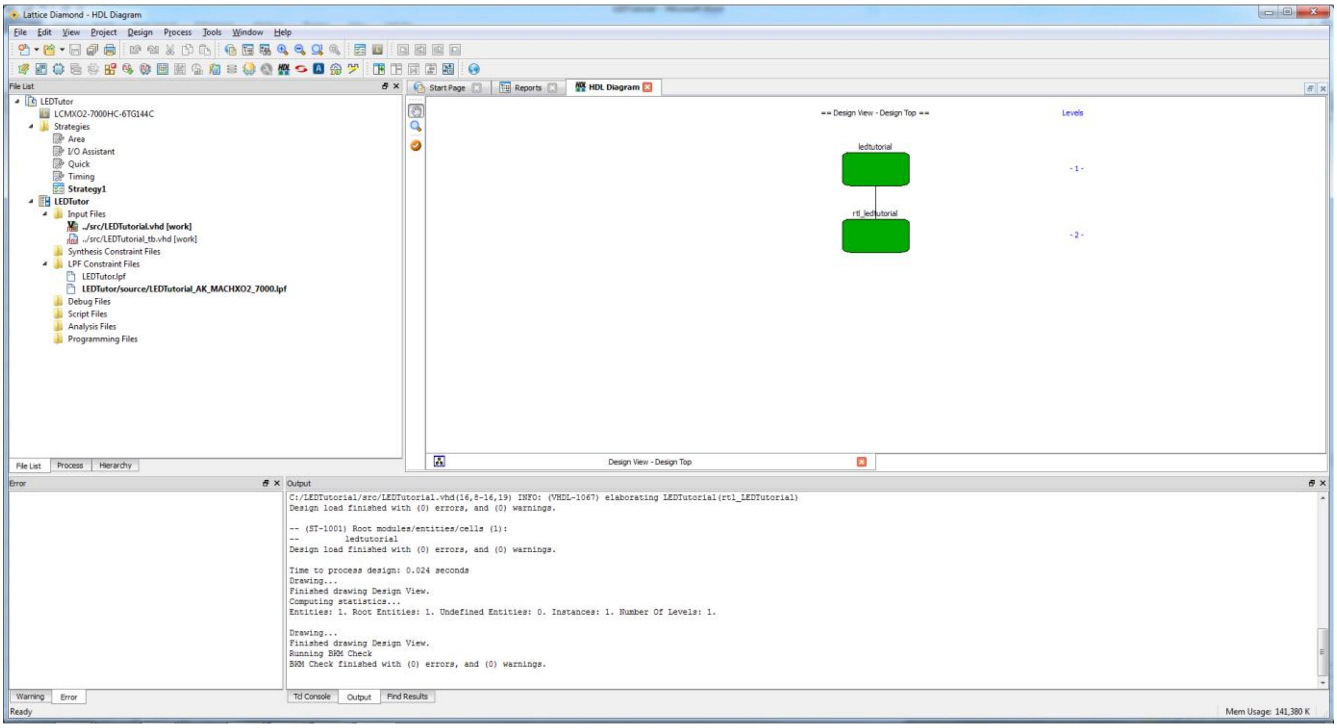
3. Choose Design > Run BKM Check. Best Known Methods (BKM) analysis is run.

The HDL Diagram with the results of the BKM analysis appears, as shown in Figure 15. Results shown in output log and HDL diagram and Hierarchy view are color coded and based on results from the BKM analysis. Best Known Methods (BKM) are design guidelines that HDL Diagram uses to analyze your design. BKM checks include the following:

- Connectivity – Checks the pin connectivity of instances throughout the design.
- Synthesis – Checks for violations of the Sunburst Design coding styles, as well as other potential synthesis problems.
- Structural Fan-Out – Checks for maximum structural fan-out violations.
- Coding Styles – Colors modules based on their line count, colors pins and ports based on their width, validates module names, and also performs big-endian or little-endian checks on all ports.
- Verification – Validates the existence and timestamps of VCD files. A series of Lint-like RTL rule checks are run. Modules that have rule violations are color coded in the HDL Diagram view.

The checks performed during a BKM run can be customized in the **Options** dialog box (**Tools > Options** from the Diamond main window) HDL Diagram section. It is a good practice to run RTL analysis before synthesis to detect coding style that could lead to mismatches between pre-synthesis and post-synthesis simulation results. The analysis views are also excellent documentation output for your design.

Figure 15: HDL Diagram

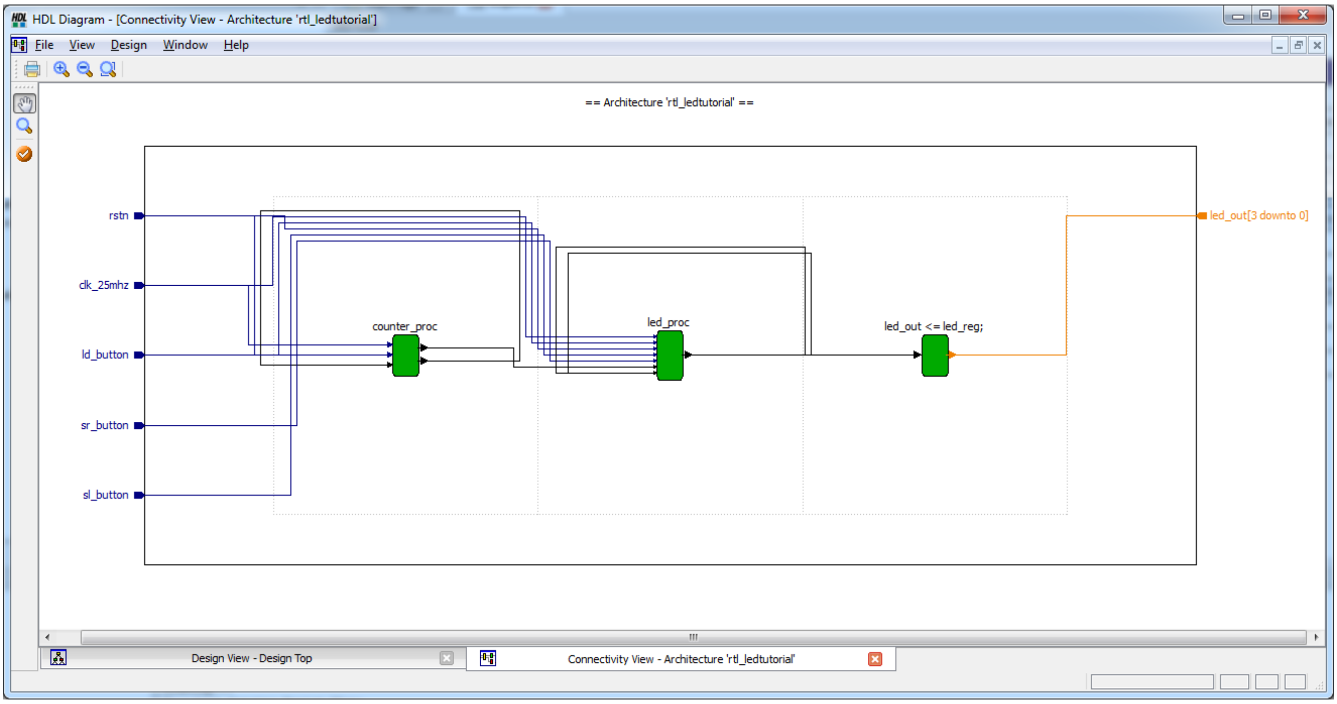


```
C:/LEDTutorial/src/LEDTutorial.vhd(16,8-16,19) INFO: (VHDL-1067) elaborating LEDTutorial(rtl_LEDTutorial)
Design load finished with (0) errors, and (0) warnings.

-- (ST-1001) Root modules/entities/cells (1):
--          ledtutorial
Design load finished with (0) errors, and (0) warnings.

Time to process design: 0.024 seconds
Drawing...
Finished drawing Design View.
Computing statistics...
Entities: 1. Root Entities: 1. Undefined Entities: 0. Instances: 1. Number Of Levels: 1.

Drawing...
Finished drawing Design View.
Running BKM Check
BKM Check finished with (0) errors, and (0) warnings.
```

4. After running the BKM check, you may encounter warning and error messages.

Error and warning messages are displayed in the Output, Warning, and Error frames. If you double-click the message in the Output, Error, or Warning frames, the source indicated in the message will be opened in the associated editor. This cross-probing function can ease your check of the source file. Double-click the rtl_ledtutorial block from the HDL Diagram Design view, or right-click the rtl_ledtutorial block and choose View Connectivity. The Connectivity view, shown in Figure 16, appears as a new tab. You can click the Detach Tool icon on the upper right corner the HDL Diagram to make it a separate window. The Connectivity view shows signal flow between module ports, internal instances and the behavioral blocks within a particular instance or module. It enables you to explore the signal connectivity — signals and bundles between instances and behavioral blocks within the current module.

Figure 16: Connectivity View


HDL Diagram - [Connectivity View - Architecture 'rtl_ledtutorial']
File
View
Design
Window
Help
== Architecture 'rtl_ledtutorial' ==
rstn
clk_25mhz
ld_button
sr_button
sl_button
counter_proc
led_proc
led_out <= led_reg;
led_out[3 downto 0]
Design View - Design Top
Connectivity View - Architecture 'rtl_ledtutorial'

5. When you finish checking the signal connectivity, you can choose **Window > Attach Window** from the separated HDL Diagram to attach it back to the Diamond main window.

Task 3: Verify Functionality with Simulation

Diamond provides an interface to create a new simulation project file that can be imported into a standalone simulator. Diamond supports Active-HDL and ModelSim® simulation file for file exports.
Aldec® Active-HDL™ is an integrated environment designed for simulation of VHDL, Verilog/SystemVerilog, EDIF, and SystemC designs.
In this task, you will simulate the design using Active-HDL and analyze the resulting waveforms.

To simulate the design:

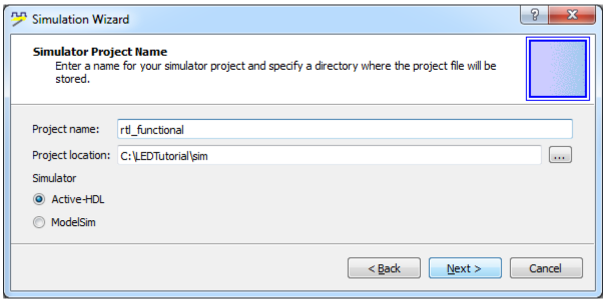
1. Select **Tool** > **Simulation Wizard**. The Simulation Wizard dialog box appears.

2. Click **Next.**

The Simulator Project Name dialog box appears.

3. Perform the following as shown in figure 17:
- Specify Project name: **rtl_functional**.
- Make sure Active-HDL is selected for Simulator.
- Click the button to browse to <*path_LEDTutorial*>/sim where to store the project's file

Figure 17: Simulation Project Name



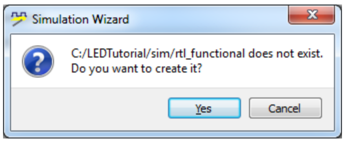
Simulator Project Name
Enter a name for your simulator project and specify a directory where the project file will be stored.

Project name: rtl_functional
Project location: C:\LEDTutorial\sim
Simulator
● Active-HDL
○ ModelSim

4. Click **Next**.

5. As reported in figure 18, click **Yes** to create a new folder

Figure 18: Simulation Wizard, New Folder

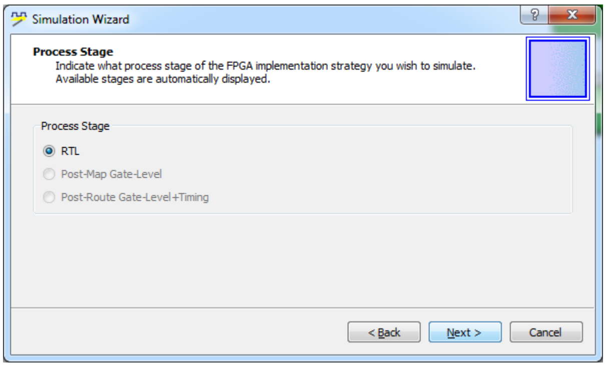


C:/LEDTutorial/sim/rtl_functional does not exist.
Do you want to create it?

The Process Stage dialog box of figure 19 appears.

6. Select **RTL** in the Process Stage box. Click **Next**.

Figure 19: Simulation Wizard, Process Stage



Process Stage
Indicate what process stage of the FPGA implementation strategy you wish to simulate. Available stages are automatically displayed.

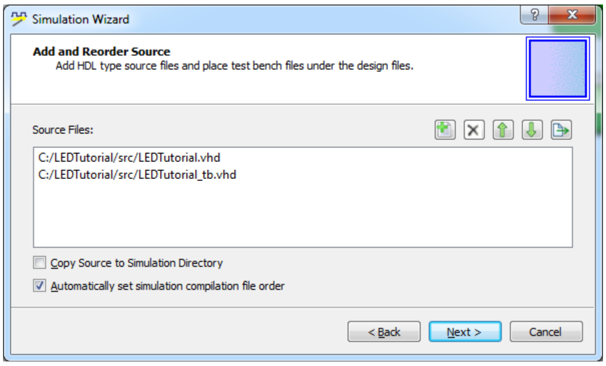
Process Stage
● RTL
○ Post-Map Gate-Level
○ Post-Route Gate-Level+Timing

The Add and Reorder Source dialog box appears as in figure 20.

7. Make sure all source files are present in the Source Files list.

8. Click **Next**.

Figure 20: Simulation Project Name



Add and Reorder Source
Add HDL type source files and place test bench files under the design files.

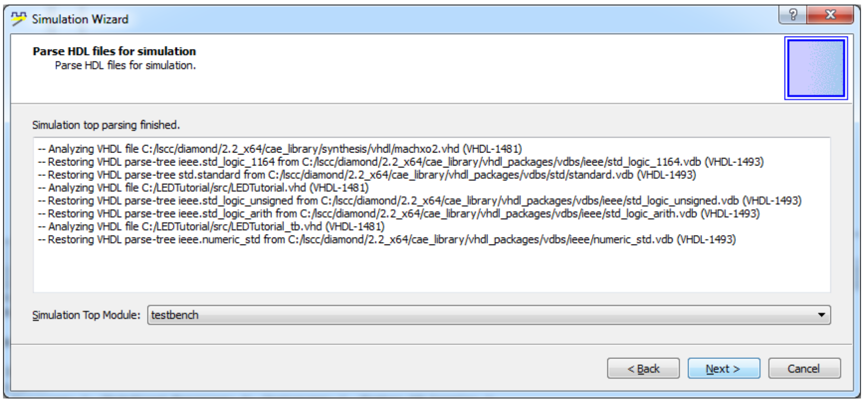
Source Files:

C:/LEDTutorial/src/LEDTutorial.vhd
C:/LEDTutorial/src/LEDTutorial_tb.vhd

☐ Copy Source to Simulation Directory
☑ Automatically set simulation compilation file order

The Parse HDL Files for Simulation dialog box of figure 21 appears.

Figure 21: Simulation Project Name



Parse HDL files for simulation
Parse HDL files for simulation.

Simulation top parsing finished.

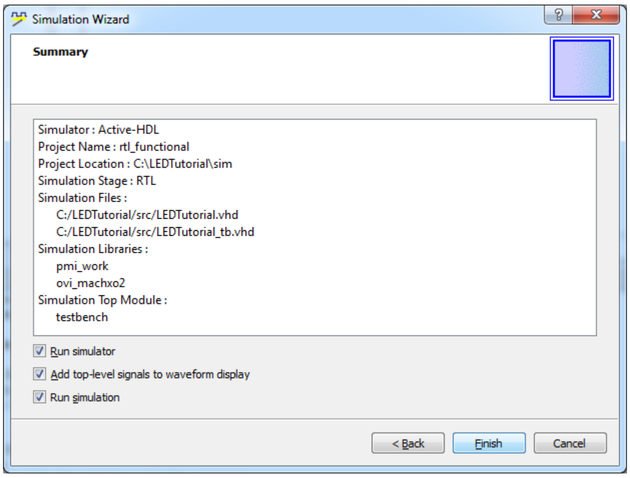
```
-- Analyzing VHDL file C:/lscc/diamond/2.2_x64/cae_library/synthesis/vhdl/machxo2.vhd (VHDL-1481)
-- Restoring VHDL parse-tree ieee.std_logic_1164 from C:/lscc/diamond/2.2_x64/cae_library/vhdl_packages/vdbs/ieee/std_logic_1164.vdb (VHDL-1493)
-- Restoring VHDL parse-tree std.standard from C:/lscc/diamond/2.2_x64/cae_library/vhdl_packages/vdbs/std/standard.vdb (VHDL-1493)
-- Analyzing VHDL file C:/LEDTutorial/src/LEDTutorial.vhd (VHDL-1481)
-- Restoring VHDL parse-tree ieee.std_logic_unsigned from C:/lscc/diamond/2.2_x64/cae_library/vhdl_packages/vdbs/ieee/std_logic_unsigned.vdb (VHDL-1493)
-- Restoring VHDL parse-tree ieee.std_logic_arith from C:/lscc/diamond/2.2_x64/cae_library/vhdl_packages/vdbs/ieee/std_logic_arith.vdb (VHDL-1493)
-- Analyzing VHDL file C:/LEDTutorial/src/LEDTutorial_tb.vhd (VHDL-1481)
-- Restoring VHDL parse-tree ieee.numeric_std from C:/lscc/diamond/2.2_x64/cae_library/vhdl_packages/vdbs/ieee/numeric_std.vdb (VHDL-1493)
```

Simulation Top Module: testbench

9. Click **Next.**

The Summary dialog box of figure 22 appears. Make sure that **Run simulator**, **Add top-level signals to waveform display**, and **Run simulation** are selected.

Figure 22: Simulation Project Name



Summary

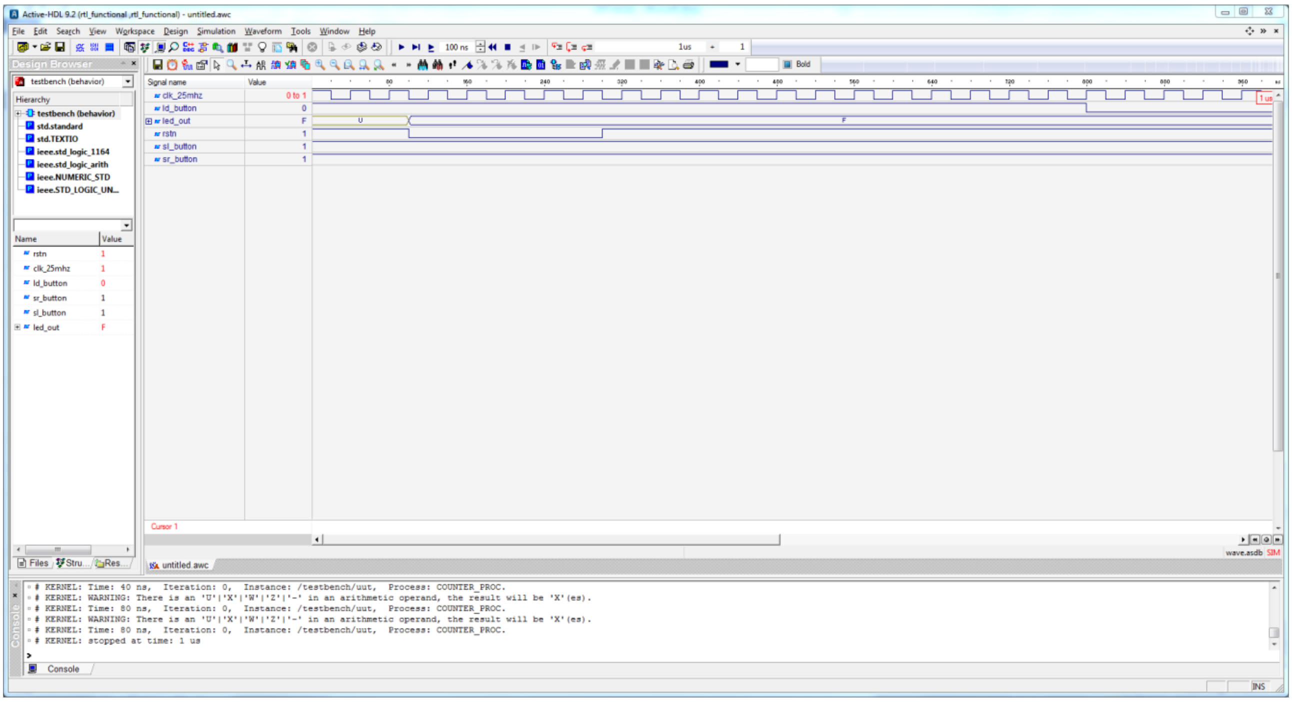
Simulator : Active-HDL
Project Name : rtl_functional
Project Location : C:\LEDTutorial\sim
Simulation Stage : RTL
Simulation Files :
    C:/LEDTutorial/src/LEDTutorial.vhd
    C:/LEDTutorial/src/LEDTutorial_tb.vhd
Simulation Libraries :
    pmi_work
    ovi_machxo2
Simulation Top Module :
    testbench

☑ Run simulator
☑ Add top-level signals to waveform display
☑ Run simulation

10. Click **Finish**.

The Aldec Active-HDL software is launched and the simulation starts automatically as shown in figure 22.

Figure 23: Active-HDL 9.1 Window



Name	Value
rstn	1
clk_25mhz	1
ld_button	0
sr_button	1
sl_button	1
led_out	F

Signal name	Value
clk_25mhz	0 to 1
ld_button	0
led_out	F
rstn	1
sl_button	1
sr_button	1

```
# KERNEL: Time: 40 ns,  Iteration: 0,  Instance: /testbench/uut,  Process: COUNTER_PROC.
# KERNEL: WARNING: There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic operand, the result will be 'X'(es).
# KERNEL: Time: 80 ns,  Iteration: 0,  Instance: /testbench/uut,  Process: COUNTER_PROC.
# KERNEL: WARNING: There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic operand, the result will be 'X'(es).
# KERNEL: Time: 80 ns,  Iteration: 0,  Instance: /testbench/uut,  Process: COUNTER_PROC.
# KERNEL: stopped at time: 1 us
>
```

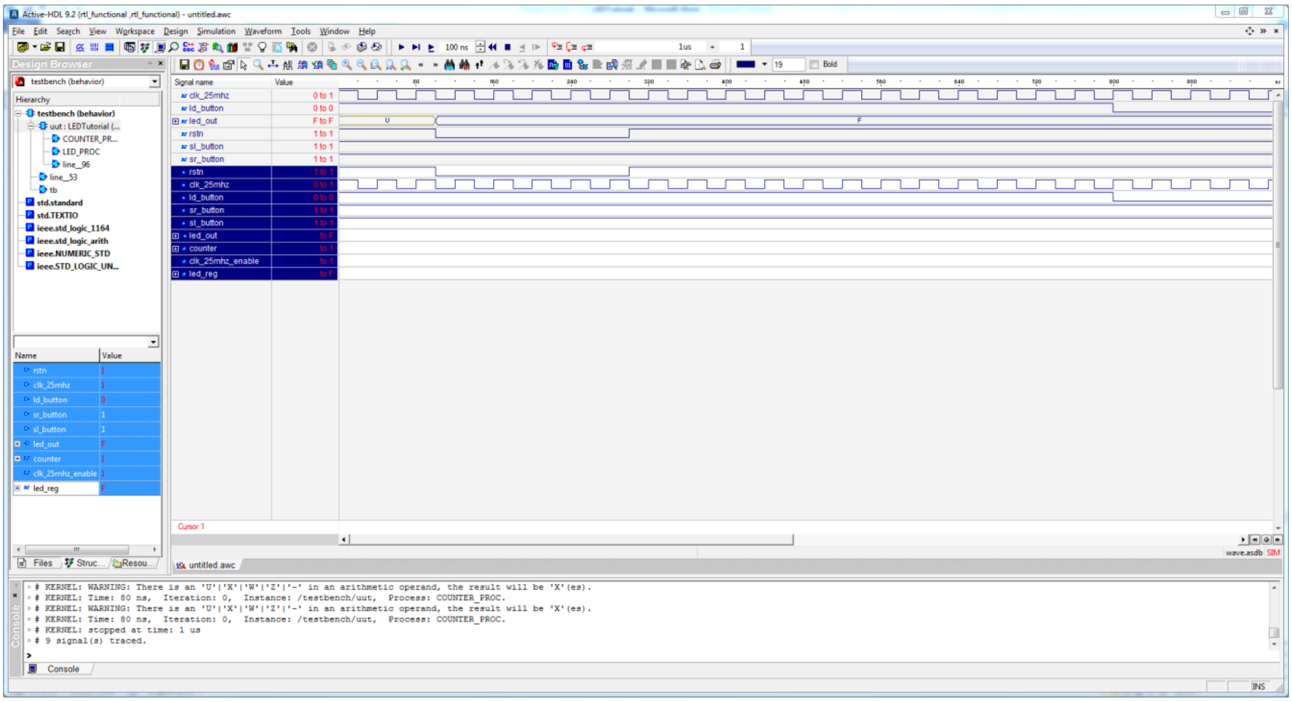
To simulate the design manually:

1. In the Active-HDL Design Browser, select **testbench** from the drop down menu.

2. Select the **Structure** tab in the Design browser.

3. In the **Structure** tab, select uut: LEDTutorial(rtl_ledtutorial), right-click and then Add to Waveform. Same operation may be done by selecting all signals and drug-and-drop to waveform under Signal name column as displayed in Figure 23.

Figure 24: Active-HDL 9.1 Window


Active-HDL 9.2 (rtl_functional .rtl_functional) - untitled.awc
File Edit Search View Workspace Design Simulation Waveform Tools Window Help
Design Browser
testbench (behavior)
Hierarchy
Signal name
Value
Cursor 1
Files
Struc...
Resou...
untitled.awc
Console
wave.asdb
INS

Name	Value
rstn	1
clk_25mhz	1
ld_button	0
sr_button	1
sl_button	1
led_out	F
counter	1
clk_25mhz_enable	1
led_reg	F

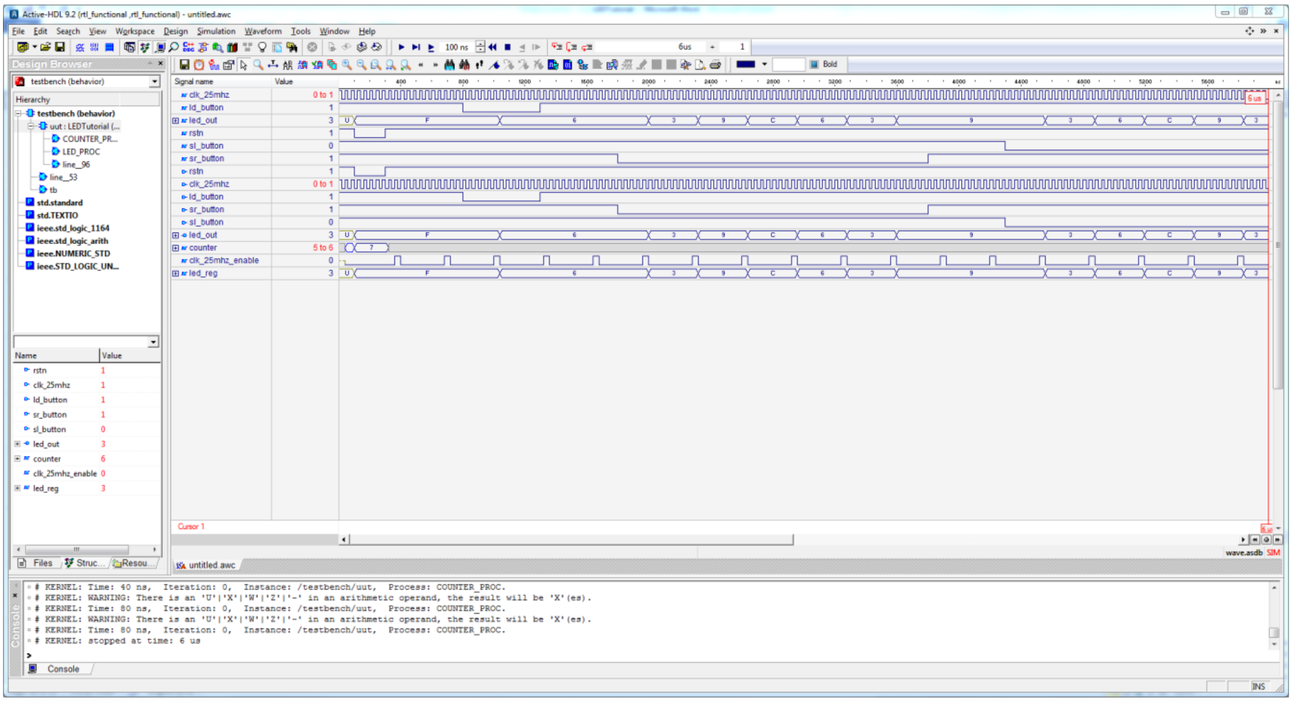
```
# KERNEL: WARNING: There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic operand, the result will be 'X'(es).
# KERNEL: Time: 80 ns,  Iteration: 0,  Instance: /testbench/uut,  Process: COUNTER_PROC.
# KERNEL: WARNING: There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic operand, the result will be 'X'(es).
# KERNEL: Time: 80 ns,  Iteration: 0,  Instance: /testbench/uut,  Process: COUNTER_PROC.
# KERNEL: stopped at time: 1 us
# 9 signal(s) traced.
>
```

6. In **Console** window, write **restart** and then press Enter

7. Always in Console window, write **run 6 us** and press Enter

After completing the simulation, the waveform appears, as shown in Figure 25.

Figure 25: Simulated Waveform


Active-HDL 9.2 (rtl_functional .rtl_functional) - untitled.awc
File Edit Search View Workspace Design Simulation Waveform Tools Window Help
Design Browser
testbench (behavior)
Hierarchy
Signal name
Value
Cursor 1
Files
Struc...
Resou...
untitled.awc
Console
wave.asdb
INS

Name	Value
rstn	1
clk_25mhz	1
ld_button	1
sr_button	1
sl_button	0
led_out	3
counter	6
clk_25mhz_enable	0
led_reg	3

```
# KERNEL: Time: 40 ns,  Iteration: 0,  Instance: /testbench/uut,  Process: COUNTER_PROC.
# KERNEL: WARNING: There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic operand, the result will be 'X'(es).
# KERNEL: Time: 80 ns,  Iteration: 0,  Instance: /testbench/uut,  Process: COUNTER_PROC.
# KERNEL: WARNING: There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic operand, the result will be 'X'(es).
# KERNEL: Time: 80 ns,  Iteration: 0,  Instance: /testbench/uut,  Process: COUNTER_PROC.
# KERNEL: stopped at time: 6 us
>
```

Task 4: Run Synthesis Process

Synthesis is the process of translating a register-transfer-level design into a process-specific, gate-level netlist that is optimized for Lattice Semiconductor FPGAs. Diamond can be used with almost any synthesis tool. Diamond comes with two tools fully integrated: Synopsys Synplify Pro for Lattice and Lattice Synthesis Engine (LSE). “Fully integrated” means that you can set options and run synthesis entirely from within Diamond. As already set during Project wizard, you will be using Synopsis Synplify Pro for Lattice to synthesize your design for the MachXO2 FPGA. To change the synthesis tool, from the Diamond main window, choose **Project > Synthesis Tool**.

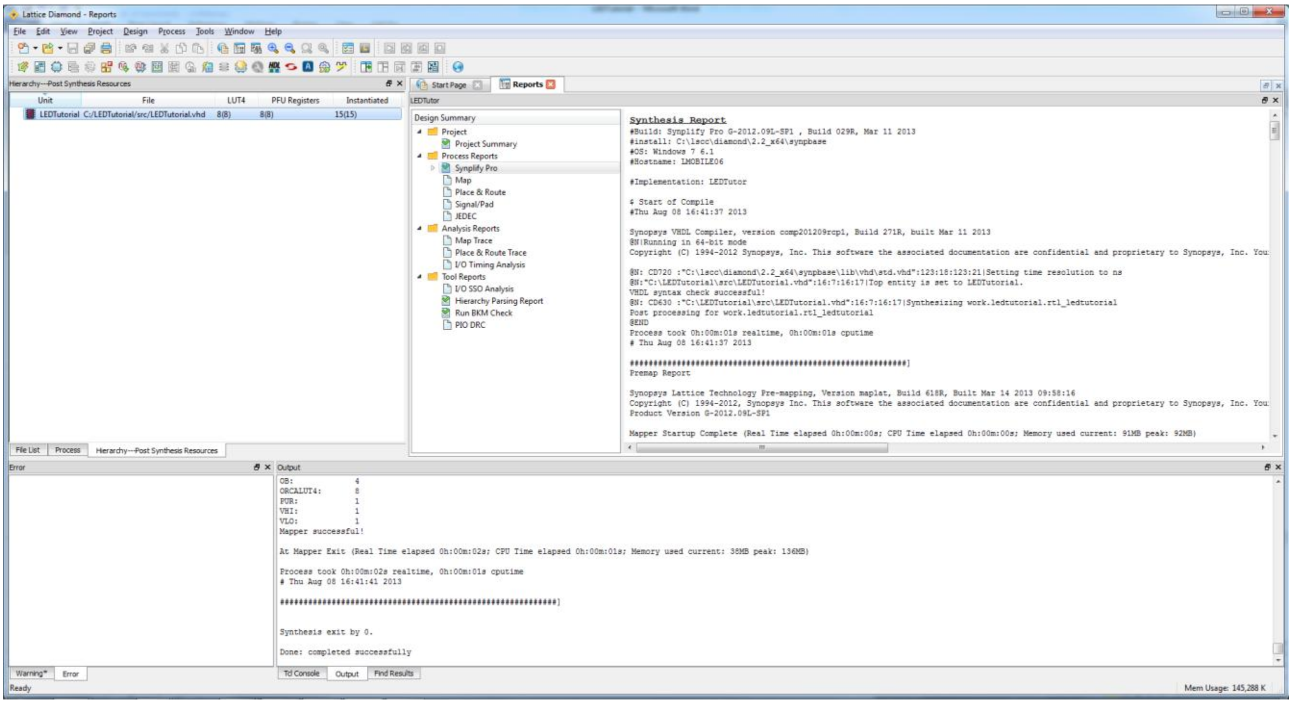
To synthesize the design and examine resource utilization:

1. From the Process View, double-click **Synthesize Design**.

When finished, check the icon next to Synthesize Design in the Process frame. A green check mark indicates success; a yellow triangle indicates success with warnings; a red X indicates failure.

2. When the synthesis process is complete, select **View > Show Views > Post Synthesis Resources**. Select the **Hierarchy – Post Synthesis Resources** tab, shown in Figure 26.

Figure 26: Post Synthesis Resources



Unit	File	LUT4	PFU Registers	Instantiated
LEDTutorial	C:/LEDTutorial/src/LEDTutorial.vhd	8(8)	8(8)	15(15)

```
Synthesis Report
#Build: Synplify Pro G-2012.09L-SP1 , Build 029R, Mar 11 2013
#install: C:\lscc\diamond\2.2_x64\synpbase
#OS: Windows 7 6.1
#Hostname: LMOBILE06

#Implementation: LEDTutor

$ Start of Compile
#Thu Aug 08 16:41:37 2013

Synopsys VHDL Compiler, version comp201209rcp1, Build 271R, built Mar 11 2013
@N|Running in 64-bit mode
Copyright (C) 1994-2012 Synopsys, Inc. This software the associated documentation are confidential and proprietary to Synopsys, Inc. You:

@N: CD720 :"C:\lscc\diamond\2.2_x64\synpbase\lib\vhd\std.vhd":123:18:123:21|Setting time resolution to ns
@N:"C:\LEDTutorial\src\LEDTutorial.vhd":16:7:16:17|Top entity is set to LEDTutorial.
VHDL syntax check successful!
@N: CD630 :"C:\LEDTutorial\src\LEDTutorial.vhd":16:7:16:17|Synthesizing work.ledtutorial.rtl_ledtutorial
Post processing for work.ledtutorial.rtl_ledtutorial
@END
Process took 0h:00m:01s realtime, 0h:00m:01s cputime
# Thu Aug 08 16:41:37 2013

###########################################################]
Premap Report

Synopsys Lattice Technology Pre-mapping, Version maplat, Build 618R, Built Mar 14 2013 09:58:16
Copyright (C) 1994-2012, Synopsys Inc. This software the associated documentation are confidential and proprietary to Synopsys, Inc. You:
Product Version G-2012.09L-SP1

Mapper Startup Complete (Real Time elapsed 0h:00m:00s; CPU Time elapsed 0h:00m:00s; Memory used current: 91MB peak: 92MB)
```

```
OB:             4
ORCALUT4:       8
PUR:            1
VHI:            1
VLO:            1
Mapper successful!

At Mapper Exit (Real Time elapsed 0h:00m:02s; CPU Time elapsed 0h:00m:01s; Memory used current: 38MB peak: 136MB)

Process took 0h:00m:02s realtime, 0h:00m:01s cputime
# Thu Aug 08 16:41:41 2013

###########################################################]


Synthesis exit by 0.

Done: completed successfully
```

The Post-Synthesis Hierarchy View displays the number of logical resources within each level of the design. In the Hierarchy table shown in Figure 20, LEDTutorial is the top module displaying the resource utilization.

- LUT4 8(8) – 8 represents the total LUT4 count utilization throughout the design and 8, round brackets, represents the LUT4 utilized only in the design module LEDTutorial,
- PFU Registers 8(8) – 8 represents the total PFU register utilization throughout the design and 8, round brackets, represents the PFU registers utilized only in the design module LEDTutorial. Similar utilization is shown for the I/O registers, carry cells and SLICEs.

Task 5: Set Timing and Location Assignments

Timing and location assignments constrain logic synthesis, as well as backend map, place, and route programs to help meet your design requirements. A well constrained design helps optimization algorithms work as efficiently as possible. In this section you'll set default timing constraints for the operating frequency and I/O timing then assign package pins to specific I/O signals.

To set timing and location assignments:

1. From the Process view, double-click **Translate Design** and then **Map Design**.

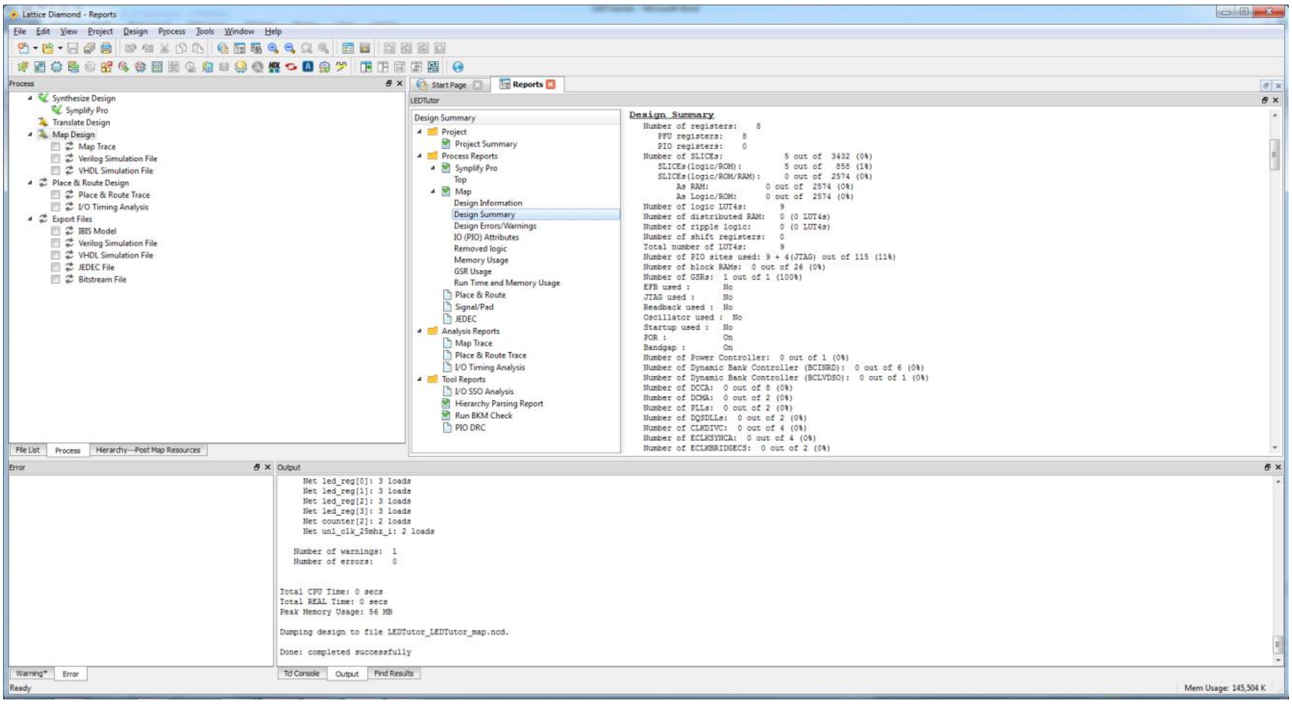
The batch interface to logic synthesis, EDIF translation, and the design mapper run. Report files appears in the Reports view. To view each process report, select the process in the **Design Summary** pane. Each major stage of an FPGA implementation is illustrated as a milestone in the Process view: Synthesize Design, Translate Design, Map Design, Place&Route Design, and Export Files. The status of any stage is represented by the following color-coded icons:

- Completed (Green check mark) - The stage completed successfully and produced output.
- Warning (Yellow Exclamation mark) - The stage completed with warning messages generated. You can go to the Warning panel to view the warning messages.
- Error (Red cross mark) - The stage failed. You can go to the Error panel to view the error messages.

2. From the **Design Summary** pane of the Reports view, select **Process Reports > Map**, click the sign before Map to display the sections in a list and then then select **Design Summary**.

The report highlights the Design Summary section of the report. To view the contents of the entire report, click on the report to be viewed. The entire report is then displayed in the right pane of the Reports view. Use the scroll bar to navigate through the report. Some of the reports are divided into sections (for example, Map, Place & Route, and Signal/Pad). Click the sign before the report to display the sections in a list. Choose the desired section. The whole report will be displayed with the selected section displayed at the top of the right pane of the Reports view.

Figure 27: MAP Report



```
Design Summary
   Number of registers:    8
      PFU registers:    8
      PIO registers:    0
   Number of SLICEs:            5 out of  3432 (0%)
      SLICEs(logic/ROM):        5 out of   858 (1%)
      SLICEs(logic/ROM/RAM):    0 out of  2574 (0%)
          As RAM:            0 out of  2574 (0%)
          As Logic/ROM:      0 out of  2574 (0%)
   Number of logic LUT4s:      9
   Number of distributed RAM:   0 (0 LUT4s)
   Number of ripple logic:      0 (0 LUT4s)
   Number of shift registers:   0
   Total number of LUT4s:      9
   Number of PIO sites used: 9 + 4(JTAG) out of 115 (11%)
   Number of block RAMs:  0 out of 26 (0%)
   Number of GSRs:  1 out of 1 (100%)
   EFB used :        No
   JTAG used :       No
   Readback used :   No
   Oscillator used :  No
   Startup used :    No
   POR :             On
   Bandgap :         On
   Number of Power Controller:  0 out of 1 (0%)
   Number of Dynamic Bank Controller (BCINRD):  0 out of 6 (0%)
   Number of Dynamic Bank Controller (BCLVDSO):  0 out of 1 (0%)
   Number of DCCA:  0 out of 8 (0%)
   Number of DCMA:  0 out of 2 (0%)
   Number of PLLs:  0 out of 2 (0%)
   Number of DQSDLLs:  0 out of 2 (0%)
   Number of CLKDIVC:  0 out of 4 (0%)
   Number of ECLKSYNCA:  0 out of 4 (0%)
   Number of ECLKBRIDGECS:  0 out of 2 (0%)
```

```
     Net led_reg[0]: 3 loads
     Net led_reg[1]: 3 loads
     Net led_reg[2]: 3 loads
     Net led_reg[3]: 3 loads
     Net counter[2]: 2 loads
     Net un1_clk_25mhz_i: 2 loads

   Number of warnings:  1
   Number of errors:    0


Total CPU Time: 0 secs
Total REAL Time: 0 secs
Peak Memory Usage: 56 MB

Dumping design to file LEDTutor_LEDTutor_map.ncd.

Done: completed successfully
```

4. Choose **Tools > Spreadsheet View**.

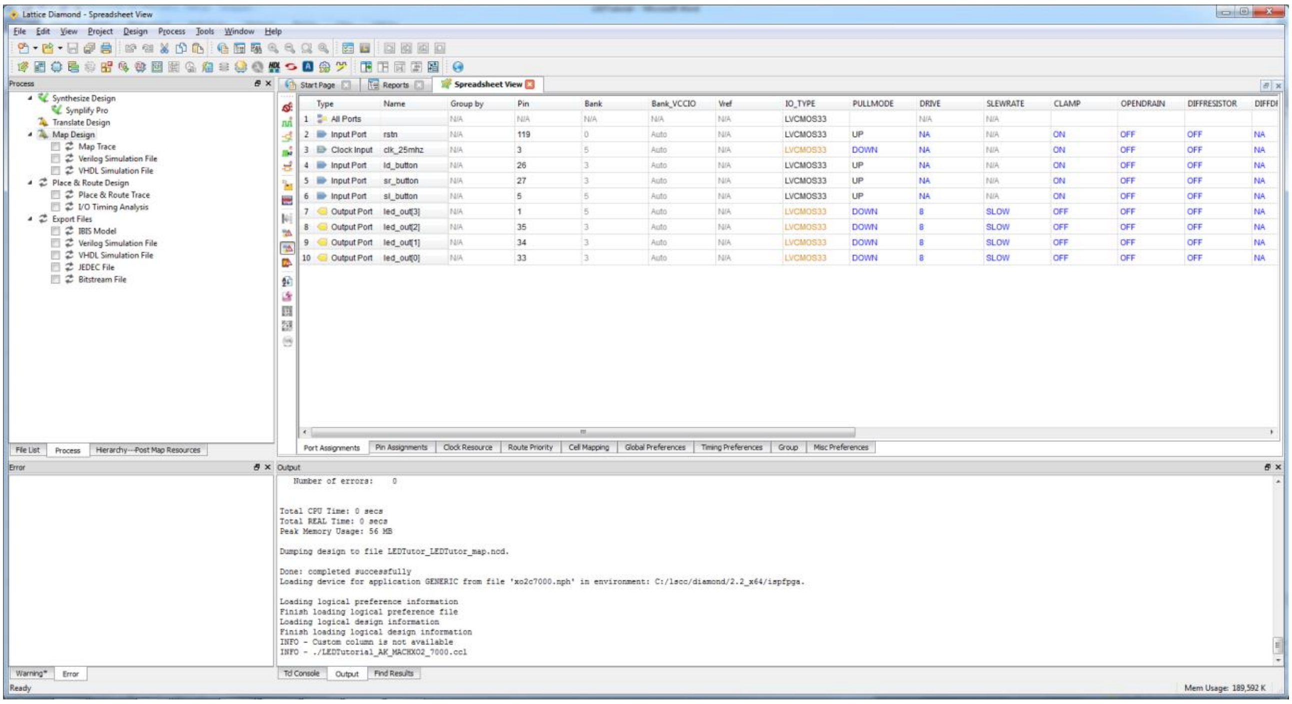
The Spreadsheet View appears like in Figure 28. The Spreadsheet View is one of several preference editors available to you to define timing, I/O and floorplan constraints for the place and route tools. Preferences are organized by type into separate tabs of the Spreadsheet View.

5. Select the Port Assignments sheet from the Spreadsheet View.

6. Right click the IO_Type cell of the All Ports row. A pull-down menu of signal standards appears. Select **LVCMOS33**, if it is not already selected. The port attributes display is updated with the new IO_TYPE. Cell entries in the Spread Sheet view are color-coded to indicate the source of a preference setting:

- Black - User-defined setting.
- Blue - Default.
- Orange - Implied by another user-defined setting.

Figure 28: Spreadsheet View



	Type	Name	Group by	Pin	Bank	Bank_VCCIO	Vref	IO_TYPE	PULLMODE	DRIVE	SLEWRATE	CLAMP	OPENDRAIN	DIFFRESISTOR	DIFFDI
1	All Ports		N/A	N/A	N/A	N/A	N/A	LVCMOS33		N/A	N/A				
2	Input Port	rstn	N/A	119	0	Auto	N/A	LVCMOS33	UP	NA	N/A	ON	OFF	OFF	NA
3	Clock Input	clk_25mhz	N/A	3	5	Auto	N/A	LVCMOS33	DOWN	NA	N/A	ON	OFF	OFF	NA
4	Input Port	ld_button	N/A	26	3	Auto	N/A	LVCMOS33	UP	NA	N/A	ON	OFF	OFF	NA
5	Input Port	sr_button	N/A	27	3	Auto	N/A	LVCMOS33	UP	NA	N/A	ON	OFF	OFF	NA
6	Input Port	sl_button	N/A	5	5	Auto	N/A	LVCMOS33	UP	NA	N/A	ON	OFF	OFF	NA
7	Output Port	led_out[3]	N/A	1	5	Auto	N/A	LVCMOS33	DOWN	8	SLOW	OFF	OFF	OFF	NA
8	Output Port	led_out[2]	N/A	35	3	Auto	N/A	LVCMOS33	DOWN	8	SLOW	OFF	OFF	OFF	NA
9	Output Port	led_out[1]	N/A	34	3	Auto	N/A	LVCMOS33	DOWN	8	SLOW	OFF	OFF	OFF	NA
10	Output Port	led_out[0]	N/A	33	3	Auto	N/A	LVCMOS33	DOWN	8	SLOW	OFF	OFF	OFF	NA

```
   Number of errors:     0


Total CPU Time: 0 secs
Total REAL Time: 0 secs
Peak Memory Usage: 56 MB

Dumping design to file LEDTutor_LEDTutor_map.ncd.

Done: completed successfully
Loading device for application GENERIC from file 'xo2c7000.nph' in environment: C:/lscc/diamond/2.2_x64/ispfpga.

Loading logical preference information
Finish loading logical preference file
Loading logical design information
Finish loading logical design information
INFO - Custom column is not available
INFO - ./LEDTutorial_AK_MACHXO2_7000.ccl
```

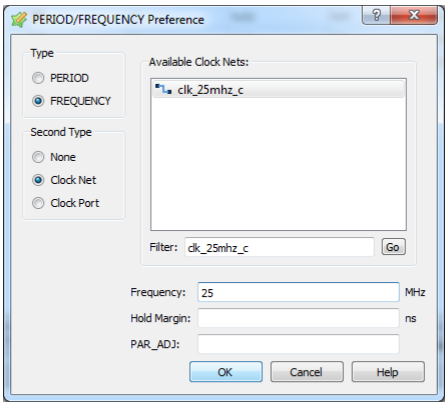
7. Click the **Period/Frequency** icon on the Spreadsheet View tool bar. The Period/Frequency Preference dialog box appears like in figure 29.

8. Enter the following preference settings:

- Type: **FREQUENCY**
- Second Type: **Clock Net**
- Available Clock Nets: **clk_25mhz_c**
- Frequency: **25MHz**

Click **OK**.

Figure 29: PERIOD/FREQUENCY Preference window



Type
○ PERIOD
● FREQUENCY

Second Type
○ None
● Clock Net
○ Clock Port

Available Clock Nets: clk_25mhz_c
Filter: clk_25mhz_c
Frequency: 25 MHz
Hold Margin: ns
PAR_ADJ:

The Timing Preferences tab of the Spreadsheet View appears with the new **FREQUENCY** preference defined.

9. Choose **File > Save LEDTutorial_AK_MACHXO2_7000.lpf** from the detached Spreadsheet View. The project Logical Preference File (.lpf) is updated. Close the Spreadsheet View.

10. From the File List view of the Diamond main window, LPF Constraints Files folder, double-click the **LEDTutorial_AK_MACHXO2_7000**.lpf file. The Source Editor appears with the ASCII LPF file. Note the timing and location preferences defined so far. Close the Source Editor.

Task 6: Running Place and Route

Use the Process view to run the Translate Design, Map Design, and Place&Route Design process stages.

To run place and route:

1. From the Process List double-click **Place & Route Design**. The place and route tools are run. Intermediate results appear in the Output frame of the Diamond main window.

2. From the **Design Summary** pane of the Reports view, find the **Process Reports** section. You will find a green check mark appears before the reports generated successfully. Expand the **Process Reports** section. Select **Place & Route**. Details about Place & Route appear in the pane to the right.

3. From the Process List double-click **Place & Route Trace**. The TRACE timing analyzer is run.

4. From the **Design Summary** pane of the Reports view, expand **Analysis Reports**, and then select **Place & Route Trace** to view the report in the pane to the right.

Task 7: Examine Post Place and Route Results

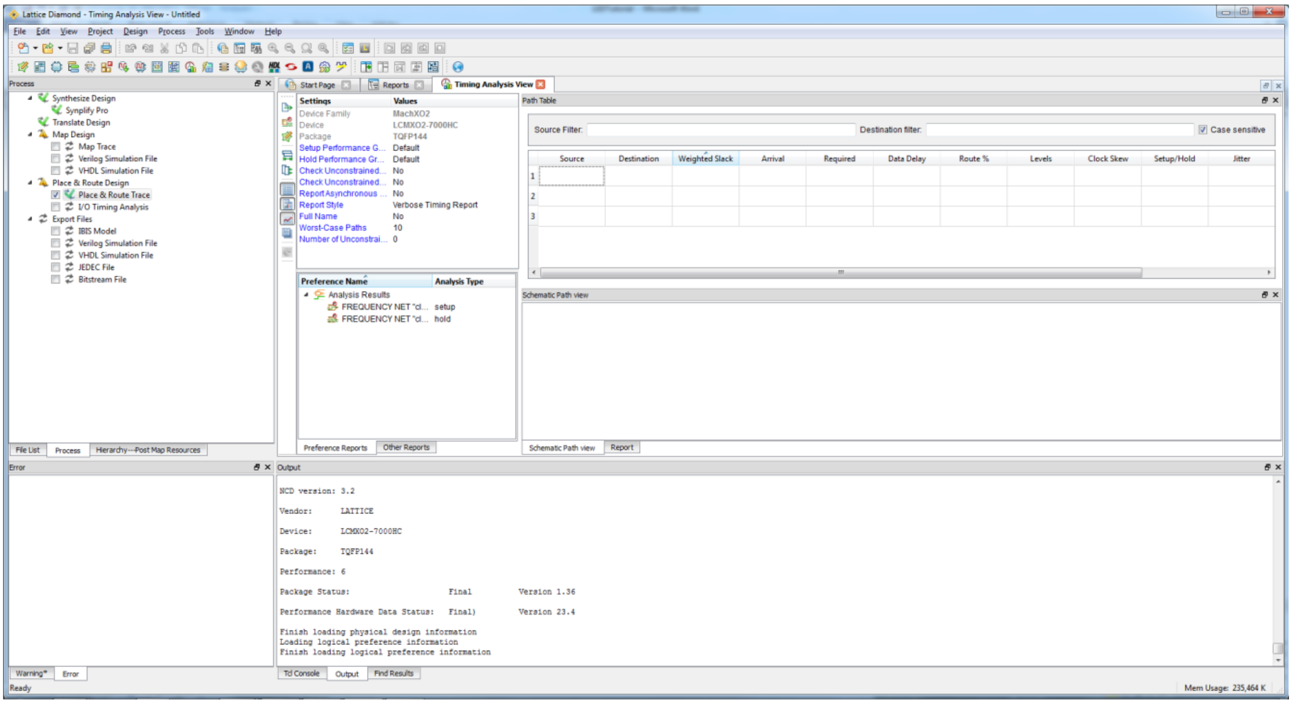
Static timing analysis (STA) is a method for determining if your circuit design meets timing constraints. It is a method that does not require the use of simulation. The STA process employs conservative modeling of gate and interconnect delays that reflect different ranges of operating conditions on various dies, providing complete verification coverage. In this task, you will view the results of the Static timing analysis.

Examine timing analysis results:

1. Choose Tools > Timing Analysis View

The Timing Analysis view appears like in figure 30.

Figure 30: Timing Analysis View



Settings	Values
Device Family	MachXO2
Device	LCMXO2-7000HC
Package	TQFP144
Setup Performance G...	Default
Hold Performance Gr...	Default
Check Unconstrained...	No
Check Unconstrained...	No
Report Asynchronous ...	No
Report Style	Verbose Timing Report
Full Name	No
Worst-Case Paths	10
Number of Unconstrai...	0

Preference Name	Analysis Type
Analysis Results	
FREQUENCY NET "cl...	setup
FREQUENCY NET "cl...	hold

```
NCD version: 3.2

Vendor:      LATTICE

Device:      LCMXO2-7000HC

Package:     TQFP144

Performance: 6

Package Status:                     Final          Version 1.36

Performance Hardware Data Status:   Final)         Version 23.4

Finish loading physical design information
Loading logical preference information
Finish loading logical preference information
```

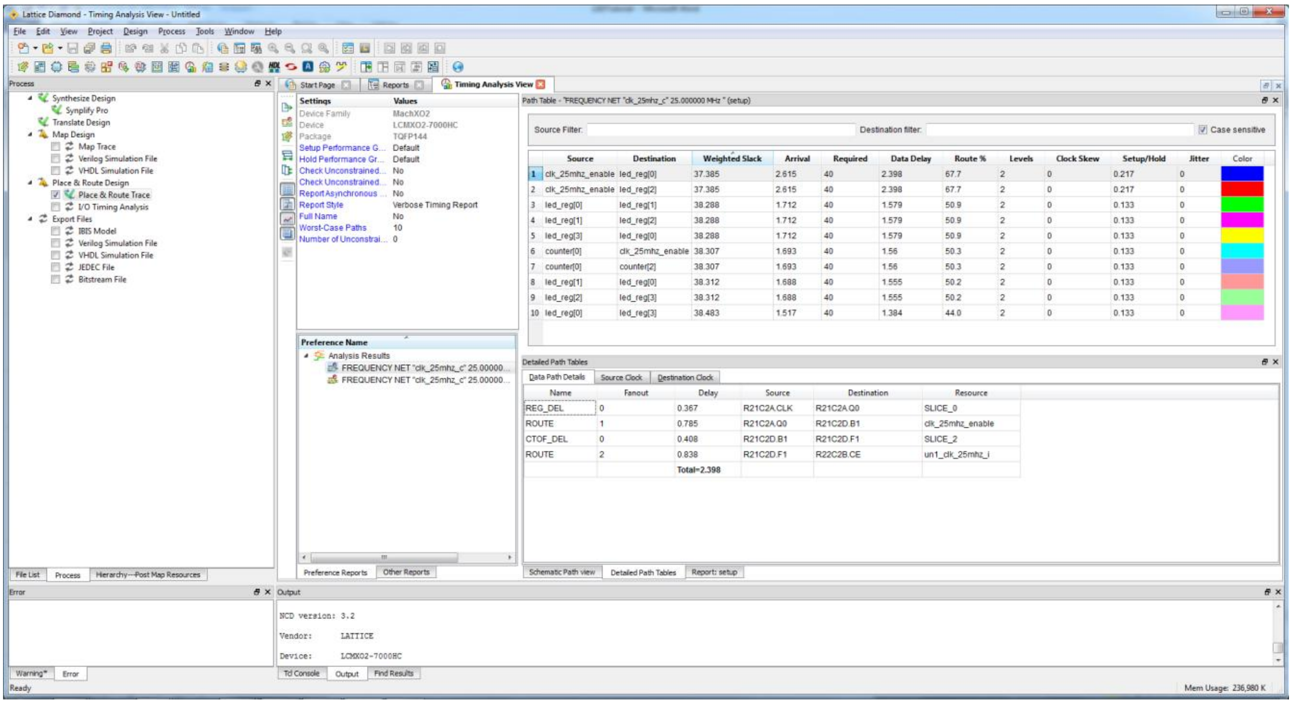
A summary of the post-route static timing analysis settings such as target device information, preference file, performance grade, and environment index of the available analysis results. Related timing preferences appear in each analysis section. The lower left pane provides an index of the available analysis results. Related timing preferences appear in each analysis section.

2. From the Analysis pane (on the lower left of the Timing Analysis view), select **FREQUENCY NET "clk_25mhz_c" 25.000000 MHz setup**. Select the first row in the **Path Table**. The Detailed Path Tables are updated.

3. Select the **Data Path Details** tab.

Each component of the data path delay is identified alternating between route delays and combinatorial or clock-to-output type delays, as shown in Figure 31.

Figure 31: Timing Analysis



Settings	Values
Device Family	MachXO2
Device	LCMXO2-7000HC
Package	TQFP144
Setup Performance G...	Default
Hold Performance Gr...	Default
Check Unconstrained...	No
Check Unconstrained...	No
Report Asynchronous ...	No
Report Style	Verbose Timing Report
Full Name	No
Worst-Case Paths	10
Number of Unconstrai...	0

Preference Name
- Analysis Results
  - FREQUENCY NET "clk_25mhz_c" 25.00000...
  - FREQUENCY NET "clk_25mhz_c" 25.00000...

Path Table - "FREQUENCY NET "clk_25mhz_c" 25.000000 MHz " (setup)

Source Filter: Destination filter: ☑ Case sensitive

	Source	Destination	Weighted Slack	Arrival	Required	Data Delay	Route %	Levels	Clock Skew	Setup/Hold	Jitter	Color
1	clk_25mhz_enable	led_reg[0]	37.385	2.615	40	2.398	67.7	2	0	0.217	0	🟦
2	clk_25mhz_enable	led_reg[2]	37.385	2.615	40	2.398	67.7	2	0	0.217	0	🟥
3	led_reg[0]	led_reg[1]	38.288	1.712	40	1.579	50.9	2	0	0.133	0	🟩
4	led_reg[1]	led_reg[2]	38.288	1.712	40	1.579	50.9	2	0	0.133	0	🟪
5	led_reg[3]	led_reg[0]	38.288	1.712	40	1.579	50.9	2	0	0.133	0	🟨
6	counter[0]	clk_25mhz_enable	38.307	1.693	40	1.56	50.3	2	0	0.133	0	🟦
7	counter[0]	counter[2]	38.307	1.693	40	1.56	50.3	2	0	0.133	0	🟪
8	led_reg[1]	led_reg[0]	38.312	1.688	40	1.555	50.2	2	0	0.133	0	🟥
9	led_reg[2]	led_reg[3]	38.312	1.688	40	1.555	50.2	2	0	0.133	0	🟩
10	led_reg[0]	led_reg[3]	38.483	1.517	40	1.384	44.0	2	0	0.133	0	🟪

Detailed Path Tables

Data Path Details | Source Clock | Destination Clock

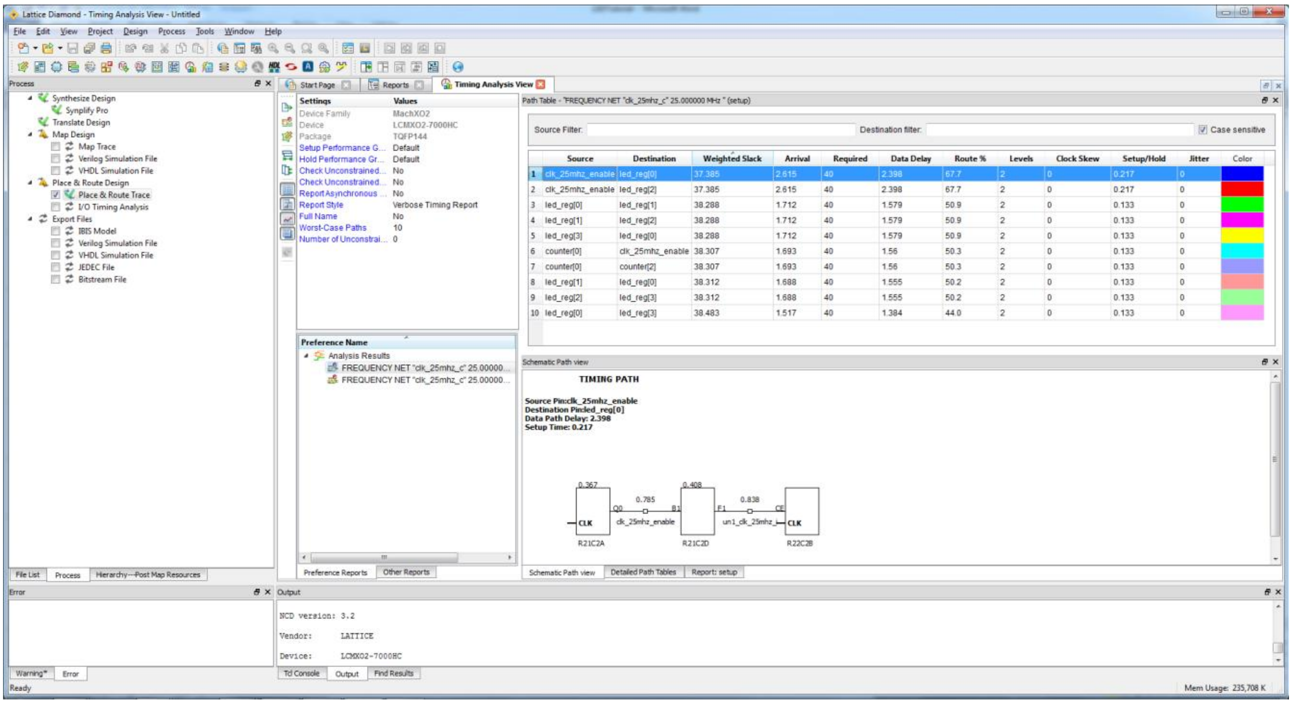
Name	Fanout	Delay	Source	Destination	Resource
REG_DEL	0	0.367	R21C2A.CLK	R21C2A.Q0	SLICE_0
ROUTE	1	0.785	R21C2A.Q0	R21C2D.B1	clk_25mhz_enable
CTOF_DEL	0	0.408	R21C2D.B1	R21C2D.F1	SLICE_2
ROUTE	2	0.838	R21C2D.F1	R22C2B.CE	un1_clk_25mhz_i
		Total=2.398			

Output

```
NCD version: 3.2
Vendor:      LATTICE
Device:      LCMXO2-7000HC
```

4. Select the Schematic Path view. A schematic graphic of the data path timing path appears as shown in Figure 32.

Figure 32: Schematic Path View



Settings	Values
Device Family	MachXO2
Device	LCMXO2-7000HC
Package	TQFP144
Setup Performance G...	Default
Hold Performance Gr...	Default
Check Unconstrained...	No
Check Unconstrained...	No
Report Asynchronous ...	No
Report Style	Verbose Timing Report
Full Name	No
Worst-Case Paths	10
Number of Unconstrai...	0

Path Table - "FREQUENCY NET "clk_25mhz_c" 25.000000 MHz " (setup)

	Source	Destination	Weighted Slack	Arrival	Required	Data Delay	Route %	Levels	Clock Skew	Setup/Hold	Jitter	Color
1	clk_25mhz_enable	led_reg[0]	37.385	2.615	40	2.398	67.7	2	0	0.217	0	🟦
2	clk_25mhz_enable	led_reg[2]	37.385	2.615	40	2.398	67.7	2	0	0.217	0	🟥
3	led_reg[0]	led_reg[1]	38.288	1.712	40	1.579	50.9	2	0	0.133	0	🟩
4	led_reg[1]	led_reg[2]	38.288	1.712	40	1.579	50.9	2	0	0.133	0	🟪
5	led_reg[3]	led_reg[0]	38.288	1.712	40	1.579	50.9	2	0	0.133	0	🟨
6	counter[0]	clk_25mhz_enable	38.307	1.693	40	1.56	50.3	2	0	0.133	0	🟦
7	counter[0]	counter[2]	38.307	1.693	40	1.56	50.3	2	0	0.133	0	🟪
8	led_reg[1]	led_reg[0]	38.312	1.688	40	1.555	50.2	2	0	0.133	0	🟥
9	led_reg[2]	led_reg[3]	38.312	1.688	40	1.555	50.2	2	0	0.133	0	🟩
10	led_reg[0]	led_reg[3]	38.483	1.517	40	1.384	44.0	2	0	0.133	0	🟪

TIMING PATH

Source Pin:clk_25mhz_enable
Destination Pin:led_reg[0]
Data Path Delay: 2.398
Setup Time: 0.217

```
NCD version: 3.2
Vendor:      LATTICE
Device:      LCMXO2-7000HC
```

5. Close Spreadsheet View. In the Save dialog box, click **No** to discard the change.

Task 8: Run Export Utility Programs

Use the Process view to generate files for exporting. One of the files exported will be a JEDEC file (.jed) which will be used to program a MachXO2 device in the next task.

1. From the Process view, choose **Export Files**. A set of export files appear under the Export Files process.

2. Select **JEDEC File** in the Export Files section

3. Click the **Run** button on the Diamond toolbar or double-click on JEDEC File

Diamond generates the selected file and saves them in your project directory.

Task 9: Download a JEDEC programming file

This task requires that you have an ARTEKIT AK-MACHXO2-7000 Development Kit (http://www.artekit.eu/products/devboards/ak-machx02-7000/).
In the previous section, you have generated JEDEC file LEDTutor_LEDTutor.jed. In this section, you will use Diamond Programmer tool to download a JEDEC programming file to a MachXO2 FPGA mounted on an AK-MACHXO2-7000 Development Kit board.

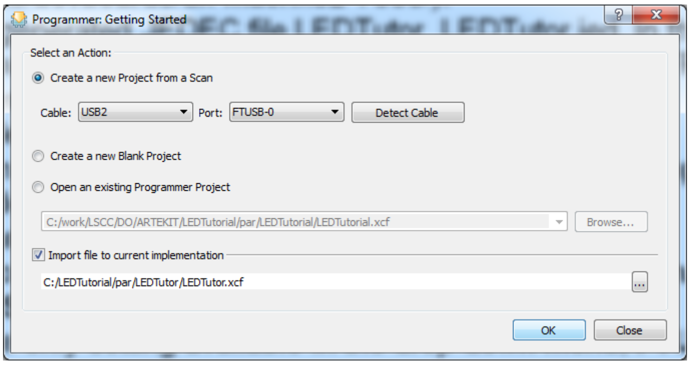
To download the JEDEC file to the FPGA on the board:

1. Connect a USB cable from your computer to the AK-MACHXO2-7000 Development Kit board. Give the computer a few seconds to detect the USB device moving to next step.

2. Choose **Tools > Programmer**

3. In the Getting Started dialog box, choose **Create a new Project from a Scan**, and then like figure 33:
    a. In the Cable box, select **USB2**.
    b. In the Port box, choose the only setting available in the drop-down menu, **FTUSB-0**.
    c. Click **OK**.

Figure 33: Programmer Setup windows



Programmer: Getting Started

Select an Action:

● Create a new Project from a Scan

Cable: USB2 Port: FTUSB-0

○ Create a new Blank Project

○ Open an existing Programmer Project

C:/work/LSCC/DO/ARTEKIT/LEDTutorial/par/LEDTutorial/LEDTutorial.xcf

☑ Import file to current implementation

C:/LEDTutorial/par/LEDTutor/LEDTutor.xcf

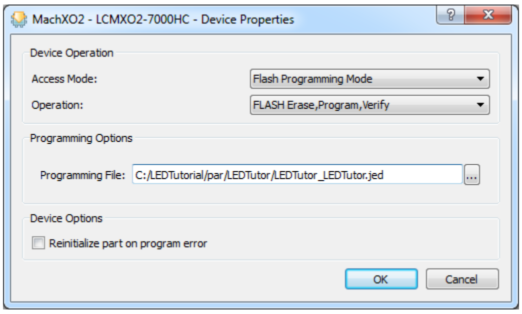
Programmer scans the device database, and then the Programmer view displays in Diamond.

4. Ensure that the device **LCMXO2-7000HC** is selected in the Device column.

5. Double-click the Operation column to display the Device Properties dialog box and verify the following settings as shown in figure 34:

a. For Access Mode, **Flash Programming Mode** from the pull-down menu.
b. For Operation, FLASH Erase, Program, Verify from the pull-down menu.
c. Ensure that the file named LEDTutor_LEDTutor.jed is selected as the programming file.

Figure 34: Programmer settings



MachXO2 - LCMXO2-7000HC - Device Properties

Device Operation

Access Mode: Flash Programming Mode

Operation: FLASH Erase,Program,Verify

Programming Options

Programming File: C:/LEDTutorial/par/LEDTutor/LEDTutor_LEDTutor.jed

Device Options

☐ Reinitialize part on program error

6. Click **OK**

7. Click the Program button on the Programmer toolbar to initiate the download.

8. If the programming process succeeded, you will see a green-shaded PASS in the Programmer Status column. Check the Programmer output console to see if the download passed reporting message **Operation: successful**.

9. At the end of this process, the FPGA is loaded with the LEDTutorial design. The programming JEDEC file allows you to test the functionality of the AK-MACHXO2-7000 Development Kit board.

To further test the design:

a) press K2 button, LD_BUTTON, to load internal register with binary value 0110. That value is also displayed on leds: D7 and D8 are switched-on, D6 and D9 are switched-off
b) press K3 button, SR_BUTTON, to activate a right shift of internal register. “Leds simulated movement” is from left to right
c) press K1 button, SL_BUTTON, to activate a left shift of internal register. “Leds simulated movement” is from right to left

10. In Diamond, choose **File > Save**. In the “Save .xcf File As” dialog box, enter LEDTutor.xcf in the File Name box, and click **Save.**

Summary of Accomplishments

You have completed the *Lattice Diamond Tutorial*. In this tutorial, you have learned how to:

- Create a new Lattice Diamond project
- Check Hardware Description Language (HDL)
- Verify functionality with simulation
- Examine resources
- Run synthesis process
- Set timing and location assignments
- Run place and route
- Examine post place and route results
- Download a programming file to an FPGA

Recommended References

You can find additional information on the subjects covered by this tutorial in the online Help.