

Lattice Diamond Tutorial



April 2013

Copyright

Copyright © 2013 Lattice Semiconductor Corporation.

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

Trademarks

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

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

Disclaimers

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

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

Type Conventions Used in This Document

Convention	Meaning or Use
Bold	Items in the user interface that you select or click. Text that you type into the user interface.
<i><Italic></i>	Variables in commands, code syntax, and path names.
Ctrl+L	Press the two keys at the same time.
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

Lattice Diamond Tutorial	1
Learning Objectives	1
Time to Complete This Tutorial	2
System Requirements	2
Accessing Online Help	2
About the Tutorial Design	2
About the Tutorial Data Flow	2
Task 1: Create a New Lattice Diamond Project	4
Task 2: Create an IPexpress Module	10
Task 3: Check Hardware Description Language	12
Task 4: Verify Functionality with Simulation	15
Task 5: Inspect Strategy Settings	18
Task 6: Examine Resources	20
Task 7: Run Synthesis Process	23
Task 8: Set Timing and Location Assignments	24
Task 9: Running Place and Route	29
Task 10: Examine Post Place and Route Results	32
Task 11: Adjust Static Timing Constraints and Review Results	36
Task 12: Comparing Multiple Place and Route Runs	39
Task 13: Analyze Power Consumption	40
Task 14: Run Export Utility Programs	42
Task 15: Download a Bitstream to an FPGA	42
Task 16: Convert a File Using Deployment Tool	44
Task 17: Use Reveal Inserter to Add On-chip Debug Logic	48
Setting Up the Trigger Units	49
Setting Up the Trigger Expressions	50
Inserting the Debug Logic	51

Generating a Bitstream and Programming the FPGA	53
Task 18: Use Reveal Logic Analyzer Perform Logic Analysis	53
Creating a New Reveal Logic Analyzer Project	53
Running the Logic Analyzer	55
Summary of Accomplishments	56
Recommended References	57

Lattice Diamond Tutorial

The next generation design tool for FPGA design, Lattice Diamond™, is designed to address the needs of high-density FPGA designers.

This tutorial leads you through all the basic steps of designing and implementing a mixed VHDL, Verilog, and EDIF design targeted to the LatticeECP3 device family. It shows you how to use several processes, tools, and reports from the Lattice Diamond software to import sources, run design analysis, view design hierarchy, and inspect strategy settings. The tutorial then proceeds to step through the processes of adding and editing a strategy, specifying the synthesis requirements, examining the device resources, setting timing and location assignments, and editing preferences to configure the filter to implement the design to the target device.

Learning Objectives

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

- ▶ Create a new Lattice Diamond project
- ▶ Create an IPexpress module
- ▶ Check Hardware Description Language (HDL)
- ▶ Verify functionality with simulation
- ▶ Inspect strategy settings
- ▶ Examine resources
- ▶ Run synthesis processes
- ▶ Set timing and location assignments
- ▶ Run place and route
- ▶ Examine post place and route results

- ▶ Adjust static timing constraints and review results
- ▶ Compare multiple place and route runs
- ▶ Analyze power consumption
- ▶ Run export utility programs
- ▶ Download a bitstream to an FPGA
- ▶ Convert a file using Deployment Tool
- ▶ Use Reveal Inserter to add on-chip debug logic
- ▶ Use Reveal Logic Analyzer perform logic analysis

Time to Complete This Tutorial

The time to complete this tutorial is about 90 minutes.

System Requirements

The following software is required to complete the tutorial:

- ▶ Lattice Diamond software
- ▶ (Optional) LatticeECP3 Versa Development Kit

Accessing Online Help

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

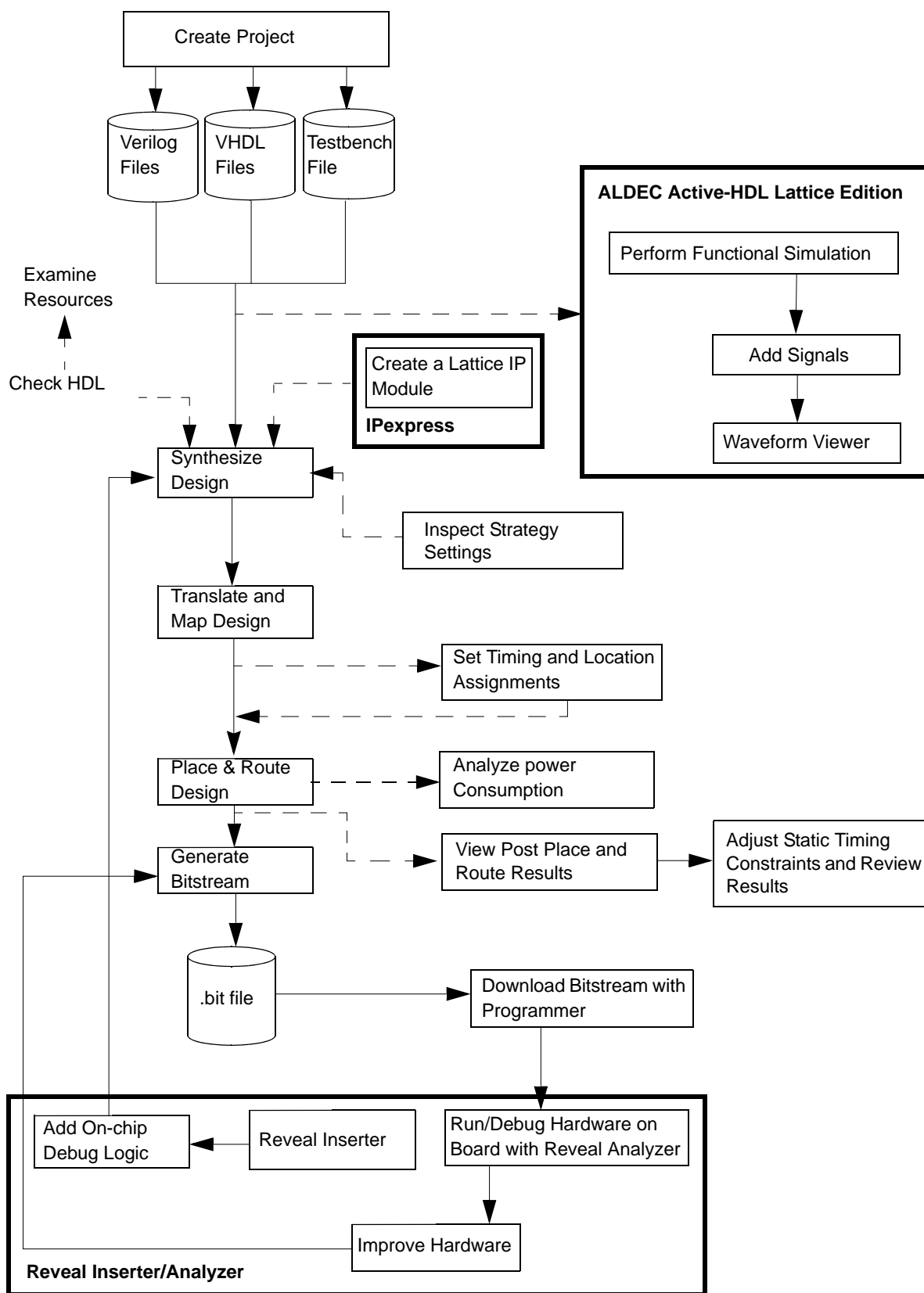
About the Tutorial Design

The design in this tutorial consists of a Verilog HDL module, two VHDL modules, and one EDIF module. The design that you create is targeted to LatticeECP3 device families.

About the Tutorial Data Flow

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


Figure 1: Tutorial Data Flow



Task 1: Create a New Lattice Diamond Project

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

To create a new project:

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

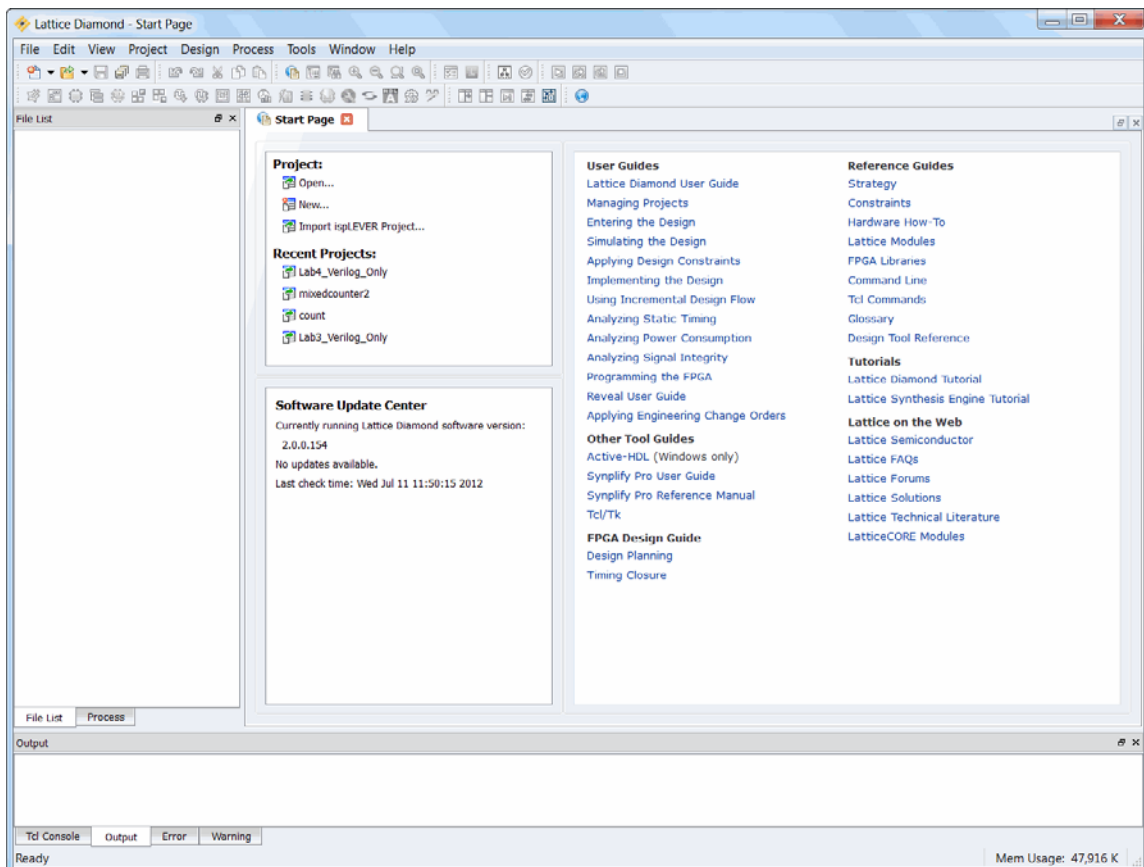
Note

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

- ▶ From your Linux platform shell window or C-shell window, execute:
`<install_path>/bin/linux/diamond`

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



Figure 2: Diamond Design Environment.



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

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

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

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

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

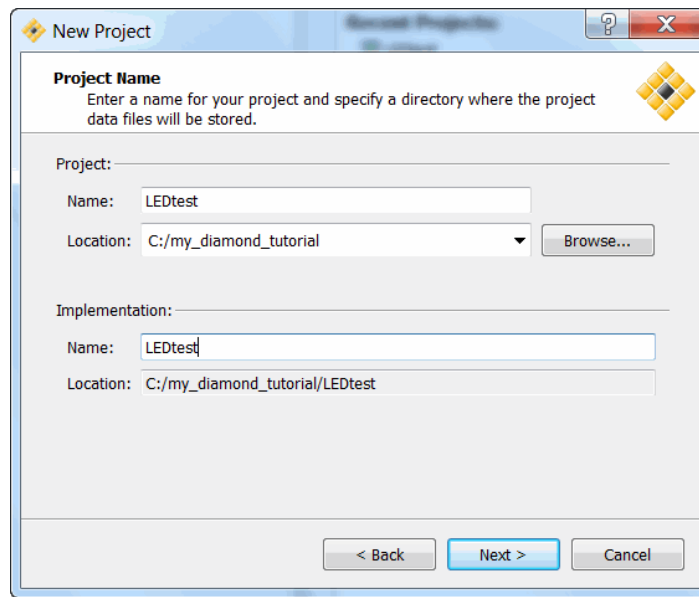
3. Specify Project name: **LEDtest**

Note

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

4. Click **Browse**. In the Project Location dialog box, browse to where you want to store the project's files. Click **Select Folder**.

By default, when you specify the project name, the implementation name is simultaneously specified the same, as shown in Figure 3. For this tutorial, leave the default implementation name as **LEDtest**. The directory to store the implementation is automatically displayed in the Location area. We will talk about creating a new implementation later in this tutorial.

Figure 3: New Project Window.

5. Click **Next**. The Add Source dialog box appears.
6. Click **Add Source**. The **Import File** dialog box appears.
7. Navigate to the folder containing the source files, which are located in the `<diamond_install_directory>/docs/tutorial/Diamond_tutorial` directory. Select the following files in the directory:

- ▶ clockDivider.v
- ▶ count4.v
- ▶ count8.vhd
- ▶ LEDtest.v,
- ▶ testbench.v
- ▶ topcount.v
- ▶ typepackage.v

and click **Open**.

The Add Source step of the Wizard appears with all the selected source files added.

8. Select **Copy source to implementation directory** and click **Next**.

The Device Selector dialog box appears.

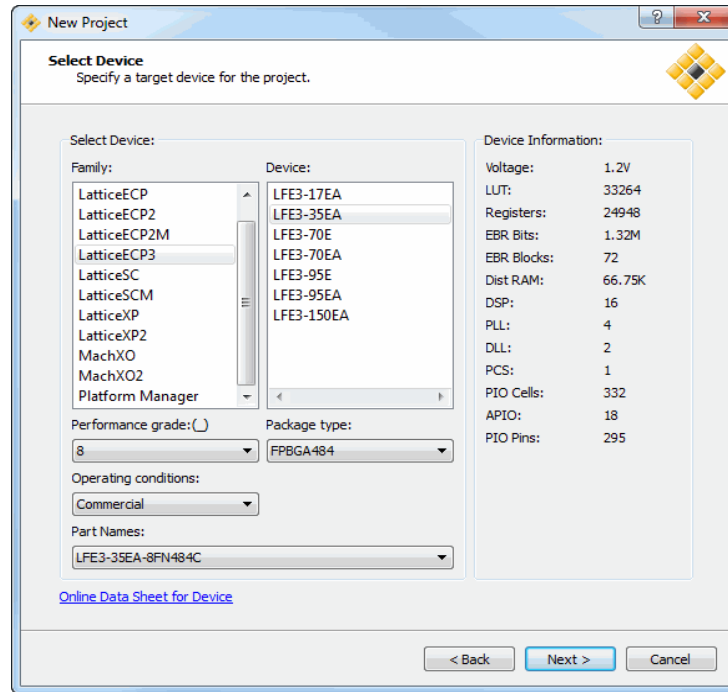
9. Select the following device options:

- ▶ Family: **LatticeECP3**
- ▶ Device: **LFE3-35EA**
- ▶ Performance Grade: **8**
- ▶ Package type: **FPBGA484**
- ▶ Operating Conditions: **Commercial**

► Part Names: **LFE3-35EA-8FN484C**

The dialog box should resemble Figure 4.

Figure 4: New Project Wizard Device Selector Dialog Box



10. Click **Next**.

The Select Synthesis Tool dialog box opens. For this tutorial, choose **Synplify Pro**.

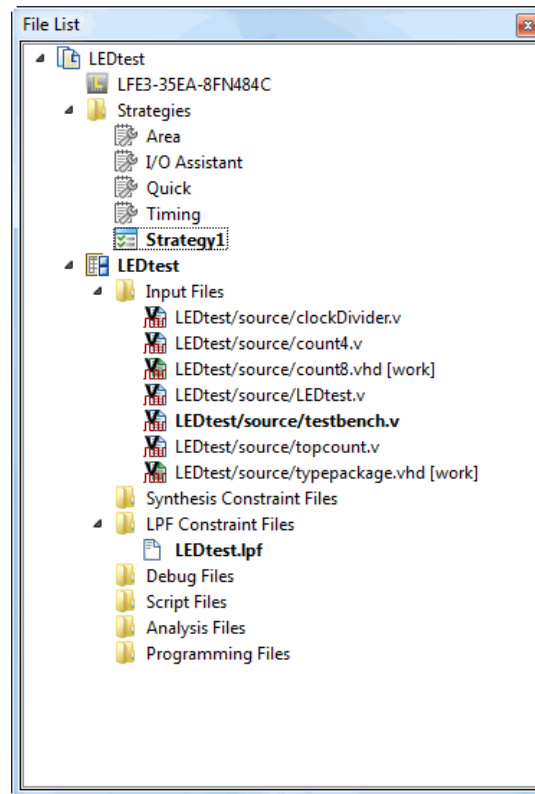
11. Click **Next**.

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

12. Click **Finish**.

The File List and Process views are populated and the Reports view appears.

The File List view, shown in Figure 5, displays the components of the project. The imported VHDL, Verilog, and EDIF files appear in the Input Files folder in the File List view. The File List view organizes project files by categories: Strategies, and Implementation including Input Files, Constraint Files, Debug Files, Script Files, and Analysis Files. You may adjust file order by dragging and dropping of the filenames in the list. Properties of each file are accessed by highlighting a file, clicking the right mouse button, and selecting Properties from the pop-up menu.

Figure 5: File List View**Note**

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

When you create a new project in Diamond, a logical preference file (.lpf) is automatically generated and assigned the same name as the FPGA project.

For this tutorial a logical preference file named **pin_assignments.lpf** is provided and contains all the pin assignments needed to program this design project onto the LatticeECP3 FPGA. All changes that you make to logical constraints will be saved in this file until you create a new logical preference file or add another existing one.

13. In the File List, right-click on **LPF Constraint file**, and select **Add > Existing File**.

Add Existing File dialog box appears.

14. Navigate to <diamond_install_directory>/docs/tutorial/Diamond_tutorial and select the file **pin_assignments.lpf**. Choose **Copy file to Implementation's Source directory**, and click **Add**.

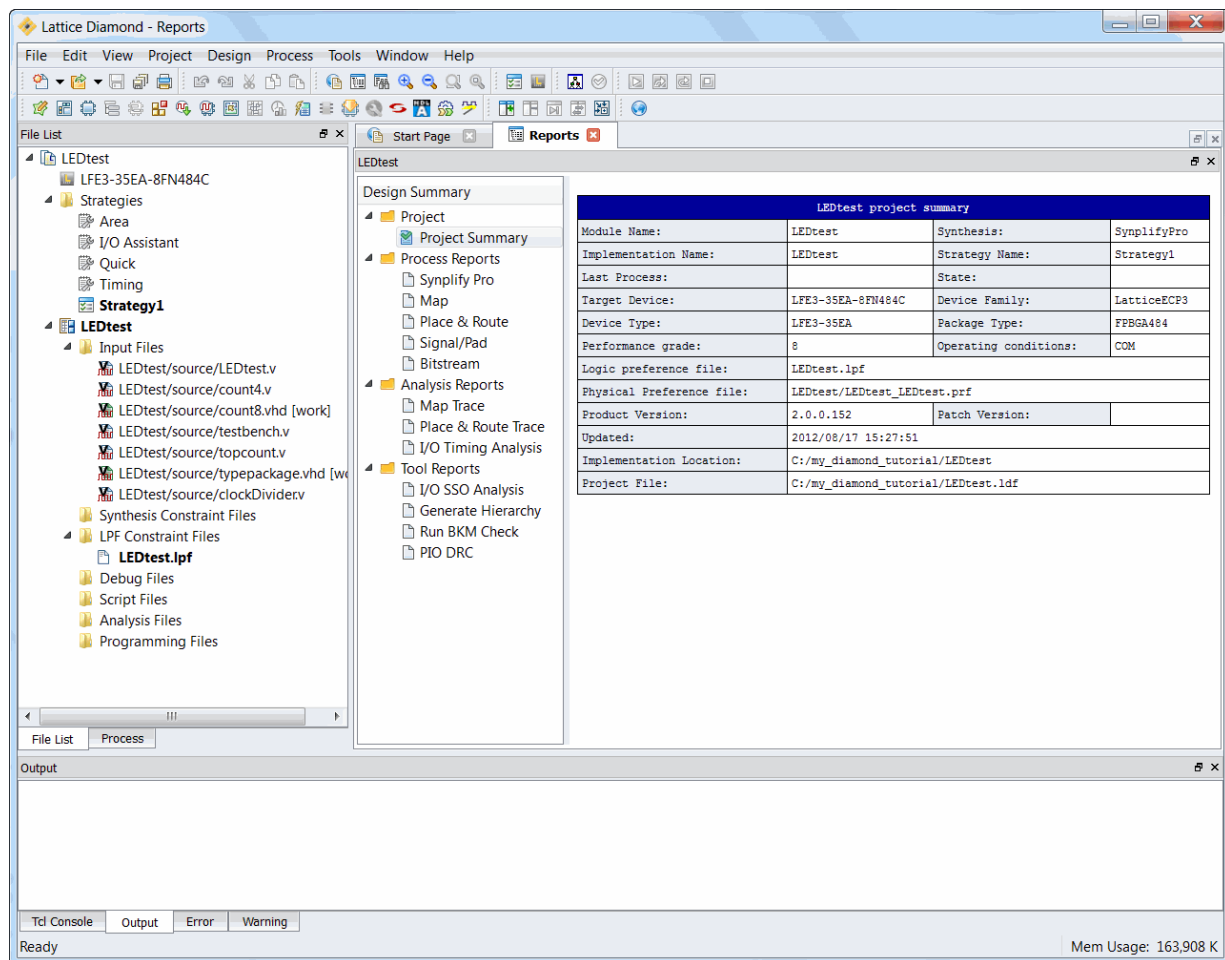
15. In the File List, right-click on pin_assignments.lpf and choose **Set as Active Preference File**.
16. In the File List, right-click on testbench.v and choose **Include for > Simulation**.

The Process view, shown in Figure 6, lists all the processes available, such as Synthesize Design, Translate Design, Map Design, Place & Route Design, and Export Files.

The Reports view provides a way to examine and print process reports. The Reports view displays reports for the major processes. There are two panes in the Reports view. The left pane lists the reports. The right pane displays the reports.

Log messages are displayed in the Output frame of the Diamond main window.

Figure 6: Process View and Reports View



Task 2: Create an IPexpress Module

IPexpress is an easy way to use a collection of modules from Lattice Semiconductor. With IPexpress these modules can be extensively customized. They can be created as part of a specific project or as a library for multiple projects.

In this task, you will generate a phase lock loop (PLL) module that will be imported into your design.

To Generate and Import a Module with IPexpress:


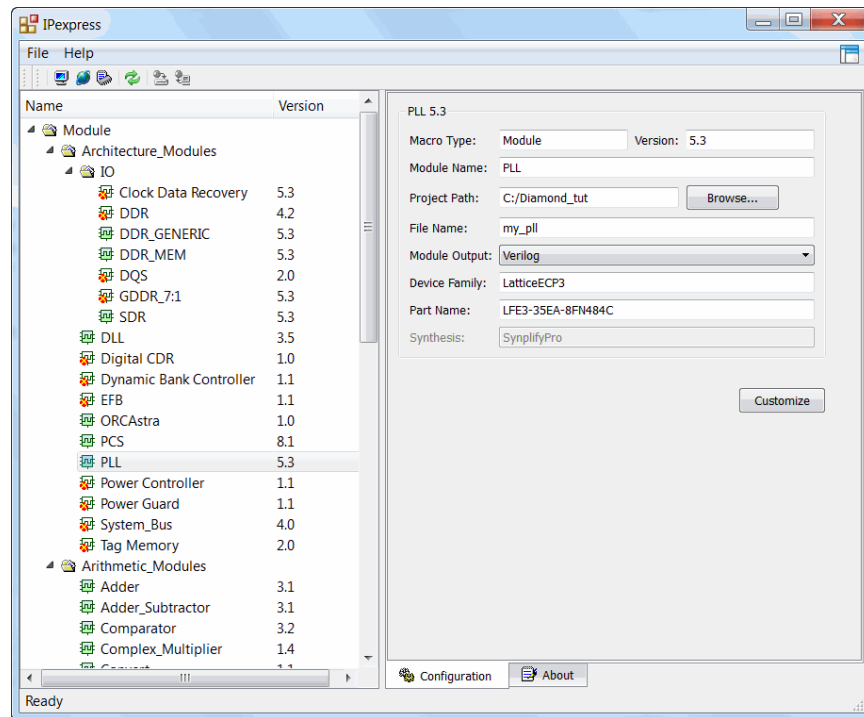
1. Choose **Tools > IPexpress**. The IPexpress tool appears.
2. Click the  icon in the upper right corner to detach the IPexpress window. An index of the available modules for the target device appear in the module tree on the left, as shown in Figure 7.

Figure 7: New Project Wizard Device Selector Dialog Box



3. In the module tree, under Module > Architecture_Modules, select **PLL**.
4. In the Configuration tab, all information is filled in from the design project except for File Name and Module Output. For this tutorial enter **my_pll** as File Name, **Verilog** as the Module Output, then click **Customize**.
5. In the Configuration tab:
 - ▶ Select **Frequency Mode**.
 - ▶ For CLKI Frequency enter **100**.

- ▶ For CLKOP/CLKOS (Non Bypass Mode) Desired Frequency enter **500**.
 - ▶ Select **Enable CLKOK**
 - ▶ For CLKOK Desired Frequency enter **50**
6. Click **Calculate**.
 7. Click **Import IPX to Diamond project**.
 8. Click **Generate**.

The IPExpress .ipx file is generated.

9. Click the **Generated Log** tab to view the log file, as shown in Figure 9, then click Close.

The IPExpress .ipx file is imported into the Diamond project and appears in the File List as shown in Figure 9 on page 12.

Figure 8: IPExpress Generate Log Tab

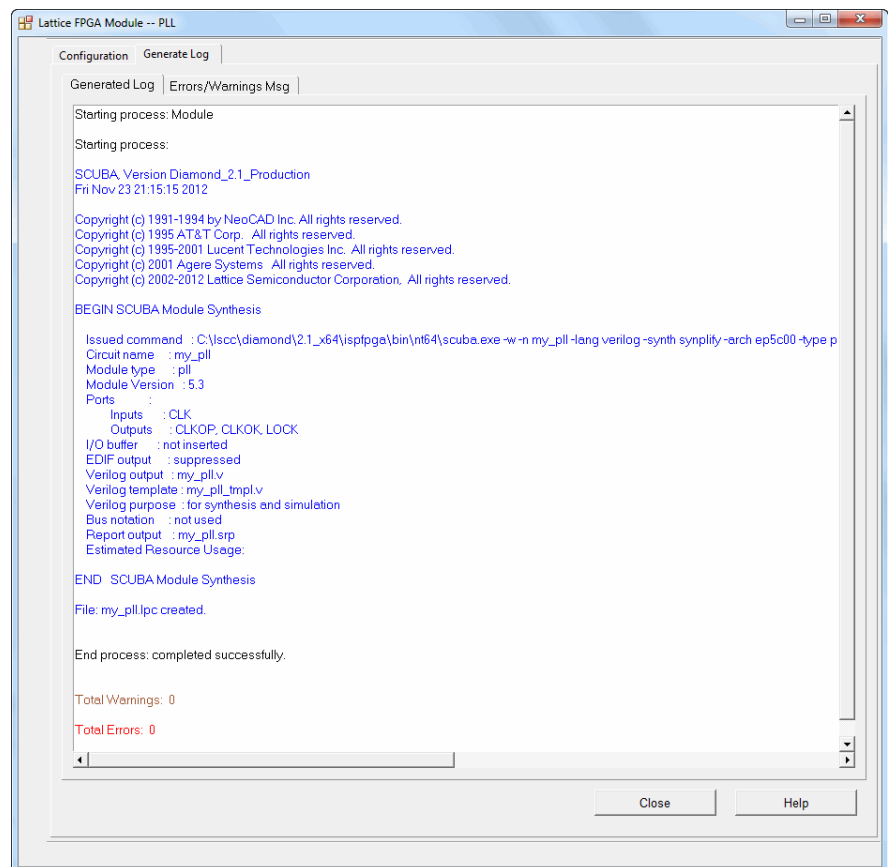
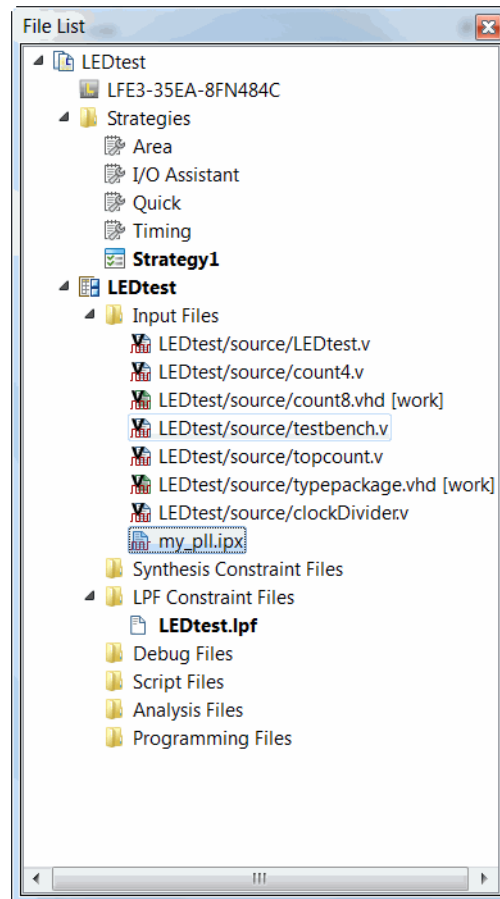


Figure 9: PLL Generated by IP Express Imported into Diamond Project

Task 3: Check Hardware Description Language

Diamond provides hardware description language (HDL) visualization and rule-checks to detect coding style violations that may lead to pre-/post-synthesis simulation mismatches.

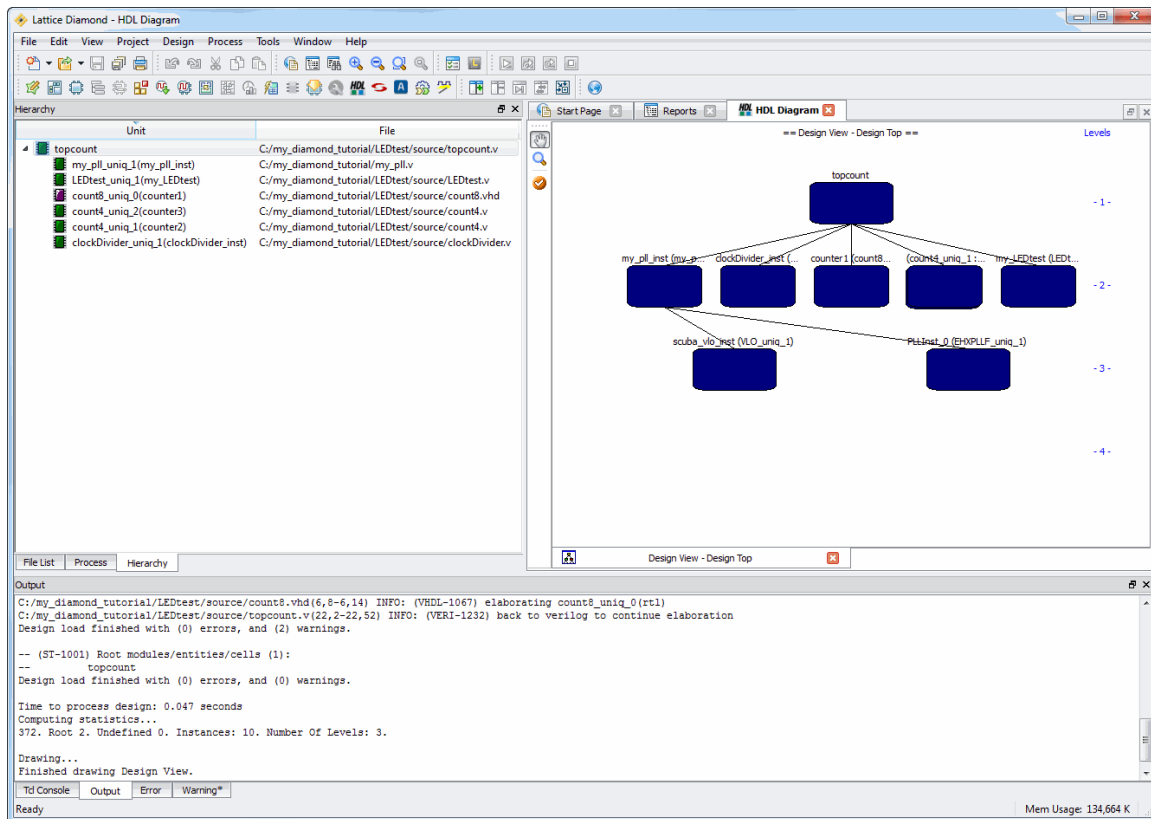
The Hierarchy view mirrors the portion of the design hierarchy that is displayed in the active graphical views. The Hierarchy view opens automatically when you open a project.

To analyze and view the HDL design:

1. Click the Hierarchy tab to open the Hierarchy view. If the Hierarchy view is not open, choose **View > Show Views > Hierarchy** from the Diamond main window.
2. Choose **Tools > HDL Diagram**.

The Hierarchy view appears as shown in Figure 10.

Figure 10: Hierarchy View



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

The HDL Diagram with the results of the BKM analysis appears, as shown in Figure 11. Results shown in output log and HDL diagram and Hierarchy view are color coded and based on results from the BKM analysis.

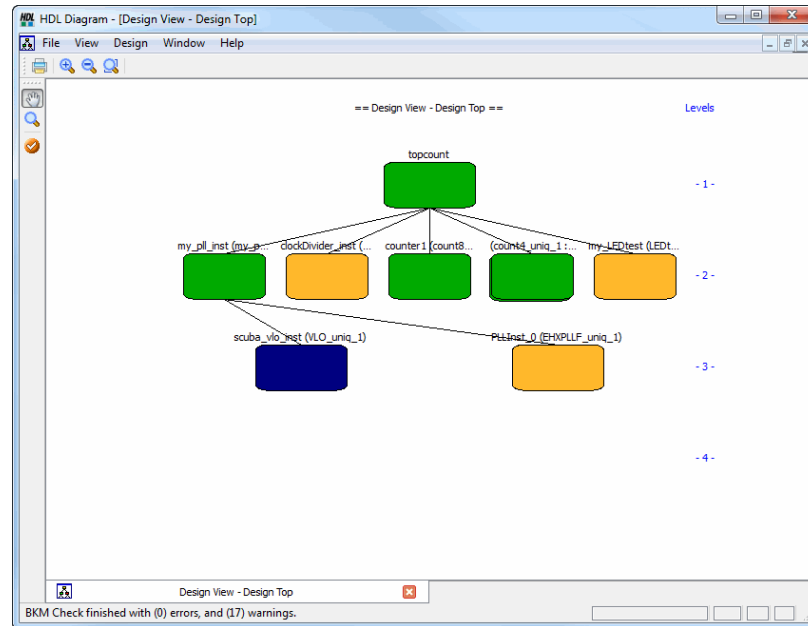
Best Known Methods (BKM) are design guidelines that HDL Diagram uses to analyze your design. BKM checks include the following:

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

The checks performed during a BKM run can be customized in the **Options** dialog box (**Tools > Options** from the Diamond main window) HDL Diagram section.

It is a good practice to run RTL analysis before synthesis to detect coding style that could lead to mismatches between pre-synthesis and post-synthesis simulation results. The analysis views are also excellent documentation output for your design.

Figure 11: HDL Diagram



4. After running the BKM check, you may encounter warning and error messages. Error and warning messages are displayed in the Output, Warning, and Error frames. If you double-click the message in the Output, Error, or Warning frames, the source indicated in the message will be opened in the associated editor. This cross-probing function can ease your check of the source file.


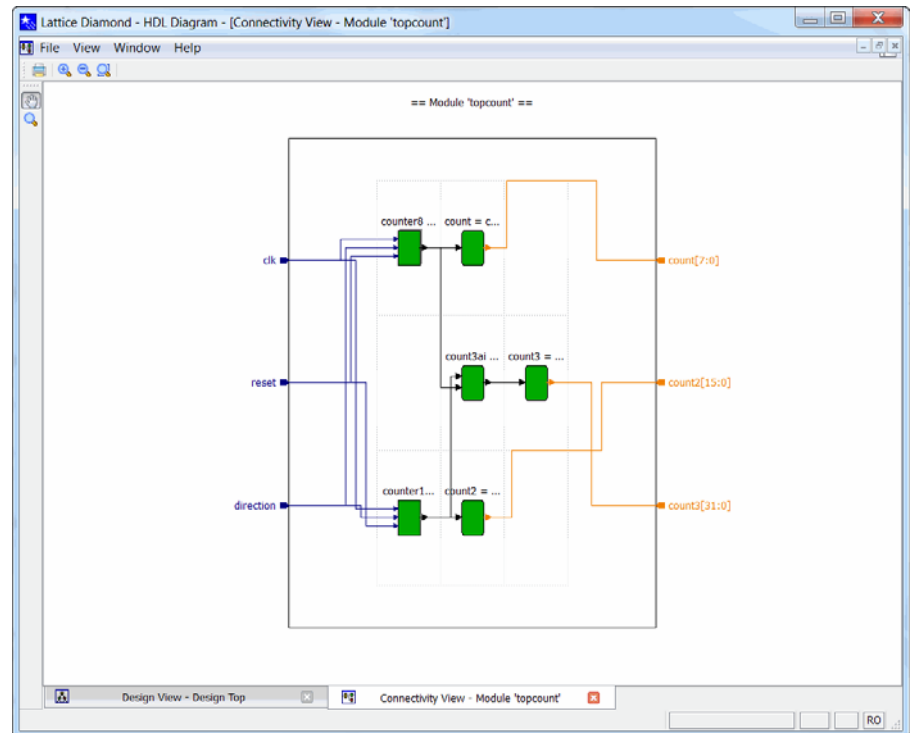
Double-click the topcount block from the HDL Diagram Design view, or right-click the topcount block and choose View Connectivity. The Connectivity view, shown in Figure 12, appears as a new tab. You can click the Detach Tool icon  on the upper right corner the HDL Diagram to make it a separate window. The Connectivity view shows signal flow between module ports, internal instances and the behavioral blocks within a particular instance or module. It enables you to explore the signal connectivity — signals and bundles between instances and behavioral blocks within the current module.

Figure 12: Connectivity View

5. When you finish checking the signal connectivity, you can choose **Window > Attach Window** from the separated HDL Diagram to attach it back to the Diamond main window.
6. After running the design analysis tool, you can see that the top-level source **topcount** is bold-faced in the File List view.

Task 4: Verify Functionality with Simulation

Diamond provides an interface to create a new simulation project file that can be imported into a standalone simulator. Diamond supports Active-HDL and ModelSim® simulation file for file exports.

Aldec® Active-HDL™ is an integrated environment designed for simulation of VHDL, Verilog/SystemVerilog, EDIF, and SystemC designs.

In this task, you will simulate the design using Active-HDL and analyze the resulting waveforms.

To simulate the design:

Make sure all the Source files are included in the simulation. In the File List pane, under **Verilog_VHDL > Input Files**, right-click on all of the source files and select **Include For > Synthesis and Simulation**.


1. Select **Tool > Simulation Wizard**.

The Simulation Wizard dialog box appears.

2. Click **Next**.

The Simulator Project Name dialog box appears.

3. Perform the following:


- ▶ Specify Project name: **simulationfile**.
- ▶ Make sure Active-HDL is selected for Simulator.
- ▶ Click the  button to browse to where you want to store the project's file.

4. Click **Next**.

The Process Stage dialog box appear.

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

The Add and Reorder Source dialog box appears.

6. Make sure all source files are present in the Source Files list. If the file my_pll.v that you created in "Task 2: Create an IPexpress Module" on page 10 is not present in the Source files list, click the Add File button , browse to your project directory, and choose the file **my_pll.v**.

7. Click **Next**.

The Parse HDL Files for Simulation dialog box appears.

8. Click **Next**.

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

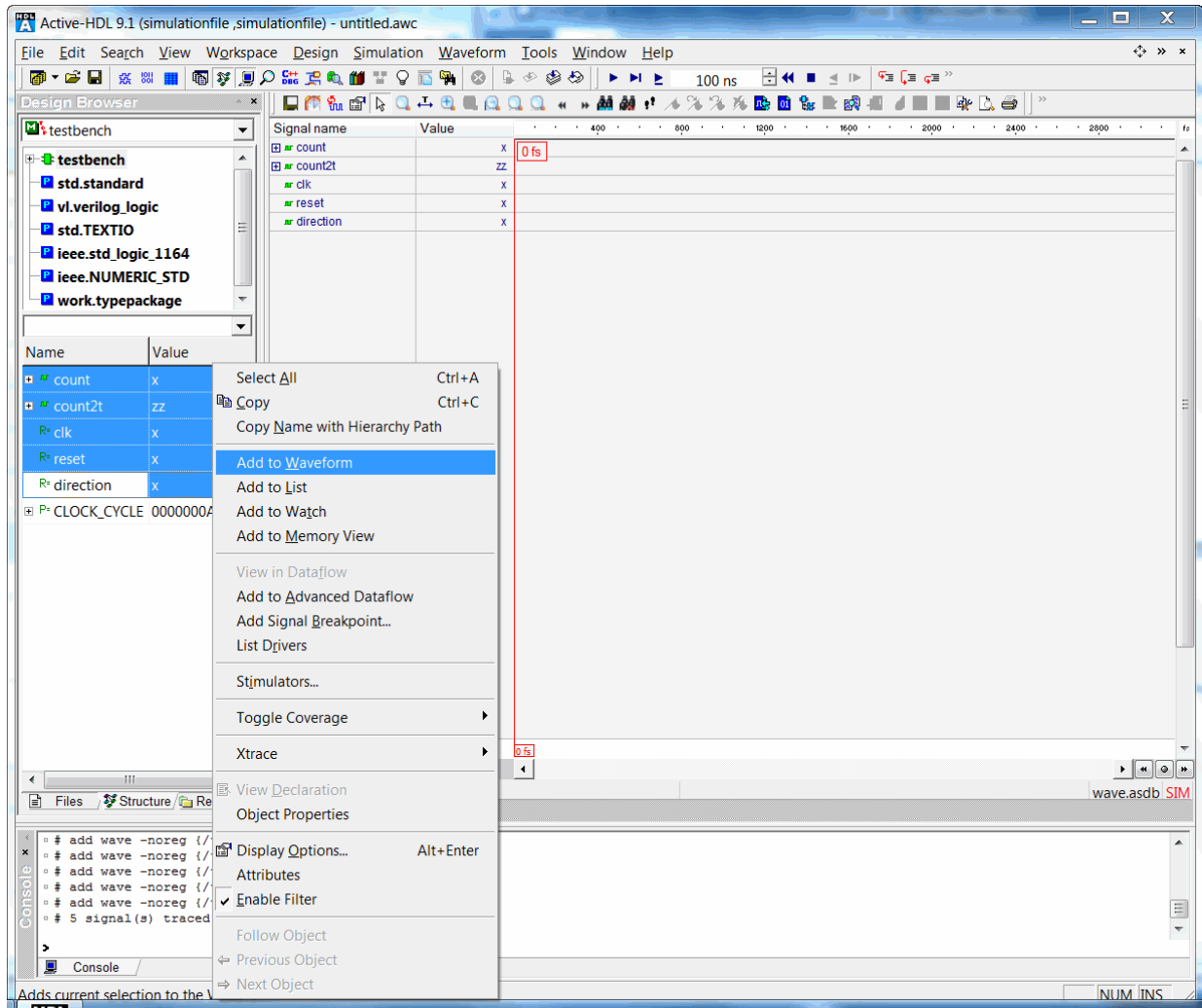
9. Click **Finish**.

The Aldec Active-HDL software is launched and the simulation starts automatically.

To simulate the design manually:

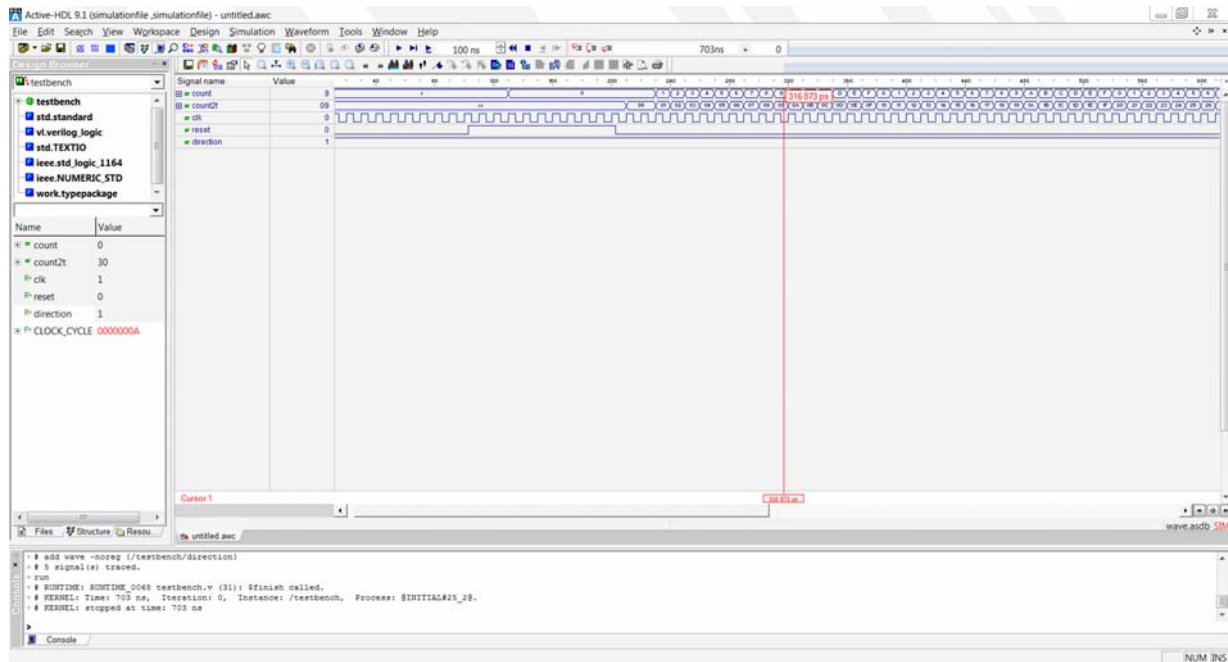
1. In the Active-HDL Design Browser, select **testbench** from the drop down menu.
2. Select the **Structure** tab in the Design browser.
3. Initialize the simulation by choosing **Simulation > Initialize Simulation**.
4. In the **Structure** tab, select all the following signals:
 - ▶ count
 - ▶ count2t
 - ▶ clk
 - ▶ reset
 - ▶ direction
5. Right-click, and in the popup menu, chose **Add to Waveform**, as shown in Figure 13.

Figure 13: Active-HDL 9.1 Window



6. Compile the simulation file by choosing **Design > Compile All**.
7. Start the simulation by choosing **Simulation > Run**. After completing the simulation, the waveform appears, as shown in Figure 14.

Figure 14: Simulated Waveform

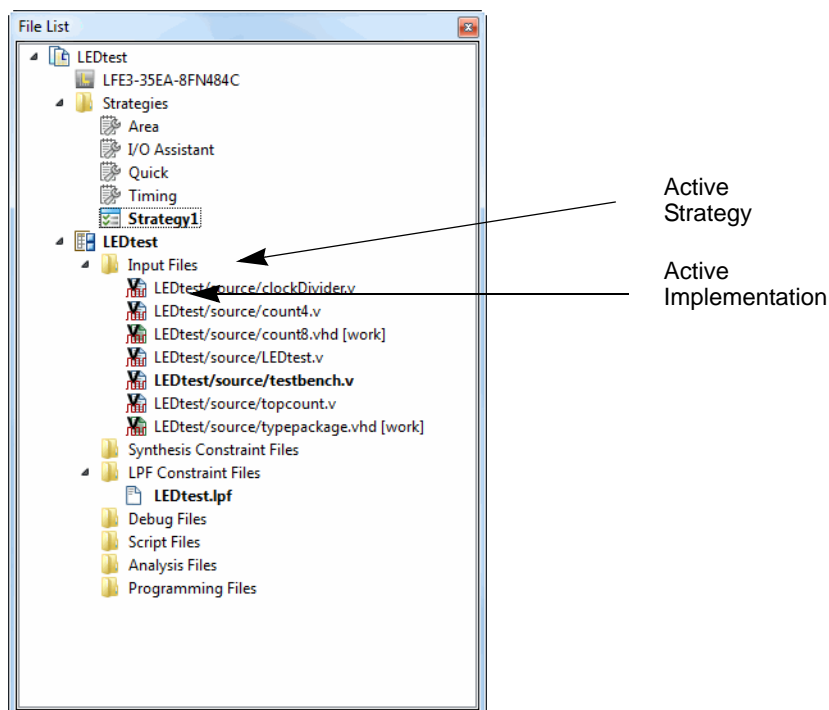


8. Close Active-HDL, and click **NO** in the Save File dialog box.

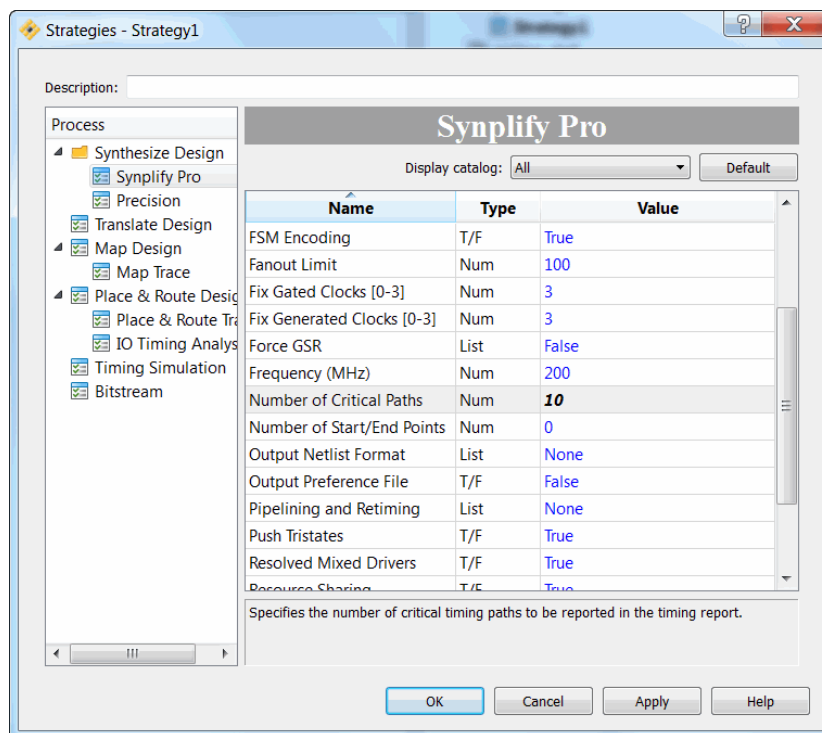
Task 5: Inspect Strategy Settings

Implementations define the design structural elements for a project including source code, constraint files, and debug insertion. Implementation contains all source files, constraint files, debug files, scripts, and analysis files. Source can mix VHDL, Verilog, & EDIF. Files can be referenced or included in the implementation. Referenced files can be shared between implementations.

A strategy is a collection of settings for controlling the different stages of the implementation process (synthesis, map, place & route, and so on). Strategies can control whether the design is optimized for area or speed, how long place and route takes, and many other factors. Diamond provides a default strategy, which may be a good collection to start with, and some variations that you can try. You can modify Strategy1, as shown in Figure 15, and create other strategies to experiment with or to use in different circumstances. Predefined strategies can also be cloned and then modified.

Figure 15: Implementations and Strategies**To adjust synthesis settings:**

- From the File List view, double-click **Strategy1**.
The Strategies - Strategy1 dialog box, shown in Figure 16, appears.
- Browse to **Synthesize Design > Synplify Pro**.
A set of default global synthesis timing constraints and optimization settings appear in the panel. Synplicity Synplify Pro® settings are displayed as the default in the dialog box.
For information on FPGA Design Constraints File (.fdc) usage in Synplify, see the *Synplify and Synplify Pro for Lattice Reference Manual* in the Synplify Pro for Lattice installation directory.
- Specify the following setting for Synplify Pro, as shown in Figure 16:
Number of Critical Paths: **10**

Figure 16: Strategies -- Strategy 1**Note**

When each strategy is selected, descriptive text appears in the lower panel of the dialog box. Default values in the strategies dialog box are shown in blue while changed values are shown in black.

3. Choose **OK**. Global synthesis options are now set for the design.


Task 6: Examine Resources

Diamond provides visualization tools to help you understand and document the physical resources of the target device and the utilization of resources. You can browse and locate device features independent of the project's source files. After synthesis, you can view the calculated resource utilization.


To browse device resources:

1. Choose **Tools > Device View**.

The Device view appears.

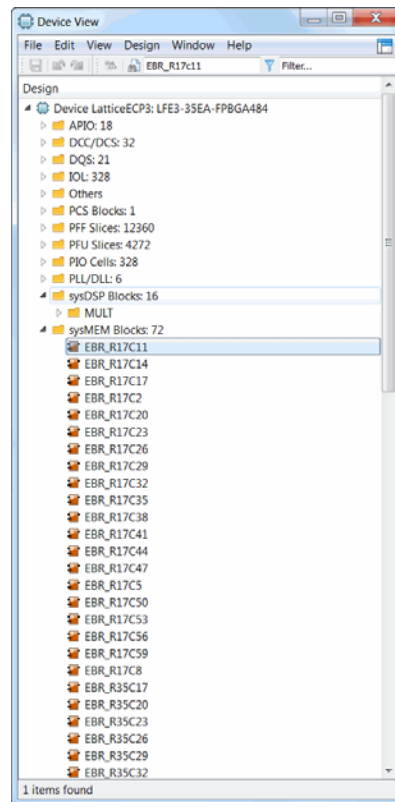
2. Click the Detach Tool icon  on the upper right corner the Device view to make it a separate window.

An index of the physical resources of the target device appear.

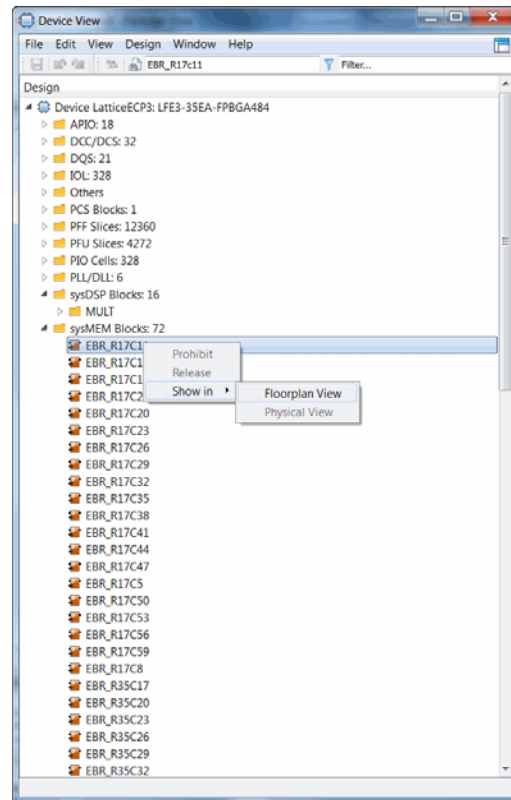
3. Click the  icon to expand the Device folder. Several folders organized by feature type appear.
4. Expand the sysDSP Blocks and sysMEM Blocks folders.
5. Type **EBR_R17C11** (Embedded Block RAM Row 17, Column 11) into the Find entry box at the top of the Device View, then press Enter. The EBR design symbol is highlighted, as shown in Figure 17.

LatticeECP3 devices contain one or more rows of sysMEM EBR blocks. EBRs are large, dedicated 18 kbit speed memory blocks. Each sysMEM block can be configured in a variety of depths and widths as RAM and ROM functions.

Figure 17: Device View

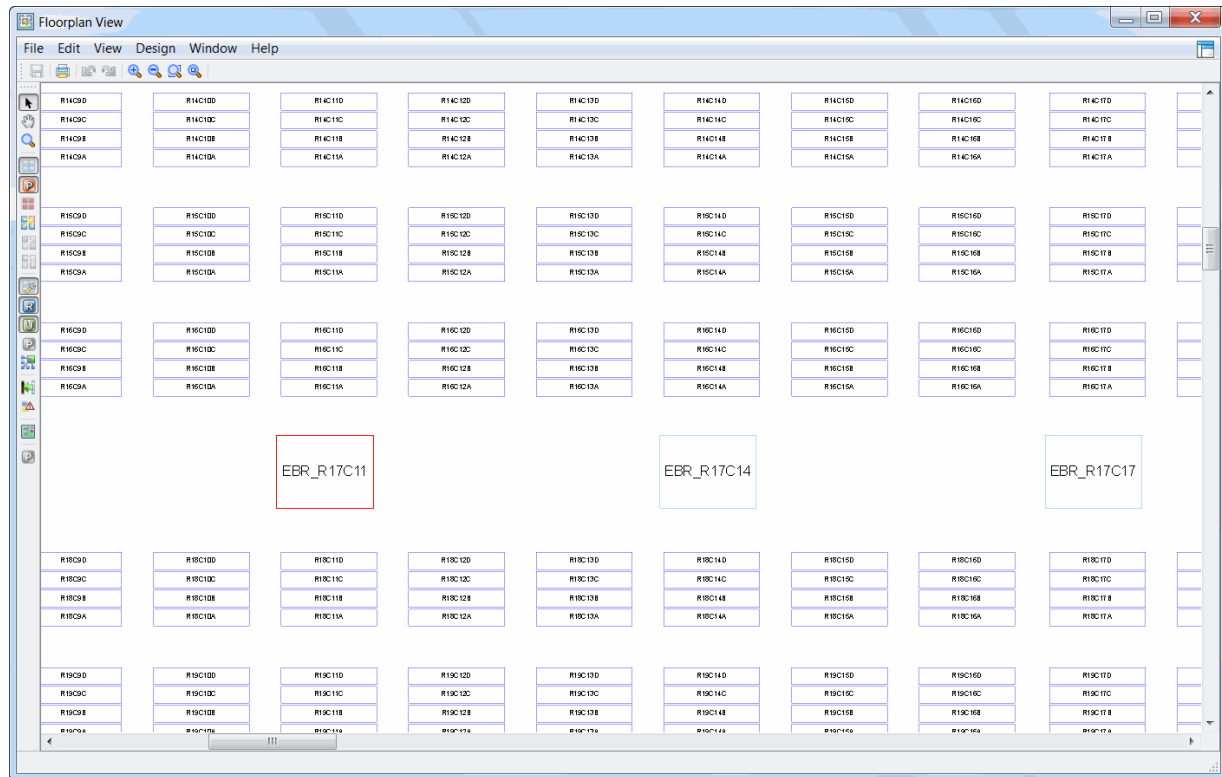


6. Right-click EBR_R17C11 and choose **Show in > Floorplan View**, as shown in Figure 18.

Figure 18: Device View with Cross Probing

Floorplan View, shown in Figure 19, provides a large-component layout of your design. It displays user constraints from the logical preference file (.lpf) and placement and routing information.

Figure 19: Floorplan View



7. Close the Floorplan View and the Device view.

Task 7: Run Synthesis Process

Synthesis is the process of translating a register-transfer-level design into a process-specific, gate-level netlist that is optimized for Lattice Semiconductor FPGAs. Diamond can be used with almost any synthesis tool. Diamond comes with two tools fully integrated: Synopsys Synplify Pro for Lattice and Lattice Synthesis Engine (LSE). “Fully integrated” means that you can set options and run synthesis entirely from within Diamond.

You will be using Synopsys Synplify Pro for Lattice to synthesize your design for the LatticeECP3 FPGA. If you are designing for MachXO, MachXO2, or Platform Manager, you have the option of using Lattice Synthesis Engine or another third-party synthesis tool as your synthesis tool instead of Synplify Pro for Lattice. To change the synthesis tool, from the Diamond main window, choose **Project > Synthesis Tool**.

To synthesize the design and examine resource utilization:

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

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

- When the synthesis process is complete, select **View > Show Views > Post Synthesis Resources**. Select the **Hierarchy – Post Synthesis Resources** tab, shown in Figure 20.

Figure 20: Post Synthesis Resources

Unit	File	LUT4	PFU Registers	
topcount	C:/Iscc/diamond/2.1_x64/docs/tutorial/Diamond_tutorial/LED_test/source/topcount.v	73(8)	102(1)	27(0)
my_pll_uniq_2(my_pll_inst)	C:/Iscc/diamond/2.1_x64/docs/tutorial/Diamond_tutorial/my_pll.v	1(1)	0(0)	0(0)
LEDtest_uniq_2(my_LEDtest)	C:/Iscc/diamond/2.1_x64/docs/tutorial/Diamond_tutorial/LED_test/source/LEDtest.v	15(15)	0(0)	0(0)
count8_uniq_1(counter1)	C:/Iscc/diamond/2.1_x64/docs/tutorial/Diamond_tutorial/LED_test/source/count8.vhd	1(1)	16(16)	0(0)
count4_uniq_4(counter3)	C:/Iscc/diamond/2.1_x64/docs/tutorial/Diamond_tutorial/LED_test/source/count4.v	6(6)	12(12)	0(0)
count4_uniq_3(counter2)	C:/Iscc/diamond/2.1_x64/docs/tutorial/Diamond_tutorial/LED_test/source/count4.v	5(5)	8(8)	0(0)
clockDivider_uniq_2(clockDivider_inst)	C:/Iscc/diamond/2.1_x64/docs/tutorial/Diamond_tutorial/LED_test/source/clockDivider.v	37(37)	65(65)	0(0)

The Post-Synthesis Hierarchy View displays the number of logical resources within each level of the design.

In the Hierarchy table shown in Figure 20, topcount is the top module displaying the resource utilization.

- ▶ LUT4 73(8) – 73 represents the total LUT4 count utilization throughout the design and 8 represents the LUT4 utilized only in the design module topcount,
- ▶ PFU Registers 102(1) – 102 represents the total PFU register utilization throughout the design and 1 represents the PFU registers utilized only in the design module topcount. Similar utilization is shown for the I/O registers, carry cells and SLICES.
- ▶ my_pll_uniq_2, LEDtest_uniq_2, count8_uniq_1, count4_uniq_4, count4_uniq_3 and clockDivider_uniq_2 are the sub-modules (instances) of the design. For example, the sub-module count4_uniq_3, LUT4 5(5) represents the total LUT4 count in the sub-module count4_uniq_3.
- ▶ PFU Registers 8(8) represents the total PFU Registers in the sub-module count4_uniq_3.
- ▶ SLICE 5(5) represents the total number of SLICES in the sub-module count4_uniq_3.

Hence, the total number of logic resources (adding the resources from the individual module) is reflected in the top level module topcount.

Task 8: Set Timing and Location Assignments

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

To set timing and location assignments:

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

The batch interface to logic synthesis, EDIF translation, and the design mapper run. Report files appear in the Reports view. To view each process report, select the process in the **Design Summary** pane.

Each major stage of an FPGA implementation is illustrated as a milestone in the Process view: Synthesize Design, Translate Design, Map Design, Place&Route Design, and Export Files. The status of any stage is represented by the following color-coded icons:

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

2. From the **Design Summary** pane of the Reports view, select **Process Reports > Map**.

The Map Report, as shown in Figure 21, appears in the right panel.

3. Right click in the right pane of the Reports view, choose **Find in Text**. Type in **Design Summary**.

The report highlights the Design Summary section of the report.





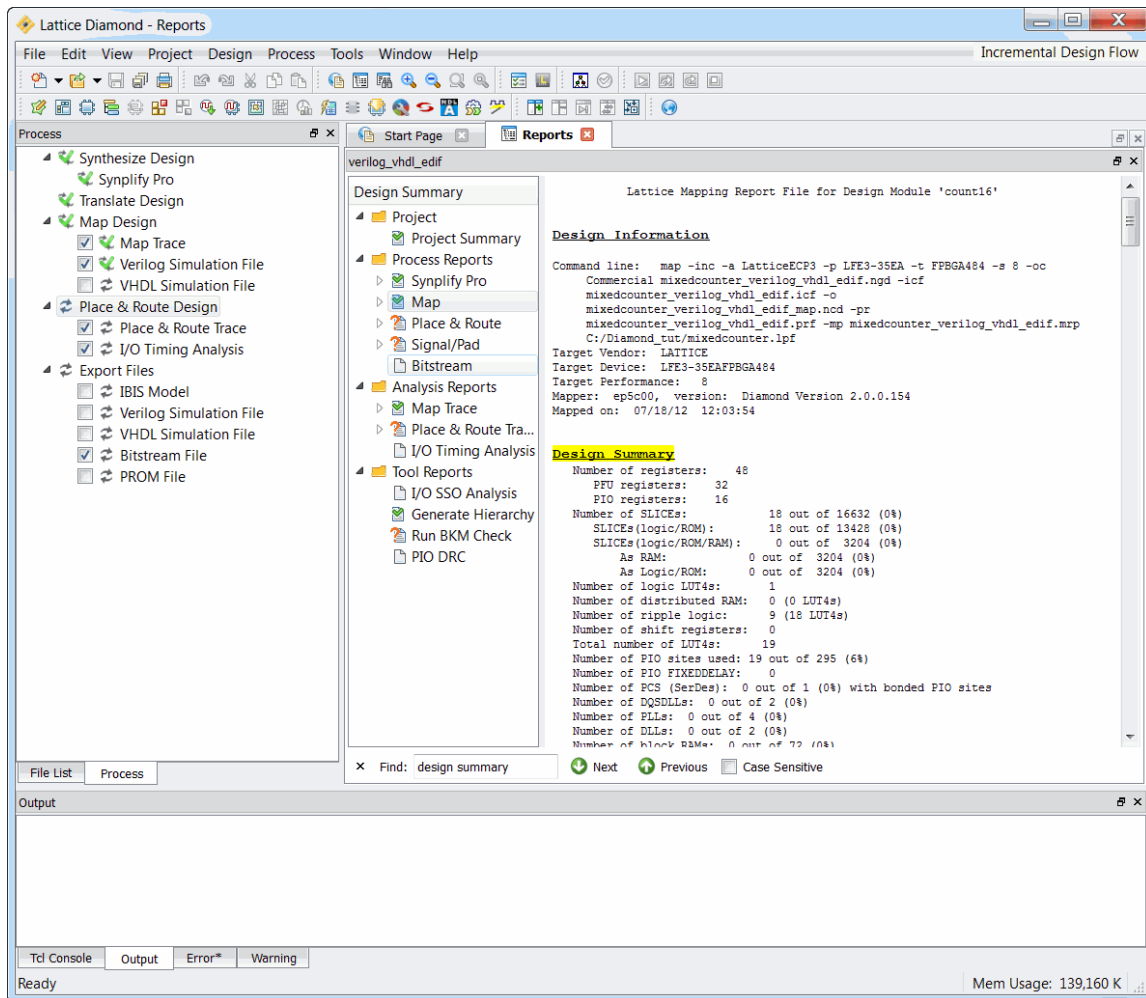

In the Design Summary pane, there is the report icon . If a report has been generated, the icon appears as . If the report is not the most recent version, the icon appears as . To view the contents of the entire report, click on the report to be viewed. The entire report is then displayed in the right pane of the Reports view. Use the scroll bar to navigate through the report. Some of the reports are divided into sections (for example, Map, Place & Route, and Signal/Pad). Click the plus  sign before the report to display the sections in a list. Choose the desired section. The whole report will be displayed with the selected section displayed at the top of the right pane of the Reports view.

Figure 21: MAP Report



4. Choose **Tools > Spreadsheet View**, or click on .

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

5. Click the **Detach Tool** icon  at the upper right corner of the Spreadsheet View.

The Spreadsheet View is detached from the Diamond main window.

6. Click the **Period/Frequency** icon  on the Spreadsheet View tool bar.
The Period/Frequency Preference dialog box appears.

7. Enter the following preference settings:

Type:	FREQUENCY
Second Type:	Clock Net
Available Clock Nets:	clk_c
Frequency:	100MHz

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

8. Click the **Input_setup/Clock_to_out** button  on the Spreadsheet View toolbar.

9. The INPUT_SETUP/CLOCK_TO_OUT Preference dialog box appears.

10. Enter the following preference settings:

Type: **INPUT_SETUP**

Second Type: **All Ports**

Clock Ports/Nets: **clk**

Time: **10ns**

Click **OK**.

The Timing Preferences tab of the Spreadsheet View appears with the new **INPUT_SETUP** preference defined. So, you can define preferences in the relevant preference dialog box.

11. From the Timing Preference tab, right-click the INPUT_SETUP entry, and select **New INPUT_SETUP**.

The INPUT_SETUP/CLOCK_TO_OUTPUT Preference dialog box appears.

12. Enter the following settings:

Type: **INPUT_SETUP**

Second Type: **Individual Ports**

Available Input Ports: **reset**

Clock Ports/Nets; **clk**

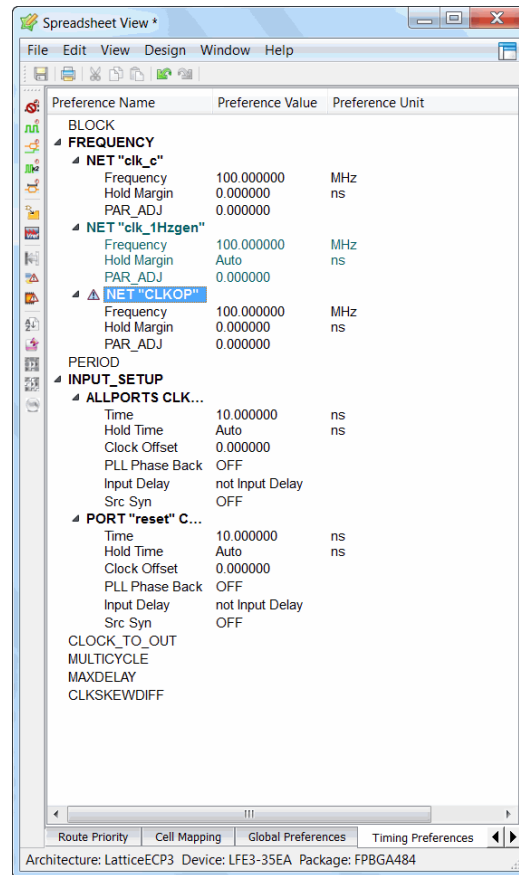
Time: **10ns**

Click **OK**.

The Timing Preference tab of the Spreadsheet View appears, as shown in Figure 22, with the new INPUT_SETUP preference defined.

The preference dialog box can be invoked from the toolbar icon, the menu item (**Edit > Preferences** from the Spreadsheet View), or from the right-click menu of the Spreadsheet View. You can also double click on a value in Timing Preferences tab and edit the value directly.

Figure 22: Spreadsheet View



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

14. Right click the IO_Type cell of the All Ports row.

A pull-down menu of signal standards appears. Select **LVC MOS33**, if it is not already selected. The port attributes display is updated with the new IO_TYPE. Cell entries in the Spread Sheet view are color-coded to indicate the source of a preference setting:

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

15. Choose **File > Save LEDtest.lpf** from the detached Spreadsheet View.

The project Logical Preference File (.lpf) is updated. Close the Spreadsheet View.

16. From the File List view of the Diamond main window, LPF Constraints Files folder, double-click the LEDtest.lpf file.

The Source Editor appears with the ASCII LPF file. Note the timing and location preferences defined so far. Close the Source Editor.

Task 9: Running Place and Route

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

To run place and route:

1. From the Process List double-click **Place & Route Design**.

The place and route tools are run. Intermediate results appear in the Output frame of the Diamond main window.

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

Details about Place & Route appear in the pane to the right.

3. From the Process List double-click **Place & Route Trace**.

The TRACE timing analyzer is run.

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

5. From the Process List double-click **I/O Timing Analysis**.

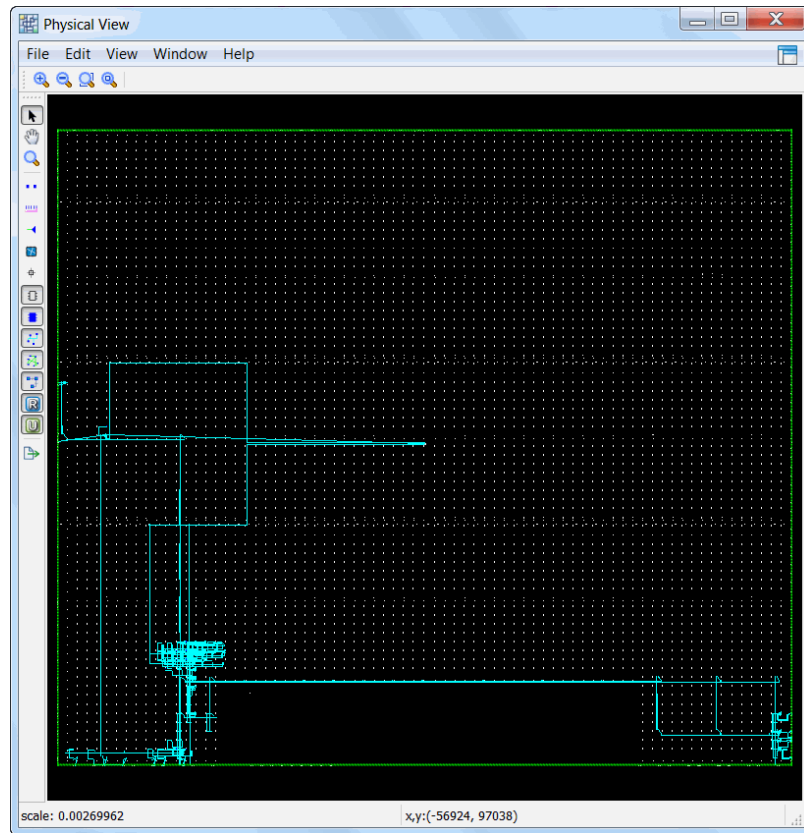
The timing analysis is run.

6. From the **Design Summary** pane of the Reports view, select the **I/O Timing Analysis** section of Analysis Reports.

The I/O Timing Report appears in the right pane of the Reports view.

7. Choose **Tools > Physical View**.

The Physical View appears, shown in Figure 23,. Physical View provides a read-only detailed layout of your design that includes switch boxes and physical wire connections.

Figure 23: Physical View


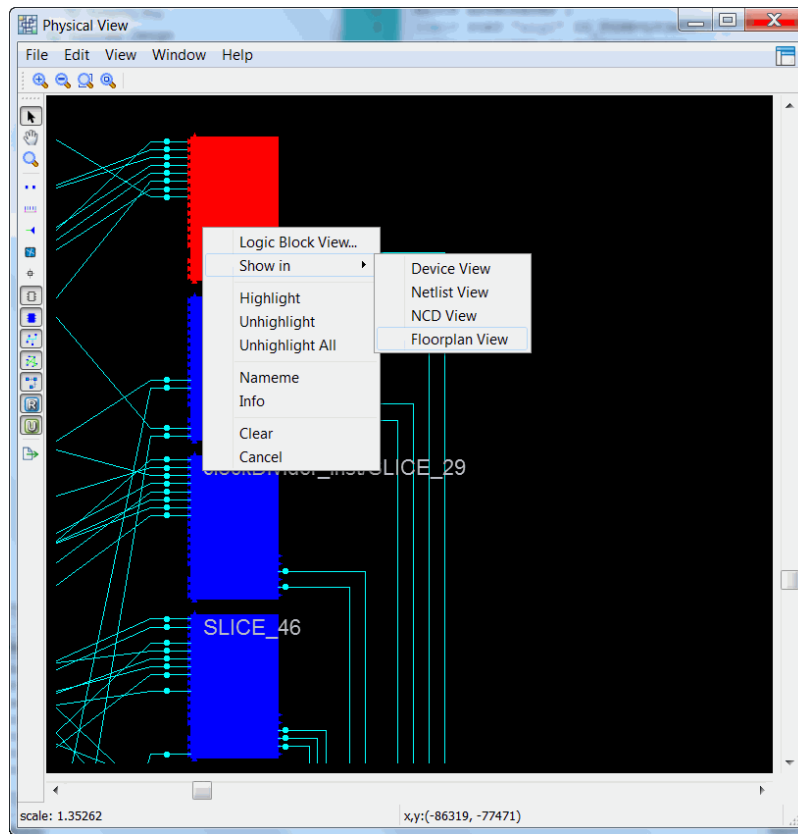
8. Use the  button to zoom into a component. Right click on the component and choose **Show in > Floorplan View**, as shown in Figure 24, to display the Floorplan View.

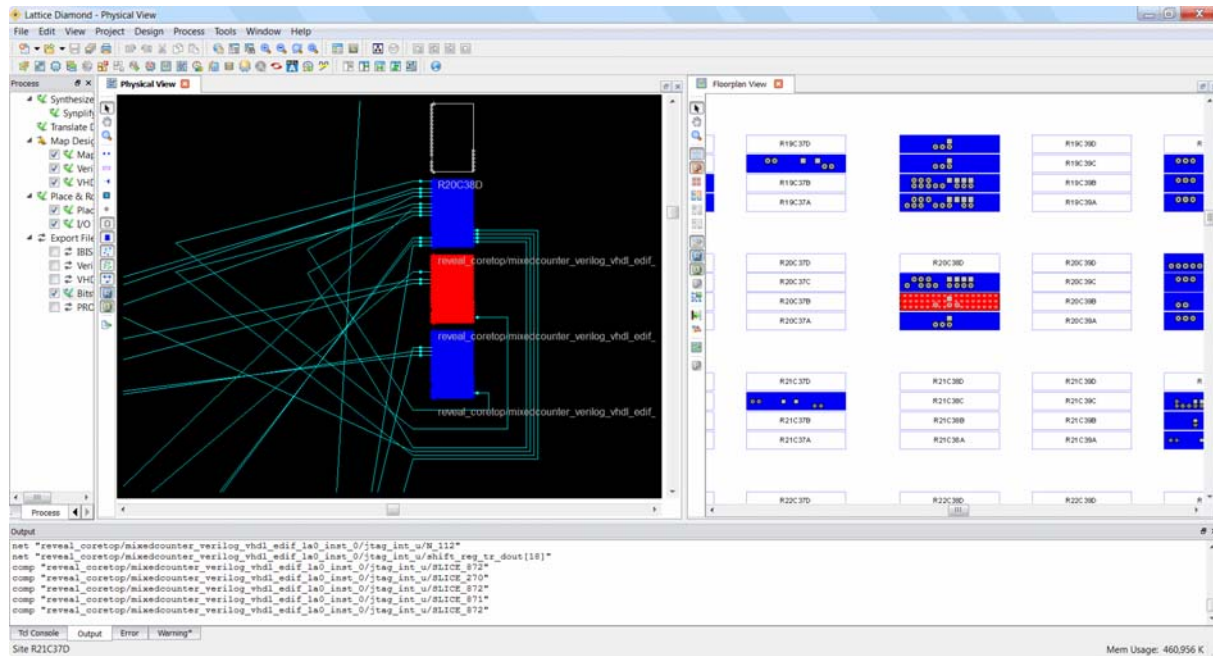
Figure 24: Physical View

9. To auto cross-probe between Floorplan and Physical Views, right-click on the Floorplan View tab and select **Split Tab Group**.

The two views display in parallel, as shown in Figure 25.

When both Floorplan View and Physical View are open, an item that you select in one of these views is automatically selected in the other. Auto cross-probing is especially useful for immediately examining connections in both views.

Figure 25: Cross Probing



10. Right click on the Floorplan View tab and choose **Move to Another Tab Group**.

Now both tabs are merged into a single group as before.

11. Close Floorplan View and Physical View tabs.

Task 10: Examine Post Place and Route Results

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

In this task, you will view the results of the Static timing analysis and then use the Timing Analysis view to enter clock jitter values.

Examine timing analysis results:

1. Choose **Tools > Timing Analysis View**, or click  in the Diamond toolbar

The Timing Analysis view appears.

2. Click the **Detach Tool** icon from the right corner of the Timing Analysis view.

The Timing Analysis view is detached from the Diamond main window.

A summary of the post-route static timing analysis settings such as target device information, preference file, performance grade, and environment conditions appear in the upper left pane. The lower left pane provides an

index of the available analysis results. Related timing preferences appear in each analysis section.

- From the Analysis pane (on the lower left of the Timing Analysis view), select **INPUT_SETUP ALLPORTS 10ns CLKPORT "clk" setup**.

The Path Table is populated in the upper right of the Timing Analysis view, with the Source, Destination, Weighted Slack, Arrival, Required, Data Delay, Route%, Levels, and other details.

- Select Row 1 of the Path Table.

The Detailed Path Tables in the lower pane are populated with details.

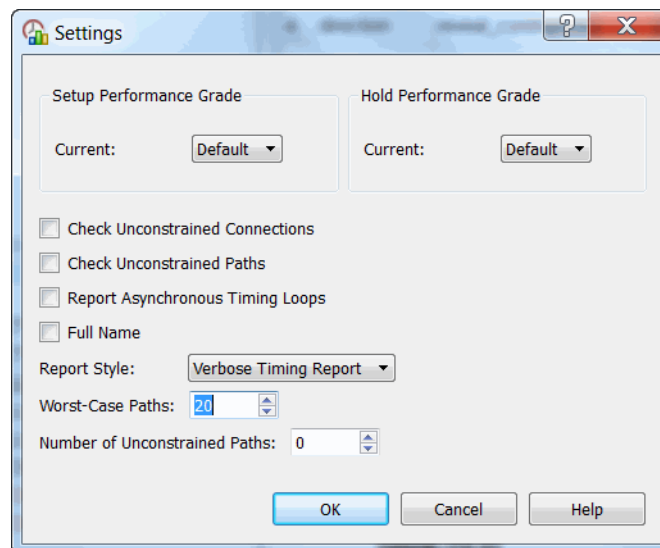
- Choose **Edit > Settings**, or click  from the toolbar in the Timing Analysis view.

The Settings dialog box appears, as shown in Figure 26.

- Enter **20** into the Worst-Case Paths field and click **OK**.

The Timing Analysis view is refreshed with the additional path data.

Figure 26: Timing Setting Dialog Box



- You can use the Source Filter field of the Path Table to filter out all the wanted paths. Delete the text from the Source Filter field, all the sources appear in the Source list again.

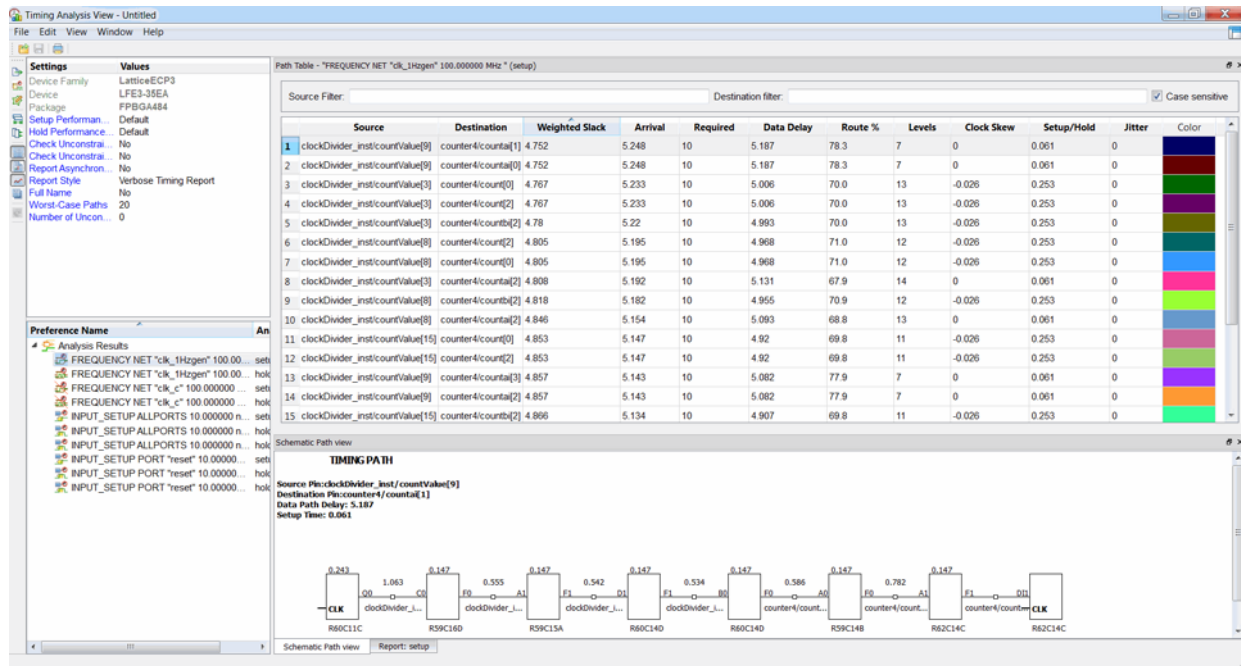
- Select the first row in the **Path Table**.

The Detailed Path Tables are updated.

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

Each component of the data path delay is identified alternating between route delays and combinatorial or clock-to-output type delays, as shown in Figure 27.

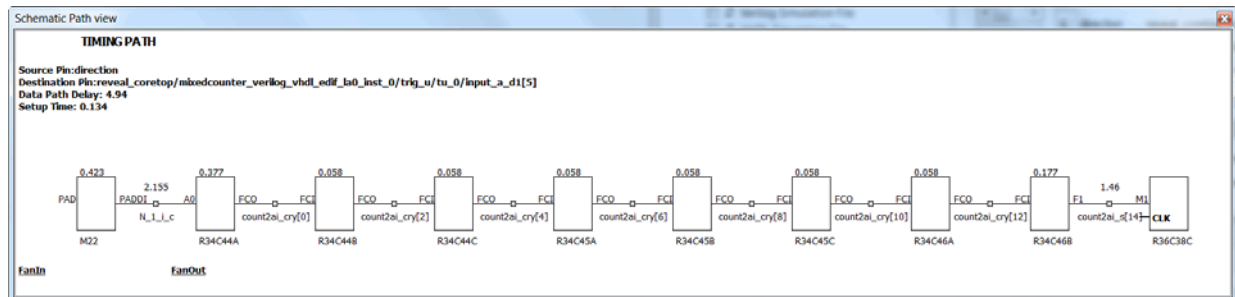
Figure 27: Timing Analysis



10. Select the Schematic Path view.

A schematic graphic of the data path timing path appears as shown in Figure 28.

Figure 28: Schematic Path View



Diamond allows user to enter peak-to-peak clock jitter on an input CLOCK PORT. The jitter is propagated through to various design modules that use this clock. The trace will use half of p-p jitter in the direction that will cause the total timing slack to reduce.

11. To enter clock jitter values, select the **Change Timing Preferences** button in the Timing Analysis view.

The TPF Spreadsheet View appears.

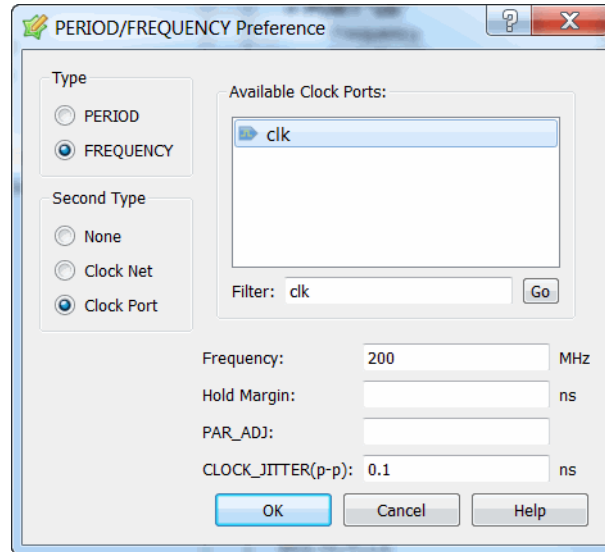
12. Select the Period/Frequency  button in the TPF Spreadsheet View.


The Period/Frequency Preference dialog box appears, shown in Figure 29.

13. Enter the following preference settings:

Type: **Frequency**
 Second Type: **Clock Port**
 Available Clock Ports: **clk**
 Frequency: **200MHz**
 CLOCK_JITTER(P-P): **0.1ns**
 Click **OK**.

Figure 29: Period/Frequency Preference



14. In Timing Analysis view, click on the update  button.

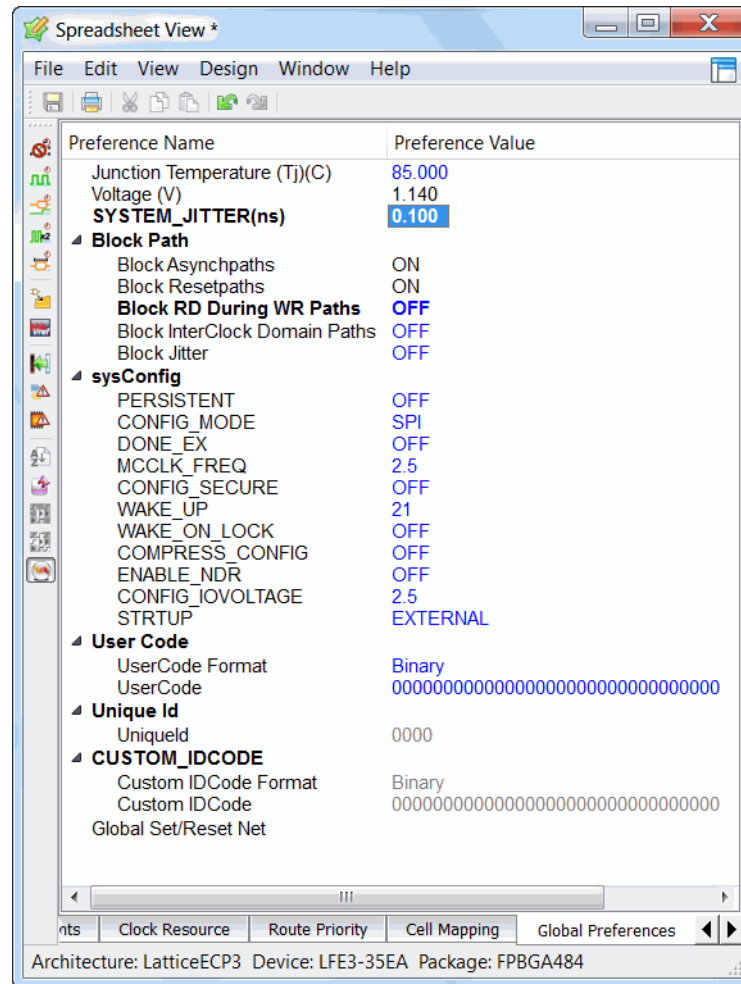
The resulting clock jitter values can be viewed in the path table details tab on the right.

15. Close the Timing Analysis view.

Diamond allows the user to specify the peak-to-peak system jitter for timing analysis. When not used, a default value of 0 is used for all analysis. System Jitter affects all clocks in the design.

16. To enter System Jitter Values, choose **Tools > Spreadsheet View**. Select the **Global Preferences** tab.
17. In the Global Preferences tab, double-click on the Preference Value column of the **SYSTEM_JITTER(ns)** row, shown in Figure 30. Enter **0.1** into the dialog box.

Figure 30: System Jitter



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

Task 11: Adjust Static Timing Constraints and Review Results

In this task, you will edit timing constraints for STA (Static Timing Analysis) using the Timing Preference File (TPF) version of Spreadsheet View, and then you will use Timing Analysis view to review the results.

Timing analysis within Lattice Diamond can be performed at three points in a typical design flow: post-synthesis, post-map when the post-synthesis netlist of the design has been translated to the target device, post-placement, and post-route. Each stage provides a progressively more accurate report of delay characteristics. Timing analysis at the synthesis stage is performed by the respective synthesis tool: Synplify Pro or Precision. Additional features are provided by Diamond for post-map stages of STA.

By default, the timing analysis engine, TRACE, uses those timing constraints applied by timing-driven map, place, and route. However, timing preferences can be modified, which allows you to manage the timing objectives of the implementation tools independent of static timing analysis. To accommodate an experimental static timing analysis loop, the TPF Spreadsheet View allows you to edit the timing preferences for use with the Timing Analysis view. This allows you to establish modified or additional timing preferences independent of the constraint set used for MPAR.

Tighten the timing objective of a preference and examine the results:

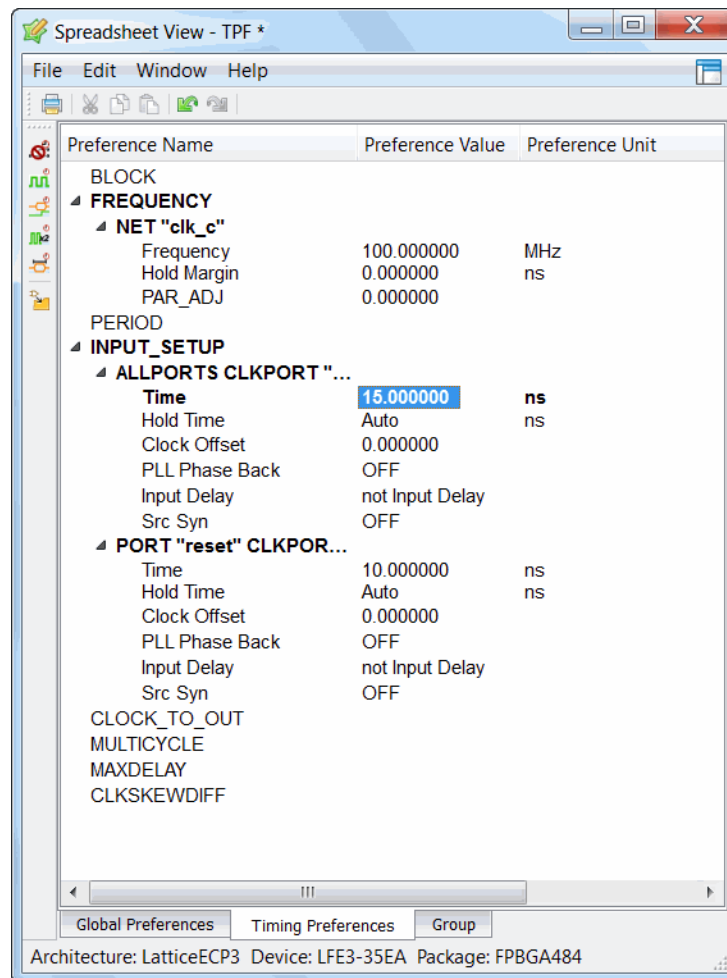
1. Choose **Tools > Timing Analysis View**, or click  in the Diamond toolbar

The Timing Analysis view appears.


2. From the Preference Name list on the lower left of Timing Analysis View, select **INPUT_SETUP ALLPORTS 10ns CLKPORT "clk" Setup**, right-click and choose **TPF Preferences**.

Spreadsheet View – TPF appears, as shown in Figure 31.

3. Select the Timing Preferences sheet of the TPF Spreadsheet View. Right-click 10.00000ns in the Preference Value column for the ALLPORTS CLKPORT "clk" and choose **Edit Value**. Enter **15ns** into the Preference Value field and press Enter.

Figure 31: Spreadsheet View

- After a few moments, return to Timing Analysis View.

The Update button  on the toolbar is now rotating.

- Click the **Update** button.

After a short while, the indicator stops rotating and the new analysis results become available in Timing Analysis View. In the title bar of Timing Analysis View, "Untitled" appears with an asterisk, which indicates an in-memory change to the timing preferences. You can save the change to a Timing Preference File (.tpf) by choosing **Save > Save Untitled As** and giving it a name and location. The .tpf file will then appear in the Analysis Files folder of the File List Pane. These .tpf files enable you to experiment with different timing settings without affecting the .lpf source file. For more information, see **Analyzing Static Timing > Using Timing Analysis View** in the Diamond online Help.


- Close Timing Analysis View and Spreadsheet View – TPF. In the Save dialog box, click **No** to discard the change.

Task 12: Comparing Multiple Place and Route Runs

Use the Run Manager to run multiple synthesis and place and route passes, compare the timing score results, and load the native circuit description (NCD) database of the best run into the workspace for further analysis.

You can create multiple strategies or implementations for the design. Then compare the runs with different implementation and strategy combination. One implementation can only be bound with one active strategy.

To create a new implementation:

1. Choose **File > New > Implementation** from the Diamond main window. Or, right click on the project name icon  from the File List view and choose **Add > New Implementation**.
2. In the New Implementation dialog box, type **verilog_vhdl** in the Name text box.

By default, the directory and location will be the same name as the implementation name. You can change the directory or location to a desired one.
3. Choose **Strategy1** from the default strategy drop-down menu.
4. Click **Add Source** and choose **From Existing Implementation > LEDtest**.
5. Select `<project_directory>/docs/testbench.v` and click the **Remove Source** button.
6. Select the “**Copy source to implementation directory**” option and click **OK**.

The new implementation **verilog_vhdl** is now displayed in the File List pane.

Note

If you want to make this new implementation active, right-click **verilog_vhdl** and choose **Set as Active Implementation**. You can have multiple implementations in your project, but you can make only one implementation active in your project at one time.

Now you will compare the run results of the **LEDtest** and **verilog_vhdl** implementations.


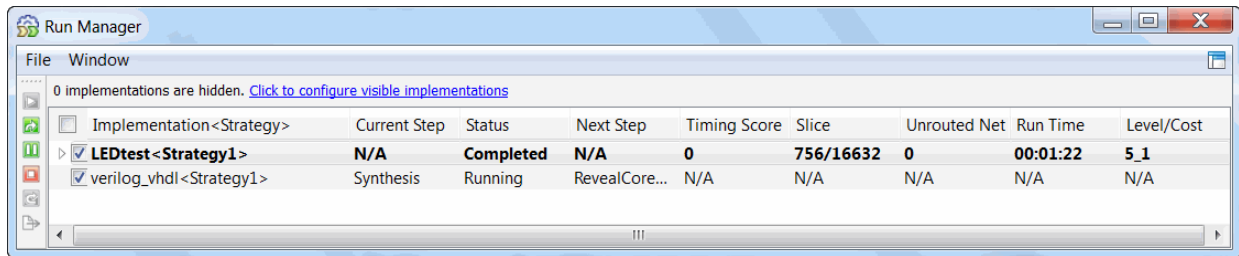
1. From the Diamond main window, choose **Tools > Run Manager**.
The Run Manager displays a table of implementation<strategy>: **LEDtest<Strategy1>** and **verilog_vhdl<Strategy1>**.
2. Enable the **LEDtest<Strategy1>** and **verilog_vhdl<Strategy1>** by setting the check boxes for each implementation, as shown in Figure 32
3. Click the **Rerun**  button on the Run Manager toolbar.
The two implementations start to run simultaneously.

Figure 32: Run Manager


Implementation<Strategy>	Current Step	Status	Next Step	Timing Score	Slice	Unrouted Net	Run Time	Level/Cost
LEDtest<Strategy1>	N/A	Completed	N/A	0	756/16632	0	00:01:22	5.1
verilog_vhdl<Strategy1>	Synthesis	Running	RevealCore...	N/A	N/A	N/A	N/A	N/A

In a few minutes, the results of the run appear in the table. Statistics such as Start time, Run time, Score, Unrouted, Level/Cost, and Description appear. The row in bold font indicates the active implementation that is loaded. The table provides a quick review of the quality of results produced by a particular strategy. To closely examine a particular run with analysis tools, such Timing Analysis View or Power Calculator, you can set the active strategy to be loaded.

If your system provides a multiple-core processor, you can set more implementations to be run concurrently. Go to the **Options** dialog box (**Tools > Options**) of the Diamond main window, **Environment > General** tab, the **Maximum number of processes in run manager** option. Enter a number in the box in front of this option. The default value is 2. The maximum allowed value is 32.


4. Choose **View > Reports**.

In Reports View, you can view results related to the run of the current active implementation. The report for **LEDtest** appears in Reports View.

Task 13: Analyze Power Consumption

Included with the Diamond software is Power Calculator, which estimates the power dissipation for a given design. Power Calculator 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.

To analyze power consumption:

1. Choose **Tools > Power Calculator** or click the  button on the toolbar. Power Calculator opens in Calculation mode.

Power Calculator provides two modes for reporting power consumption:

▶ **Estimation Mode:**

In estimation mode, Power Calculator provides estimates of power consumption based on the device resources or template that you provide. This mode enables you to estimate the power consumption for your design before the design is complete or even started

▶ **Calculation Mode:**

In calculation mode, Power Calculator calculates power consumption on the basis of device resources taken from a design's .ncd file, or from an external file such as a value change dump (.vcd) file, after placement and routing. This mode is intended for accurate calculation of power consumption, because it is based on the actual device utilization.

Editing data in cells colored white: voltages, frequency, activity factor, thermal data does not change mode. Editing data in cells colored blue (design data) will change calculation mode to estimation mode.


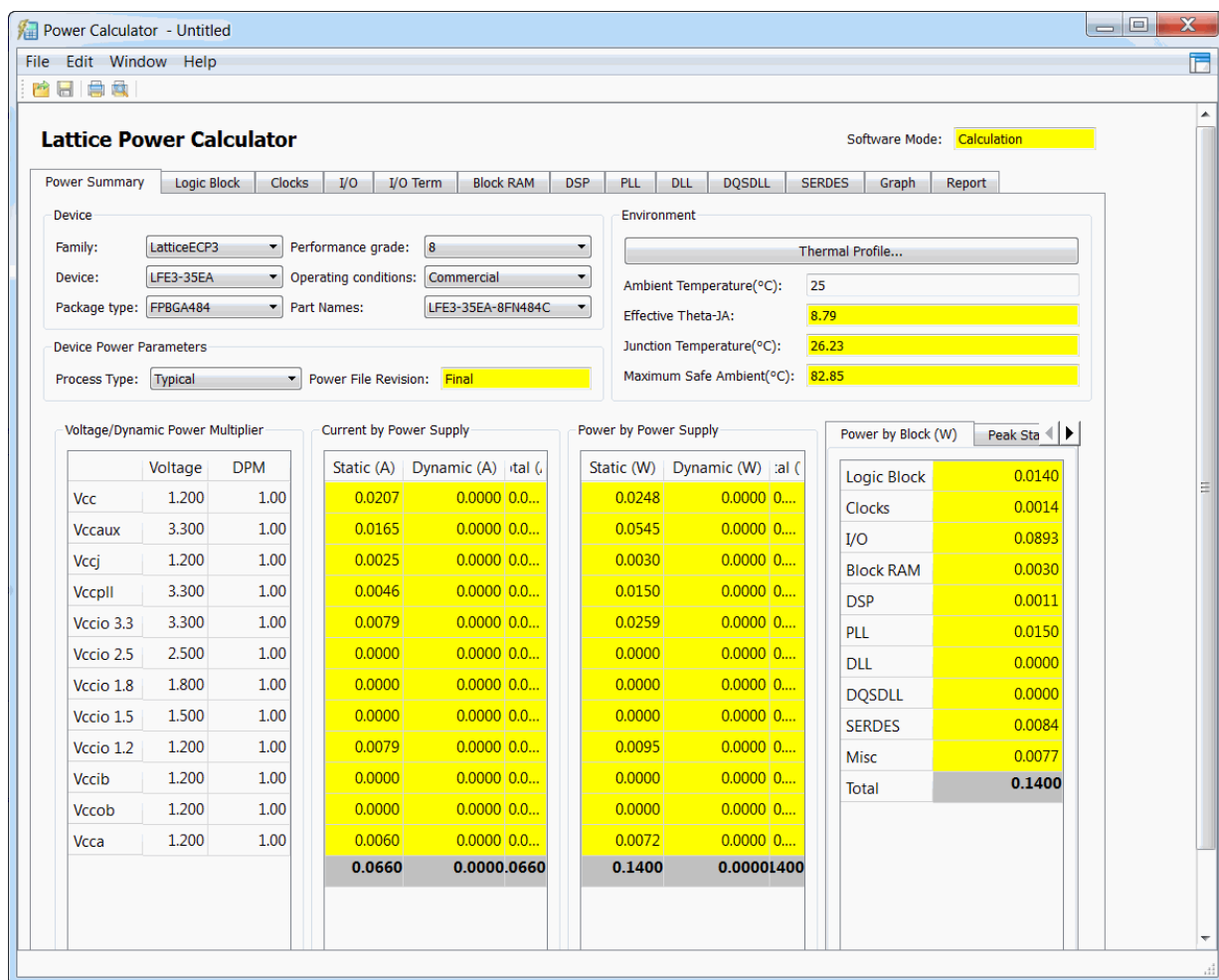
- Click the  icon the upper right corner to detach Power Calculator from the Diamond main window, as shown in Figure 33.

Figure 33: Power Calculator



- In the Device Power Parameters section select the following parameter:
Process Type: **Worst**.
- Select the **Thermal Profile** button in the Environment section.
The Thermal Profile dialog box appears.
- In the Board Selection section select the following parameter:

Board Selection: **Small board**.

6. Click **OK**.

After a short while the new power analysis results become available in the Power summary tab.

In the title bar of Power Calculator, "Untitled" appears with an asterisk, which indicates an in-memory change to the timing preferences. You can save the change to a Power Calculator File (.pcf) by choosing **Save > Save File As** and giving it a name and location. The .pcf file will then appear in the Analysis Files folder of the File List Pane. These .pcf files enable you to experiment with Power Analysis settings without affecting the .lpf source file.

7. Close Power Calculator. In the Save dialog box, click **No** to discard the change.

Task 14: Run Export Utility Programs

Use the Process view to generate files for exporting. One of the files exported will be a bitstream file (.bit) which will be used to program a LatticeECP3 device in the next task.

1. From the Process view, choose **Export Files**.

A set of export files appear under the Export Files process.

2. Select the following Export Files:

IBIS Model

VHDL Simulation File

Bitstream File

3. Click the **Run** button  on the Diamond toolbar.

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


Task 15: Download a Bitstream to an FPGA

This task requires that you have a LatticeECP3 Versa Development Kit.

In the previous section, you generated export files including a Bitstream File (.bit). In this section, you will use Diamond Programmer to download a bitstream to a LatticeECP3 FPGA mounted on a LatticeECP3 Versa Development Kit board.

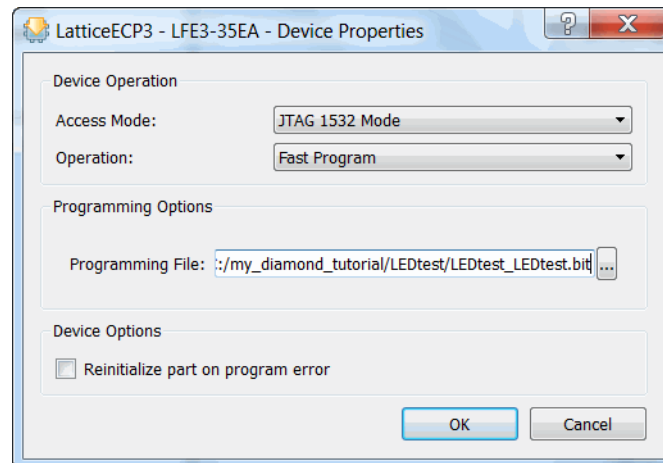
To download the bitstream to the FPGA on the board:

1. Remove any Lattice USB Programming cables from your system.
2. Connect the power supply to the development board.

3. Connect a USB cable from your computer to the LatticeECP3 Versa Development Kit board. Give the computer a few seconds to detect the USB device moving to step 4.
4. Choose **Tools > Programmer**, or click the  icon on the toolbar.
5. In the Getting Started dialog box, choose **Create a new Project from a Scan**.
 - a. In the Cable box, select **USB2**.
 - b. In the Port box, choose the only setting available in the drop-down menu, **FTUSB-0**.
 - c. Click **OK**.

Programmer scans the device database, and then the Programmer view displays in Diamond.
6. Ensure that the device **LAE3-35EA** is selected in the Device column.
7. Double-click the Operation column to display the Device Properties dialog box and choose the following settings:
 - ▶ For Access Mode, choose **JTAG 1532 Mode** from the pull-down menu.
 - ▶ For Operation, choose **Fast Program** from the pull-down menu.
 - ▶ Ensure that the bitstream file named LEDTest_LEDTest.bit is selected as the programming file, as shown in Figure 34.


Figure 34: Device Properties Dialog Box



8. Click **OK**.
9. On the LatticeECP3 Versa board, ensure that user switches (dip switches) J6 and J7 are in the **OFF** position, and all of the rest of the user switches are in the **ON** position.

Note

Refer to the [LatticeECP3 Versa Evaluation Board User's Guide](#) for more information about the LatticeECP3 Versa evaluation board.

10. Click the Program button  on the Programmer toolbar to initiate the download.
11. If the programming process succeeded, you will see a green-shaded PASS in the Programmer Status column. Check the Programmer output console to see if the download passed.
12. At the end of this process, the FPGA is loaded with the sample test bitstream. This bitstream allows you to test the functionality of the LatticeECP3 Versa Development Kit board. If the design is successfully downloaded onto the LatticeECP3 device, the multi-segment LED display will display **0**.

To further test the design:
 - a. Push the user switch (dip switch) J7 to the **ON** position to activate the forward counter on the LED display.
 - b. Push the user switch (dip switch) J6 to the **ON** position while also keeping switch J7 in the **ON** position to activate the reverse counter on the LED display. The general purpose LEDs will now start flashing.
13. In Diamond, choose **File > Save**. In the Save .xcf File As dialog box, enter ecp3_versa_test.xcf in the File Name box, and click **Save**.

Task 16: Convert a File Using Deployment Tool

In Task 15, you used Diamond Programmer to download a bitstream (.bit) to a LatticeECP3 FPGA.

You will now use the Deployment Tool to convert the .bit to an industry-standard Hex file.

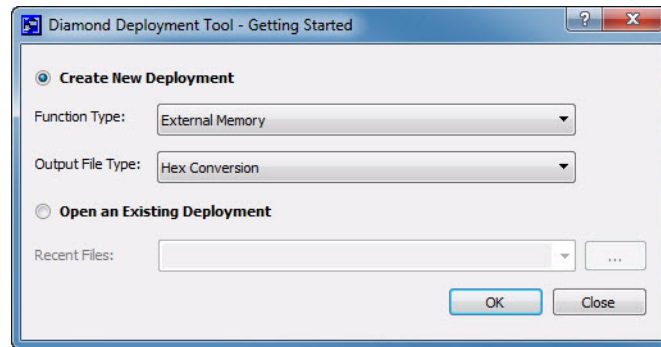
The Deployment Tool is a stand-alone tool available from the Diamond Accessories. The Deployment Tool graphical user interface is separate from the Diamond design environment. The Deployment Tool allows you to generate files for deployment for single devices and for a chain of devices. The Deployment Tool can also convert data files to other formats and use the data files it produces to generate other data file formats.


For the purpose of this tutorial, you will convert the same .bit file from Task 15 into an Intel Hex file.

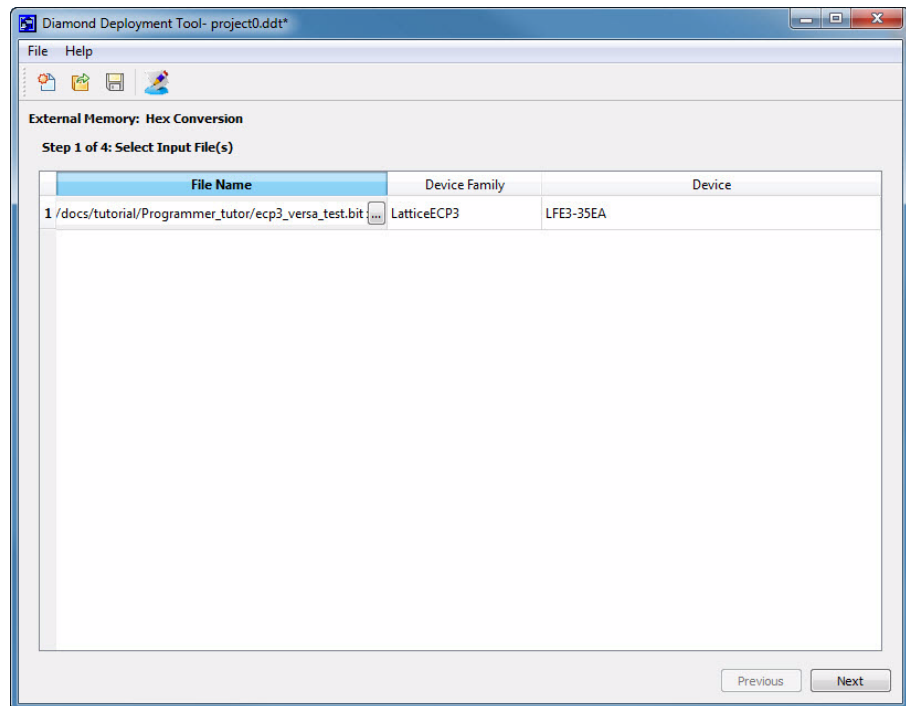
To convert the .bit to an Intel Hex file using Deployment Tool:

1. Choose **Programs > Lattice Diamond Programmer <version_number> > Deployment Tool** from the Windows Start menu.
2. In the Getting Started dialog box, choose **Create New Deployment**.
 - a. In the Function Type dropdown, choose **External Memory**.

- b. In the Output File Type dropdown, choose **Hex Conversion**.

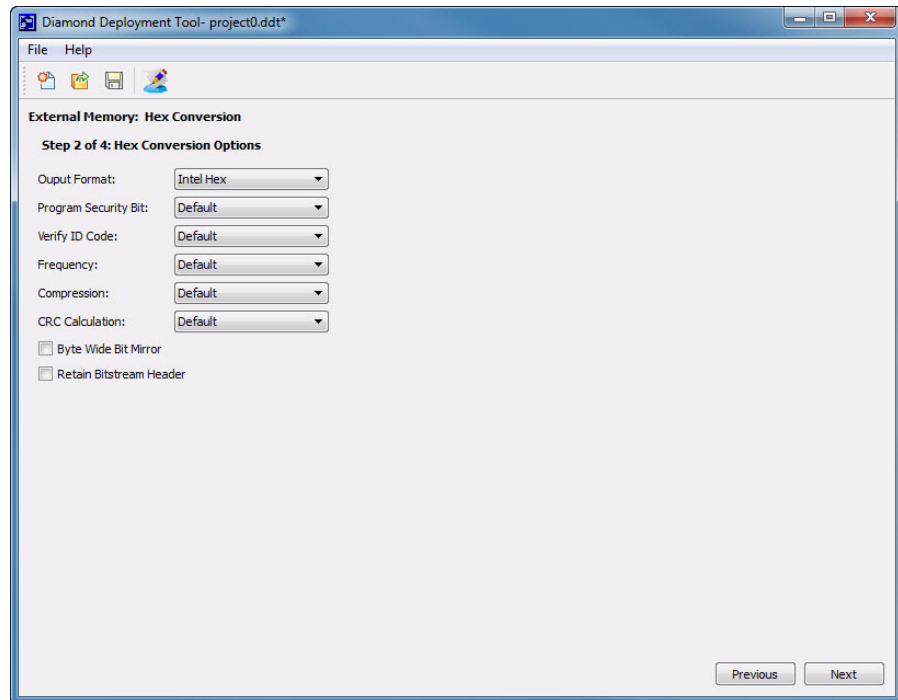


- c. Click **OK**.
3. In the Step 1 of 4: Select Input File(s) dialog box:
- Click in the File Name box.
 - Click  to display the Open File dialog box.
 - Browse to the ecp3_vera_test.bit file located in the tutorial directory.
 - Click **Open**.
 - Click **Next**.

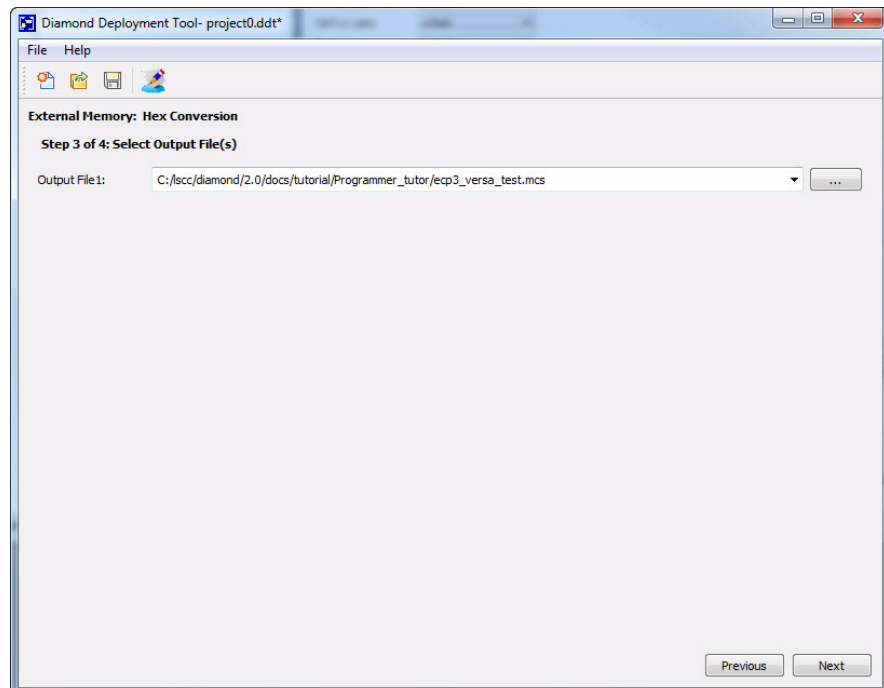


4. In the Step 2 of 4: Hex Conversion dialog box:
- Choose Output Format as **Intel Hex**.
 - Leave all other options (Program Security Bit, Verify ID Code, Frequency, Compression, and ORC Calculation) as **Default**.

- c. Leave Byte Wide Bit Mirror and Retain Bitstream Header unchecked.
- d. Click **Next**.

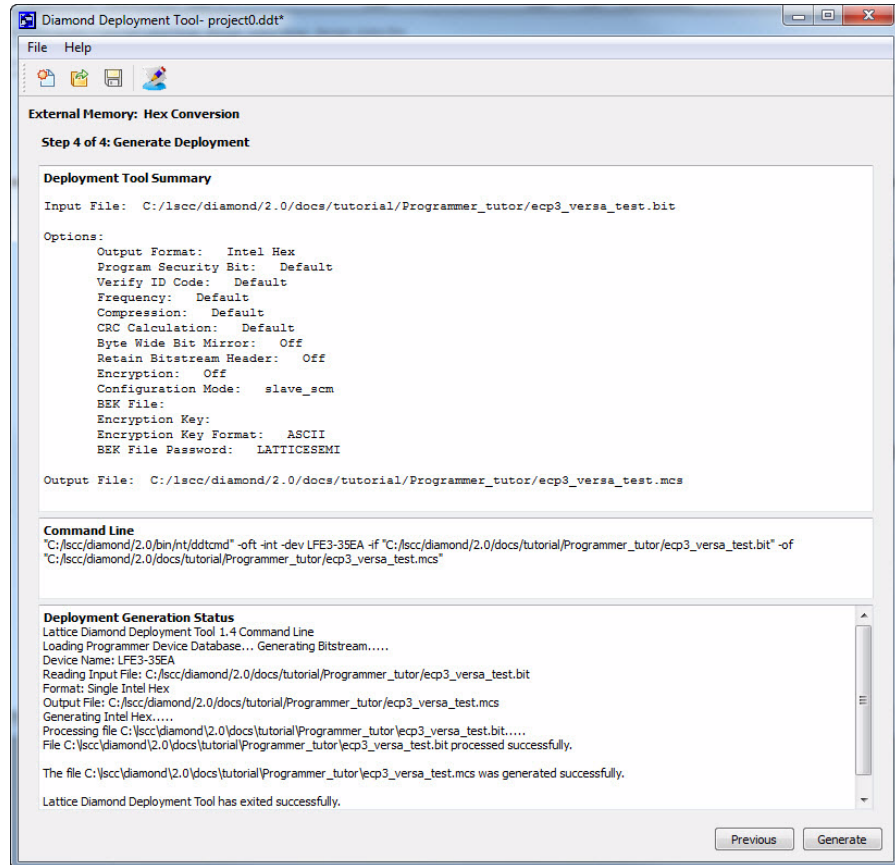


- 5. In the Step 3 of 4: Hex Conversion dialog box:
 - a. Ensure that the Output File1 is ecp3_versa_test.mcs.
 - b. Click **Next**.



6. In the Step 4of 4: Hex Conversion dialog box:
 - a. Review the Deployment Tool Summary.
 - b. Click **Generate**.

The Hex file (ecp3_versa_test.mcs) is created in the tutorial directory.



The .mcs file can be used to program the SPI Flash on the LatticeECP3 Versa Development Kit board using Programmer.



To save the Deployment Tool project:

1. Choose **File > Save**,
2. In the Save As dialog box, browse to the tutorial directory,
3. Save the Deployment Tool (.ddt) file using either the default file name or another file name.

Task 17: Use Reveal Inserter to Add On-chip Debug Logic

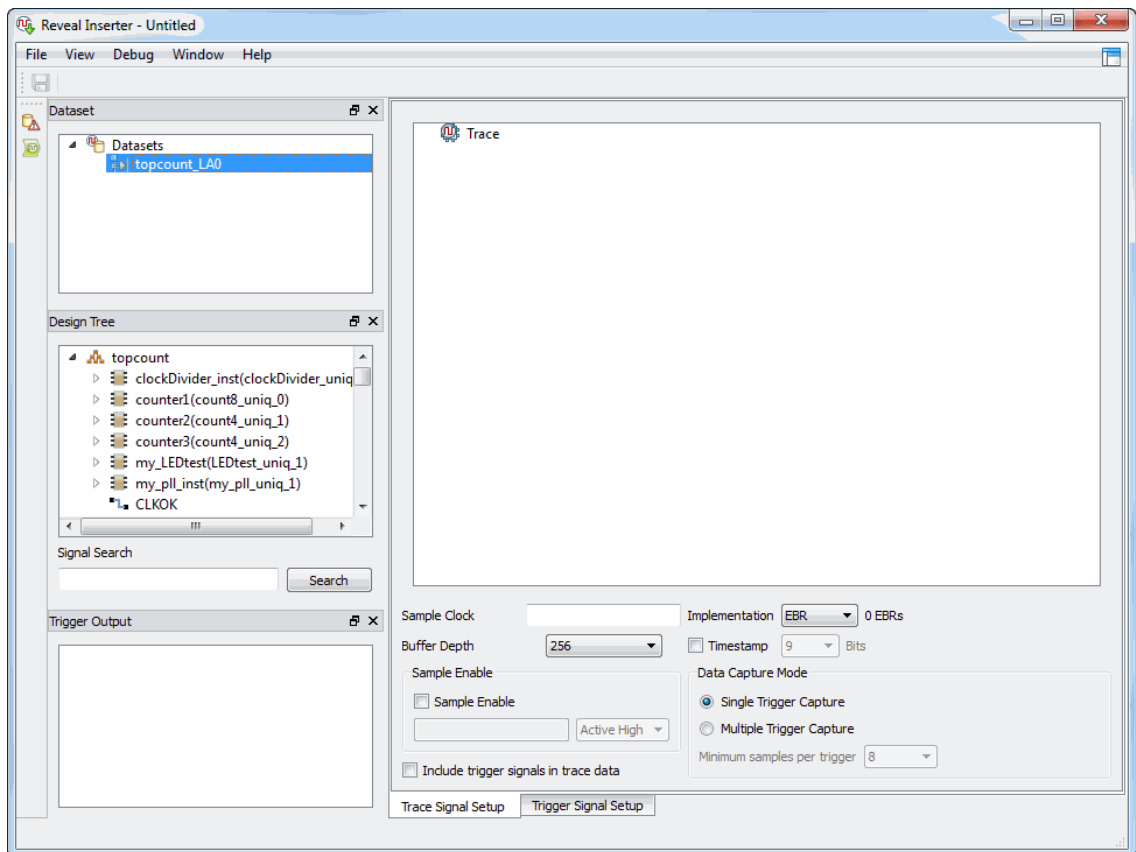
In this task, you will use the Reveal Inserter to configure a Reveal core based on triggering conditions and the desired trace buffer. The primary output of the Reveal Inserter is a modified version of your design with one or more cores instantiated and the core logic ready for mapping, placement, and routing.

To generate and add a Reveal core:

1. Choose **Tools > Reveal Inserter** or click the  icon on the toolbar.
2. Click the  icon the upper right corner to detach Reveal Inserter.

The Reveal Inserter is detached from the Diamond main window, as shown in Figure 35.

Figure 35: Reveal Inserter Main Window



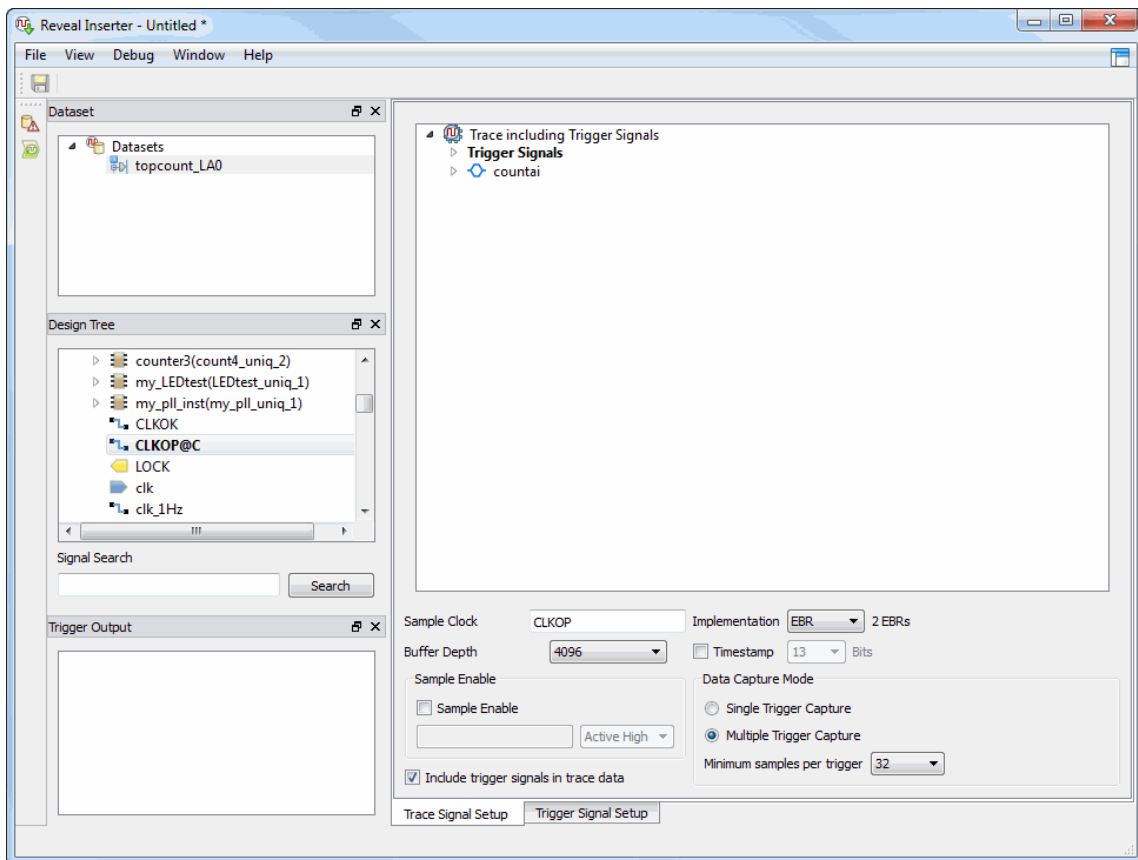
3. Click on the **Trace Signal Setup** tab, if it is not already selected.
4. From the Design Tree pane, drag the **counter1/counterai[7:0]** bus to the Trace Data pane. Right-click the trace bus and choose **Rename Trace Bus**, and name the bus **countai**.
5. Select the **Include Trigger Signals in Trace Data** option.

The name of the bus now appears in bold font in the Design Tree pane.

6. Drag the **CLKOP** signal from the Design Tree pane to the Sample Clock box, or type **CLKOP** in the Sample Clock box.
7. From the pulldown menu in the Buffer Depth box, select **4096**.
8. Set Data Capture Mode to **Multiple Trigger Capture** and Minimum Samples Per Trigger to **32**.

The Trace Signal Setup tab should now resemble Figure 36.

Figure 36: Trace Signal Setup Tab



Setting Up the Trigger Units

You will set up the trigger units in the Trigger Unit section of the Trigger Signal Setup tab.

To set up the trigger units:

1. Click on the **Trigger Signal Setup** tab.

One line appears in the Trigger Unit section of the tab with a default name of TU1.

2. Double-click the TU1 name in the Name box, backspace over “TU1,” and type **countbi**.
3. Drag the **counter1/countbi[0-7]** signals from the Design Tree pane to the Signals (MSB:LSB) box
4. In the Operator box of the trigger unit, use the default of **==**.
5. In the Radix box, select **Hex** from the drop-down menu.
6. In the Value box, double-click, backspace, and type **88**.
7. Click **Add** to add a second trigger unit.
8. In the Name box, double-click TU2, backspace over “TU2,” and type **dir**.
9. Drag the **directionR** signal from the Design Tree pane to the Signals (MSB:LSB) box.
10. In the Operator box, select **rising edge** from the drop-down menu.
11. In the Radix box, select the default of **Bin**.
12. In the Value box, double-click, backspace, and type **1**.

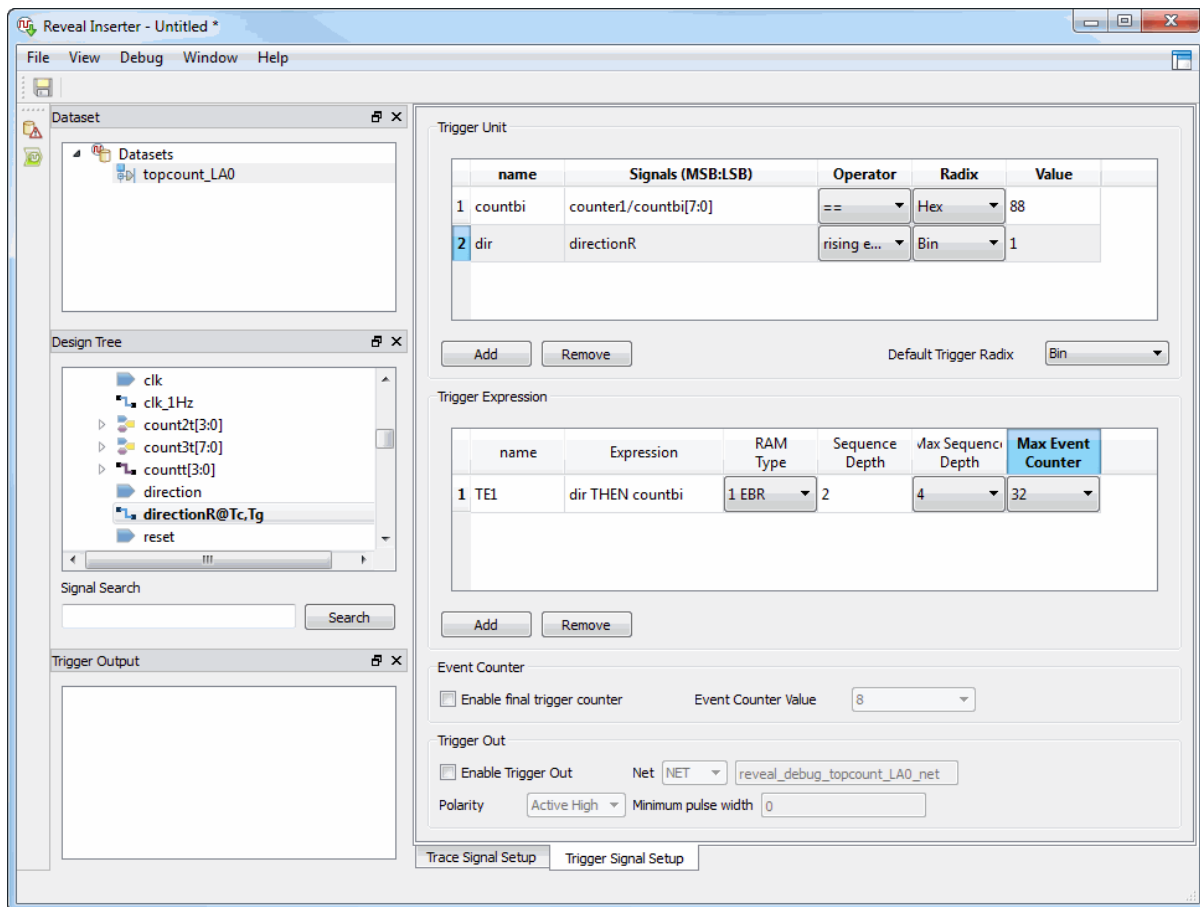
Setting Up the Trigger Expressions

Now you will set up the trigger expressions in the Trigger Expression section of the tab.

To set up the trigger expressions:

1. In the Name box in the Trigger Expressions section, use the default name of TE1.
2. In the Expression box, select the countbi and dir trigger units by typing **dir THEN countbi**.
3. In the RAM Type box, select **1 EBR** from the drop-down menu.
4. In the Sequence Depth box, make sure a value of **2** appears.
5. In the Max Sequence Depth box, select **4** from the drop-down menu.
6. In the Max Event Counter box, select **32** from the drop-down menu.
7. The Trigger Signal Setup tab should now resemble Figure 37.

Figure 37: Trigger Signal Setup Tap

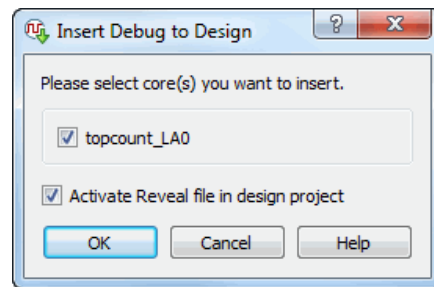


Inserting the Debug Logic

Now you will insert the debug logic into the design project.

To insert the debug logic:

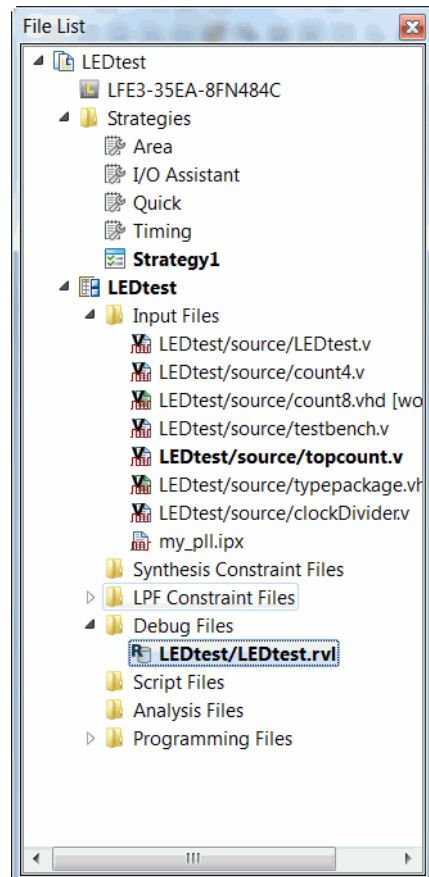
1. Choose **Debug > Insert Debug**.
2. In the Insert Debug to Design dialog box, shown in Figure 38, be sure that the **Activate Reveal File in Design Project** option is selected.

Figure 38: Insert Debug to Design Dialog Box

3. Click **OK**. Save the file as **<project_directory>/LEDtest.rvl**.

The Reveal Inserter now invokes the Synplify Pro synthesis tool, adds the debug logic, and imports the Reveal project (.rvl) file into Diamond.

The Reveal (.rvl) file is now added to the Debug Files in the File List, as shown in Figure 39.

Figure 39: File List with Debug File

4. In the Reveal Inserter window, choose **File > Close Window**.

Generating a Bitstream and Programming the FPGA

Use the Process view to generate files for exporting, and use Programmer to download the bitstream to the FPGA.

To generate a bitstream:

1. From the Process view, choose **Export Files**.
A set of export files appear under the Export Files process.
2. Select the following Export Files:
Bitstream File
3. Click the **Run** button  on the Diamond toolbar.
Diamond generates the selected files and saves them in your project directory.
4. Choose **Tools > Programmer**, or click the  icon on the toolbar.
5. Click the Program button  on the Programmer toolbar to initiate the download.
6. If the programming process succeeded, you will see a green-shaded PASS in the Programmer Status column. Check the Programmer output console to see if the download passed.

Task 18: Use Reveal Logic Analyzer Perform Logic Analysis

In this task, you will use the Reveal Logic Analyzer to set up trigger conditions and view trace buffer data from the on-chip Reveal core operating within the device on the LatticeECP3 standard evaluation board. The trigger setup influences under what specific conditions and how the Reveal core trace signal states are displayed in the Reveal Logic Analyzer's graphical user interface. You will explore just a few of the many ways to trigger and trace the system.

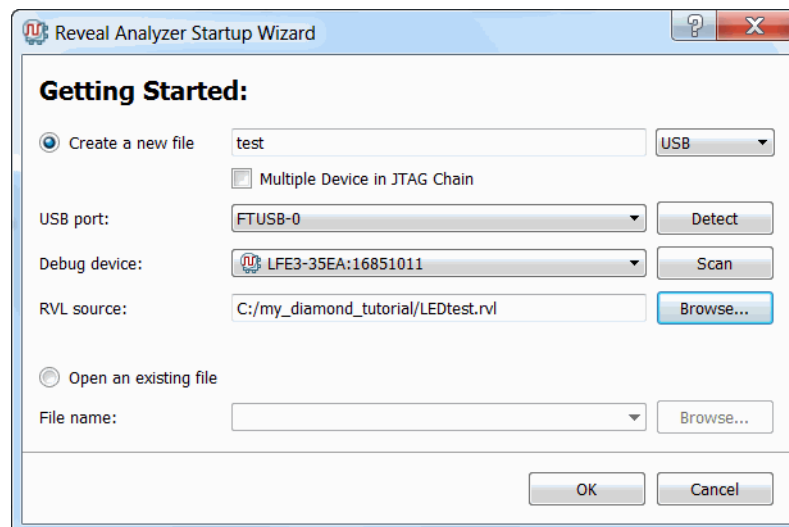
This task assumes that the LatticeECP3 Versa board is connected to your computer with a download cable, as described in Task 15.

Creating a New Reveal Logic Analyzer Project

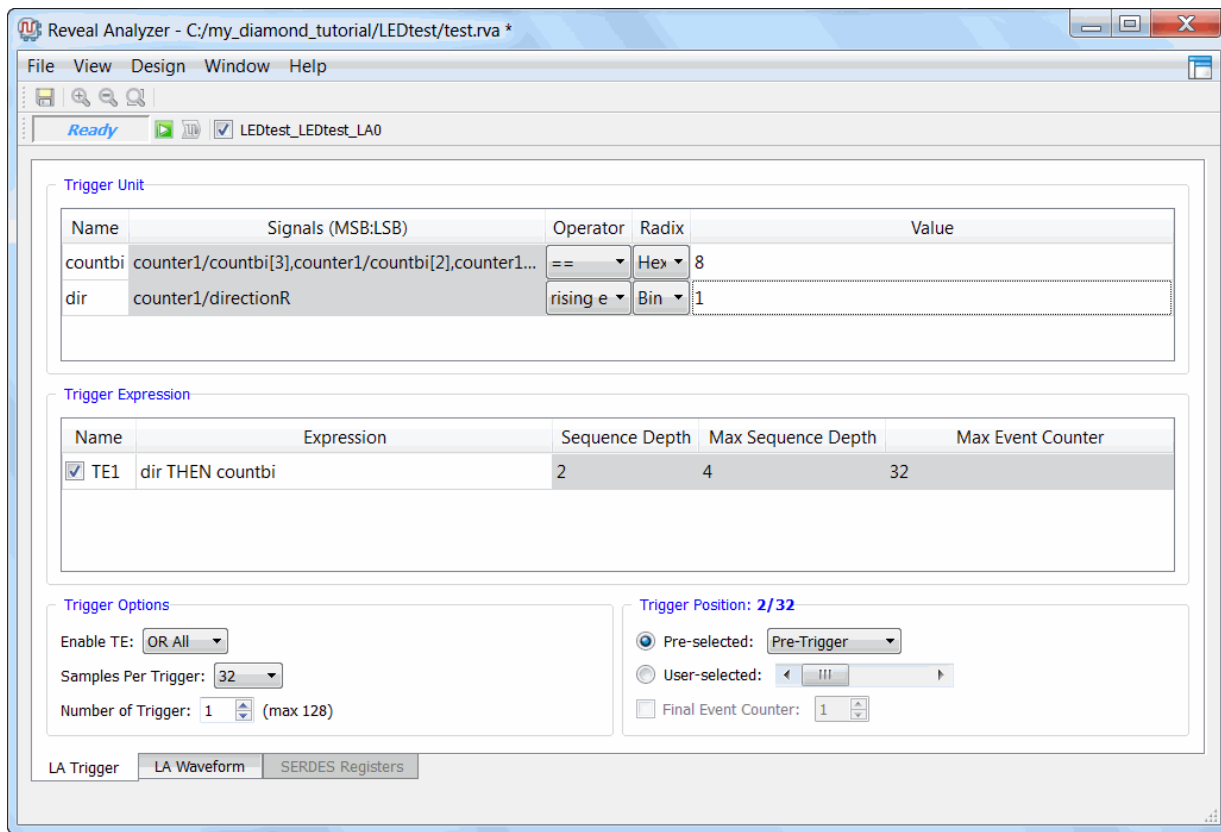
You must first create a Reveal Analyzer project.

To create a new Reveal Logic Analyzer project:

1. In the Diamond main window, choose **Tools > Reveal Analyzer**.
The Reveal Logic Analyzer Startup Wizard dialog box appears, as shown in Figure 40.
2. In the upper left of the Reveal Analyzer Startup Wizard dialog box, select **Create a new File**.
3. Type **Test** in the box to name the file.
The .rva extension is added automatically.
4. In the drop-down menu on the top row, choose **USB**, if it is not already selected.
5. Select **Multiple Device in JTAG Chain**.
6. Click **Detect**.
The cable connected to the PC is detected, and is listed in the USB Port box.
7. Click **Scan** to find the FPGA.
The LatticeECP3 device on the Versa board is displayed in the Debug device box.
8. In the RVL Source box, navigate to **<project_directory>/LEDtest.rvl**.

Figure 40: Reveal Analyzer Startup Wizard

9. Click **OK**.
The Reveal Logic Analyzer main window now appears with the Trigger Signal Setup tab selected, as shown in Figure 41. It contains the same trigger units and trigger expressions that you set up in the Reveal Inserter.

Figure 41: Reveal Analyzer

10. Choose Trigger Options:

Samples Per Trigger: **128**.

11. Choose Trigger Position: **Pre-selected: Pre-Trigger**.


In the Trigger position section, you can specify the trigger position relative to the trace data. The numbers in the section title show the current position. The two options to choose from include:


- ▶ **Pre-selected** and choose one of the standard positions
 - ▶ Pre-Trigger: 8/128 of the way from the beginning of the samples.
 - ▶ Center-Trigger: 64/128 of the way from the beginning of the samples.
 - ▶ Post-Trigger: 120/128 of the way from the beginning of the samples.
- ▶ **User-selected** and choose a position with the slider.

Running the Logic Analyzer

Now that the Reveal Logic Analyzer is set up, you can run the Logic Analyzer.


To capture data:

1. Click the Run  button in the Reveal Analyzer toolbar.

The Run button changes into the Stop  button and the status bar next to the button shows the progress.

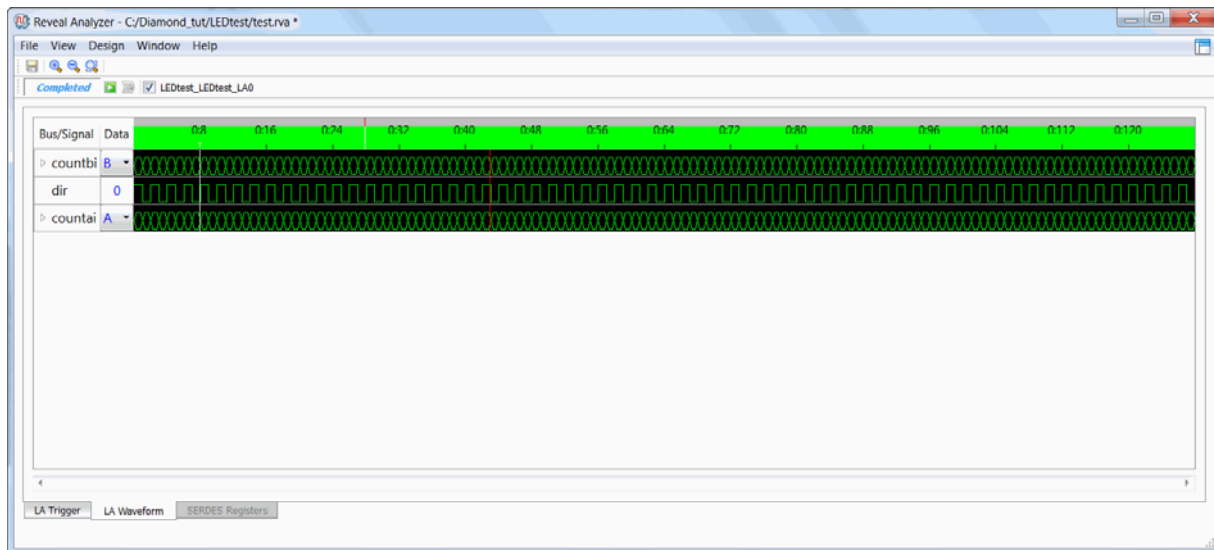
Reveal Analyzer first configures the modules selected for the correct trigger condition, then waits for the trigger conditions to occur. When a trigger occurs, the data is uploaded to your computer. The resulting waveforms appear in the Waveform View tab.

On the row of eight dip-switches on your board, push down the seventh and eighth switches from the left to set the direction signal low and reset the counter. Then pull the seventh switch up to bring the signal level high, generating a rising edge that will cause the “dir” trigger unit to be true. The trigger expression can now evaluate the next trigger unit and generate a trigger for data to be captured.

If no trigger occurs click the Manual Trigger  button.

You now see the waveforms displayed, as shown in Figure 42.

Figure 42: Reveal Analyzer Waveform



Summary of Accomplishments

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

- ▶ Create a new Lattice Diamond project
- ▶ Create an IPexpress module
- ▶ Check Hardware Description Language (HDL)
- ▶ Inspect strategy settings

- ▶ Examine resources
- ▶ Run synthesis processes
- ▶ Set timing and location assignments
- ▶ Run place and route
- ▶ Examine post place and route results
- ▶ Adjust static timing constraints and review results
- ▶ Compare multiple place and route runs
- ▶ Verify functionality with simulation
- ▶ Analyze power consumption
- ▶ Run export utility programs
- ▶ Download a bitstream to an FPGA
- ▶ Convert a file using Deployment Tool
- ▶ Use Reveal Inserter to add on-chip debug logic
- ▶ Use Reveal Logic Analyzer perform logic analysis

Recommended References

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

