Lattice Diamond Design Flow Overview for Xilinx Vivado Users

User Guide

FPGA-UG-02169-1.0

August 2022

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults and associated risk the responsibility entirely of the Buyer. Buyer shall not rely on any data and performance specifications or parameters provided herein. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. No Lattice products should be used in conjunction with mission- or safety-critical or any other application in which the failure of Lattice's product could create a situation where personal injury, death, severe property or environmental damage may occur. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Contents

Figures

Tables

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
ASC	Analog Sense and Control
CDT	C/C++ Development Tools
CPLD	Complex Programmable Logic Device
CPU	Central Processing Unit
DDR	Double Data Rate
DLL	Delay-Locked Loop
DSP	Digital Signal Processing
EDA	Electronic Design Automation
FPGA	Field Programmable Gate Array
GNU	GNU's Not Unix
HDL	Hardware Description Language
IDE	Integrated Development Environment
I/O	Input/Output
IP	Intellectual Property
LSE	Lattice Synthesis Engine
PAR	Place and Route
PCS	Physical Coding Sublayer
PFF	Programmable Functional Unit without RAM/ROM
PFU	Programmable Functional Unit
PIO	Programmable Input/Output
PLC	Programmable Logic Controller
PLL	Phase Locked Loop
RTL	Register Transfer Level
SDF	Standard Delay Format
SDK	Software Development Kit
SoC	System-on-Chip
STA	Static Timing Analysis
TCL	Tool Command Language
VCD	Value Change Dump
XPE	Xilinx's Power Estimator

1. Introduction

The Lattice Semiconductor Field Programmable Gate Array (FPGA)/Complex Programmable Logic Device (CPLD) design flow is similar in conception and implementation to the Vivado®[1] FPGA design flow. At its core, a hardware description language (HDL) code of the register transfer level (RTL) can be imported into one of Lattice's design software and then configured to one of Lattice's FPGA.

This document guides Xilinx® FPGA designers familiar with the Xilinx Vivado[1] Prime software, specifically version 20.1, in migrating existing designs to the Lattice Diamond® software. It also highlights some of the differences and similarities between the design flows of Xilinx Vivado and Diamond.

This user guide starts with an overview and comparison of the design software and a mapping of the tools and file extensions between the two. The next section compares the design flows of the design software. The succeeding sections provides a step-by-step walkthrough about the following:

- Project creation and management
- Design entry
- Compilation flow
- Simulation
- Programming/Configuration

For more details on the Diamond design flow, refer to the Diamond video tutorials listed in Table 1.1.

Table 1.1. Diamond Video Tutorial Links

Title	Description
Diamond Overview	This video overview briefly covers several new features and abilities such as the new user interface, design flow, and several tool views that are available.
Diamond Installation Overview	Preview of step by step installation process of Lattice Diamond software.
Diamond Licensing Overview	Instructional video on obtaining license and installation.
Setting up a Floating License Tutorial	Instructional video on how to setup a floating license on a server and how to setup environment variables on a client machine in order to access the server license.
Diamond Key Concepts	This video discusses the structure of Diamond projects and the use of implementations, strategies, and folders within projects. Additionally, the video discusses shared design memory use, and context sensitive views.
Diamond Design Flow Changes	This video describes the design process flow and the use of the Process view, File List view, and Run Manager view.
Diamond Timing Analysis Overview	This video describes the management of the Timing Analyzer files, the new Timing Analyzer UI, and how to make timing constraint changes and generate new timing results.
Diamond Power Calculator	This video describes the management of the Power Calculator files and the behavior of the Power Calculator view.
Diamond Reveal™ Hardware Debugger	This video describes the management of the Reveal debug files and the new Reveal Analyzer waveform changes.
Diamond Simulation Flow	This video describes the simulation features provided with the software and the basic usage.
Diamond TCL Scripting Support	This video describes the available TCL dictionaries and how to run TCL commands from the user interface or the TCL console.
Diamond Programmer	This video describes how to use it from the user interface or outside of DIamond.

[1] *Xilinx, the Xilinx logo, and Xilinx Vivado are trademarks of Xilinx Corporation or its subsidiaries.*
Xilinx Vivado software is used to show the differences and similarities between the design flows of Xilinx Vivado and Lattice Diamond software.

2. Design Software Overview

A design software is required to be able develop and implement FPGA/CPLD designs for market usage. The design software must have the following characteristics to reduce the design's time-to-market:

- Full-featured – Offers all necessary tools for design development.
- High-performance – Performs powerful optimizations and analyses.
- Intuitive user interface – Provides the best user experience with a graphical user interface that is both modular and wizard driven.

The Diamond software has these characteristics, making it one of the best industry solutions for low-end to mid-range FPGA designs. The Diamond software integrated tool environment provides a modern, comprehensive user interface for controlling the Lattice Semiconductor FPGA implementation process with support for the older Lattice FPGA device families as compared to Lattice Radiant™ software, which supports relatively newer FPGA device families.

The Diamond software uses an expanded project-based design flow and integrated tool views so that design alternatives and what-if scenarios can easily be created and analyzed. The Implementations and Strategies concepts provide a convenient way for the user to try alternate design structures and manage multiple tool settings. System-level information—including process flow, hierarchy, and file lists—is available, along with integrated HDL code checking and consolidated reporting features. A fast Timing Analysis loop, ECO Editor, and Programmer provide capabilities in the integrated framework. The cross-probing feature and the shared memory architecture ensure fast performance and better memory utilization. The Diamond software is highly customizable and provides TCL scripting capabilities from either its built-in console or from an external shell.

Most of the capabilities available in the Xilinx Vivado software are available as well in the Diamond software. However, the terminology of the individual tools, features, and file formats may differ from one software to the other. Table 2.1 lists some of the tools, features, and file formats in Xilinx Vivado and its corresponding name in Diamond.

Table 2.1. Mapping of Xilinx Vivado and Diamond Tools, Features, and File Extensions

Description	Xilinx Vivado Tools	Diamond Tools
Optimization or implementation settings	Run strategies	Design strategies
Design entry	HDL file, IP Catalog, IP Integrator	HDL file, IPexpress or Clarity Designer, Schematic editor
Design constraints	Xilinx Vivado IDE Constraints Manager, Timing Constraints Wizard, Timing Constraints Window	Spreadsheet View, LDC Editor
Synthesis tools	Xilinx Vivado Design Suite Synthesis	LSE, Synplify Pro
Schematic tools	Schematic	Netlist Analyzer, Technology Viewer
Place and Route tools	Xilinx Vivado Implementation	Place and Route (PAR)
Supported simulators	Xilinx Vivado Simulator, Mentor Graphics Questa Advanced Simulator, Mentor Graphics ModelSim Simulator, Aldec Rivera-PRO™ Simulator, Aldec Active-HDL™, Synopsys® Verilog Compiler Simulator® (VCS), Cadence® Incisive® Enterprise Simulator (IES), Cadence Xcelium Parallel Simulator	Mentor Graphics Questa Advanced Simulator, Mentor Graphics ModelSim Simulator, Aldec Rivera-PRO Simulator, Aldec Active-HDL, Synopsys Verilog Compiler Simulator (VCS), Cadence NC-Verilog, Cadence NC-VHDL, Cadence NCSim

Description	Xilinx Vivado Tools	Diamond Tools
Simulation levels	Behavioral Simulation, Post-Synthesis Simulation (Functional and Timing), Post-Implementation Simulation (Functional and Timing)	RTL Simulation, Post-Route Gate-Level Simulation, Post-Route Gate-Level+Timing Simulation
Power Analysis	Xilinx Power Estimation, Report Power	Power Calculator
On-chip debug tool	Xilinx Vivado logic analyzer (Hardware Manager)	Reveal
Hardware programming tool	Xilinx Vivado Programmer (Hardware Manager)	Programmer
Project file extension	.xpr	.ldf
IP configuration file extension	.xci	.sbx
Soft processor development and generator	Vitis™ Core Development Kit, Xilinx Software Development Kit (XSDK)	Lattice Propel™ Builder
Soft processor	Zynq, MicroBlaze™	RISC-V

3. FPGA Design Flow Overview

When creating designs for Field Programmable Gate Arrays (FPGAs), Lattice and Xilinx software tools have similarities in terms of concepts, approach, and functionality. Lattice Diamond software framework technology uses the typical FPGA design flow (see Figure 3.1) that adheres to a sequence of steps, which initially requires setting up the design environment and ends with the generation of programming files that are used to program the hardware. In cases where Xilinx designers who are familiar with Xilinx Vivado Prime would like to convert the existing Xilinx Vivado designs to Lattice Diamond software environment, the designers can simply import the HDL (hardware description language) files to Lattice Diamond and begin implementing the project-based methodology (see Figure 3.2).

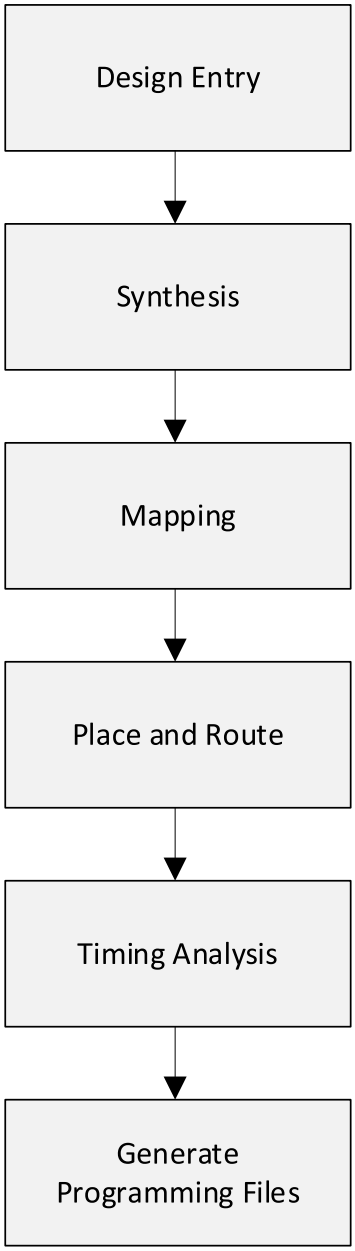


Figure 3.1. Typical FPGA Design Flow Used by Xilinx Vivado and Lattice Diamond

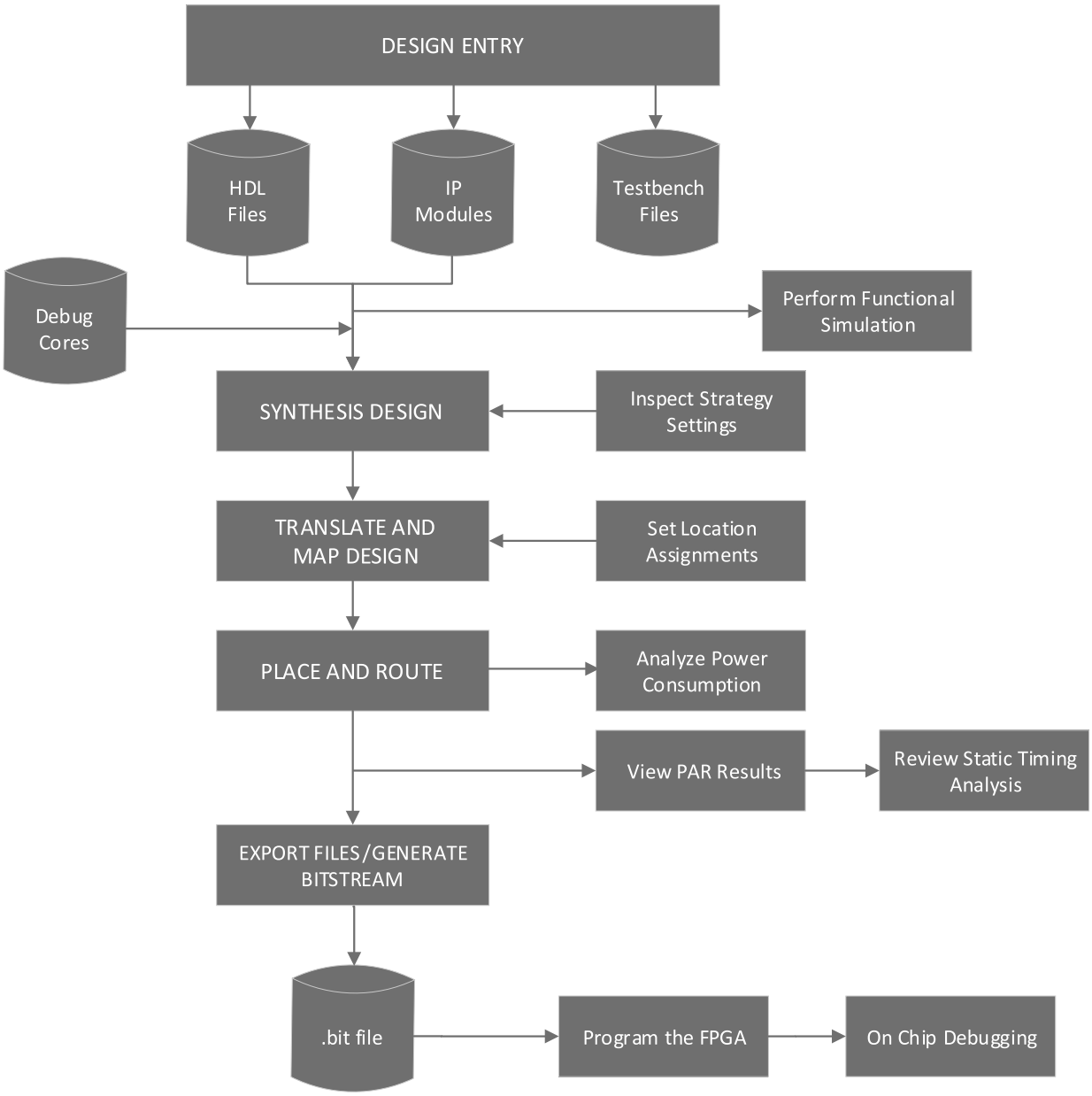


Figure 3.2. Diamond Software Detailed Design Flow

4. FPGA Design Flow Tools and Views

This section shows how Lattice Diamond and Xilinx Vivado software handle the FPGA design flow steps with various tools and views.

4.1. Project Creation and Management

In setting up the initial design environment, both Xilinx Vivado and Lattice Diamond provide a new project wizard functionality that guides the user through the steps of project creation. The Diamond New Project option is shown in Figure 4.1.

The Diamond New Project wizard allows the user to specify the project name and location, as shown in Figure 4.2, add existing source files as shown in Figure 4.3, select the target device for the design, as shown in Figure 4.4, and choose the synthesis tool, as shown in Figure 4.5.

Note: All project settings may be changed at later stages in the design process.

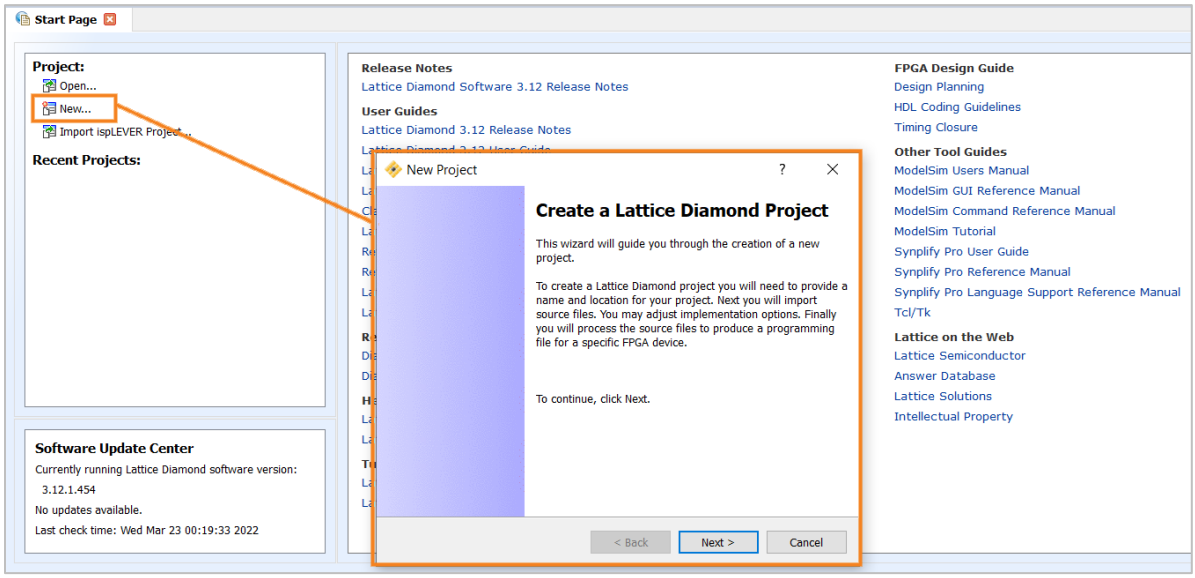


Figure 4.1. New Project Wizard

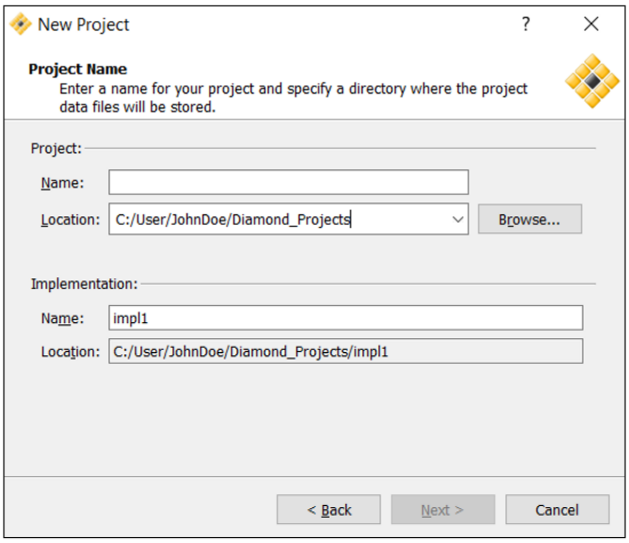


Figure 4.2. Project Settings

In the Project Name page, the following items are available.

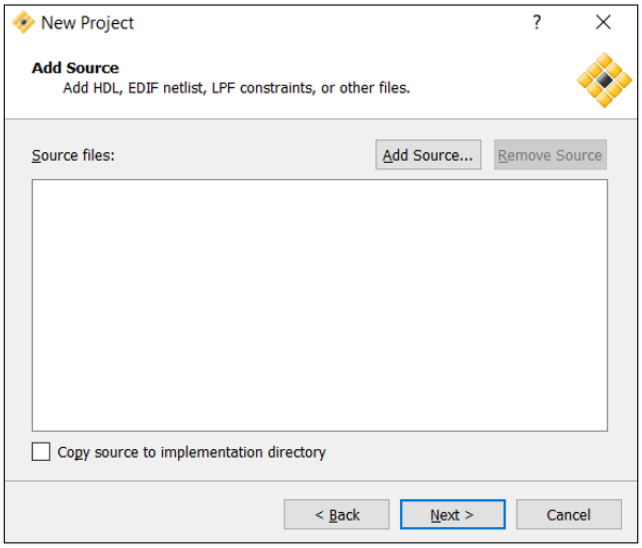
Under the Project group:

- Name – Name of project
- Location – File path where the project is saved

The following items are available under the Implementation group:

- Name – Implementation name
- Location – File path where the implementation is saved

Implementations define the design structural elements for a project. An implementation contains the structure of a design and can be thought of as the source and constraints to create 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 at least one implementation.



Diamond Add Source File Types	
File Description	**File Types**
Input Files	*.vhd *.v *.sv *.ed *.sch *.sch *.lpc *.ipx *.sbx
Platform Design Files	*.ptm
Synthesis Constraint Files	*.sdc *.fdc *.ldc
LPF Constraint Files	*.lpf
Debug Files	*.rvl *.rva
Script Files	*.spf
Analysis Files	*.pcf *.tpf
Programming Files	*.xcf
Other Files	*.*

Figure 4.3. Adding Existing Source

In the Add Source page, the following items are available.

- Add Source – Adds HDL files, constraint files or testbench in the current project.
- Copy source to implementation directory – Copies the source files added in the current project directory.

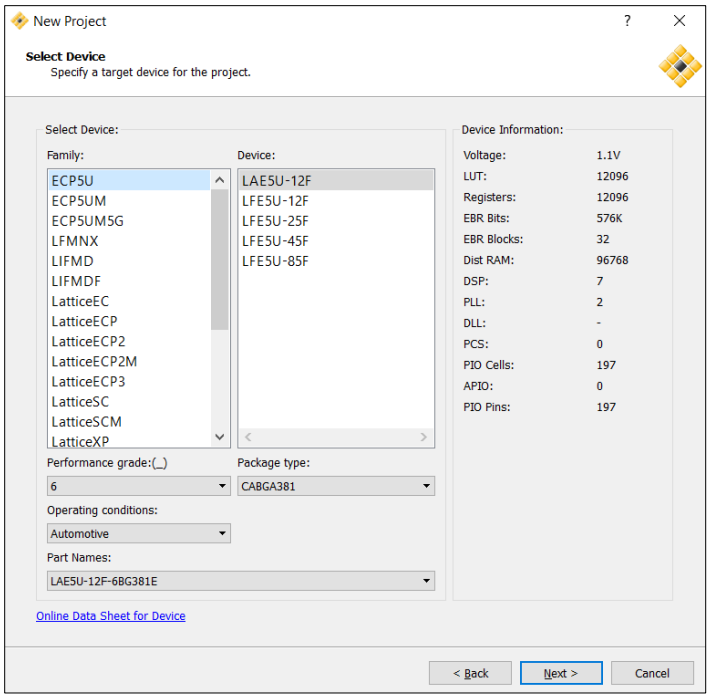


Figure 4.4. Target Device Selection

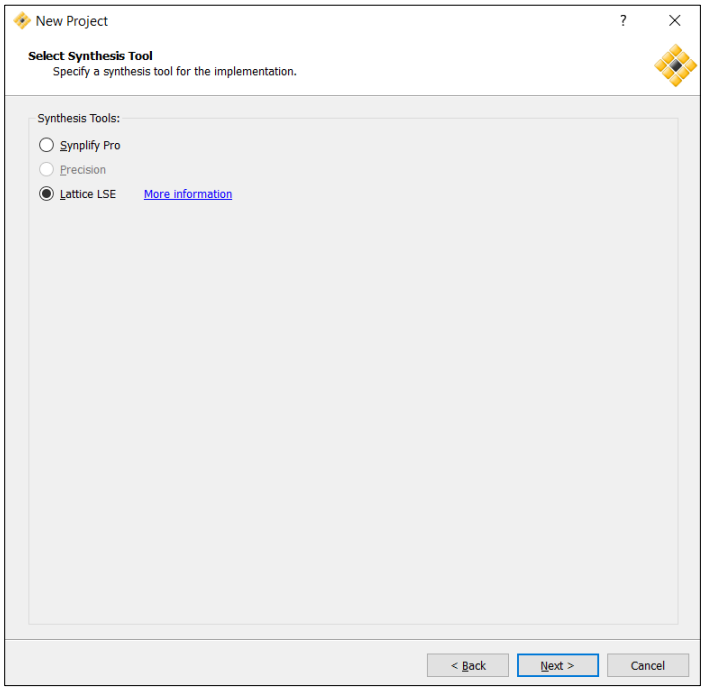


Figure 4.5. Synthesis Tool Selection

Once the project is created, choosing the logical and consistent way of organizing each source file allows the user to locate and modify as needed. Both Xilinx Vivado and Diamond support this practice by combining design sources into different categories and listing them in a Files/Files List View. In the Diamond software, the sources are categorized by different types and are identified with different icons. The Bold items indicates that the file is the top-module, the active constraint file, reveal project, programming file and are used when processing the design project. Grayed out items are those inactive files that are not used for any part of the design process, as shown in Figure 4.7.

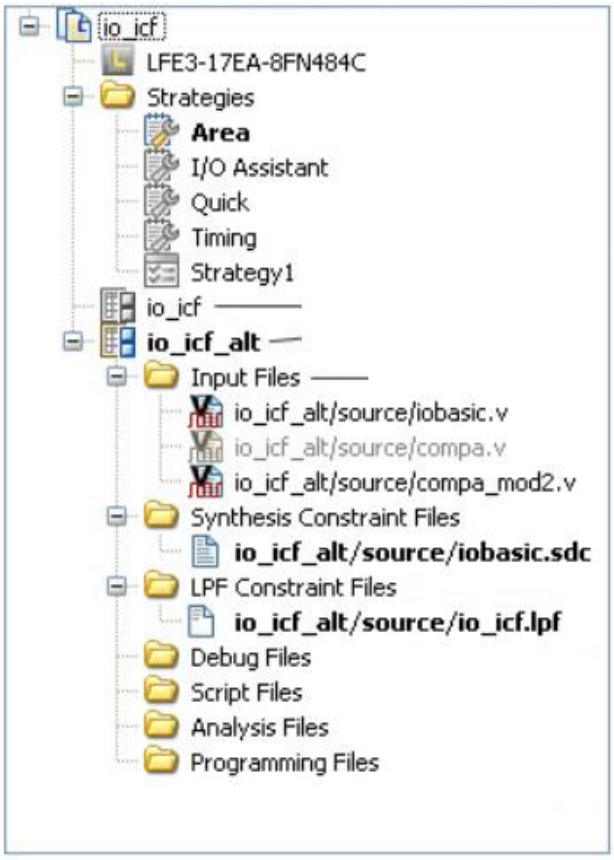


Figure 4.6. Lattice Diamond File List View

Refer to Table 4.1 for the file types and extensions.

Table 4.1. Files and Fields Listed in Diamond File List View

File	Description
Project Name	—
Device Part Number	—
Strategies	First four (wrench icon) are prebuilt and cannot be changed. Customize Strategy1 and others the user adds.
Implementations	Different versions of the design.
Input Files	Design files such as Verilog and VHDL. Includes files for the design and the testbench.
Synthesis Constraint Files	SDC files used by synthesis tools.
LPF Constraint Files	Lattice preference files used by map and place-and-route stages.
Debug Files	Reveal files used in on-chip debugging.
Script Files	Simulation Wizard files.
Analysis Files	Power Calculator and Timing Preference files.
Programming File	Files for programming a device, such as bitstream or JEDEC.

User can also see implementations listed in the File List view, as shown in Figure 4.7.

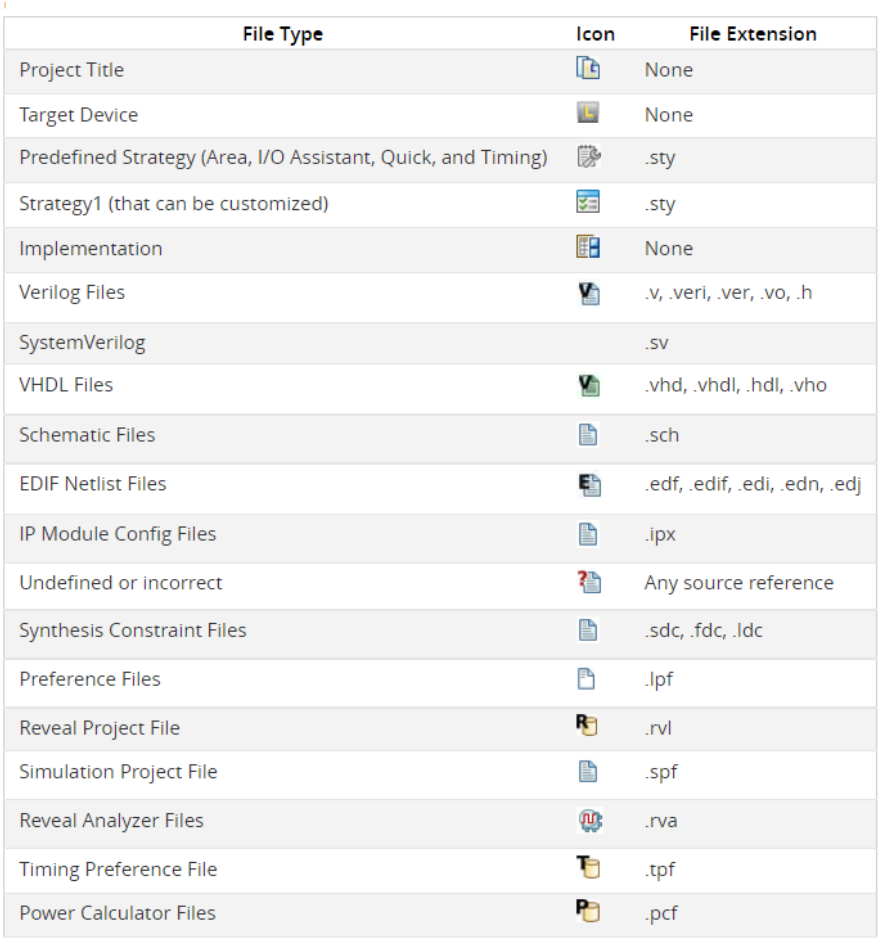


Figure 4.7. Files Listed in Diamond File List View

Diamond also offers different ways of involving a source file in the design. If the user right-click on each source file, the user is given different options depending on the file type.

These options are:

- Excluding the file from implementation
- Setting the design file as top-level
- Removing the file for either or both synthesis and/or simulation
- Checking for properties

For implementation and strategy, user can create a different version or a clone of each, edit the properties, select the synthesis tool, and select the top-level unit.

4.2. Design Strategy

In Xilinx Vivado, users can optimize the design or resolve synthesis and place-and-route challenges using a set of pre-configured command line options. This is called a run strategy. Xilinx Vivado IDE provides several commonly used strategies that are tested against internal benchmarks.

In Diamond, this is equivalent to the design strategy. Diamond allows user to choose or create its own custom strategy for the implementation. A strategy is a collection of implementation-related tool settings or recipes for how the design is implemented.

The Diamond software provides four pre-defined strategies:

- Area – for area optimization; its purpose is to minimize the total logic gates used while enabling the tight packing option available in Map.
- I/O Assistant – for I/O design; its purpose is to help designers select a legal device pinout and produce LOCATE and IOBUF preferences for optimal I/O placement.
- Quick – for initial quick run; its purpose is to allow the tool to get results with the minimum time using very low effort in placement and routing.
- Timing – for timing optimization; its purpose is to achieve timing closure.

It also enables user to create customized strategies, which can be edited, cloned, and removed. All strategies are available to all of the implementations, and any strategy can be set as the active one for an implementation.

User can view the strategies on the File List View. Double-click on the strategy to tool settings for each of the steps of the design flow.

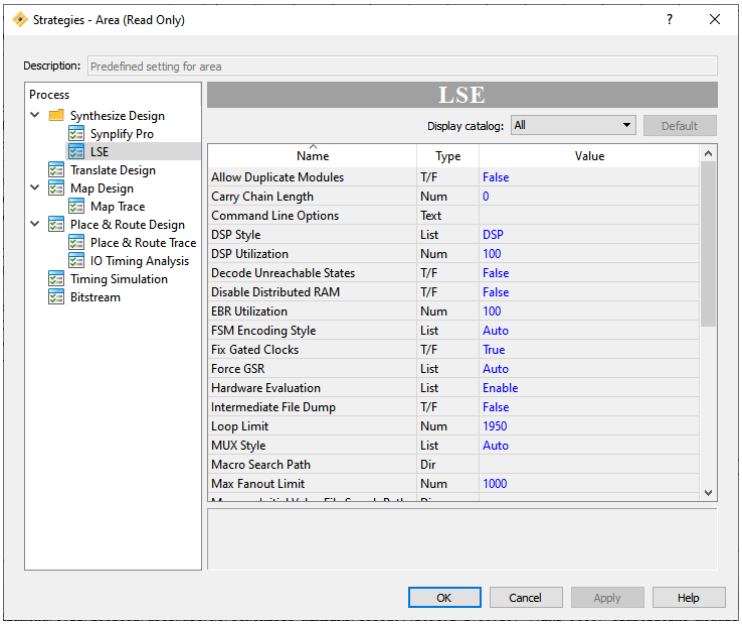


Figure 4.8. Area Pre-defined Strategy Settings

For detailed information, refer to the Diamond Help or to *Strategies (Chapter 5)* in Lattice Diamond 3.12 User Guide.

4.3. Design Entry

Lattice Diamond software supports HDL source files, IP catalog, and schematic entry as design entry methods.

4.3.1. HDL Source Files

In Xilinx Vivado, user creates a new HDL source file by clicking on Add Source under Project Manager. An Add Source window opens from which the user can choose to Add or create design sources. In the Lattice Diamond software, user can do this similarly by clicking File > New or by right-clicking the Input Files folder on the File List tab. From the dropdown menu click on Add > New File. A New File dialog box opens where user can choose the HDL file type to add to the design.

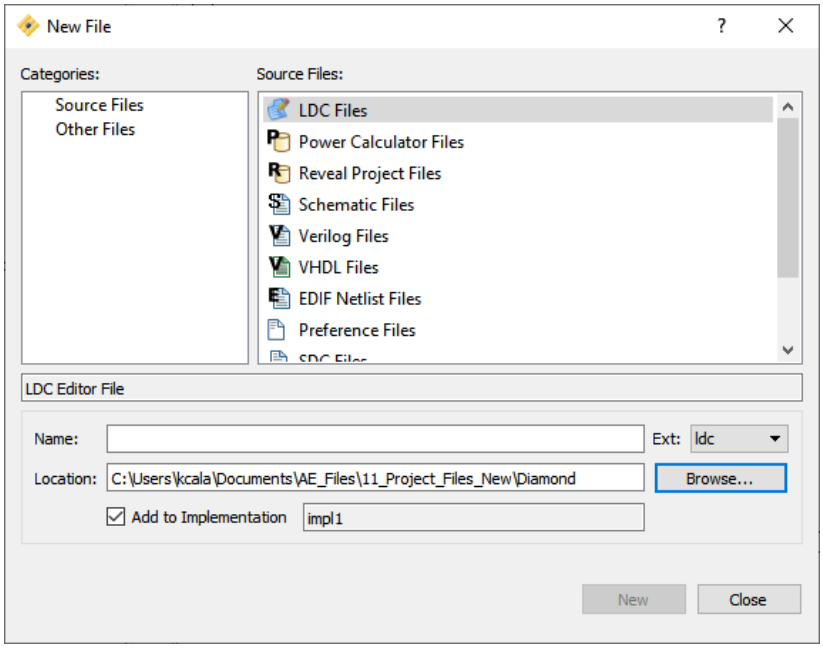


Figure 4.9. New File Dialog Box

4.3.2. IPexpress and Clarity Designer

In Xilinx Vivado, user can open the IP catalog by clicking on the IP Catalog under Project Manager. The IP catalog pane opens from where user can select which IP cores to customize and add to the design. In the Diamond software, the IPs can be accessed either through the IPexpress or the Clarity Designer.

Note: Clarity Designer is only available on the ECP5™ device and CrossLink™ device families.

User can click the IPexpress icon (or the Clarity Designer icon) on the Toolbar to open the corresponding window.

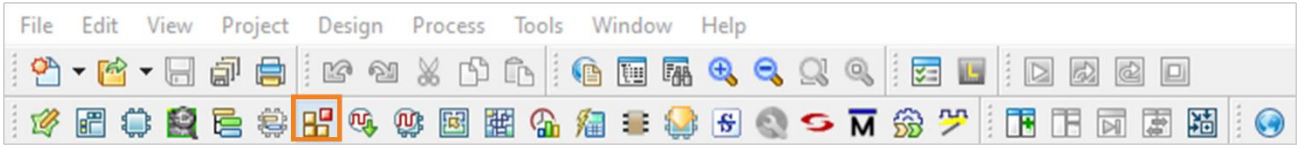


Figure 4.10. IPexpress (or Clarity Designer) Icon

On the IPexpress tab, user can select between installed IPs on local folder or can choose to download and install additional IPs from the IP Server.

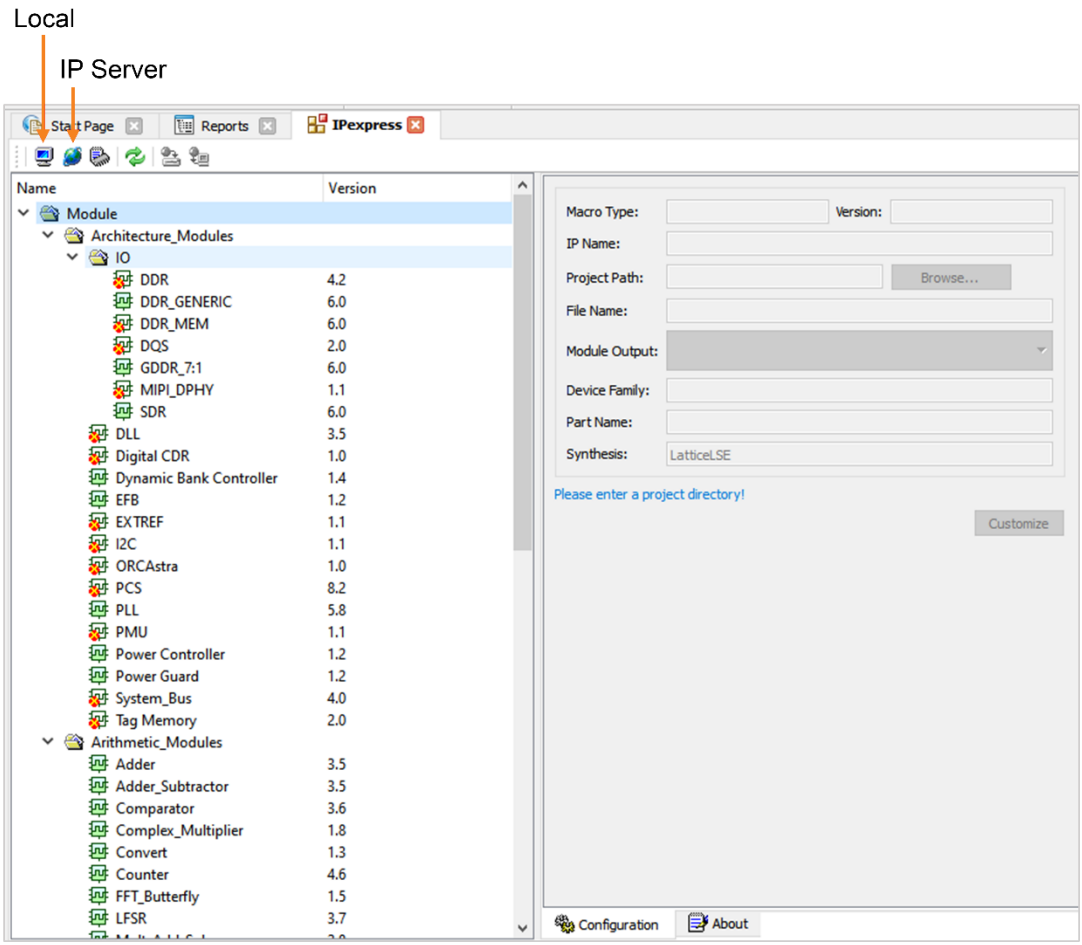


Figure 4.11. IPexpress Window

Similar to IPexpress, Clarity Designer provides a way to use a collection of functional blocks from Lattice Semiconductor. However, Clarity Designer does not only customize intellectual property (IP) but actually creates a package of modules. This includes wiring the modules together and creating placement constraints for DDR and PCS pins. This package can contain a single module or the entire design.

The resulting Clarity Designer module can aid in developing the board design and firmware while developing the FPGA design, cutting the total development time.

Clarity Designer features three major tabs:

- Catalog provides a list of the available modules and IP. Use Catalog to select the modules the user wants in the project and to customize them. User can also download more IP from the Lattice website.
- Builder helps user make connections between the customized modules.
- Planner provides a graphic way to place Double-Data Rate (DDR) pins and Physical Coding Sublayer (PCS) components before synthesis. User can run design rule checks on the placement while still in the earliest stages of designing.

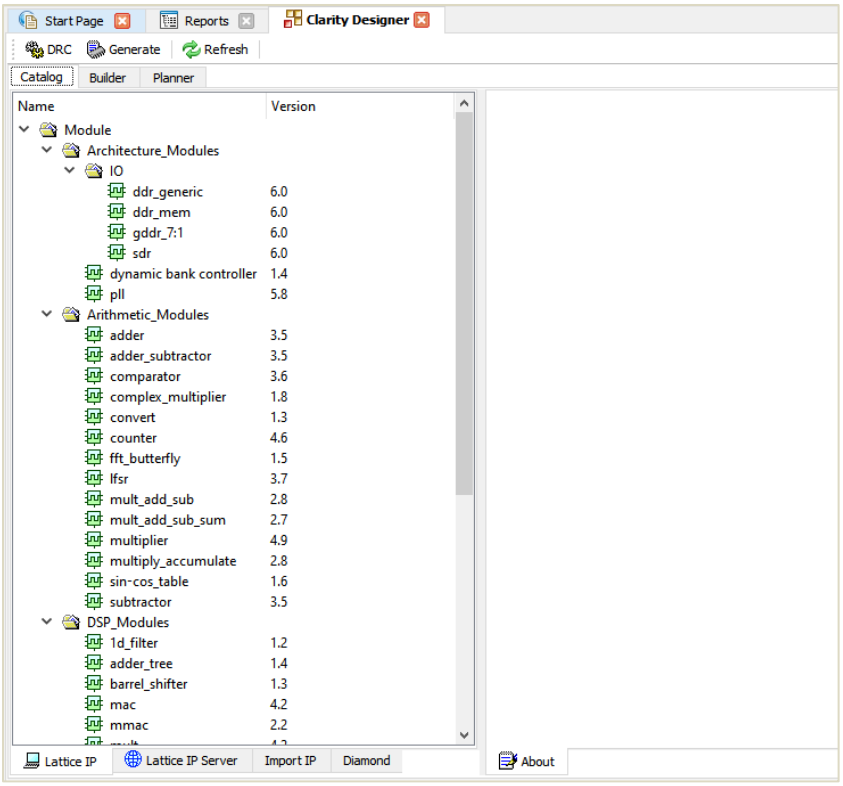


Figure 4.12. Clarity Designer Window

For detailed information, refer to the Diamond Software Help and the following sections in the Lattice Diamond 3.12 User Guide:

- IPexpress (Chapter 7)
- Clarity Designer (Chapter 7)

4.3.3. Schematic Editor

In Xilinx Vivado, there is no option of adding a schematic type source file on the project.

In the Diamond software, user can also do this using the Schematic Editor, which works in conjunction with Symbol Editor and Symbol Library Manager. User can design by laying out symbols for basic functions and other modules and drawing wires between the ports. User can also create its own library of custom schematic symbols. To begin a new schematic design, choose File > New > File.

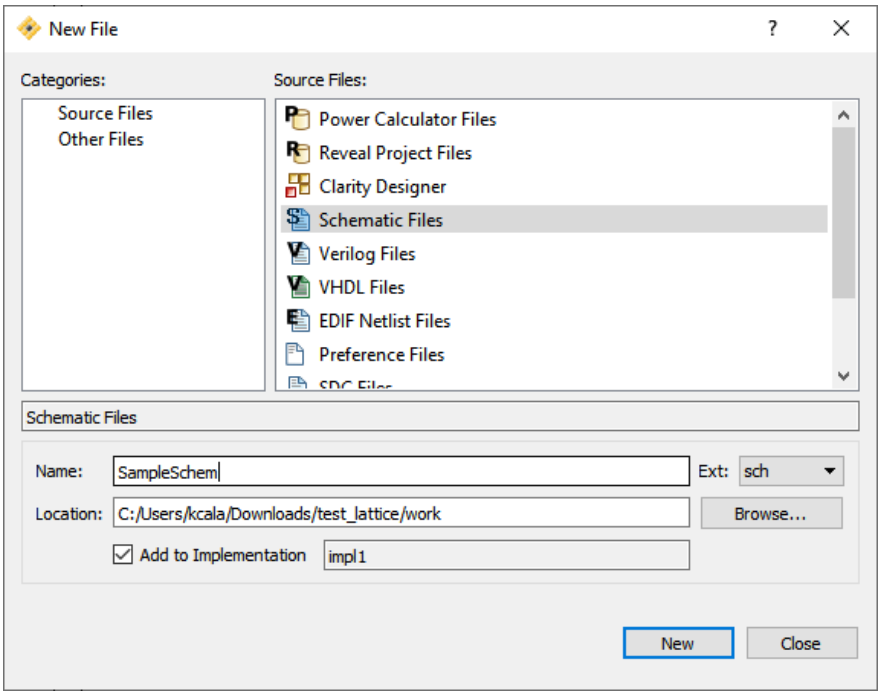


Figure 4.13. Creating a New Schematic File

When the Schematic Editor window opens, right-click on the window to view options on what user can add to the design, such as wires, buses, symbols or ports.

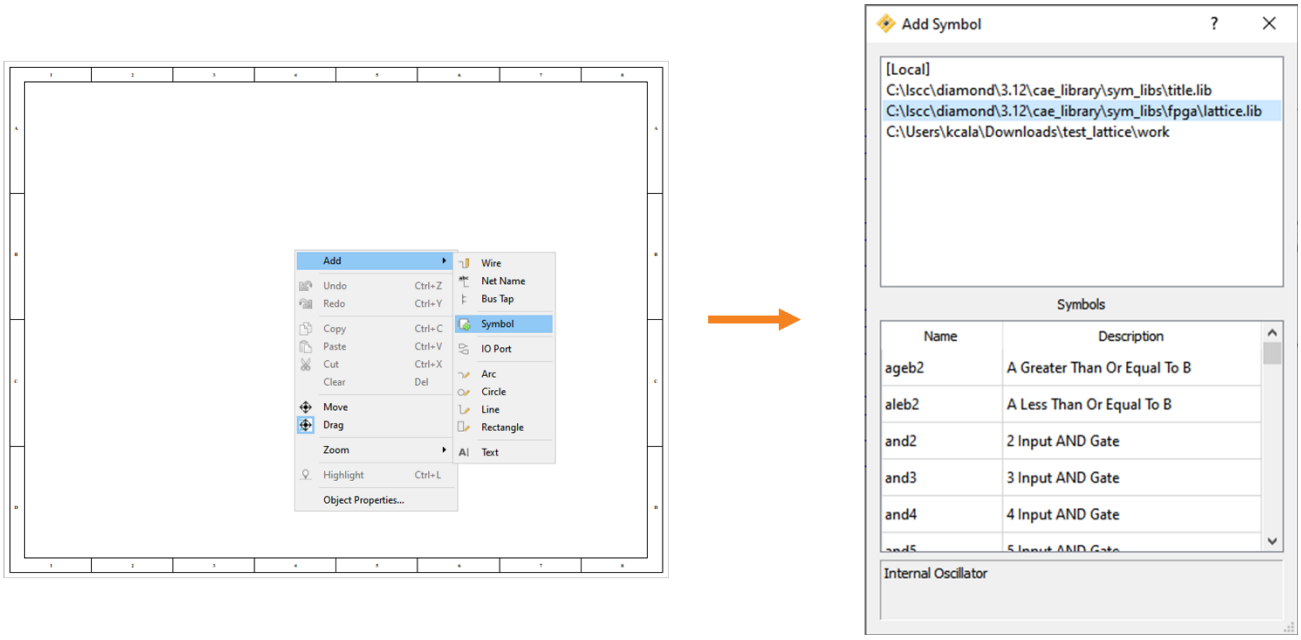


Figure 4.14. Adding Symbols to the Schematic Design

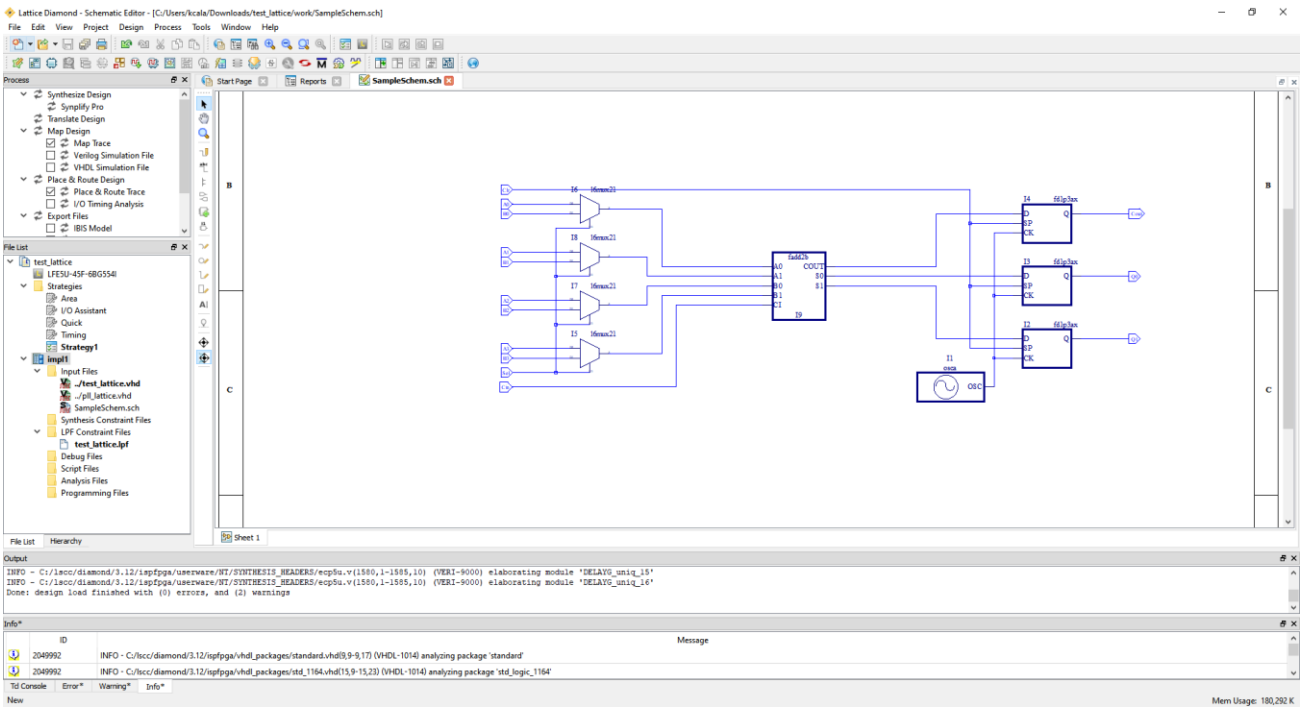


Figure 4.15. Example Schematic Design.

Upon entering the design, elaboration can be performed and the Hierarchy can show the design hierarchy as a nested list of modules. The Hierarchy view is open by default as a tab beside the File List. If the Hierarchy view is not open, choose *View > Show Views > Hierarchy* from the Diamond main window.

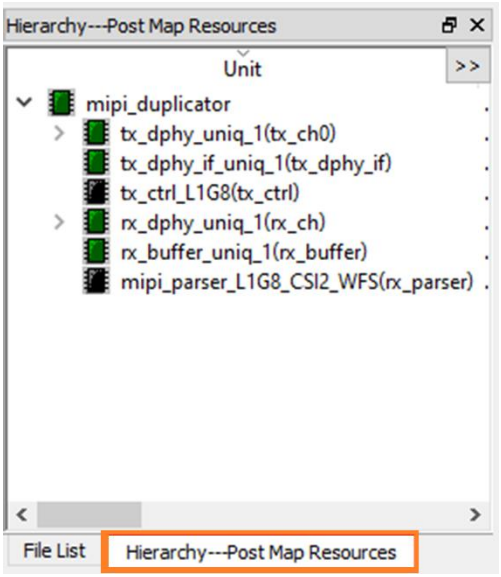


Figure 4.16. Hierarchy View

For detailed information, see the *Entering the Design* section of the Lattice Diamond Help.

4.4. Design Constraints

Design constraints are used to specify the performance requirements desired for the FPGA design. Various tools help the designers to meet those conditions. Constraints are instructions applied to the design elements that guide the design toward desired results and performance goals. They are critical to achieving timing closure or managing reusable intellectual property (IP). The most common constraints are those for timing and pin assignments, but constraints are also available for placement, routing, and many other functions.

Table 4.2. Design Constraint Tools Comparison

Xilinx Vivado	Lattice Diamond
Device, Physical and Timing Constraints window	Lattice Synthesis Engine (LSE) LDC Editor, Synplify Pro SCOPE Constraints Editor

Similar to how Xilinx Vivado applies different constraints, Lattice Diamond is able to apply constraints from multiple sources in multiple ways, such as timing constraints from SDC or LDC synthesis files; constraints defined in HDL source files; or with post-synthesis constraints known as logical preferences that are defined in the logical preference file (.lpf).

- SDC is Synopsys Design Constraint for timing (originally for ASIC)
- FDC is Synopsys FPGA Design Constraint added on top of SDC, targeted for FPGA
- LDC is Lattice Design Constraint that supports some popular SDC timing constraints command and Lattice-specific physical (device) constraints
- LPF is Logical Preference File for storing constraints. It is the constraint file used as input for developing and implementing the design. The .lpf file can also be used to override attributes that exist in the HDL source. When the user modifies these constraints in the .lpf file, it takes precedence over those in the HDL file.

The Diamond software provides tools with user interface for assigning constraints.

4.4.1. Lattice Synthesis Engine (LSE) LDC Editor

The Diamond Lattice Synthesis Engine (LSE) enables user to set Synopsys Design Constraints (SDC), which are directly interpreted by the synthesis engine. When user uses LSE, the SDC constraints are saved to a Lattice Design Constraints file (.ldc). Lattice Design Constraints (LDC) Editor, as well as Source Editor, are available for creating and editing .ldc files. LDC Editor provides a spreadsheet style user interface that enables user to quickly create and edit Synopsys Design Constraints.

4.4.2. Synplify Pro SCOPE Constraints Editor

The SCOPE™ (Synthesis Constraints Optimization Environment) presents a spreadsheet-like editor with a number of panels for entering and managing timing constraints and synthesis attributes. The SCOPE user interface also includes an advanced text editor that can help the user edit constraints easily. These constraints are saved to the FPGA Design Constraint (FDC) file.

Constraints can include timing constraints defined in the synthesis constraint files (SDC) or HDL attributes. SDC is used for the synthesis tool. Post-synthesis constraints known as preferences can also be specified. The flow combines these together. All three sources of constraints specify design goals. Synthesis, map, and place-and-route work to meet these goals. Timing analysis reports whether or not the goals are met.

Note: Not all existing SDC constraints are supported by Lattice Diamond.

Synthesis constraint creation flow depends on synthesis tool selection.

- .ldc (through LSE)
- .sdc (through Synplify Pro)
- .fdc (through Synplify Pro)

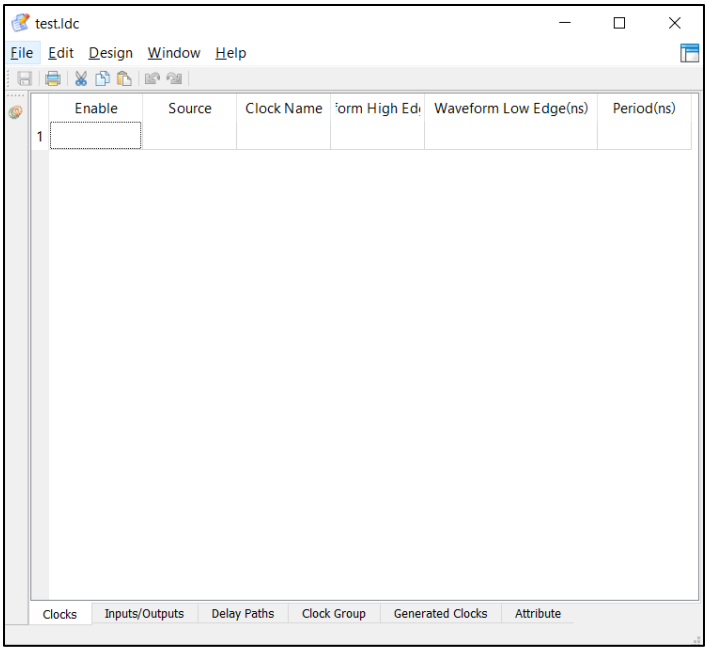


Figure 4.17. LDC Constraint Editor Window

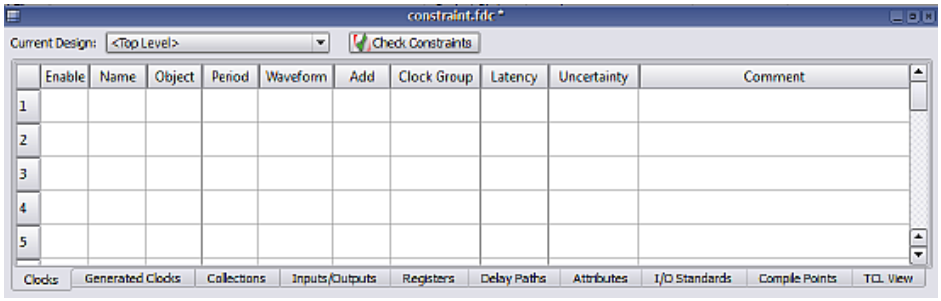


Figure 4.18. SCOPE Constraint Editor Window

For Post-synthesis, constraints are stored to the .lpf file, which is used for input for developing and implementing the design. The Map Design process depends on the .lpf file. If the user maps the design and then modify the active .lpf file, the map process and any downstream processes must be rerun.

User can create and modify logical preferences in the .lpf file using the Diamond preference-editing views, or user can modify the .lpf file directly using a text editor.

Xilinx Vivado Assignment Editor provides a spreadsheet-like interface for assigning all instance-specific settings and constraints while Lattice Diamond provides similar tools to view and define new constraints, called preferences, or to modify existing constraints from the source files and save them as preferences:

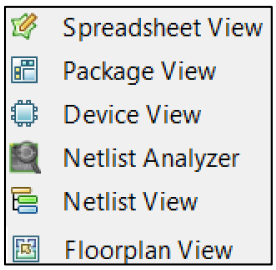


Figure 4.19. Lattice Diamond Tools

- Spreadsheet View – Modify timing constraints that are defined in the synthesis tool and save them as logical preferences to the active .lpf file. Set timing objectives such as f_{MAX} 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 and perform SSO analysis. Define groups of ports, cells, or ASIC blocks. Create UGROUPs from selected instances to guide placement and routing. Establish REGIONs for UGROUPs or to reserve areas of the floorplan. Run PIO design rule checking.
- Device View – Examine FPGA device resources. Reserve sites that should be excluded from placement and routing.
- Netlist View – View the design tree by ports, instances, and nets. Assign pins for selected signals. Set timing constraints. Define groups from selected ports or registers. Create UGROUPs from selected instances to guide placement and routing.
- 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 the status of SSO pins. Run PIO design rule checking.
- Floorplan View – View the device layout. Draw bounding boxes for UGROUPs. Draw REGIONs for the assignment of groups or to reserve areas. Reserve sites and REGIONs that should be excluded from placement and routing. Run PIO design rule checking.

For detailed information, refer to the following:

- Diamond Software Help
- The following sections in the Lattice Diamond 3.12 User Guide:
  - Synthesis Constraint Files (Chapters 5 and 6)
  - LPF Constraint Files (Chapters 5 and 6)
  - Tools: Spreadsheet, Device, Netlist, Package, and Floorplan View (Chapter 7)
- Lattice Synthesis Engine User Guide

4.5. Synthesis

Xilinx Vivado has the Xilinx Vivado Design Suite Synthesis to perform synthesis on designs with user logic and Xilinx IP. Lattice Diamond has a proprietary synthesis tool in Lattice Synthesis Engine (LSE) and supports a third party synthesis tool in Synplify Pro. Both synthesis tools support synthesis of user logic and Lattice IP.

Table 4.3. Synthesis Tools Comparison

Xilinx Vivado	Lattice Diamond
Xilinx Vivado Design Suite Synthesis	LSE and Synplify Pro

User can select between available synthesis tools in Xilinx Vivado by going to the *Analysis & Synthesis > Edit Settings > Electronic Design Automation (EDA) Tool Settings*. In Diamond, user can do this by right-clicking the current implementation and then choosing Select Synthesis Tool from the dropdown menu.

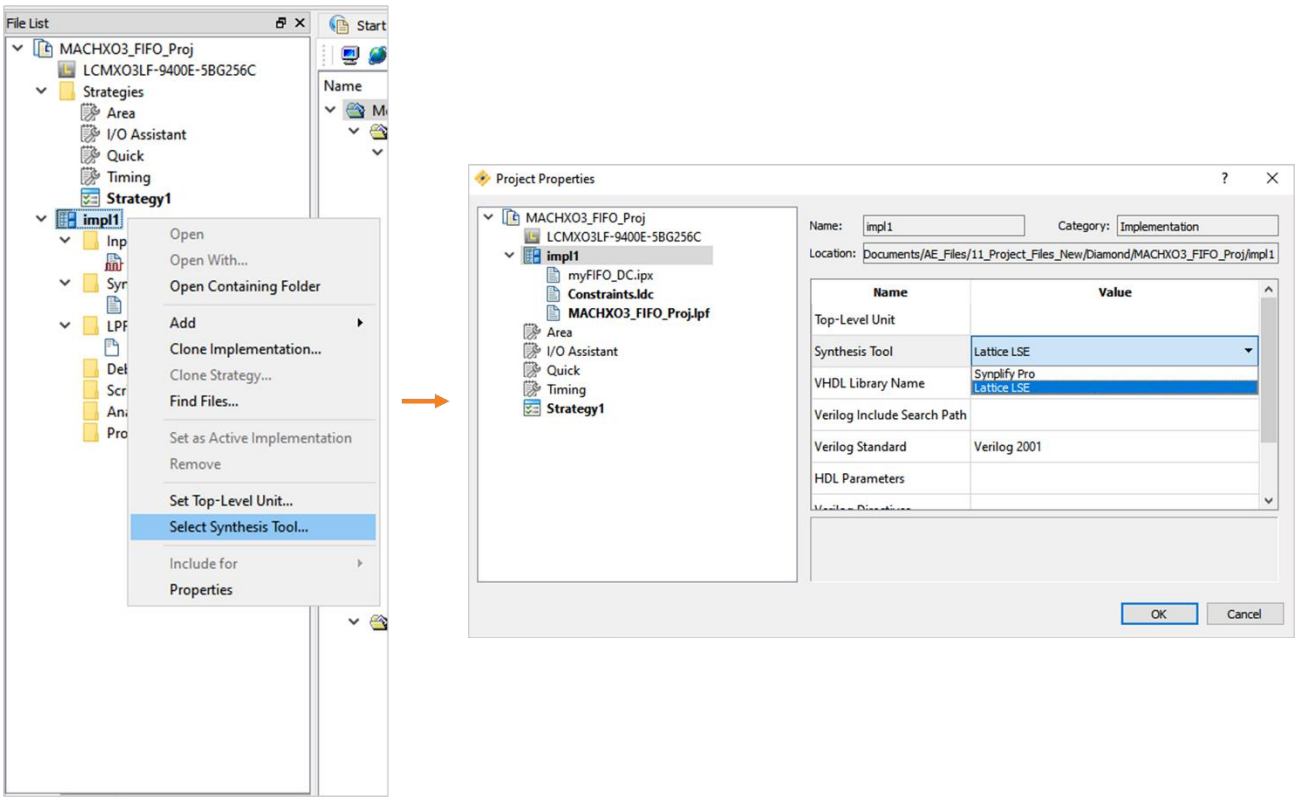


Figure 4.20. Selecting between Available Synthesis Tools

In Xilinx Vivado, synthesis is performed by clicking on Run Synthesis under Synthesis of the Flow Navigator pane. In Diamond, synthesis can be performed by double-clicking on the Synthesize Design on the Process tab or by right-clicking Synthesize Design and then clicking Run on the dropdown menu. It is important to note that when Synplify Pro is selected, synthesis is divided into two tasks, Synthesize Design and Translate; compared to LSE which only have the Synthesize Design task.

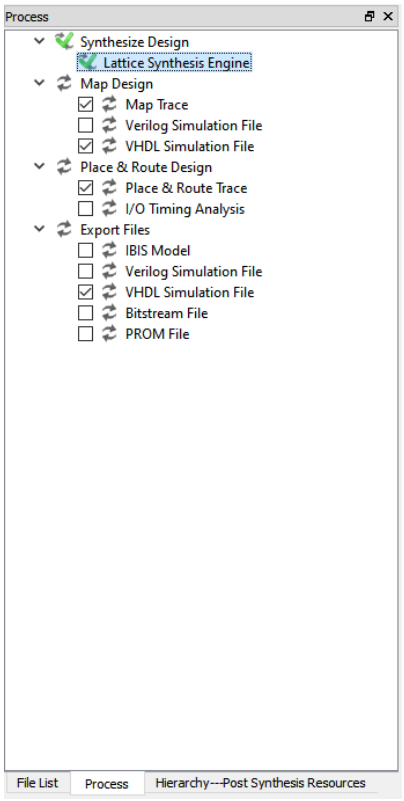


Figure 4.21. Running Synthesis on the Process Tab

Depending on which tasks are checked, user can use the Process toolbar to run a task, re-run a task, re-run all tasks, or stop a task.

Figure 4.22. Process Toolbar

Aside from the Diamond tool, user can open the standalone Synplify tool from which the user can also perform synthesis. To modify the optimization and other settings related to synthesis, refer to the Design Strategy section.

4.6. Netlist Viewer

A generated netlist can be viewed through one or more schematic views and a browser that shows the lists of modules, instances, ports, and nets. In Xilinx Vivado, this can be performed going to Synthesis > Open Synthesized Design > Schematic on the Flow Navigator pane. While in Lattice Diamond, this can be performed using Netlist Analyzer of LSE or HDL Analyst of Synplify Pro.

Table 4.4. Tools Comparison

Xilinx Vivado	Lattice Diamond
RTL Viewer and Technology Map Viewer	Netlist Analyzer and HDL Analyst

To open Netlist Analyzer, click on the Netlist Analyzer icon on the toolbar.

Figure 4.23. Opening Netlist Analyzer

To open HDL Analyst, open Synplify Pro from the Lattice Diamond and then click on the Technology View icon on the toolbar.

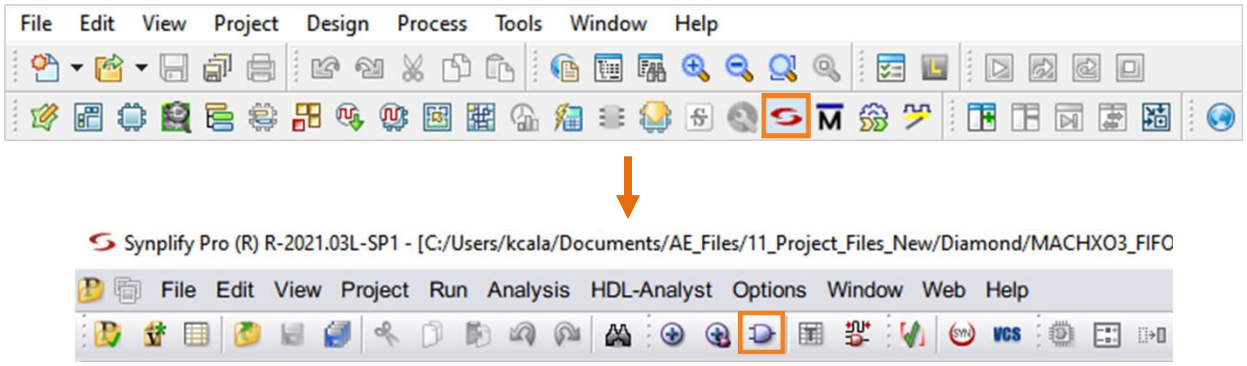


Figure 4.24. Opening Technology View

For detailed information, refer to the Diamond Software Help and to the Netlist Analyzer (Chapter 7) in the Lattice Diamond 3.12 User Guide.

4.7. Design Mapping

Design mapping converts the logical design into a network of physical components or configurable logic blocks. In Xilinx Vivado, this process is combined with the Implementation process. In Lattice Diamond, this is a separate process in the design flow that can be optimized through the design strategy settings.

To map a design, double-click Map Design or right-click Map Design and select Run from the dropdown menu.

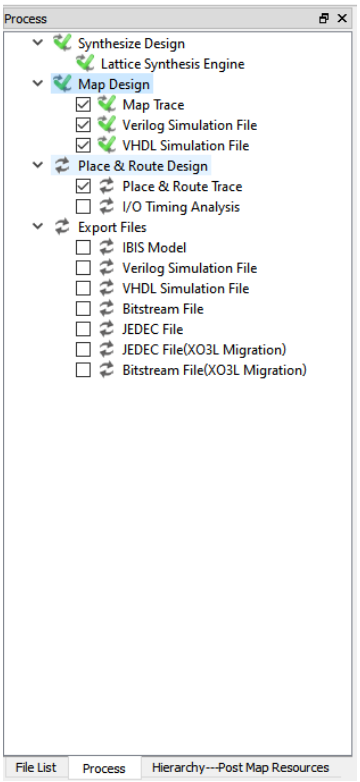


Figure 4.25. Running Map on the Process Tab

4.8. Place and Route

After a design has undergone the necessary translation to bring it into the physical design format during mapping, it is ready for placement and routing. Placement is the process of assigning the device-specific components produced by the mapping process to specific locations on the device floorplan. After placement is complete, the route phase establishes physical connections to join components in an electrical network. The place and route process takes a mapped physical design and places and routes the design. Placement and routing of a design can be cost-based or timing driven. In Xilinx Vivado software, the Xilinx Vivado II Fitter, which is also known as the PowerFit Fitter, performs place and route, also referred to as *fitting* in the Xilinx Vivado II software while in the Diamond software environment, the Place & Route Design process automatically assigns device-specific components to locations and connects them.

Table 4.5. Place and Route Tools Comparison

Xilinx Vivado Prime Pro Edition	Lattice Diamond
Xilinx Vivado Fitter (Plan, Early Place, Place, Route, Retime and Finalize)	Place and Route

Placement and routing options can influence the performance and utilization of the design implementation and ease incremental design changes. Some options affect the way the results are reported. Experimenting with place and route settings in the Strategies dialog box can help improve the placement and routing results.

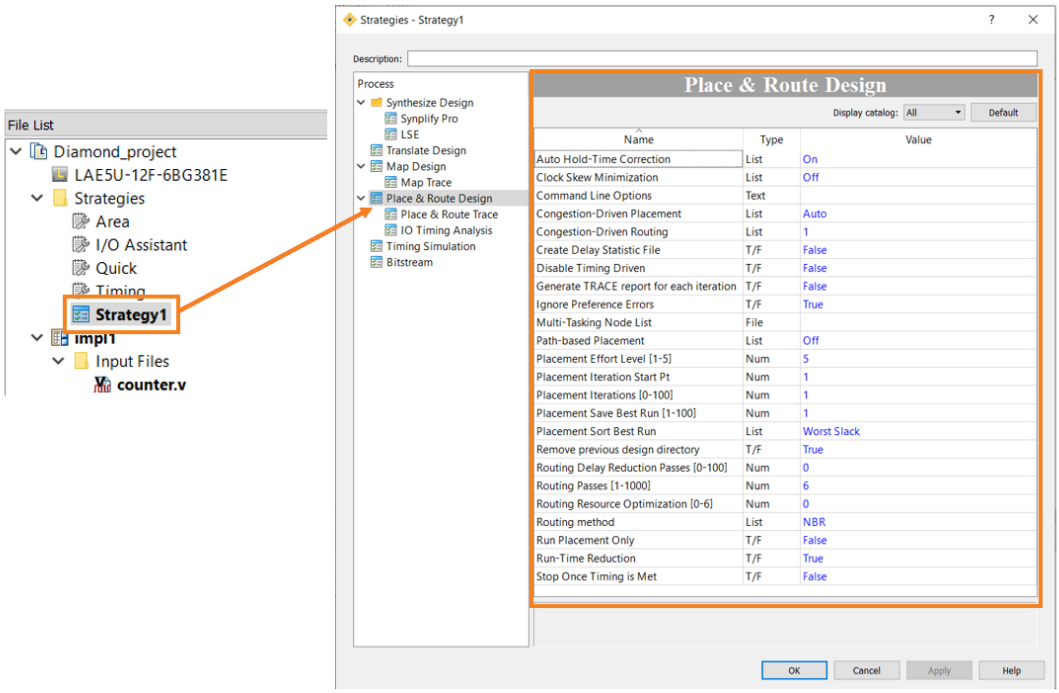


Figure 4.26. PAR Design Strategy Options

In most cases, the user design requires timing-driven placement and routing, where the timing criteria specified influences the implementation of the design. Static timing analysis results show how constrained nets meet or do not meet the timing preferences.

In the Diamond Process view under Place & Route Design, a Place & Route Trace process is available that runs static timing analysis on the place and routed .ncd file. This process reports any timing errors associated with preferences and generates a report. The Place & Route Trace and I/O Timing Analysis report can be accessed from the Analysis Reports folder of the Reports window. These reports help ensure that the I/O plan meets the I/O standards and power integrity requirements of the PCB design.

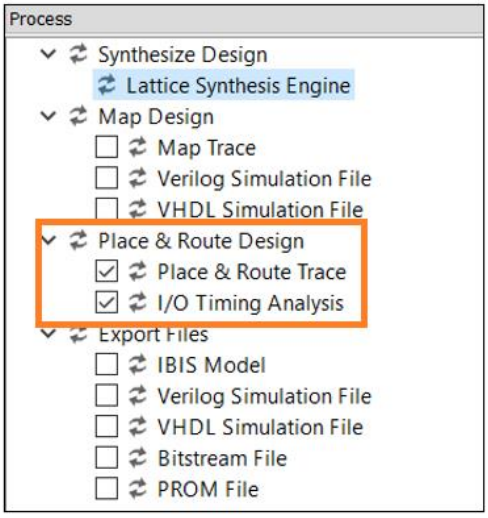


Figure 4.27. PAR Timing Analysis Option

4.9. Cross-Probing

Cross-probing is a feature where it allows the software for seamless bi-directional communication between the different tools within the software itself. Users can select a design element from one tool and locate them in another tool. Both Xilinx Vivado and Lattice Diamond software has cross-probing ability.

The Diamond preference-editing views allow users to select an element in one view and quickly display the corresponding logical or physical element in a different view. For example, user can cross-probe a component in Floorplan View to view the logical elements in Netlist View; or user can cross-probe a signal in Spreadsheet View to examine its placement on the pin layout of Package View.

To cross-probe an element from one view to another:

1. Right-click an element in any view.
2. Choose **Show in** and choose the desired view from the pop-up menu.

Table 4.6 shows the cross-probing availability for each Preference view.

Table 4.6. Cross-Probing

Cross Probe From	Element	Cross Probe To
Spreadsheet View (Port Assignments)	Pin	Package View Device View Netlist View NCD View Floorplan View
Spreadsheet View (Pin Assignments)	Signal	Package View
	Port Group	Package View
Spreadsheet View (Group)	Anchored UGROUP	Floorplan View
Spreadsheet View (Misc)	REGION	Floorplan View
Package View	Assigned pin	Netlist View Spreadsheet View Floorplan View Device View
	Site	Floorplan View Device View
Device View	DQS, IOL, PFF, PFU, PLL/DLL, DCC/DCS, sysDSP, sysMEM	Floorplan View Physical View
	Others: GSR, JTAG, Oscillator, and others.	Floorplan View Physical View
	Assignable PIO Cell	Package View Floorplan View Physical View
	Unassignable PIO Cell	Floorplan View Physical View
	DDR Support with bonded DQS	Package View Floorplan View Physical View
Netlist View	Assigned port	Package View
	Instance	Floorplan View Physical View NCD View
	Net	Floorplan View Physical View

Cross Probe From	Element	Cross Probe To
NCD View	IOL instance, PCS block, PFF slice, PFU slice, PLL/DLL, sysDSP block, sysMEM block,	Netlist View Floorplan View Physical View
	User PIO	Netlist View Package View Floorplan View Physical View
	Others: GSR, JTAG, Oscillator, etc., Net	Floorplan View Physical View
Floorplan View	Assignable PIO site	Device View Physical View Package View
	Placed PIO	Device View Netlist View NCD View Physical View Package View
	Placed IOL	Device View Netlist View NCD View Physical View
	Any non-PIO site	Device View Physical View
	Any non-PIO placed component Note: To cross-probe PFF or PFU components, select individual slices.	Device View Netlist View NCD View Physical View
	Port	Physical View
	UGROUP, REGION	Spreadsheet View
Physical View	Site	Device View Floorplan View
	Placed component	Device View Netlist View NCD View Floorplan View
	Net	NCD View Floorplan View

4.10. Programming Files

After creating and verifying the design, user can use the final output data file to download or upload a bitstream to or from an FPGA device using the Diamond Programmer. The bitstream file contains all of the configuration information from the physical design that define the internal logic and interconnections of the FPGA, as well as device-specific information from other files associated with the target device.

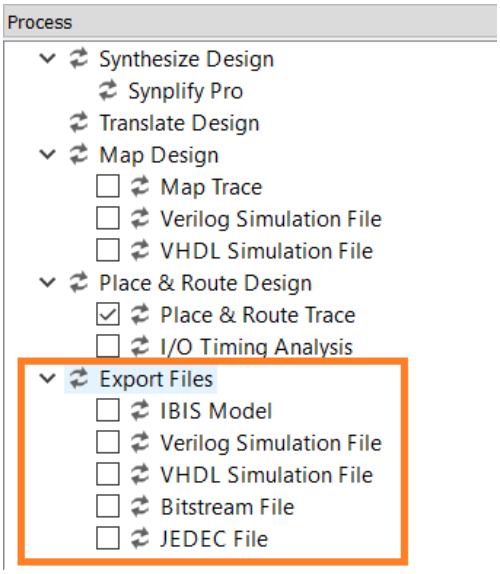


Figure 4.28. Export Files Process Options

Table 4.7. File Formats

File Format	Description
Bit File (binary) (.bit)	Binary bitstream files are the default output of the bitstream process and contain the configuration information in bitstream (zeroes and ones) that is represented in the physical design (.ncd) file.
Raw Bit File (ASCII) (.rbt)	The Raw Bit File is a text file containing ASCII ones and zeros representing the bits in the bitstream file. If the user is using a microprocessor to configure a single FPGA, user can include the Raw Bit file in the source code as a text file to represent the configuration data. The sequence of characters in the Raw Bit file is the same as the bit sequence that is written into the FPGA. The .rbt file differs from the .bit file as it contains design information in the first six lines.
Hex Mask File (.msk)	Used to compare relevant bit locations for executing a read back of configuration data contained in an operating FPGA.
Bit Generation Report File (.bgn)	Outputs information on a bit generation (bitgen) run and displays information on options that are set. This file is output by default and given the name, design_name>.bgn
JEDEC	The programming standard from the Joint Electron Design Engineering Committee, a committee of programmer and semiconductor manufacturers that provide common standards for programmable issues. Examples include acceptable test characters for PLDs and standard data transfer/programming formats for PLDs. The JEDEC Standard is the industry standard for PLD formats. In Diamond Programmer, JEDEC usually refers to the JEDEC fuse map of the design for the device that the user selected.

For more information, refer to Diamond Help or to the following sections in the Lattice Diamond 3.12 Programming Tools User Guide.

- Programming Files (Chapter 5)
- Programmer (Chapter 7)

4.11. Reports

In Xilinx Vivado, there are report strategies which defines the reports that are created during synthesis or implementation. User can check the report strategies by going to the Tool Settings > Strategies > Report Strategies on the Settings window. User can also generate reports for Timing Summary, Clock Networks, Clock Interaction, Methodology, DRC, Noise, Utilization, and Power on the Flow Navigator pane by going to Synthesis (or Implementation) > Open Synthesized Design (or Open Implemented Design). In Lattice Diamond, user can click on the Reports tab, which is open by default on the Diamond workspace. Otherwise, go to *View* > *Reports*.

Each step on the design flow has its own set of reports from resource usage to timing analysis.

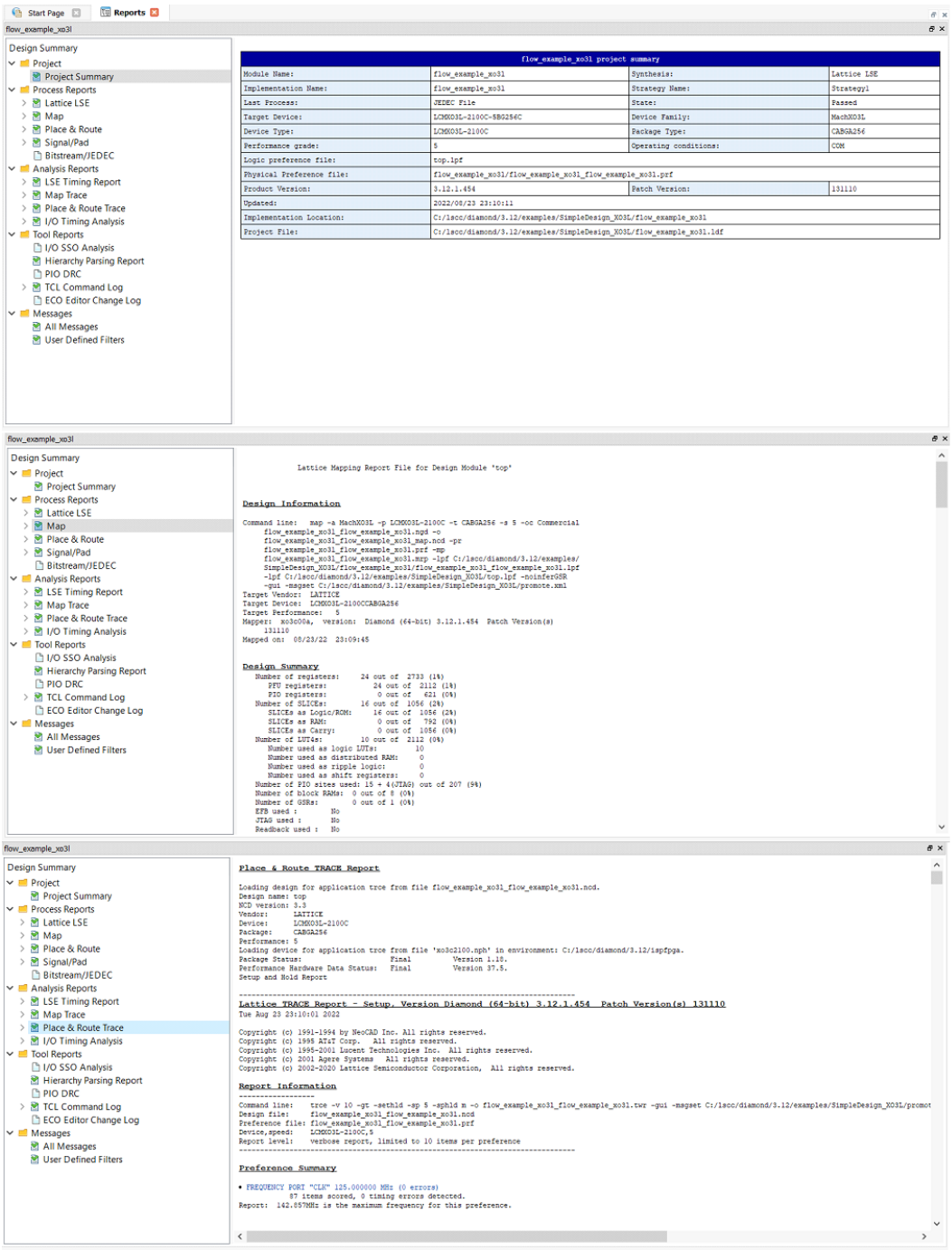


Figure 4.29. Reports Tab

For more information, refer to Diamond Help or to the following sections in the Lattice Diamond 3.12 User Guide:

- Reports (Chapter 4)
- Reports (Chapter 7)

5. Design Verification Tools

5.1. Simulation

Xilinx Vivado and Lattice Diamond both support a number of third-party simulators, as shown in Table 5.1.

Table 5.1. Supported Simulators

	Xilinx Vivado	Lattice Diamond
Mentor Graphics Questa Advanced Simulator	✓	✓
Mentor Graphics ModelSim Simulator	✓	✓
Synopsys Verilog Compiler Simulator (VCS)	✓	✓
Aldec Rivera-PRO Simulator	✓	✓
Aldec Active-HDL	✓	✓
Cadence Xcelium Parallel Simulator	✓	—
Cadence Incisive, Enterprise Simulator (IES)	✓	—
Xilinx Vivado simulator	✓	—
Cadence NC-VHDL	—	✓
Cadence NCSim	—	✓
Cadence NC-Verilog	—	✓

In Xilinx Vivado, user can run a simulation by clicking Run Simulation under Simulation on the Flow Navigator pane. User can choose between Behavioral Simulation, Post-Synthesis Simulation, and Post-Implementation Simulation. In Diamond, user can click on the Simulation Wizard icon to run simulation using Modelsim OEM tool, which is a directly-linked third-party simulator. Simulation wizard is used to generate Simulation Wizard Project (.spf) file and a simulation script DO file that is executed by ModelSim.

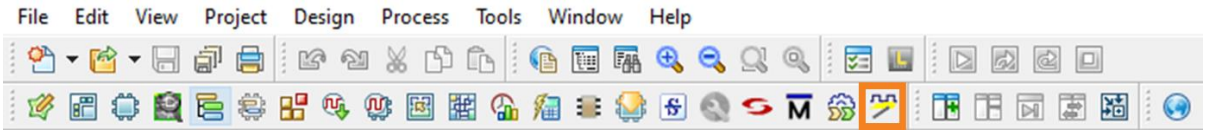


Figure 5.1. Opening Simulation Wizard from the Toolbar

Lattice Diamond also comes with a standalone version of ModelSim that can be used for project simulation. For third-party simulators, cmpl_libs TCL script is used to compile the device libraries needed by the project. For details on performing simulation using the third-party simulators, refer to the Third-Party Simulators section of the Diamond Help.

5.1.1. Simulation Levels

Depending on the output files chosen on the Process toolbar, user can perform different simulation levels. In Xilinx Vivado, there are three levels of simulation: behavioral, post-synthesis, and post-implementation with options of performing a functional or a timing simulation for post-synthesis and post-implementation. In Diamond, there are three simulation levels: the RTL, the post-map gate-level, and the post-route gate-level+timing simulations, which gives user more options when simulating designs before performing hardware verification.

Table 5.2. Simulation Level Comparison between Xilinx Vivado and Lattice Radiant

Simulation Level in Xilinx Vivado	Equivalent Simulation Level in Lattice Diamond
Behavioral	RTL
Post-synthesis (Functional or Timing)	Post-map gate-level
Post-implementation (Functional or Timing)	Post-route gate-level+timing

User can choose between the simulation levels on the Process Stage of the Simulation Wizard window.

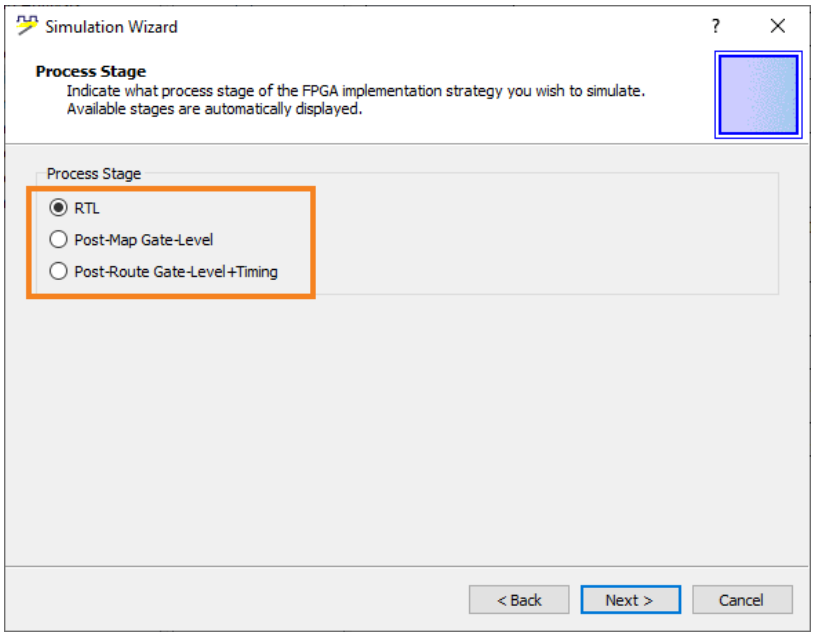


Figure 5.2. Simulation Wizard Window Showing the Simulation Levels

Each of these simulation requires specific output files that can be generated on the design flow before it can be performed:

- RTL Simulation — This can be run with just an HDL file and a testbench file.
- Post-map Gate-Level — Aside from an HDL file and a testbench file, a map gate-level netlist file and standard delay format file (SDF) are needed, which are generated during the Map Design process when Verilog (or VHDL) Simulation File is ticked.
- Post-Route Gate-Level+Timing — Aside from an HDL file and a testbench file, a post PAR gate-level netlist file and an SDF are needed, which are also generated during the Export Files process when Verilog (or VHDL) Simulation File is ticked.

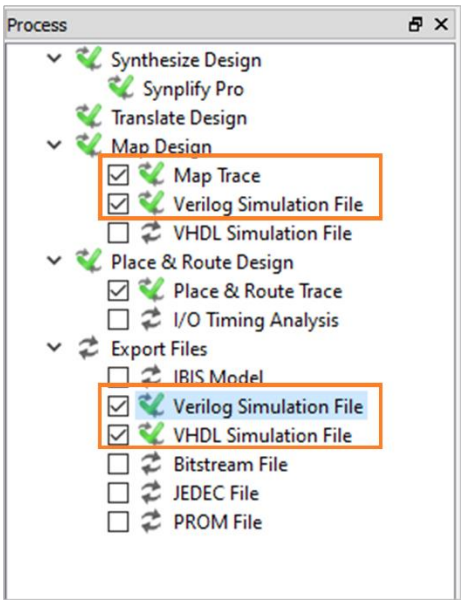


Figure 5.3. Generating a Post-Synthesis Simulation File

For detailed information, refer to the Diamond Software Help and the following sections in the Lattice Diamond 3.12 User Guide:

- Simulation Flow (Chapter 6)
- Simulation Wizard (Chapter 7)
- Mentor ModelSim (Chapter 7)
- Compile Lattice FPGA Simulation Libraries (Chapter 8)

5.2. Static Timing Analysis (STA)

Xilinx Vivado has a number of reports related to STA that would guide the user on how to close timing. In Lattice Diamond, TRACE analyzes the timing preferences that are present in the logical preference file (LPF). These timing constraints are defined in the Timing Preferences sheet of the Spreadsheet View or in a text editor and applied to the design before the design is mapped.

A TRACE report file, which shows the results of timing preferences, is generated each time the user runs the Map Trace process or the Place & Route Trace (PAR) process. The results can then be viewed in the Report View window. The Map TRACE report (.tw1) contains estimated routing that can be used to verify the expected paths and to provide an estimate of the delays before users run Place & Route. The PAR TRACE report (.twr) contains delays based on the actual placement and routing and is a more realistic estimate of the actual timing.

Timing Analysis View (Figure 5.5) is a graphical view of the post-route TRACE report. It provides path tables, schematic views of timing paths, and a report of each timing preference. It also allows the user to cross-probe to Floorplan View or Physical View to see where these paths exist on the chip.

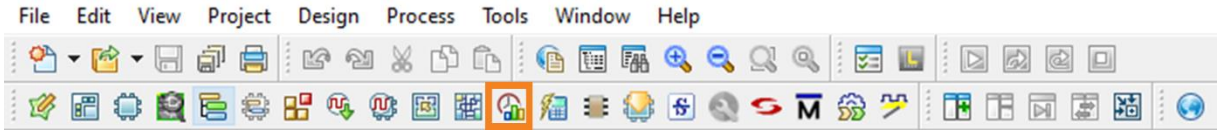


Figure 5.4. Opening the Timing Analyzer Tab

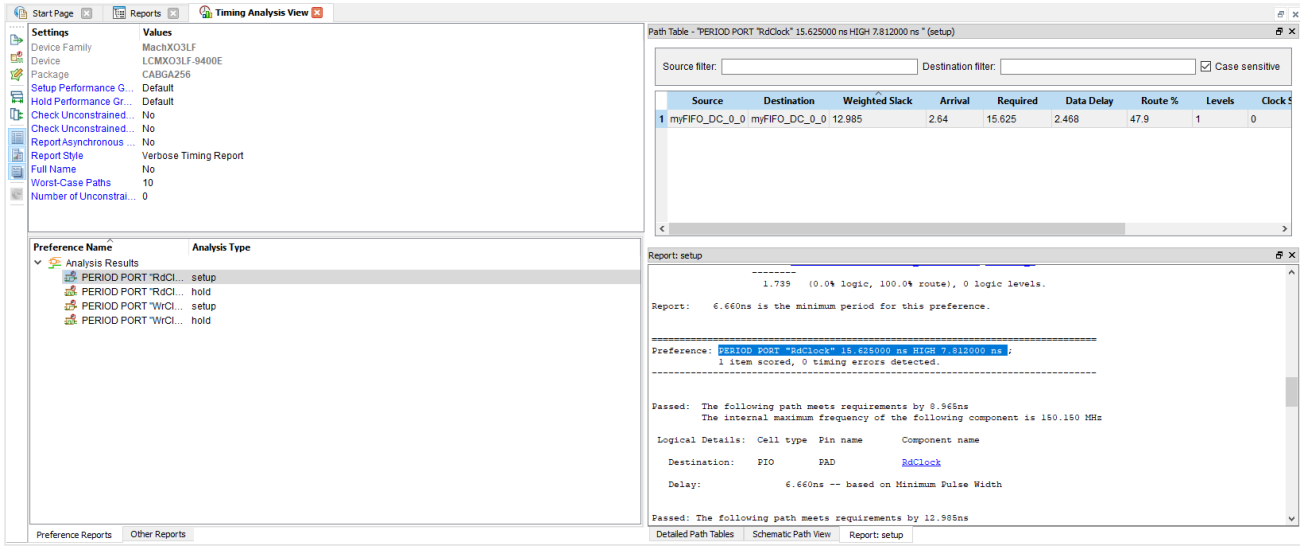


Figure 5.5. Timing Analysis View

For detailed information about TRACE and Timing Analysis View, see *Analyzing Static Timing* in the Diamond Software Help and the following sections in the Lattice Diamond 3.12 User Guide:

- Timing (Chapter 5)
- Timing Analysis View (Chapter 7)

5.3. On-Chip Hardware Debugging using Reveal

The final stage of developing the design is the actual verification process on either a test board or in the user's system. The on-chip debugging tools allow live hardware aspect checking in the user design, which helps to quickly do a verification without the use of any external equipment.

While Xilinx Vivado software offers multiple portfolios of on chip debugging tools, in Lattice Diamond environment, it offers two tools for debugging both the hardware aspect of the design tool to continuously monitor signals within the FPGA for specific conditions, which can range from simple to quite complex and see what is happening inside the FPGA and to even change register values while the user's system is running. Lattice Diamond provides sample designs to get some hands-on experience with the Reveal tools. Go to *File > Open > Design Example* and choose the design with *reveal* on its project name.

- Reveal Analyzer to check for and to analyze specific events on signals
- LatticeMico Debugger to debug LatticeMico32 microprocessor software

Note: LatticeMico Debugger does not support the LatticeMico8 microcontroller.

While Reveal Analyzer and LatticeMico Debugger work independently, these can run at the same time. Before the user starts debugging, each tool requires that a special interface module be added to the design. Then, the user reruns the design implementation process (Synthesize Design, Translate Design, Map Design, Place & Route Design) and generates bitstream data or a JEDEC file (depending on the device family) to program the FPGA. The tools can share the same ispDOWNLOAD cable and JTAG port that Programmer uses to download the design.

For detailed information, refer to the following existing documentations on using Reveal:

- Reveal Troubleshooting Guide
- Lattice Diamond 3.12 User Guide
  - Reveal Inserter (Chapter 7)
  - Reveal Analyzer (Chapter 7)

5.4. Power Analysis

Similar to Xilinx Vivado Power Estimator (EPE) and Power Analyzer tool, Diamond offers the Power Calculator, which estimates the power dissipation for a given design.

Table 5.3. Power Analysis Tools Comparison

Xilinx Vivado	Lattice Diamond
Power Estimator (EPE) and Power Analyzer tool	Power Calculator

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 estimated power consumption. The tool also allows user to import frequency and activity factors from the post-PAR simulation *value change dump* file (.vcd file). After the design information is added, Power Calculator provides accurate power consumption analysis for the design.

Power Calculator provides two modes for reporting power consumption:

- Estimation mode is used before completing the design.
- Calculation mode is based on the physical netlist file (.udb) after placement and routing.

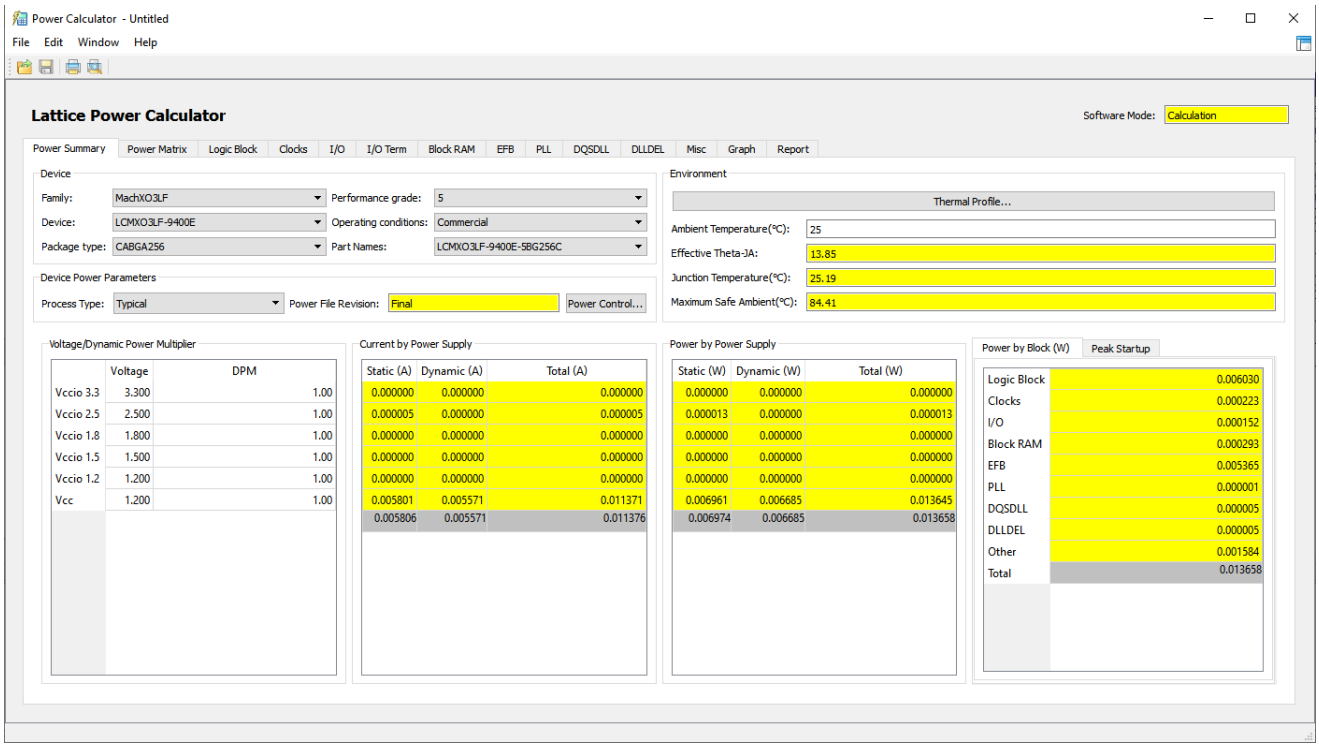


Figure 5.6. Power Calculator

For detailed information, refer to the Diamond Help or to Power Calculator (Chapter 7) in the Lattice Diamond 3.12 User Guide.

6. TCL Scripting

Similar to Xilinx Vivado, Lattice Diamond also support TCL (Tool Command Language) scripting feature that enables a batch capability for running tools in the Diamond software graphical interface. TCL commands can be used through the command line/terminal or the Lattice Diamond stand-alone TCL console that is included in the software package. The command set and the TCL Console used to run it affords user the speed, flexibility and power to extend the range of useful tasks that the Diamond software tools are already designed to perform.

Using the command line tools allows user to do the following:

- Develop a repeatable design environment and design flow that eliminates setup errors that are common in user interface design flows.
- Create test and verification scripts that allow designs to be checked for correct implementation.
- Run jobs on demand automatically without user interaction.

Note: The environments for both the Diamond TCL Console window or Diamond Standalone TCL Console window are already set. User can start entering TCL tool commands or core tool commands in the console and the software executes them.

When running the Diamond software from the Windows command line (through cmd.exe) or Linux terminal (bash), user must set up the environment variables as stated in the *Setting Up the Environment to Run Command Line* section of the Lattice Diamond software Help.

For detailed information, refer to the Diamond Help or to Chapter 9 in the Lattice Diamond 3.12 User Guide.

7. Platform Designer Tool

Diamond includes the Platform Designer tool (as shown in Figure 7.1), which enables user to build and control a complete hardware management system. Platform Designer provides an integrated design environment that enables user to configure the device, implement the hardware management algorithm, simulate, assign pins, and finally generate the JEDEC files required to program and configure the device on the circuit board. It also allows user to import other HDL files to integrate other desired functions

Lattice devices supported in Platform Designer:

- Platform Manager 2
- Platform Manager 2 in a two-device (LPTM21 and LPTM21L) configuration
- Certain devices in the MachXO2™ HC/HE family with external Analog Sense and Control (ASC) devices
- Certain devices in the MachXO3D™ family with external ASC devices
- Certain devices in the MachXO3LF™ family with external ASC devices
- Certain devices in the ECP5 family (LFE5U/LFE5UM) with external ASC devices
- Larger LPTM21 device can configure both ASC and/or LPTM21L as slave devices

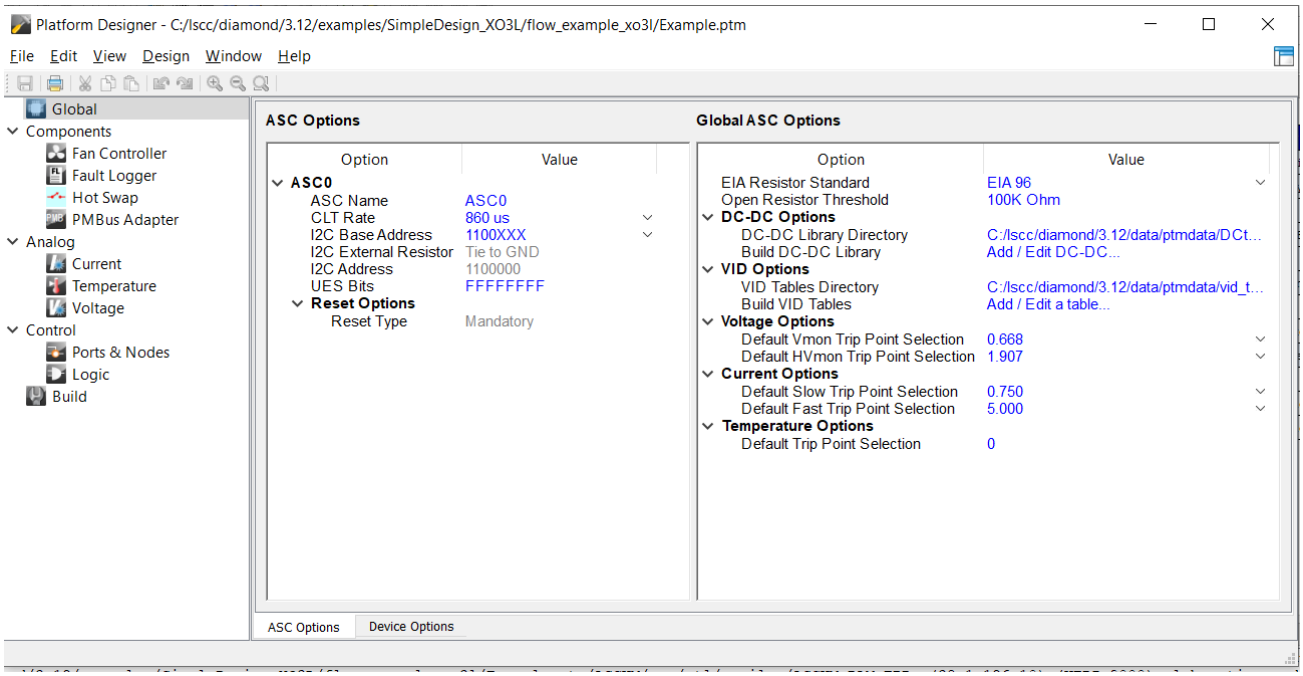


Figure 7.1. Platform Designer User Interface

For detailed information, refer to the Lattice Diamond 3.12 Platform Designer User Guide.

8. Lattice Propel

Similar to the Vitis Core Development Kit in Xilinx, the Lattice Propel is a design environment for Lattice FPGA-based processor system designs. Its development suite includes:

- Integrated development environment (IDE)
- Lattice Propel Builder graphical user interface for System-on-Chip (SoC) design
- Lattice Propel Software Development Kit (SDK) for system software development based on Eclipse Embedded C/C++ Development tools (CDT)

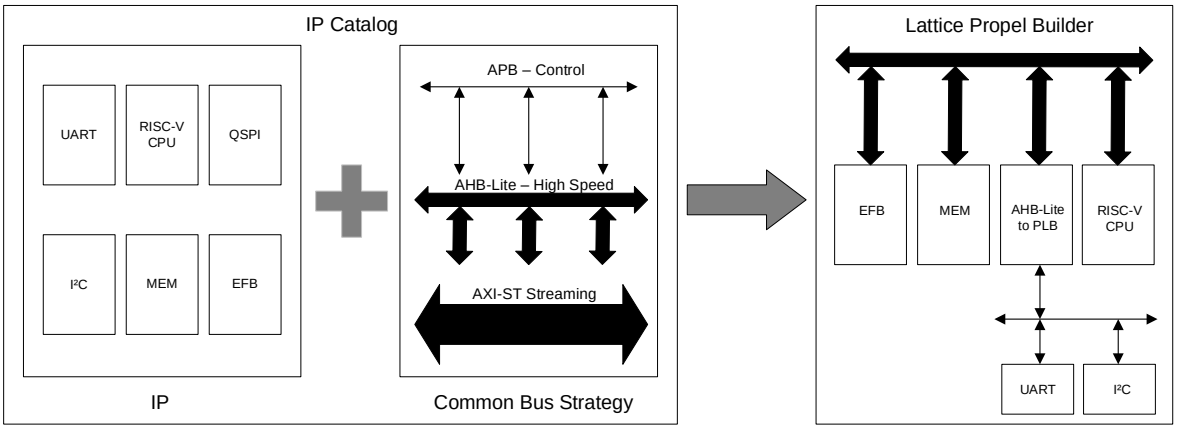


Figure 8.1. Lattice Propel Builder Design Flow

The Lattice Propel flow (as shown in Figure 8.1) starts with the creation on an SoC system design in Lattice Propel Builder using the RISC-V CPU together with APB and AHB-L buses, and peripherals.

A C/C++ project is then created using the environment file of the SoC design. User can then input the executable code on the SoC design as part of the RTL. Typically, this process can be done by loading the memory file on the system memory of the SoC design.

The next step is to generate the RTL files and then go through the design flow using Lattice Diamond or Lattice Radiant Software. Once the configuration is loaded onto the SRAM, the design can be verified through Reveal or through GNU debugger of the SDK.

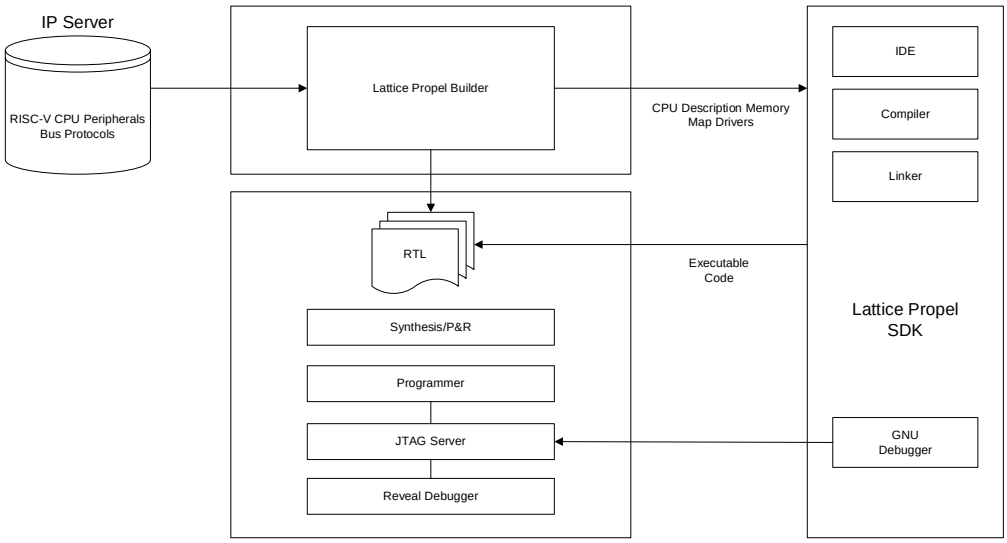


Figure 8.2. Lattice Propel Design Environment

For detailed information on Lattice Propel, see the Lattice Propel web page.

References

- Lattice Diamond 3.12 User Guide
- Lattice Diamond 3.12 Platform Designer User Guide
- Lattice Diamond 3.12 Programming Tools User Guide
- Lattice Synthesis Engine User Guide
- Clarity Designer User Manual
- Xilinx Documentation Portal
- Xilinx Vivado Design Suite User Guide: Getting Started

The following documents are referenced in the default installation path of Lattice Diamond, which is included in the software package. If the user has installed Lattice Diamond other than the default installation path, the documents can be found in the directories below:

Under *<Diamond Installation Path>/<Diamond Version>/modeltech/docs/pdfdocs/*:

- ModelSim User's Manual
- ModelSim GUI Reference Manual
- ModelSim Command Reference Manual
- ModelSim Tutorial

Under *<Diamond Installation Path>/<Diamond Version>/synpbase/doc/*:

- Synopsys Synplify Pro for Lattice User Guide
- Synopsys Synplify Pro for Lattice Reference Manual
- Synopsys Synplify Pro for Lattice Language Support Reference Manual

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

Revision History

Revision 1.0, August 2022

Section	Change Summary
All	Initial release.

www.latticesemi.com