Lattice Diamond User Guide

Copyright

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

June 2010

Trademarks

Lattice Semiconductor Corporation, L Lattice Semiconductor Corporation (logo), L (stylized), L (design), Lattice (design), LSC, E^{2}CMOS, Extreme Performance, FlashBAK, 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, LatticeEC, LatticeECP, LatticeECP-DSP, LatticeECP2, LatticeECP2M, LatticeMico8, LatticeMico32, LatticeSC, LatticeSCM, LatticeXP, LatticeXP2, MACH, MachXO, MACO, ORCA, PAC, PAC-Designer, PAL, Performance Analyst, PURESPEED, Reveal, Silicon Forest, Speedlocked, Speed Locking, SuperBIG, SuperCOOL, SuperFAST, SuperWIDE, sysCLOCK, sysCONFIG, sysDSP, sysHSI, sysI/O, sysMEM, The Simple Machine for Complex Design, 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.

HyperTransport is a licensed trademark of the HyperTransport Technology Consortium in the U.S. and other jurisdictions.

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^{TM} is the leading-edge software design environment for cost sensitive, low-power Lattice FPGA architectures. Lattice Diamond is the next generation replacement for ispLever.

This user guide provides a description of 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.

Lattice Diamond overview

The Lattice Diamond software design environment is an integrated tool environment that provides a modern, comprehensive user interface for controlling the Lattice Semiconductor FPGA implementation process.

Lattice Diamond features ease of use, design exploration, improved design flow and numerous other enhancements. The combination of new and enhanced features allows users to complete designs faster, easier, and with better results than ever before.

Refer to the Release Notes on the Lattice website for a list of supported devices.

Lattice Diamond uses an expanded project based design flow and integrated tool views so that design alternatives and what-if scenarios are easily created and analyzed. New concepts of *Implementations* and *Strategies* enable a convenient, structured methodology 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 View provide new capabilities in the integrated framework. There is cross probing between tools as well as a new shared memory architecture to ensure fast performance and better memory utilization.

Lattice Diamond is highly customizable and includes Tcl scripting capabilities, either from within its built in console, or from an external shell.

User guide organization

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

Key concepts and work flows are explained in the chapters on *Design Environment Fundamentals* and *Lattice Diamond Design Flow*.

Basic operation of the design environment is described in the *User Interface Operation* chapter.

The second half of the book provides greater detail and practical usage information. *Working with Projects* shows how to set up project implementations and strategies. *Working with Tools and Views* describes the many tool views available.

2

Getting Started

This chapter explains how to run Lattice Diamond and open or create a project. Importing a project from ispLever is covered, and some key differences between Lattice Diamond and ispLever are described.

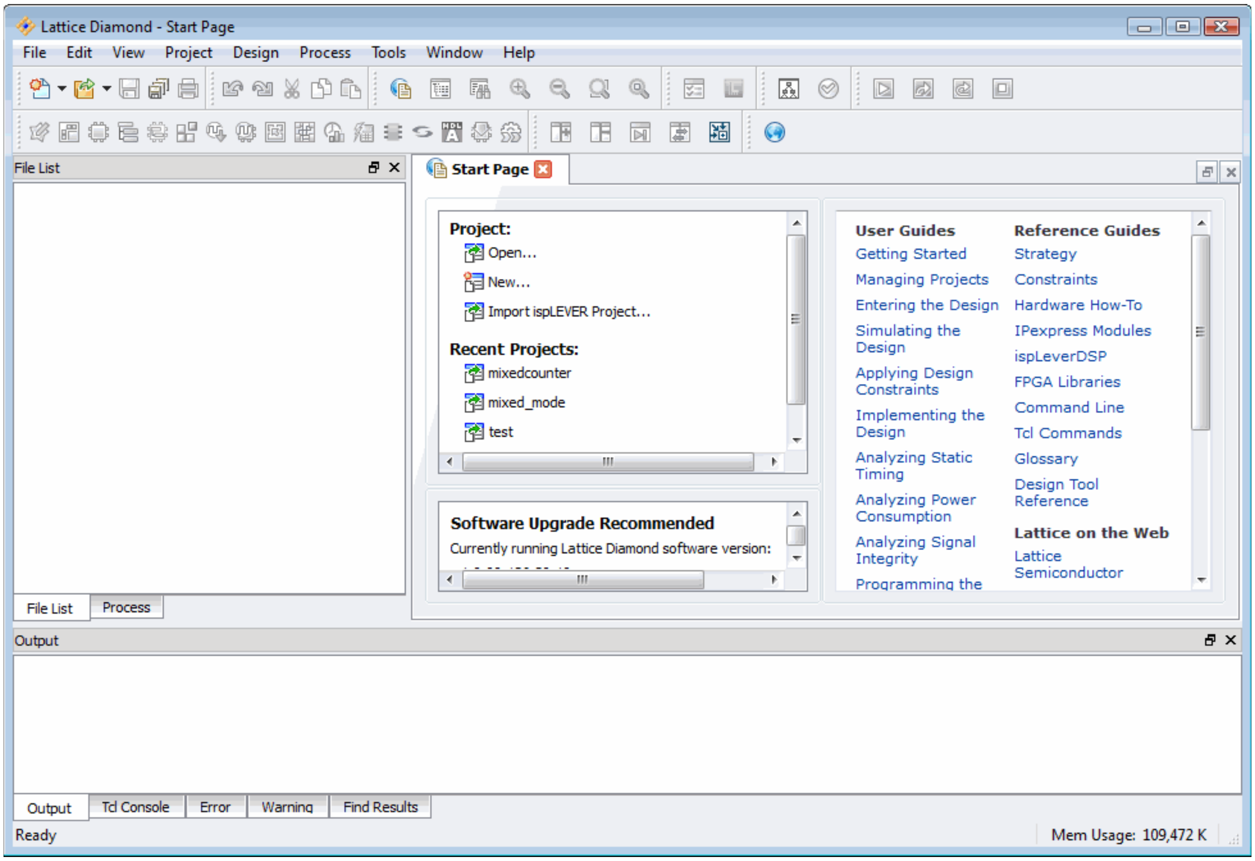
Prerequisites

It is assumed you have already installed the Lattice Diamond software and product license. See the Lattice Diamond Release notes for more information on product installation.

Running Lattice Diamond

To run Lattice Diamond select *Lattice Diamond* from the installation location. This brings up the default Start Page.

Figure 1: Default Start Page



Software Upgrade Recommended

Currently running Lattice Diamond software version:

Ready

Mem Usage: 109,472 K

Creating a new Project

New projects are created using the *New Project* wizard. You can invoke the wizard from two locations:

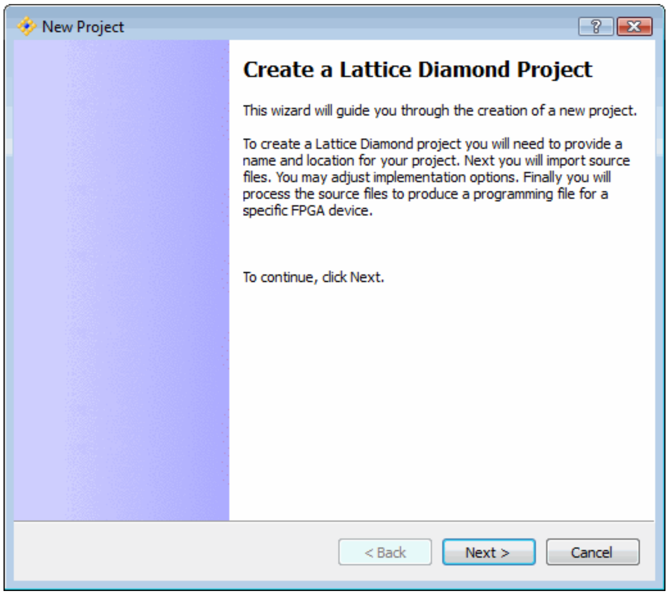
- On the Start Page select **Project > New**
- From the File menu select **New > Project**

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.

Several example project design files are included in Lattice Diamond. Here is an example of creating a new project using the mixedcounter example.

*On the Start Page select **Project > New***

Figure 2: Create a New Project



Create a Lattice Diamond Project

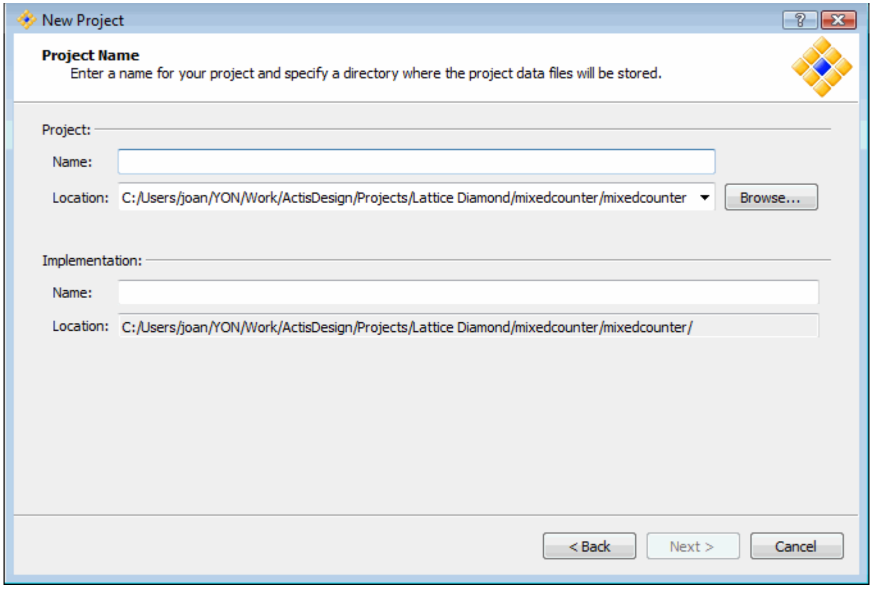
This wizard will guide you through the creation of a new project.

To create a Lattice Diamond project you will need to provide a name and location for your project. Next you will import source files. You may adjust implementation options. Finally you will process the source files to produce a programming file for a specific FPGA device.

To continue, click Next.

Select Next to bring up the Project Name dialog.

Figure 3: Project Name



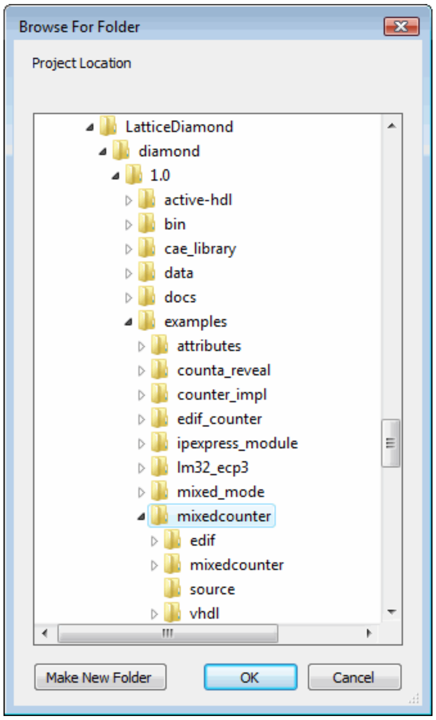
Project Name
Enter a name for your project and specify a directory where the project data files will be stored.

Project:
Name:
Location: C:/Users/joan/YON/Work/ActisDesign/Projects/Lattice Diamond/mixedcounter/mixedcounter

Implementation:
Name:
Location: C:/Users/joan/YON/Work/ActisDesign/Projects/Lattice Diamond/mixedcounter/mixedcounter/

Select **Browse** to enter a Location. Navigate to the Lattice Diamond examples directory and select *mixedcounter* as shown:

Figure 4: Project Browse

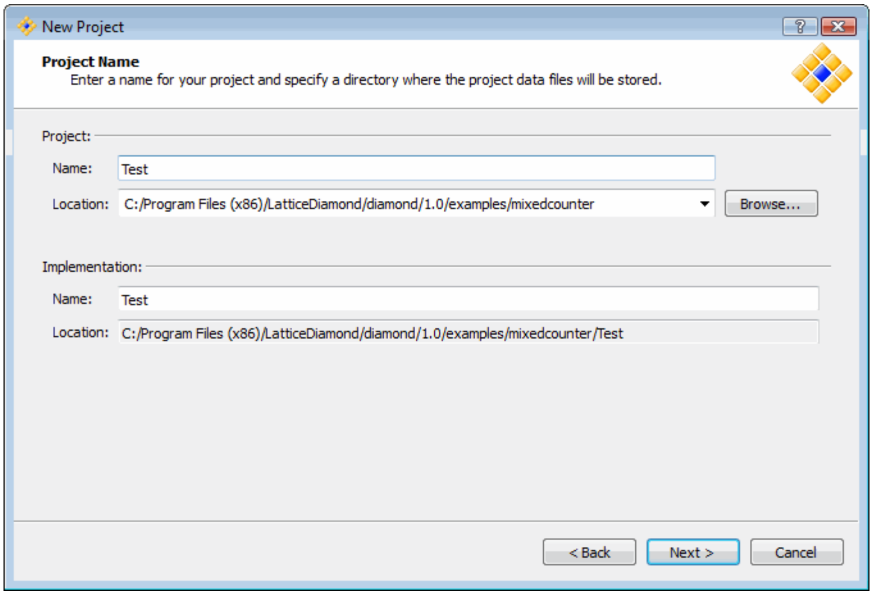

Browse For Folder

Project Location

- LatticeDiamond
  - diamond
    - 1.0
      - active-hdl
      - bin
      - cae_library
      - data
      - docs
      - examples
        - attributes
        - counta_reveal
        - counter_impl
        - edif_counter
        - ipexpress_module
        - lm32_ecp3
        - mixed_mode
        - mixedcounter
          - edif
          - mixedcounter
          - source
          - vhdl

Enter a Project Name. Notice that the default is for the Implementation Name to be the same, but this is not required. We'll leave it the same for this example and call both Project and Implementation *Test*.

Figure 5: Enter Project and Implementation Name


New Project

Project Name

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

Project:

Name: Test

Location: C:/Program Files (x86)/LatticeDiamond/diamond/1.0/examples/mixedcounter

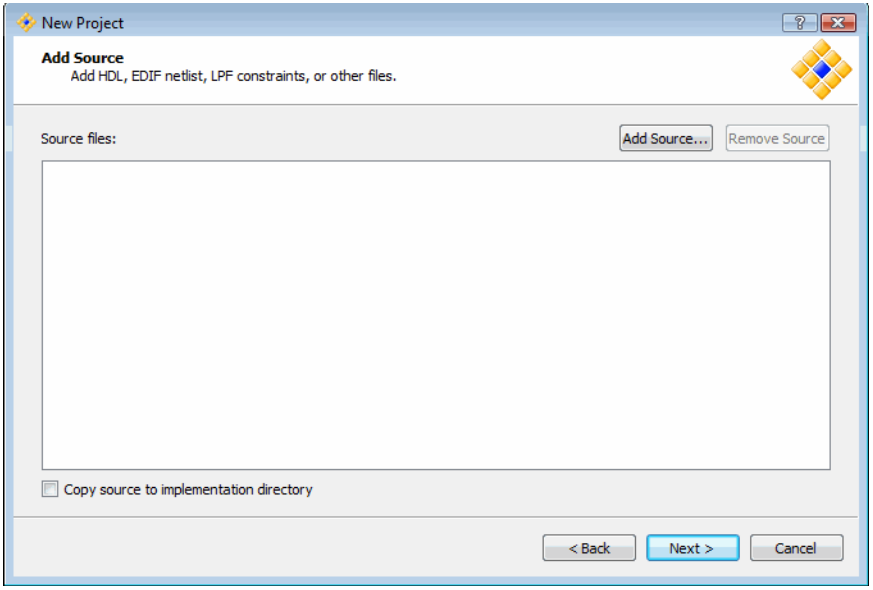
Implementation:

Name: Test

Location: C:/Program Files (x86)/LatticeDiamond/diamond/1.0/examples/mixedcounter/Test

Select **Next** to go to the *Add Source* dialog.

Figure 6: Add Source



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

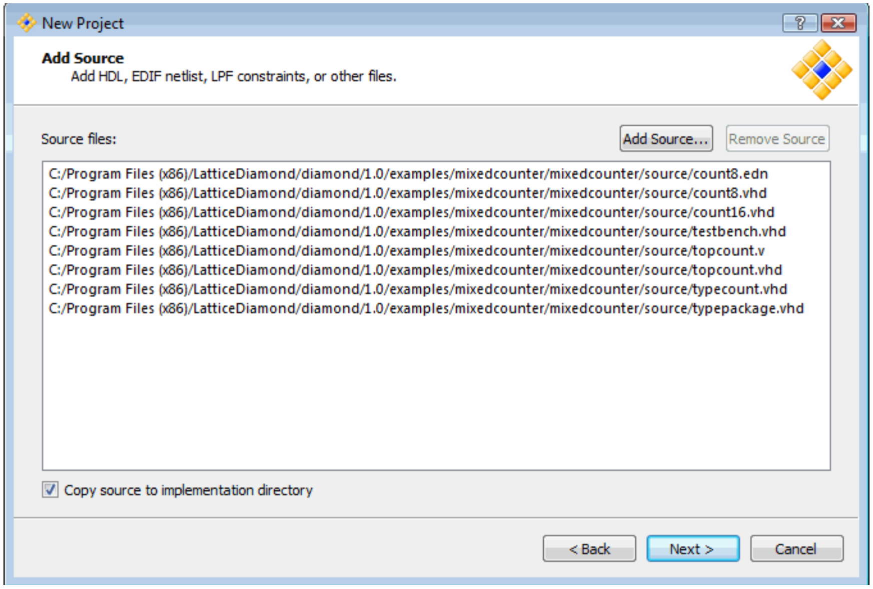
Source files:

☐ Copy source to implementation directory

From this dialog you can add HDL source files, EDIF netlist files, LPF constraint files or any other project files.

Select **Add Source** to bring up a file browser in the project example location. Go into the source directory and select all Verilog and VHDL files to add. Select **Open** to add the files to the project.

Figure 7: Source



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

Source files:

```
C:/Program Files (x86)/LatticeDiamond/diamond/1.0/examples/mixedcounter/mixedcounter/source/count8.edn
C:/Program Files (x86)/LatticeDiamond/diamond/1.0/examples/mixedcounter/mixedcounter/source/count8.vhd
C:/Program Files (x86)/LatticeDiamond/diamond/1.0/examples/mixedcounter/mixedcounter/source/count16.vhd
C:/Program Files (x86)/LatticeDiamond/diamond/1.0/examples/mixedcounter/mixedcounter/source/testbench.vhd
C:/Program Files (x86)/LatticeDiamond/diamond/1.0/examples/mixedcounter/mixedcounter/source/topcount.v
C:/Program Files (x86)/LatticeDiamond/diamond/1.0/examples/mixedcounter/mixedcounter/source/topcount.vhd
C:/Program Files (x86)/LatticeDiamond/diamond/1.0/examples/mixedcounter/mixedcounter/source/typecount.vhd
C:/Program Files (x86)/LatticeDiamond/diamond/1.0/examples/mixedcounter/mixedcounter/source/typepackage.vhd
```

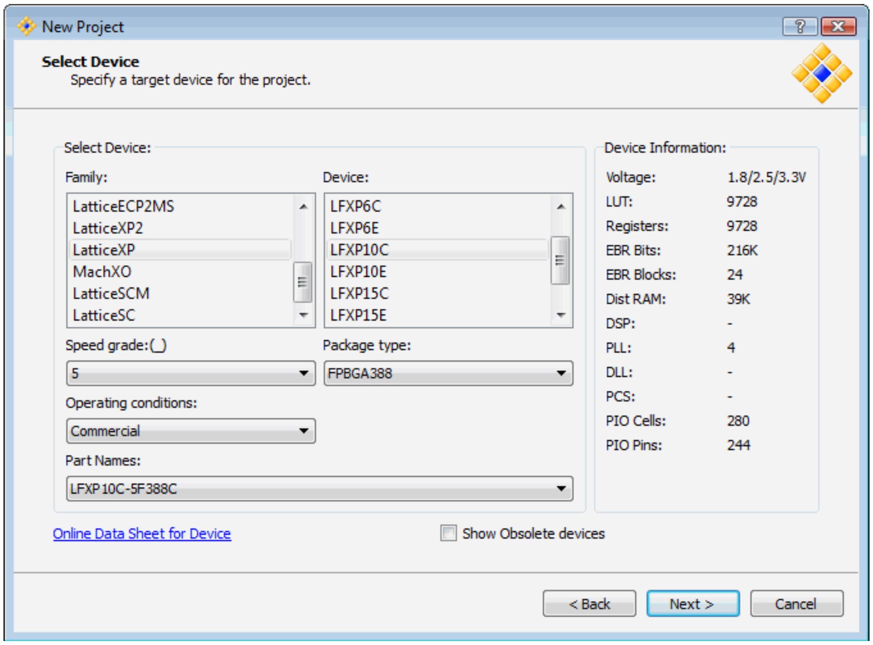
☑ Copy source to implementation directory

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

Notice there is an option to *Copy source to implementation directory*. This is selected if you want to have separate copies of the source files in the two different implementations. If you want to use one version of the source across both implementations make sure this is not selected.

*Select **Next** to go to the Select Device step.*

Figure 8: Select Device



New Project

Select Device

Specify a target device for the project.

Select Device:

Family:

- LatticeECP2MS
- LatticeXP2
- LatticeXP
- MachXO
- LatticeSCM
- LatticeSC

Device:

- LFXP6C
- LFXP6E
- LFXP10C
- LFXP10E
- LFXP15C
- LFXP15E

Speed grade:(_): 5

Package type: FPBGA388

Operating conditions: Commercial

Part Names: LFXP10C-5F388C

Device Information:

Voltage:	1.8/2.5/3.3V
LUT:	9728
Registers:	9728
EBR Bits:	216K
EBR Blocks:	24
Dist RAM:	39K
DSP:	-
PLL:	4
DLL:	-
PCS:	-
PIO Cells:	280
PIO Pins:	244

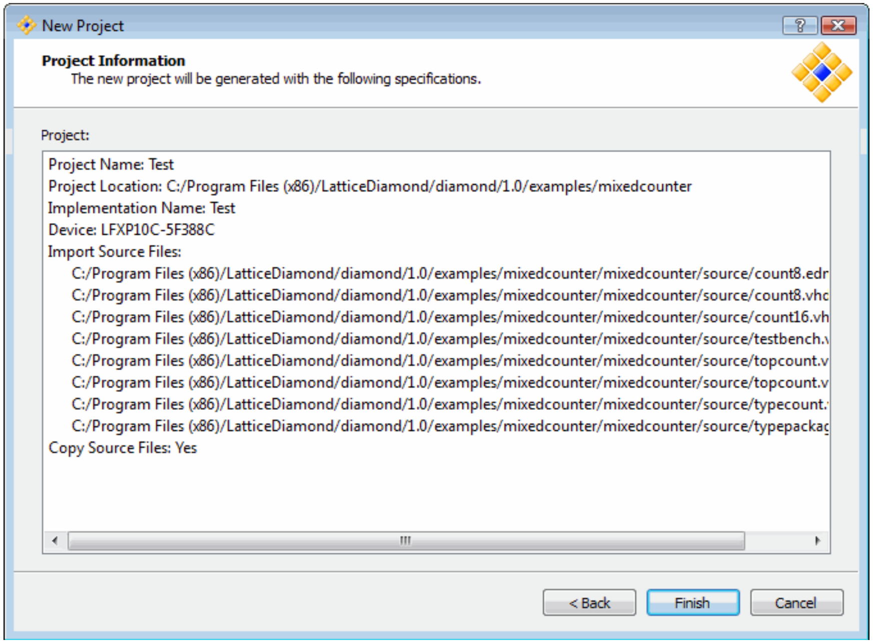
Online Data Sheet for Device

□ Show Obsolete devices

In this step you can select the target device including speed grade, package type, operating conditions and part name. For our example we'll use the defaults.

*Select **Next** to see the Project Information summary.*

Figure 9: Project Summary



Project Information

The new project will be generated with the following specifications.

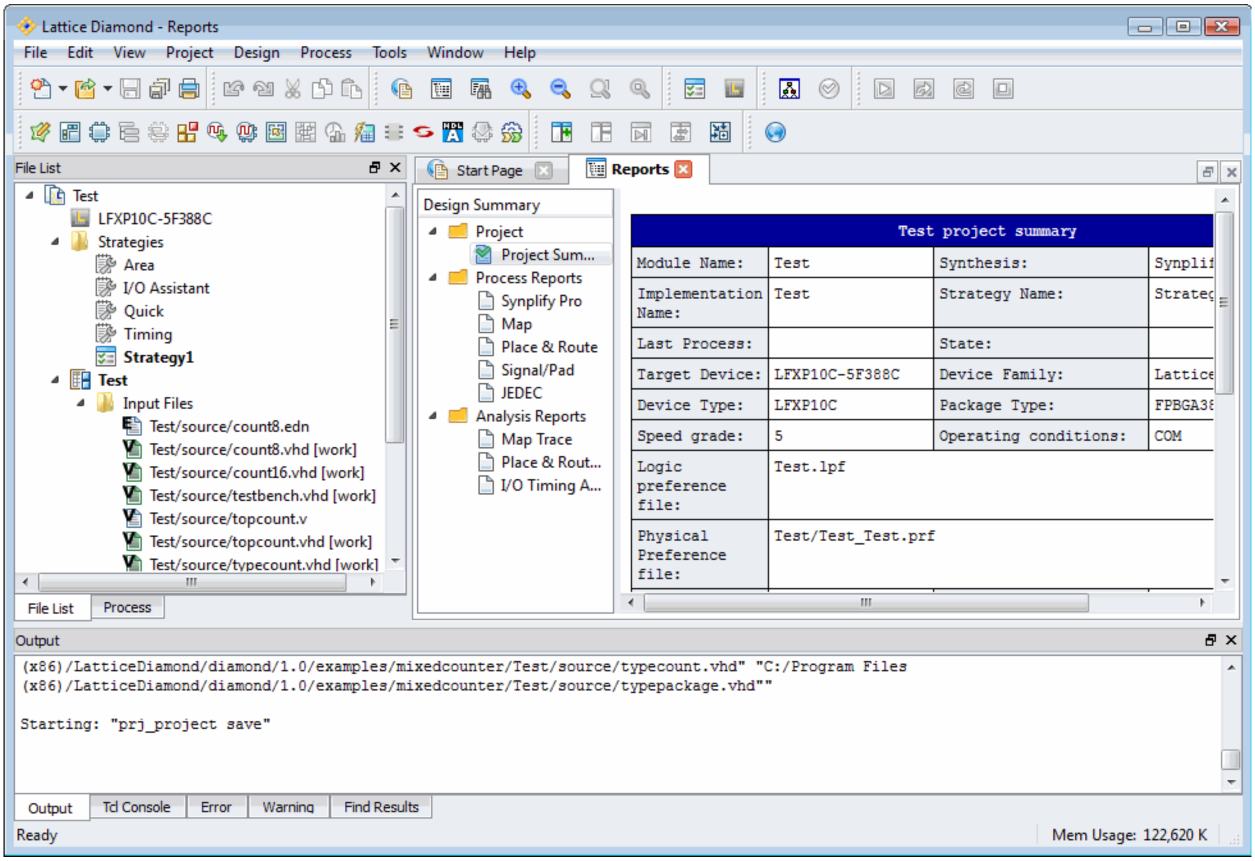
Project:

```
Project Name: Test
Project Location: C:/Program Files (x86)/LatticeDiamond/diamond/1.0/examples/mixedcounter
Implementation Name: Test
Device: LFXP10C-5F388C
Import Source Files:
    C:/Program Files (x86)/LatticeDiamond/diamond/1.0/examples/mixedcounter/mixedcounter/source/count8.edr
    C:/Program Files (x86)/LatticeDiamond/diamond/1.0/examples/mixedcounter/mixedcounter/source/count8.vhc
    C:/Program Files (x86)/LatticeDiamond/diamond/1.0/examples/mixedcounter/mixedcounter/source/count16.vh
    C:/Program Files (x86)/LatticeDiamond/diamond/1.0/examples/mixedcounter/mixedcounter/source/testbench.
    C:/Program Files (x86)/LatticeDiamond/diamond/1.0/examples/mixedcounter/mixedcounter/source/topcount.v
    C:/Program Files (x86)/LatticeDiamond/diamond/1.0/examples/mixedcounter/mixedcounter/source/topcount.v
    C:/Program Files (x86)/LatticeDiamond/diamond/1.0/examples/mixedcounter/mixedcounter/source/typecount.
    C:/Program Files (x86)/LatticeDiamond/diamond/1.0/examples/mixedcounter/mixedcounter/source/typepackag
Copy Source Files: Yes
```

At this step (or any step in the process) you can select **Back** to review or change your selections.

Select **Finish**. The newly created project is now created and open.

Figure 10: Opened Project



Test project summary			
Module Name:	Test	Synthesis:	Synplif
Implementation Name:	Test	Strategy Name:	Strateg
Last Process:		State:	
Target Device:	LFXP10C-5F388C	Device Family:	Lattice
Device Type:	LFXP10C	Package Type:	FPBGA38
Speed grade:	5	Operating conditions:	COM
Logic preference file:	Test.lpf		
Physical Preference file:	Test/Test_Test.prf		

```
(x86)/LatticeDiamond/diamond/1.0/examples/mixedcounter/Test/source/typecount.vhd" "C:/Program Files
(x86)/LatticeDiamond/diamond/1.0/examples/mixedcounter/Test/source/typepackage.vhd""

Starting: "prj_project save"
```

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

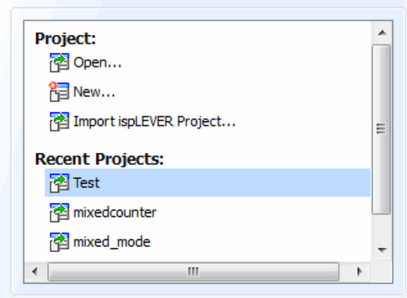
To close a project from the **File** menu select **Close Project**.

Opening an existing Project

Existing Lattice Diamond projects can be opened in these ways:

- On the Start Page select **Project > Open**
- From the File menu select **Open > Project**
- On the Start Page select **Recent Projects > Test**, or whatever selection you desire from the list. There is also a Recent Projects selection in the File menu.

Figure 11: Open Recent Project

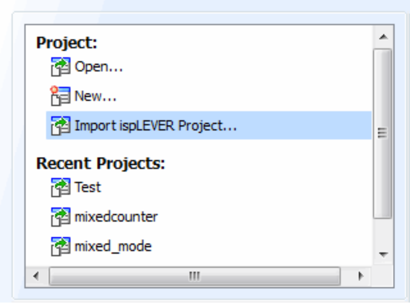


Importing an ispLever Project

Projects created using ispLever can be imported into Lattice Diamond.

- On the **Start Page** select **Project > Import ispLever Project**
- From the **File** menu select **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.

Limitations to the import/conversion process include:

- NGO files in ispLever projects will need to be manually copied into the Lattice Diamond project if the NGO files were originally copied into the ispLever project. For example NGO files that were copied from Lattice IP generation.
- Tool settings and options for MAP, PAR, etc. are not imported. You will need to manually edit your Strategy settings in Lattice Diamond for your preferred settings.
- lpc files are replaced with ipx files in Lattice Diamond. You will need to re-generate your IP by double-clicking on the lpc file. The resultant wizard will help you generate the new ipx file, replacing the old lpc file.

Next steps

Once you have a project up and running you can go sequentially through the rest of this guide to learn how to work with the entire design environment, or you can go directly to any topics of interest. The chapters on *Design Environment Fundamentals* and the *Lattice Diamond Design Flow* provide background on the key concepts and methodologies in Lattice Diamond. *User Interface Operation* goes over the functions and controls available in the environment. The chapters on *Working with Projects* and *Working with Tools and Views* explain how to run processes and tools. *Tcl Scripting* provides an introduction to the scripting capabilities available and *Advanced Topics* includes discussions of special topics.

Differences from ispLever

There are a number of differences between Lattice Diamond and ispLever. Some of the key differences, especially regarding how projects are managed, are:

- ispLever has multiple project types, but there is only one Diamond project type. In ispLever you need different project types for each type of source, for example one project for Verilog and a different project for VHDL. In Lattice Diamond there is only one project type and it can include sources of different types. For example one Lattice Diamond project can have both Verilog and VHDL source.
- Lattice Diamond must have Implementations and Strategies. These don't exist in ispLever.
- ispLever consists of a number of separate tools. Lattice Diamond is an integrated tool environment.
- All 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.

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 make sure you only see the data that is available for the current state in the process flow.

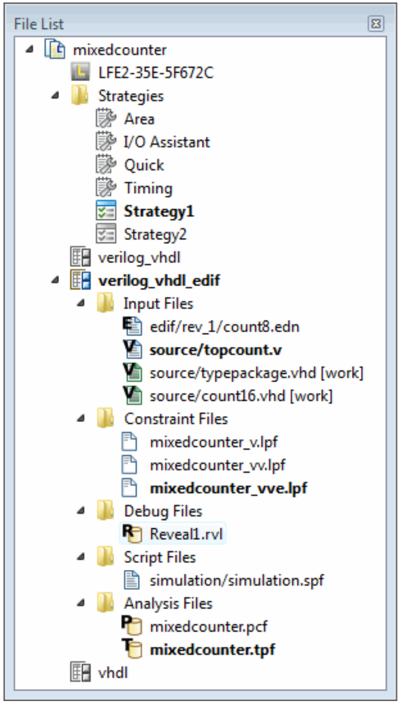
Shared memory is a key technology enabling many of the advanced functions in Lattice Diamond. Cross probing between tool views and faster process loops are a few of the benefits.

Project based environment

A project in Lattice Diamond consists of HDL source files, EDIF netlist files, LPF constraint files, Reveal debug files, script files for simulation and analysis files for power calculations and timing analysis, plus a target device and tool settings. The project data is organized into Implementations, defining the project structural elements, and Strategies, collections of tool settings.

Here is the File List showing the items in a sample project.

Figure 13: File List



```
File List
◢ mixedcounter
    LFE2-35E-5F672C
  ◢ Strategies
      Area
      I/O Assistant
      Quick
      Timing
      Strategy1
      Strategy2
    verilog_vhdl
  ◢ verilog_vhdl_edif
    ◢ Input Files
        edif/rev_1/count8.edn
        source/topcount.v
        source/typepackage.vhd [work]
        source/count16.vhd [work]
    ◢ Constraint Files
        mixedcounter_v.lpf
        mixedcounter_vv.lpf
        mixedcounter_vve.lpf
    ◢ Debug Files
        Reveal1.rvl
    ◢ Script Files
        simulation/simulation.spf
    ◢ Analysis Files
        mixedcounter.pcf
        mixedcounter.tpf
    vhdl
```

All of 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 may be set active or inactive. This is different than ispLever where the existence of Reveal debugger files meant that debug was active.

The project is the top level organizational element in Lattice Diamond and can contain multiple Implementations and multiple Strategies. If you want to have a Verilog version of your design, then you would make an Implementation consisting of only the Verilog source files. If you'd like another version of the design with primarily Verilog files but an EDIF netlist for one module, then you would create a new Implementation using the Verilog and EDIF source files. Same project, same design, just a different set of modular blocks.

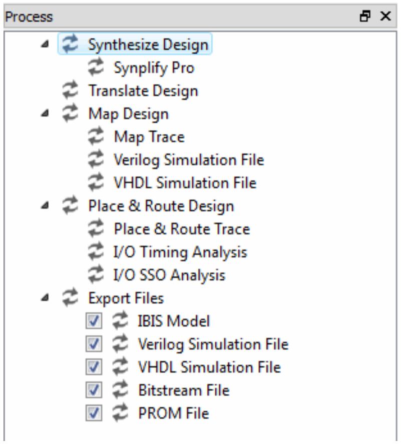
Similarly if you want to try different timing values you can create a new Strategy with the new timing parameters.

These multiple implementations and strategies are managed by being set 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 implementation design flow. Each major step in the design process is listed along with an icon showing its status. In the Export Files portion of the flow you can select which models or files you want to be exported. For example, if Bitstream File is checked then it will be generated and exported, if it is not checked it won't be generated and exported when the Export Files process step is run.

Figure 14: Process flow

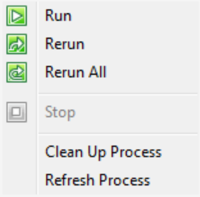


The status icons are defined as follows:

<img> TABLE

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

Figure 15: Process run pop-up menu

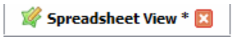


Shared memory

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 if you change a data element in one view of your design, all other views will see the change, whether or not the other views are active at the time of the change.

When project data has been changed, but not yet saved, an asterix * will display in the title tab of the view.

Figure 16: Title tab with changed memory indication



Spreadsheet View *

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

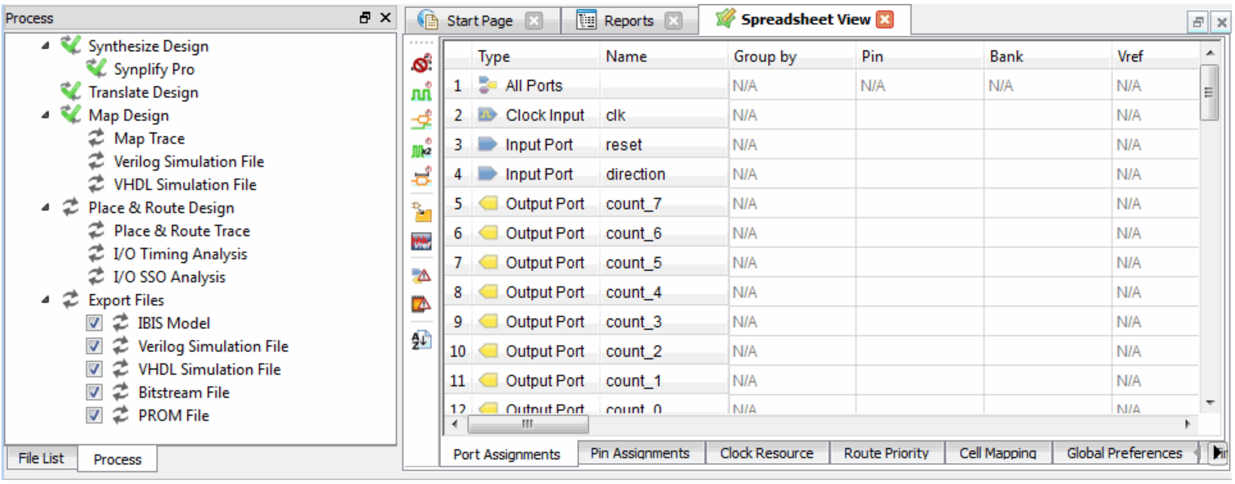
If a tool view becomes unavailable then the Lattice Diamond environment will need to be closed and restart

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 which process steps have been completed.

For example, the following sequence first shows the Process flow as complete through Map Design but not yet through Place & Route Design. Notice that the Spreadsheet view shows no pin assignments.

Figure 17: Process through Map Design

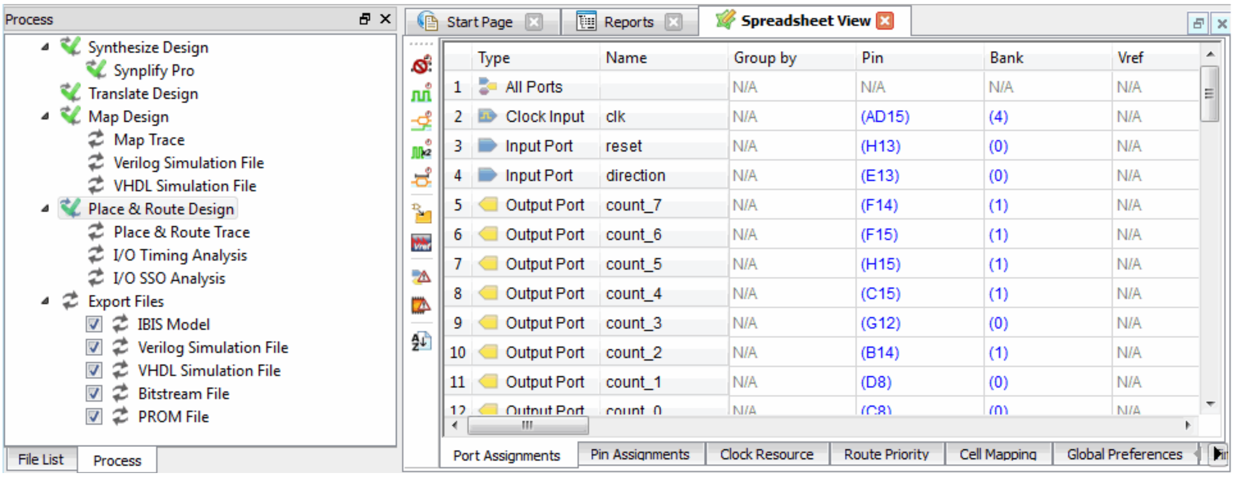


- Synthesize Design
  - Synplify Pro
- Translate Design
- Map Design
  - Map Trace
  - Verilog Simulation File
  - VHDL Simulation File
- Place & Route Design
  - Place & Route Trace
  - I/O Timing Analysis
  - I/O SSO Analysis
- Export Files
  - ☑ IBIS Model
  - ☑ Verilog Simulation File
  - ☑ VHDL Simulation File
  - ☑ Bitstream File
  - ☑ PROM File

	Type	Name	Group by	Pin	Bank	Vref
1	All Ports		N/A	N/A	N/A	N/A
2	Clock Input	clk	N/A			N/A
3	Input Port	reset	N/A			N/A
4	Input Port	direction	N/A			N/A
5	Output Port	count_7	N/A			N/A
6	Output Port	count_6	N/A			N/A
7	Output Port	count_5	N/A			N/A
8	Output Port	count_4	N/A			N/A
9	Output Port	count_3	N/A			N/A
10	Output Port	count_2	N/A			N/A
11	Output Port	count_1	N/A			N/A
12	Output Port	count_0	N/A			N/A

Now after Place & Route has completed, the Spreadsheet view shows that pins have been assigned.

Figure 18: Process through Place & Route Design

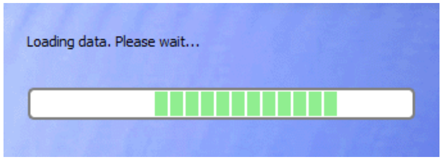


- Synthesize Design
  - Synplify Pro
- Translate Design
- Map Design
  - Map Trace
  - Verilog Simulation File
  - VHDL Simulation File
- Place & Route Design
  - Place & Route Trace
  - I/O Timing Analysis
  - I/O SSO Analysis
- Export Files
  - ☑ IBIS Model
  - ☑ Verilog Simulation File
  - ☑ VHDL Simulation File
  - ☑ Bitstream File
  - ☑ PROM File

	Type	Name	Group by	Pin	Bank	Vref
1	All Ports		N/A	N/A	N/A	N/A
2	Clock Input	clk	N/A	(AD15)	(4)	N/A
3	Input Port	reset	N/A	(H13)	(0)	N/A
4	Input Port	direction	N/A	(E13)	(0)	N/A
5	Output Port	count_7	N/A	(F14)	(1)	N/A
6	Output Port	count_6	N/A	(F15)	(1)	N/A
7	Output Port	count_5	N/A	(H15)	(1)	N/A
8	Output Port	count_4	N/A	(C15)	(1)	N/A
9	Output Port	count_3	N/A	(G12)	(0)	N/A
10	Output Port	count_2	N/A	(B14)	(1)	N/A
11	Output Port	count_1	N/A	(D8)	(0)	N/A
12	Output Port	count_0	N/A	(C8)	(0)	N/A

When you see a *Loading Data... Please Wait* message like this:

Figure 19: Loading Data



Loading data. Please wait...

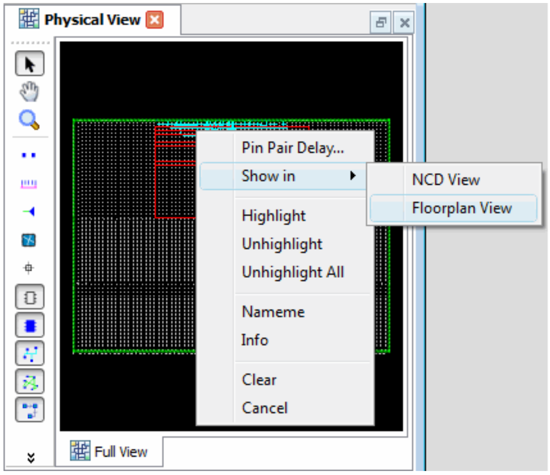
it means that a process has completed and the shared memory is being updated with new data. All tool views are dynamically updated when new data becomes available. Note that this means if a tool view is open and is displaying data and then process steps are re-run to an earlier point in the flow, the tool view will remain open but it will be grayed out 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.

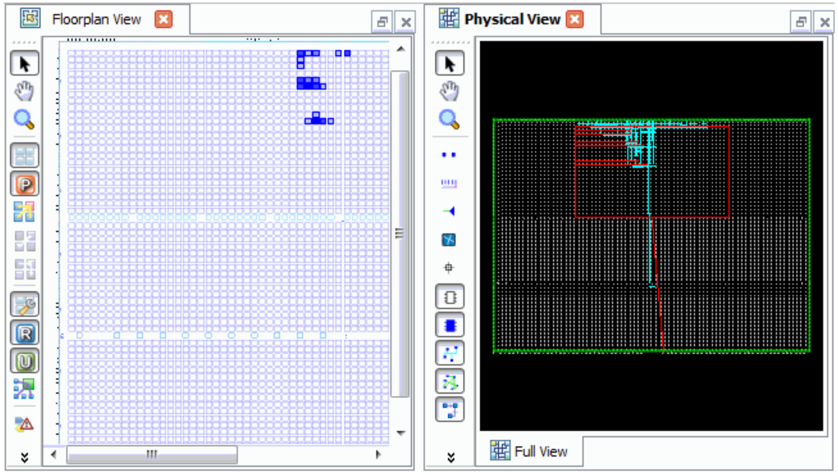
To see how this works select a pin or signal or port in one view and right-click on it. Select **Show In** and you will get a list of the potential cross probing views for the selected element.

Figure 20: Show In pop-up menu



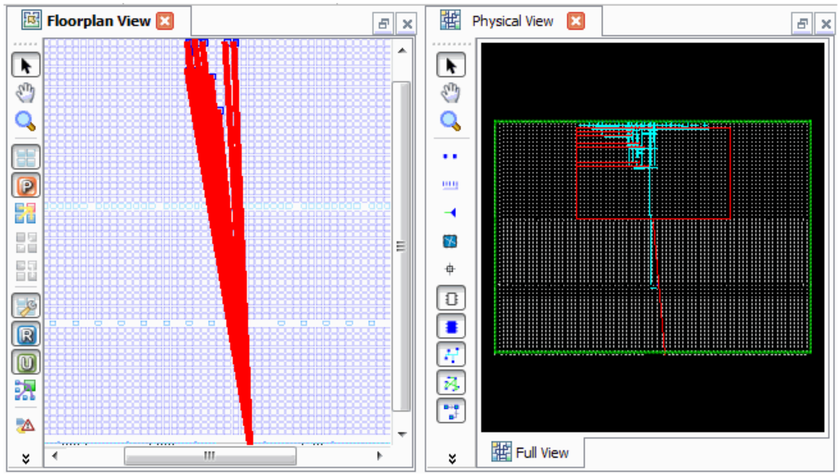
For example, here is a Physical View with a selected signal.

Figure 21: Physical View with selected signal



By right-clicking and selecting **Show In > Physical View**, we can see the same selected signal in the Physical View.

Figure 22: Selected signal 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.

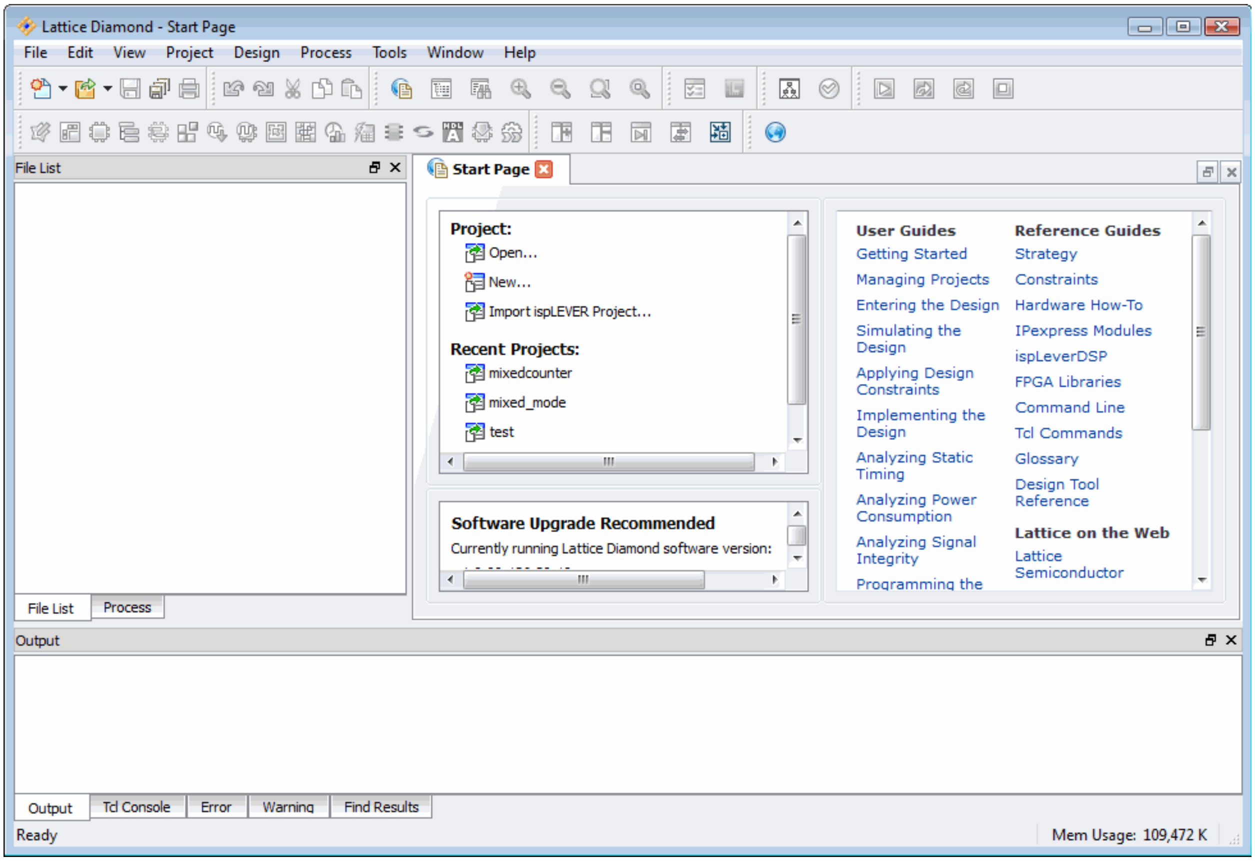
Overview

The Diamond Lattice user interface (UI) provides a comprehensive, integrated tool environment consisting of controls, views, reports, outputs and a TCL console. The UI is very flexible and configurable with the ability to store user 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. A good initial command to know is the **Window > Reset Layout** menu selection. If you ever want to quickly return to the default layout use this command.

Here is the main window of the UI shown in the default state:

Figure 23: Main Window



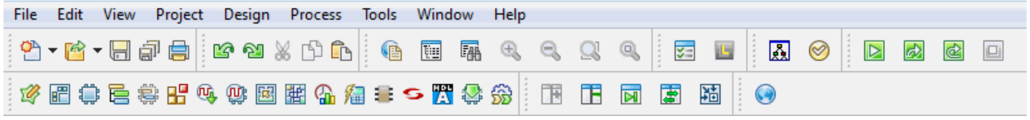
Software Upgrade Recommended

Currently running Lattice Diamond software version:

Menus and Toolbars

At the top of the main window is the menu and toolbar area. High level controls for starting tools, file and project management and controlling the display layout are contained here. All of the functionality in the toolbars is also found in the menus, plus the menus have some additional functions for system, project and toolbar control.

Figure 24: Menu and Toolbar area



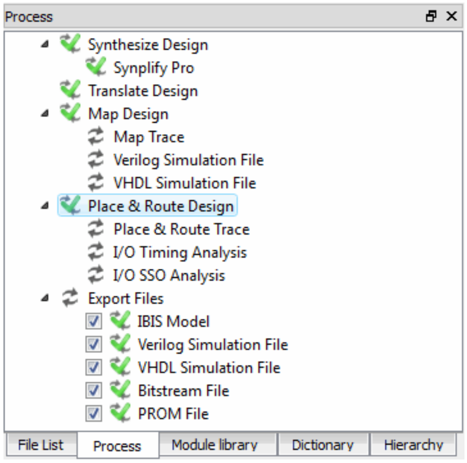
The toolbars are organized into functional sets. The display of each toolbar is controlled in the **View > Toolbars** menu and also by right-clicking in the Menu and Toolbar area. Toolbars may also be repositioned by dragging and dropping a toolbar to a new location.

<img> TABLE of toolbars

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



Process

- Synthesize Design
  - Synplify Pro
- Translate Design
- Map Design
  - Map Trace
  - Verilog Simulation File
  - VHDL Simulation File
- Place & Route Design
  - Place & Route Trace
  - I/O Timing Analysis
  - I/O SSO Analysis
- Export Files
  - ☑ IBIS Model
  - ☑ Verilog Simulation File
  - ☑ VHDL Simulation File
  - ☑ Bitstream File
  - ☑ PROM File

File List | Process | Module library | Dictionary | Hierarchy

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

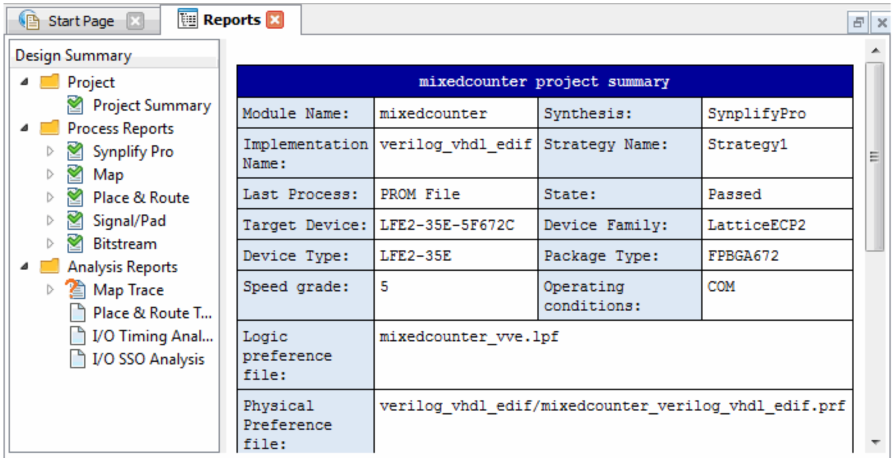
- **File List** - shows the files in the project organized by implementation and strategy. Note: this is not a hierarchical listing of the design.
- **Process** - shows the overall process flow and status for each step
- **Module library** - library of modules in the active implementation of the design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

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

Tool Views

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.

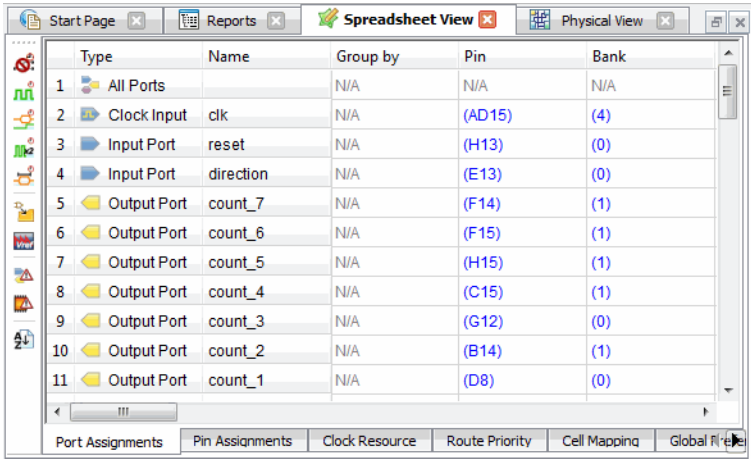
Figure 26: Tool View area



mixedcounter project summary			
Module Name:	mixedcounter	Synthesis:	SynplifyPro
Implementation Name:	verilog_vhdl_edif	Strategy Name:	Strategy1
Last Process:	PROM File	State:	Passed
Target Device:	LFE2-35E-5F672C	Device Family:	LatticeECP2
Device Type:	LFE2-35E	Package Type:	FPBGA672
Speed grade:	5	Operating conditions:	COM
Logic preference file:	mixedcounter_vve.lpf		
Physical Preference file:	verilog_vhdl_edif/mixedcounter_verilog_vhdl_edif.prf		

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

Figure 27: Multiple tools



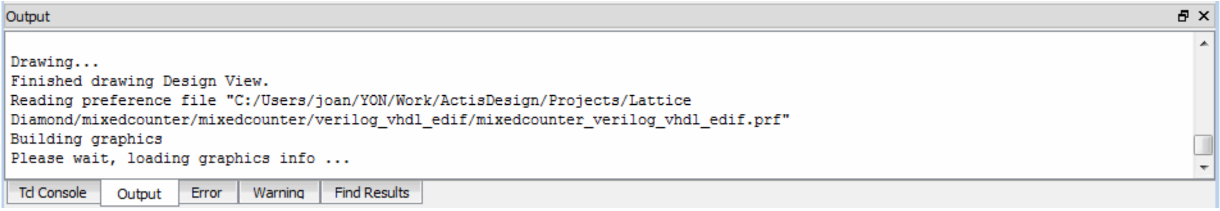
	Type	Name	Group by	Pin	Bank
1	All Ports		N/A	N/A	N/A
2	Clock Input	clk	N/A	(AD15)	(4)
3	Input Port	reset	N/A	(H13)	(0)
4	Input Port	direction	N/A	(E13)	(0)
5	Output Port	count_7	N/A	(F14)	(1)
6	Output Port	count_6	N/A	(F15)	(1)
7	Output Port	count_5	N/A	(H15)	(1)
8	Output Port	count_4	N/A	(C15)	(1)
9	Output Port	count_3	N/A	(G12)	(0)
10	Output Port	count_2	N/A	(B14)	(1)
11	Output Port	count_1	N/A	(D8)	(0)

Each tool view is specific to its tool and may contain additional toolbars, multiple panes or multiple windows controlled by additional tabs. The *Working with Tools and Views* chapter contains more information on each Tool and View.

Outputs and TCL Console

Near the bottom of the main window is the Outputs and TCL Console area.

Figure 28: Output and TCL Console area



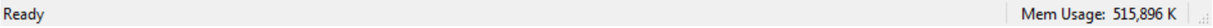
```
Drawing...
Finished drawing Design View.
Reading preference file "C:/Users/joan/YON/Work/ActisDesign/Projects/Lattice
Diamond/mixedcounter/mixedcounter/verilog_vhdl_edif/mixedcounter_verilog_vhdl_edif.prf"
Building graphics
Please wait, loading graphics info ...
```

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 tabs.

Cursor information and Memory Usage

At the very bottom of the main window is status information depending on cursor position and also the current memory usage. For more information on memory usage see *Advanced Topics*.

Figure 29: Cursor information and memory usage



Ready

Mem Usage: 515,896 K

Basic UI controls

Lattice Diamond is a modem design environment based on industry standard user interface concepts. The menus, toolbars and mouse clicks all behave in familiar ways. You can resize any of the window panes, drag and drop elements, right-click to see available actions on any design element and hover over objects to get pop-up descriptions.

All of the Project and Tool views as well as the Outputs and TCL Console items can be detached from the main UI window and operated as independent windows. You can simply grab the tool tab and drag it away from its position in the main window or you can select the detach button 🗗 .

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

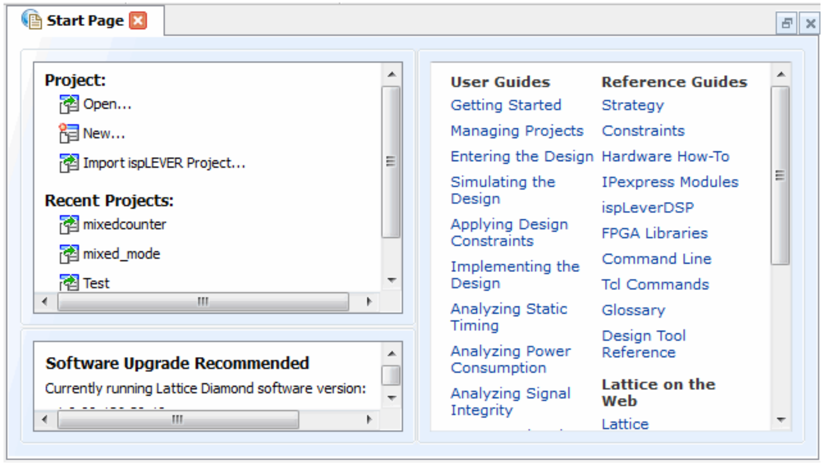
- Project views, Outputs and the TCL Console are attached using the double-click method. Double-click in the window title bar and the window will be attached back into the main UI window.
- Tool views are attached by the attach button method. Single click on the attach button ▣ and the window will be attached back into the main UI window.

Additionally there is an **Integrate All Tools** button ⊞ in the Window toolbar. This control takes all detached views and integrates them back into the main window.

Start Page

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

Figure 30: Start Page



Software Upgrade Recommended

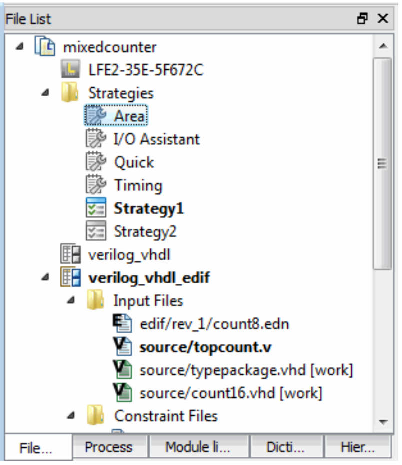
Currently running Lattice Diamond software version:

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. The File List is not a hierarchical listing of the design, but rather a list of all the design source, configuration and control files making 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**.

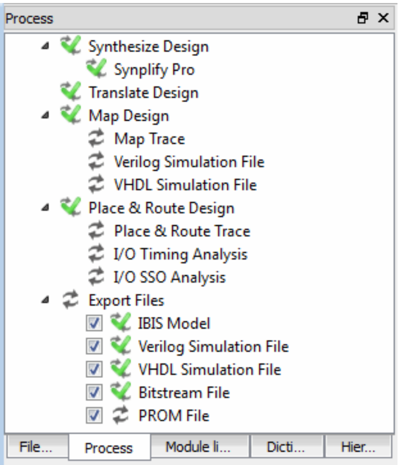
You can right-click on any file or item in the File List and a pop-up menu will give you the currently available actions for that item. The pop-up menu contents vary dependent on what type of item you select.

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

Process

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

Figure 32: Process View



Process

- ✓ Synthesize Design
  - ✓ Synplify Pro
- ✓ Translate Design
- ✓ Map Design
  - ⟳ Map Trace
  - ⟳ Verilog Simulation File
  - ⟳ VHDL Simulation File
- ✓ Place & Route Design
  - ⟳ Place & Route Trace
  - ⟳ I/O Timing Analysis
  - ⟳ I/O SSO Analysis
- ⟳ Export Files
  - ☑ ✓ IBIS Model
  - ☑ ✓ Verilog Simulation File
  - ☑ ✓ VHDL Simulation File
  - ☑ ✓ Bitstream File
  - ☑ ⟳ PROM File

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

<table w/images>

Initial

Completed

Warning

Error

Right-click on any step in the Process view and a menu will give you the currently available actions for that item.

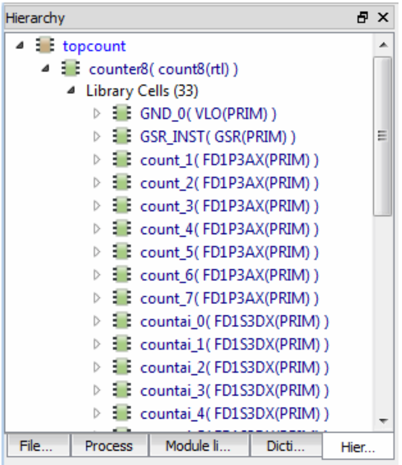
The Process view may be selected, closed, opened, attached (using the double-click method) and detached.

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. Its display is controlled from the **View > Show Views > Hierarchy** menu or by the pop-up View control menu resulting from right-clicking in the menu and toolbar area.

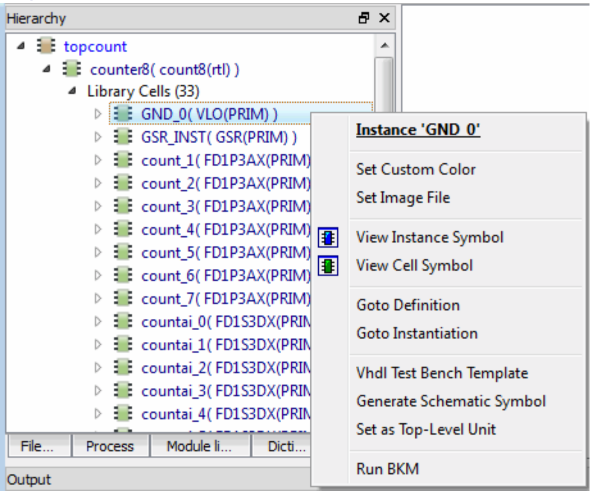
The Hierarchy view will be empty until the hierarchy for the design has been generated. Select the **Generate Hierarchy** function from the toolbar or the **Design** menu to generate the hierarchy and populate the Hierarchy view with data. In addition to displaying the HDL Diagram view, the action of generating the hierarchy causes the Hierarchy, Module Library and Dictionary project views to be displayed.

Figure 33: Hierarchy View



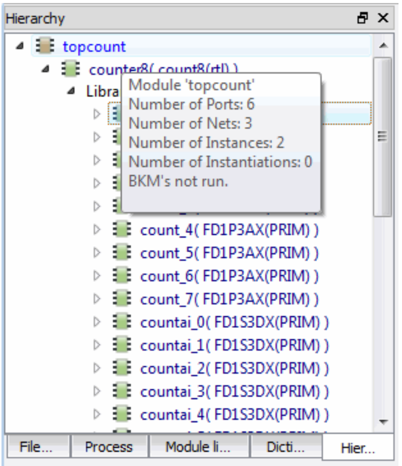
Right-click on any of the objects in the Hierarchy view to see its type and name as well as the possible actions.

Figure 34: Hierarchy item pop-up menu



Notice that if you hover the cursor over an item in the list a pop-up description of that item will be displayed.

Figure 35: Hierarchy View item description



Module 'topcount'
Number of Ports: 6
Number of Nets: 3
Number of Instances: 2
Number of Instantiations: 0
BKM's not run.

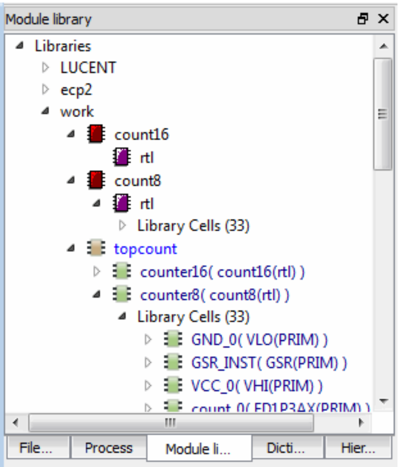
The amount of information shown in the Hierarchy view can be controlled by Tools > Options. The Hierarchy view may be selected, closed, opened, attached (through the double-click method) and detached.

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. Its display is controlled from the **View > Show Views > Module library** menu or by the pop-up View control menu resulting from right-clicking in the menu and toolbar area.

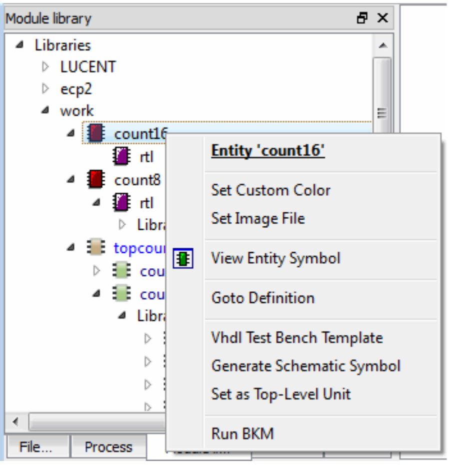
The Module library view will be empty until the hierarchy for the design has been generated. Select the **Generate Hierarchy** function from the toolbar or the **Design** menu to generate the hierarchy and populate the Module library view with data. In addition to displaying the HDL Diagram view, the action of generating the hierarchy causes the Hierarchy, Module Library and Dictionary project views to be displayed.

Figure 36: Module library view



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

Figure 37: Module item description and actions



The Module library view may be selected, closed, opened, attached (through the double-click method) and detached.

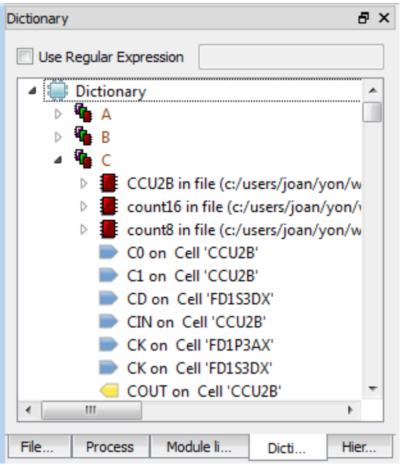
Dictionary

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. Its display is controlled from the **View > Show Views > Module library** menu or by the pop-up View control menu resulting from right-clicking in the menu and toolbar area.

The Dictionary view will be empty until the hierarchy for the design has been generated. Select the **Generate Hierarchy** function from the toolbar or the

Design menu to generate the hierarchy and populate the Dictionary view with data. In addition to displaying the HDL Diagram view, the action of generating the hierarchy causes the Hierarchy, Module Library and Dictionary project views to be displayed.

Figure 38: Dictionary view



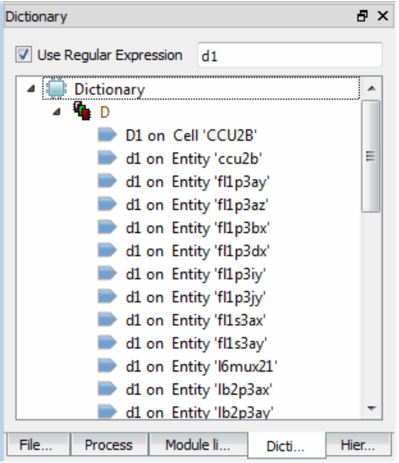
Dictionary

□ Use Regular Expression

Dictionary
A
B
C
CCU2B in file (c:/users/joan/yon/w
count16 in file (c:/users/joan/yon/
count8 in file (c:/users/joan/yon/w
C0 on Cell 'CCU2B'
C1 on Cell 'CCU2B'
CD on Cell 'FD1S3DX'
CIN on Cell 'CCU2B'
CK on Cell 'FD1P3AX'
CK on Cell 'FD1S3DX'
COUT on Cell 'CCU2B'

The Dictionary list can be filtered by selecting the *Regular Expression* box at the top of the view. When this is selected, a regular expression may be entered into the text box and the Dictionary list will display only those items matching the regular expression parameters. Note that you need to enter a keyboard return before the filtering will occur.

Figure 39: Dictionary view with regular expression filter



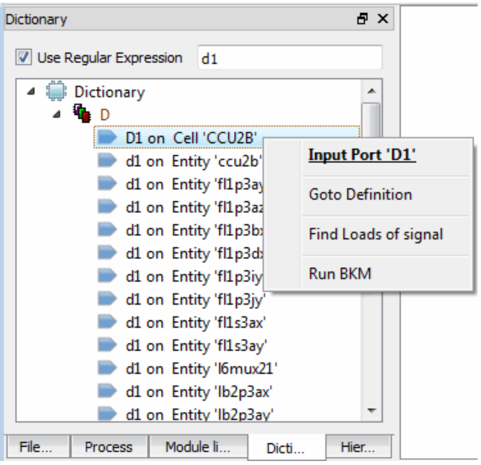
Dictionary

☑ Use Regular Expression d1

Dictionary
D
D1 on Cell 'CCU2B'
d1 on Entity 'ccu2b'
d1 on Entity 'fl1p3ay'
d1 on Entity 'fl1p3az'
d1 on Entity 'fl1p3bx'
d1 on Entity 'fl1p3dx'
d1 on Entity 'fl1p3iy'
d1 on Entity 'fl1p3jy'
d1 on Entity 'fl1s3ax'
d1 on Entity 'fl1s3ay'
d1 on Entity 'l6mux21'
d1 on Entity 'lb2p3ax'
d1 on Entity 'lb2p3ay'

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

Figure 40: Dictionary item pop-up

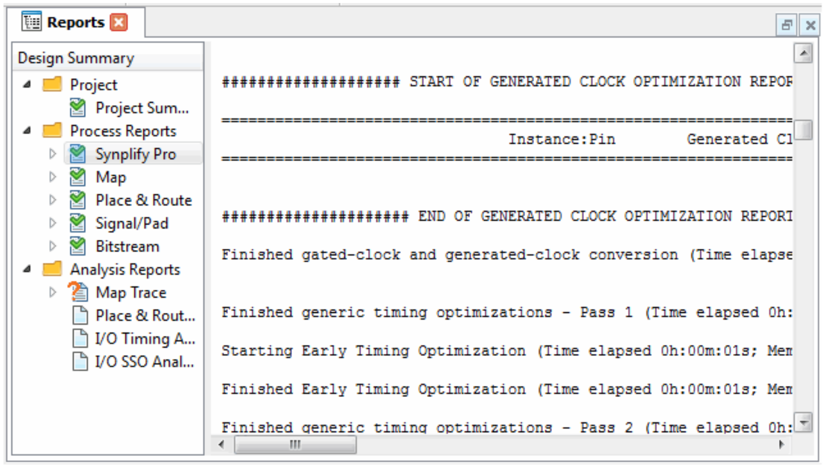


The Dictionary view may be selected, closed, opened, attached (through the double-click method) and detached.

Reports

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

Figure 41: Report view

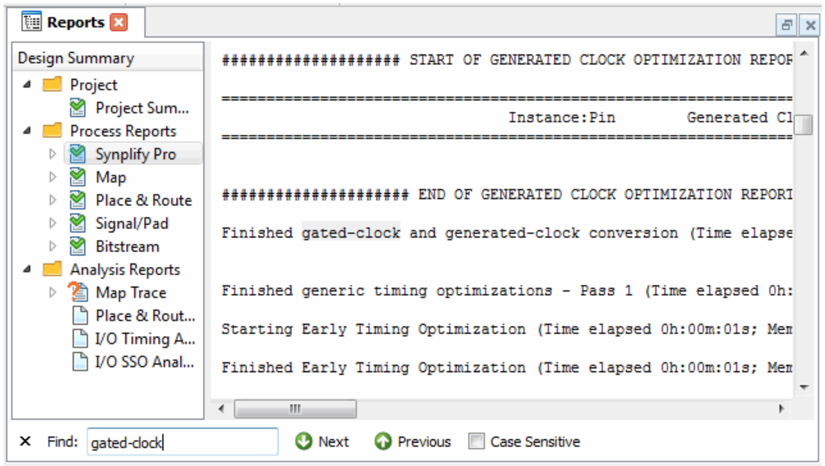


```
#################### START OF GENERATED CLOCK OPTIMIZATION REPOR

=================================================================
                                Instance:Pin        Generated Cl
=================================================================


##################### END OF GENERATED CLOCK OPTIMIZATION REPORT

Finished gated-clock and generated-clock conversion (Time elapse


Finished generic timing optimizations - Pass 1 (Time elapsed 0h:

Starting Early Timing Optimization (Time elapsed 0h:00m:01s; Mem

Finished Early Timing Optimization (Time elapsed 0h:00m:01s; Mem

Finished generic timing optimizations - Pass 2 (Time elapsed 0h:
```

The left side of the Reports view displays a Design Summary containing Report information for each step of the design process flow. The right side of the Report display shows the report for the selected step.

Right-click on a report in the Design Summary list to see the**Find in Text** function. Selecting this function brings up a text search area at the bottom of the Report view.

Figure 42: Report view find text



```
#################### START OF GENERATED CLOCK OPTIMIZATION REPOR

===================================================================
                                Instance:Pin        Generated Cl
===================================================================


#################### END OF GENERATED CLOCK OPTIMIZATION REPORT

Finished gated-clock and generated-clock conversion (Time elapse


Finished generic timing optimizations - Pass 1 (Time elapsed 0h:

Starting Early Timing Optimization (Time elapsed 0h:00m:01s; Mem

Finished Early Timing Optimization (Time elapsed 0h:00m:01s; Mem
```

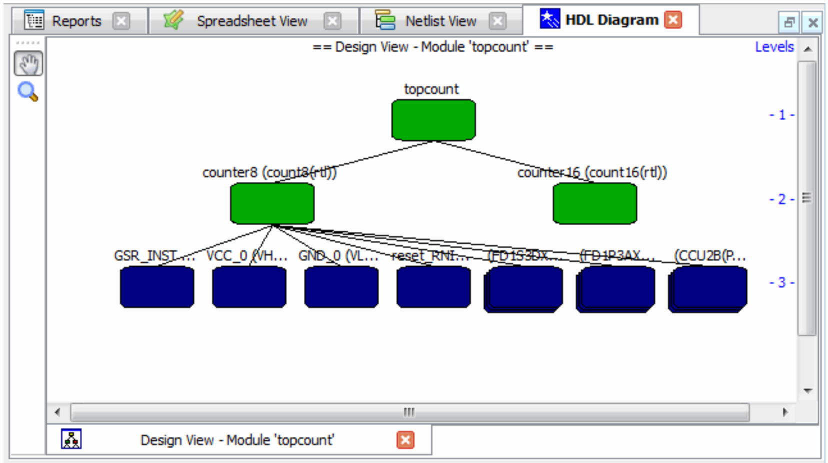
Find: gated-clock

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

Tool Views

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

Figure 43: Tool view area

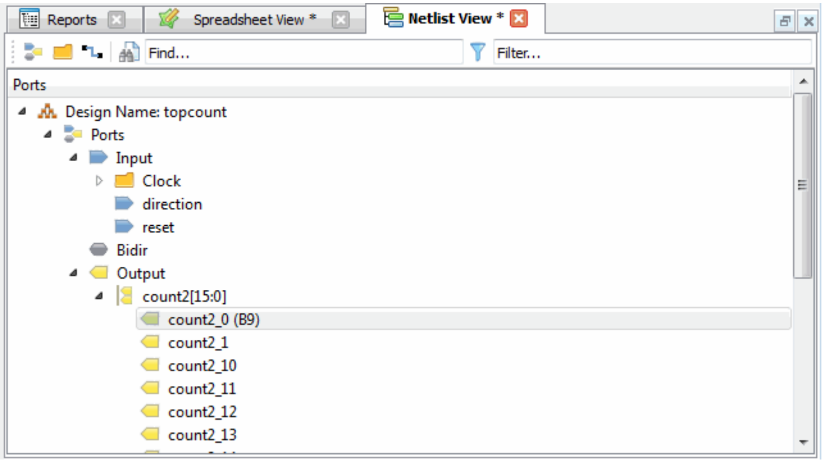


When multiple tools are active their display can be controlled with the tab group functions in the tab toolbar. See the Common Tasks section of this chapter for more information on tab group functions.

Each tool view is specific to its tool and may contain additional toolbars, multiple panes or multiple windows controlled by additional tabs. The *Working with Tools and Views* chapter contains information on each Tool and View plus more details on controlling their display.

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

Figure 44: Tool view title showing changed data

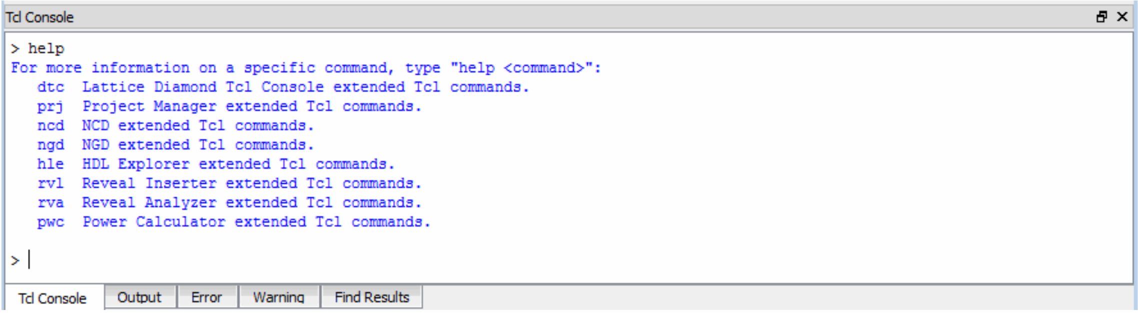


The Tool views may be selected, closed, opened, attached (through the attach icon method) and detached.

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 45: TCL Console



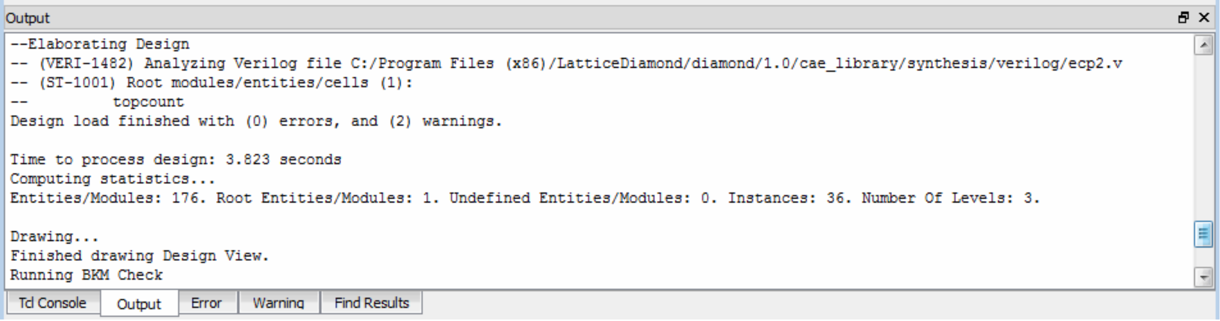
```
> help
For more information on a specific command, type "help <command>":
   dtc  Lattice Diamond Tcl Console extended Tcl commands.
   prj  Project Manager extended Tcl commands.
   ncd  NCD extended Tcl commands.
   ngd  NGD extended Tcl commands.
   hle  HDL Explorer extended Tcl commands.
   rvl  Reveal Inserter extended Tcl commands.
   rva  Reveal Analyzer extended Tcl commands.
   pwc  Power Calculator extended Tcl commands.

> |
```

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

Output

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

Figure 46: Output area



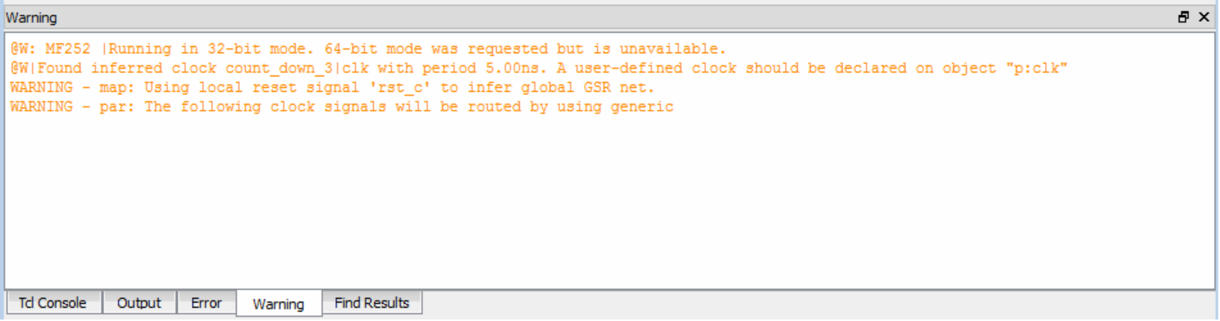
```
--Elaborating Design
-- (VERI-1482) Analyzing Verilog file C:/Program Files (x86)/LatticeDiamond/diamond/1.0/cae_library/synthesis/verilog/ecp2.v
-- (ST-1001) Root modules/entities/cells (1):
--          topcount
Design load finished with (0) errors, and (2) warnings.

Time to process design: 3.823 seconds
Computing statistics...
Entities/Modules: 176. Root Entities/Modules: 1. Undefined Entities/Modules: 0. Instances: 36. Number Of Levels: 3.

Drawing...
Finished drawing Design View.
Running BKM Check
```

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

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 and 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.

Figure 47: Error and Warning display



```
@W: MF252 |Running in 32-bit mode. 64-bit mode was requested but is unavailable.
@W|Found inferred clock count_down_3|clk with period 5.00ns. A user-defined clock should be declared on object "p:clk"
WARNING - map: Using local reset signal 'rst_c' to infer global GSR net.
WARNING - par: The following clock signals will be routed by using generic
```

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

Common Tasks

The UI controls many tools and processes. A good understanding of the features and basic operations will allow you to explore the rich functionality contained within the Lattice Diamond design environment. The following sections describe some of the more commonly performed tasks.

Controlling Views

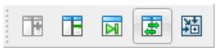
Although there are many different types of views containing widely varied information, they all are controlled similarly. Here are a few of the most common operations you can do with views:

- Open - use the **View > Show Views** menu selections or right-click in the menu or toolbar areas to open a view
- Select - if a view is already open you can select its tab to select it and bring it to the front
- Close - select the x in the upper right corner of the view, or right-click in the menu or toolbar area to toggle the view
- Detach - select the detach tool button in the upper right corner of the view
- Attach - use one of these two methods:
    - Project views, Outputs and the TCL Console are attached using the double-click method. Double-click in the window title bar and the window will be attached back into the main UI window.
    - Tool views are attached by the attach button method. Single click on the attach button and the window will be attached back into the main UI window.
- Move - click on a view's tab and use drag and drop to move a view's position within the open views

Grouping Tabs

The tab grouping controls are in the window toolbar.

Figure 48: Window toolbar



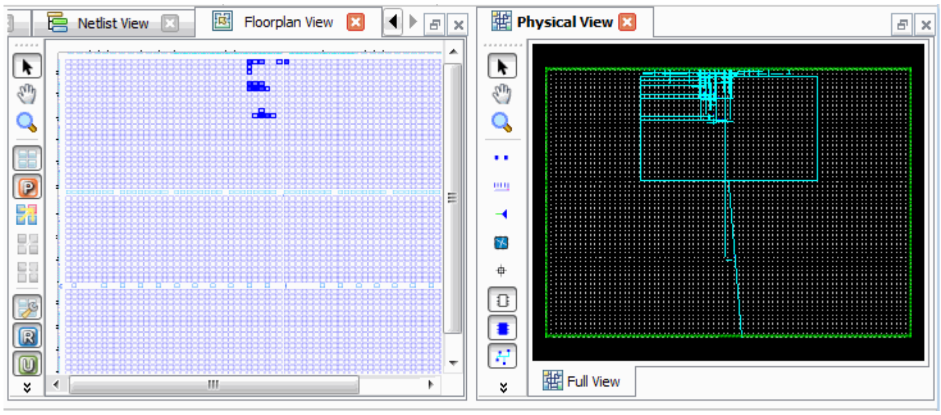
The controls work as follows:

- Split Tab Group - displays two views side by side
- Merge Tab Group - goes from a split tab group back to one primary view
- Move to another Tab Group - moves the current selected tab to another tab group
- Switch Tab Group Position - switches the positions between the two views in a split tab group
- 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 displayed views.

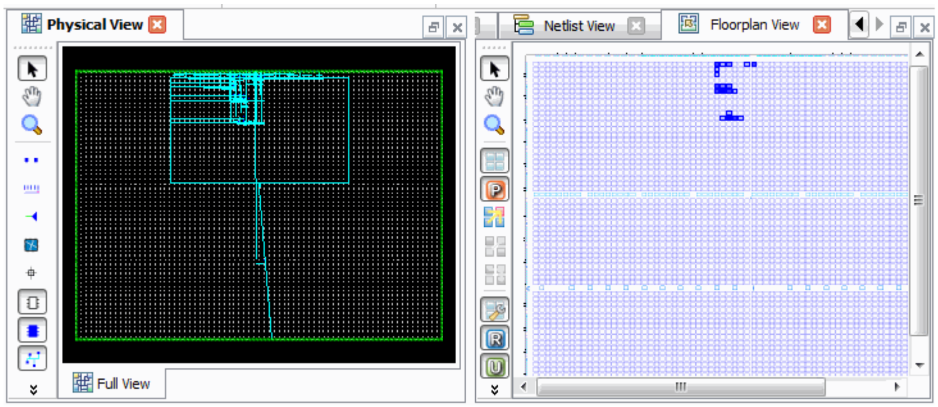
Here is the display after selecting **Split Tab Group**.

Figure 49: Split Tab Group



You can switch the position of the tool views within a tab group using the **Switch Tab Group Position** control.

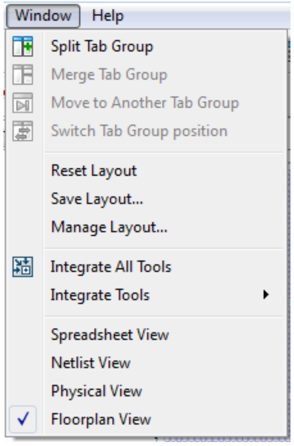
Figure 50: Switch Tab Group Position



Managing Layouts

When you have the UI set up the way you like you can save the layout. You can save as many layouts as you want, giving each one a unique name. By contrast, at any time you can easily reset the layout to the factory default. The controls for layout management are in the **Window** menu.

Figure 51: Window menu



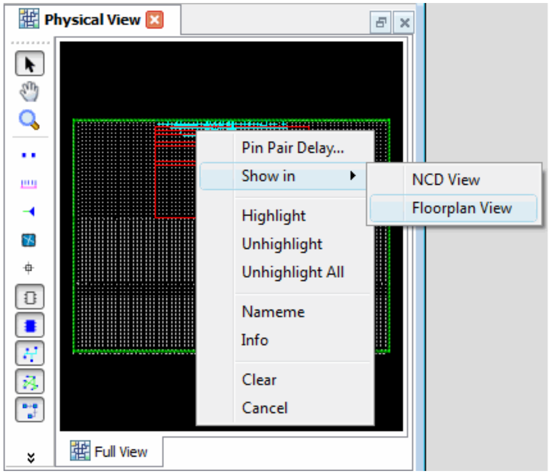
The **Window > Save Layout** selection allows you to name a layout, save it and optionally set it to be the default layout when Lattice Diamond is launched. When you save a layout there is an option to *Launch view when loaded*. If this option is selected it can take some time to load a new layout if the layout contains many tool views.

The **Window > Reset Layout** selection resets the layout to the factory default. The **Window > Manage Layouts** selection allows you to preview your saved layouts and modify or delete them and also select one to use at launch.

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 on a selected object and if cross probing is available for that object you will see a **Show In** selection in the menu selections. The **Show In** selection is available for most tool views.

Figure 52: Show In



If you select a view that is not open it will be opened automatically.

5

Working with Projects

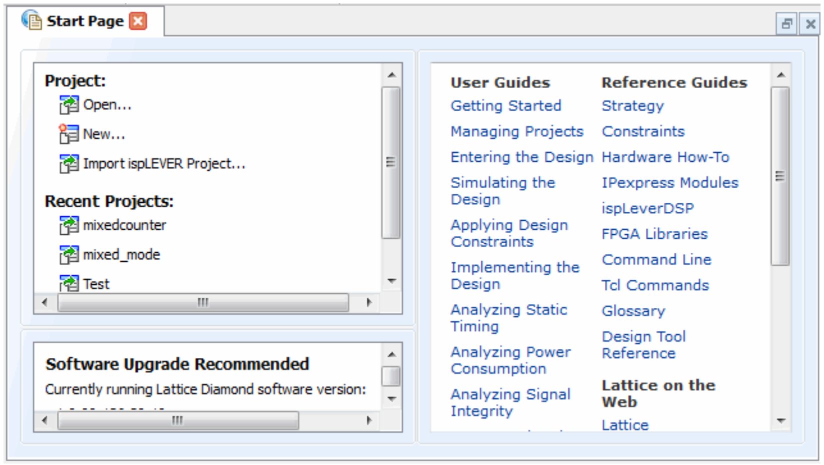
This chapter covers projects and their elements. Implementations and strategies are explained and some common project tasks are shown.

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. Detailed instructions on creating a new project are in the *Getting Started* chapter.

Figure 53: Open a project from the Start Page

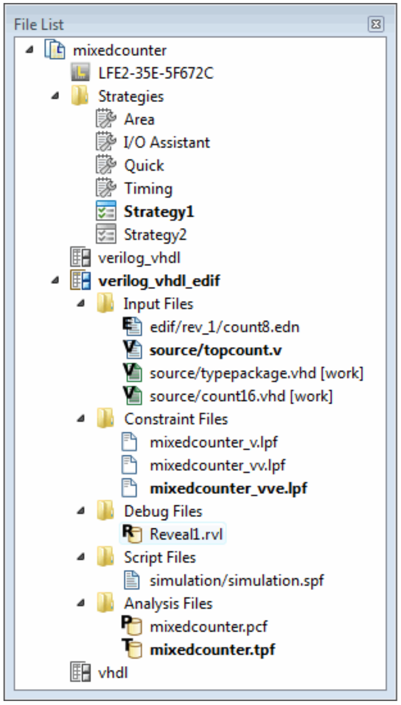


Software Upgrade Recommended

Currently running Lattice Diamond software version:

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

Figure 54: Project files in the File List

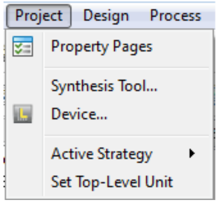


File List

- mixedcounter
  - LFE2-35E-5F672C
  - Strategies
    - Area
    - I/O Assistant
    - Quick
    - Timing
    - **Strategy1**
    - Strategy2
  - verilog_vhdl
  - **verilog_vhdl_edif**
    - Input Files
      - edif/rev_1/count8.edn
      - **source/topcount.v**
      - source/typepackage.vhd [work]
      - source/count16.vhd [work]
    - Constraint Files
      - mixedcounter_v.lpf
      - mixedcounter_vv.lpf
      - **mixedcounter_vve.lpf**
    - Debug Files
      - Reveal1.rvl
    - Script Files
      - simulation/simulation.spf
    - Analysis Files
      - mixedcounter.pcf
      - **mixedcounter.tpf**
  - vhdl

The Project menu contains selections to show 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 55: Project menu



Implementations

Implementations define the design structural elements as well as the constraint and analysis parameters for a project. An Implementation is the structure of a design and can be thought of as *what* is in the design. For example, one Implementation may use inferred memory and another Implementation may use instantiated memory.

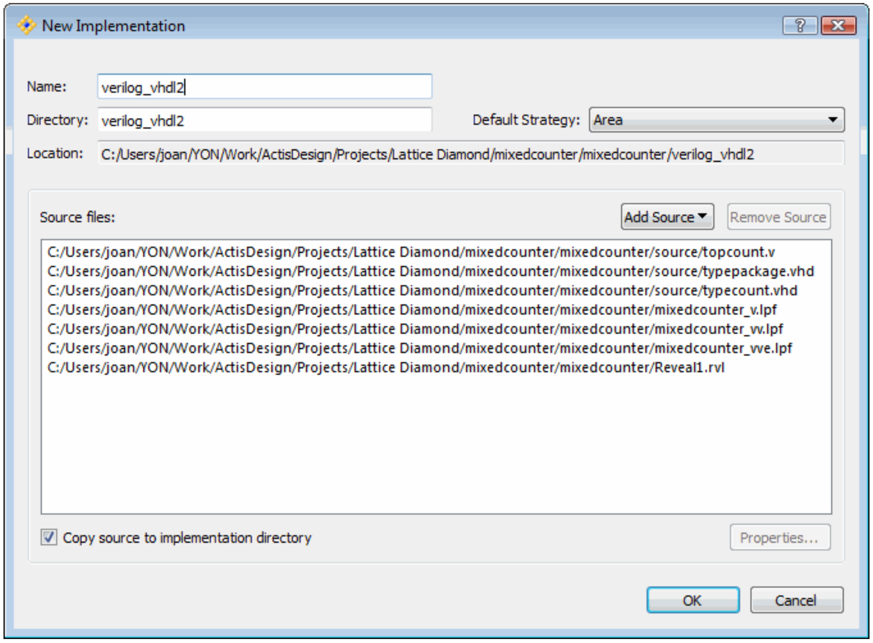
There can be multiple Implementations in a project, but only one Implementation can be active at a time. And there must be one active implementation. Every Implementation has an associated active Strategy. Strategies are a shared pool of resources for all implementations and are discussed in the next section. An Implementation is created whenever you create a new project. Implementations consist of:

- Input files

- Constraint files
- Debug files
- Script files
- Analysis 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** to bring up the New Implementation dialog. In the New Implementation dialog you can set the Implementation name, directory, default strategy and also add source files. When you select **Add Source** you have a choice of browsing for the source files or using source from an existing implementation.

Figure 56: New Implementation



Name: verilog_vhdl2

Directory: verilog_vhdl2

Default Strategy: Area

Location: C:/Users/joan/YON/Work/ActisDesign/Projects/Lattice Diamond/mixedcounter/mixedcounter/verilog_vhdl2

Source files:

```
C:/Users/joan/YON/Work/ActisDesign/Projects/Lattice Diamond/mixedcounter/mixedcounter/source/topcount.v
C:/Users/joan/YON/Work/ActisDesign/Projects/Lattice Diamond/mixedcounter/mixedcounter/source/typepackage.vhd
C:/Users/joan/YON/Work/ActisDesign/Projects/Lattice Diamond/mixedcounter/mixedcounter/source/typecount.vhd
C:/Users/joan/YON/Work/ActisDesign/Projects/Lattice Diamond/mixedcounter/mixedcounter/mixedcounter_v.lpf
C:/Users/joan/YON/Work/ActisDesign/Projects/Lattice Diamond/mixedcounter/mixedcounter/mixedcounter_vv.lpf
C:/Users/joan/YON/Work/ActisDesign/Projects/Lattice Diamond/mixedcounter/mixedcounter/mixedcounter_vve.lpf
C:/Users/joan/YON/Work/ActisDesign/Projects/Lattice Diamond/mixedcounter/mixedcounter/Reveal1.rvl
```

☑ Copy source to implementation directory

Notice that you have the choice 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 this option is not selected.

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

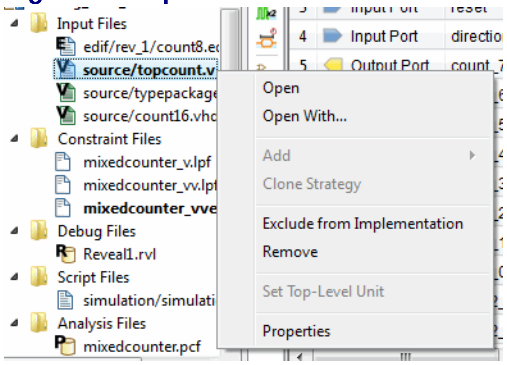
To add a file to an Implementation right-click on the Implementation name or on any of the file folders in the Implementation and select **Add > New File** or **Add > Existing File**.

Input Files

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

Right-click on an input file to bring up a pop up menu with the possible actions for that file.

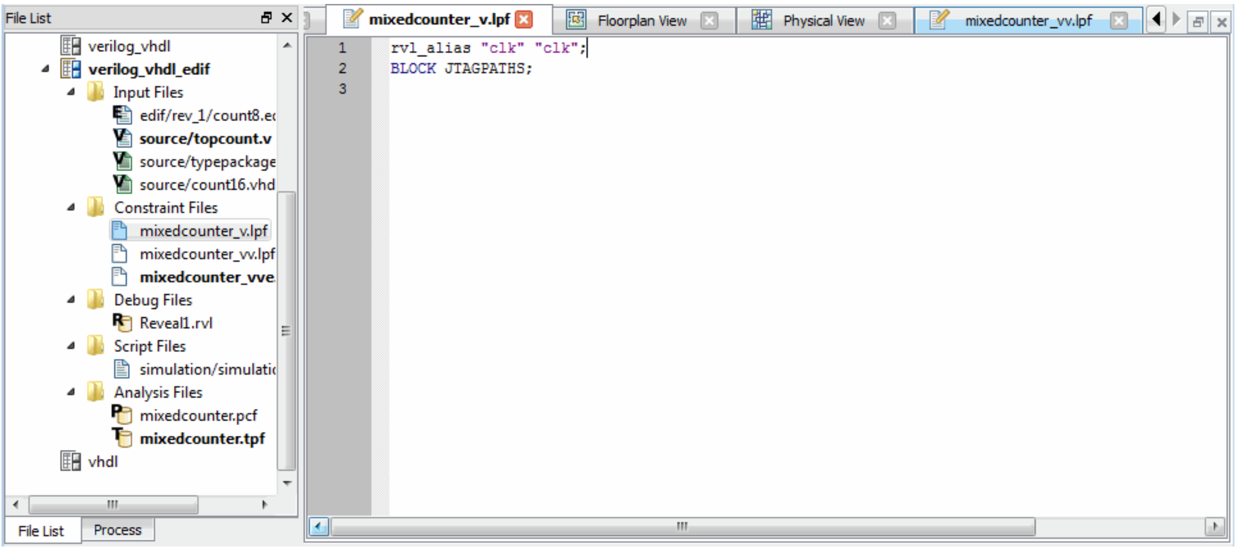
Figure 57: Input file actions



Constraint Files

Constraint files contain the design constraints for the implementation. These are the LPF files. An Implementation can have multiple Constraint files, but one and only one can be active at a time.

Figure 58: Constraint file

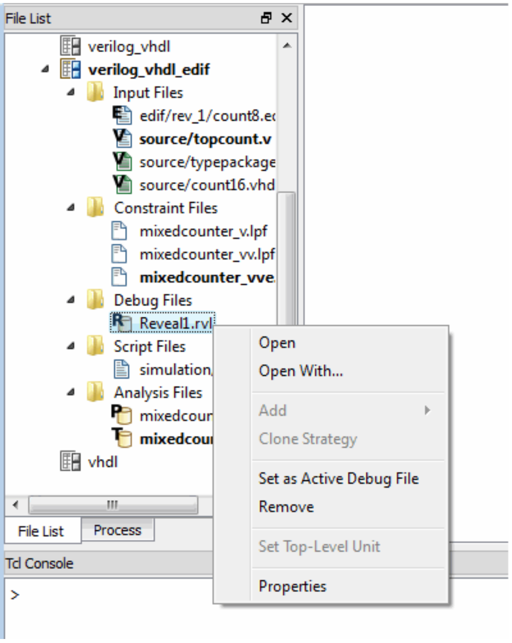


```
rvl_alias "clk" "clk";
BLOCK JTAGPATHS;
```

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 active. To insert hardware debug into your design right-click on a debug file and select **Set as Active Debug File** in the pop-up menu. Remember **bold** indicates if an element is active.

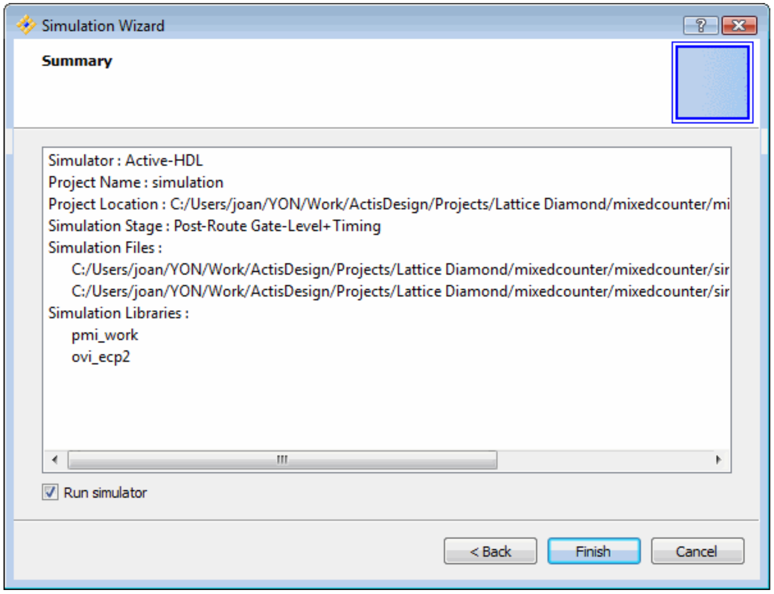
Figure 59: Reveal debug file actions



Script Files

The Script Files folder contains scripts generated by the Simulation Wizard. Once you have run the Simulation Wizard the steps are stored in a script file which may be used to control launching the simulator.

Figure 60: Simulation script file



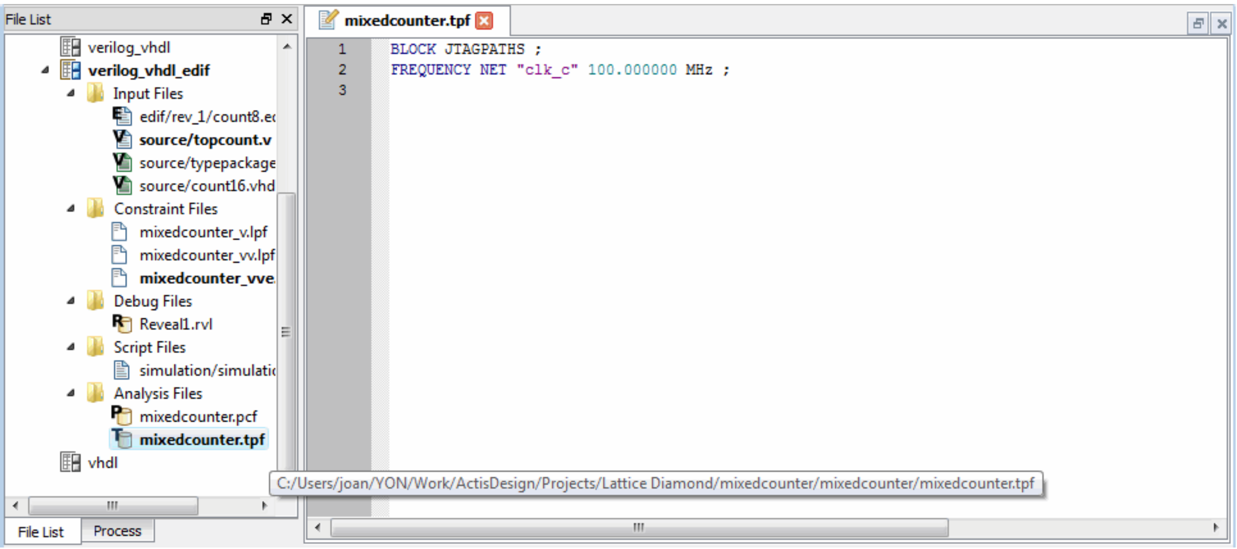
```
Simulator : Active-HDL
Project Name : simulation
Project Location : C:/Users/joan/YON/Work/ActisDesign/Projects/Lattice Diamond/mixedcounter/mi
Simulation Stage : Post-Route Gate-Level+Timing
Simulation Files :
    C:/Users/joan/YON/Work/ActisDesign/Projects/Lattice Diamond/mixedcounter/mixedcounter/sir
    C:/Users/joan/YON/Work/ActisDesign/Projects/Lattice Diamond/mixedcounter/mixedcounter/sir
Simulation Libraries :
    pmi_work
    ovi_ecp2
```

☑ Run simulator

Analysis Files

The Analysis Files folder contains Power Calculator (pcf) and Timing Analysis (tpf) files. There can be multiple analysis files and one or none can be set active. Whether these files are active or not effects the behavior of their associated tool views.

Figure 61: Timing analysis file



```
verilog_vhdl
verilog_vhdl_edif
    Input Files
        edif/rev_1/count8.ec
        source/topcount.v
        source/typepackage
        source/count16.vhd
    Constraint Files
        mixedcounter_v.lpf
        mixedcounter_vv.lpf
        mixedcounter_vve.
    Debug Files
        Reveal1.rvl
    Script Files
        simulation/simulatic
    Analysis Files
        mixedcounter.pcf
        mixedcounter.tpf
vhdl
```

```
BLOCK JTAGPATHS ;
FREQUENCY NET "clk_c" 100.000000 MHz ;
```

C:/Users/joan/YON/Work/ActisDesign/Projects/Lattice Diamond/mixedcounter/mixedcounter/mixedcounter.tpf

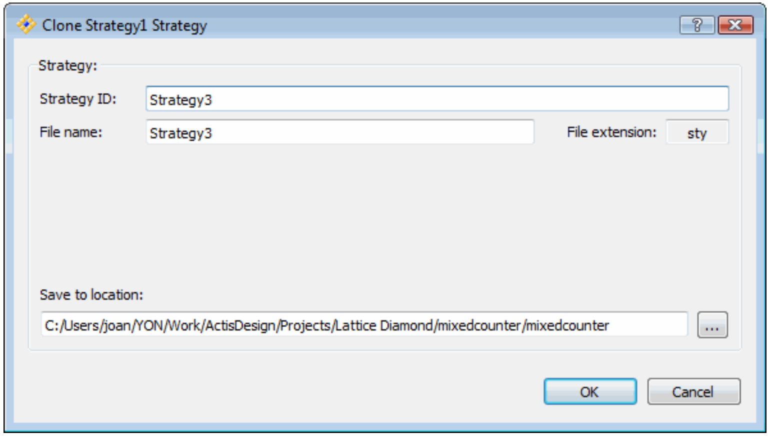
Strategies

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.

There are four predefined Strategies in Lattice Diamond plus customized user Strategies. Predefined Strategies cannot be edited, but they can be cloned and then modified and saved as custom user strategies. Predefined Strategies can be set as the active strategy. Custom user strategies can be edited, cloned, set as the active strategy and removed. All of the Strategies are available to all of the Implementations.

To create a new strategy right-click on an existing strategy and select the **Clone <strategy name> Strategy** selection. You can set the new strategy's ID and file name.

Figure 62: Clone to create a new strategy



Strategy ID: Strategy3

File name: Strategy3

File extension: sty

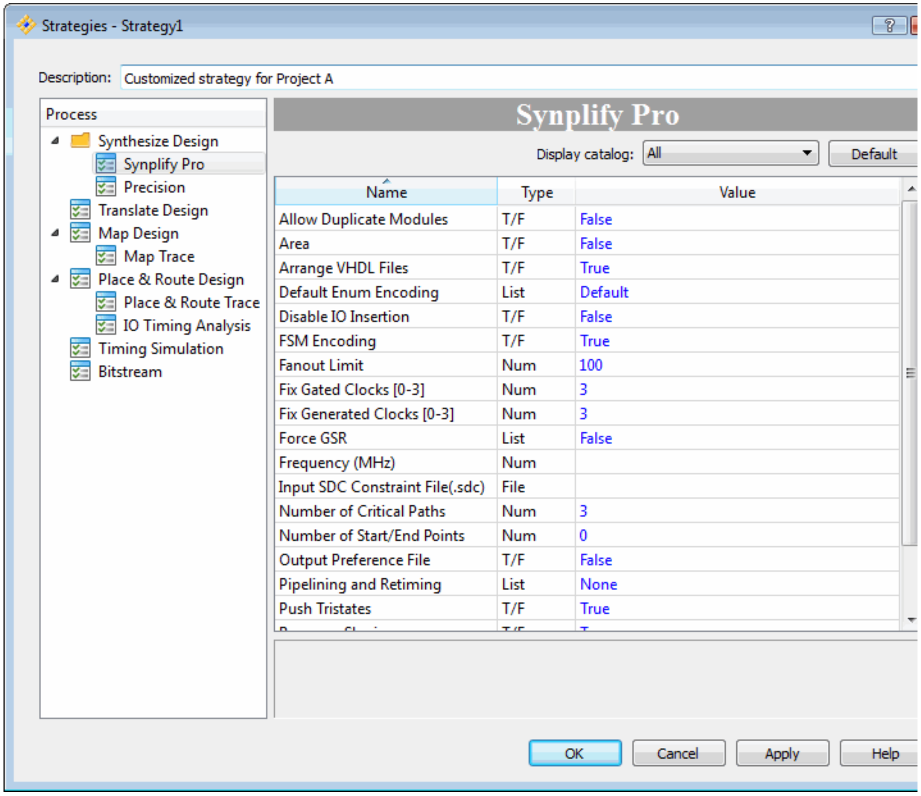
Save to location: C:/Users/joan/YON/Work/ActisDesign/Projects/Lattice Diamond/mixedcounter/mixedcounter

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

To change the settings in a strategy:

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

Figure 63: Strategy settings



Strategies - Strategy1

Description: Customized strategy for Project A

Process
- Synthesize Design
  - Synplify Pro
  - Precision
- Translate Design
- Map Design
  - Map Trace
- Place & Route Design
  - Place & Route Trace
  - IO Timing Analysis
- Timing Simulation
- Bitstream

Synplify Pro

Display catalog: All

Name	Type	Value
Allow Duplicate Modules	T/F	False
Area	T/F	False
Arrange VHDL Files	T/F	True
Default Enum Encoding	List	Default
Disable IO Insertion	T/F	False
FSM Encoding	T/F	True
Fanout Limit	Num	100
Fix Gated Clocks [0-3]	Num	3
Fix Generated Clocks [0-3]	Num	3
Force GSR	List	False
Frequency (MHz)	Num	
Input SDC Constraint File(.sdc)	File	
Number of Critical Paths	Num	3
Number of Start/End Points	Num	0
Output Preference File	T/F	False
Pipelining and Retiming	List	None
Push Tristates	T/F	True

The default options are listed in blue font color.

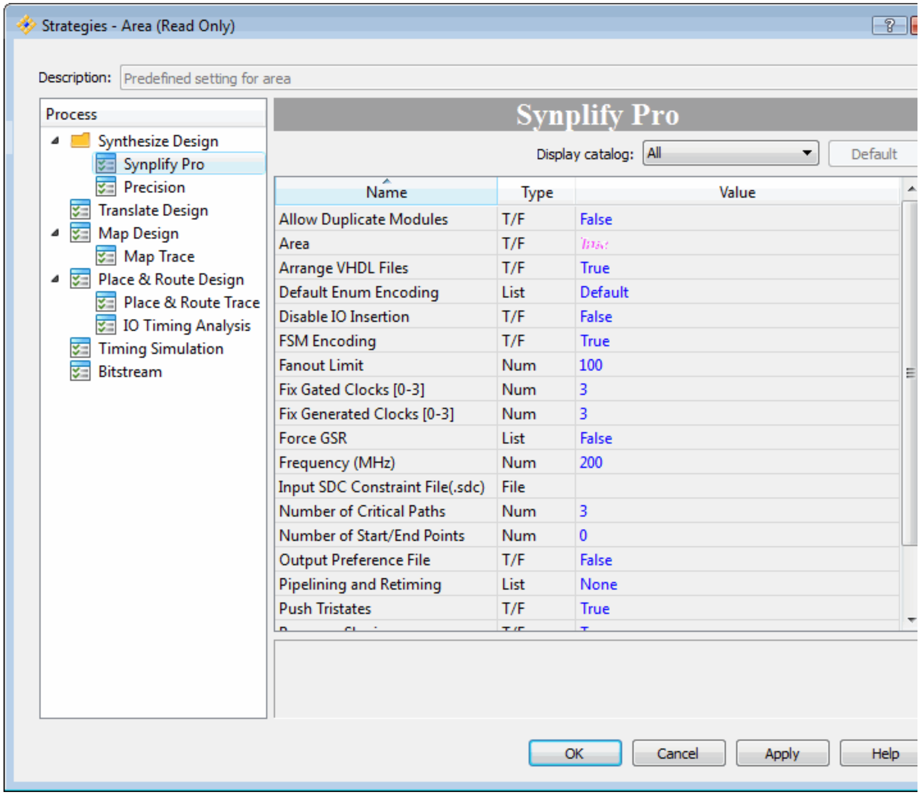
Strategies are design data independent and can be exported and used in multiple projects.

Area

The Area Strategy is a predefined Strategy for area optimization. This Strategy tries to minimize area 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 may cause some difficulties in the place and route process with longer time or incomplete routing.

Figure 64: Area predefined strategy



Description: Predefined setting for area

Display catalog: All

Name	Type	Value
Allow Duplicate Modules	T/F	False
Area	T/F	True
Arrange VHDL Files	T/F	True
Default Enum Encoding	List	Default
Disable IO Insertion	T/F	False
FSM Encoding	T/F	True
Fanout Limit	Num	100
Fix Gated Clocks [0-3]	Num	3
Fix Generated Clocks [0-3]	Num	3
Force GSR	List	False
Frequency (MHz)	Num	200
Input SDC Constraint File(.sdc)	File	
Number of Critical Paths	Num	3
Number of Start/End Points	Num	0
Output Preference File	T/F	False
Pipelining and Retiming	List	None
Push Tristates	T/F	True

I/O Assistant

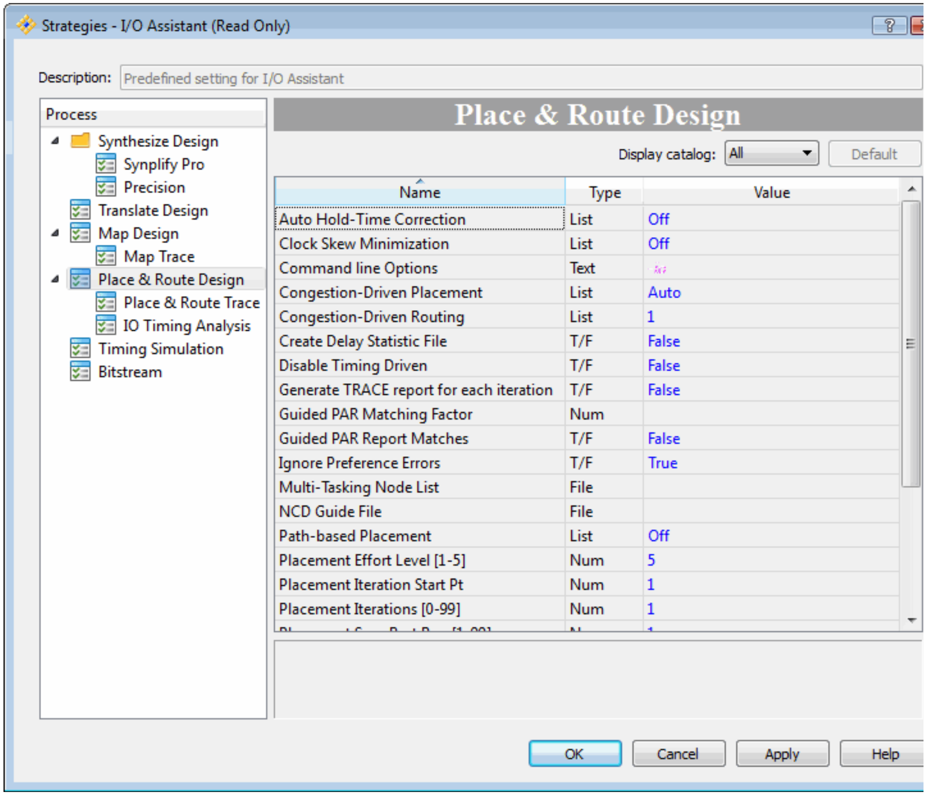
The I/O Assistant strategy is a predefined Strategy useful for I/O design. This Strategy is used to place I/O efficiently for your design at an early stage. It helps to check if you set a legal I/O location that affects board-level pin-outs.

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

Note that if you use the I/O Assistant Strategy for your project, the generated NCD file is incomplete. Running the Export Files > Bitstream File or Export Files > JEDEC File process may fail.

If you want to implement a complete design, you need to choose another strategy and rerun all processes again. See the *Lattice Diamond Design Flow* chapter for more information.

Figure 65: I/O Assistant predefined Strategy



Strategies - I/O Assistant (Read Only)

Description: Predefined setting for I/O Assistant

Place & Route Design

Display catalog: All

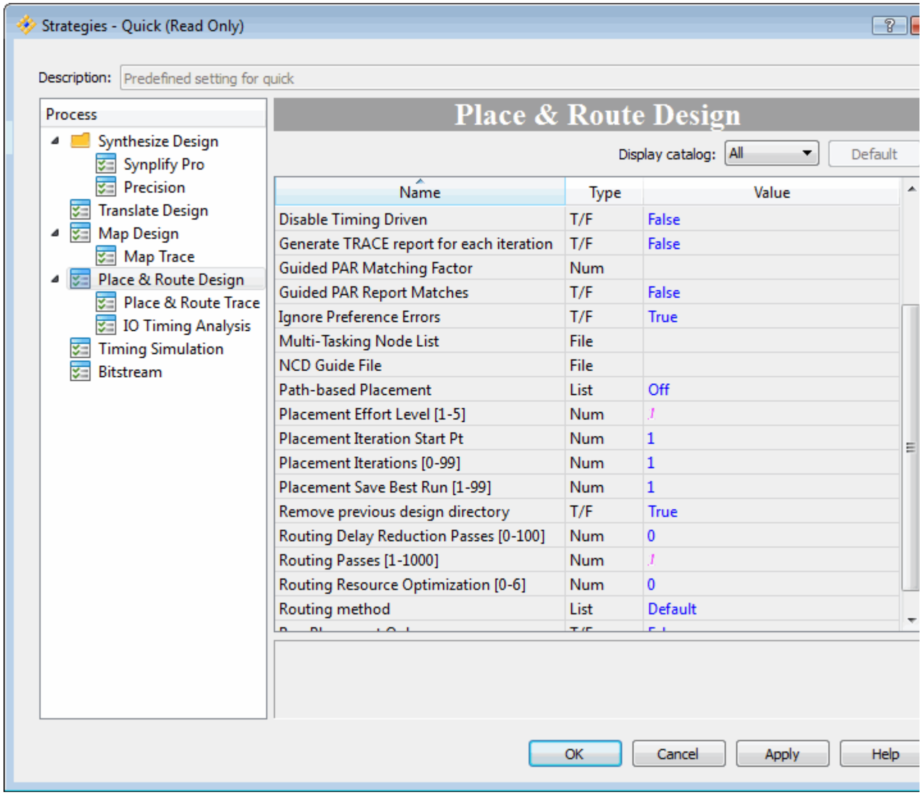
Name	Type	Value
Auto Hold-Time Correction	List	Off
Clock Skew Minimization	List	Off
Command line Options	Text	
Congestion-Driven Placement	List	Auto
Congestion-Driven Routing	List	1
Create Delay Statistic File	T/F	False
Disable Timing Driven	T/F	False
Generate TRACE report for each iteration	T/F	False
Guided PAR Matching Factor	Num	
Guided PAR Report Matches	T/F	False
Ignore Preference Errors	T/F	True
Multi-Tasking Node List	File	
NCD Guide File	File	
Path-based Placement	List	Off
Placement Effort Level [1-5]	Num	5
Placement Iteration Start Pt	Num	1
Placement Iterations [0-99]	Num	1
Placement Save Best Run [1-99]	Num	1

Quick

The Quick strategy is a predefined Strategy for doing an in it al quick run. This Strategy uses very low effort level in place and route to get results with minimum runtime. 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 should be expected to be low, and large or dense designs may not complete routing.

Figure 66: Quick predefined Strategy



Description: Predefined setting for quick

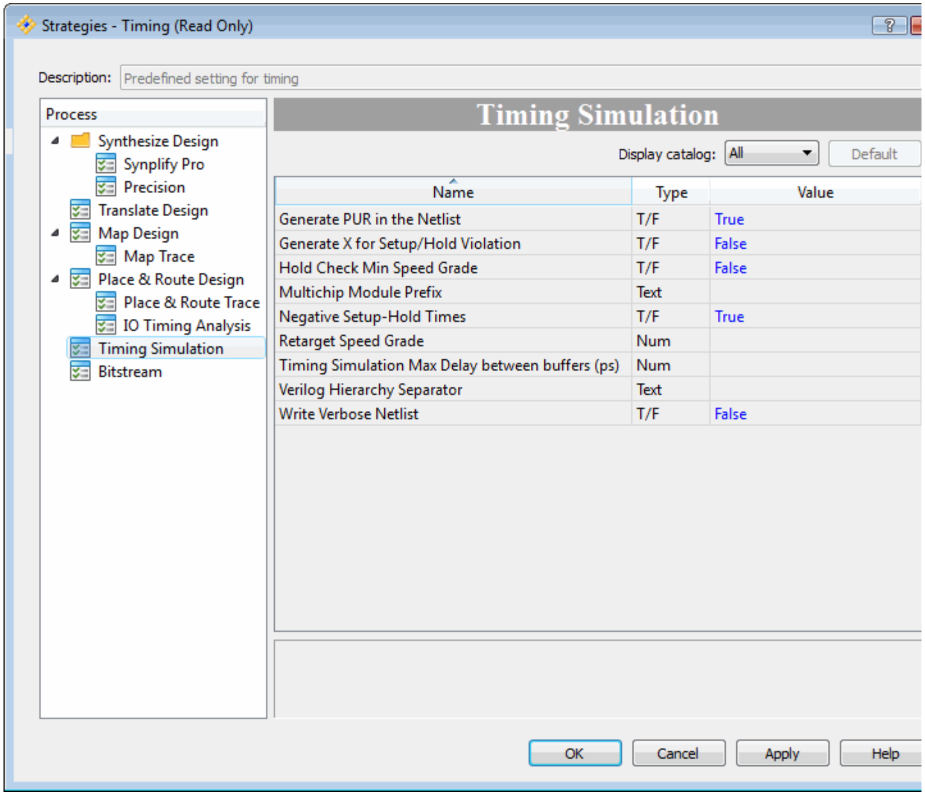
Display catalog: All

Name	Type	Value
Disable Timing Driven	T/F	False
Generate TRACE report for each iteration	T/F	False
Guided PAR Matching Factor	Num	
Guided PAR Report Matches	T/F	False
Ignore Preference Errors	T/F	True
Multi-Tasking Node List	File	
NCD Guide File	File	
Path-based Placement	List	Off
Placement Effort Level [1-5]	Num	1
Placement Iteration Start Pt	Num	1
Placement Iterations [0-99]	Num	1
Placement Save Best Run [1-99]	Num	1
Remove previous design directory	T/F	True
Routing Delay Reduction Passes [0-100]	Num	0
Routing Passes [1-1000]	Num	1
Routing Resource Optimization [0-6]	Num	0
Routing method	List	Default

Timing

The Timing strategy is a predefined strategy for timing optimization. This Strategy tries to achieve timing closure. The Timing Strategy uses very high effort level in place and route. 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 make a custom Strategy with refined settings for your design. This Strategy may increase your runtime on place and route comparing to the Quick and Area Strategies.

Figure 67: Timing predefined Strategy



Description: Predefined setting for timing

Display catalog: All

Name	Type	Value
Generate PUR in the Netlist	T/F	True
Generate X for Setup/Hold Violation	T/F	False
Hold Check Min Speed Grade	T/F	False
Multichip Module Prefix	Text	
Negative Setup-Hold Times	T/F	True
Retarget Speed Grade	Num	
Timing Simulation Max Delay between buffers (ps)	Num	
Verilog Hierarchy Separator	Text	
Write Verbose Netlist	T/F	False

User defined

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

Common Tasks

Working with projects includes many tasks, from initial project creation, to editing design files, modifying tool settings, trying different Implementations and Strategies and saving your data. The following are some of the most common project tasks.

Creating a project

See the *Creating a new project* section in the *Getting Started* chapter for step by step instructions to create a project.

Changing the target device

There are two ways to bring up the Device Selector dialog to change the target device:

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

Editing files

You can edit any of the files by double-clicking or by right-clicking and selecting **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

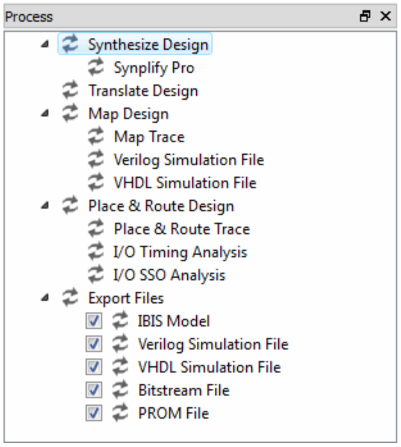
Lattice Diamond Design Flow

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 68: Design flow



Process

- Synthesize Design
  - Synplify Pro
- Translate Design
- Map Design
  - Map Trace
  - Verilog Simulation File
  - VHDL Simulation File
- Place & Route Design
  - Place & Route Trace
  - I/O Timing Analysis
  - I/O SSO Analysis
- Export Files
  - ☑ IBIS Model
  - ☑ Verilog Simulation File
  - ☑ VHDL Simulation File
  - ☑ Bitstream File
  - ☑ PROM File

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, then the netlist will also be read into the design in this process step.

Map Design

This process maps the design to the target FPGA. Map Design converts a design represented as general logical components into placeable components.

- Map Trace

  Trace can be used to run timing analysis in the post-mapping stage in the FPGA design flow. You can run this process after having mapped a design. Trace will create a timing report file (.twr) that will help a designer to determine where timing constraints will not be met. In post-map timing analysis, Trace will determine component delays and use 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

  This process back annotates the mapped design with estimated timing information so that you can run a simulation of your design. The backannotated design is a Verilog netlist.

- VHDL Simulation File

  This process back annotates the mapped design with estimated timing information so that you can run a simulation of your design. The backannotated design is a VHDL netlist.

Place & Route Design

After a design has undergone the necessary translation to bring it into the Native Circuit Description (NCD) format, you can run the Place & Route Design process. This process takes a mapped physical design .ncd file, and places and routes the design. The output is a file that can be processed by the design implementation tools.

- Place & Route Trace

  Trace can be used to run timing analysis in the post-routing stage in the FPGA design flow. You can run this process after having routed a design. Trace will create a timing report (.twr) that will allow a designer to verify timing. In post-route timing analysis, Trace analyzes your path delays and will report where these occur in the design.

- I/O Timing Analysis

  This process runs I/O timing analysis and generates an I/O Timing Report. The goal of this report is to analyze all IOs (both inputs and outputs) across all potential silicon that could be used to help ensure that the board design is compatible. The I/O TIming Report is useful to show constraints to which the board design will need to adhere.

  For each input port in the design, this process generates the worst case setup and hold time requirements and for each output port it will report the worst case min/max clock-to-out delay. The computation is performed over all speed 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

You can check the desired files you want to export and run this process.

- **IBIS Model**

  This process generates a design-specific IBIS model file (.ibs). IBIS stands for I/O Buffer Information Specification. 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**

  This process back annotates the routed design with timing information so that you can run a simulation of your design. The backannotated design is a Verilog netlist.

- **VHDL Simulation File**

  This process back annotates the routed design with timing information so that you can run a simulation of your design. The backannotated design is a VHDL netlist.

- **JEDEC File**

  This process produces 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.

Running Processes

For each step in the flow you can do the following actions:

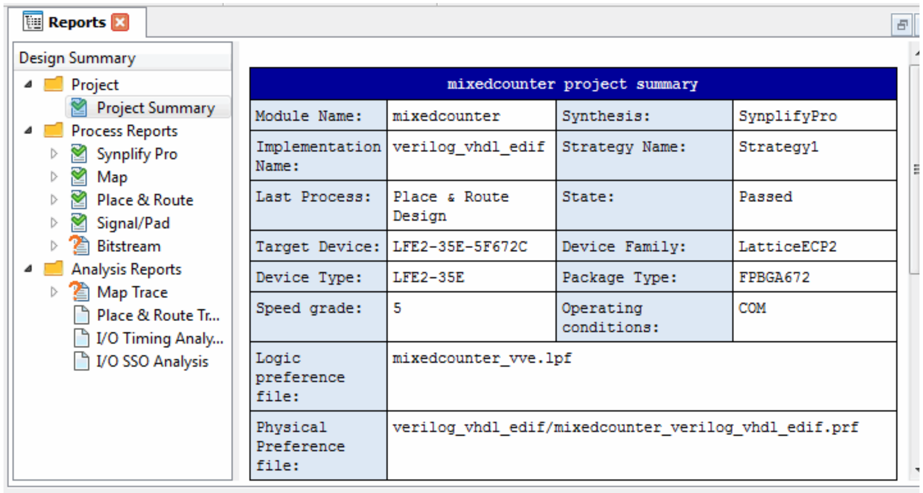
- Run - run the process, will not rerun if process has already been run
- Rerun - rerun a process that has already been run
- Rerun All - reruns all processes from the start to the selected process
- Stop - stops a running process
- Clean Up Process - clears the process state and puts a process into 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:

<img> process state: 4 icons in table

The Reports view displays detailed information about the results of running processes, including the last process run.

Figure 69: Reports shows 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
Speed grade:	5	Operating conditions:	COM
Logic preference file:	mixedcounter_vve.lpf		
Physical Preference file:	verilog_vhdl_edif/mixedcounter_verilog_vhdl_edif.prf		

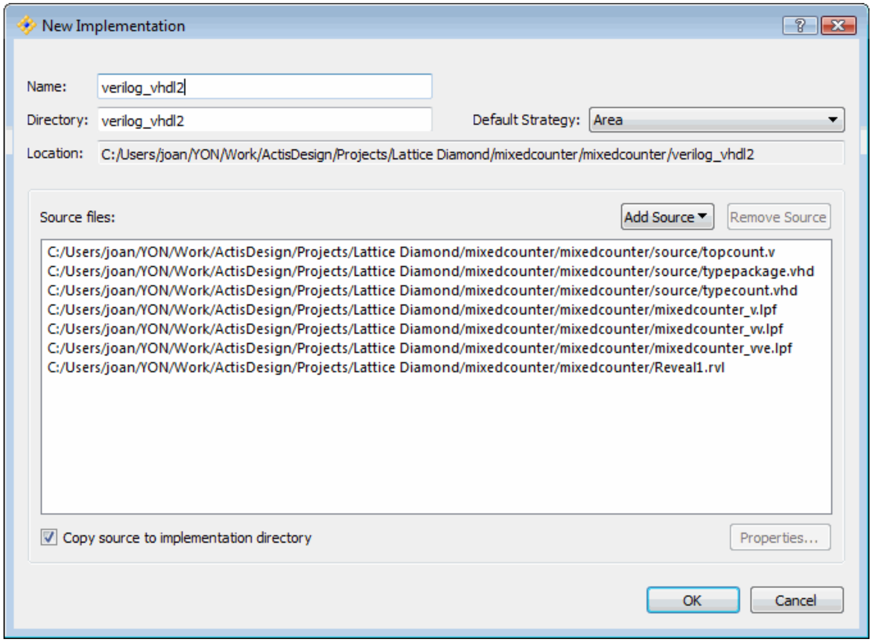
Implementation Flow and Tasks

Understanding how to use Implementations in your process flow is an important concept in Lattice Diamond. Implementations organize the structure of your design and allow you to try alternate structures and tool settings to determine which will give you the best results.

You may 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 on the project name in the File List and select **Add > New Implementation**. In the dialog the **Add Source** selection allows you to use source from an existing Implementation. The **Default Strategy** selection lets you choose from the currently defined Strategies to set the Implementation’s active Strategy.

Figure 70: Adding source to a new implementation



Name: verilog_vhdl2

Directory: verilog_vhdl2

Default Strategy: Area

Location: C:/Users/joan/YON/Work/ActisDesign/Projects/Lattice Diamond/mixedcounter/mixedcounter/verilog_vhdl2

Source files:

```
C:/Users/joan/YON/Work/ActisDesign/Projects/Lattice Diamond/mixedcounter/mixedcounter/source/topcount.v
C:/Users/joan/YON/Work/ActisDesign/Projects/Lattice Diamond/mixedcounter/mixedcounter/source/typepackage.vhd
C:/Users/joan/YON/Work/ActisDesign/Projects/Lattice Diamond/mixedcounter/mixedcounter/source/typecount.vhd
C:/Users/joan/YON/Work/ActisDesign/Projects/Lattice Diamond/mixedcounter/mixedcounter/mixedcounter_v.lpf
C:/Users/joan/YON/Work/ActisDesign/Projects/Lattice Diamond/mixedcounter/mixedcounter/mixedcounter_vv.lpf
C:/Users/joan/YON/Work/ActisDesign/Projects/Lattice Diamond/mixedcounter/mixedcounter/mixedcounter_vve.lpf
C:/Users/joan/YON/Work/ActisDesign/Projects/Lattice Diamond/mixedcounter/mixedcounter/Reveal1.rvl
```

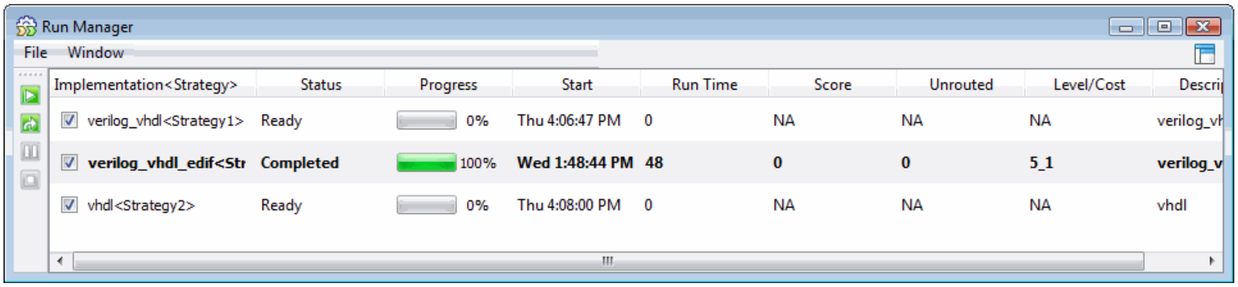
☑ Copy source to implementation directory

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* selection is not selected. By not copying the source you are making sure that your source is kept in sync between the two Implementations.

Run management

The Run Manager is used to run different Implementations. Each Implementation will uses its active Strategy. Select **Tools > Run Manager** or use the icon from the toolbar.

Figure 71: Run Manager



Implementation<Strategy>	Status	Progress	Start	Run Time	Score	Unrouted	Level/Cost	Descri
☑ verilog_vhdl<Strategy1>	Ready	0%	Thu 4:06:47 PM	0	NA	NA	NA	verilog_vh
☑ **verilog_vhdl_edif<Str**	**Completed**	100%	**Wed 1:48:44 PM**	**48**	**0**	**0**	**5_1**	**verilog_v**
☑ vhdl<Strategy2>	Ready	0%	Thu 4:08:00 PM	0	NA	NA	NA	vhdl

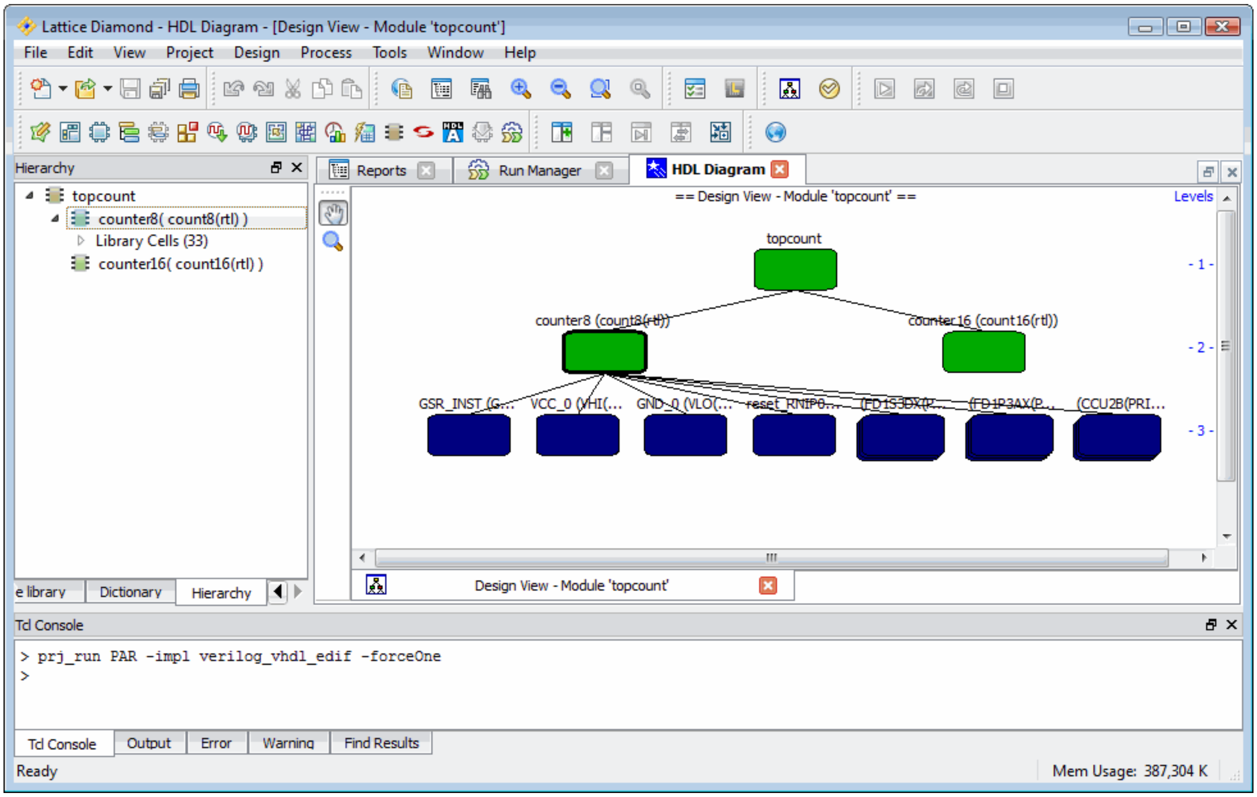
The Run Manager runs the entire process flow for each selected Implementation. If you are running multiple Implementation runs on a multicore system the run manager will distribute them to execute in parallel.

To examine the reports from each process, first make an Implementation active in the File List and then select the Reports view.

HDL Design Hierarchy and Checking

The HDL Diagram view of your design is displayed whenever the Generate Hierarchy or Run BKM Check functions are run.

Figure 72: HDL Diagram


Lattice Diamond - HDL Diagram - [Design View - Module 'topcount']
File Edit View Project Design Process Tools Window Help
Hierarchy
topcount
counter8(count8(rtl))
Library Cells (33)
counter16(count16(rtl))
e library Dictionary Hierarchy
Reports
Run Manager
HDL Diagram
== Design View - Module 'topcount' ==
Levels
topcount
counter8 (count8(rtl))
counter16 (count16(rtl))
GSR_INST (G...
VCC_0 (VHI(...
GND_0 (VLO(...
reset_RNIP8...
(FD1S3DX(P...
(FD1P3AX(P...
(CCU2B(PRI...
- 1 -
- 2 -
- 3 -
Design View - Module 'topcount'
Tcl Console
Tcl Console Output Error Warning Find Results

```
> prj_run PAR -impl verilog_vhdl_edif -forceOne
>
```

Ready

Mem Usage: 387,304 K

To generate the design hierarchy select **Design > Generate Hierarchy** from the menu or the toolbar. If you want to automatically generate the design hierarchy whenever a design is loaded or changed select **Design > Auto Generate Hierarchy**. Note that when auto generation is set it will be run 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.

Best Known Methods (BKM) are design guidelines used to analyze your design. BKM design checks include:

- 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 select **Design > Run BKM Check** from the menu or the icon from the toolbar. If you want to automatically run these checks whenever a design is loaded select **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.

Constraint Creation

The following steps illustrate how you might assign design constraints and implement them at each stage of the design flow.

1. You can include some constraints at the HDL level using HDL attributes. These will be included in the EDIF netlist.

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 to use to set 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** – View 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** – View 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** – View 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.

3. Save the preferences to the logical preference file LPF.

4. Run the Map Design process (map). This process reads the NGD and LPF files and produces a native circuit description NCD file and a physical preference PRF file. The PRF contains preferences used by the PAR engine. The PRF file is an internal file generated by the Map engine and 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 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.

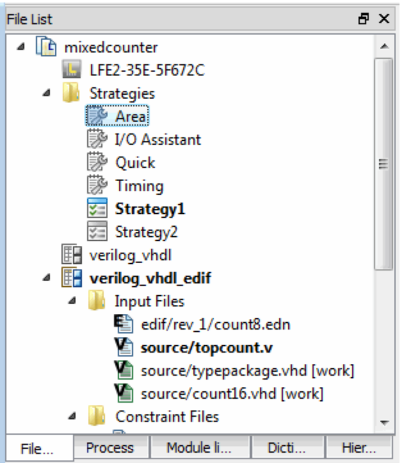
8. Modify preferences or create new ones using any of the views. Save the preference changes and rerun the Place & Route Design process.

Simulation Flow

The Simulation Flow in Lattice Diamond has been optimized to place emphasis on the design module source. As a result there are no testbench files contained in a Lattice Diamond project. Testbench templates can be created after using the Simulation Wizard and a simulator can be launched from Lattice Diamond. This allows faster, easier creation of new module source files while still providing a simple mechanism to create testbench templates and export your design to a simulator.

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.

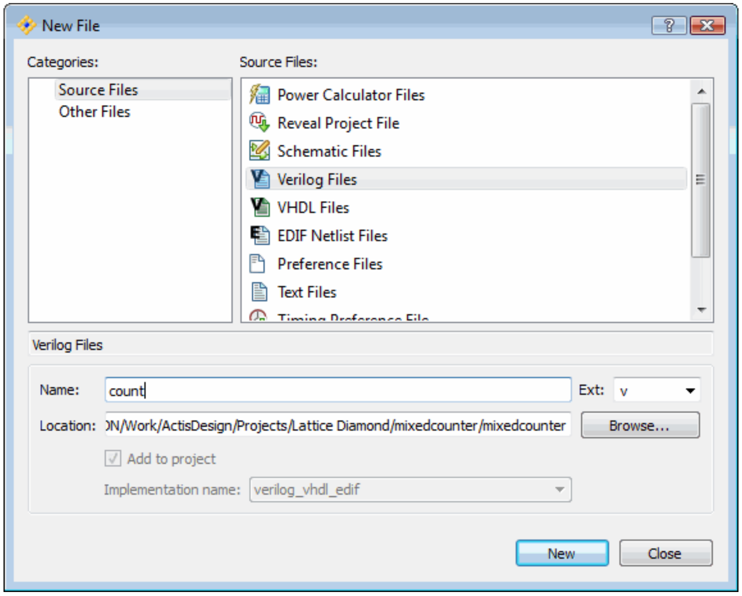
Figure 73: Input files for simulation



To create a new module file:

- Right-click on the Implementation in the File List and select **Add > New File**
- Or from the menus select **File > New > File**

Figure 74: Adding a new module file



Verilog Files

Name: count

Ext: v

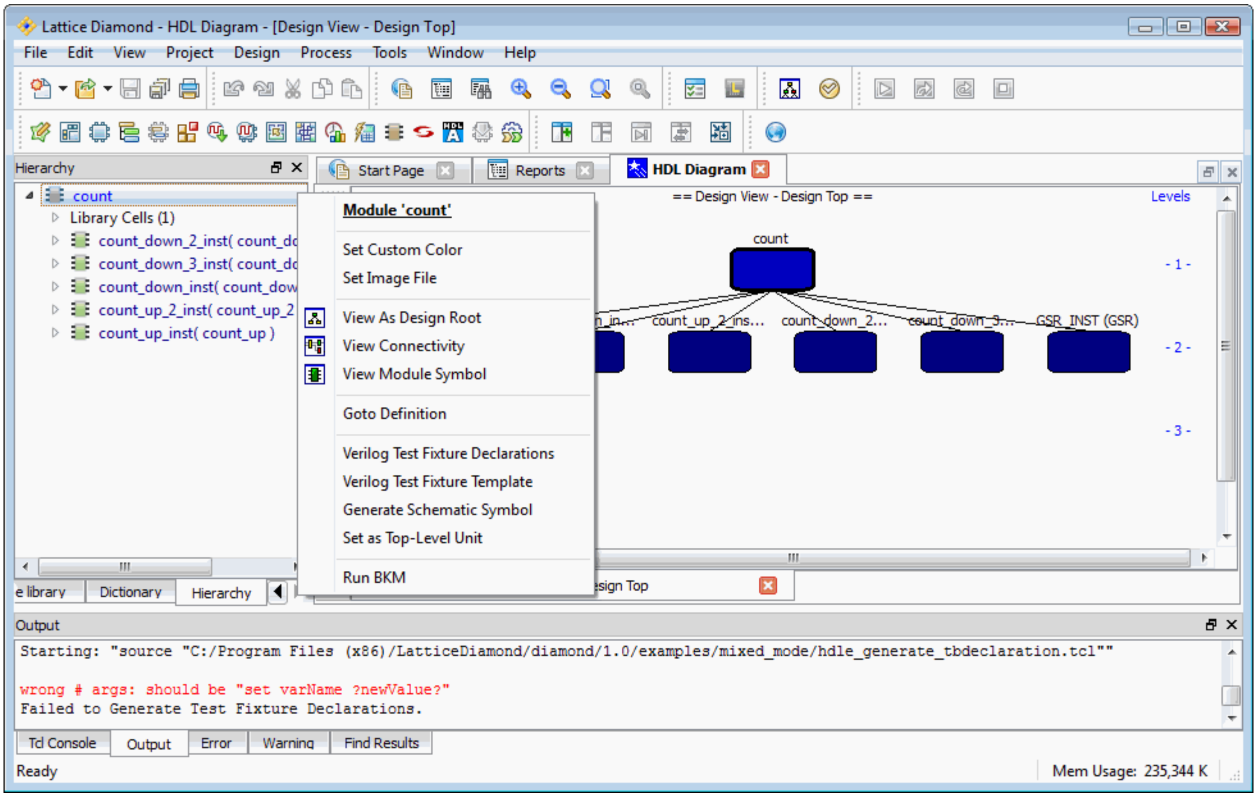
Location: ON/Work/ActisDesign/Projects/Lattice Diamond/mixedcounter/mixedcounter

☑ Add to project

Implementation name: verilog_vhdl_edif

To generate a testbench template you first need to generate the design hierarchy. Select Generate Hierarchy and select the project Hierarchy view of the design.

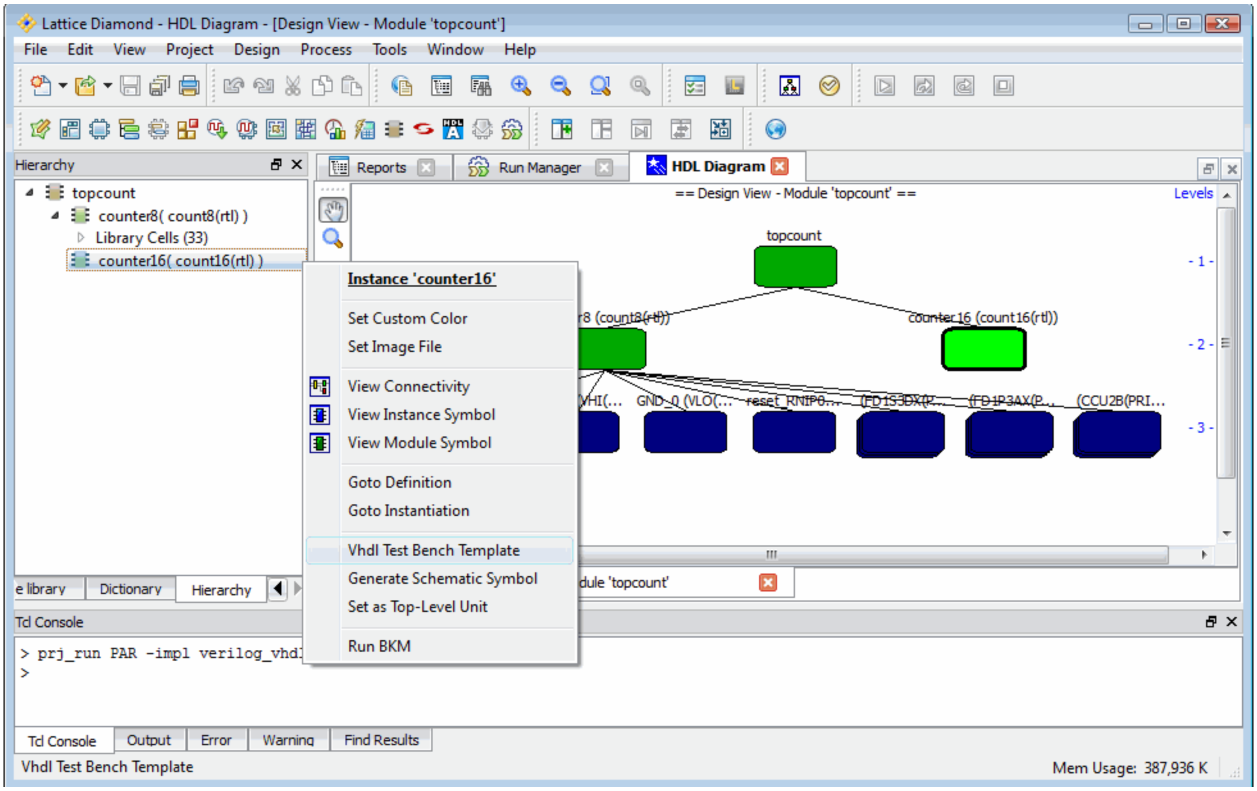
Figure 75: Hierarchical view


Lattice Diamond - HDL Diagram - [Design View - Design Top]
File Edit View Project Design Process Tools Window Help
Hierarchy
count
Library Cells (1)
count_down_2_inst(count_dc
count_down_3_inst(count_dc
count_down_inst(count_dow
count_up_2_inst(count_up_2
count_up_inst(count_up)
e library Dictionary Hierarchy
Module 'count'
Set Custom Color
Set Image File
View As Design Root
View Connectivity
View Module Symbol
Goto Definition
Verilog Test Fixture Declarations
Verilog Test Fixture Template
Generate Schematic Symbol
Set as Top-Level Unit
Run BKM
Start Page
Reports
HDL Diagram
== Design View - Design Top ==
Levels
count
- 1 -
- 2 -
- 3 -
count_up_2_ins...
count_down_2...
count_down_3...
GSR_INST (GSR)
sign Top
Output
Tcl Console Output Error Warning Find Results
Ready
Mem Usage: 235,344 K

```
Starting: "source "C:/Program Files (x86)/LatticeDiamond/diamond/1.0/examples/mixed_mode/hdle_generate_tbdeclaration.tcl""

wrong # args: should be "set varName ?newValue?"
Failed to Generate Test Fixture Declarations.
```

Right-click on a module in the Hierarchy view and the menu includes a selection to generate a test fixture template for that module. The Output view will display the filename and path to the generated test fixture source file.

Figure 76: Test fixture template generation



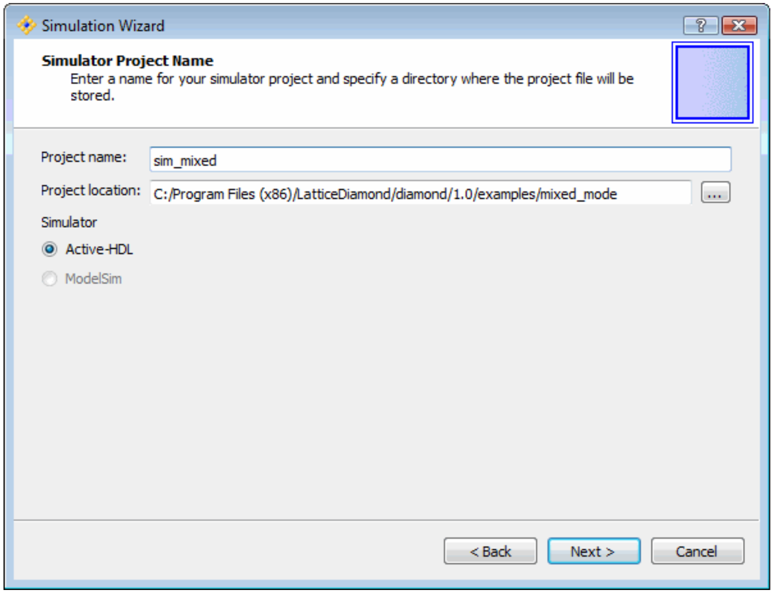
```
> prj_run PAR -impl verilog_vhdl
>
```

Vhdl Test Bench Template

Mem Usage: 387,936 K

When you are ready to simulate you can export the design using the **Tools > Simulation Wizard**. The wizard will lead you through a series of steps including selecting a simulation project name and location, which simulator to use (if you have more than one installed), selecting the process stage to use (from RTL to Post-Route Gate-Level + Timing), language (VHDL or Verilog) and source files. You can optionally run the simulation directly from the wizard.

Figure 77: Simulation wizard



Simulation Wizard

Simulator Project Name

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

Project name: sim_mixed

Project location: C:/Program Files (x86)/LatticeDiamond/diamond/1.0/examples/mixed_mode

Simulator

● Active-HDL

○ ModelSim

To launch a simulation from Lattice Diamond select **Tools >**<simulator> or use the simulator icon in the toolbar. You will need to add the testbench file to the simulation environment.

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 backannotated 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.

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 the 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
- Simulation module files only in Lattice Diamond. Simulation testbench files are not supported in Lattice Diamond.
- Exporting designs to simulation is done with the Simulation Wizard in Lattice Diamond
- Timing analysis can be performed using the Timing Analysis View without having to re-implement the design. See the Timing Analysis View section for more details
- Design hierarchy is not generated by default, you must use the Generate Hierarchy function
- 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. Common tool controls and report viewing provide an easy to use methodology to increase productivity and enable higher quality results.

This chapter includes an overview of each tool contained in Lattice Diamond as well as project level information views. A variety of tool views provide unique ways to control, view and analyze different aspects of the FPGA implementation.

Shared design data

Lattice Diamond uses shared memory that is accessed by all tools and views. When multiple tools are active at the same time an asterix * in the tool title tab indicates that the design data has been changed and not yet saved.

Cross probing

Shared design data allows data objects in one view of the design to be selected and displayed in other views. This cross probing capability is very useful when wanting to display the physical location of a component or net after it has been implemented.

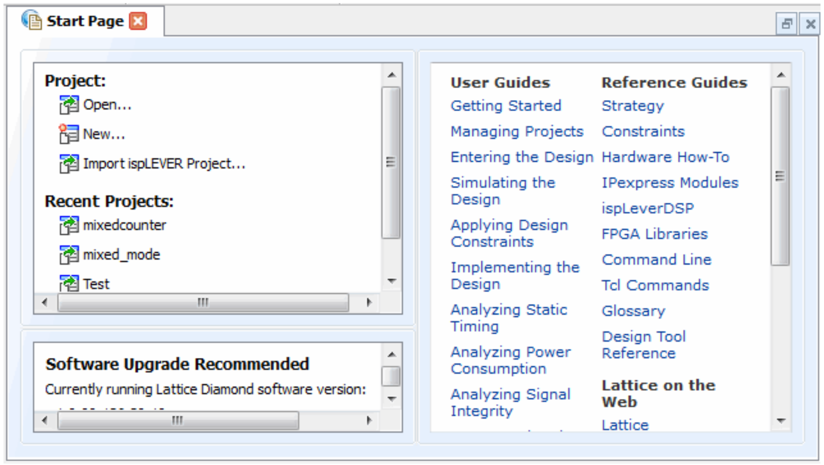
View Highlights

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

Start Page

The Start Page is displayed by default when Lattice Diamond is run. The three panes within the Start Page are for opening projects, product documentation and software version and updates. Startup behavior can be modified using **Tools > Options**.

Figure 78: Start Page



Software Upgrade Recommended

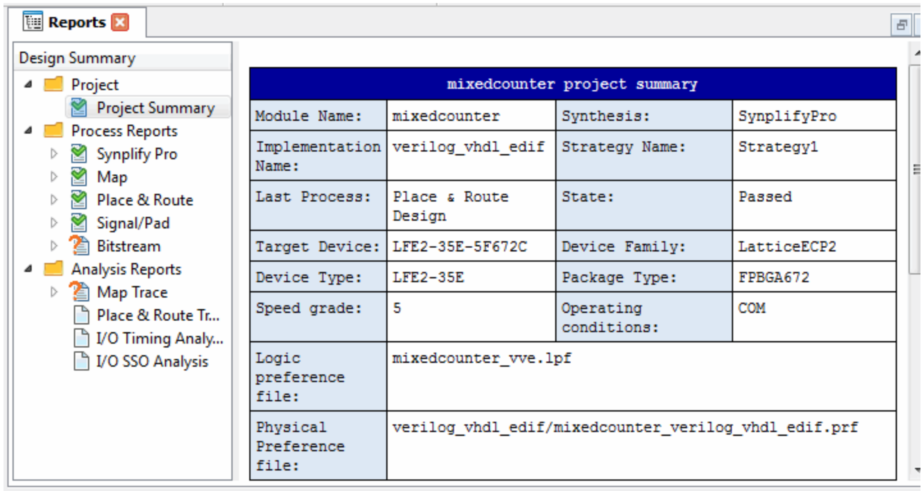
Currently running Lattice Diamond software version:

The Start Page is a handy place to open projects and find 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 79: Reports view



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
Speed grade:	5	Operating conditions:	COM
Logic preference file:	mixedcounter_vve.lpf		
Physical Preference file:	verilog_vhdl_edif/mixedcounter_verilog_vhdl_edif.prf		

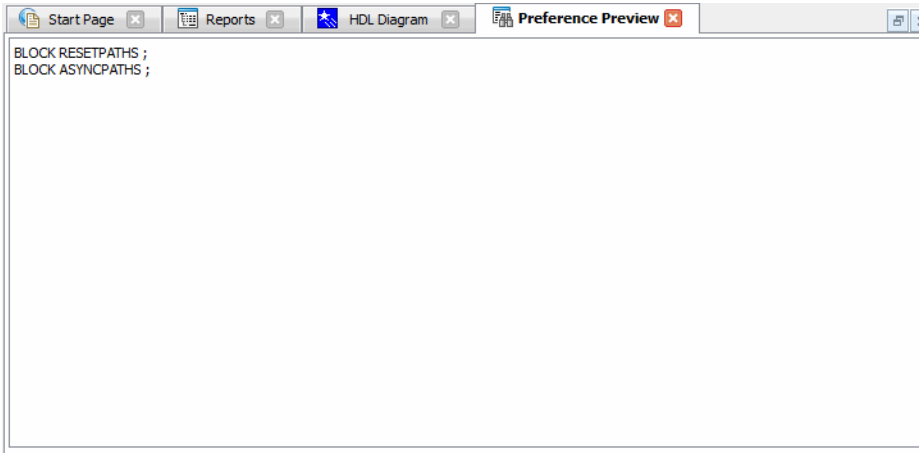
In the left pane of the Reports view is a Design Summary section organized into Project, Process Reports and Analysis Reports. Select any item to see its report. There may be no report for a selection because a process step has not been run. The different file icons indicate if a report has never been generated, completed (green check mark) or is out of date (orange question mark).

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

The Preference Preview shows the constraints for the current Implementation. It is a read only view of the active LPF file.

Figure 80: Preference preview



```
BLOCK RESETPATHS ;
BLOCK ASYNCPATHS ;
```

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

Tools

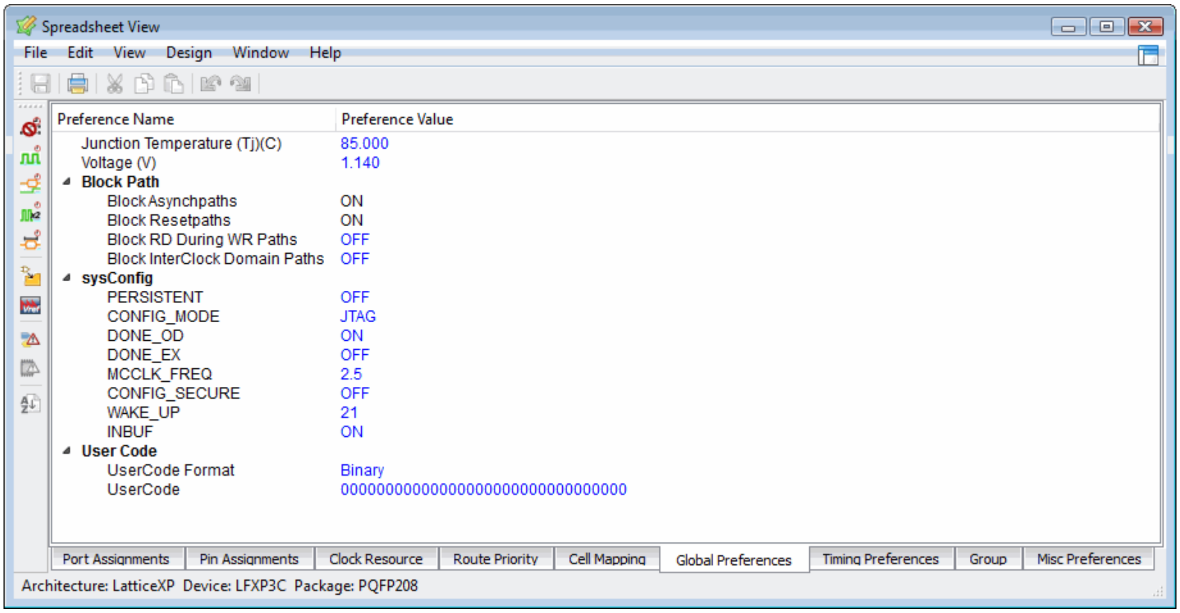
The entire FPGA implementation process tool set is contained in Lattice Diamond. Tools are run by selecting them in the **Tools** menu or toolbar.

The following sections give an overview of each tool. Tools that are internal to Lattice Diamond are explained in some detail. Detailed descriptions of external tools can be found in their product documentation.

Spreadsheet View

The Spreadsheet View provides an interactive spreadsheet format for viewing and assigning design constraints. Its collection of spreadsheets enables you to assign preferences such as PERIOD, FREQUENCY, I/O timing, and LOCATE to optimize placement and routing. 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.

Figure 81: Spreadsheet view



Preference Name	Preference Value
Junction Temperature (Tj)(C)	85.000
Voltage (V)	1.140
Block Path	
Block Asynchpaths	ON
Block Resetpaths	ON
Block RD During WR Paths	OFF
Block InterClock Domain Paths	OFF
sysConfig	
PERSISTENT	OFF
CONFIG_MODE	JTAG
DONE_OD	ON
DONE_EX	OFF
MCCLK_FREQ	2.5
CONFIG_SECURE	OFF
WAKE_UP	21
INBUF	ON
User Code	
UserCode Format	Binary
UserCode	00000000000000000000000000000000

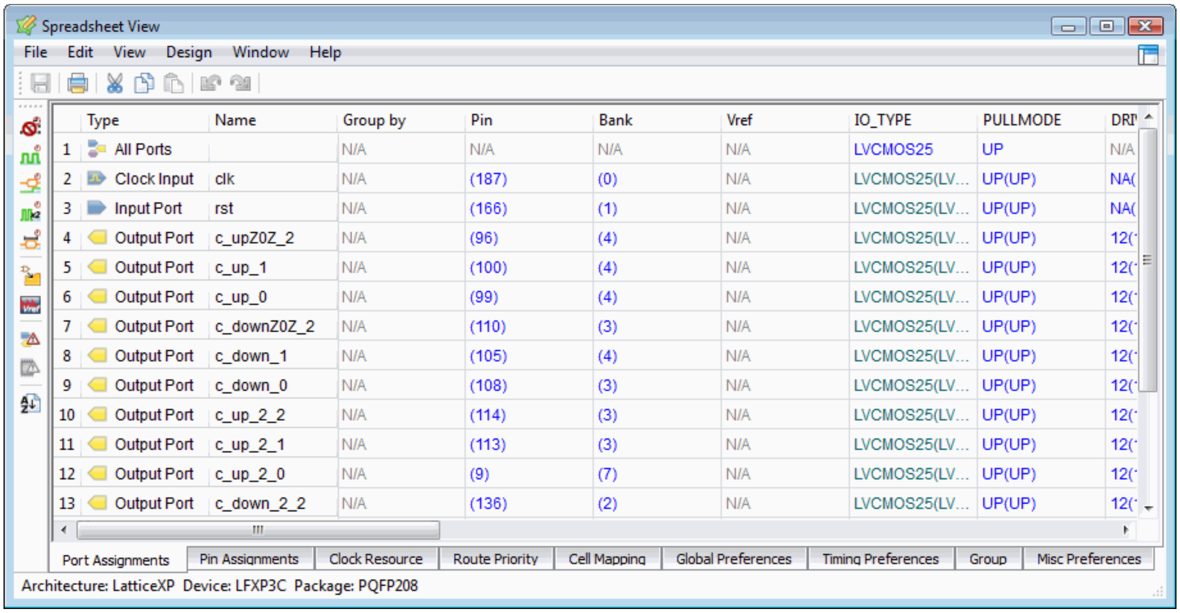
Architecture: LatticeXP Device: LFXP3C Package: PQFP208

As soon as the target device has been specified, Spreadsheet View enables you to set global preferences. 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 82: Spreadsheet view port assignments



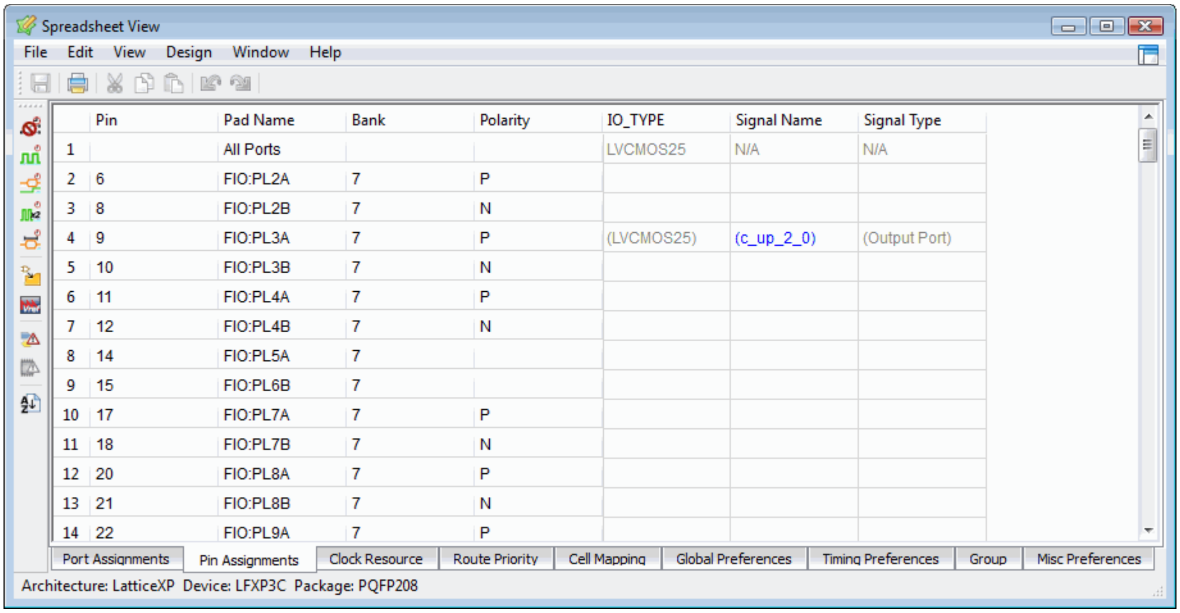
	Type	Name	Group by	Pin	Bank	Vref	IO_TYPE	PULLMODE	DRI
1	All Ports		N/A	N/A	N/A	N/A	LVCMOS25	UP	N/A
2	Clock Input	clk	N/A	(187)	(0)	N/A	LVCMOS25(LV...	UP(UP)	NA(
3	Input Port	rst	N/A	(166)	(1)	N/A	LVCMOS25(LV...	UP(UP)	NA(
4	Output Port	c_upZ0Z_2	N/A	(96)	(4)	N/A	LVCMOS25(LV...	UP(UP)	12(
5	Output Port	c_up_1	N/A	(100)	(4)	N/A	LVCMOS25(LV...	UP(UP)	12(
6	Output Port	c_up_0	N/A	(99)	(4)	N/A	LVCMOS25(LV...	UP(UP)	12(
7	Output Port	c_downZ0Z_2	N/A	(110)	(3)	N/A	LVCMOS25(LV...	UP(UP)	12(
8	Output Port	c_down_1	N/A	(105)	(4)	N/A	LVCMOS25(LV...	UP(UP)	12(
9	Output Port	c_down_0	N/A	(108)	(3)	N/A	LVCMOS25(LV...	UP(UP)	12(
10	Output Port	c_up_2_2	N/A	(114)	(3)	N/A	LVCMOS25(LV...	UP(UP)	12(
11	Output Port	c_up_2_1	N/A	(113)	(3)	N/A	LVCMOS25(LV...	UP(UP)	12(
12	Output Port	c_up_2_0	N/A	(9)	(7)	N/A	LVCMOS25(LV...	UP(UP)	12(
13	Output Port	c_down_2_2	N/A	(136)	(2)	N/A	LVCMOS25(LV...	UP(UP)	12(

Architecture: LatticeXP Device: LFXP3C Package: PQFP208

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 83: Spreadsheet view pin assignments



	Pin	Pad Name	Bank	Polarity	IO_TYPE	Signal Name	Signal Type
1		All Ports			LVCMOS25	N/A	N/A
2	6	FIO:PL2A	7	P			
3	8	FIO:PL2B	7	N			
4	9	FIO:PL3A	7	P	(LVCMOS25)	(c_up_2_0)	(Output Port)
5	10	FIO:PL3B	7	N			
6	11	FIO:PL4A	7	P			
7	12	FIO:PL4B	7	N			
8	14	FIO:PL5A	7				
9	15	FIO:PL6B	7				
10	17	FIO:PL7A	7	P			
11	18	FIO:PL7B	7	N			
12	20	FIO:PL8A	7	P			
13	21	FIO:PL8B	7	N			
14	22	FIO:PL9A	7	P			

Architecture: LatticeXP Device: LFXP3C Package: PQFP208

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 select **Update**.

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.

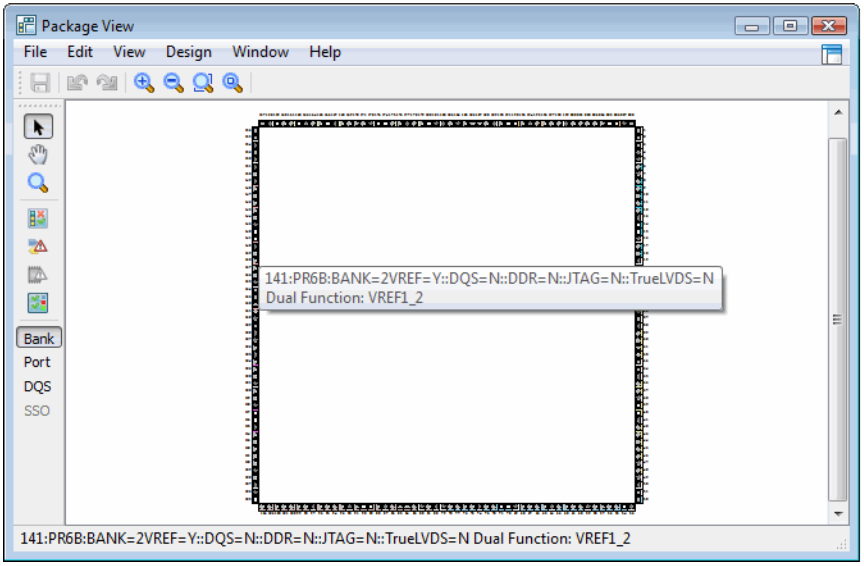
Misc Preferences

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.

Figure 84: Package view



141:PR6B:BANK=2VREF=Y::DQS=N::DDR=N::JTAG=N::TrueLVDS=N
Dual Function: VREF1_2

141:PR6B:BANK=2VREF=Y::DQS=N::DDR=N::JTAG=N::TrueLVDS=N Dual Function: VREF1_2

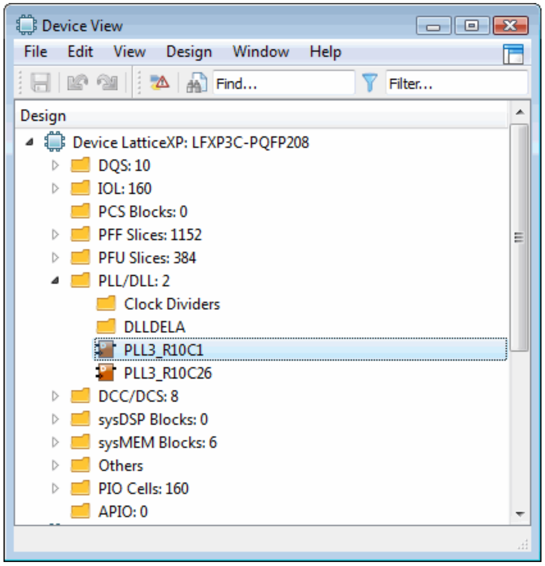
As you move your mouse pointer over the layout, pin descriptions and locations are displayed in pop-ups and in the status bar. The **View > Show Differential Pairs** command displays fly wires between differential pin pairs and identifies the positive differential pins.

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 85: Device view



Device LatticeXP: LFXP3C-PQFP208
- DQS: 10
- IOL: 160
- PCS Blocks: 0
- PFF Slices: 1152
- PFU Slices: 384
- PLL/DLL: 2
  - Clock Dividers
  - DLLDELA
  - PLL3_R10C1
  - PLL3_R10C26
- DCC/DCS: 8
- sysDSP Blocks: 0
- sysMEM Blocks: 6
- Others
- PIO Cells: 160
- APIO: 0

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

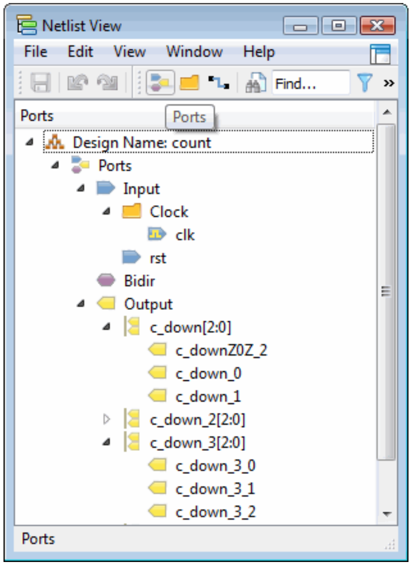
Device View is available as soon as the target device has been specified.

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.

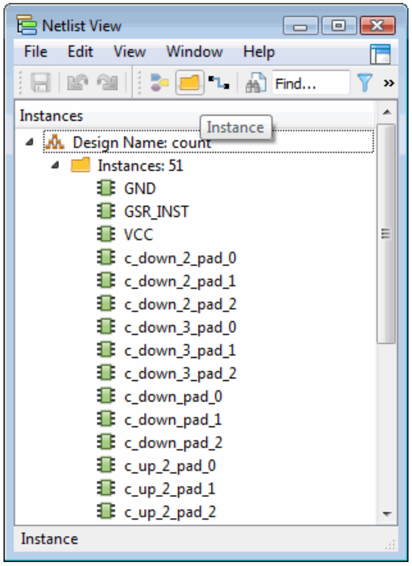
Netlist View organizes the netlist by ports, instances, and nets, and it provides a toolbar button and design tree view for each of these categories to make it easier to create timing or location preferences.

Figure 86: Netlist view ports design tree



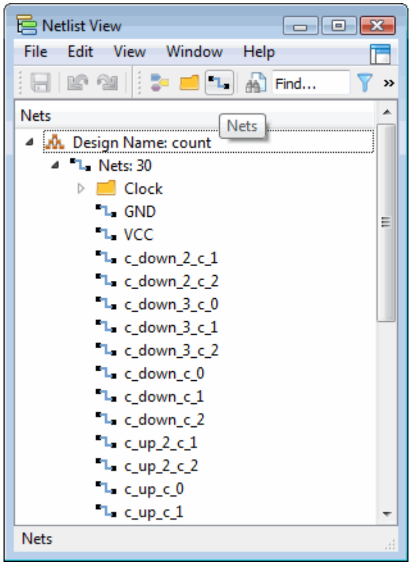
```
Design Name: count
  Ports
    Input
      Clock
        clk
      rst
    Bidir
    Output
      c_down[2:0]
        c_downZ0Z_2
        c_down_0
        c_down_1
      c_down_2[2:0]
      c_down_3[2:0]
        c_down_3_0
        c_down_3_1
        c_down_3_2
```

Figure 87: Netlist view instances design tree



```
Design Name: count
  Instances: 51
    GND
    GSR_INST
    VCC
    c_down_2_pad_0
    c_down_2_pad_1
    c_down_2_pad_2
    c_down_3_pad_0
    c_down_3_pad_1
    c_down_3_pad_2
    c_down_pad_0
    c_down_pad_1
    c_down_pad_2
    c_up_2_pad_0
    c_up_2_pad_1
    c_up_2_pad_2
```

Figure 88: Netlist view nets design tree



- Design Name: count
  - Nets: 30
    - Clock
    - GND
    - VCC
    - c_down_2_c_1
    - c_down_2_c_2
    - c_down_3_c_0
    - c_down_3_c_1
    - c_down_3_c_2
    - c_down_c_0
    - c_down_c_1
    - c_down_c_2
    - c_up_2_c_1
    - c_up_2_c_2
    - c_up_c_0
    - c_up_c_1

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

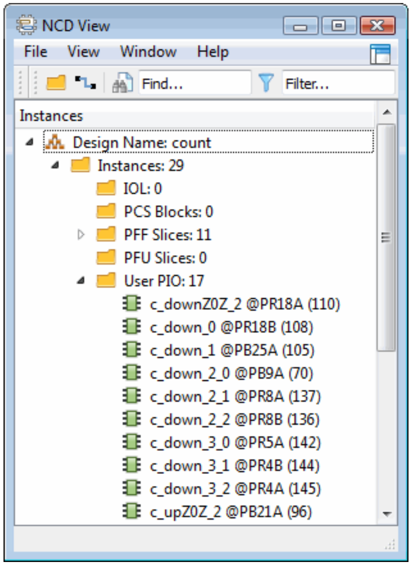
Netlist View is available after synthesis and translation.

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.

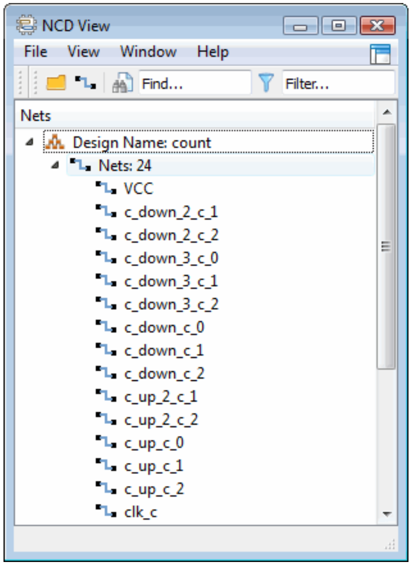
Figure 89: NCD view instances design tree



Instances

- Design Name: count
  - Instances: 29
    - IOL: 0
    - PCS Blocks: 0
    - PFF Slices: 11
    - PFU Slices: 0
    - User PIO: 17
      - c_downZ0Z_2 @PR18A (110)
      - c_down_0 @PR18B (108)
      - c_down_1 @PB25A (105)
      - c_down_2_0 @PB9A (70)
      - c_down_2_1 @PR8A (137)
      - c_down_2_2 @PR8B (136)
      - c_down_3_0 @PR5A (142)
      - c_down_3_1 @PR4B (144)
      - c_down_3_2 @PR4A (145)
      - c_upZ0Z_2 @PB21A (96)

Figure 90: NCD view nets design tree



Nets

- Design Name: count
  - Nets: 24
    - VCC
    - c_down_2_c_1
    - c_down_2_c_2
    - c_down_3_c_0
    - c_down_3_c_1
    - c_down_3_c_2
    - c_down_c_0
    - c_down_c_1
    - c_down_c_2
    - c_up_2_c_1
    - c_up_2_c_2
    - c_up_c_0
    - c_up_c_1
    - c_up_c_2
    - clk_c

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

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.

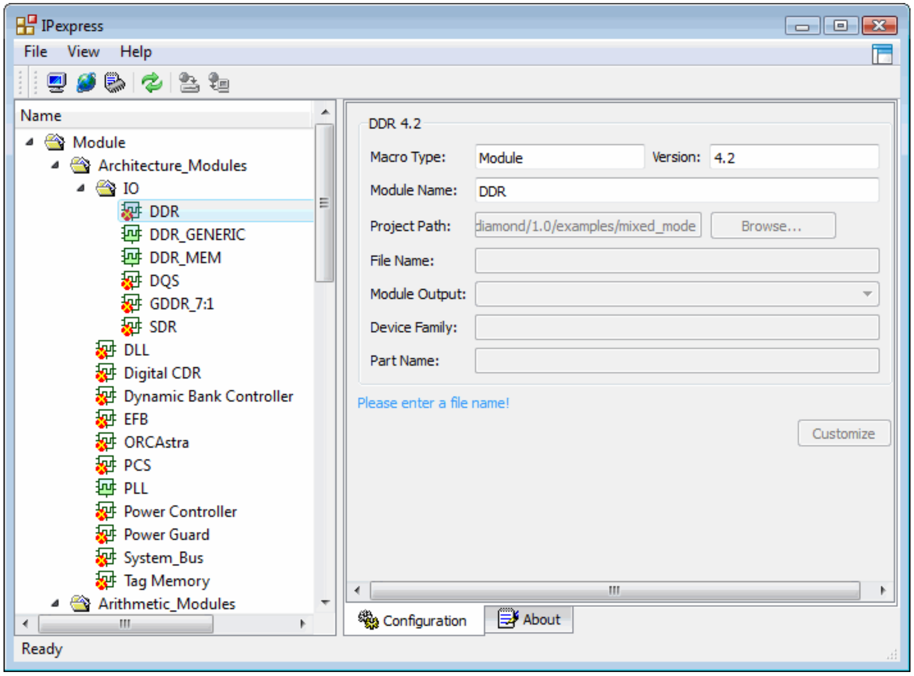
IPexpress

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 checkbox which will include an ipx file in your source list. This file can be used to regenerate the module for any changes.

Select **Tools > IPexpress** to see the full list of what's available.

Figure 91: IPexpress



DDR 4.2

Macro Type: Module Version: 4.2

Module Name: DDR

Project Path: diamond/1.0/examples/mixed_mode

File Name:

Module Output:

Device Family:

Part Name:

Please enter a file name!

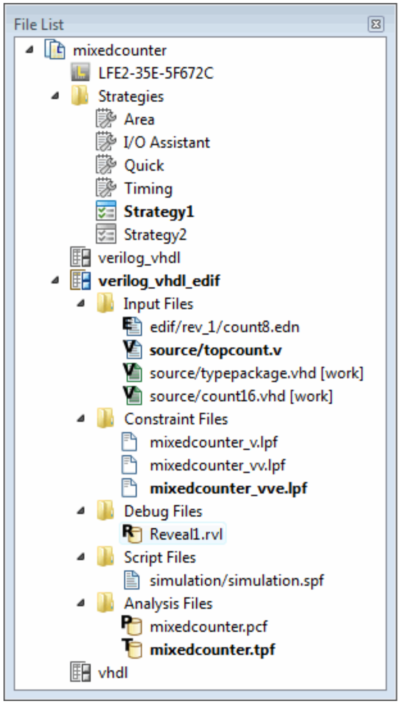
In addition to the included modules there is a Server tab that allows you to connect to the Lattice server and find available IP to be downloaded and installed. Lattice IP can be purchased or used in a trial mode. Refer to the Lattice website for a list of the available IP functions. For more information see the Advanced Topics chapter or the IPexpress Module Reference Guide.

Reveal Inserter

The Reveal Inserter tool allows you to add debug information to your design for use with Reveal Analyzer. Reveal Inserter enables you to select which design signals to use for debug tracing or triggering, then generate a core on the basis of these signals and their use. After generating the required core, it generates a modified design with the necessary debug connections and links it to the signals. Reveal Inserter supports VHDL, Verilog, and EDIF flows for debug insertion. Once 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 92: File list view with Reveal debug project file



File List

- mixedcounter
  - LFE2-35E-5F672C
  - Strategies
    - Area
    - I/O Assistant
    - Quick
    - Timing
    - **Strategy1**
    - Strategy2
  - verilog_vhdl
  - **verilog_vhdl_edif**
    - Input Files
      - edif/rev_1/count8.edn
      - **source/topcount.v**
      - source/typepackage.vhd [work]
      - source/count16.vhd [work]
    - Constraint Files
      - mixedcounter_v.lpf
      - mixedcounter_vv.lpf
      - **mixedcounter_vve.lpf**
    - Debug Files
      - Reveal1.rvl
    - Script Files
      - simulation/simulation.spf
    - Analysis Files
      - mixedcounter.pcf
      - **mixedcounter.tpf**
  - vhdl

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

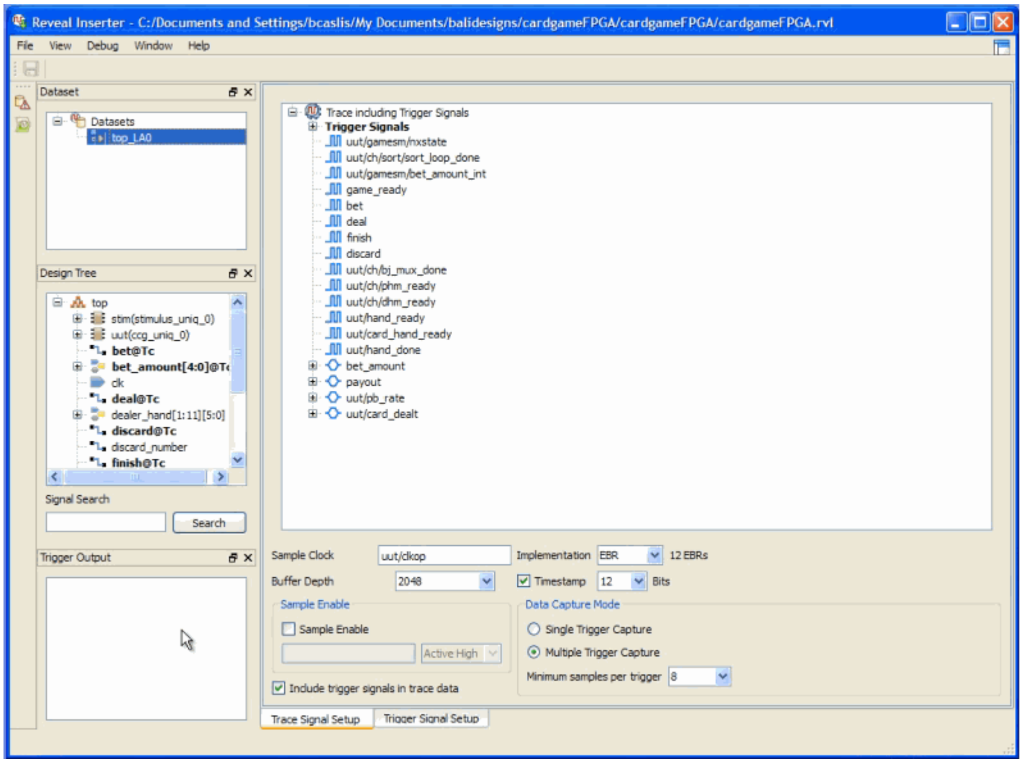
Reveal Inserter can be launched in multiple ways and will use different debug files according to the following conditions:

- If there is an active debug file it will be used if Reveal Inserter is launched from the menu selection **Tools > Reveal Inserter** or the toolbar icon. Also if there is only one debug file, even if it is inactive, it will be used at launch.
- Any debug file, active or inactive, can be double-clicked in the File List Debug folder and that debug file will be used to launch Reveal Inserter.
- If you launch Reveal Inserter and there are multiple debug files and none are selected as active, a dialog will ask you if you want to use one of the inactive debug files.

- If no debug files exist Reveal Inserter will be launched with a default configuration.

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

Figure 93: Reveal Inserter Trace Signal Setup


Reveal Inserter - C:/Documents and Settings/bcaslis/My Documents/balidesigns/cardgameFPGA/cardgameFPGA/cardgameFPGA.rvl

Trace including Trigger Signals
- Trigger Signals
  - uut/gamesm/nxstate
  - uut/ch/sort/sort_loop_done
  - uut/gamesm/bet_amount_int
  - game_ready
  - bet
  - deal
  - finish
  - discard
  - uut/ch/bj_mux_done
  - uut/ch/phm_ready
  - uut/ch/dhm_ready
  - uut/hand_ready
  - uut/card_hand_ready
  - uut/hand_done
- bet_amount
- payout
- uut/pb_rate
- uut/card_dealt

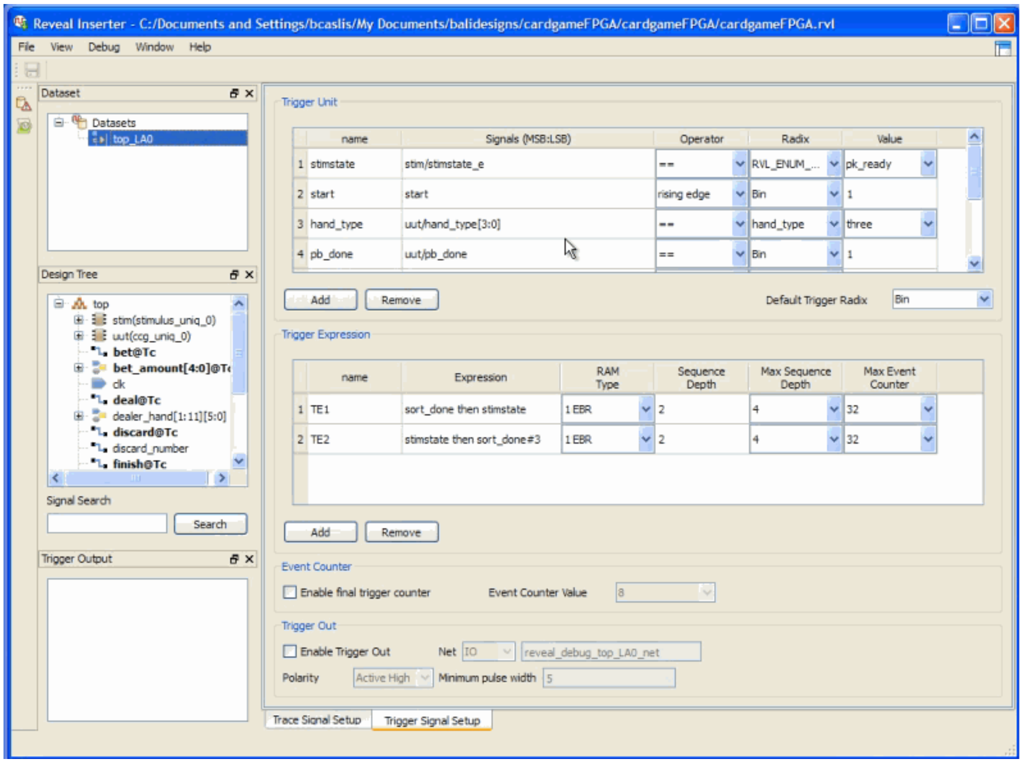
Sample Clock: uut/clkop
Implementation: EBR 12 EBRs
Buffer Depth: 2048
☑ Timestamp 12 Bits

Sample Enable
□ Sample Enable Active High

Data Capture Mode
○ Single Trigger Capture
● Multiple Trigger Capture
Minimum samples per trigger: 8

☑ Include trigger signals in trace data

Figure 94: Reveal Inserter Trigger Signal Setup



Trigger Unit

	name	Signals (MSB:LSB)	Operator	Radix	Value
1	stimstate	stim/stimstate_e	==	RVL_ENUM_...	pk_ready
2	start	start	rising edge	Bin	1
3	hand_type	uut/hand_type[3:0]	==	hand_type	three
4	pb_done	uut/pb_done	==	Bin	1

Add Remove Default Trigger Radix: Bin

Trigger Expression

	name	Expression	RAM Type	Sequence Depth	Max Sequence Depth	Max Event Counter
1	TE1	sort_done then stimstate	1 EBR	2	4	32
2	TE2	stimstate then sort_done#3	1 EBR	2	4	32

Add Remove

Event Counter

□ Enable final trigger counter Event Counter Value: 8

Trigger Out

□ Enable Trigger Out Net: IO reveal_debug_top_LA0_net

Polarity: Active High Minimum pulse width: 5

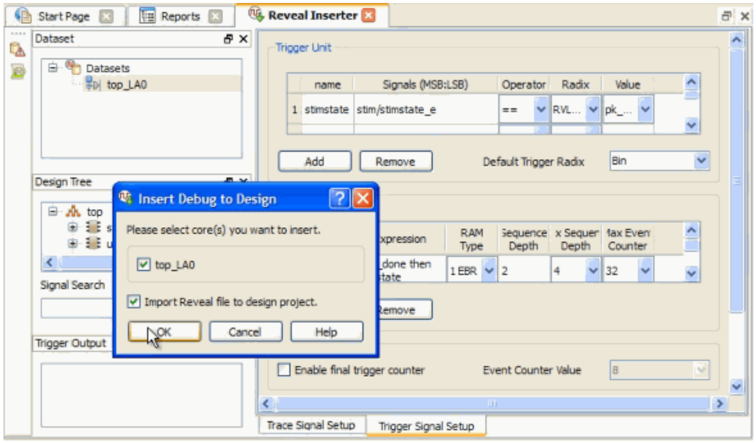
Setting up and Inserting Debug

A **Debug** menu is displayed when Reveal Inserter is launched. It contains controls for managing cores, managing trace signals, running DRC, inserting debug and for managing token sets.

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

Once you have your debug setup, use the **Insert Debug** control to insert debug to your design. The Insert Debug control is found in the Debug menu and in the toolbar in the Reveal Inserter view. This will cause the current debug file to be set active.

Figure 95: Insert Debug



	name	Signals (MSB:LSB)	Operator	Radix	Value
1	stimstate	stim/stimstate_e	==	RVL...	pk_...

Default Trigger Radix: Bin

xpression	RAM Type	Sequence Depth	x Sequer Depth	lax Even Counter
done then tate	1 EBR	2	4	32

□ Enable final trigger counter   Event Counter Value: 8

Insert Debug to Design

Please select core(s) you want to insert.

☑ top_LA0

☑ Import Reveal file to design project.

OK   Cancel   Help

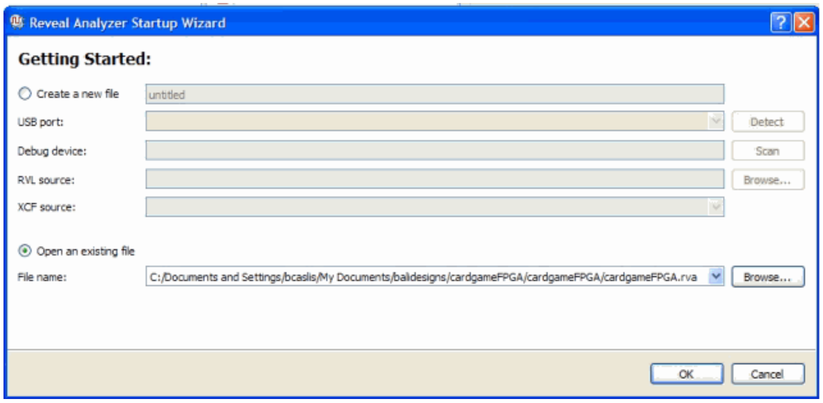
When the design is full implemented and programmed you can then run Reveal Analyzer to debug your design.

Reveal Analyzer

After you generate the bitstream or JEDEC file, Reveal Analyzer can be used to debug your FPGA circuitry by giving you access to internal nodes inside the device so you can observe their behavior. It enables you to set and change various values and combinations of trigger signals. Once 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.

The Reveal Analyzer tool is started with the Reveal Analyzer Startup wizard where you can use an existing Reveal Analyzer file or create a new one.

Figure 96: Reveal Analyzer Setup Wizard



Reveal Analyzer Startup Wizard

Getting Started:

○ Create a new file: untitled

USB port:

Debug device:

RVL source:

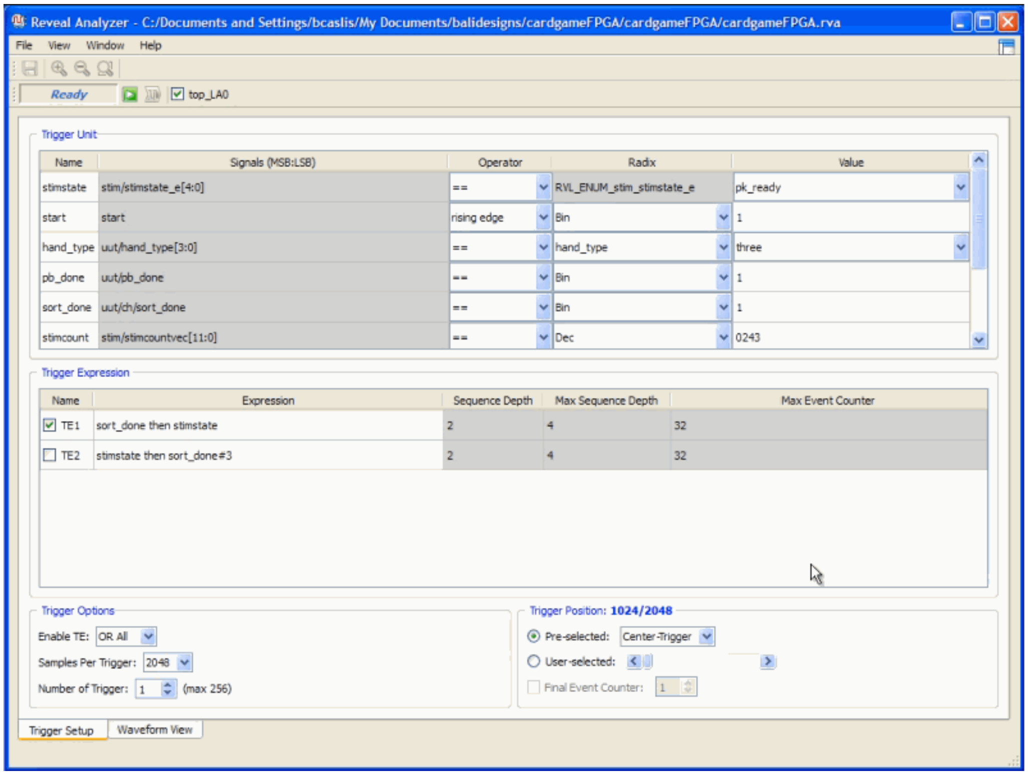
XCF source:

● Open an existing file

File name: C:/Documents and Settings/bcaslis/My Documents/baldesigns/cardgameFPGA/cardgameFPGA/cardgameFPGA.rva

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 97: Reveal Analyzer



Reveal Analyzer - C:/Documents and Settings/bcaslis/My Documents/balidesigns/cardgameFPGA/cardgameFPGA/cardgameFPGA.rva

☑ top_LA0

Trigger Unit

Name	Signals (MSB:LSB)	Operator	Radix	Value
stimstate	stim/stimstate_e[4:0]	==	RVL_ENUM_stim_stimstate_e	pk_ready
start	start	rising edge	Bin	1
hand_type	uut/hand_type[3:0]	==	hand_type	three
pb_done	uut/pb_done	==	Bin	1
sort_done	uut/ch/sort_done	==	Bin	1
stimcount	stim/stimcountvec[11:0]	==	Dec	0243

Trigger Expression

Name	Expression	Sequence Depth	Max Sequence Depth	Max Event Counter
☑ TE1	sort_done then stimstate	2	4	32
☐ TE2	stimstate then sort_done#3	2	4	32

Trigger Options

Enable TE: OR All

Samples Per Trigger: 2048

Number of Trigger: 1 (max 256)

Trigger Position: **1024/2048**

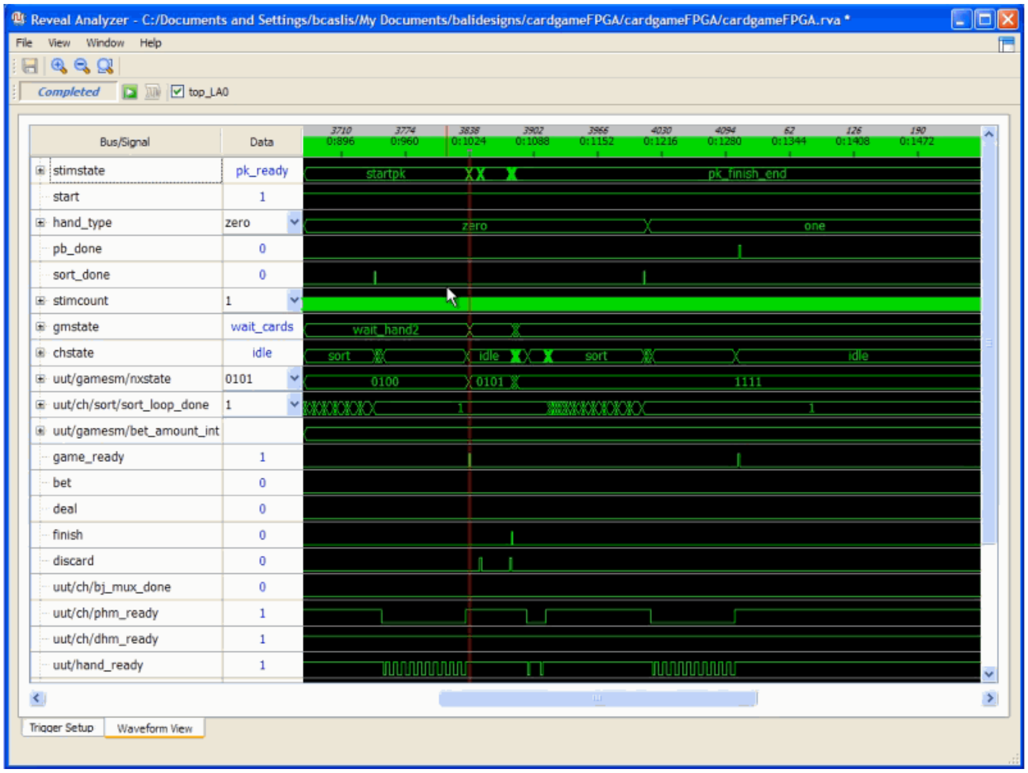
● Pre-selected: Center-Trigger

○ User-selected:

☐ Final Event Counter: 1

When you select **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 selection of Run also switches the display to the Waveform view.

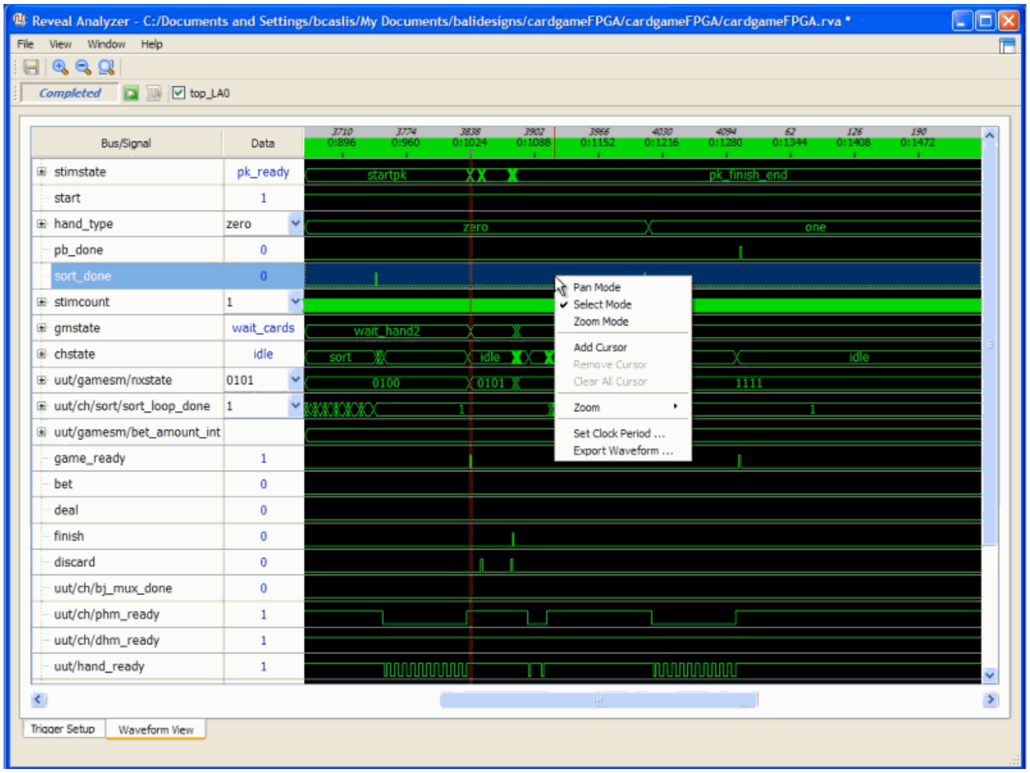
Figure 98: Reveal Analyzer waveform view



Bus/Signal	Data
stimstate	pk_ready
start	1
hand_type	zero
pb_done	0
sort_done	0
stimcount	1
gmstate	wait_cards
chstate	idle
uut/gamesm/nxstate	0101
uut/ch/sort/sort_loop_done	1
uut/gamesm/bet_amount_int	
game_ready	1
bet	0
deal	0
finish	0
discard	0
uut/ch/bj_mux_done	0
uut/ch/phm_ready	1
uut/ch/dhm_ready	1
uut/hand_ready	1

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 99: Reveal Analyzer waveform cursor controls



Reveal Analyzer - C:/Documents and Settings/bcaslis/My Documents/balidesigns/cardgameFPGA/cardgameFPGA/cardgameFPGA.rva *

Bus/Signal	Data
stimstate	pk_ready
start	1
hand_type	zero
pb_done	0
sort_done	0
stimcount	1
gmstate	wait_cards
chstate	idle
uut/gamesm/nxstate	0101
uut/ch/sort/sort_loop_done	1
uut/gamesm/bet_amount_int	
game_ready	1
bet	0
deal	0
finish	0
discard	0
uut/ch/bj_mux_done	0
uut/ch/phm_ready	1
uut/ch/dhm_ready	1
uut/hand_ready	1

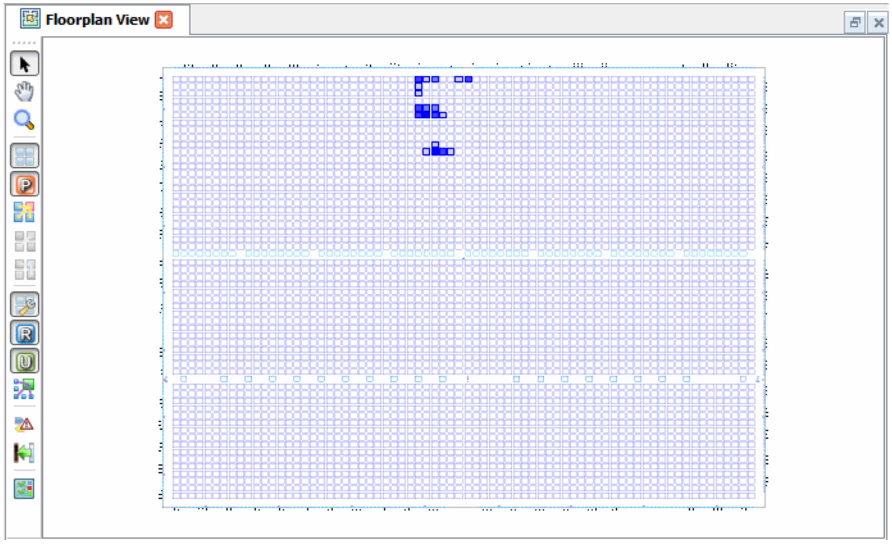
The Data column in the view shows the data for the active cursor. The Reveal Analyzer supports multiple cursors which can be added, removed and position changed within the waveform. Selecting the cursor and dragging it will produce a rubber band effect which can be used for measuring time intervals.

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 100: 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 displayed in tooltips and in the status bar include:

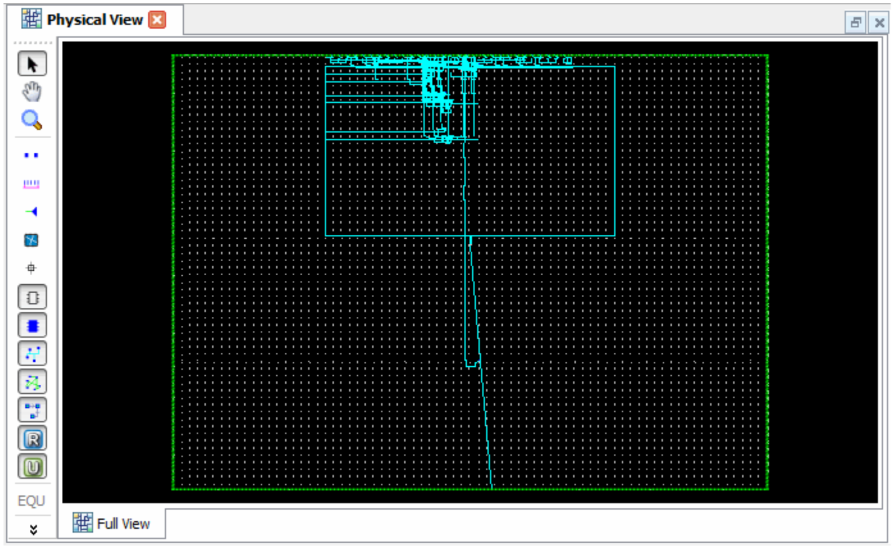
- number of resources for each UGROUP and REGION
- number of utilized slices for each PLC component
- name and location of each component, port, net, and site

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 flylines.

Figure 101: Physical View



As you move your mouse 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.

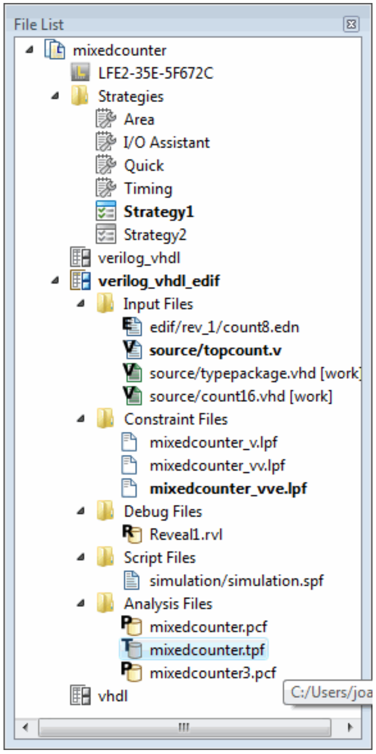
The Physical View toolbar allows you to select the types of elements that will be displayed on the layout, including components, empty sites, switch boxes, switches, pin wires, routes, and timing paths.

Timing Analysis View

The Timing Analysis view provides a graphical way to navigating timing information and a fast timing analysis loop allowing dynamic path changes without having to re-implement or re-map 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 102: TPF file

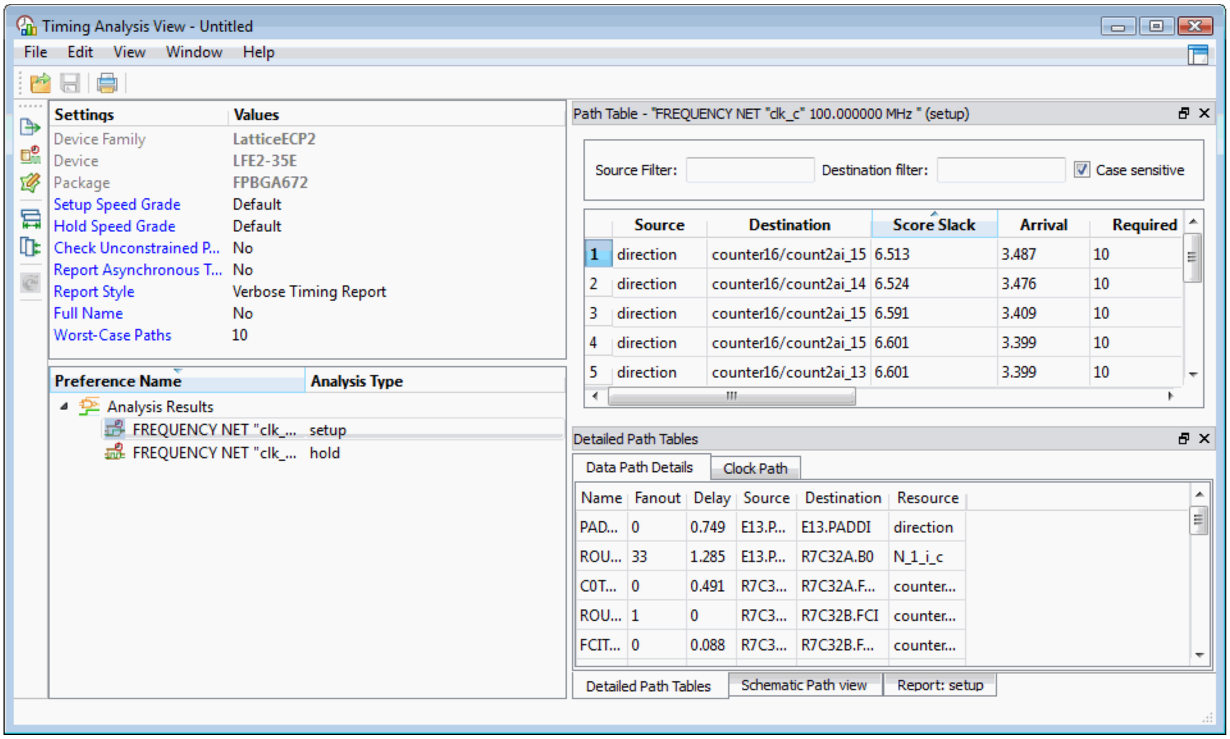


```
File List
mixedcounter
    LFE2-35E-5F672C
    Strategies
        Area
        I/O Assistant
        Quick
        Timing
        Strategy1
        Strategy2
    verilog_vhdl
    verilog_vhdl_edif
        Input Files
            edif/rev_1/count8.edn
            source/topcount.v
            source/typepackage.vhd [work]
            source/count16.vhd [work]
        Constraint Files
            mixedcounter_v.lpf
            mixedcounter_vv.lpf
            mixedcounter_vve.lpf
        Debug Files
            Reveal1.rvl
        Script Files
            simulation/simulation.spf
        Analysis Files
            mixedcounter.pcf
            mixedcounter.tpf
            mixedcounter3.pcf
    vhdl                                C:/Users/joa
```

Launching Timing Analysis

When the Timing Analysis tool is launched it uses the active TPF file or, if it is launched when there is no active TPF file, it will read timing preferences from the LPF file for its initial timing calculations. Additionally you can double-click on any active or inactive TPF file in the File List and the Timing Analysis view will be launched using those TPF settings.

Figure 103: Timing Analysis View



Settings	Values
Device Family	LatticeECP2
Device	LFE2-35E
Package	FPBGA672
Setup Speed Grade	Default
Hold Speed Grade	Default
Check Unconstrained P...	No
Report Asynchronous T...	No
Report Style	Verbose Timing Report
Full Name	No
Worst-Case Paths	10

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

Path Table - "FREQUENCY NET "clk_c" 100.000000 MHz " (setup)

Source Filter: Destination filter: ☑ Case sensitive

	Source	Destination	Score Slack	Arrival	Required
1	direction	counter16/count2ai_15	6.513	3.487	10
2	direction	counter16/count2ai_14	6.524	3.476	10
3	direction	counter16/count2ai_15	6.591	3.409	10
4	direction	counter16/count2ai_15	6.601	3.399	10
5	direction	counter16/count2ai_13	6.601	3.399	10

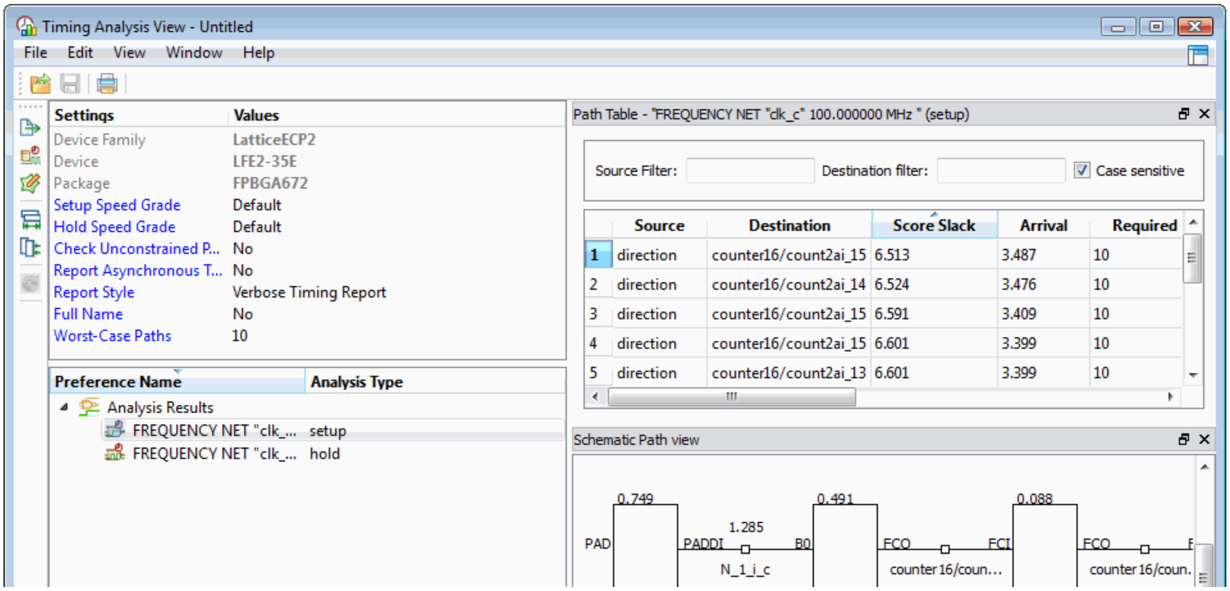
Detailed Path Tables

Data Path Details | Clock Path

Name	Fanout	Delay	Source	Destination	Resource
PAD...	0	0.749	E13.P...	E13.PADDI	direction
ROU...	33	1.285	E13.P...	R7C32A.B0	N_1_i_c
COT...	0	0.491	R7C3...	R7C32A.F...	counter...
ROU...	1	0	R7C3...	R7C32B.FCI	counter...
FCIT...	0	0.088	R7C3...	R7C32B.F...	counter...

The Timing Analysis view consists of sections displaying the trace settings from your active Strategy, preferences and views of the path table, detailed path tables, schematic path and report. The Preferences section lists any preferences from the active LPF or TPF file. Here is the Schematic path view.

Figure 104: Timing Analysis Schematic Path View



Settings	Values
Device Family	LatticeECP2
Device	LFE2-35E
Package	FPBGA672
Setup Speed Grade	Default
Hold Speed Grade	Default
Check Unconstrained P...	No
Report Asynchronous T...	No
Report Style	Verbose Timing Report
Full Name	No
Worst-Case Paths	10

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

Path Table - "FREQUENCY NET "clk_c" 100.000000 MHz " (setup)

Source Filter: Destination filter: ☑ Case sensitive

	Source	Destination	Score Slack	Arrival	Required
1	direction	counter16/count2ai_15	6.513	3.487	10
2	direction	counter16/count2ai_14	6.524	3.476	10
3	direction	counter16/count2ai_15	6.591	3.409	10
4	direction	counter16/count2ai_15	6.601	3.399	10
5	direction	counter16/count2ai_13	6.601	3.399	10

Additionally the Timing Analysis specific toolbar on the left of the view contains the following controls:

Export - timing paths to cvs file

Settings - change settings for specific timing analysis run. To change the settings permanently must be changed in the trace settings of your Strategy.

Change Timing Preferences - displays the Timing Preferences tab of the Spreadsheet view. This is a view of the timing preferences file (TPF).

Fit All Columns - display control

View All Columns - display control

Update - rotates to indicate a preference has been changed. Select to recalculate timing.

Importing LPF settings

To load the LPF settings into the currently open TPF use the **File > Import > Copy LPF to TPF** selection. This is a good way to update your TPF file with any changes you may have made in the LPF file.

Updating and saving TPF settings

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

1. From the Timing Analysis view, select the **Change Timing Preferences** button on the left. This opens up the Spreadsheet View > Timing Preferences view.

2. Modify a preference in the Timing Preferences view. Notice that an asterix indicating a data change appears in both the Spreadsheet and Timing Analysis tabs.

3. Select the Timing Analysis view again and select the rotating **Update** button on the left. This will recalculate timing.

4. To save the TPF file first make sure the Timing Analysis view is active and then select **File > Save**.

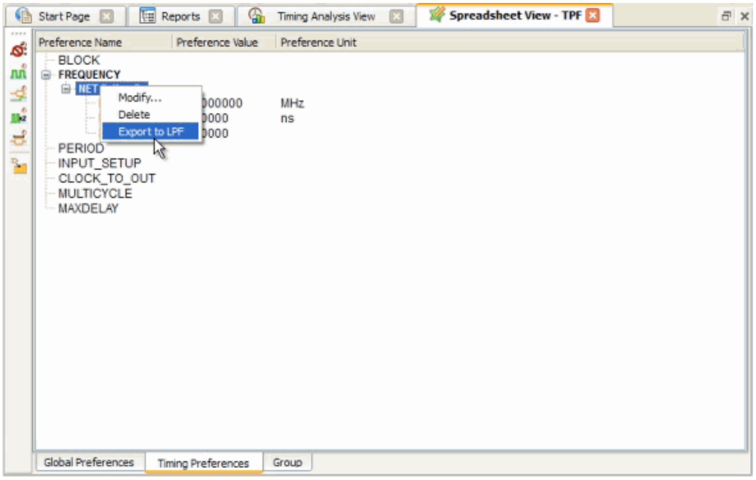
Note that the Timing Analysis view must be active to use File > Save. You can't save timing preferences from the Spreadsheet >Timing Preferences view.

Also notice that the Update button will be rotating only when a preference has been changed but the timing has not yet been recalculated.

Exporting TPF settings

If you want to save your TPF settings into the LPF file first select the Timing Preferences view option in the Spreadsheet View. Right-click on the item you want to export and select **Export to LPF**. This will export the selected setting to the active LPF file.

Figure 105: Export TPF

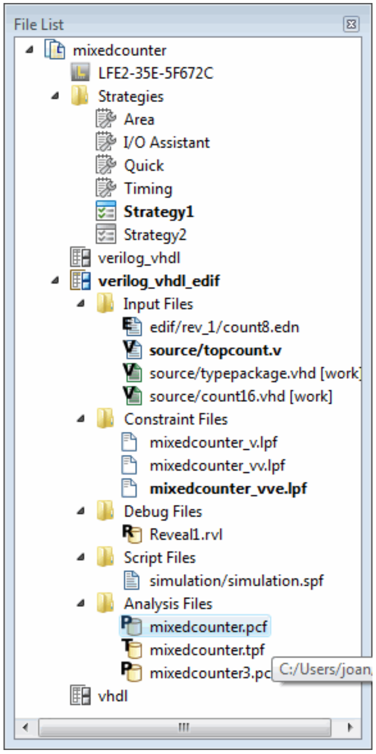


Power Calculator

The Power Calculator tool estimates the power dissipation for your 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.

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

Figure 106: Power Calculator file



File List

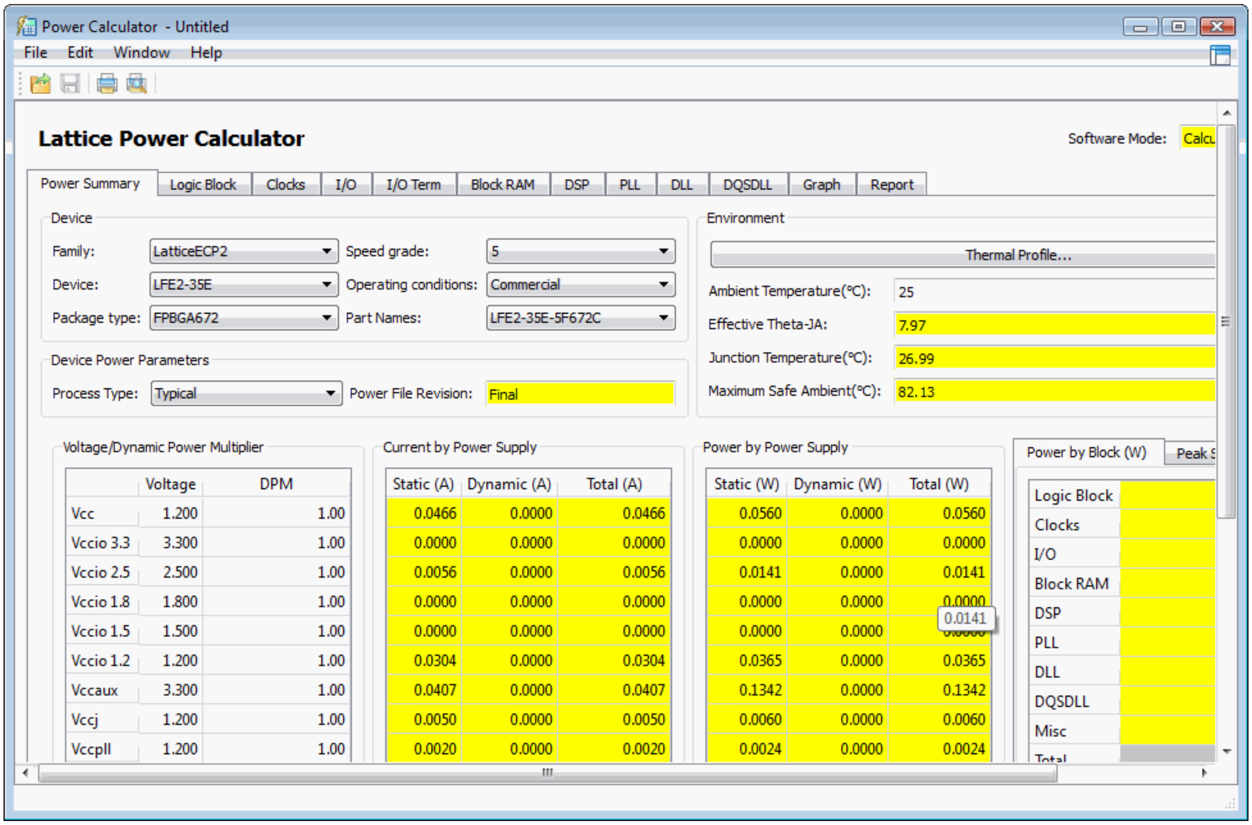
- mixedcounter
  - LFE2-35E-5F672C
  - Strategies
    - Area
    - I/O Assistant
    - Quick
    - Timing
    - **Strategy1**
    - Strategy2
  - verilog_vhdl
  - **verilog_vhdl_edif**
    - Input Files
      - edif/rev_1/count8.edn
      - **source/topcount.v**
      - source/typepackage.vhd [work]
      - source/count16.vhd [work]
    - Constraint Files
      - mixedcounter_v.lpf
      - mixedcounter_vv.lpf
      - **mixedcounter_vve.lpf**
    - Debug Files
      - Reveal1.rvl
    - Script Files
      - simulation/simulation.spf
    - Analysis Files
      - mixedcounter.pcf
      - mixedcounter.tpf
      - mixedcounter3.pc C:/Users/joan
  - vhdl

Power Calculator can be launched in multiple ways and will use different PCF files according to the following conditions:

- If there is an active PCF file it will be used if Power calculator is launched from the menu selection **Tools > Power Calculator** or the toolbar icon. Also if there is only one PCF file, even if it is inactive, it will be used at launch.
- Any PCF file, active or inactive, can be double-clicked in the File List Analysis Files folder and that PCF file will be used to launch Power Calculator.
- If no PCF files exist Power Calculator will launch and perform power calculations based on the current open design.

The Power Calculator view consists of a Power Summary tab as well as tabs based on the specific target device.

Figure 107: Power Summary



Power Calculator - Untitled
File Edit Window Help
Lattice Power Calculator
Power Summary | Logic Block | Clocks | I/O | I/O Term | Block RAM | DSP | PLL | DLL | DQSDLL | Graph | Report
Thermal Profile...
0.0141

Software Mode: Calcu

Device

Family:	LatticeECP2	Speed grade:	5
Device:	LFE2-35E	Operating conditions:	Commercial
Package type:	FPBGA672	Part Names:	LFE2-35E-5F672C

Device Power Parameters

Process Type: Typical   Power File Revision: Final

Environment

Ambient Temperature(°C):	25
Effective Theta-JA:	7.97
Junction Temperature(°C):	26.99
Maximum Safe Ambient(°C):	82.13

Voltage/Dynamic Power Multiplier — Current by Power Supply — Power by Power Supply

	Voltage	DPM	Static (A)	Dynamic (A)	Total (A)	Static (W)	Dynamic (W)	Total (W)
Vcc	1.200	1.00	0.0466	0.0000	0.0466	0.0560	0.0000	0.0560
Vccio 3.3	3.300	1.00	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Vccio 2.5	2.500	1.00	0.0056	0.0000	0.0056	0.0141	0.0000	0.0141
Vccio 1.8	1.800	1.00	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Vccio 1.5	1.500	1.00	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Vccio 1.2	1.200	1.00	0.0304	0.0000	0.0304	0.0365	0.0000	0.0365
Vccaux	3.300	1.00	0.0407	0.0000	0.0407	0.1342	0.0000	0.1342
Vccj	1.200	1.00	0.0050	0.0000	0.0050	0.0060	0.0000	0.0060
Vccpll	1.200	1.00	0.0020	0.0000	0.0020	0.0024	0.0000	0.0024

Power by Block (W) | Peak S

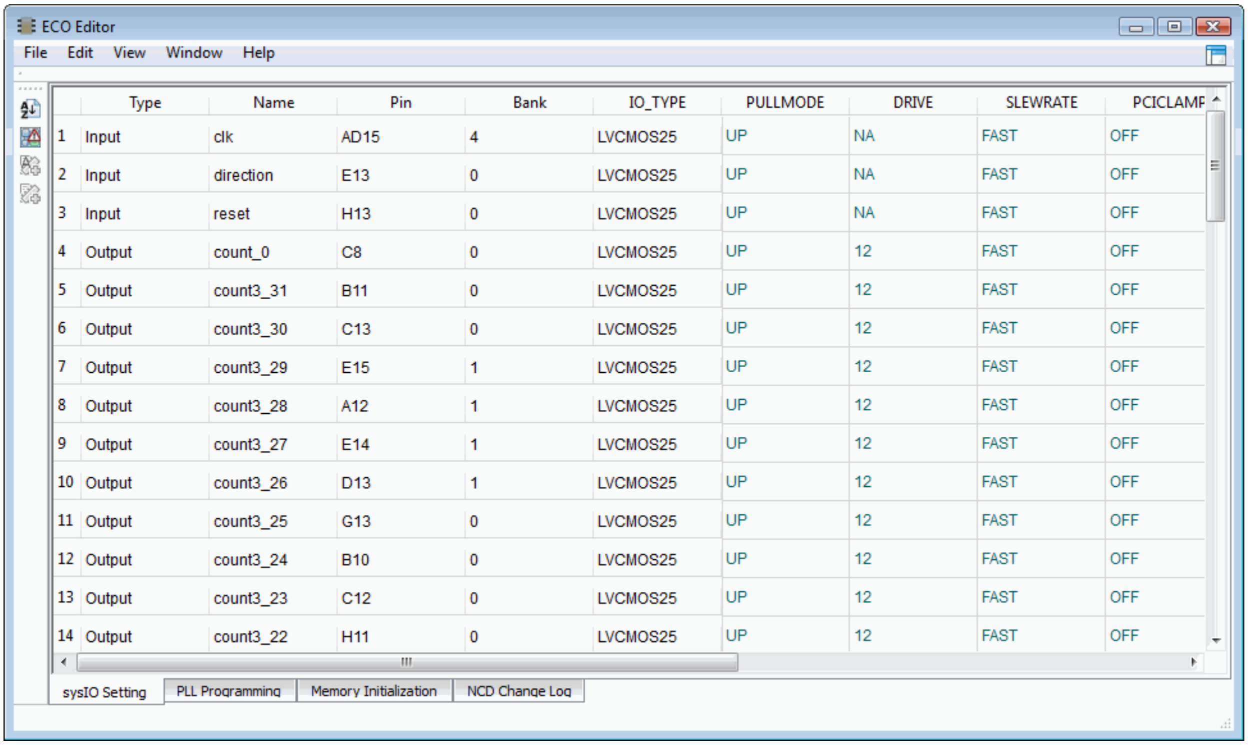
Logic Block	
Clocks	
I/O	
Block RAM	
DSP	
PLL	
DLL	
DQSDLL	
Misc	
Total	

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

ECO Editor

The Engineering Change Order (ECO) Editor allows you to safely and conveniently to select changes to the NCD netlist after place and route.

Figure 108: ECO Editor



	Type	Name	Pin	Bank	IO_TYPE	PULLMODE	DRIVE	SLEWRATE	PCICLAMP
1	Input	clk	AD15	4	LVCMOS25	UP	NA	FAST	OFF
2	Input	direction	E13	0	LVCMOS25	UP	NA	FAST	OFF
3	Input	reset	H13	0	LVCMOS25	UP	NA	FAST	OFF
4	Output	count_0	C8	0	LVCMOS25	UP	12	FAST	OFF
5	Output	count3_31	B11	0	LVCMOS25	UP	12	FAST	OFF
6	Output	count3_30	C13	0	LVCMOS25	UP	12	FAST	OFF
7	Output	count3_29	E15	1	LVCMOS25	UP	12	FAST	OFF
8	Output	count3_28	A12	1	LVCMOS25	UP	12	FAST	OFF
9	Output	count3_27	E14	1	LVCMOS25	UP	12	FAST	OFF
10	Output	count3_26	D13	1	LVCMOS25	UP	12	FAST	OFF
11	Output	count3_25	G13	0	LVCMOS25	UP	12	FAST	OFF
12	Output	count3_24	B10	0	LVCMOS25	UP	12	FAST	OFF
13	Output	count3_23	C12	0	LVCMOS25	UP	12	FAST	OFF
14	Output	count3_22	H11	0	LVCMOS25	UP	12	FAST	OFF

ECOs are requests to make small changes to your design after place and route. The changes are directly written into the native circuit description (NCD) database file 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, or to facilitate changes that are made to the design specification to compensate for design problems that are introduced while integrating other FPGAs or components of your PC board design.

The ECO Editor’s spreadsheet includes tabs for editing I/O settings, PLL settings and memory initialization values. It also provides a Change Log window to track all changes to the NCD file since it was originally written by the place and route process.

Be aware that once you edit your post-PAR, routed NCD file, your functional simulation and timing simulation will no longer match.

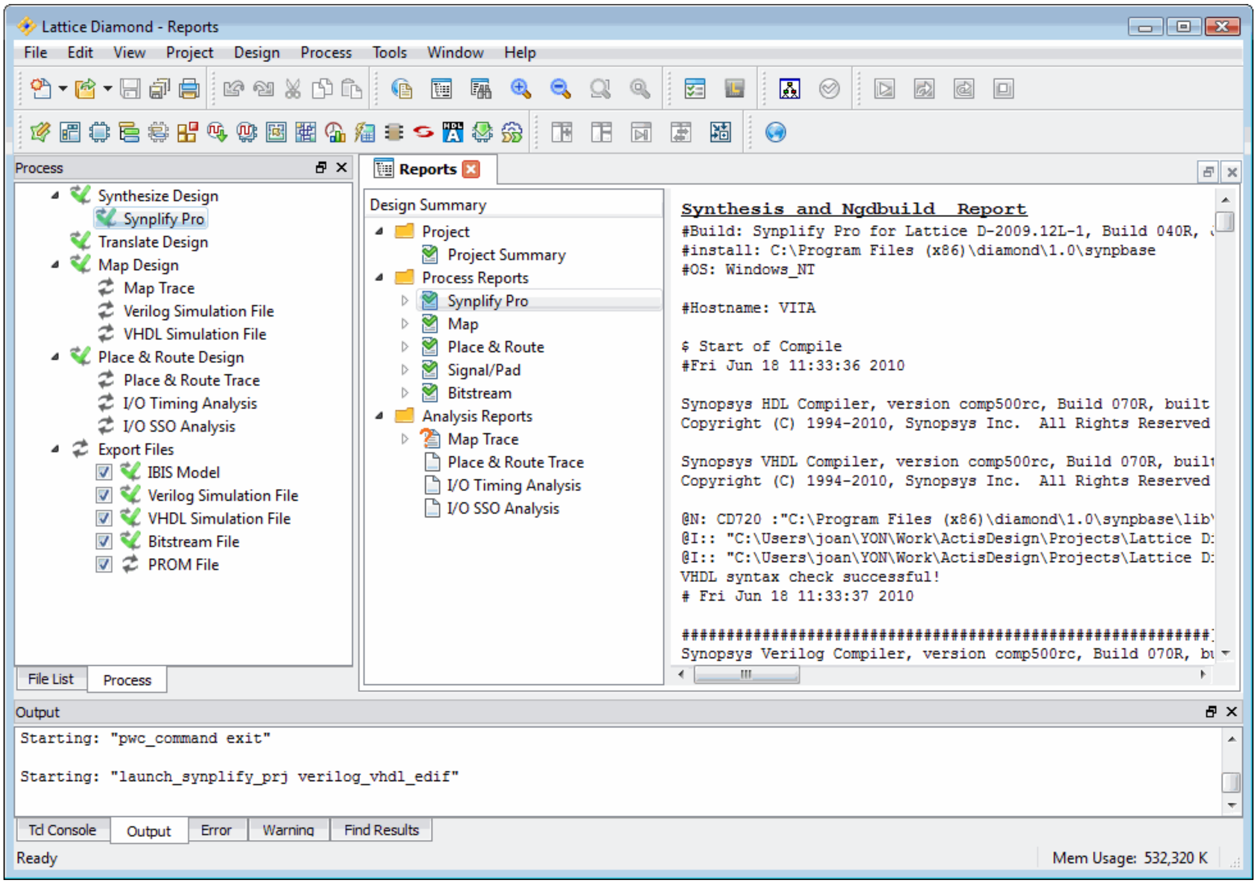
For more information see *Applying Engineering Change Orders* in the Lattice Diamond online documentation.

Synplify Pro for Lattice

The Synplify Pro for Lattice tool is an external synthesis tool used in the Lattice Diamond design flow. It is launched by selecting the Synthesize Design step in the Process view or from the Tool menu or toolbar.

Synplify Pro is run in batch mode when run as part of the Process flow in Lattice Diamond. To look at the output report from Synplify Pro select the **Process Reports > Synplify Pro** selection in the Reports view Design Summary section.

Figure 109: Synthesis report



```
Synthesis and Ngdbuild  Report
#Build: Synplify Pro for Lattice D-2009.12L-1, Build 040R, J
#install: C:\Program Files (x86)\diamond\1.0\synpbase
#OS: Windows_NT

#Hostname: VITA

$ Start of Compile
#Fri Jun 18 11:33:36 2010

Synopsys HDL Compiler, version comp500rc, Build 070R, built
Copyright (C) 1994-2010, Synopsys Inc.  All Rights Reserved

Synopsys VHDL Compiler, version comp500rc, Build 070R, built
Copyright (C) 1994-2010, Synopsys Inc.  All Rights Reserved

@N: CD720 :"C:\Program Files (x86)\diamond\1.0\synpbase\lib\
@I:: "C:\Users\joan\YON\Work\ActisDesign\Projects\Lattice D:
@I:: "C:\Users\joan\YON\Work\ActisDesign\Projects\Lattice D:
VHDL syntax check successful!
# Fri Jun 18 11:33:37 2010

################################################################
Synopsys Verilog Compiler, version comp500rc, Build 070R, bu
```

```
Starting: "pwc_command exit"

Starting: "launch_synplify_prj verilog_vhdl_edif"
```

To run Synplify Pro in interactive mode it can be launched from Lattice Diamond with the **Tools > Synplify Pro** menu or toolbar selection.

For more information on Synplify Pro see the Synplify User Guide in the Lattice Diamond product documentation.

Active-HDL Lattice Edition

The Active-HDL Lattice Edition tool is a simulation tool closely linked but external to the Lattice Diamond environment. It is not run as part of the Process implementation flow. Active-HDL is launched by selecting the Active-HDL selection in the Tool menu or toolbar.

See the *Simulation Flow* section of this manual for more information on simulating your design and see the *Simulation Wizard* section of this manual for information on creating a simulation project to run in Active-HDL.

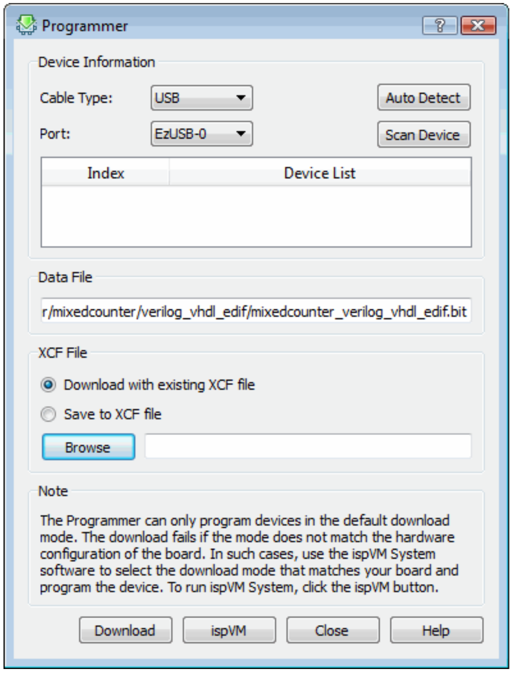
For more information on Active-HDL see the Active-HDL User Guide in the Lattice Diamond product documentation.

Programmer

The Programmer tool provides a place within the Lattice Diamond implementation environment to program the target device. Programmer incorporates a subset of the features of ispVM System, including detecting cable type, scanning device chain, creating XCF file, and downloading the data file to the device.

After placing and routing a design and generating the JEDEC or bitstream file, you can launch the Programmer to download the data file into the target device.

Figure 110: Programmer



Device Information

Cable Type: USB

Port: EzUSB-0

Index	Device List

Data File

r/mixedcounter/verilog_vhdl_edif/mixedcounter_verilog_vhdl_edif.bit

XCF File

● Download with existing XCF file

○ Save to XCF file

Note

The Programmer can only program devices in the default download mode. The download fails if the mode does not match the hardware configuration of the board. In such cases, use the ispVM System software to select the download mode that matches your board and program the device. To run ispVM System, click the ispVM button.

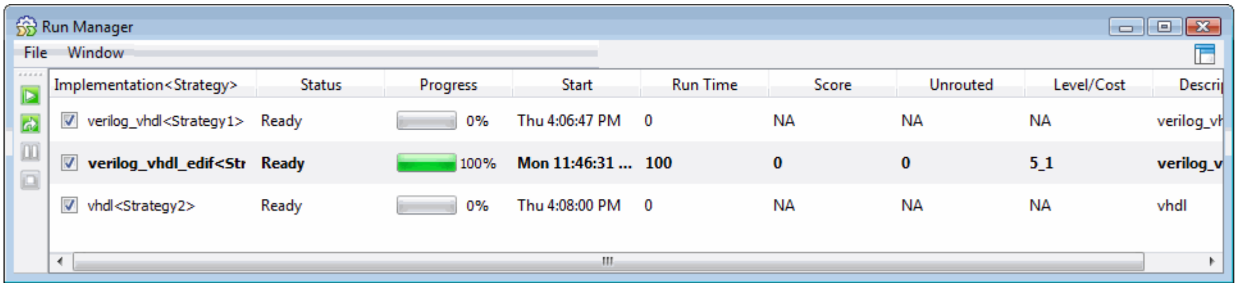
Programmer supports single device programming for the current design using the default programming options. For complex device programming tasks, use the stand-alone ispVM System software.

For more information on programming a device with the Programmer see the Lattice Diamond product documentation.

Run Manager

The Run Manager tool is used to run different implementation/strategy combinations. Select **Tools > Run Manager** or use the icon from the toolbar.

Figure 111: Run Manager



Implementation<Strategy>	Status	Progress	Start	Run Time	Score	Unrouted	Level/Cost	Descri
☑ verilog_vhdl<Strategy1>	Ready	0%	Thu 4:06:47 PM	0	NA	NA	NA	verilog_vh
☑ **verilog_vhdl_edif<Str**	**Ready**	100%	**Mon 11:46:31 ...**	**100**	**0**	**0**	**5_1**	**verilog_v**
☑ vhdl<Strategy2>	Ready	0%	Thu 4:08:00 PM	0	NA	NA	NA	vhdl

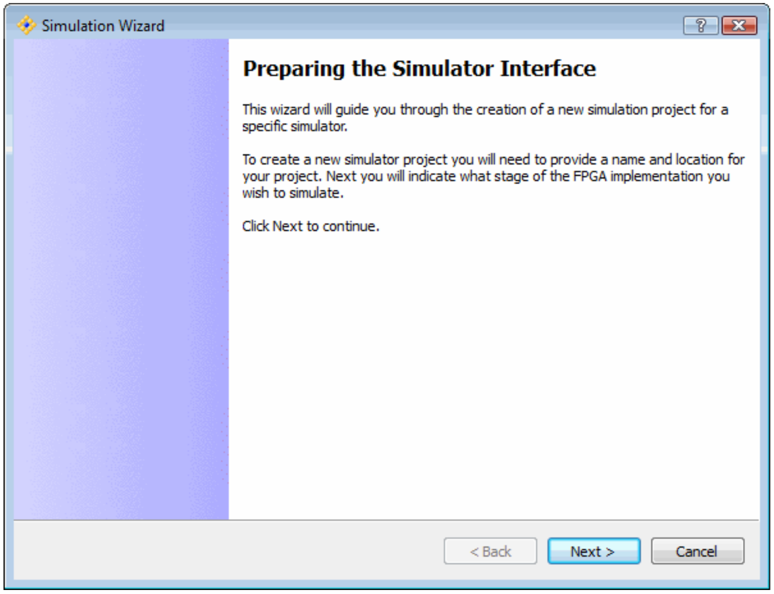
The Run Manager runs the entire process flow for each selected implementation. If you are running on a multicore system the run manager will distribute the runs to execute in parallel. There is an option in Tools > Options to set the maximum number of processes to run in parallel. Generally this should be set to the number of cores in your processor, but if you are using the Multi-Tasking node list option for PAR it should be set to one. Refer to Lattice Diamond online help for more information.

To examine the reports from each process, first make an implementation active in the File List and then select the Reports view.

Simulation Wizard

The Simulation Wizard tool leads you through the process of creating a simulation project for your design. Launch the wizard by using the **Tools > Simulation Wizard** menu selection. The wizard will lead you through a series of steps including selecting a simulation project name and location, which simulator to use (if you have more than one installed), selecting the process stage to use (from RTL to Post-Route Gate-Level + Timing), language (VHDL or Verilog) and source files. You can optionally run the simulation directly from the wizard.

Figure 112: Simulation wizard



Preparing the Simulator Interface

This wizard will guide you through the creation of a new simulation project for a specific simulator.

To create a new simulator project you will need to provide a name and location for your project. Next you will indicate what stage of the FPGA implementation you wish to simulate.

Click Next to continue.

Common Tasks

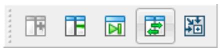
Lattice Diamond gathers the many FPGA implementation tools into one central design environment. The benefit to the user is a common location for all the tools, common controls and sharing data between the tools.

Controlling Tool Views

Tool views are highly configurable in the Lattice Diamond environment. Detaching and attaching views and manipulating tab groups are two key features that allow you to customize the tool views to meet your requirements.

When tools are initially launched they will appear in the main window as a new tab. Multiple tools can be launched and appear in the tabs. The Window toolbar contains buttons for controlling the display of multiple tool tabs.

Figure 113: Multiple tool tab controls



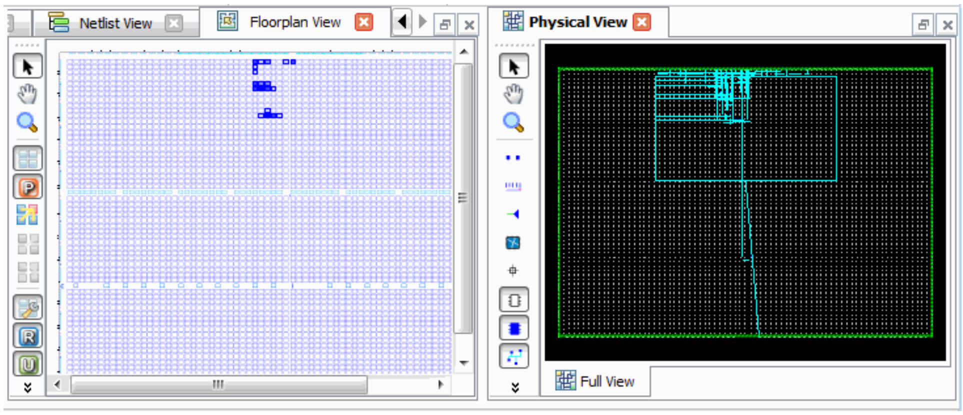
If you need to see more data, the tool view can be detached from the main window using the detach button.

Tool views can be docked back into the main window using the attach button.

If you have multiple tool views and want to dock them all back into the main window you can use the **Integrate All Tools** control.

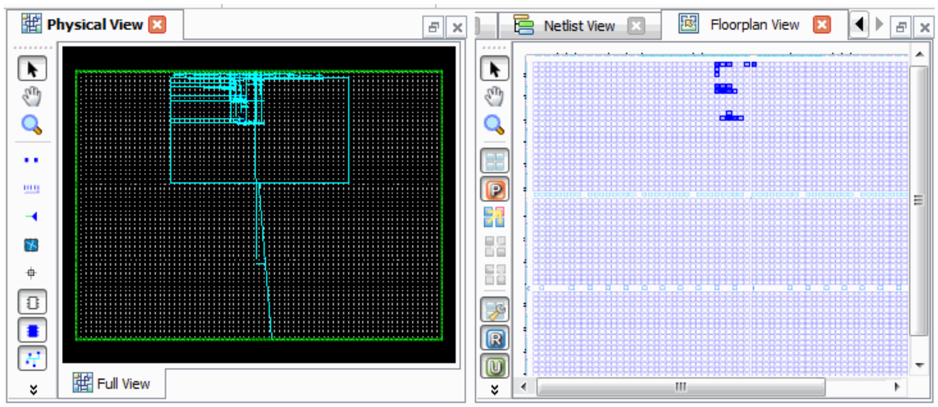
Since the tools all reference the same shared design memory and you can use cross probing to see the same data element in multiple tool views, it is very useful to be able to view two tool views side by side within the main window using **Split Tab Group**.

Figure 114: Split Tab Group shows two views



You can switch the position of the tool views within a tab group using the **Switch Tab Group Position control**.

Figure 115: Switch Tab Group Position changes the position of two tabs



You can rotate the views into different tab groups using the **Move to Another Tab Group** control.

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 for fitting a selected area to the window view area.

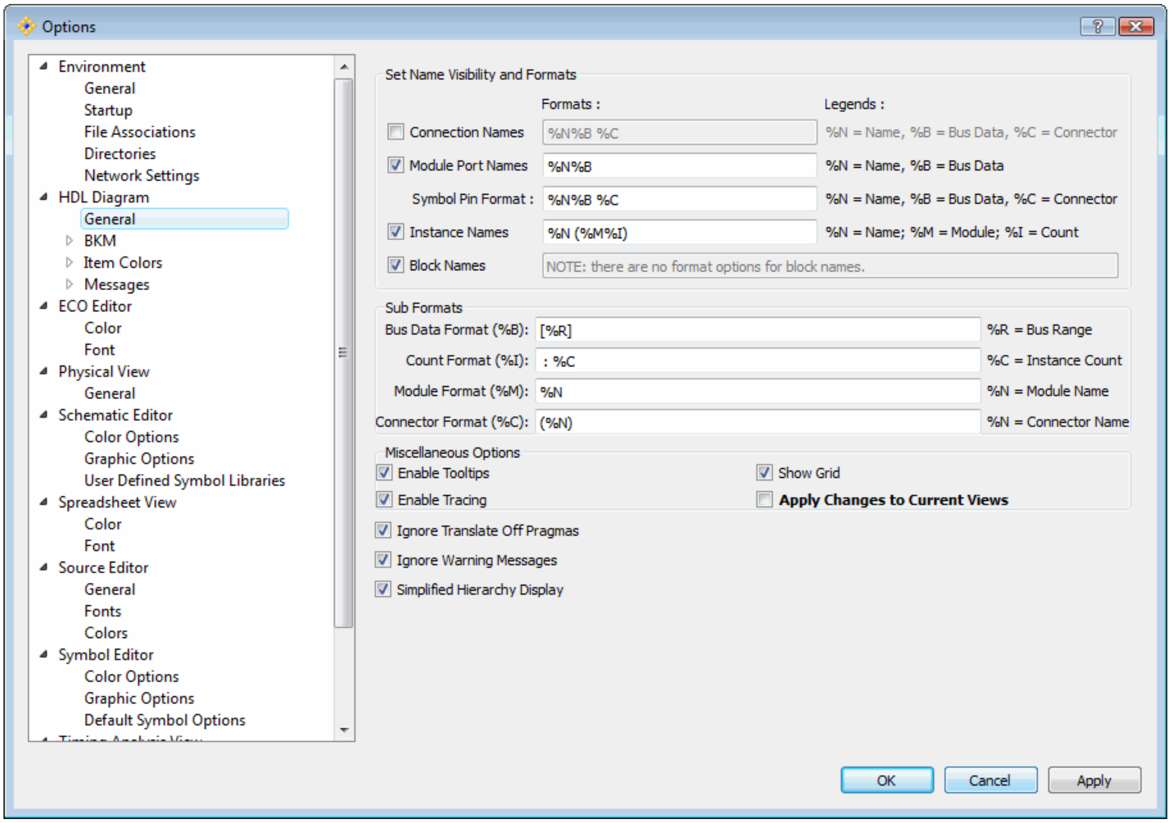
Tooltips

When you place the cursor over a graphical element in a tool view a pop-up message displays containing information on the element. These pop-ups are called *tooltips*. The same information displayed in the tooltip pop-up will also temporarily be displayed in the status bar at the lower left of the main window.

Setting display options

The **Tools > Options** selection allows you to customize display options for many of the tools. The Options selections are organized by tool and include selections for color, font and other graphic elements.

Figure 116: Tools > Options > HDL Diagram



Set Name Visibility and Formats

	Formats :	Legends :
☐ Connection Names	%N%B %C	%N = Name, %B = Bus Data, %C = Connector
☑ Module Port Names	%N%B	%N = Name, %B = Bus Data
Symbol Pin Format :	%N%B %C	%N = Name, %B = Bus Data, %C = Connector
☑ Instance Names	%N (%M%I)	%N = Name; %M = Module; %I = Count
☑ Block Names	NOTE: there are no format options for block names.	

Sub Formats

Bus Data Format (%B):	[%R]	%R = Bus Range
Count Format (%I):	: %C	%C = Instance Count
Module Format (%M):	%N	%N = Module Name
Connector Format (%C):	(%N)	%N = Connector Name

Miscellaneous Options

☑ Enable Tooltips ☑ Show Grid
☑ Enable Tracing ☐ **Apply Changes to Current Views**

☑ Ignore Translate Off Pragmas
☑ Ignore Warning Messages
☑ Simplified Hierarchy Display

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.

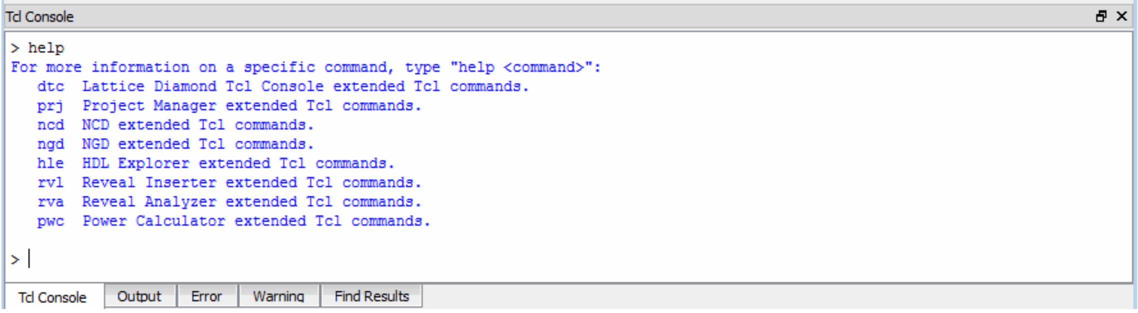
Overview

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 display the TCL Console by selecting its tab located 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 117: TCL Console and command groups



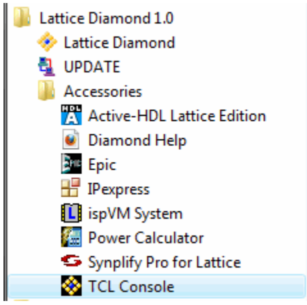
```
> help
For more information on a specific command, type "help <command>":
   dtc  Lattice Diamond Tcl Console extended Tcl commands.
   prj  Project Manager extended Tcl commands.
   ncd  NCD extended Tcl commands.
   ngd  NGD extended Tcl commands.
   hle  HDL Explorer extended Tcl commands.
   rvl  Reveal Inserter extended Tcl commands.
   rva  Reveal Analyzer extended Tcl commands.
   pwc  Power Calculator extended Tcl commands.

> |
```

The TCL Console can be opened, closed, detached and attached (using the double-click method).

External TCL Console

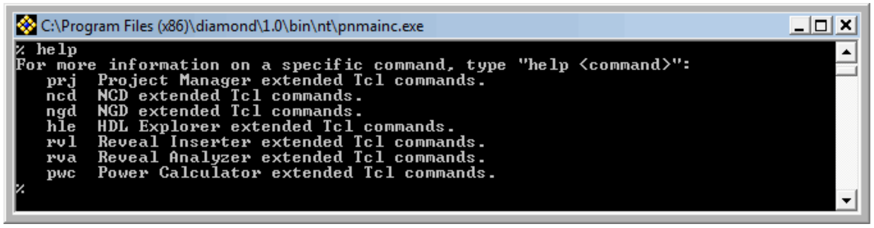
All of the Lattice Diamond Tcl commands available in the internal TCL Console are also available from an external console. In the folder where you launched Lattice Diamond is an **Accessories** folder containing an external TCL Console.

Figure 118: Launching an external TCL console



Select the 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.

Figure 119: Running commands in an external TCL shell



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

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.
```

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.

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_writereport    Power Calculator writereport command
```

See the *Advanced Topics* chapter for more information on writing Tcl scripts.

9

Advanced Topics

This chapter contains a collection of information on 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 how to best manage the environment for optimum performance, especially when your design is large.

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

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

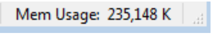
External tools referenced from with Lattice Diamond, such as for synthesis and simulation, use their own memory in addition to what is used by Lattice Diamond.

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 120: Memory usage



Mem Usage: 235,148 K

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

Clear Tool Memory

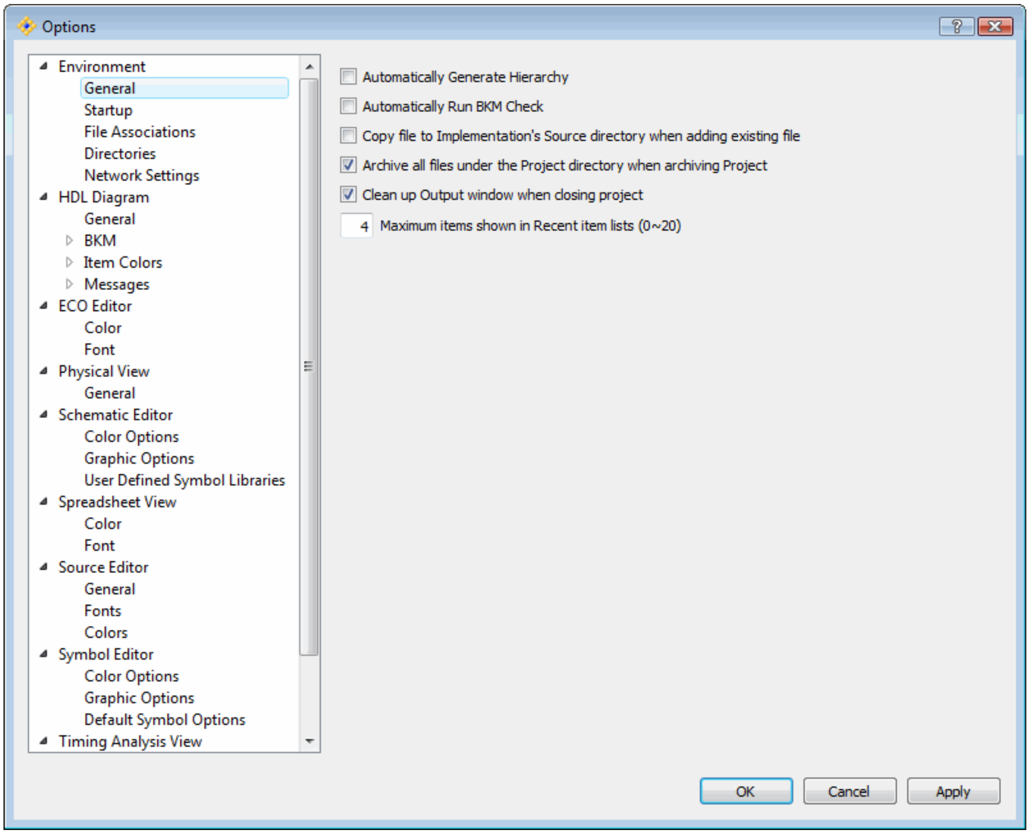
The **Tools > Clear Tool Memor**y selection clears the device, design, and preference information and the HDL Diagram database from system memory. Clearing the tool memory can speed up memory intensive processes such as place and route. If your design is very large it is a good practice to clear memory prior to running place and route.

If you have open tool views which will be effected by clearing the tool memory a confirmation dialog be displayed to give you the opportunity to cancel the memory clear.

Environment and tool options

There are many environment control and tool view display parameters you can customize in Lattice Diamond. This section shows how to configure some of the significant options for your environment. Select **Tools > Options** to display the option list.

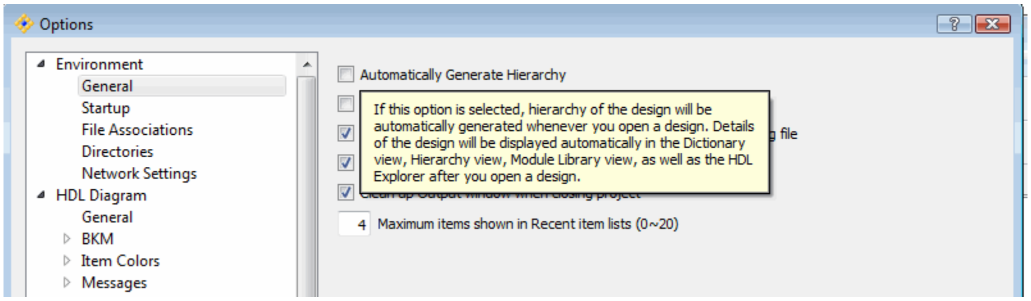
Figure 121: Environment and tool options



□ Automatically Generate Hierarchy
□ Automatically Run BKM Check
□ Copy file to Implementation's Source directory when adding existing file
☑ Archive all files under the Project directory when archiving Project
☑ Clean up Output window when closing project
4 Maximum items shown in Recent item lists (0~20)

The selections are organized into functional folders. You can use the context sensitive help function **?** in the upper right of the Options window to get information about each parameter in a folder. Select **?** and then click on the item of interest.

Figure 122: Help information



□ Automatically Generate Hierarchy

If this option is selected, hierarchy of the design will be automatically generated whenever you open a design. Details of the design will be displayed automatically in the Dictionary view, Hierarchy view, Module Library view, as well as the HDL Explorer after you open a design.

4 Maximum items shown in Recent item lists (0~20)

Some of the commonly configured items include:

- **Environment > General** includes a selection to **Automatically Generate Hierarchy**. If this is selected the Generate Hierarchy function is run whenever a project is opened.
- **Environment > Startup** includes selections 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** contains settings for configuration of the checks performed in BKM checking.

There are also a number of tool view and editor folders containing customizable display properties such as color and font.

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. Select **Reference Guides > Command Line** on the Lattice Diamond Start Page for more information.

IPexpress flow

IPexpress provides a variety of modules to use as building blocks in your design. These modules cover a variety of common functions and can be customized. They are optimized for Lattice device architectures. Use these modules to speed your design work and to get the most effective results.

Tcl scripts

Creating your own custom scripts using the Diamond Extended Tcl language can save you a lot of time. If there are tasks you perform often in Lattice Diamond, you can automate the operations using a script. Not only do scripts save time but they also provide a uniform approach to design that can be helpful when you try to find an optimal solution for your design with 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 method allows you to easily 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. (Optional) Enter the history command in the Tcl Console window to ensure 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 **save_script** script name values, but *my$script* or *my script* are invalid. File paths using forward slashes used with an identifier are valid if using an absolute file path to an existing folder.

The script will be saved in your project folder. 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 some other place you generally keep your scripts. However, you must save it 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 have 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 some conflict or exception. You will likely have 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 *notepad* or *vi* or any text editor to create a file and directly write in the Tcl commands.

Sample Tcl Script

The following Tcl example shows a very simple script that opens a project, runs the entire design flow through the Place & Route process, then closes the project. A typical script would probably contain more steps but for sake of demonstration, use this simple 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 run scripts from within the TCL Console with or without your project opened. You can also run from the standalone Tcl Console prompt window.

For example, to run a Tcl script in Lattice Diamond 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 on the TCL Console tab in the Output area at the bottom to open the console.

3. (Optional) If there are previously issued commands in the console, enter reset in the console command line to refresh your session and clear out all previous commands.

```
reset
```

4. Now, run your source command using the absolute file name and path to your script. As the sample script in the previous section, this script opens, runs the flow through PAR, and closes the project.

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

As long as there are no syntax errors or invalid arguments, this 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. Go back to your script and examine and edit your script as needed.

Index