Lattice Prompt AI Design Assistant

User Guide

FPGA-UG-02274-1.0

October 2026

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 all associated risk is the responsibility entirely of the Buyer. The information provided herein is for informational purposes only and may contain technical inaccuracies or omissions, and may be otherwise rendered inaccurate for many reasons, and Lattice assumes no obligation to update or otherwise correct or revise this information. 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. LATTICE PRODUCTS AND SERVICES ARE NOT DESIGNED, MANUFACTURED, OR TESTED FOR USE IN LIFE OR SAFETY CRITICAL SYSTEMS, HAZARDOUS ENVIRONMENTS, OR ANY OTHER ENVIRONMENTS REQUIRING FAIL-SAFE PERFORMANCE, INCLUDING ANY APPLICATION IN WHICH THE FAILURE OF THE PRODUCT OR SERVICE COULD LEAD TO DEATH, PERSONAL INJURY, SEVERE PROPERTY DAMAGE OR ENVIRONMENTAL HARM (COLLECTIVELY, "HIGH-RISK USES"). FURTHER, BUYER MUST TAKE PRUDENT STEPS TO PROTECT AGAINST PRODUCT AND SERVICE FAILURES, INCLUDING PROVIDING APPROPRIATE REDUNDANCIES, FAIL-SAFE FEATURES, AND/OR SHUT-DOWN MECHANISMS. LATTICE EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS OF THE PRODUCTS OR SERVICES FOR HIGH-RISK USES. 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.

Inclusive Language

This document was created consistent with Lattice Semiconductor’s inclusive language policy. In some cases, the language in underlying tools and other items may not yet have been updated. Please refer to Lattice’s inclusive language FAQ 6878 for a cross reference of terms. Note in some cases such as register names and state names it has been necessary to continue to utilize older terminology for compatibility.

© 2026 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

Contents

Contents 3
Abbreviations in This Document 5
1. Introduction 6
1.1. Purpose 6
1.2. Audience 6
1.3. Terms Used in This Document 7
1.4. Supported Toolchain and AI Client 7
1.5. Your Responsibilities 7
2. Lattice Prompt Overview 9
2.1. Architecture 9
2.2. What the Servers Provide 9
2.3. Data Privacy 10
2.4. Knowledge Sources 10
3. Capabilities 11
3.1. Knowledge 11
3.2. SoC Construction 11
3.3. Firmware 12
3.4. Implementation 12
3.5. Long-Running Jobs 13
3.6. Support 13
4. Installation and Setup 14
4.1. Before You Install 14
4.1.1. Operating System 14
4.1.2. Network 14
4.1.3. Python 14
4.1.4. AI Client 14
4.1.5. Lattice Propel and Lattice Radiant 14
4.1.6. Licenses 15
4.2. Installing Lattice Prompt 15
4.2.1. Option A: Install with the Agent 15
4.2.2. Option B: Install Manually from a Terminal 15
4.3. What the Installer Prints 16
4.3.1. If the Installer Fails 16
4.4. Installed Locations 16
4.5. Environment Variables 16
4.6. Verifying the Installation 17
4.7. Choosing a Model 17
4.8. Tool Permissions 17
5. Quick Start 18
6. Design Flow 19
6.1. Design Flow Phases 19
6.2. Rules the Agent Follows 20
6.3. Editing an Existing Design 20
6.4. Timing Closure 20
6.4.1. How a Fix is Tried 20
6.4.2. Radiant Session Configuration 21
6.5. Hardware Debug Flow 21
7. Tutorials 22
7.1. Building a New SoC 22
7.2. Architecting a Larger System 22
7.3. Modifying a Reference Design 22
7.4. Compiling an Existing Radiant Project 22

© 2026 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

7.5. Shorter Requests 22
8. Recommended Practices 24
8.1. Writing the Request 24
8.2. During the Session 24
8.3. Reading the Results 24
8.4. Environment 24
9. Troubleshooting and Support 25
9.1. Creating a Lattice Support Log 25
9.2. Support Log Contents and Redaction 25
9.3. Diagnostic Log and Health Check 25
9.4. Common Problems 26
10. Known Limitations 27
References 28
Technical Support Assistance 29
Revision History 30

Figures

Figure 2.1. Lattice Prompt Architecture 9
Figure 6.1. Design Flow with Approval and Validation Gates 19

Tables

Table 1.1. Lattice Prompt Servers 6
Table 1.2. Terms Used in This Document 7
Table 1.3. AI Client Configuration 7
Table 2.1. Architecture Layers 9
Table 3.1. Knowledge Capabilities 11
Table 3.2. SoC Construction Capabilities 11
Table 3.3. Firmware Capabilities 12
Table 3.4. Implementation Capabilities 12
Table 3.5. Long-Running Job Capabilities 13
Table 3.6. Support Capabilities 13
Table 4.1. Minimum Requirements 14
Table 4.2. Installer Options 15
Table 4.3. Installed Locations 16
Table 4.4. Environment Variables 16
Table 7.1. Shorter Requests 22
Table 9.1. Support Log Contents and Redaction 25
Table 9.2. Common Problems 26
Table 10.1. Known Limitations 27

© 2026 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviation	Definition
AI	Artificial Intelligence
API	Application Programming Interface
CPU	Central Processing Unit
FPGA	Field-Programmable Gate Array
GCC	GNU Compiler Collection
GSRD	Golden System Reference Design
I/O	Input/Output
IP	Intellectual Property
LDC	Lattice Design Constraints
MCP	Model Context Protocol
MIPI	Mobile Industry Processor Interface
PAR	Place and Route
PCIe	Peripheral Component Interconnect Express
PDC	Physical Design Constraints
PHY	Physical Layer
PLL	Phase-Locked Loop
PMI	Parameterizable Module Instantiation
RTL	Register-Transfer Level
SDC	Synopsys Design Constraints
SDK	Software Development Kit
SGE	System Generation Environment
SoC	System On Chip
SSH	Secure Shell
Tcl	Tool Command Language
UART	Universal Asynchronous Receiver/Transmitter

© 2026 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

1. Introduction

1.1. Purpose

Lattice Prompt™ is a software package that lets you drive the Lattice field-programmable gate array (FPGA) design flow from an artificial intelligence (AI) agent, hereafter the agent, using natural language. You describe the design intent in plain language. The agent proposes a design, and with your approval builds and compiles it with the Lattice Propel™ software and the Lattice Radiant™ software.

Internally the package runs two servers for the Model Context Protocol (MCP), the open standard that AI clients use to call external tools. You do not choose between them: the installer always installs both, and the agent calls the server that provides the operation it needs. Together they are called the servers in this guide.

Table 1.1. Lattice Prompt Servers

Component	Server Name	What It Covers
Lattice Prompt server	lattice-prompt	Design consultant: IP catalog and device knowledge, reference designs, Propel SoC construction, RISC-V firmware generation, Radiant project builds, RTL simulation, debug setup for Reveal (Radiant on-chip logic analyzer), hardware-debug analysis, support logs
Lattice Radiant MCP server	lattice-radiant-mcp	Radiant automation: persistent Radiant Tcl session, step-by-step implementation flow (map, place, route, timing, bitstream), timing-closure analysis, RTL and constraint lint, Radiant documentation search, remote execution on a compute farm

Three properties define how Lattice Prompt behaves:

- The agent decides. Lattice Prompt contains no language model of its own and makes no design decisions. It supplies Lattice knowledge, deterministic validators, and tool executors. The agent, running on your AI client model, plans the work, makes the decisions, and explains them.
- Data comes from Lattice sources. IP names, versions, parameters, ports, device families, and pin assignments are read from the installed Lattice toolchain and the catalogs supplied with the servers rather than invented. The validators catch a fabricated IP or parameter before a build runs. They check structure and catalog facts. They do not check that the design does what you intended.
- You approve the design. The agent must present a design proposal and receive your explicit approval before it generates or builds anything. Built-in design checks must pass before the agent may report a build as complete.

Note: Lattice Prompt gives the agent Lattice data and checks. It does not guarantee that a generated design is correct, meets timing, or works on hardware. The validators catch structural and catalog errors. They do not replace functional simulation, timing sign-off, and hardware validation, which remain your responsibility.

1.2. Audience

This guide is for two groups of readers:

- The first group is engineers new to FPGAs or to the Lattice toolchain, such as embedded software developers, system architects, and evaluators, who want a working design without first learning Propel Builder and Radiant. For these readers the agent performs the tool operations. This guide explains what it does and what it needs from you.
- The second group is experienced Lattice FPGA designers who want to shorten routine work and keep full control of the result. For these readers the guide explains how the agent flow maps onto the tools they already know.

The guide assumes that you can describe the system you want in engineering terms, such as the board, the interfaces, the data rates, and the processor, and that you can install software, run a command in a terminal, and set an environment variable.

One point applies to both readers. The agent produces a design, but you remain accountable for it. For more information refer to the Your Responsibilities section. Judging a proposal and verifying the built design on a board call for FPGA fundamentals: what synthesis, placement and routing, timing constraints, and a bitstream are. If these are

© 2026 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

new to you, the Lattice training and the Propel and Radiant user guides listed under References are the place to learn them, and the agent explains any term you ask about during a session.

1.3. Terms Used in This Document

This guide uses one name for each part of the system. The third column names the product that fills each role in this release.

Table 1.2. Terms Used in This Document

Term	Meaning	Embodiment in This Release
Model	Language model that produces the agent text and tool calls. Performs no actions itself.	Model supplied by your AI client
Agent	Model running in a loop that can call tools. Plans, asks, decides, and explains.	Any tool-calling agent
AI client	Application that hosts the agent and the conversation. The servers are registered with it.	Claude Code, the tested AI client for this release
MCP	Model Context Protocol, the wire contract between AI client and server.	Model Context Protocol
Lattice Prompt server	Server that provides design knowledge, validators, and executors. Registered as lattice-prompt.	Lattice Prompt
Lattice Radiant MCP server	Server that drives a Radiant Tcl session. Registered as lattice-radiant-mcp.	Lattice Prompt
Servers	Both servers together.	—
Tool	One operation a server exposes for the agent to call.	propel_connect_net, job_dashboard
Resource	Read-only data a server exposes for the agent to read.	lattice://workflow
Lattice toolchain	Propel and Radiant together, with the simulator that ships with them. Named individually where the difference matters.	Propel Builder, Propel SDK, Radiant, Questa
Reveal	Radiant on-chip logic analyzer, used to capture signals from the running FPGA.	Lattice Radiant
Donor reference design	A Lattice reference design whose clock, reset or connectivity structure is copied into your design, even when its device differs.	Lattice reference designs
Design plan	The record of a session, written beside the project as design_plan.html: your request, the proposal, your decisions, and the checks that ran.	Lattice Prompt

1.4. Supported Toolchain and AI Client

This release requires Lattice Propel 2025.2 or later (Propel Builder and Propel SDK) and Lattice Radiant 2025.2 or later. A valid Lattice license is required to build.

This release was developed and tested with Claude Code as the AI client, and the installer registers the servers with it by default.

Table 1.3. AI Client Configuration

Item	Value
Tested AI client	Claude Code
Configuration file written by the installer	The client MCP server registration file. The installer enters both servers there. The file and its location depend on the client. For the tested client, see Installed Locations.

1.5. Your Responsibilities

Lattice Prompt is a design assistant. The agent proposes and builds. You approve each step, and you are accountable for the result:

© 2026 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

- You make the design decisions. Every proposal, every timing fix that changes constraints or RTL, and every deviation from the approved plan is presented for your approval. An approved proposal is your design choice, exactly as if you had entered it in Propel Builder or Radiant.
- You perform the verification. The validators check structure, catalog facts, and timing reports. They do not prove function. Functional simulation, timing sign-off, board bring-up and qualification against your product requirements remain your work, whatever the agent reports.
- Review before use. Generated RTL, constraints, firmware, Tcl scripts, and build settings are inputs to your project, not finished deliverables. Review them as you would review code from a colleague before committing them to a product, and keep them under your own revision control.
- Licensing and terms. Use of Lattice Prompt is subject to the AI Solutions Software License Agreement shipped with the installer. Propel, Radiant, and the AI client each have their own license terms, and the IP and reference designs the catalog points to carry their own licenses, which you accept when you download them.
- Data you send to the AI provider. The agent sends your prompts, and whatever design content it reads, to the AI provider behind your AI client. Lattice Prompt neither sends nor sees that traffic. Confirm that your AI client data handling meets your organization's confidentiality requirements before using it on proprietary designs.

© 2026 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

2. Lattice Prompt Overview

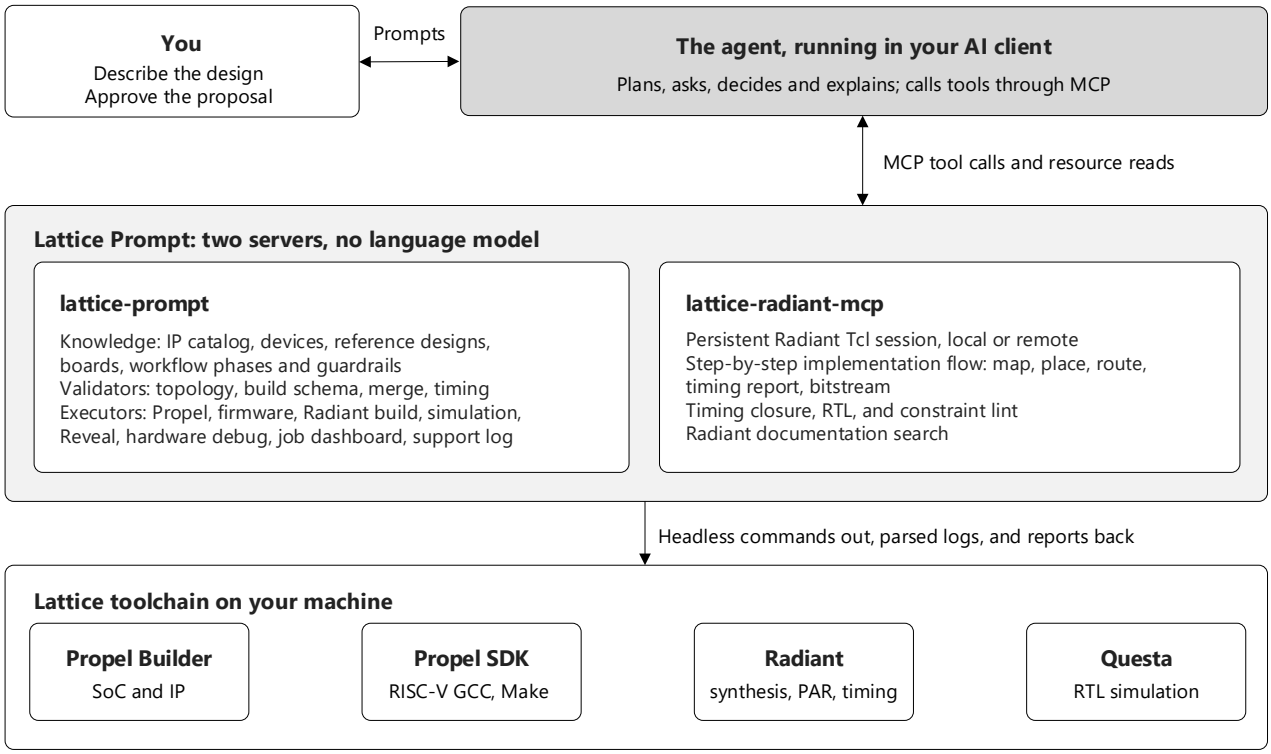


Figure 2.1. Lattice Prompt Architecture

You interact with the agent in your AI client. The agent reads Lattice knowledge from the servers and calls their validators and executors. The servers run Propel and Radiant in console mode and return parsed results to the agent.

2.1. Architecture

Table 2.1. Architecture Layers

Layer	Role	Where it Runs
Agent	Holds the conversation, reasons about requirements and architecture, chooses IP, writes constraints and firmware, decides when to call a tool, and explains results.	Your AI client, with the model of your AI provider
Servers	Supply knowledge taken from Lattice sources, deterministic validators, and tool executors. Contain no language model and make no design decisions.	A Python virtual environment on your machine, registered with your AI client
Lattice toolchain	Propel Builder, Propel SDK, Radiant, and the bundled Questa simulator perform the construction, build, and simulation work.	Your machine

2.2. What the Servers Provide

- Resources are read-only knowledge the agent loads on demand: the IP catalog, device constraints, reference designs, the firmware flow, the Radiant command-line reference, and a set of workflow resources, one per design phase. These tell the agent which checks to run, in which order, and which decisions need your approval.
- Validators are deterministic checks that return pass or fail and list each violation with its source. Some are hard facts, such as whether an IP exists, whether a family supports it, or whether a timing endpoint fails. Others are recommendations that the agent may weigh and you may override.

© 2026 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

- Executors run the toolchain: Propel sessions, IP generation, firmware compilation, Radiant builds, RTL simulation, Reveal setup. Long jobs run in the background and can be watched on the build dashboard.

2.3. Data Privacy

The servers run locally and, by default, communicate with the agent over standard input and output. Your design files, build logs, license settings, and the diagnostic log are not transmitted anywhere by Lattice Prompt. Some actions you invoke do reach online services, such as downloading a reference design or running a live Lattice IP-catalog query. These send only the request, never your design. What the agent sends to its AI provider is governed by that provider and your AI client settings, not by Lattice Prompt. Lattice Prompt writes a support log to disk for you to review and attach to a support case. It is never transmitted automatically, and because it retains project and design names for triage, review and redact it before attaching.

2.4. Knowledge Sources

The catalogs supplied with the servers are derived from Lattice sources: reference designs, IP user guides, device data sheets, and evaluation-board documentation published on the Lattice website and in Lattice GitHub repositories. The servers contain metadata derived from those sources. They do not contain the documents themselves. Two actions reach Lattice online services on your request: a live IP-catalog query and a reference-design download. Both send only the request. Some reference designs and documents cannot be downloaded by the agent, for example, those that require a Lattice account. In that case the agent provides the link to the Lattice website or repository where the file is published, and continues with the file once you have downloaded it and stated its location. Reference designs published on GitHub are cloned directly.

© 2026 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

3. Capabilities

Lattice Prompt exposes three kinds of capability to the agent: knowledge resources it reads, deterministic validators that return pass or fail with cited violations, and executors that run Propel and Radiant. The tables below group them by design phase.

3.1. Knowledge

Table 3.1. Knowledge Capabilities

Capability	Description
IP catalog	Search the installed Propel and Radiant catalogs and the Lattice IP server. Read every parameter, port, bus interface, memory map, and address space of an IP. Resolve the best installed version and check compatibility against a device, package, speed grade, and toolchain version.
Parameter dependencies	Which parameters control which others, and which ports appear or disappear when a parameter changes, before the change is made.
Device data	Device families, packages, speed grades, per-family limits, and the IP each family supports.
Reference designs and boards	Search the catalog of Lattice reference designs and evaluation boards, find related designs that use the same IP, check family compatibility, and download a design as a starting point.
Primitive and PMI lookup	Find device primitives and Parameterizable Module Instantiation (PMI) modules with their instantiation templates.
Workflow guidance	Phase-by-phase resources that tell the agent what to check, in what order, and which decisions require your approval.
Radiant documentation search	Search the Radiant documentation knowledge base and, through the web-docs skill, version-matched Lattice documents. Provided by the Radiant server.
Design-file readers	Parse a Propel .sbx design, a Radiant .rdf project, or a constraint file into structured facts, report every connection of one instance, and emit the design file list.

3.2. SoC Construction

Table 3.2. SoC Construction Capabilities

Capability	Description
Propel SoC design	Create a project from scratch, from a Propel template, or from an existing design file. Configure and add IP, connect scalar nets and bus interfaces, add top-level ports, assign address segments, add glue logic. The agent queues every change and previews it before execution.
Clock and reset	Clock-domain planning, reset-synchronizer topology derived from reference designs, and a completeness check that every domain has a reset.
Design audit and rationale	A pre-compile review that compares the built design against the approved proposal, so that what was built is what you approved, and a rationale store that records why each decision was made, as an audit trail for later review.
Design plan	The record you can hand to a reviewer, kept at design_plan.html beside your project: the prompt verbatim, the proposal, your decisions, the gates that ran, and any recorded deviation.
Validation gates	Checks on the proposed design structure (topology), on the build description before Propel runs (build schema), on the combination of several reference designs (merge), and on the timing results. Topology and timing must pass before a build is reported complete.
Advice	Advisory notes on a proposed design, grouped by application area: bus-width mismatches, missing reset or firmware plans, and duplicate or unconnected IP instances. The agent may override a note with a stated reason.
IP installation	Install IP from the Lattice IP server or from .ipk files into the Propel or Radiant catalog, and resolve where an IP is installed.
Protocol bridges	Find installed IP that converts between two bus protocols when two interfaces to be connected do not match.

© 2026 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

Capability	Description
Session recovery	Rebuild a Propel session after a server restart, resynchronize it with its saved Tcl, check that script, or import an existing Tcl script into a session.
Grounding checks	Check that every element of a proposal is traceable to its reference design and that a testbench plan matches the files on disk before either is presented.

3.3. Firmware

Table 3.3. Firmware Capabilities

Capability	Description
Project scaffold	Board support package, drivers, linker script, Eclipse project files, and self-tests generated from the system generation environment file (SGE), which Propel produces with the design. Generation needs no compiler.
Driver knowledge	The API and real usage examples of each Lattice driver, with a coverage check against the peripherals in your design.
Build and verify	Compilation with the RISC-V cross-compiler from the Propel SDK, a build verification step, and conversion of the binary to a memory-initialization (.mem) file that preloads the SoC boot memory with the firmware.
Firmware tests	Generate and run a host unit-test harness for the firmware project, generate the firmware memory-initialization file only if the tests pass, and import the sources of a reference C project.

3.4. Implementation

Table 3.4. Implementation Capabilities

Capability	Description
Radiant builds	Project setup, source management, strategy selection, synthesis, map, place and route, timing analysis, and bitstream generation. Constraint generation is included: pin and I/O constraints are generated into a post-synthesis PDC from the cataloged board or the donor design (a reference design whose structure is copied into yours), and clock definitions are written from the constraint reference the server provides (see Constraints below). Pre-synthesis SDC and LDC files supplied with a design are added to the project as well.
Constraints	Pin assignments generated from the cataloged evaluation board or a donor reference design, I/O standard checks against the target board, and a reference for every constraint keyword.
Timing	Timing report parsing, critical-path extraction, and the timing validation gate with constraint-coverage checking.
Simulation	Simulation-project discovery, model provisioning, testbench inventory and proposal, setup validation, RTL simulation with the Questa simulator bundled with Radiant, and a simulation report.
Hardware debug	Reveal probe proposal and setup generation, build-log parsing, build-error lookup, failure classification, and a design summary with references to the reports it was taken from, used for root-cause analysis.
Radiant session and Tcl	Start, inspect, and stop a persistent Radiant session and run Radiant Tcl commands in it. Provided by the Radiant server.
Step-by-step implementation flow	Load a saved Radiant design and run map, placement, routing, timing report, design report, and bitstream generation step by step, with per-step options. Provided by the Radiant server.
Remote execution	Configure and test running Radiant on another machine or a compute farm. Provided by the Radiant server.
Timing-closure helpers	List the catalog of known timing fixes, find the fixes that match a failure, check whether a strategy setting applies to a device family, lint RTL and constraints, and look up how device primitives are connected and instantiated. Provided by the Radiant server.

© 2026 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

3.5. Long-Running Jobs

Synthesis, place and route, firmware compilation, and RTL simulation run in the background. They do not block the conversation, so you can continue to work with the agent while a build runs.

Table 3.5. Long-Running Job Capabilities

Capability	Description
Build dashboard	A local web page listing every job with a live log tail, a cancel button, and copyable commands. The dashboard is a separate process: it continues to run if the session is interrupted and remains available after the session ends. It is started only on request.
Job status, history, cancel	The state of any job by its identifier, the full job history newest first, and cancellation of a running job.

3.6. Support

Table 3.6. Support Capabilities

Capability	Description
Support log	On request, a redacted support bundle beside your design: a structured report with evidence grade, hypotheses, attribution, and next step (described under Creating a Lattice Support Log), plus build logs, session replay, rationale, and the design plan. Nothing is transmitted. You attach the bundle to a Lattice support case.
Diagnostic log	A local rotating log that never leaves your machine.

© 2026 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

4. Installation and Setup

Installation requires one command, which you can run yourself or have the agent run for you. The release zip carries both servers, each in its own directory, and a top-level installer that runs each component installer in turn. Before You Install lists what must be in place first.

4.1. Before You Install

Complete these steps once, before running the Lattice Prompt installer. Table 4.1 summarizes the minimum requirements.

Table 4.1. Minimum Requirements

Requirement	Minimum
Operating system	Windows 10 or 11, or 64-bit Linux
Python	3.11 or newer, 64-bit, on the path
AI client	An MCP-capable AI coding client with an account at its AI provider. This release was tested with Claude Code
Lattice Propel	2025.2 or later, Propel Builder and Propel SDK
Lattice Radiant	2025.2 or later, including the bundled Questa simulator for RTL simulation
Licenses	A Lattice license reachable through LM_LICENSE_FILE, and SALT_LICENSE_SERVER as well if you simulate
Network	Access to a Python package index during installation, and access to the Lattice website for tool and reference-design downloads

4.1.1. Operating System

Lattice Prompt runs where the Propel and Radiant tools run: Windows 10 or 11, or 64-bit Linux. The servers are installed per user and need no administrator rights beyond what the Lattice tools themselves require.

4.1.2. Network

The installer downloads the server Python dependencies from a package index, so the machine needs that access once, at installation time. Afterward, Lattice Prompt itself needs no network. The Lattice website is reached only when you download a tool, a license, or a reference design, and your AI client reaches its own provider as it normally does.

4.1.3. Python

Install Python 3.11 or newer, 64-bit, from python.org. On Windows choose the **Windows installer (64-bit)** and select **Add python.exe to PATH** during setup. The Python from the Microsoft Store cannot host the servers. Confirm the version with the following command:

```
python --version
```

4.1.4. AI Client

Lattice Prompt runs inside an AI coding client. It does not include one. Install the client from its vendor, sign in with your account, and start it once so that its configuration file exists. This release was tested with Claude Code.

4.1.5. Lattice Propel and Lattice Radiant

Install Lattice Propel 2025.2 or later and Lattice Radiant 2025.2 or later from the Lattice website. Propel Builder and Propel SDK are both required and are installed by the one Propel installer. The default locations are found automatically: *C:\lscc\propel\<version> and C:\lscc\radiant\<version>* on Windows, and */usr/local/lattice/propel/<version> and /usr/local/lattice/radiant/<version>*, or the same under */opt/lattice*, on Linux. For any other location, set the four location variables as described under Environment Variables.

© 2026 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

4.1.6. Licenses

A Lattice license is required to build. Obtain it from the Lattice licensing page, then set two variables so that both the Lattice tools and the Questa simulator can find it. <license> is either the path to your license file or port@host for a license server.

On Windows, enter the following in a Command Prompt:

```
setx LM_LICENSE_FILE "<license>"
setx SALT_LICENSE_SERVER "<license>"
```

On Linux, enter the following in your shell profile (bash or zsh):

```
export LM_LICENSE_FILE=<license>
export SALT_LICENSE_SERVER=<license>
```

For csh or tcsh, use setenv instead:

```
setenv LM_LICENSE_FILE <license>
setenv SALT_LICENSE_SERVER <license>
```

Restart the AI client after setting them, because it passes its environment to the servers at start-up.

4.2. Installing Lattice Prompt

There are two ways to install: let the agent do it, or run the installer yourself from a terminal. Both produce the same result. The unzipped folder can be deleted after installation.

4.2.1. Option A: Install with the Agent

Once your AI client is installed and signed in, unzip the release into a folder of your choice, then enter the following, with the folder path filled in:

```
Install the Lattice Prompt bundle in <folder where you unzipped the release>
```

The agent reads the README in the bundle, checks for a suitable Python, runs the installer for both servers, and tells you what is missing. It asks for anything only you can supply, such as your Lattice account for the Propel and Radiant downloads or your license details.

4.2.2. Option B: Install Manually from a Terminal

Use this if you prefer to install it by hand. The installer installs both servers and registers them with your AI client.

On Windows, in PowerShell or a Command Prompt:

```
cd <folder where you unzipped the release>
python install.py
```

On Linux, in a terminal:

```
cd <folder where you unzipped the release>
python3 install.py
```

If python is not on your path, call the interpreter by its full path instead, for example, *C:\Program Files\Python311\python.exe* install.py. Restart your AI client afterward. The same command upgrades an existing installation, and python install.py --uninstall removes it. Run python install.py --help for the full list of options. Table 4.2 lists the options used in this guide.

Table 4.2. Installer Options

Option	Effect
--force-reinstall	Install this version of a server even when an equal or newer one is present. Only the lattice-prompt server supports it, so use it as --only lattice-prompt --force-reinstall.
--uninstall	Remove everything the installer wrote. The build-job cache is kept so build history survives a reinstall.

© 2026 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

4.3. What the Installer Prints

The installer runs the installer of each server in turn and prints one line per step: Python check, virtual environment, server install and import check, Propel and Radiant detection, execution backend, client registration. It finishes with a summary:

```
  OK    lattice-prompt
  OK    lattice-radiant-mcp
```

Both lines must read OK. If one reads FAIL, the other server is already usable and only the failed one needs attention. See If the Installer Fails below. Installation takes a few minutes on a normal network connection.

4.3.1. If the Installer Fails

The installer stops with a message that names the cause. Whichever route you took, the quickest recovery is to give the failure output to the agent:

```
The Lattice Prompt installation failed. Read the installer output, find the cause, fix it and run the installer again.
```

The agent reads the output and the bundle README, applies the fix it finds there, such as selecting a 64-bit Python, setting a missing environment variable, or adding your organization's root certificate to the operating-system certificate store (see Common Problems), and re-runs the installer, using --only lattice-prompt or --only lattice-radiant-mcp when one server needs attention. It reports what it changed and asks before anything that needs your account, your license or administrator rights. The installer never bypasses a certificate check, so a network that blocks the package index must be resolved by your network administrator. The agent reports this rather than attempting a workaround.

If the agent cannot resolve it, the causes and fixes are listed under Common Problems. For a support case, copy the installer output into the case. The support log is produced by the installed server and is available only after a successful installation.

4.4. Installed Locations

Table 4.3. Installed Locations

Item	Windows	Linux
Virtual environments	%APPDATA%\lattice-prompt\venv, %LOCALAPPDATA%\lattice-radiant-mcp\venv	~/.local/share/lattice-prompt/venv, ~/.local/share/lattice-radiant-mcp/venv
Configuration	%APPDATA%\lattice-prompt\, ~/.claude.json, %APPDATA%\radiant-mcp\config.json, ~/.claude/.mcp.json	~/.config/lattice-prompt/, ~/.claude.json, ~/.config/radiant-mcp/config.json, ~/.claude/.mcp.json
Build-job cache and diagnostic log	%APPDATA%\lattice-prompt\cache\	~/.cache/lattice-prompt/
Default project directory	%USERPROFILE%\LatticeProject	~/LatticeProject
Diagnostic log	%APPDATA%\lattice-prompt\cache\logs\lattice_prompt.log	~/.cache/lattice-prompt/logs/lattice_prompt.log

New projects, firmware workspaces, and generated IP go under the default project directory unless you name another location. Keep the path short. Windows limits a resolved path to about 260 characters, and a long project path causes failures whose messages do not identify path length as the cause.

4.5. Environment Variables

All are optional. The Lattice Prompt server auto-discovers Propel and Radiant from the standard install locations.

Table 4.4. Environment Variables

Variable	Purpose
LATTICE_PROPEL, LATTICE_PROPEL_BASE	Propel install root when it is not in a default location. Different parts of the server read the two names. Set both to the same path.

© 2026 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

Variable	Purpose
LATTICE_RADIANT, LATTICE_RADIANT_BASE	Radiant install root when it is not in a default location. Different parts of the server read the two names. Set both to the same path.
LATTICE_CACHE_DIR	Location of the Lattice Prompt cache and logs when not the default. See Installed Locations and Diagnostic Log and Health Check.
RADIANT_EXE	Radiant executable used by the persistent Radiant session. See Radiant Session Configuration.
RADIANT_MCP_CONFIG	Path of the Radiant server configuration file. See Radiant Session Configuration.
RADIANT_REMOTE_MODE, RADIANT_SSH_*	Remote execution of the Radiant session: local or ssh, and the SSH connection settings. See Radiant Session Configuration.

4.6. Verifying the Installation

Start a new session in your AI client and ask:

```
Is Lattice Prompt connected?
```

The agent reports whether the two servers, lattice-prompt and lattice-radiant-mcp, are connected and summarizes what each provides. The wording differs from session to session and between AI clients. The reply below is from a test session:

```
Yes, the lattice-prompt server is connected. Its tools are now available in this session.
lattice-prompt: Connected. It provides tools for Propel, firmware, simulation, Radiant builds, timing validation, IP search and design plans.
lattice-radiant-mcp: Also connected. It provides the step-by-step implementation flow, Radiant documentation search and timing-closure analysis.
```

Confirm that both servers are reported as connected.

If a server does not appear after a restart, open the client MCP server registration file (see Installed Locations), confirm that the lattice-prompt and lattice-radiant-mcp entries exist, and confirm that the command path in each entry is an existing Python executable (for example, run it with --version).

4.7. Choosing a Model

The design workflows involve long multi-step reasoning. Use the most capable model your AI client offers.

4.8. Tool Permissions

The AI client asks for your permission before each tool call. This is the client's own safeguard, not a Lattice Prompt setting. A design session involves dozens of tool calls, so you may be asked many times in a row. Answering each prompt as it appears is the safest choice.

Optionally, you can allow the two Lattice Prompt servers once in the client settings file so the client stops asking for those tools. Each client keeps this file in its own location and format: a user-level file such as <client settings folder>/settings.json applies to every project, and a project-level file such as <project folder>/<client folder>/settings.json applies to one project. Refer to your client documentation for the location. The example below uses the format of the tested client:

```
{
  "permissions": {
    "allow": ["mcp__lattice-prompt__*", "mcp__lattice-radiant-mcp__*"]
  }
}
```

Do not add shell commands or file writes to the allow list. Let the client keep prompting for those. The servers themselves never need elevated privileges.

© 2026 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

5. Quick Start

This chapter runs one complete session so that you can see what to expect. It uses the same LPDDR4 demo as the first tutorial, Building a New SoC, so the two chapters describe one design: this one as the sequence of what you do, the tutorial as what the agent does behind the scenes. It assumes the installation is verified and a Lattice CertusPro™-NX Versa board is the target.

1. Start a session. Create an empty folder, for example, a new folder under <default project directory>, and start your AI client in that folder. The agent writes the project there.
2. Enter the request. Type: *I want to build an SoC to demo the LPDDR4 read and write access sequence with a CPU. Show the transaction on UART: what is written, what is read back, and whether they match. Target the CertusPro-NX Versa board.*
3. Answer the questions. The agent asks about anything that would change the hardware, such as the memory test patterns to run or the UART baud rate. Each question comes with a recommendation. Respond to each clarification according to your requirements, or accept the recommendation.
4. Read the proposal. The agent lists the IP it uses with versions, among them the RISC-V processor, the LPDDR4 memory controller, and the UART, then the clock and reset plan, the address map, and the top-level ports, and offers to record the session in a design plan. Nothing is built yet. The tutorial Building a New SoC shows this proposal in full.
5. Approve. Reply, for example, *Approved*. After you approve, the agent builds exactly what the proposal describes.
6. Allow the tools. The client asks permission the first time each Lattice tool is called. Allow them once for the session, or add them to the allow list as described under Tool Permissions.
7. Wait for the build. The SoC is generated, the firmware is compiled, and Radiant runs synthesis, placement and routing in the background. To watch progress, ask in plain words, for example, Show the build dashboard. The agent starts a local web page that lists running and finished jobs with a live log tail, and it keeps updating if the session ends. Synthesis, placement and routing take time, and how long depends on the design and your machine. The agent reports when the build completes, and you can continue the conversation or leave the session while it runs. If you leave the session, check the job status or the build dashboard when you return.
8. Read the result. The agent reports whether the topology and timing checks passed, where the bitstream is, and anything it could not do. A report of *complete* means the flow finished and both checks passed. It does not prove that the design works on the board.
9. Find your files. Under the project folder: the Propel design (.sbx) and generated RTL, the Radiant project and its implementation directory with the bitstream and reports, the firmware workspace, design_plan.html, and the build logs.
10. Next steps. Run a simulation or ask for changes. Each is a new request in the same session.

© 2026 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

6. Design Flow

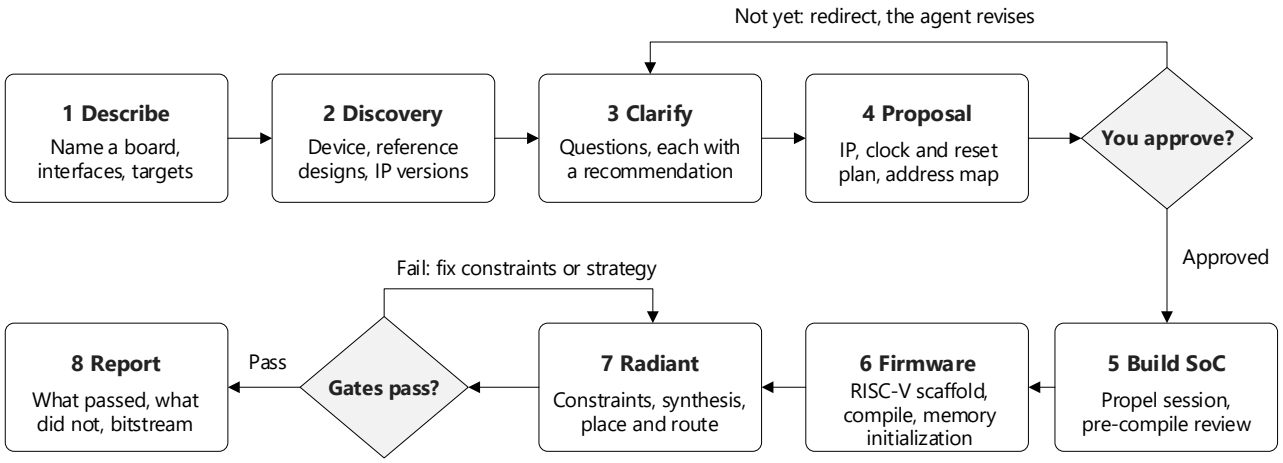


Figure 6.1. Design Flow with Approval and Validation Gates

The agent is guided to follow this flow for every design task that touches a real project. The two diamonds are the gates: your approval before anything is generated, and the validators before anything is reported complete. A complete build has finished and passed both gates. It does not prove that the design is correct.

6.1. Design Flow Phases

These are the phases the agent works through. You answer questions in Clarify, approve the Proposal, and review the Report.

1. Describe. State the outcome: the board or device, the interfaces, the data rates or video formats, the processor if you need one. You do not need to name IP.
2. Discovery. The agent resolves the board to a part number, searches for reference designs and evaluation boards, finds each candidate IP in the installed catalog, resolves its version, and checks compatibility with the target. The agent runs discovery even when your request already names specific IP or a structure, because what you name may not exist in the Lattice catalog for that device. If it does not, the agent says so and proposes the closest supported option.
3. Clarify. Where two readings of your request would produce different hardware, the agent asks. Each question comes with a recommendation, the alternatives, and what changes with the answer.
4. Proposal. The agent presents the IP list with versions, the clock plan, the reset plan, the address map, the top-level ports, and where the pin and timing constraints come from (the board or a donor design). The topology validator runs before the proposal is shown. The agent offers to open a design plan that records the prompt verbatim, the proposal, and your decisions at design_plan.html beside the project.
5. Build the SoC. This step starts only after your explicit approval. The agent opens a Propel session, queues the IP configuration and connections, previews the design for unconnected ports, validates, and executes. A mandatory pre-compile review then compares what was built against what was approved, and the agent fixes each finding, proposes a fix to you, or records the difference as intentional.
6. Firmware. For designs with a RISC-V processor: decode the system environment, scaffold the project with drivers and linker script, compile, verify the build, and apply the memory-initialization file to the SoC.
7. Radiant. Generate or verify the physical and timing constraints, then run synthesis, map, place and route in the background. Request the build dashboard to monitor progress.
8. Report. The timing validator checks failing endpoints and constraint coverage. Zero coverage is an error, and coverage below 100% is a warning that requires a recorded justification, so a build is complete only with full coverage or a justified exception. Topology and timing must both pass before the build is called complete. The

© 2026 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

report states what passed, what did not, and where the bitstream is located. RTL simulation and Reveal debug are optional follow-on steps that you request.

6.2. Rules the Agent Follows

These rules are supplied to the agent as workflow guidance. Lattice Prompt cannot enforce them: it has no language model and cannot block a tool call, so how consistently they are followed depends on the AI client and model you use.

- No assumptions. The agent must not guess parameter values, IP versions, pin locations, or clock frequencies. If a fact is not in the catalog, a reference design, the IP metadata, or the device constraints, the agent asks you.
- MCP first. The agent uses the server tools for Propel, Radiant, and catalog operations rather than direct shell commands, because the tools handle environment setup, output capture, error classification, and where the tool runs (local or remote). It falls back to the shell only when a tool is unreachable, and reports the fallback to you.
- No manual edits to generated files. If a Propel- or Radiant-generated file needs to change, it is regenerated. It is not manually edited.
- Report divergence. The agent reports anything in the approved plan that the build cannot deliver before execution, rather than silently dropping it. The agent does not introduce virtual I/O without naming it in the plan.
- Record the rationale. The starting point, every source it relied on, and every intentional deviation are written to the rationale store, so the design can be audited after the session.

6.3. Editing an Existing Design

The same flow applies when you start from a .sbx design, a Propel template, or a downloaded reference design. The agent ingests the existing project, parses its instances, nets, and address map, derives the clock and reset topology from it, and proposes only the changes. A reference design on a different family can still serve as a donor, a design whose clock and reset structure is copied into yours.

6.4. Timing Closure

When a build fails timing, the agent does more than rebuild with different settings. It reads the timing and resource reports and proposes one specific fix from a catalog of known fixes delivered with the server: a strategy setting, a constraint change, or an RTL change. When timing fails by only a small margin, the catalog proposes a placement seed sweep, five seeds by default. You approve any fix that changes constraints or RTL.

6.4.1. How a Fix is Tried

The sequence is as follows:

1. The agent reads the existing timing and resource reports.
2. It matches the failure against the catalog, proposes candidate fixes, and checks the chosen fix against the design.
3. It applies the fix to a separate copy of the project, never to the original. A fix that changes constraints or RTL needs your approval first.
4. The copy is built in the background, as any build is.
5. The agent reads the new timing result and confirms that nothing else got worse than in the original build.

The analysis steps only read reports. The only Radiant run is the build of the project copy, and the original design is never modified. The same tools also offer RTL and constraint lint and checks of what the toolchain supports, which the agent uses before a build when they apply. If the timing-closure tools are not available in a session, the agent falls back to the critical-path report and the timing-closure knowledge resource.

A second path exists for designs that already have a Radiant design saved after an earlier map step: the agent can open that saved design in a persistent Radiant session, adjust map, placement, and routing options, and iterate place and route without a full rebuild. The agent is guided to pick one path per build attempt and not to mix the two.

© 2026 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

6.4.2. Radiant Session Configuration

The persistent Radiant session used by the timing-closure path and the saved-database path resolves the Radiant executable in this order: the RADIANT_EXE environment variable, the active context block in its configuration file, then a built-in default. License variables come from the environment first and the configuration file second. The configuration file is searched at the path in RADIANT_MCP_CONFIG, then radiant-mcp.json in the working directory, then the user configuration directory. Two contexts are kept, local and remote, and the active one follows the remote-execution mode: local or ssh.

To configure the session, enter the following request:

```
Find the Radiant installation on this machine and save its path and license settings for local use.
```

Then confirm that the Radiant session is ready. The agent reports the Radiant installation it found and anything that is missing.

6.5. Hardware Debug Flow

For an existing Radiant project that does not build, or a bitstream that programs but does not behave as expected, the agent classifies the failure from the build artifacts, parses the build log, looks up each error identifier, and builds a design digest with references. It then works through the fault one test at a time, each test chosen to rule out a cause, and records every result in a case file beside the project so the investigation can resume later. For board-level faults, the Reveal tools propose probes, generate the Reveal setup, and compare the captured signals with a simulation waveform.

© 2026 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

7. Tutorials

The prompts below are taken from real sessions. Enter them in your AI client with the servers installed. Each prompt states what the design must do, such as the board, the interfaces, and the data rates. It does not name IP cores. The agent selects the IP from the catalog as part of the design work. The outcomes described were obtained in Lattice test sessions. Your results depend on the model, the installed catalog, and the toolchain version, and the generated design still needs your verification. One prompt names the GSRD, the golden system reference design that Lattice publishes for a board.

7.1. Building a New SoC

```
I want to build an SoC to demo the LPDDR4 read and write access sequence with a CPU.
Show the transaction on UART: what is written, what is read back, and whether they match.
Target the CertusPro-NX Versa board.
```

The agent resolves the board to the LFCPNX-100-9LFG672I device, selects the RISC-V processor, system memory, LPDDR4 memory controller, UART, interconnects, and bridges, proposes a two-output PLL with one reset synchronizer per domain and an address map, waits for approval, builds the SoC, generates and compiles firmware that walks test patterns over the memory, pins the Versa board signals, runs Radiant, and reports whether the checks passed.

7.2. Architecting a Larger System

```
I would like to create a solution that interfaces 12 camera sensors to NVIDIA Holoscan.
Use a Lattice FPGA. Use a RISC-V processor to configure the camera sensors.
Video format: 1080p raw12 60 fps. Ethernet: 25 Gbps.
```

No device is named, so the agent first derives the aggregate bandwidth, then chooses a Lattice Avant™-X part, then splits the design into a Propel control-plane SoC for sensor configuration and a Radiant RTL data plane for the video path, and proposes both before building either.

7.3. Modifying a Reference Design

```
Add 3 UARTs to the CertusPro-NX GSRD and rebuild the firmware to drive them.
Then launch it in Radiant and run all the way to bitstream.
Use Propel and Radiant 2025.2. Target the CertusPro-NX Versa board (LFCPNX-100-9LFG672I).
Do not edit the source GSRD. Work on a workspace copy.
```

The GSRD is the golden system reference design that Lattice publishes for the board. Because the request names it, the agent resolves it from the catalog, downloads it if it is not present, and works on a copy.

7.4. Compiling an Existing Radiant Project

```
Simulate and compile the Radiant project C:/path/to/project1/proj.rdf using Radiant 2025.2 on this
machine.
Simulate and compile the Radiant project in this archive: C:/my_designs/project1.zip
using Radiant 2025.2 on my remote Linux machine:
Radiant installation path = /tools/lscc/radiant/2025.2
LM_LICENSE_FILE = <port>@<license server>
SALT_LICENSE_SERVER = <port>@<license server>
```

The remote configuration is stored after the first session and does not need to be repeated.

7.5. Shorter Requests

Table 7.1. Shorter Requests

Goal	Prompt
Check whether an IP fits a device	Does the MIPI D-PHY receiver IP support CrossLink™-NX LIFCL-40? Which versions are installed?

© 2026 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

Goal	Prompt
Find a starting point	Which Lattice reference designs cover PCIe on CertusPro-NX, and which board do they target?
Watch a long build	Start the build dashboard and show me the running jobs.
Simulate	Set up a testbench for the SoC in C:/work/my_soc and run an RTL simulation that checks a UART transfer end to end.
Close timing	The Radiant build fails timing on the 200 MHz domain. Show the critical paths and propose strategy changes.
Debug on hardware	The board boots but the UART prints nothing. Set up Reveal probes on the UART transmit path.
Answer a Radiant question	How do I constrain a generated clock in Radiant?
Report a problem to Lattice	Create a Lattice support log for the build failure we just saw.

© 2026 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

8. Recommended Practices

8.1. Writing the Request

- Describe the result you want, not how to build it. Name the board, the interfaces, the data rate or video format, and the processor if you need one. Leave IP selection to the agent. Naming an IP that you are not certain of constrains the design around a guess.
- Provide the key constraint. *1080p raw12 at 60 fps* and *25 Gbps* narrow the device choices compared to a general phrase such as *high-resolution video*.
- Name the board, not only the family. A board resolves to a part number, a pinout, and a set of constraints. A family alone gives the agent none of those.
- Name the toolchain version when more than one is installed, for example, *use Propel and Radiant 2025.2*.
- State the project location and, when starting from a reference design, work on a copy of it in your project folder (a workspace copy), so the source is never edited.

8.2. During the Session

- Read the proposal before approving it. The IP list, clock plan, reset plan, and address map are stated before anything is created. This is the point to clarify requirements, refine the design, or change the architecture, before anything is generated.
- Expect questions. A question means two readings of your prompt would produce materially different hardware. Answer the question. Do not tell the agent to continue.
- Accept the design plan. It has no cost and gives a reviewer the prompt, the proposal, your decisions, and the gates that ran, after the session has ended.
- Ask for the build dashboard before a long build. It survives the session, so a place-and-route run left overnight can be checked later.
- Keep one session per project. Context from a previous design does not carry over cleanly, and mixing two designs in one conversation makes the rationale store ambiguous.

8.3. Reading the Results

When the agent reports a build, phase 8 of the design flow, check the following before you accept the result.

- A complete build has passed both gates. If the agent reports a build without stating that topology and timing validation passed, request both validations.
- Incomplete timing coverage is a warning, not a pass. Below full constraint coverage, either add the missing constraints or have the justification recorded in the design plan.
- Read the warnings list in the report, not only the success line. For Radiant tools, the success flag detects only known error patterns and can miss a new one. The warnings list captures every line that contains an error or failure keyword.
- Treat advisory notes as guidance. Advisory notes from the validators are heuristics. Override them when you have a reason, and record the reason.

8.4. Environment

- Use short project paths on Windows. Keep the resolved path well under 260 characters.
- Set license variables before starting the AI client. Both the Lattice license and the Questa license variables must be set in the environment the AI client inherits.
- One Radiant instance per license. Builds that run in the background each launch their own Radiant process. Confirm that your license allows the concurrency you request.
- Restart the AI client after installing or upgrading. The servers are loaded at start-up.

© 2026 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

9. Troubleshooting and Support

9.1. Creating a Lattice Support Log

When a problem occurs, ask the agent for a support log:

```
Create a Lattice support log for this issue.
```

Similar phrasing is accepted, for example, *log this issue*, *report this bug*, or *what do I send to Lattice*. The agent then:

1. Collects the machine facts: the session transcript if the AI client kept one, job records and build logs, the Propel session replay, the rationale store, the design plan, the Lattice Prompt server log, tool versions, and platform.
2. States an evidence grade. Grade A means the transcript was read and quotes are verbatim. Grade B means the order of events is reliable but the exact inputs of each tool call are paraphrased rather than quoted. Grade C means part of the session is no longer visible and the report says which part.
3. Compares the conversation history with the files on disk and reports any disagreement rather than resolving it silently.
4. Lists each issue separately with at least two hypotheses, the evidence for and against each, and a conclusion.
5. Attributes the primary cause to one of the following: the Lattice Prompt server, the workflow guidance, the Lattice toolchain, the conduct of the agent, your input, or the environment, with a confidence level.
6. Decides what happens next: whether a support case is required and against which component, whether a workaround exists and what trade-off it carries, and what you and the agent each do next.

The result is one zip beside your design containing report.html, report.json, a readme, and copies of every attachment that support would otherwise request. Nothing is transmitted. You open it, review the redaction statement, and attach it to your support case.

9.2. Support Log Contents and Redaction

Table 9.1. Support Log Contents and Redaction

Kept	Removed
Device part numbers, IP names and versions, tool versions, error codes, timings, job identifiers	Usernames, hostnames, license server addresses, environment variables
Project and design names, listed in the redaction statement so you can decide	—

Review the bundle before you attach it to a support case. Project and design names are kept so that the report remains useful. The redaction statement lists them so that you can decide whether they may leave your organization.

9.3. Diagnostic Log and Health Check

- Diagnostic log. Always on. The server writes its own log file under the cache directory listed under Installed Locations, keeps it under about 20 MB, and never sends it anywhere. You do not need to read it: the support log collects it automatically. LATTICE_CACHE_DIR moves the directory.
- Health resource. Ask the agent to read lattice://health to see the Lattice Prompt server version, the discovered Propel and Radiant paths, and where tools run (local or remote).
- Job records. Every background job keeps its command line, exit code, start and end time, and log path. Ask for the job history or open the build dashboard.

© 2026 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

9.4. Common Problems

Table 9.2. Common Problems

Symptom	Likely cause	What to do
No Lattice tools after restart	Registration missing or AI client not fully restarted	Quit and reopen the AI client. Check the configuration file named under Supported Toolchain and AI Client.
Permission prompt on every tool call	Allow list not set	Add the server tools to the AI client allow list as described under Tool Permissions.
IP catalog empty	Propel not found	Read lattice://health to see what was discovered, then set LATTICE_PROPEL and LATTICE_PROPEL_BASE to the Propel path and re-run the installer.
propelbldc or pnmainc not found	Toolchain not installed or in an unusual location	Install Propel and Radiant, or set LATTICE_PROPEL and LATTICE_RADIANT, and the same paths in LATTICE_PROPEL_BASE and LATTICE_RADIANT_BASE.
License checkout failure at build or simulation	LM_LICENSE_FILE or SALT_LICENSE_SERVER not set in the AI client environment	Set both as user environment variables and restart the AI client.
Generate step produces a stub with no metadata, or a compiler reports an existing file as missing	Windows path length over about 260 characters	Shorten the project path or enable long-path support. Do not attempt a code-level workaround.
Memory-initialization synthesis error (INIT_VALUE_*)	Several possible causes	Run the generated-source check and the memory-initialization check. Do not hand-edit the RTL.
Agent stops polling a long job	Lost track of job status	Ask for the job status by identifier, or open the build dashboard.
Installer stops on a certificate error	Network inspects HTTPS	Trust the corporate root certificate in the operating-system store. The installer never skips certificate verification.
Installer reports a newer version already installed	Incorrect release zip	Use the newer bundle, or pass --only lattice-prompt --force-reinstall if a downgrade of the lattice-prompt server is intended.

© 2026 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

10. Known Limitations

Table 10.1. Known Limitations

Area	Limitation
Toolchain versions	Requires Propel 2025.2 or later and Radiant 2025.2 or later. Earlier versions are not supported.
Enforcement	The servers cannot block a tool call. Gates and advisories are followed by the agent, not enforced by the servers. A different AI client or model may follow them less consistently.
Catalog coverage	The reference-design and IP user-guide catalogs supplied with the servers cover a subset of the Lattice portfolio. A design the catalog does not cover is proposed from available IP and precedent designs.
Simulation	RTL simulation uses the Questa simulator bundled with Radiant and runs locally only. Simulation applies to Radiant .rdf projects.
Programming and Reveal	Programming the board from Lattice Prompt is not supported in this release and is planned as a future enhancement. Program the bitstream with Radiant Programmer directly. Reveal capture is not automated: the tools prepare and validate the Reveal setup, and the capture itself runs in the Radiant Reveal Analyzer.
AI provider	The quality and privacy of a session depend on the model and AI client you use. Lattice Prompt does not control what the AI client sends to the AI provider.
Concurrent Radiant instances	Each background build and each persistent Radiant session launches its own Radiant instance. The license must allow the concurrency.
Design correctness	A generated design is not guaranteed to be correct, to meet timing, or to work on hardware. Lattice Prompt contains no language model and makes no design decisions. Functional simulation, timing sign-off, and hardware validation remain your responsibility.

© 2026 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

References

- Lattice Radiant Timing Constraints Methodology (FPGA-AN-02059)
- Model Context Protocol Specification
- Lattice Prompt web page
- Lattice Propel Design Environment web page
- Lattice Radiant Software web page
- Lattice Solutions IP Cores web page
- Lattice Solutions Reference Designs web page
- Lattice Insights for Lattice Semiconductor training courses and learning plans

© 2026 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

Technical Support Assistance

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

For frequently asked questions, refer to the Lattice Answer Database at www.latticesemi.com/Support/AnswerDatabase.

Note: When you submit a case for the Lattice Prompt, attach the support log ZIP described in Creating a Lattice Support Log. Also include the version and build time recorded in the *BUILDINFO* file located beside the installer, along with the Propel and Radiant versions from the health resource.

© 2026 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

Revision History

Revision 1.0, October 2026

Section	Change Summary
All	Initial release.

© 2026 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal.
All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.



www.latticesemi.com