Lattice Propel 2023.1 Builder

User Guide

FPGA-UG-02185-1.0

June 2023

Disclaimers

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

Contents

Figures

Tables

Glossary

A list of words or terms specialized in this document.

Glossary	Definition
BSP	Board Support Package, the layer of software containing hardware-specific drivers and libraries to function in a particular hardware environment.
CPU	Central Processing Unit
CSV	Comma Separated Values file
DRC	Design Rule Check
DUT	Design Under Test
ESI	Previous name of Propel
FPGA	Field Programmable Gate Array
GUI	Graphic User Interface
HDL	Hardware Description Language
HSM	Hardware Security Module
IDE	Integrated Development Environment
IP-XACT	An XML format that defines and describes electronic components and their designs.
IPX	Internet Packet Exchange
LHS	Left Hand Side
LSB	Least Significant Bit
MSB	Most Significant Bit
OS	Operating System
Perspective	A group of views and editors in the Workbench window.
PGE	Package Generate Engine
Programmer	A tool can program Lattice FPGA SRAM and external SPI Flash through various interfaces, such as JTAG, SPI, and I^2C.
RHS	Right Hand Side.
RISC-V	A free and open instruction set architecture (ISA) enabling a new era of processor innovation through open standard collaboration.
SBX	The files that store the spatial index of the features
SDK	Embedded System Design and Develop Kit. A set of software development tools that allows the creation of applications for software package on the Lattice embedded platform.
SoC	System on Chip. An integrated circuit that integrates all components of a computer or other electronic systems.
SRAM	Static Random Access Memory
TCL	Tool Command Language
UFM	User Flash Memory
VIP	Verification IP
Workspace	The directory where stores your work, which is used as the default content area for your projects as well as for holding any required metadata.
Workbench	The desktop development environment in Eclipse IDE platform.

1. Introduction

Lattice Propel™ 2023.1 Builder is a graphical tool used to assemble complex System-on-Chip (SoC) modules which can be used in the supported Lattice FPGA devices. These modules and/or IP can be assembled and connected easily by simply dragging and dropping the modules and/or IP into the Schematic Window.

1.1. Purpose

Embedded system solutions play an important role in FPGA system design allowing you to develop the software for a processor in an FPGA device. It provides flexibility for you to control various peripherals from a system bus.

To develop an embedded system on an FPGA, you need to design the System on Chip (SoC) with an embedded processor. Lattice Propel Builder helps you develop your system with a RISC-V processor, peripheral IP, and a set of tools by a simple drag-and-drop.

The purpose of this document is to introduce Lattice Propel 2023.1 Builder tool and design flow to help you quickly get started to build a small demo system. You can also find the recommended flows of using Lattice Propel Builder in this document.

1.2. Audience

The intended audience for this document includes embedded system designers and embedded software developers using Lattice MachXO3D, MachXO3L, MachXO3LF, MachXO2, LIFCL, LFD2NX, LFMNX, LFCPNX, LFMXO5, and LAV-AT devices. The technical guidelines assume readers have expertise in the embedded system area and FPGA technologies.

2. Lattice Propel Builder Design

The Propel Builder design includes creating a SoC project design, and a verification design, which are discussed in detail in the following sections.

2.1. Builder Environment

After Propel 2023.1 is installed, you can launch the stand-alone Propel Builder by double-clicking the **Run Propel Builder** icon () from the toolbar.

You can also invoke the Propel Builder from the command line:

- Run the following command in Windows:

```
<propel_installation_directory>\builder\rtf\bin\nt64\propelbld.exe -gui
```

- Run the following command on Linux:

```
< propel_installation_directory>/builder/rtf/bin/lin64/propelbld –gui
```

Refer to the Lattice Propel 2023.1 Installation for Windows User Guide (FPGA-AN-02067) and Lattice Propel 2023.1 Installation for Linux (FPGA-AN-02066) for details on the installation.

Note:

When you create a Propel SoC project, a linked verification project is always generated as well. Refer to the Switching SoC Project to Verification Project section for more information on verification project.

After the Propel Builder is launched, a single workbench window is displayed. The workbench contains Menu, Toolbar, Design View, IP catalog, Schematic view, address mapping, Start Page, and TCL console. Figure 2.1 shows the workbench after opening a project.

1. Menu bar
2. Toolbar
3. IP Catalog and Design View
4. Schematic View, Address Mapping, and Start Page
5. TCL Console

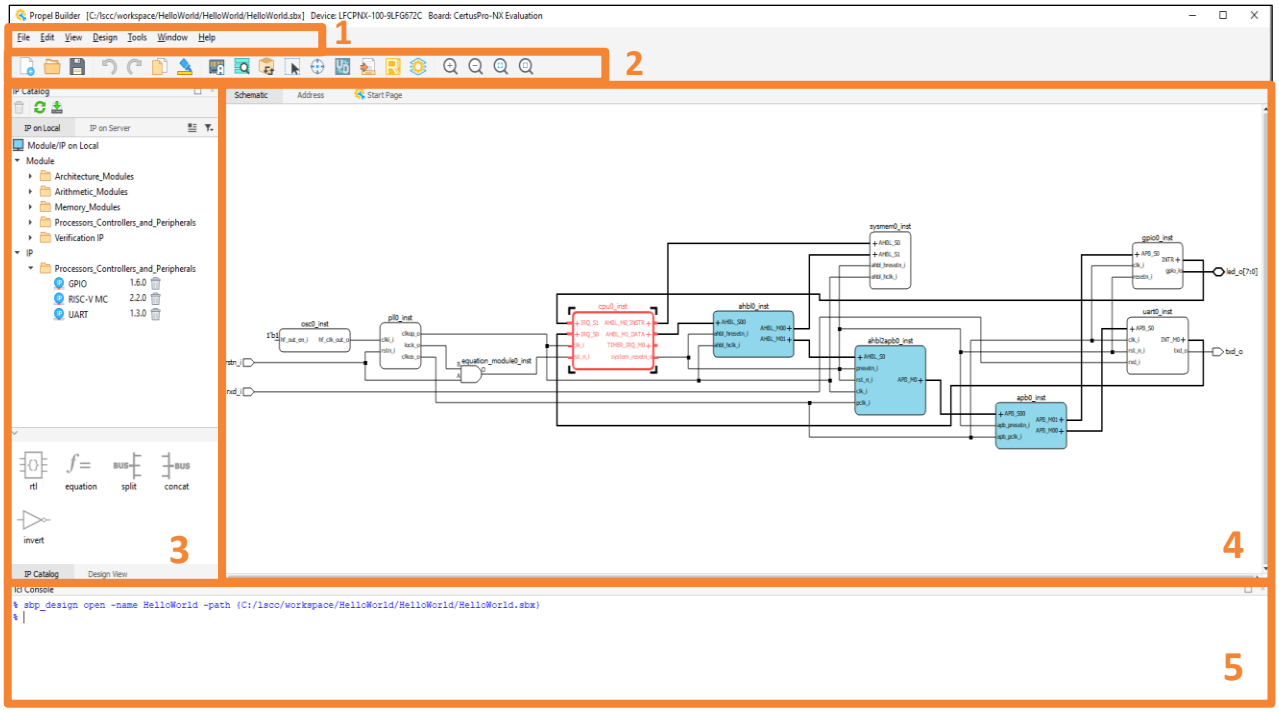


Figure 2.1. Propel Builder Workbench Window

2.2. Project Design Flow

2.2.1. Creating Template SoC Project

Templates are customized with commonly-used programs as well as pre-defined family, device, package, speed and operating condition. This HelloWorld template is programmed to run HelloWorld demo. You can make changes based on this template.

1. Choose **File** > **New Design** from the Lattice Propel Builder Menu bar. The Create System Design wizard (Figure 2.2) opens.

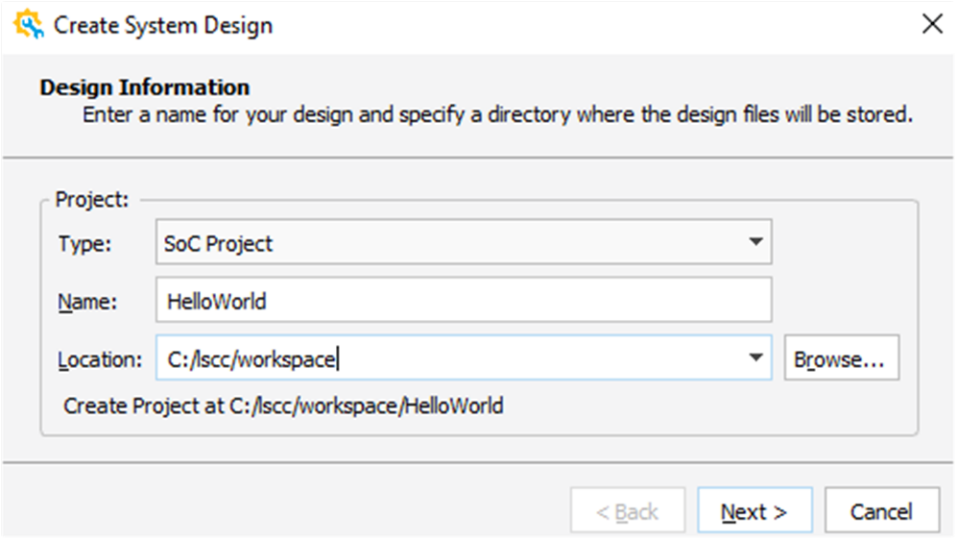


Figure 2.2. Create System Design – Design Information Wizard

2. The default Project Type is displayed in the **Type** field. Select **SoC Project** from the drop-down menu.
3. Enter a project name in the **Name** field, such as HelloWorld.
4. (Optional) The default location is shown in the **Location** field. Use the **Browse…** option to change the project workspace location.

Note: Long path is not supported well in Windows OS. When there is a file existing under that path but get a prompt of *No such file*, try moving the file to a directory with a shorter name.

5. Click **Next**. In the Configure Propel Project wizard, you can specify a device or a board for a Template SoC project. To specify a device for your new Template SoC project, use the **drop-down** menu to select desired device information (Family, Device, Package, Speed). Also, select **Hello World Project** or **Empty Project** in the **Templates** field (Figure 2.3).

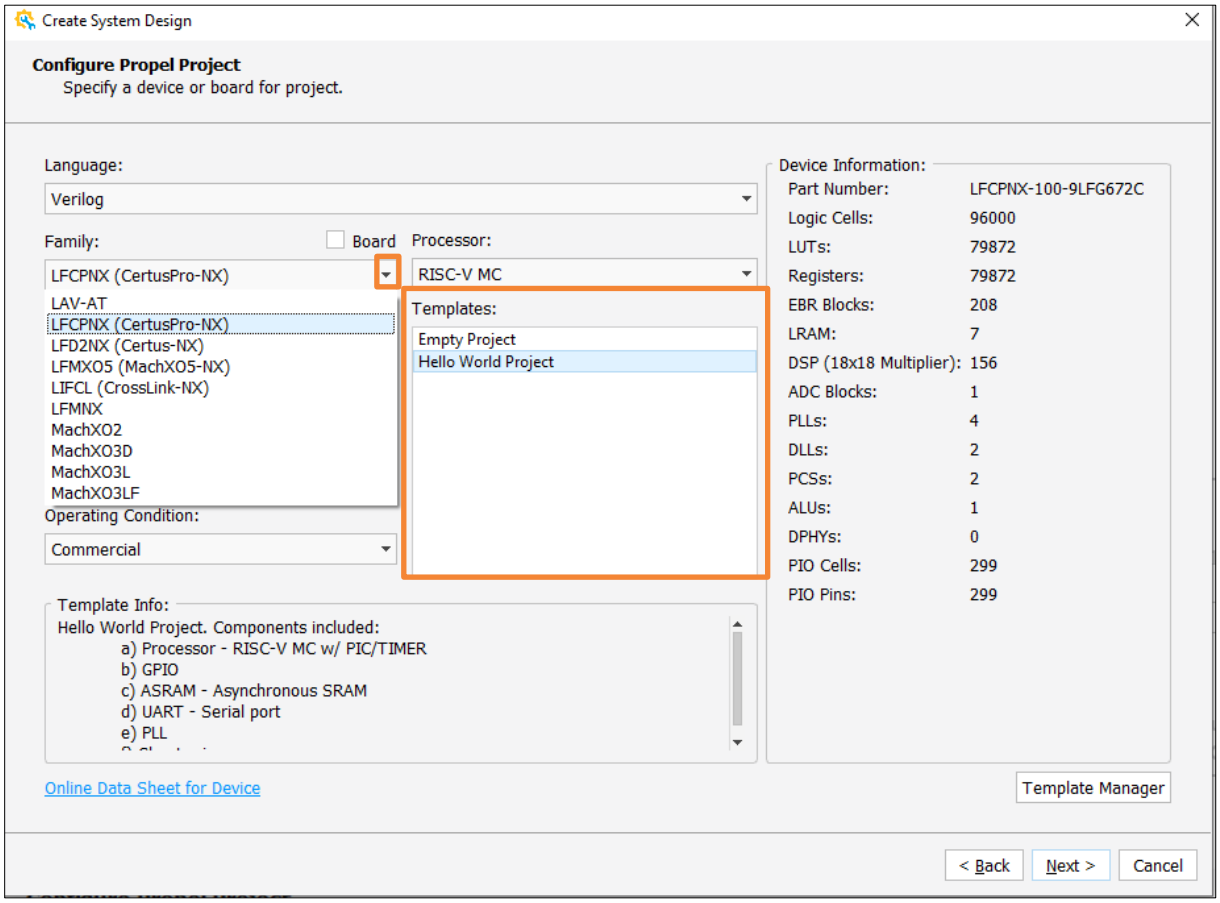


Figure 2.3. Specify a Device for Template SoC Project

- Or, to specify a board for a new Template SoC project, check the **Board** box (Figure 2.4).

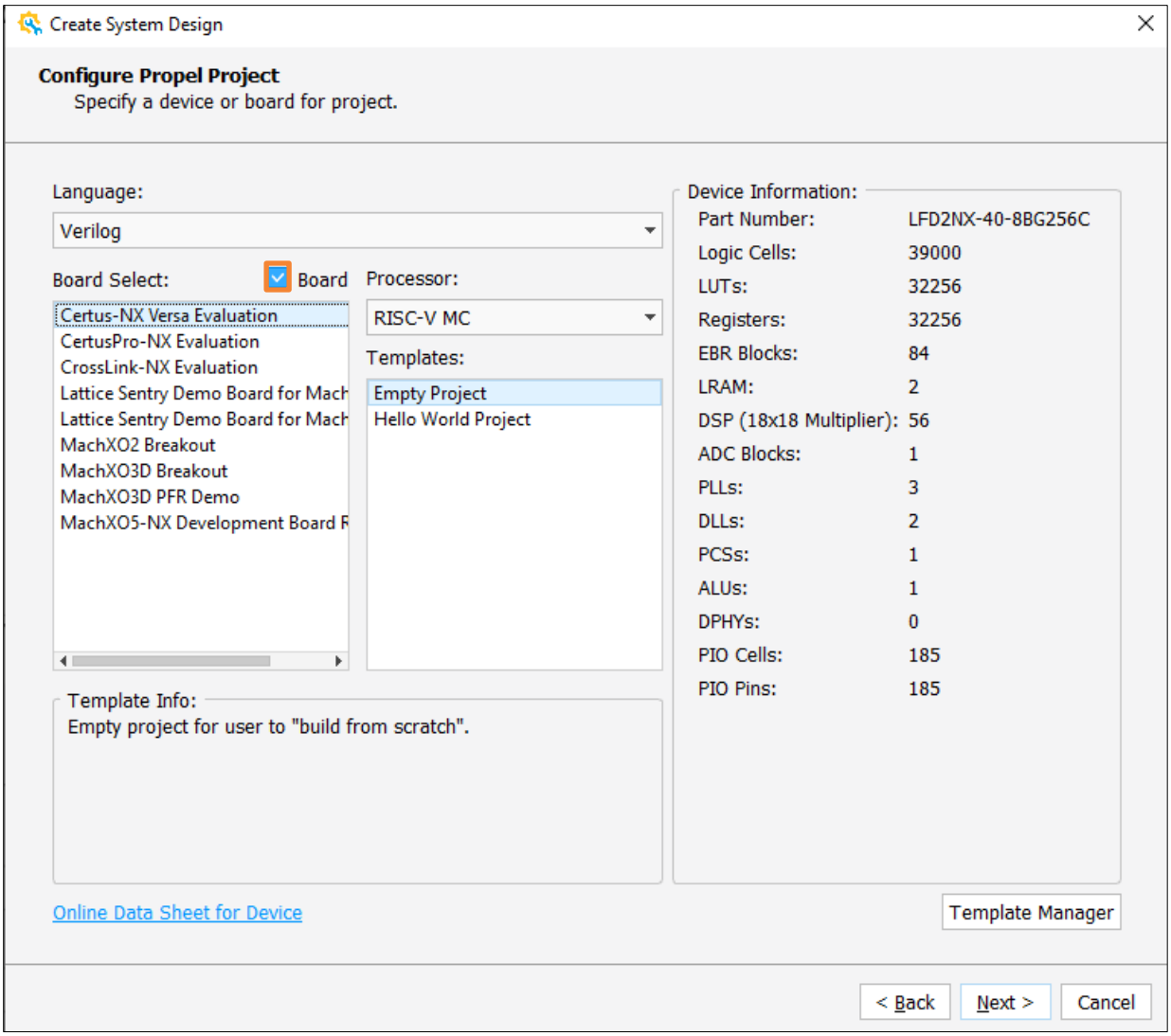


Figure 2.4. Specify a Board for Template SoC Project (1)

Notes:

- You can see both empty project and some predefined templates listed in **Templates** field.
- Select a desired template from the **Templates** area, such as Hello World Project.
- These templates usually have been validated on that board and some of them may have certain constraints. Check Lattice Propel 2023.1 Release Notes (FPGA-AN-02065) for more information on constraints.

6. The Configure Propel Project wizard is shown in Figure 2.5. From the **Board Select** area, select the desired board, such as CertusPro-NX Evaluation.

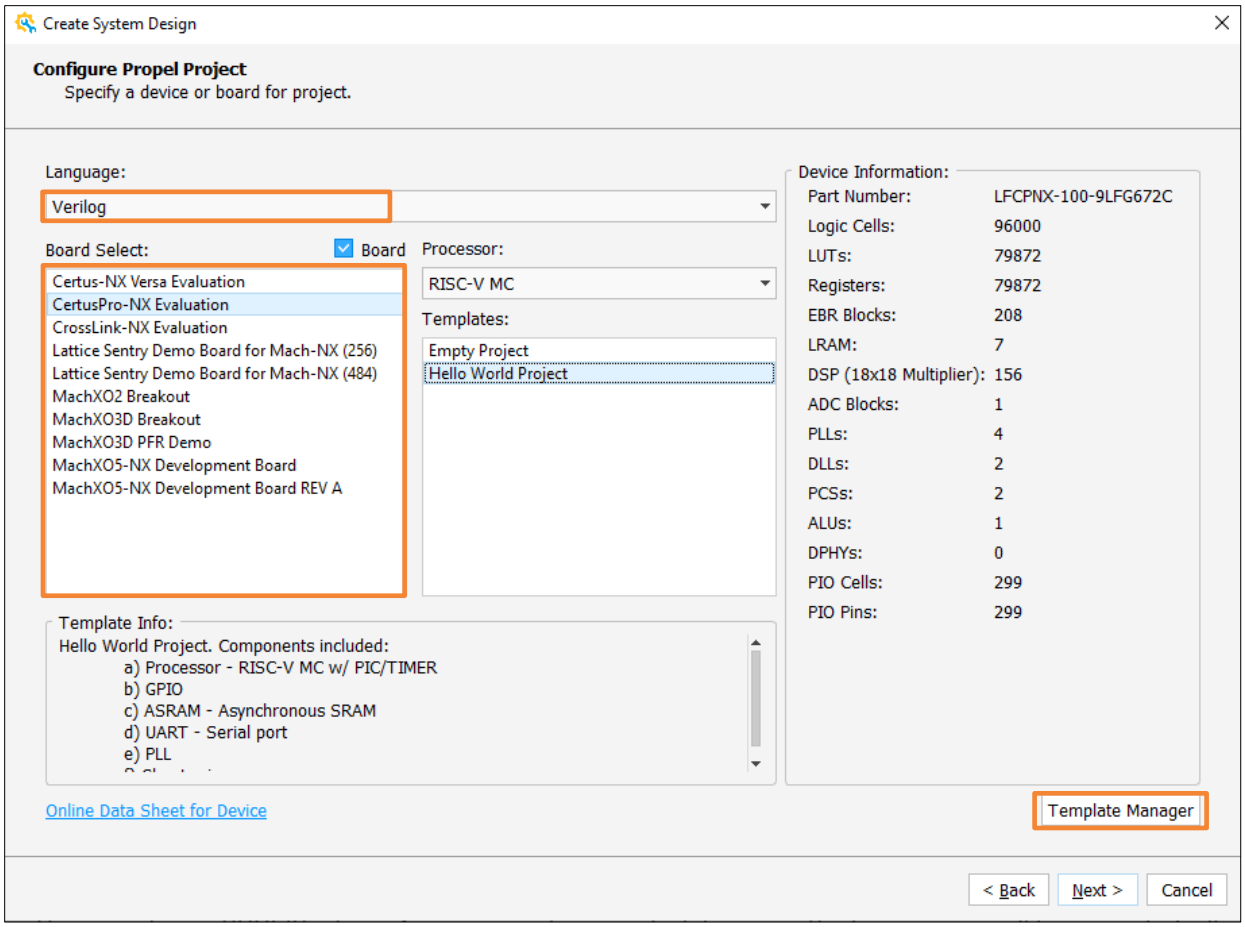


Figure 2.5. Specify a Board for Template SoC Project (2)

Notes:

- You can choose VHDL/Verilog in **Language** filed. The top wrapper is to be generated in your selected language. All other IP and modules are generated in their default language.
- **Template Manager** is for you to customize your own template that can be imported later. Refer to the Define Custom Template section for more information.

7. (Optional) About the Processor section (Figure 2.6):
   - If you create an empty project, you do not need to make any selection.
   - If you want a template with a specific processor type, you need select the correct processor.
   - Different processors are with different templates. For example, RISC-V RX processor is with different templates from those for RISC-V MC processor.

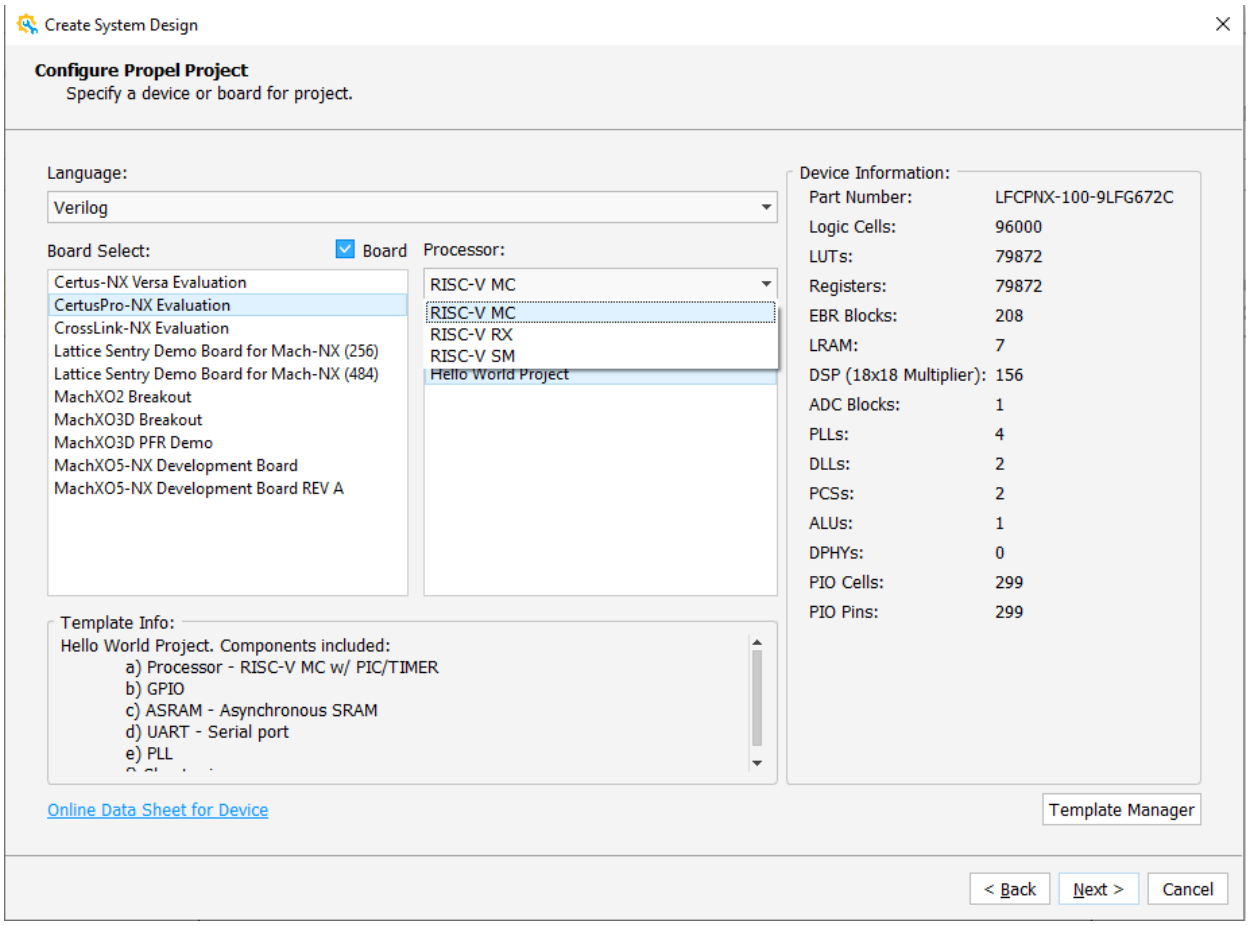


Figure 2.6. Specify a Board for Template SoC Project (3)

8. Click **Next**. The Project Information wizard opens, check and confirm the project information (Figure 2.7).

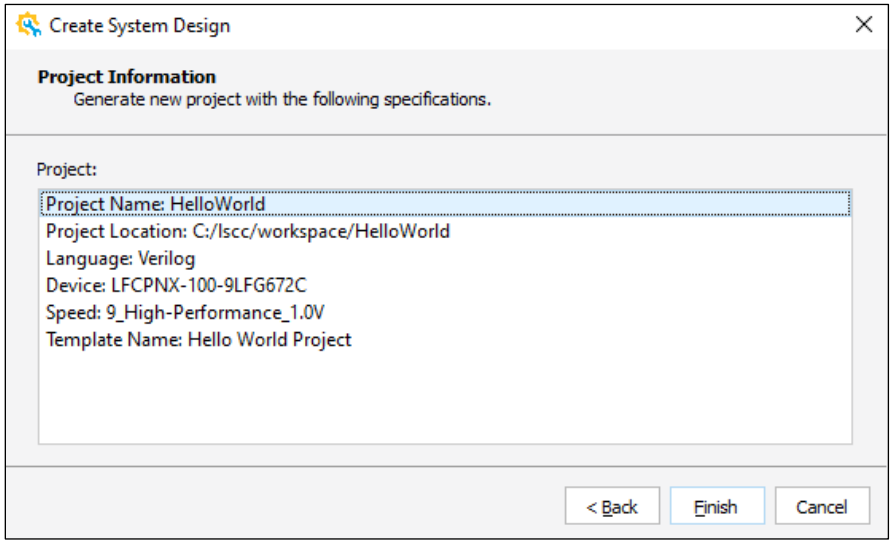


Figure 2.7. Project Information Wizard

9. Click **Finish**. Propel Builder GUI opens (Figure 2.8).

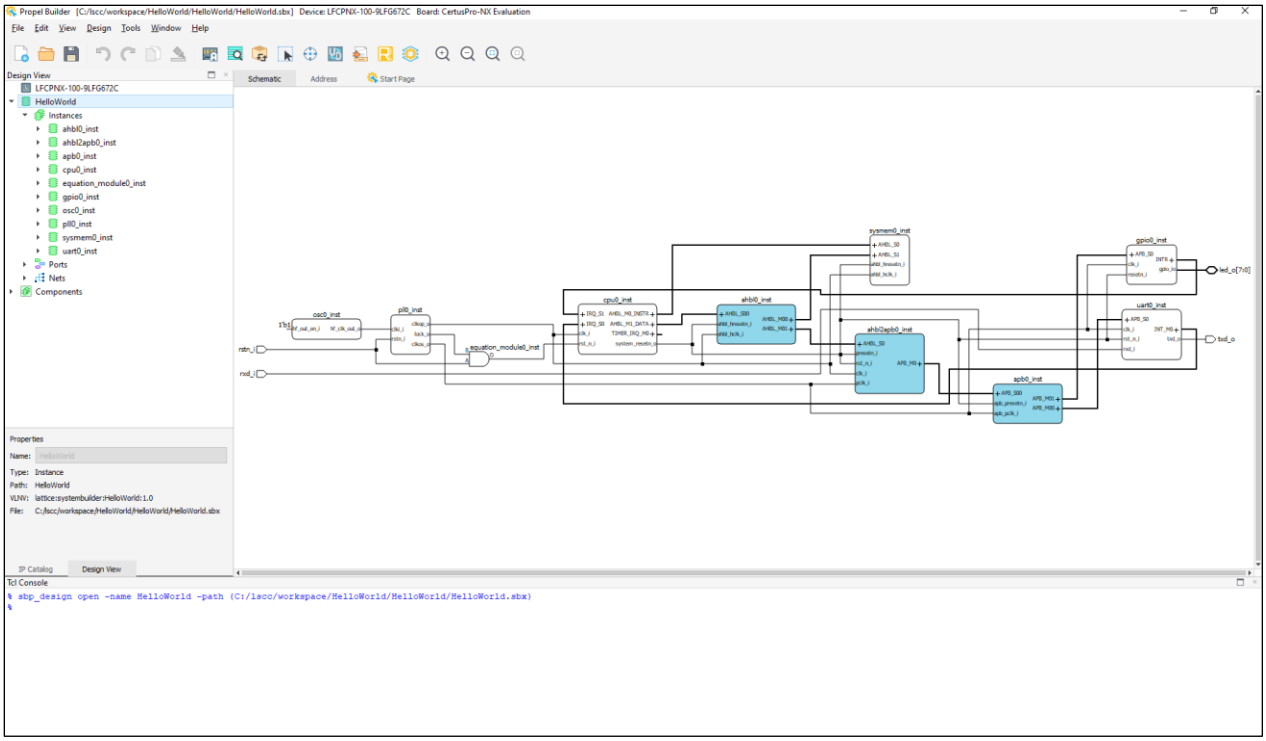


Figure 2.8. Propel Builder GUI showing HelloWorld Template SoC Project

2.2.2. Opening an Existing SoC Project

1. Choose **File** > **Open Design** from Propel Builder GUI Menu or Click **Open Design** icon from Propel Builder Toolbar. The Open sbx dialog opens (Figure 2.9).

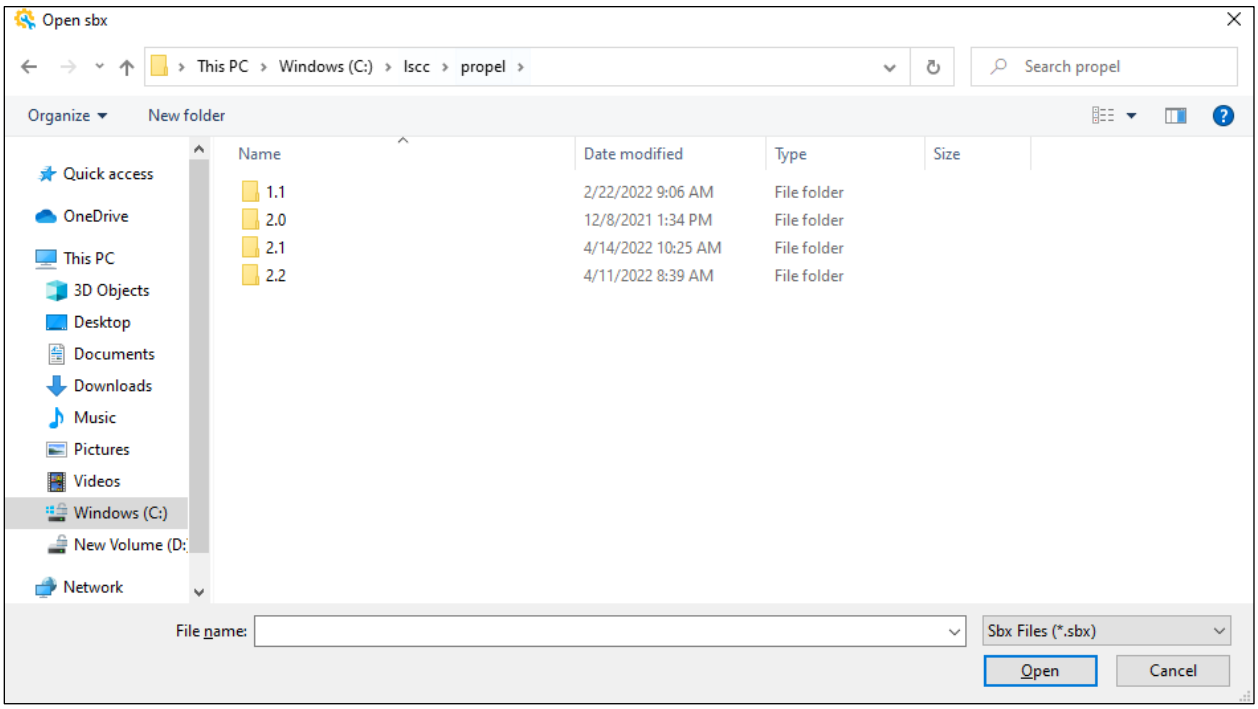


Figure 2.9. Open Sbx Dialog

2. Browse to find the default project workspace folder or your own project workspace folder. Choose the sbx file, such as HelloWorld.sbx (Figure 2.10).

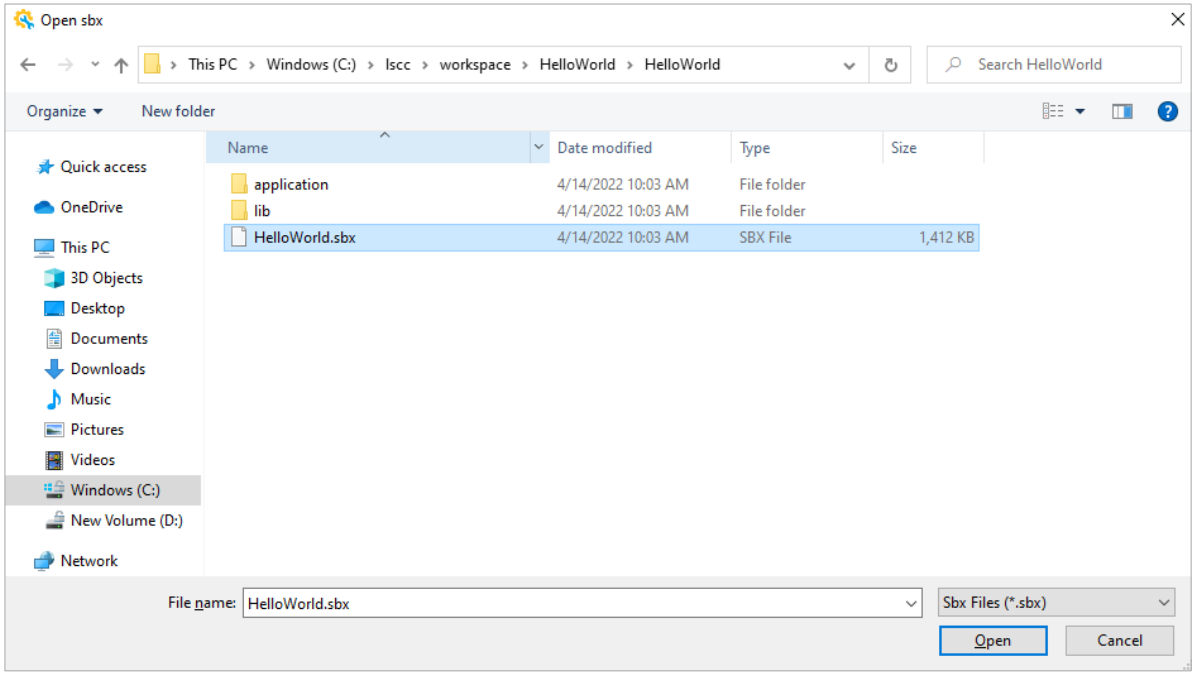


Figure 2.10. Open HelloWorld Project

3. Click **Open**. The Builder GUI shows the SoC design.
4. You can also use **File** > **Recent Designs** from Propel Builder Menu (Figure 2.11) to quickly open a recently-closed project.

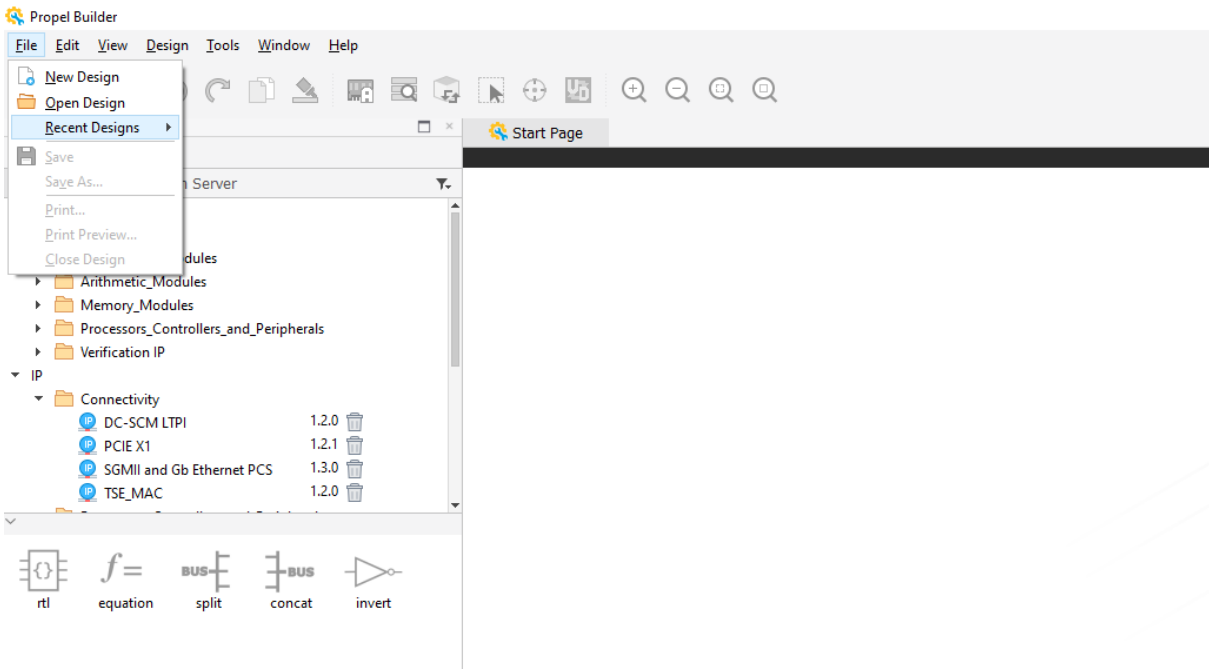


Figure 2.11. Open Recent Project

2.2.3. Generating and Instantiating IP/Module

After starting a Propel Builder project, you can add modules by dragging them from the IP Catalog view to the Schematic view. The IP Catalog view comes with a large variety of modules for common use and some glue logic modules, which can be found from the **IP on Local** tab (Figure 2.12). You can also click the **IP on Server** tab to find and download more modules for specialized use.

(Optional) From the Propel Builder GUI **IP Catalog** area (Figure 2.12), choose the **IP on Server** tab. Select a desired IP. IP version is displayed along with the IP name. For downloadable IP, a **Install** button is shown. indicates this IP is not compatible with the current version of Propel. If the IP is already installed, "Installed" is shown (Figure 2.12).

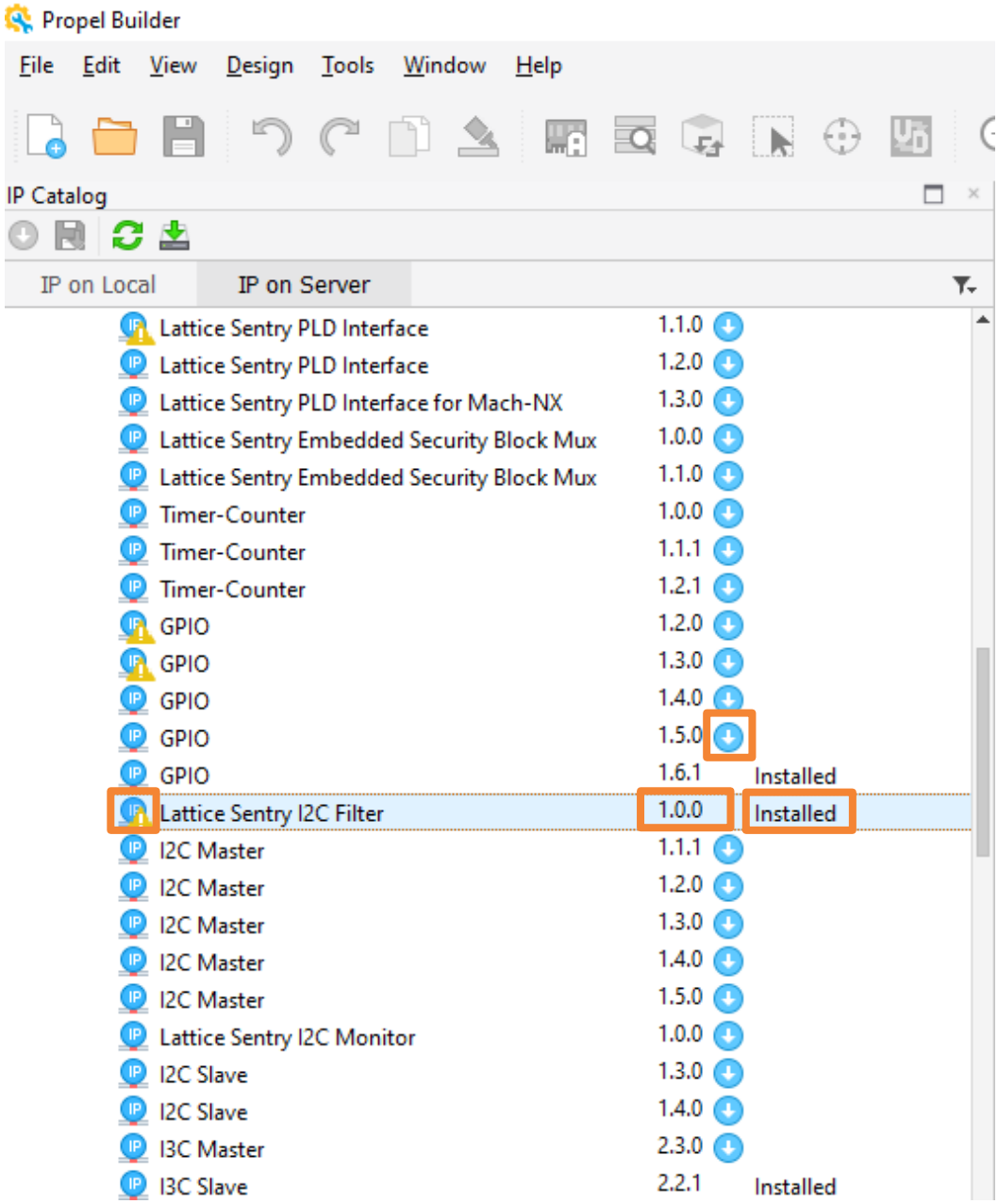


Figure 2.12. IP On Server

You can use the filter and search function in IP catalog to find the IP you desire (Figure 2.13).

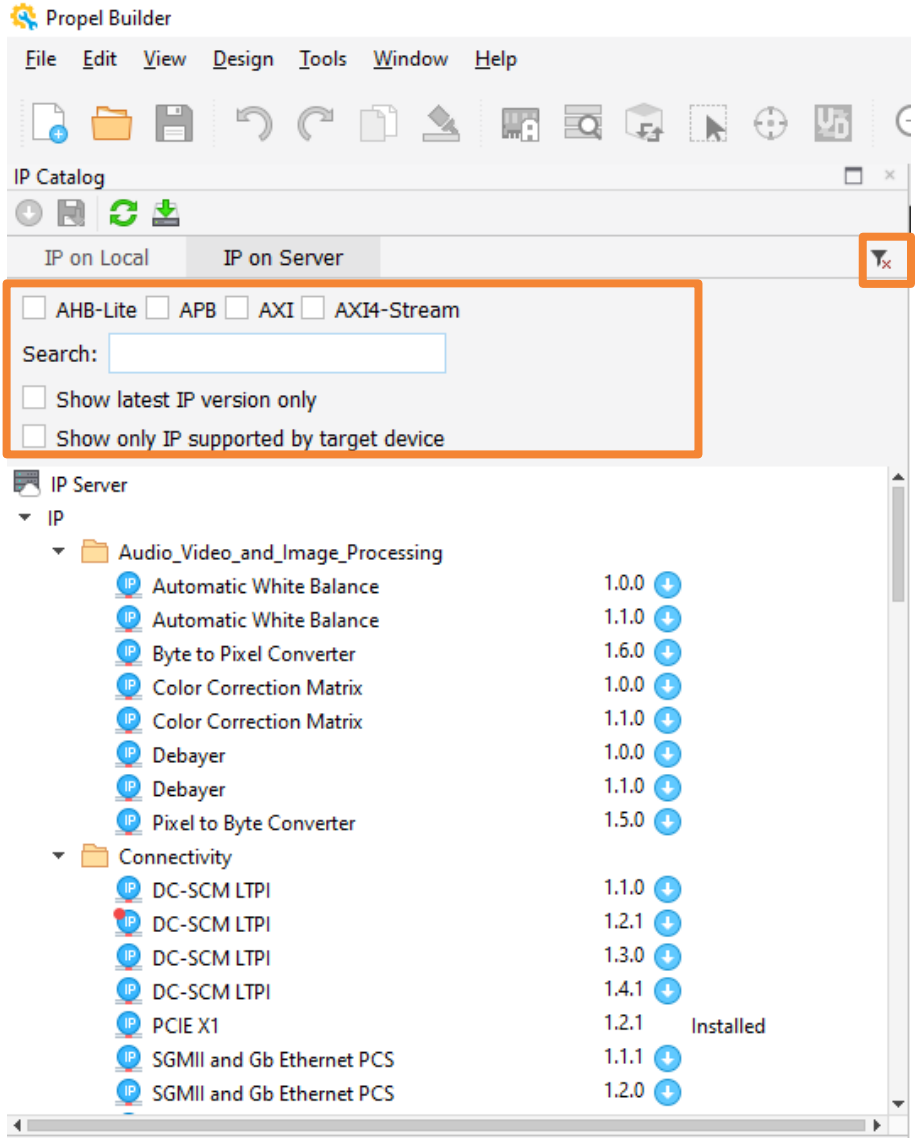


Figure 2.13. IP Filter and search

1. Being installed successfully, the new IP is shown in the **IP on Local** tab (Figure 2.14).
   Note: Only IP that are compatible with the current device can appear under **IP on Local**.

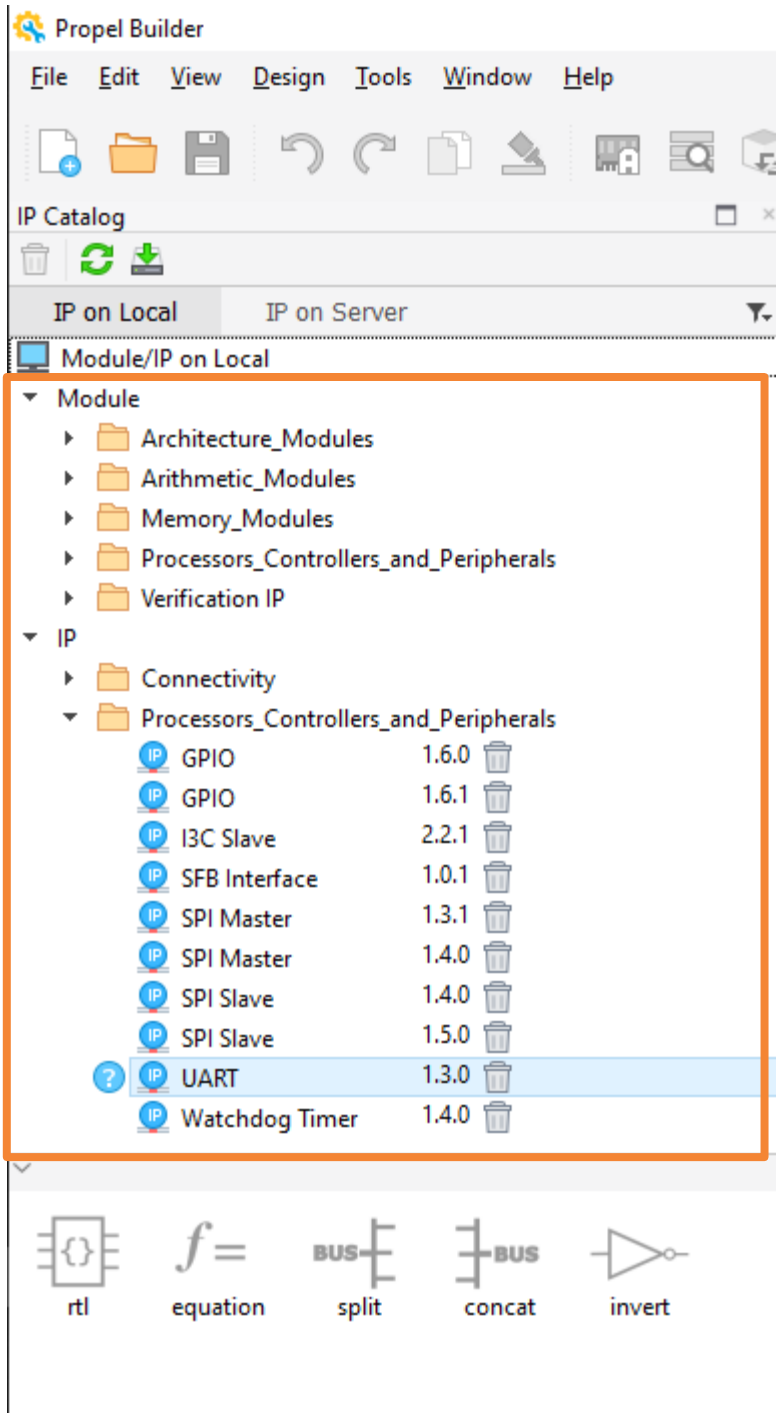


Figure 2.14. IP Catalog

2. From the **IP on Local** tab, select a desired IP, such as GPIO. Double-click the IP module, or drag and drop the IP module to the **Schematic** view. A Module/IP Block wizard pops up (Figure 2.15).

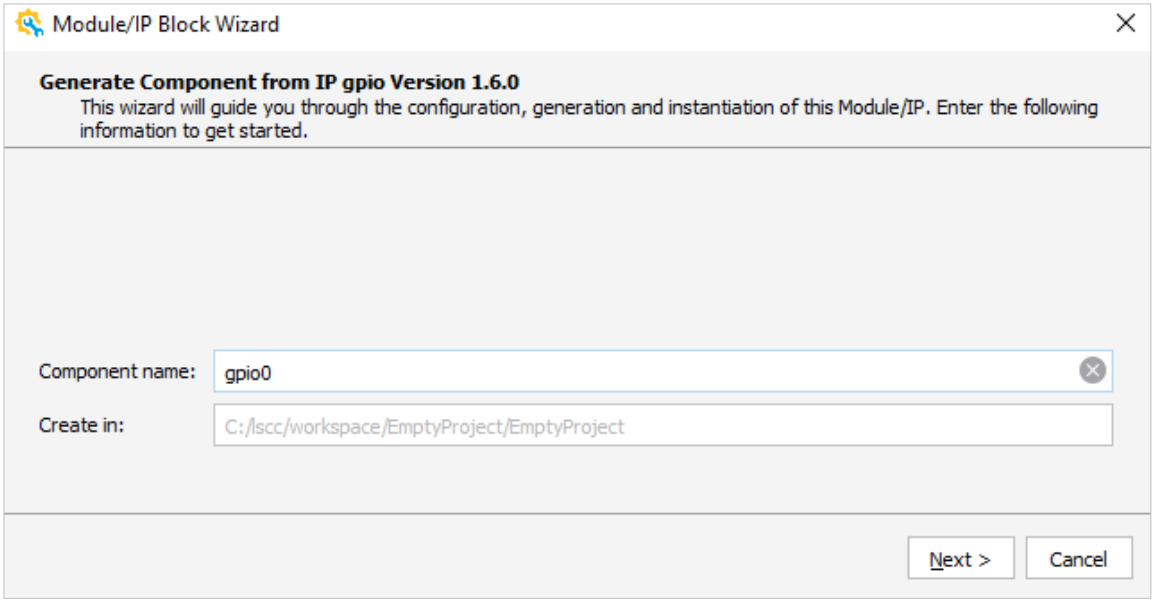


Figure 2.15. Module/IP Block Wizard – Generate Component from Module gpio Version 1.6.0

3. Enter a component name in the **Component name** area, such as gpio0. Click **Next**. Module/IP Block Wizard shows the **Configure Component from IP gpio Version 1.6.0** page (Figure 2.16).

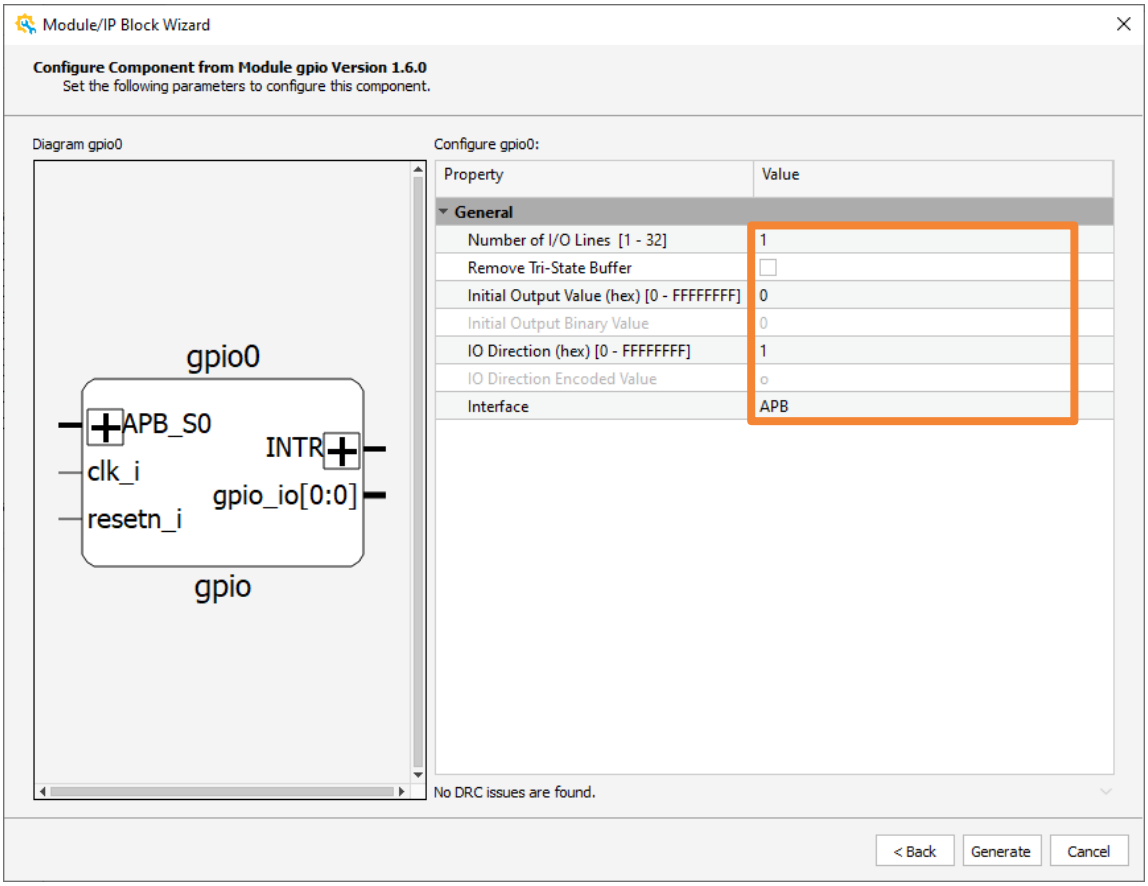


Figure 2.16. Module/IP Block Wizard - Configure Component from IP Gpio Version 1.6.0

4. (Optional) Reconfigure IP. Click the **Value** field. Enter a desired value for a property. Or check the checkbox to enable a property. You can also select a desired value from the dropdown menu for a property in the **Value** area. The property in gray is not configurable. By changing the value, the property is thus configured.

   Note: You need to generate the overall project for the IP to be generated or for any changes to take effect. Refer to Generating the RTL file section for more info on how to do this.

5. Click **Generate**. Module/IP Block wizard shows the Check Generated Result page (Figure 2.17).

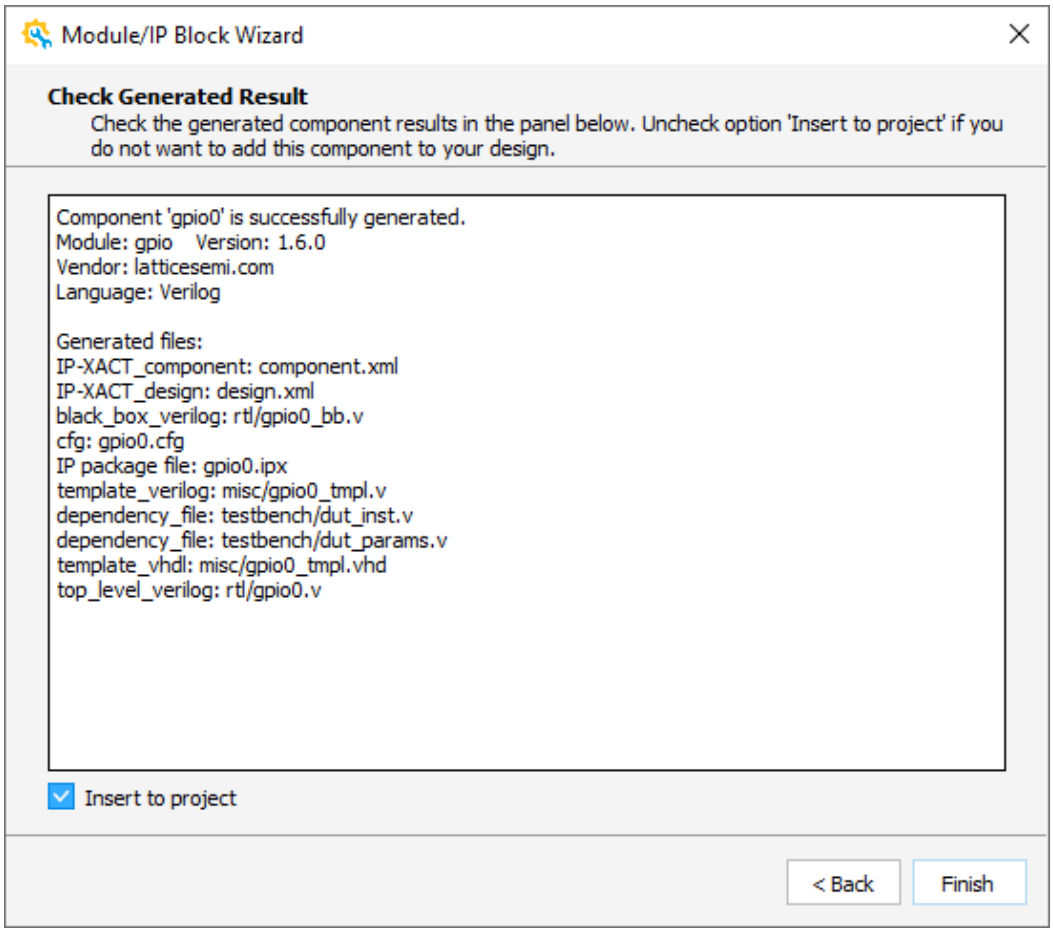


Figure 2.17. Module/IP Block Wizard - Check Generated Result

6. Select **Insert to project** (Figure 2.17) at the bottom of the Check Generate Result wizard, so that a **Define Instance** dialog box pops up as shown in next step. Or, this dialog box does not appear.

7. After selecting **Insert to project**, click **Finish** (Figure 2.17). The **Define Instance** dialog box opens (Figure 2.18).

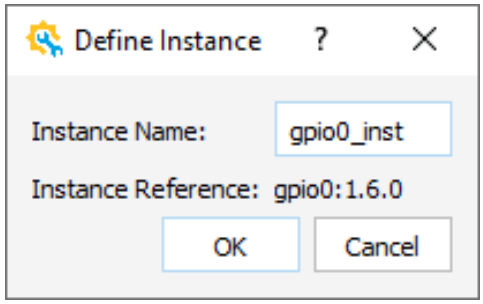


Figure 2.18. Design Instance Dialog Box

8. (Optional) Change the instance name, if desired. Space and special characters are not allowed.
9. Click **OK**. The schematic block for the module appears in the **Schematic** view (Figure 2.19).

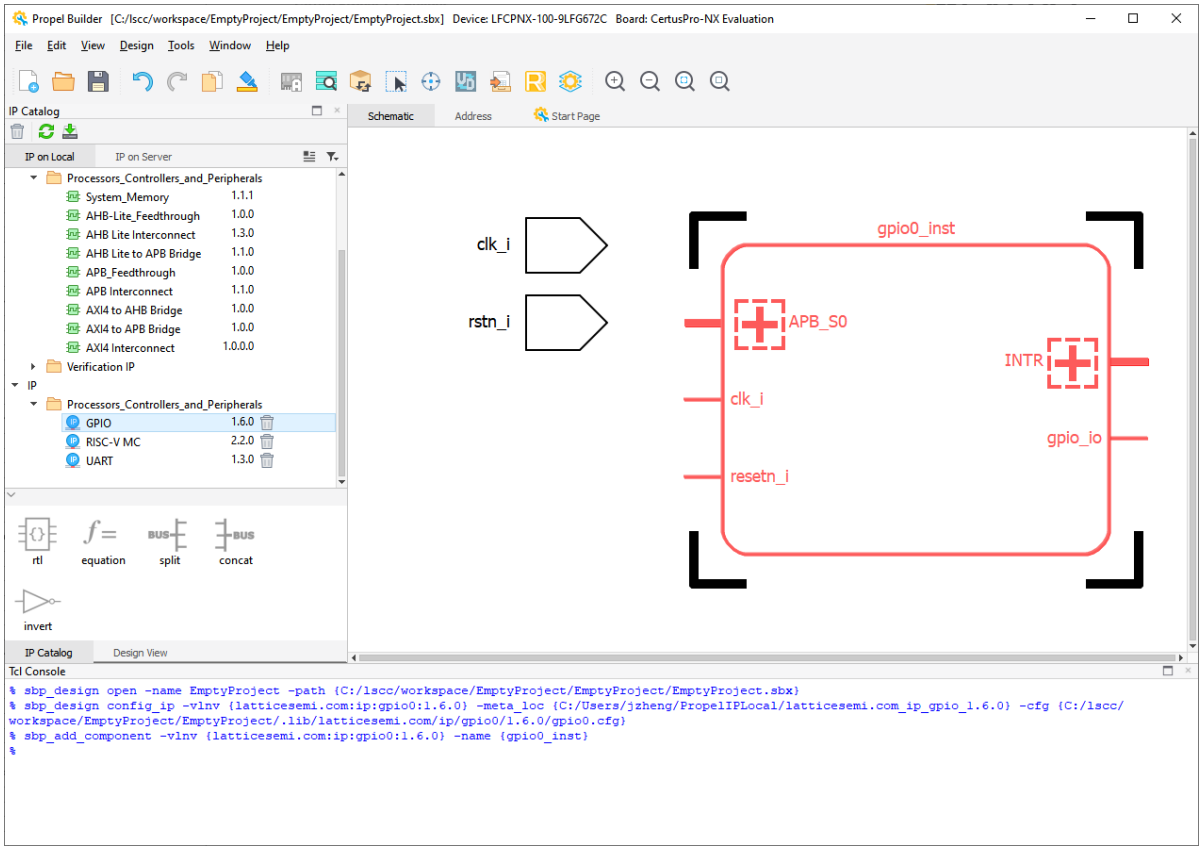


Figure 2.19. Propel Builder Schematic View Shows the Module Instance

2.2.4. Adding Glue Logic

Propel 2023.1 builder supports glue logic modules as well as IP modules. You can add glue logic modules by dragging them from the IP Catalog view (Figure 2.20) to the Schematic view.

Glue logics be modified afterwards by double click the module.

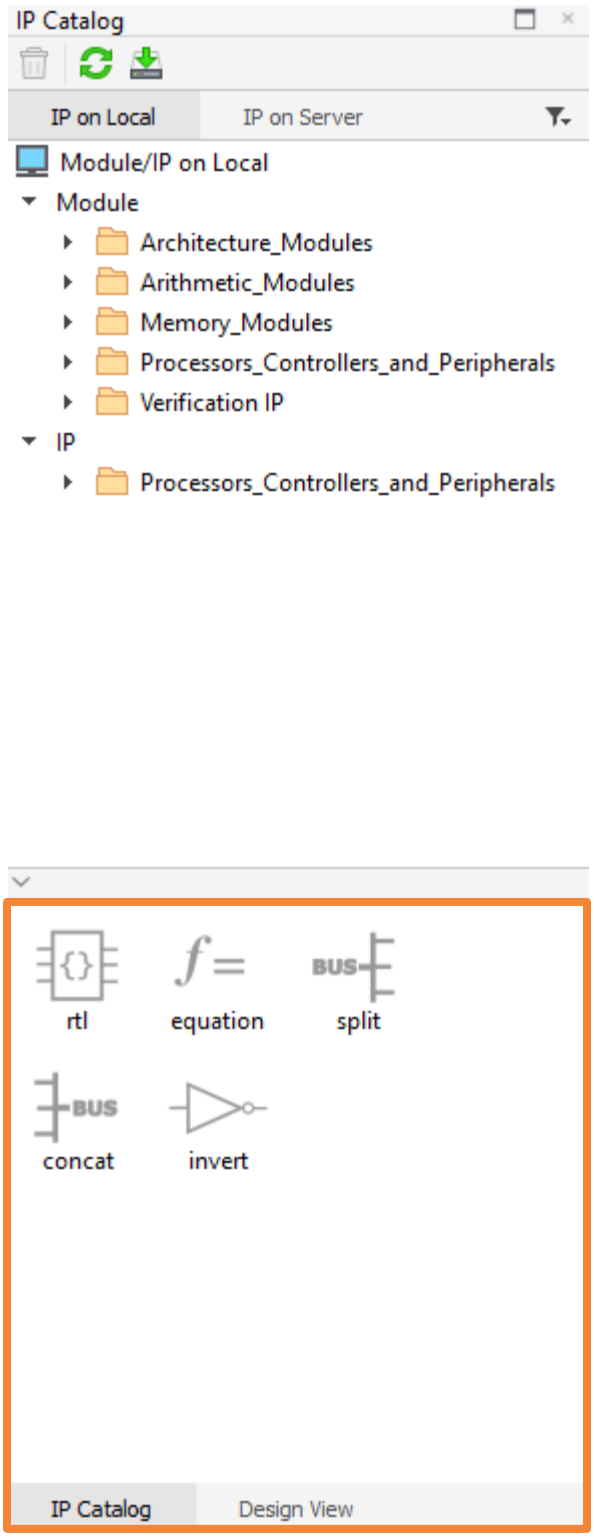


Figure 2.20. IP Catalog

2.2.4.1. Concat

1. From the IP Catalog, select the **concat** module. Double-click the concat module or drag and drop concat module to the Schematic view. A Glue Logic wizard (Figure 2.21) pops up.

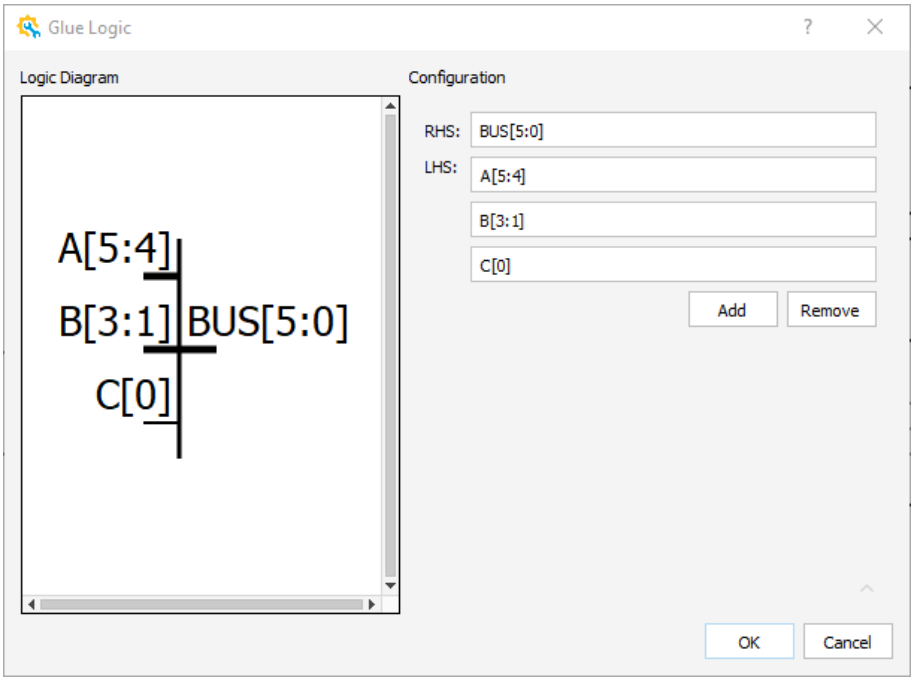


Figure 2.21. Glue Logic for Concat Module

2. Click the **RHS (right-hand side)** field to change the default right-hand side bus width to a desired one. Click the **LHS (left-hand side)** field to change the default left-hand side bus width to a desired one. Click the Add button to add LHS (Left-hand Side). Click the Remove button to remove LHS (Left-hand Side). You can only add or remove LHS but not RHS. LHS and RHS are thus configured.
3. Click **OK**. The schematic block for the concat module in red appears in the Schematic view (Figure 2.22). you can drag and connect them with the rest of design.

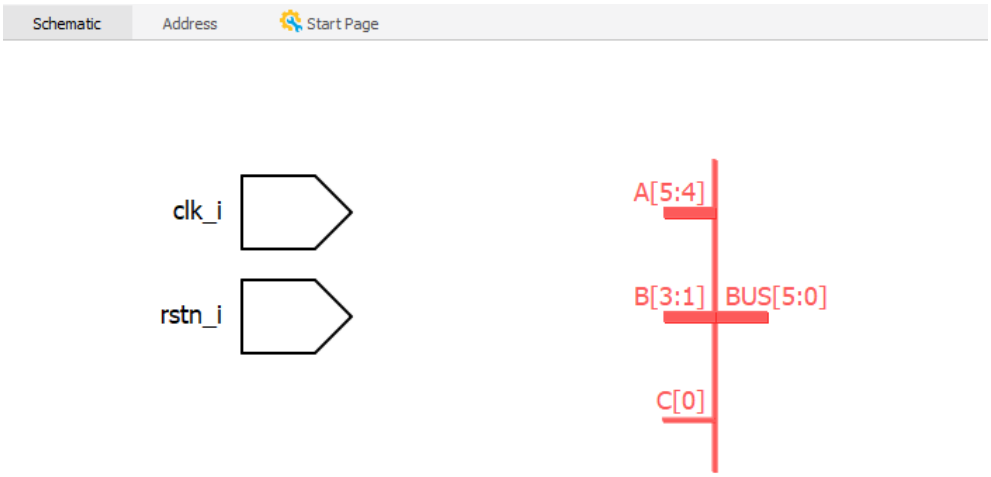


Figure 2.22. Schematic View Shows a Concat Module

2.2.4.2. Equation

1. From the IP Catalog, select the **equation** module. Double-click the equation module, or drag and drop the equation module to the Schematic view. A Glue Logic wizard (Figure 2.23) pops up.

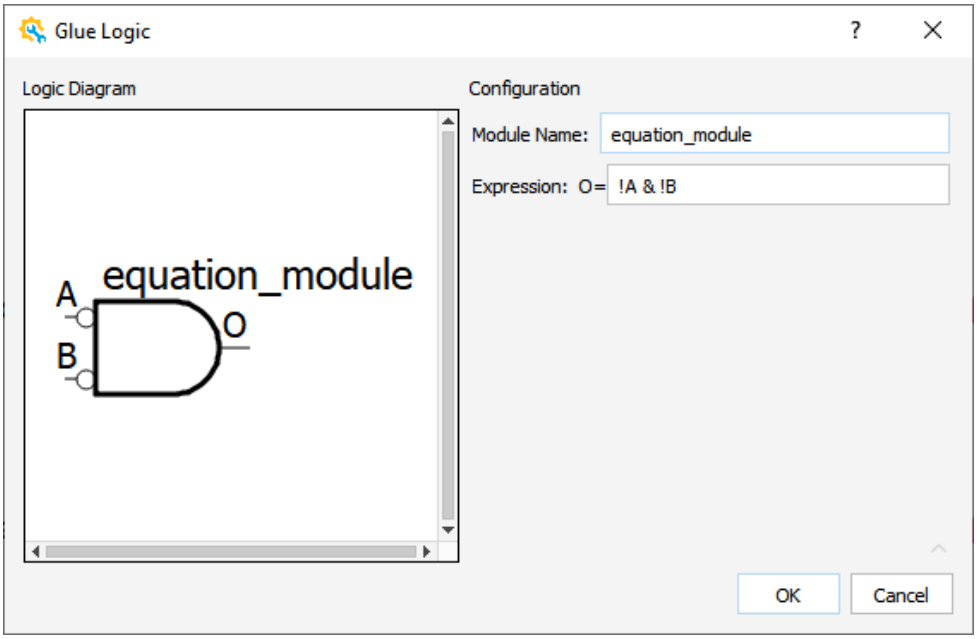


Figure 2.23. Glue Logic Wizard for Equation Module

2. Click the **Module Name** field to change the default value to a desired name. Click the **Expression** field to change the default expression to a desired one. The expression supports and (&), or (|), negation (!). The module name and expression are thus configured. Expression supports parentheses, as it complies with Verilog-HDL grammar.
3. Click **OK.** The schematic block for the equation module in red appears in the Schematic view (Figure 2.24).

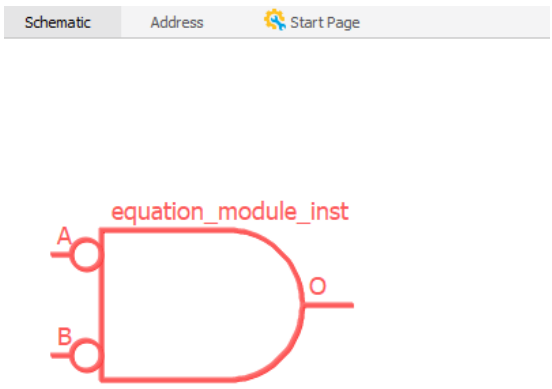


Figure 2.24. Schematic View Shows an Equation Module

2.2.4.3. Invert

1. From the IP Catalog, select the **invert** module. Double-click the invert module or drag and drop invert module to the Schematic view. The schematic block for the invert module in red appears in the Schematic view (Figure 2.25).

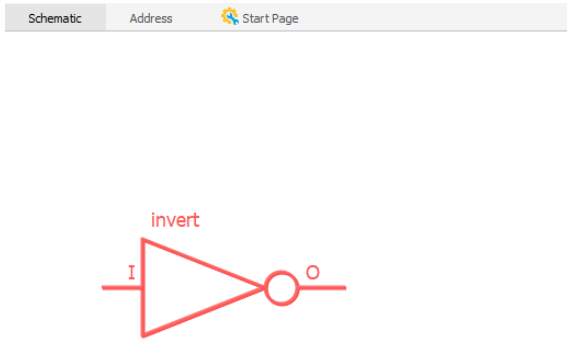


Figure 2.25. Schematic View Shows the Invert Module

2.2.4.4. Rtl

1. From the IP Catalog, select the **rtl** module. Double-click the rtl module or drag and drop the rtl module to the Schematic view. A Glue Logic wizard (Figure 2.26) pops up.

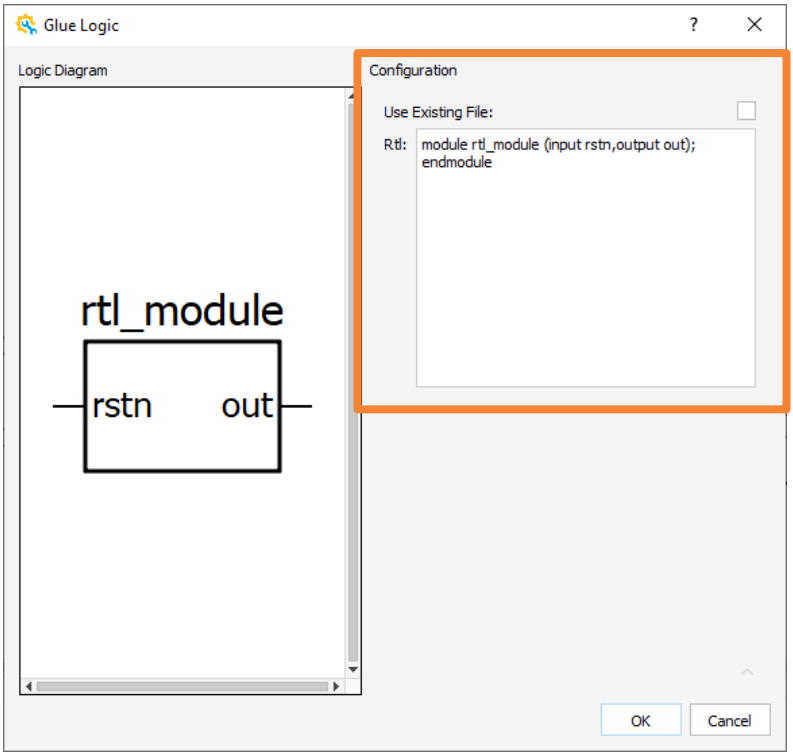


Figure 2.26. Glue Logic Wizard for Rtl Module

2. You can edit your own rtl module in the **Rtl** area by entering the rtl module function.
3. Or you can use an existing rtl module by checking the **Use Existing File** checkbox. After checking the Use Existing File checkbox (Figure 2.27), browse to find the existing RTL module file path from the **Path** filed. If you use an existing file, RTL references that file from the source. If you do not use an existing file, RTL creates a new file called <*module name*>.v. If **CopyToDesign** is enabled, a copy of the original source is created in the Propel project under <project>/lib/gluelogics/.

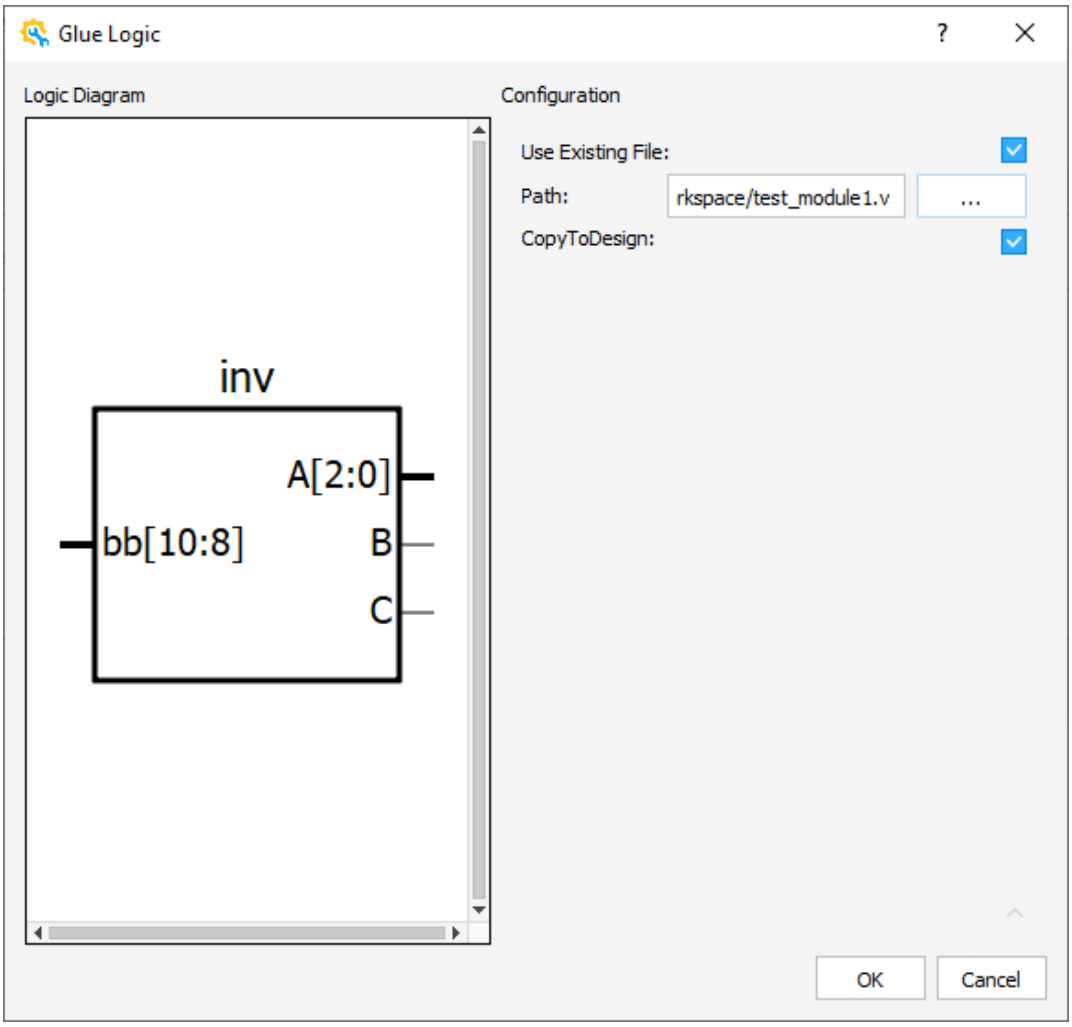


Figure 2.27. Existing Rtl Module Configuration

4. Click **OK**. The schematic block for the rtl module in red appears in the Schematic view (Figure 2.28).

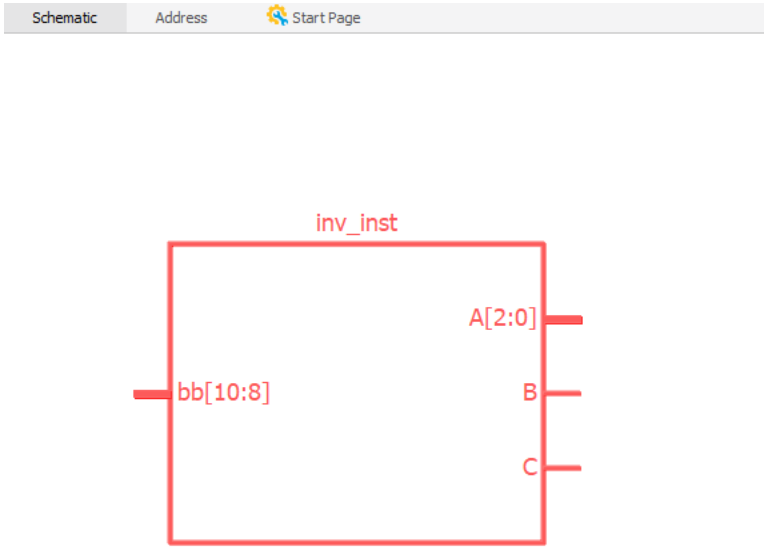


Figure 2.28. Schematic View Shows the Custom Rtl Module

Note: At present, Rtl Module only supports Verilog HDL.

2.2.4.5. Split

1. From the IP Catalog, select the **split** module. Double-click the split module, or drag and drop the split module to the Schematic view. A Glue Logic wizard (Figure 2.29) pops up.

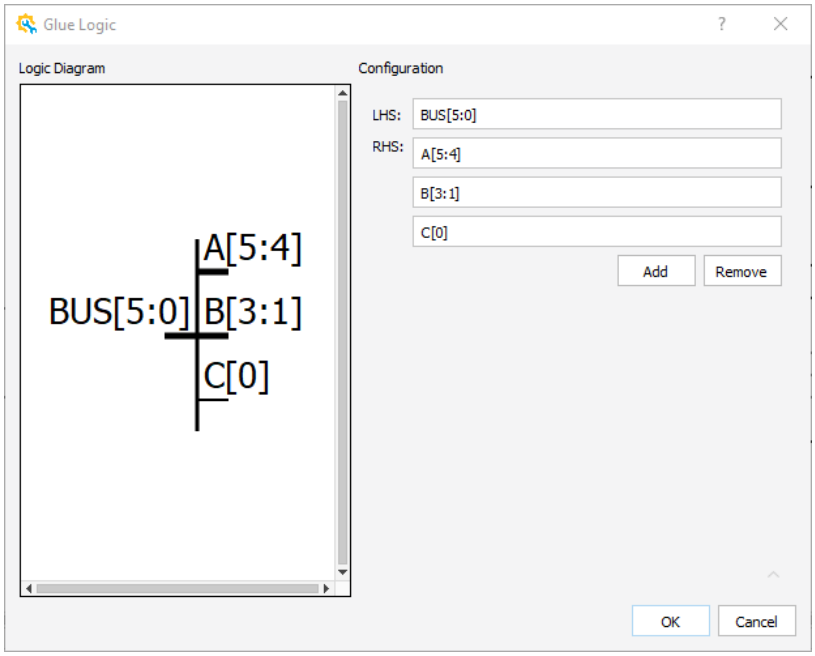


Figure 2.29. Glue Logic Wizard for Split Module

2. Click **LHS (left-hand side)** field to change the default left-hand side bus widths to a desired one. Click **RHS (right-hand side)** field to change the default right-hand side bit width to a desired one. Click the Add button to add an RHS. Click the Remove button to remove an RHS. You can only add or remove RHS but not LHS. LHS and RHS are thus configured.
3. Click **OK**. The schematic block for the split module in red appears in the Schematic view (Figure 2.30).

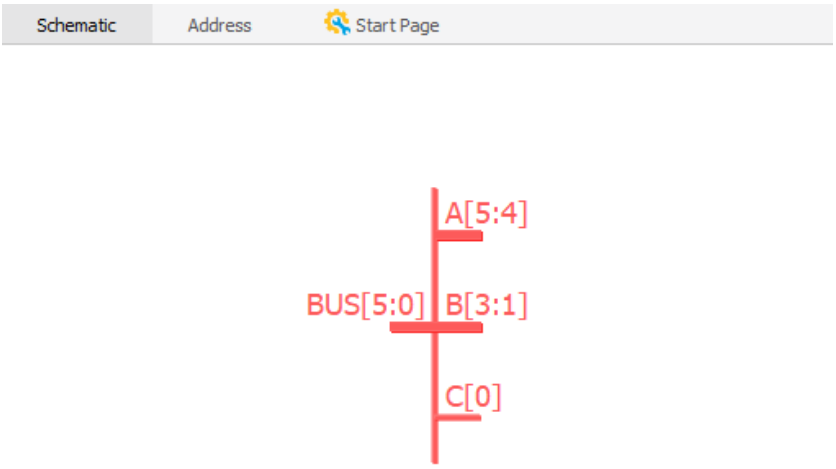


Figure 2.30. Schematic View Shows Split Module

2.2.5. Working with the Schematic View

You can make changes in the Schematic view including automatic layout to clean up the display, moving, resizing, renaming blocks manually, highlighting objects and zooming the display in and out.

2.2.5.1. View Signal List of Block

Click the plus sign of the desired block to see the signals it contains. The plus sign changes to a negative sign and shows the signal list as shown in Figure 2.31. Click the negative sign to close the expanded bus. The schematic returns to the previous form.

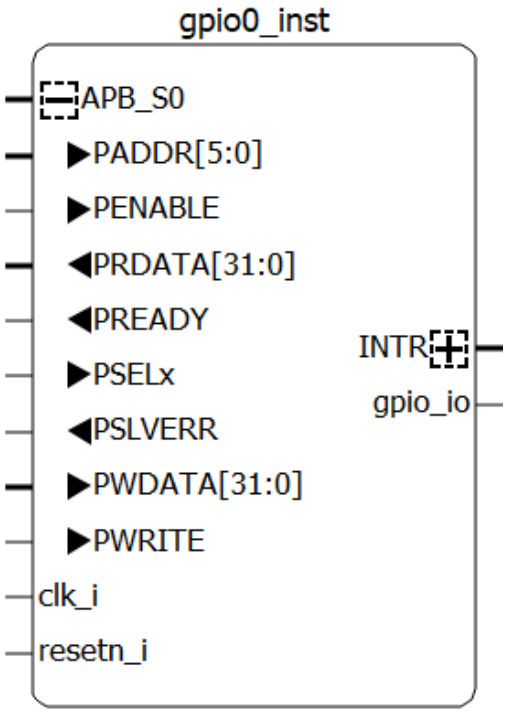


Figure 2.31. Signal List of Modules

2.2.5.2. Select One or More Objects

Select one object or more objects in one of the following ways:

- Click on the object in the Schematic view. The selected object turns to red (Figure 2.32).

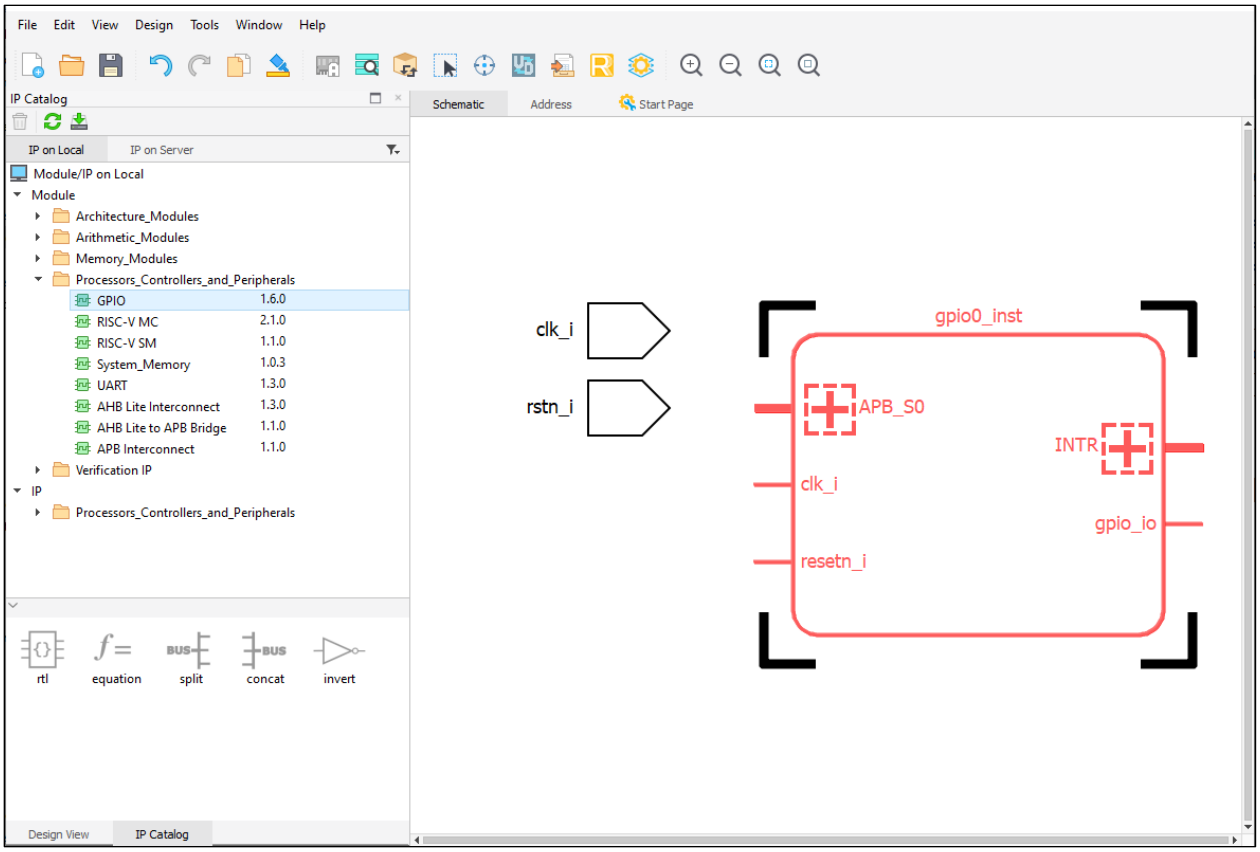


Figure 2.32. Select Object

- Ctrl-click or Shift-click in the Schematic view to select more than one objects.
- Click the Area_select icon from the Propel Builder Toolbar. Click and drag to draw a selection rectangle around the modules and ports in the Schematic view. Click the icon again to turn off the Area_select mode.
- From the Schematic view, right-click and choose Select All or press Ctrl-A to select all the objects.

  Note: Ctrl-click or Shift-click on the object in the Schematic view can also de-select the object while leaving the others selected.

2.2.5.3. Re-arrange the Schematic

Propel Builder allows re-arranging the objects in the schematic view. You can re-arrange modules and ports. Drag objects to re-arrange the schematic. Propel Builder has rules for placing objects to adjust the schematic in an organized arrangement.

1. Select the desired modules (one or more modules can be dragged at the same time) or ports (one or more ports can be dragged at the same time).
2. Click on the selected items and drag it/them to the desired location.
3. Release the mouse button.

Note: The selected objects can be moved to a specified location, or near the existing object (as near as the rules allow). Other objects in the schematic can also be moved to accommodate the new location of the selected objects.

To bring selected objects to the center of the Schematic view using the Locate Object mode:

1. Click the **Locate Object** icon from the Propel Builder Toolbar. The background turns to dark gray.
2. Select the object in the List view of the Design View. The selected object is in the center of the Schematic View (Figure 2.33).

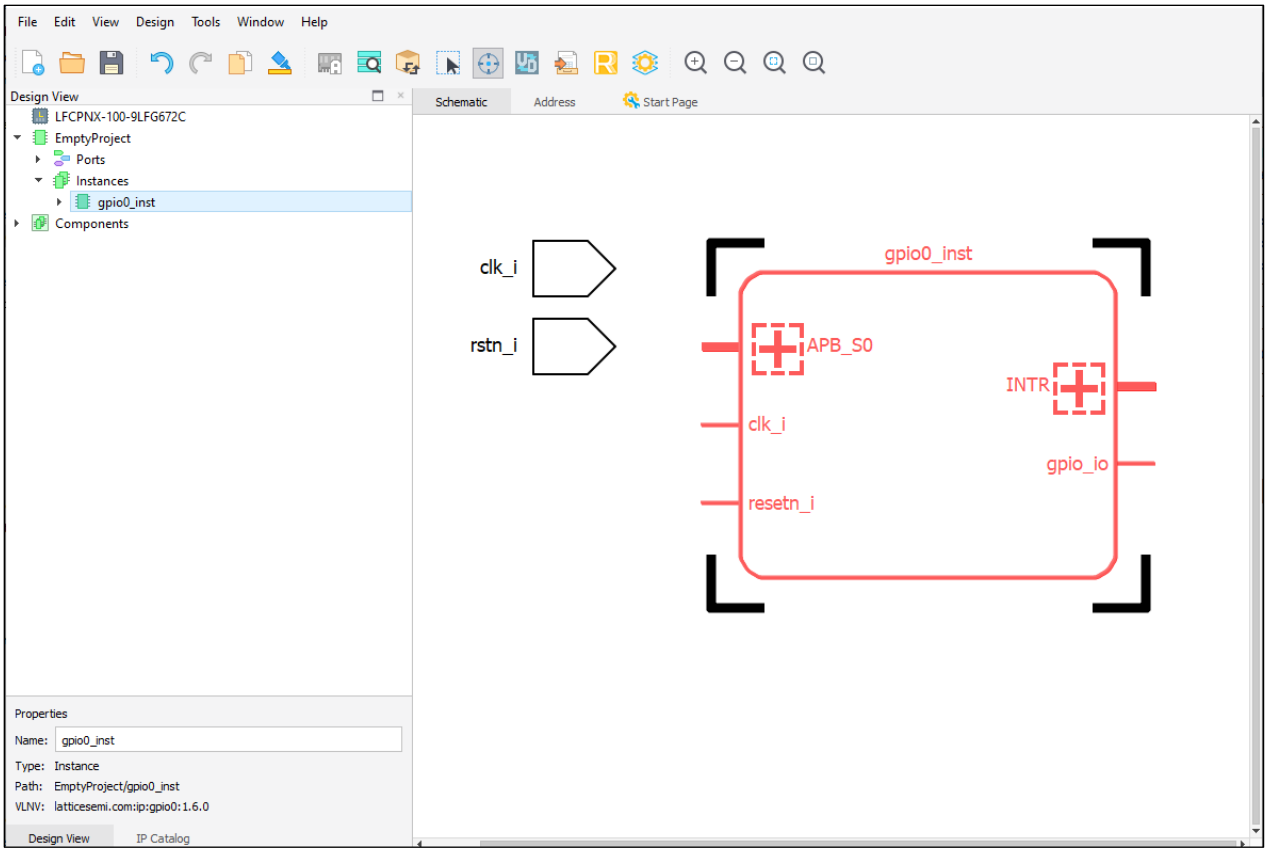


Figure 2.33. Locate Objects

To automatically simplify the layout:

1. Right-click anywhere in the Schematic view and choose **Relayout**.

2.2.5.4. Duplicate a Module

Select the desired module and click **Clone**  . Or right-click on the module and choose **Clone**  . A copy of the schematic block appears with a new instance name (Figure 2.34).

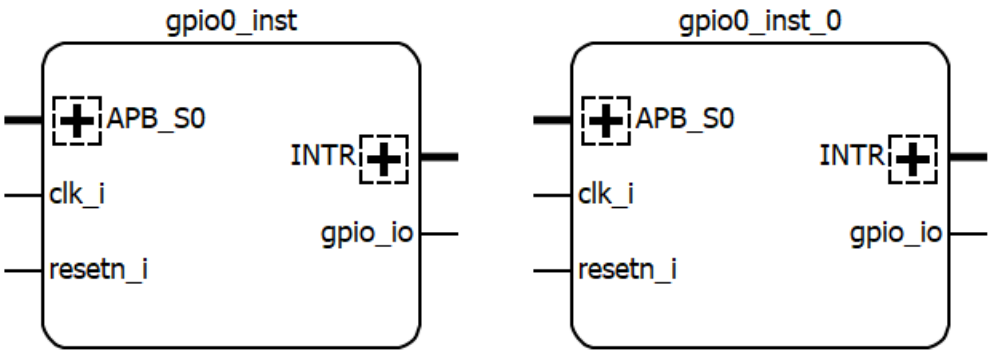


Figure 2.34. Duplicate a Module

2.2.5.5. Restore a Deleted Module

1. In the List view of the Design View (Figure 2.35), go to the Components folder. The deleted module can still be found in the List view in plain text. Those not deleted are shown in bold-faced text.

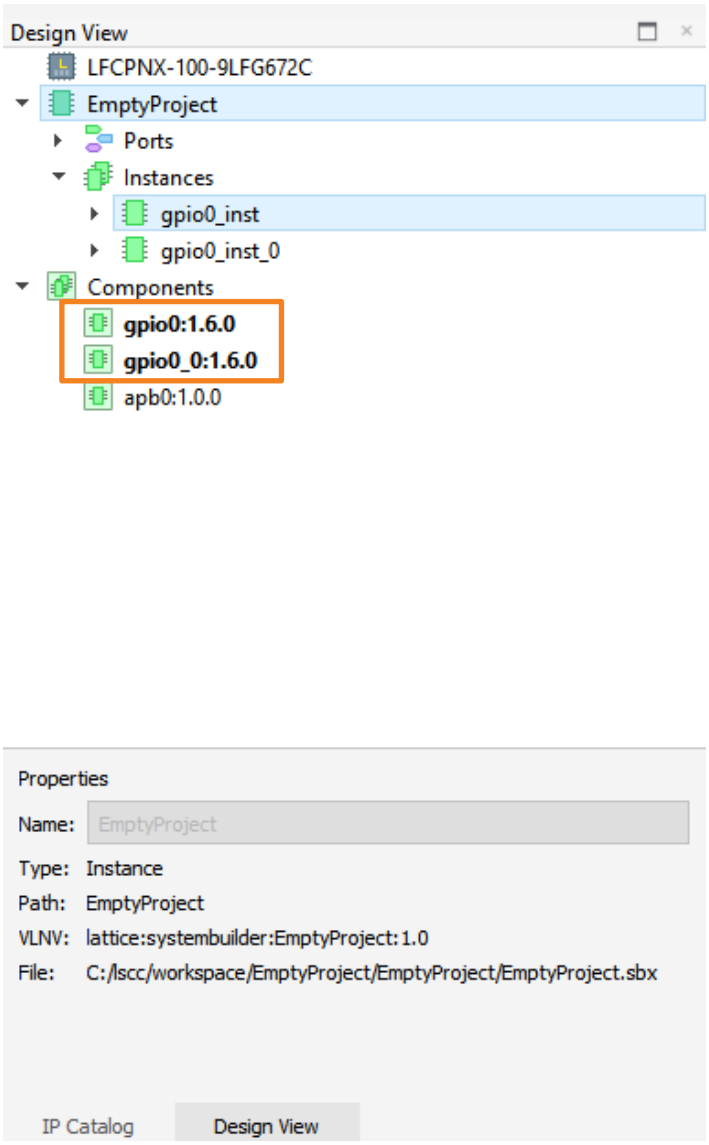


Figure 2.35. Design View

2. Right-click on the component you want to restore, and choose **Instantiate**. The Define Instance dialog box (Figure 2.36) open.

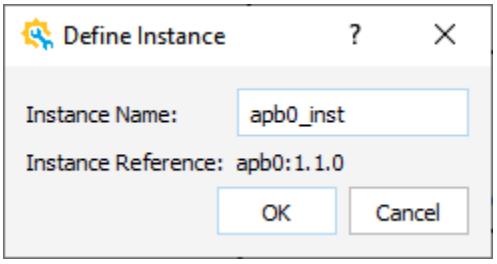


Figure 2.36. Define Instance Dialog Box

3. Enter a name for the new instance.
4. Click **OK**. The module appears in the Schematic view and the List view of Design Info, and the component name is bold-faced.

2.2.5.6. Reconfigure a Module

1. Double-click the module, or right-click the module you want to reconfigure. Choose **Reconfig**. The Module/IP Block wizard (Figure 2.37) opens.

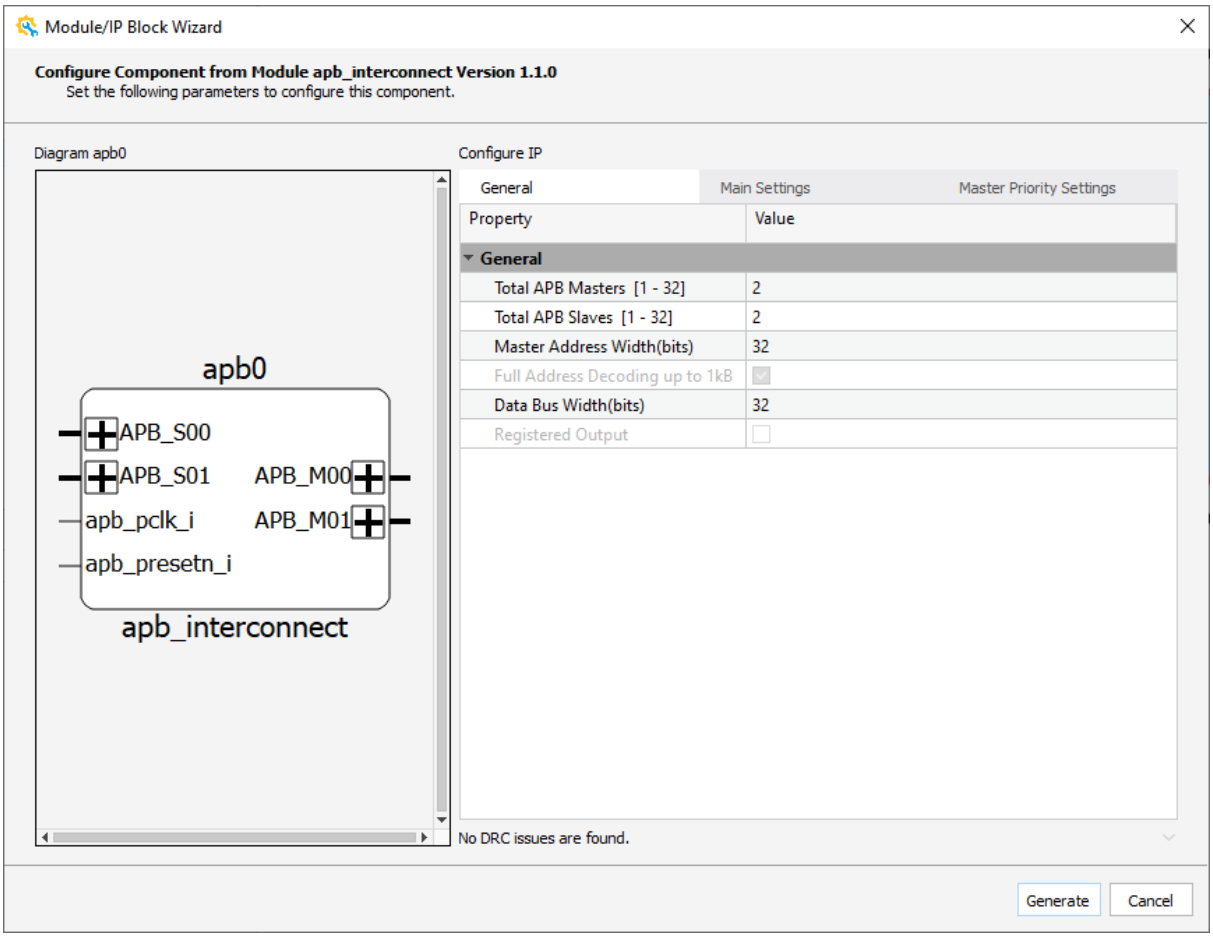


Figure 2.37. Module/IP Block Wizard – Configure Component

2. Configure component (including General property, Main Settings and Mater Priority Settings) at Configure IP table in the Module/IP Block Wizard. Click **Generate** to generate the module as usual. The schematic block for the module changes to match the new configuration.

2.2.5.7. Resize Module Blocks

1. Select the desired module block. The block is highlighted in red with black corners (Figure 2.38).

Figure 2.38. Select Module

2. Click and drag one of the corners to change the size and to shape of the block.
3. Release the mouse button. All the other objects move to make room for this block.

   Note: Right-click the module block and choose **Unresize Instance** to restore the size of the module block.

2.2.5.8. Methods to Zoom

There are a variety of methods to zoom within the Schematic view including toolbar commands and dragging in the Schematic view.

The following commands are available from the Propel Builder Toolbar.

- Zoom In (Ctrl++) — enlarges the view of the entire layout.
- Zoom Out (Ctrl+-) — reduces the view of the entire layout.
- Zoom Fit — reduces or enlarges the entire layout so that it fits inside the window.
- Zoom To — enlarges the size of one or more selected objects on the layout and fills the window with selection.

Note: The mouse wheel provides a finer zoom control by rolling the mouse wheel forward to zoom in and backward to zoom out while pressing the Ctrl key.

- To zoom by holding the mouse button and dragging.
  - To zoom to fit the window, drag up and to the left. The image adjusts to fill the window.
  - To zoom out, drag up and to the right. The dragging distance determines the amount of zoom. The image is reduced and centered in the window.
  - To zoom in, drag down and to the left. The dragging distance determines the amount of zoom. The image is enlarged and centered in the window.
  - To zoom in in a specific area, start at the upper-left corner of the area and drag to the lower-right corner of the area. The area that dragging across is adjusted to fill the window.

Note: Make sure that the **Area_select** icon is not selected in the Toolbar. If you need to use Area_select and zoom by dragging frequently, choose **Tools** > **Options** from Propel Builder menu bar. The Options Dialog opens (Figure 2.39). Select **Use right mouse button for zooming actions** and click **OK**. Then you can select an instance using the left mouse

button, zoom in or zoom out using the right mouse button, and the **Area_select** icon disappears from the Propel Builder Toolbar.

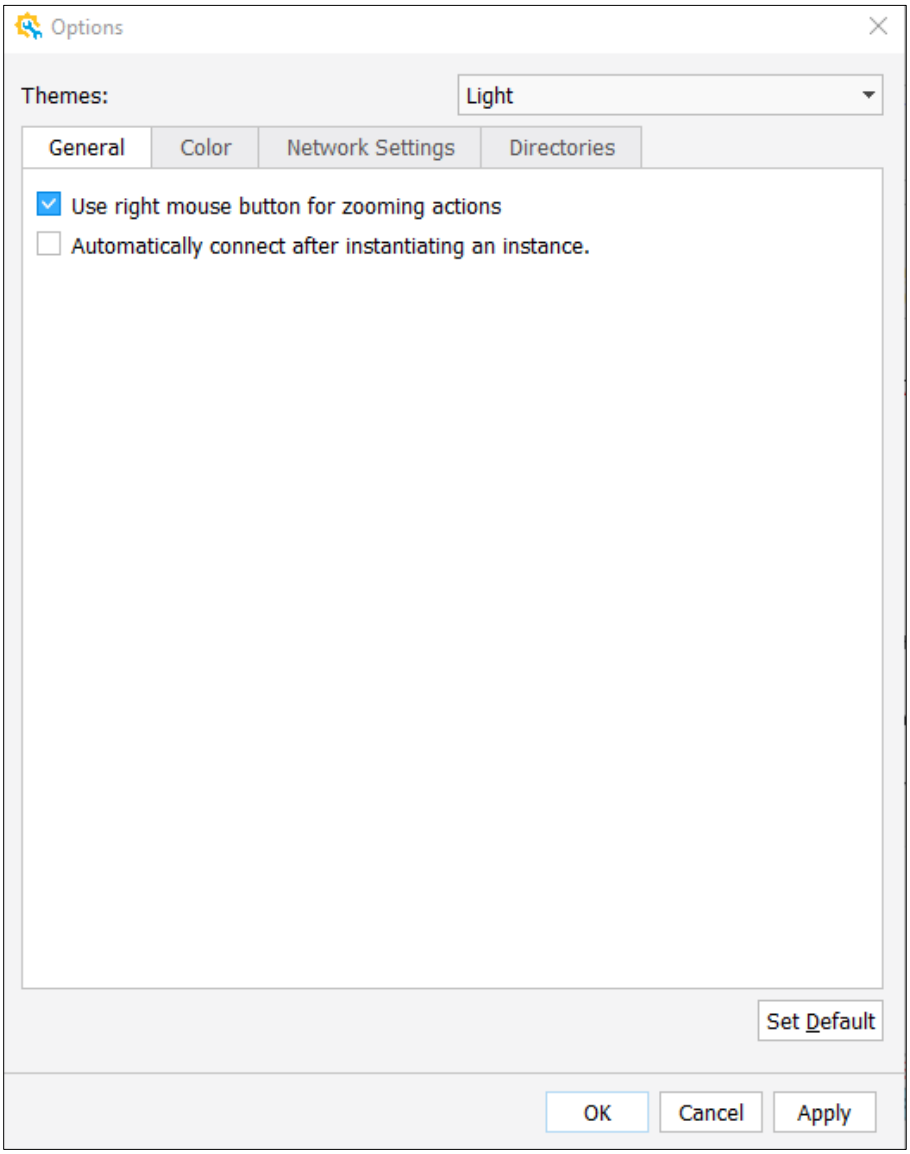


Figure 2.39. Options Dialog

2.2.5.9. Move a Schematic Image

You can move a schematic image within the Schematic view by panning and scrolling:

- To pan the image: Hold down the **Ctrl** key and the left mouse button while dragging the image.
- To scroll vertically: Rotate the mouse wheel. Or click in the vertical scroll bar.
- To scroll horizontally: Hold down the Shift key and rotate the mouse wheel. Or click in the horizontal scroll bar.

2.2.5.10. Show the Connectivity of a Module

Right-click the module and choose **Show Connectivity**. All nets connected to the module, all pins and ports connected to the nets are highlighted (Figure 2.40).

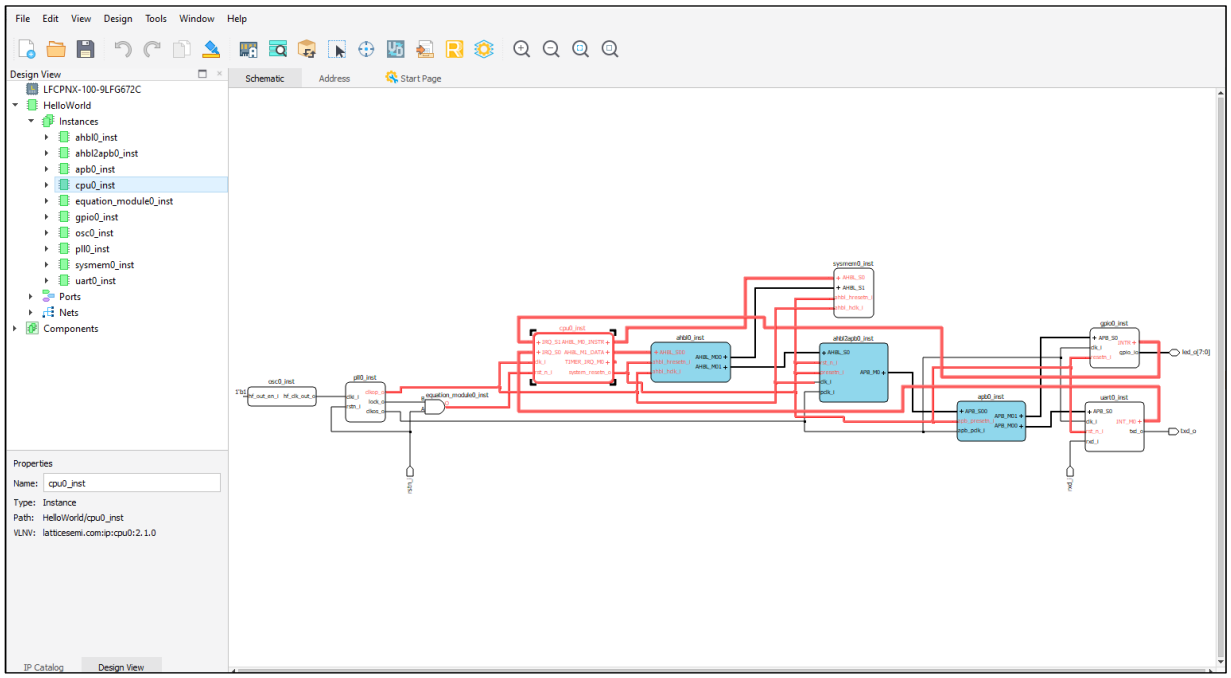


Figure 2.40. Show Connectivity of the Module

2.2.5.11. Highlight an Object

select an object and click **Highlight** , or, right-click the object and choose **Highlight** . The object is highlighted in blue (Figure 2.41). Click **Highlight** again. You can remove highlighting.

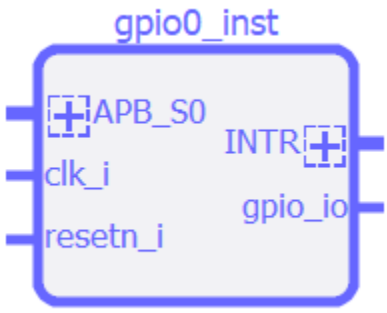


Figure 2.41. Highlight an Object

2.2.5.12. Change the Name of an Object

1. Select the object from List view of the Design View. Information of the selected object is shown in the **Properties** area (Figure 2.42).

Figure 2.42. Object Properties

2. Change the name and click **Enter.** The name changes in the Schematic view and List view of Design Info.

2.2.5.13. Print a Schematic

1. Choose **File** > **Print Preview**. The Print Preview window (Figure 2.43) opens.

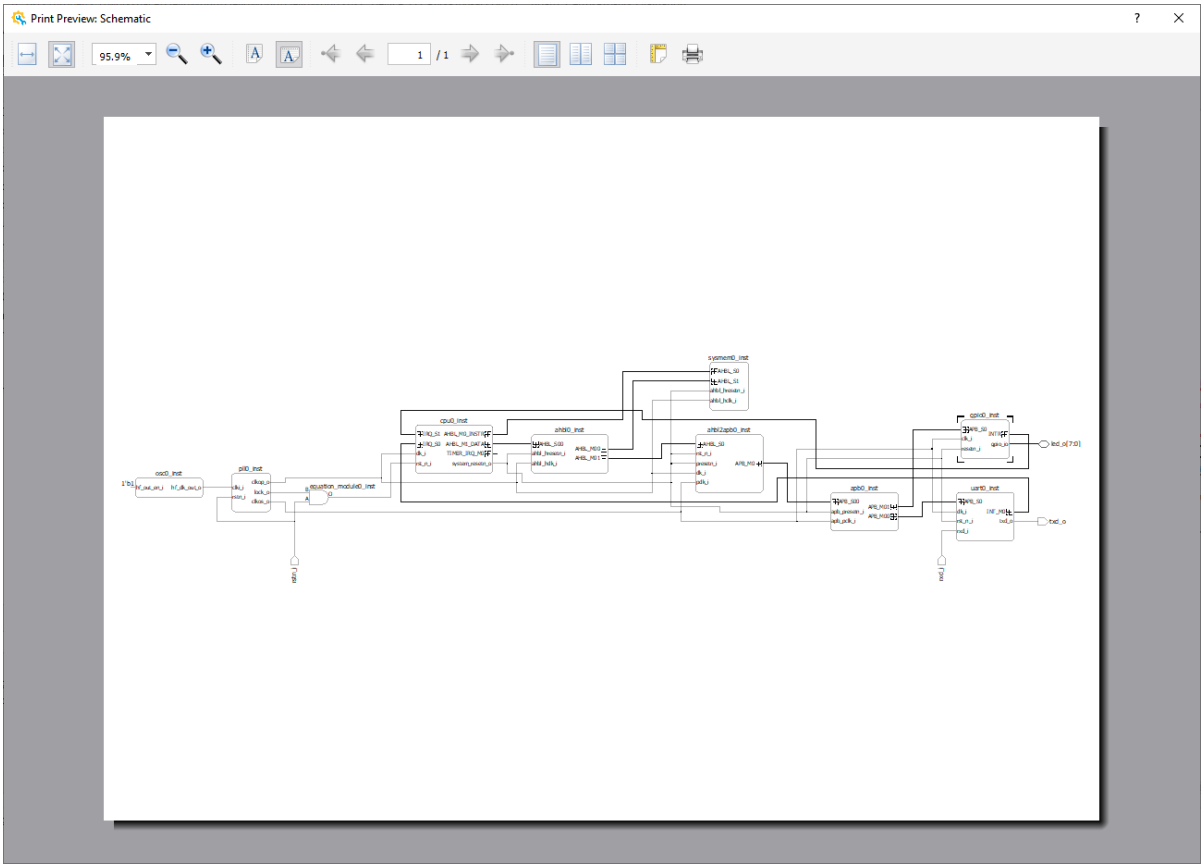


Figure 2.43. Print Preview

2. Expand the Print Preview window to the desired size.

3. Click the **Page Setup** button and adjust the paper size and margins, if necessary.
4. Click the **Print** button .
5. Adjust the printer settings, if necessary. Click **Print**.

2.2.6. Connecting Modules

You can connect the pins of modules to other modules or to top-level ports by dragging a line between them or by selecting connection points, or by assigning a constant value to an input pin or bus. Propel Builder does not allow obvious inappropriate connections, such as a connection between two output pins or mismatched buses.

Note: All AHB/APB connections of IP should be connected to the Processor so as to avoid error during the synthesis process in Radiant.

2.2.6.1. Connect Modules by Drawing

1. Move the cursor to a pin or port. The cursor changes to a pencil icon . Click and hold while dragging to another pin, port, or net. An allowed pin or port shows a green checkmark when you hover over it (Figure 2.44). An allowed net becomes bold when you hover over it (Figure 2.45).

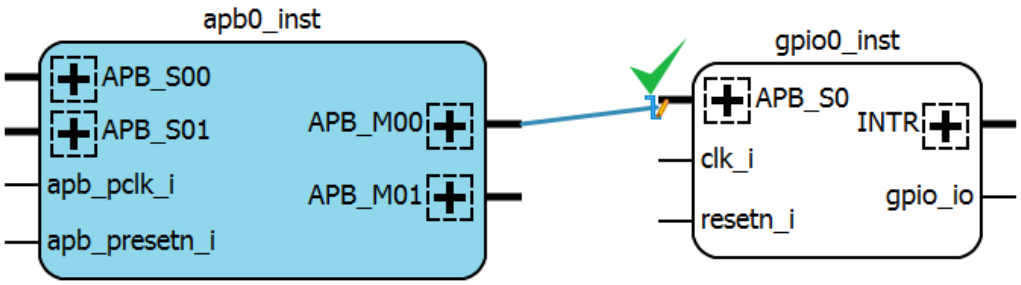


Figure 2.44. Draw a Pin or a Port

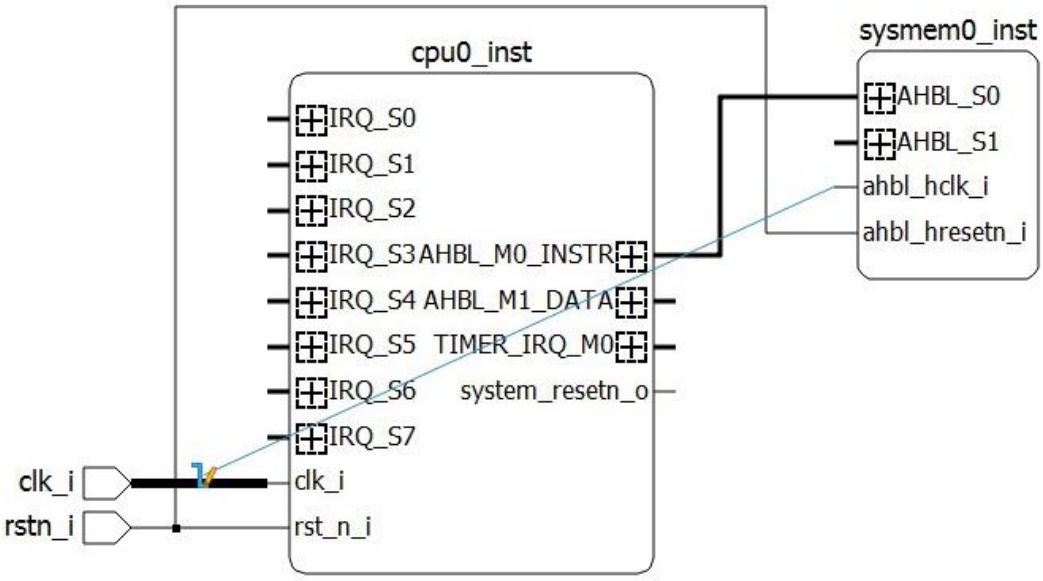


Figure 2.45. Draw Nets

2. Click on the pin, port, or net that you want to connect to. If the connection is allowed, a line appears connecting the two objects. Propel Builder creates a path around other objects.

3. You can connect multiple ports once. When more than one port are selected (Figure 2.46), click **connect** from the right-click menu to implement it.

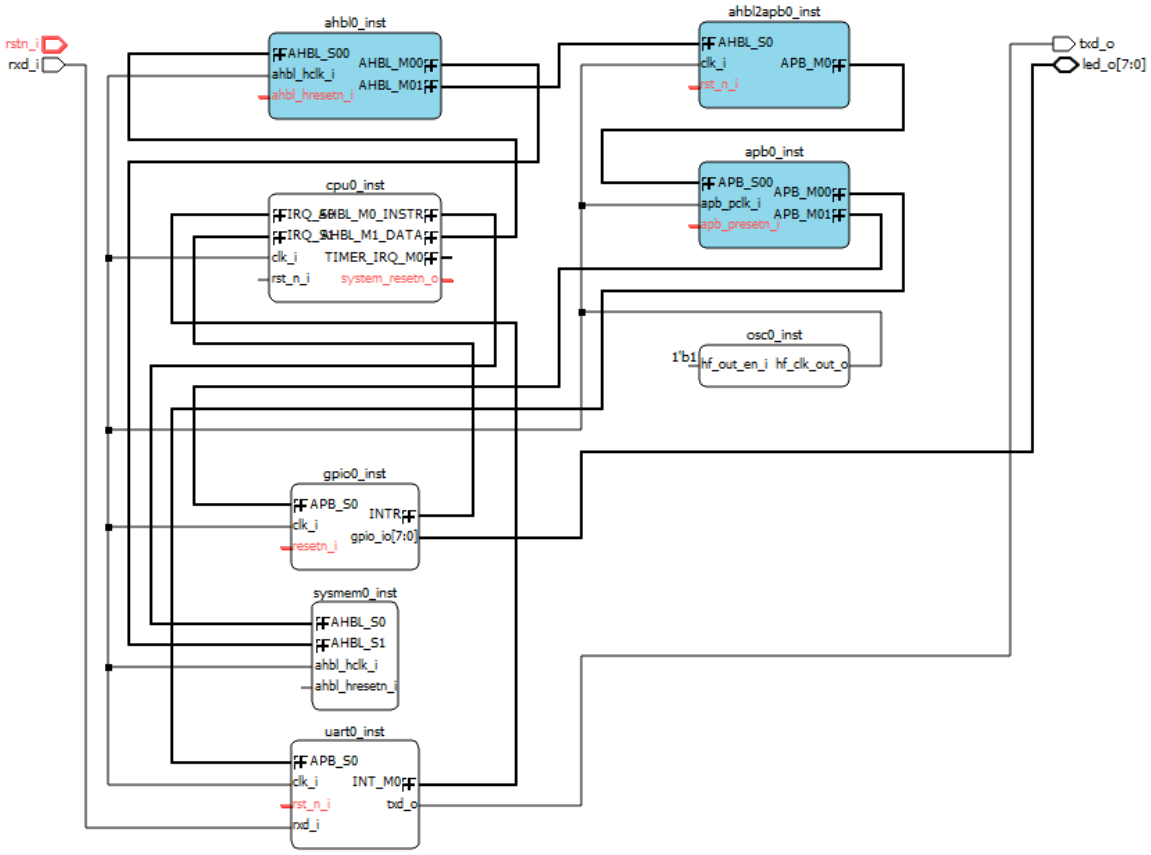


Figure 2.46. Select More than One Ports

4. When the connection is complete, right-click to leave the drawing mode.

2.2.6.2. Connect Modules by Selecting Points

1. Select the pins, ports, and nets you want to connect.
2. Right-click one of the selected objects. Choose **Connect** .

2.2.6.3. Connect Pins by Auto Connect

1. Right-click on Schematic view and choose **Auto Connect** (Figure 2.47). The Connect Ports dialog appears (Figure 2.48).

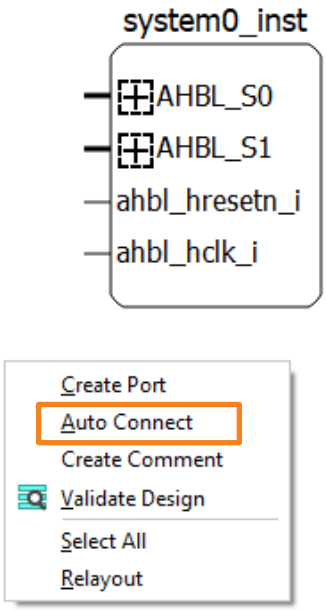


Figure 2.47. Action Menu of Right-clicking on a Blank

2. By default, **Auto Connect** selects all ports. You can also deselect the interface/clock/rest pins that you do not want to auto-connect (Figure 2.48).
3. Auto Connect can only connect one of its suggested sources at a time. For example, for PLL connections, we can only select clkos_o or clkop_o to connect to multiple destination ports, but we cannot make those connections at the same time.

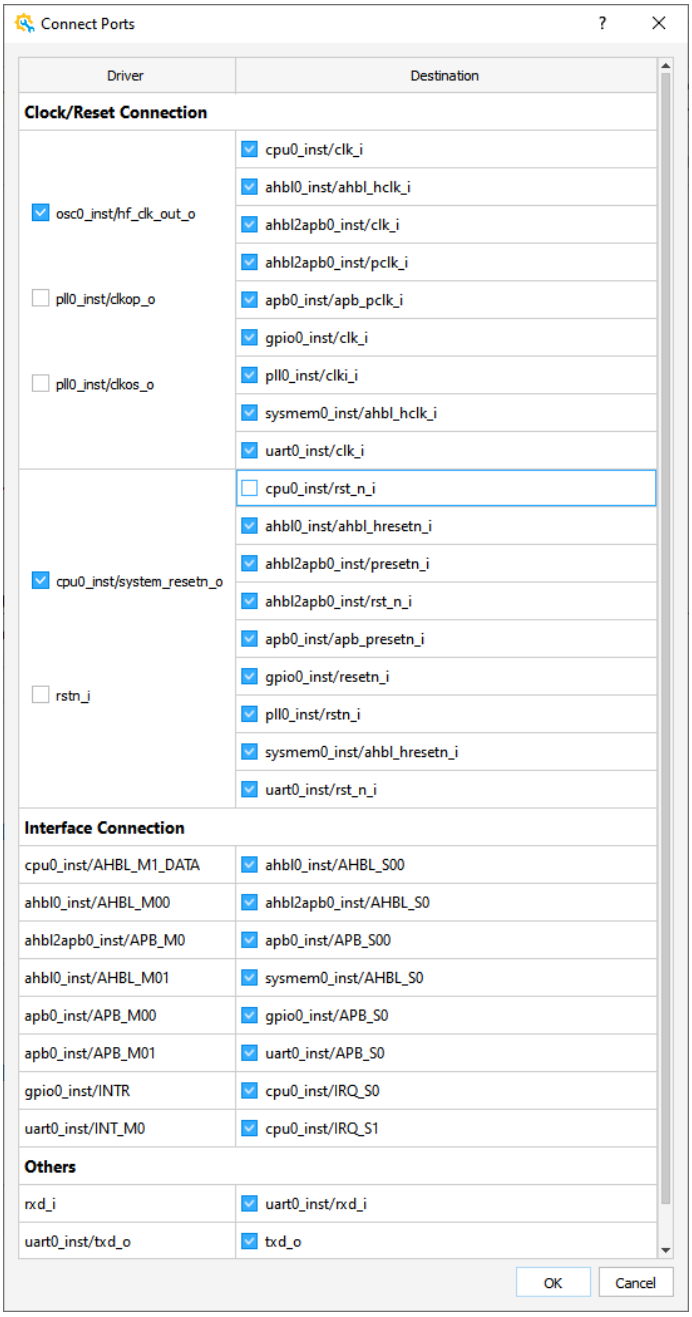


Figure 2.48. Action Menu of Connecting Ports

Auto Connect Rules

Auto Connect can initialize three types of connections: Clock/Reset connection, Interface connection, and other connection.

A. Clock/Reset Connection

For clock/reset connection, you can choose only one clock port and one reset port in the Driver selection area. In the example below, you can choose only one clock port from osc0_inst.hf_clk_out_o, pll0_inst.clkop_o, or pll0_inst.clkos_o, and one reset port from cpu0_inst.system_reset_n_0 or rstn_i (Figure 2.49).

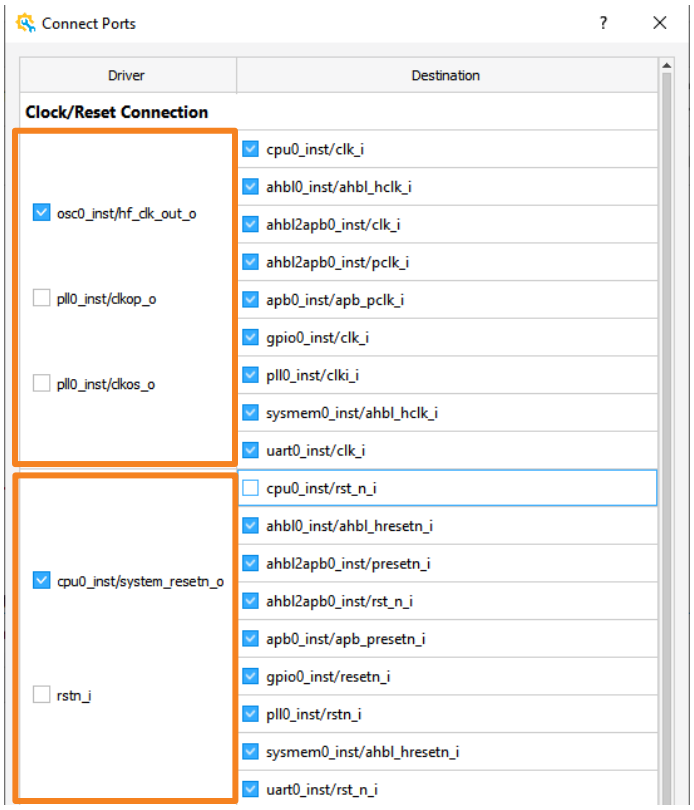


Figure 2.49. Select One Clock/Reset Port

Following are rules of how clock/reset driver port auto connects to the destination:

a. Whether or not the SoC project contains a CPU instance.

- If the SoC project contains a CPU instance, and the CPU instance has an output clock/reset port, such as cpu0_inst.system_resetn_o (Figure 2.50), the clock/reset of other instances are thus connected to the clock/reset port of the CPU Instance (Figure 2.50).

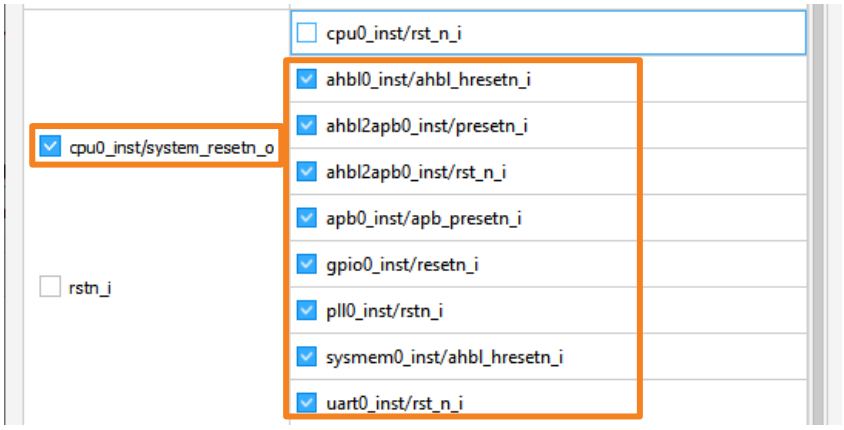


Figure 2.50. Connecting CPU Instance Reset Output Port

- If the SoC project does not contain a CPU instance, clock/reset on all instances connect to the top clock/reset port. Top clock/reset port is the port such as rstn_i in Figure 2.50.

b. The output clock/reset port on one instance should not be connected to its own clock/reset input port. For example, cpu0_inst.system_resetn_o should not connect to cpu0_inst.rst_n_i in Figure 2.50.

B. Interface Connection

Bus interfaces can only connect from one interface to another. It is a one-to-one relation, not a one-to-multiple relation. And the interfaces must have the same interface type.

In the example below (Figure 2.51), cpu0_inst.AHBL_M1_DATA is a bus interface. It can only connect to one destination, ahbl0_inst.AHBL_S00, because they have the same interface type AHBLite. To know the interface type, you can check the **Bus Type** in **Design View** (Figure 2.51).

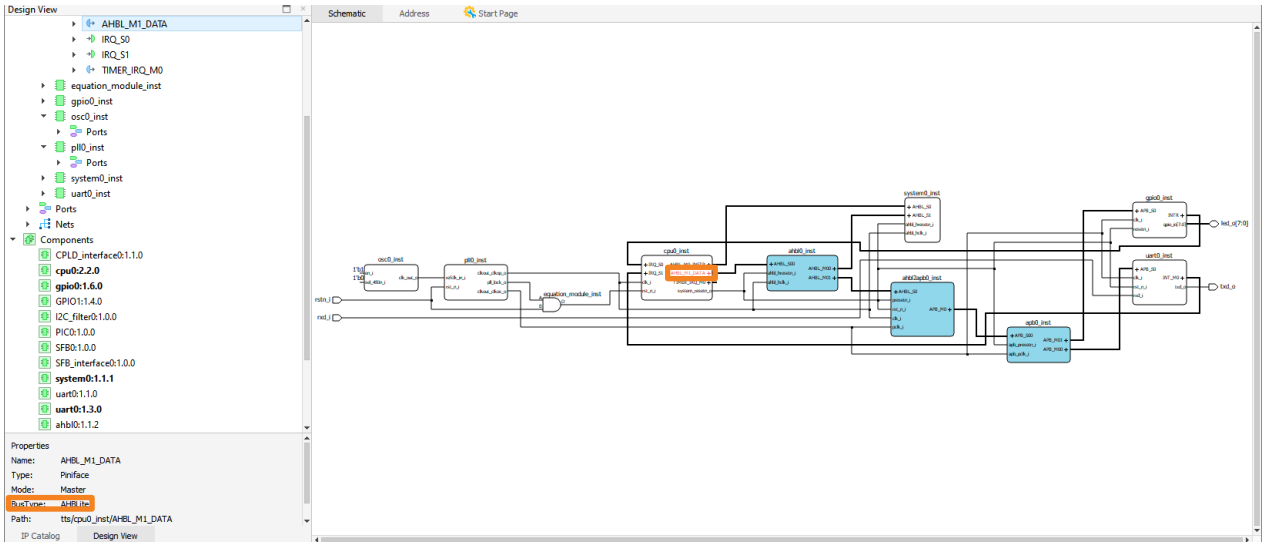


Figure 2.51. Interface Type of cpu0_inst.AHBL_M1_DATA

Interface Connects Rules

A. If a project contains a CPU instance, the interface connection starts from the CPU instance, then to one bus/bridge instance, and then to another bus/bridge instance. After connecting all bus/bridge instances one by one (cpu0 > ahbl0 > ahbl2apb0 > apb0), finally connects to other instances (gpio0, uart0). See Figure 2.52 and Figure 2.53.

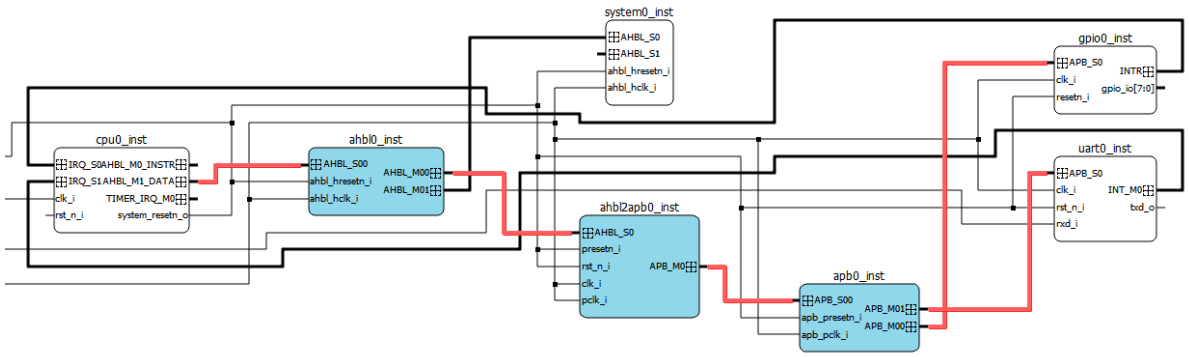


Figure 2.52. Interface Connection on from CPU to Bus/Bridge instances in Schematic View

Interface Connection	
cpu0_inst/AHBL_M1_DATA	ahbl0_inst/AHBL_S00
ahbl0_inst/AHBL_M00	ahbl2apb0_inst/AHBL_S0
ahbl2apb0_inst/APB_M0	apb0_inst/APB_S00
ahbl0_inst/AHBL_M01	sysmem0_inst/AHBL_S0
apb0_inst/APB_M00	gpio0_inst/APB_S0
apb0_inst/APB_M01	uart0_inst/APB_S0
gpio0_inst/INTR	cpu0_inst/IRQ_S0
uart0_inst/INT_M0	cpu0_inst/IRQ_S1

Figure 2.53. Interface Connection from CPU to Bus/Bridge Instances in Connect Port View

- If the last instance connected has the same bus interface type as that of the CPU instance, then it connects back to CPU instance (see Figure 2.54 and Figure 2.55). In Figure 2.56 and Figure 2.57, you can see gpio0_inst.INTR and cpu0_inst.IRQ_S0 have the bus type: Interrupt.

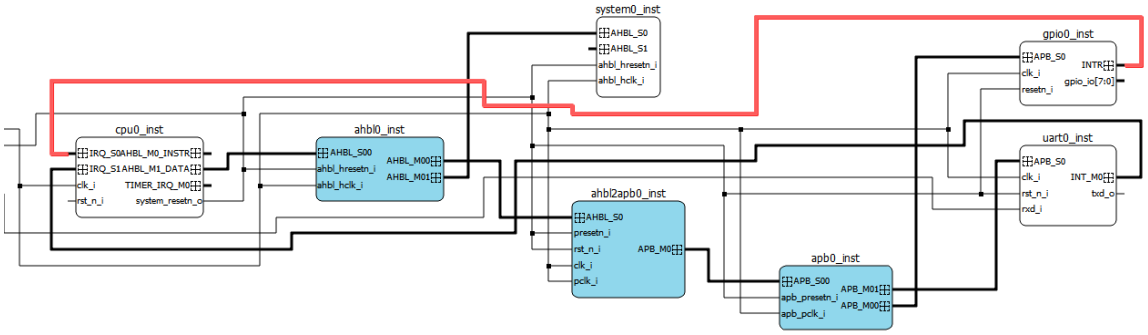


Figure 2.54. Interface Connection on Other Instances Connecting back to CPU in Schematic View

Interface Connection	
cpu0_inst/AHBL_M1_DATA	ahbl0_inst/AHBL_S00
ahbl0_inst/AHBL_M00	ahbl2apb0_inst/AHBL_S0
ahbl2apb0_inst/APB_M0	apb0_inst/APB_S00
ahbl0_inst/AHBL_M01	sysmem0_inst/AHBL_S0
apb0_inst/APB_M00	gpio0_inst/APB_S0
apb0_inst/APB_M01	uart0_inst/APB_S0
gpio0_inst/INTR	cpu0_inst/IRQ_S0
uart0_inst/INT_M0	cpu0_inst/IRQ_S1

Figure 2.55. Interface Connection on Other Instances Connecting back to CPU in Connect Port View

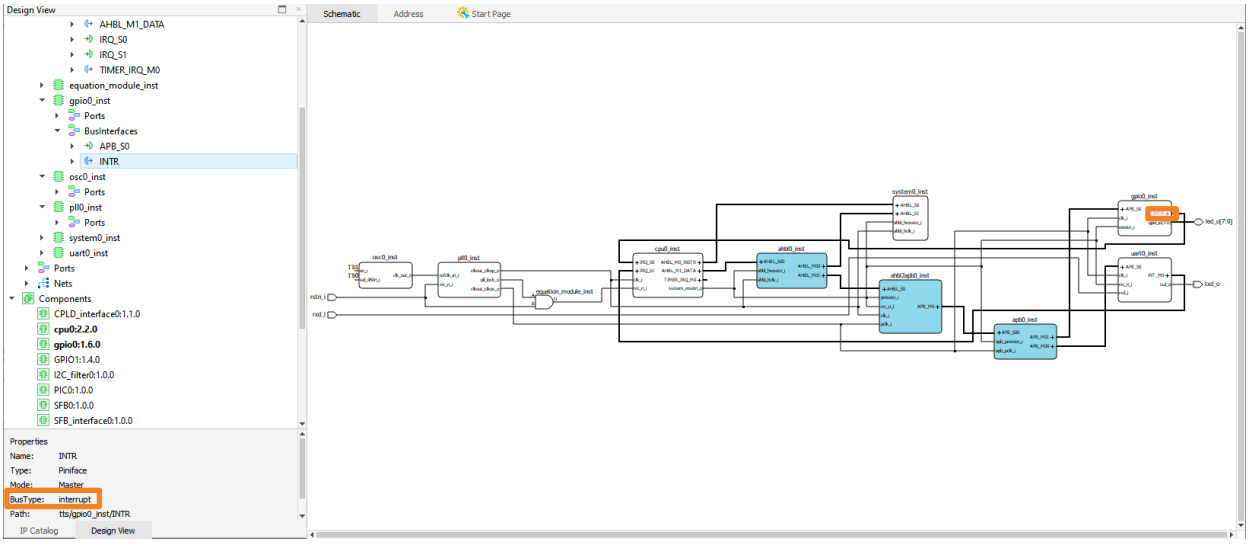


Figure 2.56. Interface Type of gpio0_inst.INTR

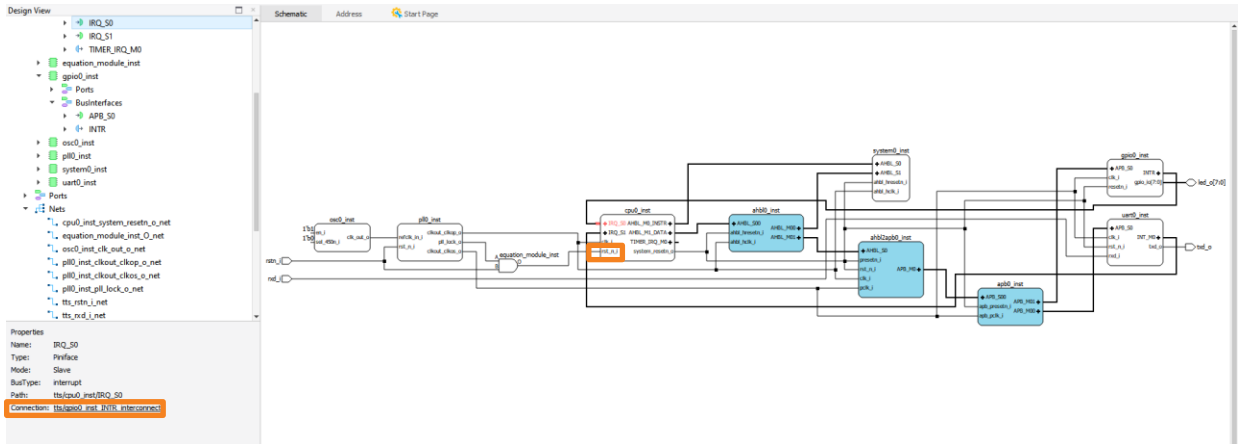


Figure 2.57. Interface Type of cpu0_inst.IRQ_S0

- If the project does not contain CPU instance, the interface connection starts from the bus/bridge instance, and then to each bus/bridge instance one by one, and finally connects to the other instances.

Note: If the bus interface is an instruction set, such as CPU0_inst.AHBL_M0_INSTR, it does not perform auto-connect.

C. Other Connection

If ports have the same port name, for example, rxd_i and uart0_ins.rdx_i, they are auto connected.

2.2.6.4. Assign a Constant Value to an Input Pin

1. Select the desired pin and right-click the pin. Choose **Assign Constant Value** (Figure 2.58).

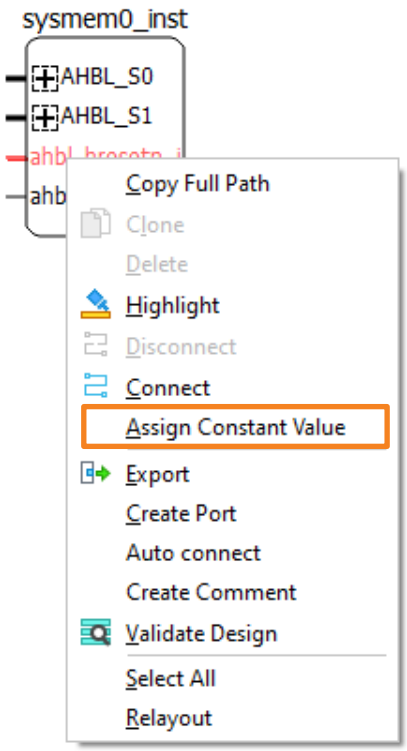


Figure 2.58. Right-click Menu of an Input Pin

2. A small dialog box appears (Figure 2.59). Enter the desired value. To erase the value and start over, click X. For all buses (more than one pin), the format is hexadecimal. Make sure the value fits the number of pins in the bus. For example, a 3-pin bus can accept 0-7, but not 8.

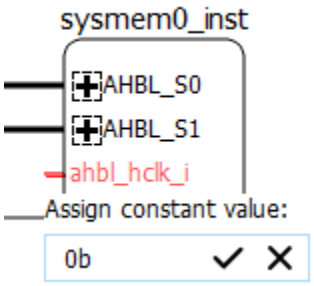


Figure 2.59. Dialog Box of Assigning Constant Value to an Input Pin

2.2.6.5. Disconnect Modules

Usually, disconnecting modules means deleting a net. If the net has multiple branches, just delete one branch, leaving the rest of the nets intact.

- To disconnect one branch of a net, right-click the pin of that branch and choose **Disconnect**. The branch going to that pin disappears.
- To disconnect a whole net, right-click the net and choose **Delete**. The whole net is deleted.

2.2.7. Creating Top-level Ports

With multiple modules in a Propel Builder project, top-level ports need to be created for a complete Propel Builder module. You can create a top-level port either manually or automatically.

For input ports that connect to more than one pin, such as for clock and reset signals, the manual way is more convenient than the automatic way. If the port names automatically-generated need to be changed, the manual way is better.

For other pins, the automatic way is usually preferred. The automatic way enables you to create the appropriate port type and connect net simultaneously. The automatic way can also create multiple ports at the same time.

The automatic way of creating ports is the most effective. If all the modules are selected, Propel Builder can automatically create ports for all the remaining unconnected pins for selected modules in one step.

- To create a port for a pin or bus manually:
  a. Right-click in the Schematic view, and choose **Create Port**. The Create Port dialog box pops up (Figure 2.60).

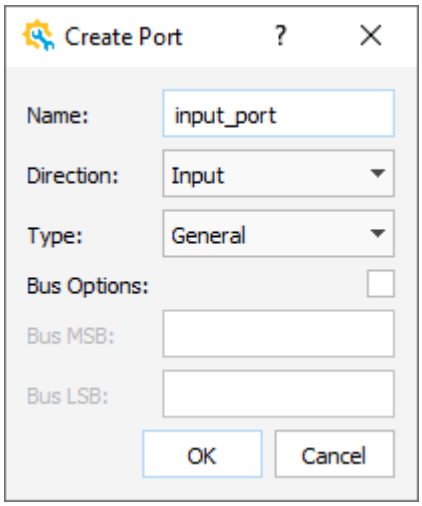


Figure 2.60. Create Port Dialog Box

  b. Enter a name for the port in the **Name** field.
  c. Choose a direction for the port, such as Input in the **Direction** field.
  d. Choose the port type from the **Type** field. The default type is General.
  e. (Optional) Select the **Bus Options**. Enter the number for the most significant bit (MSB) and the least significant bit (LSB). These two options define the bus width.
  f. Click **OK**. The port appears. Figure 2.61 shows an input port, port name of which is on the left. Figure 2.62 shows an output port and an inout port, port names of which are on the right.

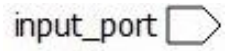


Figure 2.61. Input Port

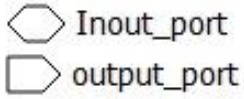


Figure 2.62. Output Port and Inout Port

  g. Connect the port to the module pins. Refer to the Connecting Modules section for more details.

- To create ports automatically:
  a. Select the pins that are to be connected to the top-level ports. All the pins in a module can be selected by clicking its block. Any pins that are already connected to a net or a constant value are skipped.
  b. Right-click on one of the selected pins, interfaces or modules, and choose **Export**. The selected pins are extended by lines to new top-level port symbols. The names of the ports and nets are added to the List view. Zoom out or scroll the image in the Schematic view to see the new ports.
  c. Port or net names can be changed, if needed.
- To modify a port:
  a. Click on a port, **Properties** window open (Figure 2.63).

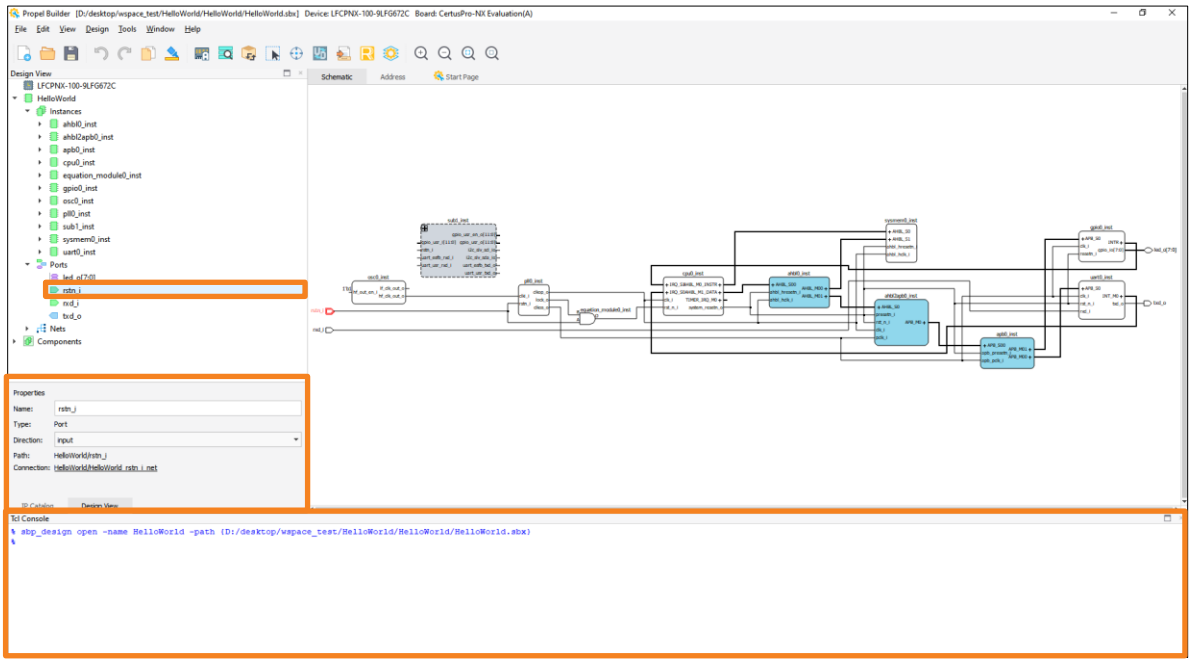


Figure 2.63. Properties of Port

  b. Port direction can be modified. You can use the drop-down menu to choose the desired port direction. TCL command is shown in **Design View** (Figure 2.64).

  Note: Port width can only be modified if it is already multi-bit. An error message is prompted out or shown in TCL console, if the input value does not meet rules (Figure 2.64).

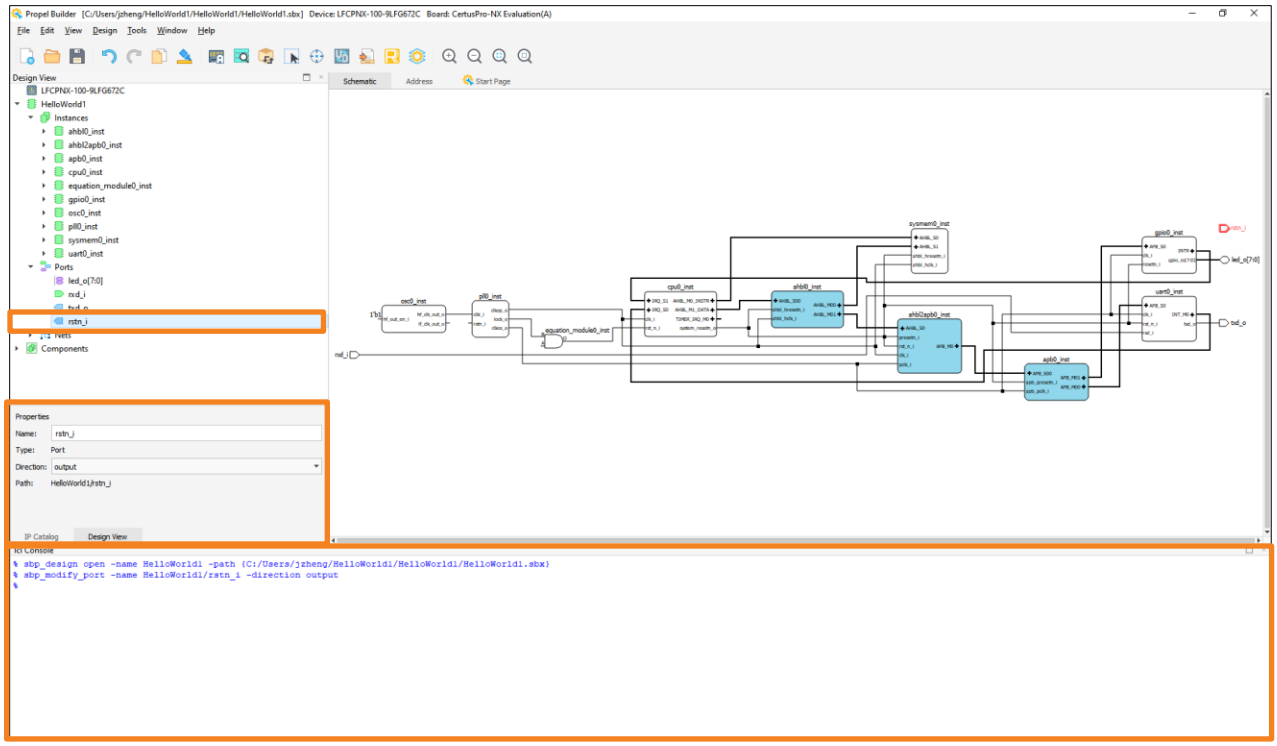


Figure 2.64. Port Direction

c. PortBus MSB, LSB and direction can be modified. Click on PortBus and then use the drop-down menu to choose the desired direction (Figure 2.65).

Note: If the input value does not meet rules, an error message prompts out or shows the in TCL console. .

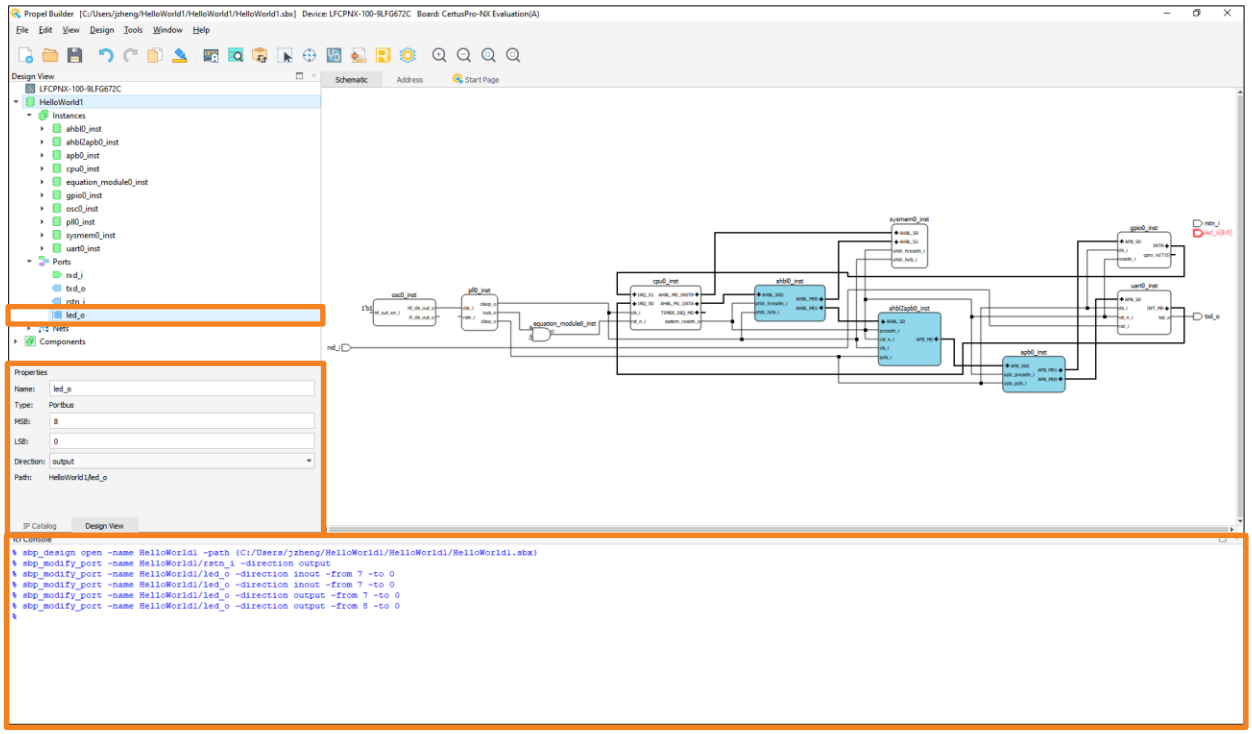


Figure 2.65. PortBus Direction

2.2.8. Adjusting Address Spaces

The Address view shows the base address, size of the address segment, and the end address for each leaf memory-mapped slave connection in the Propel Builder project. Propel Builder can automatically assign address values, while the base address value can be changed manually. The addresses are automatically assigned when major and slave components were connected. The ranges are set when the modules are configured. The end addresses are calculated.

The Lock option on each address space prevents Auto Assign from changing the base address value. The Lock option is selected automatically when you manually change the address value. To reset the address space, clear the Lock option before clicking the Auto Assign icon.

Note: There is no Lock option on LocalMemory. The value of the base address can always be changed, while Auto Assign does not reset it to its original value.

1. In the Propel Builder main window, click the Address tab. The Address view (Figure 2.66) shows.

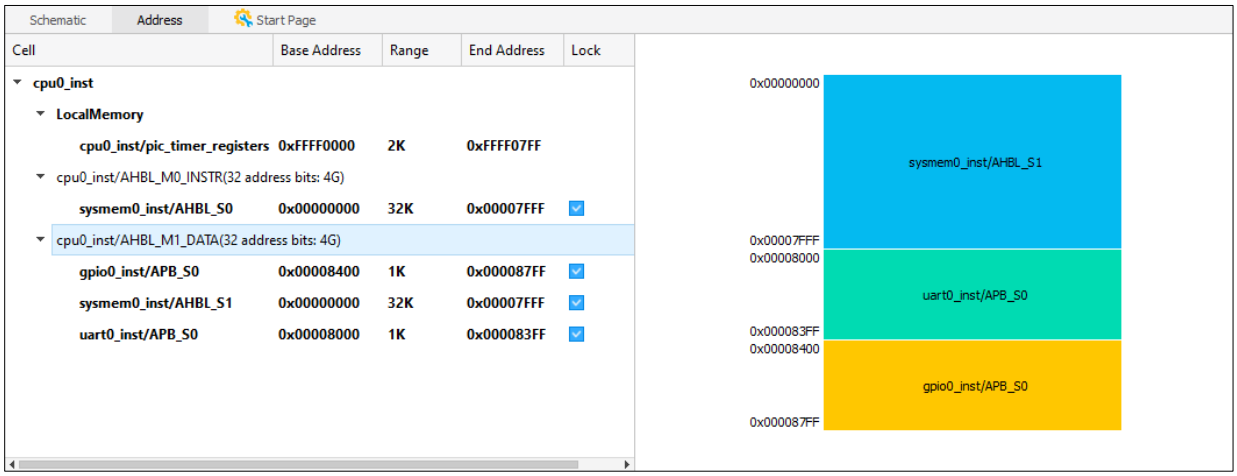


Figure 2.66. Address View

2. (Optional) Set or clear the Lock options as desired in Address view.
3. (Optional) Double-click the base address in Address view (Figure 2.67). Type the new value and press **Enter**. Values must align with 1K boundaries, such as 0x00000400, 0x00000800, or 0x00000C00. The end address value changes based on the new value. The Lock option is selected automatically at this time.

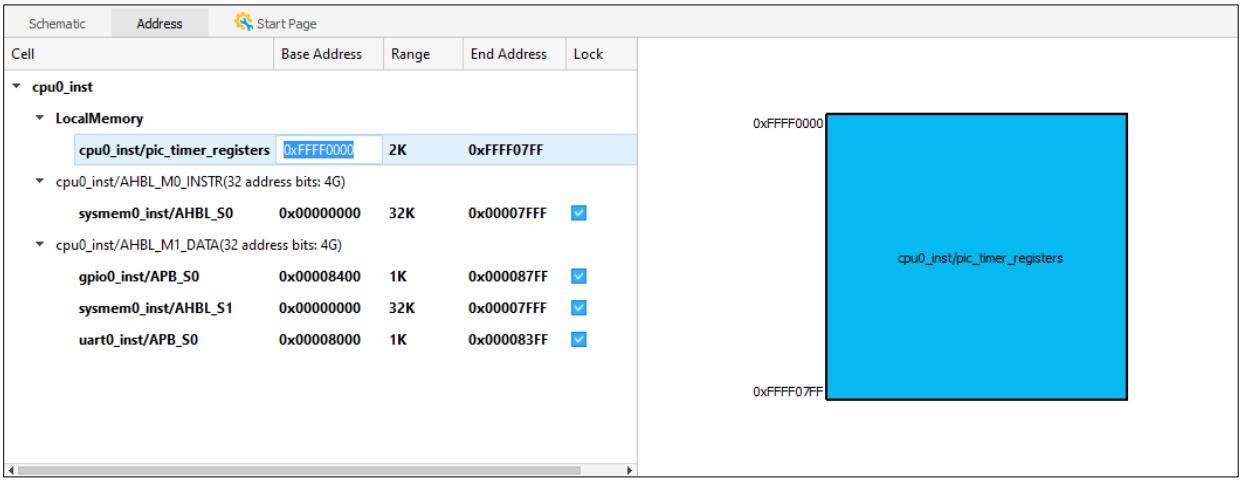


Figure 2.67. Edit Base Address

Note: If there is a conflict of a related address space, it is shown in red in the graphic.

2.2.9. Validating the Design

The design rule check (DRC) can be run at any time. It can check if there is any illegal connection or overlapping address space.

- To perform the design rule check:

From the toolbar of the Propel Builder main window, click the **Validate Design** icon  . The DRC results appear in the Tcl Console.

2.2.10. Generating the RTL file

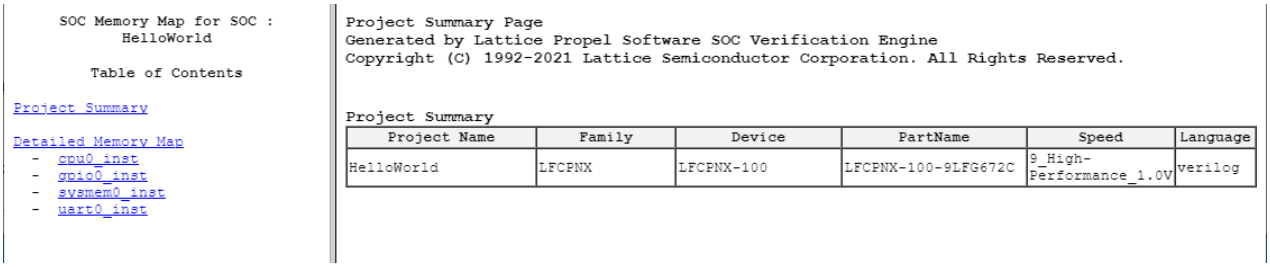
To create a Propel Builder module is to generate a .sbx file, which defines a Propel Builder project, an RTL file, and the instantiation templates. The RTL file includes the Verilog code for the module. The instantiation templates have Verilog and VHDL code to help instantiate the Propel Builder module in a design.

- To generate RTL:

From the toolbar of the Propel Builder main window, click the **Generate** icon  . This saves the Propel Builder design and runs the design rules check (DRC).

2.2.11. Generating the Memory Report

From the toolbar of the Propel Builder main window, click the **Generate Memory Report** icon  . The SoC design memory information is generated. The memory report folder (mem_report) is generated. All design memory information is shown in detail in the memory report files. The Memory Report files are in HTML format (Figure 2.68). To see more about address space, refer to the Adjusting Address Spaces section.



SOC Memory Map for SOC : HelloWorld

Table of Contents

Project Summary

Detailed Memory Map
- cpu0_inst
- gpio0_inst
- sysmem0_inst
- uart0_inst

Project Summary Page
Generated by Lattice Propel Software SOC Verification Engine
Copyright (C) 1992-2021 Lattice Semiconductor Corporation. All Rights Reserved.

Project Summary

Project Name	Family	Device	PartName	Speed	Language
HelloWorld	LFCPNX	LFCPNX-100	LFCPNX-100-9LFG672C	9_High-Performance_1.0V	verilog

Figure 2.68. Memory Report

2.2.12. Launch Project in Diamond or Radiant

In a complete SoC project, a Diamond or Radiant project can be created upon a Propel Builder design, and after that the SoC project can be launched in Diamond or Radiant software.

According to the device in the SoC project, Propel Builder can launch Diamond or Radiant software accordingly.

For Diamond support device, the **Diamond** software icon  shows in the Builder GUI Toolbar.

For Radiant support device, the **Radiant** software icon  shows in the Builder GUI Toolbar.

Note: Different device families are supported and available in Radiant/Diamond, as shown in Table 2.1.

Table 2.1. Different Device Families Supported in Radiant or Diamond

Software	Project Name	Part Number
Radiant	CertusPro-NX	LFCPNX
	Certus-NX	LFD2NX
	MachXO5-NX	LFMXO5
	CrossLink-NX	LIFCL
Diamond	MachXO2	LCMXO2
	MachXO3D	LCMXO3D
	MachXO3L	LCMXO3DL
	MachXO3LF	LCMXO3DLF
	MachNX	LFMNX

For a template SoC project, you can launch **Diamond** or **Radiant** software directly after a template SoC project is created. After launch, files from Propel Builder project are also exported to the resultant Propel SDK C/C++ project folder, such as BSP, drivers, sys_platform.h, sys_env.xml.

2.2.12.1. Launch Diamond Software

1. Click the Diamond icon  from the Propel Builder toolbar. Diamond software is launched. diamond_setup_template.tcl in project path is automatically invoked. The meaning of the tcl command in it can be found from **Help** in the menu bar.
2. Lattice Diamond is launched with a Diamond project generated for SoC at the background (Figure 2.69).

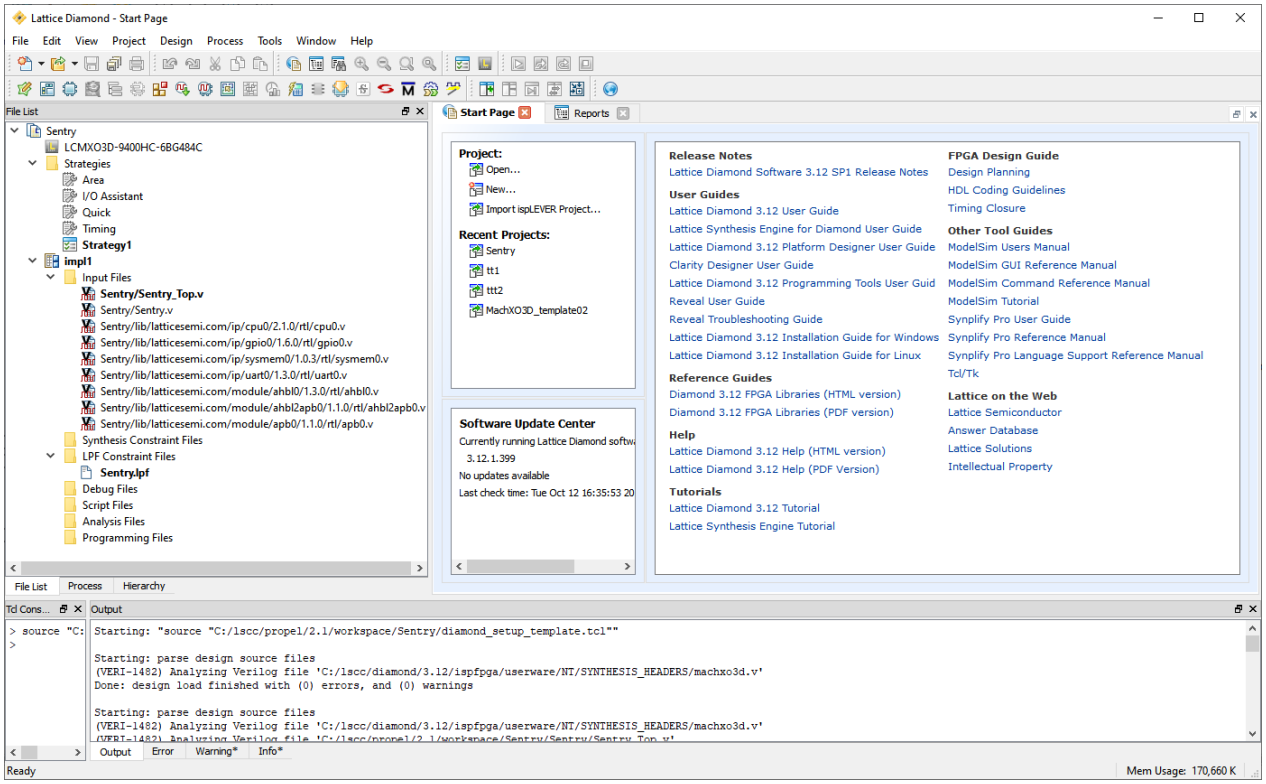


Figure 2.69. Diamond Project

3. (Optional) From the **File List** view of Diamond:
   - modify the top-level RTL file (<*proj_name*>_Top.v) to match the SoC design, presupposition of which is that there is a top-level RTL file in your SoC design.
     - create a top-level RTL file (<*proj_name*>_Top.v) to match the SoC design, if the SoC design is created from an Empty Project template and there is no top-level RTL file in your SoC design.

   a. (Optional) Modify LPF constraint file (<*proj_name*>.lpf) to match the SoC design, if you have modified the SoC design. This step is a must to the SoC design that is created from Empty Project template.

   b. Switch to **Process** view of the Diamond project (Figure 2.70). Make sure at least one programming file (IBIS Model, Verilog Simulation File, VHDL Simulation File, Bitstream File, or JEDEC file) is selected in the **Export Files** section. Available programming files can be different upon specific device included.

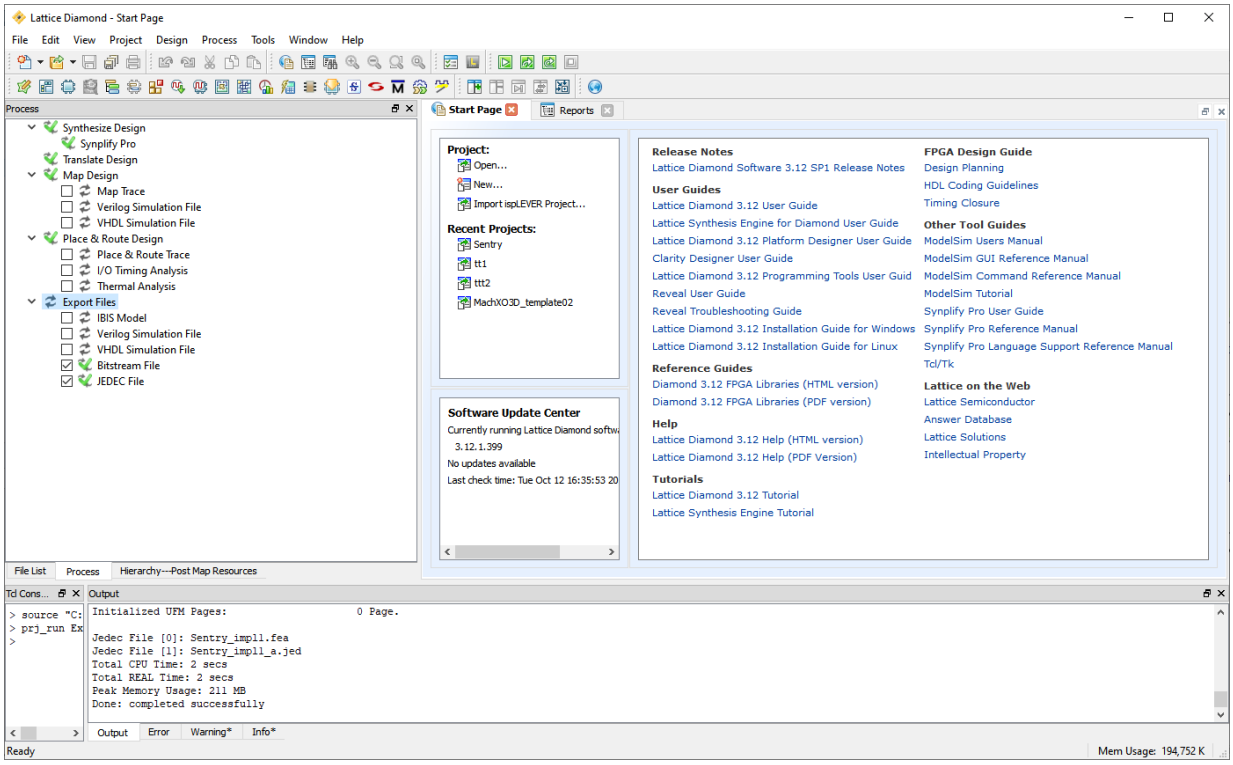


Figure 2.70. Generate Programming Files

   c. Right-click **Export Files** and choose **Run**  . The programming file is generated. The programming file can be used in the Diamond Programmer.

Notes:

- Refer to the Diamond online help for more details of the Diamond project. Re-launch the Diamond project, if the project settings are modified in Propel 2023.1 Builder.
- If there is encrypted RTL in diamond project, it is automatically detected in Diamond.
- Board template may have some constraint files. Constraint files are different per each type of the board. The constraint files are also exported to Diamond.

2.2.12.2. Launch Radiant Software

1. Click the **Radiant** icon  from the Propel Builder toolbar. Radiant software is launched. radiant_setup_template.tcl in project path is automatically invoked. The meaning of the tcl command in it can be found from **Help** in the menu bar.

2. Lattice Radiant is launched with a Radiant project generated for SoC at the background (Figure 2.71).

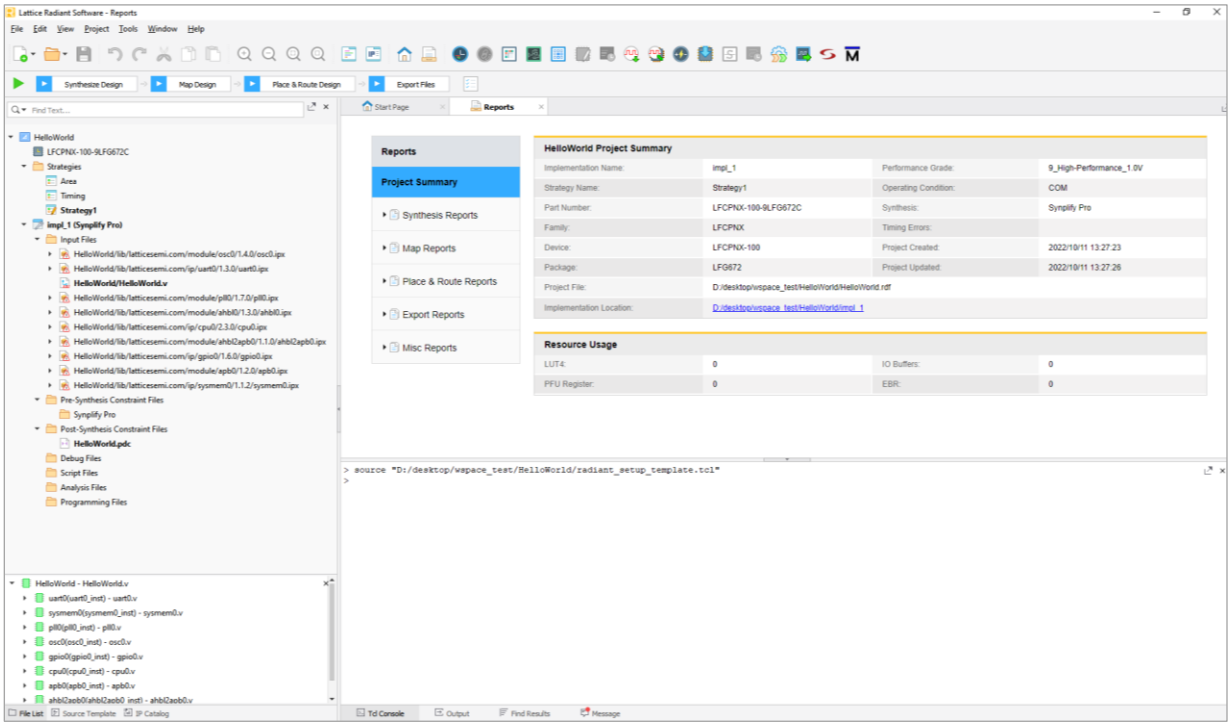


Figure 2.71. Radiant Project

a. (Optional) From the **File List** view of Radiant:
   - modify the top-level RTL file (<proj_name>_Top.v) to match the SoC design, presupposition of which is that there is a top-level RTL file in your SoC design; or
   - create a top-level RTL file (<proj_name>_Top.v) to match the SoC design, if the SoC design is created from an Empty Project template and there is no top-level RTL file in your SoC design.

b. (Optional) Modify pdc constraint file (<*proj_name*>.pdc) to match the SoC design, if you have modified the SoC design. This step is a must to the SoC design that is created from Empty Project template.

c. Click **Run all**. The Programming file is generated. It can be used in the Radiant Programmer.

Notes:

- Refer to the Radiant online help for more details of the Radiant project.
- Board template may have some constraint files. Constraint files are different per each type of board. The constraint files are also exported to Radiant.

2.2.12.3. Ipx Based Implementation Flow in Radiant

To enable Radiant constraint propagation flow, the design source files need to be switched from original HDL files to the .ipx file when generating the Radiant project.

Ipx based Radiant implementation flow is for project created in later than Propel Builder 2023.1. For project created before Propel Builder 2023.1, use a different method for exporting files. If you want to use ipx flow in these old projects, open them in Propel Builder 2023.1 and regenerate, then export to Radiant to change how it is exported to Radiant. (Figure 2.72).

Considering some IPs are available at both Propel and Radiant side, you should only update IP through ipx in Propel Builder side.

Note: You must avoid IP update (through ipx) in Radiant side. The potential out-of-sync of updating IP in Radiant side may cause unexpected issues.

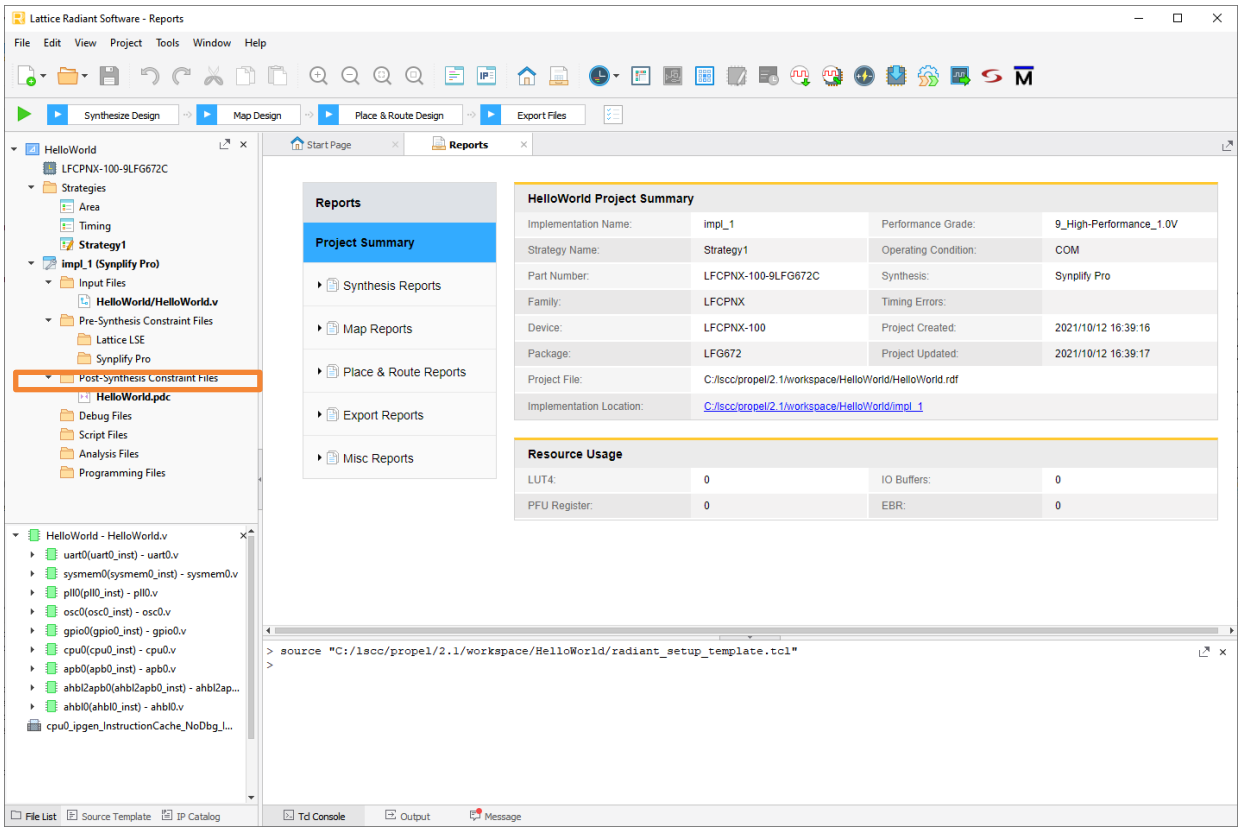


Figure 2.72. Radiant Project for Project before Propel Builder 2023.1

2.2.13. Launching SDK

After a SoC design is completed, Propel SDK can be launched in Propel Builder for software development. For template SoC project, you can launch SDK directly by clicking **Run Propel** icon from Propel Builder toolbar after template SoC project created.

1. Click the **Run Propel** icon  from the Propel Builder GUI Toolbar. Lattice Propel Launcher (Figure 2.73) opens.

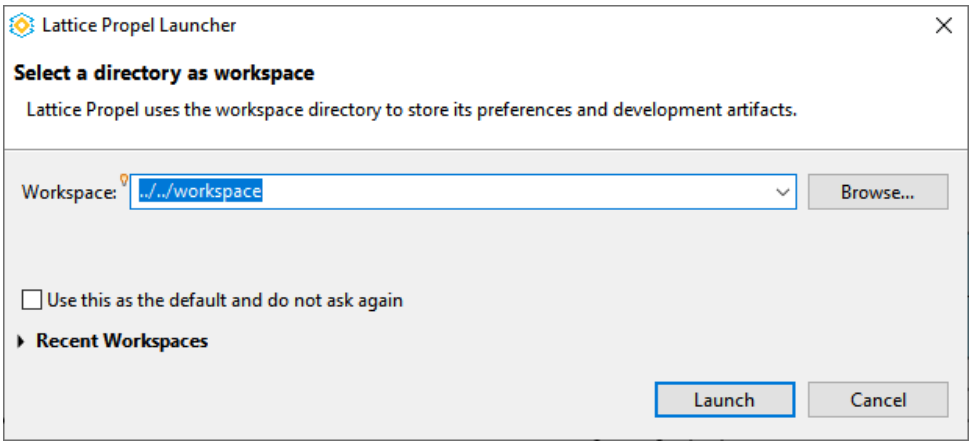


Figure 2.73. Lattice Propel Launcher Wizard

2. Click **Launch**. Propel SDK GUI opens and the C Project wizard (Figure 2.74) pops up for loading the system and the board support package (BSP) to create a C/C++ project. Enter a project name in the **Project Name** field, such as HelloWorld_C. Click **Next**.

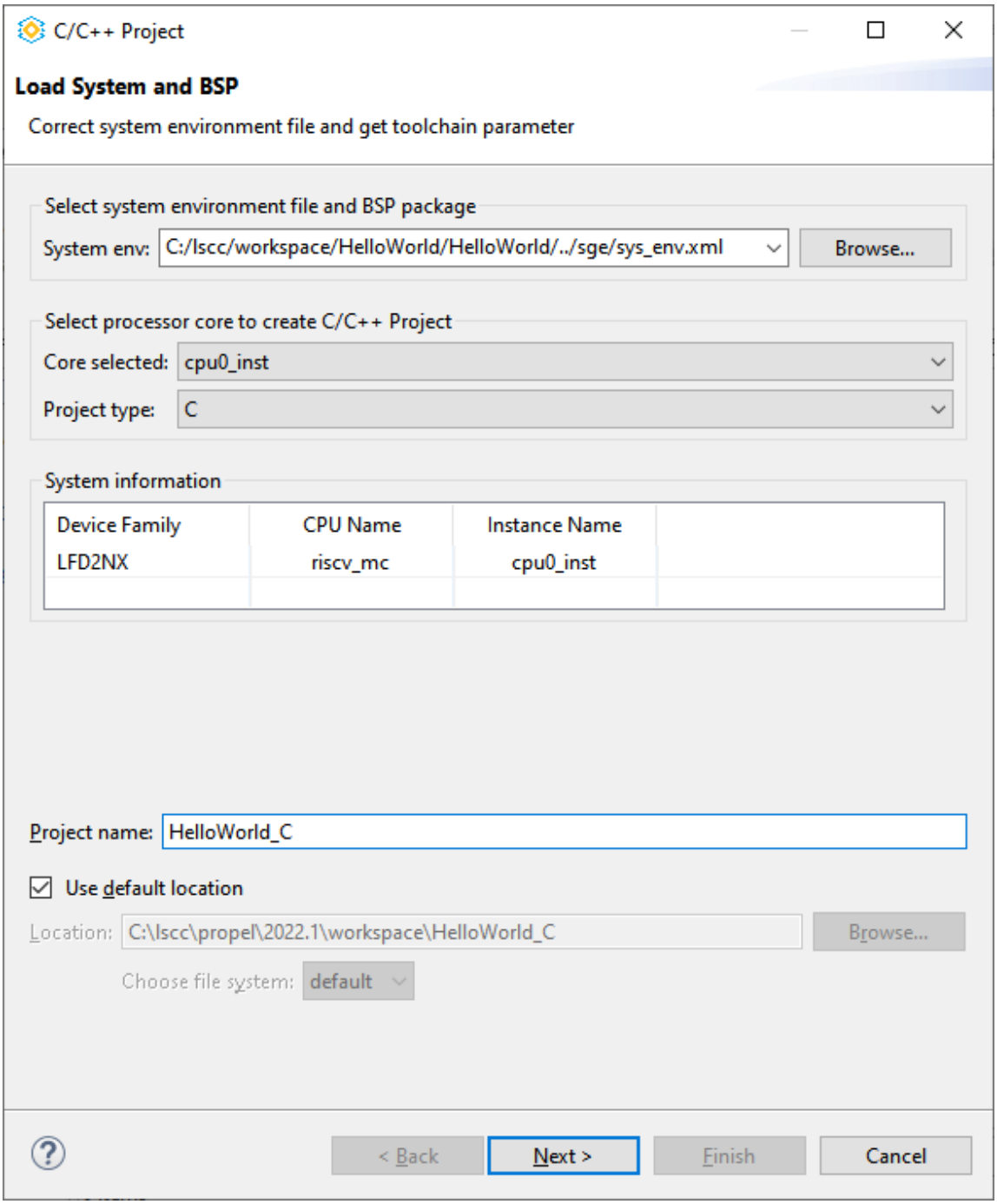


Figure 2.74. Propel SDK GUI and C/C++ Project Wizard

3. Click **Next**. The C/C++ project dialog (Figure 2.75) opens.

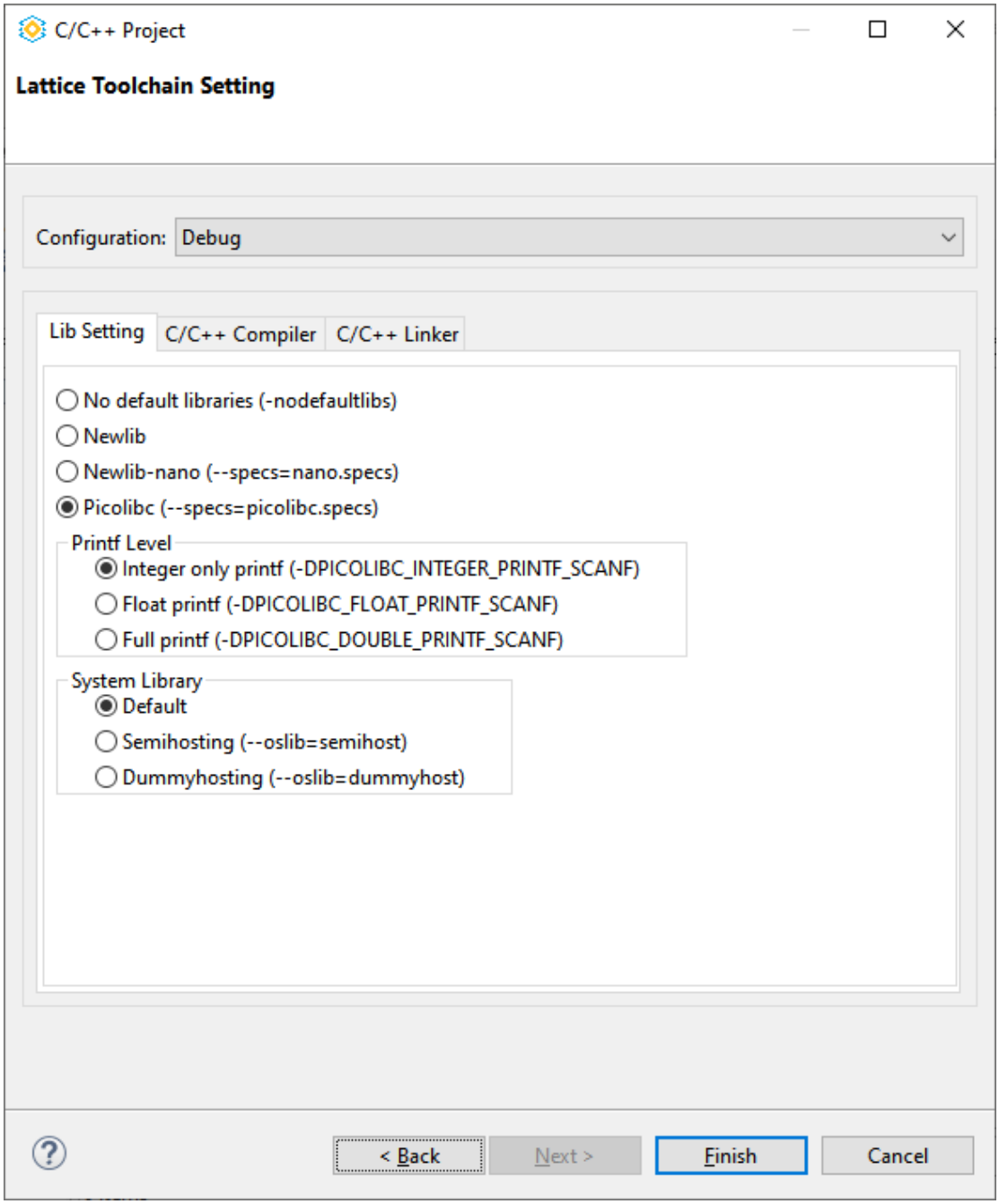


Figure 2.75. Create C/C++ Project

4. Click **Finish**. The C/C++ project is created and is displayed using the C/C++ perspective. A perspective is a collection of tool views for a particular purpose. The C/C++ perspective is for creating C/C++ programs.

Notes:

- Refer to Lattice Propel 2023.1 SDK User Guide (FPGA-UG-02186) for more details on how to create a C/C++ project and develop the C/C++ project in Propel SDK.
- For a SoC design, if the init file (mem file) of System Memory module instance is updated in a C project, the initialization section of the system memory module instance should be re-configured. Or use the ECO flow to update the information. Refer to the Diamond online help for more on how to use an ECO flow.

2.2.14. Others

2.2.14.1. Modifying the Project Settings

Propel 2023.1 Builder supports changing the projects settings. You can modify the device, package, speed and operating conditions by double clicking the device part number from the Design View.

1. Double click the device part number from the Design View (Figure 2.76), the Modify Device Info wizard pops up (Figure 2.77).

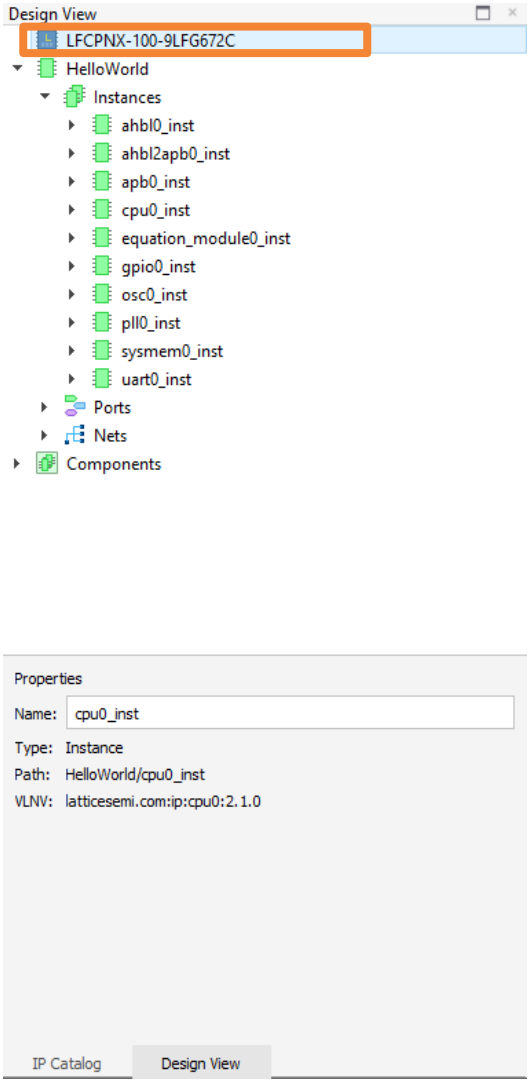


Figure 2.76. Design View of Device Part Number

2. Use the drop-down menu to choose the desired device in **Device** filed in the Modify Device Info wizard (Figure 2.77). Use the drop-down menu to change the package, speed and operating condition in **Package**, **Speed**, **Operating Condition** filed.

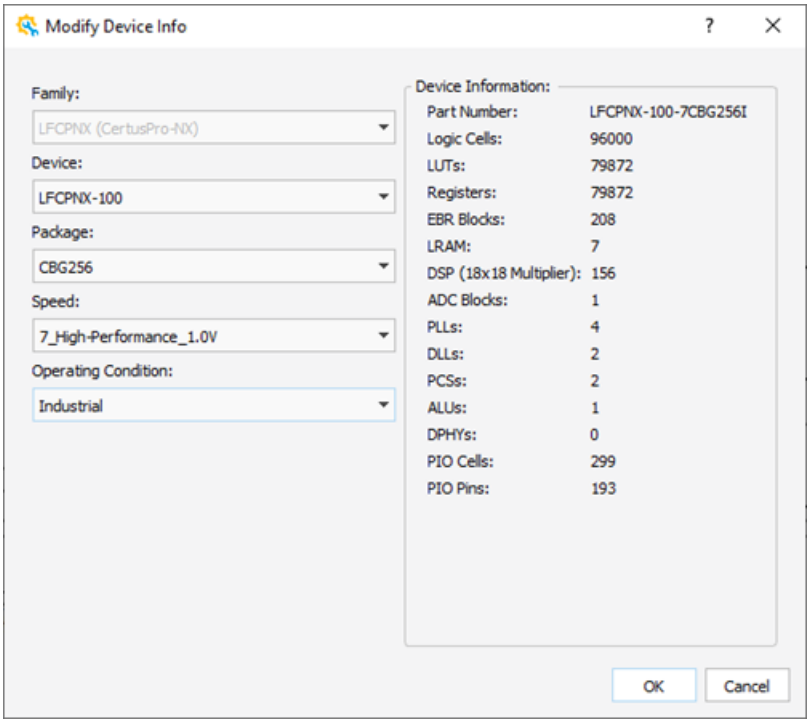


Figure 2.77. Modify Device Info

3. Click **OK**. The System Builder dialog (Figure 2.78) pops up showing a message of configuring all IPs.

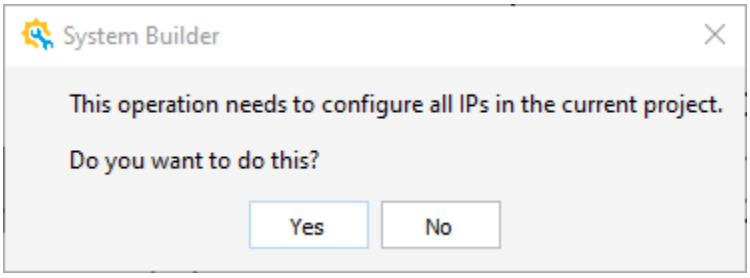


Figure 2.78. System Builder Dialog

4. Click **Yes**. By clicking **Yes**, the Builder engine regenerates each IP in the project with the updated device settings. All IPs in current project are configured. The new device part number shows in Design View (Figure 2.79).

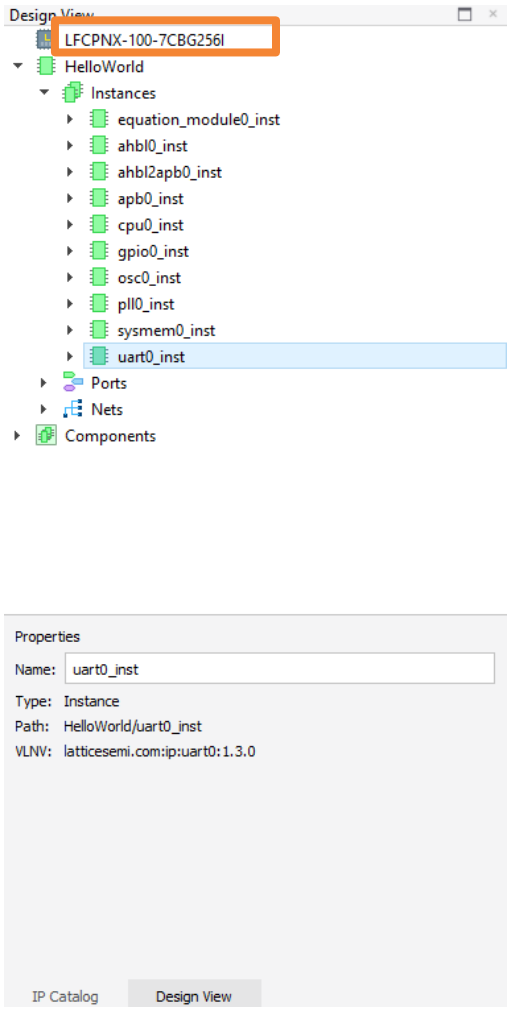


Figure 2.79. Design View of Device Part Number

2.2.14.2. Undo/Redo

Propel 2023.1 Builder supports one-level undo/redo. You can click the **Undo** icon from the Propel Builder Toolbar to go back to last action. Click the **Redo** icon from the Propel Builder Toolbar to recover last undo action.

- Undo: Choose **Edit** > **Undo** from the Lattice Propel Builder Menu bar. Or, click the **Undo** icon from Propel Builder GUI Toolbar.
- Redo: Choose **Edit** > **Redo** from the Lattice Propel Builder Menu bar. Or, click the **Redo** icon from Propel Builder Toolbar.

Note: Currently, Propel 2023.1 Builder only supports single-time undo/redo.

2.2.14.3. Tools Options

Choose **Tools** > **Options** from Propel Builder menu bar. The Options Dialog opens (Figure 2.80).

- Themes

You can choose Light/Dard theme from pull-down menu.

Note: A restart of Propel Builder is required to take effect of the theme change (Figure 2.80).

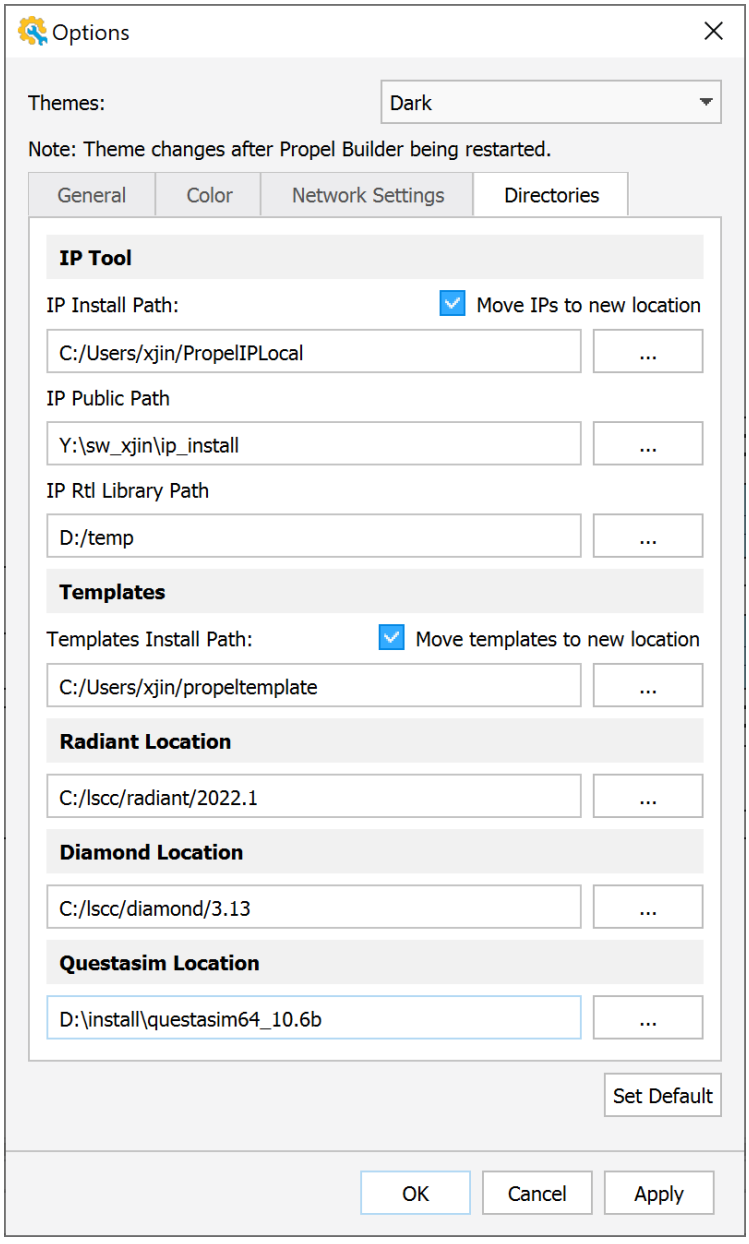


Figure 2.80. Theme Change

- General (Figure 2.81)
  - You can use the right mouse button for zooming actions. See the Methods to Zoom section for more details.
  - The "Automatically connect after instantiating an instance" option is shown in this **Options** dialog. Click **Apply** to make this function take effect.

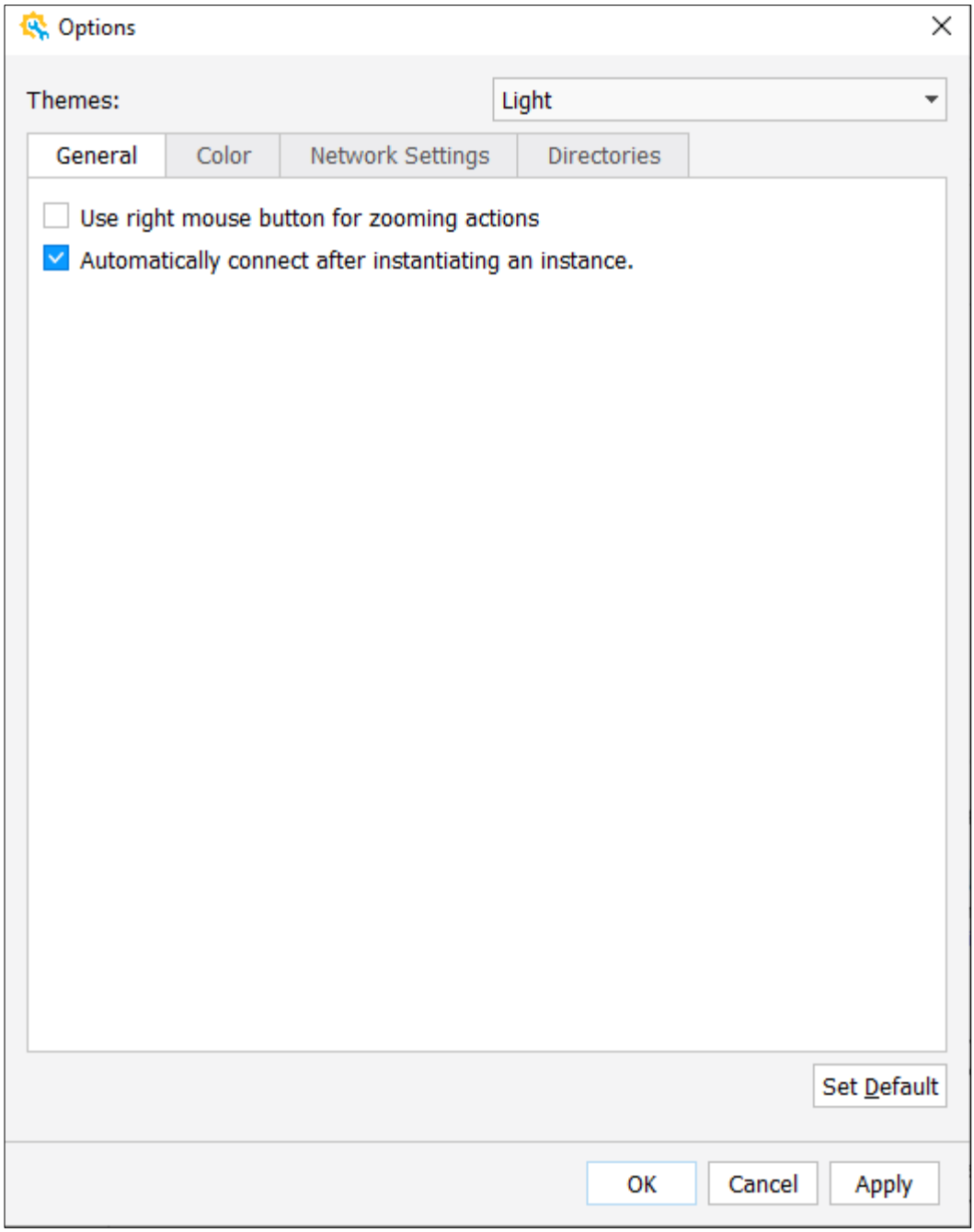


Figure 2.81. General Options

When this auto-connect function is enabled, when an instance is instantiated (see Generating and Instantiating IP/Module section for instantiate an IP), after clicking **OK** in **Define Instance** dialog (Figure 2.82), an addition dialog box pops up with some suggested Clock/Reset connection for the ports on this IP (Figure 2.83). Check/Uncheck to make the desired connections.

For functional use, refer to Connect Pins by Auto Connect for more details.

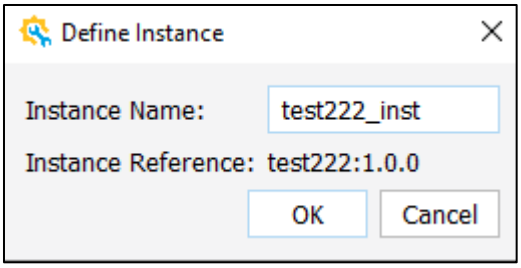


Figure 2.82. Define Instance Dialog

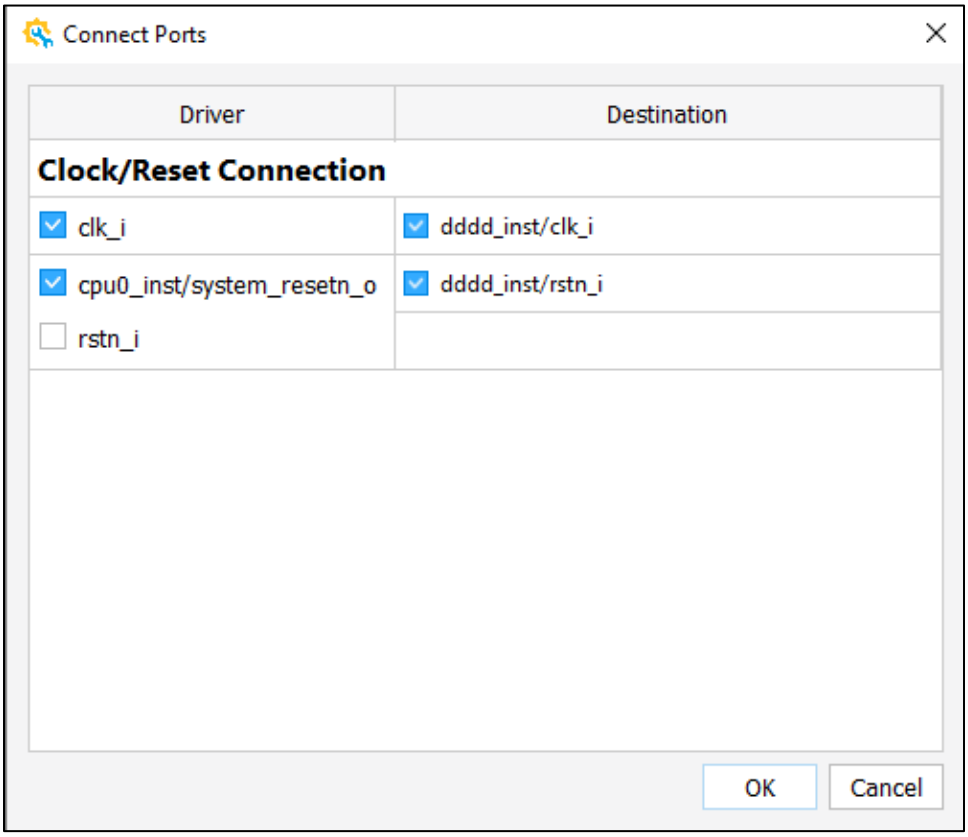


Figure 2.83. Connect Ports Dialog

- Color (Figure 2.84)

To use the highlight function on this tab, refer to the Highlight an Object section for more details. To use the selection function on this tab, refer to the Select One or More Objects section for more details.

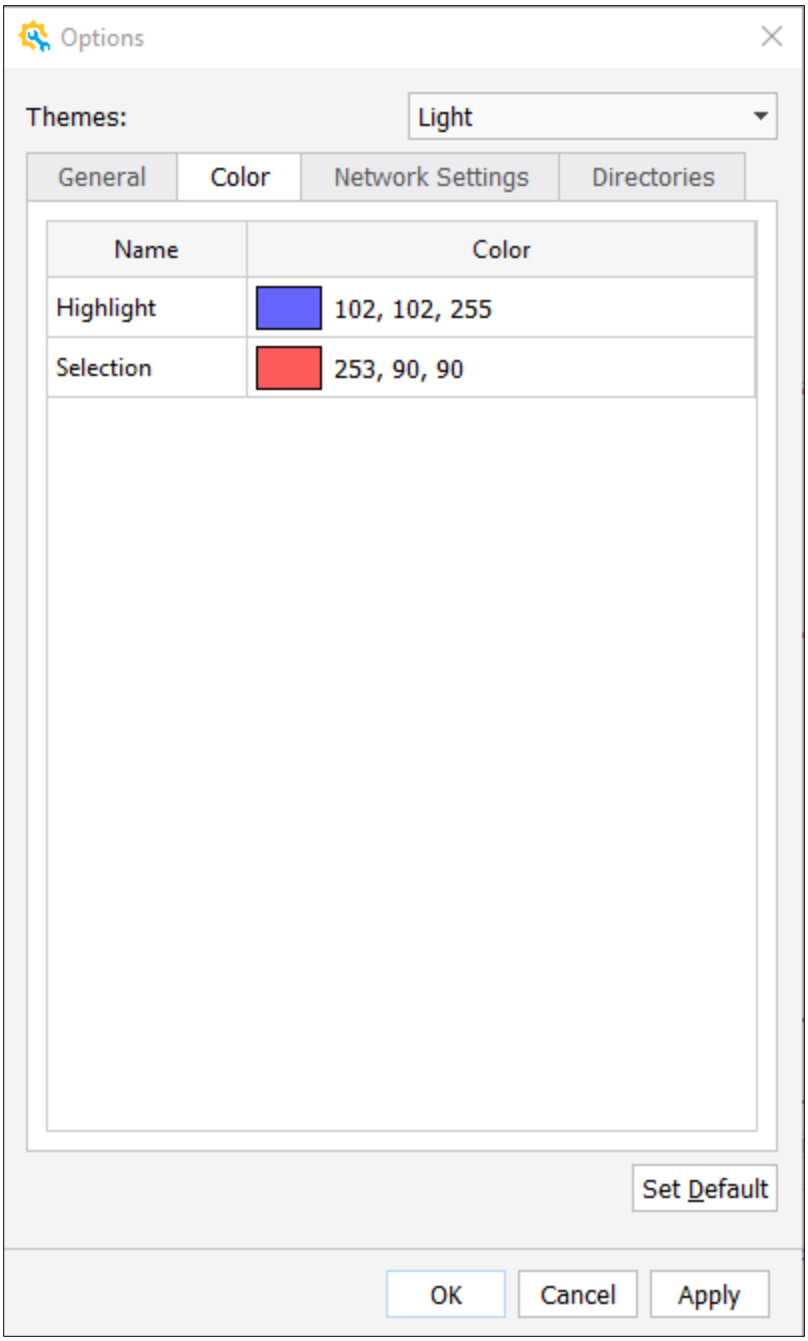


Figure 2.84. Color for Highlight and Selection

- Network Settings (Figure 2.85)

If you need to use a proxy server, click this **Network Settings** tab. Make the appropriate selections on this tab.

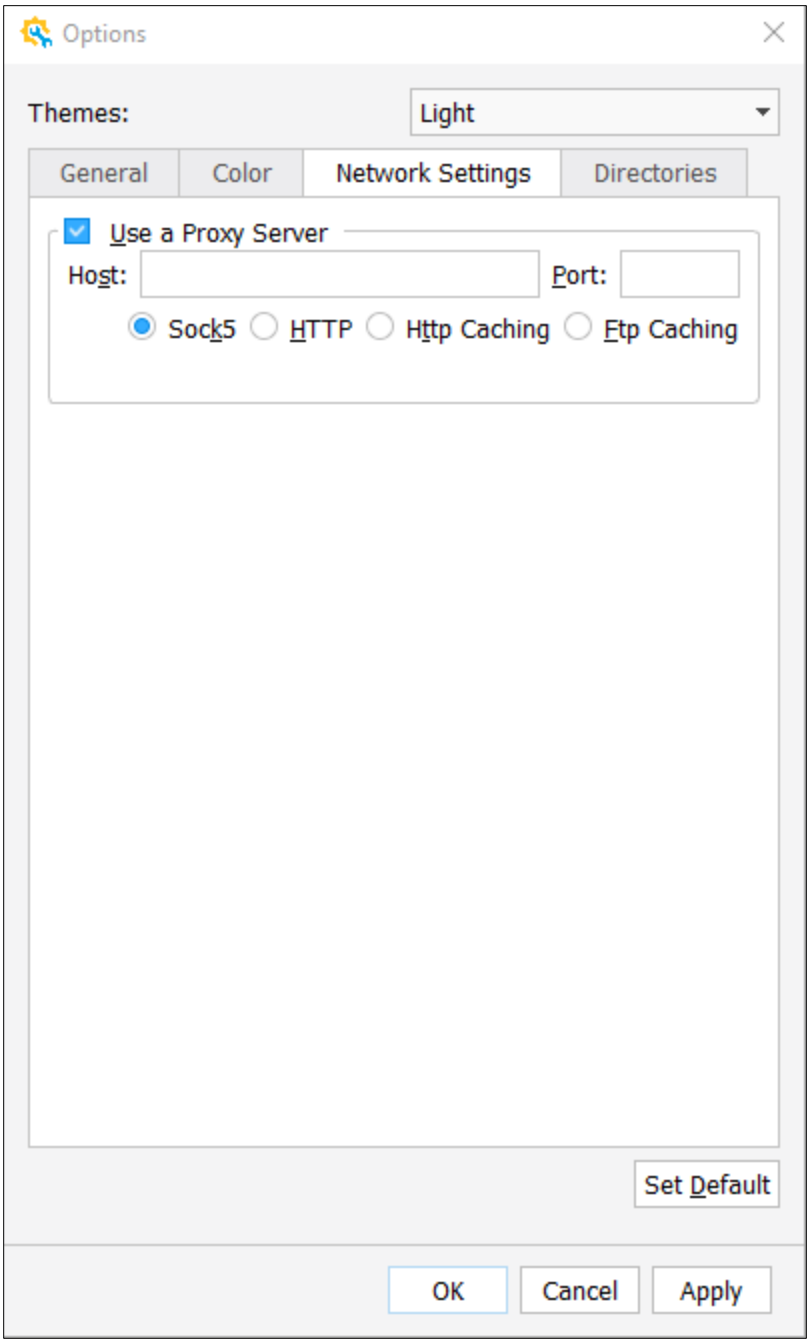


Figure 2.85. Network Settings

- Directories (Figure 2.86)

On this **Directories** tab, there are path setting for related tool.

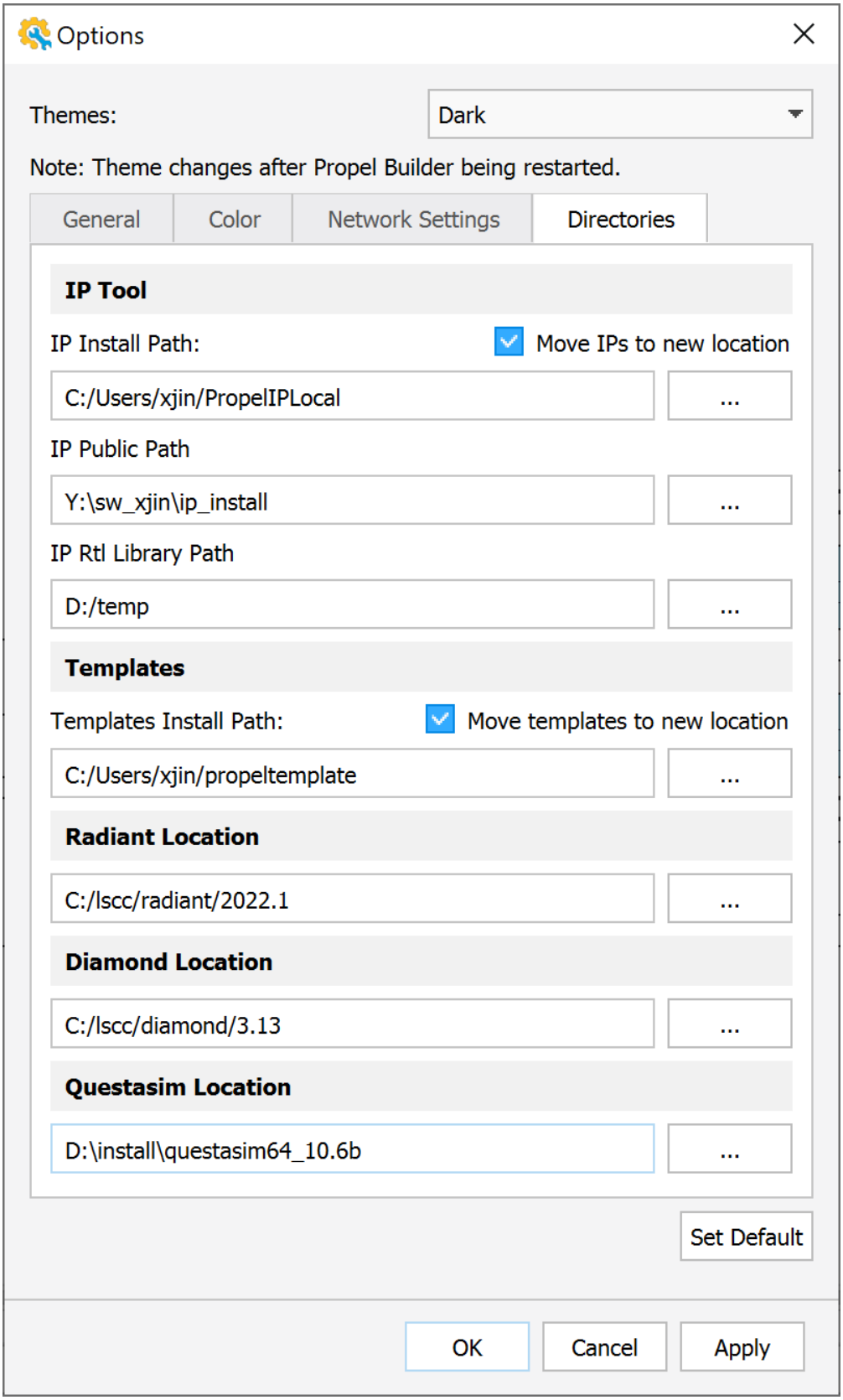


Figure 2.86. Directories

IP Tool (Figure 2.86):

- IP Installed Path (Figure 2.86): Folders of the IPs that are downloaded and installed. Refer to the Generating and Instantiating IP/Module section for more details.
- IP Public Path (Figure 2.87):

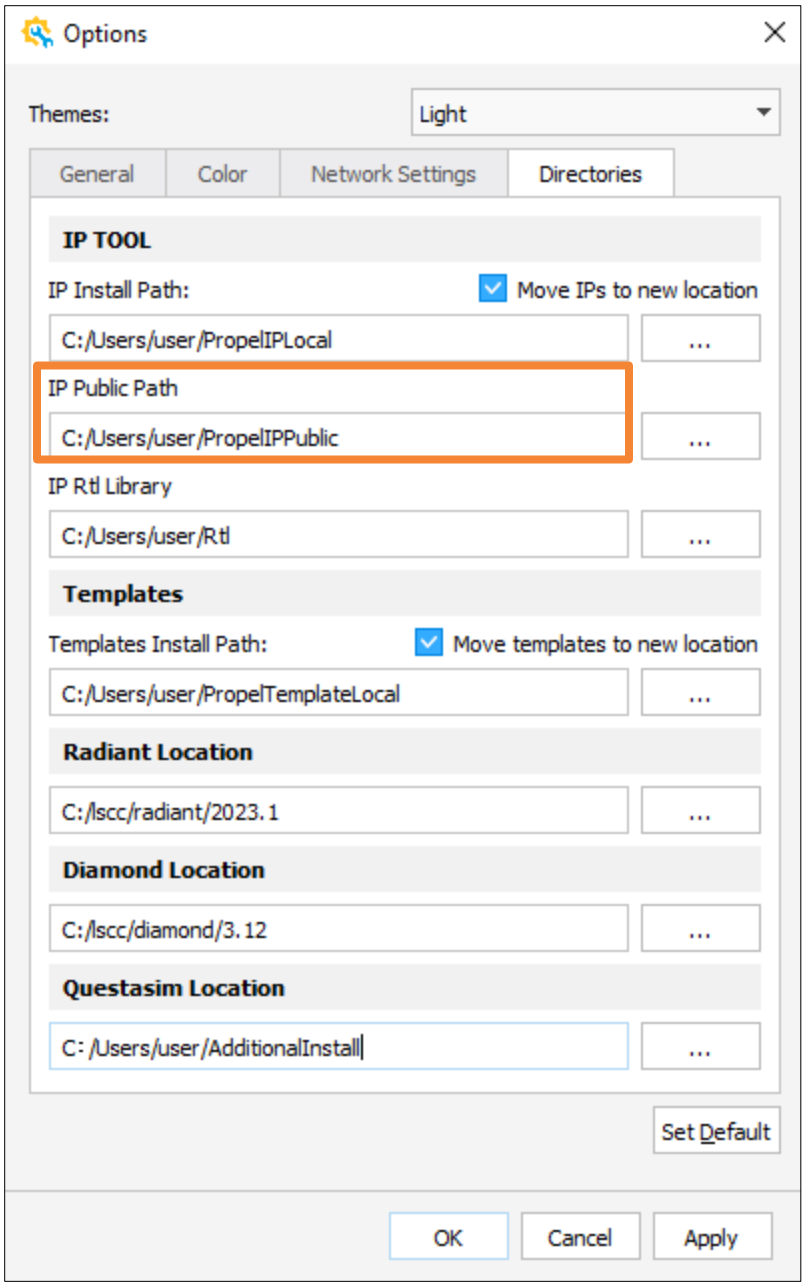


Figure 2.87. IP Public Path

IP Public Path is for administrator user to manage the public IP, and for general user to use the public IP.

Note: Make sure the path is accessible in your current system, without any additional transfer App installed.

- For administrator user to manage the public IP:
  - The administrator user must have the write permission to this path.

- If the administrator user wants to install or remove IP from the IP Public path, just set this as the same path in the **IP Install Path** and then install/remove IP from it.

- For general user to use Public IP:
  - The general user must have the read permission of this path.
  - Set the IP Public Path, and then refresh **IP catalog** to get IP from this path (Figure 2.88).

Note:

IP Public Path does not support network path in Windows Operation System (OS), for example, \\192.168.11.11\shared\ip. Workaround for this is to map the Network Drive with the network path and set the path with a local disk (Figure 2.95).

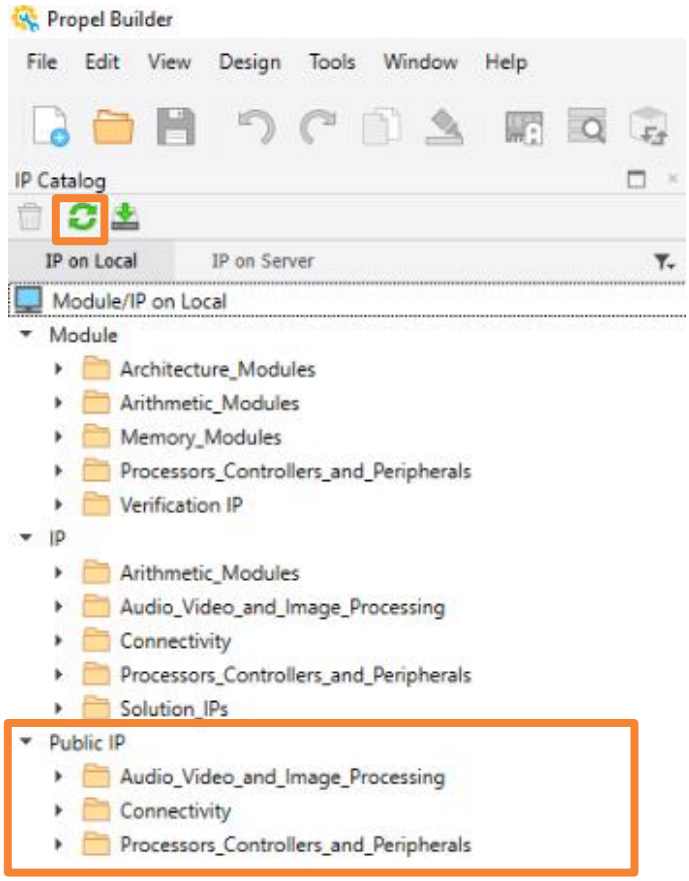


Figure 2.88. IP Catalog

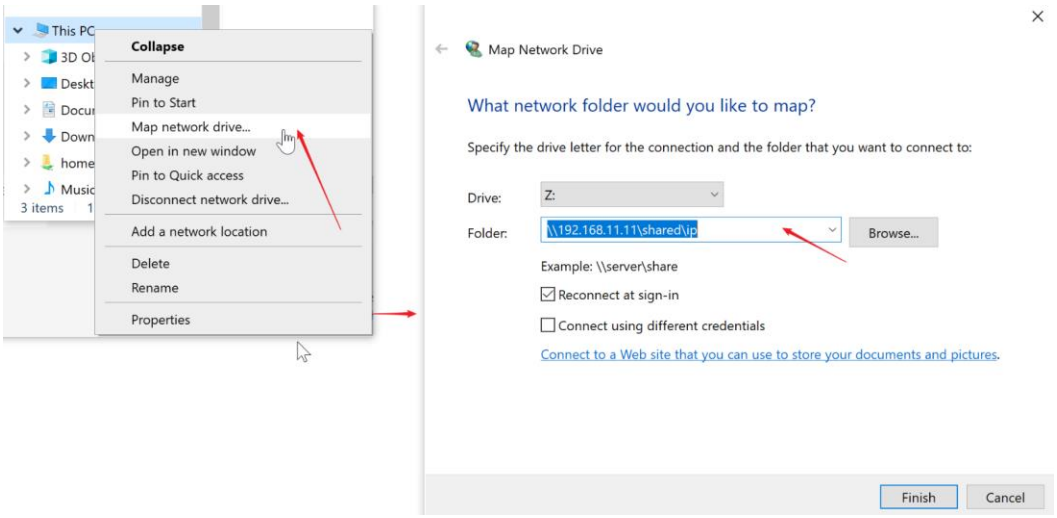


Figure 2.89. Map Network Device

- IP Rtl Library (Figure 2.86): Folders of the IP RTL library. Refer to Lattice Propel 2023.1 IP Package User Guide (FPGA-UG-02187) for more details of the IP RLT Library.

Templates (Figure 2.86):

- Templates Install Path: Folders where the templates installed. See the Define Custom Template section for more details.

Radiant Location (Figure 2.86): Folder where Radiant software installed.

Diamond Location (Figure 2.86): Folder where Diamond software installed.

Questasim Location (Figure 2.86): Folder where Questasim software installed.

2.3. Verification Project Design Flow

2.3.1. Creating a Verification Project

1. Choose **File** > **New Design** from Lattice Propel Builder Menu Bar. The Create System Design wizard opens (Figure 2.90).

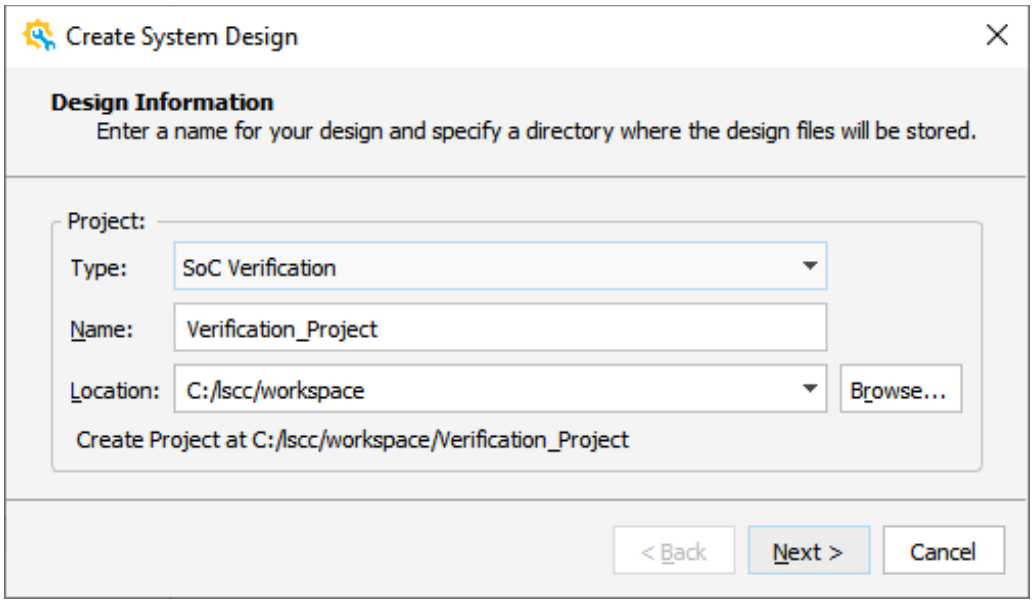


Figure 2.90. Create System Design – Design Information Wizard

2. Choose SoC Verification from the **Type** field.
3. Enter a project name in the **Name** field, such as Verification_Project.
4. (Optional) Use the "**Browse…**" option to change the project location in the **Location** field, if needed.
5. Click **Next**. The Propel Project Configure wizard is shown as Figure 2.91.

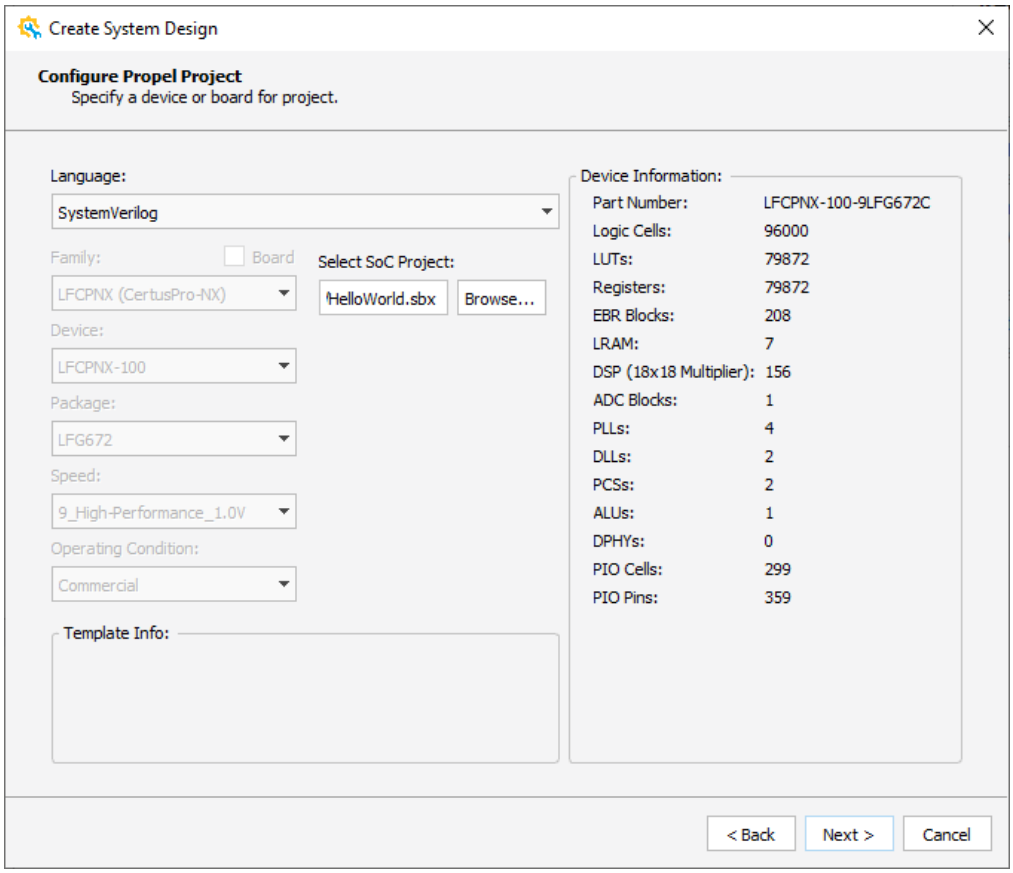


Figure 2.91. Create System Design – Propel Project Configure Wizard

6. (Optional) The default Hardware Description Language (HDL) is displayed in the Language area. Use the drop-down menu to change the default language.
7. Select an existing SBX design by using **Browse…** to choose an SBX file, such as HelloWorld.
8. Click **Next**. The Project Information wizard opens.
9. Click **Finish**. A dut_inst of SoC project can be seen in the Schematic view (Figure 2.92).

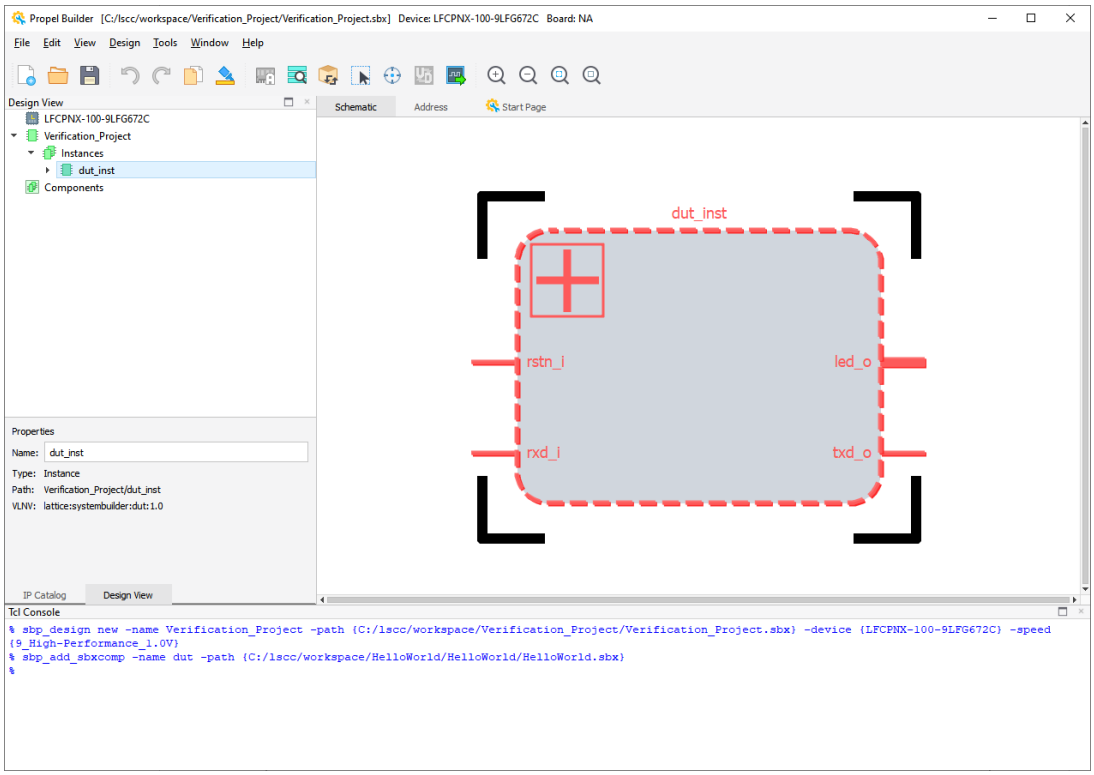


Figure 2.92. Verification Project

Note: The default instance name of an imported Design Under Test (DUT) is dut_inst. By default, its boundary is shown with dotted line, and the filled-in color is gray. If there is any change in the SOC design, the dut_inst DUT block can be updated accordingly by double-clicking this dut_inst DUT block, or by right-clicking this dut_inst DUT block and choosing Reconfig.

10. Click the plus sign  of this dut_inst DUT block. You can see the whole SoC Design (Figure 2.93). Click the negative sign  to close the expanded bus.

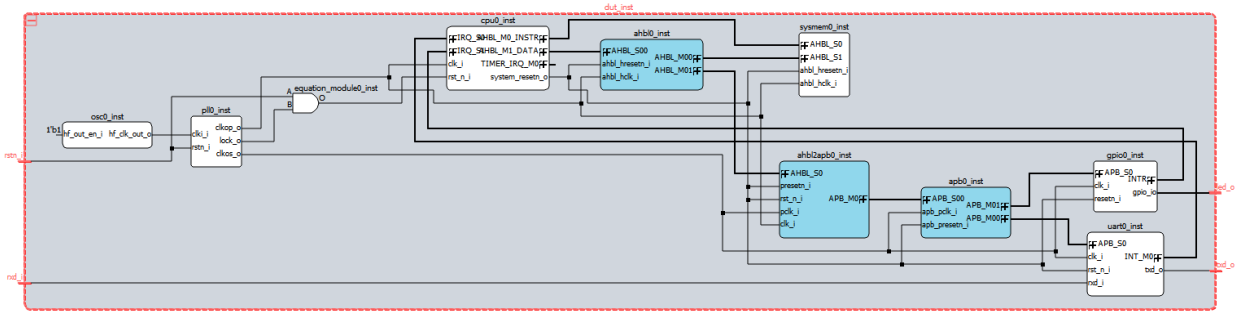


Figure 2.93. Whole SoC Design

2.3.2. Switching SoC Project to Verification Project

Propel Builder supports switching SoC project to verification project manually after creating a SoC project.

1. Create a SoC Project, such as HelloWorld project. Refer to the Creating Template SoC Project section for more details.

Note: Corresponding Verification project is created automatically when creating a new empty SOC project. Hello World Templates can have pre-developed Verification projects.

2. Click **Switch Verification and Soc Design** icon  from Propel Builder GUI Toolbar. Propel Builder dialog box opens (Figure 2.94) if a project is not saved.

   When you create a SoC project, a corresponding verification project is also generated using this icon. This is how the Design to Verification flow works, it switches between these projects which are linked.

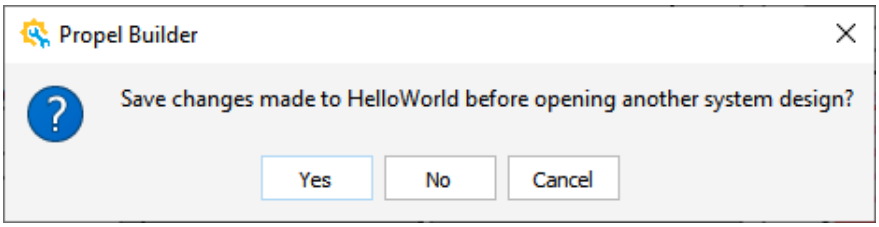


Figure 2.94. Propel Builder Dialog Box

3. Click **Yes** in the Propel Builder dialog box. The Propel Builder GUI switches to verify the project (Figure 2.95).

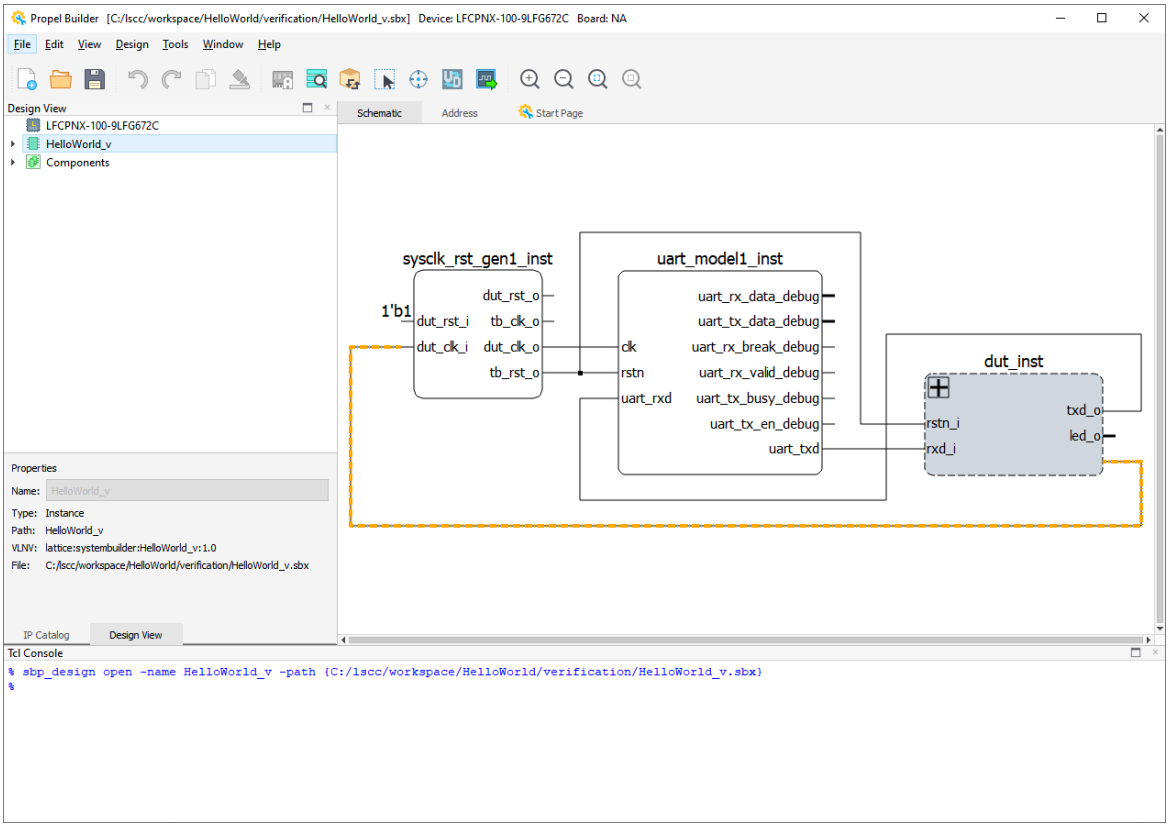


Figure 2.95. Switching to Project Verification

Note: If the SoC project needs to be re-configured, click **Switch Verification and Soc Design**  again to switch the verification project to the SoC project.

2.3.3. Opening a Verification Project

Refer to the Opening an Existing SoC Project section for procedure in detail.

2.3.4. Adding Modules, IP and VIPs

Refer to the Generating and Instantiating IP/Module section for procedure in detail.

2.3.5. Working with the Schematic View

Refer to the previous Working with the Schematic View section for procedure in detail.

2.3.6. Connecting Modules

Refer to the previous Connecting Modules section for procedure in detail.

2.3.7. Monitoring DUT

This feature is available to a SOC Verification project only. In Propel Builder, the pin inside DUT can be connected to the input pin of a VIP. The connected pins can be found at both ends of the orange line, as shown in Figure 2.96. Only pin and pin bus are supported in the current release of Propel Builder.

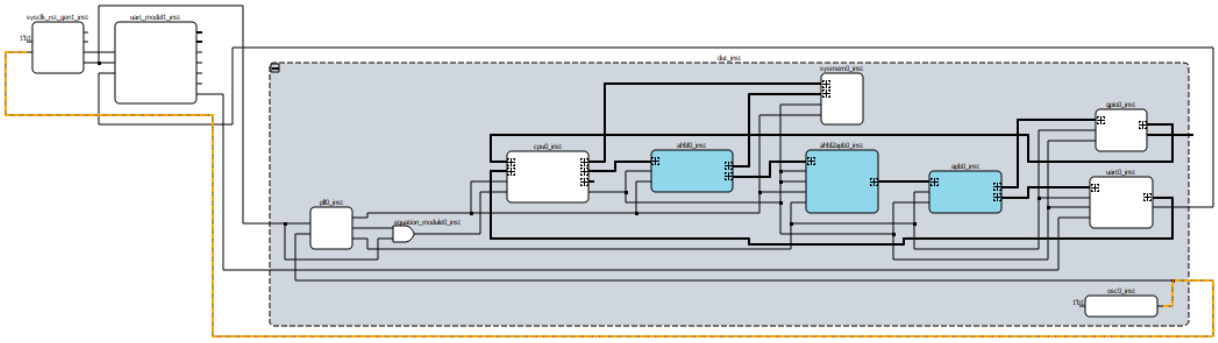


Figure 2.96. Monitoring DUT

The testbench generated for this Verification project is shown in Figure 2.97. This testbench file can be used for simulation.

```
75    logic sysclk_rst_gen1_inst_dut_clk_i_net;
76
77    /*-----------------------Connection Block-----------------------*/
78    /* This section cover all the connections that related to internal*/
79    /* signals of DUT, or interface(e.g, AHBL Master BFM).           */
80    /*--------------------------------------------------------------*/
81
82    assign sysclk_rst_gen1_inst_dut_clk_i_net = dut_inst.osc0_inst.hf_clk_out_o;
83
84    /*---------------------------BFM Block--------------------------*/
85    /* This section is reserved for user to create stimulus using BFM */
86    /* APIs.                                                        */
87    /*--------------------------------------------------------------*/
88
89    ....
90
91    sysclk_rst_gen1
92    sysclk_rst_gen1_inst
93    (
94      .dut_clk_i(sysclk_rst_gen1_inst_dut_clk_i_net),
95      .dut_clk_o(sysclk_rst_gen1_inst_dut_clk_o_net),
96      .tb_rst_o(sysclk_rst_gen1_inst_tb_rst_o_net)
97    );
```

Figure 2.97. Testbench of the Verification Project

2.3.8. Generating Simulation Environment

1. Click the **Generate** icon  from the Propel Builder Toolbar to generate a testbench file including the scripts for the chosen simulator, file list for HDLs, and some other files. The testbench file structure is shown in Figure 2.98.

```
[sim]           -- generated simulation environment
        [hdl_header]
soc_regs.v      -- register definitions of all the components in DUT/SOC
sys_platform.v  -- base address, user settings of all the components in DUT/SOC
        [misc]
*.*             -- all the mem, hex, txt files will be copied here
                  flist.f        -- file list for HDLs
                  msim.do        -- do script for simulator, it will be qsim.do
for Questasim.
this file compiles project and invokes simulator with some default settings
using the generated testbench
                  wave.do     -- do script for adding signals in waveform
window
                  <project_name>.sv    -- top testbench, SystemVerilog based
```

Figure 2.98. Testbench File Structure

Unlike those HDLs generated in the SOC design, the testbench generation in the Verification project is just a start point for you to work with. Make sure that you generate a new testbench if there is an existing simulation environment. A dialog box pops up prompting you to make a choice of Yes or No (Figure 2.99).

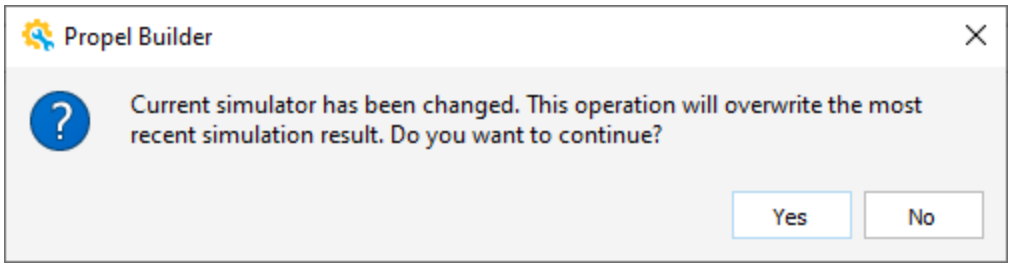


Figure 2.99. Propel Builder – A Reminding Dialog Box

2.3.9. Launching Simulation

1. Click **Launch Simulation** icon  from the Propel Builder Toolbar to launch simulation.
2. The default simulation tool, ModelSim, OEM version, opens (Figure 2.100).

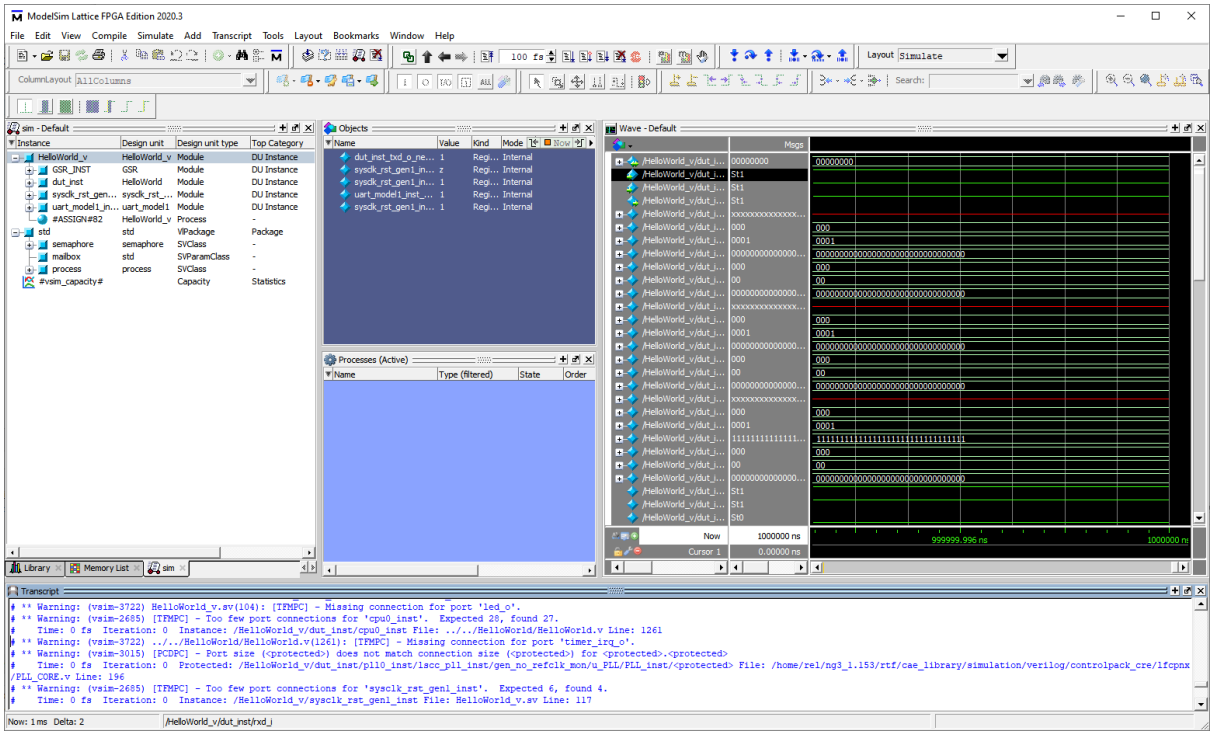


Figure 2.100. ModelSim Simulation GUI

3. Or, before launching simulation, you can change Simulator to Questasim by clicking **Design** > **Options** from the Builder Menu bar. The Builder Options Wizard opens (Figure 2.101). Click **Directories** to set a desired Questasim Location.

   Note: Set the **Radiant Location** or **Diamond Location** first, and then the **Questasim Location**. Click **Launch Simulation** icon  from the Propel Builder Toolbar to launch simulation.

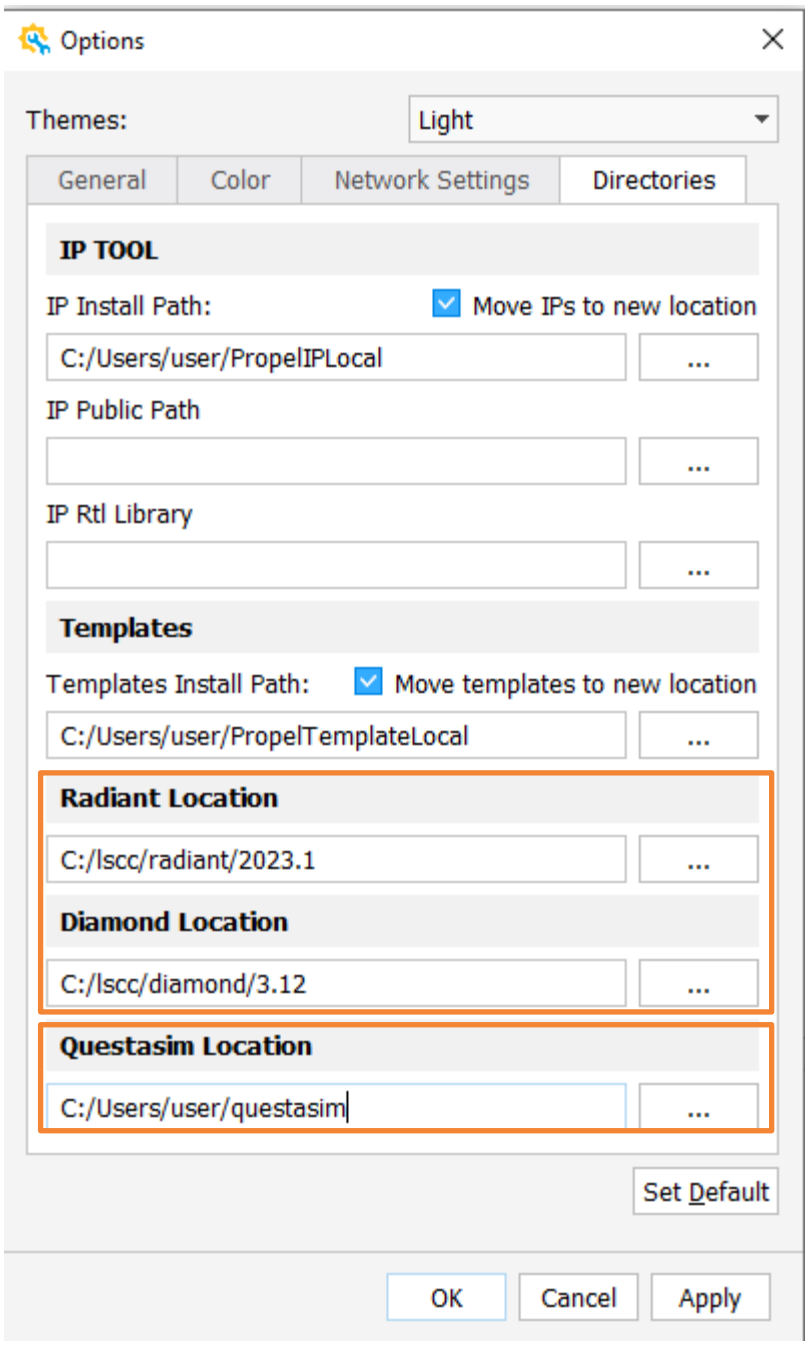


Figure 2.101. Builder Options Wizard – Radiant/Diamond Location and Questasim Location

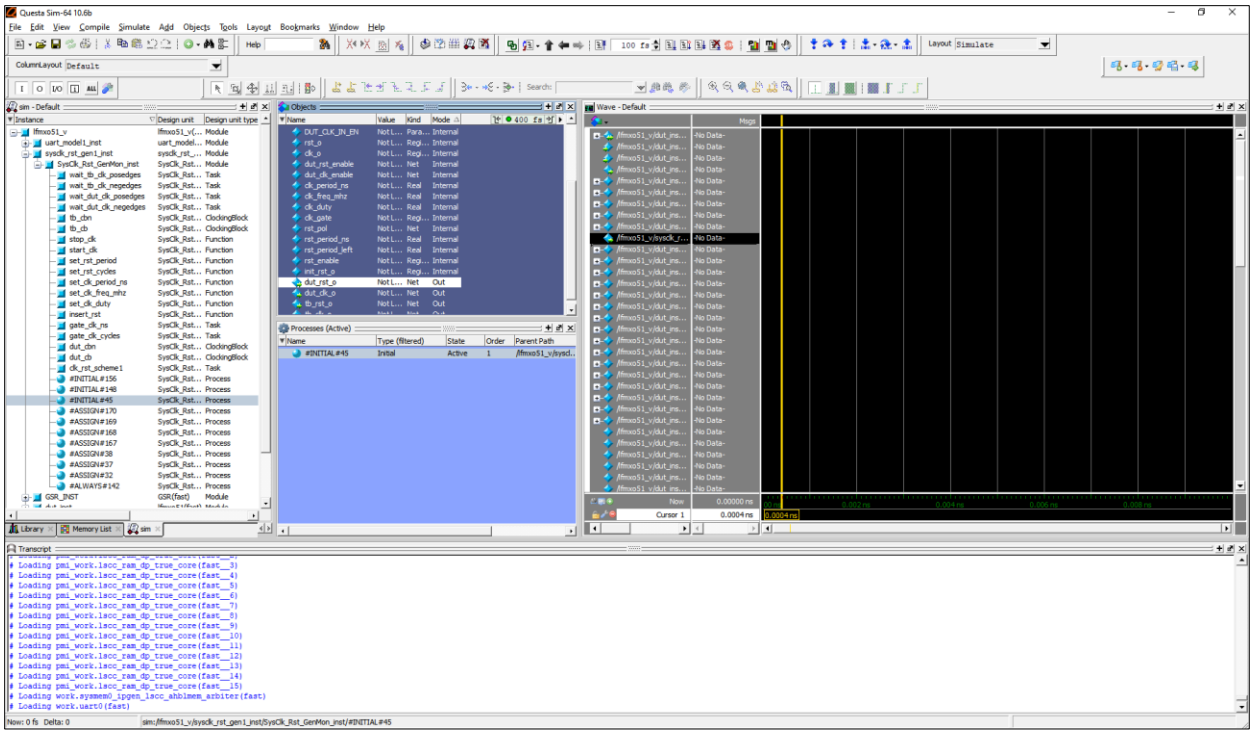


Figure 2.102. Questa Simulation GUI

Once the Questa Sim GUI is launched (Figure 2.102), the Questa sim is linked with Propel Builder. Propel Builder has an auto simulation library compilation for Questa.

Simulation project path is <*proect_name*>/verification/sim (Figure 2.111). During launch, the generating script under this path can detect whether or not there are existing simulation libraries (an ovi_<*device_family_name*> folder under the path).

If yes, simulation uses the existing library; otherwise, the device library is created and compiled once into the ovi folder for further use. And, this compiling process may take some time.

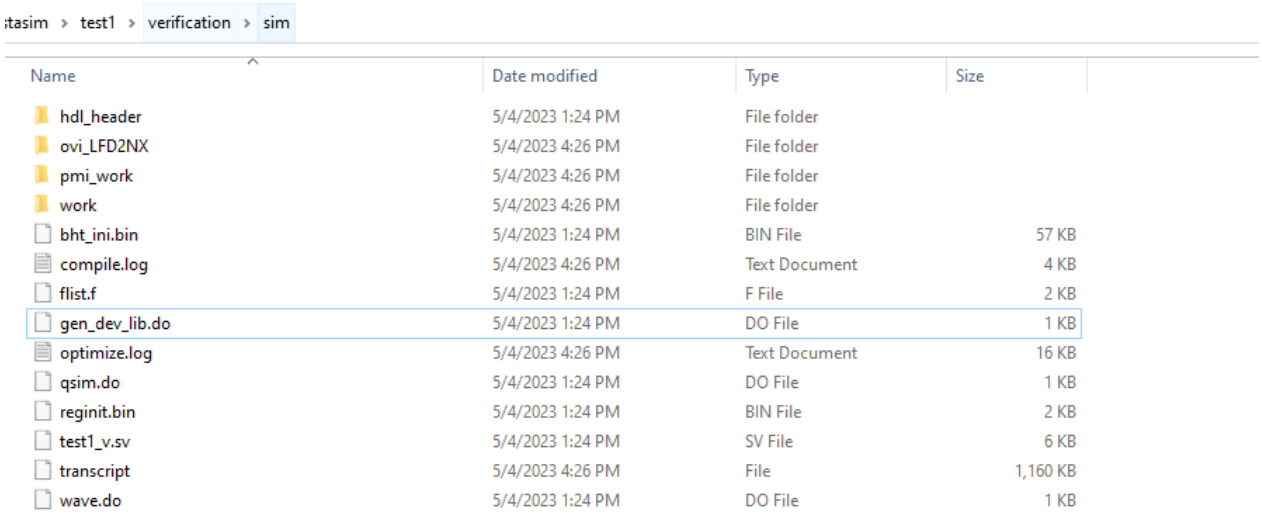


Figure 2.103. Testbench File Structure in Questasim

2.4. Advanced Usage

2.4.1. Supported Interface

Lattice Propel Builder and IP Packager share the same usage of interfaces. These interfaces are listed below.

- Interrupt
- AMBA3 AHB Lite
- AMBA3 APB
- AMBA4 AXI4
- AMBA4 AXI4 Lite
- AMBA4 AXI4 Stream
- Localbus

Refer to Lattice IP Packager 2023.1 User Guide (FPGA-UG-02187) for detailed usage of these interfaces in the IP Packager.

2.4.2. Supported Language

Propel 2023.1 Builder supports the following languages:

- Verilog HDL
- SystemVerilog
- VHDL
- Mixed Verilog/VHDL

Currently, verification projects only support Verilog HDL.

2.4.3. Supported Hierarchical IP

Hierarchical IP instantiates a hierarchical component that contains a reference to a design along with a description of the top-level interface of the component.

Click **Expand IP** sign on the left corner of the hierarchical IP (Figure 2.104) to show the detailed design scope of this IP (Figure 2.105).

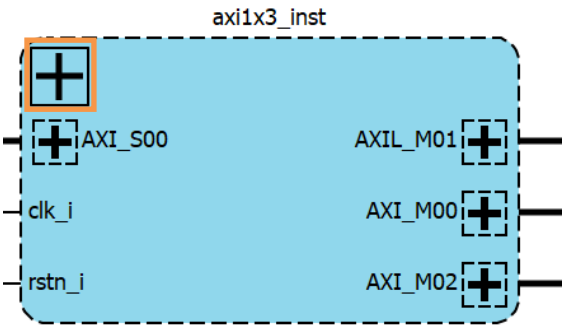


Figure 2.104. Hierarchical IP

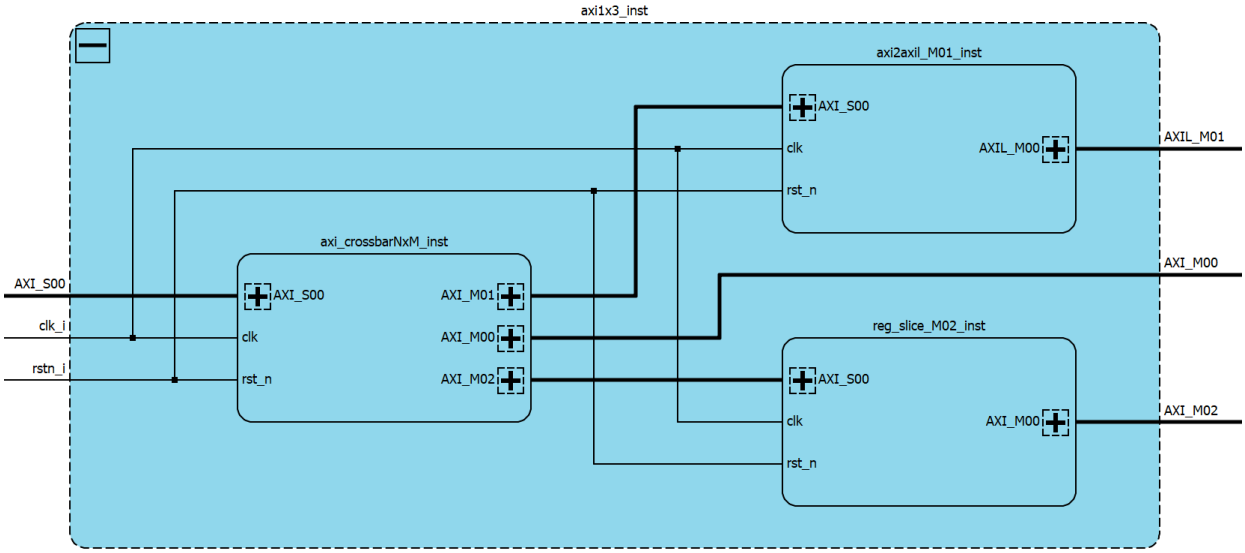


Figure 2.105. Hierarchical IP Scope in Detail

2.4.4. Export Interface

AHB Lite or APB interface ports can be connected to external IP outside of Propel Builder. You must use AHB-Lite or APB feedthrough modules to hold the address space (when feedthrough module is configured as major) or memory map (when feedthrough module is configured as slave) information. The interface to be exported should be connected first to feedthrough, then the other end of the feedthrough can be exported. An example is shown in Figure 2.106.

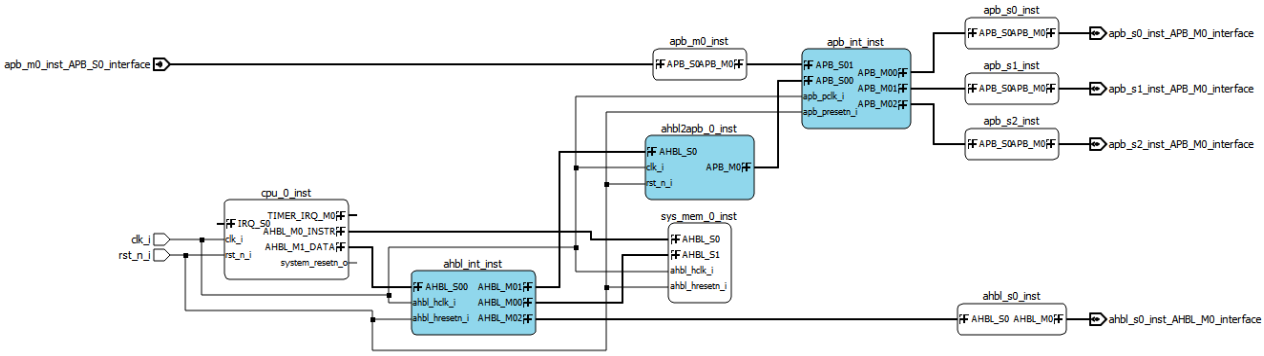


Figure 2.106. Export Interface Example

In the example above, to export interface ahbl_int_inst.AHB_M02:

1. Instantiate an AHB-Lite feedthrough module from IP catalog, and configure it as slave. The instance name here is ahbl_s0_inst.
2. Set the address width and data width for the memory map.
3. Connect ahbl_int_inst.AHB_M02 to ahbl_s0_inst.AHBL_S0.
4. Export ahbl_s0_inst.AHBL_M0 for external IP.

You can see the memory map for ahbl_s0_inst is shown in the Address Editor (Figure 2.107).

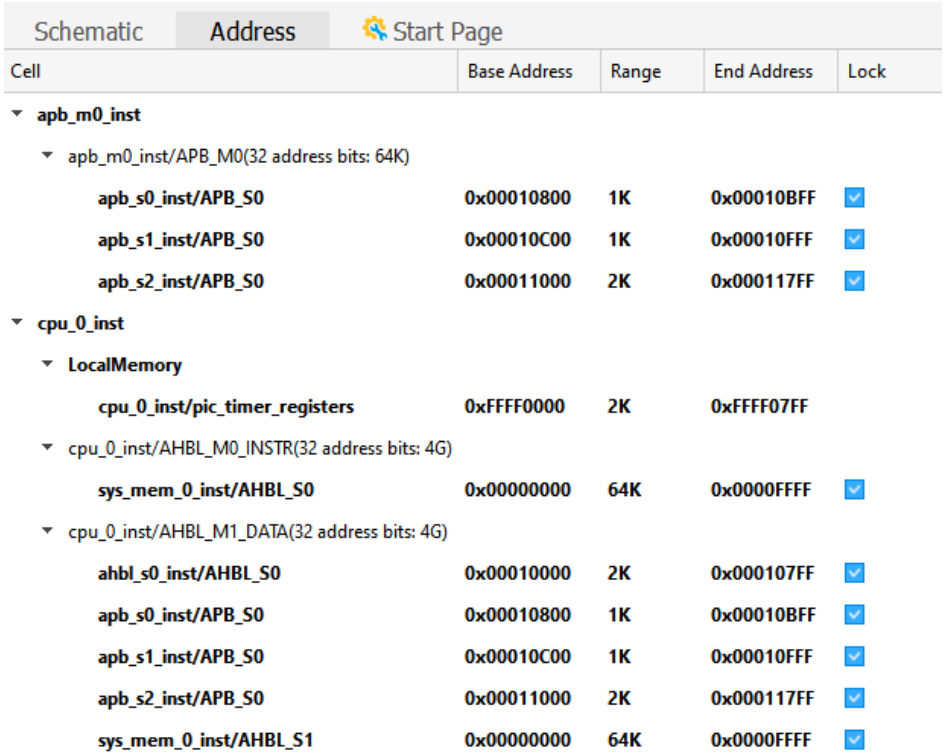


Figure 2.107. Memory Map for ahbl_s0_inst in Address Editor

2.4.5. Define Custom Template

Saving a design as a template is an enhancement feature of Propel Builder, which allows you to create your own new project template based on an existing design. You can deploy this template into Propel Builder later or share it with others by saving template into a .ptmp file. See the following on how to generate a .ptmp file.

Note: This function takes effect from Propel Builder 2023.1, so the design created in earlier Propel Builder versions cannot define its custom template.

You can access **Template Manager** in two ways:

- Access during project creation. See the Creating Template SoC Project for more information.
- Choose **Tools** > **Template Manager** from Propel Builder Menu Bar.

2.4.5.1. Export the Current Template Configuration

Choose **Tools** > **Create Template** from Propel Builder Menu Bar. Template Configuration (Figure 2.108) opens.

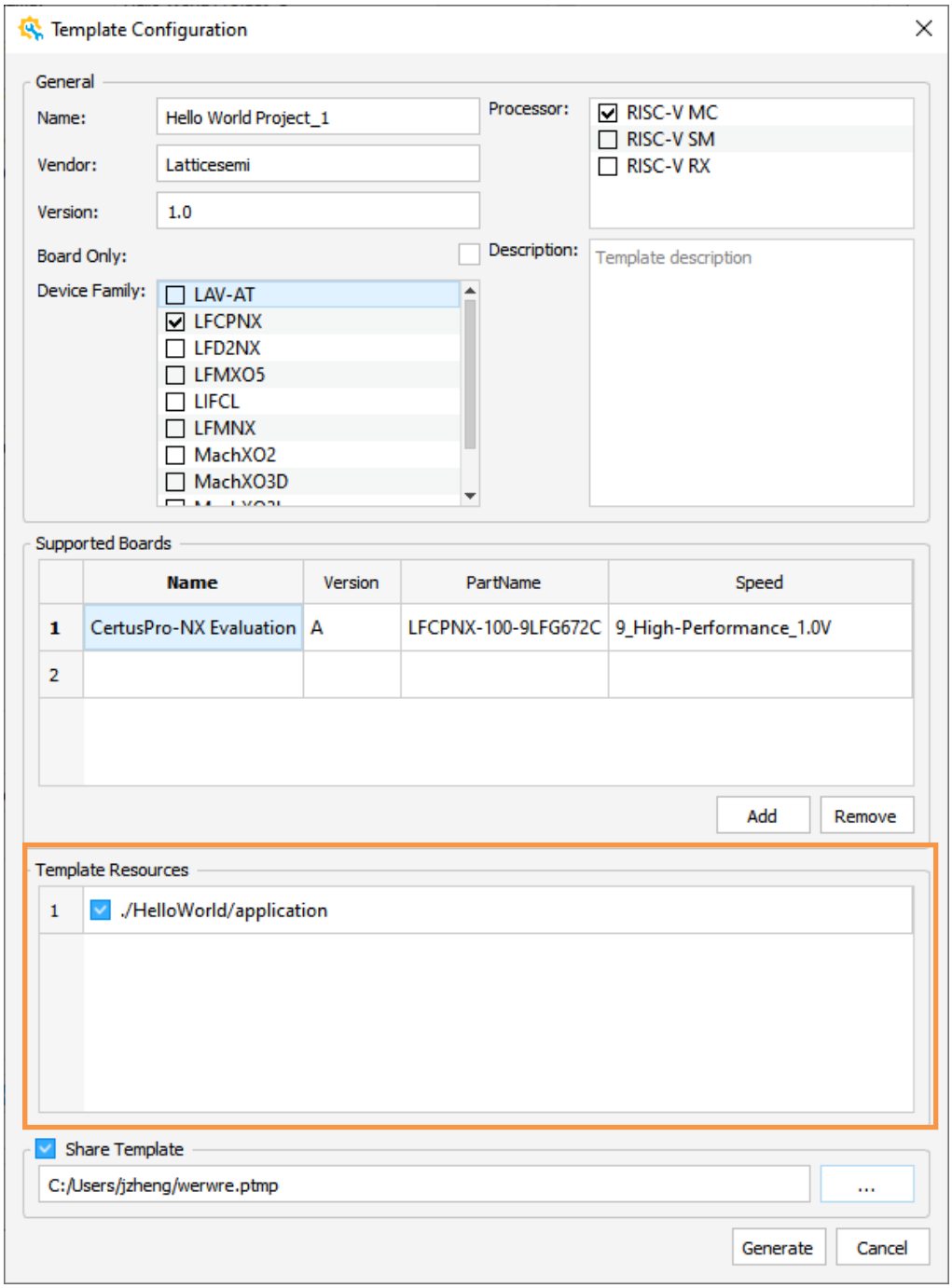


Figure 2.108. Export Template Configuration Page

1. Fill in general information such as Template name, Description, Vendor.
2. Select Processor, Device Family, and Supported Boards.
   - The device used in the current design is listed by default. You must make sure the design can work on all the devices in the list.
   - Click on an IP instance in **Schematic view**. The **Design View** shows the corresponding information of this IP.

     Check VLNV for this IP name (Figure 2.109). Go to **IP Catalog**, and click on before this IP to show the **IP Information**. In the **Device Supported** area, all the devices this IP supports are listed (Figure 2.110). Make sure

all the IP instances in the design have the device support which are listed in the Board Family list in Template Configuration (Figure 2.108).

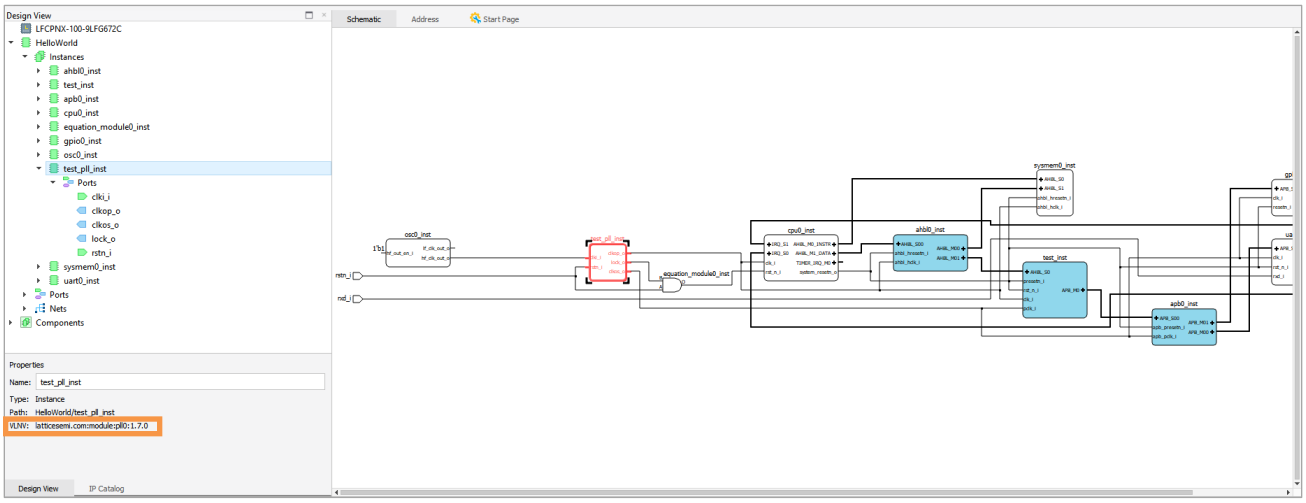


Figure 2.109. Check IP Name in Design View

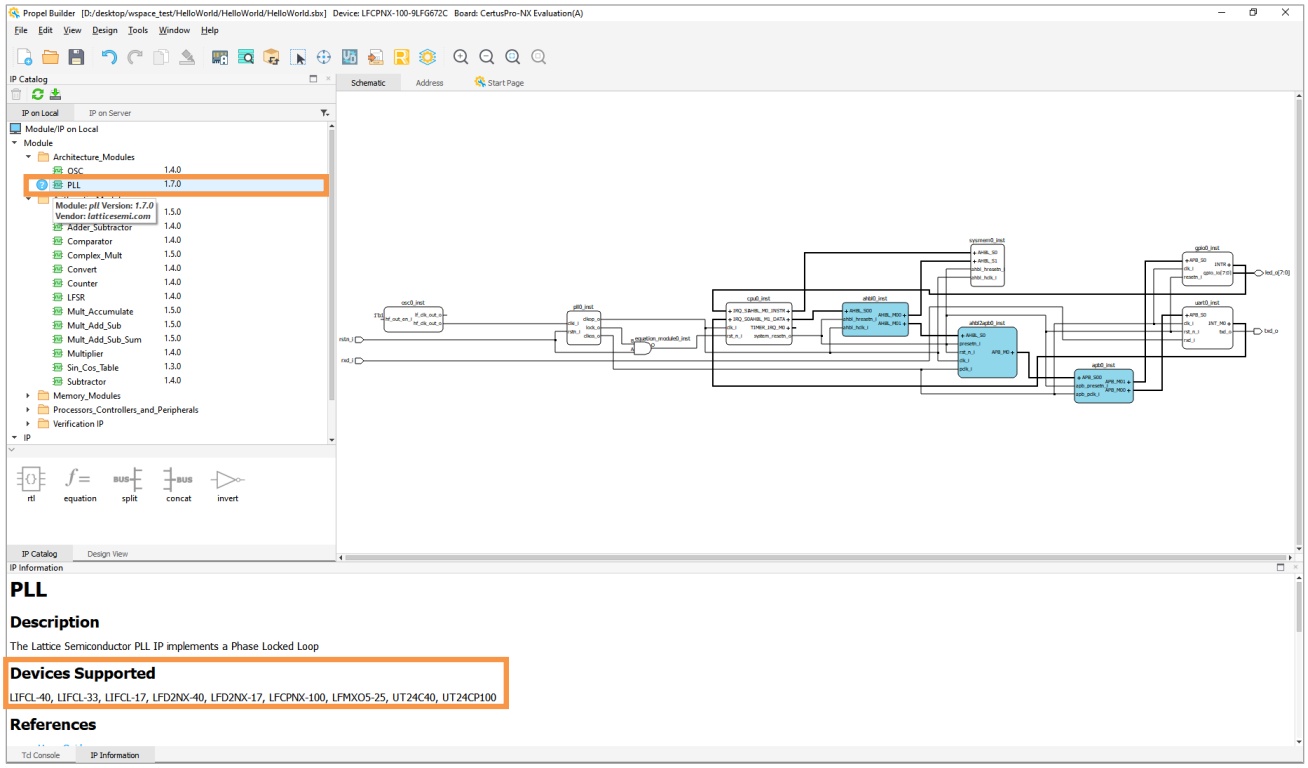


Figure 2.110. Check Device Supported in IP Information

- You can add the board information manually in **Supported Boards** (refer to Figure 2.4) for board information). You must ensure the correctness of the board information and the design can work on this board.

3. Template Resources:

   Optional resources include the newly created project such as C application project, user RTLs.

- Output:
  - By default, after you click the **Generate** button, the new template can be installed to your custom template folder, which is a folder named **PropelTemplateLocal** under $HOME (Figure 2.111).
  - Option under **Template Resources**, such as the *application* folder, contains functional-ready embedded application source codes for C project. Check the box before it (Figure 2.108), then the application under current project is also exported to destination (Figure 2.111). Other files under this option works the same way.

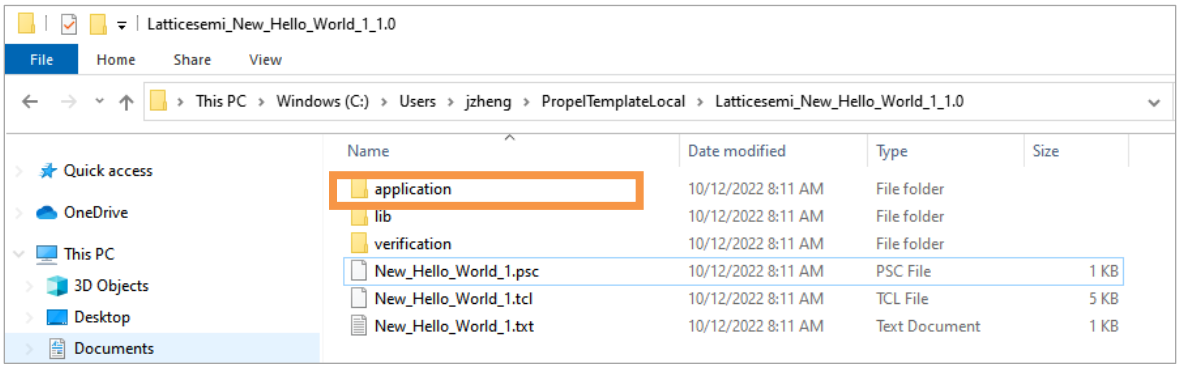


Figure 2.111. Export Template to PropelTemplateLocal Folder

  - If you want to share this template, you can click **Share Template** and output this template to a .ptmp file which can be later installed to Propel Builder by other people.
- Template manager page for custom templates.

  You can import/delete/export a selected template in **Template Manager** when creating a new system design (Figure 2.112 and Figure 2.113).

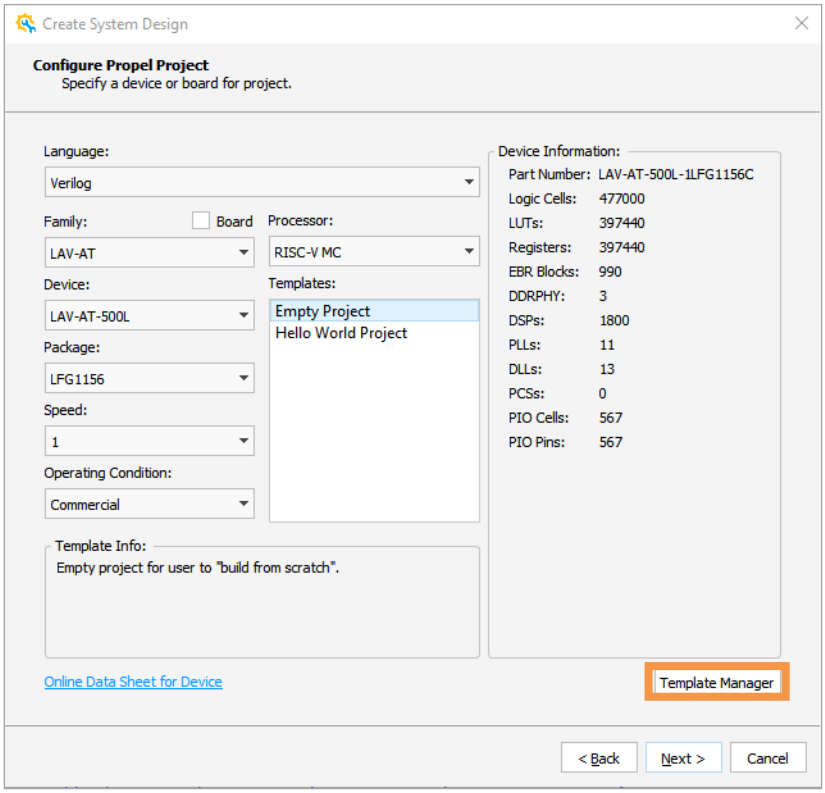


Figure 2.112. Template Manager Entry

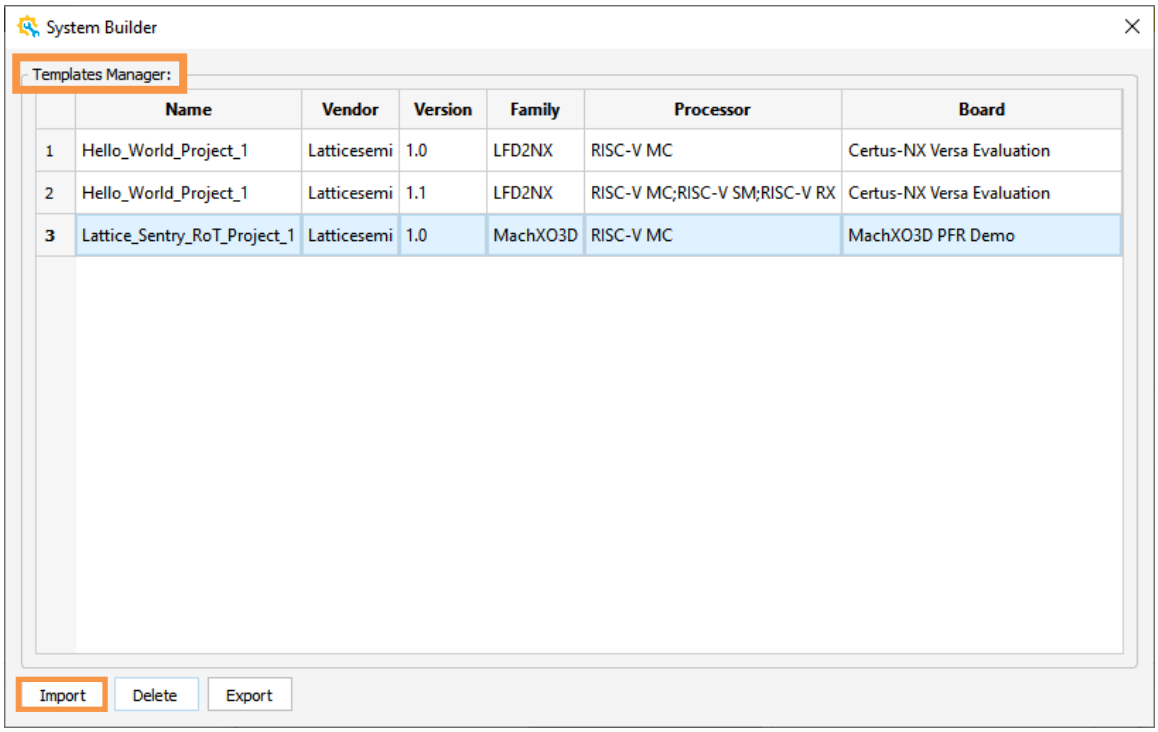


Figure 2.113. Templates Manager Page

2.4.5.2. Import a Template

1. click **Import** (Figure 2.113), and then choose the desired .ptmp file (Figure 2.114).

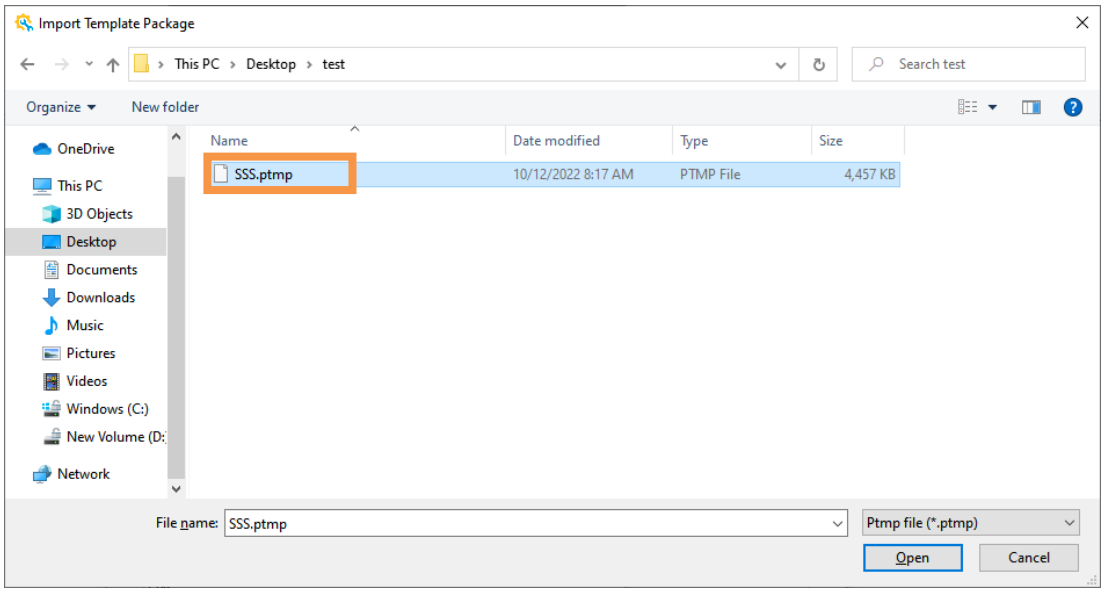


Figure 2.114. Import a .ptmp File

2. Custom template installation path.
   Custom templates are installed in $HOME/PropelTemplateLocal directory by default (Figure 2.115).

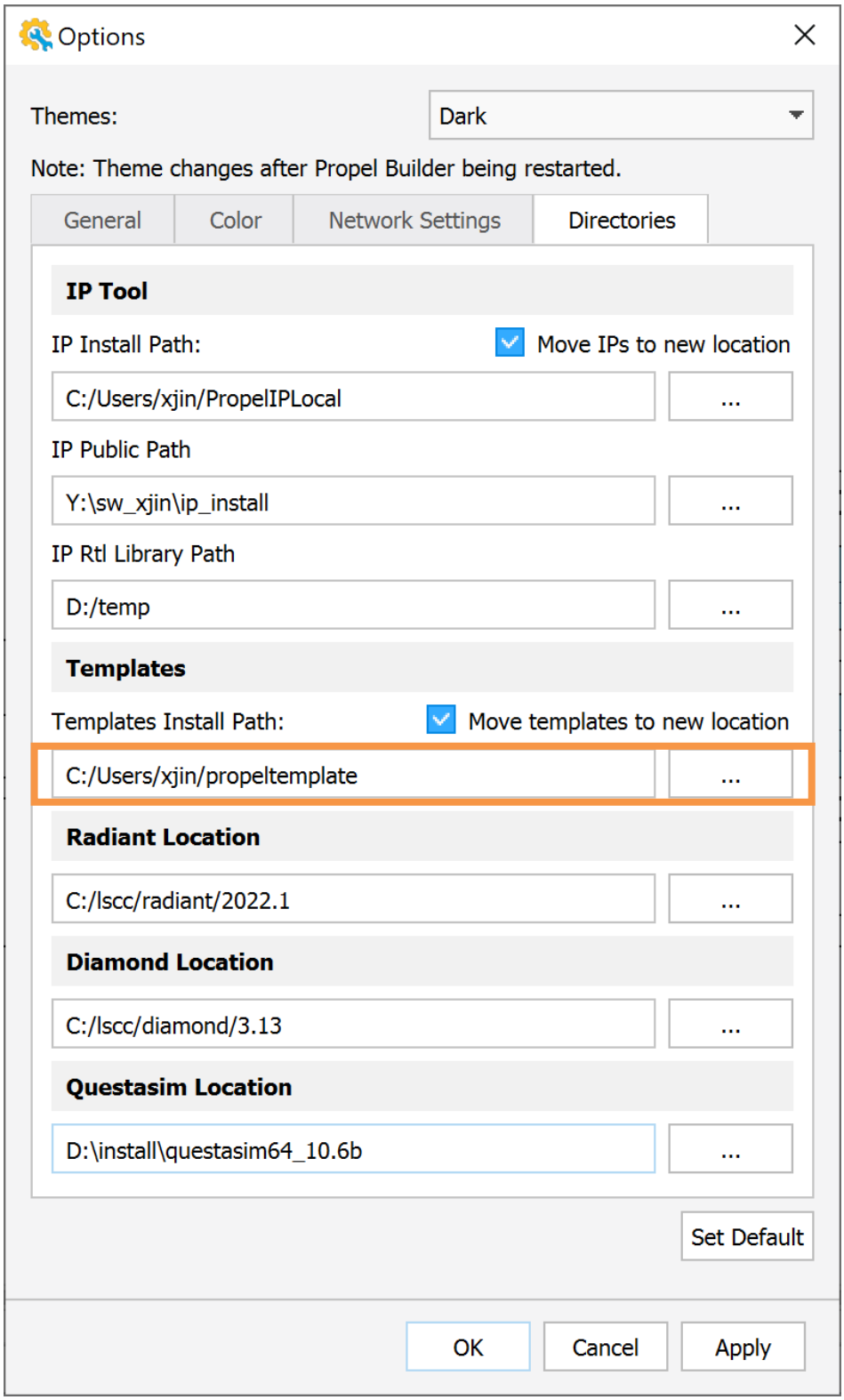


Figure 2.115. Template Manager Page

You can move it to another location by checking the *Move templates to new location* option in Directories tab of the **Options** dialog (Figure 2.116).

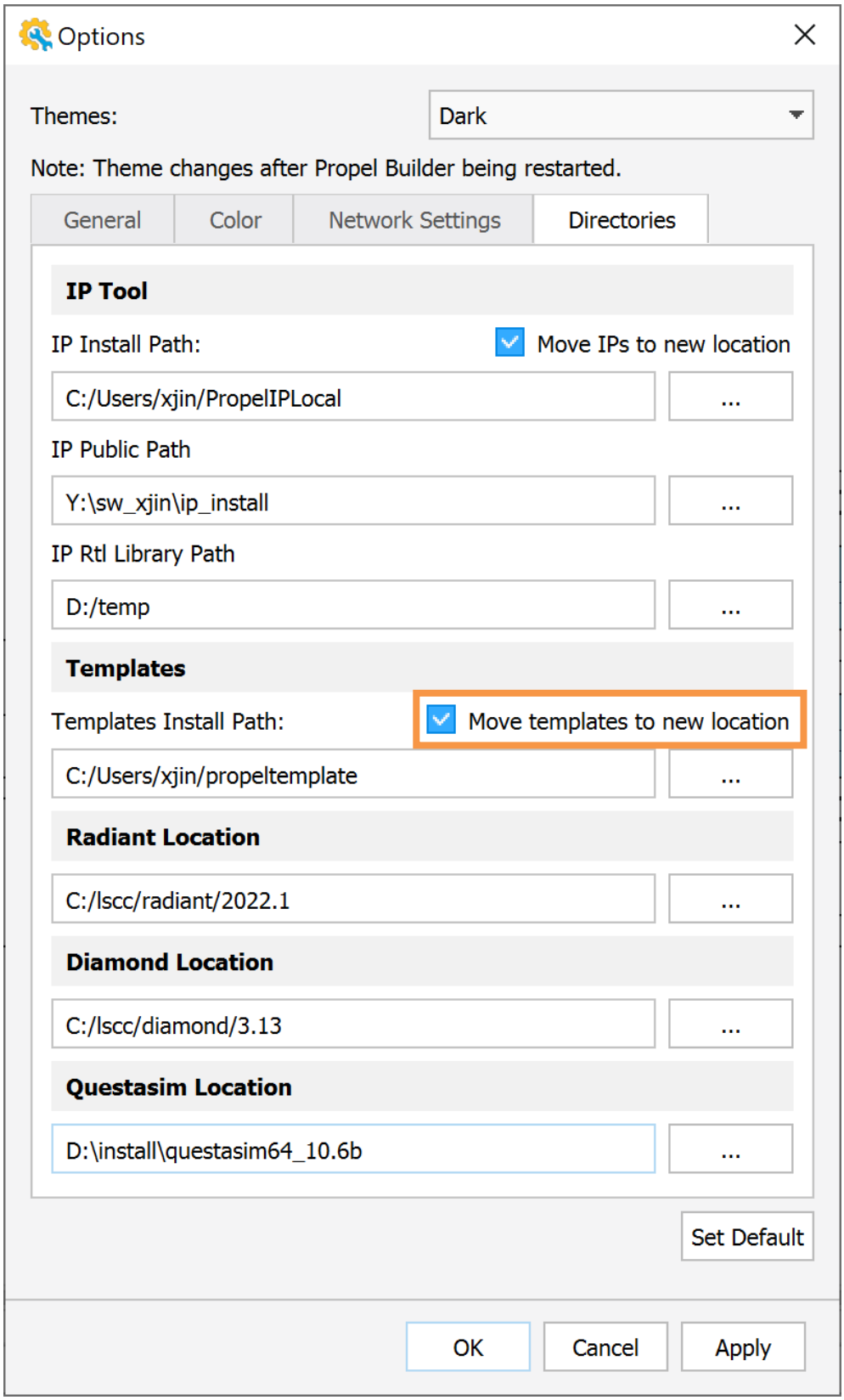


Figure 2.116. Template Manager Page – Change to New Location

2.4.5.3. Delete a Template

Click on the custom template you want to delete from the **Template Manager** Page (Figure 2.117), and then click **Delete**. The corresponding template folder under **Template Install Path** will be deleted.

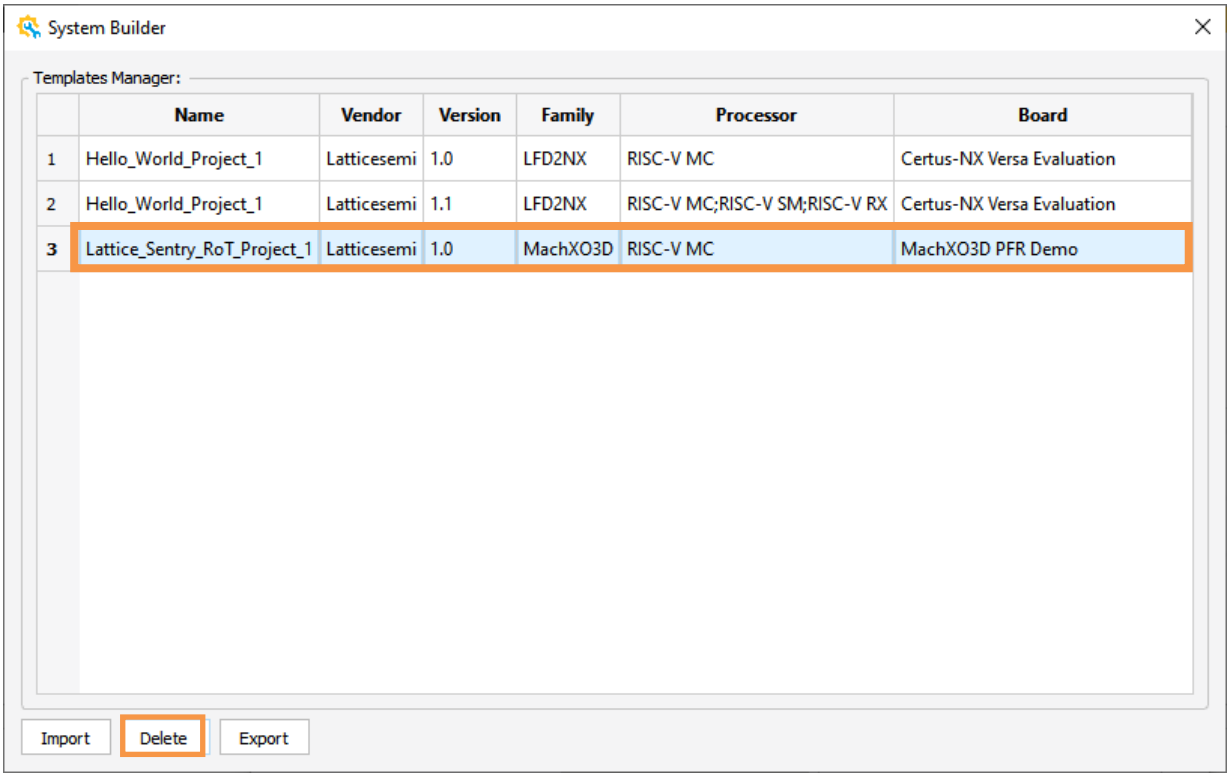


Figure 2.117. Templates Manager Page – Delete a Template

2.4.6. Include Sub Sbx File

Right Click on **Schematic View** and choose **Add Sbx Instance** (Figure 2.118), and then chose the sbx file you want to input (Figure 2.119).

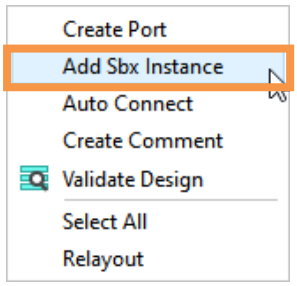


Figure 2.118. Right Click on Schematic View

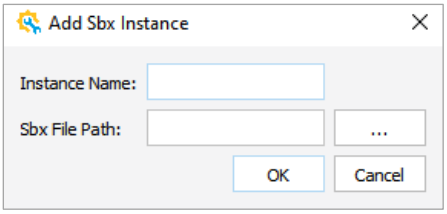


Figure 2.119. Input sbx File and Instance Name

Notes:

- The device in the sub sbx must be the same as the device in the top sbx.
- The sub sbx component name cannot be the same as the components in the top sbx file.
- Only one-level of hierarchy is supported. A sub sbx file cannot have hierarchical components.

Auto memory assignment considers memory map for a slave or address space of a manager in the sub system sbx, if there is interconnect bus to connect them. As shown in Figure 2.120, subm_inst and subs_inst are the sub sbx instances.

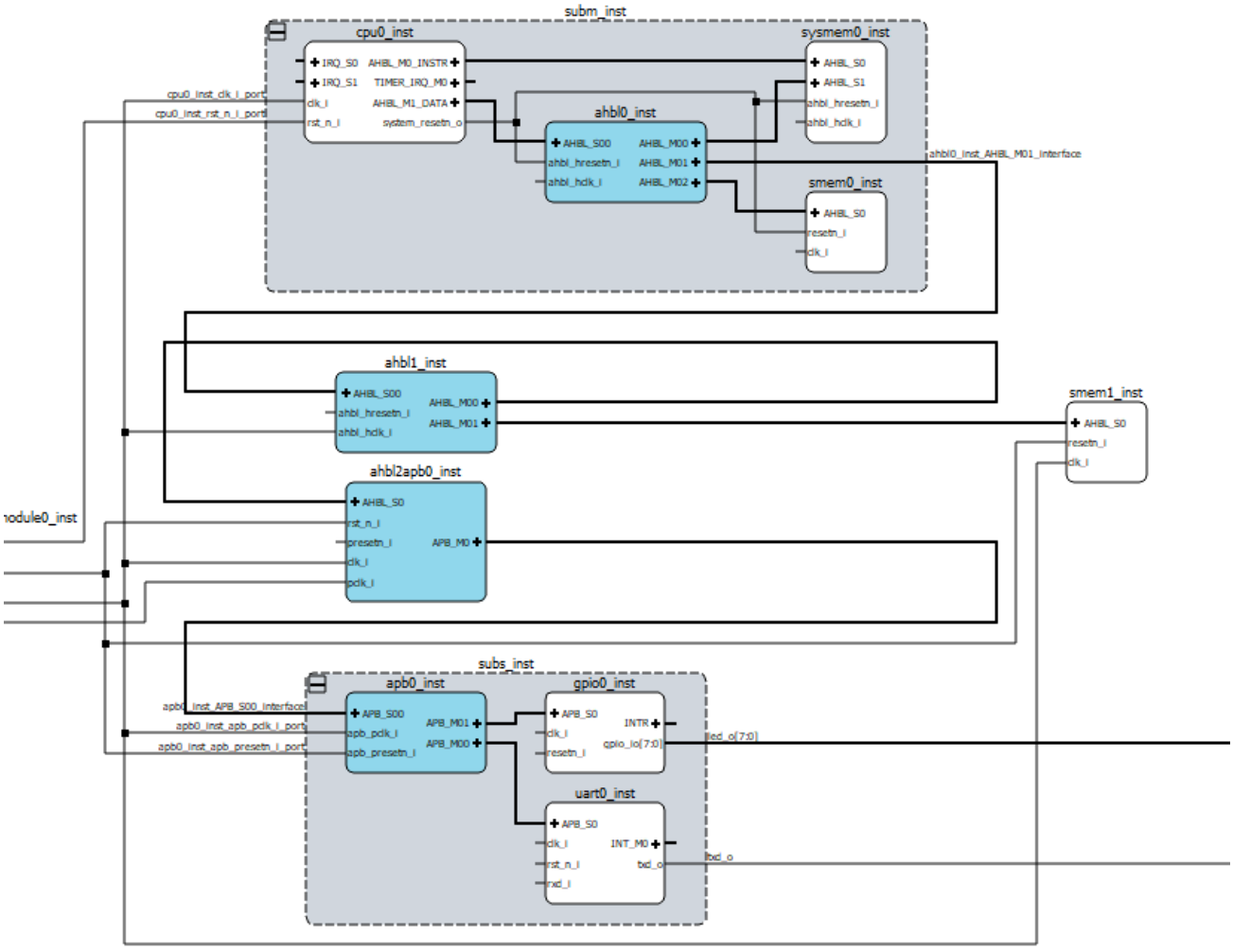
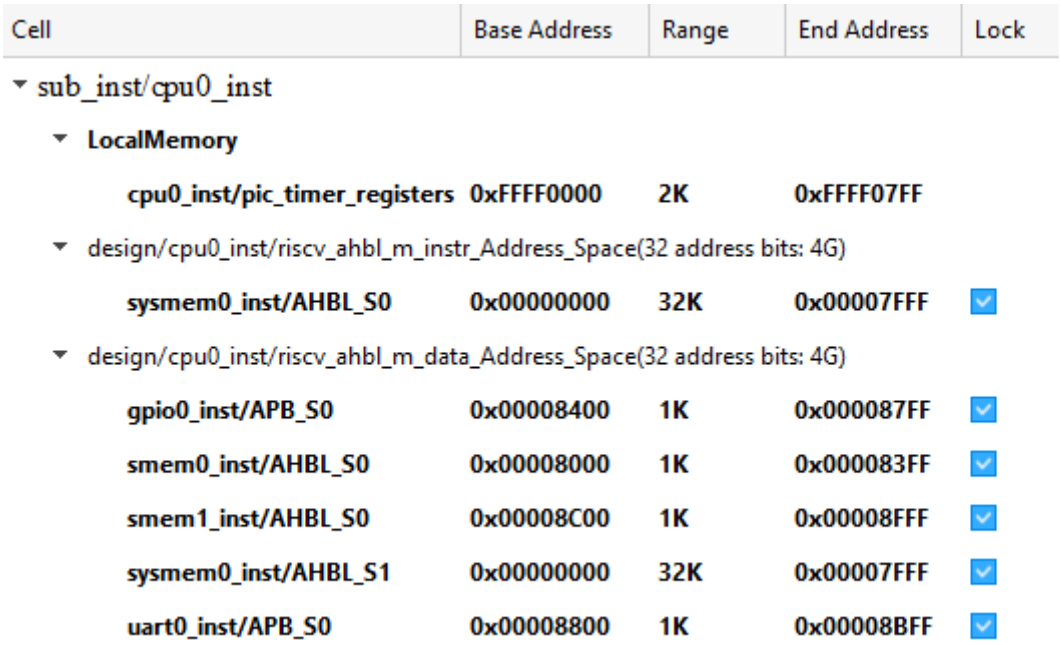


Figure 2.120. Sub major sbx and sub slave sbx

The memory information in sub system sbx is displayed in top-level address tab (Figure 2.121).



Cell	Base Address	Range	End Address	Lock
sub_inst/cpu0_inst				
LocalMemory				
cpu0_inst/pic_timer_registers	0xFFFF0000	2K	0xFFFF07FF	
design/cpu0_inst/riscv_ahbl_m_instr_Address_Space(32 address bits: 4G)				
sysmem0_inst/AHBL_S0	0x00000000	32K	0x00007FFF	☑
design/cpu0_inst/riscv_ahbl_m_data_Address_Space(32 address bits: 4G)				
gpio0_inst/APB_S0	0x00008400	1K	0x000087FF	☑
smem0_inst/AHBL_S0	0x00008000	1K	0x000083FF	☑
smem1_inst/AHBL_S0	0x00008C00	1K	0x00008FFF	☑
sysmem0_inst/AHBL_S1	0x00000000	32K	0x00007FFF	☑
uart0_inst/APB_S0	0x00008800	1K	0x00008BFF	☑

Figure 2.121. Memory Map for Sub Sbx

3. TCL Commands

Propel Builder provides TCL commands to execute actions. You can manually enter TCL commands in Tcl Console (Figure 3.1), if you prefer using command lines rather than using the GUI.

Figure 3.1. Tcl Console

3.1. sbp_design

The sbp_design command is used as one of the high-level management commands such as opening, closing, of the design files created by the Propel Builder.

3.1.1. Open

Opens an existing Propel Builder design for modification.

Usage	sbp_design open -name <design name> -path <design path> [-device <device name>]
Example	sbp_design open -name project1 -path project1.sbx

3.1.2. Close

Closes a Propel Builder design currently opened.

Usage	sbp_design close

3.1.3. New

Creates a new Propel Builder design.

Usage	sbp_design new -name <new design name> -path <new design path> -device <device name> -language <language type > –board <board name>

3.1.4. Save

Saves the current Propel Builder design to file on disk or save it as a new file. You can choose to save the design to the project location (Example 1) or to a specific path (Example 2).

Usage	sbp_design save [-path <new design path>]
Example 1	sbp_design save
Example 2	sbp_design save -path new_design.sbx

3.1.5. Drc

Runs the design rule check for the Propel Builder design file.

Usage	sbp_design drc

3.1.6. Generate

Generates RTL code to instantiate and connect the IP cores specified in the Propel Builder design file.

Usage	sbp_design generate

3.1.7. Auto Assign Addresses

Automatically assigns memory mapped addresses to all the slaves in the system. These addresses should be chosen to avoid slaves with multiple non-contiguous address ranges.

Usage	sbp_design auto_assign_addresses

3.1.8. Verify

Launches the SOC verification engine to verify design.

Usage	sbp_design verify [-h] [--workfolder <workfolder_path>] [--sbx_file <sbx_file_path>] {sim_gen,pfr_pack,regmap_gen,auto_run} positional arguments: {sim_gen,pfr_pack,regmap_gen,auto_run} specify the application sim_gen Simulation Generation Flow pfr_pack PFR Security Engine Packager Flow regmap_gen Memory Map Generation Flow auto_run Automation Flow optional arguments: -h, --help show this help message and exit --workfolder WORKFOLDER, -wf WORKFOLDER assign the working folder. Otherwise, the default one (current working folder) will be used --sbx_file SBX_FILE, -sf SBX_FILE specify the sbx file
Example	sbp_design verify –workfolder ./mem_map –sbx_file D:/ /XO3D_Initial/XO3D_Initial.sbx regmap_gen

3.1.9. PGE

Runs command with different parameters to call SGE function, DGE function, and Signature inside of PGE (Package Generate Engine).

- Runs command to call SGE to generate files for SDK.

Usage	sbp_design *pge sge –I <input path> [-o <output path>]* -i [Required] the top-level SoC project sbx file full path. -o [Optional] output full path, if the parameter is not specified, output path is the same as SoC project path. sge folder is generated at output path, at present sge folder under the SoC project root directory.

- Runs command to call DGE to generate TCL script that can be used to generate a Diamond or Radiant project.

Usage	sbp_design *pge dge –i <input path> [-o <output path>] [-diamond|-radiant]* -i [Required] the top level SoC project sbx file full path. -o [Optional] output full path, if the parameter is not specified, output path is the same as SoC project path. For DGE, tcl script and related files are generated at output path. At present, Diamond or Radiant project share the same workspace with SoC project. -diamond [Optional] flag to execute DGE for diamond project. -radiant [Optional] flag to execute DGE for Radiant project.

- Run command to generate signature.
- PGE gets partial UFM3 contents including version packetI), Sentry PFR configure data (D), and call Flash Address Tool to sign with private key from Factory HSM.

Usage	sbp_design *pge* gen_signature -cfile <c binary file> -dfile <d binary file> -output <output binary file>

3.1.10. Undo

Undo operation. Currently, software only supports single undo.

Usage	sbp_design undo

3.1.11. Redo

Redo operation. Currently, software only supports single redo.

Usage	sbp_design redo

3.1.12. Set Device

Modify the device information including device, package, speed, operating condition.

Usage	sbp_design set_device -device <device name> -speed <speed value> -package <package value> -operating <operating condition>

3.2. Other TCL Commands

The following commands are used for specifying connectivity and IP instantiation to the Propel Builder backend.

3.2.1. sbp_create_project

Create a new propel builder project.

Usage	sbp_create_project -name <new_project_name> -path <project_path> -language <Verilog/Vhdl> -psc <constrain_file_name> -device <device_name> -speed <device_speed> -board <board_name(version)>

3.2.2. sbp_add_component

Instantiates an IP component into the system. Must specify the component VLNV identifier. This corresponds to a component instance in the IP-XACT design. For example, the command below can instantiate an AHB-Lite interconnect component.

Usage	sbp_add_component -vlnv <VLNV> -name <instance_name>
Example	sbp_add_component -vlnv lattice:ip:ahbl_interconnect:1.0 -name ahblite_interconnect

3.2.3. sbp_add_sbxcomp

Instantiates a hierarchical component from sbx file into the system.

Usage	sbp_add_sbxcomp -name <hname> -path <sbx_file_absolute_path_name>
Example	sbp_add_sbxcomp -name sim_comp -path “C:/test/test.sbx”

3.2.4. sbp_add_gluelogic

Instantiates a gluelogic component into the system.

Usage	sbp_add_gluelogic –name <instance_name> -logicinfo <logic json info from sbp_create_glue_logic>
Example	sbp_add_gluelogic -name equation_module_inst -logicinfo [sbp_create_glue_logic equation equation_module {}“{ {"”x”r":"!A ” ”B","module_”a”e":"equation_mo”ule"}}]

Note: This is an internal command. Modify existing usage is not recommended.

3.2.5. sbp_create_glue_logic

Create gluelogic information when adding gluelogic component. rtl_path must be a file path if gluelogic comes from a file. Else be empty.

Usage	sbp_create_glue_logic <type> <module_name> <rtl_path> <json cfg data>
Example	sbp_add_gluelogic -name equation_module_inst -logicinfo [sbp_create_glue_logic equation equation_module {}“{ {"”x”r":"!A ” ”B","module_”a”e":"equation_mo”ule"}}]

Note: This is an internal command. Modify existing usage is not recommended.

3.2.6. sbp_ reconfig _gluelogic

Allow user to reconfig gluelogic component.

Usage	sbp_ reconfig _gluelogic –name <instance_name> -logicinfo <logic json info from sbp_create_glue_logic>
Example	sbp_reconfig_gluelogic -name equation_module0_inst -logicinfo [sbp_create_glue_logic equation equation_module0 {} { {"expr":"A & B","module_name":"equation_module0"}}]

Note: This is an internal command. Modify existing usage is not recommended.

3.2.7. sbp_add_port

Creates a top-level I/O port. Must specify the direction.

Usage	sbp_add_port [-from <bit number>] [-to <bit number>] -direction <in/out/inout> <port_name>

3.2.8. sbp_modify_port

Modify existing top-level ports to change the width and the direction.

Usage	sbp_modify_port -name <port_name> [-from <bit number>] [-to <bit number>] -direction <dir>

3.2.9. sbp_connect_net

Connects all of the specified pins and/or ports to the same net. The arguments can be pins or ports in the system design. Only one of the arguments can be the driver (output pin/port), driving all other input pin/ports. Example below connects the clk port to all components, assuming component pins are all named clk.

Usage	sbp_connect_net [-name <net name>] <pin/port> <pin/port>
Example	sbp_connect_net [sbp_get_pins clk] {CLOCK_IN}

3.2.10. sbp_connect_interface_net

Connects a bus interface pin/port to another interface pin/port. This corresponds to the interconnection element in the IP-XACT design.

Usage	sbp_connect_interface_net <pin/port> <pin/port>

3.2.11. sbp_connect_constant

Connects a constant integer to a pin/pinbus/port/portbus. To assign to a pin/pinbus, the object must be an input pin/pinbus. To assign to a port/portbus, the object must be an output port/portbus. If the integer requires multiple bits, not 0 or 1, then the object must be a bus. The tcl command can be used to assign the same constant to multiple pin/pinbus/port/portbus at the same time.

Usage	sbp_connect_constant -constant <integer> <pin/pin bus/ port/portbus> <pin/pin bus/ port/portbus>…
Example	sbp_connect_constant -constant 1 {test/i2c_mst_apb/rst_n_i} {test/riscv/clk_i}

3.2.12. sbp_connect_whitebox

User can do connection cross the hierarchy boundary for verification purpose.

Usage	sbp_connect_whitebox <design_name/vip_inst_name/pin_name> <design_name/uit_inst_name/port_name>

3.2.13. sbp_connect_ group

Allow user to connect a group of signals.

Usage	sbp_connect_group -ports {<source port1> <destination port1> <destination port2>; <source port2> <destination port3> <destination port4>} -nets {<net_name1> <destination portm1> <destination portm2>; <net_name2> <destination portm3>} -interfaces {<interface1> <interface2>; <interface3> <interface4>}

3.2.14. sbp_disconnect_whitebox

Removes the connection cross the hierarchy boundary.

Usage	sbp_disconnect_whitebox <design_name/vip_inst_name/pin_name> <design_name/uit_inst_name/port_name>

3.2.15. sbp_disconnect_interface_net

Disconnects an interface pin/port from the interface nets they attached to. Note that any interface pin or interface port can attach to one interface net at the most.

Usage	sbp_disconnect_interface_net <pin/port> <pin/port>

3.2.16. sbp_disconnect_net

Disconnects all of the specified input pins and/or ports from the nets they attached to. Note that any pin or port can attach to one net at the most. Can also be used to disconnect a constant that is connected to a pin/port with connect_constant.

Usage	sbp_disconnect_net <pin0> <pin1> <port2>

3.2.17. sbp_assign_addr_seg

Assigns a memory map between a pair of master and slave interfaces. Range specifies the range of the segment, for example, 32‘h0000400, 32‘h0001000. Offset specifies the base offset of the range, for example, 32‘h0000400

Usage	sbp_assign_addr_seg -offset <offset> < slave connection name>
Example	sbp_assign_addr_seg -offset 32‘h00001000 simple/riscv/AHBL_S00

3.2.18. sbp_unassign_addr_seg

The Tcl command unsets the fixed offset flag for a memory map allowing the auto_assign Tcl command to assign the memory map offset.

Usage	sbp_unassign_addr_seg < slave interface name>
Example	sbp_unassign_addr_seg simple/spi/AHB_S00

3.2.19. sbp_assign_local_memory

Assigns a base address to a local memory map of a master address space.

Usage	sbp_assign_local_memory -offset <offset> <master_addr_space>
Example	sbp_assign_local_memory -offset ‘h0050000 Foundation_SoC/riscv/ahbl_m_data_Address_Space

3.2.20. sbp_export_pins

Exports a list of pins or interface pins, or all not-yet-connected pins of the components to the top-level port list in the design. The function detects whether or not the argument is pin(s) or component(s). In the two examples below, Example 1 demonstrates a Tcl command to export the pin init_done, while Example 2 demonstrates a Tcl command to export all pins and interfaces of the ddr3 component.

Usage	sbp_export_pins <pin/component> <pin/component>
Example 1	sbp_export_pins {ddr3/init_done}
Example 2	sbp_export_pins {ddr3}

3.2.21. sbp_export_interface

Exports bus interfaces that are passed as arguments to the Tcl command from the component to the top-level component. Example below exports the AHBL_MASTER bus interface of the RISC-V component to the top-level component.

Usage	sbp_export_interfaces <interface> <interface> <interface>
Example	sbp_export_interfaces simple/riscv/AHBL_MASTER

3.2.22. sbp_rename

The rename Tcl command renames objects within the design. The new name of the object and the current hierarchical name of the given object are used as the parameter value. The object can be an interface connection, connection, port, interface, or component. The example below demonstrates the changing of the name of a port in milestone project from CLK to CLOCK.

Usage	sbp_rename -name <new name> <object name>
Example	sbp_rename -name CLOCK milestone/CLK

3.2.23. sbp_replace

The Replace (Re-Config) Tcl command replaces a component with a new configuration for itself. VLNV refers to the newly-generated IP, component name refers to the existing component that is to be replaced, and instance refers to the instance name of the component with new configuration.

Usage	sbp_replace -vlnv <VLNV> -name <instance> -component <component name>
Example	sbp_replace -vlnv lattice:ip:ahblite_bus_0:1.1 -name ahbl_bus_0 -component simple/ahblite

3.2.24. sbp_copy

Copies objects, IP instances and nets, from the current or other open sbp designs to the current sbp design. All objects are post-fixed with a postfix string. For example, you can write TCL code below to duplicate the components and connections with new instance names post fixed with X, by calling copy on all the components, ports and nets. Pins are automatically duplicated while the components are duplicated.

Usage	sbp_copy -postfix <postfixString> objects
Example	sbp_copy –postfix X $selected_objs

3.2.25. sbp_delete

Deletes objects, IP instances and nets, from the current sbp design. In the examples below, Example 1 demonstrates a Tcl command to delete a port, while Example 2 demonstrates a Tcl command to delete ddr3 component.

Usage	Sbp_delete objects -type <type name>
Example 1	sbp_delete [sbp_get_ports <clock>] -type port
Example 2	sbp_delete {ddr3} -type component

3.2.26. sbp_get_pins

Gets a list of pin names that match a pattern string, and/or the pins that are associated with an object. The object in the [-from <objectName>] option can be a net or a component. The example below gets the clk pin from the interconnect IP.

Usage	sbp_get_pins [-from <object Name>] [pattern]
Example	sbp_get_pins -from ahblite_interconnect clk

3.2.27. sbp_get_interface_pins

Gets a list of interface names that match a pattern string, and/or the interfaces that are associated with an object. The object in the [-from <objectName>] option can be an interface net or a component. The example below can get all AHB-Lite slave interface pins from the interconnect IP.

Usage:	sbp_get_interface_pins [-from <objectName>] [pattern]
Example:	sbp_get_interface_pins -from ahblite_interconnect S*_AHB

3.2.28. sbp_get_ports

Get a list of the names of ports that match a pattern string, and/or the ports that are associated with an object. The object in the [-from <objectName>] option can be a net.

Usage	sbp_get_ports [-from <objectName>] [pattern]

3.2.29. sbp_get_interface_ports

Gets a list of interface names that match a pattern string, and/or the interface ports that are associated with an object. The object in the [-from <objectName>] option can be an interface net.

Usage	sbp_get_interface_ports [-from <objectName>] [pattern]

3.2.30. sbp_get_nets

Gets a list of net names that match a pattern string, and/or the nets that are associated with an object. The object in the [-from <objectName>] option can be a pin or a port.

Usage	sbp_get_nets [-from <objectName>] [pattern]

3.2.31. sbp_get_interface_nets

Gets a list of interface net names that match a pattern string, and/or the interface nets that are associated with an object. The object in the [-from <objectName>] option can be an interface pin or an interface port.

Usage	sbp_get_interface_nets [-from <objectName>] [pattern]

3.2.32. sbp_set_property

Sets the properties of an input object. The first argument is a list of name value pairs. We have a list of parameters because the GUI IP configuration dialog submits changes to many parameters of an IP component at the same time when the dialog is closed. In the examples below, Example 1 changes the data width of the RAM block named “ebr_0”, while Example 2 changes the number of master interfaces to two and number of slave interface to three on AHB-Lite interconnect block named “ahbl_interconnect”.

Usage	sbp_set_property <name0 value0 name1 value1 …> object
Example 1	sbp_set_property {datawidth 32} {test/ebr_0}
Example 2	sbp_set_property {NUM_MI 2 NUM_SI 3} test/ahbl_interconnect

3.2.33. sbp_get_property

Gets the property of the object. The example below is to get the number of slave interfaces of AHB-Lite interconnect block named “ahbl_interconnect_0”.

Usage	sbp_get_property <parameter name> <object>
Example	sbp_get_property NUM_SI ahbl_interconnect_0

3.2.34. sbp_report_properties

Prints all the properties and values associated to the type of the object.

Usage	sbp_report_properties object
Example	sbp_report_properties ahbl_interconnect
Outputs	Name: ahbl_interconnect NUM_SI: 1 NUM_MI: 2 DATA_WIDTH: 32 ADDRESS_WIDTH: 32

3.2.35. sbp_get_components

Gets a list of component names that match a pattern string, and/or the components that are associated with an object. The default pattern string is a wildcard “*” that matches all components. The pattern string may consist of string segments and wildcards. The example below returns all the component names that contain “interconnect”. The command returns an empty string if no match is found.

Usage	sbp_get_components <component name>
Example	sbp_get_components {*interconnect*}

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>
```

```
  <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"/>
```

```
    <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"
```

```
            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"
```

```
            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>
```

```
            </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-->
```

```
  <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">
```

```
    <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>
```

```
  <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"/>
```

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

</xs:schema>
```

References

- Lattice Propel 2023.1 SDK User Guide (FPGA-UG-02186)
- Lattice IP Packager 2023.1 User Guide (FPGA-UG-02187)
- Lattice Propel 2023.1 Installation for Windows User Guide (FPGA-AN-02067)
- Lattice Propel 2023.1 Installation for Linux User Guide (FPGA-AN-02066)
- Lattice Propel 2023.1 Release Notes (FPGA-AN-02065)
- Lattice Diamond Online Help
- Lattice Radiant Online Help

Technical Support Assistance

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

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

Revision History

Revision 1.0, June 2023

Section	Change Summary
All	Production release.

www.latticesemi.com