Lattice IP Packager 2026.1

User Guide

FPGA-UG-02253-1.0

June 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

© 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.

Figures

© 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.

Tables

© 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 or terms specialized in this document.

Abbreviation	Definition
CPU	Central Processing Unit
CSV	Comma Separated Values file
DRC	Design Rule Check
ESI	Previous name of Propel
FPGA	Field Programmable Gate Array
GUI	Graphical User Interface
HDL	Hardware Description Language
IDE	Integrated Development Environment
IP	Intellectual Property
IP-XACT	An XML format that defines and describes electronic components and their designs.
LSE	Lattice Synthesis Engine
MPAR	Multiple Place and Route
RTL	Register Transfer Level
SoC	System on Chip. An integrated circuit that integrates all components of a computer or other electronic systems.
TCL	Tool Command Language
UDB	Lattice Radiant software Unified Database file
VM	Synthesis tool output file

© 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

An IP package is a collection of all required files related to an IP core. These related files of an IP are organized in different directories. The graphical tool Lattice IP Packager 2026.1 is used to create an IP package for the Lattice Radiant™ software and the Lattice Propel™ Builder software.

1.1. Purpose

This document introduces Lattice IP Packager 2026.1 and provides a hands-on guide for you to start creating a package for a customized IP module. A variety of IP modules that can be customized are available for you in the Lattice Radiant software and Lattice Propel Builder. These modules cover a variety of common functions and can support your design work.

1.2. Audience

The intended audience for this document includes embedded system designers and embedded software developers using Lattice devices. The technical guidelines assume readers have expertise in the embedded system area and FPGA technologies.

© 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. IP Packager

Lattice IP Packager helps you create an IP package easily. You can edit port, file, parameter, and memory in the IP Packager, and pack a customized IP directly. Figure 2.1 shows an example of directories and files of an IP package.

```
- [IP Package]
   - metadata.xml
   - [rtl]
   - [testbench]
   - [ldc]
      - constraint.ldc
   - [driver]
    - [eval]
   - [plugin]
      - plugin.py
   - [doc]
      - introduction.html
      - EULA.txt
```

Figure 2.1. Example Directories and Files of an IP Package

- metadata.xml [mandatory]: XML metadata file which mainly describes legal usage and interface of an IP.
- rtl [mandatory]: Directory for parameterized HDL source files. HDL source files contain configurable parameters for you to configure.
- testbench [optional]: Directory for test bench files.
- ldc [optional]: Directory for template constraint file. The file name should be constraint.ldc.
- driver [optional]: Directory for driver source code files.
- plugin [optional]: Directory for Python script to implement internal logic of the soft IP. The file name of the Python script should be plugin.py.
- doc[mandatory]: Directory for documentation files. It should contain one mandatory introduction file, one mandatory license agreement file, and other optional documents.

The custom IP package can be used in Lattice Propel Builder for SoC design. An IP instance package (Figure 2.2) is generated when you configure an IP in Lattice Propel Builder.

```
- <instance_name>
   - <instance_name>.cfg
   - <instance_name>.ipx
   - component.xml
   - design.xml
   - [rtl]
      - <instance_name>.v
      - <instance_name>_bb.v
   - [constraints]
      - <instance_name>.ldc
   - [driver]
   - [eval]
   - [misc]
      - <instance_name>_tmpl.v
      - <instance_name>_tmpl.vhd
```

Figure 2.2. Example Directories and Files of an IP Instance Package

© 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.

2.1. Launching IP Packager

The Lattice IP Packager tool can be launched in Lattice Propel Builder and the Lattice Radiant software. The tool can be run in both the Windows and Linux operating systems.

To launch IP Packager from Lattice Propel Builder, you can either use the executables listed below or use the Lattice Propel Builder GUI.

To launch IP Packager from Lattice Propel Builder using executables:

- Windows GUI mode
  a. Open the **Run** dialog box by pressing the Windows key + R.
  b. In the **Run** dialog box, type the following directory and press **Enter**.
  `<InstallationDirectory>/propel/2026.1/builder/rtf/bin/nt64/ippack.exe`
  c. The IP Packager tool opens and shows its GUI.
- Windows console mode
  a. Open the **Run** dialog box by pressing the Windows key + R.
  b. In the **Run** dialog box, type the following directory and press **Enter**.
  `<Installation Directory>/propel/2026.1/builder/rtf/bin/nt64/ippackc.exe`
  c. The IP Packager tool opens as a console window.
- Linux GUI mode
  a. Open a terminal.
  b. Type the following command and navigate to where the executable is located.
  `cd <Installation Directory>/Propel/2026.1/builder/rtf/bin/lin64/`
  c. Type `./ippack` on a command line and press **Enter**.
  d. The IP Packager tool opens and shows its GUI.
- Linux console mode
  a. Open a terminal.
  b. Type the following command and navigate to where the executable is located.
  `cd <Installation Directory>/propel/2026.1/builder/rtf/bin/lin64/`
  c. Type `./ippackc` on a command line and press **Enter**.
  d. The IP Packager tool opens as a console window.

To launch IP Packager from Lattice Propel Builder using the software GUI:

1. Choose **Tools** > **IP Packager** from the Lattice Propel Builder Menu Bar, as shown in Figure 2.3.

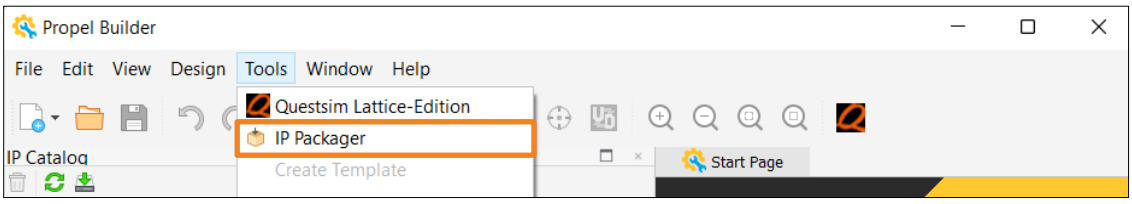

Figure 2.3. Launch IP Packager from Lattice Propel Builder

2. The IP Packager window pops up, as shown in Figure 2.4.

© 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.

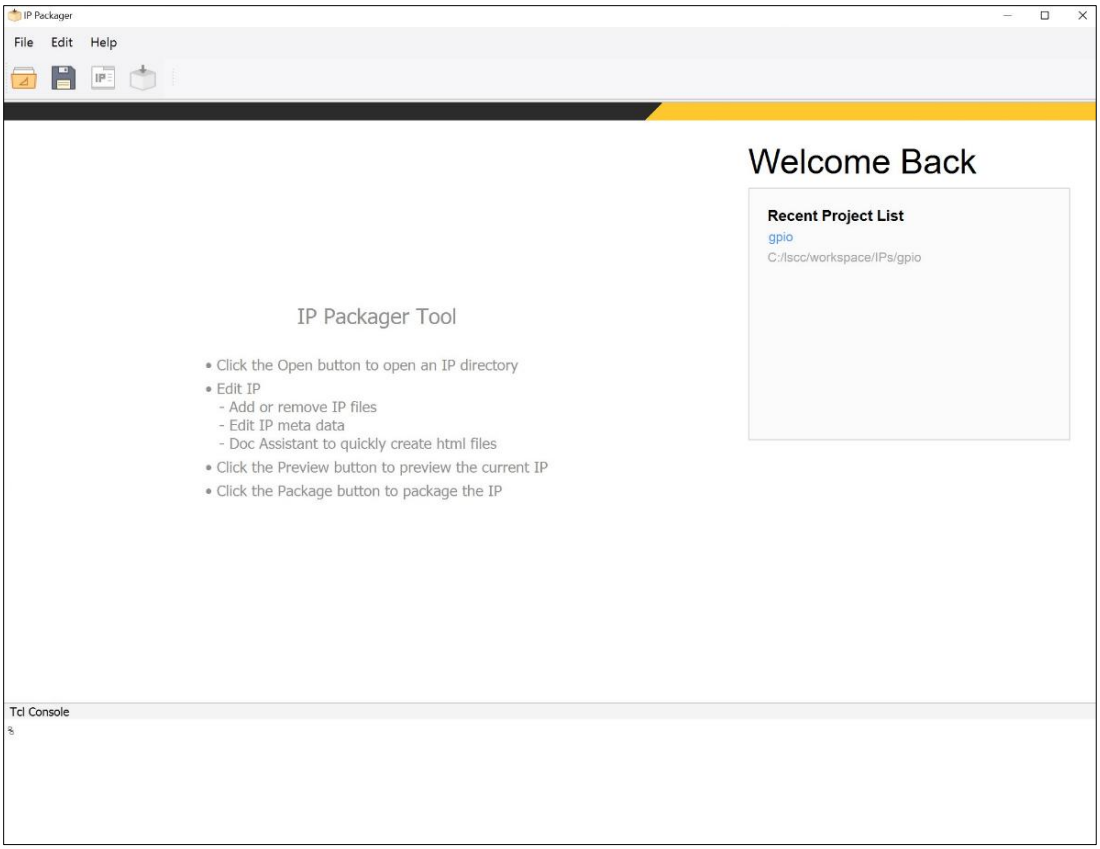

Figure 2.4. IP Packager GUI

To launch IP Packager from the Lattice Radiant software, you can either use the executables listed below or choose IP Packager from the Windows **Start** menu.

To launch IP Packager from the Lattice Radiant software using executables:

- Windows GUI mode
  a. Open the **Run** dialog box by pressing the Windows key + R.
  b. In the Run dialog box, type the following directory and press **Enter**.
  `<Installation Directory>/radiant/2026.1/bin/nt64/ippack.exe`
  c. The IP Packager tool opens and shows its GUI.
- Windows console mode
  a. Open the **Run** dialog box by pressing the Windows key + R.
  b. In the Run dialog box, type the following directory and press **Enter**.
  `<Installation Directory>/radiant/2026.1/bin/nt64/ippackc.exe`
  c. The IP Packager tool opens as a console window.
- Linux GUI mode
  a. Open a terminal.
  b. Type the following command and navigate to where the executable is located.
  `cd <Installation Directory>/radiant/2026.1/bin/lin64/`
  c. Type `./ippack` on a command line and press **Enter**.
  d. The IP Packager tool opens and shows its GUI.

© 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.

- Linux console mode
  a. Open a terminal.
  b. Type the following command and navigate to where the executable is located.
     `<Installation Directory>/radiant/2026.1/bin/lin64/`
  c. Type `./ippackc` on a command line and press **Enter**.
  d. The IP Packager tool opens as a console window.

To launch IP Packager from the Windows Start menu:

1. Go to the Windows **Start** menu and choose **Programs** > **Lattice Radiant Software** > **Accessories** > **IP Packager**.
2. The IP Packager window pops out (Figure 2.4).

2.2. Packing Custom IP Flow

2.2.1. Opening an IP Directory

1. In Lattice IP Packager, choose **File** > **Open IP Directory** from the Menu or click the Open Design icon from the Toolbar. The **Select Folder** dialog opens (Figure 2.5).

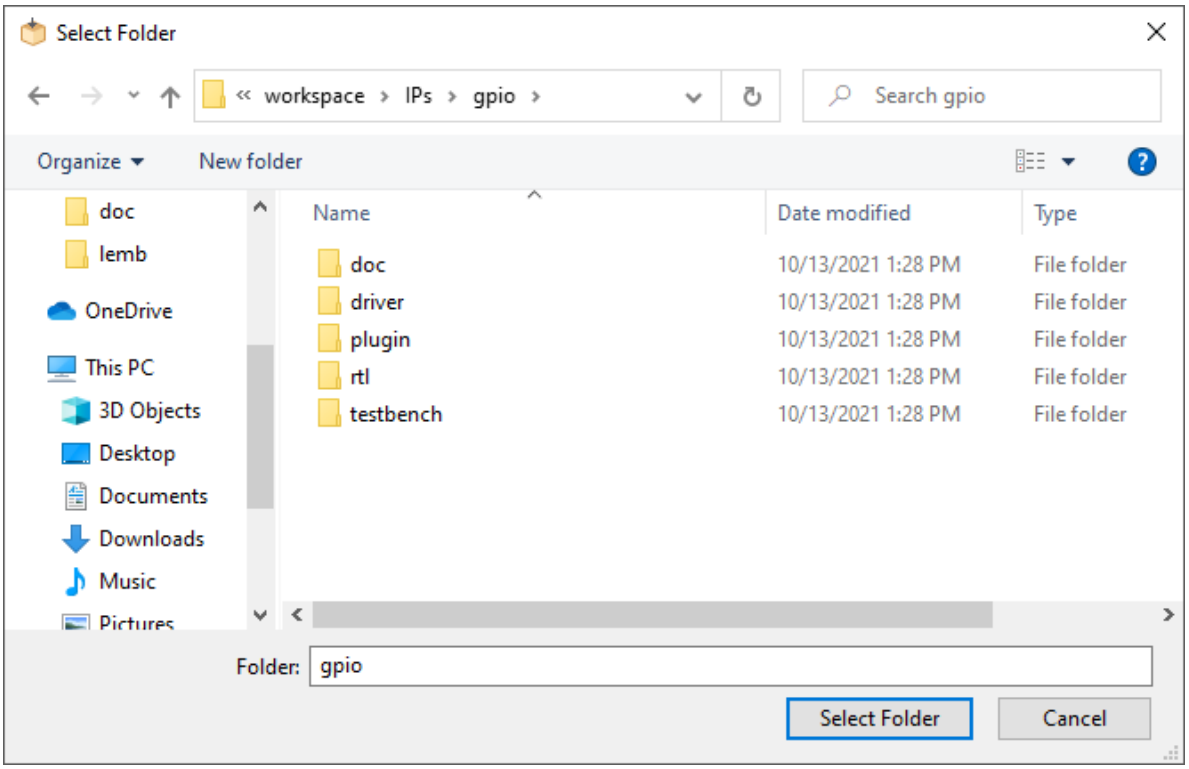

Figure 2.5. Select Folder Dialog

2. Choose a desired IP directory.
3. Click **Select Folder**. The IP Packager GUI shows the IP Package project (Figure 2.6). An IP package project may include the following parts.

   The three parts listed below are mandatory:

   - Meta Data
   - Design File
   - Doc

© 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.

The three parts listed below are optional:

- Test Bench
- Constraint
- Misc

These mandatory and optional parts need to be edited. Refer to the Editing IP section for more details.

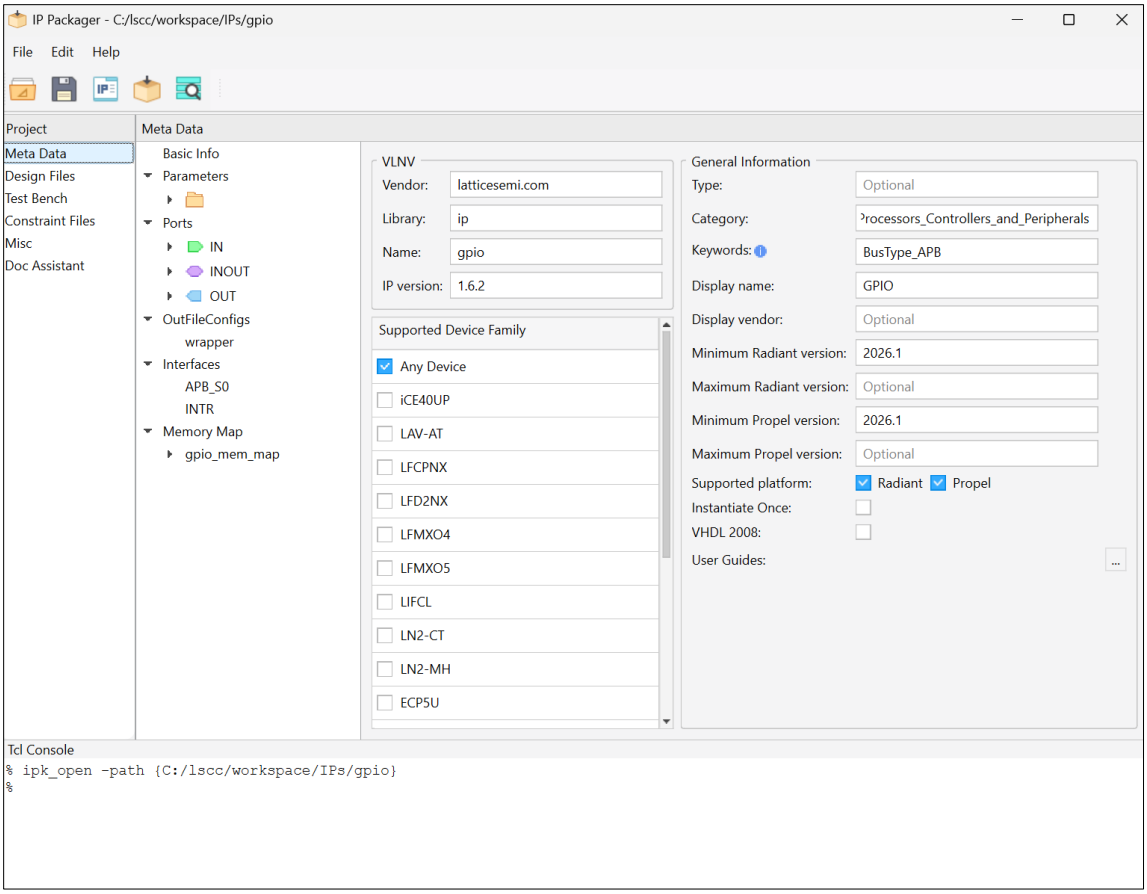

Figure 2.6. IP Packager with IP Project Details

Note: If you prepare all directories and files of an IP module in advance, the IP Packager can load the information. You can use IP Packager to open an empty folder for creating a new IP and add your own RTL files into this empty IP. If you do not want to update the IP module information, you can skip the Editing IP section below.

© 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.

2.2.2. Editing IP

1. From the IP Packager Project area, click **Meta Data**. The IP Packager GUI shows the Meta Data information in detail (Figure 2.7).

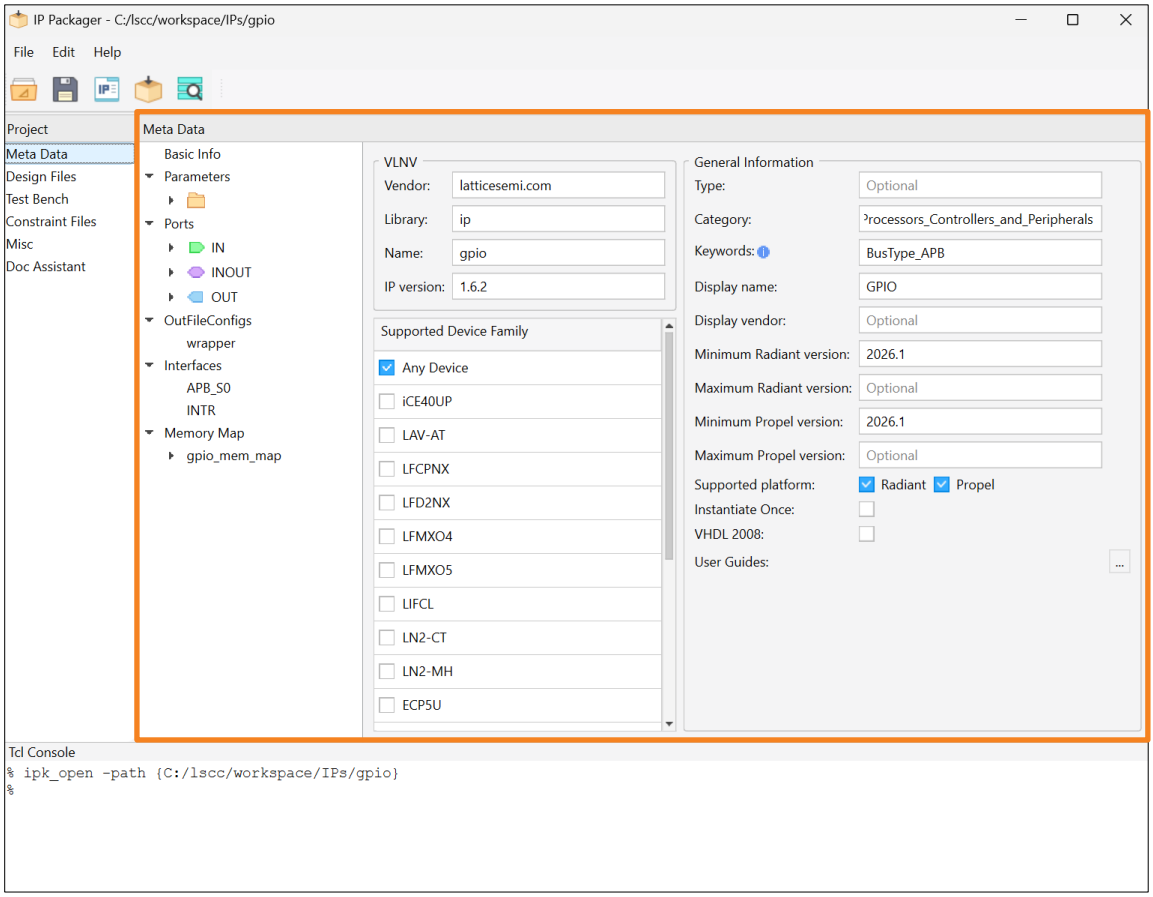

Figure 2.7. IP Packager with Meta Data Details

a. To configure basic information:
   - Click **Basic Info** from the **Meta Data** view (Figure 2.8).

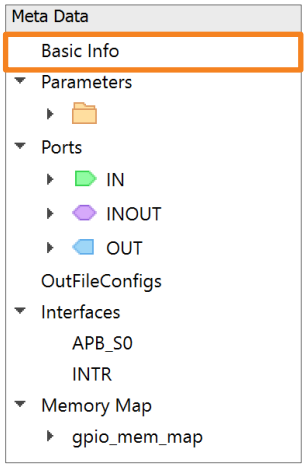

Figure 2.8. Meta Data View

© 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 can see the Basic Info properties in detail, as shown in Figure 2.9.

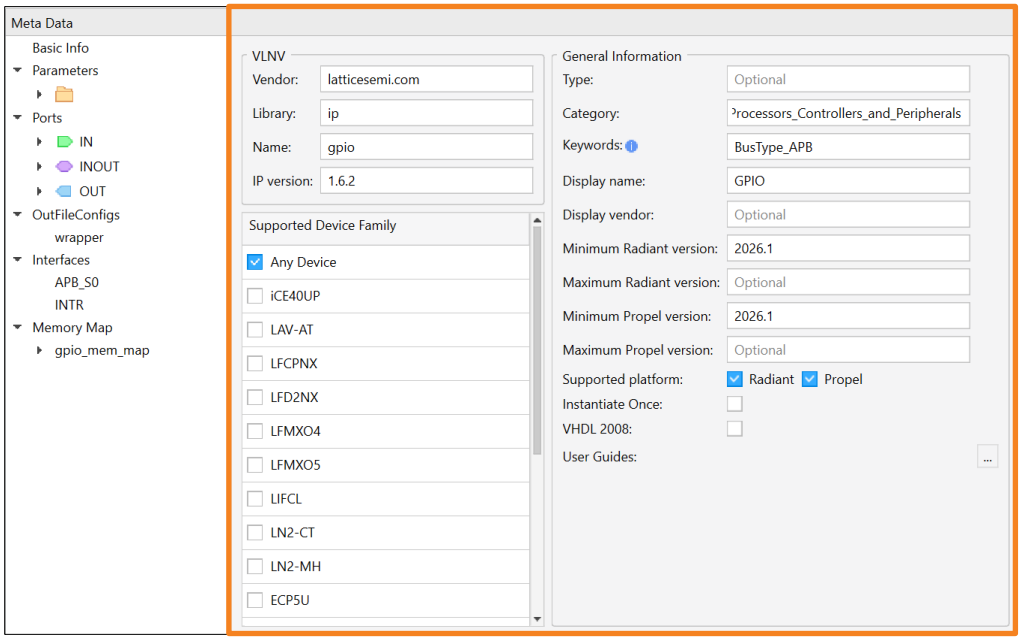

Figure 2.9. Configure Basic Info

- Configure the basic information by entering a desired value for a property or checking the checkbox for the property in the property field. For example, click the **Vendor** field to enter a desired vendor value. **Note**: IP Packager supports both the Lattice Radiant software and Lattice Propel Builder. IP Packager currently is based on the Lattice Radiant software 2026.1. It does not support the Lattice Radiant software version that is lower than 2026.1. Therefore, **Minimum Radiant version** should be set to 2026.1 or 2026.1 +. For Lattice Propel Builder, IP Packager currently is based on Lattice Propel Builder 2026.1. It does not support the Lattice Propel Builder version that is lower than 2026.1. So, **Minimum Propel version** should be set to 2026.1 or 2026.1+. If the version of the Lattice Radiant software or Lattice Propel Builder is invalid, a warning message is shown (Figure 2.10).

© 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.

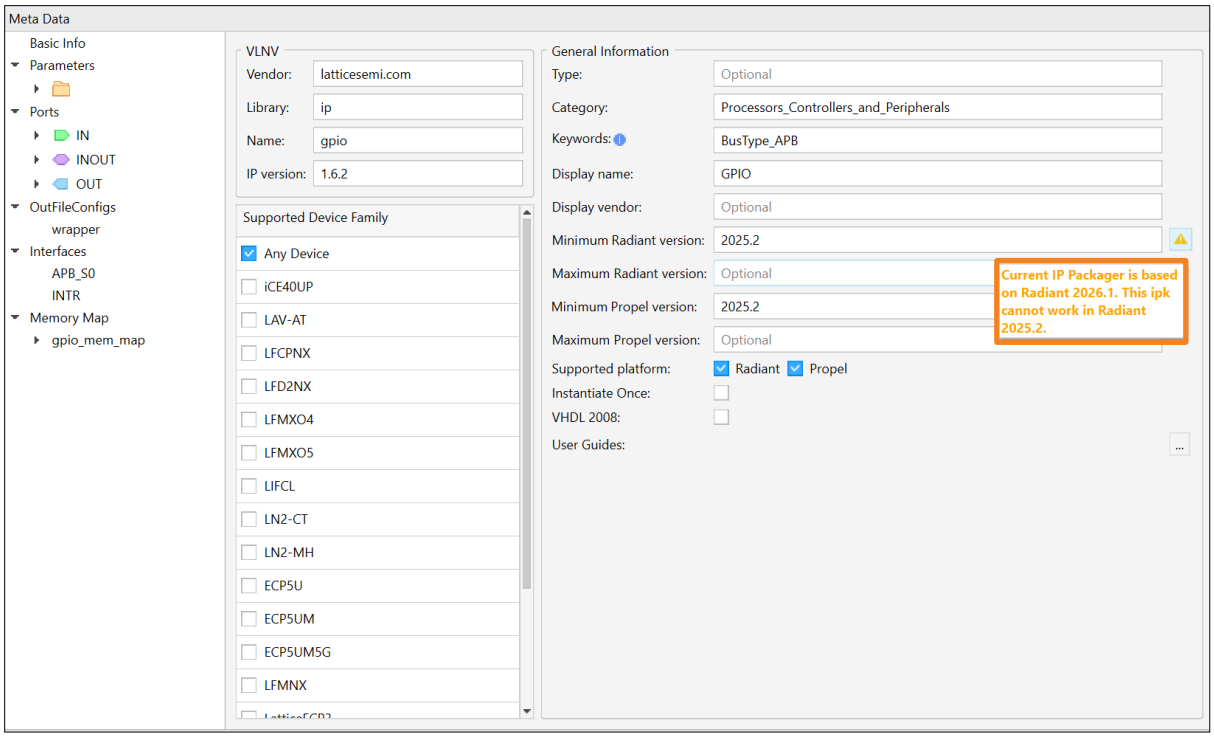

Figure 2.10. Warning Message of Invalid Radiant and Propel Version

b. To configure parameters:
   - Parameters are settings that can be used in Ports/Interfaces/Memory map configuration. They function similarly to macros. For example, if you click on the port gpio_i, you can see its configuration. In gpio_i, there is a value using EXTERNAL_BUF that is defined in Parameters (Figure 2.11).

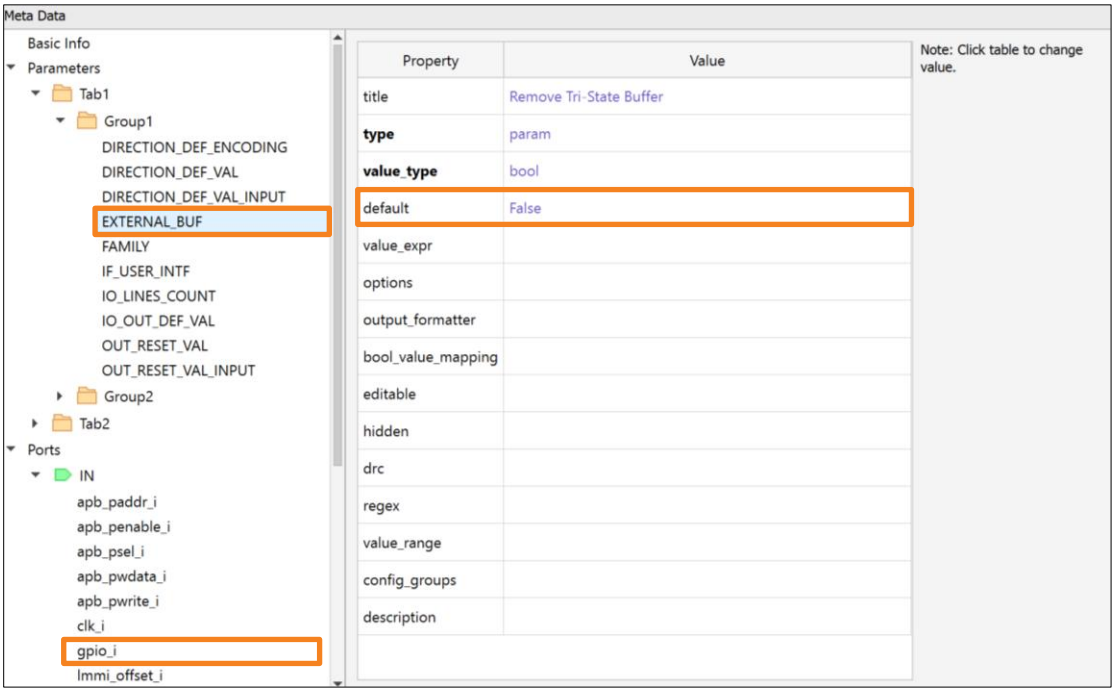

Figure 2.11. Preview on Parameters

© 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.

- Two-level hierarchy (Tab/Group) is used to categorize the parameters. Click the **IP Preview** icon from the toolbar (Figure 2.12) and you can see the hierarchy relations (Figure 2.13).

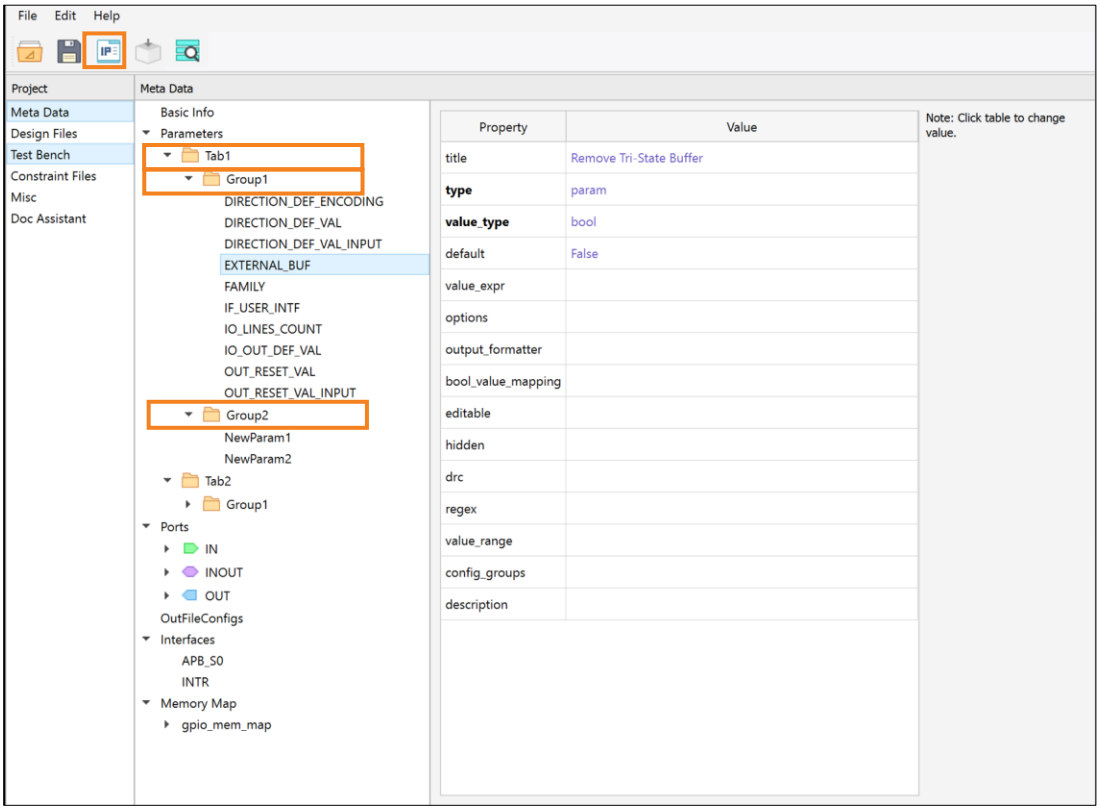

Figure 2.12. Two-Level Hierarchy to Categorize Parameters

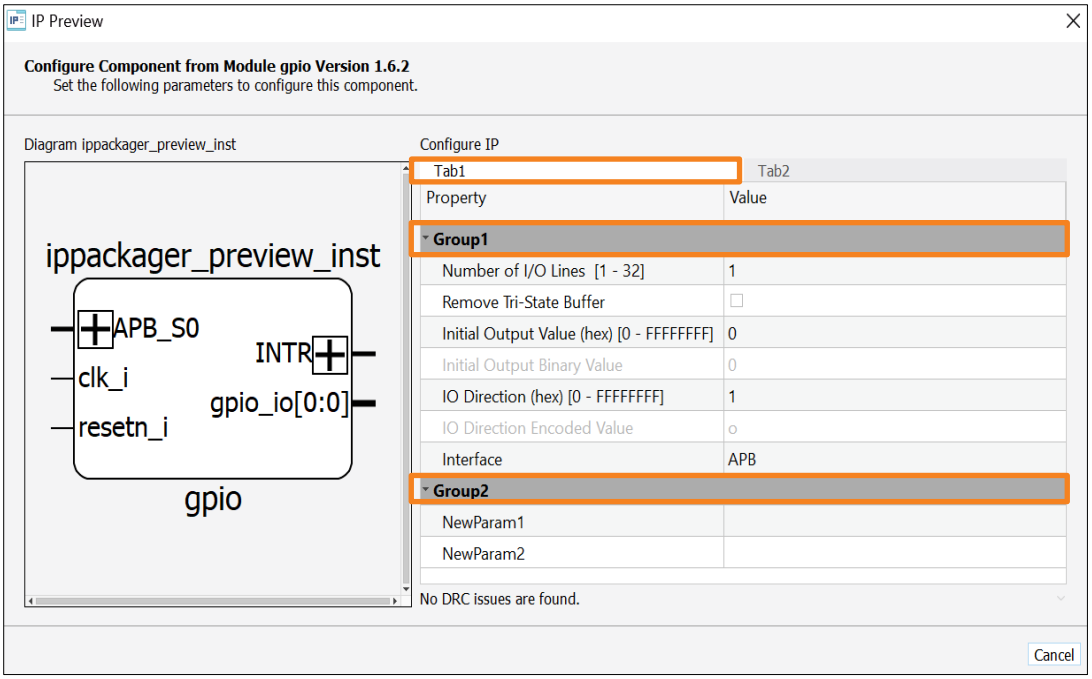

Figure 2.13. IP Preview Showing Two-level Hierarchy in Parameters

© 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.

- Naming rules for parameters:
  - Parameters in all groups should not have the same name.
  - Characters supported in naming are letters with mixed case, numbers, and underscore (_).
- To add a parameter, right-click Parameters from the **Meta Data** view and choose **Add Parameter Tab** (Figure 2.14). **Tab1** is created under **Parameters**.

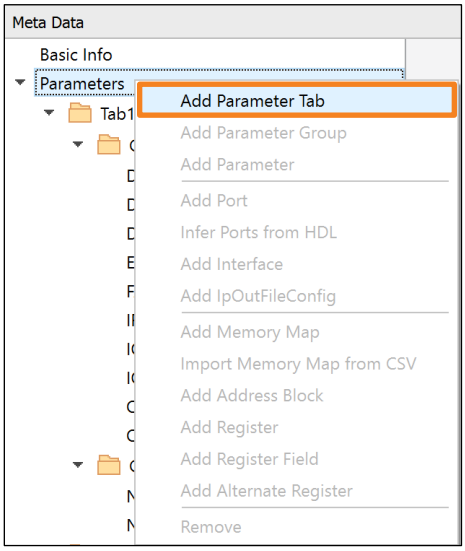

Figure 2.14. Right-click Menu of Parameters in Meta Data View

- Right-click Tab1 under **Parameters** from the **Meta Data** view, and choose **Add Parameter Group**. Group1 is newly added under Tab1 (Figure 2.15).

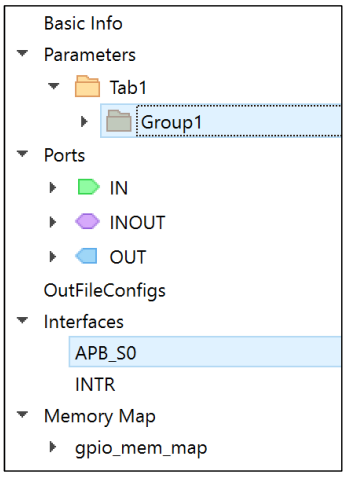

Figure 2.15. Group1 Added to Tab1

- Right-click Group1 under Tab1 from the **Meta Data** view, and choose **Add Parameter**. NewParam1 is newly added under Group1 (Figure 2.16).

© 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.

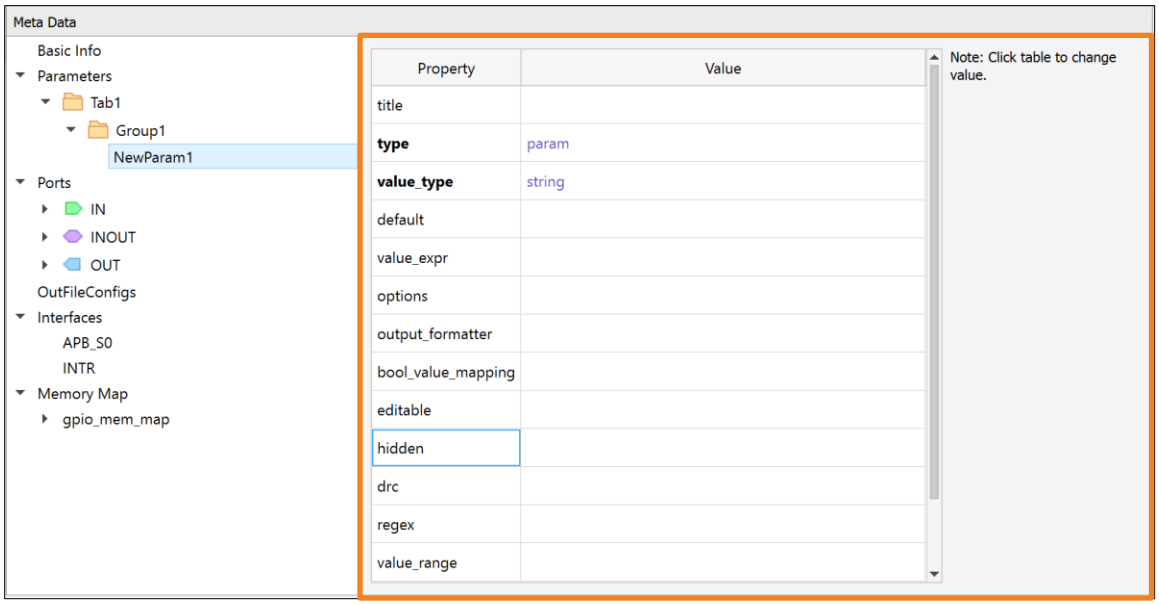

Figure 2.16. NewParam1 Added to Group1

- Double-click NewParam1 to rename the parameter (Figure 2.17). The parameter name is also used as the unique ID of the parameter setting.

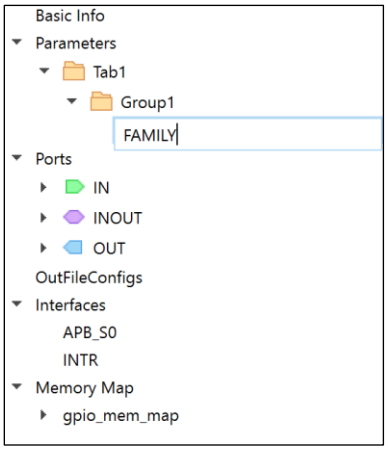

Figure 2.17. Rename a Parameter

- Double-click the property value field to enter the desired parameter value (Figure 2.18), or select the desired value from the drop-down menu (Figure 2.19). The properties are thus configured. Properties listed in bold must be configured. The details of each property are shown in Table 2.1.

© 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.

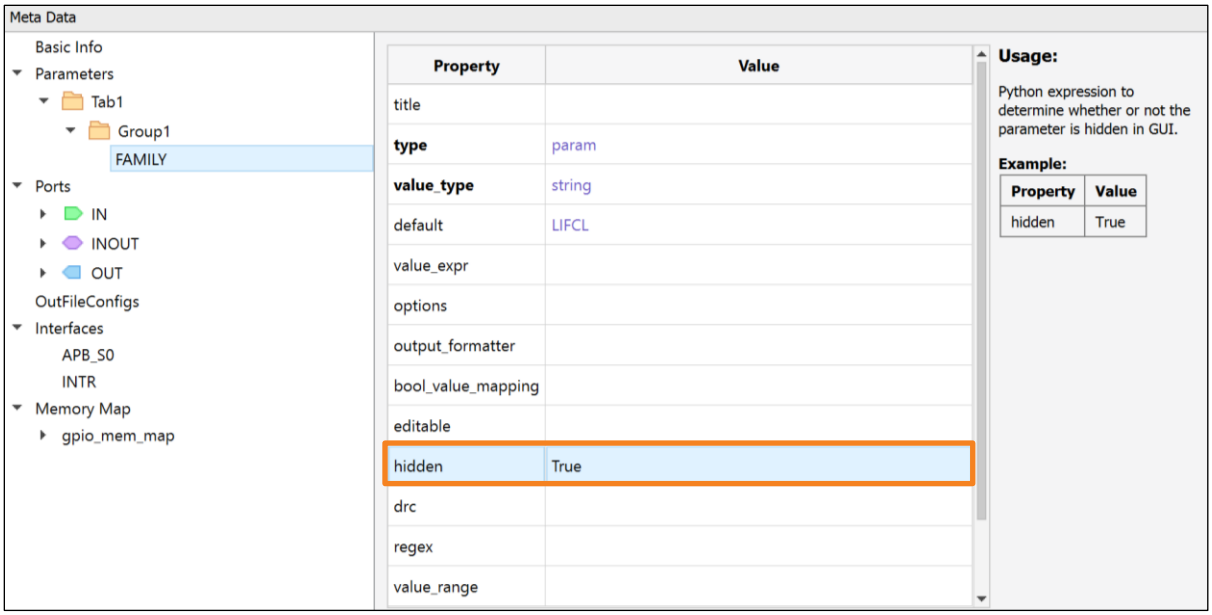

Figure 2.18. Enter Desired Parameter Value

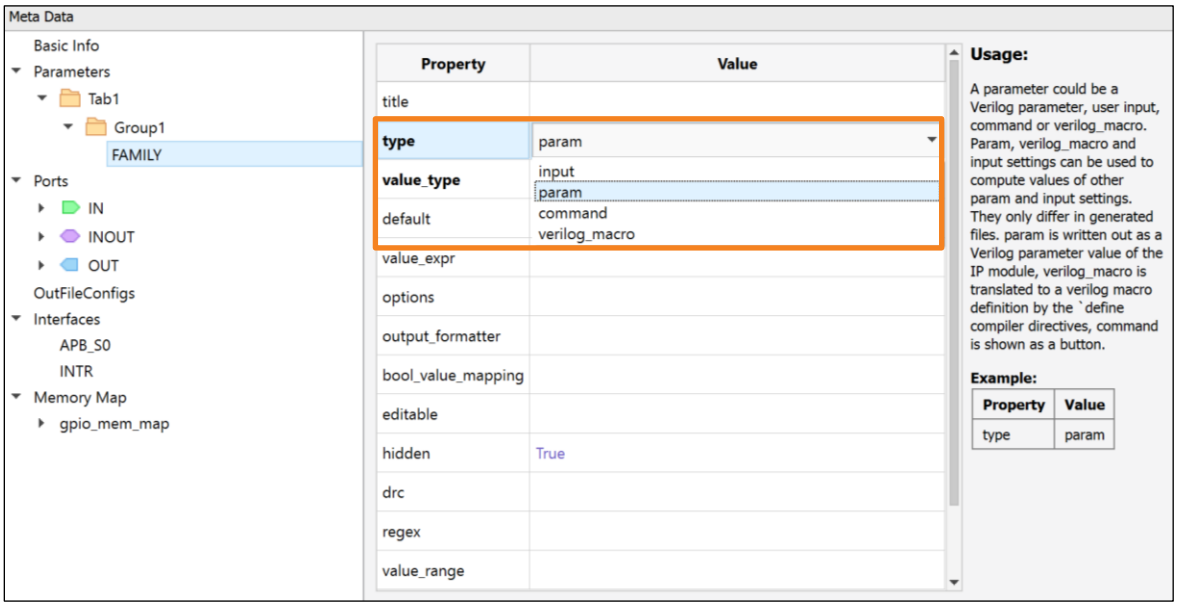

Figure 2.19. Select Desired Parameter Value from Drop-down Menu

© 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.

Table 2.1. Details of Parameter Property

Property	Value	Mandatory	Description
title	String	No	Short title of the setting. If title is not specified, name of the parameter is used. Example: title=”Device Architecture”
type	param, input, command, verilog_macro	Yes	Type of the setting. A parameter could be a Verilog parameter, user input, command, or verilog_macro. Param, input, and verilog_macro settings can be used to compute values of other param and input settings. They only differ in generated files. Param is written out as a Verilog parameter value of the IP module. verilog_macro is translated to a Verilog macro definition by the define compiler directives, while command is shown as a button. Example: type=”param”
value_type	bool, string, int, float, path	Yes	Type of the value. Supported types are bool, int, float, string, and path. The int type supports unlimited precision. The float type supports the precision of float type of C programming language. Refer to the Characteristics of the Floating Types section of the 1999 ISO/IEC C Standard for details. The path type indicates a string that represents a path. / is used as a separator. Example: value_type=”string”
default	Python expression	No	Default value of the parameter. If the parameter has no default attribute but has the options attribute, the first option is picked as the default value. If the setting has neither default attribute nor options attribute, the initial value of the setting is set to 0 for int, 0.0 for float, “” for string, and False for bool. By default, it is not allowed to reference to other setting values. Use value_expr as reference values. Example: default=”LIFCL”
value_expr	Python expression	No	Python expression to compute the value of the parameter. The result is used as the parameter value if the setting is not editable. For example, divider is calculated by frequencies. Example: value_expr=” int(round((sys_clock_freq * 250.0) / i2c_left_desired_frequency))-1”
options	Python list or list of tuples	No	Candidate options for the parameter, which is used by the GUI to display a drop-down selector. It can be set to a simple list or a list of tuples. If it is a simple list, elements are displayed and written. If it is a list of tuples, the first item in tuple is displayed and the second item in tuple is written. Example: options=”[0.1, 0.2, 0.5, 1.0]”
output_formatter	str, nostr	No	Control how parameter values are written in output RTL files. The following formatters are supported. Str: parameter values are written as strings. nostr: quotation marks of strings are removed. Example: output_formatter=”str”
bool_value_mapping	Python tuple or list with two string elements	No	The map-to-map bool values to dedicated strings. By default, bool values are written as 1, 0. Example: bool_value_mapping=”(‘True’, ‘False’)”

© 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.

Property	Value	Mandatory	Description
editable	Python expression	No	Python expression to determine if the setting is editable. When a setting is not editable, it is greyed out in the GUI display and its value is computed by value_expr. Otherwise, the user input is used. Example: editable=”(FEEDBACK_PATH == ‘PHASE_AND_DELAY’)” FEEDBACK_PATH is a parameter name in metadata.xml.
hidden	True	No	Python expression to determine if the setting is hidden in the GUI. If hidden is set to True, the item is hidden in the GUI. The default value is False. The expression is resolved to boolean value after the user setting is changed. Example: hidden=”True”
drc	Python expression	No	Python expression to do DRC on the setting. True means DRC pass. Example: drc=”check_valid_addr_pre(I2C_LEFT,i2c_left)” check_valid_addr_pre is defined in plugin.py Parameter name: I2C_LEFT i2c_left
regex	Regular expression	No	Regular expression to do DRC on the value. For example, the value should be prefixed by 0b. Example: regex=”0b[01]+”
value_range	Python tuple or list with 2 comparable elements	No	Valid range of setting value, which is used to do DRC on the setting. The maximum value can be infinity, float(‘inf’). Example: value_range=”(0, 1023) if(i2c_right_enable) else (-9999, 9999)”
config_groups	Python expression	No	A name or names to group settings together. It is copied from the IP-XACT configGroups attribute and only supports the SystemBuilder value. If it is defined, the related RTL parameter is brought out to the IP instance top module, so that the System Builder can re-define its value.
Description	String	No	Detailed description of the setting.
Macro_name	id, value	No	Specify how to name the Verilog macro in the verilog_macro type setting item, the ID, or value of this setting item. The default value is setting, which means the setting ID is defined as a Verilog macro. If it is set as value, the evaluated setting value is the Verilog macro. Example: maro_name= “value” Result: `define <setting value> Note: This is only applicable to the setting item whose value_type attribute is set to string.

- Repeat steps above to configure all parameters as desired.

c. To configure Ports:
   - Click **Ports** from **Meta Data**, three port types are listed: IN, OUT, INOUT (Figure 2.20).

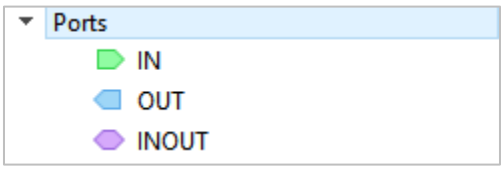

Figure 2.20. Three Port Types

© 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.

- Right-click **IN** from the **Ports** area, and choose **Add Port** (Figure 2.21).

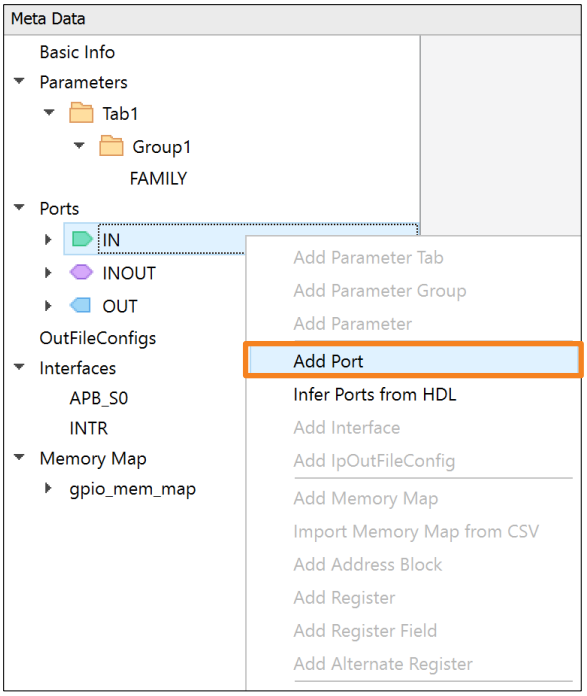

Figure 2.21. Right-click Menu of IN Port

Note: You can add all ports by using **Infer Ports from HDL**, if you prepare valid RTL files and add RTL files in the **Design Files** section. Click **Infer Ports from HDL**, the **Add inferred ports to meta data** wizard pops up (Figure 2.22). If you have added some ports and then choose **Infer Ports from HDL**, all the previously added ports are removed and only ports from the HDL file can be added.

© 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.

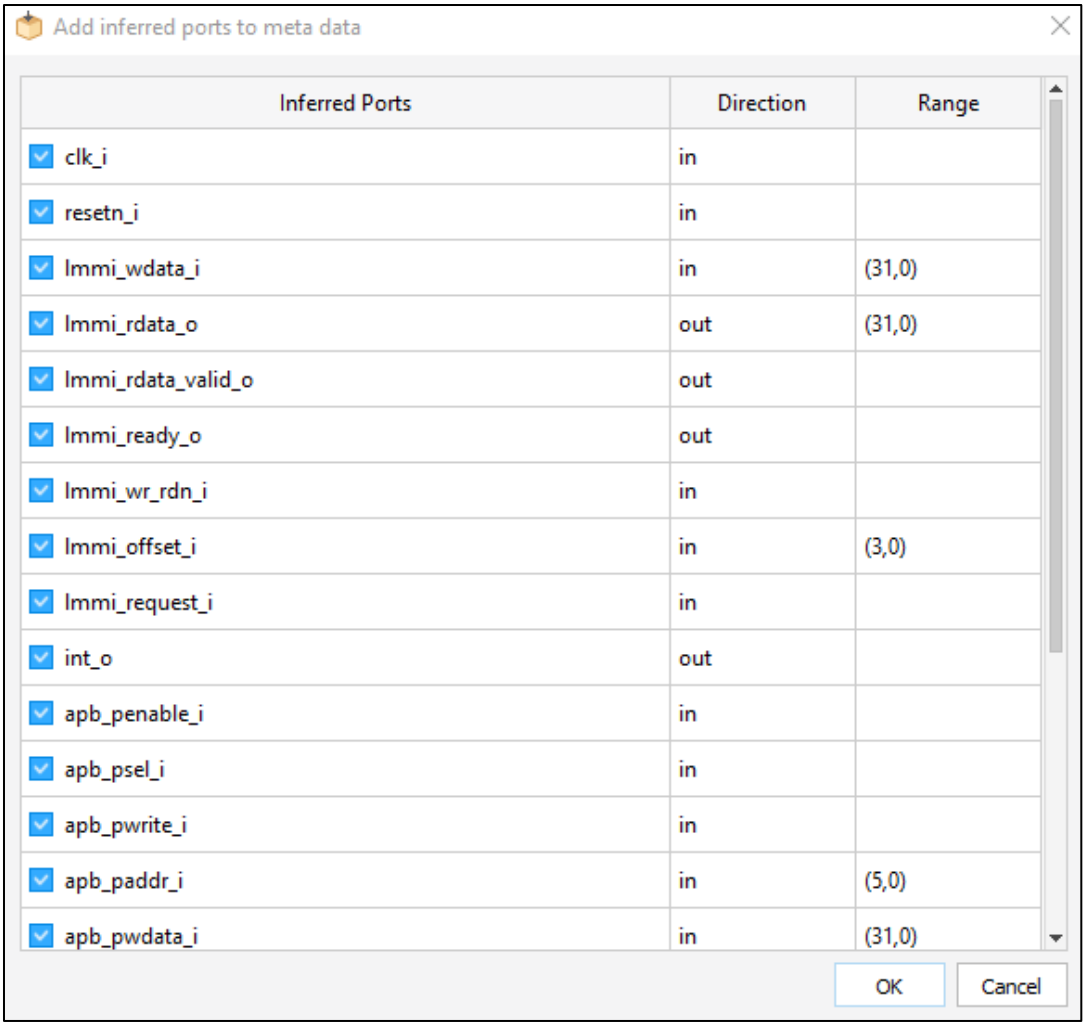

Figure 2.22. Add inferred ports to meta data Wizard

- **NewPort1** is created. Double click **NewPort1** to rename the port (Figure 2.23).

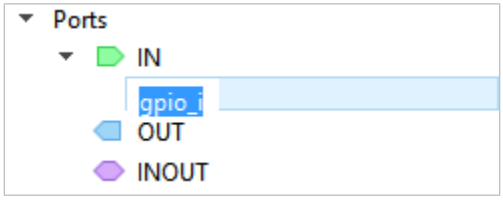

Figure 2.23. Rename an IN Port

- Configure all properties of the IN port as desired (Figure 2.24). Properties listed in bold must be configured. The details of each property are shown in Table 2.2.

© 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.

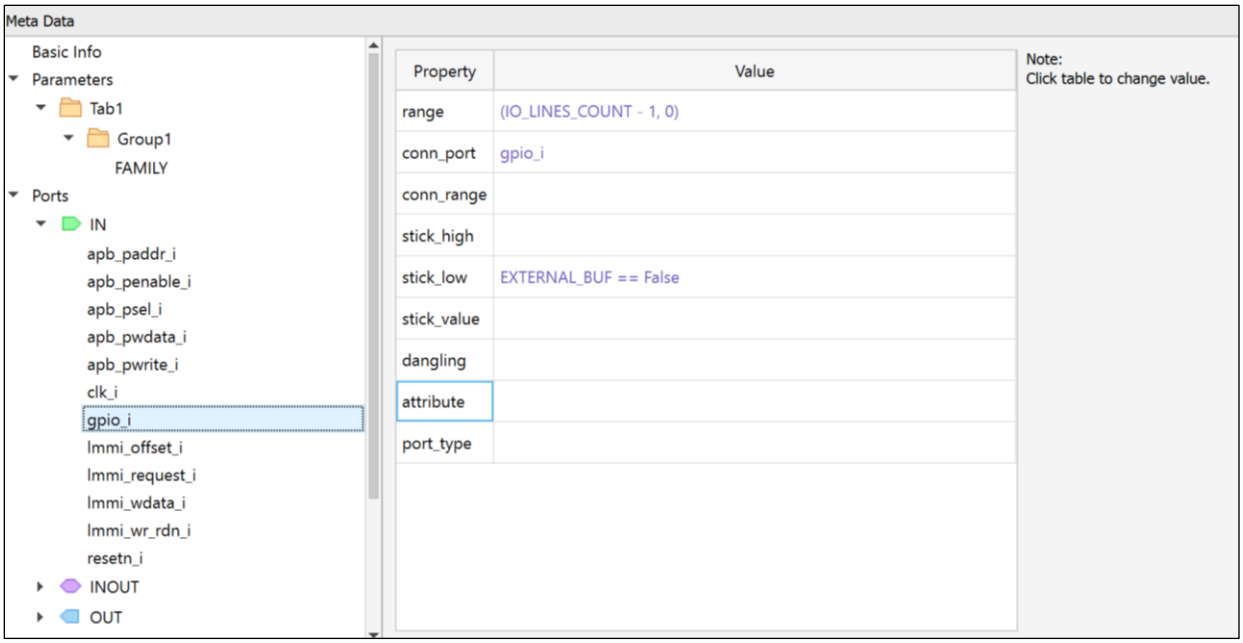

Figure 2.24. Configure an IN Port

Table 2.2. Details of Port Property

Property	Value	Mandatory	Description
range	Python tuple or list with two-integer elements	No	Range of this port. It should be a Python expression whose evaluation result is a tuple or array with two elements. Example: range=”(A_WDT-1, 0)” A_WDT is a parameter name.
conn_port	Valid Verilog module name	No	Name of the top-level IP port in RTL to which this port connects to. Value of port name of user HDL top module is used if conn_port is not specified. Example: conn_port=”Clk”
conn_range	Python tuple or list with two-integer elements	No	Range of conn_port. It should be a Python expression whose evaluation result is a tuple or array with two elements. Example: conn_range=”(A_WDT-1, 0)” A_WDT is a parameter name.
stick_high	Python expression	No	Python script. True: tie this port to 1. Example: stick_high=”True”
stick_low	Python expression	No	Python script. True: tie this port to 0. Example: stick_low=”no_seq_pins()” no_seq_pins is defined in plugin.py.
stick_value	Python expression	No	Python script. Tie port to the evaluation result of this attribute.
dangling	Python expression	No	Python script. True: keep this port unconnected. Example: dangling=”not USE_COUT” USE_COUT is a parameter name.

© 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.

Property	Value	Mandatory	Description
attribute	Python expression	No	Python script. The value is written to the .v file as the attribute of the port. Example: attribute="'(* AAA, BBB=1 *)'" => (* AAA, BBB=1*) input PORT;
port_type	String	No	Data, reset, and clock are valid values. The default value is data. Port_type is passed to the IPXact component.xml as a lsccip:isClk or lssccip:isRst node in venderExtensions in component/model/ports/port.

- Repeat steps above to configure all ports as desired.

d. To configure OutFileConfigs:

- OutFileConfigs specifies all customized output file configuration nodes <fileConfig> for the whole customized flow (Figure 2.25). A fileConfig node contains a group of attributes to specify a specific file or directory generation.
  Note: Skip the customization of OutFileConfigs if you are not familiar with the Python implementation details of IP generation. Lattice IP Packager can automatically manage OutFileConfigs settings.

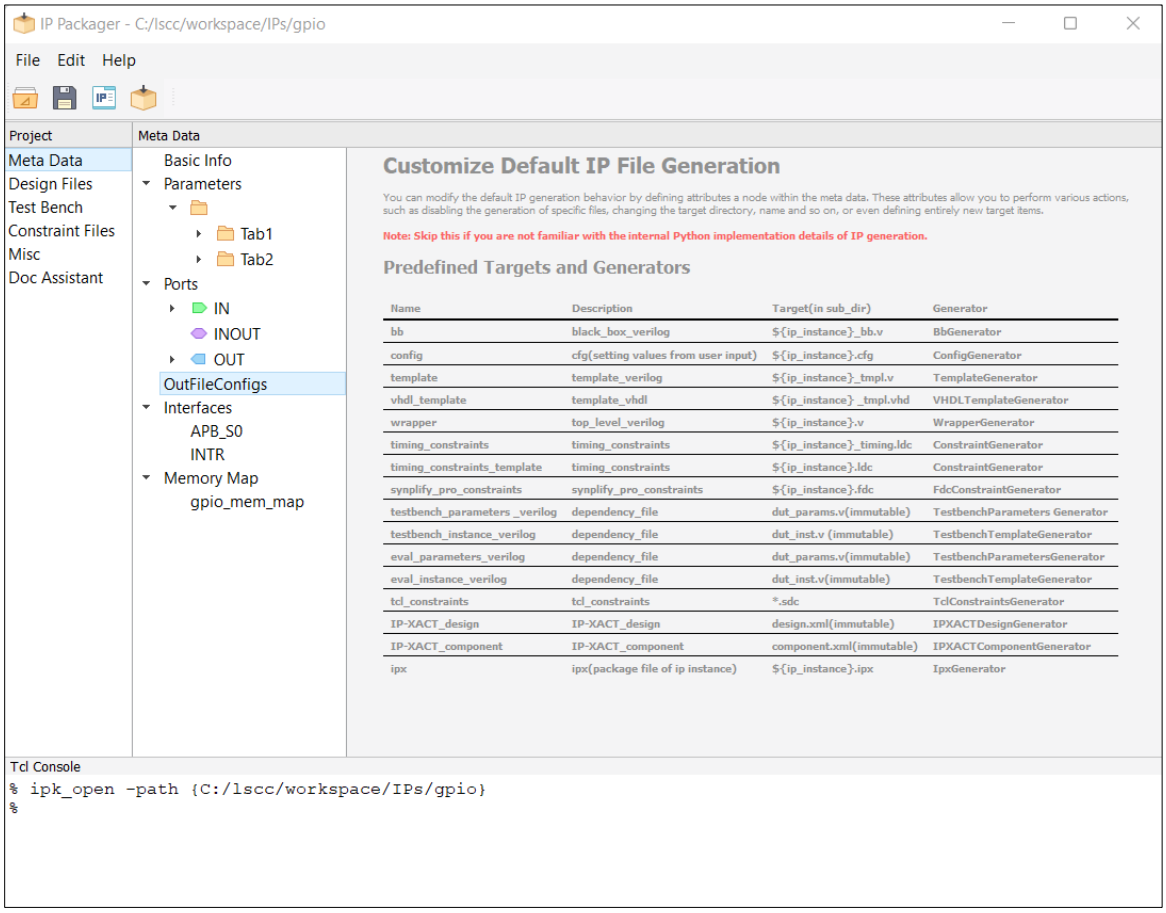

Figure 2.25. Output File Configuration

- Right-click **OutFileConfigs** from the **Meta Data** view, and select **Add IP OutFileConfig** (Figure 2.26). A new interface **tcl_constraints** is created.

© 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.

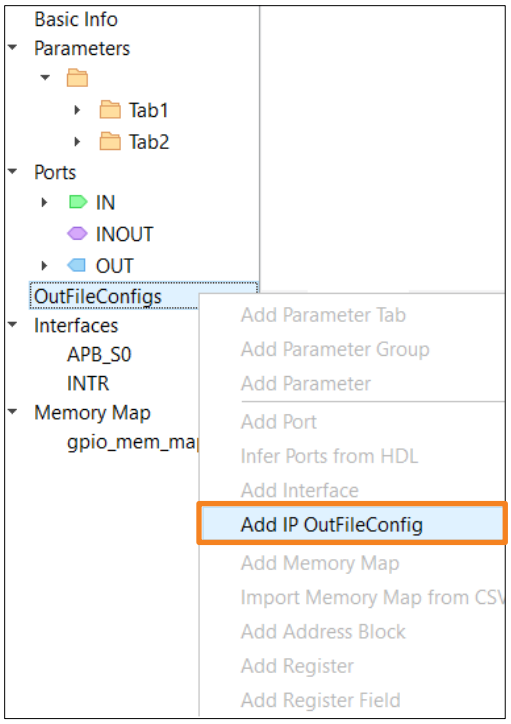

Figure 2.26. Right-click Menu of OutFileConfig

- Double click **tcl_constraints** to change the type of **OutFileConfigs** using a drop-down menu (Figure 2.27).

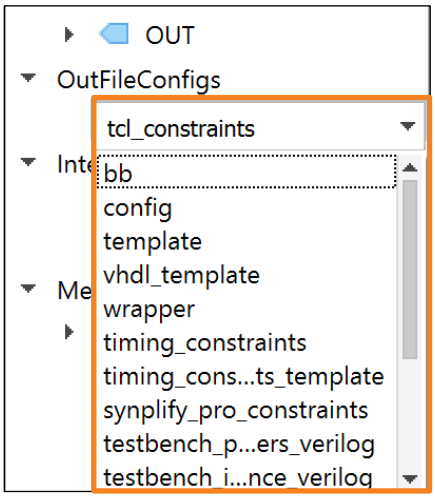

Figure 2.27. Change Type of OutFileConfigs

- Click **tcl_constraints** and configure the value of each property according to the usage (Figure 2.28).

© 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.

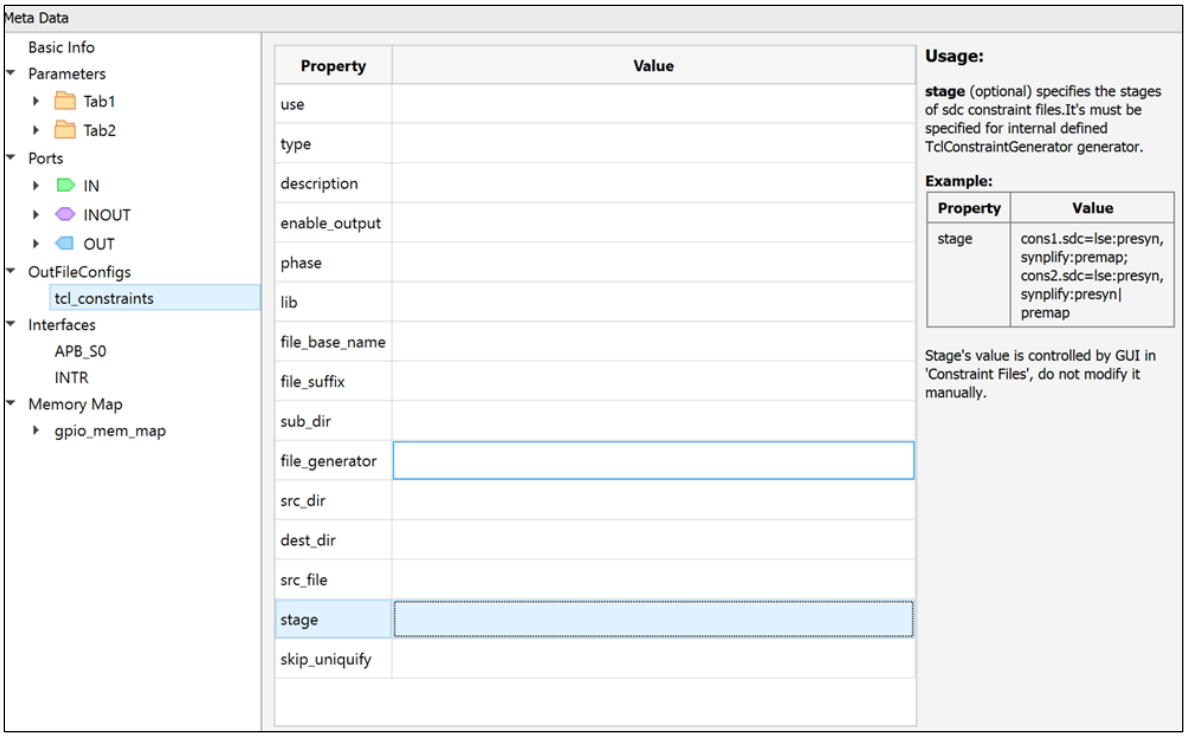

Figure 2.28. Configure Property Value in OutFileConfigs

e. To configure Interfaces:
   - Right-click **Interfaces** from the **Meta Data** view, and choose **Add Interface** (Figure 2.29). A new interface **NewInterface1** is created.

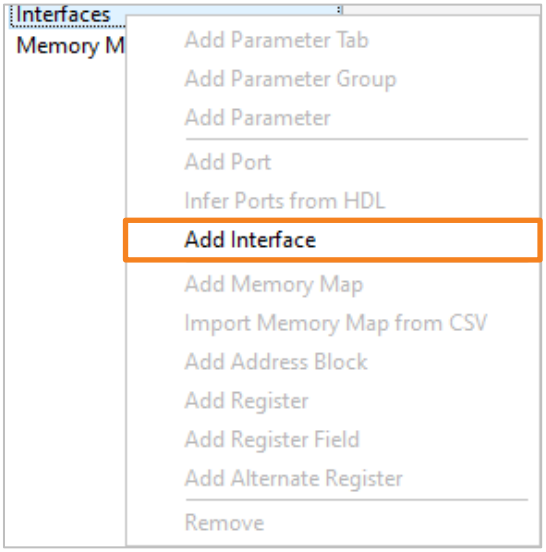

Figure 2.29. Right-click Menu of Interfaces

   - Double click **NewInterface1** to rename interface (Figure 2.30).

Figure 2.30. Rename Interface

© 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.

- Configure the interface as desired (Figure 2.31).

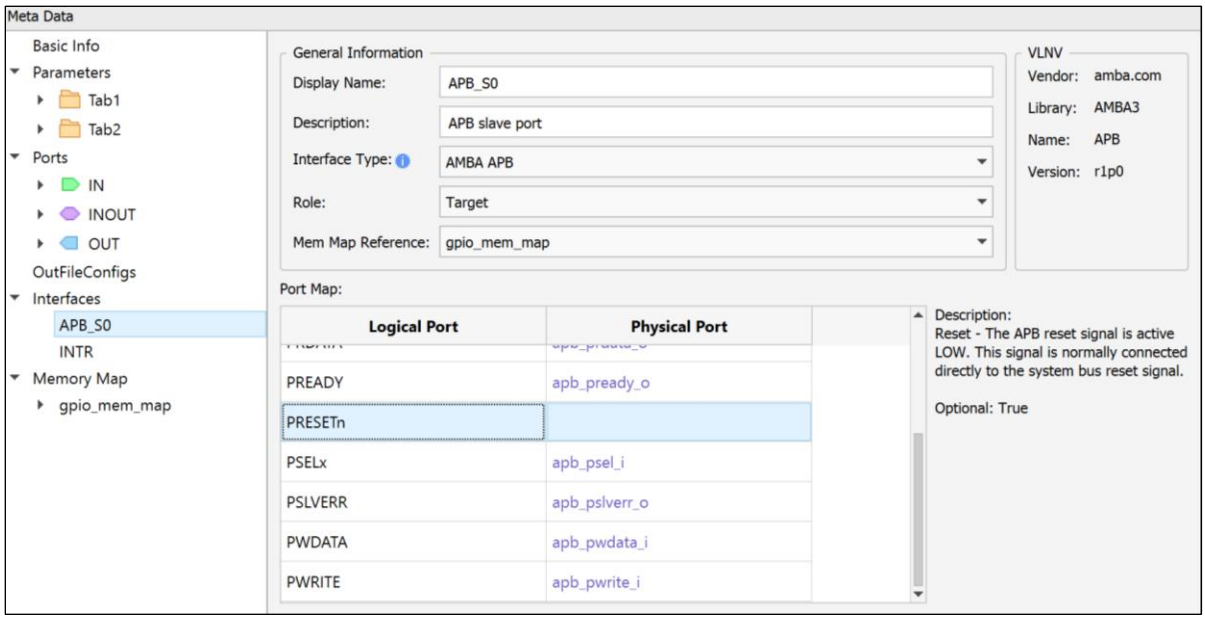

Figure 2.31. Configure Interface

- Repeat steps above to configure all interfaces as desired. Alternatively, click the **Auto Assign** button to configure interfaces (Figure 2.32). The IP Packager tool searches all ports to match current interface. **Note:** You can click the **Help** button to view more details regarding the **Auto Assign** function.

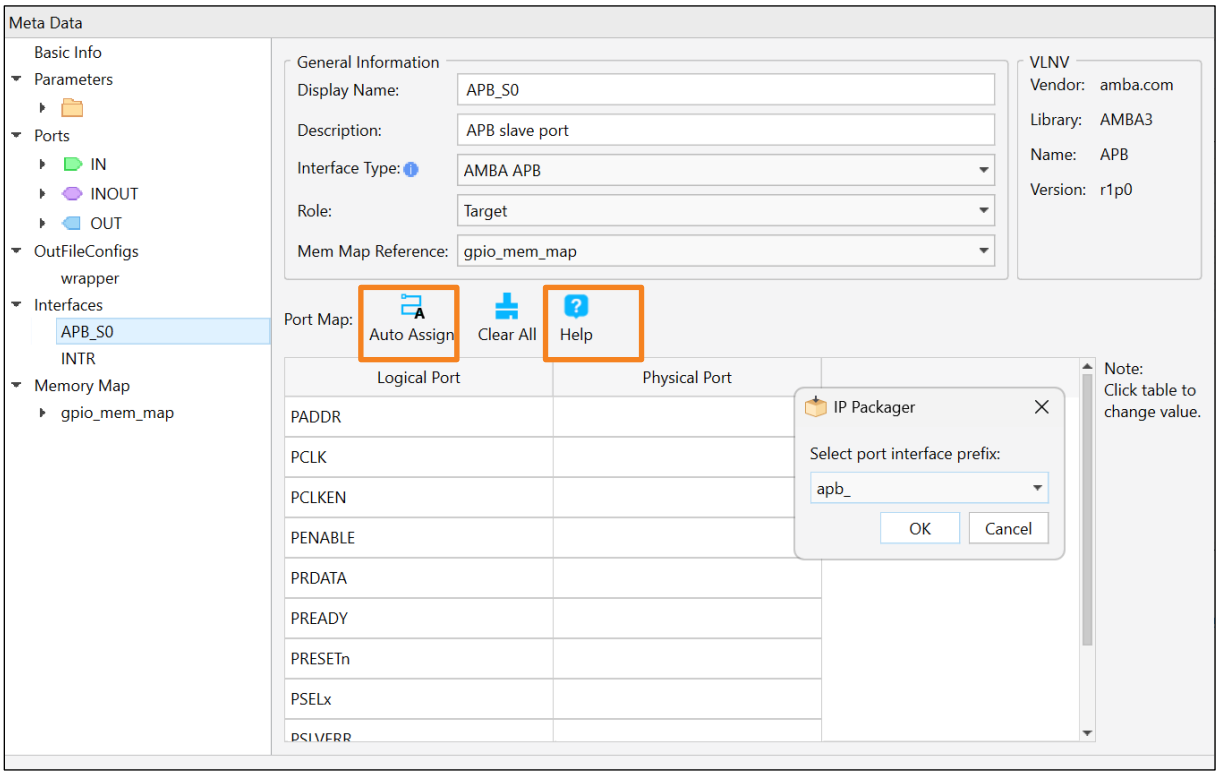

Figure 2.32. Auto Assign Button

© 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.

f. To configure Memory Map:
   - Right-click **Memory Map** from **Meta Data**, and choose **Add Memory Map** or **Import Memory Map from CSV** (Figure 2.33). A new memory map is created (Figure 2.34).

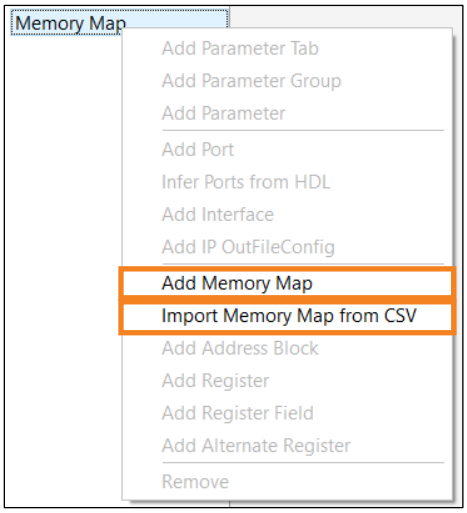

Figure 2.33. Right-click Menu of Memory Map

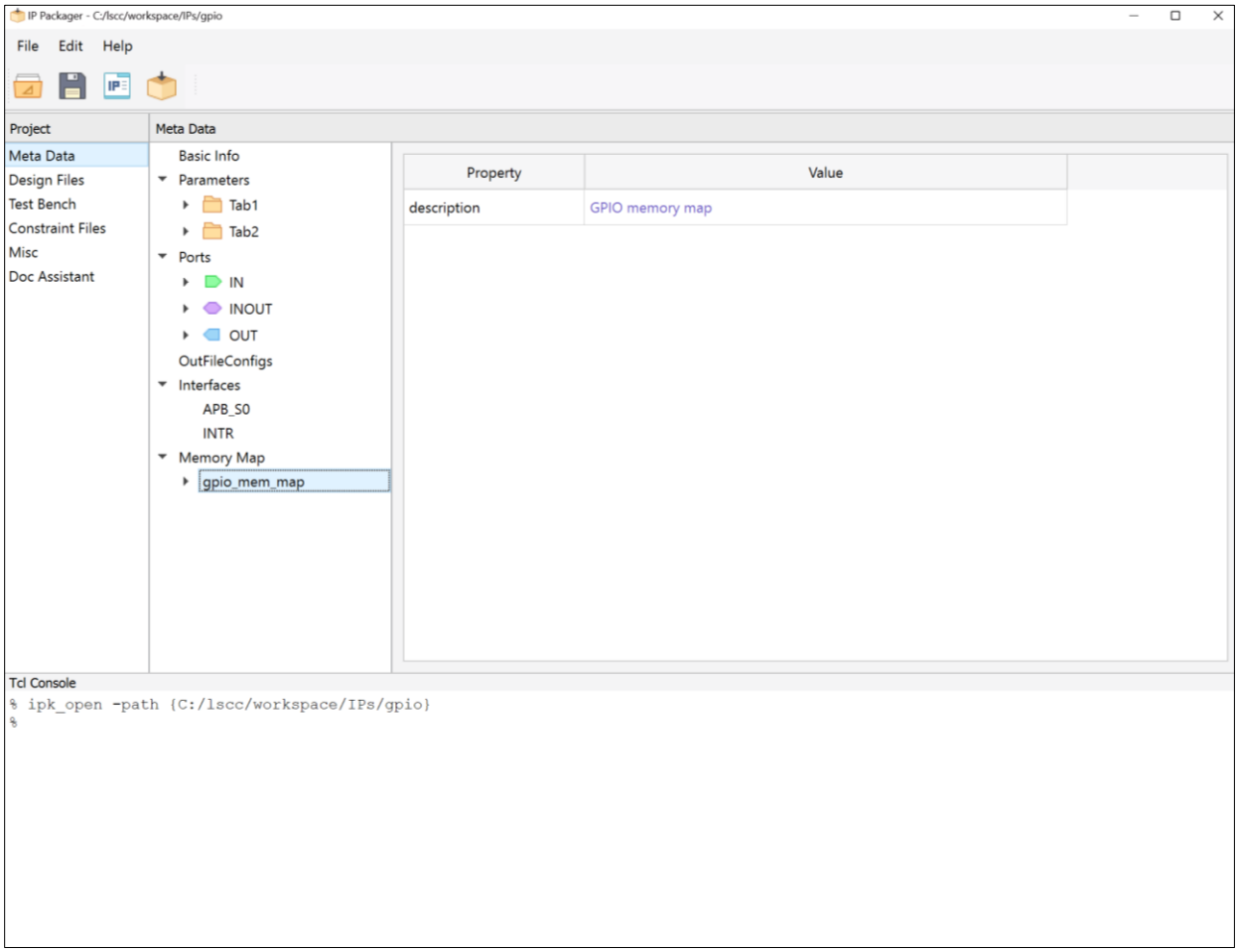

Figure 2.34. Generate New Memory Map

© 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.

g. To add Memory Map reference for interface:
Click **Mem Map Reference** to select an existing Memory Map for interface (Figure 2.35).

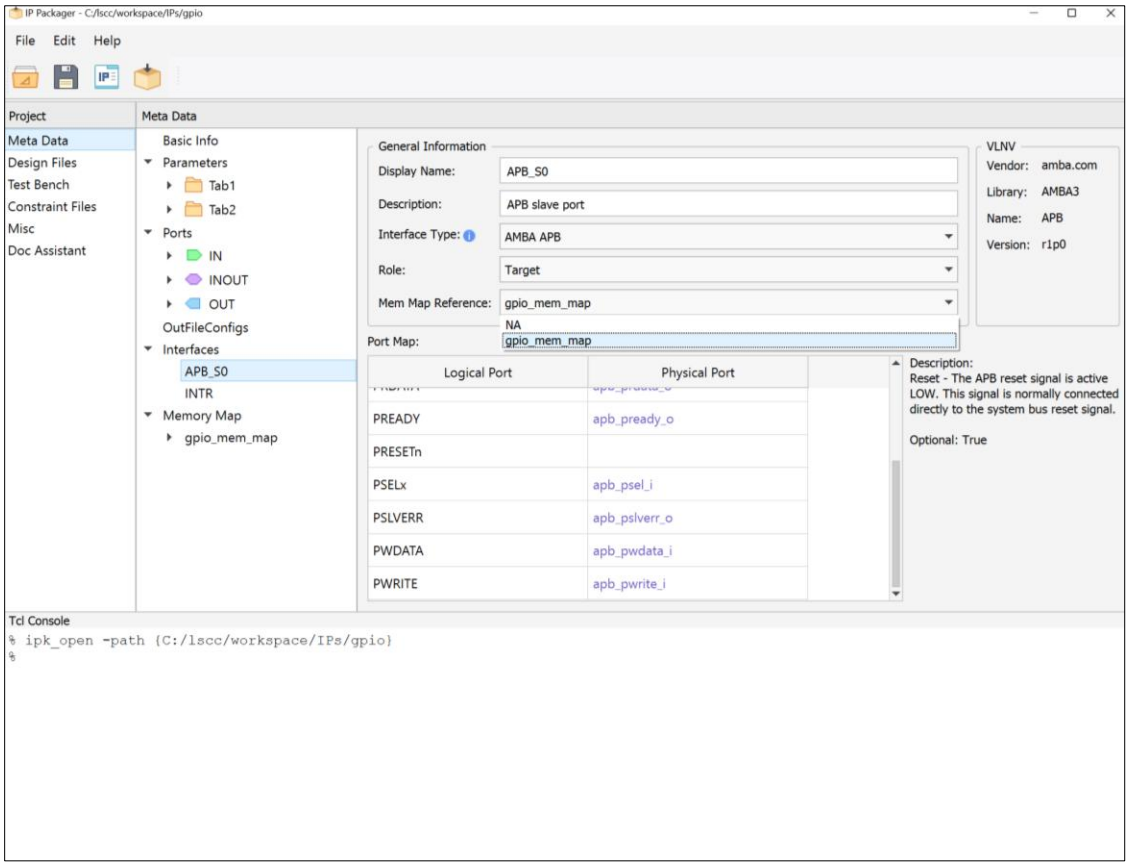

Figure 2.35. Add Memory Map for Interface

2. From the IP Packager Project area, click **Design Files**. The IP Packager GUI shows the Design Files information (Figure 2.36).

© 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.

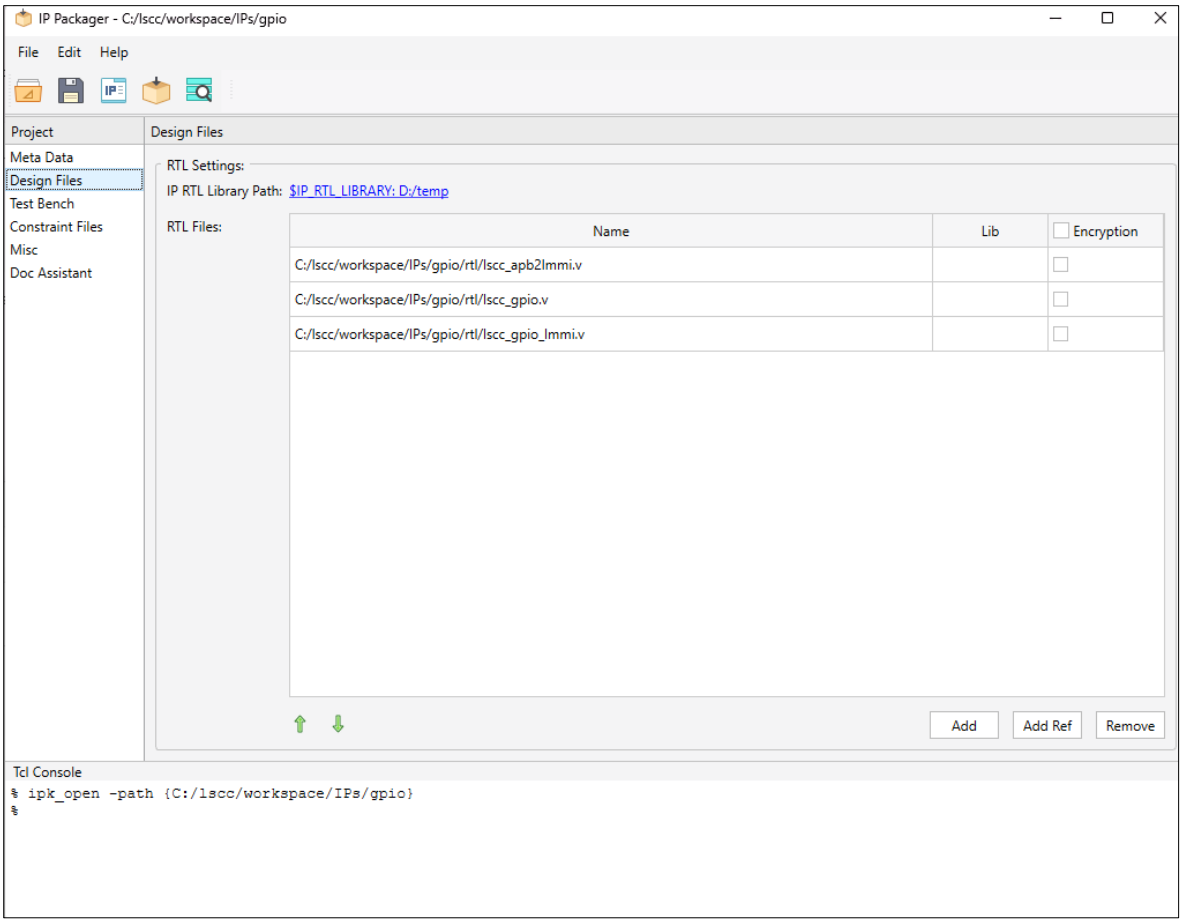

Figure 2.36. IP Packager with Design File Details

- Click the Add button to select a desired RTL file in the Design Files field. The added file is packaged into the IP.
- Click the Add Ref button to select a reference RTL file in the IP RTL Library Path. This file is not packaged into the IP.
- Click the Remove button to remove an RTL file.
- If there are syntax errors when running ipk_drc, click the ⇧ or ⇩ button to change the sequence of selected RTL files.

Note: Design file configuration is mandatory. Specify at least one RTL file. If the RTL file needs to be encrypted, check the **Encryption** option for it.

IP Packager supports importing RTL files from the library path without saving them locally. When files change, you can always get the latest files from the library path without updating the IP version or regenerating the IP. This helps avoid maintaining the RTL files in multiple locations.

- To add reference RTL file into the IP:
  - Set the path of RTL library.
  - Add reference RTL file from library by clicking the **Add Ref** button.

© 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.

- To set IP RTL Library path in Lattice Propel Builder software:
  If you want to use an IP that contains a reference RTL file in Lattice Propel Builder, you must set the value of IP RTL Library in **Menu** > **Tools** > **Options** > **Directories**, as shown in Figure 2.37. Alternatively, you can set the value to the environment variable “LATTICESEMI_IP_RTL_LIB_PATH”.

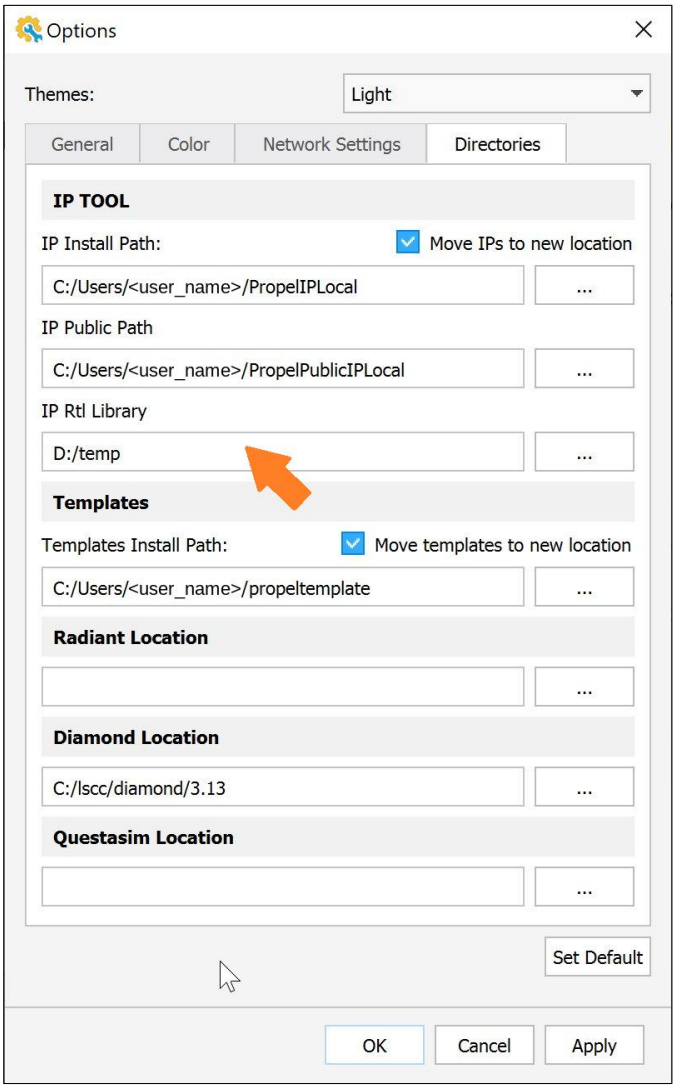

Figure 2.37. Set IP RTL Library Path in Lattice Propel Builder

© 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.

- To Set IP RTL Library path in Radiant:
  If you want to use an IP that contains a reference RTL file in the Lattice Radiant software, you must set the value of IP RTL Library in **Menu** > **Tools** > **Options** > **General** > **Directories**, as shown in Figure 2.38.
  Alternatively, you can set the value to the environment variable "LATTICESEMI_IP_RTL_LIB_PATH".

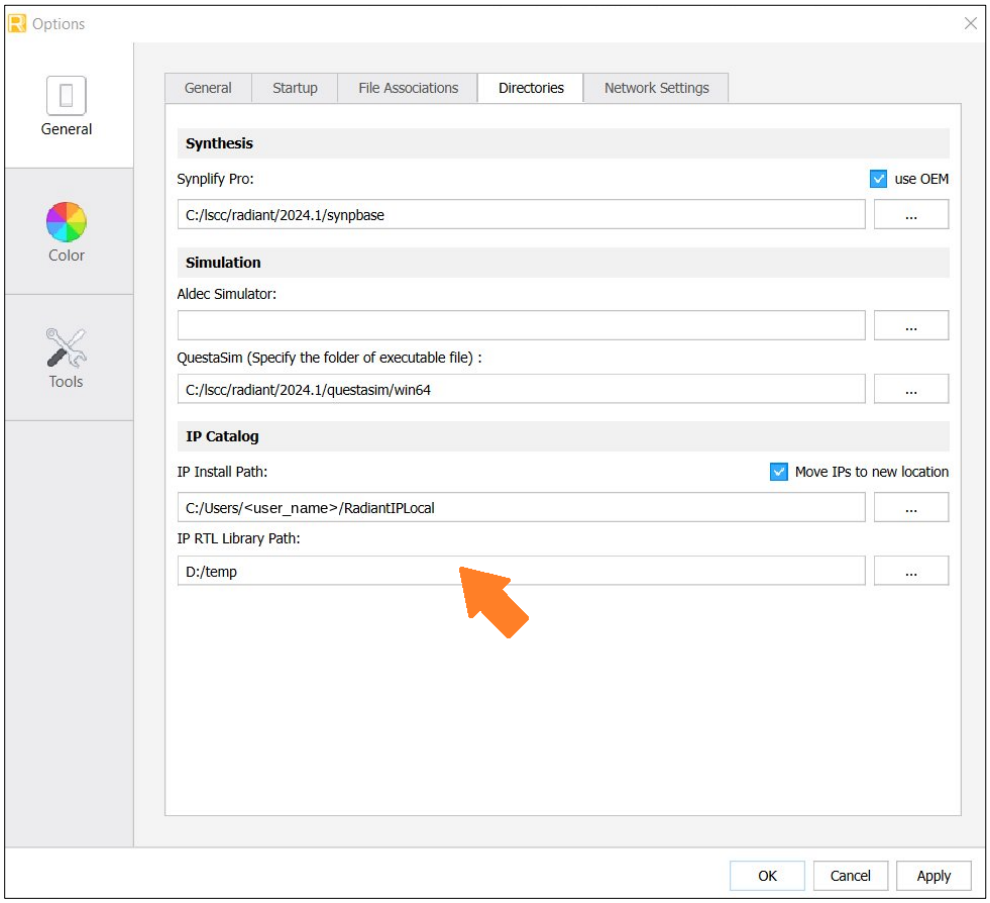

Figure 2.38. Set IP RTL Library Path in Lattice Radiant Software

3. From the IP Packager Project area, click **Test Bench**. The Test Bench information is shown in detail (Figure 2.39).

© 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.

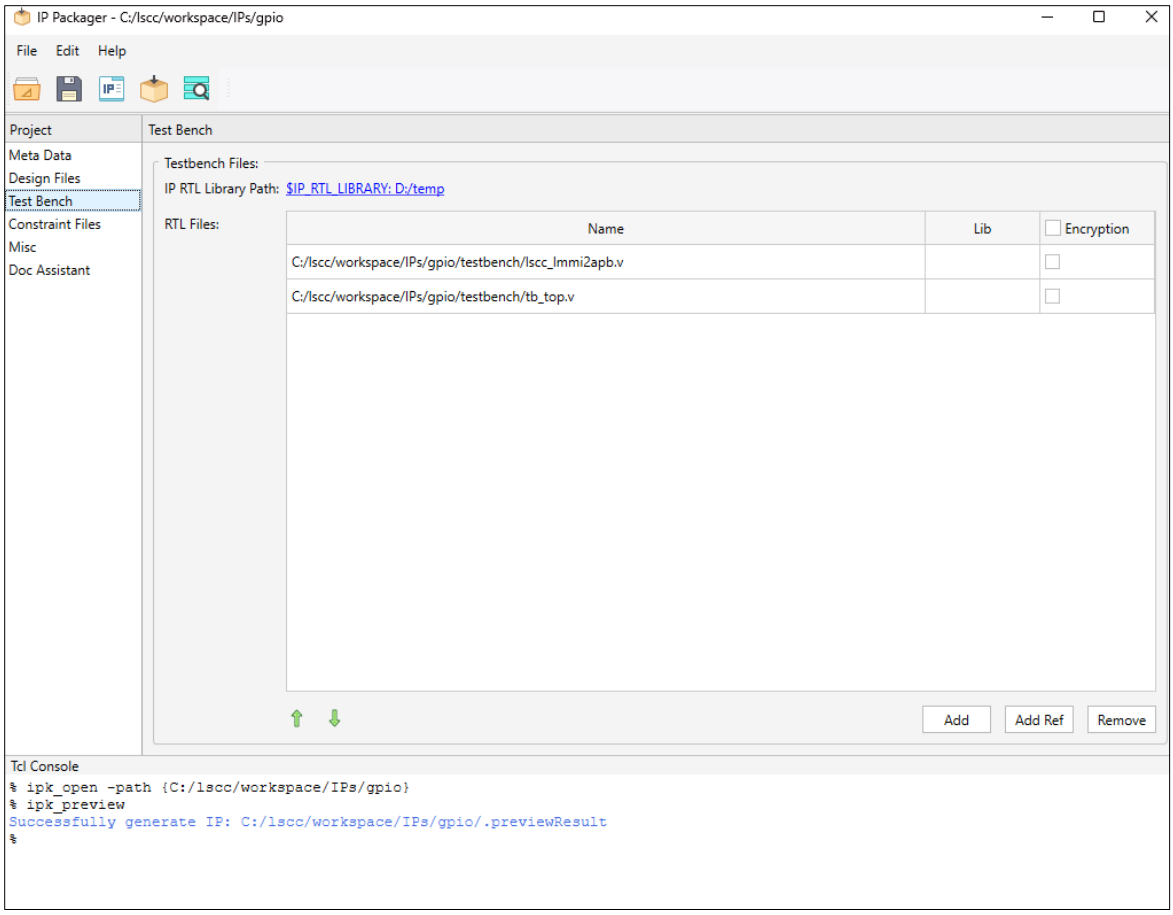

Figure 2.39. IP Packager with Test Bench Details

- Follow the steps as those for configuring the design files to configure the Test Bench.
  Note: The Test Bench files configuration is not mandatory.

© 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. From the IP Project area, click **Constraint Files**. The constraint files information is shown in detail (Figure 2.40).
   - Click Add to select a desired sdc file in the Constraint Files field. The added file is packaged into the IP.
   - Click Remove to remove an sdc file.
   - Double click the cell of **Lattice LSE** or **Synplify Pro** to change synthesis stage.

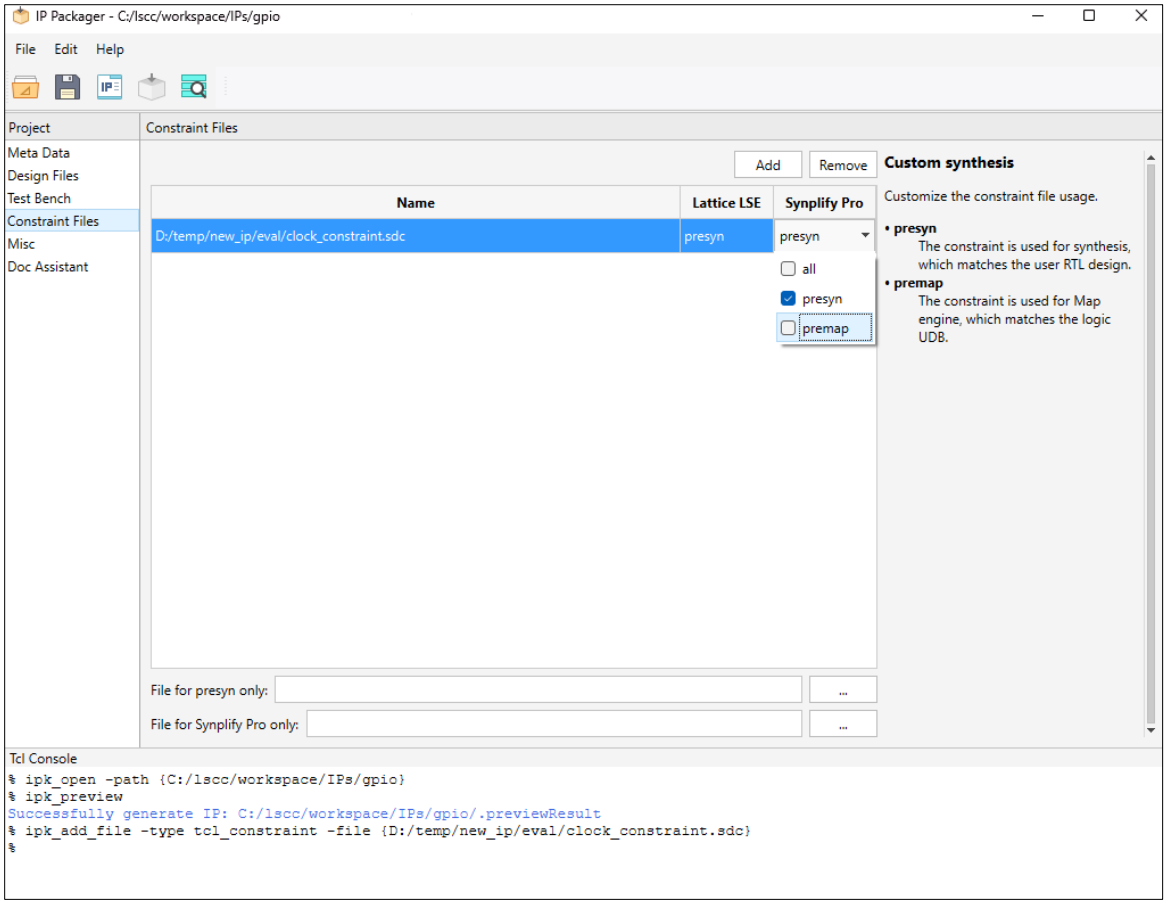

Figure 2.40. IP Packager with Constraint Files Details

Lattice Radiant design object name is different in each design stage and synthesis tool, as shown in Figure 2.41.

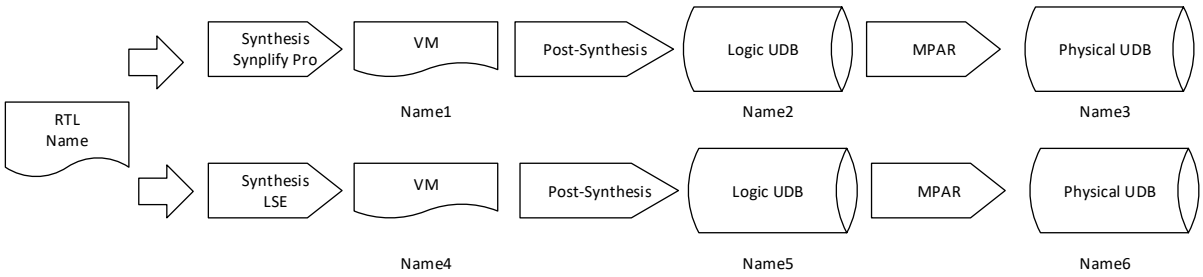

Figure 2.41. Lattice Radiant Design Object Name in Different Stages and Synthesis Tools

To support constraint in a single file, you must use the TCL scripting function to identify the corresponding stage and synthesis tool. The constraint parser is an internal TCL interpreter. The following two variables are added into this TCL interpreter and they are dynamically set via different engine tools.

© 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.

- $radiant(stage)

  Valid values:

  - presyn
  - premap
- $radiant(synthesis)

  Valid values:

  - lse
  - synplify

Notes:

- Constraints from the previous design stage are passed down to the next design stage if they are not dropped. Therefore, there is no need to duplicate the same constraints in different stages unless you want to overwrite them or add new ones.
- The constraints, which are out of if-else statement, are taken in multiple times. In the example below, the constraint set_false_path is duplicated three times for different engines.
- Single constraint file style is for IP developers only. It is not supported for user constraints.

```
set var 5
if { $radiant(stage) == "presyn" } {
      create_clock -period 10 -name myclk [get_ports clk]
}

if { $radiant(synthesis) == "lse" } {
      # LSE
      if { $radiant(stage) == "presyn" } {
            set_max_delay -from [get_cells {c[0]}] $var
      }elseif { $radiant(stage) == "premap" } {
            set_max_delay -from [get_cells {c_6__I_0.ff_inst}] [expr $var+20]
      }
} else {
      # synplify
      if { $radiant(stage) == "presyn"} {
            set_max_delay -from [get_cells {c[0]}] 10
      }elseif { $radiant(stage) == "premap" } {
            set_max_delay -from [get_cells {c_reg[0].ff_inst}] 25.0
      }
}

set_false_path -from [get_ports {q}]
```

Figure 2.42. Single Constraint Example

5. From the IP Packager Project area, click **Misc**. The Misc-related information is shown in detail (Figure 2.43). Configure Misc information as desired.

© 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.

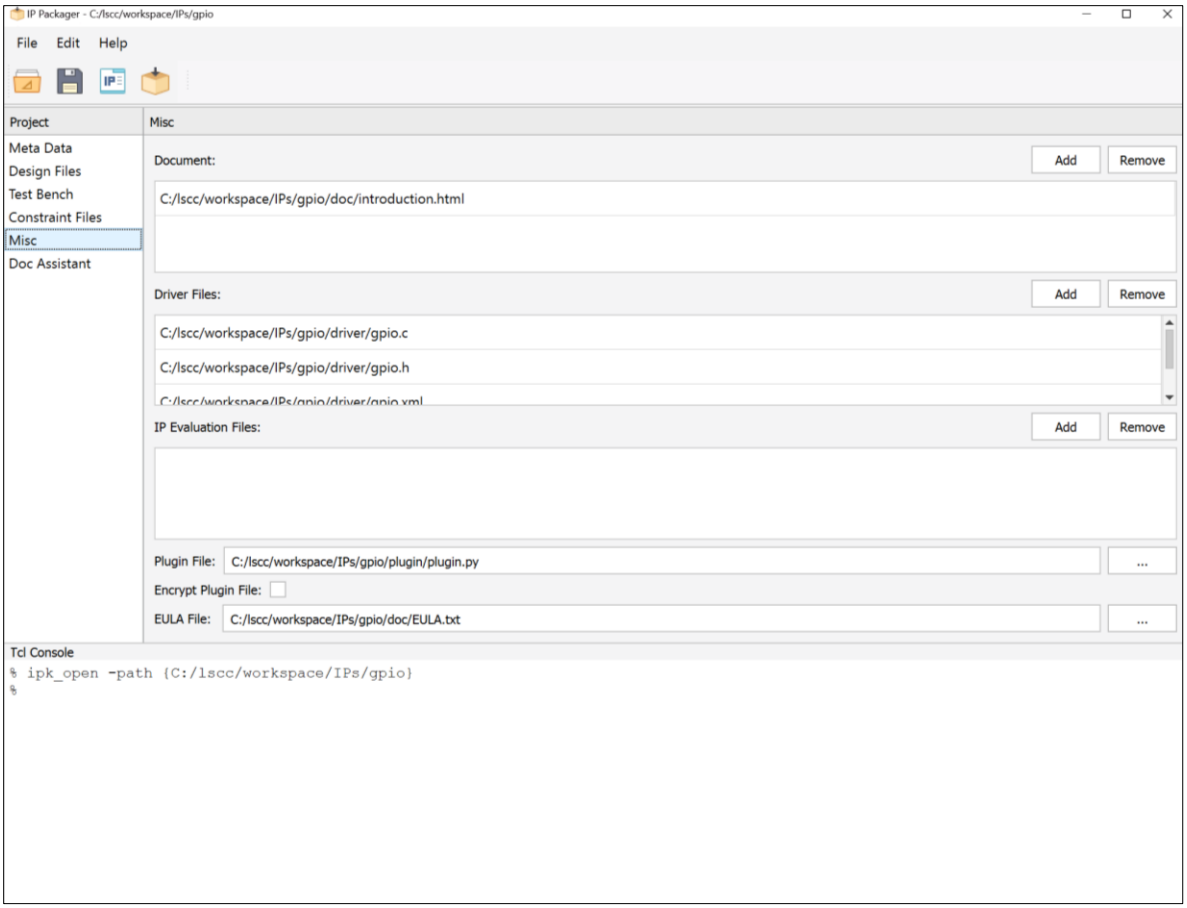

Figure 2.43. IP Packager with Misc Details

© 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. From the IP Packager Project area, click **Doc Assistant**. The Doc Assistant information is shown in detail (Figure 2.44).

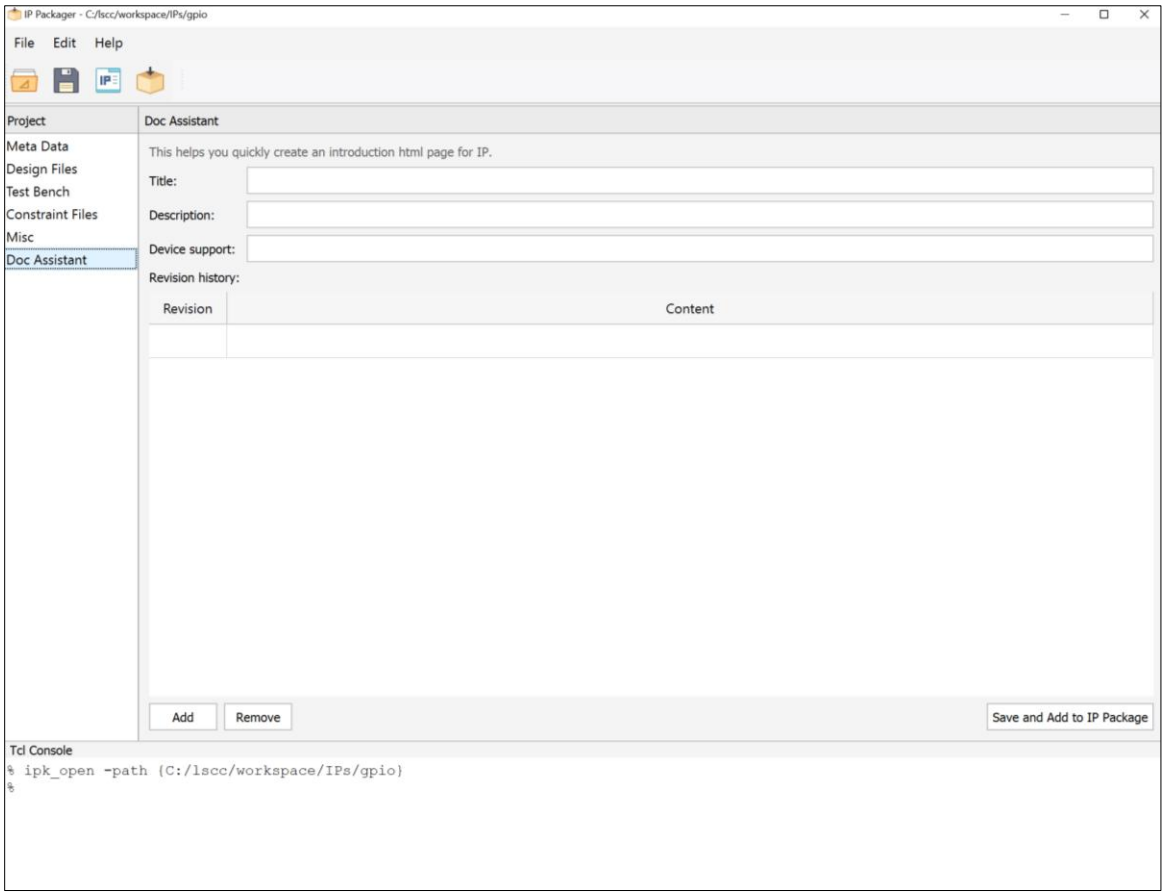

Figure 2.44. IP Packager with Doc Assistant Details

- Enter the desired title, description, and device in the **Title**, **Description**, and **Device support** fields accordingly.
- Double-click and enter a revision in **Revision** field. Double-click and enter revision contents in **Content** field.

  Refer to Figure 2.45 as an example. Click the Add button. A new blank row is added to the **Revision history** area. You can add desired revision and contents accordingly. Select the revision you want to delete, and click the Remove button to remove a Revision.
- Click the Save and Add to IP Package button to create an introduction html page and save this file to the IP package.

© 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.

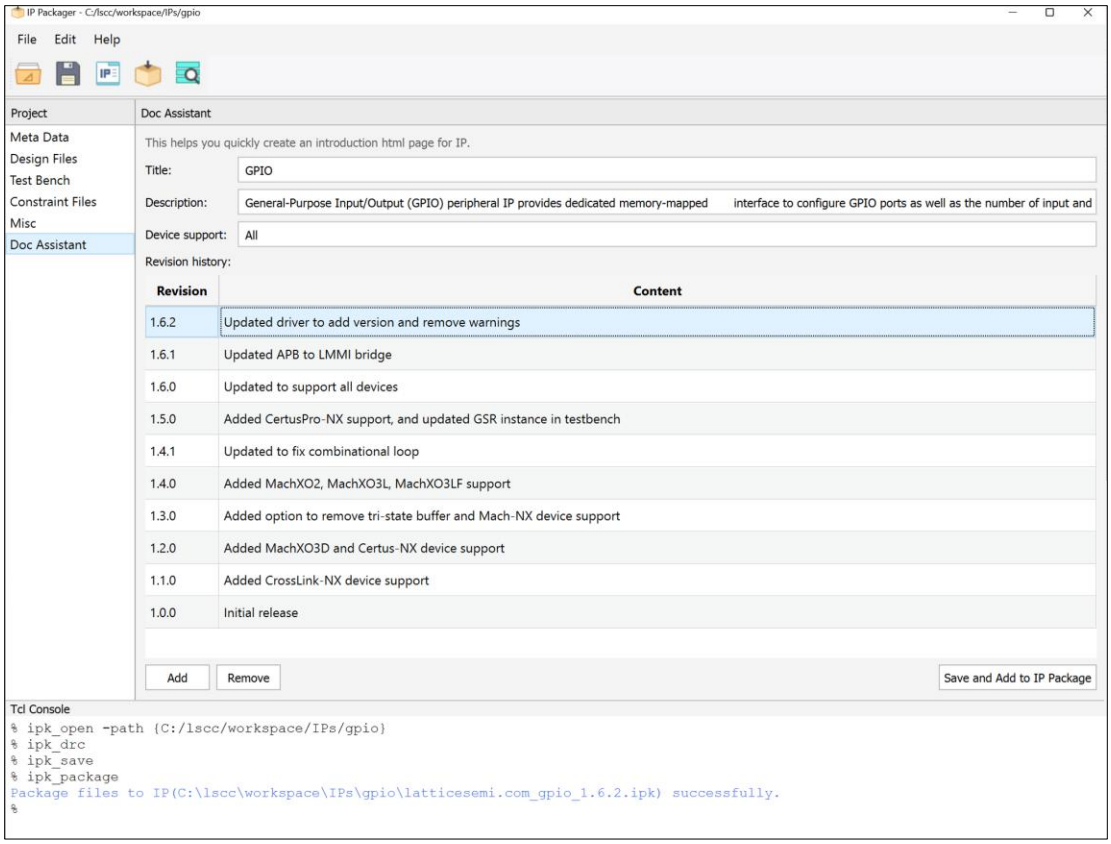

Figure 2.45. Configure Doc Assistant

7. Click the button to check if there are errors in the current IP.

© 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.

2.2.3. Previewing IP

1. Choose **Edit** > **IP Preview** from the IP Packager menu. Or, click the **IP Preview** icon from the IP Packager toolbar.
2. **IP Preview** pops up showing the configuration component for the IP module (Figure 2.46), if the IP project configuration is correct. Otherwise, an error message pops up (Figure 2.47).
3. Click the **Generate** button to check the generated IP (Figure 2.48 and Figure 2.49).

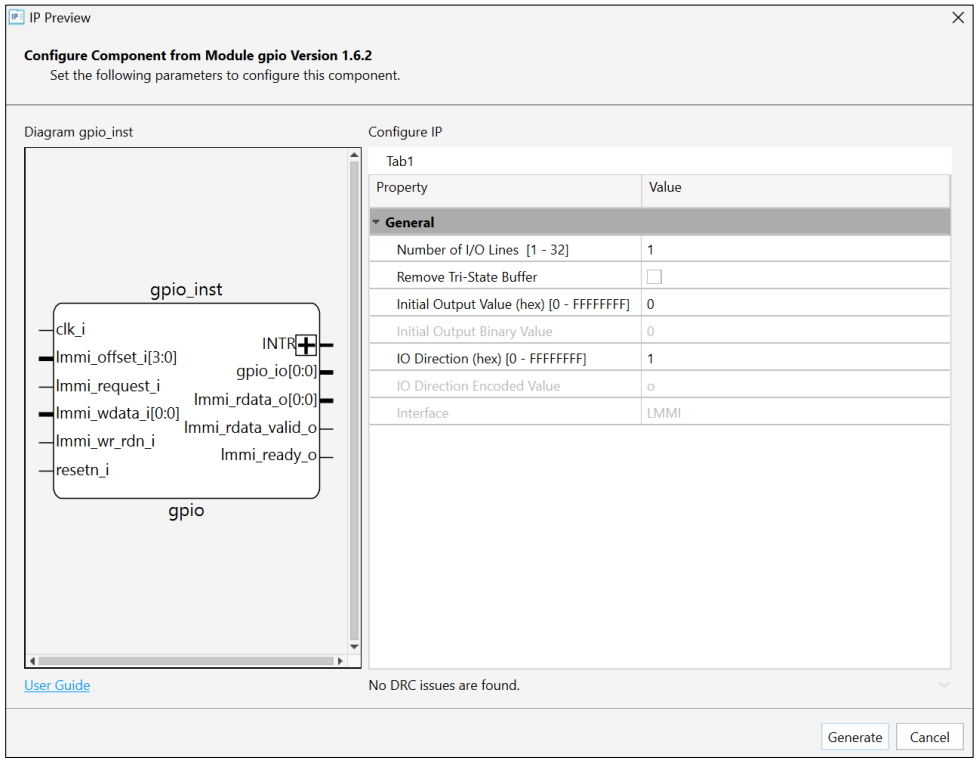

Figure 2.46. IP Preview Shows Configuration for IP Module

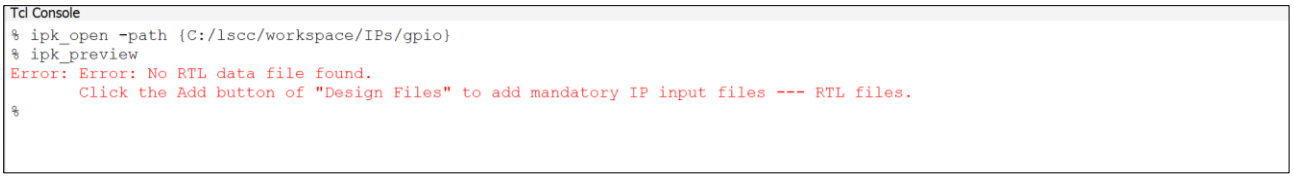

Figure 2.47. Error Message Pops Up in IP Packager

© 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.

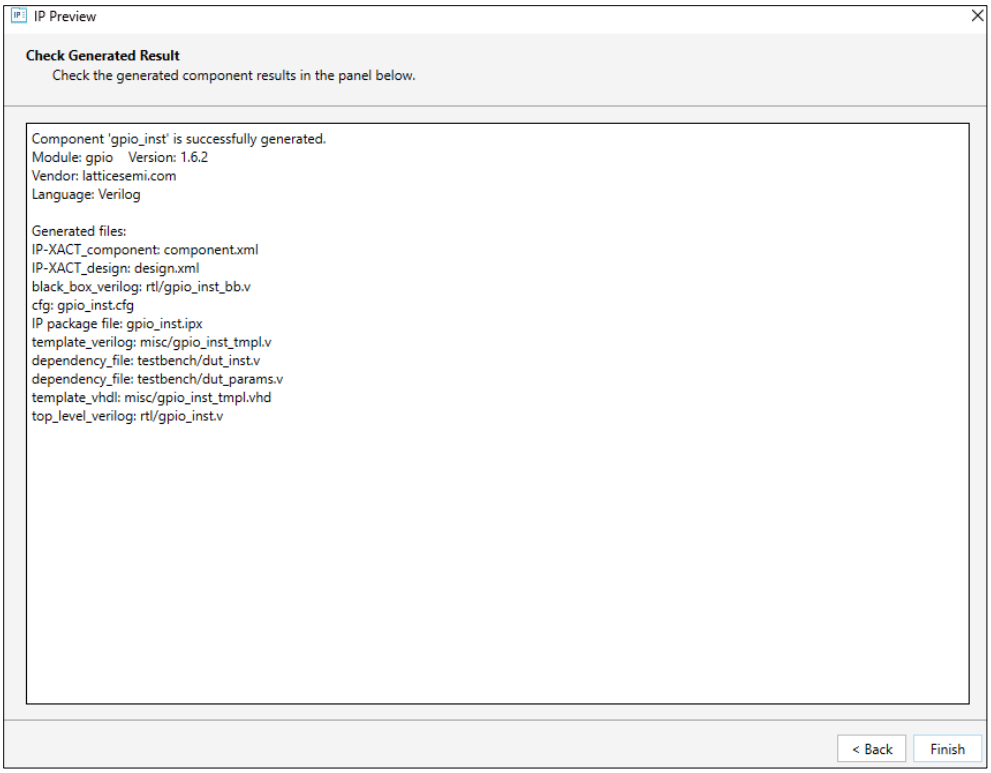

Figure 2.48. Generate Result of the IP in IP Preview

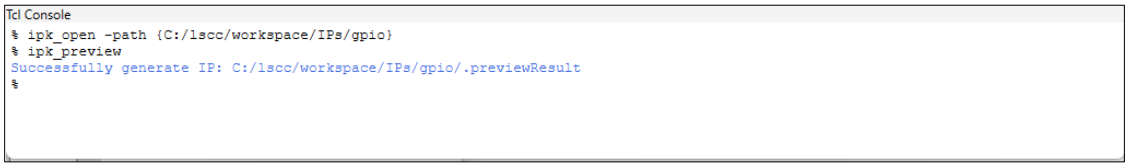

Figure 2.49. Generate Result of the IP in Tcl Console

© 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.

2.2.4. Packaging IP

1. Choose **Edit** >  **Package IP** from the IP Packager menu. Or, click the **Package IP** icon  from the Toolbar.
2. A message pops up showing the packaging is completed successfully (Figure 2.50).

Figure 2.50. IP Packager Pops Up Successful Packaging Message

Note: When the setting value is changed, the packaging operation is disabled. You must perform a preview operation before packaging. If the preview is successful, the packaging operation is enabled. Otherwise, error messages are displayed in the output widget.

2.3. Editing IP Package Files

2.3.1. Metadata File

Metadata.xml can be generated after editing the IP. You can also manually edit the file. The IP platform uses XML file to describe metadata of a soft IP. Namespace “lsccip” is used in XML file. The content of the XML file consists of three mandatory nodes including <general>, <settings>, and <ports>, five optional nodes including <busInterfaces>, <addressSpaces>, <memoryMaps>, <componentGenerators>, and <estimatedResources>. One optional new child node <outFileConfigs> is added to support the customized IP generation flow in Lattice Propel design environment (Figure 2.51).

```
<lsccip:ip xmlns:lsccip="http://www.latticesemi.com/XMLSchema/Radiant/ip"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.0">
  <lsccip:general>
    ......
  </lsccip:general>
  <lsccip:settings>
    ......
  </lsccip:settings>
  <lsccip:ports>
    ......
  </lsccip:ports>
</lsccip:ip>
```

Figure 2.51. Example XML of Metadata Layout

1. <general> node: describes the general information about a soft IP, for example, its name, version, and category. Table 2.3 shows the child nodes of <general> node.

© 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.

Table 2.3. Child Nodes of General Node

Child Node	Mandatory	Description
vendor	Yes	Soft IP vendor. An official soft IP should have the vendor name. Example: latticesemi.com
library	Yes	Library of the soft IP. If the library node is not set, the default value ip is used. Example: ip, interface
name	Yes	Name of the soft IP. The name must be unique among soft IPs of the same vendor and library. Example: adder
display_name	No	Name to be displayed in the software. If display_name is not set, software displays the IP name directly. Example: Adder
version	Yes	Version of the soft IP. Example: 1.0.0
category	Yes	Category of the soft IP. Category can be hierarchical. Levels are separated by “,”. Example: Memory_Modules,Distributed_RAM
keywords	No	Keywords of the soft IP. Multiple keywords are separated by “,”. Example: BusType_AHB,BusType_APB
min_radiant_version	Yes	The minimum Radiant version, which supports the soft IP. Example: 1.0, no service pack; 1.0.1, with service pack.
max_radiant_version	No	The maximum Radiant version, which supports the soft IP. Example: 2.0.
supported_products	No	FPGA products supported by the soft IP.
type	No	Enhancement for CPU IP.
min_esi_version	No	The minimum ESI version, which supports the soft IP. Enhancement for esi.
max_esi_version	No	The maximum ESI version, which supports the soft IP. Enhancement for esi.
Supported_platforms	No	Default is Lattice Radiant Software, specific for esi.
User_guides	No	The User guides for IP.

2. <settings> node: describes parameters information that should contain one or more <setting> nodes. In an IP instance package, Verilog parameters are used to configure the soft IP. All user configurable parameters should be added to the <settings> section as <setting> nodes. Beside parameters, you can add <setting> nodes for user-input only. Table 2.4 shows attributes of <setting> nodes.

© 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.

Table 2.4. Attributes of Setting Nodes

Attribute	Value	Mandatory	Description
id	String	Yes	The unique ID of the setting, which is also referred to as: Parameter name in RTL codes Python variable name Tcl variable name To make the value of the id valid in Verilog HDL, Python and Tcl, it should consist of only letters, digits, and underscore. The first character should be a letter. Example: id=”num_outputs”
title	String	No	Short title of the setting. If title is not specified, the value of setting is used. Example: title=”Number of Output”
type	param, input, command, verilog_macro	Yes	Type of the setting. A parameter could be a Verilog parameter, user input, command or verilog_macro. Param, input, and verilog_macro settings can be used to compute values of other param and input settings. They only differ in generated files. Param is written out as a Verilog parameter value of the IP module, verilog_macro is translated to a Verilog macro definition by the define compiler directives, while a parameter with the type of command is shown as a button. Example: type=”param”
value_type	bool, string, int, float, path	Yes	Type of the value. Supported types are bool, int, float, string, and path. The int type supports unlimited precision. The float type supports the precision of float type of C programming language. Refer to the Characteristics of Floating Types section of the 1999 ISO/IEC C Standard for details. The path type indicates a string which represents a path. / is used as separator. Example: value_type=”int”
default	Python expression	No	Default value of the setting. If the setting has no default attribute but has the options attribute, the first option is picked as default value. If the setting has neither default attribute nor options attribute, the initial value of setting is set to 0 for int, 0.0 for float, “” for string, and False for bool. Example: default=”1.0”
value_expr	Python expression	No	Python expression to compute the value of the setting. The result is used as the parameter value if the setting is not editable. For example, divider is calculated by frequencies. Example: value_expr=” int(round((sys_clock_freq * 250.0) / i2c_left_desired_frequency))-1”
options	Python list or list of tuples	No	Candidate options for the setting, which is used in the GUI to display a drop-down selector. It can be set to a simple list or a list of tuples. If it is a simple list, elements are displayed and written. If it is a list of tuples, the first item in the tuple is displayed and the second item in the tuple is written. Example: options=”[0.1, 0.2, 0.5, 1.0]”

© 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.

Attribute	Value	Mandatory	Description
output_formatter	str, nostr	No	Controls how parameter values are written in output RTL files. The following formatters are supported. str: parameter values are written as strings. nostr: quotation marks of strings are removed. Example: output_formatter=”str”
bool_value_mapping	Python tuple or list with 2 string elements	No	The map-to-map bool values to dedicated strings. By default, bool values are written as 1, 0. Example: bool_value_mapping=”(‘True’, ‘False’)”
editable	Python expression	No	Python expression to determine if the setting is editable. When a setting is not editable, it is greyed out in the GUI display and its value is computed by value_expr. Otherwise, the user input is used. Example: editable=”(FEEDBACK_PATH == ‘PHASE_AND_DELAY’) FEEDBACK_PATH is a setting ID in metadata.xml.
hidden	True	No	Python expression to determine if the setting is hidden in GUI. If hidden is set to True, the item is hidden in GUI. The default is False. The expression is resolved to boolean value after the user setting is changed. Example: hidden=”True”
drc	Python expression	No	Python expression to do DRC on the setting. True means DRC pass. Example: drc=”check_valid_addr_pre(I2C_LEFT_ADDRESSING_PRE,i2c_left_addressing_width)” (check_valid_addr_pre is defined in plugin.py setting ID: I2C_LEFT_ADDRESSING_PRE i2c_left_addressing_width)
regex	Regular expression	No	Regular expression to do DRC on the value. For example, the value should be prefixed by 0b. Example: regex=”0b[01]+”
value_range	Python tuple or list with 2 comparable elements	No	Valid range of setting value, which is used to do DRC on the setting. The maximum value can be infinity, float(‘inf’). Example: value_range=”(0, 1023) if(i2c_right_enable) else (-9999, 9999)”
config_groups	Python expression	No	A name or names to group settings together. It is copied from the IP-XACT configGroups attribute and only supports the System Builder value. If it is defined, related RTL parameter is brought out to IP instance top module, so that System Builder can re-define its value.
description	String	No	Detailed description of the setting.
group1	String	No	Groups the settings. Settings of the same group1 is displayed as sub-items under a group item on GUI. The settings with the same group1 should be written continuously if you want the GUI to group them under one group item. Otherwise, you can see multiple groups with the same name in the GUI. Example: group1= “Output Setting”
group2	String	No	Groups the group1 groups. Group1 groups of the same group2 is displayed in a separate page in the GUI. So, a two-level hierarchy is supported in GUI display. Unlike group1, you need not write setting nodes with the same group2 continuously. You must follow the rule for group1 that all the settings with the same group1 must be in the same group2.

© 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.

Attribute	Value	Mandatory	Description
Macro_name	id, value	No	Specifies how to name the Verilog macro in verilog_macro type setting item, the ID or value of this setting item. The default value is setting, which means the setting ID is defined as a Verilog macro. If it is set as value, the evaluated setting value is the Verilog macro. Example: maro_name= “value” Result: `define <setting value> Note: This is only be considered in the setting item whose value_type attribute is set to string.

3. <ports> node: IP module package has some ports in its implementation. These ports should be described in <ports> section as <port> child nodes. If an input port is stuck to fixed value, or an output port is dangling, the port is not used and is invisible after generation. Table 2.5 shows the attributes of <port> nodes.

Table 2.5. Attributes of Port Nodes

Attribute	Value	Mandatory	Description
name	Valid Verilog port name	Yes	Name of a port. Example: name=”Clk”
dir	in, out, inout	Yes	Direction of a port. Example: dir=”in”
range	Python tuple or list with 2 int elements	No	Range of this port. It should be a Python expression whose evaluation result is a tuple or array with two elements. Example: range=”(A_WDT-1, 0)” A_WDT is a setting ID.
conn_mod	Valid Verilog module name	Yes	Name of an IP core module to which this port connects. Example: conn_mod=”counter”
conn_port	Valid Verilog module name	No	Name of port of an IP core module to which this port connects. Value of name is used, if conn_port is not specified. Example: conn_port=”Clk”
conn_range	Python tuple or list with 2 int elements	No	Range of conn_port. It should be a Python expression whose evaluation result is a tuple or array with two elements. Example: conn_range=”(A_WDT-1, 0)” A_WDT is a setting ID.
stick_high	Python expression	No	Python script. True: tie this port to 1. Example: stick_high=”True”
stick_low	Python expression	No	Python script. True: tie this port to 0. Example: stick_low=”no_seq_pins()” no_seq_pins is defined in plugin.py.
stick_value	Python expression	No	Python script. Tie port to the evaluation result of this attribute.
dangling	Python expression	No	Python script. True: keep this port unconnected. Example: dangling=”not USE_COUT” USE_COUT is a setting ID.

© 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.

Attribute	Value	Mandatory	Description
bus_interface	Valid bus interface name	No	Bus interface name defined in <busInterfaces> node. Example: bus_interface=” ahb_slave_0”
attribute	Python expression	No	Python script. The value is written to the .v file as the attribute of the port. Example: attribute=”’(* AAA, BBB=1 *)’” => (* AAA, BBB=1*) input PORT;
port_type	String	No	Data, reset, and clock are valid values. The default value is data. Port_type is passed to the IPXact component.xml as a lsccip:isClk or lssccip:isRst node in venderExtensions in component/model/ports/port.

4. <busInterfaces> node: contains a list of all interface ports of the soft IP (Figure 2.52). The description follows IP-XACT format. Refer to IEEE 1685-2014: IEEE Standard for IP-XACT, Standard Structure for Packaging, Integrating, and Reusing IP within Tool Flows for more details. The busInterface has a list of portMap, which defines the logicalPort name and physicalPort name. In <ports> nodes, one port can have an optional bus_interface attribute, which binds the port with a busInterface. If one port is bound to the busInterface, its name should match the physicalPort name of one portMap of the busInterface, so that port and busInterface/portMap is bound together. Whether or not a port can be used is based on user configuration. If the port is used, the corresponding portMap in busInterface is written to the output IP-XACT file. Otherwise, the corresponding portMap in busInterface is not written to the output IP-XACT file.

© 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.

```
<lsccip:busInterfaces>
    <lsccip:busInterface>
        <lsccip:name>AHBL_S00</lsccip:name>
        <lsccip:displayName>AHBL_S00</lsccip:displayName>
        <lsccip:description>AHB-Lite slave port</lsccip:description>
        <lsccip:busType library="interface" name="ahblite"
vendor="lattice" version="1.0"/>
        <lsccip:abstractionTypes>
            <lsccip:abstractionType>
                <lsccip:abstractionRef library="interface"
name="ahblite_rtl" vendor="lattice" version="1.0"/>
                <lsccip:portMaps>
                    <lsccip:portMap>
                        <lsccip:logicalPort>
                            <lsccip:name>HSEL</lsccip:name>
                        </lsccip:logicalPort>
                        <lsccip:physicalPort>

<lsccip:name>ahbl_s00_hsel_slv_i</lsccip:name>
                        </lsccip:physicalPort>
                    </lsccip:portMap>
                    <lsccip:portMap>
                        <lsccip:logicalPort>
                            <lsccip:name>HADDR</lsccip:name>
                        </lsccip:logicalPort>
                        <lsccip:physicalPort>

<lsccip:name>ahbl_s00_haddr_slv_i</lsccip:name>
                        </lsccip:physicalPort>
                    </lsccip:portMap>
                </lsccip:portMaps>
            </lsccip:abstractionType>
        </lsccip:abstractionTypes>
        <lsccip:slave>
            <lsccip:memoryMapRef memoryMapRef="ahbs_mem_map"/>
        </lsccip:slave>
    </lsccip:busInterface>
</lsccip:busInterfaces>
```

Figure 2.52. Example of busInterface Node

© 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. <addressSpaces> node: specifies the addressable area seen by bus interfaces of type master (Figure 2.53). The description follows the IP-XACT format. Refer to IEEE 1685-2014: IEEE Standard for IP-XACT, Standard Structure for Packaging, Integrating, and Reusing IP within Tool Flows for more details.

```
<lsccip:addressSpaces>
   <lsccip:addressSpace>
      <lsccip:name>ahbm_addr_space</spirit:name>
      <lsccip:range>4k</lsccip:range>
      <lsccip:width>32</lsccip:width>
   </lsccip:addressSpace>
</lsccip:addressSpaces>
```

Figure 2.53. Example of addressSpaces Node

6. <memoryMaps> node: specifies the information about the range of registers, memory, or other address blocks accessible through the subordinate interface (Figure 2.54). The description follows the IP-XACT format. Refer to IEEE 1685-2014: IEEE Standard for IP-XACT, Standard Structure for Packaging, Integrating, and Reusing IP within Tool Flows for more details and examples.
   To represent dynamic availability of some elements, <register> and <field> nodes are extended with an optional isPresent element that defines whether the enclosing element is present or not, as introduced in IEEE Std 1685-2014. The value of isPresent element is a Python expression. Parameters defined in the <settings> node can be used in the expression.
   If a register is used based on user configuration, the register is written to an output IP-XACT file. Otherwise, the register is not written to the output IP-XACT file.

© 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.

```
<lsccip:memoryMaps>
    <lsccip:memoryMap>
        <lsccip:name>ahbs_mem_map</lsccip:name>
        <lsccip:description>AHB-Lite Slave 0 memory map</lsccip:description>
        <lsccip:addressBlock>
            <lsccip:name>registers</lsccip:name>
            <lsccip:displayName>registers</lsccip:displayName>
            <lsccip:description>Register Block</lsccip:description>
            <lsccip:baseAddress>0</lsccip:baseAddress>
            <lsccip:range>4096</lsccip:range>
            <lsccip:width>32</lsccip:width>
            <lsccip:usage>register</lsccip:usage>
            <lsccip:access>read-write</lsccip:access>
            <lsccip:register>
                <lsccip:name>Status</lsccip:name>
                <lsccip:displayName>Status Register</lsccip:displayName>
                <lsccip:description>Status Register</lsccip:description>
                <lsccip:addressOffset>0x10</lsccip:addressOffset>
                <lsccip:size>4</lsccip:size>
                <lsccip:volatile>true</lsccip:volatile>
                <lsccip:access>read-only</lsccip:access>
                <lsccip:field>
                    <lsccip:name>FIFO_Empty</lsccip:name>
                    <lsccip:displayName>FIFO_Empty</lsccip:displayName>
                    <lsccip:description>Indicates current status of the
interface in the receive direction:  0 - There is data available.  1 - The FIFO
is empty.</lsccip:description>
                    <lsccip:isPresent> 1 != int_setting </lsccip:isPresent>
                    <lsccip:bitOffset>0</lsccip:bitOffset>
                    <lsccip:bitWidth>1</lsccip:bitWidth>
                    <lsccip:volatile>true</lsccip:volatile>
                    <lsccip:access>read-only</lsccip:access>
                    <lsccip:writeValueConstraint>
                        <lsccip:minimum>0</lsccip:minimum>
                        <lsccip:maximum>0</lsccip:maximum>
                    </lsccip:writeValueConstraint>
                    <lsccip:testable
testConstraint="unConstrained">false</lsccip:testable>
                </lsccip:field>
            </lsccip:register>
        </lsccip:addressBlock>
</lsccip:memoryMap>
</lsccip:memoryMaps>
```

Figure 2.54. Example of memoryMaps Node

© 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. <componentGenerators> node: contains a list of componentGenerator elements. Each componentGenerator element defines a generator that is run on generated IP instance package. Each generator is called after other IP instance package files are generated. For example, a component generator can generate memory initialization file for a memory IP (Figure 2.55).

```
<lsccip:componentGenerators>
  <lsccip:componentGenerator>
    <lsccip:name>memGenerator</lsccip:name>
    <lsccip:generatorExe>script/mem_gen.py</lsccip:generatorExe>
  </lsccip:Generator>
</lsccip:Generators>
```

Figure 2.55. Example of componentGenerators Node

8. <estimatedResources> node: contains a list of estimatedResource element. Each estimatedResource element defines the formula to calculate one type of resource used in the IP instance package. Table 2.6 shows the elements in the < estimatedResource> node.

Table 2.6. Elements in estimatedResources Node

Element	Value	Mandatory	Description
name	String	Yes	Name of the resource.
Number	Python expression	Yes	Python script to calculate the number of the resource.

9. <outFileConfigs> node: specifies all customized output file configuration nodes <fileConfig> for the whole customized flow. FileConfig node contains a group of attributes to specify a specific file or directory generation.

A full description of a soft IP might be large. Metadata.xml supports to build a large XML file from small manageable chunks. The approach is implemented by Xinclude (Figure 2.56).

metadata.xml

```
<lsccip:ip version="1.0"
xmlns:lsccip="http://www.latticesemi.com/XMLSchema/Radiant/ip"
xmlns:xi="http://www.w3.org/2001/XInclude">
    <lsccip:general> … </lsccip:general>
<xi:include href="setting.xml" parse="xml" />
<xi:include href="memory_map.xml" parse="xml" />
<xi:include href="address_space.xml" parse="xml" />
<xi:include href="bus_interface.xml" parse="xml" />
<lsccip:ports> … </lsccip:ports>
</lsccip:ip>
```

setting.xml

```
<lsccip:settings xmlns:lsccip="http://www.latticesemi.com/XMLSchema/Radiant/ip">
<lsccip:setting id="int_setting" type="input" value_type="int"
conn_mod="example" default="1"   title="Integer Setting" />
<lsccip:setting id="bool_with_value_mapping" type="param"
value_type="bool" conn_mod="example"
              default="False" bool_value_mapping="('Enabled', 'Disabled')" />
</lsccip:settings>
```

Figure 2.56. Example of Xinclude Usage

© 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 can use the metadata.xsd in Appendix A. to check the metadata file. For those elements borrowed from IPXact, refer to the related xsd files of IPXact.

2.3.2. Implementation RTL Files

You should put all IP implementation RTL files in the rtl sub-directory of the whole IP package. The Lattice Propel Builder platform does not support hierarchical rtl directories. All RTL sources can be put in one directory level. Three file suffix names, .v, .sv, and .vhd are supported for RTL files to represent Verilog, SystemVerilog, and VHDL language types accordingly.

The IP Platform always generates a wrapper module by default for the whole IP generation. This wrapper is in Verilog HDL no matter what the language type of the IP implementation RTLs is. Providing a Verilog-HDL top to handle the interface translation among different language types is strongly recommended.

If the IP is implemented in Verilog HDL only, including SystemVerilog, all the contents are copied in one .v/.sv RTL file and modules are defined in the wrapper top module. All module names follow a unified naming rule to avoid name collision such as Top module name, PMI module name, Primitive module name, Blackbox module name.

If some portions of an IP module in the RTL file are encrypted, the corresponding portions generated in Verilog file are also encrypted.

All the RTLs are kept and copied to an IP instance directory, if there is VHDL in IP implementation RTLs. Therefore, the module-naming rule does not need to be unified. You can specify different lib for VHDL source that is specified in the wrapper file generator (Figure 2.57).

```
<lsccip:outFileConfigs>
      <lsccip:fileConfig name="wrapper" lib='test2.vhd=libA;test3.vhd=libB' />
</lsccip:outFileConfigs>
```

Figure 2.57. Example of Specifying Lib for VHDL

2.3.3. Python Script Plugin File

Python expressions can be used in metadata files to implement complex logic. To support complex logic, you can add any python functions in the plugin.py file (Figure 2.58) of an IP package, and then call the functions in Python expressions in the metadata file.

Each setting item value can be referred to in a Python expression in metadata.xml by its ID as a Python variable. An expression is evaluated on demand. The plugin has its individual namespace and all the setting item values are exported to plugin as a global map variable IP_SETTINGS. You can use IP_SETTINGS [item ID] to refer to an item or set its value in the plugin code.

A global variable __PLUGIN_VER is defined with value 2.0. You can check this variable if you want to provide a plugin that can work on both current and legacy IP platforms.

© 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.

```
def cntr_opt():
  #return {"Down"  :0,
  #        "Up"    :1,
  #        "UpDown":2}
  return [("Down"  , 0),
          ("Up"    , 1),
          ("UpDown", 2)]


def cntr_ldir():
  return ((CNTR_DIR == 0) | (CNTR_DIR == 1))


def cntr_wdt():
  if (CNTR_WIDTH < 2):
    return 2
  else:
    return CNTR_WIDTH


def cntr_hval_check():
  if (CNTR_HVALUE <= CNTR_LVALUE):
    ret = 0
    PluginUtil.post_error("Higher count value should be greater than the Lower count value.")
  else:
    ret = 1
  return ret


def get_device_name(value):
    x = runtime_info.device_info.architecture(value)
    return x
```

Figure 2.58. Template of Plugin File

© 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.

2.3.4. Memory Map CSV File

Lattice IP Packager 2026.1 supports importing memory maps from a CSV file to create memory maps. You can edit a CSV file. Figure 2.59 shows the CSV file template. Each row to be imported is with one of the keywords among MEMORYMAP, REGISTER, and FIELD. A row with no keyword is ignored, such as the header row (orange squared part in Figure 2.59). The header row is to help identify the column.

	A	B	C	D	E	F	G	H	I	J
1	0	1	2	3	4	5		7	8	9
2		name	description	baseAddress	range	width				
3	MEMORYMAP	ABC	desc1	0x0	32	64				
4		name	displayName	description	addressOffset	size	volatile	access		
5	REGISTER	QSPI_CTRL	QSPI_CTRL	QSPI Control Register	0x00	32	TRUE	read-write		
6		field	displayName	description	bitOffset	bitWidth	volatile	access	default	testable
7	FIELD	spi_mode	spi_mode	Sets the SPI mode	0	2	TRUE	read-write	DEFAULT_SPI_MODE	TRUE
8	FIELD	sck_div	sck_div	Sets the SPI clock divider	2	3	TRUE	read-write	DEFAULT_SCK_DIV	TRUE
9	FIELD	reserved	reserved	Reserved bits	5	26	TRUE	read-write	0	TRUE
10	FIELD	soft_reset	soft_reset	Resets internal soft logic	31	1	TRUE	read-write	0	TRUE
11		name	displayName	description	addressOffset	size	volatile	access		
12	REGISTER	CMD_DATA	CMD_DATA	Command Data Register	0x04	32	TRUE	read-write		
13		field	displayName	description	bitOffset	bitWidth	volatile	access	default	testable
14	FIELD	cmd_data	cmd_data	Command data to transmit in transaction phase 1 (always big endian)	0	32	TRUE	read-write	0	TRUE
15		name	displayName	description	addressOffset	size	volatile	access		
16	REGISTER	TX_FIFO_DATA	TX_FIFO_DATA	Tx FIFO Data Register	0x08	32	TRUE	write-only		
17		name	displayName	description	bitOffset	bitWidth	volatile	access	default	testable
18	FIELD	tx_fifo_data	tx_fifo_data	Data to transmit in transaction phase 2.	0	32	TRUE	write-only	NA	FALSE

Figure 2.59. Example of Memory Map CSV File

© 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. TCL Commands

Lattice IP Packager 2026.1 provides TCL commands to execute actions. You can manually enter TCL commands in the Tcl Console (Figure 3.1).

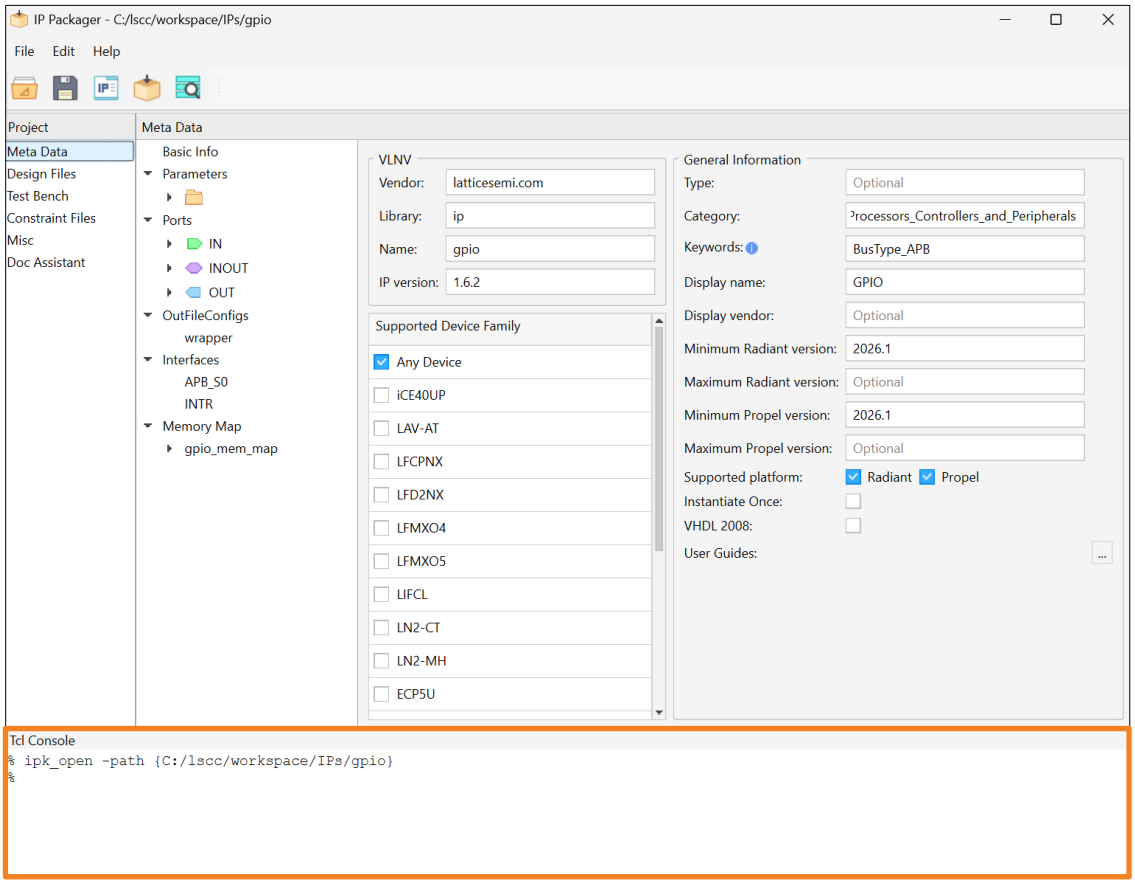

Figure 3.1. Tcl Console

An IP core typically contains the following types of objects:

- Parameter
- Parameters are IP settings that can be used in Ports/Interfaces/Memory map configuration.
- Port
- Ports are pins of an instantiated IP.
- Interface
- Interfaces of an instantiated IP are composed of ports.
- Memory map
- A memory map is associated with a target which contains an address block specifying the base address and the range of the target segment.
- Address block
- An address block describes a single, contiguous block of memory that is part of a memory map.
- Register
- A register element describes a register in an address block or register file.
- The following commands for the above objects are designed to be used through the IP Packager GUI interface.

© 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.

3.1. ipk_open

Opens an existing IP or an empty folder to create a new IP.

Usage	`ipk_open -path <ip path>`

3.2. ipk_save

Saves the current IP.

Usage	`ipk_save`

3.3. ipk_close

Closes the current IP.

Usage	`ipk_close`

3.4. ipk_drc

Runs the IP rule checker on the current IP.

Usage	`ipk_drc`

3.5. ipk_package

Packages the current IP to an ipk file.

Usage	`ipk_package`

3.6. ipk_add_param

Creates a new parameter.

Usage	`ipk_add_param`
	`-name <parameter name>`
	`-type <parameter type>`
	`-value_type <value type>`
	`[-tab <tab name>]`
	`[-group <group name>]`
	-type: input, param, command, verilog_macro
	-value_type: string, bool, int, float, 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.

3.7. ipk_add_port

Creates a new port. You must specify the direction.

Usage	`ipk_add_port -dir <in/out/inout> -name <port name> [-range <"(MSB, LSB)">]`

3.8. ipk_add_fileconfig

Creates a new fileconfig item. You must specify the fileconfig type.

Usage	`ipk_add_fileconfig -type <type name>` -type: bb, config, template, vhdl_template, wrapper, timing_constraints, timing_constraints_template, synplify_pro_constraints, testbench_parameters_verilog, testbench_instance_verilog, eval_parameters_verilog, eval_instance_verilog, tcl_constraints, IP-XACT_design, IP-XACT_component, ipx

3.9. ipk_add_interface

Creates a new interface. You must specify the role and type.

Usage	`ipk_add_interface`
	`-role <controller/target>`
	`-type <interface vlnv>`
	`-name <interface name>`
	-type: use usage ipk_list_interface_types to get type list.

3.10. ipk_add_mem_map

Creates a new memory map.

Usage	`ipk_add_mem_map -name <memory map name>`

3.11. ipk_add_addr_block

Creates a new address block. You must specify the parent memory map.

Usage	`ipk_add_addr_block`
	`-name <mem_map_name/addr_block_name>`
	`-base_addr <value>`
	`-width <value>`
	`-range <value>`

© 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.

3.12. ipk_add_register

Creates a new register. You must specify the parent address block.

Usage	ipk_add_register -name <mem_map_name/addr_block_name/register_name> -offset <value> -size <value>

3.13. ipk_add_field

Creates a new register field or alter_register field. You must specify the parent address block.

Usage	ipk_add_field (-name <mem_map_name/addr_block_name/register_name/field_name> | -name <mem_map_name/addr_block_name/register_name/alter_register_ name/field_name>) -bit_offset <value> -bit_width <value>

3.14. ipk_add_alter_register

Creates a alter_register field. You must specify the parent address block.

Usage	ipk_add_alter_field -name <mem_map_name/addr_block_name/register_name/alter_ register_name>

3.15. ipk_delete

Deletes an existing parameter/port/interface/memory map/address block/register item.

Usage	ipk_delete -type <param | port | fileconfig | interface | mem_map | addr_block | register> -name <item name>

3.16. ipk_rename

Renames an existing parameter/port/interface/memory map/address block/register item.

Usage	ipk_rename -type <param | port | interface | mem_map | addr_block | register | fileconfig> -name <item name> <new name>

© 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.

3.17. ipk_get_items

Gets item list by type.

Usage	ipk_get_items -type <param | port | interface | mem_map | addr_block | register | fileconfig>

3.18. ipk_get_item_property

Gets the property list of an item.

Usage	ipk_get_item_property -type <param | port | interface | mem_map | addr_block | register | fileconfig> -item_name <item name> [-prop_name <property name>]

3.19. ipk_list_interface_types

Gets the list of supported interface types of IP Packager.

Usage	ipk_list_interface_types

3.20. ipk_set_item_property

Sets the property value of an item.

Usage	ipk_set_item_property -type <param | port | interface | mem_map | addr_block | register | fileconfig> -item_name <item name> -prop_name <property name> -prop_value <value>

3.21. ipk_set_ip_info

Sets general information of the current IP.

Usage	ipk_set_ip_info -type <vendor | library | name | version | type | category | keywords | display_name | display_vendor | min_radiant_version | max_radiant_version | min_propel_version | max_propel_version | instantiate_once | supported_platforms | vhdl_2008> -value <value>

© 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.

3.22. ipk_get_ip_info

Gets general information of the current IP.

Usage	ipk_get_ip_info -type < vendor | library | name | version | type | category | keywords | display_name | display_vendor | min_radiant_version | max_radiant_version | min_propel_version | max_propel_version | instantiate_once | supported_platforms | vhdl_2008> -type: value * can get all items' value.

3.23. ipk_get_interface_ports

Gets port map of an interface.

Usage	ipk_get_interface_ports -interface_name <interface name> [-log_port <logical port name>]

3.24. ipk_set_interface_port

Sets a physical port to the logical port of an interface.

Usage	ipk_set_interface_port -interface_name <interface name> -log_port <logical port name> -phy_port <physical port name>

3.25. ipk_import_mem_map

Imports a csv file for memory map.

Usage	ipk_import_mem_map -file <csv file path>

3.26. ipk_set_library_path

Sets the library path for reference hdl files.

Usage	ipk_set_library_path -path <path>

3.27. ipk_add_file

Adds an HDL file for the current IP.

Usage	ipk_add_file -type < rtl| testbench | doc | driver | eval | plugin | eula |tcl_constraint | ldc_constraint | fdc_constraint> (-file <file path> | -ref_file <file path (only for rtl file)>) [-lib <value(only for vhdl)>]

© 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.

3.28. ipk_get_files

Gets file list from the current IP.

Usage	ipk_get_files -type < rtl | testbench | doc | driver | eval | plugin | eula | tcl_constraint | ldc_constraint | fdc_constraint >

3.29. ipk_get_file_property

Gets properties of files in the current IP.

Usage	ipk_get_file_property -type < rtl | testbench | doc | driver | eval | plugin | eula | tcl_constraint | ldc_constraint | fdc_constraint > -file <file path> -prop_name <property name> -prop_name: value * can get all properties' value.

3.30. ipk_set_file_property

Sets property value of a file in the current IP.

Usage	ipk_set_file_property -type < rtl | testbench | doc | driver | eval | plugin | eula | constraint> -file <file path> -prop_name <property name> -prop_value <property value>

3.31. ipk_remove_file

Removes existing files from the current IP.

Usage	ipk_remove_file -type < rtl | testbench | doc | driver | eval | plugin | eula | tcl_constraint | ldc_constraint | fdc_constraint > -file <file path>

3.32. ipk_add_device

Adds devices for the current IP.

Usage	ipk_add_device -family <family name> [-device <device name>] [-speed <speed name>] [-package <package name>] [-operating <operating name>]

© 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.

3.33. ipk_remove_device

Removes devices for the current IP.

Usage	ipk_remove_device -family <family name> [-device <device name>] [-speed <speed name>] [-package <package name>] [-operating <operating name>]

3.34. ipk_get_device

Gets device information of the current IP.

Usage	ipk_get_device

3.35. ipk_preview

Opens the preview dialog for the current IP.

Usage	ipk_preview

© 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.

Appendix A. metadata.xsd

```
<?xml version="1.0" encoding="gb2312"?>
<!-- IP Metadata Schema 0.0.5 -->
<!--
Change Log:
    0.0.0  21-Jul-2016 initial revision.
    0.0.1  22-Jul-2016 1. removed enum_value from <setting> 2. change value_mapping to
scriptType
    0.0.2  29-Jul-2016 1. removed GUI out of metadata scope
    0.0.3  30-Jul-2016 1. add value list support
    0.0.4  07-Jan-2021 1. new features in Radiant 3.0
    0.0.5  06-Apr-2021 1. fully support Verilog macro
-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://www.latticesemi.com/XMLSchema/Radiant/ip"
           xmlns:lsccip="http://www.latticesemi.com/XMLSchema/Radiant/ip"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
  <xs:import schemaLocation="xml.xsd" namespace="http://www.w3.org/XML/1998/namespace"/>
  <xs:include schemaLocation="busInterface.xsd"/>
  <xs:include schemaLocation="memoryMap.xsd"/>
  <xs:include schemaLocation="file.xsd"/>
  <xs:include schemaLocation="generator.xsd"/>
  <xs:include schemaLocation="design.xsd"/>

  <xs:attributeGroup name="ipany.att">
    <xs:anyAttribute processContents="lax"/>
  </xs:attributeGroup>
  <!-- version string type definition-->
  <xs:simpleType name="threeFigureVersionType">
    <xs:annotation>
      <xs:documentation>
        The syntax for version string.
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:pattern value="[0-9]+\.[0-9]+\.[0-9]+"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="twoFigureVersionType">
    <xs:annotation>
      <xs:documentation>
        The syntax for Radiant version string.
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:pattern value="[0-9]+\.[0-9]+"/>
    </xs:restriction>
  </xs:simpleType>
```

© 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.

```
  <xs:simpleType name="scriptType">
    <xs:annotation>
      <xs:documentation>
        Any python expressions. Could be a simple const variable like &quot;1&quot;,
        or a tuple &quot;(0, 10)&quot;, or a function call &quot;user_func1()&quot;,
        or a complex expression &quot;a==12&quot;
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string"/>
  </xs:simpleType>

  <xs:simpleType name="settingTypeType">
    <xs:annotation>
      <xs:documentation>
       Is this setting variable for parameter of IP core module, verilog macro, or just
for user input?
       Valid values are &quot;input&quot;, &quot;verilog_macro&quot;,
&quot;param&quot;, and &quot;command&quot;.
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:enumeration value="param"/>
      <xs:enumeration value="input"/>
      <xs:enumeration value="verilog_macro"/>
      <xs:enumeration value="command"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="settingMacroNameType">
    <xs:annotation>
      <xs:documentation>
     Specify what is used to name the Verilog macro, ID or Value?
     Valid values are &quot;id&quot; and &quot;value&quot;.
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:enumeration value="id"/>
      <xs:enumeration value="value"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="settingValueType">
    <xs:annotation>
      <xs:documentation>
       Value type of setting variable. Valid types are &quot;bool&quot;,
&quot;string&quot; and &quot;int&quot;
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:enumeration value="bool"/>
      <xs:enumeration value="string"/>
      <xs:enumeration value="int"/>
      <xs:enumeration value="float"/>
```

© 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.

```
      <xs:enumeration value="path"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="settingDefaultType">
    <xs:annotation>
      <xs:documentation>
        Default value of setting variable.
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string"/>
  </xs:simpleType>

  <xs:simpleType name="settingOutputFormatterType">
    <xs:annotation>
      <xs:documentation>
        Formatter of output.
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:enumeration value="str"/>
      <xs:enumeration value="nostr"/>
    </xs:restriction>
  </xs:simpleType>

  <!-- ip.settings.setting-->
  <xs:complexType name="settingElementType">
      <xs:annotation>
            <xs:documentation>
                  Setting variable definition.
            </xs:documentation>
      </xs:annotation>

      <xs:attribute name="id" type="xs:string" use="required" />
      <xs:attribute name="type" type="lsccip:settingTypeType"
            use="required" />
      <xs:attribute name="value_type" type="lsccip:settingValueType"
            use="required" />
      <xs:attribute name="conn_mod" type="xs:string" use="required" />
      <xs:attribute name="domain" type="xs:string" use="optional" />
      <xs:attribute name="default" type="lsccip:settingDefaultType"
            use="optional" />
      <xs:attribute name="value_expr" type="lsccip:scriptType"
            use="optional" />
      <xs:attribute name="drc" type="lsccip:scriptType" use="optional" />
      <xs:attribute name="editable" type="lsccip:scriptType"
            use="optional" />
      <xs:attribute name="description" type="xs:string" use="optional" />
      <xs:attribute name="title" type="xs:string" use="optional" />
      <xs:attribute name="hidden" type="xs:string" use="optional" />
      <xs:attribute name="regex" type="xs:string" use="optional" />
      <xs:attribute name="options" type="lsccip:scriptType"
```

© 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.

```
            use="optional" />
      <xs:attribute name="value_range" type="lsccip:scriptType"
            use="optional" />
      <xs:attribute name="group1" type="xs:string" use="optional" />
      <xs:attribute name="group2" type="xs:string" use="optional" />
      <xs:attribute name="bool_value_mapping" type="lsccip:scriptType"
            use="optional" />
      <xs:attribute name="output_formatter"
            type="lsccip:settingOutputFormatterType" use="optional" />
      <xs:attribute name="config_groups" type="lsccip:scriptType"
            use="optional" />
      <xs:attribute name="process_path" type="xs:boolean" use="optional"
            default="true">
      </xs:attribute>
      <xs:attribute name="macro_name" type="lsccip:settingMacroNameType"
use="optional"
          default="id">
      </xs:attribute>
    <xs:attribute ref="xml:base" use="optional"/>
    <xs:attribute name="no_dependency" type="xs:string" use="optional" />
  </xs:complexType>

  <!-- ip.settings -->
  <xs:complexType name="settingsType">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="setting"
type="lsccip:settingElementType" />
    </xs:sequence>
    <xs:attribute ref="xml:base" use="optional"/>
  </xs:complexType>

  <!-- ip.ports.port-->
  <xs:complexType name="portElementType">
      <xs:annotation>
            <xs:documentation>IP port definition.</xs:documentation>
      </xs:annotation>
      <xs:attribute name="name" type="xs:string" use="required" />
      <xs:attribute name="dir" type="xs:string" use="required" />
      <xs:attribute name="conn_mod" type="xs:string" use="required" />
      <xs:attribute name="conn_port" type="xs:string" use="optional" />
      <xs:attribute name="conn_range" type="xs:string" use="optional" />
      <xs:attribute name="range" type="lsccip:scriptType"
            use="optional" />
      <xs:attribute name="stick_high" type="lsccip:scriptType"
            use="optional" />
      <xs:attribute name="stick_low" type="lsccip:scriptType"
            use="optional" />
      <xs:attribute name="stick_value" type="lsccip:scriptType"
            use="optional" />
      <xs:attribute name="dangling" type="lsccip:scriptType"
            use="optional" />
      <xs:attribute name="bus_interface" type="xs:string"
```

© 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.

```
            use="optional" />
      <xs:attribute name="attribute" type="lsccip:scriptType"
            use="optional" />
      <xs:attribute name="port_type" use="optional" default="data">
            <xs:simpleType>
                  <xs:restriction base="xs:string">
                        <xs:enumeration value="data"></xs:enumeration>
                        <xs:enumeration value="reset"></xs:enumeration>
                        <xs:enumeration value="clock"></xs:enumeration>
                  </xs:restriction>
            </xs:simpleType>
      </xs:attribute>
    <xs:attribute ref="xml:base" use="optional"/>
  </xs:complexType>
  <!-- ip.outFileConfigs -->
  <xs:complexType name="outFileConfigsType">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" name="fileConfig" type="lsccip:fileConfigType"
/>
    </xs:sequence>
  </xs:complexType>
  <!-- ip.outFileConfigs.fileConfig -->
  <xs:complexType name="fileConfigType">
      <xs:annotation>
            <xs:documentation>Configure output file</xs:documentation>
      </xs:annotation>
      <xs:attribute name="name" type="xs:string" use="required" />
      <xs:attribute name="use" use="optional" default="merge">
            <xs:annotation>
                  <xs:documentation>
                        Merge the new attributes to existing configuration or
                        replace all
                  </xs:documentation>
            </xs:annotation>
            <xs:simpleType>
                  <xs:restriction base="xs:string">
                        <xs:enumeration value="merge"></xs:enumeration>
                        <xs:enumeration value="replace"></xs:enumeration>
                  </xs:restriction>
            </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="type" use="optional" default="file">
            <xs:annotation>
                  <xs:documentation>
                        Indicate the item is generate a file or directory
                  </xs:documentation>
            </xs:annotation>
            <xs:simpleType>
                  <xs:restriction base="xs:string">
                        <xs:enumeration value="file"></xs:enumeration>
                        <xs:enumeration value="directory"></xs:enumeration>
                  </xs:restriction>
```

© 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.

```
            </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="description" type="xs:string"
            use="optional">
      </xs:attribute>
      <xs:attribute name="enable_output" type="xs:string" use="optional"
            default="True">
            <xs:annotation>
                  <xs:documentation>
                        Python expression represent a boolean value to
                        indicate the output is enabled or disabled
                  </xs:documentation>
            </xs:annotation>
      </xs:attribute>
      <xs:attribute name="phase" type="xs:int" use="optional"
            default="0">
      </xs:attribute>
      <xs:attribute name="file_base_name" type="xs:string"
            use="optional">
      </xs:attribute>
      <xs:attribute name="file_suffix" type="xs:string"
            use="optional">
      </xs:attribute>
      <xs:attribute name="sub_dir" type="xs:string" use="optional"></xs:attribute>
      <xs:attribute name="file_generator" type="xs:string"
            use="optional">
      </xs:attribute>
      <xs:attribute name="src_dir" type="xs:string"
          use="optional">
      </xs:attribute>
      <xs:attribute name="dest_dir" type="xs:string"
        use="optional">
    </xs:attribute>
    <xs:attribute name="src_file" type="xs:string"
        use="optional">
    </xs:attribute>
    <xs:attribute name="export" use="optional" default="input_only">
        <xs:annotation>
            <xs:documentation>
                Indicate export all setting items or input only
            </xs:documentation>
        </xs:annotation>
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="all"></xs:enumeration>
                <xs:enumeration value="input_only"></xs:enumeration>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
      <xs:attributeGroup ref="lsccip:ipany.att"/>
  </xs:complexType>
  <!-- ip.interface-->
```

© 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.

```
  <xs:complexType name="portsType">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="1" name="port"
type="lsccip:portElementType" />
    </xs:sequence>
    <xs:attribute ref="xml:base" use="optional"/>
  </xs:complexType>

  <!-- ip.interface-->
  <xs:complexType name="generalType">
    <xs:sequence>
      <xs:element maxOccurs="1" minOccurs="0" name="vendor"
            type="xs:string" />
      <xs:element maxOccurs="1" minOccurs="0" name="library"
            type="xs:Name" />
      <xs:element maxOccurs="1" minOccurs="1" name="name"
            type="xs:string" />
      <xs:element maxOccurs="1" minOccurs="0" name="display_name"
            type="xs:string" />
      <xs:element maxOccurs="1" minOccurs="1" name="version"
            type="xs:string" />
      <xs:element maxOccurs="1" minOccurs="1" name="category"
            type="xs:string" />
      <xs:element maxOccurs="1" minOccurs="0" name="keywords"
            type="xs:string" />
      <xs:element maxOccurs="1" minOccurs="0" name="type"
            type="xs:string" />
      <xs:element maxOccurs="1" minOccurs="0" name="instantiatedOnce"
            type="xs:boolean" />
      <xs:element maxOccurs="1" minOccurs="1"
            name="min_radiant_version" type="lsccip:twoFigureVersionType" />
      <xs:element maxOccurs="1" minOccurs="0"
            name="max_radiant_version" type="lsccip:twoFigureVersionType" />
      <xs:element maxOccurs="1" minOccurs="0" name="min_esi_version"
            type="lsccip:twoFigureVersionType" />
      <xs:element maxOccurs="1" minOccurs="0" name="max_esi_version"
            type="lsccip:twoFigureVersionType" />
      <xs:element maxOccurs="1" minOccurs="1"
            name="supported_products" type="lsccip:supportedProductsType" />
      <xs:element maxOccurs="1" minOccurs="0"
            name="supported_platforms" type="lsccip:supportedPlatformsType" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="supportedPlatformsType">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="1" name="supported_platform"
type="lsccip:supportedPlatformType" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="supportedPlatformType">
```

© 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.

```
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="supported_templates"
type="lsccip:supportedTemplatesType" />
    </xs:sequence>
    <xs:attribute name="name" type="xs:string"/>
  </xs:complexType>

  <xs:complexType name="supportedTemplatesType">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="supported_template"
type="lsccip:supportedTemplateType" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="supportedTemplateType">
    <xs:attribute name="processor" type="xs:string"/>
    <xs:attribute name="family" type="xs:string"/>
    <xs:attribute name="name" type="xs:string"/>
    <xs:attribute name="version" type="xs:string" use="optional"/>
  </xs:complexType>

  <xs:complexType name="supportedProductsType">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="1" name="supported_family"
type="lsccip:supportedFamilyType" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="supportedFamilyType">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="supported_device"
type="lsccip:supportedDeviceType" />
    </xs:sequence>
    <xs:attribute name="name" type="xs:string"/>
  </xs:complexType>

  <xs:complexType name="supportedDeviceType">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="supported_speed_grade"
type="lsccip:supportedSpeedType" />
    </xs:sequence>
    <xs:attribute name="name" type="xs:string"/>
  </xs:complexType>

  <xs:complexType name="supportedSpeedType">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="supported_package"
type="lsccip:supportedPackageType" />
    </xs:sequence>
    <xs:attribute name="name" type="xs:string"/>
  </xs:complexType>
```

© 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.

```
  <xs:complexType name="supportedPackageType">
    <xs:attribute name="name" type="xs:string"/>
  </xs:complexType>

  <!-- ip.estimatedResources -->
  <xs:complexType name="estimatedResourcesType">
    <xs:annotation>
      <xs:documentation>
          IP estimated resources definition.
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="estimatedResource" type="lsccip:estimatedResourceType"
minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="estimatedResourceType">
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="number" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>

  <!-- ip -->
  <xs:element name="ip">
    <xs:complexType>
      <xs:sequence>
       <xs:element name="general" type="lsccip:generalType" />
       <xs:element name="settings" type="lsccip:settingsType" />
       <xs:element name="ports" type="lsccip:portsType" />
       <xs:element name="outFileConfigs" type="lsccip:outFileConfigsType" maxOccurs="1"
minOccurs="0">
          <xs:unique name="fileCfgKey">
            <xs:selector xpath="lsccip:fileConfig"/>
            <xs:field xpath="@name"/>
          </xs:unique>
       </xs:element>
       <xs:element ref="lsccip:busInterfaces" minOccurs="0" />
       <xs:element ref="lsccip:addressSpaces" minOccurs="0" />
       <xs:element ref="lsccip:memoryMaps" minOccurs="0" />
       <xs:element ref="lsccip:componentGenerators" minOccurs="0" />
       <xs:element ref="lsccip:choices" minOccurs="0" />
       <xs:element ref="lsccip:fileSets" minOccurs="0" />
       <xs:element ref="lsccip:design" minOccurs="0" maxOccurs="unbounded"/>
       <xs:element name="estimatedResources"
              type="lsccip:estimatedResourcesType" minOccurs="0" />
       <xs:element ref="lsccip:parameters" minOccurs="0" />
      </xs:sequence>
      <xs:attribute name="version" type="lsccip:twoFigureVersionType"/>
      <xs:attribute name="platform" type="xs:string" use="optional"/>
```

© 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.

```
      <xs:attribute name="platform_version" type="lsccip:twoFigureVersionType"
use="optional"/>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

© 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 Propel 2026.1 Builder User Guide (FPGA-UG-02254)
- Lattice Diamond Online Help
- Lattice Radiant Online Help
- Lattice Insights for Lattice Semiconductor Training Series 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.

© 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, June 2026

Section	Change Summary
All	Production 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