



Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!



Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832

Email & Skype: info@chipsmall.com Web: www.chipsmall.com

Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China



Lattice Diamond User Guide



August 2013

Copyright

Copyright © 2013 Lattice Semiconductor Corporation.

This document may not, in whole or part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without prior written consent from Lattice Semiconductor Corporation.

Trademarks

Lattice Semiconductor Corporation, L Lattice Semiconductor Corporation (logo), L (stylized), L (design), Lattice (design), LSC, CleanClock, Custom Mobile Device, DiePlus, E²CMOS, Extreme Performance, FlashBAK, FlexiClock, flexiFLASH, flexiMAC, flexiPCS, FreedomChip, GAL, GDX, Generic Array Logic, HDL Explorer, iCE Dice, iCE40, iCE65, iCEblink, iCEcable, iCEchip, iCEcube, iCEcube2, iCEman, iCEprog, iCEsab, iCEsocket, IPexpress, ISP, ispATE, ispClock, ispDOWNLOAD, ispGAL, ispGDS, ispGDX, ispGDX2, ispGDXV, ispGENERATOR, ispJTAG, ispLEVER, ispLeverCORE, ispLSI, ispMACH, ispPAC, ispTRACY, ispTURBO, ispVIRTUAL MACHINE, ispVM, ispXP, ispXPGA, ispXPLD, Lattice Diamond, LatticeCORE, LatticeEC, LatticeECP, LatticeECP-DSP, LatticeECP2, LatticeECP2M, LatticeECP3, LatticeECP4, LatticeMico, LatticeMico8, LatticeMico32, LatticeSC, LatticeSCM, LatticeXP, LatticeXP2, MACH, MachXO, MachXO2, MACO, mobileFPGA, ORCA, PAC, PAC-Designer, PAL, Performance Analyst, Platform Manager, ProcessorPM, PURESPEED, Reveal, SensorExtender, SiliconBlue, Silicon Forest, Speedlocked, Speed Locking, SuperBIG, SuperCOOL, SuperFAST, SuperWIDE, sysCLOCK, sysCONFIG, sysDSP, sysHSI, sysI/O, sysMEM, The Simple Machine for Complex Design, TraceID, TransFR, UltraMOS, and specific product designations are either registered trademarks or trademarks of Lattice Semiconductor Corporation or its subsidiaries in the United States and/or other countries. ISP, Bringing the Best Together, and More of the Best are service marks of Lattice Semiconductor Corporation.

Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

Disclaimers

NO WARRANTIES: THE INFORMATION PROVIDED IN THIS DOCUMENT IS “AS IS” WITHOUT ANY EXPRESS OR IMPLIED WARRANTY OF ANY KIND INCLUDING WARRANTIES OF ACCURACY, COMPLETENESS, MERCHANTABILITY, NONINFRINGEMENT OF INTELLECTUAL PROPERTY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL LATTICE SEMICONDUCTOR CORPORATION (LSC) OR ITS SUPPLIERS BE LIABLE FOR ANY DAMAGES WHATSOEVER (WHETHER DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL, INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OF OR INABILITY TO USE THE INFORMATION PROVIDED IN THIS DOCUMENT, EVEN IF LSC HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. BECAUSE SOME JURISDICTIONS PROHIBIT THE EXCLUSION OR LIMITATION OF CERTAIN LIABILITY, SOME OF THE ABOVE LIMITATIONS MAY NOT APPLY TO YOU.

LSC may make changes to these materials, specifications, or information, or to the products described herein, at any time without notice. LSC makes no commitment to update this documentation. LSC reserves the right to discontinue any product or service without notice and assumes no obligation to correct any errors contained herein or to advise any user of this document of any correction if such be made. LSC recommends its customers obtain the latest version of the relevant information to establish, before ordering, that the information being relied upon is current.

Type Conventions Used in This Document

Convention	Meaning or Use
Bold	Items in the user interface that you select or click. Text that you type into the user interface.
<i><Italic></i>	Variables in commands, code syntax, and path names.
Ctrl+L	Press the two keys at the same time.
<code>Courier</code>	Code examples. Messages, reports, and prompts from the software.
<code>...</code>	Omitted material in a line of code.
<code>.</code> <code>.</code> <code>.</code>	Omitted lines in code and report examples.
[]	Optional items in syntax descriptions. In bus specifications, the brackets are required.
()	Grouped items in syntax descriptions.
{ }	Repeatable items in syntax descriptions.
	A choice between items in syntax descriptions.

Contents

Chapter 1	Introduction	1
	Lattice Diamond Overview	1
	User Guide Organization	2
Chapter 2	Getting Started	3
	Prerequisites	3
	Running Lattice Diamond	3
	Creating a New Project	5
	Opening an Existing Project	11
	Importing an ispLEVER Project	12
	Next Steps	13
	Differences from ispLEVER	13
Chapter 3	Design Environment Fundamentals	15
	Overview	15
	Project-Based Environment	16
	Process Flow	17
	Shared Memory	18
	Context-Sensitive Data Views	18
	Cross-Probing	20
Chapter 4	User Interface Operation	23
	Overview	23
	Menus and Toolbars	24
	Project Views	25
	Tool View Area	26
	Output and Tcl Console	26
	Status Information	27
	Basic UI Controls	27

	Start Page	28
	File List	28
	Process	30
	Hierarchy	30
	Reports	32
	Tool Views	34
	Tcl Console	35
	Output	36
	Error, Warning, and Info	36
	Find Results	36
	Common Tasks	37
	Controlling Views	37
	Managing Layouts	38
	Cross-Probing Between Views	42
Chapter 5	Working with Projects	43
	Overview	43
	Implementations	45
	Input Files	46
	Synthesis Constraint Files	47
	LPF Constraint Files	47
	Debug Files	48
	Script Files	49
	Analysis Files	49
	Programming Files	49
	Strategies	50
	Area	51
	I/O Assistant	52
	Quick	53
	Timing	55
	User-Defined	55
	Common Tasks	56
	Creating a Project	56
	Changing the Target Device	56
	Setting the Top Level of the Design	56
	Editing Files	57
	Saving Project Data	57
Chapter 6	Lattice Diamond Design Flow	59
	Overview	59
	Design Flow Processes	60
	Running Processes	61
	Implementation Flow and Tasks	63
	Run Management	64
	HDL Design Hierarchy and Checking	64
	Synthesis Constraint Creation	65
	LPF Constraint Creation	67
	Simulation Flow	68
	I/O Assistant Flow	71

	Summary of Changes from ispLEVER	73
Chapter 7	Working with Tools and Views	75
	Overview	75
	Shared Memory	75
	Cross Probing	75
	View Menu Highlights	76
	Start Page	76
	Reports	77
	Preference Preview	78
	Tools	78
	Spreadsheet View	79
	Package View	83
	Device View	84
	Netlist View	85
	NCD View	87
	IPexpress	88
	Platform Designer	89
	Reveal Inserter	90
	Reveal Analyzer	93
	Floorplan View	97
	Physical View	98
	Logic Block View	99
	Timing Analysis View	99
	LDC Editor	104
	Schematic Editor	105
	Power Calculator	106
	ECO Editor	111
	Programmer	113
	Deployment Tool	115
	Model 300 Programmer	115
	Programming File Utility	116
	Download Debugger	117
	Partition Manager	117
	HDL Diagram	118
	Run Manager	119
	Synplify Pro for Lattice	121
	Active-HDL Lattice Edition	121
	Simulation Wizard	122
	Common Tasks	122
	Controlling Tool Views	122
	Using Zoom Controls	125
	Displaying Tool Tips	125
	Setting Display Options	125
Chapter 8	Tcl Scripting	127
	Overview	127
	Tcl Console	127
	External Tcl Console	128
	Commands	129
	Lattice Diamond Tcl Console	130
	Project Manager	130

	NCD	130
	NGD	131
	HDL Diagram	131
	Reveal Inserter	131
	Reveal Analyzer	132
	Power Calculator	132
	Programmer	132
	Incremental Design Flow	133
	Compile Lattice FPGA Simulation Libraries	133
	Tcl Scripting From Command-Line Shells	133
	Lattice Diamond Example Tcl Script	133
	DOS Script for Running Lattice Diamond Tcl Script	134
	Bash Script for Running Lattice Diamond Tcl Script on Windows	134
	Bash Script for Running Lattice Diamond Tcl Script on Linux	134
Chapter 9	Advanced Topics	137
	Shared Memory Environment	137
	Memory Usage	138
	Clear Tool Memory	138
	Environment and Tool Options	139
	Pin Migration	140
	Batch Tool Operation	141
	Tcl Scripts	141
	Creating Tcl Scripts from Command History	141
	Creating Tcl Scripts from Scratch	142
	Sample Tcl Script	142
	Running Tcl Scripts	142
	Project Archiving	143
Appendix	File Descriptions	145
	Index	151

Introduction

Lattice Diamond® software is the leading-edge software design environment for cost-sensitive, low-power Lattice FPGA architectures. It is the next-generation replacement for ispLEVER. Lattice Diamond's integrated tool environment provides a modern, comprehensive user interface for controlling the Lattice Semiconductor FPGA implementation process. Its combination of new and enhanced features allows users to complete designs faster, more easily, and with better results than ever before.

This user guide describes the main features, usage, and key concepts of the Lattice Diamond design environment. It should be used in conjunction with the Release Notes and reference documentation included with the product software. The Release Notes document is also available on the Lattice Web site and provides a list of supported devices.

Lattice Diamond Overview

Lattice Diamond uses an expanded project-based design flow and integrated tool views so that design alternatives and what-if scenarios can easily be created and analyzed. The new *Implementations* and *Strategies* concepts provide a convenient way for users to try alternate design structures and manage multiple tool settings.

System-level information—including process flow, hierarchy, and file lists—is available, along with integrated HDL code checking and consolidated reporting features.

A fast Timing Analysis loop, ECO Editor, and Programmer provide new capabilities in the integrated framework. The cross-probing feature and the new shared memory architecture ensure fast performance and better memory utilization.

Lattice Diamond is highly customizable and provides Tcl scripting capabilities from its built-in console or from an external shell.

User Guide Organization

This user guide contains all the basic information for using the Lattice Diamond software. It is organized in a logical sequence from introductory material, through operational descriptions, to advanced topics.

Key concepts and work flows are explained in “Design Environment Fundamentals” on page 15 and “Lattice Diamond Design Flow” on page 59.

Basic operation of the design environment is described in “User Interface Operation” on page 23.

Other parts of the book provide greater detail and practical usage information. The chapter “Working with Projects” on page 43 shows how to set up project implementations and strategies. “Working with Tools and Views” on page 75 describes the many tool views available.

Getting Started

This chapter explains how to run Lattice Diamond and open or create a project. It includes instructions for importing a project from ispLEVER and explains key differences between Lattice Diamond and ispLEVER.

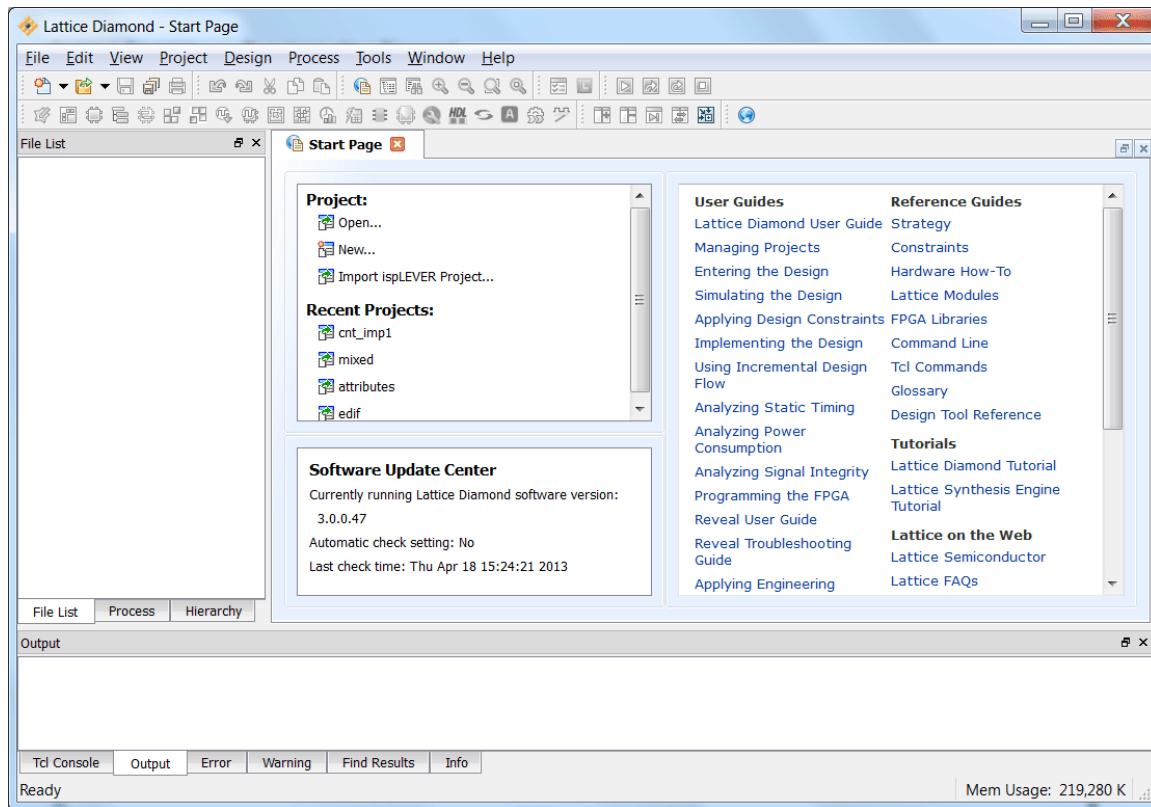
For more information about project fundamentals, see the chapters “Design Environment Fundamentals” on page 15 and “Working with Projects” on page 43.

Prerequisites

It is assumed you have already installed the Lattice Diamond software and product license. See the Lattice Diamond Installation Notice for complete information on product installation.

Running Lattice Diamond

To run Lattice Diamond, select **Lattice Diamond** from the installation location. This opens the default Start Page.

Figure 1: Default Start Page

Note

If you are using Windows 7, you can use the "pin" command from the Start menu to place a Lattice Diamond shortcut on the Start menu or the Taskbar.

Do not use the "pin" command that is available from the Taskbar while Diamond is running. If you do so, the shortcut will fail when you try to use it to launch Diamond.

Creating a New Project

The New Project wizard steps you through the process of creating a new project, allowing you to name the project and its implementation, add source files, and select a target device.

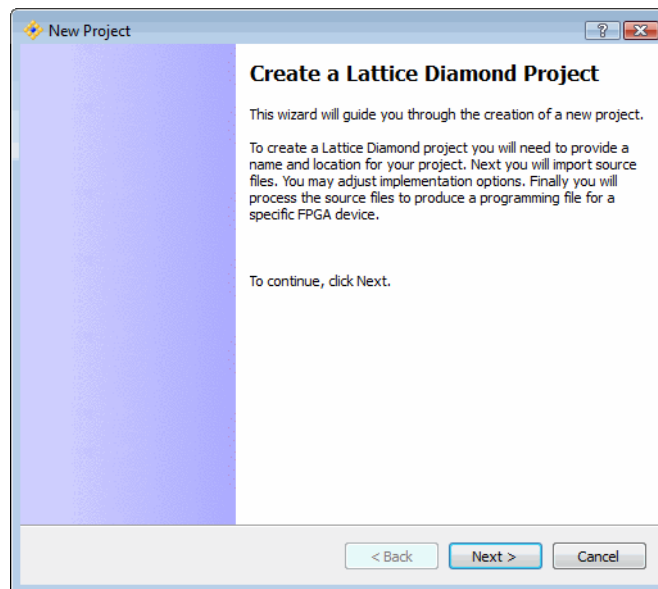
You can open the New Project Wizard using one of the following methods:

- ▶ On the Start Page, select **New** in the Project pane.
- ▶ From the File menu, choose **New > Project**

Several example project design files are included in Lattice Diamond. The following example procedure shows how to create a new project using the “mixedcounter” example.

On the Start Page, select Project > New.

Figure 2: Create a New Project



Click **Next** to open the Project Name dialog box.

Click **Browse** to open the Browse for Folder dialog box. Navigate to the Lattice Diamond examples directory and select the **mixed_mode** folder, as shown:

Figure 3: Project Name

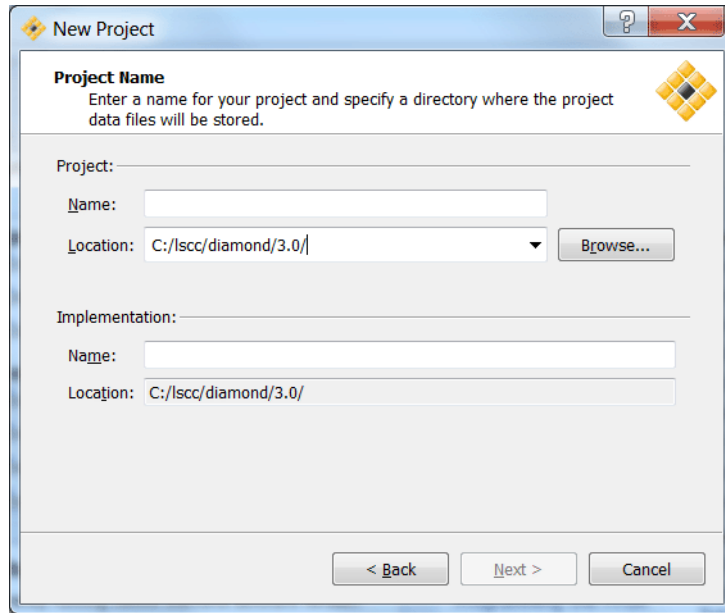
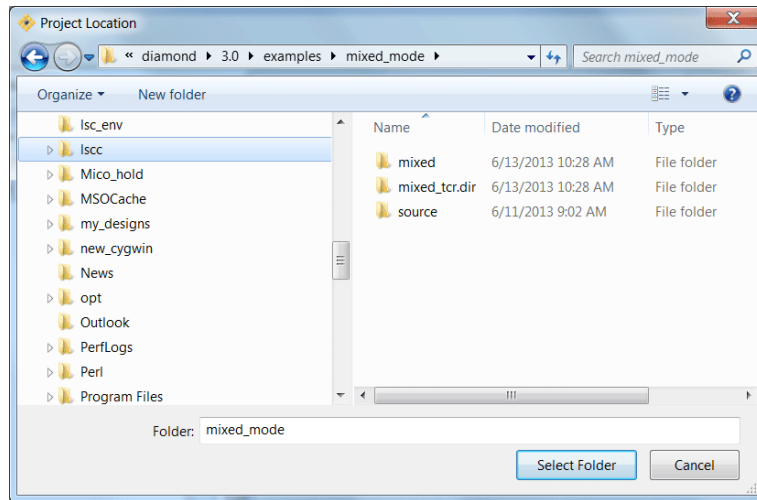
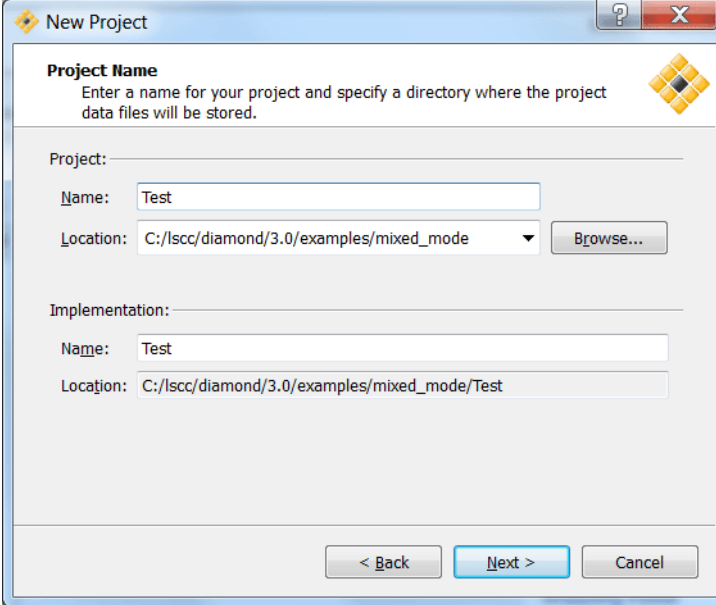


Figure 4: Project Browser



Enter a Project Name. Notice that the implementation is given the same name by default, but this is not required. For this example, we will leave them the same, naming both the project and the initial implementation “Test.”

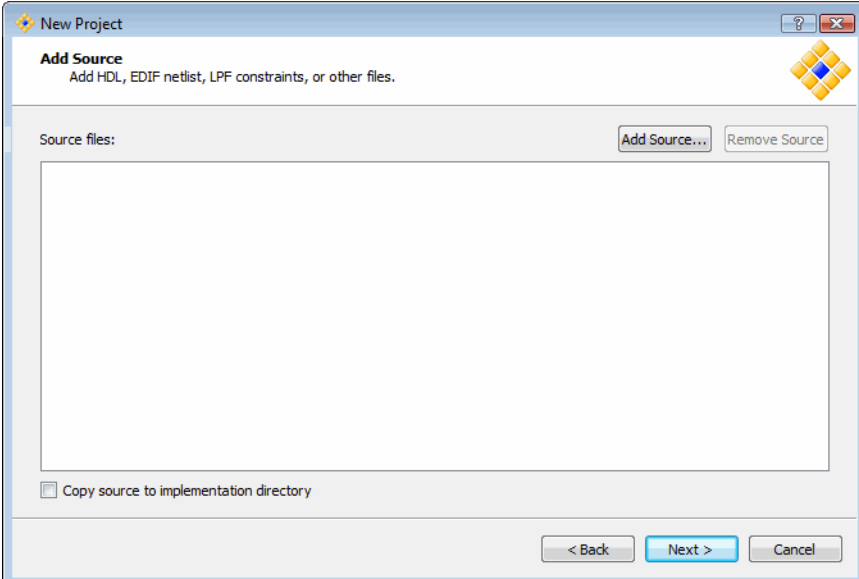
Figure 5: Enter Project and Implementation Name



The screenshot shows a dialog box titled "New Project". It has a "Project Name" section with the instruction "Enter a name for your project and specify a directory where the project data files will be stored." Below this, there are two sections: "Project:" and "Implementation:". The "Project:" section has a "Name:" field containing "Test" and a "Location:" dropdown menu showing "C:/lsc/diamond/3.0/examples/mixed_mode" with a "Browse..." button. The "Implementation:" section has a "Name:" field containing "Test" and a "Location:" field containing "C:/lsc/diamond/3.0/examples/mixed_mode/Test". At the bottom, there are three buttons: "< Back", "Next >" (highlighted in blue), and "Cancel".

Click **Next** to open the Add Source dialog box.

Figure 6: Add Source



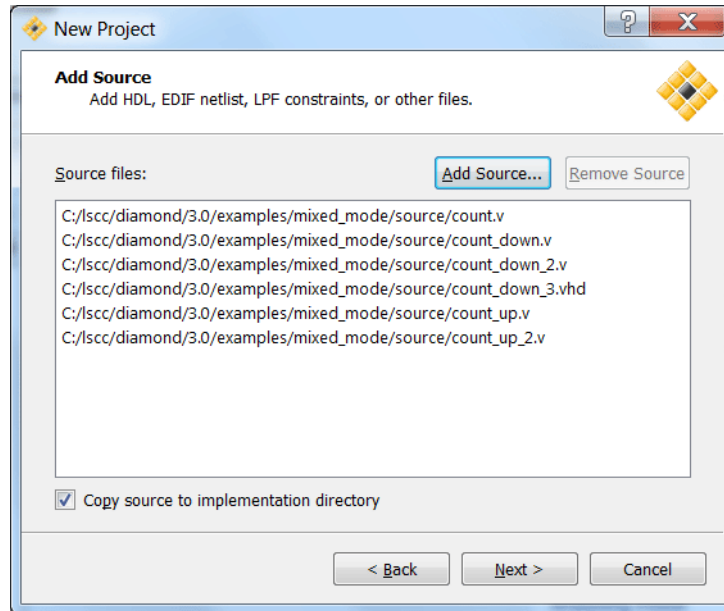
The screenshot shows a dialog box titled "New Project" with the "Add Source" section. The instruction is "Add HDL, EDIF netlist, LPF constraints, or other files." Below this, there is a "Source files:" label and a large empty list box. To the right of the list box are "Add Source..." and "Remove Source" buttons. At the bottom left, there is a checked checkbox labeled "Copy source to implementation directory". At the bottom right, there are three buttons: "< Back", "Next >" (highlighted in blue), and "Cancel".

From this dialog box, you can add Verilog or VHDL source files, EDIF netlist files, LPF constraint files, schematic, debug and analysis files or any other project files. Diamond takes the source files and places them into the correct folders for the new project.

Click **Add Source** to open a file browser in the project example location.

Open the “source” folder, select all Verilog and VHD files and click **Open** to add the files to the project.

Figure 7: Source

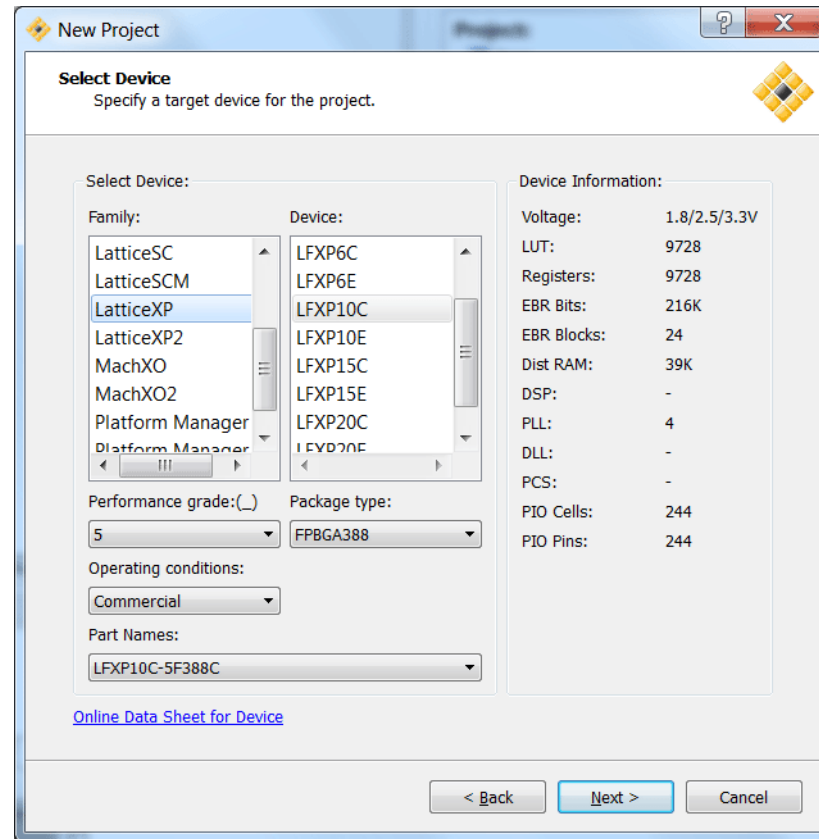


This process of browsing and adding source files can be done as many times as needed before going to the next step.

Notice the option to “Copy source to implementation directory.” Select this option if you want to copy the external source files to the project’s initial implementation. If you prefer to reference these files, clear this option. See “Implementations” on page 45 for further details.

Click **Next** to select the device.

Figure 8: Select Device

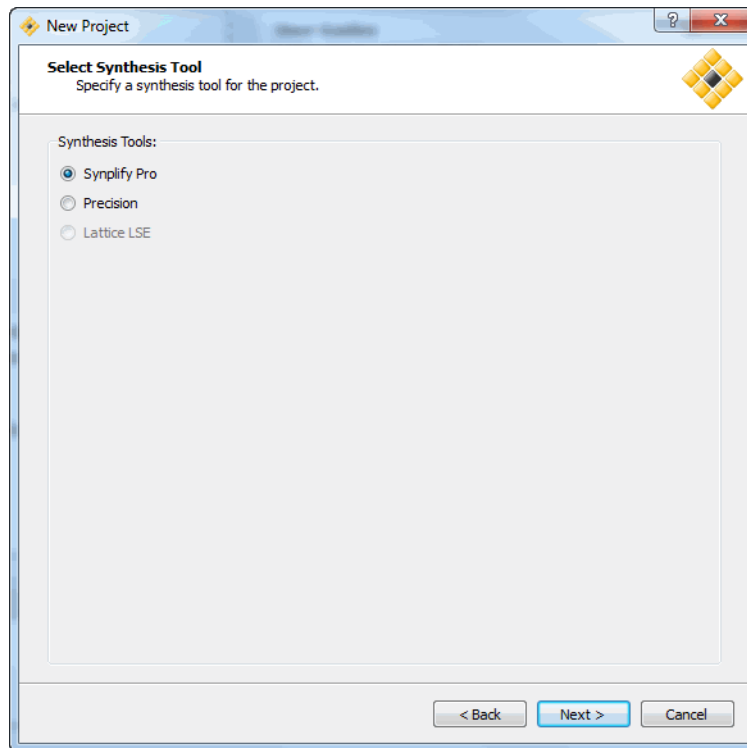


In this step you can select the target device, performance grade, package type, operating conditions and part name. For our example, we will use the default settings.

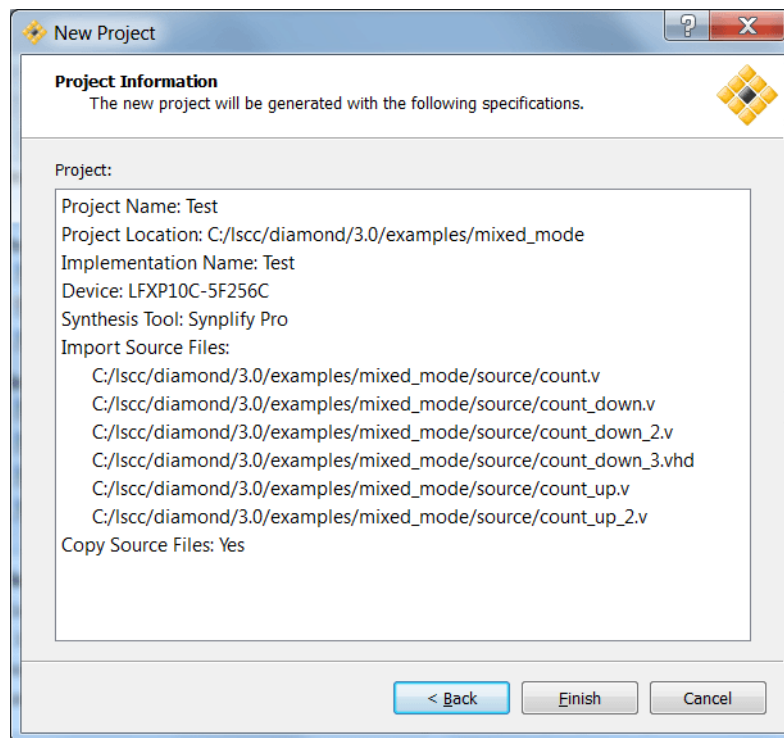
Click **Next** to open the Select Synthesis Tool dialog box.

Select the synthesis tool that you want to use.

If you are designing for MachXO, MachXO2, or Platform Manager, you have the option of using Lattice Synthesis Engine (LSE) as your synthesis tool instead of Synplify Pro for Lattice or another third-party synthesis tool. LSE is a synthesis tool custom-built for Lattice products and fully integrated with Diamond.

Figure 9: Select Synthesis Tool

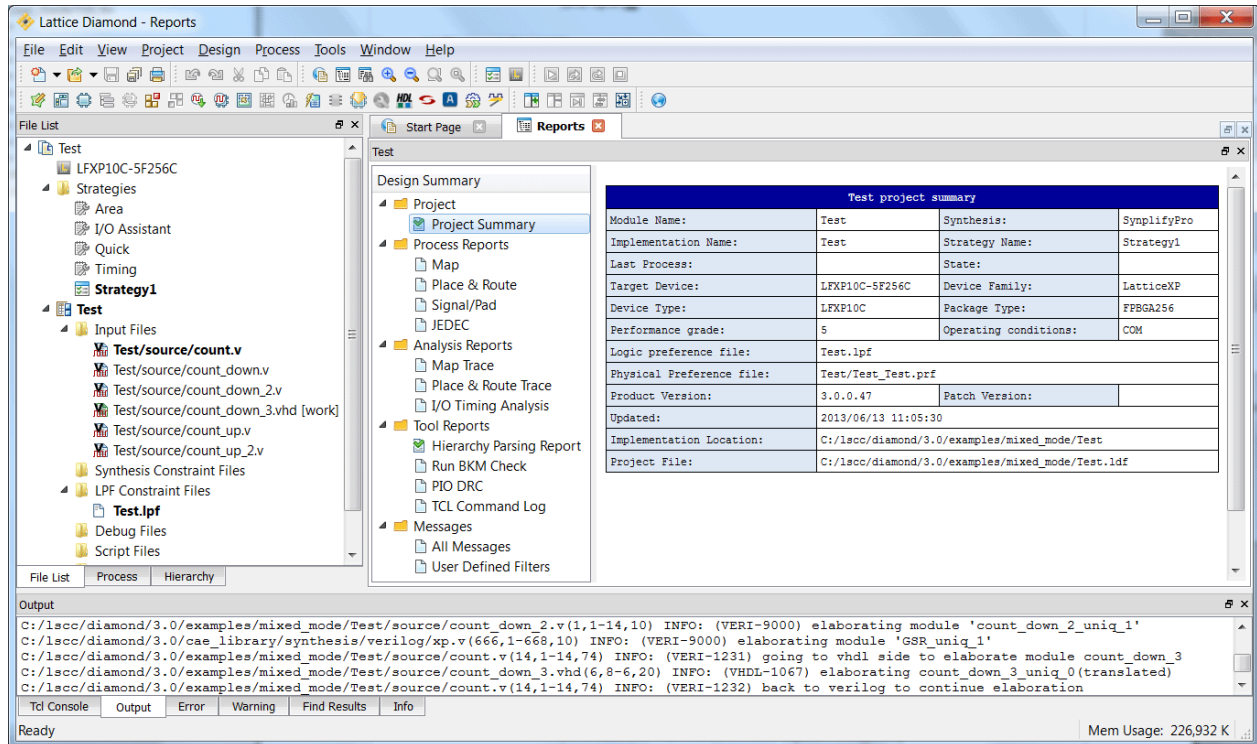
Click **Next** to open the Project Information summary.

Figure 10: Project Summary

At this step or any other step in the process, you can click the **Back** button to review or change your selections.

Click **Finish**. The newly created project, shown in Figure 11, is now created and open.

Figure 11: Opened Project



Select the **File List** tab under the left pane, to view the Test project file list. Select the **Process** tab, to view the design flow processes and status.

To close a project, choose **File > Close Project**.

Opening an Existing Project

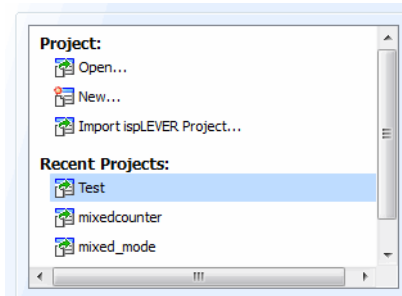
Use one of the following methods to open an existing Lattice Diamond project:

- ▶ On the Start Page, select **Open** from the Project pane.
- ▶ From the File menu, choose **Open > Project**.
- ▶ On the Start Page, select the desired project from the Recent Projects list. Alternatively, choose a recent project from the **File > Recent Projects** menu.

You can use the Options dialog box to increase the number of projects that are shown in the Recent Projects list and to automatically load the previous project at startup. Choose **Tools > Options** to open the dialog box. To increase the number of recent projects listed, select **General** from the

Environment section, and then enter a number for “Maximum items shown in Recent items list.” To automatically open the previous project during startup, select **Startup** from the Environment section, and then choose **Open Previous Project** from the “At Lattice Diamond startup” menu.

Figure 12: Open Recent Project

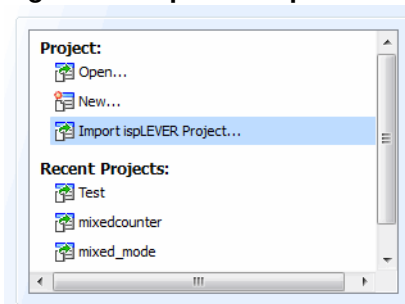


Importing an ispLEVER Project

Use one of the following methods to import an ispLEVER project into Lattice Diamond:

- ▶ On the Start Page select **Import ispLEVER Project** from the Project pane.
- ▶ From the File menu, choose **Open > Import ispLEVER Project**

Figure 13: Import an ispLEVER Project



The file browser applies a ***.syn** file filter to help you find ispLEVER project files. The ispLEVER project is converted to a Lattice Diamond project.

The import/conversion process has the following limitations:

- ▶ Any **.ngo** files will need to be manually copied into the Lattice Diamond project if these files were originally copied into the ispLEVER project; for example, **.ngo** files that were copied from Lattice IP generation.
- ▶ All **.lpc** files are replaced with **.ipx** files in Lattice Diamond. You will need to re-generate your IP by double-clicking the **.lpc** file name. The wizard will then open and help you generate the new **.ipx** file, replacing the **.lpc** file.

- ▶ If you select the “Copy source to Implementation directory” option, the following additional limitations will apply:
 - ▶ Verilog include files specified within the Verilog source files will not be copied.
 - ▶ Files associated with IPexpress Module .lpc files (such as .v, .txt, .ngo) will not be copied.
 - ▶ User-specified schematic symbols (.sym) will not be copied.

Next Steps

After you have a project opened in Lattice Diamond, you can go sequentially through the rest of this user guide to learn how to work with the entire design environment, or you can go directly to any topics of interest.

- ▶ The chapters “Design Environment Fundamentals” on page 15 and “Lattice Diamond Design Flow” on page 59 provide explanations of key concepts.
- ▶ “User Interface Operation” on page 23 provides descriptions of the functions and controls that are available in the Diamond environment.
- ▶ The chapters “Working with Projects” on page 43 and “Working with Tools and Views” on page 75 explain how to run processes and use the design tools.
- ▶ “Tcl Scripting” on page 127 provides an introduction to the scripting capabilities available, plus command-line shell examples.
- ▶ “Advanced Topics” on page 137 provides further details about environment options, shared memory, and Tcl scripting.

Differences from ispLEVER

There are a number of differences between Lattice Diamond and ispLEVER. Key differences, especially regarding how projects are managed, include the following:

- ▶ ispLEVER has multiple project types, but there is only one Diamond project type. In ispLEVER, you need different projects types for each type of source; for example, one project for Verilog and a different project for VHDL. In Lattice Diamond, the project can include sources of different types. For example, one Lattice Diamond project can contain both Verilog and VHDL source files.
- ▶ Lattice Diamond includes implementations and strategies. These do not exist in ispLEVER.
- ▶ ispLEVER parses source file hierarchy when a project is opened, and it will question the existence of multiple top-level modules. Lattice Diamond does not display hierarchy by default (though it can be configured to do so), and you need to set the top-level design unit if multiple top-level modules exist.

- ▶ ispLEVER consists of a number of separate tools. Lattice Diamond is an integrated tool environment.
- ▶ All of the Lattice Diamond tool views share a common memory image of design data. This means that changes to the design data are seen by all tools.
- ▶ Lattice Diamond projects do not allow simulation test benches as source; only modules are contained within a Lattice Diamond project.
- ▶ Lattice Diamond 1.3 and later supports the ability to mark individual source files for simulation, synthesis, or both. This supports multiple file test benches and modules with different representations for simulation and synthesis. ispLEVER only supported a single test bench file for simulation and did not support different representations for the same module.

Design Environment Fundamentals

This chapter provides background and discussion on the technology and methodology underlying the Lattice Diamond software design environment. Important key concepts and terminology are defined.

Overview

Understanding some of the fundamental concepts behind the Lattice Diamond framework technology will increase your proficiency with the tool and allow you to quickly come up to speed on its use.

Lattice Diamond is a next-generation software design environment that uses a new project-based methodology. A single project can contain multiple implementations and strategies to provide easily managed alternate design structures and tool settings.

The process flow is managed at a system level with run management controls and reporting. Context-sensitive views ensure that you only see the data that is available for the current state in the process flow.

The shared memory technology enables many of the advanced functions in Lattice Diamond. Easy cross-probing between tool views and faster process loops are among the benefits.

Note

You can run multiple instances of Lattice Diamond, by loading Lattice Diamond multiple times. This enables you to run different Diamond projects at the same time. However, you must not load the same project in more than one Lattice Diamond software instance, because this can cause conflicts in the software.

You can also run Diamond remotely. Refer to the Lattice Diamond Installation Notice for more information.

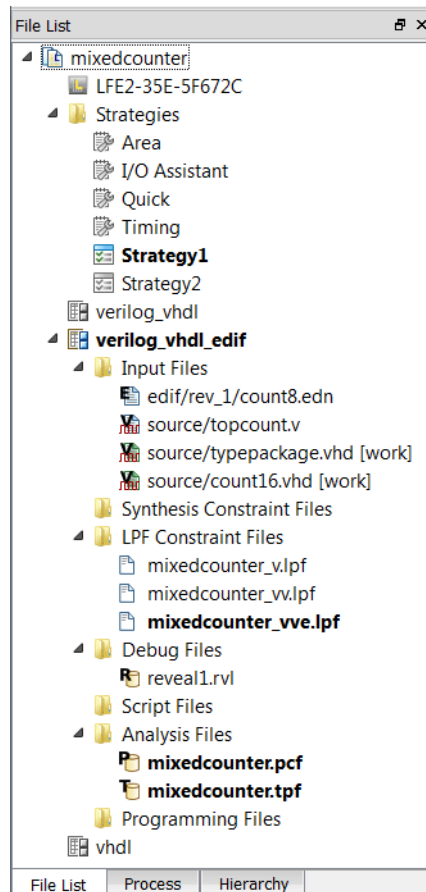
Project-Based Environment

A project in Lattice Diamond consists of HDL source files, EDIF netlist files, synthesis constraint files, LPF constraint files, Reveal debug files, script files for simulation, analysis files for power calculation and timing analysis, and programming files.

It also includes settings for the targeted device and the different tools. The project data is organized into implementations, which define the project structural elements, and strategies, which are collections of tool settings.

The following File List shows the items in a sample project.

Figure 14: File List



Each item that is displayed in **bold** means that it has been selected as the active item for an implementation. An implementation displayed in bold means that it has been selected as the currently active implementation for the project. Your project must have one active implementation, and the implementation must have one active strategy. Optional items, such as Reveal hardware debugger files, can be set as active or inactive. This differs

from ispLEVER, where the existence of Reveal debugger files means that debug is active.

The project is the top-level organizational element in Lattice Diamond, and it can contain multiple implementations and multiple strategies. This enables you to try different design approaches within the same project. If you want to have a Verilog version of your design, you will make an implementation that consists of only the Verilog source files. If you want another version of the design with primarily Verilog files but an EDIF netlist for one module, you will create a new implementation using the Verilog and EDIF source files. It will be the same project and design, but with a different set of modular blocks.

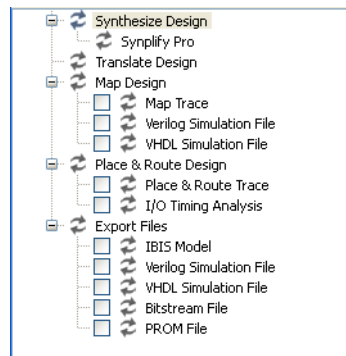
Similarly, if you want to try different implementation tool options, you can create a new strategy with the new option values.

You manage these multiple implementations and strategies for your project by setting them as active. There can only be one active implementation with its one active strategy at a time.




Process Flow

The Process View provides a system-level overview of the FPGA design flow. Each major step in the design process is shown, along with an icon that indicates its status. In the Map and Place & Route sections, you can select optional subtasks to be run every time. These selections are saved on a project basis. In the Export Files section, you can select the models or files that you want to be exported with the Export Files process. For example, if Bitstream File is checked, it will be generated and exported; if it is not checked, it will not be generated and exported.

Figure 15: Process Flow



The process status icons are defined as follows:

-  Process in initial state
-  Process completed successfully
-  Process completed with warnings, see Warning output