Lattice Diamond User Guide

Lattice Semiconductor Corporation
5555 NE Moore Court
Hillsboro, OR 97124
(503) 268-8000

Diamond 1.4 November 2011

Copyright

Copyright © 2011 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, E^2CMOS, Extreme Performance, FlashBAK, FlexiClock, flexiFlash, flexiMAC, flexiPCS, FreedomChip, GAL, GDX, Generic Array Logic, HDL Explorer, IPexpress, ISP, ispATE, ispClock, ispDOWNLOAD, ispGAL, ispGDS, ispGDX, ispGDXV, ispGDX2, 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, LatticeMico8, LatticeMico32, LatticeSC, LatticeSCM, LatticeXP, LatticeXP2, MACH, MachXO, MachXO2, MACO, ORCA, PAC, PAC-Designer, PAL, Performance Analyst, Platform Manager, ProcessorPM, PURESPEED, Reveal, 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.
<*Italic*>	Variables in commands, code syntax, and path names.
Ctrl+L	Press the two keys at the same time.
`Courier`	Code examples. Messages, reports, and prompts from the software.
`...`	Omitted material in a line of 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

1

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, modules, 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 61.

Basic operation of the design environment is described in “User Interface Operation” on page 21.

Other parts of the book provide greater detail and practical usage information. The chapter “Working with Projects” on page 45 shows how to set up project implementations and strategies. “Working with Tools and Views” on page 77 describes the many tool views available.

2

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 45.

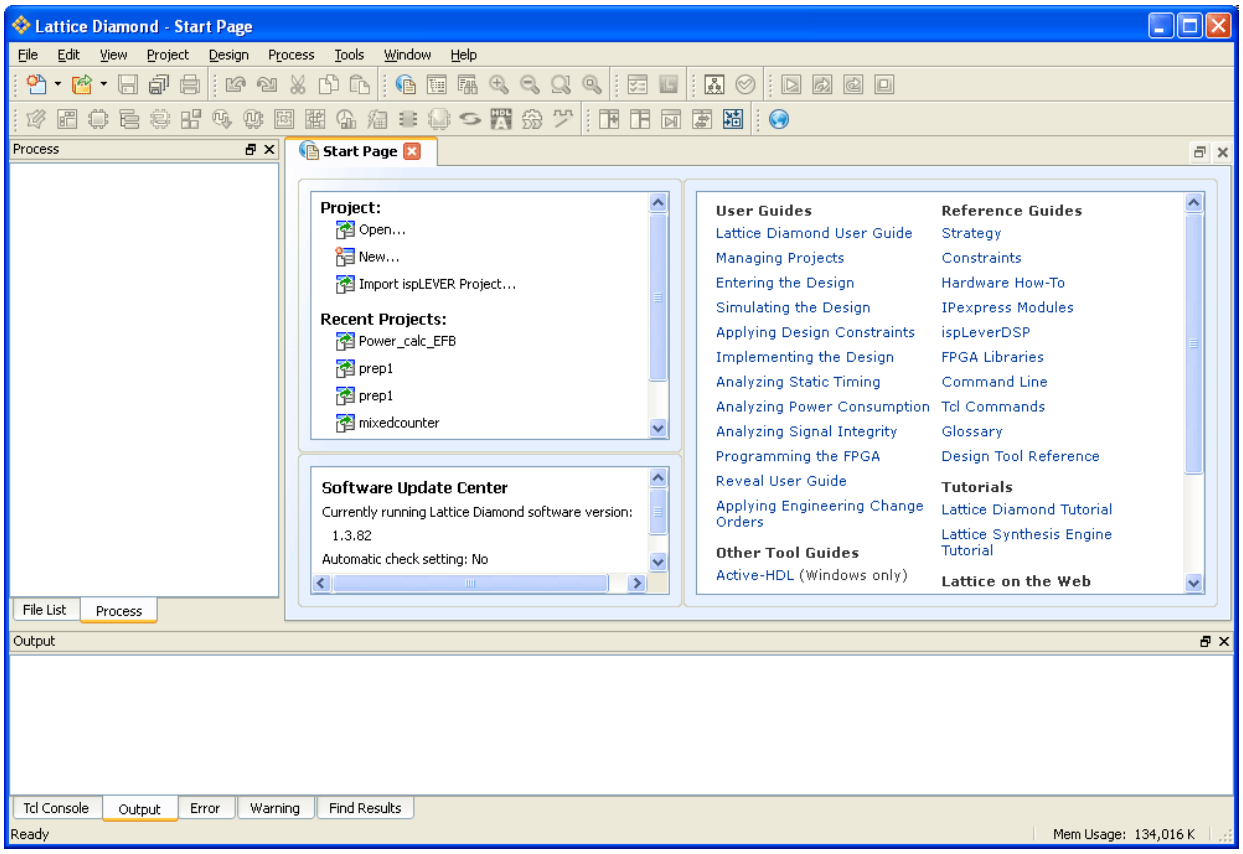
Prerequisites

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:

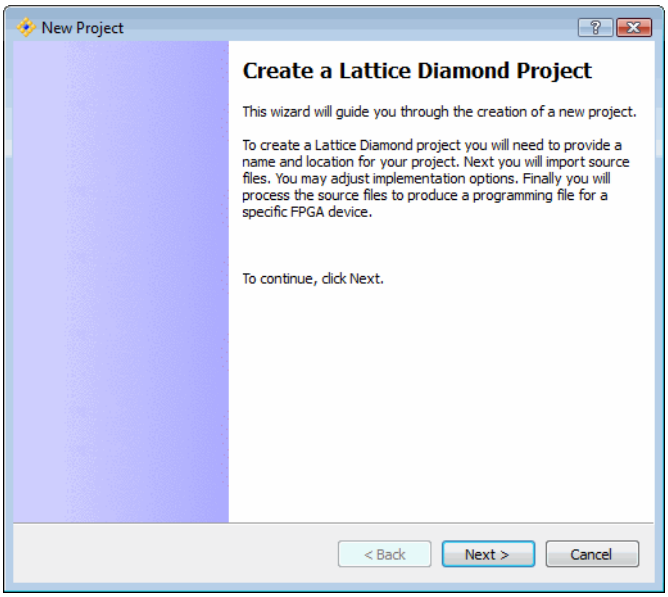
- On the Start Page, select **New** in the Project pane.
- From the File menu, choose **New > Project**

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**.

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

Figure 2: Create a New Project



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

Figure 3: Project Name

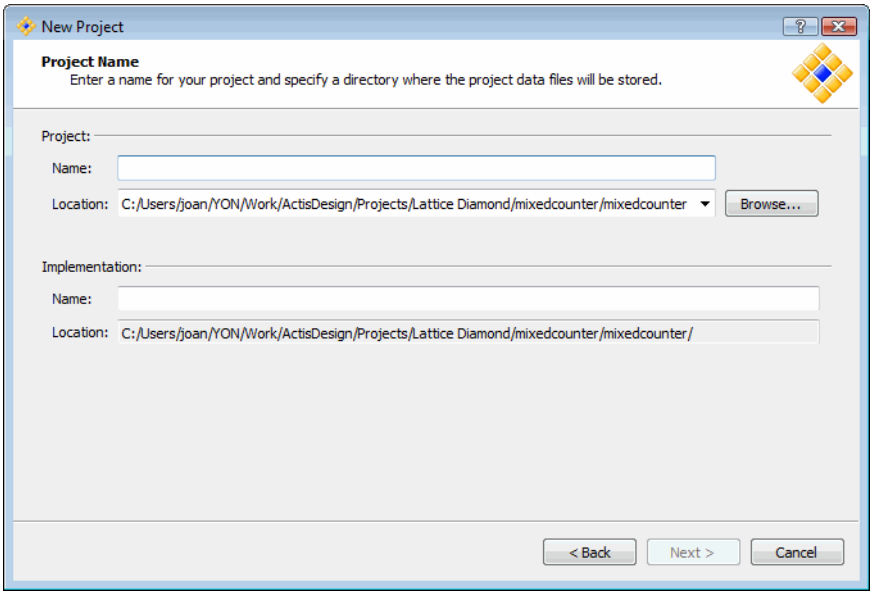
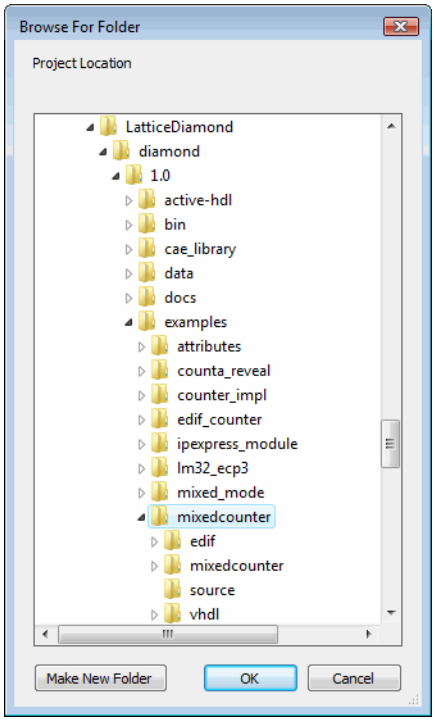
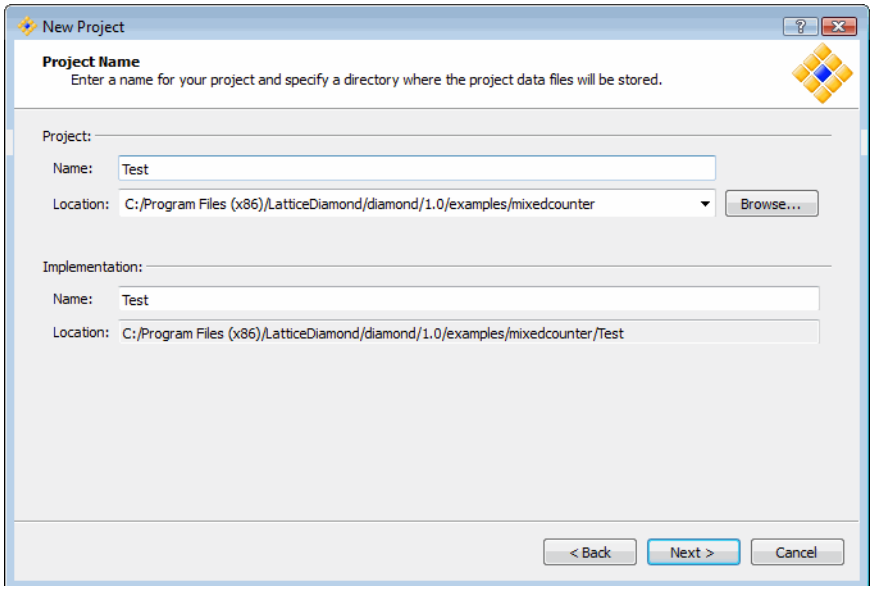


Figure 4: Project Browser



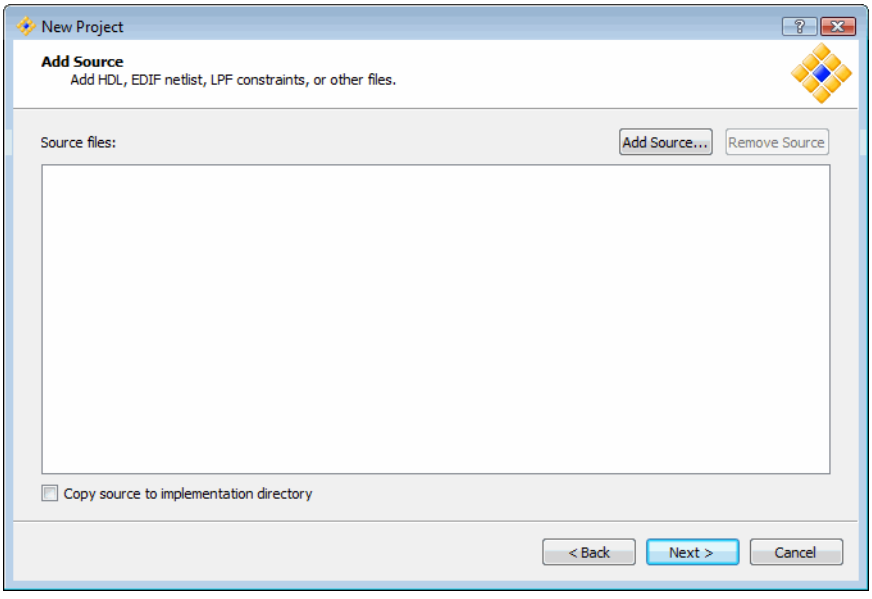
Enter a Project Name. Notice that the implementation is given the same name by default, but this is not required. For this example, we will leave them the same, naming both the project and the initial implementation “Test.”

Figure 5: Enter Project and Implementation Name



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

Figure 6: Add Source



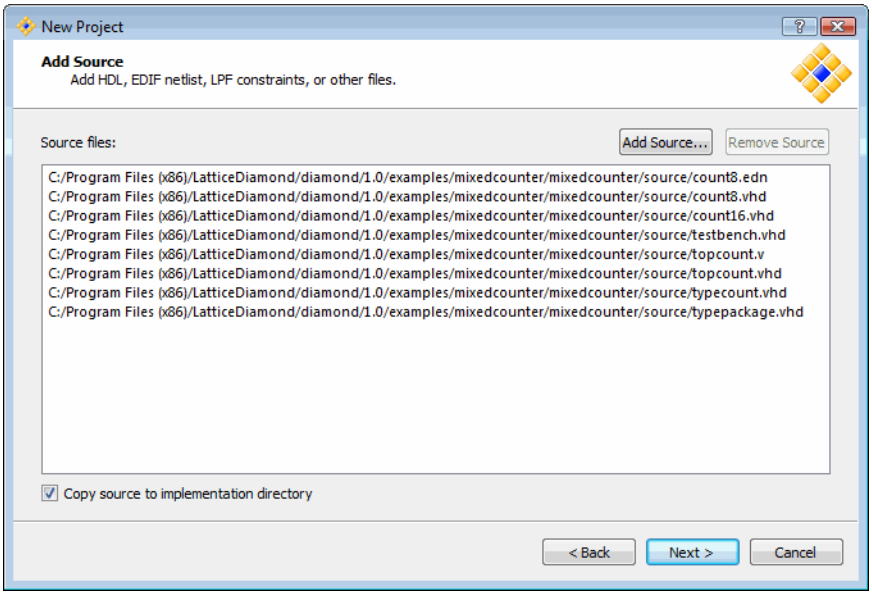
From this dialog box, you can add Verilog or VHDL source files, EDIF netlist files, LPF constraint files, schematic, debug and analysis files or any other

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 VHDL files, and click **Open** to add the files to the project.

Figure 7: Source

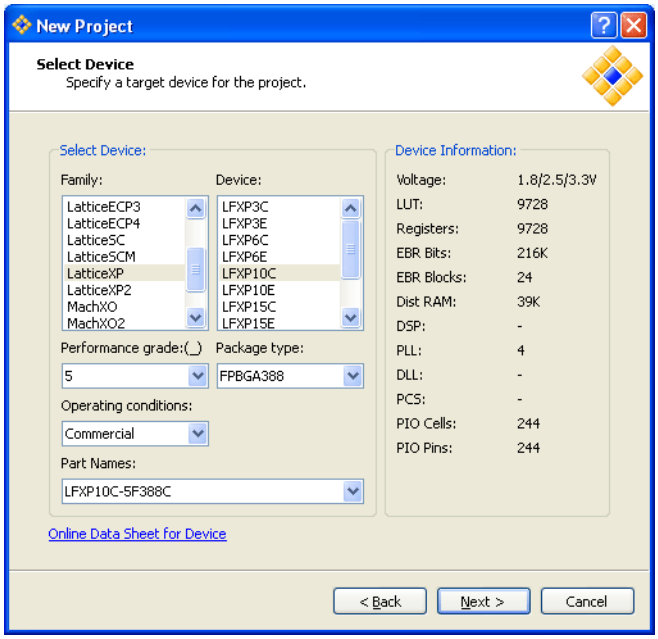


This process of browsing and adding source files can be done as many times as needed before going to the next step.

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 46 for further details.

Click **Next** to select the device.

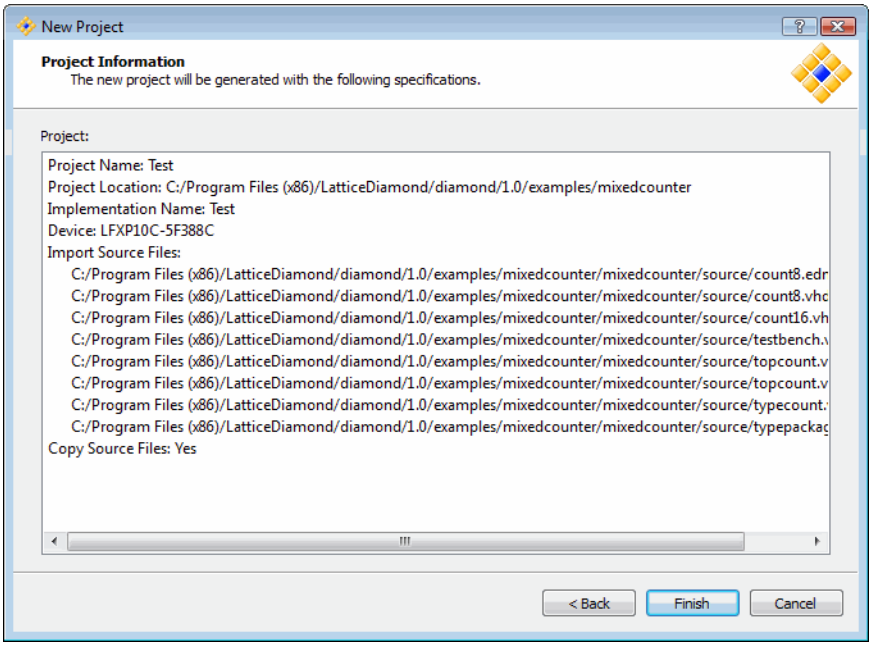
Figure 8: Select Device



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 see the Project Information summary.

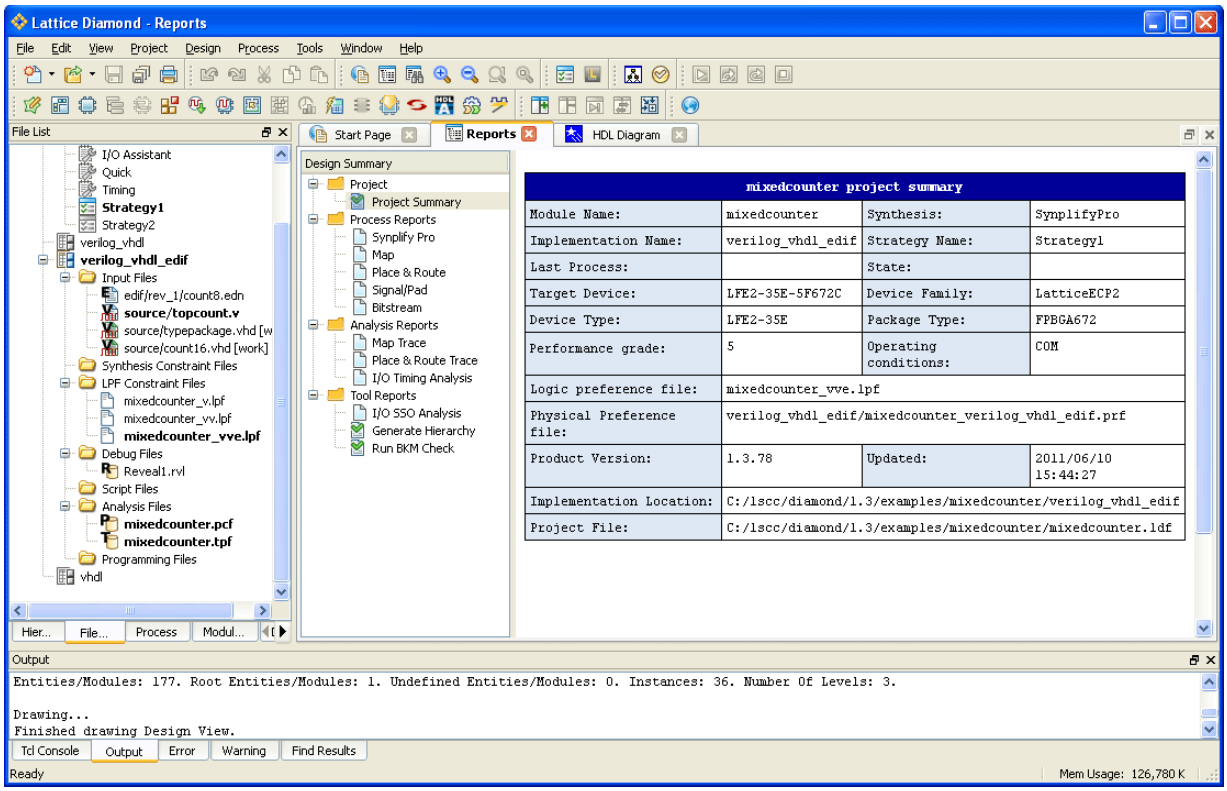
Figure 9: Project Summary



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 is now created and open.

Figure 10: Opened Project



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

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

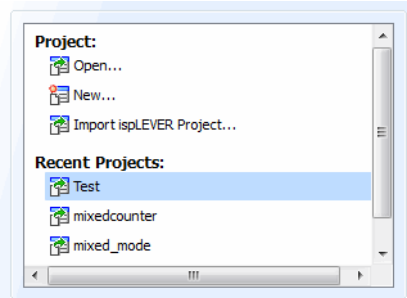
Opening an Existing Project

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 11: Open Recent Project

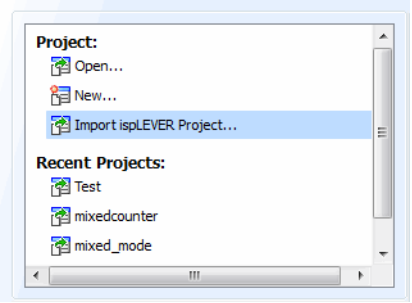


Importing an ispLEVER Project

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 12: 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 61 provide explanations of key concepts.
- “User Interface Operation” on page 21 provides descriptions of the functions and controls that are available in the Diamond environment.
- The chapters “Working with Projects” on page 45 and “Working with Tools and Views” on page 77 explain how to run processes and use the design tools.
- “Tcl Scripting” on page 123 provides an introduction to the scripting capabilities available, plus command-line shell examples

- “Advanced Topics” on page 131 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 testbenches 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 testbenches and modules with different representations for simulation and synthesis. ispLEVER only supported a single testbench file for simulation and did not support different representations for the same module.

3

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, as this can cause conflicts in the software.

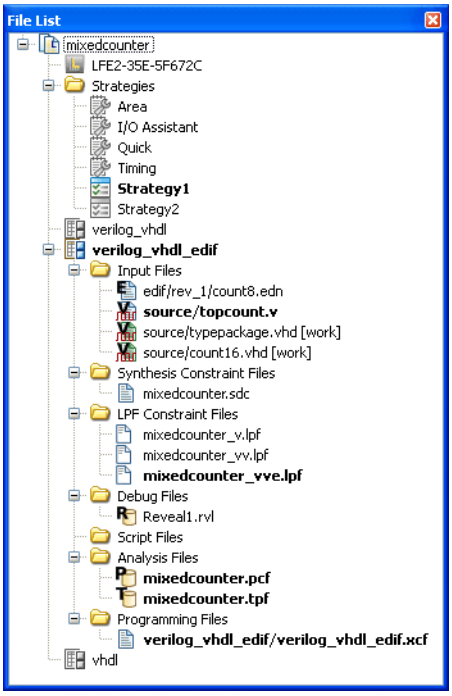
You can also run Diamond remotely. Refer to the Lattice Diamond Installation Notice for more information.

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 13: File List



The items in **bold** are active. You must have one active implementation, and the implementation must have one active strategy. Optional items, such as Reveal hardware debugger files, can be set 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. 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.

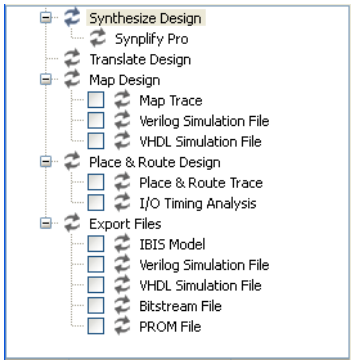
Similarly, if you want to try different implementation tool options, you can create a new strategy with the new option values.

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 portion, 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 14: Process Flow

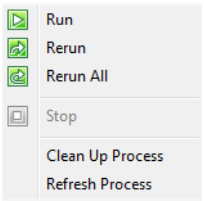


The process status icons are defined as follows:

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

Right-clicking a process opens the controls for running, stopping, cleaning up or refreshing that process.

Figure 15: Process Run Pop-up Menu

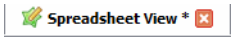


Shared Memory

Lattice Diamond uses a shared memory architecture. All tool and data views are looking at the same design data at any point in time. This means that when you change a data element in one view of your design, all other views will see the change, whether they are active or not.

When project data has been changed but not yet saved, an asterisk * is displayed in the title tab of the view.

Figure 16: Title Tab with Changed Memory Indication



Notice that the asterisks indicating changed data will appear in all views referencing the changed data.

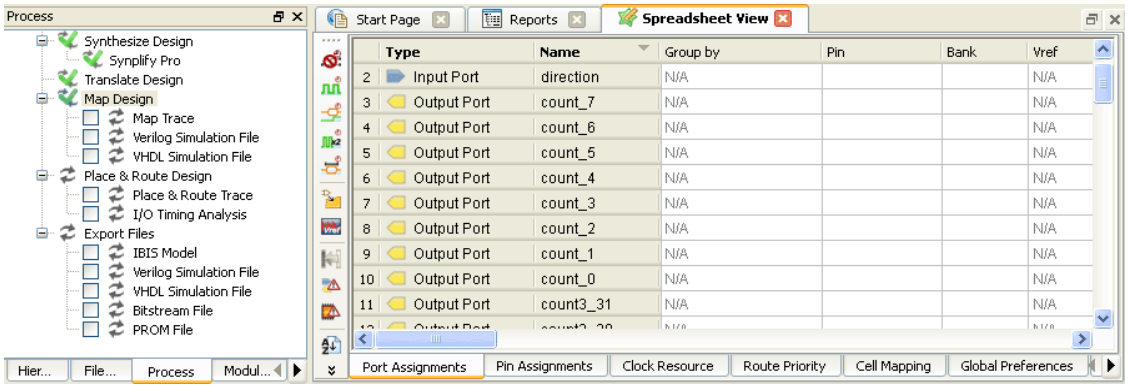
If a tool view becomes unavailable, the Lattice Diamond environment will need to be closed and restarted.

Context-Sensitive Data Views

The data in shared memory reflects the state or context of the overall process flow. This means that views such as Spreadsheet View will display only the data that is currently available, depending on process steps that have been completed.

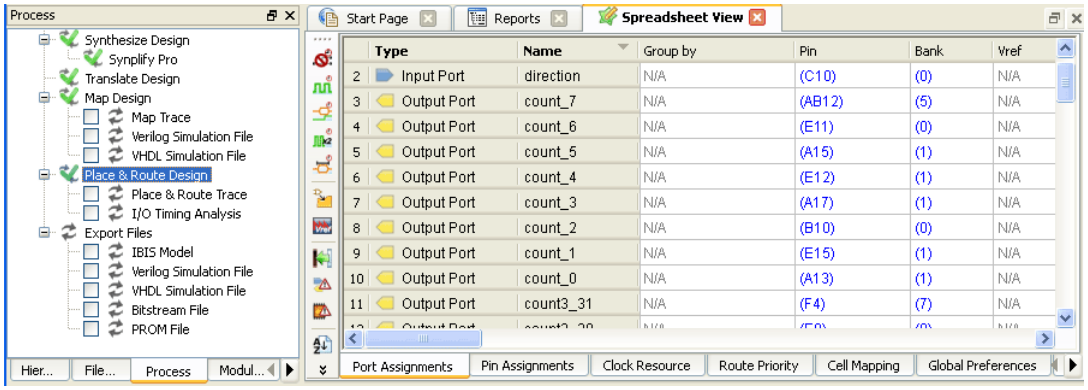
For example, Figure 17 shows that the Process flow has been completed through Map Design but not through Place & Route Design. Therefore, Spreadsheet View shows no pin assignments.

Figure 17: Process Through Map Design



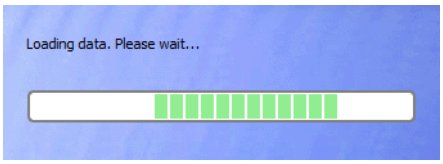
After Place & Route Design has been completed, Spreadsheet View displays the pin assignments.

Figure 18: Process Through Place & Route Design



When you see a “Loading Data” message, it means that a process has been completed and that the shared memory is being updated with new data.

Figure 19: Loading Data



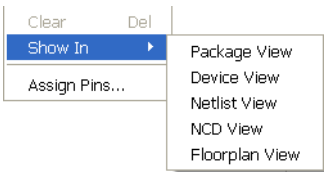
All tool views are dynamically updated when new data becomes available. This means that when you rerun an earlier process while a view is open and displaying data, the view will remain open but dimmed because its data is no longer available.

Cross-Probing

Cross-probing is a feature found in most tool views. Cross-probing allows common data elements to be viewed in multiple tool views.

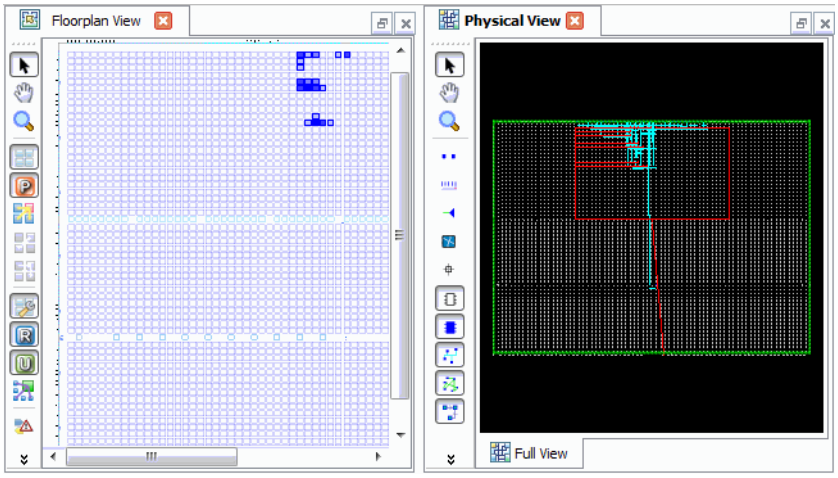
To see how this works, select a pin or signal in one view and right-click it. Select **Show In** to see a list of cross-probing views for the selected element.

Figure 20: Show In Pop-up Menu



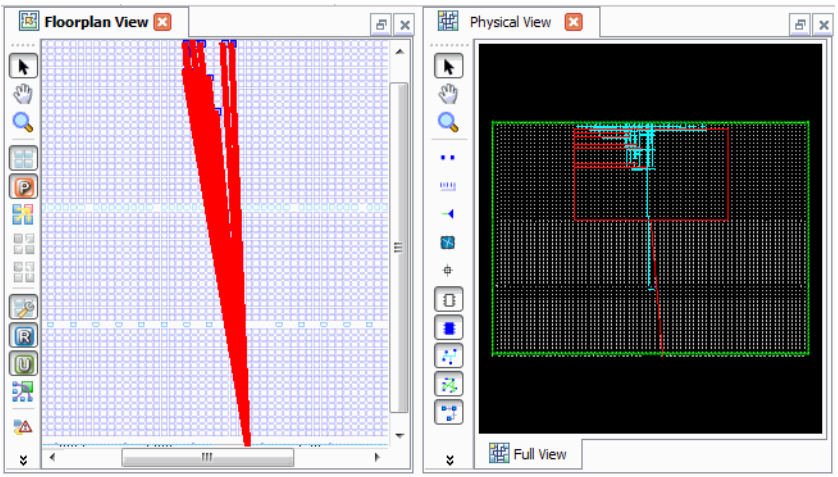
For example, Figure 21 shows a tab group consisting of Floorplan View and Physical View. A signal is selected in Physical View.

Figure 21: Physical View with Selected Signal



When you right-click the selected signal in Physical View and choose **Show In > Floorplan View**, the same selected signal is highlighted on the Floorplan View layout.

Figure 22: Selection in Floorplan and Physical Views



4

User Interface Operation

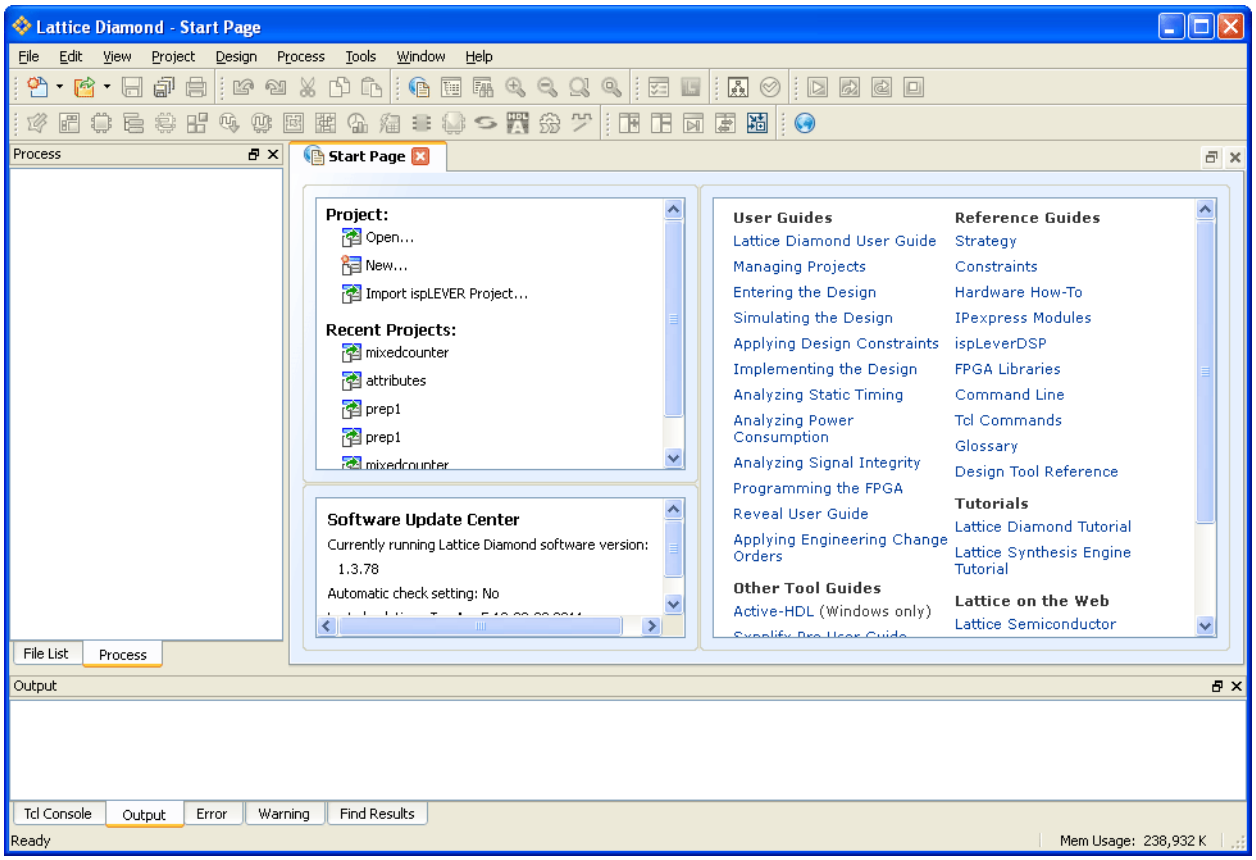
This chapter describes the user interface features, controls and basic operation. Each major element of the interface is explained. The last section in the chapter describes common user interface tasks.

Overview

The Diamond Lattice user interface (UI) provides a comprehensive, integrated tool environment. The UI is very flexible and configurable, enabling you to store layout preferences.

This chapter will take you through the operation of the main elements of the UI, but you should also explore the controls at your own pace. Figure 23 shows the Lattice Diamond main window in the default state.

Figure 23: Main Window



Menus and Toolbars

At the top of the main window is the menu and toolbar area. High-level controls for accessing tools, managing files and projects, and controlling the layout are contained here. All of the functionality in the toolbars is also contained in the menus. The menus also have functions for system, project and toolbar control.

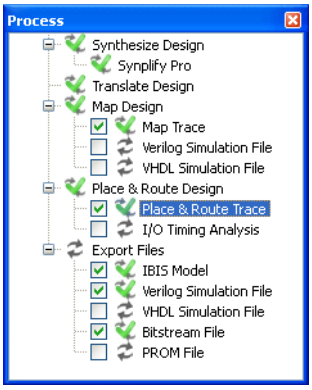
Figure 24: Menu and Toolbar Area

The toolbars are organized into functional sets. The display of each toolbar is controlled in the **View > Toolbars** menu and also by right-clicking in the Menu and Toolbar area. Each toolbar can be repositioned by dragging and dropping it to a new location.

Project Views

In the middle of the main window on the left side is the Project View area. This is where the overall project and process flow is displayed and controlled.

Figure 25: Project View Area



Tabs at the bottom of the Project View allow you to select between the following views:

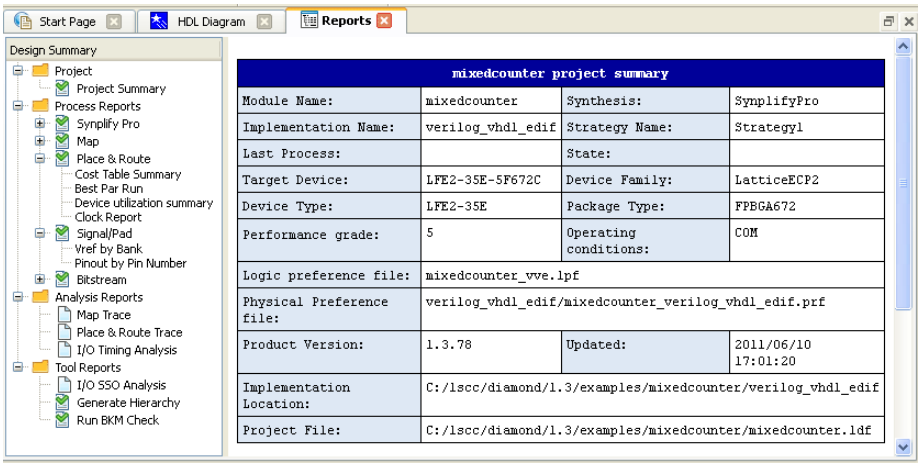
- File List – shows the files in the project organized by implementations and strategies. This is not a hierarchical listing of the design.
- Process – shows the overall process flow and status for each step
- Module library – library of modules in the active implementation of the design
- Dictionary – alphabetical listing of all design elements
- Hierarchy – hierarchical design representation

The File List and Process views are displayed by default. The Module library, Dictionary and Hierarchy views are displayed only when selected or when the **Generate Hierarchy** function has been run and project data is available for these views to display.

Tool View Area

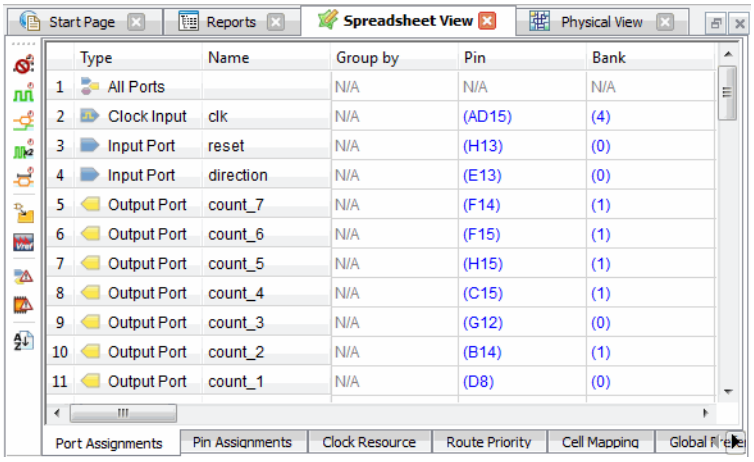
In the middle of the main window on the right side is the Tool View area. This is where the Start Page, Reports View and all the Tool views are displayed.

Figure 26: Tool View Area



Multiple tools can be displayed at the same time. The Window toolbar includes controls for grouping the tool views as well as integrating all tool views back into the main window.

Figure 27: Multiple Tools

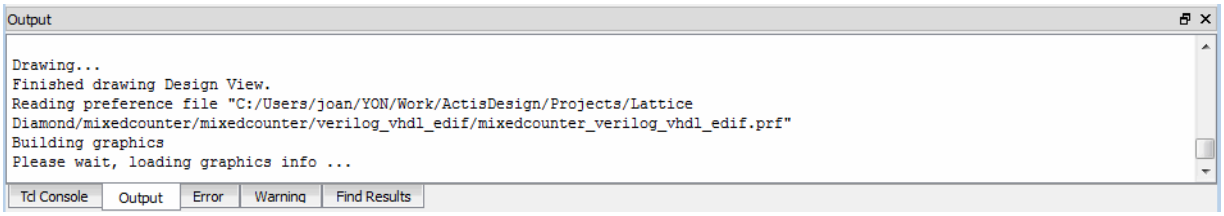


Each tool view is specific to its tool and can contain additional toolbars and multiple panes or windows controlled by additional tabs. The chapter “Working with Tools and Views” on page 77 provides more details about each tool and view.

Output and Tcl Console

Near the bottom of the main window is the Output and Tcl Console area.

Figure 28: Output and Tcl Console Area



Tabs at the bottom of this area allow you to select between Tcl Console, Output, Error, Warning and Find Results. Tool output is automatically sent to the Output tab, and Errors and Warnings are automatically sent to their respective tabs.

Status Information

At the very bottom of the main window is status information. The information shown depends on the position of the mouse pointer, current memory usage, and whether unsaved preference changes are in memory. For more information on memory usage see “Advanced Topics” on page 131.

Figure 29: Status Information

Basic UI Controls

The Lattice Diamond environment is based on modern industry standard user interface concepts. The menus, toolbars, and mouse pointer all behave in familiar ways. You can resize any of the window panes, drag and drop elements, right-click a design element to see available actions, and hold the mouse pointer over an object to view the tooltip.

Each of the Project and Tool views as well as the Outputs and Tcl Console items can be detached from the main window and operated as independent windows. Simply click the detach button in the upper-right corner.

After a view or item has been detached from the main window it can be reattached by one of two methods:

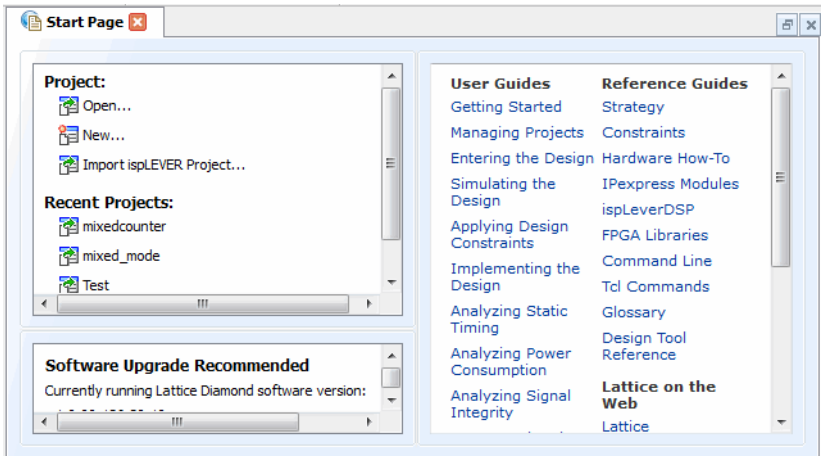
- For Project views, Outputs and the Tcl Console, double-click inside the window title bar, and the window will be re-attached to the main window.
- For Diamond Tool views, single-click the attach button in the upper-right corner, and the window will be re-attached to the main window.

Additionally, an **Integrate All Tools** button is available in the Window toolbar. This control gathers all detached views and reintegrates them inside the main window.

Start Page

The Start Page contains selections for opening projects, hyperlinks to product documentation, and software status and upgrade information. The Start Page appears in the Tool View area by default when Lattice Diamond is first launched.

Figure 30: Start Page

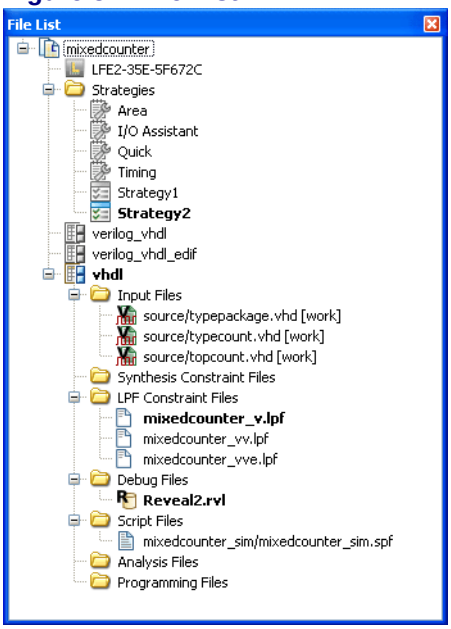


The Start Page can be closed, opened, detached and attached (using the attach button).

File List

The File List is a project view that shows the files in the project, including implementations and strategies. It is not a hierarchical listing of the design, but rather a list of all the design source, configuration and control files that make up the project.

Figure 31: File List



At the top level in the File List is the project name. Directly below the project name is the target device, followed by the strategies, and then the implementations. There must be one active implementation, and it must have one active strategy. Active elements are indicated in **bold**.

You can right-click any file or item in the File List to access a pop-up menu of currently available actions for that item. The pop-up menu contents vary, depending on the type of item selected.

The File List view can be closed, opened, detached, and attached (using the double-click method).

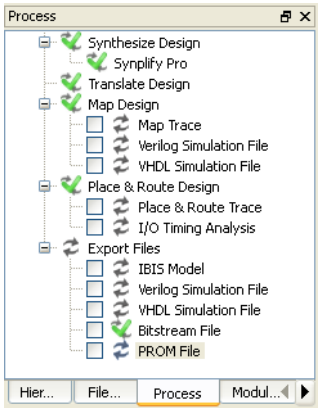
Process

The Process View is a project view that displays the high-level process flow for the project.

The icons to the left of each step indicate the process status and are defined as follows:

- Process in initial state

Figure 32: Process View



Process completed successfully

Process completed with warnings, see Warning output

Process failed, see Error output

You can right-click any step in the Process view to access a pop-up menu of currently available actions for that item.

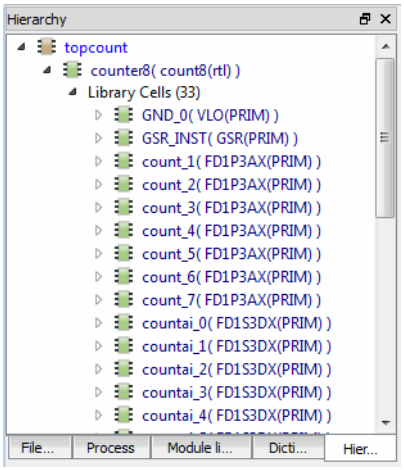
The Process view can be selected, closed, opened, detached, and attached (using the double-click method).

Hierarchy

The Hierarchy View is a project view that displays the design hierarchy. This view is automatically displayed when the Generate Hierarchy function is selected. Along with the Module library and Dictionary views, the Hierarchy View is not displayed by default. To display the hierarchy, choose **View > Show Views > Hierarchy** or right-click in the menu and toolbar area and choose Hierarchy from the pop-up menu.

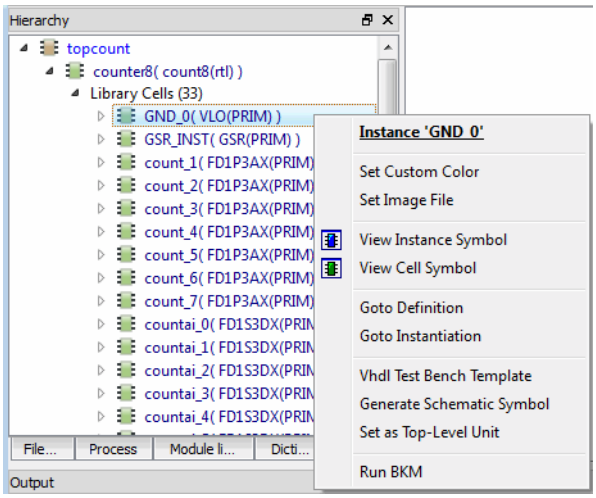
The Hierarchy View remains empty until the hierarchy for the design has been generated. Click the **Generate Hierarchy** button on the toolbar, or choose it from the **Design** menu to generate the hierarchy and populate the Hierarchy View with data. In addition to displaying the HDL Diagram, generating the hierarchy causes the Hierarchy, Module Library and Dictionary views to be displayed.

Figure 33: Hierarchy View



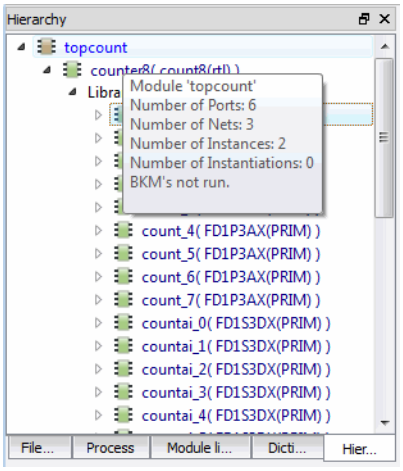
Right-click any of the objects in the Hierarchy View to see its type and name as well as the available actions.

Figure 34: Hierarchy Item Pop-up Menu



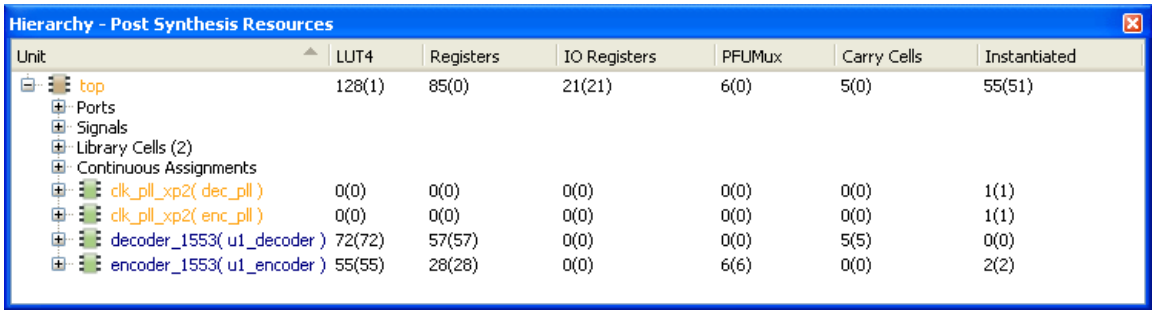
When you hold the mouse pointer over an item in the list, a tooltip description for that item is displayed.

Figure 35: Hierarchy View Item Description



After synthesis, the Hierarchy View displays the calculated resource utilization data, such as LUT4, registers, IO registers. Device resources that would be consumed are shown to the right of each module. The first number in each column is the number of that resource that would be used by that module and all of its submodules. The second number, in parentheses, is the number of that resource that would be used by just that module, not including any of its submodules.

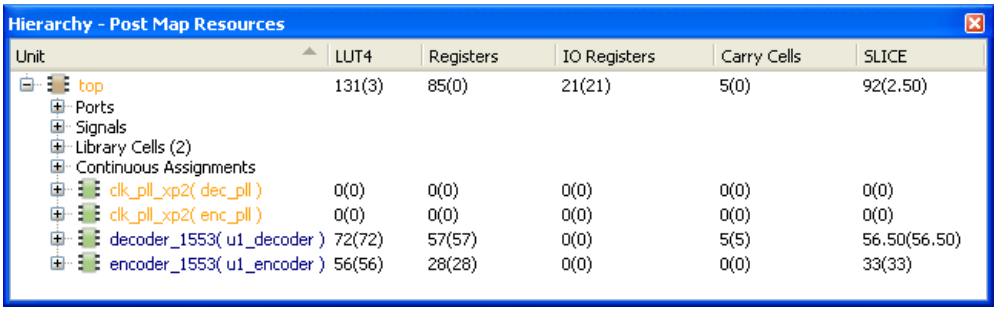
Figure 36: Post-Synthesis Resource Utilization



Unit	LUT4	Registers	IO Registers	PFUMux	Carry Cells	Instantiated
top	128(1)	85(0)	21(21)	6(0)	5(0)	55(51)
Ports						
Signals						
Library Cells (2)						
Continuous Assignments						
clk_pll_xp2(dec_pll)	0(0)	0(0)	0(0)	0(0)	0(0)	1(1)
clk_pll_xp2(enc_pll)	0(0)	0(0)	0(0)	0(0)	0(0)	1(1)
decoder_1553(u1_decoder)	72(72)	57(57)	0(0)	0(0)	5(5)	0(0)
encoder_1553(u1_encoder)	55(55)	28(28)	0(0)	6(6)	0(0)	2(2)

The numbers are updated after the synthesis and map stages of the implementation process.

Figure 37: Post-Map Resource Utilization



The amount of information shown in the Hierarchy View can be controlled by **Tools > Options**. In the HDL Diagram section, select **Custom Resources**, and then select the resource headers that you want displayed.

Resource information for a selected module can be exported to a .csv file or an ASCII text file. Right-click the desired module and choose the appropriate Export Resource command.

The Hierarchy View can be selected, closed, opened, detached, and attached (through the double-click method).

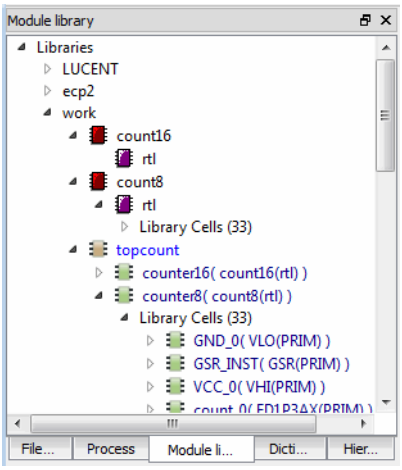
Module Library

The Module Library View is a project view that displays the modules in the design. This view is automatically displayed when the Generate Hierarchy function is selected.

Along with the Hierarchy and Dictionary views, the Module Library View is not displayed by default. To display the module library, choose **View > Show**

Views > Module library or right-click in the menu and toolbar area and choose **Module library** from the pop-up menu.

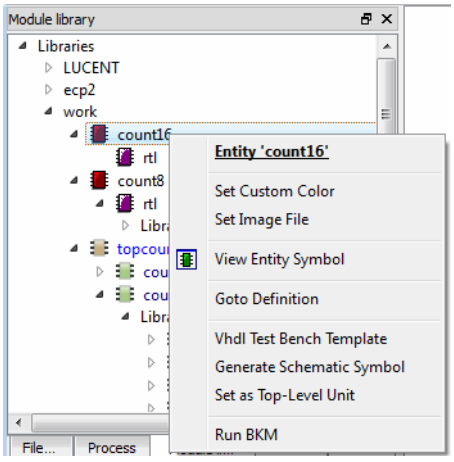
Figure 38: Module Library View



The Module Library View remains empty until the hierarchy for the design has been generated. Click the **Generate Hierarchy** button on the toolbar, or choose it from the Design menu to generate the hierarchy and populate the Module Library View with data. In addition to displaying the HDL Diagram view, generating the hierarchy causes the Hierarchy, Module Library and Dictionary project views to be displayed.

Right-click any of the objects in the Module Library View to see its type and name as well as the possible actions.

Figure 39: Module Item Description and Actions



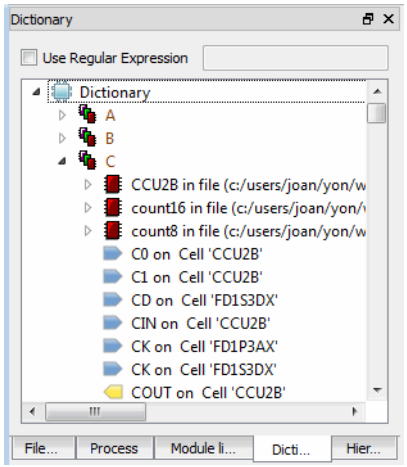
The Module Library View can be selected, closed, opened, detached, and attached (through the double-click method).

Dictionary

The Dictionary View is a project view that displays all of the data elements in the design. This view is automatically displayed when the Generate Hierarchy function is selected. Along with the Hierarchy and Module library views, the Dictionary view is not displayed by default. To display the dictionary, choose **View > Show Views > Dictionary** or right-click in the menu and toolbar area and choose **Dictionary** from the pop-up menu.

The Dictionary view remains empty until the hierarchy for the design has been generated. Click the **Generate Hierarchy** button on the toolbar, or choose it from the Design menu to generate the hierarchy and populate the Dictionary View with data. In addition to displaying the HDL Diagram, generating the hierarchy causes the Hierarchy, Module Library, and Dictionary views to be displayed.

Figure 40: Dictionary View



The “Use Regular Expression” option at the top of the view enables you to filter the Dictionary list. After you select this option, type a regular expression in the text box, and press Enter. The Dictionary list will then display only those items that match the parameters.

Right-click any of the objects in the Dictionary view to see its type and name as well as the possible actions.

Figure 41: Dictionary View with Regular Expression Filter

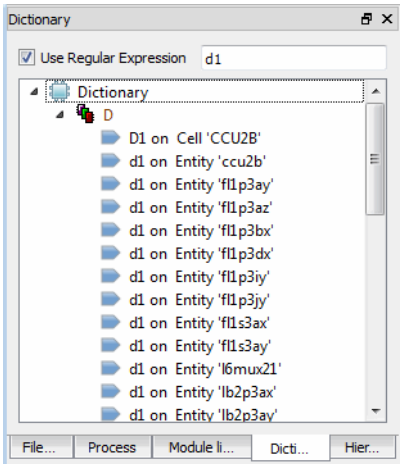
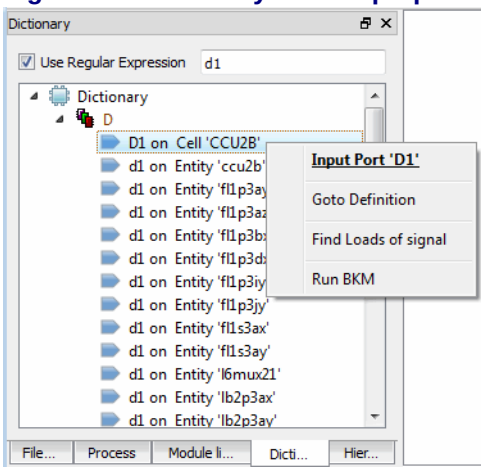


Figure 42: Dictionary Item Pop-up



The Dictionary view can be selected, closed, opened, detached, and attached (through the double-click method).

Reports

The Reports View provides a centralized reporting mechanism in the Tools view area. The Reports View is automatically displayed and updated when processes are run.

The Design Summary pane on the left provides hyperlinks to the Project Summary and to report information for each step of the design process flow. The right pane displays the report for the selected step.

Right-click a report in the Design Summary list to access the Find in Text command. Selecting this command opens a text search area at the bottom of the Report View.

Figure 43: Reports View

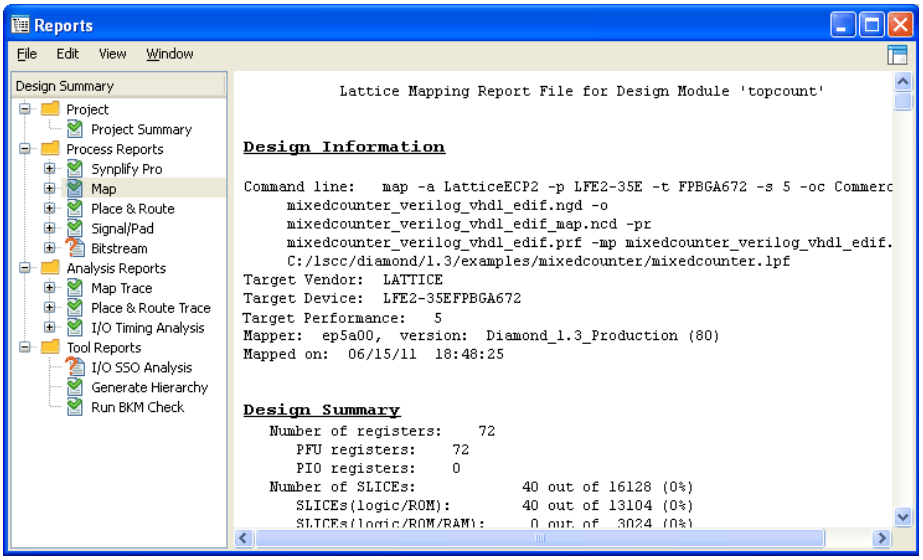
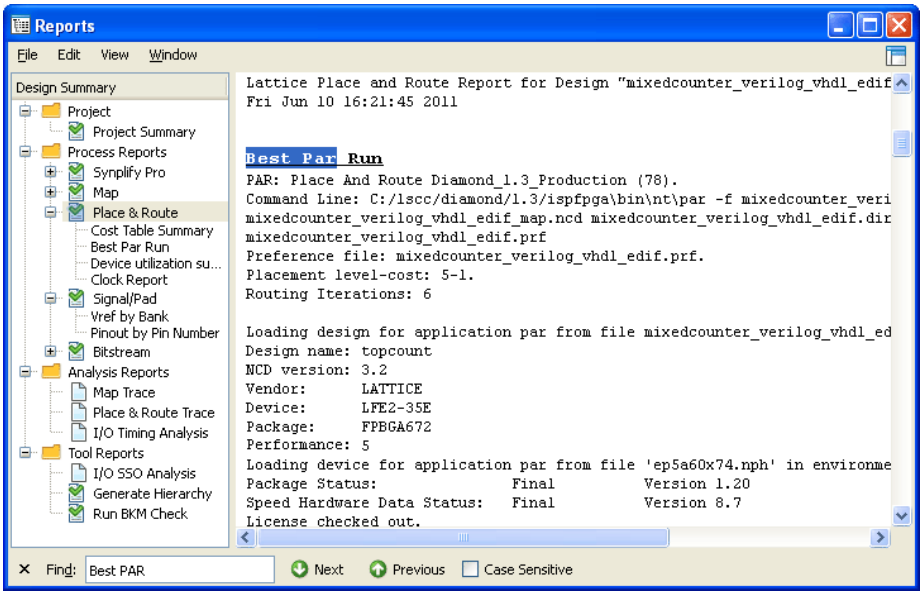


Figure 44: Report View Find in Text

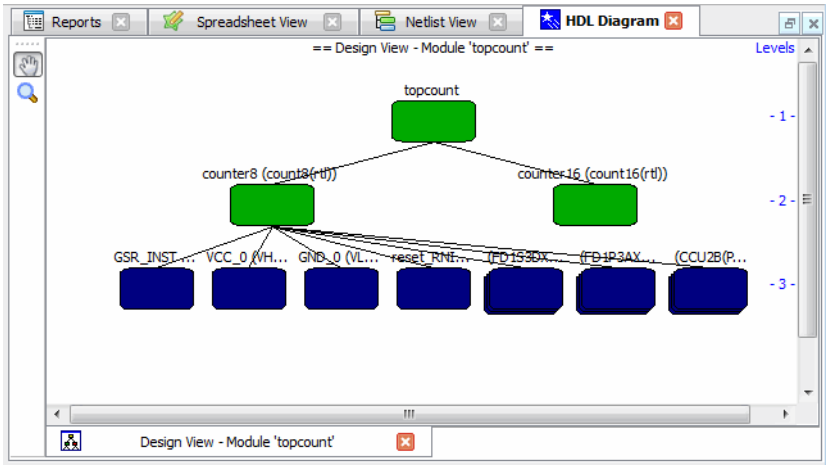


The Report View can be selected, closed, opened, detached, and attached (through the attach icon method).

Tool Views

The Tool View area of the UI displays the active tools. For example, Figure 45 shows the Tool view area with the Reports, Spreadsheet View, Netlist View and HDL Diagram displayed:

Figure 45: Tool View Area

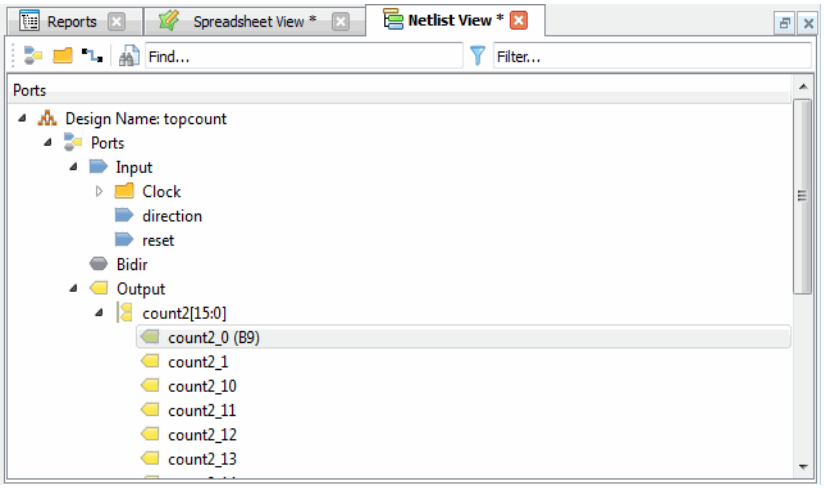


When multiple tools are active, their display can be controlled with the tab group functions in the Window toolbar. See “Common Tasks” on page 38 for more information on tab group functions.

Each tool view is specific to its tool and can contain additional toolbars, multiple panes, or multiple windows controlled by additional tabs. See “Working with Tools and Views” on page 77 for descriptions of each tool and view, plus details on controlling their display.

You will notice an asterisk “*” in the tool view tab title when there has been a change to the shared memory.

Figure 46: Tool View Tab Title Showing Changed Data

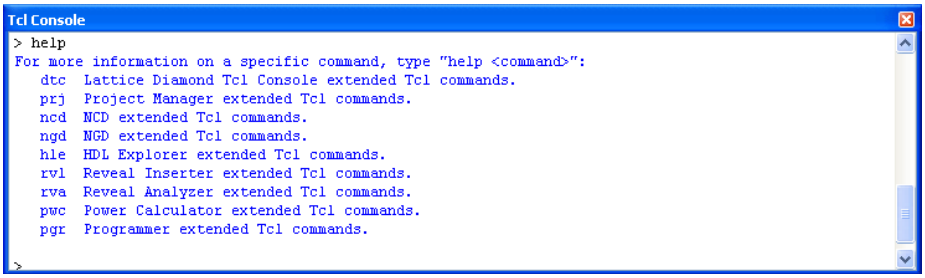


The Tool views can be selected, closed, opened, detached, and attached (through the attach icon method).

Tcl Console

The Tcl Console is an integrated console for Tcl scripting. You can enter Tcl commands in the console to control all of the functionality of Lattice Diamond. Use the Tcl help command (help) to display a listing of the groups of Lattice Diamond extended Tcl commands.

Figure 47: Tcl Console

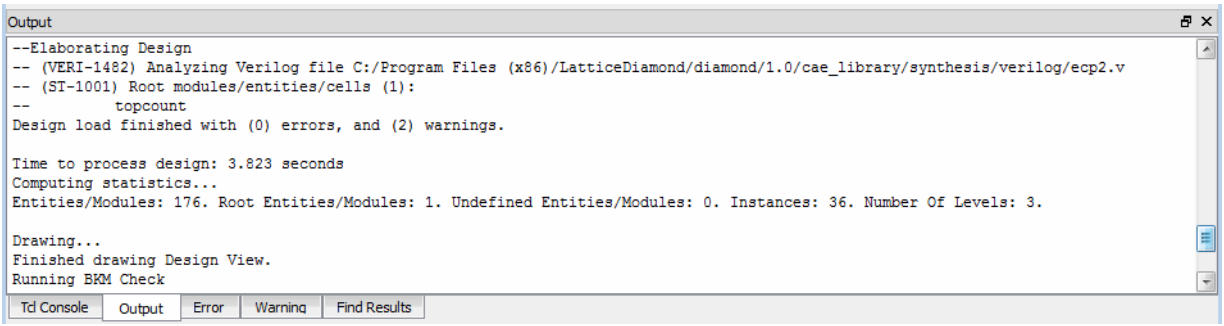


The Tcl Console can be selected, closed, opened, detached, and attached (through the double-click method).

Output

The Output View is a read-only area where tool output is displayed.

Figure 48: Output View

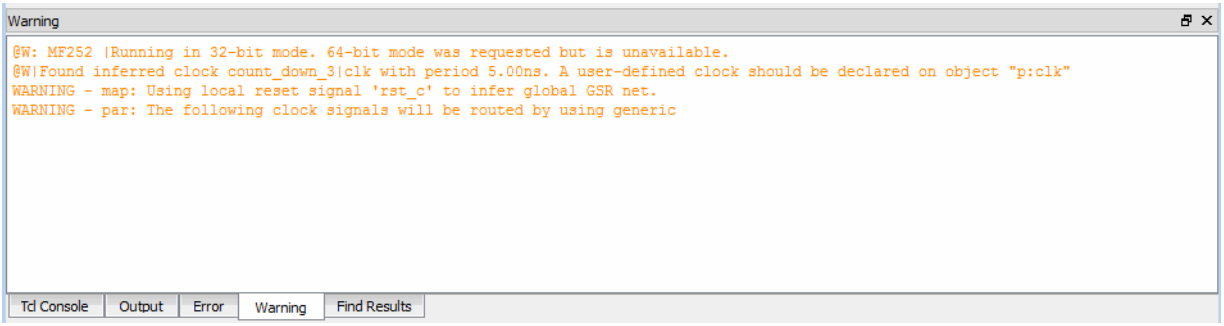


The Output view can be selected, closed, opened, detached, and attached (through the double-click method).

Error and Warning

The Error and Warning views are read-only areas where tool errors and warnings are displayed. The error and warning information is cumulative, so you will see the history of all errors and warnings from potentially multiple process runs. Error and warning information is not cleared with each new run of a process. You can clear this information manually from the view by right-clicking inside the view and choosing **Clear** from the pop-up menu.

Figure 49: Error and Warning Display



The Error and Warning views can be selected, closed, opened, detached, and attached (through the double-click method).

Common Tasks

The Lattice Diamond UI controls many tools and processes. The following sections describe some of the more commonly performed tasks.

Controlling Views

All of the views in Lattice Diamond are controlled in a similar manner, even the information they contain varies widely. Here are some of the most common operations:

- Open – Use the **View > Show Views** menu selections or right-click in the menu or toolbar areas to select a view from the pop-up menu.
- Select – If a view is already open you can select its tab to bring it to the front.
- Close – Click the **x** in the upper right corner of the view, or right-click in the menu or toolbar area and select the view from the pop-up menu to clear the check mark.
- Detach – Click the detach button in the upper right corner of the view
- Attach – Use one of the two following methods:
  - For project views, Output and the Tcl Console, double-click in the window title bar, and the window will be attached back into the main UI window.

- For tool views, click the attach button , and the window will be attached back into the main UI window.

- Move – Click and hold a view's tab, and then drag and drop the view to a different position within the open views

Grouping Tabs

The tab grouping controls are in the window toolbar.

Figure 50: Window Toolbar

The controls work as follows:

- Split Tab Group – displays two views side by side
- Merge Tab Group – merges a split tab group back into the primary view
- Move to Another Tab Group – moves the selected tab to the other tab group
- Switch Tab Group Position – switches the positions of the two tab groups
- Integrate all Tools – brings all detached views back into the main window

You can also drag and drop the tabs to change the position of the open views.

Figure 51 shows the display after selecting **Split Tab Group**.

Figure 51: Split Tab Group

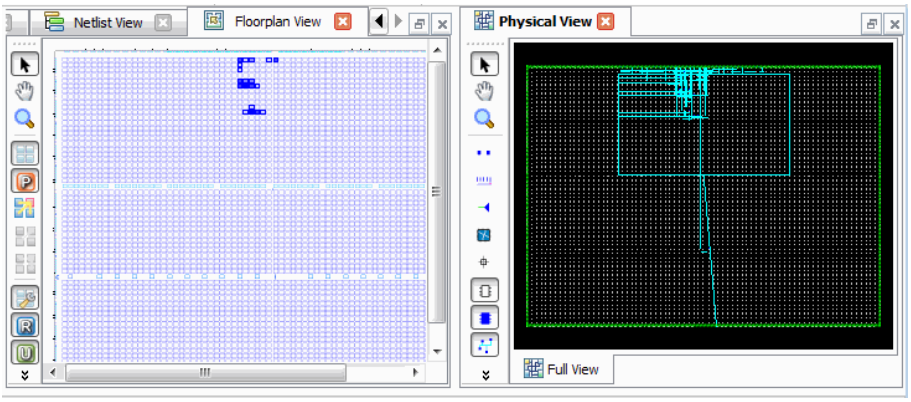
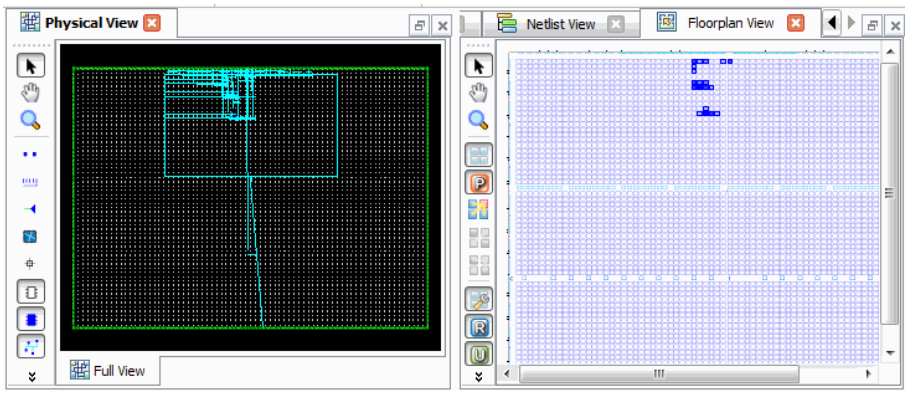


Figure 52 shows the display after selecting **Switch Tab Group Position**.

Figure 52: Switch Tab Group Position



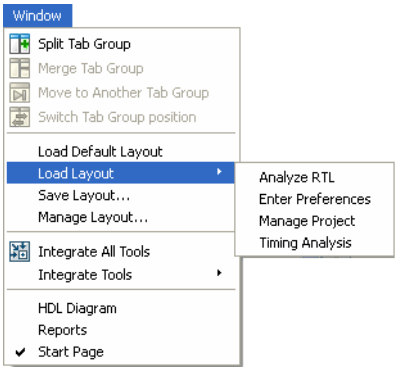
Managing Layouts

Diamond's layout management utilities allow you to load a predefined layout of views and to create your own customized layouts. Predefined and customized layouts enable you to get to work immediately on a specific task, such as analyzing your source code or setting design constraints.

Using Predefined Layouts

Four predefined layouts are included with Diamond and are available from the Window menu.

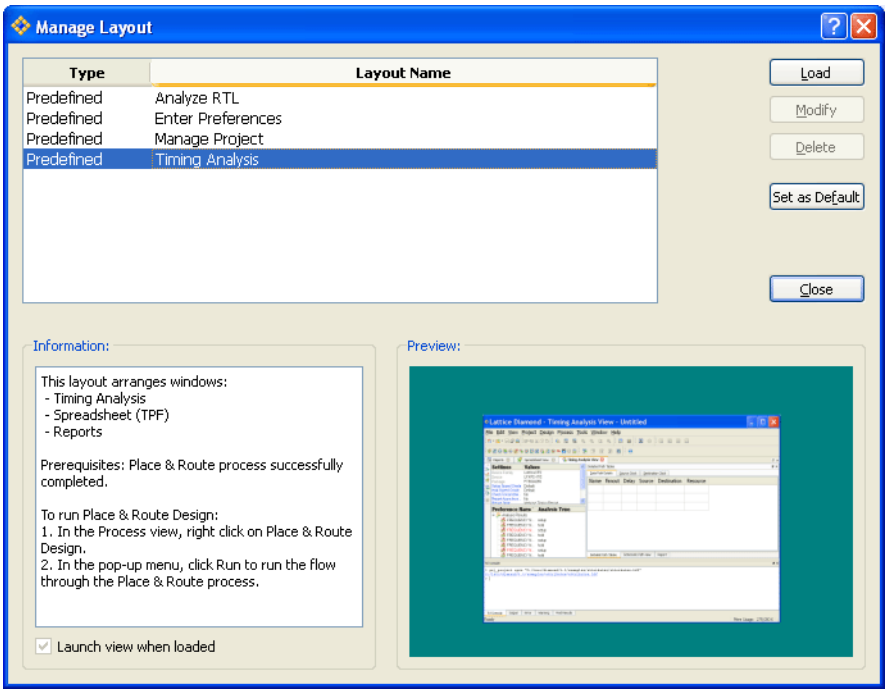
Figure 53: Diamond Predefined Layouts



Each of these predefined layouts is described in the Manage Layout dialog box, which is also available from the Window menu. The Information pane provides prerequisites and instructions for viewing all the windows of the layout. For example, if you load the Timing Analysis predefined layout and your design is still in the pre-route stage, you will need to run Place & Route in order to view the Timing Analysis View.

You cannot modify or delete any of the predefined layouts from the Manage Layout dialog box. However, you can load one of the predefined layouts into

Figure 54: Predefined Layouts



Diamond's main window, modify it by opening and closing or detaching views, and then save the arrangement as a customized layout.

Loading a Predefined Layout You can load a predefined layout by selecting it from the Window menu or from the Manage Layout dialog box. When you load a predefined layout, Diamond does not close any of the tool views that are already open. Any currently open tool views that are part of the predefined layout will be arranged according to the layout when it is loaded. Other open tool views will remain in their current positions. For example, if you have Power Calculator detached as a separate window when you load the "Analyze RTL" predefined layout, Power Calculator will remain open and in its detached position.

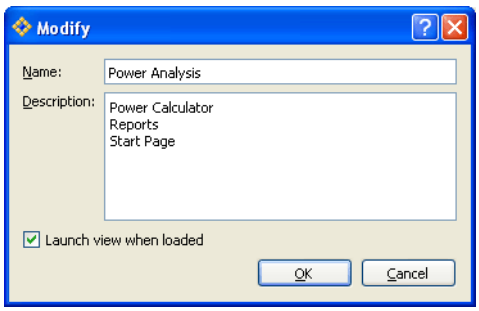
Creating Customized Layouts

When you have the UI set up the way you like, you can save the arrangement as a customized layout by using the Window > Save Layout command. You can save as many of these customized layouts as you want, giving each a unique name. The names of saved customized layouts are added to the Window > Load Layout menu, and they appear as "Customized Layouts" in the Manage Layouts dialog box.

You can use the Manage Layout dialog box to delete a customized layout, and you can open the Modify dialog box to make minor changes. The Modify dialog box enables you to change the name of the customized layout and to select or clear the "Launch view when loaded" option. However, it does not allow you to add or delete tool views.

Using the "Launch view when loaded" Option The "Launch view when loaded" option will cause all of the layout's tool views to open when you load the customized layout. This gives you quick access to the tools and is very useful for layouts that include only a few tool views. For more complex layouts, this option can cause a long delay while each tool view gets loaded with the layout. If your customized layout includes a lot of tool views, you should clear this option.

Figure 55: Modify Layout



Loading a Customized Layout You can load a customized layout by selecting it from the Window menu or from the Manage Layout dialog box. When you load a customized layout, Diamond does not close any of the tool views that are already open. If the customized layout is set to "Launch view when loaded," the Load Layout command will open all of the layout's tool views that are currently unopened. Any of the layout's tool views that were already open will be arranged according to the layout. Any other open tool views will remain in their current positions.

If the customized layout is not set to "Launch view when loaded," the Load Layout command will not open any of the layout's tool views. For these types of layouts, first open the layout's tool views that you need to work with, and then choose the Load Layout command. The tool views will be arranged according to the customized layout. You can later open any or all of the remaining tool views that are part of the customized layout and choose the Load Layout command again. These tool views will also be arranged according to the layout.

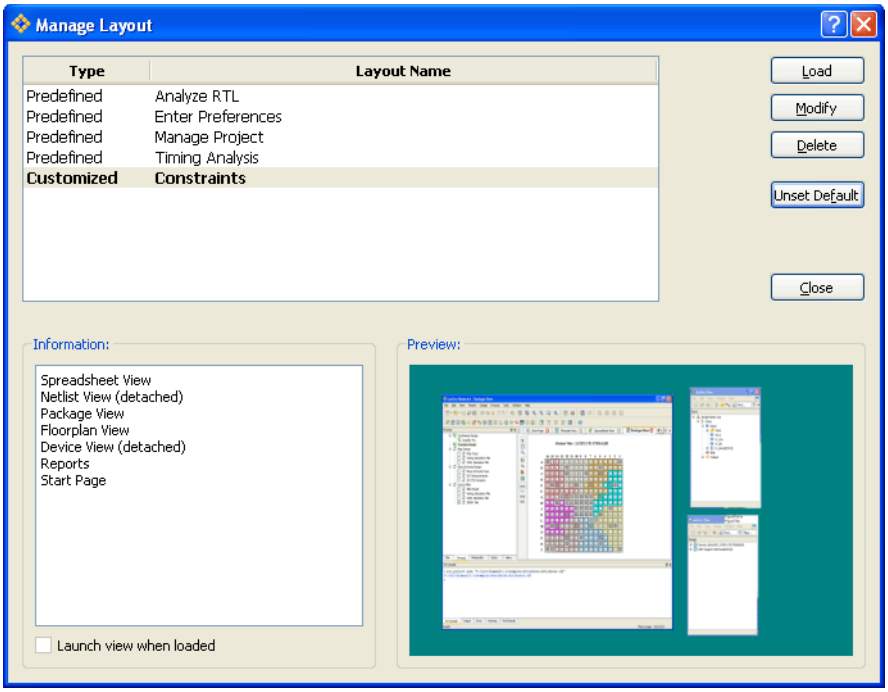
Layout Management Commands

All of the commands for layout management are available from the **Window** menu:

- Load Default Layout – Loads the predefined or customized layout that has been specified as the default in the Manage Layout dialog box. This command gives you quick access to your default layout after you have opened your project in Diamond.
- Load Layout – Provides quick access to the predefined and customized layouts.
- Save Layout – Enables you to save the arrangement of all open views in your current session as a customized layout. The Save Layout dialog box shows a preview of the layout and lists all the views that will be included in the customized layout.

- Manage Layouts – Opens the Manage Layout dialog box, which allows you to preview and load a predefined or customized layout, delete a customized layout, and select a layout as the default so that it will open with the Load Default Layout command. The Layout Name list displays the default layout in bold type. If a lot of views are listed for a customized layout, you should leave the “Launch view when loaded” option cleared to avoid a long wait as the layout is loaded.

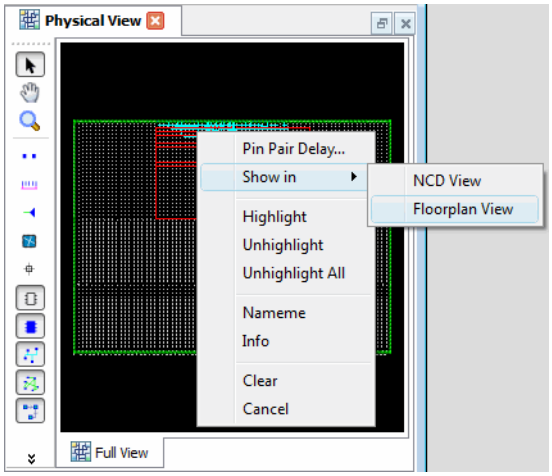
Figure 56: Customized Layout as Default



Cross-Probing Between Views

It is possible to select a data object in one view and see that same data object in a different view or views. Right-click a selected object; if cross-probing is available for that object you will see a **Show In** sub-menu with available views listed. If you select a view that is not yet open, Diamond will open it automatically. Cross-probing is available for most tool views.

Figure 57: Show In



5

Working with Projects

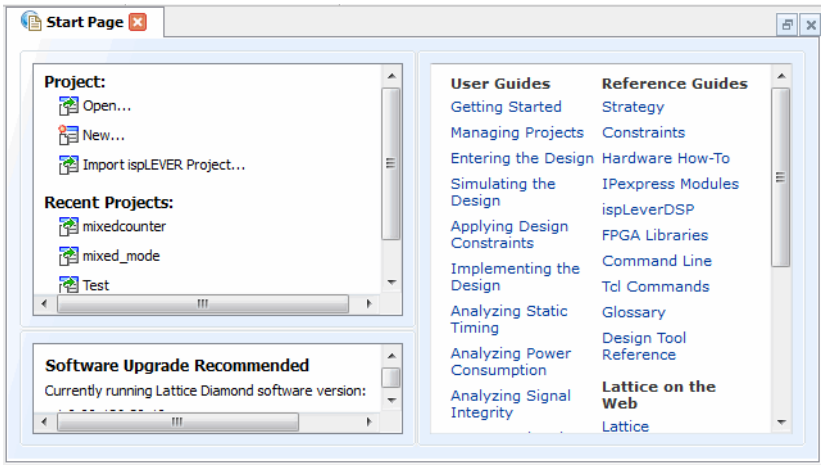
This chapter covers projects and their elements. Implementations and strategies are explained and some common project tasks are shown.

Overview

A project is the top organizational element in the Lattice Diamond design environment. Projects consist of design, constraint, configuration and analysis files. There is only one project open at a time, and a single project can include multiple design structures and tool settings.

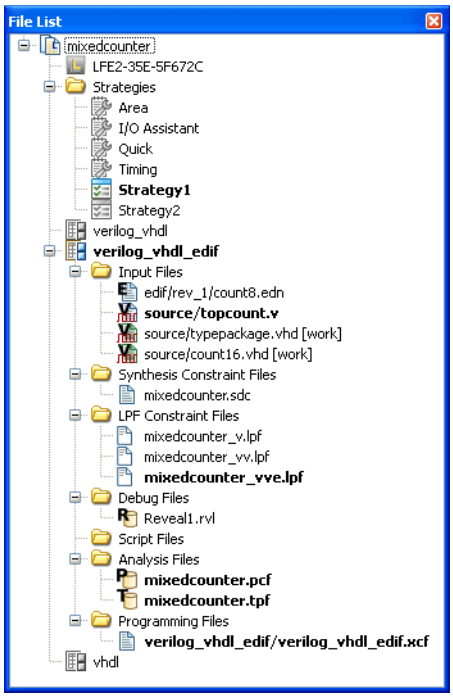
You can create, open, or import a project from the Start Page. See the chapter “Getting Started” on page 3 for instructions on creating a new project.

Figure 58: Opening a Project from the Start Page



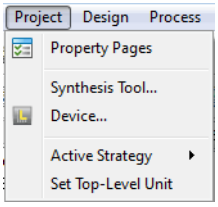
The File List view shows a project and its main elements.

Figure 59: Project Files in the File List



The Project menu commands enable you examine the project properties, change the target device, set the synthesis tool, show the active strategy tool settings and set the top level design unit.

Figure 60: Project Menu



Implementations

An implementation is the structure of a design and can be thought of as *what* is in the design. For example, one implementation might use inferred memory while another implementation uses instantiated memory. Implementations also define the constraint and analysis parameters for a project.

There can be multiple implementations in a project, but only one implementation can be active at a time. And there must be one active implementation. Every implementation has an associated active strategy. Strategies are a shared pool of resources for all implementations and are

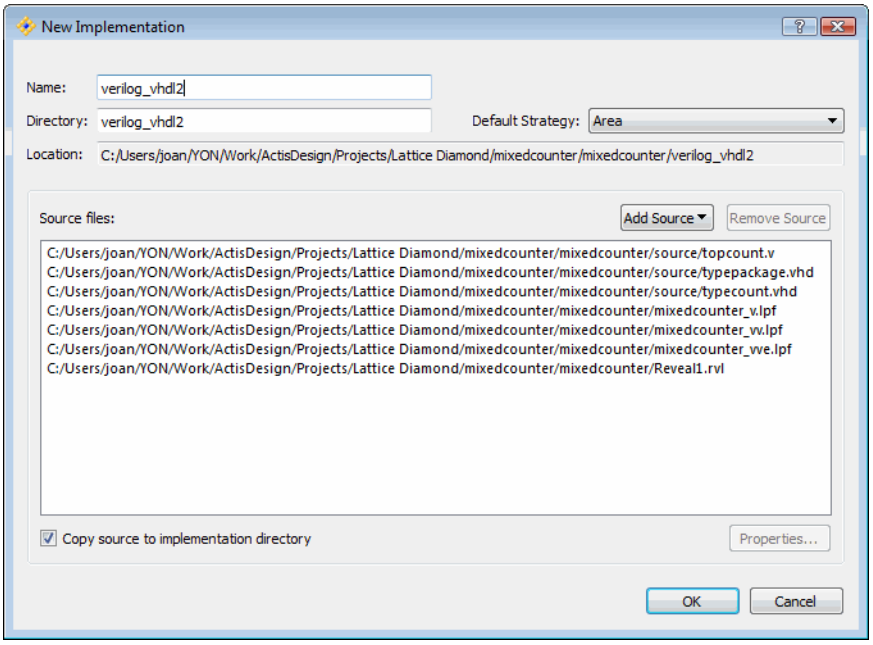
discussed in the next section. An implementation is created whenever you create a new project.

Implementations consist of the following files:

- Input files
- Synthesis constraint files
- LPF constraint files
- Debug files
- Script files
- Analysis files
- Programming files

To add a new implementation to an existing project, right-click on the project name in the File List project view and select **Add > New Implementation**. In the New Implementation dialog box, you can set the implementation name, directory, default strategy, and you can add source files. When you select **Add Source** you have a choice of browsing for the source files or using a source from an existing implementation.

Figure 61: New Implementation



Notice that you have the option to “Copy source to implementation directory.” If this option is selected, the source files will be copied from the existing implementation to the new implementation, and you will be working with different source files in the two implementations. If you want the two implementations to share the same source files and stay in sync, make sure that this option is not selected.

To make an implementation active, right-click its name in the File List and choose **Set as Active Implementation**.

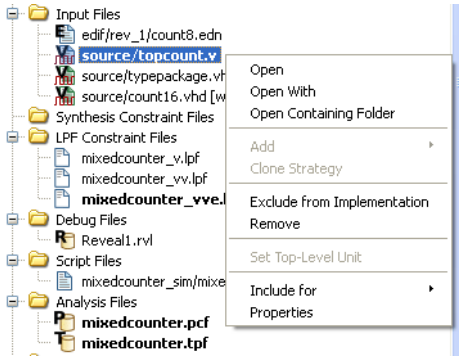
To add a file to an implementation, right-click the implementation name or any file folder in the implementation and choose **Add > New File**, or choose **Add > Existing File**.

Input Files

Input files are the design source files for the project. Input files can be any combination of Verilog, VHDL, and EDIF files.

Right-click an input file name to open a pop-up menu of possible actions for that file.

Figure 62: Input File Actions



You can use the “Include for” commands to specify that a source file be included for both synthesis and simulation, synthesis only, or simulation only.

Synthesis Constraint Files

Synthesis constraint files are constraint files that are directly interpreted by the synthesis engine. SDC constraint files can use either Synplify® or Synopsys® format constraints. An .sdc file can be added to an implementation if the selected synthesis tool is Synplify Pro or Precision. If the selected synthesis tool is the Lattice Synthesis Engine (LSE), which is available for MachXO/2 and Platform Manager devices, a Lattice design constraint (.ldc) synthesis file can be added. Constraints in the .ldc .file use the Synopsys constraint format.

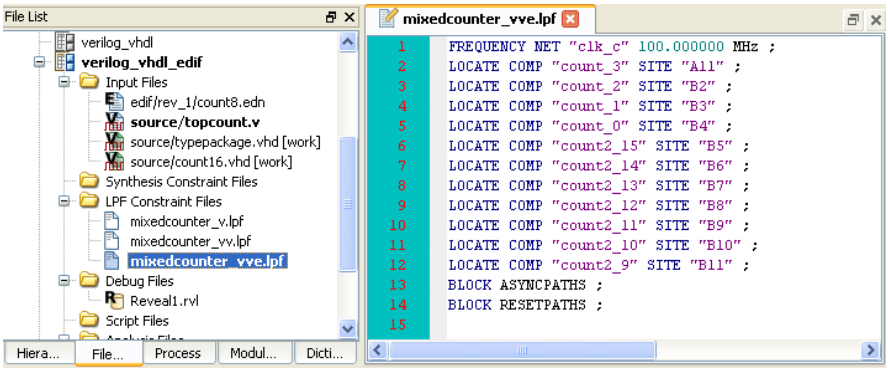
An implementation can have multiple synthesis constraint files. Multiple .sdc files can be active at the same time, but only one .ldc file can be active at a time. Unlike LPF constraints, a synthesis constraint file must be set active by the user.

LPF Constraint Files

LPF constraint files are logical preference files (.lpf), source files for storing logical constraints called preferences. Preferences that you add and edit using Diamond's preference-editing views, such as Spreadsheet View, are saved to the active .lpf file. The active logical preference file is then used as input for post-synthesis processes.

An implementation can have multiple .lpf files, but only one can be active at a time.

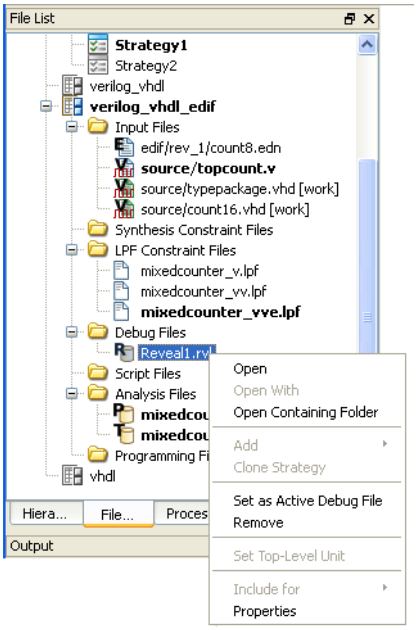
Figure 63: LPF Constraint File



Debug Files

The files in the Debug folder are project files for the Reveal Inserter. They are used to insert hardware debug into your design. There can be multiple debug files, and one or none can be set as active. To insert hardware debug into your design, right-click a debug file name and choose **Set as Active Debug File** from the pop-up menu. The debug file name becomes bold, indicating that it is active.

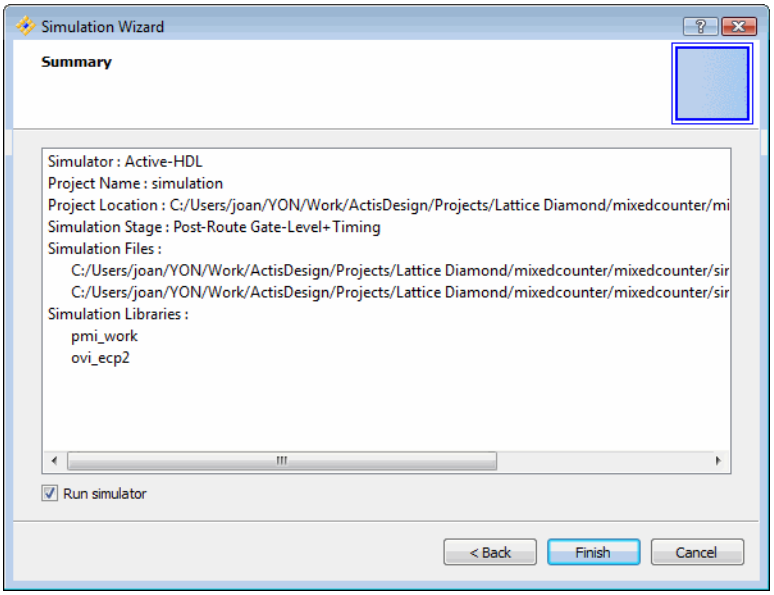
Figure 64: Reveal Debug File Actions



Script Files

The Script Files folder contains the scripts that are generated by the Simulation Wizard. After you run the Simulation Wizard, the steps are stored in a simulation project file (.spf), which can be used to control the launching of the simulator.

Figure 65: Simulation Script File



Analysis Files

The Analysis Files folder contains Power Calculator files (.pcf) and Timing Preference files (.tpf). The folder can contain multiple analysis files, and one or none can be set as active. The active or non-active status of an analysis file affects the behavior of the associated tool view.

Programming Files

Programming files (.xcf) are configuration scan chain files used by the Diamond Programmer for programming devices in a JTAG daisy chain. The .xcf file contains information about each device, the data files targeted, and the operations to be performed.

An implementation can have multiple .xcf files, but only one can be active at a time.The file must be set active by the user.

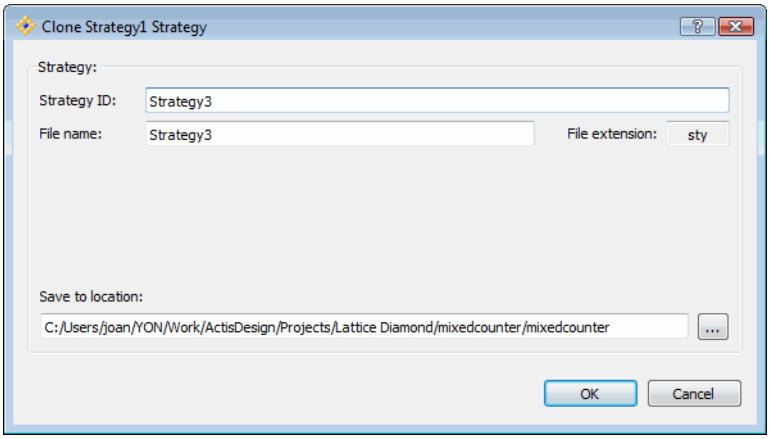
Strategies

Strategies are collections of all the implementation-related tool settings in one convenient location. Strategies can be thought of as recipes for how the design will be implemented. An implementation defines *what* is in the design, and a strategy defines *how* that design will be run. There can be many strategies, but only one can be active at a time. There must be one active strategy for each implementation.

Lattice Diamond provides four predefined strategies. It also enables you to create customized strategies. Predefined strategies cannot be edited, but they can be cloned, modified, and saved as customized user strategies. Customized user strategies can be edited, cloned, and removed. All strategies are available to all of the implementations, and any strategy can be set as the active one for an implementation.

To create a new strategy from an existing one, right-click the existing strategy and choose **Clone <strategy name> Strategy**. Set the new strategy's ID and file name.

Figure 66: Cloning to Create a New Strategy

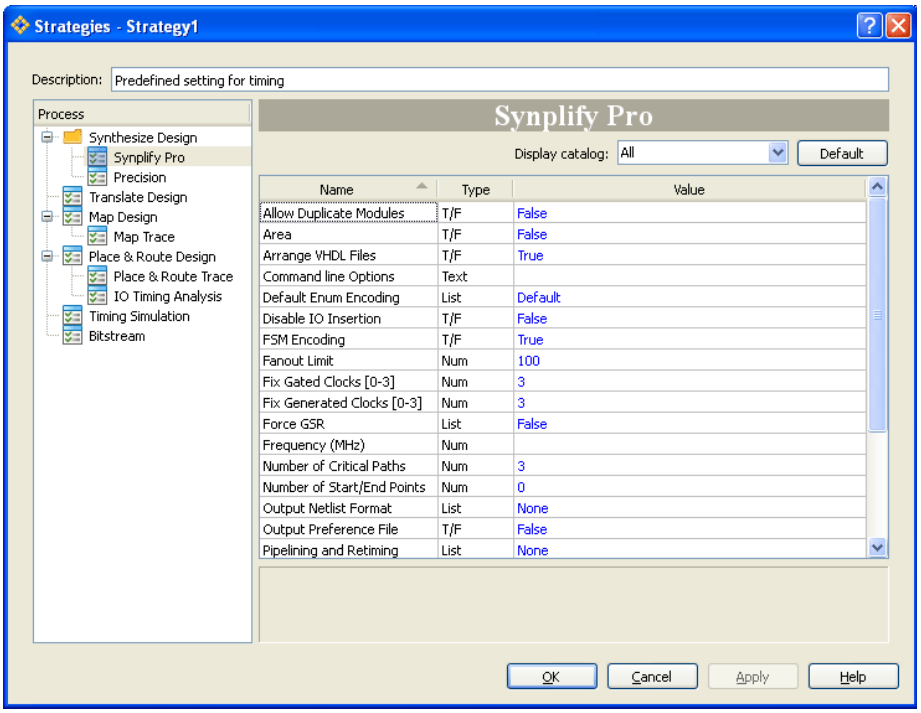


To make a strategy active, right-click the strategy name and choose **Set as Active Strategy**.

To change the settings in a strategy:

- Double-click the strategy name in the File List view
- Select the option type to modify
- Double-click the Value of the option to be changed

Figure 67: Strategy Settings



The default options are displayed in blue font.

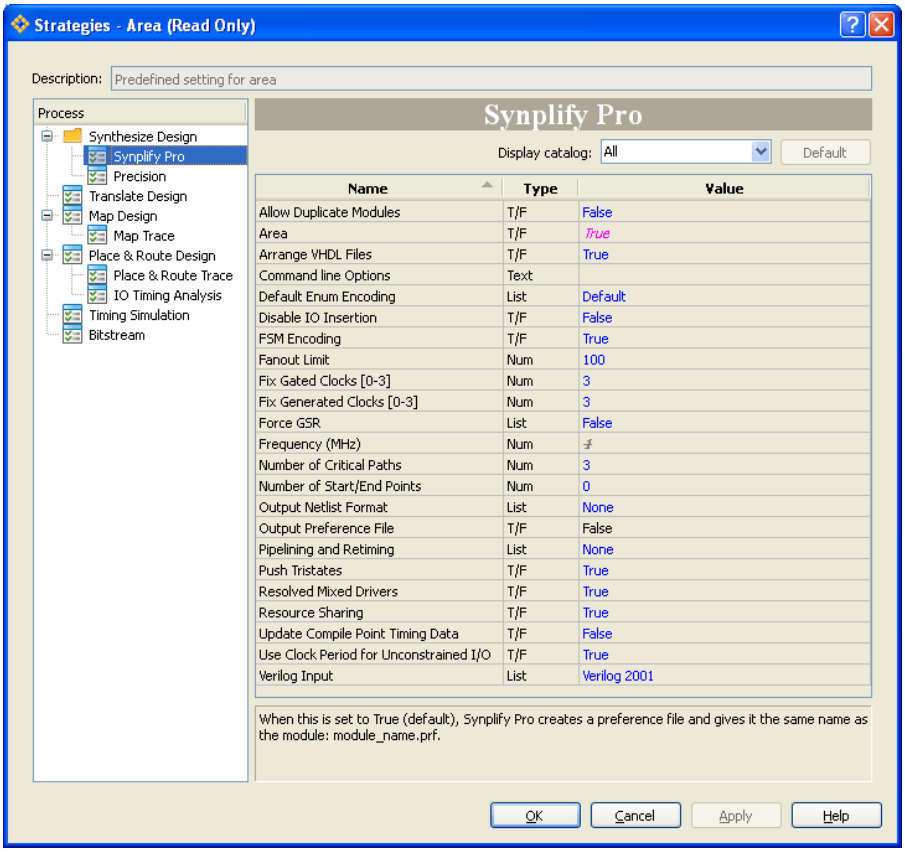
Strategies are design data independent and can be exported and used in multiple projects.

Area

The Area strategy is a predefined strategy for area optimization. Its purpose is to minimize the total logic gates used while enabling the tight packing option available in Map. It is commonly used for low-density devices such as MachXO.

Applying this strategy to large and dense designs might cause difficulties in the place and route process, such as longer time or incomplete routing.

Figure 68: Area Predefined Strategy



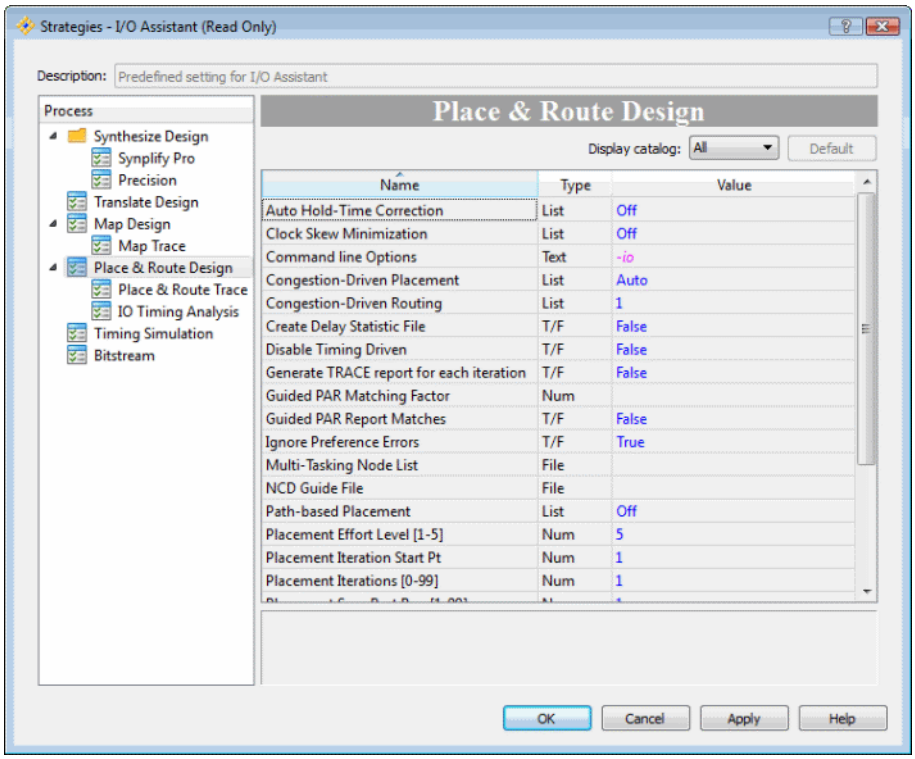
I/O Assistant

The I/O Assistant strategy is a predefined strategy that is useful for I/O design. It helps you select a legal device pinout and produce LOCATE and IOBUF preferences for optimal I/O placement.

The benefit is that you will get results in I/O placement information early on, without having a complete design or any long runtimes after finishing place and route. However, applying this strategy to your design might take extra runtime, because it executes logic synthesis, translation, map, and I/O placement process.

If you use the I/O Assistant strategy for your project, the generated .ncd file will be incomplete. Running the Export Files > Bitstream File process or the Export Files > JEDEC File process will fail. If you want to implement a complete design, you will need to choose another strategy and rerun all processes again. See “Lattice Diamond Design Flow” on page 61 for more information.

Figure 69: I/O Assistant Predefined Strategy

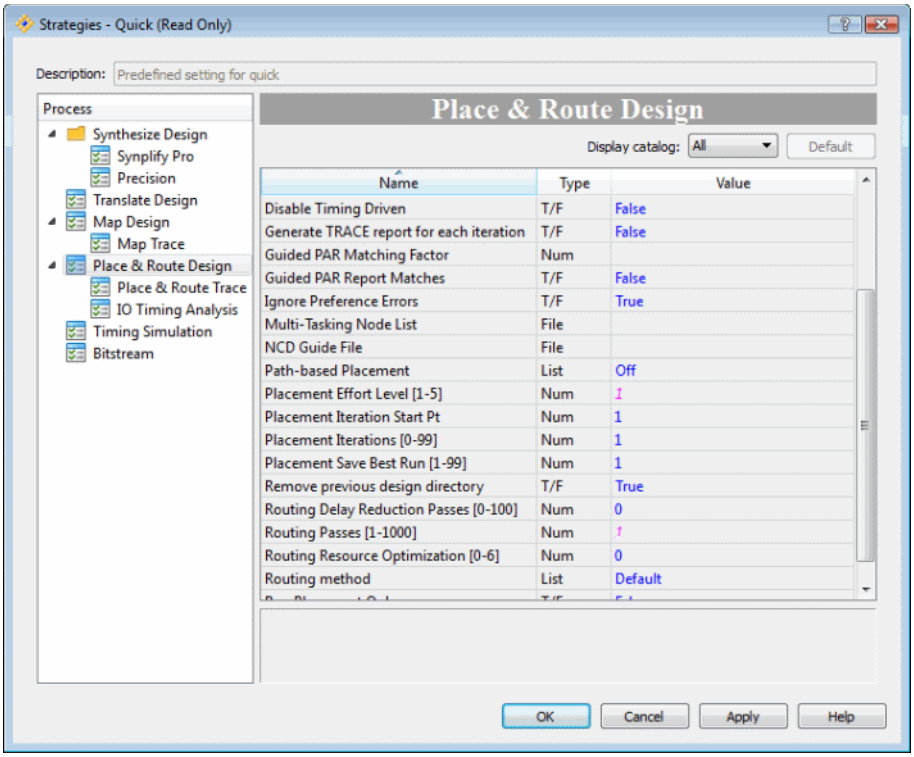


Quick

The Quick strategy is a predefined strategy for doing an initial quick run. This strategy uses very low effort level in placement and routing to get results with minimum run time. If your design is small and your target frequencies are low, this is a good strategy to try. Even if your design is large, you might want to start with this strategy to get a first look at place-and-route results and to tune your preference file with minimum runtime.

The Quick strategy will give you results in the least possible time. However, the quality of these results in terms of achieved frequency will probably be low, and large or dense designs might not complete routing.

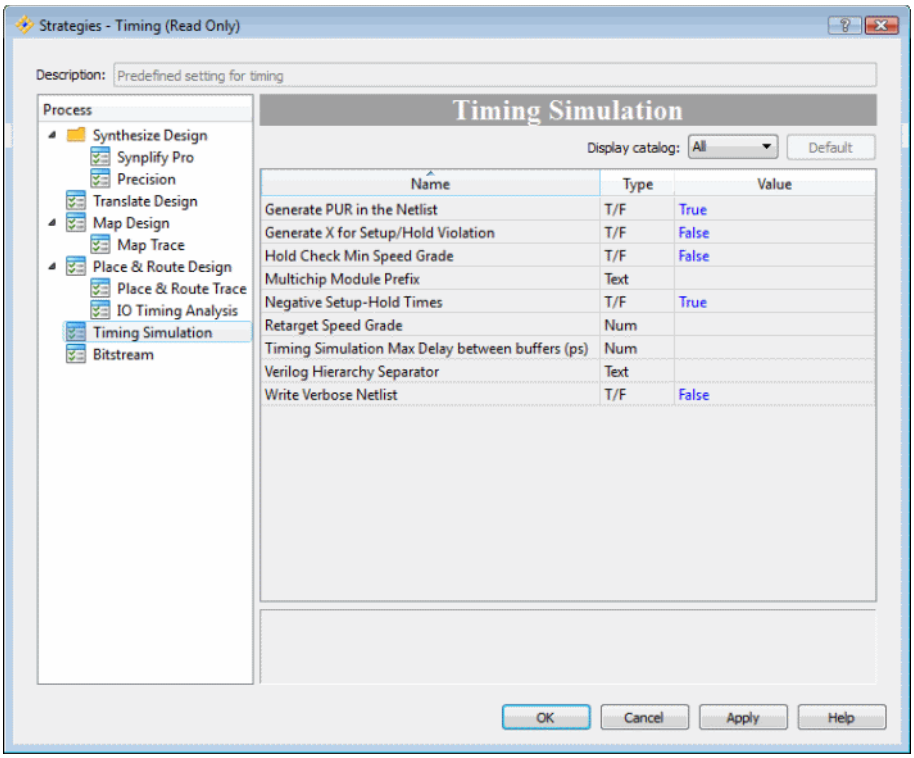
Figure 70: Quick Predefined Strategy



Timing

The Timing strategy is a predefined strategy for timing optimization. Its purpose is to achieve timing closure. The Timing strategy uses very high effort level in placement and routing. Use this strategy if you are trying to reach the maximum frequency on your design. If you cannot meet your timing requirements with this strategy, you can clone it and create a customized strategy with refined settings for your design. This strategy might increase your place-and-route run time compared to the Quick and Area strategies.

Figure 71: Timing Predefined Strategy



User-Defined

You can define your own customized strategy by cloning and modifying any existing strategy. You can start from either a predefined or a customized strategy.

Common Tasks

Working with projects includes many tasks: creating the project, editing design files, modifying tool settings, trying different implementations and strategies, saving your data.

Creating a Project

See “Creating a New Project” on page 5 in the “Getting Started” chapter for step-by-step instructions.

Changing the Target Device

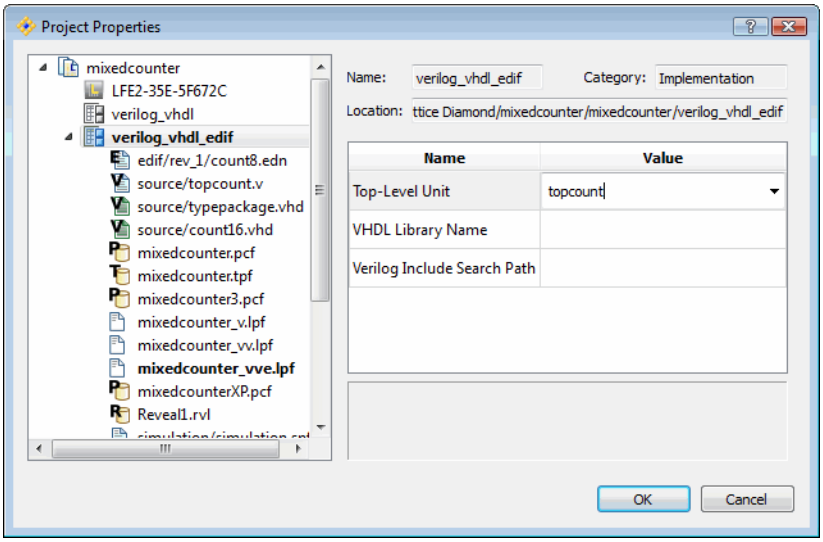
There are two ways to access the Device Selector dialog box for changing the target device:

- Right-click the device in the project File List view
- Choose **Project > Device**

Setting the Top Level of the Design

If multiple top levels exist in the hierarchy of your HDL source files, you will need to set the top-level design unit. After generating the hierarchy, choose **Project > Set Top-Level Unit**.

Figure 72: Top-Level Design Unit



In the Project Properties dialog box, select **Value** next to **Top-Level Unit** and select the desired top level from the list.

Editing Files

You can open any of the files for editing by double-clicking or by right-clicking and choosing **Open** or **Open with**.

Saving Project Data

In the File menu are selections for saving your design and project data.

- Save – saves the currently active item
- Save As – opens the Save As dialog to save the active item
- Save All – saves all changed documents
- Save Project – saves the current project
- Archive Project – creates a zip file of the current project in a location you specify

6

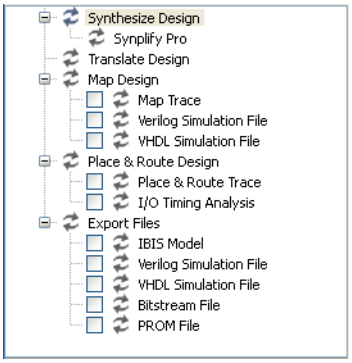
Lattice Diamond Design Flow

This chapter describes the design flow in Lattice Diamond. Running processes and controlling the flow for alternate what-if scenarios is explained. A summary of the major differences from the ispLever flow is included.

Overview

The FPGA implementation design flow in Lattice Diamond provides extensive what-if analysis capabilities for your design. The design flow is displayed in the Process view.

Figure 73: Design Flow



Design Flow Processes

The design flow is organized into discrete processes, where each step allows you to focus on a different aspect of the FPGA implementation.

Synthesize Design This process runs the selected synthesis tool (Synplify Pro is the default) in batch mode to synthesize your HDL design.

Translate Design This process converts the EDIF file output from synthesis to NGD format. If the design utilizes Lattice NGO netlist files, such as generated Lattice IP, the netlist will also be read into the design.

Map Design This process maps the design to the target FPGA and produces a mapped Native Circuit Description (.ncd) design file. Map Design converts a design's logical components into placeable components.

The Map Design process also generates timing analysis and simulation files when you have selected them before running Map Design:

- Map Trace – creates a post-map timing report file (.tw1) that helps determine where timing constraints will not be met. This report can be viewed in Report View. In post-map timing analysis, Trace determines component delays and uses estimated routing delays. The estimation method used is based on the setting for "Route Estimation Algorithm" in the Map Trace section of the active strategy. This can be used to detect severe timing issues, such as deep levels of logic, without incurring the runtime of PAR.
- Verilog Simulation File – generates a Verilog netlist of the mapped design that is back annotated with estimated timing information. This generated _map_vo.vo file enables you to run a simulation of your design.

  back annotates the mapped design with estimated timing information so that you can run a simulation of your design. The back annotated design is a Verilog netlist.
- VHDL Simulation File – generates a VHDL netlist of the routed design that is back annotated with estimated timing information. This generated _map_vho.vho file enables you to run a simulation of your design.

The timing analysis and simulation files can also be generated separately by double-clicking each one.

Place & Route Design This process takes a mapped physical .ncd design file and places and routes the design. The output is an .ncd file that can be processed by the design implementation tools.

The Place & Route Design process also generates timing and SSO analysis files when you have selected them before running Place & Route Design:

- Place & Route Trace – creates a timing report (.twr) that enables you to verify timing. This report can be viewed in Report View. In post-route timing analysis, Trace analyzes path delays and reports where these occur in the design.
- I/O Timing Analysis – runs I/O timing analysis and generates an I/O Timing Report that can be viewed in Report VIew. The report is an

analysis of inputs and outputs across all potential silicon to help ensure that the board design is compatible; it shows the constraints to which the board design will need to adhere.

For each input port in the design, this report shows the worst case setup and hold time requirements. For each output port, it shows the worst case min/max clock-to-out delay. The computation is performed over all performance grades available for the device and at the voltage and temperature specified in the preference file. I/O timing analysis also automatically determines the clocks and their associated data ports.

Export Files This process generates the IBIS, simulation, and programming files that you have selected for export:

- IBIS Model – generates a design-specific I/O Buffer Information Specification model file (.ibs). IBIS models provide a standardized way of representing the electrical characteristics of a digital IC's pins (input, output, and I/O buffers).
- Verilog Simulation File – generates a Verilog netlist of the routed design that is back annotated with timing information. This generated .vo file enables you to run a timing simulation of your design.
- VHDL Simulation File – generates a VHDL netlist of the routed design that is back annotated with timing information. This generated .vho file enables you to run a timing simulation of your design.
- JEDEC File – generates a JEDEC file for programming the device. JEDEC is the industry standard for PLD formats. In the Lattice Diamond software, JEDEC refers to the fuse map of your design for the selected device. This option is applicable only to non-volatile FPGAs such as LatticeXP, LatticeXP2, MachXO and MachXO2.
- Bitstream File – generates a configuration bitstream (bit images) file, which contains all of the design's configuration information that defines the internal logic and interconnections of the FPGA, as well as device-specific information from other files. This option is applicable only to volatile SRAM-based FPGAs, such as LatticeECP/2/M, LatticeECP3, LatticeSC/M.
- PROM File – generates an output file in one of several programmable read-only memory (PROM) file formats. This option is applicable only to volatile SRAM-based FPGAs, such as LatticeECP/2/M, LatticeECP3, LatticeSC/M

The files for export can also be generated separately by double-clicking each one.

Running Processes

For each step in the process flow you can perform the following actions:

- Run – runs the process, if it has not yet been run
- Rerun – reruns a process that has already been run
- Rerun All – reruns all processes from the start to the selected process
- Stop –- stops a running process

- Clean Up Process – clears the process state and puts a process into an initial state as if it had not been run
- Refresh Process – reloads the current process state

The state of each process step is indicated with an icon to the left of the process:

Process in initial state

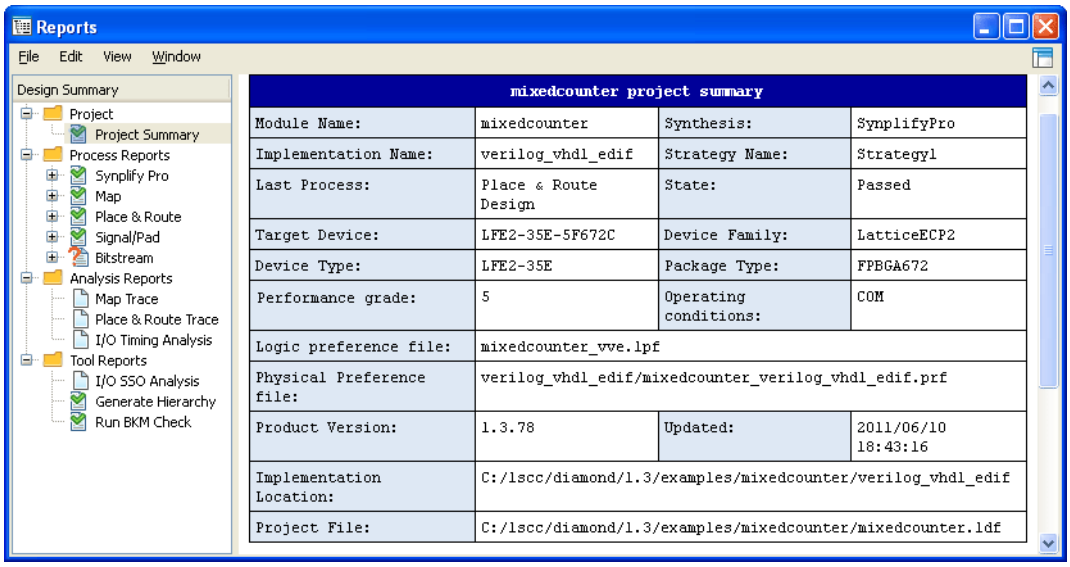
Process completed successfully

Process completed with warnings, see Warning output

Process failed, see Error output

The Reports View displays detailed information about the process results, including the last process run. In the Tool Reports section, it shows the results of SSO analysis that has been set up in Spreadsheet View. It also reports the results of hierarchy generation and Best Known Methods (BKM) analysis. The Generate Hierarchy and Run BKM Check commands are available from the Design menu.

Figure 74: Reports View of Last Process Run



mixedcounter project summary

Module Name:	mixedcounter	Synthesis:	SynplifyPro
Implementation Name:	verilog_vhdl_edif	Strategy Name:	Strategy1
Last Process:	Place & Route Design	State:	Passed
Target Device:	LFE2-35E-5F672C	Device Family:	LatticeECP2
Device Type:	LFE2-35E	Package Type:	FPBGA672
Performance grade:	5	Operating conditions:	COM
Logic preference file:	mixedcounter_vve.lpf		
Physical Preference file:	verilog_vhdl_edif/mixedcounter_verilog_vhdl_edif.prf		
Product Version:	1.3.78	Updated:	2011/06/10 18:43:16
Implementation Location:	C:/lscc/diamond/1.3/examples/mixedcounter/verilog_vhdl_edif		
Project File:	C:/lscc/diamond/1.3/examples/mixedcounter/mixedcounter.ldf		

Implementation Flow and Tasks

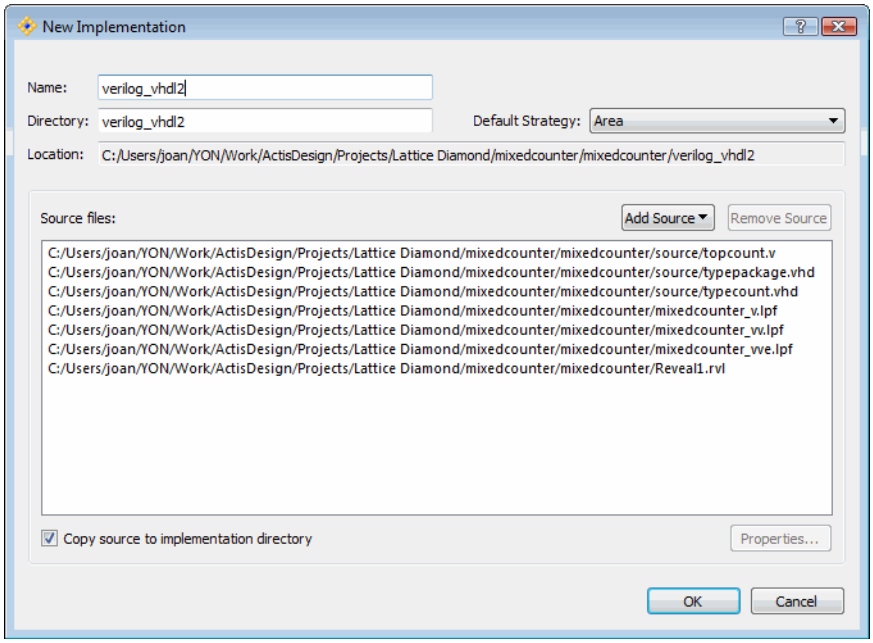
Implementations organize the structure of your design and allow you to try alternate structures and tool settings to determine which ones will give you the best results.

You might want to try different implementations of a design using the same tool strategy, or try running the exact same implementation with different strategies to see which scenario will best meet your project goals. Each

implementation has an associated active strategy, and when you create a new implementation you must select its active strategy.

To try the same implementation with different strategies you will need to create a new implementation/strategy combination. Right-click the project name in the File List and choose **Add > New Implementation**. In the dialog box, the Add Source selection allows you to use source from an existing implementation. The Default Strategy selection allows you to choose from the currently defined strategies.

Figure 75: Adding a Source to a New Implementation

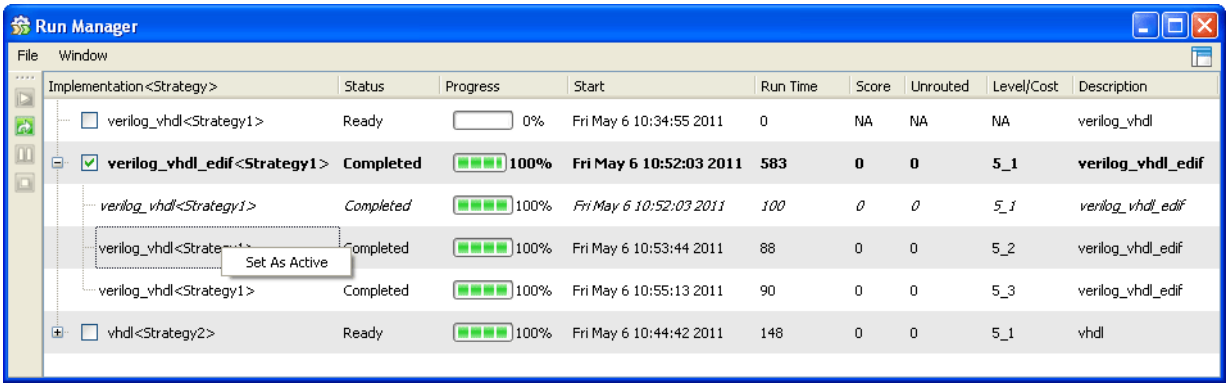


If you want to use the exact same source for the new and the existing implementations, make sure that the “Copy source to implementation directory” option is not selected. This will ensure that your source is kept in sync between the two implementations.

Run Management

Use the Run Manager to run different implementations. Each implementation will use its active strategy. Choose **Tools > Run Manager** or click the Run Manager button on the toolbar.

Figure 76: Run Manager



The Run Manager runs the entire process flow for each selected implementation. If you are running multiple implementations on a multicore system, the run manager will distribute them so that they are executed in parallel.

You can use the run Manager list to set an implementation as active. Right-click the implementation/strategy pair and choose **Set as Active**.

For an implementation that uses multiple iterations of place-and-route, you can select the run that you want to use as the active netlist for further processes. Right-click the desired run, and choose **Set as Active**. Multiple place-and-route (PAR) iterations are specified in the Place & Route section of the active Strategy.

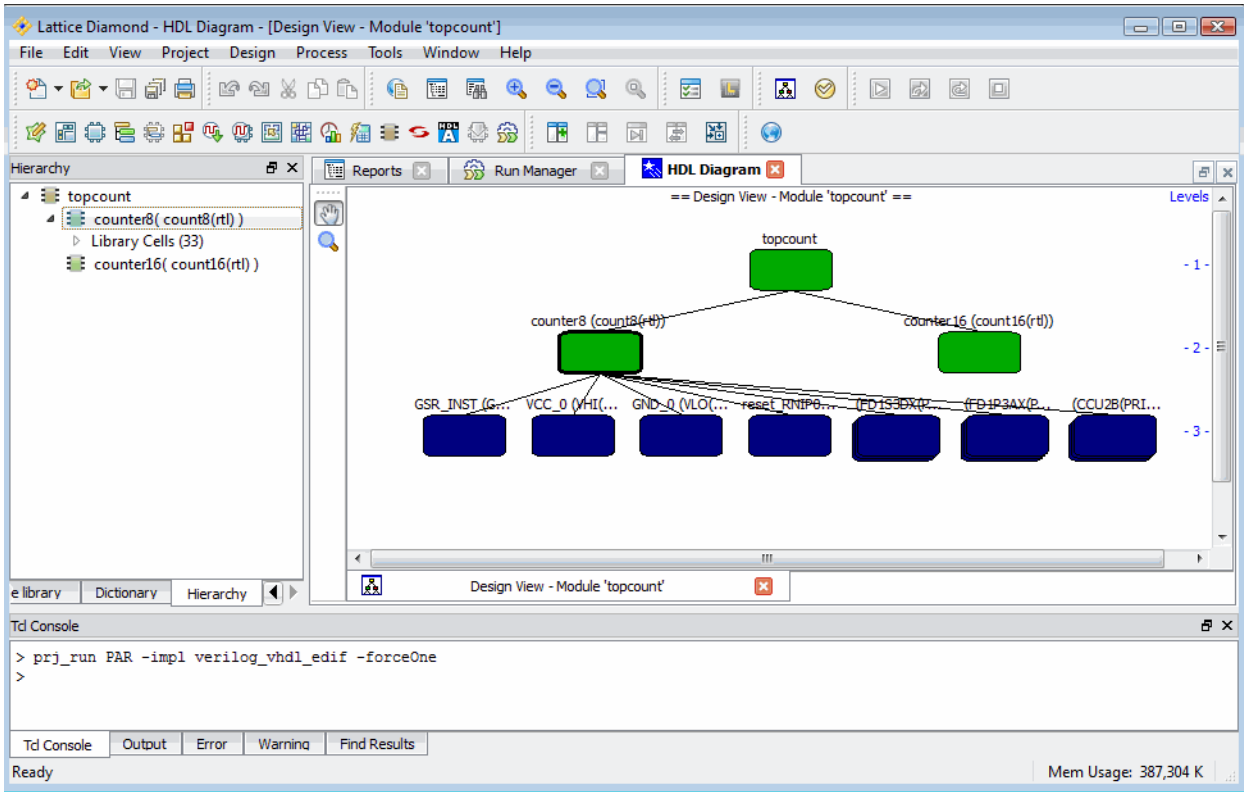
To examine the reports from each process, make an implementation active, and then select the Reports View. For multiple Place & Route iterations, the PAR report is not updated when the iteration is made active.

HDL Design Hierarchy and Checking

The HDL Diagram of your design is displayed whenever you generate the design hierarchy or run BKM Check.

To generate the design hierarchy, choose **Design > Generate Hierarchy** or click the Generate Hierarchy button on the toolbar. If you want to automatically generate the design hierarchy whenever a design is loaded or changed, choose **Design > Auto Generate Hierarchy**. When you choose “Auto Generate Hierarchy,” the hierarchy will be generated not only when the design is loaded, but also whenever the design changes. The generation of the hierarchy data causes the Hierarchy, Module Library and Dictionary views to be displayed in the project view area.

Figure 77: HDL Diagram



Best Known Methods (BKM) are design guidelines that are used to analyze your design. BKM includes the following design checks:

- 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

To run BKM checks, choose **Design > Run BKM Check** from the menu or click the button on the toolbar. If you want to automatically run these checks whenever a design is loaded, choose **Design > Auto Run BKM Check**.

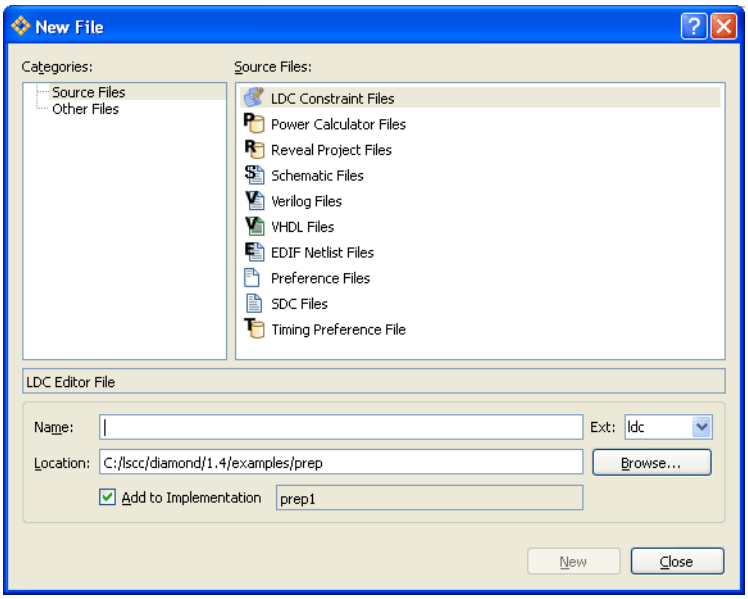
While running a BKM check, errors and warnings are listed in the Output panel. The BKM checks also color-highlight design elements in the graphical and textual views when they have associated BKM violations.

Synthesis Constraint Creation

Synthesis constraints, in the format of the Synopsys® Design Constraint (SDC) language, can be added to a design implementation. If you are using Synplify Pro or Precision for synthesis, the constraints will be included in an .sdc file. If you are using the Lattice Synthesis Engine, which is available for MachXO, MachXO2, and Platform Manager designs, the synthesis will be included in an .ldc file.

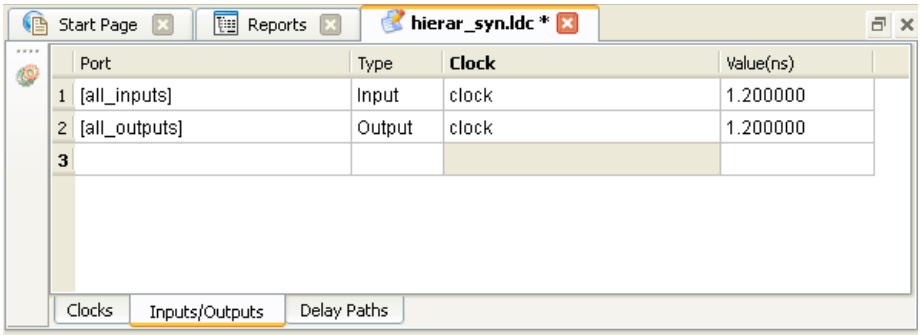
To create a new synthesis constraint file, right-click the Synthesis Constraint Files folder in the File List pane and choose **Add > New File**. In the New File dialog box, select LDC Constraint Files or SDC Files from the Source Files list and give the file a name.

Figure 78: New Synthesis Constraint File



If you selected SDC Files, open the file in the Source Editor to add the constraints. If you selected LDC Constraint Files, the LDC Editor will open displaying the spreadsheet and three tabs for creating and editing synthesis constraints.

Figure 79: LDC Editor



For detailed information about setting SDC constraints, see *Applying Lattice Synthesis Engine Constraints* and the *Constraints Reference Guide* in the Lattice Diamond online Help.

LPF Constraint Creation

The logical preference file (.lpf) is the source file for storing constraints called preferences. Logical preferences in the .lpf file are used as input to post-synthesis processes. These preferences can be created using Diamond's preference-editing views. The following steps illustrate how you might assign and edit logical preferences in Lattice Diamond and implement them at each stage of the design flow.

1. If desired, define some constraints at the HDL level using HDL attributes. These attributes from source files will be included in the EDIF netlist, and they will be displayed in Lattice Diamond after you run the Translate Design process.
2. Open one or more of the following views to create new constraints or to modify existing constraints from the source files and save them as preferences.
   - Spreadsheet View – This is the primary view for setting constraints. Set timing objectives such as fMAX and I/O timing, define signaling standards, and make pin assignments. Assign clocks to primary or secondary routing resources. Set parameters for simultaneous switching outputs for SSO analysis. Define groups of ports, cells, or ASIC blocks. Create UGROUPs from selected instances to guide placement. Establish REGIONs for UGROUPs or for reserving resources.
   - Package View – Examine the pin layout of the design. Modify signal assignments and reserve pin sites that should be excluded from placement and routing. Examine SSO analysis by pin. Run PIO design rule check to check for legal placement of signals to pins.
   - Device View – Examine FPGA device resources. Reserve sites that should be excluded from placement and routing.
   - Netlist View – Examine the design tree by the element names of ports, instances, and nets so that the names can be used when defining preferences. Assign selected signals by dragging them to Package View. Set timing and location constraints. Create UGROUPs of logical instances to guide placement and routing. Set BLOCK preferences for selected nets.
   - Floorplan View – Examine the device layout of the design. Draw bounding boxes for UGROUPs. Draw REGIONs for the assignment of groups or to reserve an area. Reserve sites and REGIONs that should be excluded from placement.

   When you modify preferences in Diamond, the "Preferences Modified" indicator appears in the status bar on the right.
3. Save the preferences to the logical preference file (.lpf).

   The "Preferences Modified" indicator is cleared from the status bar.

4. Run the Map Design process (map). This process reads the .ngd and .lpf files and produces a native circuit description file (.ncd) and a physical preference file (.prf). The .prf file is an internal file generated by the Map engine that contains preferences used by the PAR engine. The .prf file is is not intended to be edited by the user because edits will be lost when it is regenerated.
5. Run the Map TRACE process and examine the timing analysis report. This is an optional step, but it can be a quick and useful way to identify serious timing issues in design and/or preference errors (syntax & semantic). Modify preferences as needed and save them.
6. Run the Place & Route Design process (par). This process reads the post-MAP .ncd file and the .prf file, and it appends placement and routing to a post-PAR .ncd file.
7. Open views directly or by cross-probing to examine timing and placement and create new UGROUPs. Also examine the Place & Route Trace report.
   - Timing Analysis View – Examine details of timing paths. Cross probe selected paths to Floorplan and Physical Views. Create one or more timing preference files (.tpf) and experiment with sets of modified preferences for the purpose of timing analysis, using the TPF Spreadsheet View. Copy the best results to the .lpf file.
   - NCD View– Examine placement assignments. Cross probe to Physical View and open detailed views of selected blocks. Create new UGROUPs, as needed, from selected instances in NCD View.
8. Modify preferences or create new ones using any of the views. Save the preference changes and rerun the Place & Route Design process.

Simulation Flow

The simulation flow in Diamond 1.3 or later has been enhanced to support source files that can be set in the File List view to be used for the following purposes:

- Simulation & Synthesis (default)
- Simulation only
- Synthesis only

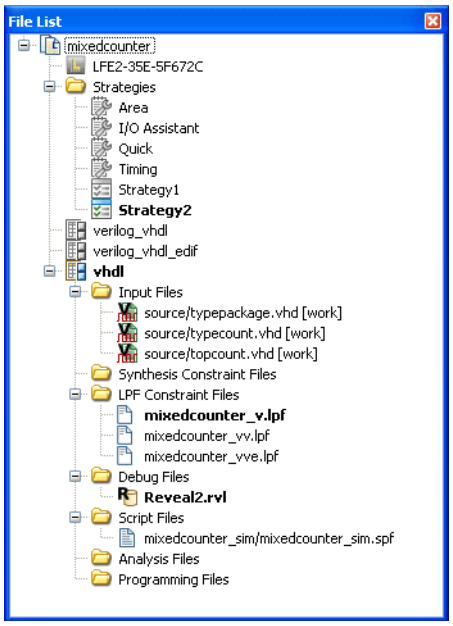
This allows the use of testbenches, including multiple file testbenches. Additionally, multiple representations of the same module can be supported, such as one for simulation only and one for synthesis only.

The Simulation Wizard has been enhanced to support these changes. The Simulation Wizard will automatically include any files that have been set for simulation only or for both simulation and synthesis. The user can select the top of the design for simulation independent of the implementation design top. This allows easy support for testbench files, which are normally at the top of the design for simulation but not included for implementation. The implementation wizard will export the design top to the simulator, along with

source files, and set the correct top for the SDF file if running timing simulation.

The File List view shows an implementation's input files for simulation. This is a listing of source files and does not show design hierarchy.

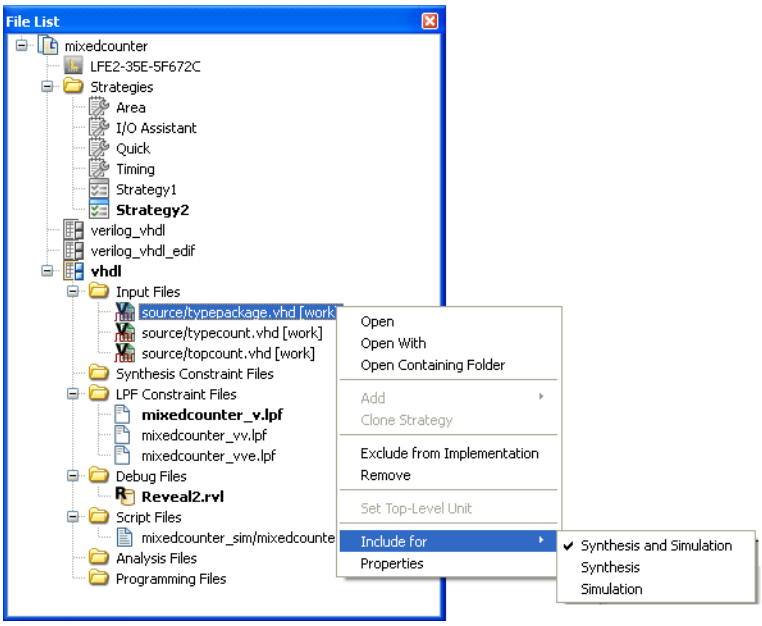
Figure 80: Input Files for Simulation



After you add a module, use the **Include For** menu to specify how the module file is to be used in the design.

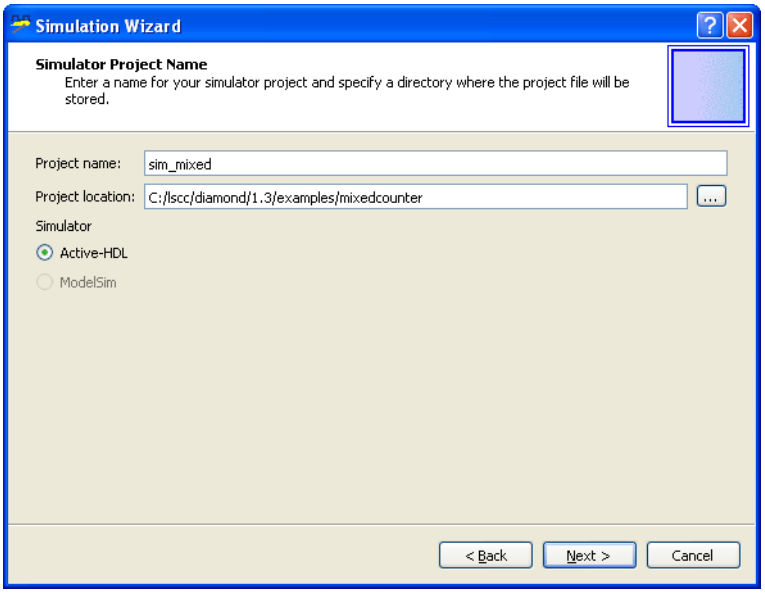
When you are ready to simulate, you can export the design using the Simulation Wizard. Choose **Tools > Simulation Wizard** or click the Simulation Wizard button on the toolbar .The wizard will lead you through a series of steps: selecting a simulation project name and location, specifying the simulator to use (if you have more than one installed), selecting the process stage to use (from RTL to Post-Route Gate-Level + Timing), and

Figure 81: Include For Commands



selecting the language (VHDL or Verilog) and source files. You can also run the simulation directly from the wizard.

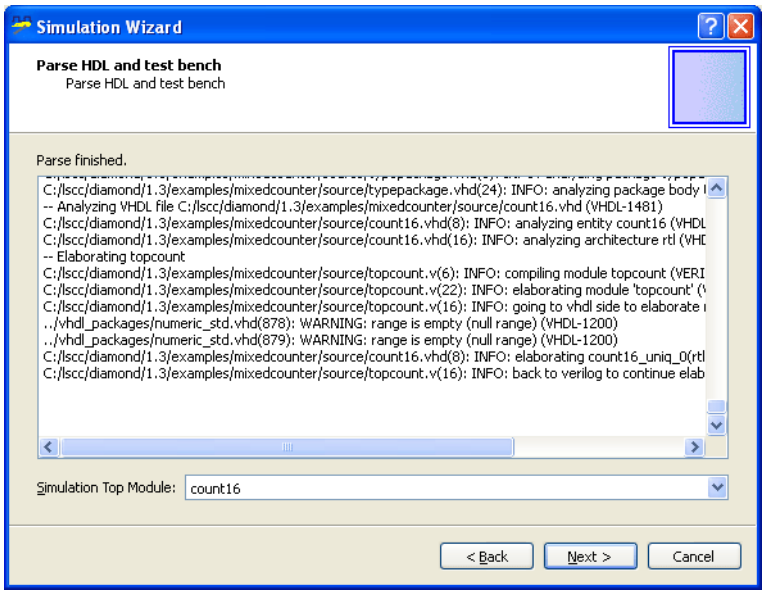
Figure 82: New Simulation Project



After you have set up the simulator project and specified the implementation stage and source files to be included, the Simulation Wizard parses the HDL and test bench. The last step is to specify the simulation top module.

In some designs, the compile order of the HDL files passed to the simulator might result in compilation warnings. In most cases, these compilation

Figure 83: Simulation Top Module



warnings can be safely ignored. The warnings can be eliminated in one of two ways:

- The correct compilation order for the HDL files can be set in the File List view. After the correct order for the files is set manually, the files will be sent to the simulator, which will eliminate any compilation warnings.
- The correct compilation order for the HDL files can be set in the Simulation Wizard during the "Add and Reorder" step. After the correct order for the files is set manually, the files will be sent to the simulator, which will eliminate any compilation warnings.

I/O Assistant Flow

Defining a device pinout can be a complicated process because of constraints in the PC board layout and the FPGA architecture, and it is typically done long before the entire FPGA design is complete. The I/O Assistant, a special predefined strategy within Diamond, assists you with this task, enabling you to produce an FPGA-verified pinout early on based upon PC board layout requirements.

The I/O Assistant strategy helps you select a legal device pinout and produce LOCATE and IOBUF preferences for optimal I/O placement. The only design content required to validate an I/O plan is an HDL model of the I/O ports. Details of the internal logic can be treated as a black box. The primary output of the I/O Assistant flow is a validated placement of I/O signals that can be back annotated to the logical preference file.

The I/O Assistant strategy is a read-only predefined set of properties for the design flow. The following sequential steps are typical for the I/O Assistant design flow:

1. Create a top-level module in HDL that describes all of the ports in the design. You can do so manually or use the I/O modules generated by IPexpress.
2. Make the I/O Assistant strategy active for your project. From the Strategies folder in the File list pane, right-click **I/O Assistant** and choose **Set as Active Strategy**.
3. Synthesize your HDL as you would normally.

   If you are using Synplify Pro, Lattice Diamond will automatically pass the required attributes and header files for I/O Assistant flow when you run the Translate Design process. It will also automatically pass the required attributes and header files if you are using the integrated Lattice Constraint Engine (LSE) for a MachXO, MachXO2, or Platform Manager design.

   If you are running synthesis in stand-alone mode, you will need to include these attributes and header library files in the source code before synthesis. See the synthesis tool documentation for more information.
4. Constrain your design to add banking location preferences, I/O types, I/O ordering, and minor customizations. You can set these preferences using Spreadsheet View or you can do this manually.

   - To set the preferences in Spreadsheet View, choose **Tools > Spreadsheet View** and edit the I/Os.
   - To set I/O preferences manually, double-click the name of the project's logical preference file (.lpf) from the Constraints folder in the File List view.

   When implementing DDR interfaces, it is recommended that you generate the required DDR modules using IPExpress along with the port definitions. This will enable the tool to check for any DDR-related rules that are being violated.
5. Run the Place & Route Design process.

   The process maps and places the I/Os based on the preferences, the I/O Assistant strategy, and the architectural resources. The output is a pad report (.pad) to guide future placement and a placed and routed native circuit description .ncd that contains only I/Os.
6. Examine the I/O Placement results by doing one or more of the following:

   - From the Process Reports folder in the Reports window, select **Signal/Pad** to open the PAD Specification File and examine the pinout.
   - Choose **Tools > Package View**, and then choose **View > Display IO Placement** to view the pin assignments on the layout to cite areas for minor customization.

   To view the results of timing constraints:

   - Run the **Place & Route Trace** process and open the Place & Route Trace report from the Reports window.

- Run the **I/O Timing Analysis** process and open the I/O Timing Report from the Reports window.

7. Make any needed adjustments to the I/O preferences, as you did in Step 4.
8. Rerun **Place & Route Design**.
9. Repeat steps 5 through 7 as necessary to achieve your I/O placement objectives.
10. From Package View, choose **Design > Backannotate Assignments** to copy the I/O preferences to the logical preference file, and then choose **File > Save**.

    I/O placement preferences are written to the end of the .lpf file and will take precedence over any existing preferences that may conflict with them.
11. Create a new strategy or add an existing one. Set the strategy as the active one, and take your design through the regular flow.

Summary of Changes from ispLEVER

Lattice Diamond is the next-generation FPGA design environment, replacing the ispLEVER tool. Although the design processes are very similar between the two environments, there are a number of improvements and differences to be aware of if you are an experienced ispLEVER user.

- Synthesize Design and Translate Design steps in Lattice Diamond replace the Build Database step in ispLEVER.
- Exporting designs to simulation is done with the Simulation Wizard in Lattice Diamond.
- Timing analysis can be performed using the Timing Analysis View in Lattice Diamond without having to re-implement the design. See “Timing Analysis View” on page 102 for more details.
- Design hierarchy is not generated by default. You must use the Generate Hierarchy command.
- Reports from the design process steps are viewed independently of the process state. Therefore, viewing a process report will not cause a process to be rerun. In ispLEVER, viewing a report causes a process to be rerun.

7

Working with Tools and Views

This chapter covers the tools and views controlled from the Lattice Diamond framework. Tool descriptions are included and common tasks are described.

Overview

The Lattice Diamond design environment streamlines the implementation process for FPGAs by combining the tool control and data views into one common location. Two main features of this design environment make it easy to keep track of unsaved changes in your design and examine data objects in different view.

Shared Memory

Lattice Diamond uses shared memory that is accessed by all tools and views. As soon as design data has been changed, an asterisk * appears in the tab title of the open views, notifying you that unsaved changes are in memory.

Cross Probing

Shared design data in Lattice Diamond enables you to select a data object in one view and display it in other views. This cross-probing capability is especially useful for displaying the physical location of a component or net after it has been implemented.

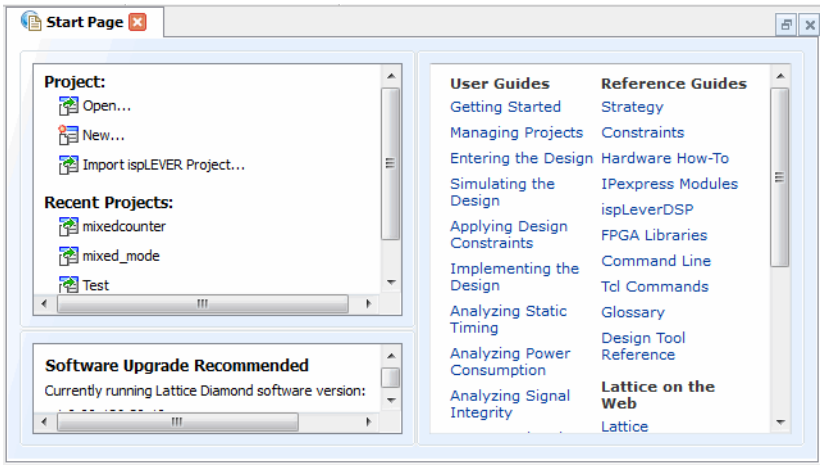
View Menu Highlights

The View menu and toolbar control the display of toolbars, project views and display control. Also included in the View menu are the important project-level features Start Page, Reports and Preference Preview.

Start Page

The Start Page is displayed by default when you run Lattice Diamond. The three panes within the Start Page enable you to open projects, read product documentation, and view the software version and updates. You can modify startup behavior by choosing **Tools > Options**.

Figure 84: Start Page

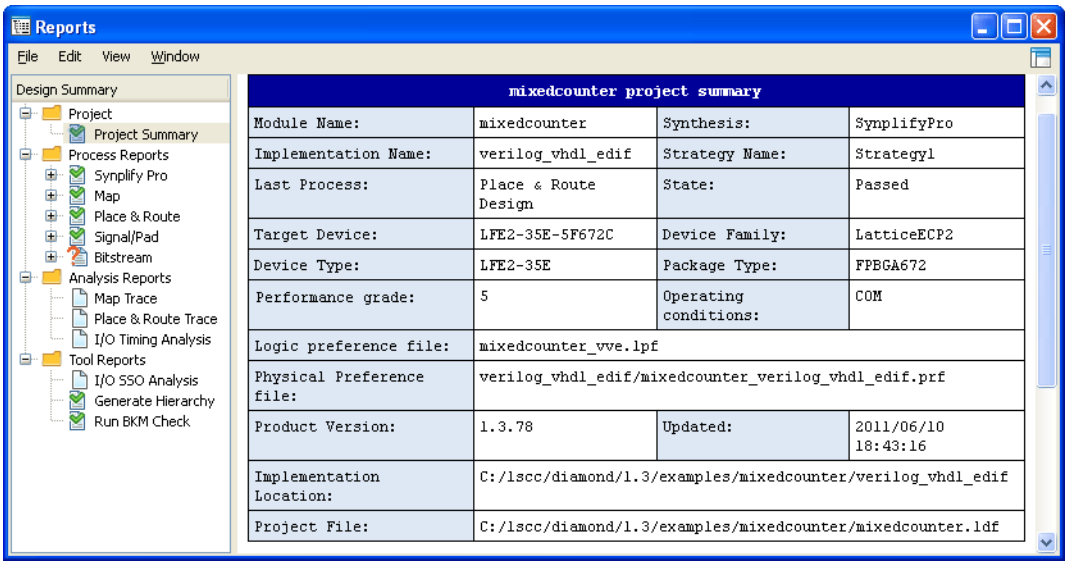


The Start Page gives you quick access to recent projects and to product documentation. It can be opened, closed, detached and attached (using the icon method).

Reports

The Reports View provides one central location for all project and tool report information. It is displayed by default when a project is open.

Figure 85: Reports View



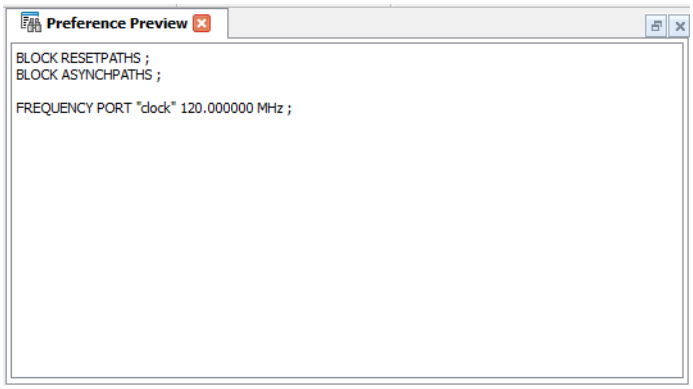
The Design Summary pane of the Reports View is organized into Project, Process Reports, Analysis Reports, and Tool Reports. The different file icons indicate whether a report has been completed (green check mark), has never been generated (blank note paper), or is out of date (orange question mark). Select any item to see its report.

The Reports View is the primary, central view for all process report information. It can be opened, closed, detached and attached (using the icon method).

Preference Preview

Preference Preview shows the design's logical preferences as they exist in shared memory, which includes unsaved preferences as well as those in the logical preference file (.lpf). When you first open Lattice Diamond, Preference Preview shows the contents of the active .lpf file. As you use preference views such as Spreadsheet View and Package View to add or modify constraints, Preference Preview reflects those changes. What you see in Preference Preview will be reflected in the .lpf file when you use the Save command.

Figure 86: Preference Preview



The Preference Preview is a read-only view of logical preferences and can be opened, closed, detached and attached (using the attach button).

Tools

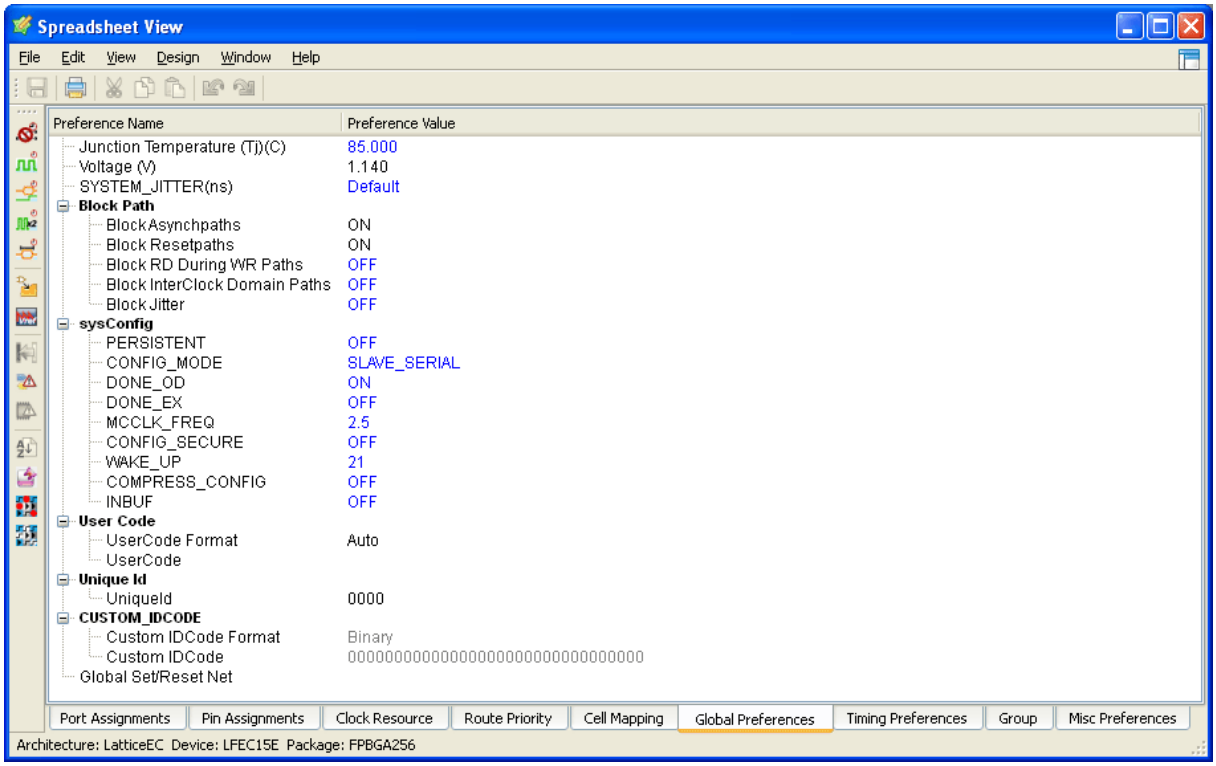
The entire FPGA implementation process tool set is contained in Lattice Diamond. You can run a tool by selecting it from the Tools menu or toolbar.

This section provides an overview of each of these tools. More detailed information is available in the user guides, which you can access from the Start Page or from the Lattice Diamond Help. Detailed descriptions of external tools can be found in their product documentation.

Spreadsheet View

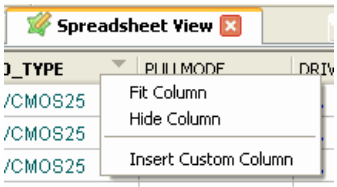
Spreadsheet View provides an interactive spreadsheet format for viewing and assigning design constraints. Its collection of preference sheets enables you to assign preferences such as PERIOD, FREQUENCY, I/O timing, and LOCATE to optimize placement and routing. Preferences can also be set for SSO Analysis and clock jitter.

Figure 87: Spreadsheet View



The Port and Pin Assignments sheets allow you to view I/Os by signal or pin attributes and use the Assign Pins or Assign Signals functions to make assignments. Custom columns are also available on the Port and Pin Assignments sheet. Custom columns enable you to add your own information, such as notes for specific signals or pins or design data for third-party tools. You can create an unlimited number of custom columns, and you can include the column when you export to a Lattice CSV file or a Pin Layout File. Right-click any column heading to add a custom column.

Figure 88: Adding a Custom Column



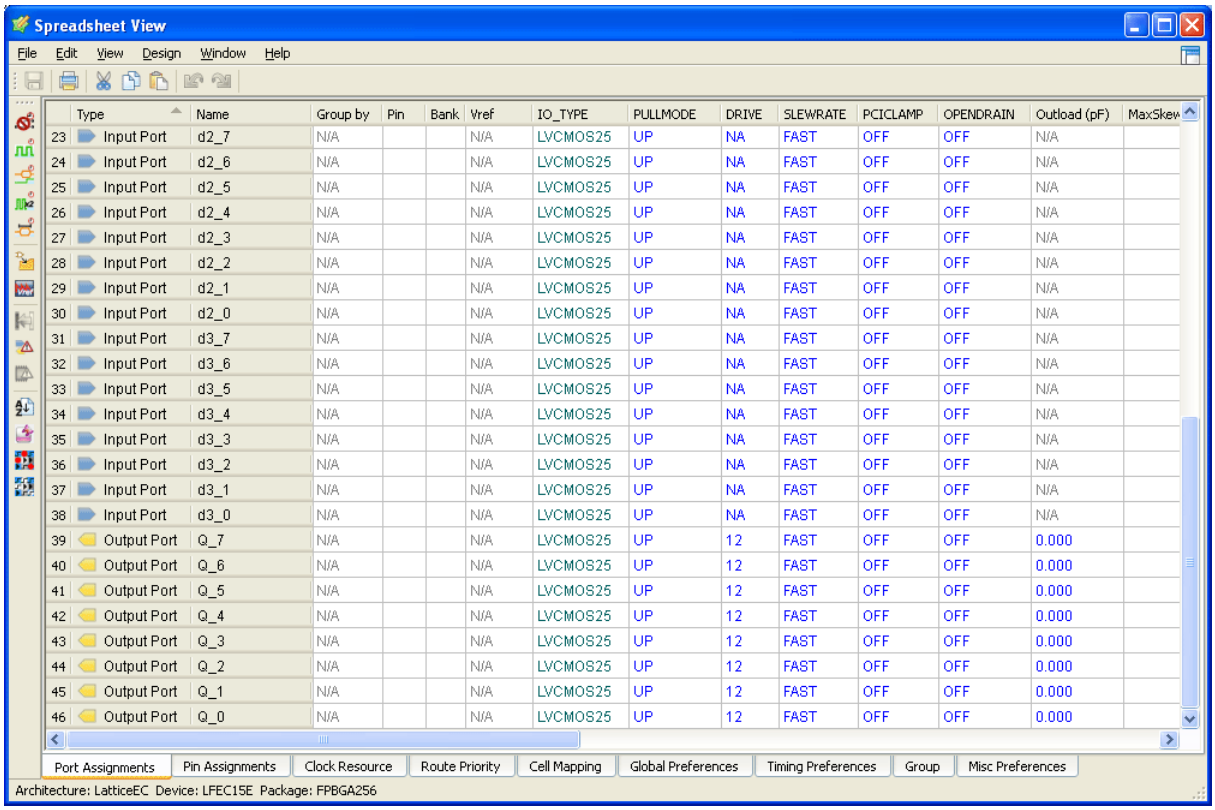
As soon as the target device has been specified, Spreadsheet View enables you to set global preferences. After synthesis and translation, it allows you to explore other devices of the same family for possible pin migration, as explained in “Pin Migration” on page 134.

After synthesis and translation, all of the following preference sheets become available for editing:

Port Assignments

The Port Assignments sheet provides a signal list of the design and shows any pin assignments that have been made. It enables you to assign or edit pin locations and other attributes by entering them directly on the spreadsheet. It also enables you to assign pins in the Assign Pins dialog box by right-clicking selected signals and selecting **Assign Pins** from the pop-up menu.

Figure 89: Spreadsheet View Port Assignments

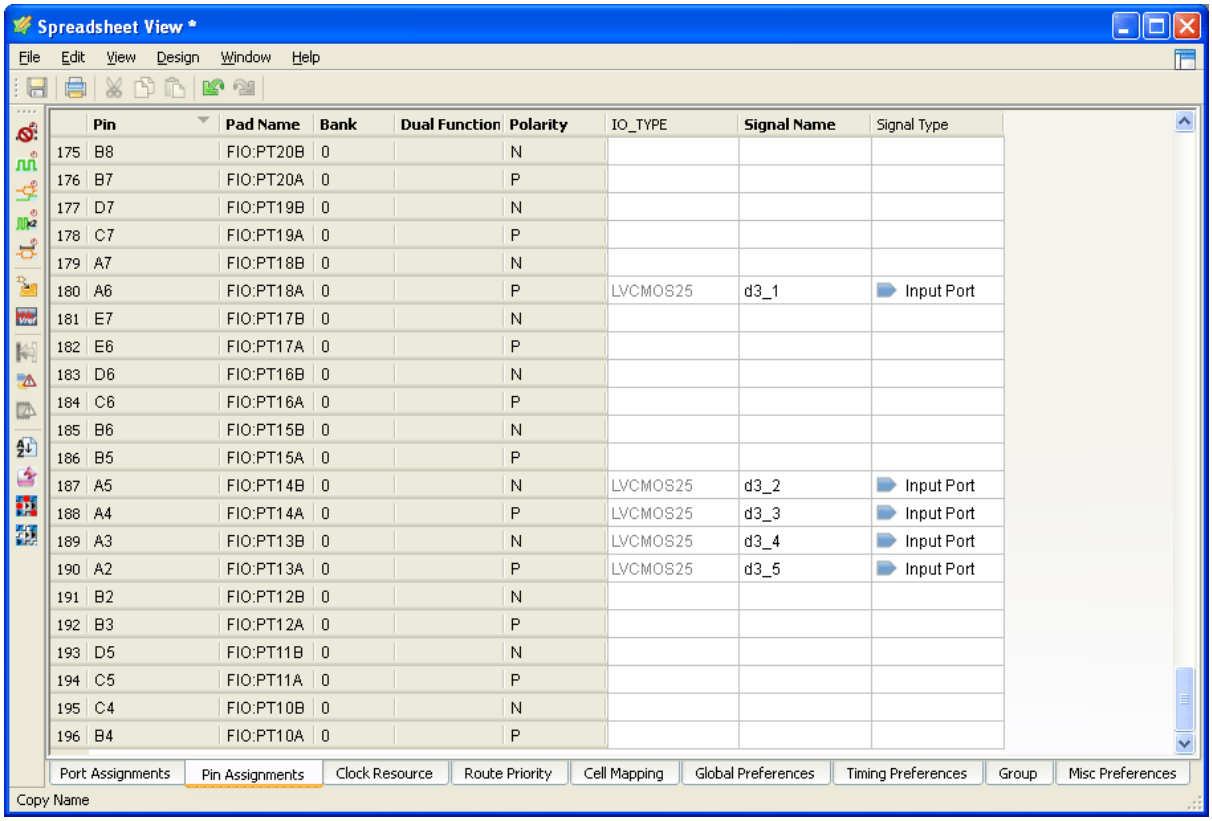


	Type	Name	Group by	Pin	Bank	Vref	IO_TYPE	PULLMODE	DRIVE	SLEWRATE	PCICLAMP	OPENDRAIN	Outload (pF)	MaxSkew
23	Input Port	d2_7	N/A			N/A	LVCMOS25	UP	NA	FAST	OFF	OFF	N/A	
24	Input Port	d2_6	N/A			N/A	LVCMOS25	UP	NA	FAST	OFF	OFF	N/A	
25	Input Port	d2_5	N/A			N/A	LVCMOS25	UP	NA	FAST	OFF	OFF	N/A	
26	Input Port	d2_4	N/A			N/A	LVCMOS25	UP	NA	FAST	OFF	OFF	N/A	
27	Input Port	d2_3	N/A			N/A	LVCMOS25	UP	NA	FAST	OFF	OFF	N/A	
28	Input Port	d2_2	N/A			N/A	LVCMOS25	UP	NA	FAST	OFF	OFF	N/A	
29	Input Port	d2_1	N/A			N/A	LVCMOS25	UP	NA	FAST	OFF	OFF	N/A	
30	Input Port	d2_0	N/A			N/A	LVCMOS25	UP	NA	FAST	OFF	OFF	N/A	
31	Input Port	d3_7	N/A			N/A	LVCMOS25	UP	NA	FAST	OFF	OFF	N/A	
32	Input Port	d3_6	N/A			N/A	LVCMOS25	UP	NA	FAST	OFF	OFF	N/A	
33	Input Port	d3_5	N/A			N/A	LVCMOS25	UP	NA	FAST	OFF	OFF	N/A	
34	Input Port	d3_4	N/A			N/A	LVCMOS25	UP	NA	FAST	OFF	OFF	N/A	
35	Input Port	d3_3	N/A			N/A	LVCMOS25	UP	NA	FAST	OFF	OFF	N/A	
36	Input Port	d3_2	N/A			N/A	LVCMOS25	UP	NA	FAST	OFF	OFF	N/A	
37	Input Port	d3_1	N/A			N/A	LVCMOS25	UP	NA	FAST	OFF	OFF	N/A	
38	Input Port	d3_0	N/A			N/A	LVCMOS25	UP	NA	FAST	OFF	OFF	N/A	
39	Output Port	Q_7	N/A			N/A	LVCMOS25	UP	12	FAST	OFF	OFF	0.000	
40	Output Port	Q_6	N/A			N/A	LVCMOS25	UP	12	FAST	OFF	OFF	0.000	
41	Output Port	Q_5	N/A			N/A	LVCMOS25	UP	12	FAST	OFF	OFF	0.000	
42	Output Port	Q_4	N/A			N/A	LVCMOS25	UP	12	FAST	OFF	OFF	0.000	
43	Output Port	Q_3	N/A			N/A	LVCMOS25	UP	12	FAST	OFF	OFF	0.000	
44	Output Port	Q_2	N/A			N/A	LVCMOS25	UP	12	FAST	OFF	OFF	0.000	
45	Output Port	Q_1	N/A			N/A	LVCMOS25	UP	12	FAST	OFF	OFF	0.000	
46	Output Port	Q_0	N/A			N/A	LVCMOS25	UP	12	FAST	OFF	OFF	0.000	

Pin Assignments

The Pin Assignments sheet provides a pin list of the device and shows the signal assignments that have been made. It enables you to edit signal assignments or assign new signals by right-clicking selected pins and selecting **Assign Signals** from the pop-up menu.

Figure 90: Spreadsheet View Pin Assignments



Clock Resource

The Clock Resource sheet enables you to apply a clock domain to the device's primary or secondary clock or prohibit the use of primary and secondary clock resources to route the net. For LatticeECP2 devices, it enables you to use edge clock resources. For LatticeECP3 devices, it enables you to assign a secondary clock to a clock REGION that has already been defined.

Route Priority

The Route Priority sheet enables you to set the PRIORITIZE preference, which assigns a weighted importance to a net or bus. To set this preference, drag the desired nets from Netlist View to the Route Priority sheet. You can then select a priority value for each net. Values range from 0 to 100.

Cell Mapping

The Cell Mapping sheet enables you to set the USE DIN and USE DOUT cell preferences for flip-flops in your design. The PIO Register column allows you to set the register to True or False. The True setting moves registers into the I/Os. The False setting moves registers out of the I/Os. To set these preferences, drag the desired registers from Netlist View to the Cell Mapping sheet.

Global Preferences

The Global Preferences sheet enables you to set preferences that affect the entire design, such as junction temperature and voltage; BLOCK preferences applied to all paths of a particular type; and USERCODE. Also included in the Global sheet are sysCONFIG preferences for FPGA devices that support the sysCONFIG configuration port.

Timing Preferences

The Timing Preferences sheet displays all timing preferences that have been set in the design, including BLOCK preferences for specific nets, FREQUENCY, PERIOD, INPUT_SETUP, CLOCK_TO_OUT, MULTICYCLE, and MAXDELAY. You can create a new timing preference by double-clicking the preference name, which opens the dialog box. To modify an existing timing preference, double-click the preference name, edit the information in the dialog box, and click the Update button.

Group

The Group sheet displays any groups that have been created and enables you to define a new cell, port, or ASIC group or create a new universal group (UGROUP). Double-click the group type to open the dialog box and create a new group preference. To modify an existing group preference, double-click the group name, edit the information in the group dialog box, and click the Update button.

Misc Preferences

The Miscellaneous sheet enables you to define REGIONs, assign Vref locations, and reserve resources by setting a PROHIBIT preference. To set a new miscellaneous preference, double-click the preference type to open the

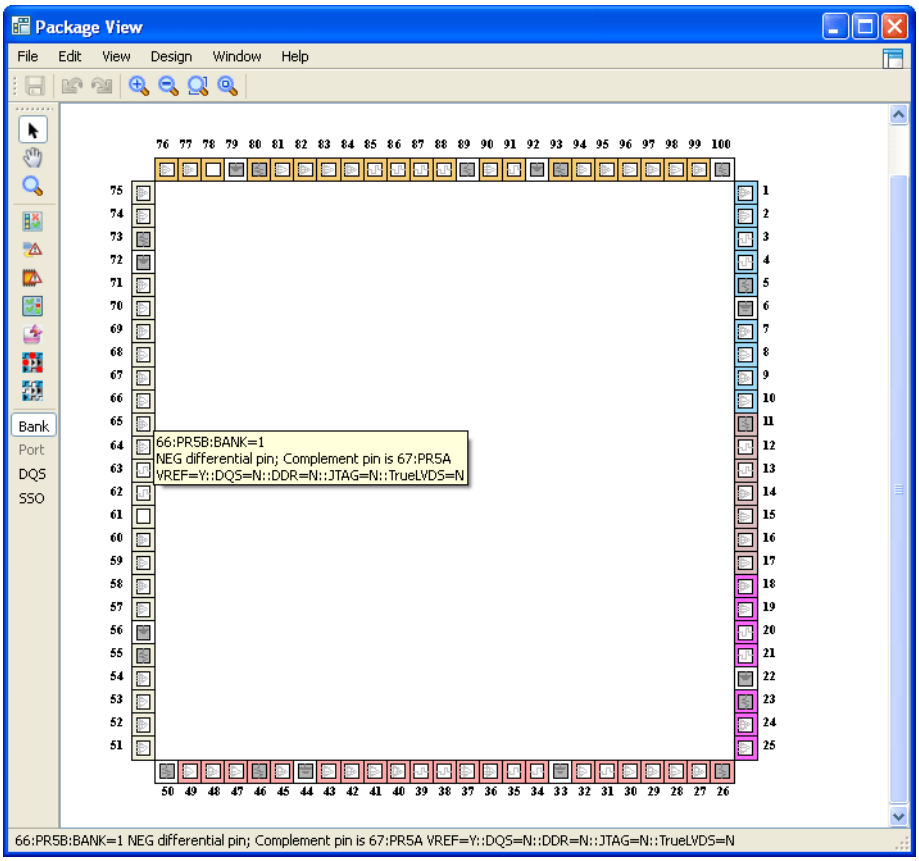
dialog box. To modify an existing miscellaneous preference, double-click the preference name, edit the information in the dialog box, and click the Update button.

Package View

Package View shows the pin layout of the target device and displays the assignments of signals to device pins. Package View interacts with Netlist View for assigning pins, enabling you to drag selected signals to the desired locations on the pin layout to establish LOCATE preferences. Each pin that is assigned with a LOCATE preference is color-coded to indicate the port direction of the related signal port. Package View allows you to edit these assignments, and it allows you to reserve sites on the layout that you want to exclude from placement and routing.

After synthesis and translation, Package View allows you to explore other devices of the same family for possible pin migration, as explained in “Pin Migration” on page 134.

Figure 91: Package View



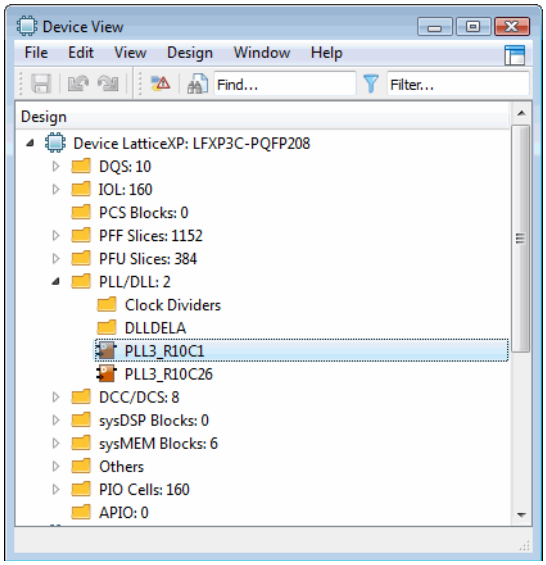
As you move your mouse pointer over the layout, pin descriptions and locations are displayed in tooltips and in the status bar. The **View > Show Differential Pairs** command displays fly wires between differential pin pairs and identifies the positive differential pins.

Package View is available as soon as the target device has been specified.

Device View

Device View provides a categorized index of device resources based on the target device. Statistics cover System PIOs, User PIOs, PFUs, PFFs, sysDSP blocks, sysMEM blocks, IOLOGIC, PLL/DLLs, and other embedded ASIC blocks. Device View enables you to use the Prohibit command to reserve sites that you want to exclude from placement and routing.

Figure 92: Device View



From Device View, you can cross probe selected resources to their sites in Package View, Floorplan View, and Physical View.

Device View is available as soon as the target device has been specified.

Netlist View

Netlist View displays the design elements of the post-synthesis native generic database (NGD) netlist. The NGD is a binary speed-optimized data structure that is used by the system to browse the logical netlist.

Netlist View organizes the netlist by ports, instances, and nets, and it provides a toolbar button and design tree view for each of these categories to make it easier to create timing or location preferences.

Each design tree view is equipped with utilities for filtering the list and searching for elements.

From Netlist View, you can drag selected signals to Package View to assign them, drag selected nets to Spreadsheet View's Route Priority sheet to prioritize them, and drag registers to the Cell Mapping sheet to specify registers for flip-flops. You can use the right-click menu to set timing

Figure 93: Netlist View Ports Design Tree

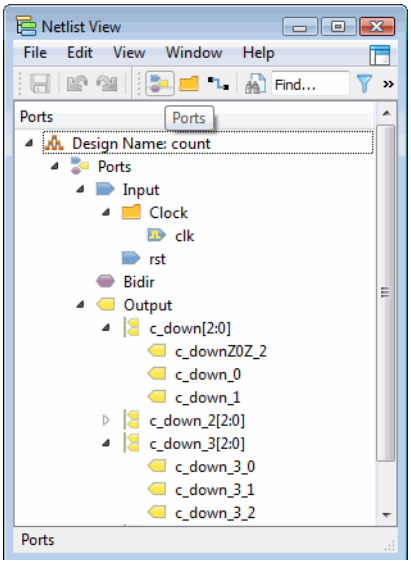
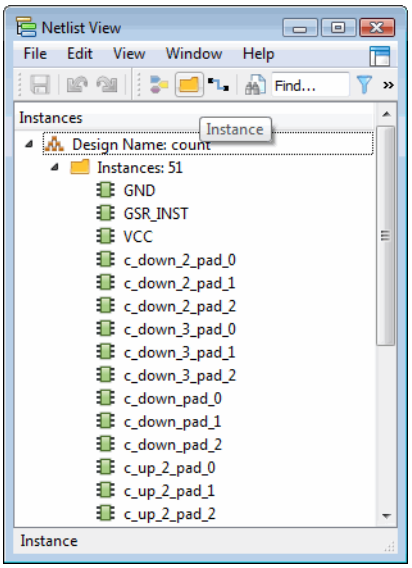


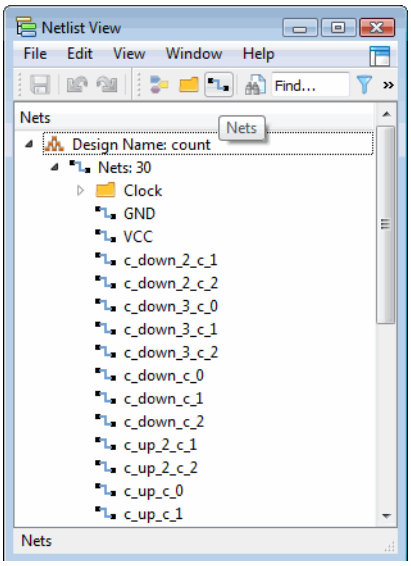
Figure 94: Netlist View Instances Design Tree



preferences for selected nets and to create logical groups from selected instances.

Netlist View is available after synthesis and translation.

Figure 95: Netlist View Nets Design Tree

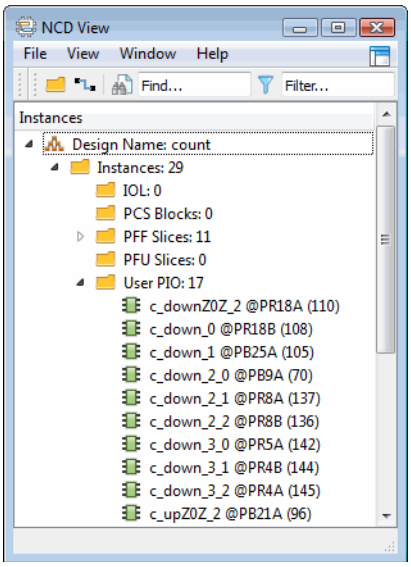


NCD View

NCD View provides a categorized index of synthesized design resources and consumption based on the target device and the in-memory native circuit description (NCD) database. Statistics cover System PIOs, User PIOs, PFUs, PFFs, sysDSP blocks, sysMEM blocks, IOLOGIC, PLL/DLLs, and other embedded ASIC blocks.

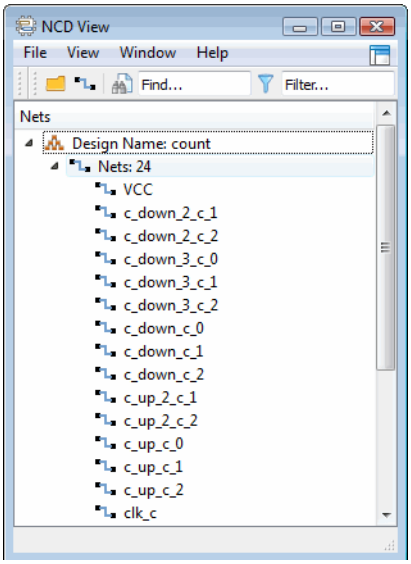
NCD View is organized by nets and instances and provides a toolbar button and design tree view for each of these categories.

Figure 96: NCD View Instances Design Tree



Each design tree view is equipped with utilities for filtering the list and searching for elements.

Figure 97: NCD View Nets Design Tree



From NCD View, you can create a new UGROUP preference from selected instances. You can also access schematic or tabular detailed views for selected instances.

NCD View is available after a successful run of Place & Route.

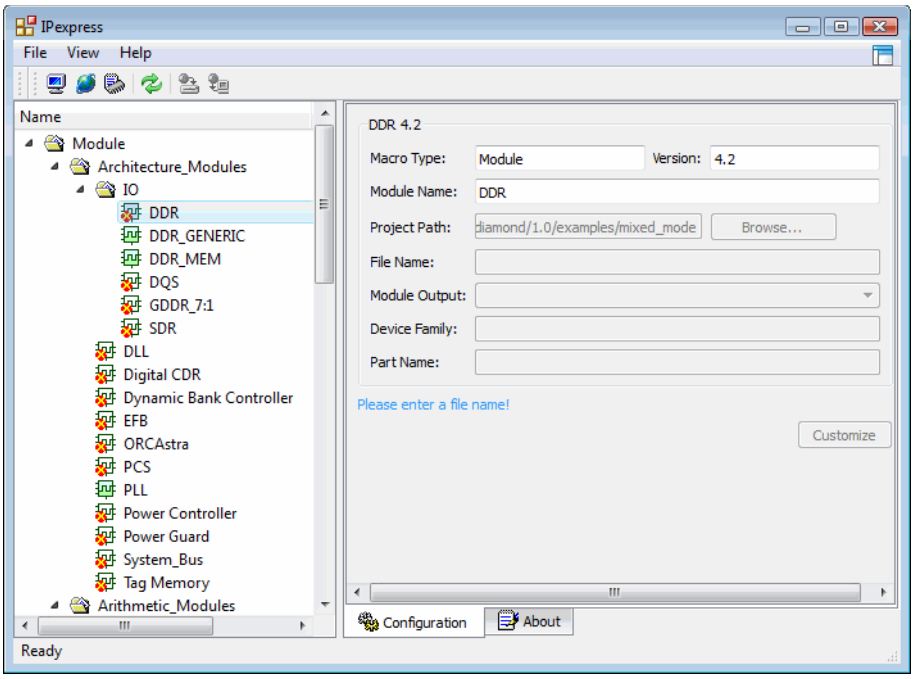
IPexpress

IPexpress is a collection of functional modules that can be used to generate Verilog or VHDL source for use in your design. Modules are functional blocks of design that can be reused wherever that function is needed. They are optimized for Lattice device architectures and can be customized. Use these modules to speed your design work and to get the most effective results.

Many basic modules are included in IPexpress. They provide a variety of functions including I/O, arithmetic, memory, and more. A recommended way to use IP express is to select the import module option which will include an ipx file in your source list. This file can be used to regenerate the module for any changes.

Choose **Tools > IPexpress** to see the full list of available modules.

Figure 98: IPexpress

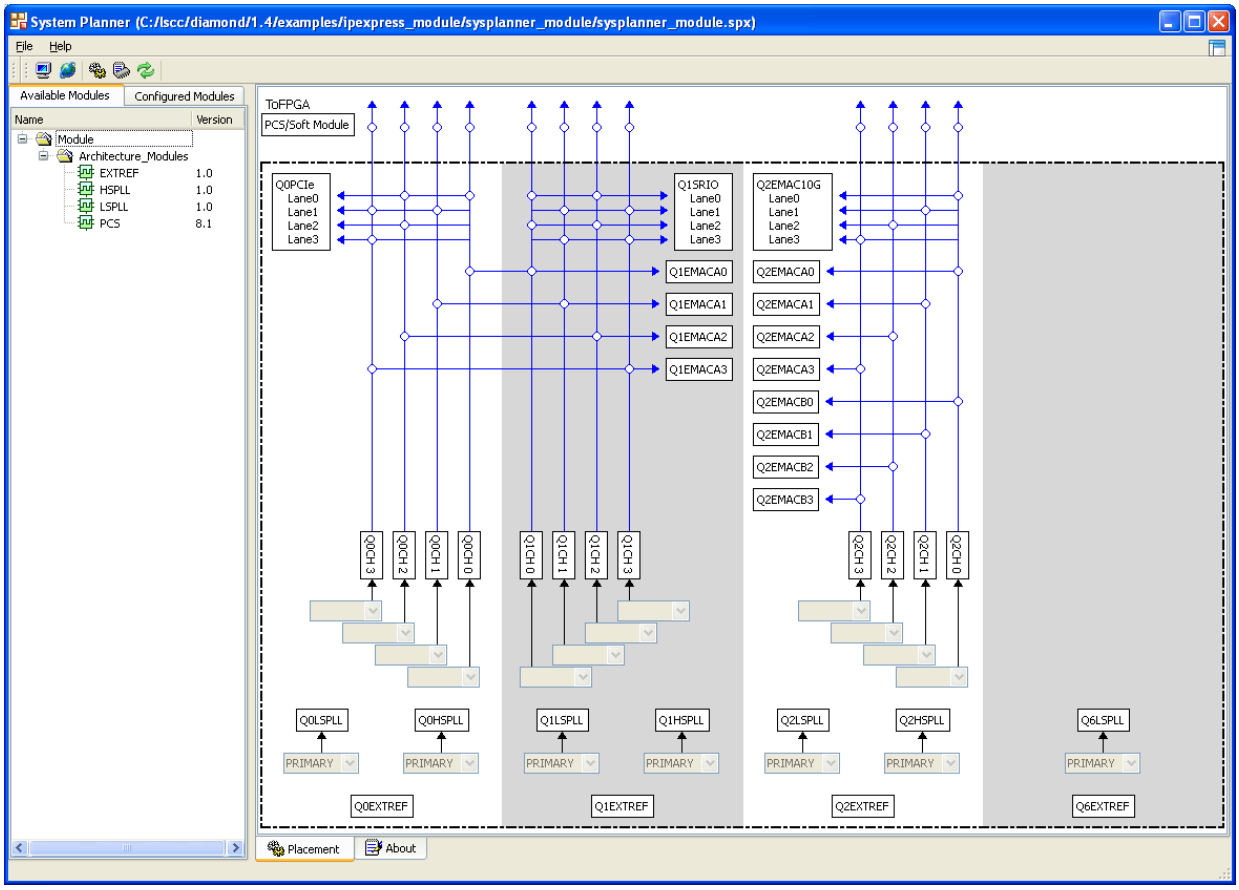


In addition to the included modules, an IP Server button is provided that enables you to connect to the Lattice IP server, explore the available IP, and select those that you would like to download and install. Lattice IP can be purchased or used in a trial mode. Refer to the Lattice Web site for a list of the available IP functions. For more information, see the *IPexpress Module Reference Guide*.

System Planner

System Planner enables you to customize IP that use the PCS/serdes channels, MACO cores, and ASB PLLs of LatticeECP4 designs. Choose **Tools > System Planner** to see the list of available System Planner modules.

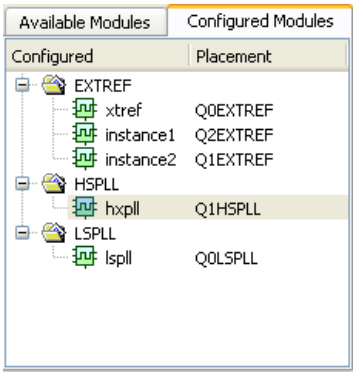
Figure 99: System Planner



To configure a module, double-click the module name, enter an Instance name, and click **Customize**. In the dialog box, select from the available options, and click **Configure**.

Select the Configuration tab to view all the configured module instances.You can then place a module instance by dragging it from the Configured Modules list over to the Placement layout.

Figure 100: System Planner Configured Modules

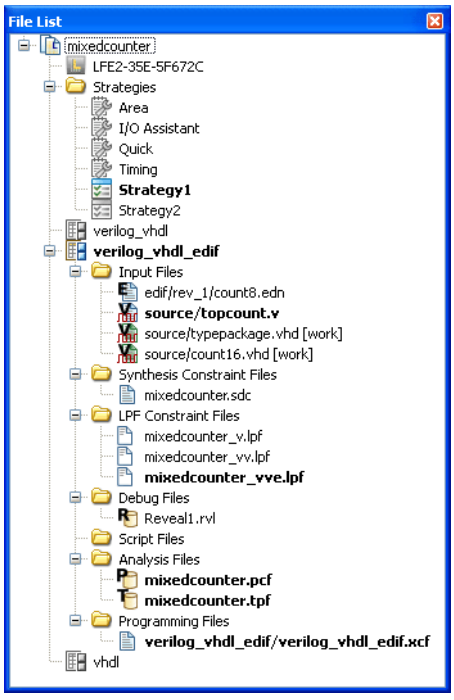


Reveal Inserter

Reveal Inserter allows you to add debug information to your design to allow hardware debugging using Reveal Analyzer. Reveal Inserter enables you to select the design signals to use for tracing, triggering, and clocking. Reveal Inserter will automatically generate the debug core(s), and insert it into a modified design with the necessary debug connections and signals. Reveal Inserter supports VHDL, Verilog, and EDIF sources. Mixed-HDL designs are represented by the synthesis EDIF netlist. After the design has been modified for debug, it is mapped, placed and routed with the normal design flow in Lattice Diamond.

The File List Debug file folder contains the debug files for Reveal Inserter.

Figure 101: File List View with Reveal Debug Project File



One or none of the debug files can be active at a time. If no debug file is

active, hardware debug will not be inserted into the design when it is implemented.

Launching Reveal Inserter

To launch Reveal Inserter, choose **Tools > Reveal Inserter** or click the Reveal Inserter button on the toolbar. When Reveal Inserter is launched, the debug file it uses will depend on the following conditions:

- If an active debug file exists, it will be used. An inactive debug file will be used if it is the only one available.
- If an active or inactive debug file in the File List Debug folder is double-clicked, it will be used.
- If multiple debug files exist and no debug file is set as active, a dialog box will enable you to select one of the inactive debug files.
- If no debug files exist, Reveal Inserter will use a default configuration.

The sections of Reveal Inserter view display the Dataset, Design Tree, and Trigger Output, plus Trace Signal Setup and Trigger Signal setup views.

Figure 102: Reveal Inserter Trace Signal Setup

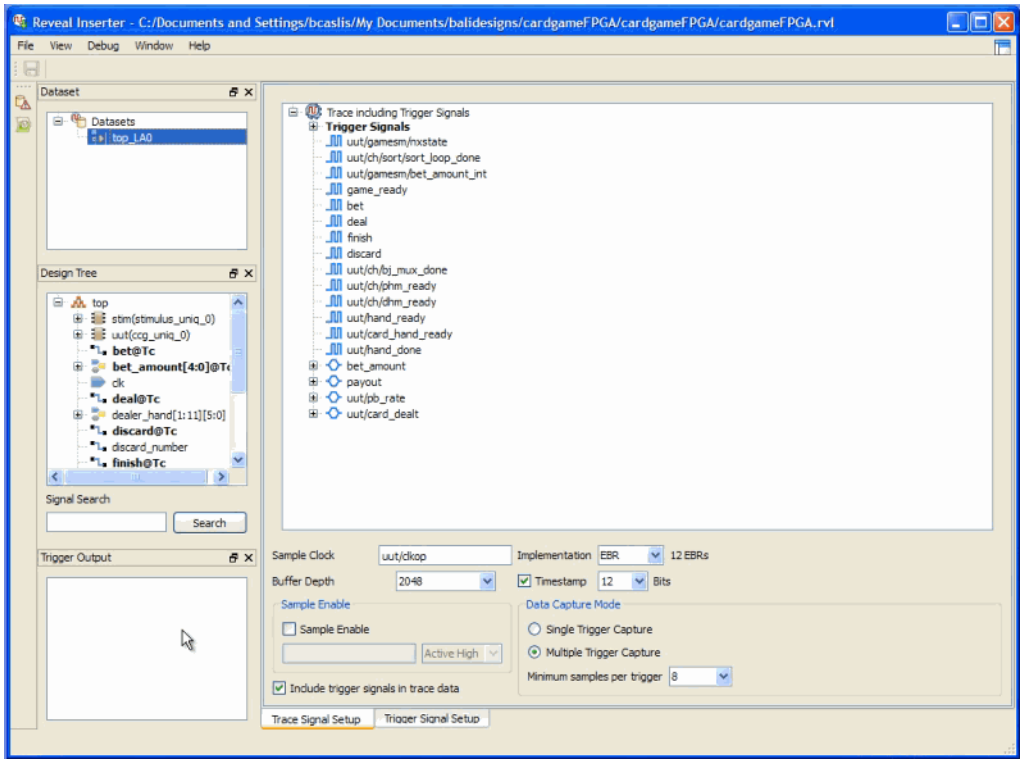
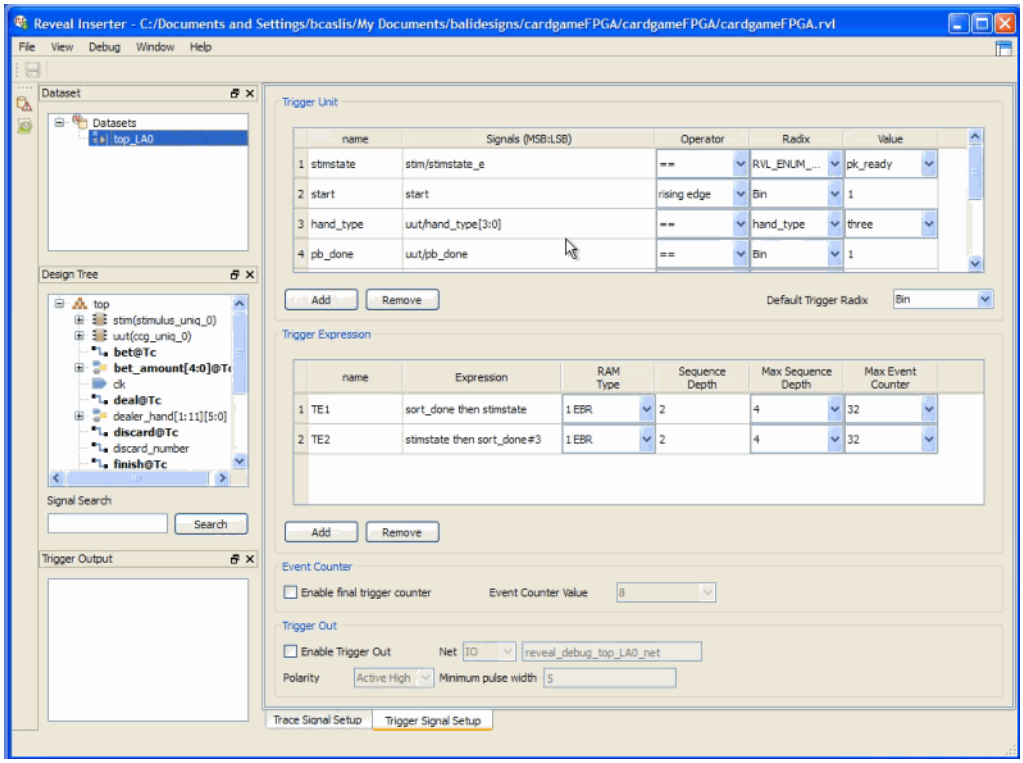


Figure 103: Reveal Inserter Trigger Signal Setup



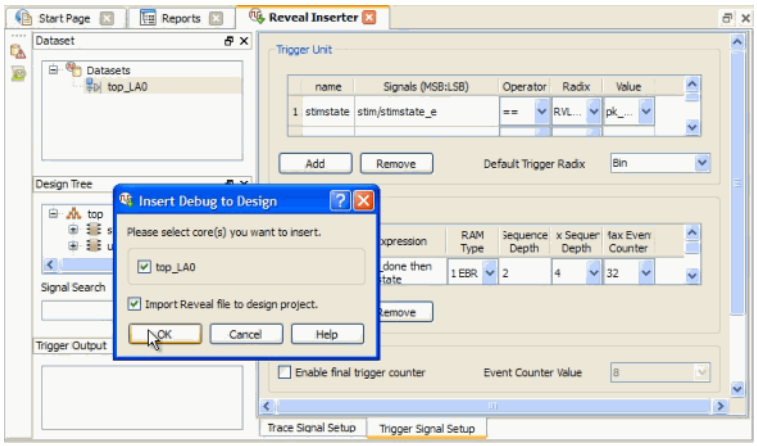
Setting up and Inserting Debug

A Debug menu becomes available after Reveal Inserter is launched. It includes controls for managing cores, managing trace signals, running DRC, and inserting debug.

See the Reveal User Guide for more information on setting up debug information with Reveal Inserter.

After you have your debug set up, choose **Debug > Insert Debug** or click the Insert Debug button on the vertical toolbar to insert debug into your design. This will set the current debug file as active.

Figure 104: Insert Debug



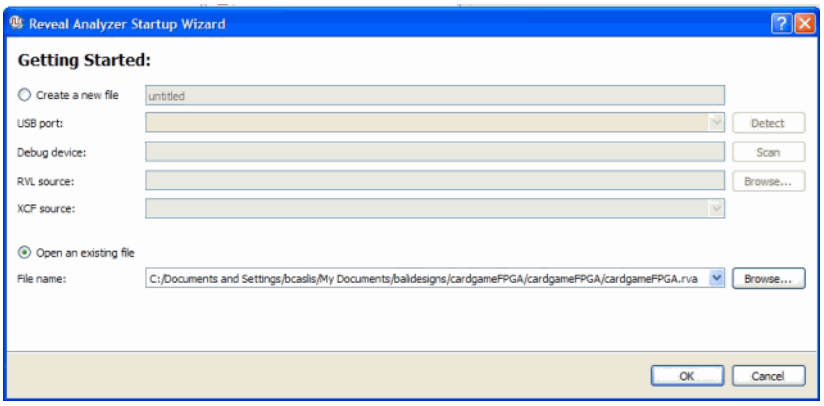
When the design is fully implemented and programmed, you can run Reveal Analyzer to debug your design.

Reveal Analyzer

After you generate the bitstream or JEDEC file, you can use Reveal Analyzer to debug your FPGA circuitry. Reveal Analyzer gives you access to internal nodes inside the device so that you can observe their behavior. It enables you to set and change various values and combinations of trigger signals. After the specified trigger condition is reached, the data values of the trace signals are saved in the trace buffer. After the data is captured, it is transferred from the FPGA through the JTAG ports to the PC.

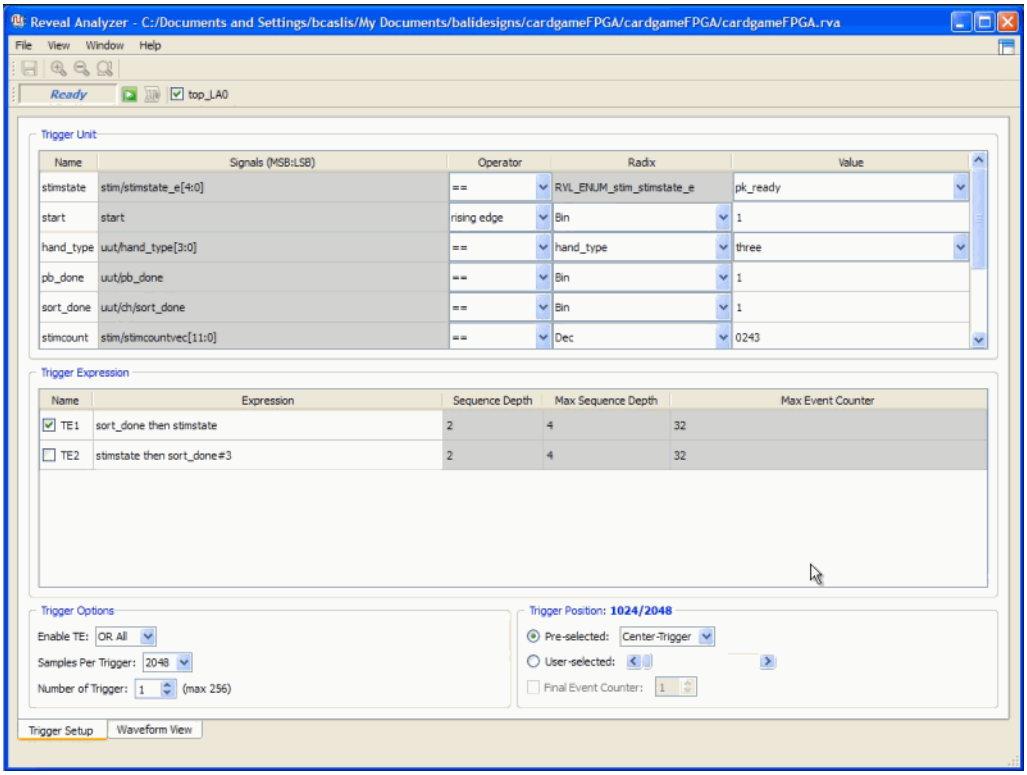
To launch Reveal Analyzer, choose **Tools > Reveal Analyzer** or click the Reveal Analyzer button on the toolbar. The Reveal Analyzer Startup Wizard allows you to use an existing Reveal Analyzer file or create a new one.

Figure 105: Reveal Analyzer Setup Wizard



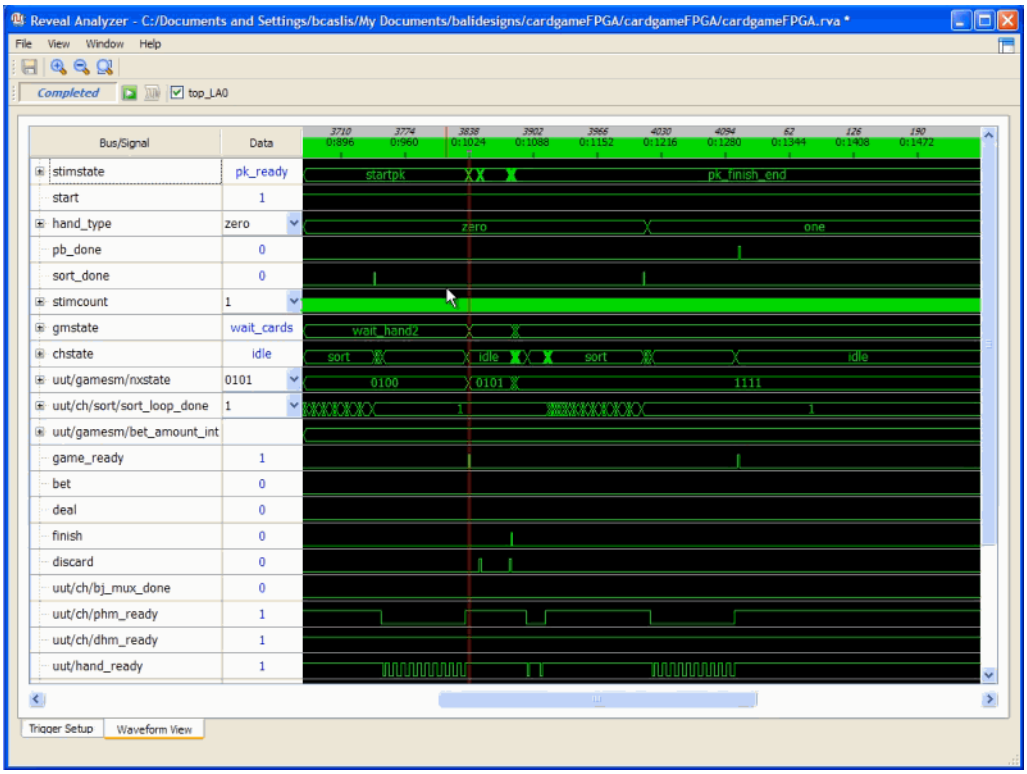
The Reveal Analyzer view consists of a Trigger Setup view and a Waveform view. In the Trigger Setup view are areas displaying the Trigger Unit, Trigger Expression, Trigger Options and Trigger Position. Also in this view are controls to select which core to use to enable for triggering the analyzer.

Figure 106: Reveal Analyzer



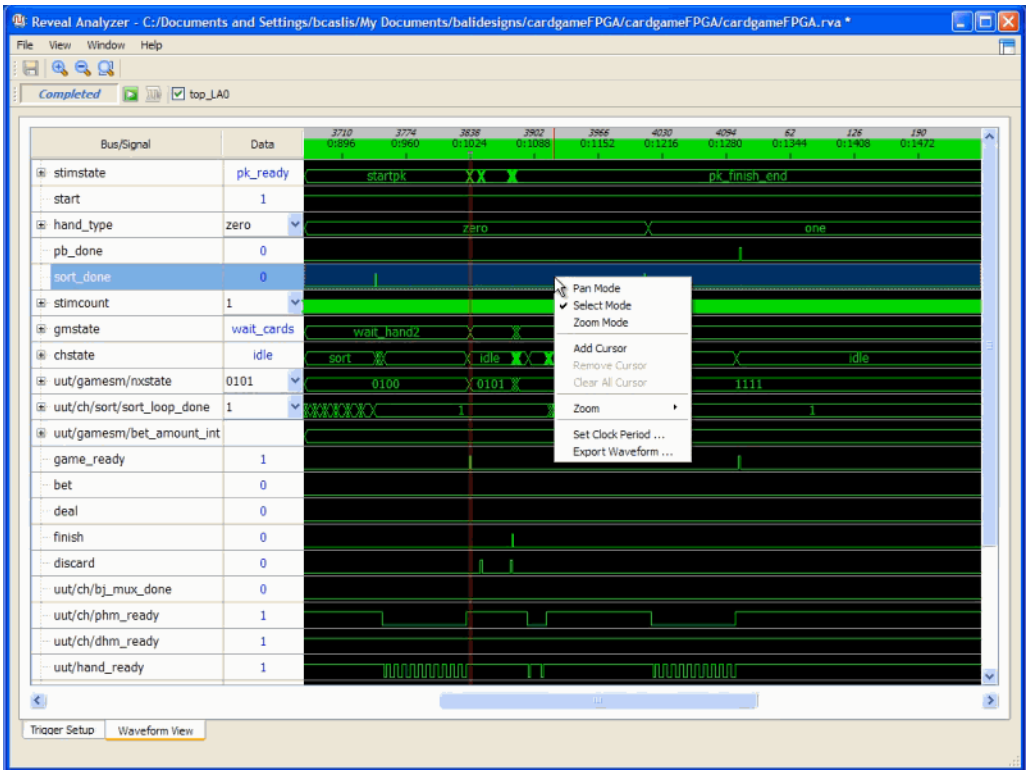
When you choose **Run**, the analyzer connects to the hardware, configures the debug logic, and waits for the trigger conditions. Once triggered, the data is uploaded to the analyzer. The Run command also switches the display to the Waveform view.

Figure 107: Reveal Analyzer Waveform View



The Waveform view has controls for running, zooming and window controls in the menu and toolbar areas. Right-clicking in the waveform area brings up a pop-up menu with selections for changing the cursor mode for panning, zooming or selecting plus selections for managing cursors, changing the clock period and exporting waveforms.

Figure 108: Reveal Analyzer Waveform Cursor Controls



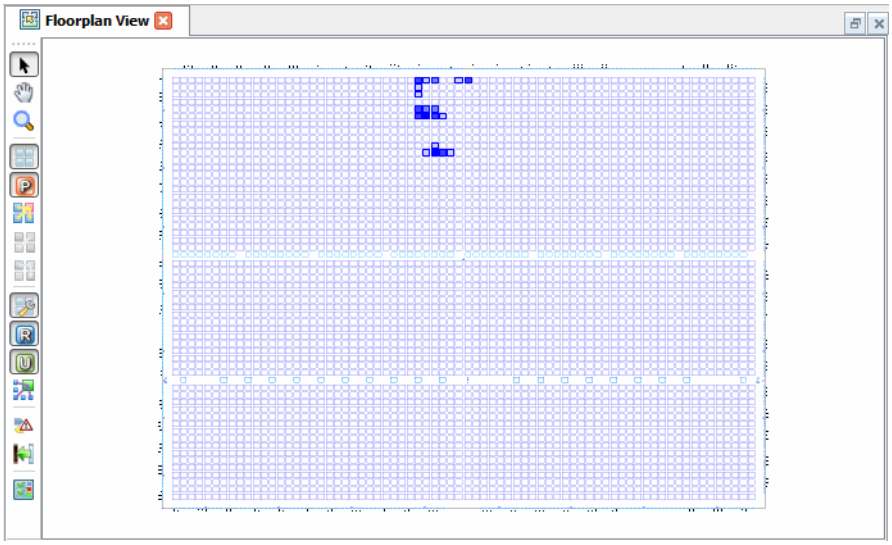
The Data column in the view shows the data for the active cursor. The Reveal Analyzer supports multiple cursors which can be added, removed and position changed within the waveform. Selecting the cursor and dragging it will produce a rubber band effect which can be used for measuring time intervals.

See the Reveal User Guide for more information on using Reveal Analyzer.

Floorplan View

Floorplan View provides a large-component layout of your design. It displays user constraints from the logical preference file (.lpf) and placement and routing information. All connections are displayed as fly-lines.

Figure 109: Floorplan View



Floorplan View allows you to create REGIONs and bounding boxes for UGROUPs and specify the types of components and connections to be displayed. As you move your mouse pointer over the floorplan layout, details are displayed in tooltips and in the status bar:

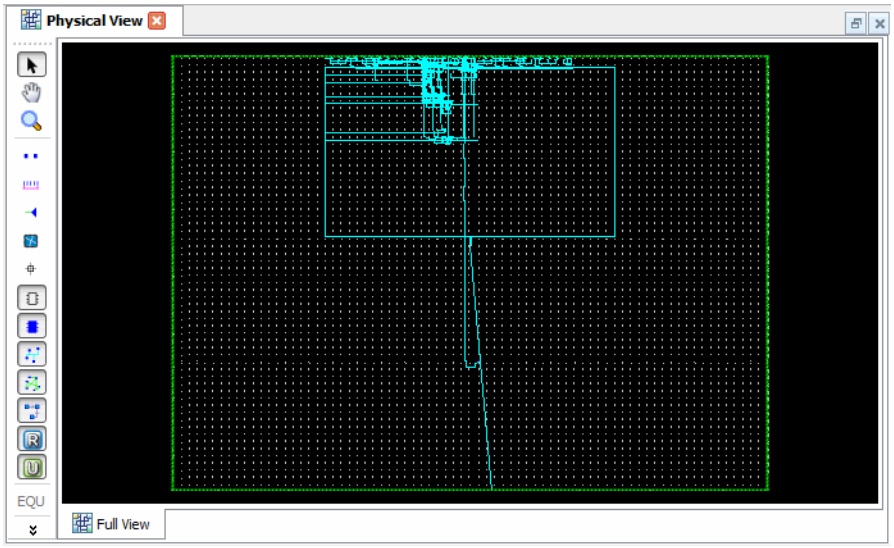
- the number of resources for each UGROUP and REGION
- the number of utilized slices for each PLC component
- the name and location of each component, port, net, and site

Floorplan View is available as soon as the target device has been specified.

Physical View

Physical View provides a read-only detailed layout of your design that includes switch boxes and physical wire connections. Routed connections are displayed as Manhattan-style lines, and unrouted connections are displayed as fly-lines.

Figure 110: Physical View



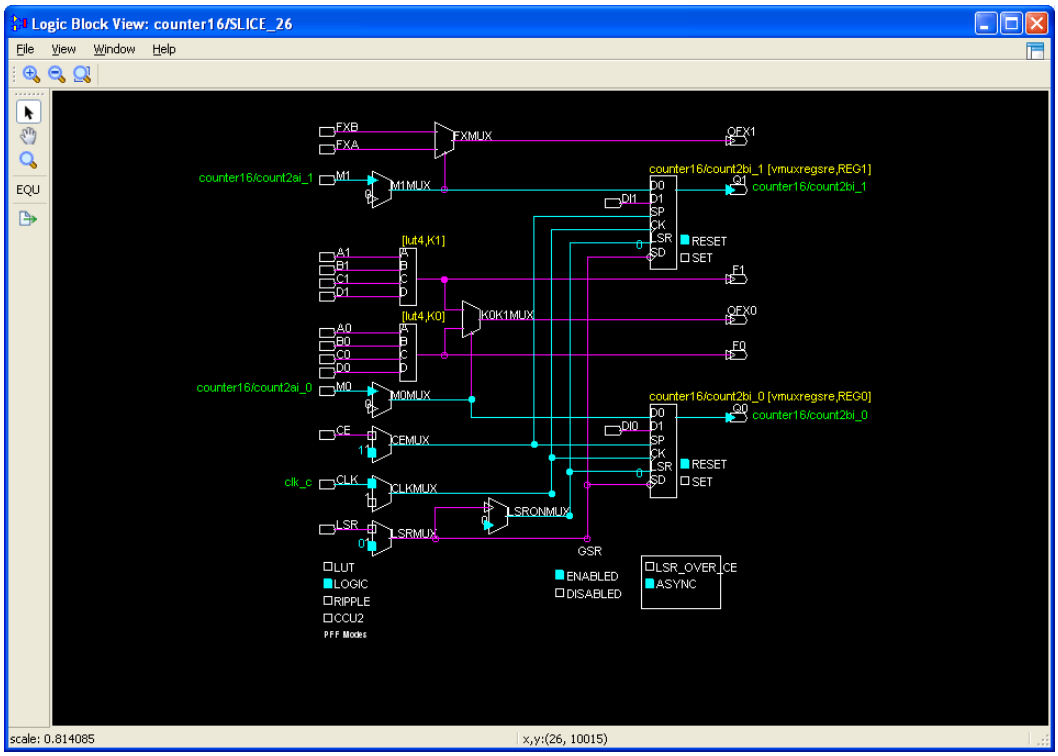
As you move your mouse pointer slowly over the layout, the name and location of each REGION, group, component, port, net, and site are displayed as tooltips and also appear in the status bar. The tooltips and status bar also display the group name for components that are members of a group.

The Physical View toolbar allows you to select the types of elements that will be displayed on the layout, including components, empty sites, switch boxes, switches, pin wires, routes, and timing paths.

Logic Block View

Logic Block View enables you to examine logic details of one or more placed and routed components. It provides either a schematic or tabular view, depending on the type of component selected.

Figure 111: Logic Block Schematic View



Schematic views of PIO and PFU/PFF components can be accessed from NCD View, Floorplan View, and Physical View. Tabular views of PLL, EBR, and DSP blocks can be accessed from Floorplan View and Physical View. Right-click one or more selected components and choose **Logic Block View**.

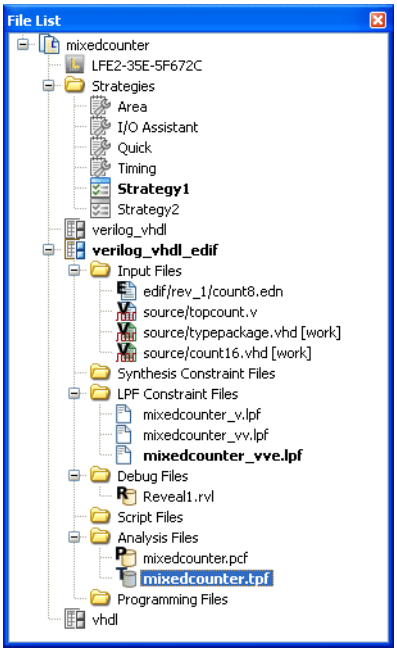
Multiple Logic Block Views can be opened at one time, up to the maximum that has been set in the Tool > Options dialog box for Physical View.

Timing Analysis View

Timing Analysis View provides a graphical way to navigate timing information and a fast timing analysis loop that allows dynamic path changes without having to re-implement or remap your design.

The File List view of your project contains the Timing Analysis preference files (.tpf) in the Analysis folder. One or none of the .tpf files can be active.

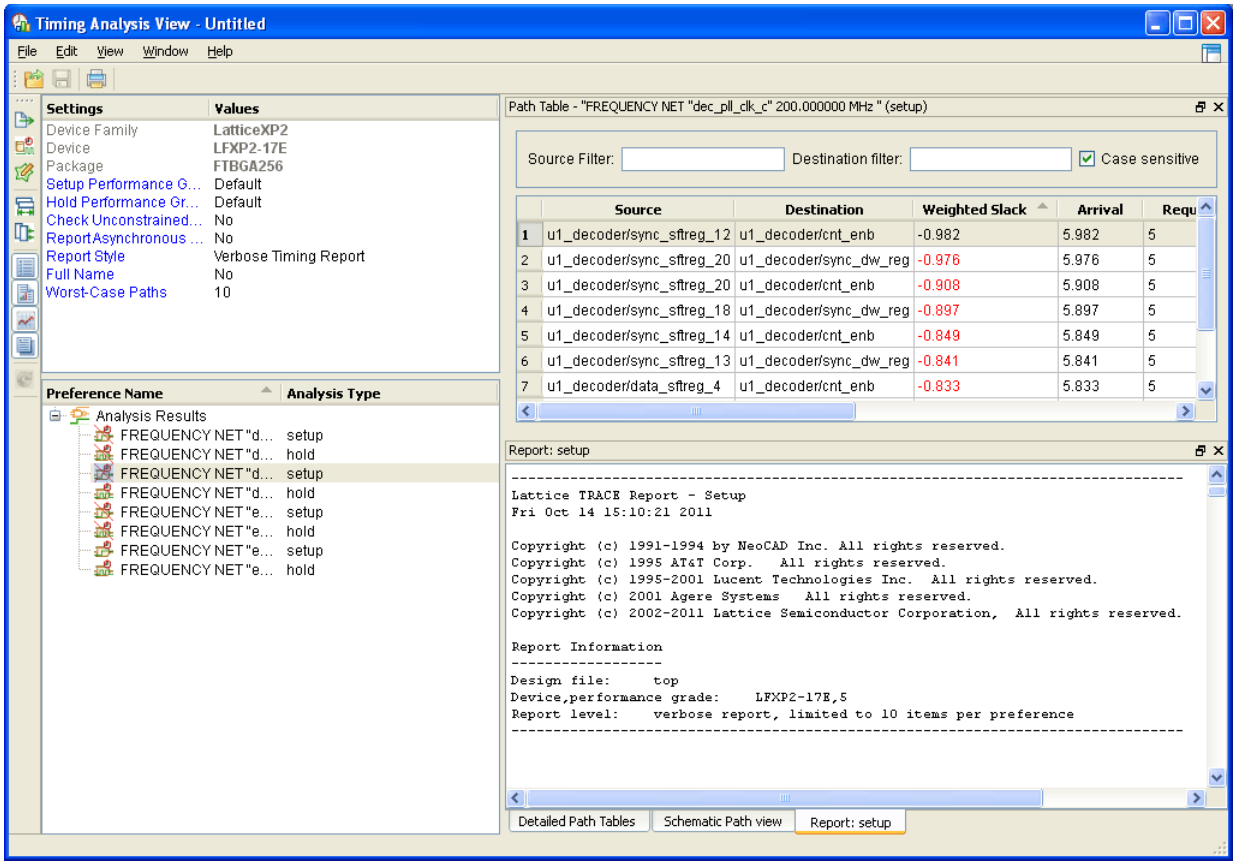
Figure 112: Timing Analysis Preference File



Launching Timing Analysis View

When Timing Analysis View is launched, it uses the active .tpf file. If there is no active .tpf file, it will read timing preferences from the logical preference file (.lpf) for its initial timing calculations. You can also double-click any active or inactive .tpf file in the File List to launch Timing Analysis View using the settings from the selected .tpf file.

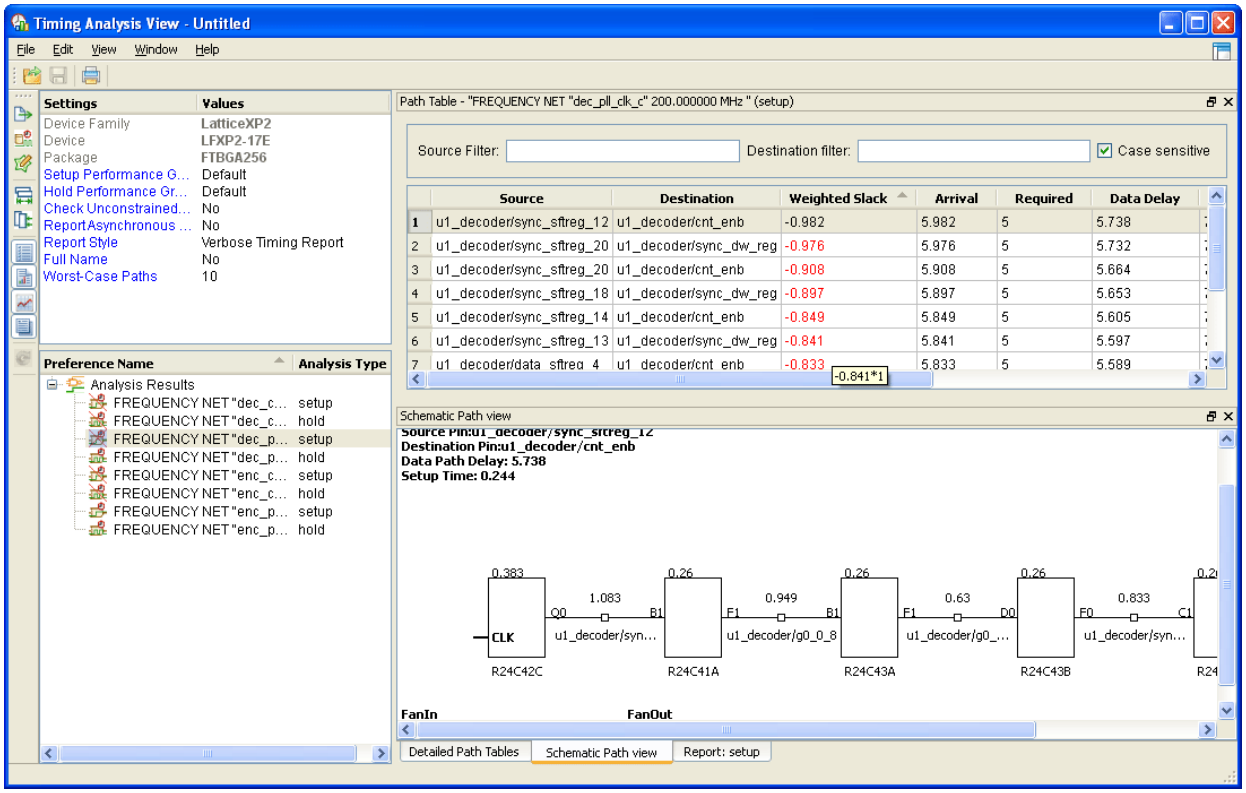
Figure 113: Timing Analysis View



The sections in Timing Analysis View display the trace settings from your active strategy and the preferences from the active .tpf or .lpf file. Timing

Analysis View also provides views of the path table, detailed path tables, schematic path and report.

Figure 114: Timing Analysis Schematic Path View



The Timing Analysis vertical toolbar on the left contains the following controls:

- Export – exports the timing paths to a cvs file.
- Settings – allows you to change settings for a timing analysis run. To change the settings permanently, you must edit the trace settings of the active strategy.
- Change Timing Preferences – displays the Timing Preferences tab of the Spreadsheet view. This is a graphical view of the .tpf file.
- Fit All Columns – sizes each column of the Detailed Path Table to fit the information displayed.
- View All Columns – sizes all columns of the Detailed Path Table to fit inside the window.
- Path Table – displays or hides the path table.
- Report: setup – displays or hides the Report:Setup tab.
- Schematic Path view – displays or hides the Schematic Path View tab.
- Detailed Path Tables – displays or hides the Detailed Path Tables tab.
- Update – when this button is visible and rotating, indicates a preference has been changed. When clicked, it recalculates timing.

Updating and Saving Timing Preference File Settings

When you change a timing preference, you must run **Update** to recalculate the timing. Use the following steps to change timing preferences, recalculate timing and save your preferences:

1. In Timing Analysis View, click the Change Timing Preferences button on the vertical toolbar. This opens Spreadsheet TPF View.
2. Modify a preference in Spreadsheet TPF View. An asterisk indicating a data change appears in both the Spreadsheet and Timing Analysis tabs.
3. Select Timing Analysis View again.

   The Update button in the vertical toolbar is now visible and rotates, indicating that a timing preference has changed and that the timing has not yet been recalculated.
4. Click the rotating Update button to recalculate the timing.
5. To save the .tpf file, make sure that Timing Analysis View is active, and then choose **File > Save**.

   Timing Analysis view must be active in order to use the File > Save command. You cannot save timing preferences from Spreadsheet TPF View.

Exporting TPF Settings

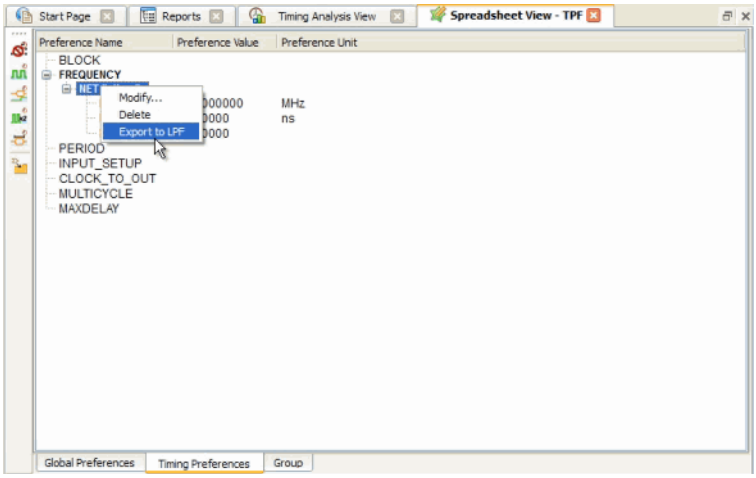
To save your TPF settings to the logical preference file, first select the Timing Preferences tab of Spreadsheet TPF View. Right-click the preference you want to export and choose **Export to LPF**. This will export the selected setting to the active .lpf file.

Importing In-Memory Timing Preferences

If you edit timing preferences in the regular Spreadsheet View after you have launched Timing Analysi View, you must import these in-memory timing preferences in order to run timing analysis on them.

1. In Timing Analysis View, click the Change Timing Preferences button on the vertical toolbar.
2. In the TPF Spreadsheet View, choose **File > Import > Copy LPF to TPF**.
3. Return to the Timing Analysis View main window and click the Update button.

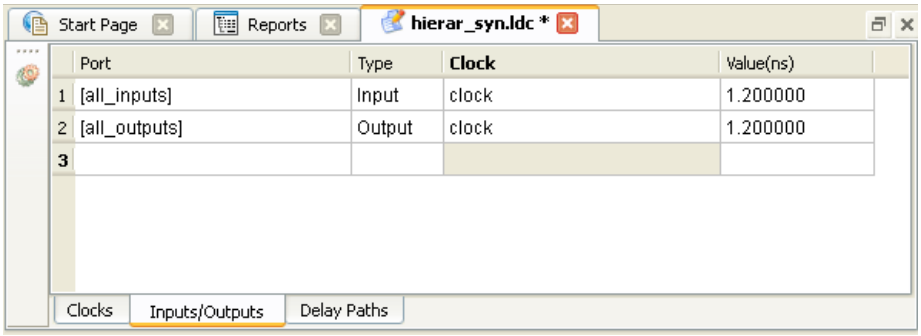
Figure 115: Export TPF



LDC Editor

The Lattice Design Constraints (LDC) Editor is a synthesis constraint tool for use with the Lattice Synthesis Engine. Currently, the Lattice Synthesis Engine and LDC Editor support the MachXO, MachXO2, and Platform Manager device families. The LDC Editor uses a spreadsheet style user interface that enables you to quickly create and edit Synopsys Design Constraints and save them to a Lattice Design Constraint file (.ldc). You can create several .ldc files and select one of them to serve as the active synthesis constraint file for the current implementation.

Figure 116: LDC Editor



After you have selected the Lattice Synthesis Engine (LSE) as the synthesis tool, the LDC Editor will open automatically each time you create or open an .ldc file. You also have the option of viewing and editing .ldc files in the Source Editor. The LDC Editor includes individual tabs for Clocks, Inputs/Outputs, and Delay Paths. Each sheet enables you to define synthesis constraints by double-clicking a cell and selecting or typing a value.

Clocks The Clocks tab allows you to define an alias to be associated with an existing clock port or net from the source fiile. Double-click the Source cell to select an existing clock port or net, and then enter an alias for the clock in the Clock Name cell. Enter a clock period in nanoseconds.

Inputs/Outputs The Inputs/Outputs tab enables you to specify an input or output delay relative to a clock. Double-click the Type cell to select the type of delay (input or output), and then select from the input or output ports in the Port cell. In the Clock cell, select an existing clock or an alias name that has been defined for a clock. Enter a delay value in nanoseconds.

Delay Paths The Delay Paths tab allows you to define a Multicycle path, specify a Max_Delay for a timing path, and identify a False path that is to be excluded from timing analysis. Double-click the Delay Type cell to select the type of delay, and then specify the path information, delay, and cycles as appropriate.

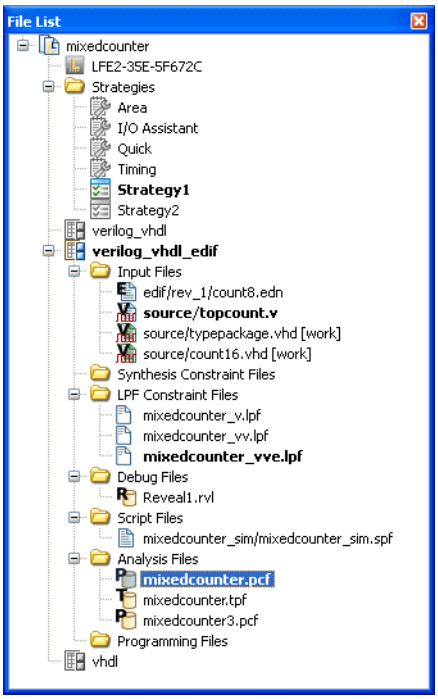
For detailed information about setting SDC constraints, see *Applying Lattice Synthesis Engine Constraints* and the *Constraints Reference Guide* in the Lattice Diamond online Help.

Power Calculator

Power Calculator estimates the power dissipation for your design. It uses parameters such as voltage, temperature, process variations, air flow, heat sink, resource utilization, activity and frequency to calculate the device power consumption. It reports both static and dynamic power consumption.

Power Calculator files (.pcf) are managed in the Analysis Files folder of the File List.

Figure 117: Power Calculator File

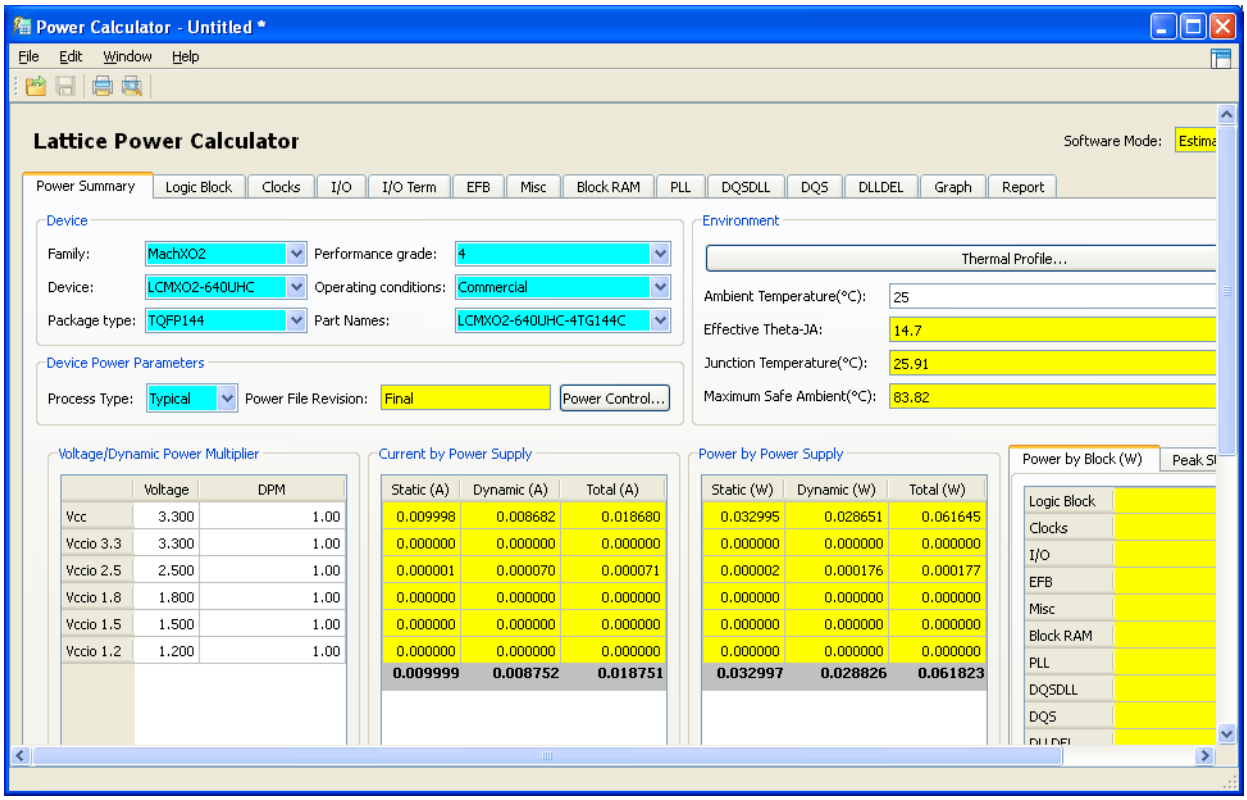


To launch Power Calculator, choose **Tools > Power Calculator** or click the Power Calculator button on the toolbar. When Power Calculator is launched, the .pcf file it uses will depend on the following conditions:

- If an active .pcf file exists, it will be used. An inactive .pcf file will be used if it is the only one available.
- If an active or inactive .pcf file in the File List Analysis Files folder is double-clicked, it will be used.
- If no .pcf file exists, Power Calculator will perform power calculations based on the current open design.

When Power Calculator opens, it displays the Power Summary page, which enables you to change the targeted device, operating conditions, voltage, and other basic parameters. Updated estimates of power consumption are then displayed based on these changes. Tabs for other pages, including Logic Block, I/O, I/O Term, Block RAM, Graph, and Report, are arranged across the top. The number and types of these pages will depend on the target device.

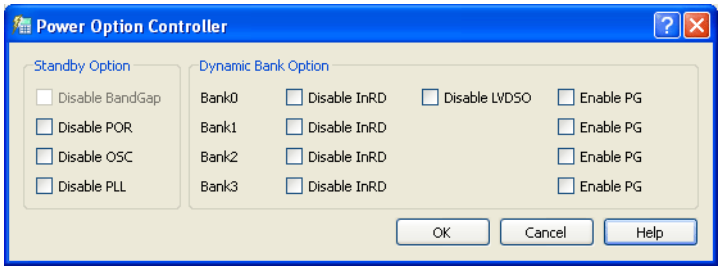
Figure 118: Power Summary



Power Calculator can also be opened as a stand-alone tool, allowing you to create or open an existing Power Calculator project file without using a Diamond project. Choose **Start > Programs > Lattice Diamond > Accessories > Power Calculator**.

Added Features for Low-Power Architecture Three power modules are available for MachXO2 and LatticeECP4 devices: Power Controller, Dynamic Bank Controller, and Power Guard. You can generate these modules from IPexpress. Refer to the *IPexpress Module Reference Guide* for more information. Power Calculator provides tools for taking advantage of these features, including the Power Option Controller, which is accessible on the Power Summary page. It enables you to conserve power by turning on and off blocks in the chip that consume power, manage the entrances and exits of signals, and use power guard (PG) to stop signals from entering the chip.

Figure 119: Power Option Controller for MachXO2



First, set the standby, shut-off, and power guard options that you want to allow, using the appropriate Power Calculator pages. Afterwards, return to the Power Summary page, click the **Power Control** button to open the Power Option Controller, and select the elements to be turned off during low-power operation.

Power Option Controller For MachXO2 devices, the Power Option Controller includes the following options for turning off elements that have been placed in standby mode:

- Disable Bandgap – turns off the Bandgap. When this options is selected, analog circuitry such as the PLLs, on-chip oscillator, and referenced and differential I/O buffers are also turned off.
- Disable POR – turns off the power-on-reset circuit, which monitors VCC levels. When the POR circuitry is turned off, limited power detection circuitry is still active. This option is only recommended for applications in which the power supply rails are reliable.
- Disable OSC – turns off the on-chip oscillator.
- Disable PLL – turns off the selected phase-locked loop.

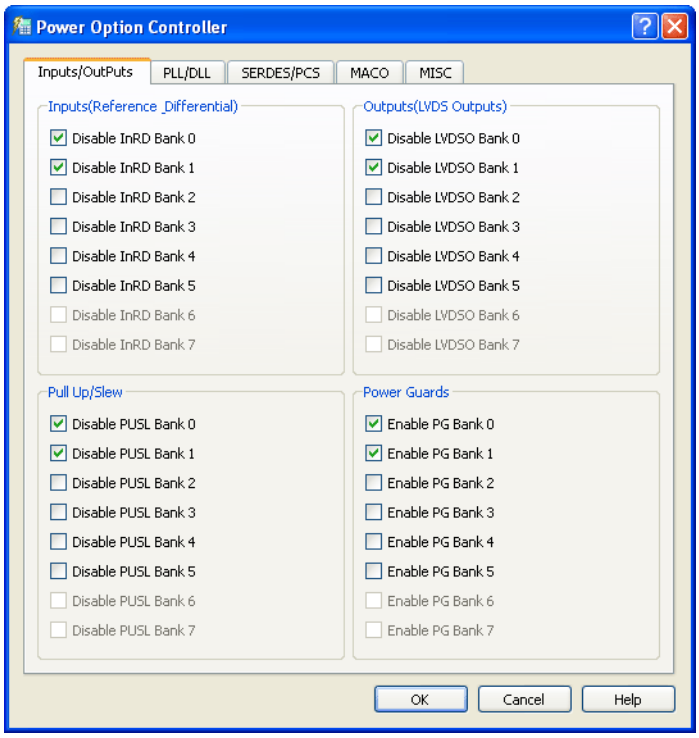
The Power Option Controller includes the following dynamic bank options for MachXO2 devices:

- Disable InRd – turns off the referenced and differential input buffers for a selected bank.
- Disable LVDSO – turns off the LVDS output buffer for a selected bank.
- Enable PG – enables Power Guard for the selected bank.

For LatticeECP4 devices, the Power Option Controller comprises separate tabs for Inputs/Outputs, PLL/DLL, SERDES/PCS, MACO, and MISC.

- I/O Tab
  - Disable InRd – turns off the referenced and differential input buffers for a selected bank.
  - Disable LVDSO – turns off the LVDS output buffer for the selected bank.
  - Disable PUSL – turns off the pull-up/slew control for the selected bank.
  - Enable PG – enables Power Guard for the selected bank.
- PLL/DLL Tab
  - Disable PLL – turns off the selected phase-locked loop.
  - Disable DDRDLL – turns off the selected double-data-rate delay-locked loop.
- SERDES/PCS Tab
  - Disable Channel – turns off the selected Receive, Transmit, or Los of Signal (LOS) for the SERDES PCS channel.
  - Disable HSPLL – turns off the high-performance LC-based PLL for the SERDES quad.

Figure 120: Power Option Controller for LatticeECP4



- Disable LSPLL – turns off the ring-based lower-performance PLL for the SERDES quad.

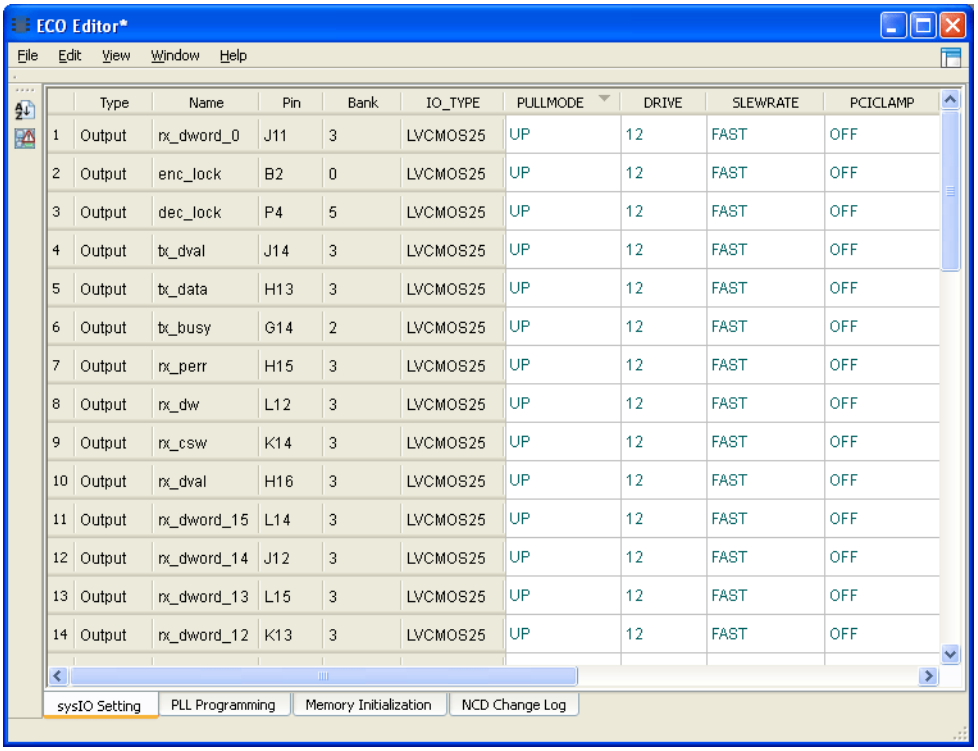
- MACO tab
  - Disable <MACO> Core – turns off the selected MACO core.
- MISC tab
  - Disable OSC – turns off the on-chip oscillator.

For more information on Power Calculator see *Analyzing Power Consumption* in the Lattice Diamond product documentation.

ECO Editor

The Engineering Change Order (ECO) Editor enables you to safely make changes to an implemented design without having to rerun the entire process flow. Choose **Tools > ECO Editor** or click the ECO Editor button on the toolbar.

Figure 121: ECO Editor



	Type	Name	Pin	Bank	IO_TYPE	PULLMODE	DRIVE	SLEWRATE	PCICLAMP
1	Output	rx_dword_0	J11	3	LVCMOS25	UP	12	FAST	OFF
2	Output	enc_lock	B2	0	LVCMOS25	UP	12	FAST	OFF
3	Output	dec_lock	P4	5	LVCMOS25	UP	12	FAST	OFF
4	Output	tx_dval	J14	3	LVCMOS25	UP	12	FAST	OFF
5	Output	tx_data	H13	3	LVCMOS25	UP	12	FAST	OFF
6	Output	tx_busy	G14	2	LVCMOS25	UP	12	FAST	OFF
7	Output	rx_perr	H15	3	LVCMOS25	UP	12	FAST	OFF
8	Output	rx_dw	L12	3	LVCMOS25	UP	12	FAST	OFF
9	Output	rx_csw	K14	3	LVCMOS25	UP	12	FAST	OFF
10	Output	rx_dval	H16	3	LVCMOS25	UP	12	FAST	OFF
11	Output	rx_dword_15	L14	3	LVCMOS25	UP	12	FAST	OFF
12	Output	rx_dword_14	J12	3	LVCMOS25	UP	12	FAST	OFF
13	Output	rx_dword_13	L15	3	LVCMOS25	UP	12	FAST	OFF
14	Output	rx_dword_12	K13	3	LVCMOS25	UP	12	FAST	OFF

ECOs are requests for small changes to be made to your design after it has been placed and routed. The changes are directly written into the native circuit description database file (.ncd) without requiring that you go through the entire design implementation process.

ECOs are usually intended to correct errors found in the hardware model during debugging. They are also used to facilitate changes that had to be made to the design specification because of problems encountered when other FPGAs or components of the PC board design were integrated.

The ECO Editor includes windows for editing I/O settings, PLL settings, and memory initialization values. It also provides a Change Log window for you to track changes between the modified .ncd file and the post-PAR .ncd file.

Note

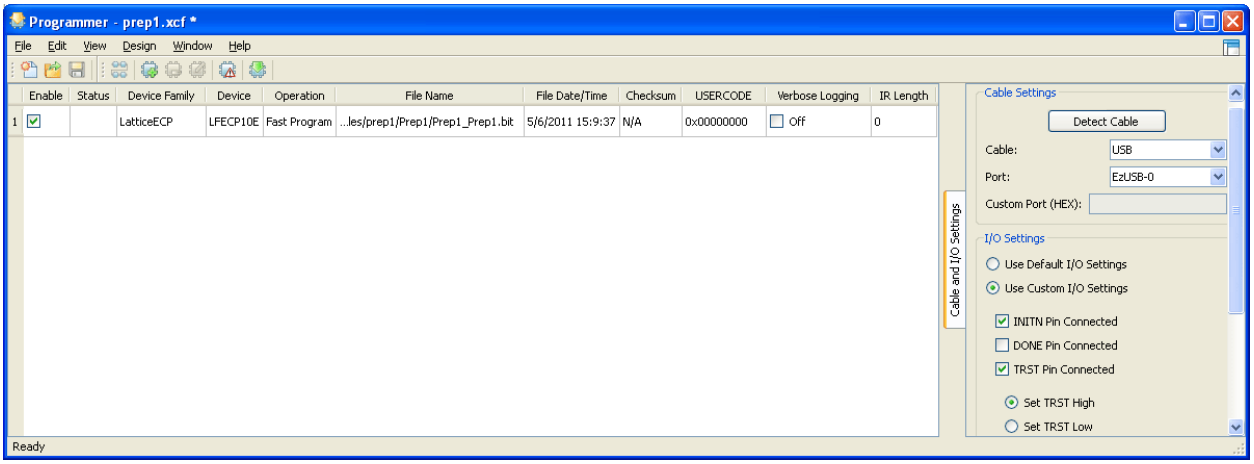
After you edit your post-PAR, routed .ncd file, your functional simulation and timing simulation will no longer match.

For more information see *Applying Engineering Change Orders* in the Lattice Diamond online documentation.

Programmer

After you have placed and routed the design and generated the JEDEC or bitstream file, you can use Diamond's integrated Programmer to program the target device. Choose **Tools > Programmer**. Programmer detects the cable type, scans the device chain, creates the XCF file, and downloads the data file to the device.

Figure 122: Programmer



Programmer supports serial, concurrent (turbo), and microprocessor programming of Lattice devices in PC and Linux environments. Device chains can be scanned automatically using the Programmer graphical user interface.

Programmer is integrated into the Diamond software environment, and is also available in a standalone version.

Features include:

- Scan chain and display chain contents (.xcf file)
- Download data files to devices
- Create/modify/display .xcf file
- Generate files based on the .xcf file (Tcl/command line only)

Programmer uses a Single Document Interface (SDI) where a single .xcf project is displayed per Programmer instance. Opening additional .xcf files in Diamond-integrated mode will close the current .xcf and open the specified .xcf.

The main view displays the devices in the current Programmer project resulting from the Scan action, or from manual creation in a table.

Double-clicking on an uneditable cell or right-clicking and selecting Device Properties opens a dialog that displays more information about the selected

device. Additionally, some entries may be edited directly on the table by clicking.

Columns can be displayed or hidden by choosing View > Columns. The default columns that are displayed are Process, Status, Device Family, Device, Operation, File Name, File Date/Time, Checksum, USERCODE, and Verbose Logging.

Additional columns are available, but are hidden by default. The columns are Device Vendor, Device Full Name, and Device Description.

The following columns are directly editable from the table:

- Enable
- Device Family
- Device
- File Name
- Verbose Logging

Some of these columns become un-editable (grayed out) if the selected operation or other option does not support it. For example, File Name will be grayed out for a 'Bypass' operation.

Each row has a column enabling the device. Devices where the enable option is not selected will not be programmed. They will effectively be treated as a bypass operation. This allows one .xcf file to be used whether programming all devices in a scan chain or just a single device.

Each row has a column for the device status. The status indicates whether the operation performed was successful or not. This field is also used for read back operations to display what is read back if the data to display is short. For larger data sets that are read back, a dialog box is displayed.

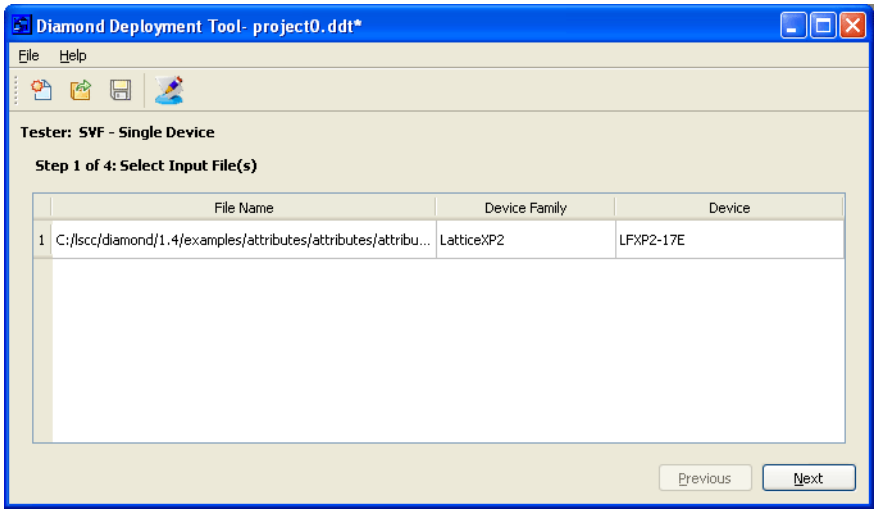
Programmer can also be used as a stand-alone tool. From the Windows Start menu, choose **Programs > Lattice Diamond > Accessories > Diamond Programmer**.

For more information about programming a device with Programmer, see the “Programming the FPGA” section of the Lattice Diamond online Help.

Deployment Tool

Deployment Tool is a stand-alone tool available from the Diamond Accessories. The Deployment Tool enables you to convert data files to other formats and use the data files to generate other data file formats. A four-step wizard helps you create a new deployment and select the deployment type, input file type, and output file type.

Figure 123: Deployment Tool

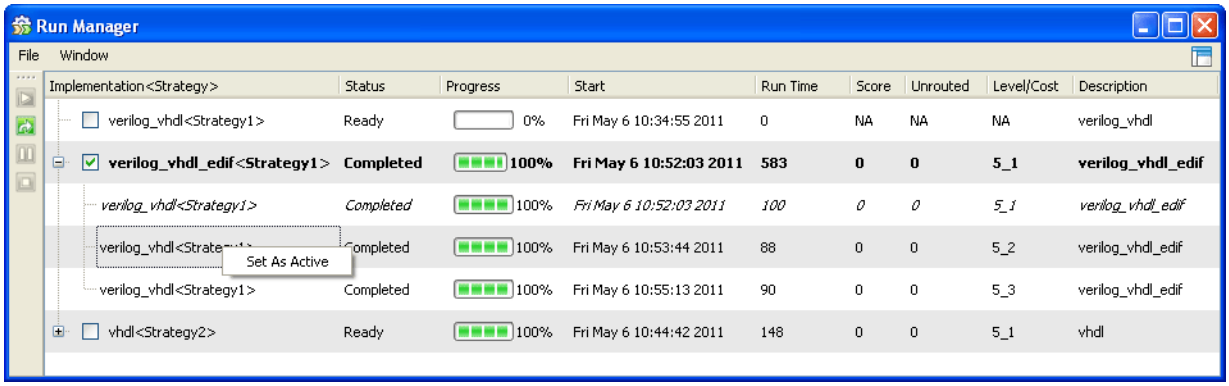


For more information about using the Deployment Tool, see the “Deploying the Design with the Deployment Tool” in the Lattice Diamond online Help.

Run Manager

Run Manager runs the processes for the different implementation/strategy combinations. Choose **Tools > Run Manager** or click the Run Manager button on the toolbar.

Figure 124: Run Manager



Run Manager takes the design through the entire process flow for each selected implementation. If you are running on a multicore system, Run Manager will distribute the iterations so that they are executed in parallel. The

options “Maximum number of implementation processes in run manager” and “Maximum number of multi-par processes in run manager” are available in the Environment > General section of the Tool Options dialog box. Choose **Tools > Options** to access it. These options enable you to set the maximum number of processes to run in parallel. Generally, the maximum number of processes should be the same as the number of cores in your processor; but if the strategy is using the “Multi-Tasking Node List” option for Place & Route Design, this number should be set to one.

You can use the Run Manager list to set an implementation as active. Right-click the implementation/strategy pair and choose **Set as Active**.

For an implementation that uses multiple iterations of place-and-route, you can select the iteration that you want to use as the active netlist for further processes. Expand the implementation list, right-click the desired iteration, and choose **Set as Active**. The active iteration is displayed in italics.

To examine the reports from each process, set an implementation as active, and then select the Reports View.

See the “Managing Projects” section of the Lattice Diamond online Help for more information about using implementations, strategies, and Run Manager.

Synplify Pro for Lattice

Synplify Pro for Lattice is an OEM synthesis tool used in the Lattice Diamond design flow. Synplify Pro runs in batch mode when you run the Synthesize Design step in Process View. To examine the output report, select **Synplify Pro** in the Process Reports folder of Reports View.

You can also run Synplify Pro in interactive mode. Choose **Tools > Synplify Pro for Lattice** or click the Synplify Pro button  on the toolbar.

For more information, see the *Synplify Pro User Guide*, which is available from the Lattice Diamond Start Page or the Synplify Pro Help menu.

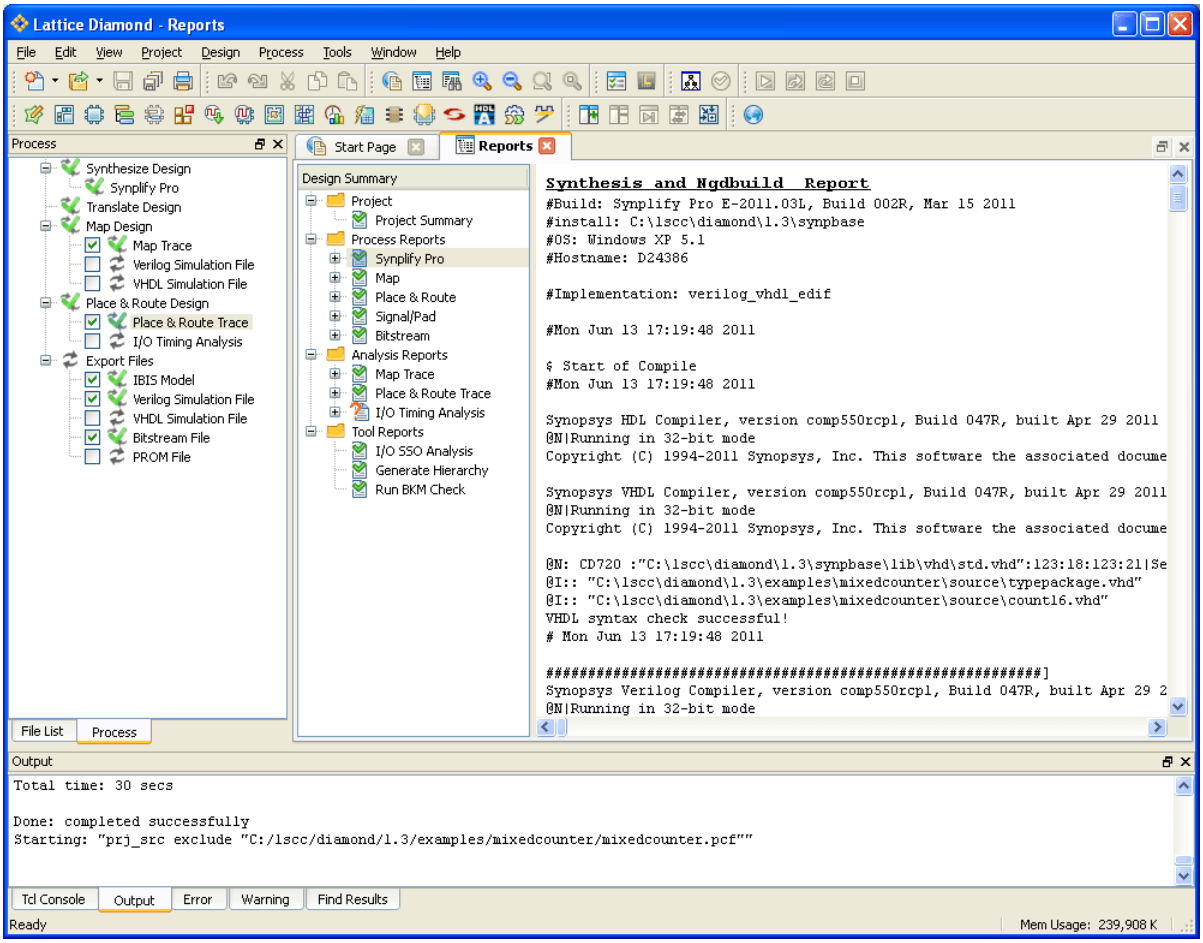
Active-HDL Lattice Edition

The Active-HDL Lattice Edition tool is an OEM simulation tool that is closely linked to the Lattice Diamond environment. It is not run as part of the Process implementation flow. To run Active-HDL, choose **Tools > Active-HDL Lattice Edition** or click the Active-HDL button  on the toolbar.

See “Simulation Flow” on page 70 for more information about simulating your design. See “Simulation Wizard” on page 118 for information about creating a simulation project to run in Active-HDL.

For complete information about Active-HDL, see the *Active-HDL Online Documentation*, which is available from the Lattice Diamond Start Page or the Active-HDL Help menu.

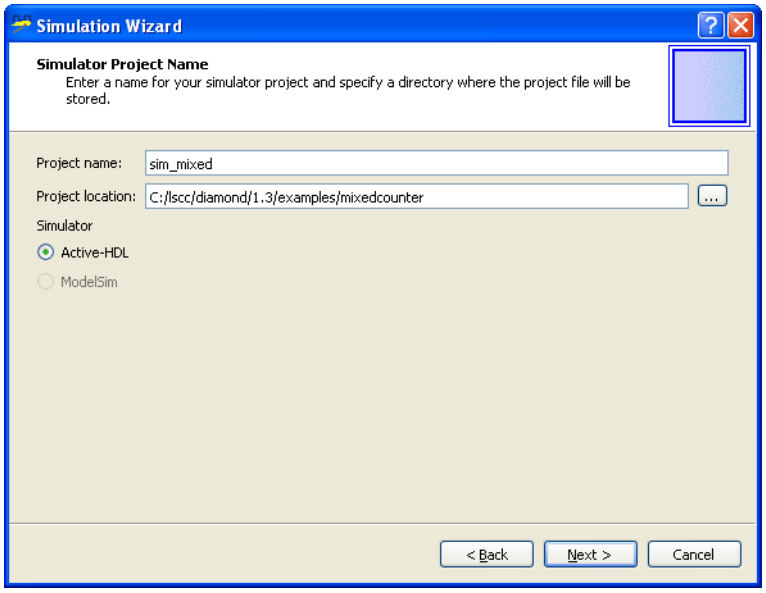
Figure 125: Synthesis Report



Simulation Wizard

The Simulation Wizard enables you to create a simulation project for your design. To open Simulation Wizard, choose **Tools > Simulation Wizard** or click the Simulation Wizard button on the toolbar. The wizard leads you through a series of steps that include selecting a simulation project name and location, specifying the simulator to use (if you have more than one installed), selecting the process stage to use (from RTL to Post-Route Gate-Level + Timing), specifying the language (VHDL or Verilog), and selecting the source files. You can optionally run the simulation directly from the wizard.

Figure 126: Simulation Wizard



Common Tasks

Lattice Diamond gathers the many FPGA implementation tools into one central design environment. This gives you common controls for active tools, and it provides shared data between views.

Controlling Tool Views

Tool views are highly configurable in the Lattice Diamond environment. You detach a tool view to work with it as a separate window, and you can create tab groups to display two views side-by-side.

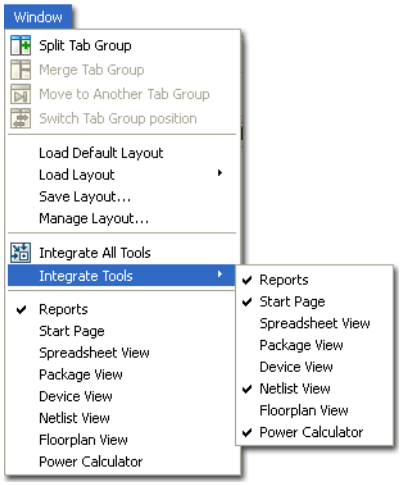
Detaching and Attaching a Tool View

Each Diamond integrated tool view contains a Detach button in the upper-right corner that allows you to work with the tool view as a separate window.

After a tool view is detached, the Detach button changes to an Attach button, which reintegrates the view into the Lattice Diamond main window.

You can detach as many tool views as desired. The Window menu keeps track of all open tool views and allows you to reintegrate one or all of them with the main window or detach one of them. Those that are already integrated are displayed with a check mark.

Figure 127: Window Integrate Tools Menu



You can also use the Integrate All Tools button on the toolbar to dock all detached views back into the main window.

Using Tab Groups

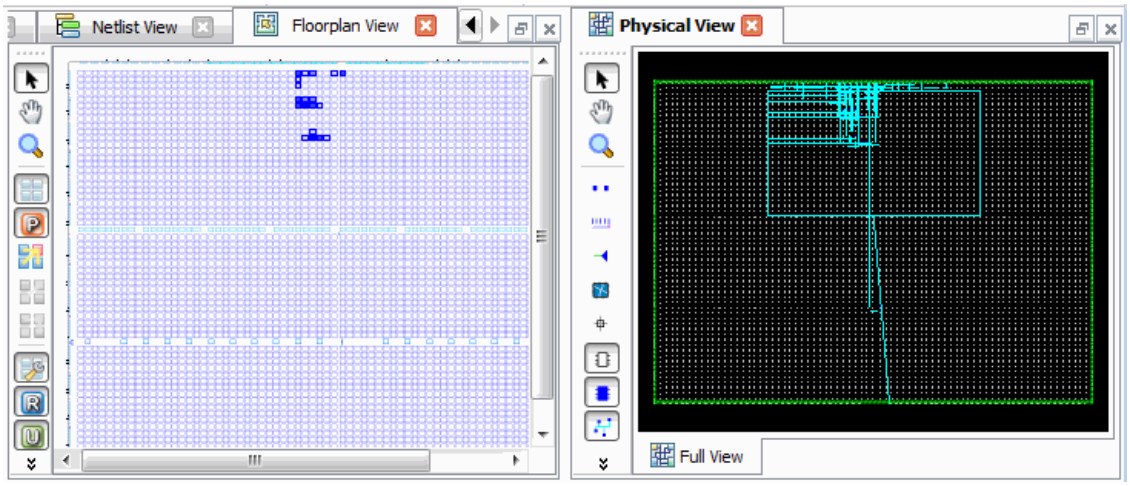
Lattice Diamond allows you to split one or more active tools into a separate tab group. You can use the Window menu or the toolbar buttons to create the tab group and control the display.

Figure 128: Tool Tab Controls

The Split Tab Group button separates the currently active tool into a separate tab group. Having two separate tab groups enables you to work with two tool views side-by-side. This is especially useful for dragging and dropping to make preference assignments; for example, dragging a port from Netlist View to Package View to assign it to a pin or dragging nets to the Route Priority preference sheet to prioritize them.

Having two separate tab groups is also useful for examining the same data element in two different views, such as the Floorplan and Physical View layouts.

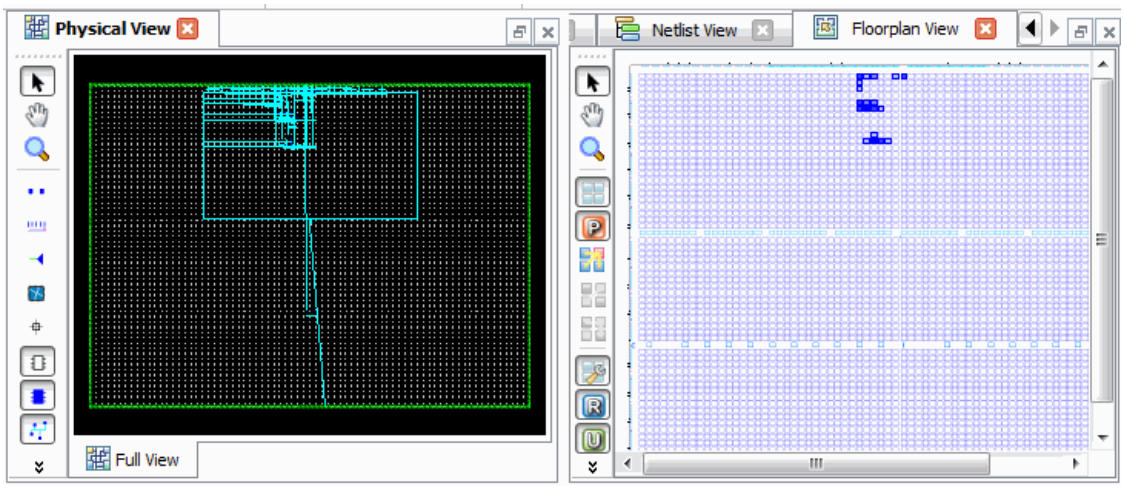
Figure 129: Split Tab Group with Side-by-Side Layout Views



You can move an active tool view from one tab group to another by dragging and dropping it, or you can use the Move to Another Tab Group button on the toolbar.

To switch the positions of the two tab groups, click the Switch Tab Group Position button on the toolbar.

Figure 130: Split Tab Group with Switched Positions



To merge the split tab group back into the main group, click the Merge Tab Group button on the toolbar.

Using Zoom Controls

Lattice Diamond includes display zoom controls in the View toolbar. There are controls for increasing or reducing the scale of the view, fitting the display contents to the window view area, and fitting a selected area or object to the window view area.

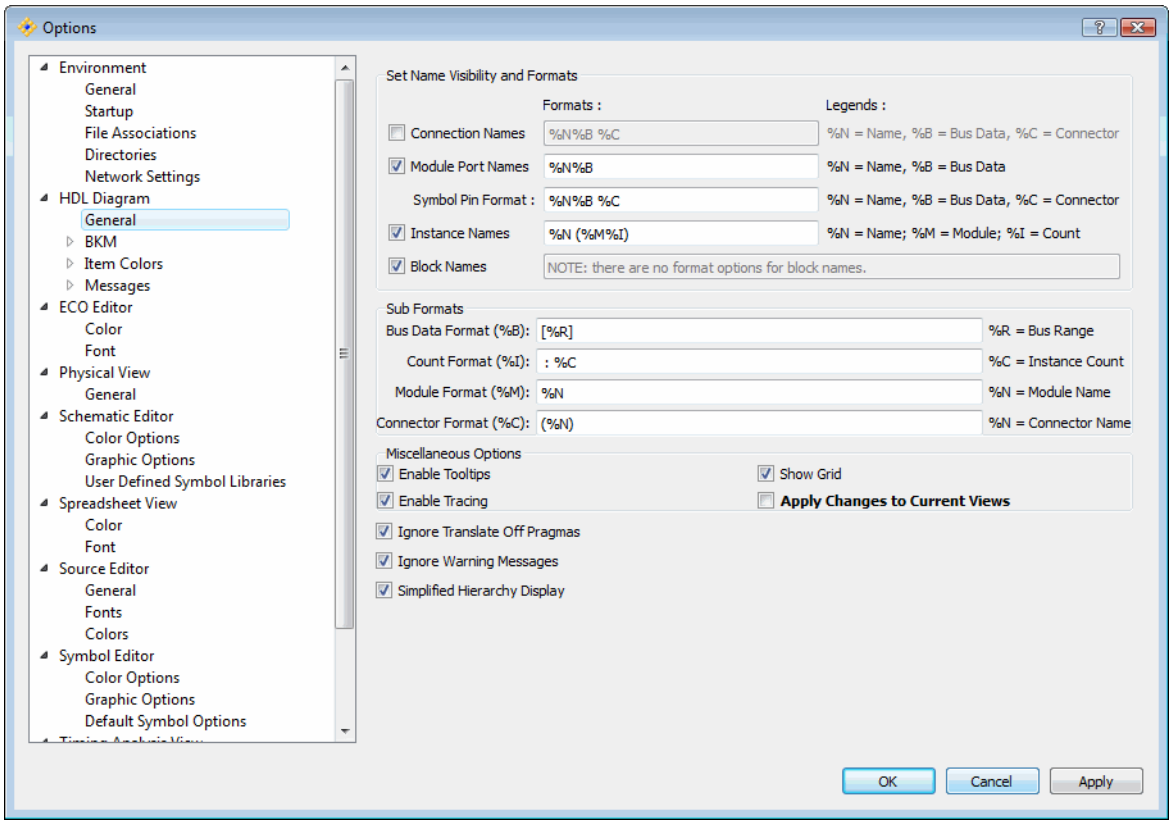
Displaying Tooltips

When you place the cursor over a graphical element in a tool view, a tooltip appears with information on the element. The same information displayed in the tooltip will also be displayed temporarily in the status bar on the lower left of the main window.

Setting Display Options

The Options dialog box enables you to specify general environment options as well as customize the display for the different tools. Tool options include selections for color, font and other graphic elements.

Figure 131: Options for HDL Diagram



8

Tcl Scripting

This chapter describes the Tcl scripting capabilities in Lattice Diamond. The internal TCL Console and external TCL Console are described as well as some of the extended Tcl commands for Lattice Diamond control.

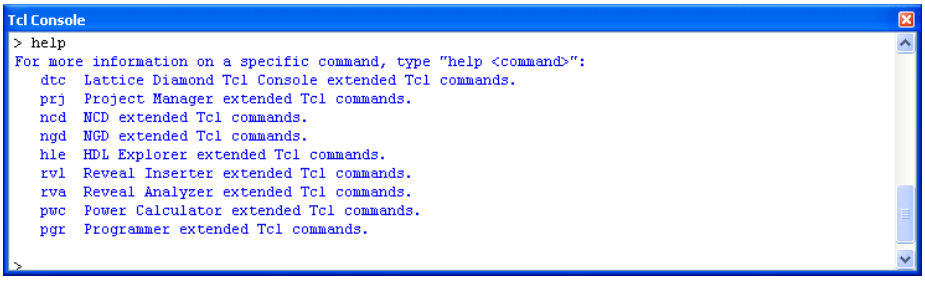
Overview

Tool Command Language (Tcl) is a scripting language used for controlling software tools and automating tool control and testing. It is very useful for controlling batch operation of processes. The Lattice Diamond design environment includes an interactive Tcl Console and extended Lattice Diamond Tcl commands.

Tcl Console

You can view the integrated Tcl Console by selecting its tab at the bottom of the Lattice Diamond main window. You can enter “help” at the Tcl prompt to see the major groups of Tcl commands for Lattice Diamond.

Figure 132: Tcl Console and Command Groups



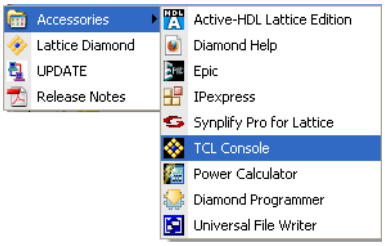
The integrated Tcl Console can be opened, closed, detached and attached

(using the double-click method).

External Tcl Console

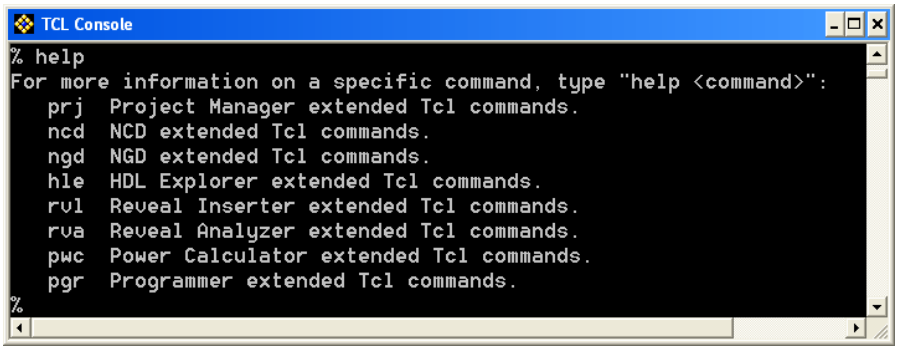
All of the Lattice Diamond Tcl commands that are available in the integrated Tcl Console can also be used in the external console that is included with the Lattice Diamond installation. In the folder where you launched Lattice Diamond is an **Accessories** folder that contains the external Tcl Console.

Figure 133: Launching an External Tcl Console



Select this Tcl Console to run the external Tcl shell. You can enter “help” at the prompt to see the major groups of Tcl commands for Lattice Diamond. Help on individual commands is available both in the online Help and from the “help” command within the console. The console’s help command can be used to get the top level help, help for a particular dictionary, or help for an individual command.

Figure 134: Running Commands in an External Tcl Shell



To use an existing Tcl script file use “source <script_file.tcl>”. To save commands in the Tcl Console to a script file use “save_script <script_file.tcl>”.

See “Advanced Topics” on page 131 for more information on writing Tcl scripts.

Commands

Every function available in the user interface is available to you as part of the extended Tcl commands for Lattice Diamond. You can create scripts to run design flow processes and manage project data as well as perform all standard Tcl operations.

The help command (help) is very useful for getting listings of available commands and their syntax.

```
> help
```

For more information on a specific command, type "help <command>":

```
   dtc  Lattice Diamond Tcl Console extended Tcl commands.
   prj  Project Manager extended Tcl commands.
   ncd  NCD extended Tcl commands.
   ngd  NGD extended Tcl commands.
   hle  HDL Explorer extended Tcl commands.
   rvl  Reveal Inserter extended Tcl commands.
   rva  Reveal Analyzer extended Tcl commands.
   pwc  Power Calculator extended Tcl commands.
   pgr  Programmer extended Tcl commands.
```

The following sections show the help command and output for each major grouping of the Lattice Diamond extended Tcl commands. You can use the help command (*help*) for more information on each specific group or command. See “Advanced Topics” on page 131 for more information on writing Tcl scripts.

Lattice Diamond Tcl Console

```
> help dtc
```

Lattice Diamond Tcl Console extended Tcl commands

```
history:            Shows the commands history
reset:              Reset History and clear up console
clear:              Clear up console
save_script:        Saves a script of executed commands
                        Usage: save_script <script_name>
set_prompt:         Set a new prompt
                        Usage: set_prompt <newPrompt>
```

Project Manager

```
> help prj
```

Project Manager extended Tcl commands
For more information on a specific command, type hlp command-name:

```
  prj_project  Project commands to manipulate project
  prj_src      Project source commands to manipulate project
sources
  prj_impl     Project implementation commands to manipulate
implementation
```

```
  prj_strgy    Project strategy commands to manipulate
strategies
  prj_run      Project flow running command to run a flow
process
  prj_syn      Project synthesis tool commands to list or set
synthesis tool
  prj_dev      Project device commands to list or set the
device used in the project
```

NCD

```
> help ncd
```

NCD extended Tcl commands
For more information on a specific command, type the command without any options:

```
ncd_port          NCD port command
ncd_inst          NCD instance command
ncd_net           NCD net command
ncd_attr          NCD attribute command
```

NGD

```
> hlp ngd
```

NGD extended Tcl commands
For more information on a specific command, type the command without any options:

```
ngd_port          NGD port command
ngd_inst          NGD instance command
ngd_net           NGD net command
ngd_attr          NGD attribute command
```

HDL Explorer

```
> help hle
```

HLE extended Tcl commands
For more information on a specific command, type hlp command-name:

```
hle_design        HLE design command
hle_module        HLE module command
hle_message       HLE messge command
```

Reveal Inserter

```
> help rvl
```

Reveal Inserter extended Tcl commands
For more information on a specific command, type hlp command-name:

```
rvl_project   RVL project commands to manipulate reveal insert
project
rvl_core      RVL core comands to manipulate cores in current
project
```

```
rvl_trace     RVL trace commands to manipulate trace signals
and optins for a debug core in current project
rvl_tu        RVL trigger unit commands to manipulate trigger
units for a debug core in current project
rvl_te        RVL trigger expression commands to manipulate
trigger expressions for a debug core in current project
rvl_tokenmgr  RVL token manager commands to manipulate tokens
in current project
```

Reveal Analyzer

```
> help rva
```

Reveal Analyzer extended Tcl commands
For more information on a specific command, type hlp command-name:

```
rva_trace        Reveal Analyzer trace commands
rva_core         Reveal Analyzer core commands
rva_tu           Reveal Analyzer tu commands
rva_te           Reveal Analyzer te commands
rva_trigoptn     Reveal Analyzer trigger options
rva_project      Reveal Analyzer project commands
```

Power Calculator

```
> help pwc
```

Power Calculator extended Tcl commands
For more information on a specific command, type hlp command-name:

```
pwc_command         Power Calculator command commands
pwc_device          Power Calculator device command
pwc_parameters      Power Calculator parameters command
pwc_thermal         Power Calculator thermal command
pwc_settings        Power Calculator settings command
pwc_supply          Power Calculator supply command
pwc_logicblocks     Power Calculator logicblocks command
pwc_clocks          Power Calculator clocks command
pwc_inout           Power Calculator inout command
pwc_blockram        Power Calculator blockram command
pwc_dspblock        Power Calculator dspblock command
pwc_plldll          Power Calculator plldll command
pwc_maco            Power Calculator maco command
pwc_serdes          Power Calculator serdes command
pwc_writereport     Power Calculator writereport command
pwc_efb             Power Calculator efb command
pwc_misc            Power Calculator misc command
pwc_power           Power Calculator power control command
```

Programmer

> help pgr

Programmer extended Tcl commands

For more information on a specific command, type help command-name:

```
pgr_project      Programmer project commands
pgr_program      Programmer program commands
```

Tcl Scripting From Command-Line Shells

Lattice Diamond Tcl Scripts can also be run directly from a command line shell such as DOS or bash. For this to be accomplished, certain environment variables need to be set up and a Tcl script created.

Lattice Diamond Example Tcl Script

The following example shows a Tcl script created for implementing an existing project.

```
prj_project open "C:/example_path_name/file_name.ldf"
prj_run Synthesis -impl cardgameFPGA
prj_run Translate -impl cardgameFPGA
prj_run Map -impl cardgameFPGA
prj_run PAR -impl cardgameFPGA
prj_run PAR -impl cardgameFPGA -task PARTrace
prj_run Export -impl cardgameFPGA -task Bitgen
prj_project close
```

DOS Script for Running Lattice Diamond Tcl Script

To run the Lattice Diamond Example Tcl script in a Windows DOS shell, the commands from the following DOS script need to be entered.

```
set %LSC_INI_PATH%=
set %LSC_DIAMOND%=true
set %TCL_LIBRARY%=C:\lscc\diamond\1.3\tcltk\lib\tcl8.4
set %FOUNDRY%=C:\lscc\diamond\1.3\ispFPGA
set %PATH%=%FOUNDRY%\bin\nt;%PATH%
C:\lscc\diamond\1.3\bin\nt\pnmainc.exe project.tcl > output.txt
```

The DOS script assumes that the default directory installation is used for the Lattice Diamond software. If the software is installed into a different directory, the script will need to be modified.

Bash Script for Running Lattice Diamond Tcl Script on Windows

Similar to the DOS script, the following example shows how to run a Lattice Diamond Tcl in a Cygwin bash shell on Windows.

```
export TEMP=/cygdrive/c/TEMP
export LSC_INI_PATH=""
export LSC_DIAMOND=true
export TCL_LIBRARY=/cygdrive/c/lscc/diamond/1.3/tcltk/lib/
tcl8.4
export FOUNDRY=/cygdrive/c/lscc/diamond/1.3/ispFPGA
export PATH=$FOUNDRY/bin/nt:$PATH
/cygdrive/c/lscc/diamond/1.3/bin/nt/pnmainc.exe project.tcl >
output1.txt
```

Like the DOS script, the bash script assumes that the default directory installation is used for the Lattice Diamond software. If the software is installed into a different directory, the script will need to be modified.

Bash Script for Running Lattice Diamond Tcl Script on Linux

Lattice Diamond software is supported on Linux also. The following is an example for running the Lattice Diamond Tcl script example in a Linux bash shell.

```
export TEMP=/tmp
export LSC_INI_PATH=""
export LSC_DIAMOND=true
export TCL_LIBRARY=/usr/local/lscc/diamond/1.3/tcltk/lib/tcl8.4
export FOUNDRY=/usr/local/lscc/diamond/1.3/ispFPGA
export PATH=$FOUNDRY/bin/nt:$PATH
/usr/local/lscc/diamond/1.3/bin/lin/diamondc project.tcl >
output1.txt
```

As in the previous examples, the Linux example assumes that the default directory installation is used for the Lattice Diamond software. If the software is installed into a different directory, the script will need to be modified.

9

Advanced Topics

This chapter explains advanced concepts, features and operational methods for Lattice Diamond.

Shared Memory Environment

The Lattice Diamond design environment uses a shared memory architecture. Shared memory allows all internal tool views to access the same image of the design at any point in time. Understanding how shared memory is being used can give you insight into managing the environment for optimum performance, especially when your design is large.

There is one shared database that contains the device, design, and preference information in system memory.

Generating the hierarchy of your design uses an additional database separate from the main shared memory database.

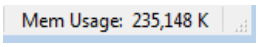
External tools referenced from within Lattice Diamond, such as those for synthesis and simulation, use their own memory in addition to what is used by Lattice Diamond.

When you launch the first tool view that accesses shared memory, it will take longer than the launch of subsequent views.

Memory Usage

The main window of the UI displays a memory usage figure in the lower right corner.

Figure 135: Memory Usage



This indicates the current memory usage for the Lattice Diamond environment, including all open tools.

Clear Tool Memory

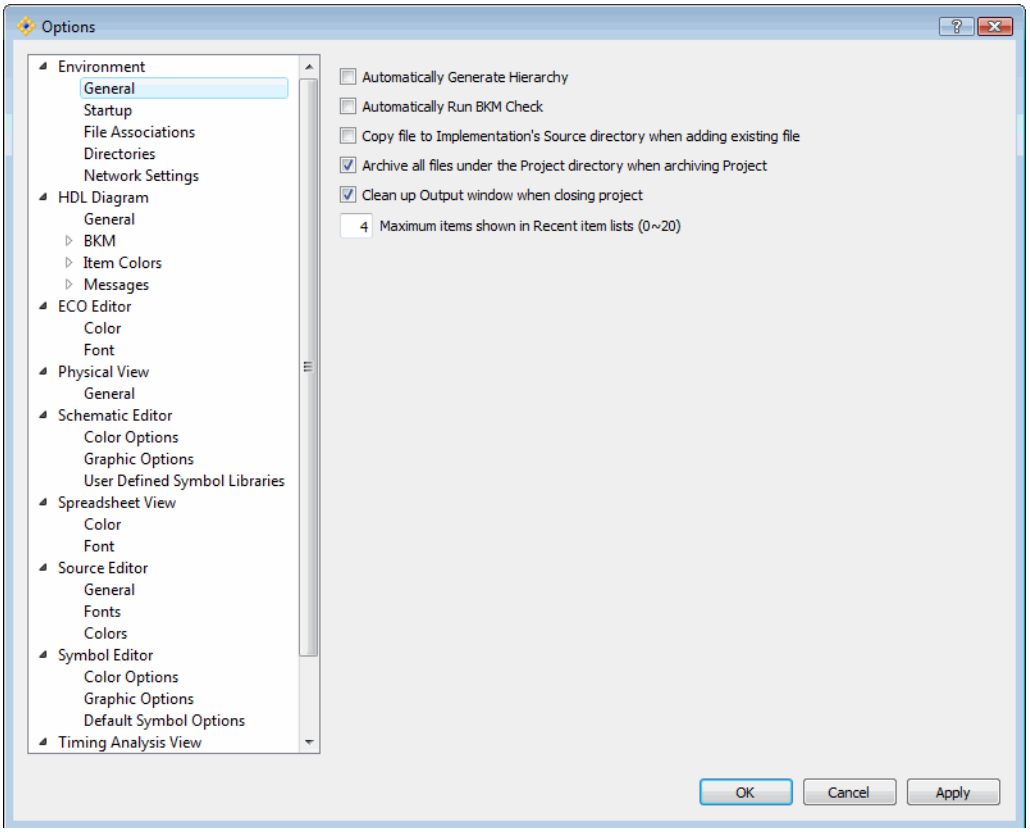
The “Clear Tool Memory command, available from the Tools menu, clears the device, design, and preference information and the HDL Diagram database from system memory. Clearing the tool memory can speed up memory-intensive processes such as place and route. When your design is very large, it is good practice to clear memory prior to running place and route.

If you have open tool views that will be affected by clearing the tool memory, a confirmation dialog box will open to give you the opportunity to cancel the memory clear.

Environment and Tool Options

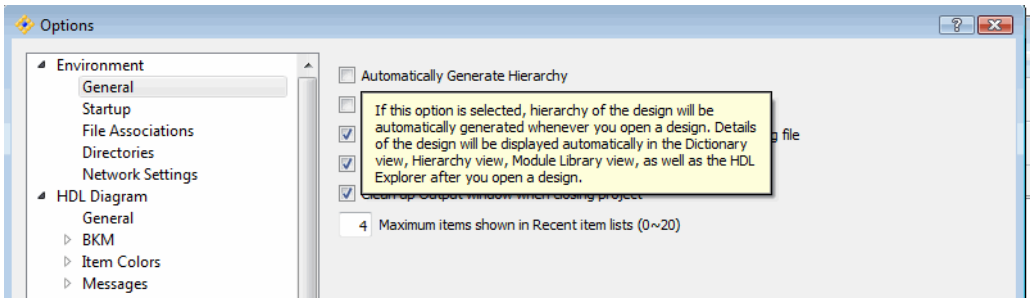
Lattice Diamond provides many environment control and tool view display options that enable you to customize settings. Choose **Tools > Options** to access these options.

Figure 136: Environment and Tool Options



The Options dialog box is organized into functional folders. You can use the context-sensitive help button in the upper right to view information about each parameter in a folder. Click the context-sensitive help button, and then click the item of interest.

Figure 137: Context-Sensitive Help Information



Commonly configured items include:

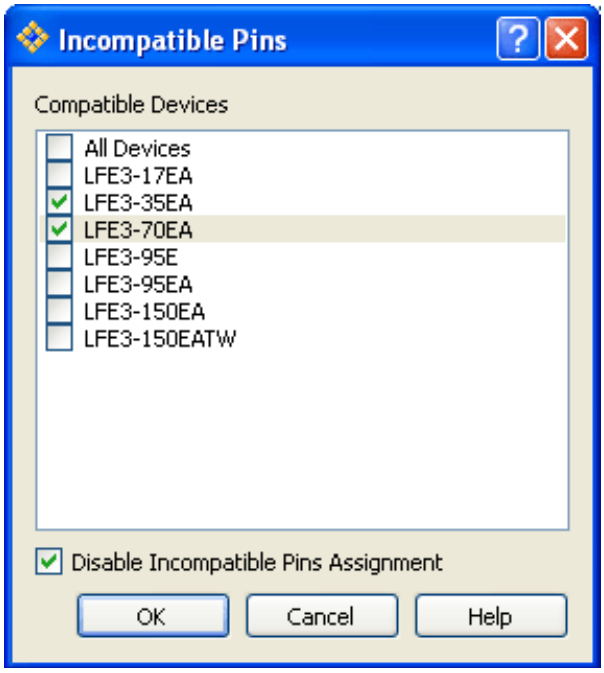
- Environment > General > Automatically Generate Hierarchy – runs the Generate Hierarchy function whenever a project is opened.
- Environment > Startup – enables you to configure the default action at startup and also to control the frequency of checking for software updates.
- Environment> File Associations – allows you to set the programs to be associated with different file types based on the file extensions.
- HDL Diagram > BKM – allows you to configure the settings for checks to be performed in BKM checking.

The tool view folders include customizable display properties such as color and font.

Pin Migration

The “Incompatible Pins” dialog box helps you migrate pin assignments to a different device of the same family and package. This information enables you to save your current pinout while you explore other devices.

Figure 138: Incompatible Pins Dialog Box



The “Incompatible Pins” dialog box is available from the View menu in Spreadsheet View or Package View after the Translate Design process. After selecting one or more devices for possible pin migration, you can view the incompatible pins in Package View and in the Pin Assignments sheet of Spreadsheet View.

For more information about pin migration, refer to “Migrating Pin Assignments” in the Lattice Diamond online Help.

Batch Tool Operation

The core tools in the FPGA implementation design flow can all be run in batch mode using command-line tool invocation or scripts. For detailed information, see the *Command Line Reference Guide*, available from the Lattice Diamond Start Page.

Tcl Scripts

The Diamond Extended Tcl language enables you to create customized scripts for tasks that you perform often in Lattice Diamond. Automating these operations through Tcl scripts not only saves time, but also provides a uniform approach to design. This is especially useful when you try to find an optimal solution using numerous design implementations.

Creating Tcl Scripts from Command History

A good first step in Tcl programming is to create a Tcl script by saving some command history and then modifying it as needed. This allows you to get started by using existing command information.

To create a Tcl command script using command history:

1. In the Tcl Console window, first perform a *reset* command so that your script won’t contain any of the actions that may have already executed.

   ```
   reset
   ```

2. Now perform the commands that you want to save as a script.
3. Optionally, enter the history command in the Tcl Console window to ensure that the commands you wish to save are in the console’s memory.
4. In the Tcl Console window type

   ```
   save_script <script_name>
   ```

   where <*script_name*> is any identifier that has no spaces and contains no special characters except underscores. For example, *myscript* or *design_flow_1* are acceptable script name values, but *my$script* and *my script* are invalid.

   It is possible to give your <*script_name*> value a file path and name if you want to keep it in a subfolder in your project or in a folder where you generally keep your scripts. However, you must save it to an existing folder. File paths, using forward slashes, with an identifier are valid if using an absolute file path to an existing folder.

5. Navigate to your script file and use the text editing tool of your choice to make any necessary changes, such as deleting extraneous lines or invalid arguments.

In most cases, you will need to edit the script you saved and take out any invalid arguments or any commands that cannot be performed in the Lattice Diamond environment due to a conflict or exception. You will need to revisit this step later if, after running your script, you experience any run errors due to syntax errors or technology exceptions.

Creating Tcl Scripts from Scratch

Tcl commands can be written directly into a script file. You can use any text editor, such as Notepad or vi, to create a file and type in the Tcl commands.

Sample Tcl Script

The following Tcl example shows a simple script that opens a project, runs the entire design flow through the Place & Route process, and then closes the project. A typical script would probably contain more steps, but you can use this example as a general guideline.

```
prj_project open "C:/lscc/diamond/edif_counter/edif.ldf"
prj_run PAR -impl edif -forceAll
prj_project close
```

Running Tcl Scripts

You can run scripts from the Lattice Diamond integrated Tcl Console whether your project is opened or not. You can also run scripts from the external Tcl Console prompt window. The following example procedure uses the integrated Tcl Console and the sample Tcl script from the previous section:

To run a Tcl script that opens a project, runs processes and closes the project:

1. Open Lattice Diamond but do not open your project. If your project is open, choose **File > Close Project**.
2. In the Diamond main window, click the Tcl Console tab at the bottom to open the console.
3. If there are previously issued commands in the console, type **`reset`** in the console command line to refresh your session and clear out all previous commands.

   ```
   reset
   ```

4. Run the source command using the absolute file name and path to your script.

   ```
   source C:/lscc/diamond/1.0/examples/edif_counter/myscript2
   ```

   The sample Tcl script opens the project, runs the flow through Place & Route, and closes the project.

As long as there are no syntax errors or invalid arguments, this procedure should open your project, and you should see the processes running to completion.

If there are errors in the script, you will see the errors in red in the Tcl Console after you attempt to run it. You will then need to return to your script and edit it as needed.

Project Archiving

A Diamond project archive is a single compressed file (.zip) of your project. The project archive can contain all of the files in your project directory, or it can be limited to source-related files. When you use the **File > Archive Project** command, the dialog box provides the option to “Archive all files under the Project directory.” When you select this option, the entire project is archived. When you clear this option, only the project’s source-related files, including strategies, are archived. Many of these source-related files must be archived in order to achieve the same bitstream results for a fully implemented design.

Whichever archiving method you select, if your project contains source files stored outside the project folder, the remote files will be compressed under the remote_files subdirectory in the archive; for example:

<project_name>/remote_files/sources
<project_name>remote_files/strategies

Appendix A “File Descriptions” on page 139 provides lists of the file types used in Diamond, including those generated during design implementation. The Archive column indicates the files that must be archived in order to achieve the same bitstream results.

A

File Descriptions

The following tables describe many of the files that are used in Lattice Diamond, including files generated by the implementation processes. A check mark in the Archive column indicates that a file of this particular type, when used in implementing the design, must be included in the project's archive in order to create the same bitstream results.

Table 1: Source Files

File Type	Definition	Function	Archive?
.edf	Default EDIF source file generated by Precision	Used in design translation to create the .ngd netlist.	✓
.edn	Default EDIF source file generated by Synplify	Used in design translation to create the .ngd netlist.	✓
.ipx	Manifest file generated by IPexpress.	Enables changes to be made to a module or IP IPexpress.	✓
.ldc	SDC constraints file	Used for specifying design-specific constraints for the Lattice Synthesis Engine (LSE).	✓
.ldf	Diamond Project file	Used for managing and implementing all project files in Diamond.	✓
.lib	Schematic symbol library	Used for adding symbols to a schematic source file.	✓
.lpf	Diamond project logical preference file	Stores logical constraints for developing and implementing the design.	✓
.mem	Memory Generator source file	Specifies the initial contents for the memory modules in your design. Required for creating ROM and optional for creating RAM modules.	✓
.ngo	Native Generic Object (NGO) netlist format	Used as an alternative to HDL for creating a black-box module.	✓

Table 1: Source Files (Continued)

File Type	Definition	Function	Archive?
.pcf	Power Calculator project file	Stores power analysis results from information extracted from the design project.	✓
.rva	Reveal Analyzer file	Defines the Reveal Analyzer project and contains data about the display of signals in Waveform View.	✓
.rvl	Reveal Inserter debug file	Defines the Reveal project and its modules, identifies the trace and trigger signals, and stores the trace and trigger options.	✓
.sch	Schematic source file	Schematic representation of the circuit in terms of the components used and how they connect to each other.	✓
.sdc	SDC constraints file	Used for specifying design-specific constraints for SynplifyPro or Precision synthesis tools.	✓
.sym	Symbol Editor source file	Used for creating symbols or primitive elements that represent an independent schematic module.	✓
.tpf	Timing Analysis Preference file	Used for obtaining quick analysis of timing preferences.	✓
.v	Verilog source file	Verilog description of the circuit structure and function	✓
.vhd	VHDL source file	VHDL description of the circuit structure and function.	✓
.wdl	Waveform Editor source file	Used for creating test stimulus files for simulation.	✓
.xcf	Configuration chain file	Used for programming devices in a JTAG daisy chain.	✓

Table 2: IPexpress Files

File Type	Definition	Function	Archive?
Module Files			
%instName.lpc	Module parameter configuration file	Stores the user configurations for the module.	✓
%instName.v	Verilog module file	Verilog netlist generated by IPexpress to match the user configuration of the module.	✓
%instName.vhd	VHDL module file	VHDL netlist generated by IPexpress to match the user configuration of the module.	✓
%instName_tmpl.v	Verilog template file	A template for instantiating the generated module. This file can be copied into a user Verilog file.	✓

Table 2: IPexpress Files (Continued)

File Type	Definition	Function	Archive?
%instName_tmpl.vhd	VHDL module template file	A template for instantiating the generated module. This file can be copied into a user VHDL file.	✓
tb_%instName_tmpl.v	Verilog testbench template	A simple Verilog testbench for the generated module. Can be edited to add more vectors, which will be overwritten during module regeneration.	✓
tb_%instName_tmpl.vhd	VHDL module testbench file	A simple VHDL testbench for the generated module. Can be edited to add more vectors, which will be overwritten during module generation.	✓
IP Files			
%instName.lpc	IP parameter configuration file	Used for re-creating or modifying the core in the IPexpress tool.	✓
%instName_bb.v	Verilog IP black box file	Provides the Verilog synthesis black box for the IP core and defines the port list.	✓
%instName_beh.v	Verilog IP behavioral file	Provides a behavioral simulation model for the IP core.	✓
%instName_inst.v	Verilog IP instantiation file	Provides a Verilog instance template for the IP core.	✓
%instName_inst.vhd	VHDL IP instantiation file	Provides a VHDL instance template for some IP cores. Archiving this file, though optional, will make it easier to write a top-level VHDL file and copy and paste the instantiation section from this file.	
%instName.ngo	IP database file	Provides the netlist in encrypted format for the IP core and is used by Map and PAR tools.	✓
pmi_*.ngo	Support database files	Provides the netlist files in encrypted format for the Lattice parameterizable module instances (PMI) used within the IP core.	✓
%IPName_params.v	IP parameters file	Contains the Verilog parameters used by the IP core simulation model.	✓
*.txt	SERDES configuration file	Contains the SERDES configuration that is required during bitgen process.	✓

Table 3: Implementation Files

File Type	Definition	Function	Archive?
.bgn	Bitstream generation report file	Reports results of a bit generation (bitgen) run and displays information on options that were set.	
.bit	Bitstream file	Used for SRAM FPGA programming.	

Table 3: Implementation Files (Continued)

File Type	Definition	Function	Archive?
.edi	SynplifyPro EDIF output file	Netlist file generated by Diamond. The file extension must be changed to .edf or .edn in order to import it into a project.	
.ibs	Post-Route I/O buffer information specification file (IBIS)	Used for analyzing signal integrity and electromagnetic compatibility (EMC) on printed circuit boards	
.ior	Post-PAR I/O Timing Analysis file	For each input data port, reports setup and hold time requirements and min/max clock-to-out delay for each output port.	
.jed	JEDEC file	Used for Flash FPGA programming	
_map.ncd	Mapped native circuit description netlist file.	Includes mapping information.	
_mapvho.sdf	Post-map SDF simulation file for VHDL	Used for post-map functional simulation.	
_mapvho.vho	Post-map simulation file for VHDL	Used for post-map functional simulation.	
_mapvo.sdf	Post-map SDF simulation file for Verilog	Used for post-map functional simulation.	
_mapvo.vo	Post-map simulation file for Verilog	Used for post-map functional simulation.	
.mcs	PROM file	Used for SRAM FPGA programming	
.mrp	Map Report file	Provides statistics about component usage in the mapped design.	
.ncd	Post-route native circuit description netlist file.	Includes placement and routing information.	
.ngd	Native generic description file produced by the ngdbuild translation process.	Used by Map to map logical elements to physical elements in the native circuit description file (.ncd).	
.ngo	Native generic object file generated by the edif2ngo translation process. Contains logical descriptions of the design's components and hierarchy	Used as input to ngdbuild, which converts the .ngo to an .ngd file.	
.pad	Post-Route PAD report file	Lists all programmable I/O cells used in the design and their associated primary pins.	
.par	Post-Route Place & Route report file	Summarizes information from all iterations and shows the steps taken to achieve a placement and routing solution.	
.prf	Physical preference file produced by the Map Design process.	Used as an input file to placement, routing and TRACE.	
.sso	Post-PAR SSO analysis file	Reports the noise caused by simultaneously switching outputs.	
.tw1	Post-Map TRACE analysis file	Estimates pre-route timing.	
.twr	Post-PAR TRACE analysis file	Reports post-route timing.	
.vho	Post-Route VHDL simulation file	Used for post-route simulation.	

Table 3: Implementation Files (Continued)

File Type	Definition	Function	Archive?
.vo	Post-Route Verilog simulation file	Used for post-route simulation.	
_vho.sdf	Post-Route SDF simulation file for VHDL	Used for timing simulation.	
_vo.sdf	Post-Route SDF simulation file for Verilog	Used for timing simulation.	

Index