



# Lattice Propel 2024.1 Builder

## User Guide

FPGA-UG-02212-1.0

June 2024

## Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults, and all associated risk is the responsibility entirely of the Buyer. The information provided herein is for informational purposes only and may contain technical inaccuracies or omissions, and may be otherwise rendered inaccurate for many reasons, and Lattice assumes no obligation to update or otherwise correct or revise this information. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. LATTICE PRODUCTS AND SERVICES ARE NOT DESIGNED, MANUFACTURED, OR TESTED FOR USE IN LIFE OR SAFETY CRITICAL SYSTEMS, HAZARDOUS ENVIRONMENTS, OR ANY OTHER ENVIRONMENTS REQUIRING FAIL-SAFE PERFORMANCE, INCLUDING ANY APPLICATION IN WHICH THE FAILURE OF THE PRODUCT OR SERVICE COULD LEAD TO DEATH, PERSONAL INJURY, SEVERE PROPERTY DAMAGE OR ENVIRONMENTAL HARM (COLLECTIVELY, "HIGH-RISK USES"). FURTHER, BUYER MUST TAKE PRUDENT STEPS TO PROTECT AGAINST PRODUCT AND SERVICE FAILURES, INCLUDING PROVIDING APPROPRIATE REDUNDANCIES, FAIL-SAFE FEATURES, AND/OR SHUT-DOWN MECHANISMS. LATTICE EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS OF THE PRODUCTS OR SERVICES FOR HIGH-RISK USES. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

## Inclusive Language

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

# Contents

Contents .....	3
Abbreviations in This Document.....	8
1. Introduction.....	9
1.1. Purpose .....	9
1.2. Audience .....	9
2. Lattice Propel Builder Design.....	10
2.1. Builder Environment .....	10
2.2. Project Design Flow.....	11
2.2.1. Creating SoC Project .....	11
2.2.2. Opening an Existing SoC Project .....	20
2.2.3. Generating and Instantiating IP/Module .....	22
2.2.4. Adding Glue Logic .....	29
2.2.5. Working with the Schematic View.....	36
2.2.6. Connecting Modules .....	46
2.2.7. Creating Top-level Ports .....	58
2.2.8. Adjusting Address Spaces .....	61
2.2.9. Validating the Design .....	62
2.2.10. Generating the Propel Design.....	63
2.2.11. Generating the Memory Report .....	64
2.2.12. Launch Project in Diamond or Radiant Software.....	65
2.2.13. Launching SDK and Generate Mem File.....	70
2.2.14. Others .....	76
2.3. Verification Project Flow .....	88
2.3.1. Creating a Verification Project.....	88
2.3.2. Switching from SoC Design Project to Verification Project.....	90
2.3.3. Opening a Verification Project.....	92
2.3.4. Adding Modules, IP and VIPs .....	92
2.3.5. Working with the Schematic View .....	92
2.3.6. Connecting Modules .....	92
2.3.7. Viewing Address Maps.....	92
2.3.8. Monitoring DUT .....	93
2.3.9. Generating Simulation Environment .....	94
2.3.10. Launching Simulation.....	96
2.4. Advanced Usage.....	98
2.4.1. Supported Interface .....	98
2.4.2. Supported Language .....	98
2.4.3. Supported Hierarchical IP .....	98
2.4.4. Export Interface .....	99
2.4.5. Define Custom Template .....	100
2.4.6. Include Sub Sbx File .....	109
3. TCL Commands .....	112
3.1. sbp_design Commands .....	112
3.1.1. Open .....	112
3.1.2. Close .....	112
3.1.3. New.....	112
3.1.4. Save.....	112
3.1.5. Drc.....	112
3.1.6. Generate.....	113
3.1.7. Auto Assign Addresses.....	113
3.1.8. Verify.....	113
3.1.9. PGE.....	113
3.1.10. Undo .....	114

3.1.11. Redo .....	114
3.1.12. Set Device .....	114
3.2. Other TCL Commands.....	114
3.2.1. sbp_create_project.....	114
3.2.2. sbp_add_component.....	114
3.2.3. sbp_add_sbxcamp .....	115
3.2.4. sbp_config_ip.....	115
3.2.5. ip_catalog_list.....	115
3.2.6. ip_catalog_install.....	115
3.2.7. ip_catalog_uninstall.....	115
3.2.8. sbp_upgrade_component.....	116
3.2.9. sbp_add_glue_logic.....	116
3.2.10. sbp_create_glue_logic.....	116
3.2.11. sbp_reconfig_glue_logic.....	116
3.2.12. sbp_add_port.....	116
3.2.13. sbp_modify_port.....	116
3.2.14. sbp_connect_net.....	116
3.2.15. sbp_connect_interface_net.....	117
3.2.16. sbp_connect_constant.....	117
3.2.17. sbp_connect_whitebox.....	117
3.2.18. sbp_connect_group.....	117
3.2.19. sbp_disconnect_whitebox.....	117
3.2.20. sbp_disconnect_interface_net.....	117
3.2.21. sbp_disconnect_net.....	117
3.2.22. sbp_assign_addr_seg.....	118
3.2.23. sbp_unassign_addr_seg.....	118
3.2.24. sbp_assign_local_memory.....	118
3.2.25. sbp_export_pins.....	118
3.2.26. sbp_export_interface.....	118
3.2.27. sbp_rename.....	118
3.2.28. sbp_replace.....	119
3.2.29. sbp_copy.....	119
3.2.30. sbp_delete.....	119
3.2.31. sbp_get_pins.....	119
3.2.32. sbp_get_interface_pins.....	119
3.2.33. sbp_get_ports.....	119
3.2.34. sbp_get_interface_ports.....	120
3.2.35. sbp_get_nets.....	120
3.2.36. sbp_get_interface_nets.....	120
3.2.37. sbp_set_property.....	120
3.2.38. sbp_get_property.....	120
3.2.39. sbp_report_properties.....	120
3.2.40. sbp_get_components.....	121
3.2.41. sbp_design set_prj_option.....	121
3.2.42. sbp_design gen_tcl.....	121
References .....	122
Technical Support Assistance .....	123
Revision History .....	124

## Figures

Figure 2.1. Propel Builder Workflow .....	10
Figure 2.2. Propel Builder Workbench Window .....	11
Figure 2.3. Create System Design – Design Information Wizard .....	12
Figure 2.4. Specify a Device for Template SoC Project .....	13
Figure 2.5. Specify a Board for Template SoC Project (1) .....	14
Figure 2.6. Specify a Board for Template SoC Project (2) .....	15
Figure 2.7. Specify a Board for Template SoC Project (3) .....	16
Figure 2.8. Project Information Wizard .....	16
Figure 2.9. Propel Builder GUI showing Template SoC Project .....	17
Figure 2.10. New SoC Project in Another Way .....	17
Figure 2.11. New SoC Project – Select Template .....	18
Figure 2.12. New SoC Project – Select Custom Template .....	19
Figure 2.13. New SoC Project – Select Device .....	20
Figure 2.14. Open Design .....	20
Figure 2.15. Open Existing Project .....	21
Figure 2.16. Open Recent Project .....	21
Figure 2.17. IP on Server .....	22
Figure 2.18. Hover Mouse over the Icon .....	23
Figure 2.19. IP Filter and Search .....	24
Figure 2.20. IP Catalog .....	25
Figure 2.21. Module/IP Block Wizard – Generate Component from Module gpio Version 1.6.0 .....	26
Figure 2.22. Module/IP Block Wizard – Configure Component from IP Gpio Version 1.6.0 .....	26
Figure 2.23. Module/IP Block Wizard – Check Generated Result .....	27
Figure 2.24. Design Instance Dialog Box .....	27
Figure 2.25. Propel Builder Schematic View Shows the Module Instance .....	28
Figure 2.26. Regenerate Module/IP .....	28
Figure 2.27. Upgrade All IPs .....	29
Figure 2.28. Glue Logic Section of the IP Catalog .....	30
Figure 2.29. Glue Logic for Concat Module .....	31
Figure 2.30. Schematic View Shows a Concat Module .....	31
Figure 2.31. Glue Logic Wizard for Equation Module .....	32
Figure 2.32. Schematic View Shows an Equation Module .....	32
Figure 2.33. Schematic View Shows the Invert Module .....	33
Figure 2.34. Glue Logic Wizard for RTL Module .....	33
Figure 2.35. Existing RTL Module Configuration .....	34
Figure 2.36. Schematic View Shows the Custom RTL Module .....	35
Figure 2.37. Glue Logic Wizard for Split Module .....	35
Figure 2.38. Schematic View Shows Split Module .....	36
Figure 2.39. Signal List of Modules .....	37
Figure 2.40. Select Object .....	38
Figure 2.41. Locate Objects .....	39
Figure 2.42. Duplicate a Module .....	39
Figure 2.43. Design View .....	40
Figure 2.44. Define Instance Dialog Box .....	40
Figure 2.45. Module/IP Block Wizard – Configure Component .....	41
Figure 2.46. Select Module .....	42
Figure 2.47. Options Dialog .....	43
Figure 2.48. Show Connectivity of the Module .....	44
Figure 2.49. Highlight an Object .....	44
Figure 2.50. Object Properties .....	45
Figure 2.51. Print Preview .....	45
Figure 2.52. Draw a Pin or a Port .....	46

Figure 2.53. Draw Nets .....	46
Figure 2.54. Select More than One Ports.....	47
Figure 2.55. Pin/Net/Pinface/Netface Display Filter.....	47
Figure 2.56. Pin/Pinface is Hidden .....	48
Figure 2.57. Action Menu of Right-clicking on a Blank .....	49
Figure 2.58. Action Menu of Connecting Ports.....	50
Figure 2.59. Select One Clock/Reset Port .....	51
Figure 2.60. Connecting CPU Instance Reset Output Port.....	52
Figure 2.61. Interface Type of cpu0_inst.AHBL_M1_DATA .....	53
Figure 2.62. Interface Connection on from CPU to Bus/Bridge instances in Schematic View .....	53
Figure 2.63. Interface Connection from CPU to Bus/Bridge Instances in Connect Port View .....	54
Figure 2.64. Interface Connection on Other Instances Connecting back to CPU in Schematic View .....	54
Figure 2.65. Interface Connection on Other Instances Connecting back to CPU in Connect Port View .....	55
Figure 2.66. Interface Type of gpio0_inst.INTR .....	55
Figure 2.67. Interface Type of cpu0_inst.IRQ_S0.....	56
Figure 2.68. Right-click Menu of an Input Pin.....	57
Figure 2.69. Dialog Box of Assigning Constant Value to an Input Pin .....	57
Figure 2.70. Create Port Dialog Box.....	58
Figure 2.71. Input Port.....	58
Figure 2.72. Output Port and Inout Port.....	58
Figure 2.73. Port Edit Dialog .....	59
Figure 2.74. Properties of Port.....	59
Figure 2.75. Port Direction.....	60
Figure 2.76. Open TCL History .....	60
Figure 2.77. TCL History Window.....	61
Figure 2.78. PortBus Direction .....	61
Figure 2.79. Address View .....	62
Figure 2.80. Edit Base Address.....	62
Figure 2.81. Optional Bus Ports Handling .....	63
Figure 2.82. Memory Report .....	64
Figure 2.83. Memory Report .....	64
Figure 2.84. Memory Report of Each Instance .....	65
Figure 2.85. Diamond Project .....	66
Figure 2.86. Generate Programming Files .....	67
Figure 2.87. Radiant Project .....	68
Figure 2.88. Radiant Project for Project before Propel Builder 2024.1 .....	69
Figure 2.89. Radiant Project for Project Since Propel Builder 2024.1 .....	69
Figure 2.90. Lattice Propel Launcher Wizard .....	70
Figure 2.91. Propel SDK GUI and C/C++ Project Wizard .....	71
Figure 2.92. Create C/C++ Project.....	72
Figure 2.93. Build C/C++ Project .....	73
Figure 2.94. Mem File for Simulation .....	73
Figure 2.95. Mem File Location .....	74
Figure 2.96. Initialize Memory for System Memory Module (1) .....	74
Figure 2.97. Initialize Memory for System Memory Module (2) .....	75
Figure 2.98. ECO flow in Radiant .....	76
Figure 2.99. Design View of Device Part Number .....	77
Figure 2.100. Modify Device Info.....	78
Figure 2.101. System Builder Dialog .....	78
Figure 2.102. Design View of Device Part Number .....	79
Figure 2.103. Theme Change .....	80
Figure 2.104. General Options.....	81
Figure 2.105. Define Instance Dialog .....	82
Figure 2.106. Connect Ports Dialog.....	82

Figure 2.107. Color .....	83
Figure 2.108. Network Settings .....	84
Figure 2.109. Directories .....	85
Figure 2.110. IP Public Path .....	86
Figure 2.111. IP Catalog .....	87
Figure 2.112. Map Network Device .....	87
Figure 2.113. Create System Design – Design Information Wizard .....	88
Figure 2.114. Create System Design – Propel Project Configure Wizard .....	89
Figure 2.115. Verification Project .....	90
Figure 2.116. Whole SoC Design .....	90
Figure 2.117. SoC Design Project .....	91
Figure 2.118. Verification Project .....	91
Figure 2.119. Reload dut_inst after Each SoC Project Design Change .....	92
Figure 2.120. Address Maps .....	93
Figure 2.121. Monitoring DUT .....	93
Figure 2.122. Testbench of the Verification Project .....	94
Figure 2.123. Initialization of Mem File .....	94
Figure 2.124. Generate Simulation Environment on Verification Project .....	95
Figure 2.125. Testbench File Structure .....	95
Figure 2.126. Remind for Simulation Tool Change .....	96
Figure 2.127. Launch Simulation in Verification Project .....	96
Figure 2.128. Questa Simulation GUI .....	96
Figure 2.129. Testbench File Structure in Simulation Project .....	97
Figure 2.130. Hierarchical IP .....	98
Figure 2.131. Hierarchical IP Scope in Detail .....	99
Figure 2.132. Export Interface Example .....	99
Figure 2.133. Memory Map for ahbl_s0_inst in Address Editor .....	100
Figure 2.134. Export Template Configuration Page .....	101
Figure 2.135. Check IP Name in Design View .....	102
Figure 2.136. Check Device Supported in IP Information .....	102
Figure 2.137. Double Click on the Constraint File Name .....	103
Figure 2.138. Export Template to PropelTemplateLocal Folder .....	104
Figure 2.139. Template Manager Entry .....	105
Figure 2.140. Templates Manager Page .....	105
Figure 2.141. Import a .ptmp File .....	106
Figure 2.142. Template Manager Page .....	107
Figure 2.143. Template Manager Page – Change to New Location .....	108
Figure 2.144. Templates Manager Page – Delete a Template .....	109
Figure 2.145. Right Click on Schematic View .....	109
Figure 2.146. Input sbx File and Instance Name .....	109
Figure 2.147. Sub major sbx and sub subordinate sbx .....	110
Figure 2.148. Memory Map for Sub Sbx .....	111
Figure 3.1. TCL Console .....	112

## Tables

Table 2.1. Different Device Families Supported in Radiant or Diamond .....	65
--	----

## Abbreviations in This Document

A list of abbreviations used 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
DGE	Design Generator Engine
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 <sup>2</sup> C.
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.
SGE	Software Generator Engine
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™ 2024.1 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 2024.1 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. The workflow is described below, and the details are explained in the following sections.

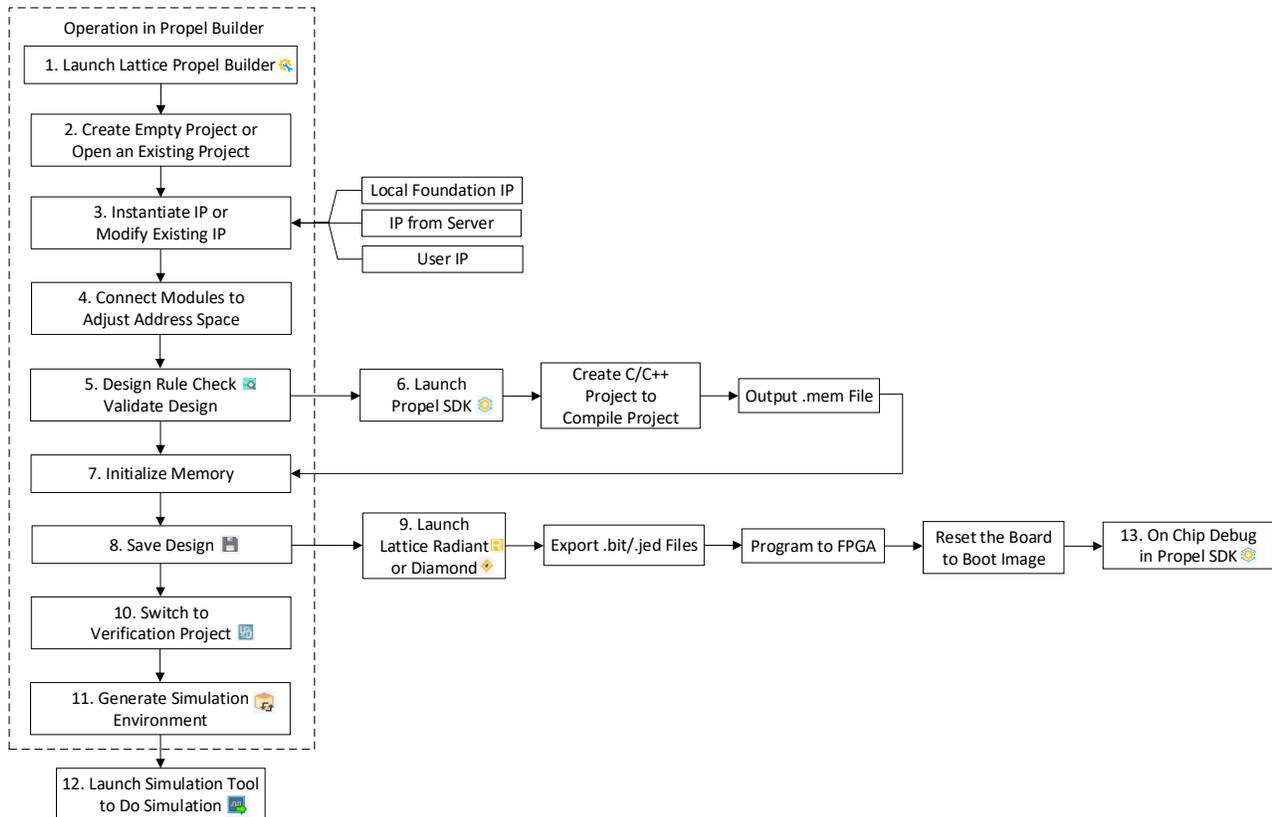


Figure 2.1. Propel Builder Workflow

### 2.1. Builder Environment

After Propel 2024.1 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/propelbldwrap -gui`

Refer to the [Lattice Propel 2024.1 Installation for Windows User Guide \(FPGA-AN-02088\)](#) and [Lattice Propel 2024.1 Installation for Linux User Guide \(FPGA-AN-02089\)](#) 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 from SoC Design 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 2.2](#) 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

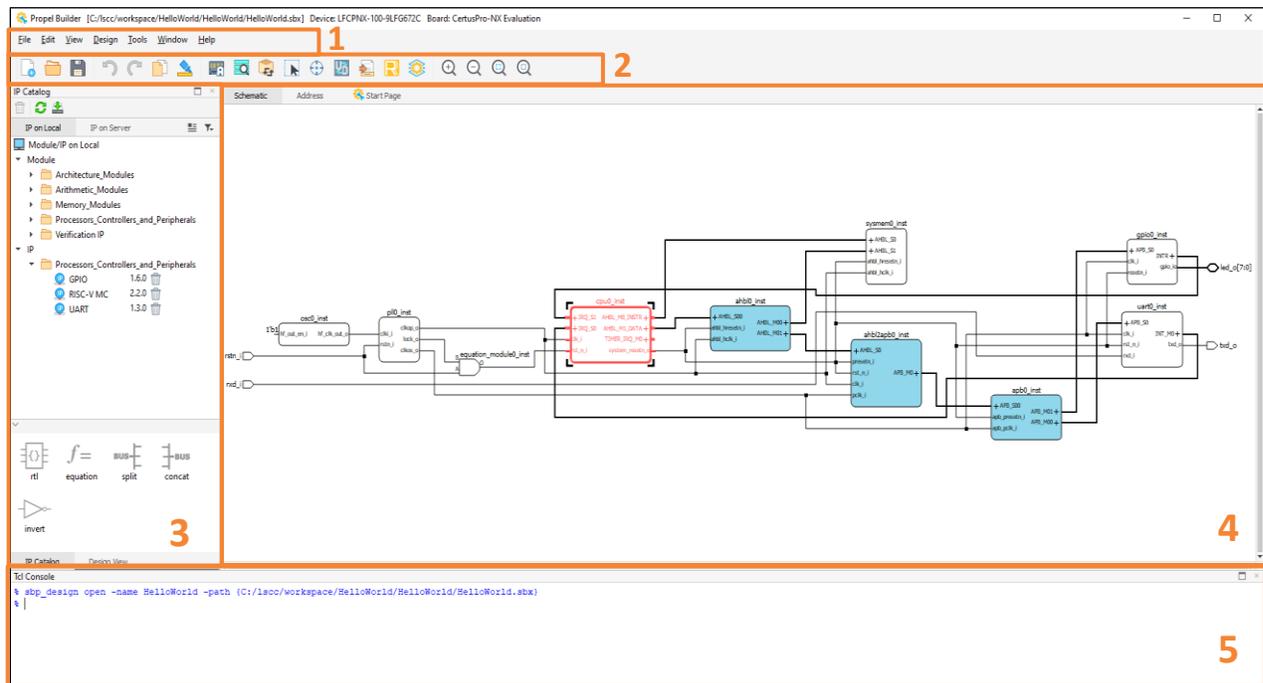


Figure 2.2. Propel Builder Workbench Window

## 2.2. Project Design Flow

### 2.2.1. Creating SoC Project

Choose **File** > **New Design** from the Lattice Propel Builder Menu bar. The Create System Design wizard ([Figure 2.3](#)) opens.

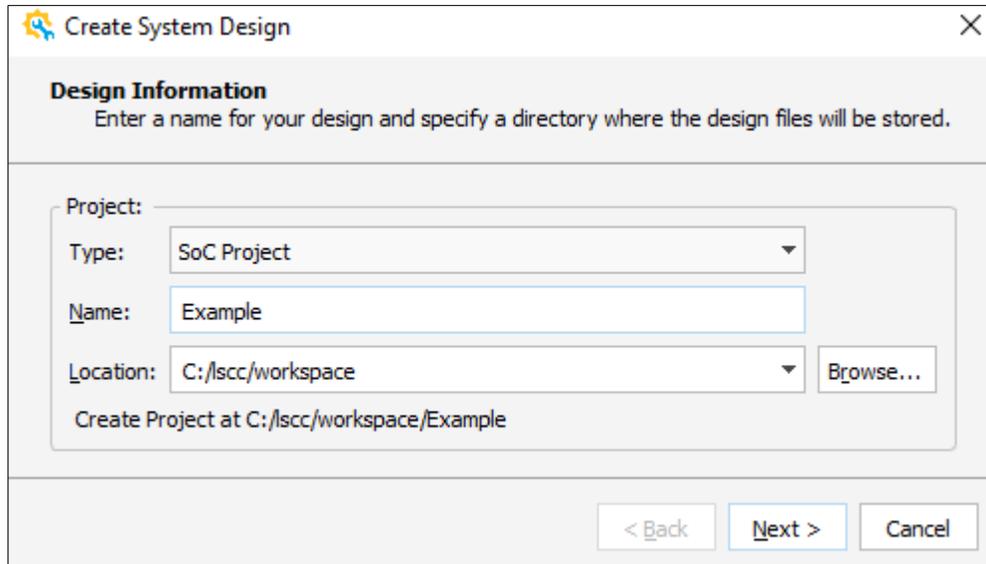


Figure 2.3. Create System Design – Design Information Wizard

1. The default Project Type is displayed in the **Type** field. Select **SoC Project** from the drop-down menu.
2. Enter a project name in the **Name** field, such as 'Example'.
3. (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. If there is a file existed under the path but get a prompt of No such file, try moving this file to a directory with a shorter path.

4. Click **Next**. In the Configure Propel Project wizard, you can specify a device or a board for a SoC project. To specify a device for your new SoC project, use the **drop-down** menu to select desired device information (Family, Device, Package, Speed).
5. Select **Template SoC Project (RISC-V \*\*\* Project)** or **Empty Project** in the **Templates** field (Figure 2.4).  
Templates are customized with commonly used programs as well as pre-defined family, device, package, speed, and operating condition. There are several templates here, which are programmed to run demo. You can make changes based on these templates.

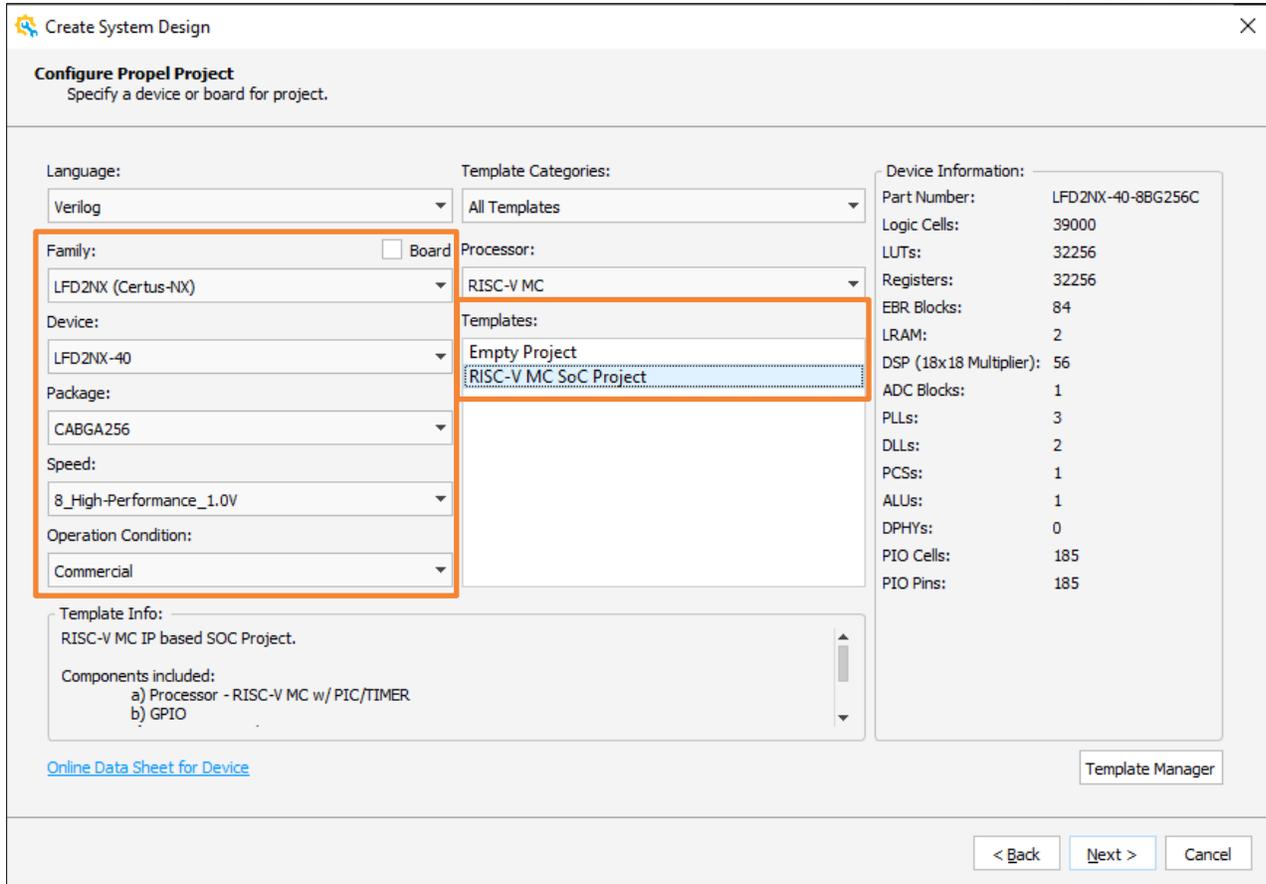


Figure 2.4. Specify a Device for Template SoC Project

- Or, to specify a board for a new Template SoC project, check the **Board** box (Figure 2.5).

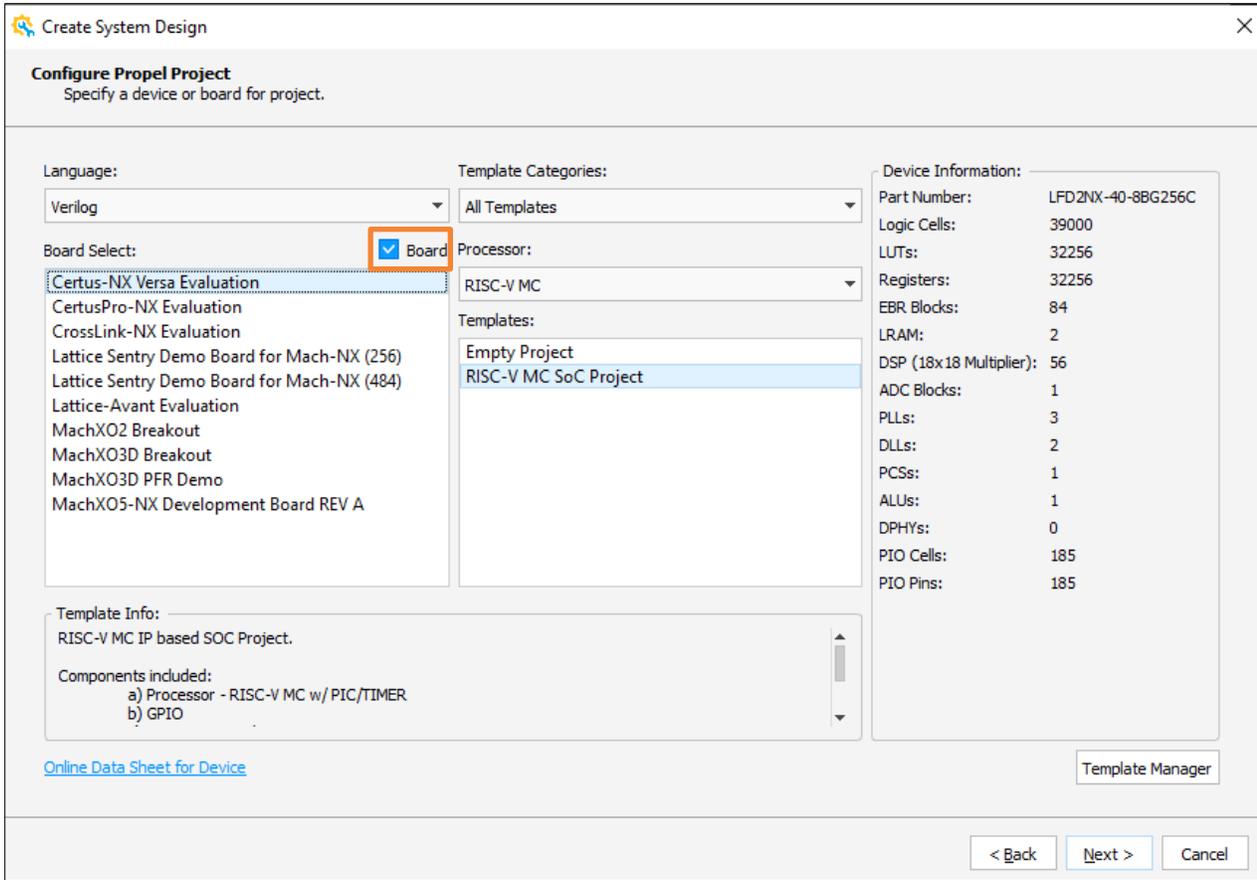


Figure 2.5. Specify a Board for Template SoC Project (1)

**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 RISC-V MC SoC Project.
  - Templates usually have been validated on that board and some of them may have certain constraints. Check [Lattice Propel 2024.1 Release Notes \(FPGA-AN-02090\)](#) for more information on constraints.
6. The Configure Propel Project wizard is shown in [Figure 2.6](#). From the **Board Select** area, select the desired board.

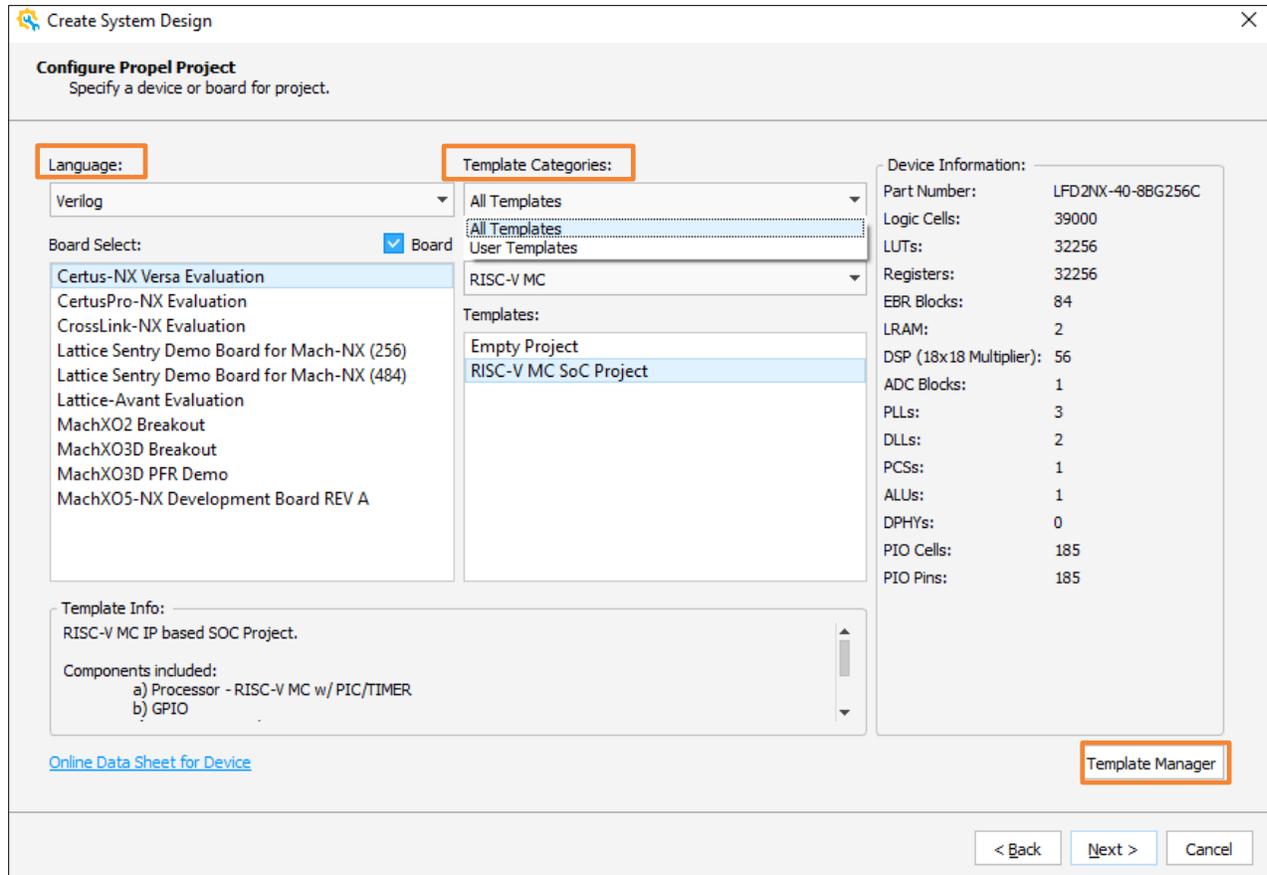
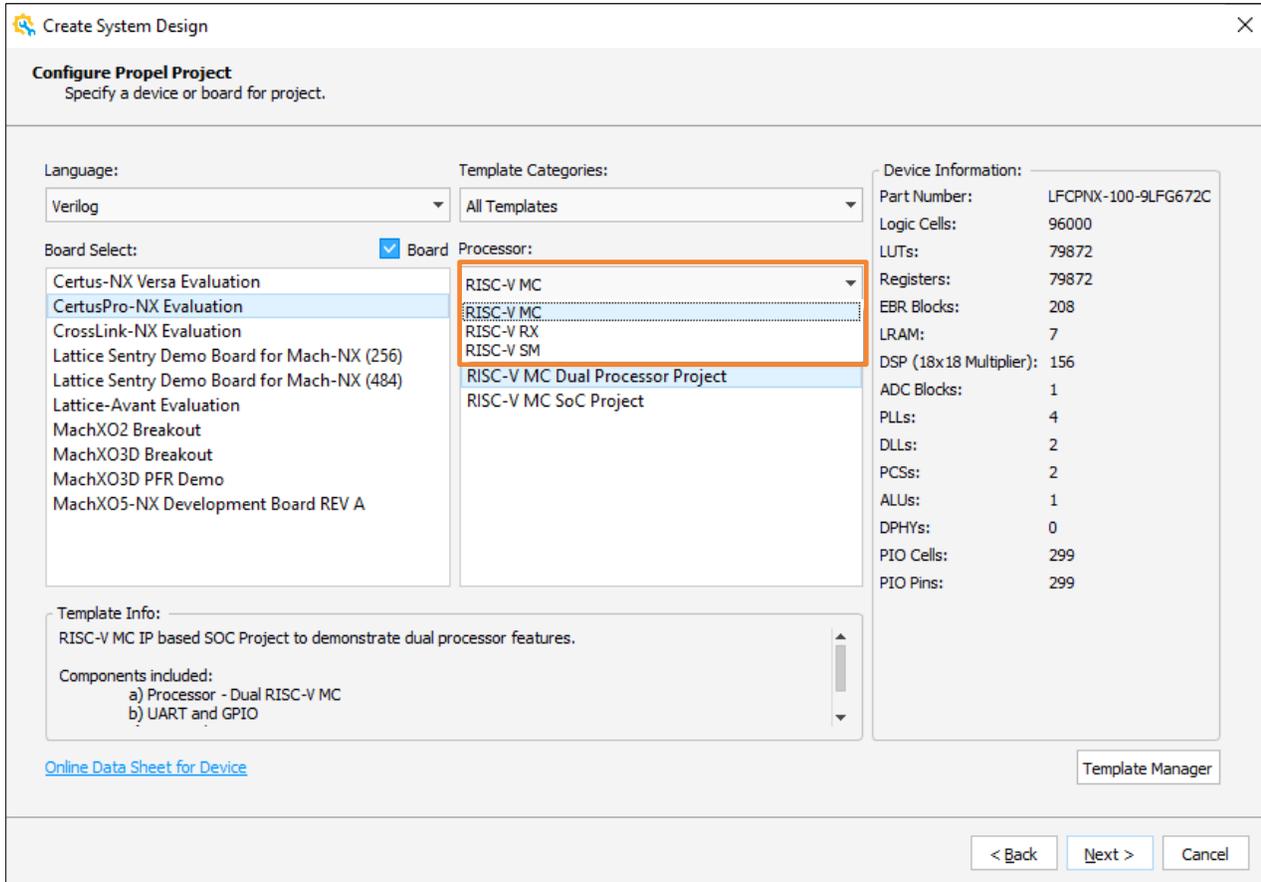


Figure 2.6. Specify a Board for Template SoC Project (2)

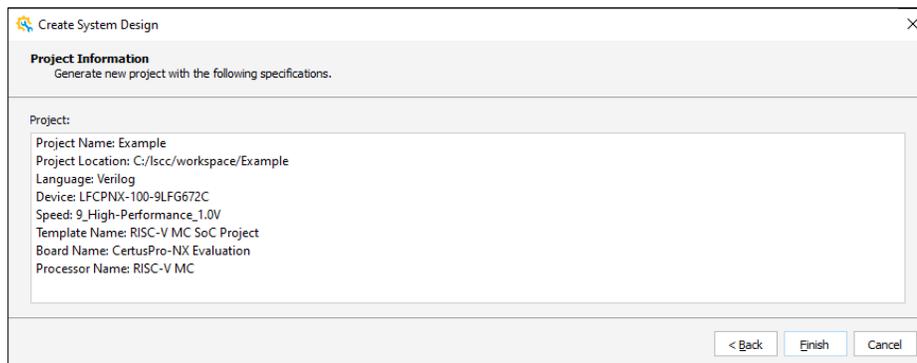
**Notes:**

- You can choose VHDL/Verilog in **Language** field. 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.
  - Custom templates (User Templates) can also be selected from this page, from the drop-down menu of **Template Categories**.
7. (Optional) About the Processor section (Figure 2.7):
- 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 2.7. Specify a Board for Template SoC Project (3)**

- Click **Next**. The Project Information wizard opens, check and confirm the project information (Figure 2.8).



**Figure 2.8. Project Information Wizard**

- Click **Finish**. Propel Builder GUI opens (Figure 2.9).

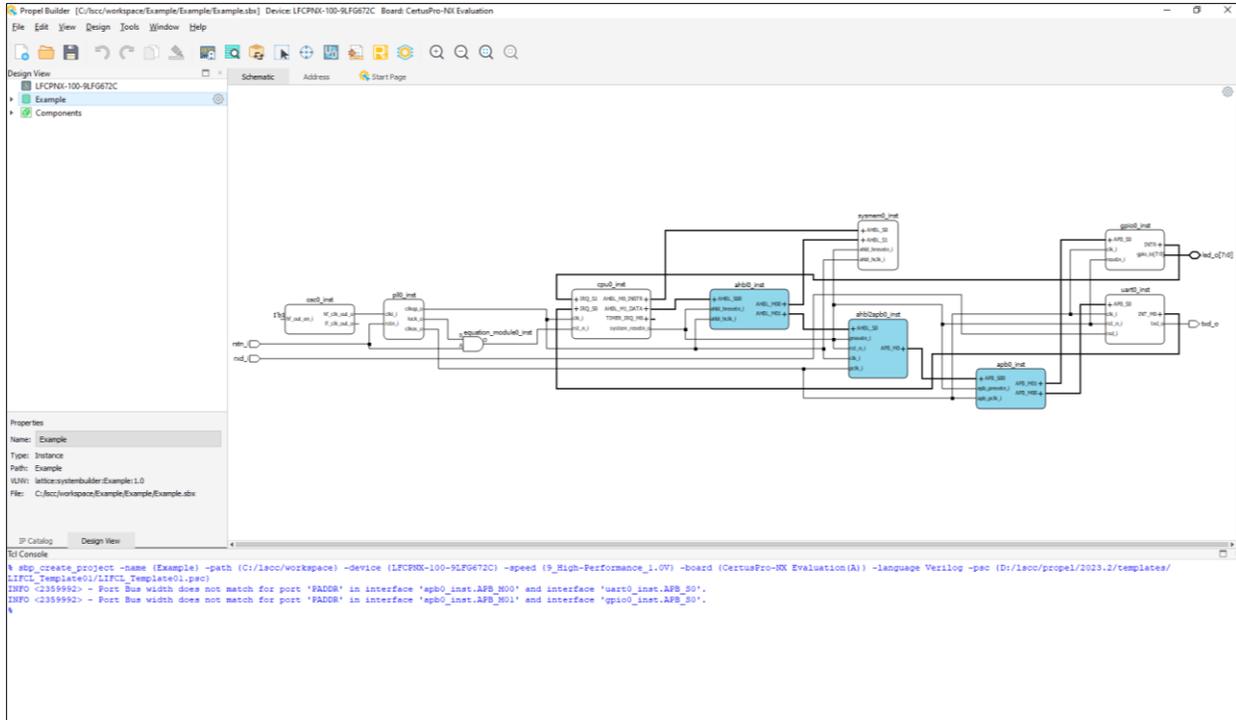


Figure 2.9. Propel Builder GUI showing Template SoC Project

10. (Optional) If you want to see all templates while creating a SoC project, you can use **New SoC Design**.

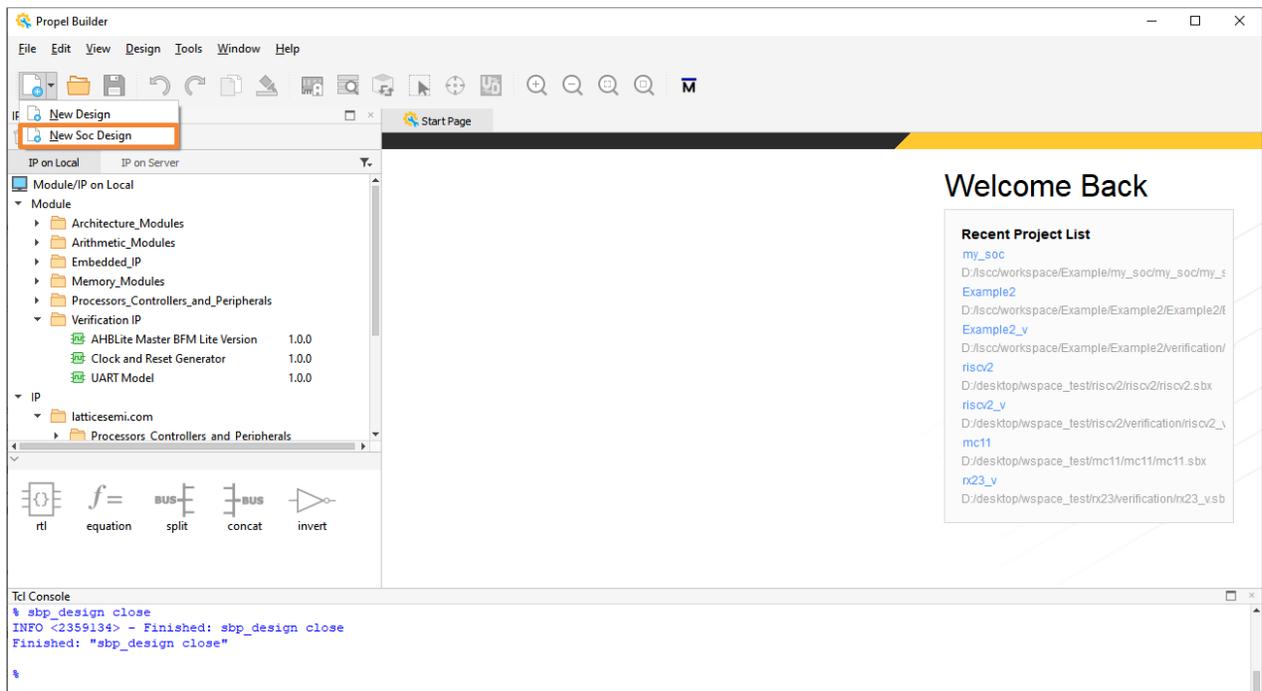
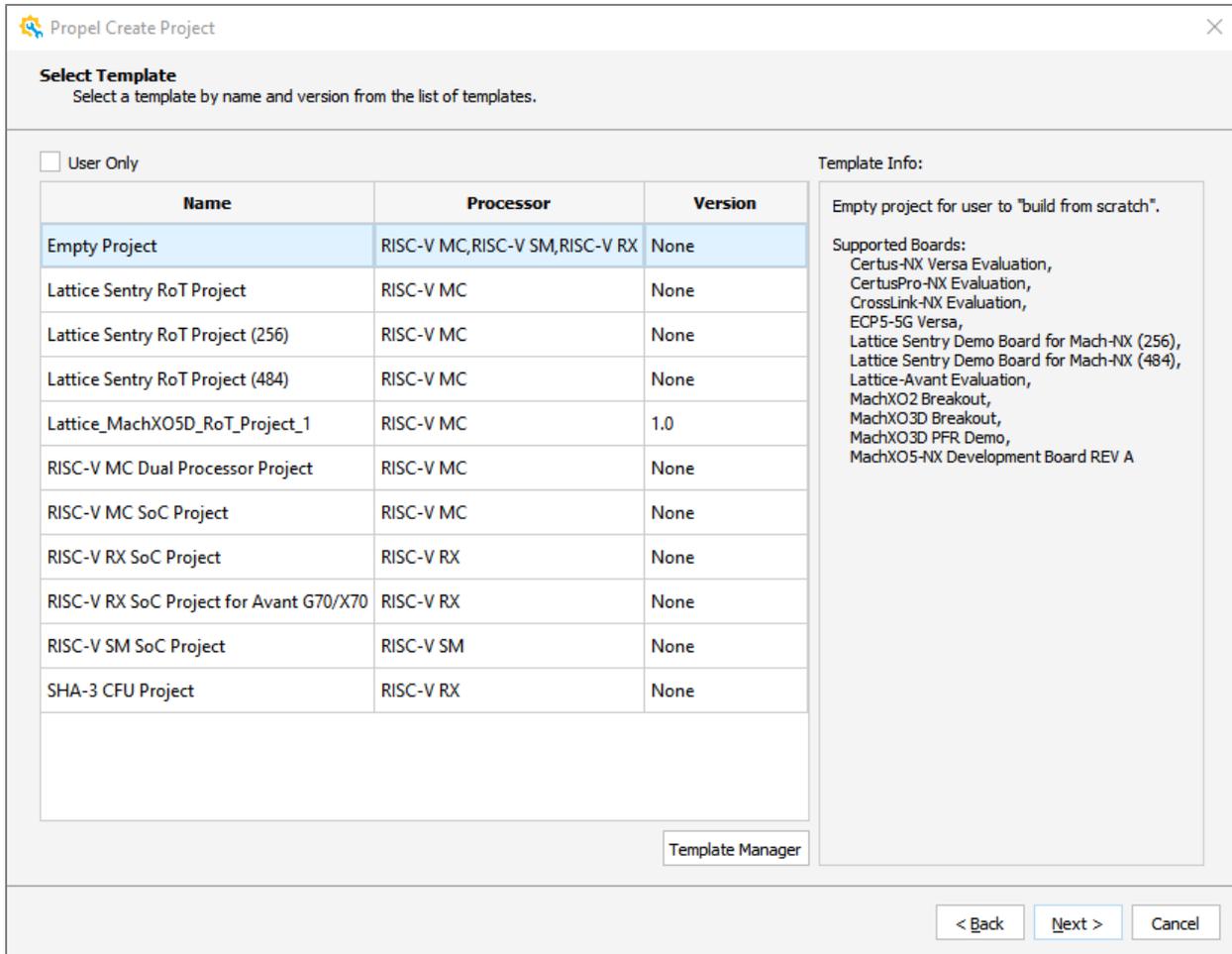


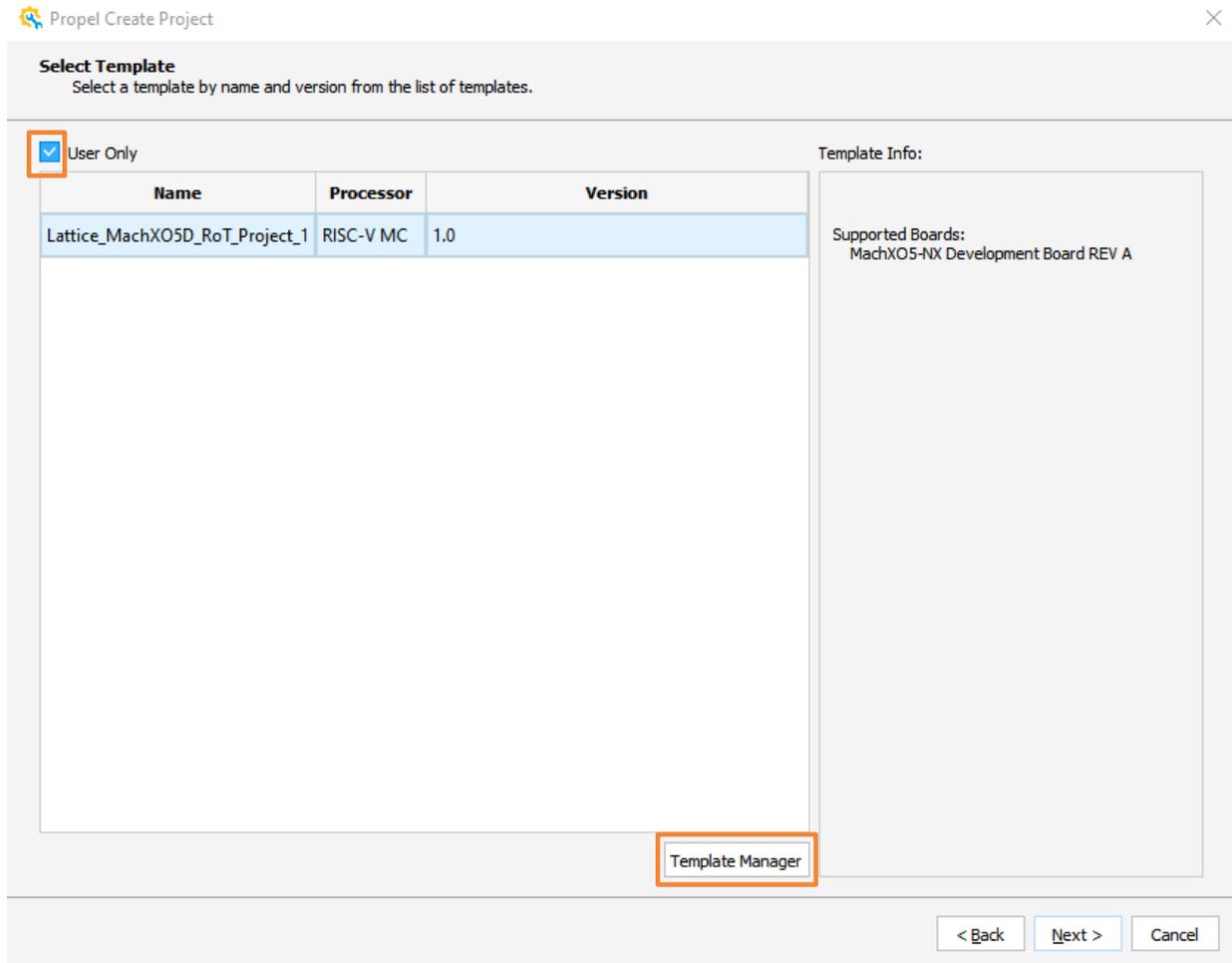
Figure 2.10. New SoC Project in Another Way

You can get detailed information on Template in this flow (Figure 2.11).



**Figure 2.11. New SoC Project – Select Template**

Click **User Only** to see user custom template (Figure 2.12). Refer to the [Define Custom Template](#) section for how to use **Template Manager**.



**Figure 2.12. New SoC Project – Select Custom Template**

You can choose Device/Board in this flow as well (Figure 2.13).

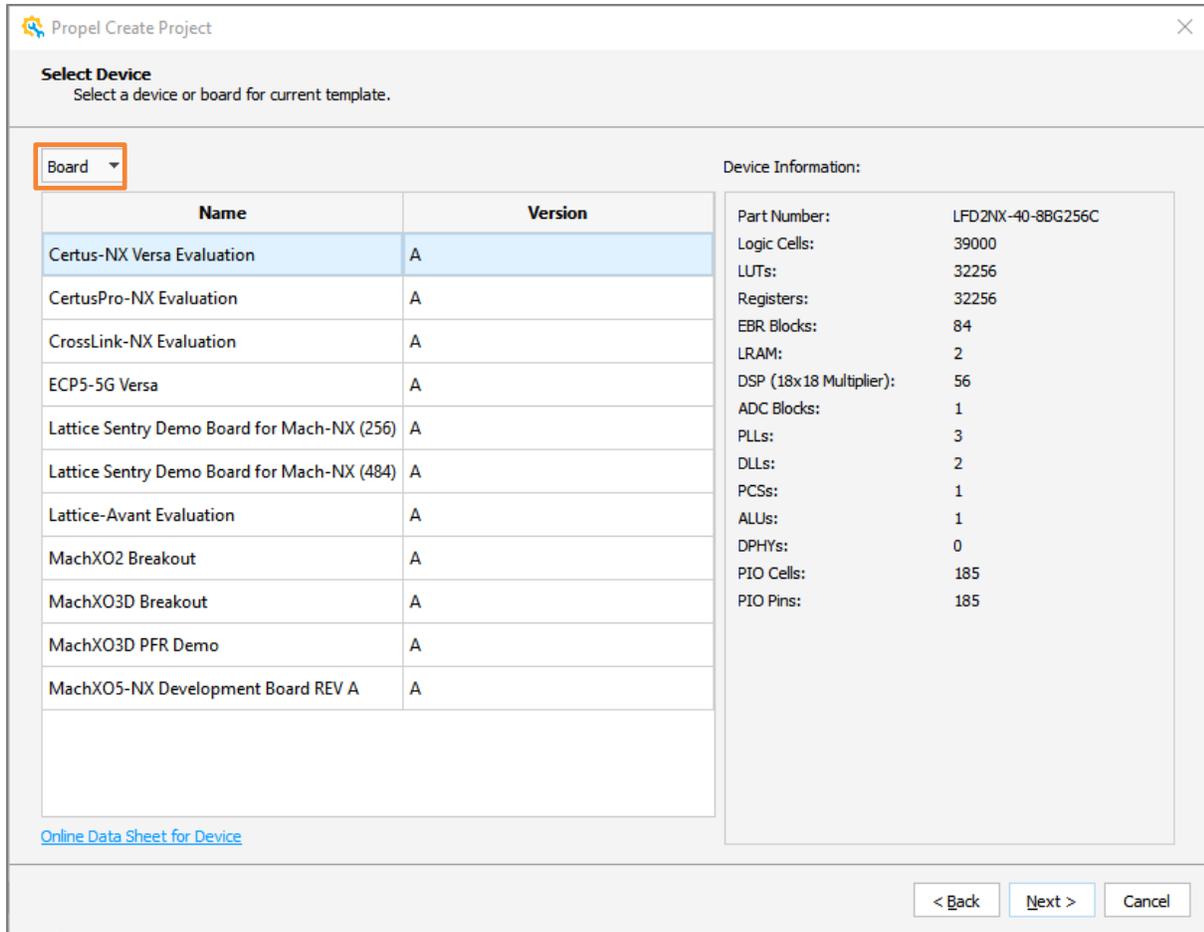


Figure 2.13. New SoC Project – Select Device

## 2.2.2. Opening an Existing SoC Project

- To open an existing SoC project, choose **File** >  **Open Design** from Propel Builder GUI menu or click **Open Design** icon  from Propel Builder Toolbar (Figure 2.14).

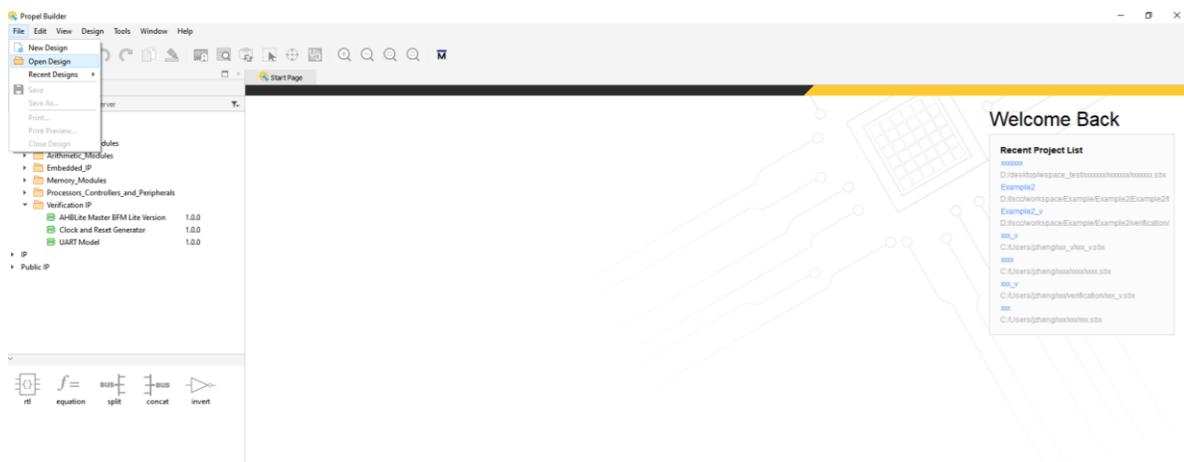


Figure 2.14. Open Design

2. Browse to find the default project workspace folder or your own project workspace folder. Choose the sbx file, such as HelloWorld.sbx (Figure 2.15).

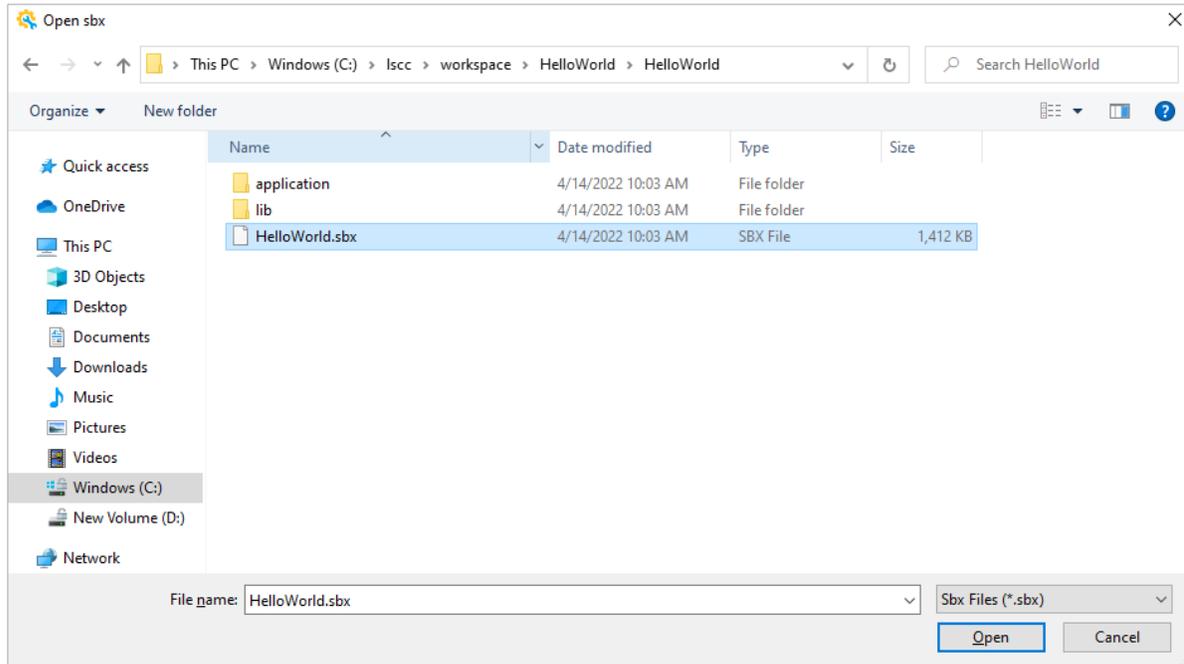


Figure 2.15. Open Existing Project

3. Click **Open**. The Builder GUI shows the SoC design.
4. You can also use **File > Recent Designs** from Propel Builder Menu (Figure 2.16) to quickly open a recently closed project.

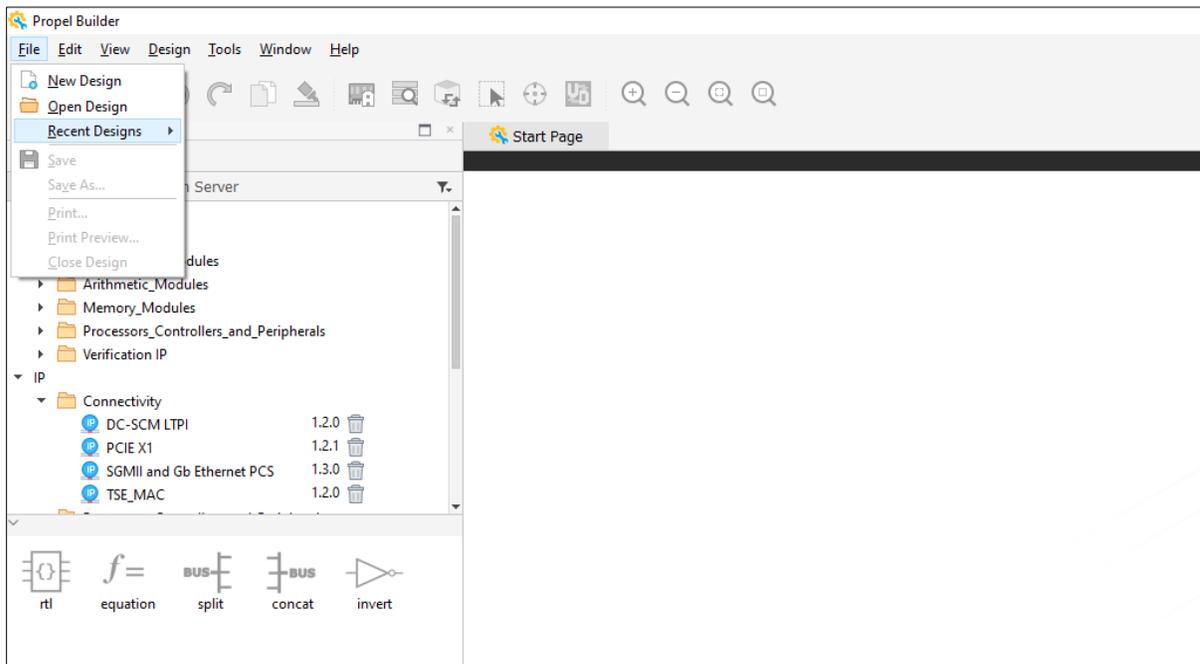


Figure 2.16. Open Recent Project

### 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. You can also click the **IP on Server** tab to find and download more modules for specialized use (Figure 2.17).

(Optional) From the Propel Builder GUI **IP Catalog** area (Figure 2.17), 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 or the current device. You can hover the mouse over the to see the warning message details about why the IP is not supported (Figure 2.18). If the IP is already installed, “Installed” is shown (Figure 2.17).

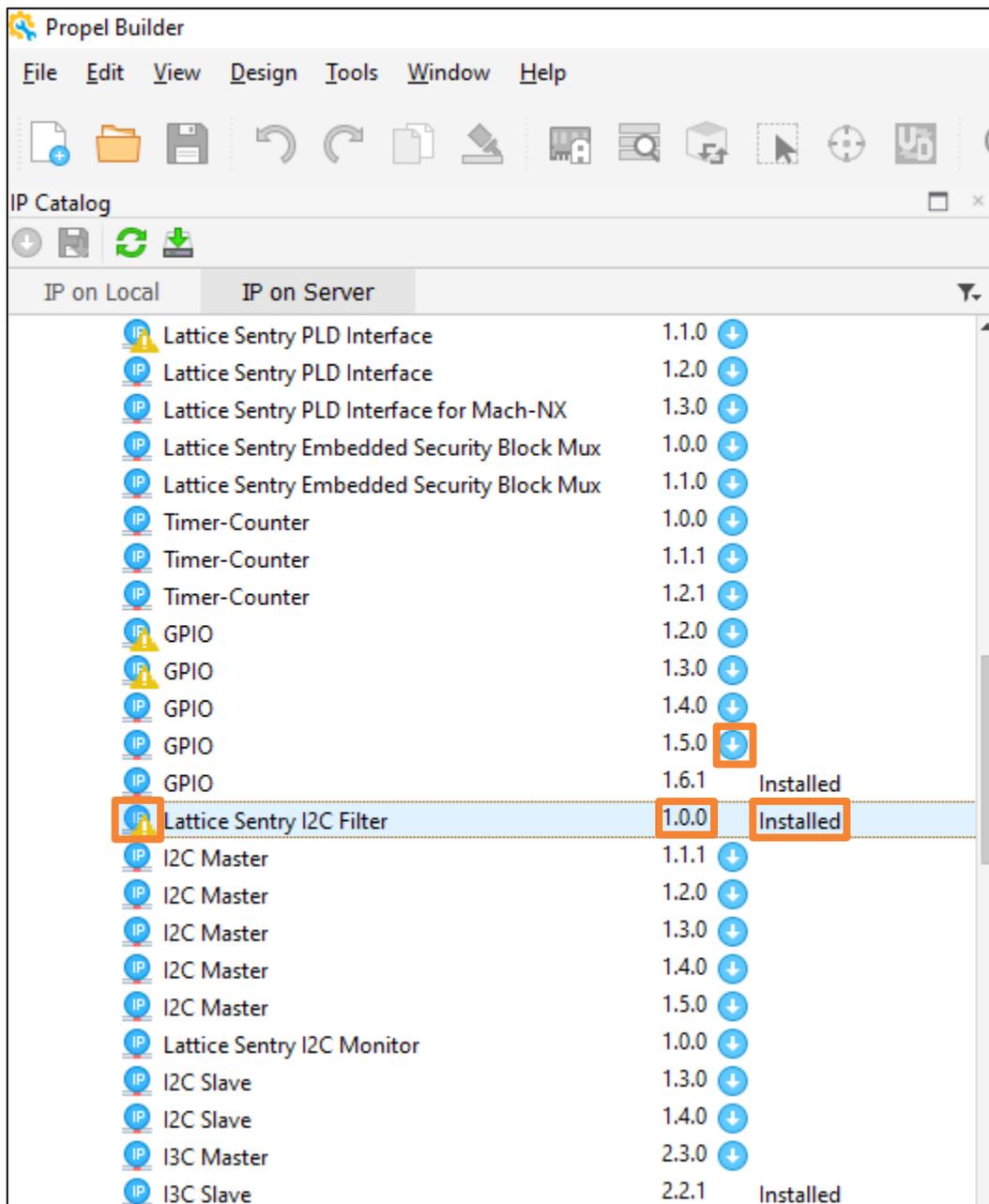


Figure 2.17. IP on Server

IP on Local	IP on Server
CNN Coprocessor Unit	1.2.0
Lattice Sentry PLD Interface	1.0.0
Lattice Sentry PLD Interface	1.1.0
Lattice Sentry PLD Interface	1.2.0
Lattice Sentry PLD Interface for Mach-NX	1.3.0
Lattice Sentry Embedded Security Block Mux	1.0.0
Lattice Sentry Embedded Security Block Mux	1.1.0
Timer-Counter	1.0.0
Timer-Counter	1.1.1
Timer-Counter	1.2.2
GPIO	1.2.0
<b>GPIO</b>	1.3.0
GPIO	1.4.0
GPIO	1.5.0
This IP can only be used in Propel Software	1.6.1 <b>Installed</b>
version 1.1 and lower. Click  to install this IP.	1.0.0 <b>Installed</b>
I2C Master	1.1.1
I2C Master	1.2.0
I2C Master	1.3.0
I2C Master	1.4.0
I2C Master	1.5.0
Lattice Sentry I2C Monitor	1.0.0
I2C Slave	1.3.0
I2C Slave	1.4.0
I3C_Controller	3.0.0
I3C Master	2.3.0
I3C Slave	2.2.1 <b>Installed</b>
I3C Target	3.0.0
Internal Flash Controller for MachXO5-NX	1.0.2

Figure 2.18. Hover Mouse over the Icon

You can use the filter and search function in IP catalog to find the IP you desire (Figure 2.19).

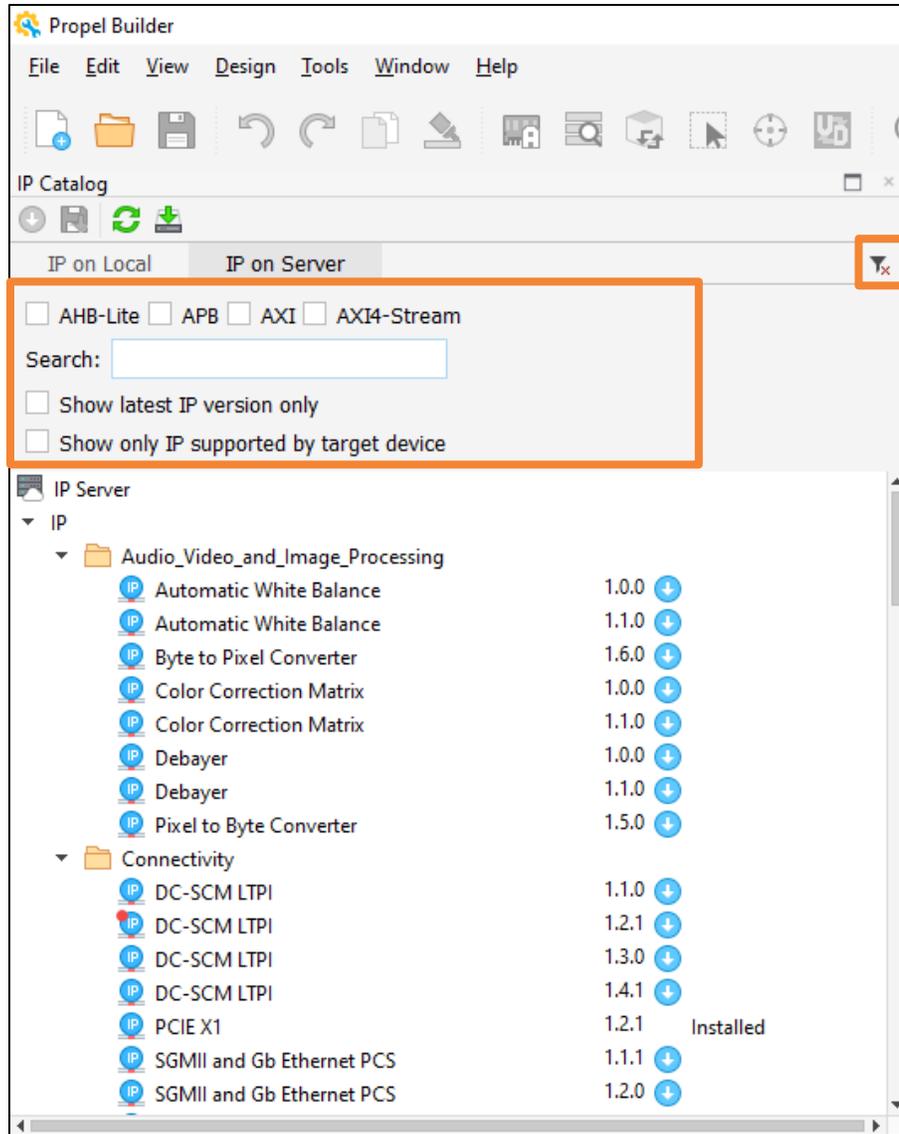


Figure 2.19. IP Filter and Search

1. Being installed successfully, the new IP is shown in the **IP on Local** tab (Figure 2.20).

**Note:** Only IP that are compatible with the current device can appear under **IP on Local**.

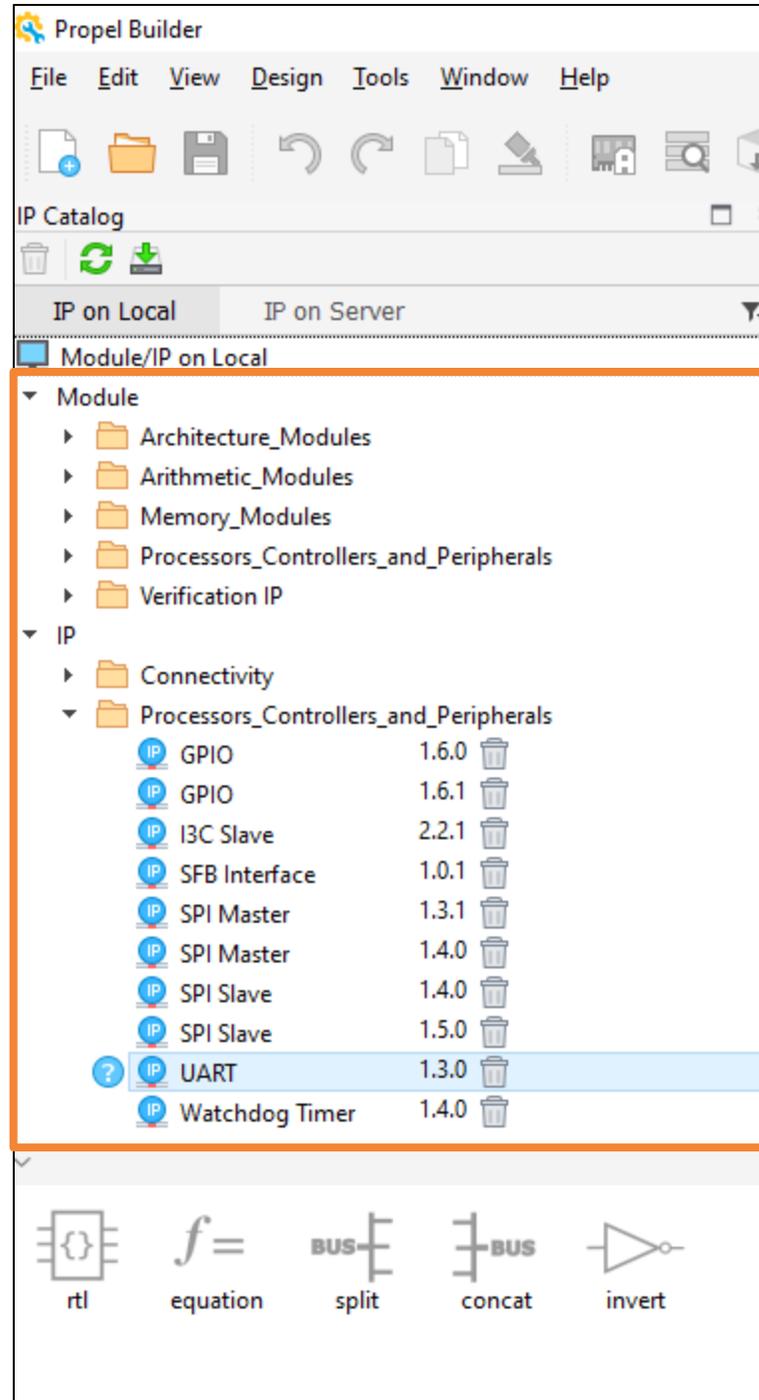
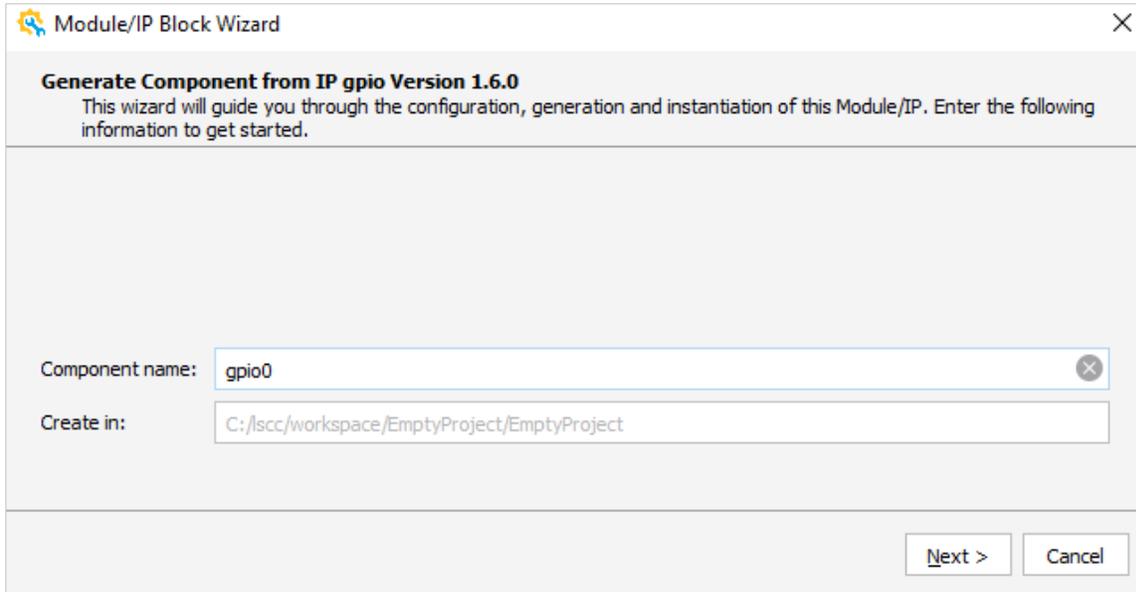


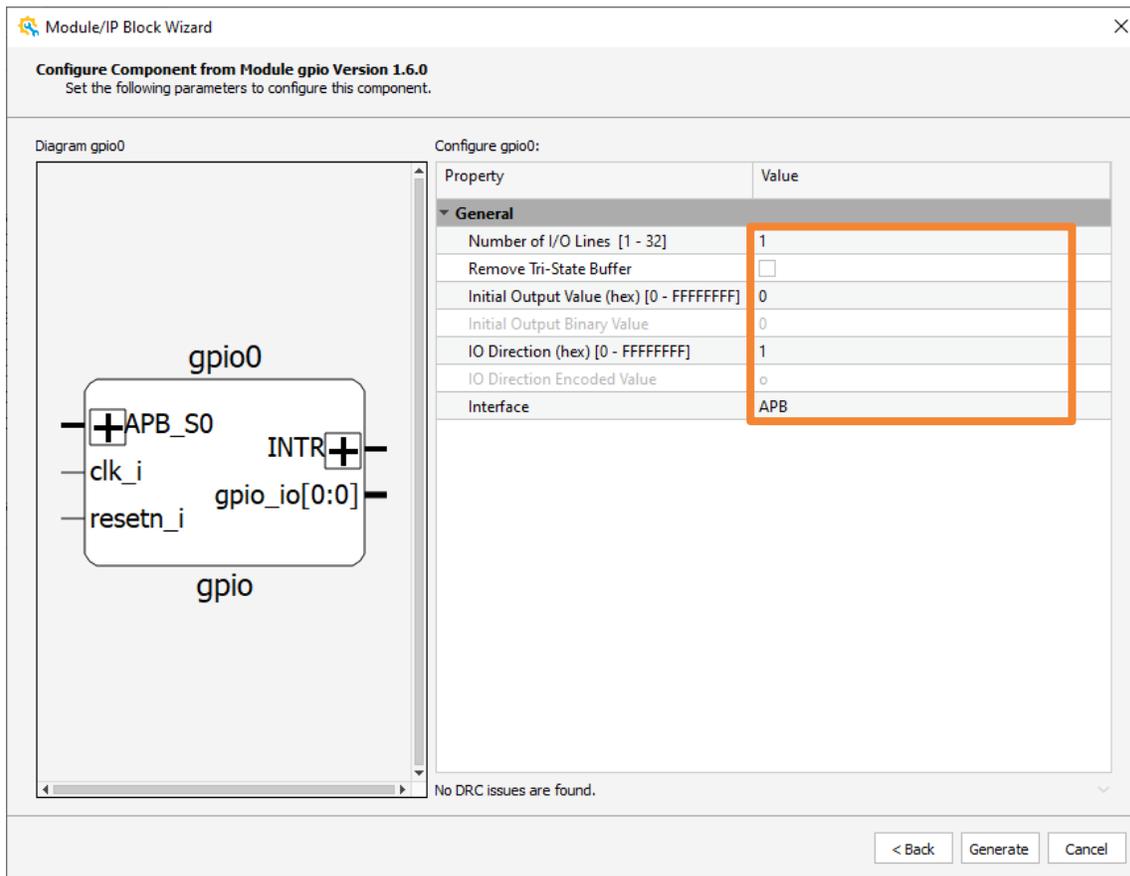
Figure 2.20. IP Catalog

- 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 2.21).



**Figure 2.21. 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 2.22).



**Figure 2.22. Module/IP Block Wizard – Configure Component from IP Gpio Version 1.6.0**

- (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 Propel Design section for more info on how to do this.

- Click **Generate**. Module/IP Block wizard shows the Check Generated Result page (Figure 2.23).

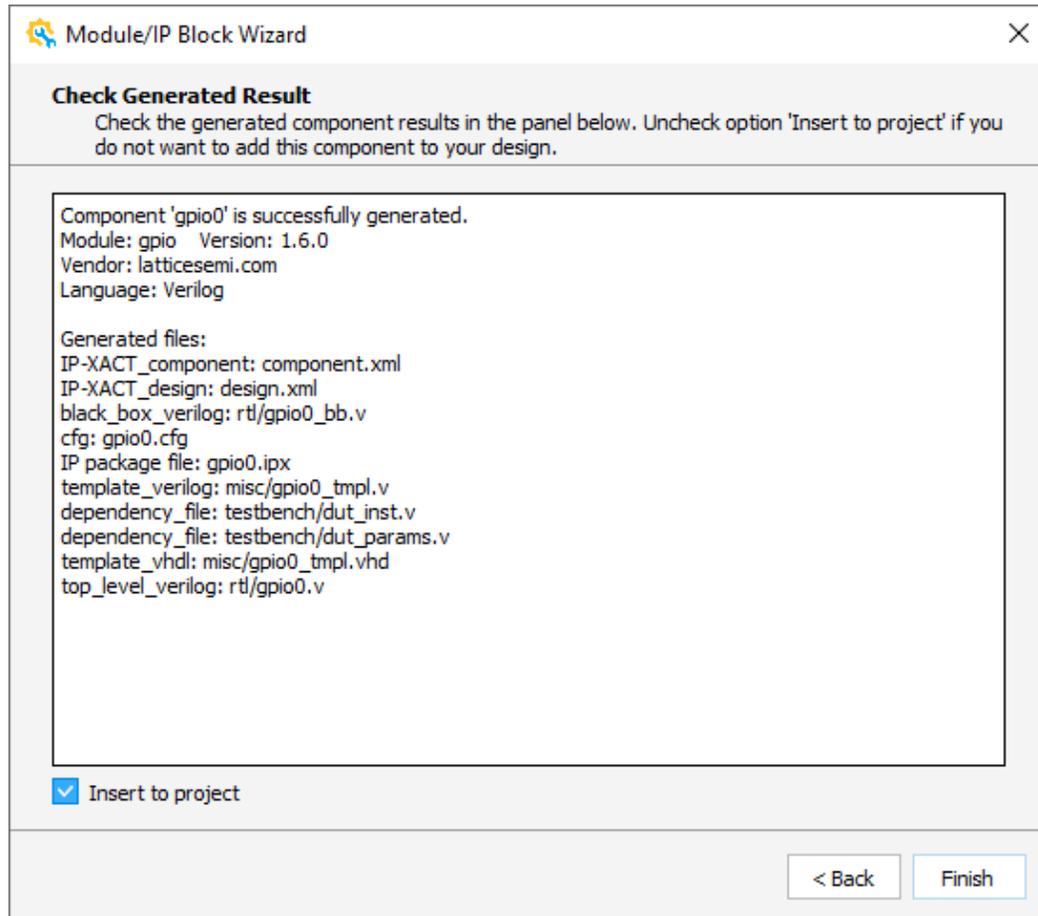


Figure 2.23. Module/IP Block Wizard – Check Generated Result

- Select **Insert to project** (Figure 2.23) 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.
- After selecting **Insert to project**, click **Finish** (Figure 2.23). The **Define Instance** dialog box opens (Figure 2.24).

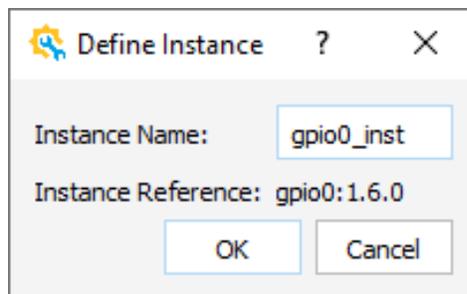


Figure 2.24. Design Instance Dialog Box

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 2.25).

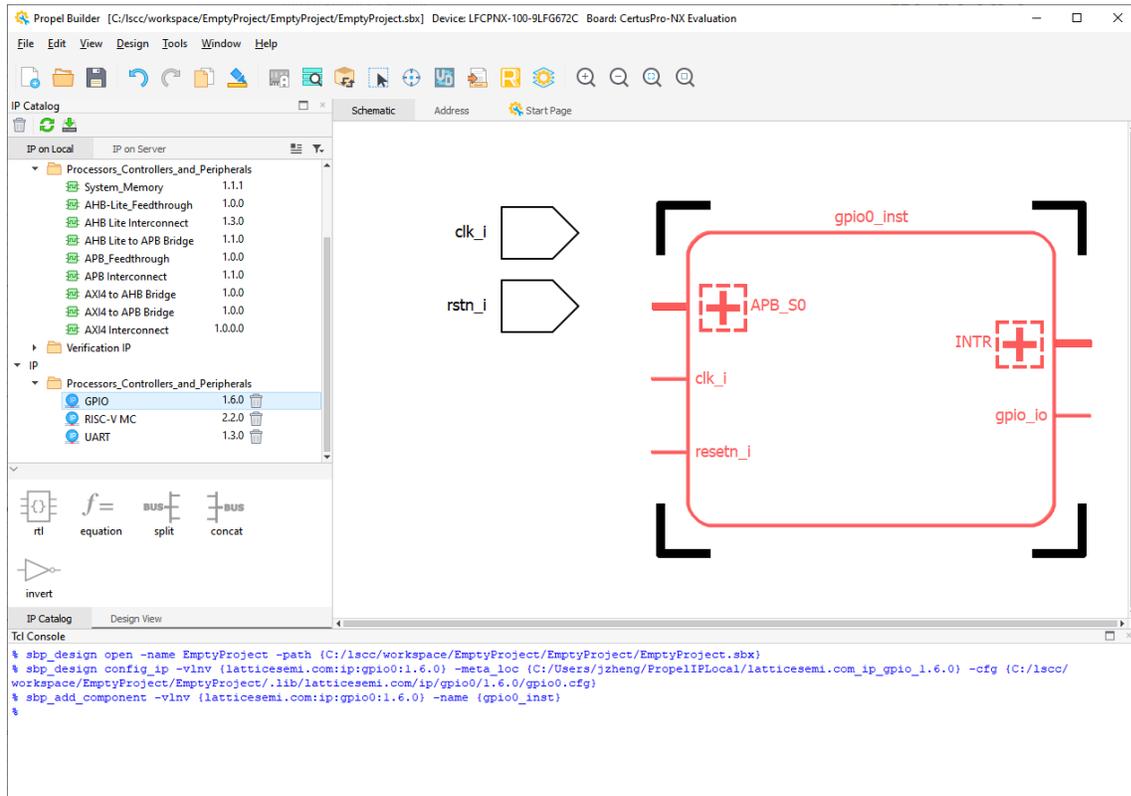


Figure 2.25. Propeller Builder Schematic View Shows the Module Instance

10. (Optional) To re-generate this IP, double click on the instance. A Module/IP Block wizard pops up (Figure 2.26).

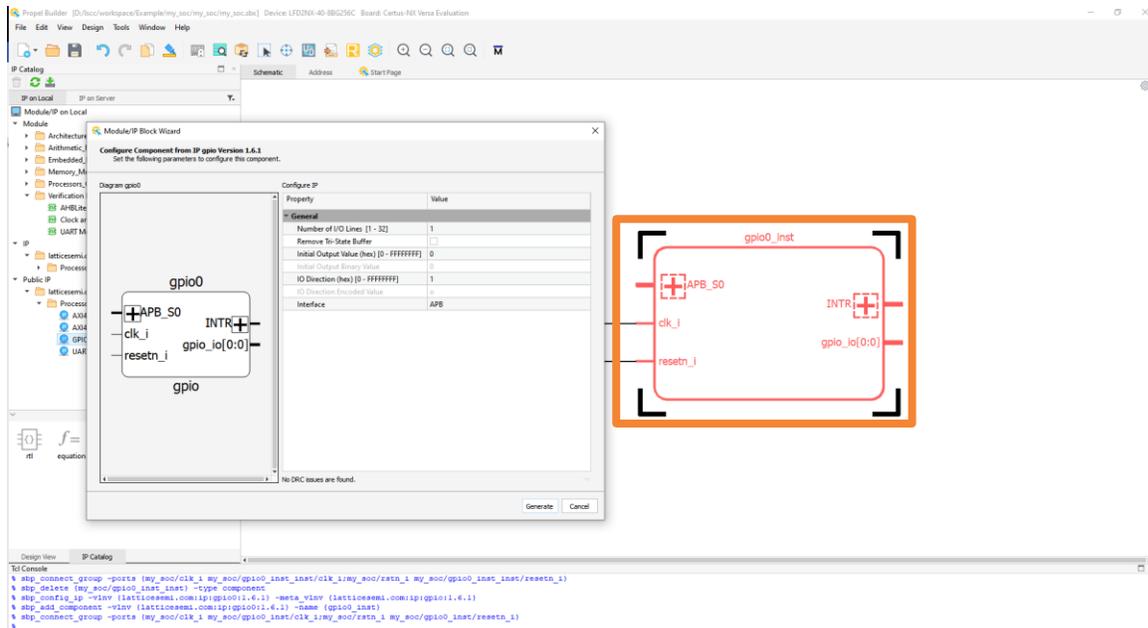


Figure 2.26. Regenerate Module/IP

- (Optional) Propel Builder also supports to update all IPs. Choose **Edit > Upgrade ALL Ips**. Wait a few seconds, depending on how sophisticated the design is, to see the result (Figure 2.27).

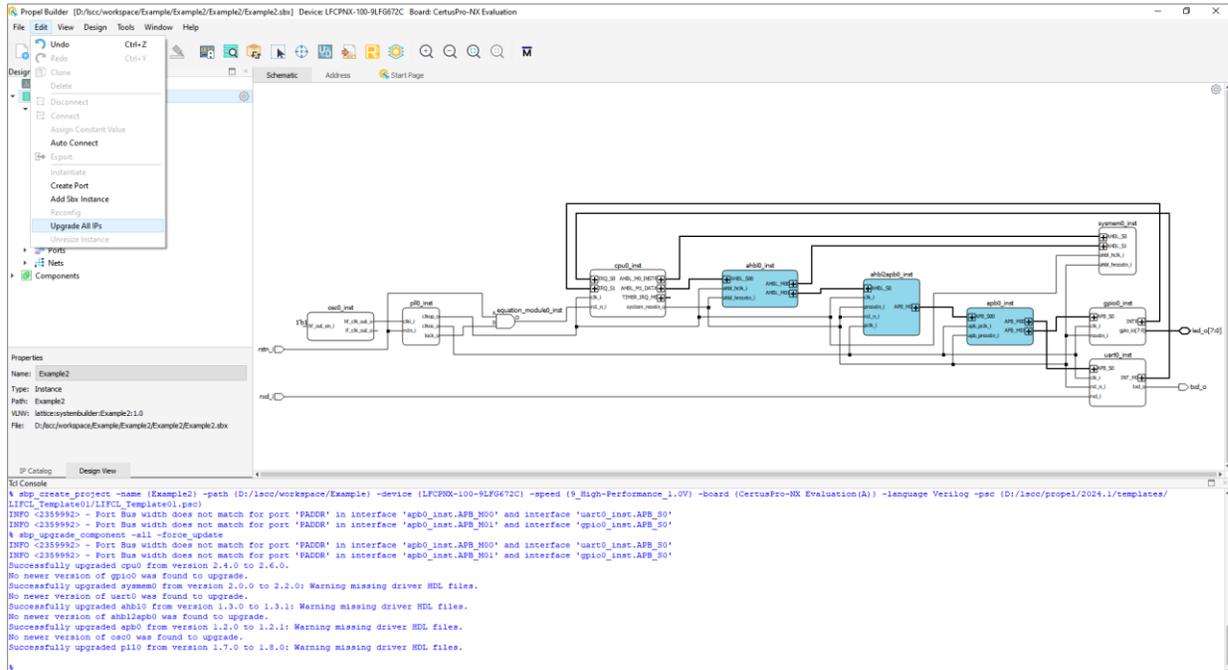


Figure 2.27. Upgrade All IPs

## 2.2.4. Adding Glue Logic

Propel 2024.1 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 2.28) to the Schematic view.

Glue logics be modified afterwards by double click the module.

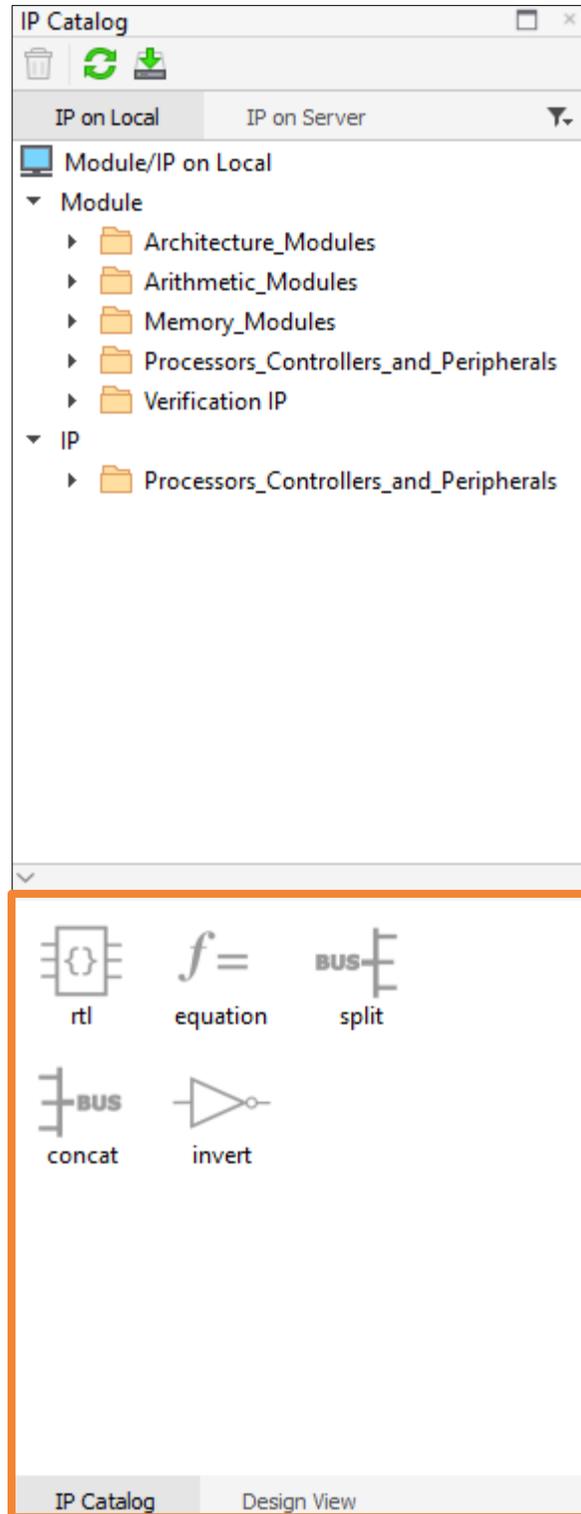
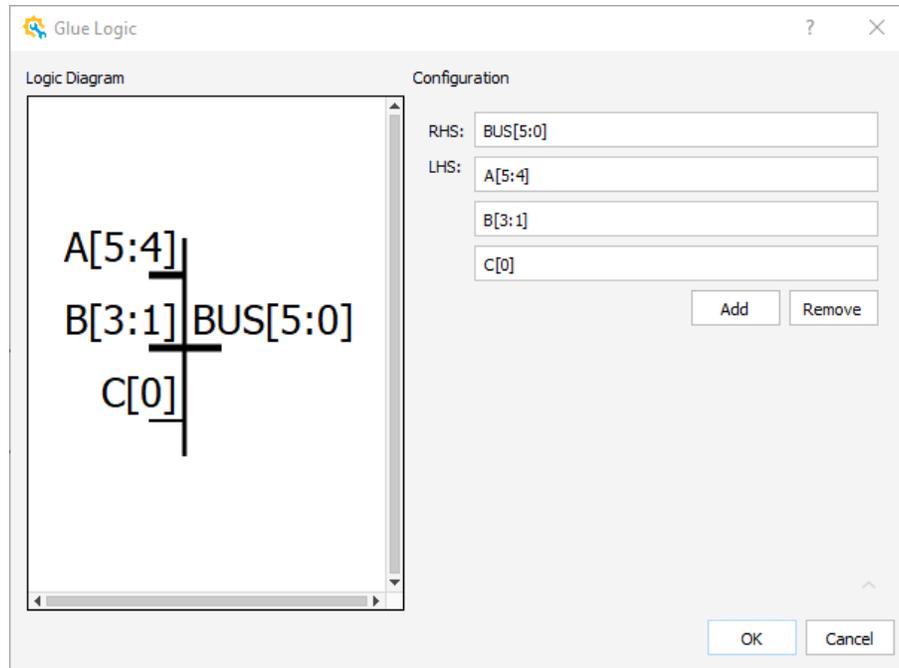


Figure 2.28. Glue Logic Section of the IP Catalog

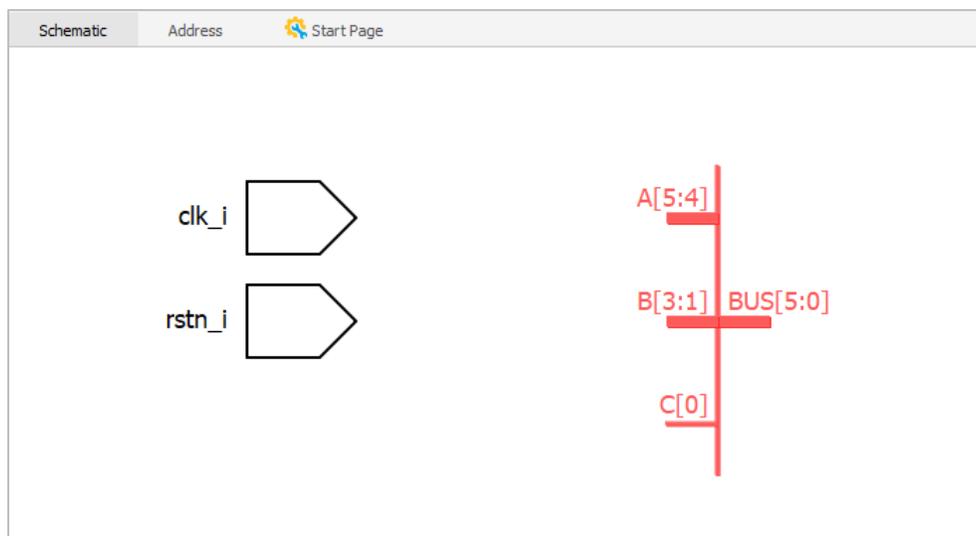
#### 2.2.4.1. Concat

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 2.29) pops up.



**Figure 2.29. 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 2.30](#)). you can drag and connect them with the rest of design.



**Figure 2.30. 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 2.31](#)) pops up.

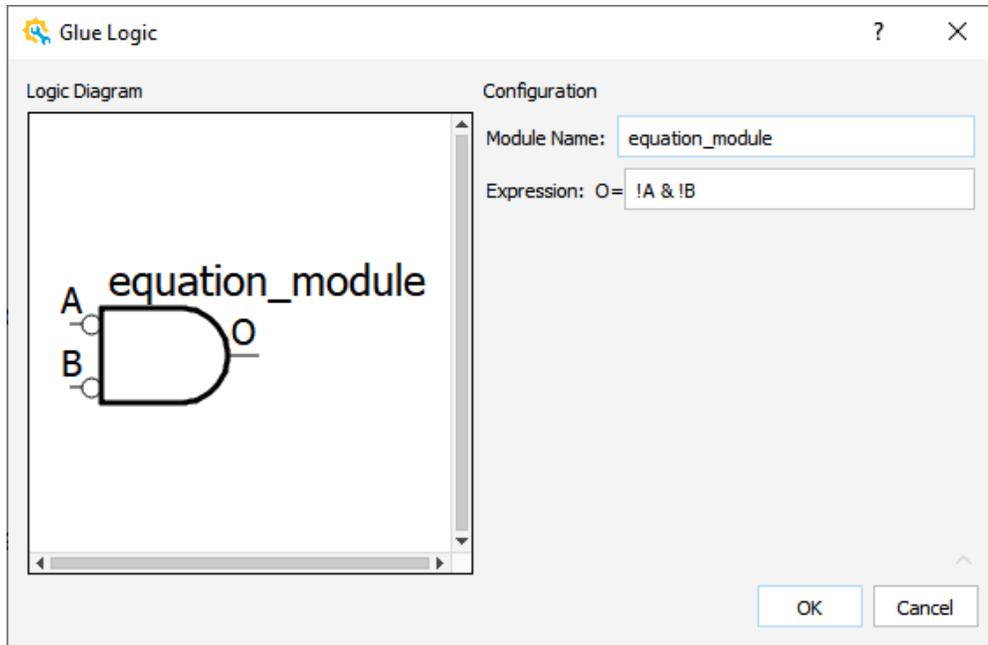


Figure 2.31. Glue Logic Wizard for Equation Module

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 (!), caret (^). 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 2.32).

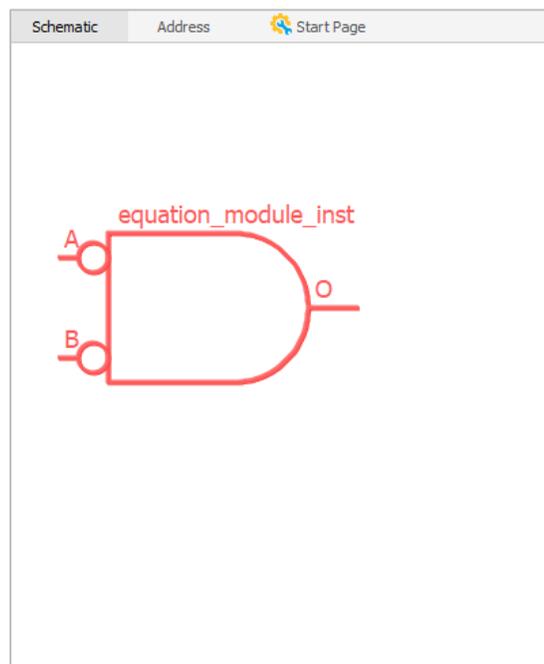


Figure 2.32. Schematic View Shows an Equation Module

### 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 2.33).

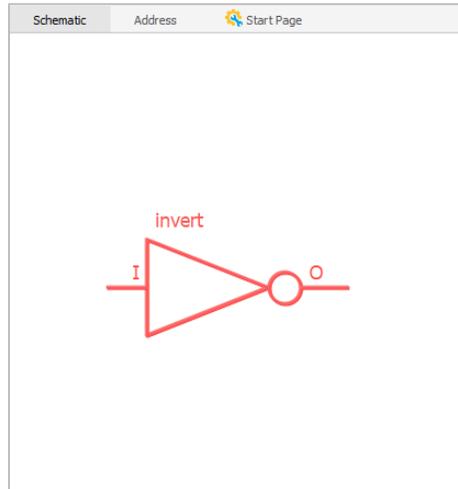


Figure 2.33. Schematic View Shows the Invert Module

### 2.2.4.4. RTL

**Note:** At present, RTL Module only supports Verilog HDL.

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 pops up (Figure 2.34). Glue Logic supports both Verilog HDL and VHDL. You can use the drop-down menu to choose a desired language.

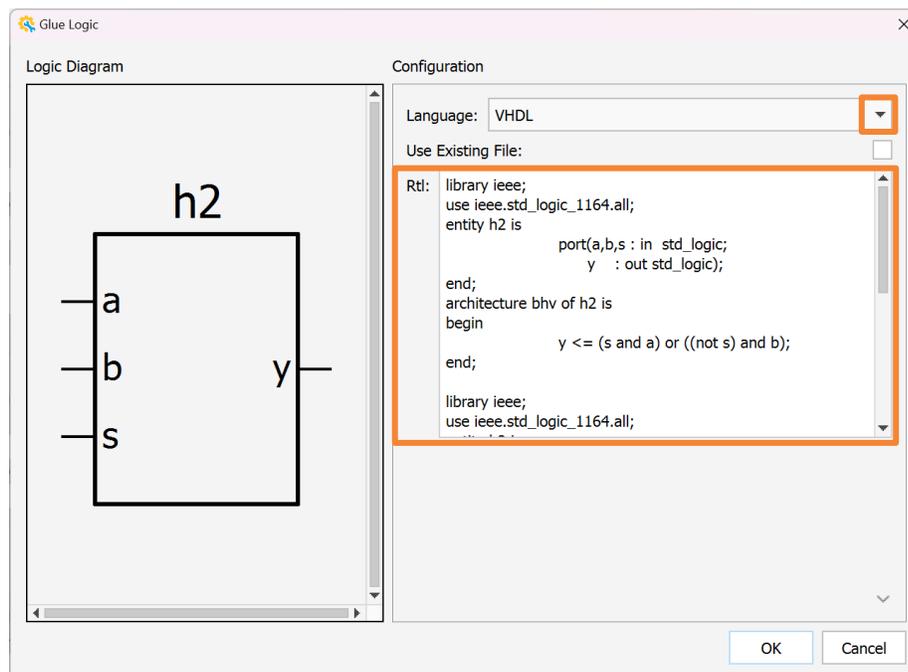


Figure 2.34. Glue Logic Wizard for RTL Module

2. You can edit your own RTL module in the RTL area by entering the RTL module function (Figure 2.34).

- Or, you can use an existing RTL module by checking the **Use Existing File** checkbox. After checking the Use Existing File checkbox (Figure 2.35), browse to find the existing RTL module file path from the **Path** field. 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/`, if not enabled, it is referenced from its current location.

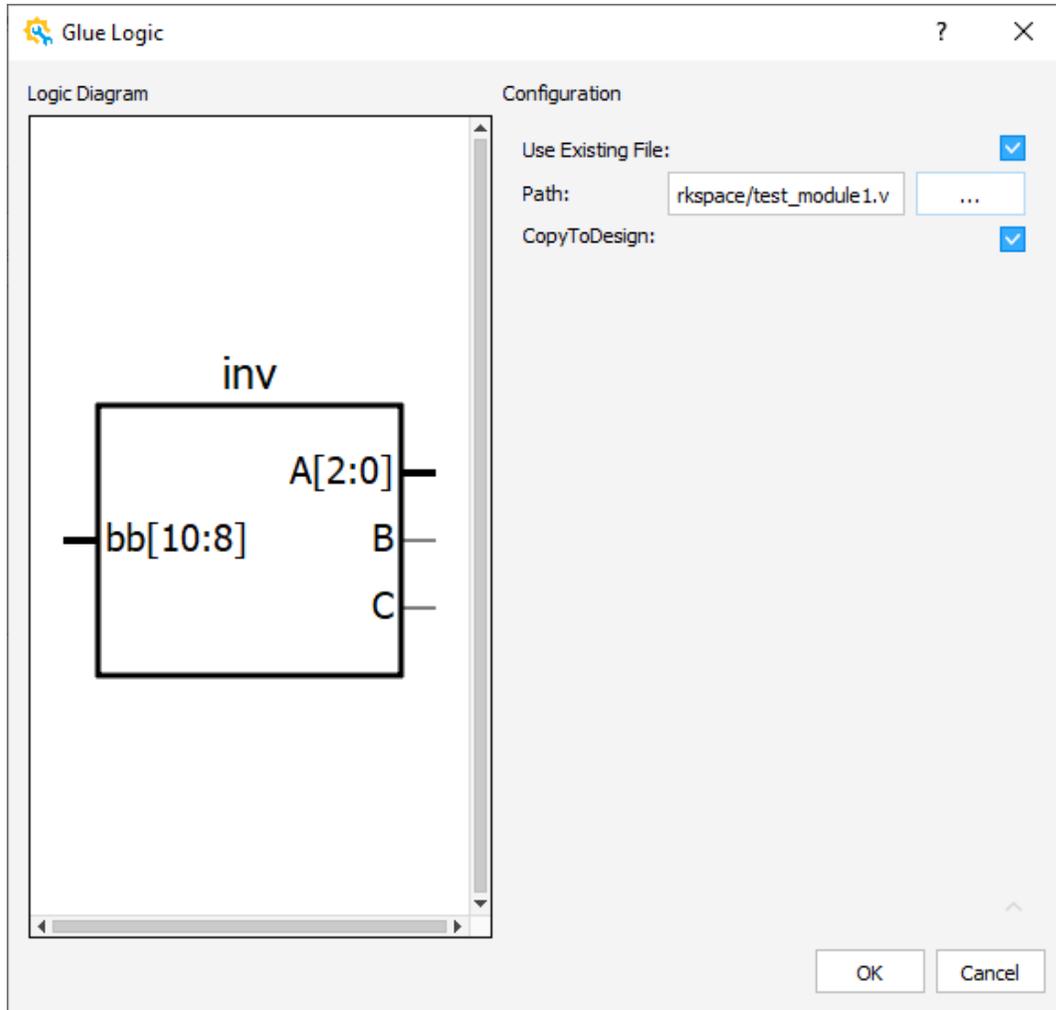


Figure 2.35. Existing RTL Module Configuration

- Click **OK**. The schematic block for the RTL module in red appears in the Schematic view (Figure 2.36).

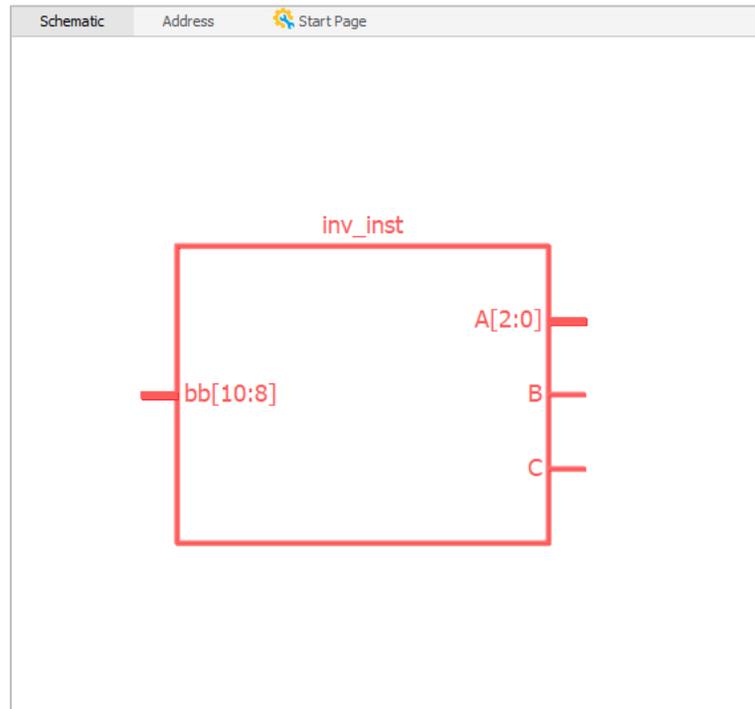


Figure 2.36. Schematic View Shows the Custom RTL Module

#### 2.2.4.5. Split

- 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 2.37) pops up.

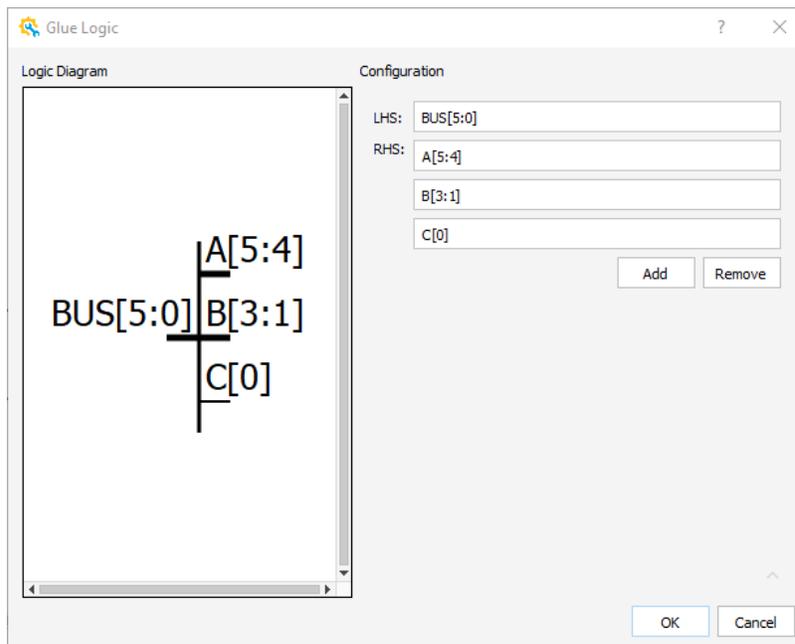
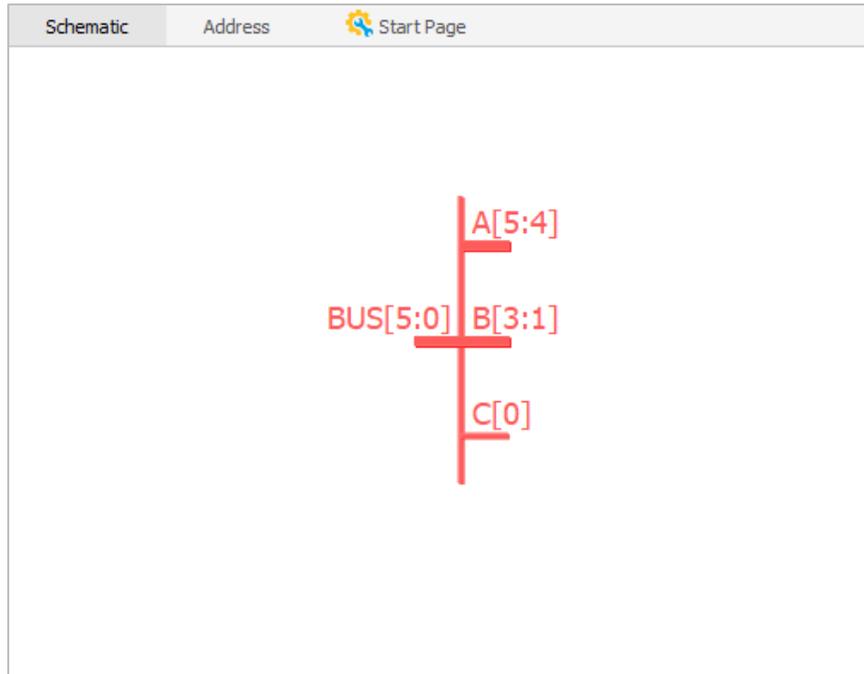


Figure 2.37. Glue Logic Wizard for Split Module

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  button to add an RHS. Click the  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 2.38](#)).



**Figure 2.38. 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 2.39](#). Click the negative sign  to close the expanded bus. The schematic returns to the previous form.

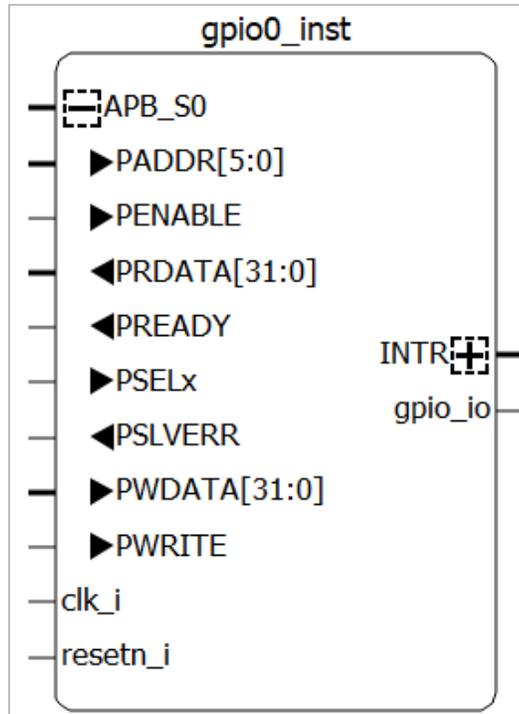


Figure 2.39. Signal List of Modules

### 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 2.40).

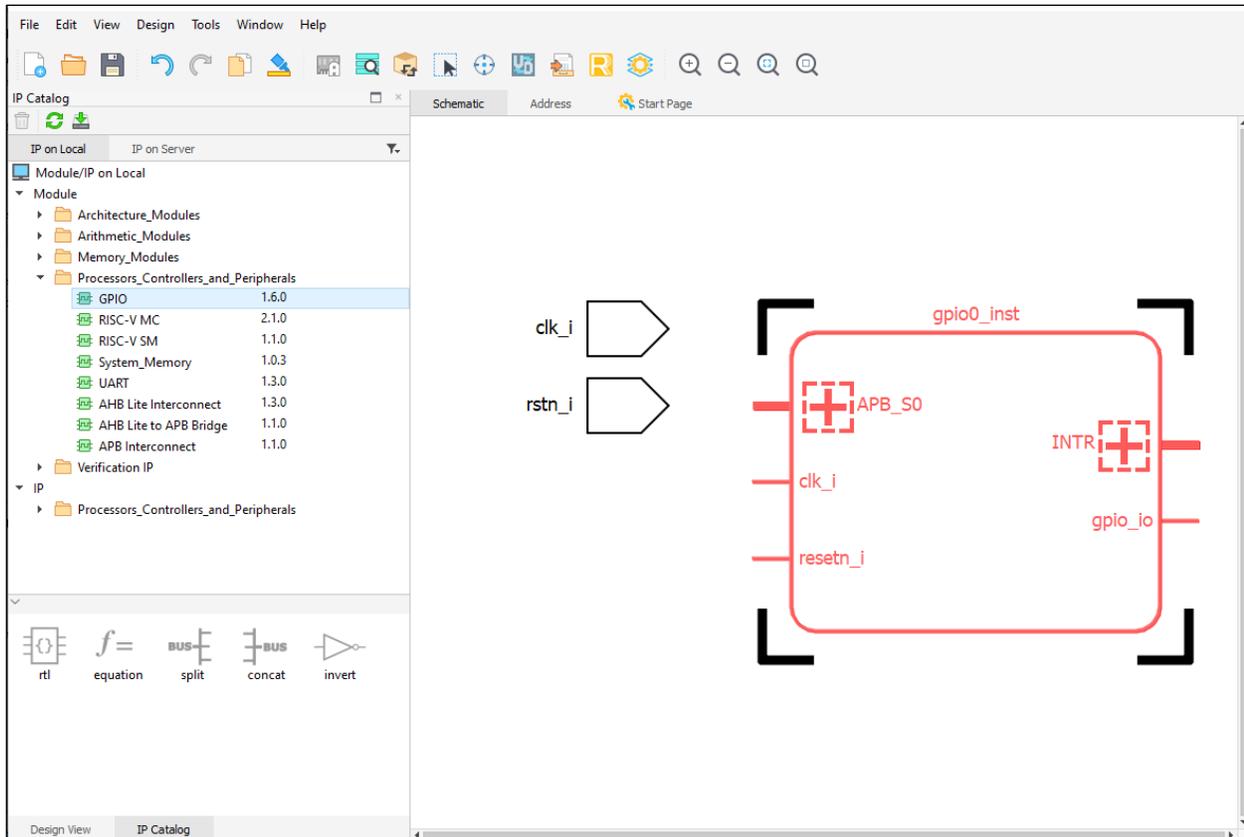


Figure 2.40. Select Object

- 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 2.41).

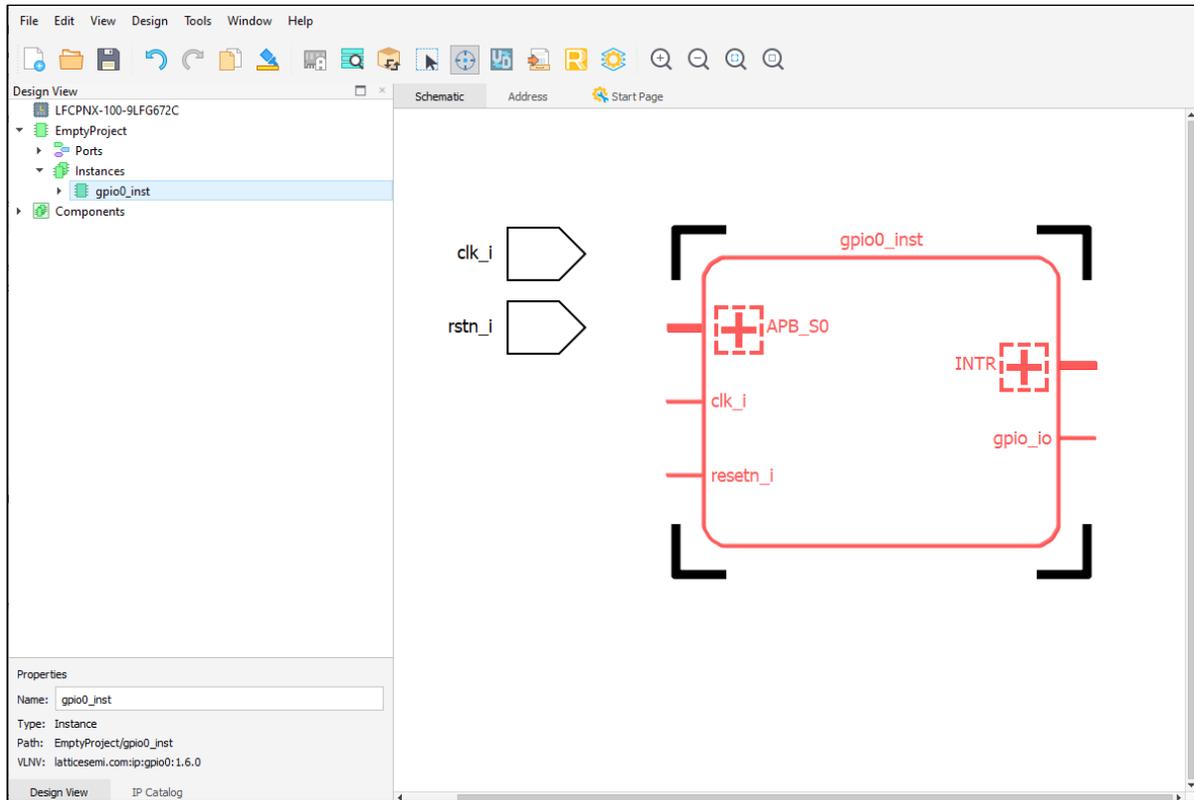


Figure 2.41. Locate Objects

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 2.42). It creates a separate copy of the original module using all the same settings and is essentially its own component which can be modified and managed separately.

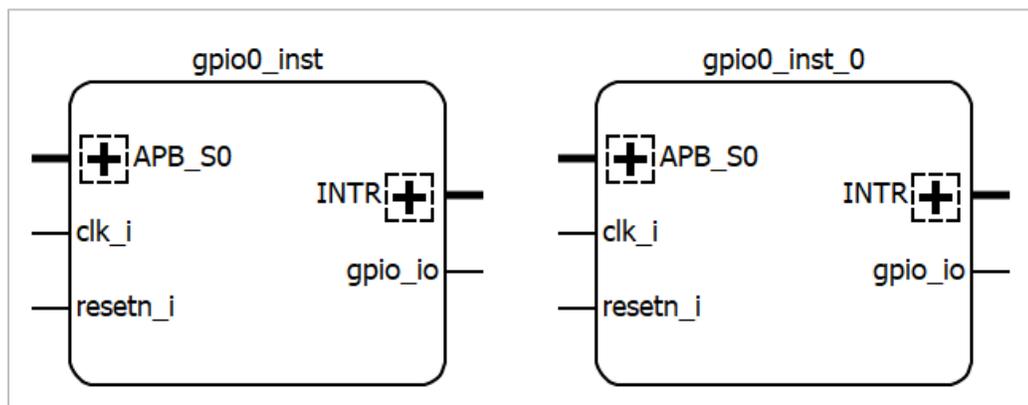


Figure 2.42. Duplicate a Module

### 2.2.5.5. Restore a Deleted Module

1. In the List view of the Design View (Figure 2.43), 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.

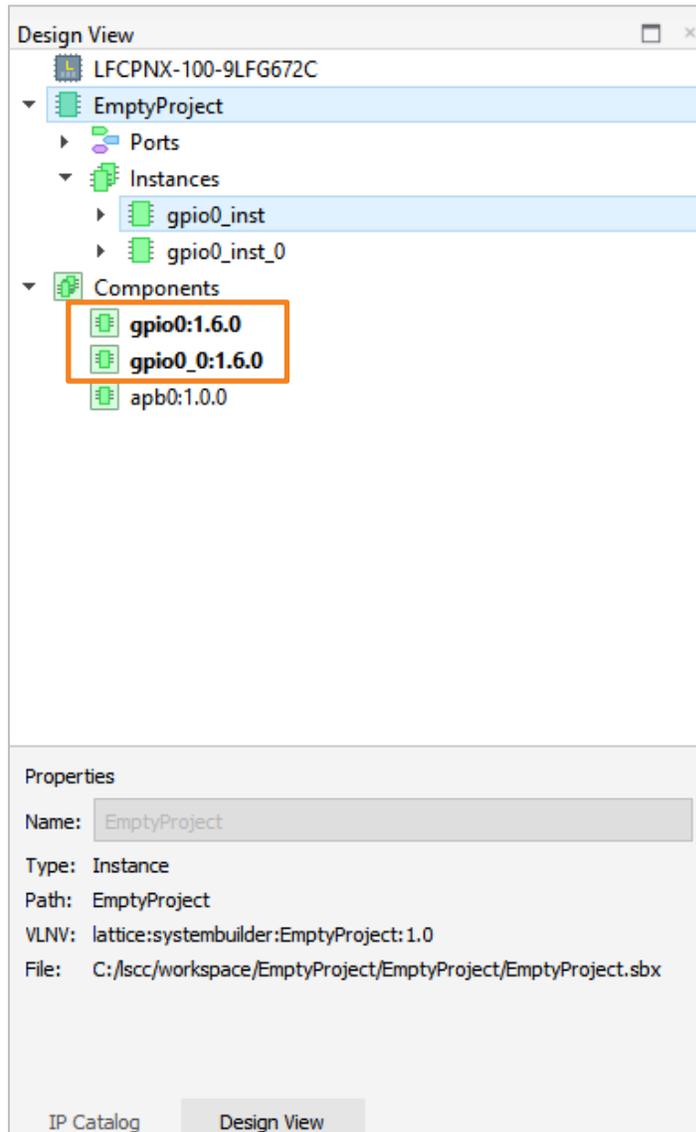


Figure 2.43. Design View

2. Right-click on the component you want to restore, and choose **Instantiate**. The Define Instance dialog box (Figure 2.44) open.

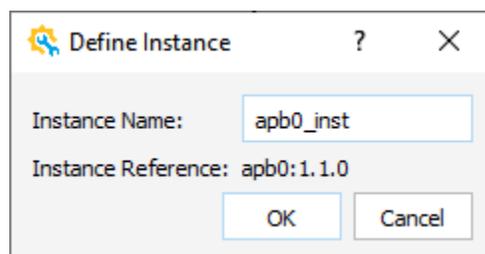
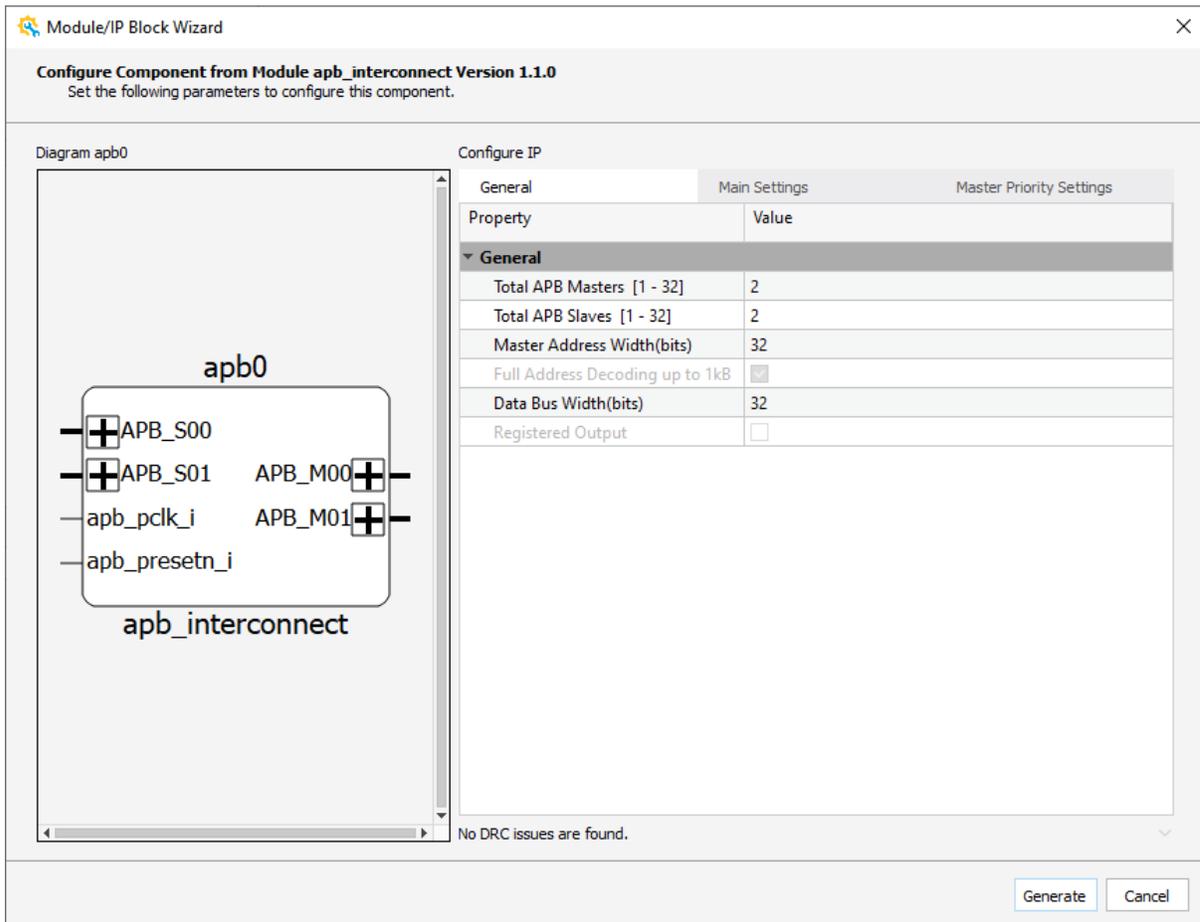


Figure 2.44. Define Instance Dialog Box

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 2.45) opens.



**Figure 2.45. 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 2.46).

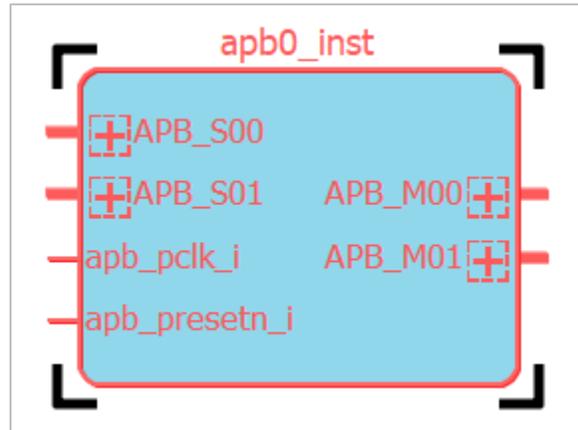


Figure 2.46. Select Module

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.

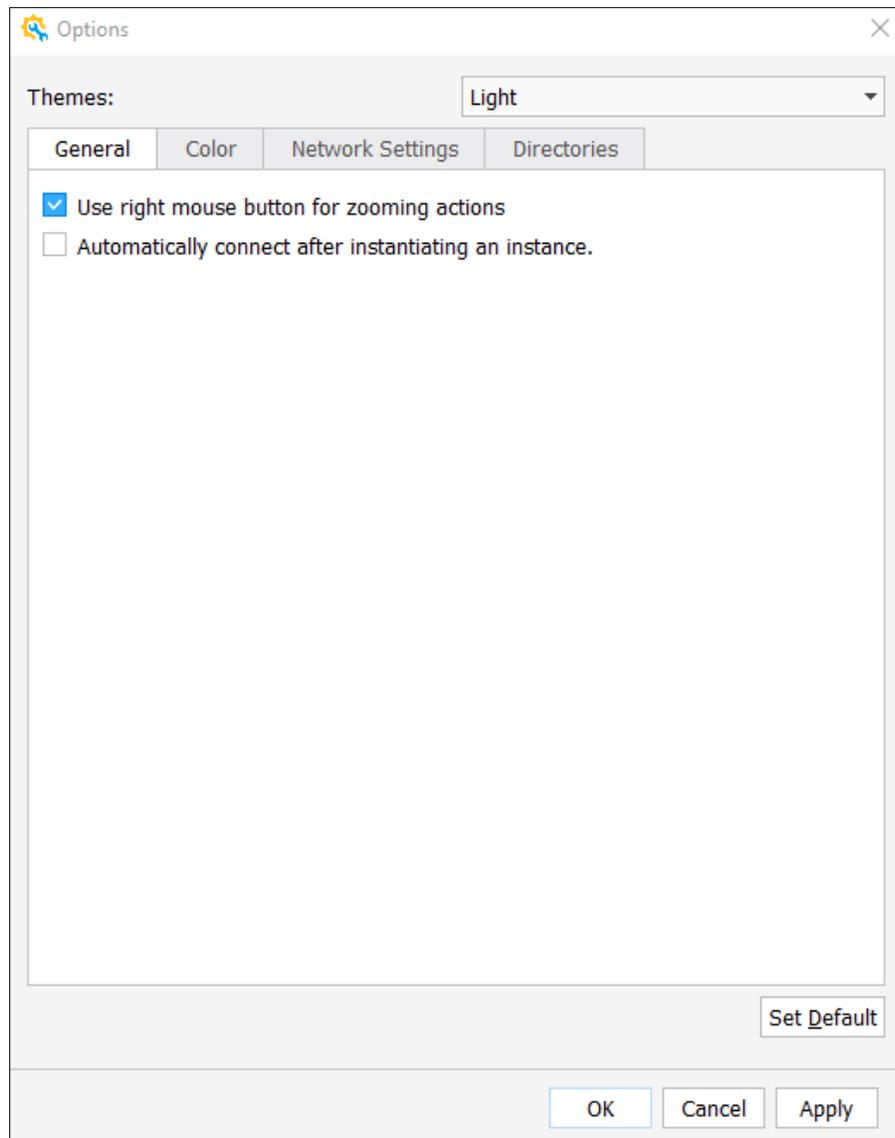
- Zoom In (Ctrl++) — enlarges the view of the entire layout.
- Zoom Out (Ctrl+-) — reduces the view of the entire layout.
- Zoom Fit — reduces or enlarges the entire layout so that it fits inside the window.
- Zoom To — enlarges the size of one or more selected objects on the layout and fills the window 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 2.47). 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 2.47. 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 2.48).

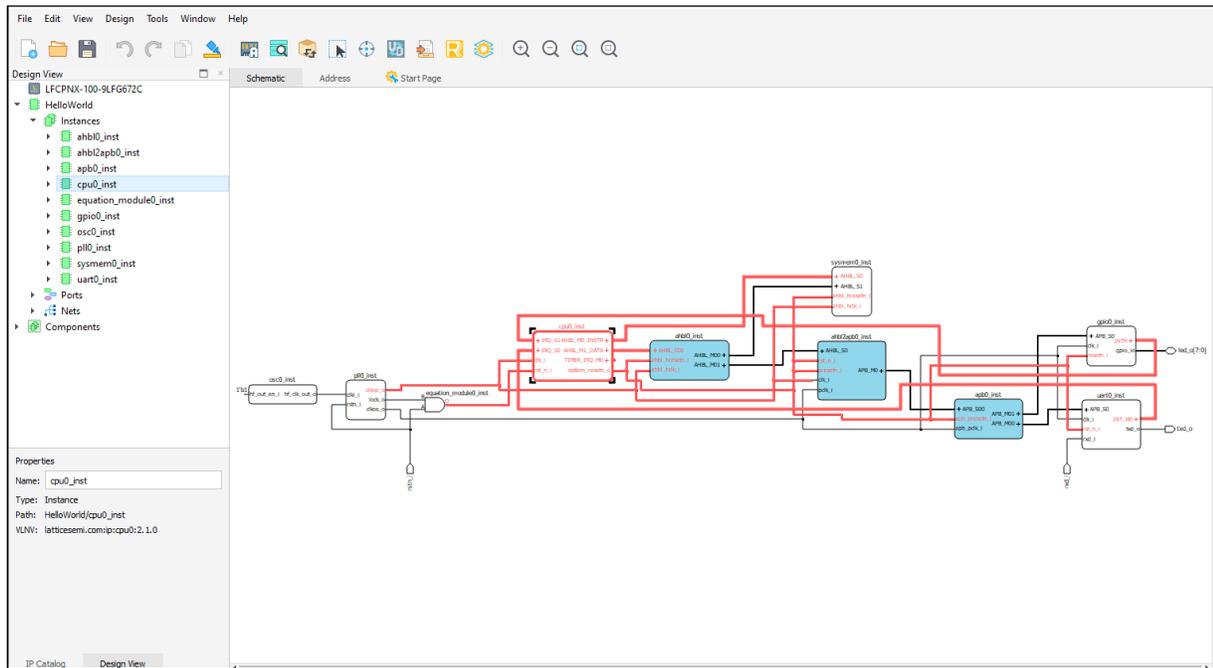


Figure 2.48. Show Connectivity of the Module

### 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 2.49). Click **Highlight**  again. You can remove highlighting. In **Tools Options** section, you can change the highlight color.

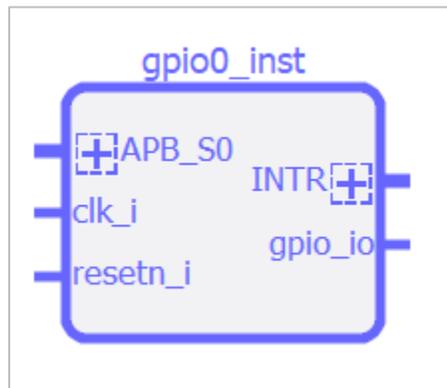


Figure 2.49. Highlight an Object

### 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 2.50).

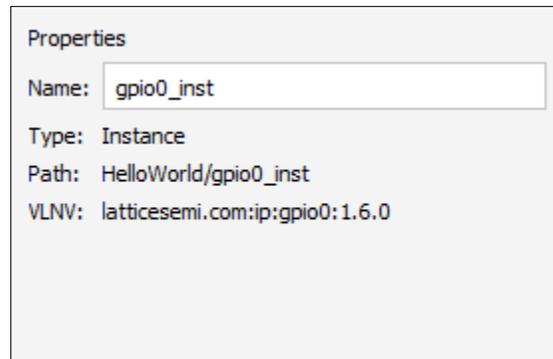


Figure 2.50. Object Properties

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 2.51) opens.

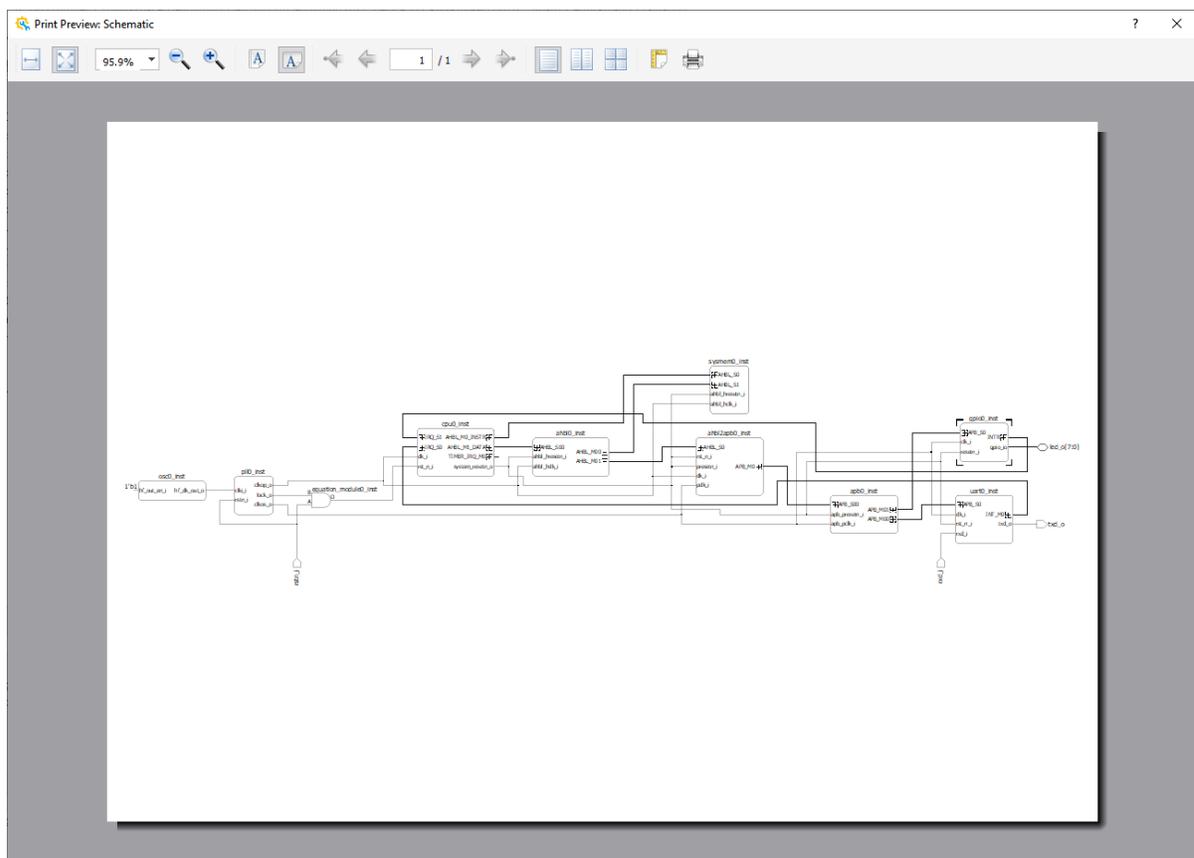


Figure 2.51. Print Preview

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.

### 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 2.52). An allowed net becomes bold when you hover over it (Figure 2.53).

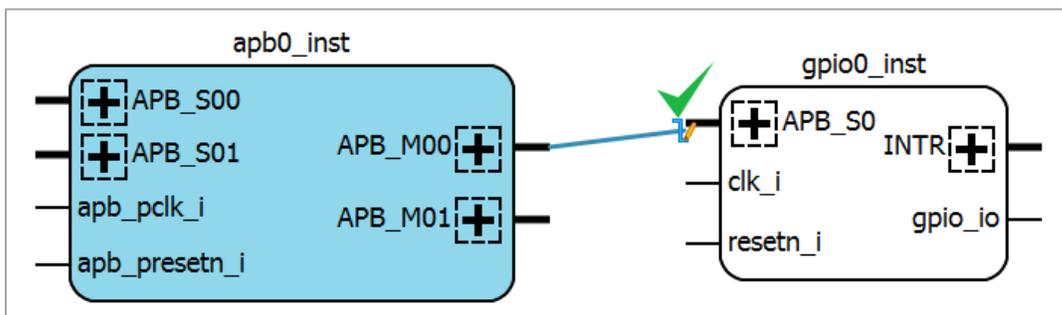


Figure 2.52. Draw a Pin or a Port

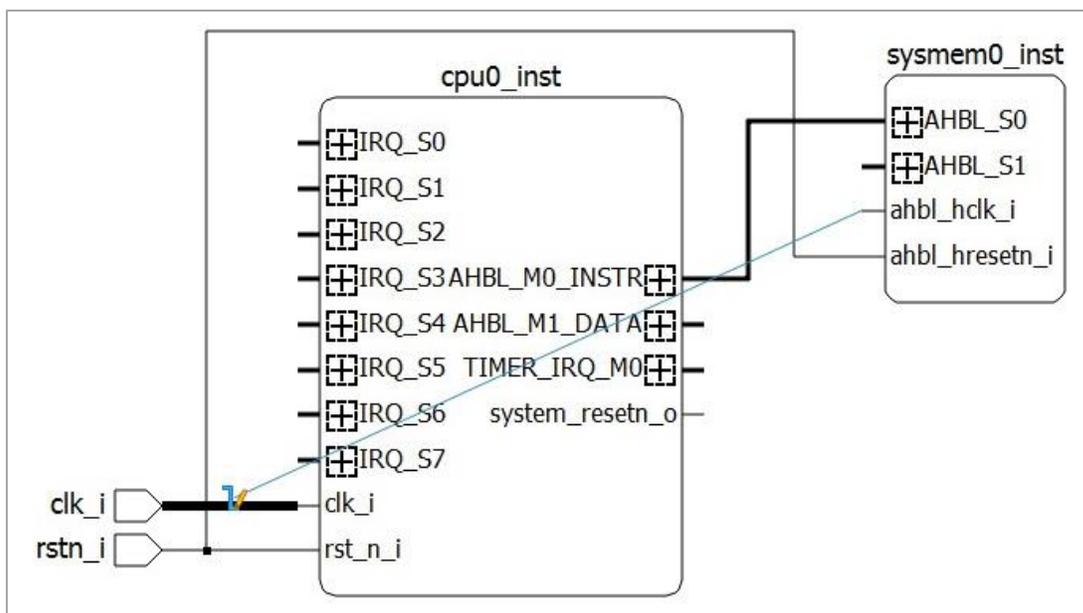


Figure 2.53. Draw Nets

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.

- You can connect multiple ports once. When more than one port is selected (Figure 2.54), click **connect**  from the right-click menu to implement it.

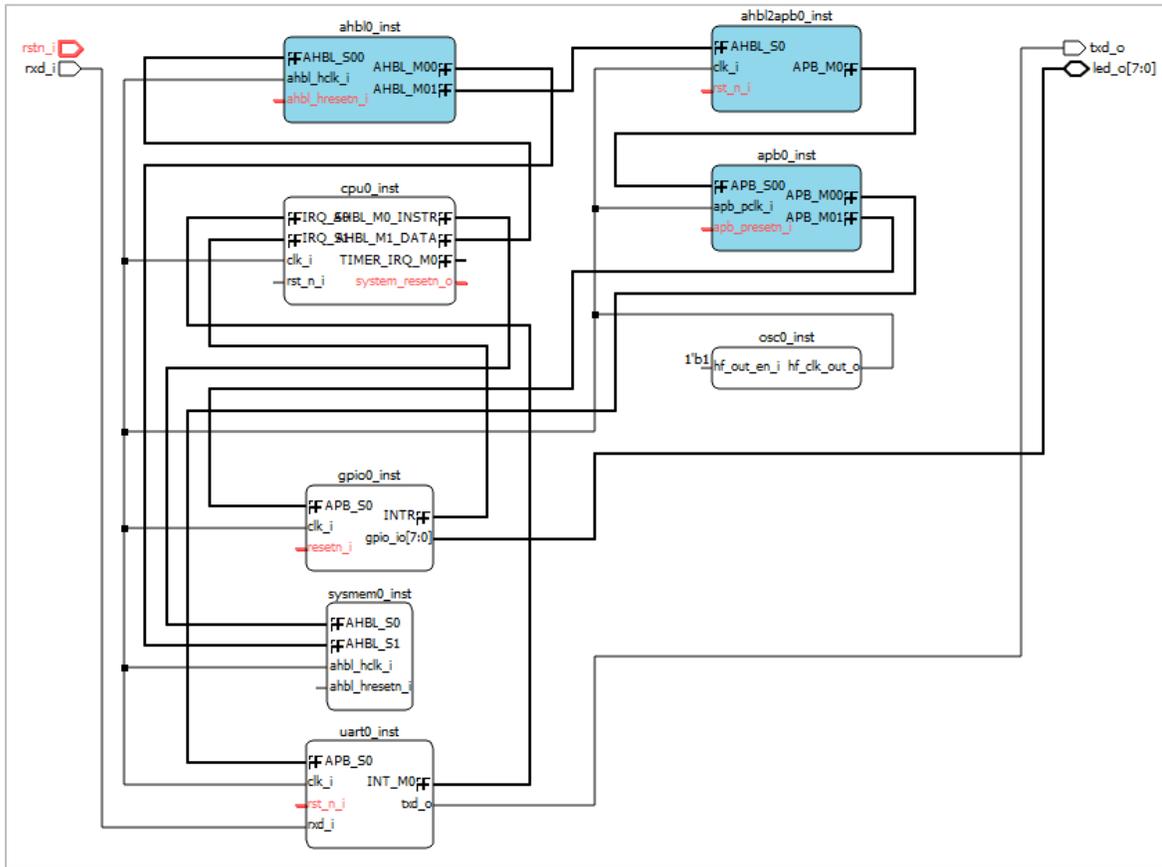


Figure 2.54. Select More than One Ports

- When the connection is completed, right-click to leave the drawing mode.  
Pin/Net/Pinface/Netface display filter:

Click the  icon on Schematic view, the filter shows. By clicking it again, filter hides.

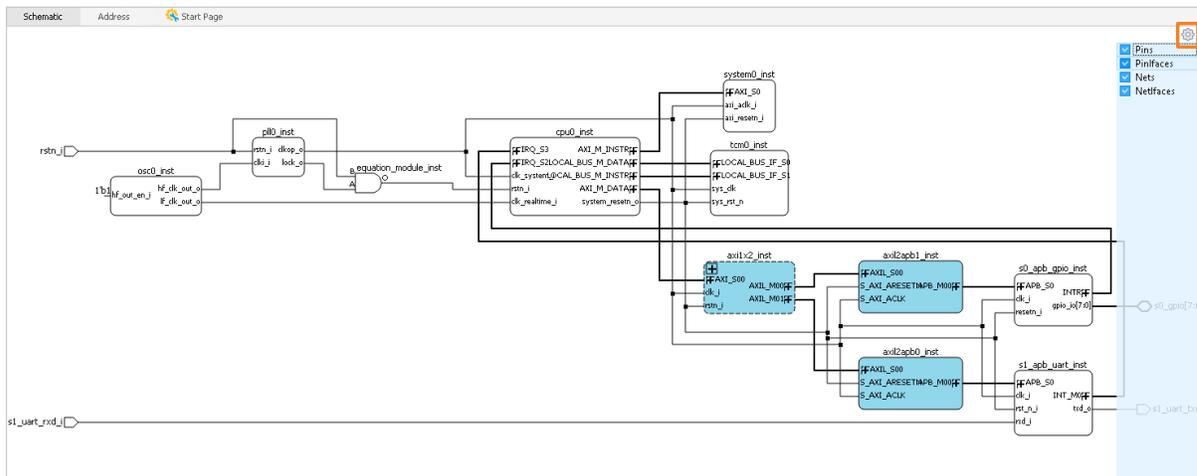


Figure 2.55. Pin/Net/Pinface/Netface Display Filter

When Pin/Pinface is hidden, the connection is hidden. The hidden status is not to be saved in design. It is only used to view design easily.

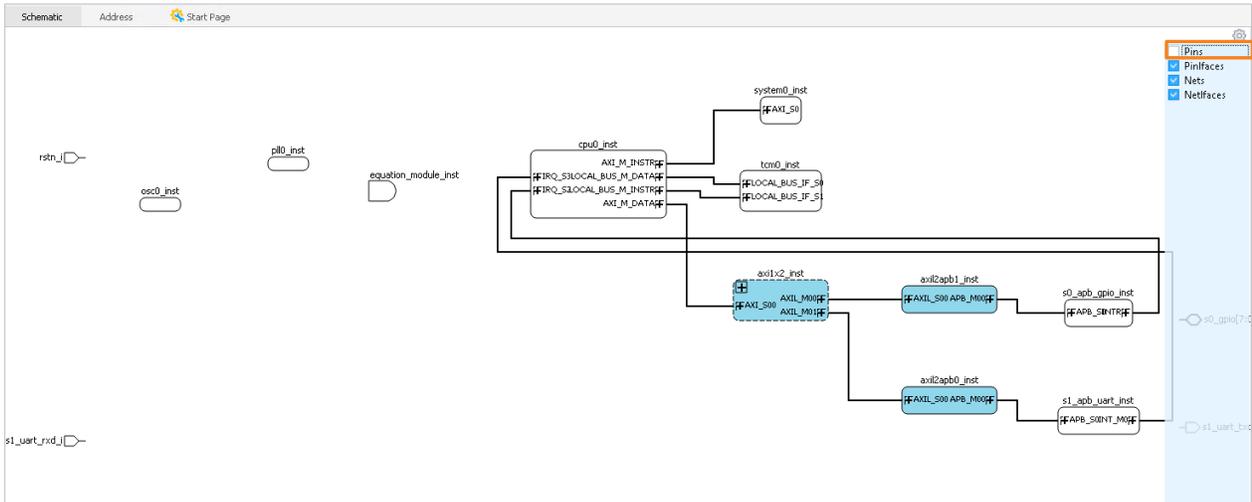


Figure 2.56. Pin/Pinface is Hidden

### 2.2.6.2. Connect Modules by Selecting Points

1. Select the pins, ports, and nets you want to connect, using Ctrl + Left click.
2. Right-click one of the selected objects. Choose **Connect**  from the menu, and the objects will be connected.

### 2.2.6.3. Connect Pins by Auto Connect

1. Right-click on Schematic view and choose **Auto Connect** (Figure 2.57). The Connect Ports dialog appears (Figure 2.58).

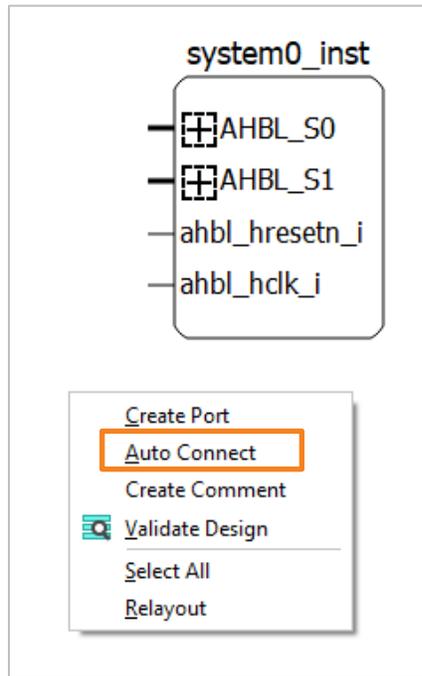


Figure 2.57. Action Menu of Right-clicking on a Blank

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 2.58).
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.

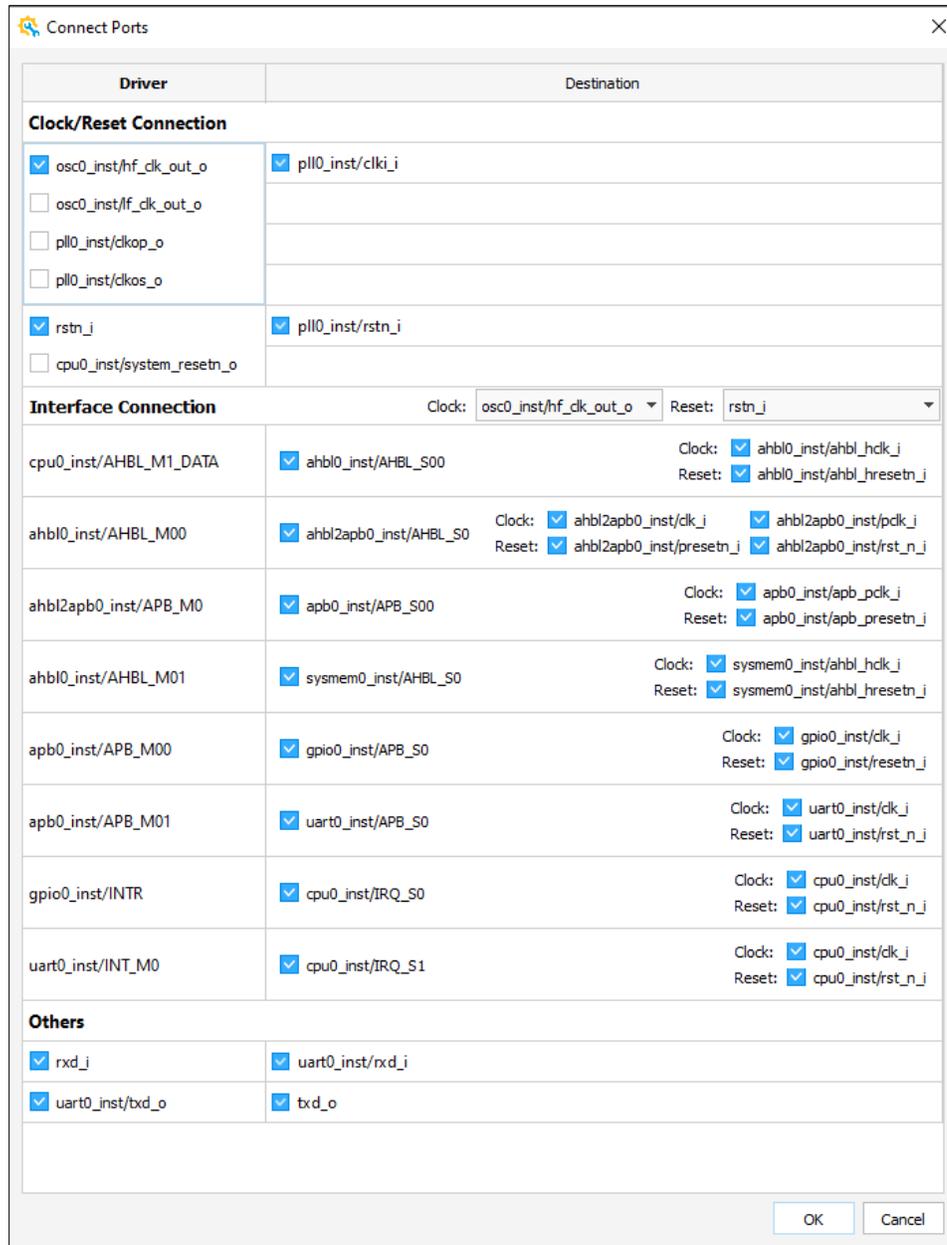


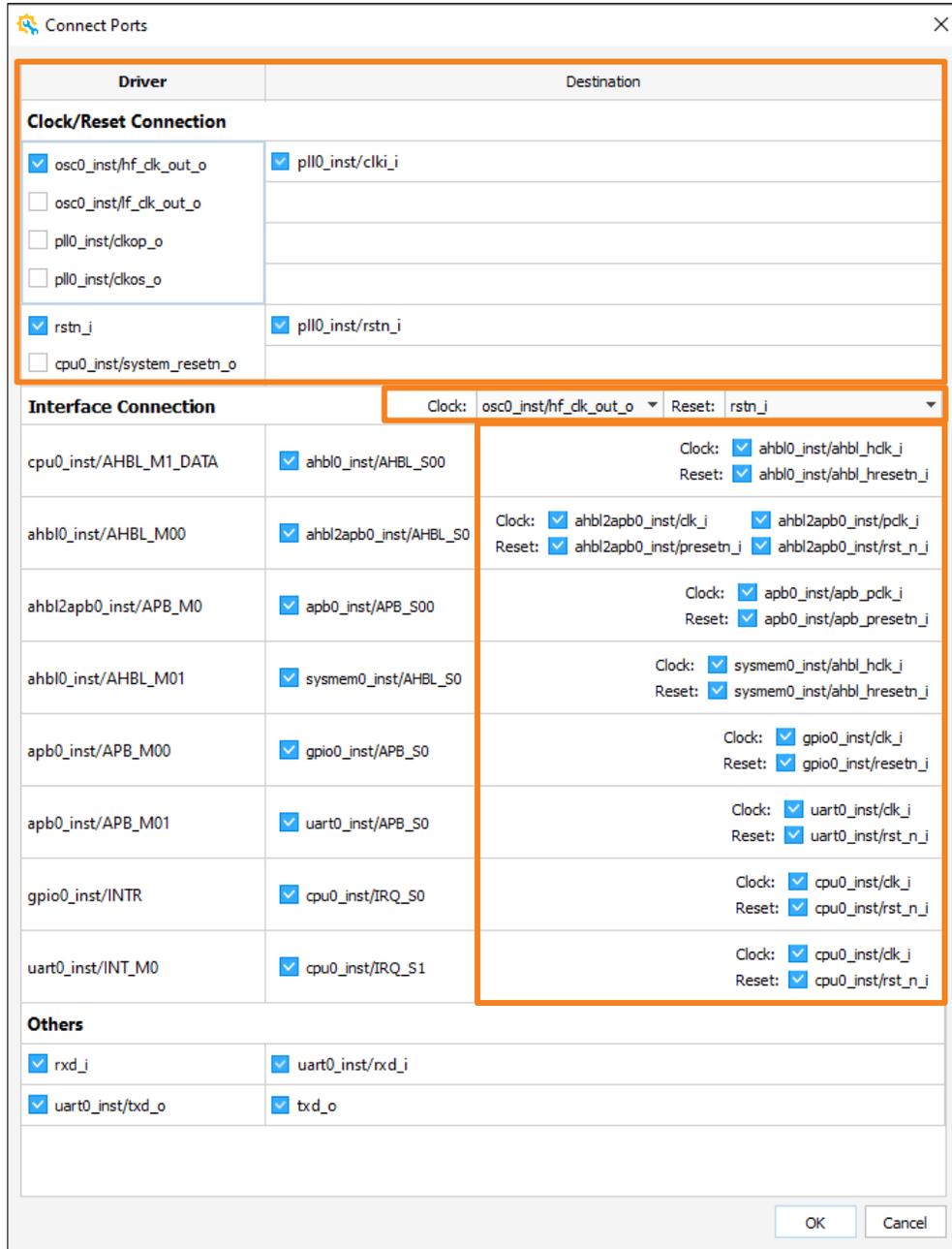
Figure 2.58. Action Menu of Connecting Ports

### 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 2.59).



**Figure 2.59. 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 2.60), the clock/reset of other instances are thus connected to the clock/reset port of the CPU Instance (Figure 2.60).

<input type="checkbox"/> rstn_i	<input checked="" type="checkbox"/> pll0_inst/rstn_i		
<input checked="" type="checkbox"/> cpu0_inst/system_resestn_o			
<b>Interface Connection</b>		Clock: osc0_inst/hf_clk_out_o	Reset: rstn_i
cpu0_inst/AHBL_M1_DATA	<input checked="" type="checkbox"/> ahbl0_inst/AHBL_S00	Clock: <input checked="" type="checkbox"/> ahbl0_inst/ahbl_hclk_j	Reset: <input checked="" type="checkbox"/> ahbl0_inst/ahbl_hresestn_j
ahbl0_inst/AHBL_M00	<input checked="" type="checkbox"/> ahbl2apb0_inst/AHBL_S0	Clock: <input checked="" type="checkbox"/> ahbl2apb0_inst/clk_j <input checked="" type="checkbox"/> ahbl2apb0_inst/pclk_j	Reset: <input checked="" type="checkbox"/> ahbl2apb0_inst/presestn_j <input checked="" type="checkbox"/> ahbl2apb0_inst/rst_n_j
ahbl2apb0_inst/APB_M0	<input checked="" type="checkbox"/> apb0_inst/APB_S00	Clock: <input checked="" type="checkbox"/> apb0_inst/apb_pclk_j	Reset: <input checked="" type="checkbox"/> apb0_inst/apb_presestn_j
ahbl0_inst/AHBL_M01	<input checked="" type="checkbox"/> system0_inst/AHBL_S0	Clock: <input checked="" type="checkbox"/> system0_inst/ahbl_hclk_j	Reset: <input checked="" type="checkbox"/> system0_inst/ahbl_hresestn_j
apb0_inst/APB_M00	<input checked="" type="checkbox"/> gpio0_inst/APB_S0	Clock: <input checked="" type="checkbox"/> gpio0_inst/clk_j	Reset: <input checked="" type="checkbox"/> gpio0_inst/resestn_j
apb0_inst/APB_M01	<input checked="" type="checkbox"/> uart0_inst/APB_S0	Clock: <input checked="" type="checkbox"/> uart0_inst/clk_j	Reset: <input checked="" type="checkbox"/> uart0_inst/rst_n_j
gpio0_inst/INTR	<input checked="" type="checkbox"/> cpu0_inst/IRQ_S0	Clock: <input checked="" type="checkbox"/> cpu0_inst/clk_j	Reset: <input checked="" type="checkbox"/> cpu0_inst/rst_n_j
uart0_inst/INT_M0	<input checked="" type="checkbox"/> cpu0_inst/IRQ_S1	Clock: <input checked="" type="checkbox"/> cpu0_inst/clk_j	Reset: <input checked="" type="checkbox"/> cpu0_inst/rst_n_j

**Figure 2.60. 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 2.60.
- 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\_resestn\_o should not connect to cpu0\_inst.rst\_n\_i in Figure 2.60.

## 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 2.61), `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 2.61).

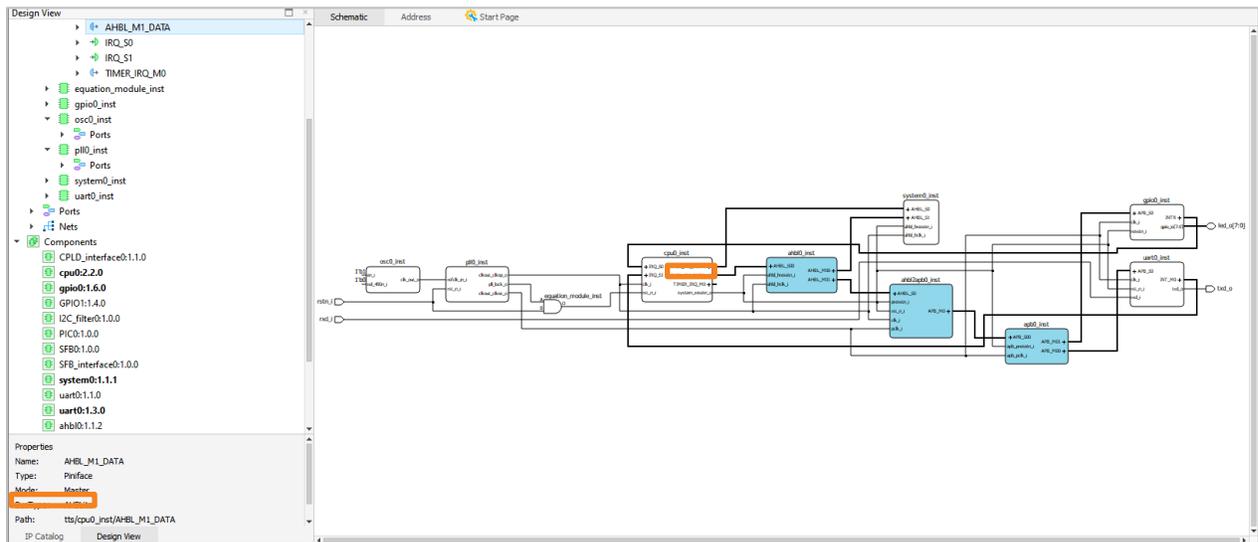


Figure 2.61. Interface Type of `cpu0_inst.AHBL_M1_DATA`

### Interface Connects Rules

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 2.62 and Figure 2.63.

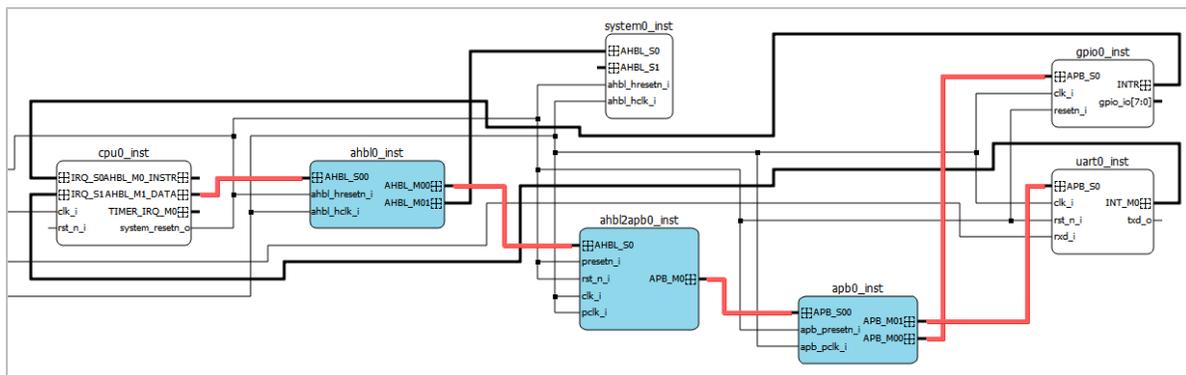


Figure 2.62. Interface Connection on from CPU to Bus/Bridge instances in Schematic View

Interface Connection		Clock: osc0_inst/hf_clk_out_o	Reset: rstn_j
cpu0_inst/AHBL_M1_DATA	<input checked="" type="checkbox"/> ahbl0_inst/AHBL_S00	Clock: <input checked="" type="checkbox"/> ahbl0_inst/ahbl_hclk_j	Reset: <input checked="" type="checkbox"/> ahbl0_inst/ahbl_hresetn_j
ahbl0_inst/AHBL_M00	<input checked="" type="checkbox"/> ahbl2apb0_inst/AHBL_S0	Clock: <input checked="" type="checkbox"/> ahbl2apb0_inst/clk_j <input checked="" type="checkbox"/> ahbl2apb0_inst/pclk_j	Reset: <input checked="" type="checkbox"/> ahbl2apb0_inst/presetn_j <input checked="" type="checkbox"/> ahbl2apb0_inst/rst_n_j
ahbl2apb0_inst/APB_M0	<input checked="" type="checkbox"/> apb0_inst/APB_S00	Clock: <input checked="" type="checkbox"/> apb0_inst/apb_pclk_j	Reset: <input checked="" type="checkbox"/> apb0_inst/apb_presetn_j
ahbl0_inst/AHBL_M01	<input checked="" type="checkbox"/> system0_inst/AHBL_S0	Clock: <input checked="" type="checkbox"/> system0_inst/ahbl_hclk_j	Reset: <input checked="" type="checkbox"/> system0_inst/ahbl_hresetn_j
apb0_inst/APB_M00	<input checked="" type="checkbox"/> gpio0_inst/APB_S0	Clock: <input checked="" type="checkbox"/> gpio0_inst/clk_j	Reset: <input checked="" type="checkbox"/> gpio0_inst/resetn_j
apb0_inst/APB_M01	<input checked="" type="checkbox"/> uart0_inst/APB_S0	Clock: <input checked="" type="checkbox"/> uart0_inst/clk_j	Reset: <input checked="" type="checkbox"/> uart0_inst/rst_n_j
gpio0_inst/INTR	<input checked="" type="checkbox"/> cpu0_inst/IRQ_S0	Clock: <input checked="" type="checkbox"/> cpu0_inst/clk_j	Reset: <input checked="" type="checkbox"/> cpu0_inst/rst_n_j
uart0_inst/INT_M0	<input checked="" type="checkbox"/> cpu0_inst/IRQ_S1	Clock: <input checked="" type="checkbox"/> cpu0_inst/clk_j	Reset: <input checked="" type="checkbox"/> cpu0_inst/rst_n_j

Figure 2.63. 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 2.64 and ). In Figure 2.66 and Figure 2.67, you can see gpio0\_inst.INTR and cpu0\_inst.IRQ\_S0 have the bus type: Interrupt.

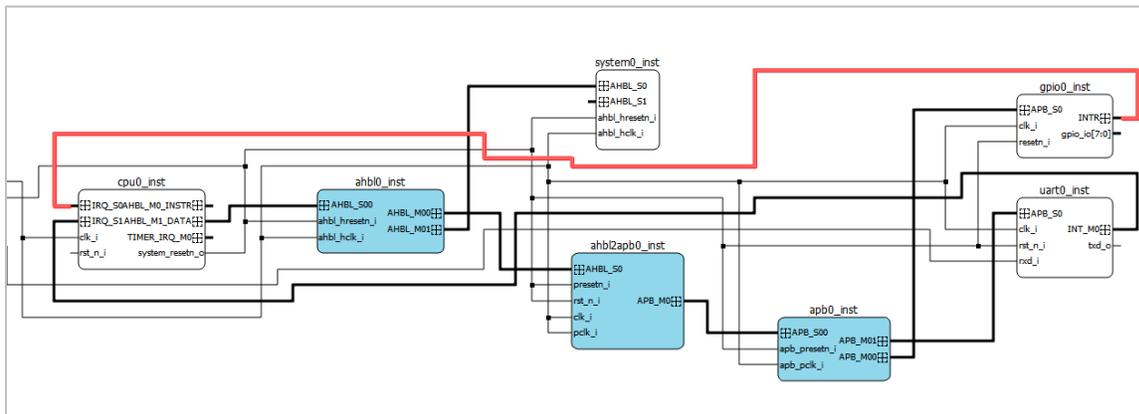


Figure 2.64. Interface Connection on Other Instances Connecting back to CPU in Schematic View

Interface Connection		Clock: osc0_inst/hf_clk_out_o	Reset: rstn_j
cpu0_inst/AHBL_M1_DATA	<input checked="" type="checkbox"/> ahbl0_inst/AHBL_S00	Clock: <input checked="" type="checkbox"/> ahbl0_inst/ahbl_hclk_j	Reset: <input checked="" type="checkbox"/> ahbl0_inst/ahbl_hresetn_j
ahbl0_inst/AHBL_M00	<input checked="" type="checkbox"/> ahbl2apb0_inst/AHBL_S0	Clock: <input checked="" type="checkbox"/> ahbl2apb0_inst/dk_j <input checked="" type="checkbox"/> ahbl2apb0_inst/pdk_j	Reset: <input checked="" type="checkbox"/> ahbl2apb0_inst/presetn_j <input checked="" type="checkbox"/> ahbl2apb0_inst/rst_n_j
ahbl2apb0_inst/APB_M0	<input checked="" type="checkbox"/> apb0_inst/APB_S00	Clock: <input checked="" type="checkbox"/> apb0_inst/apb_pdk_j	Reset: <input checked="" type="checkbox"/> apb0_inst/apb_presetn_j
ahbl0_inst/AHBL_M01	<input checked="" type="checkbox"/> system0_inst/AHBL_S0	Clock: <input checked="" type="checkbox"/> system0_inst/ahbl_hclk_j	Reset: <input checked="" type="checkbox"/> system0_inst/ahbl_hresetn_j
apb0_inst/APB_M00	<input checked="" type="checkbox"/> gpio0_inst/APB_S0	Clock: <input checked="" type="checkbox"/> gpio0_inst/dk_j	Reset: <input checked="" type="checkbox"/> gpio0_inst/resetn_j
apb0_inst/APB_M01	<input checked="" type="checkbox"/> uart0_inst/APB_S0	Clock: <input checked="" type="checkbox"/> uart0_inst/dk_j	Reset: <input checked="" type="checkbox"/> uart0_inst/rst_n_j
gpio0_inst/INTR	<input checked="" type="checkbox"/> cpu0_inst/IRQ_S0	Clock: <input checked="" type="checkbox"/> cpu0_inst/dk_j	Reset: <input checked="" type="checkbox"/> cpu0_inst/rst_n_j
uart0_inst/INT_M0	<input checked="" type="checkbox"/> cpu0_inst/IRQ_S1	Clock: <input checked="" type="checkbox"/> cpu0_inst/dk_j	Reset: <input checked="" type="checkbox"/> cpu0_inst/rst_n_j

Figure 2.65. Interface Connection on Other Instances Connecting back to CPU in Connect Port View

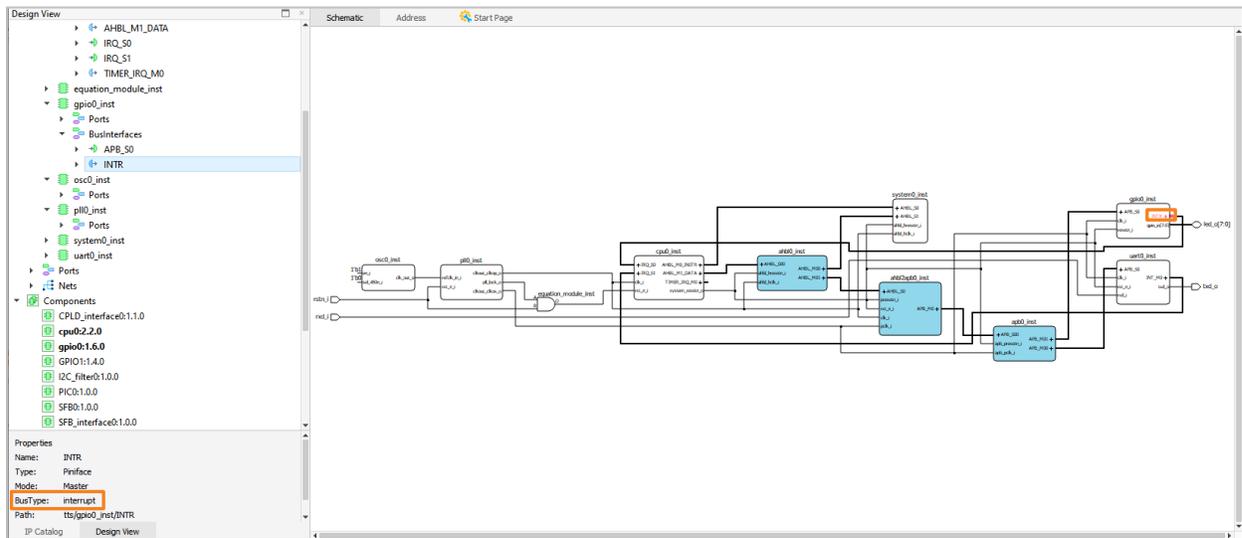
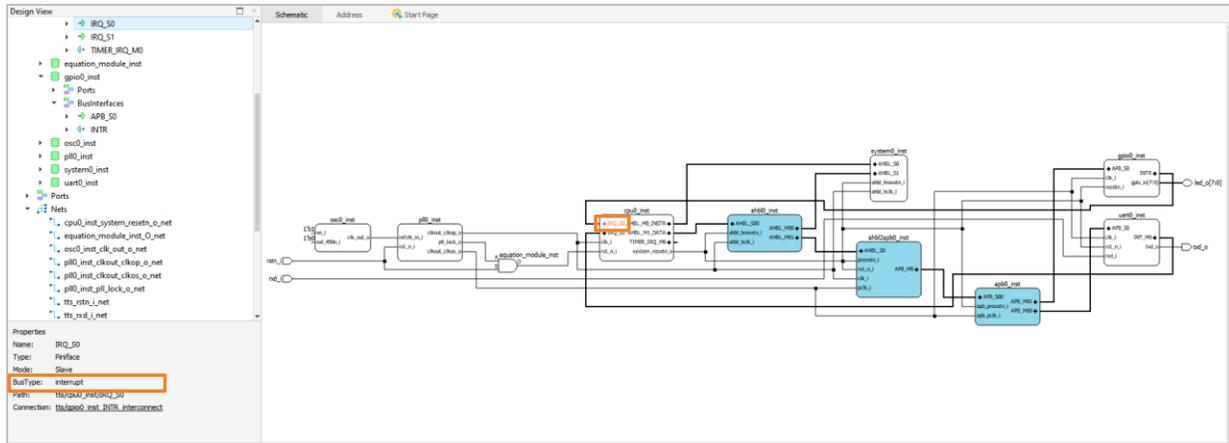


Figure 2.66. Interface Type of gpio0\_inst.INTR



**Figure 2.67. 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\_MO\_INSTR, it does not perform auto-connect.

### C. Other Connection

If ports have the same port name, for example, rxd\_i and uart0\_ins.rxd\_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 2.68).

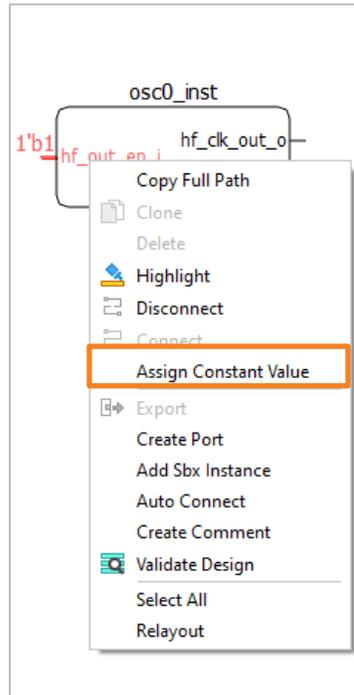


Figure 2.68. Right-click Menu of an Input Pin

2. A small dialog box appears (Figure 2.69). 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.

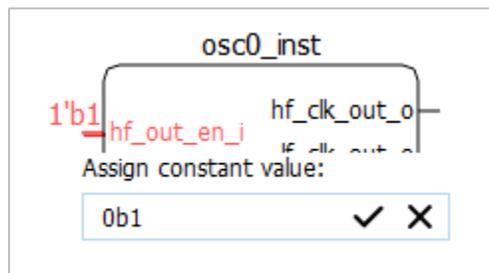


Figure 2.69. Dialog Box of Assigning Constant Value to an Input Pin

### 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 2.70).

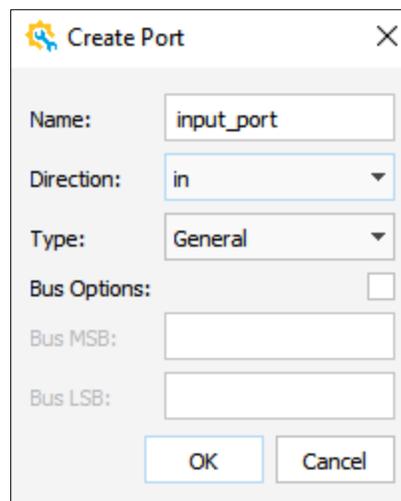


Figure 2.70. Create Port Dialog Box

- 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 2.71 shows an input port, port name of which is on the left. Figure 2.72 shows an output port and an inout port, port names of which are on the right.

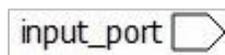


Figure 2.71. Input Port

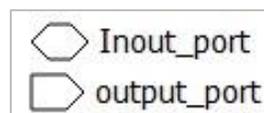


Figure 2.72. Output Port and Inout Port

- g. Connect the port to the module pins. Refer to the [Connecting Modules](#) section for more details.

h. (Optional) Double click on the Port/ Portbus in schematic view, will open the port edit dialog (Figure 2.73).

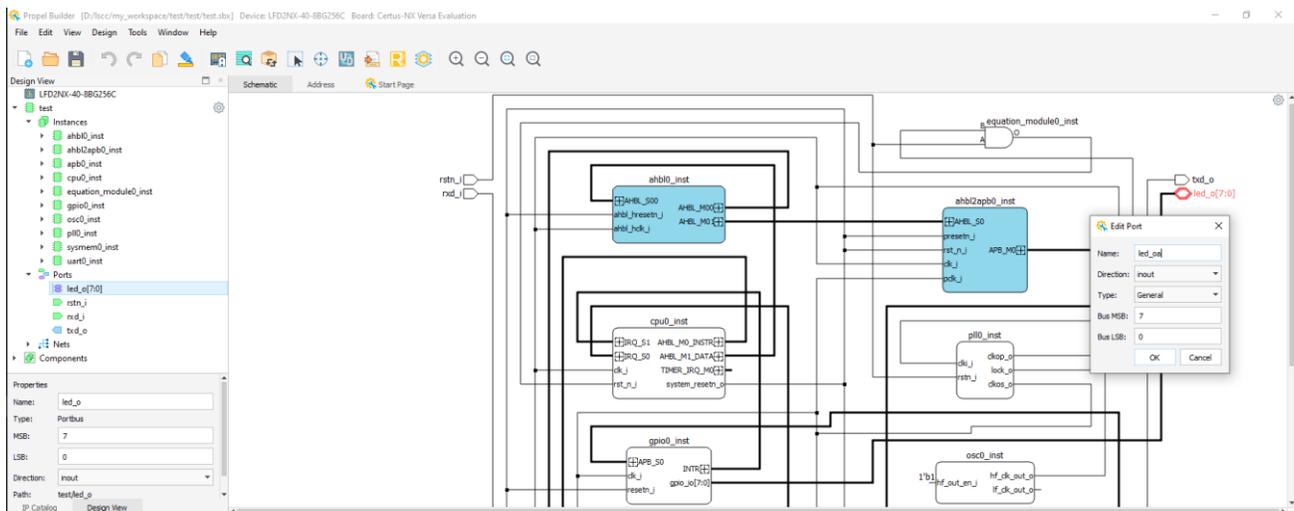


Figure 2.73. Port Edit Dialog

- 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 2.74).

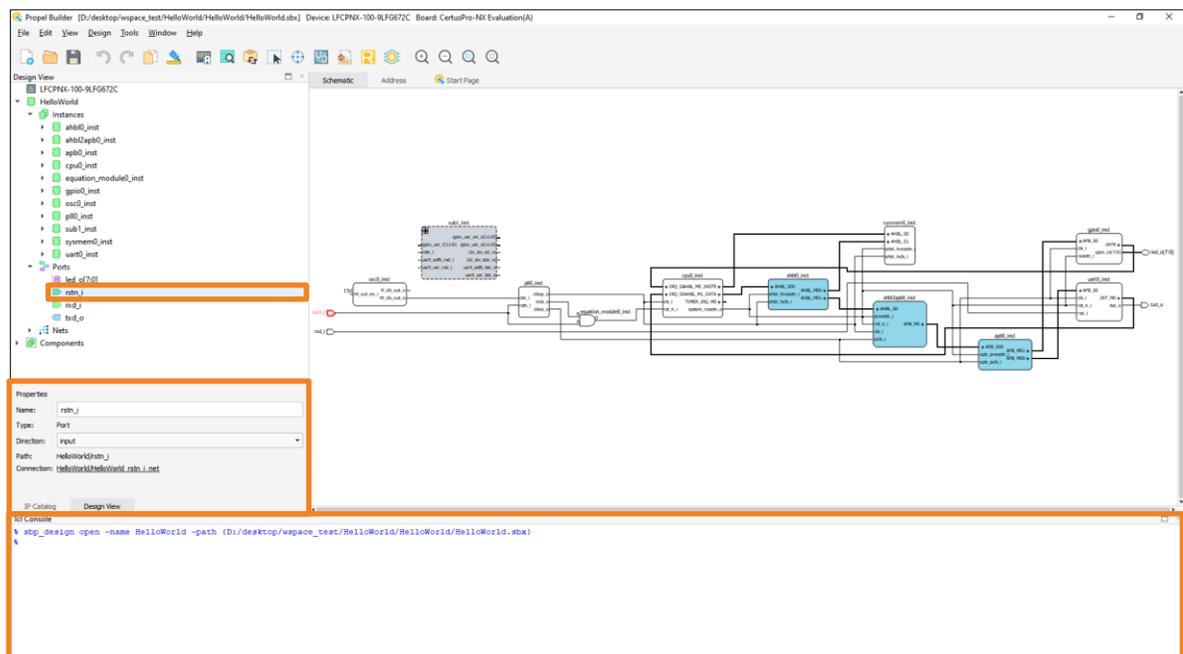


Figure 2.74. Properties of Port

- 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 2.75).
- c. 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 2.75).

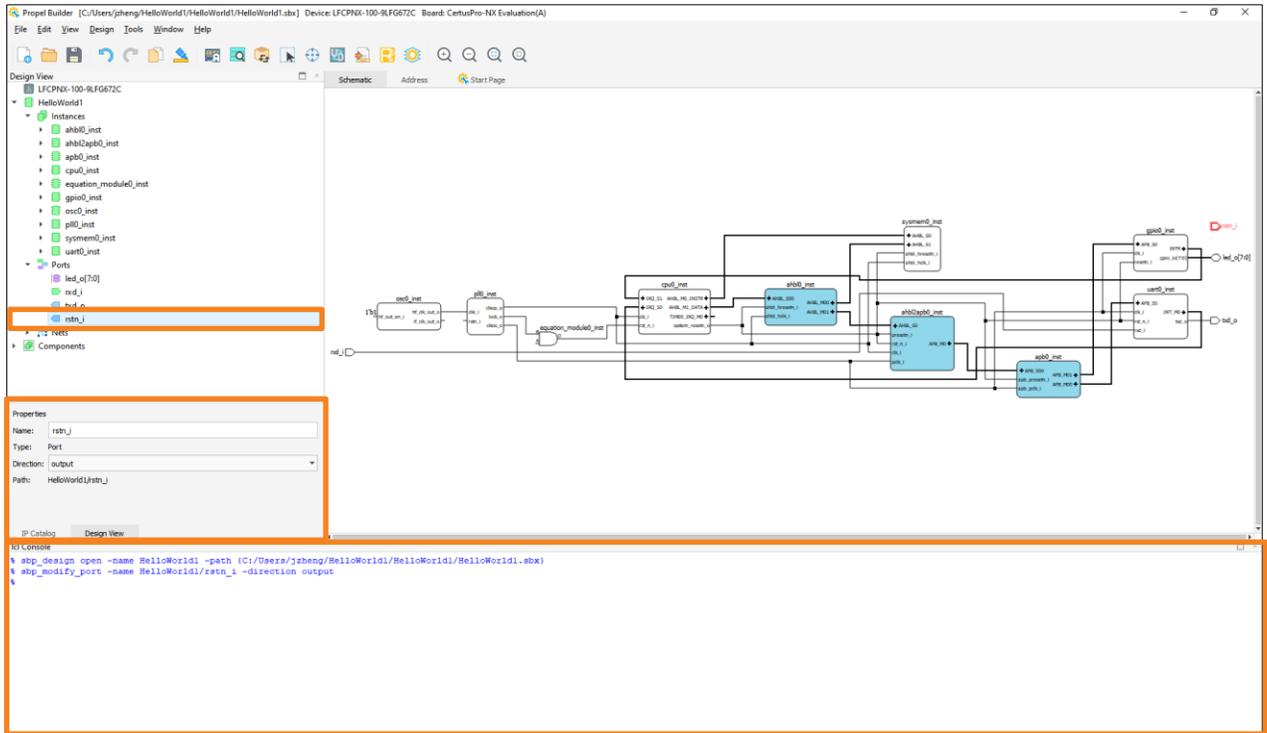


Figure 2.75. Port Direction

- d. TCL history can be viewed in TCL History window (Figure 2.77) by choosing **Design > TCL History** (Figure 2.76).

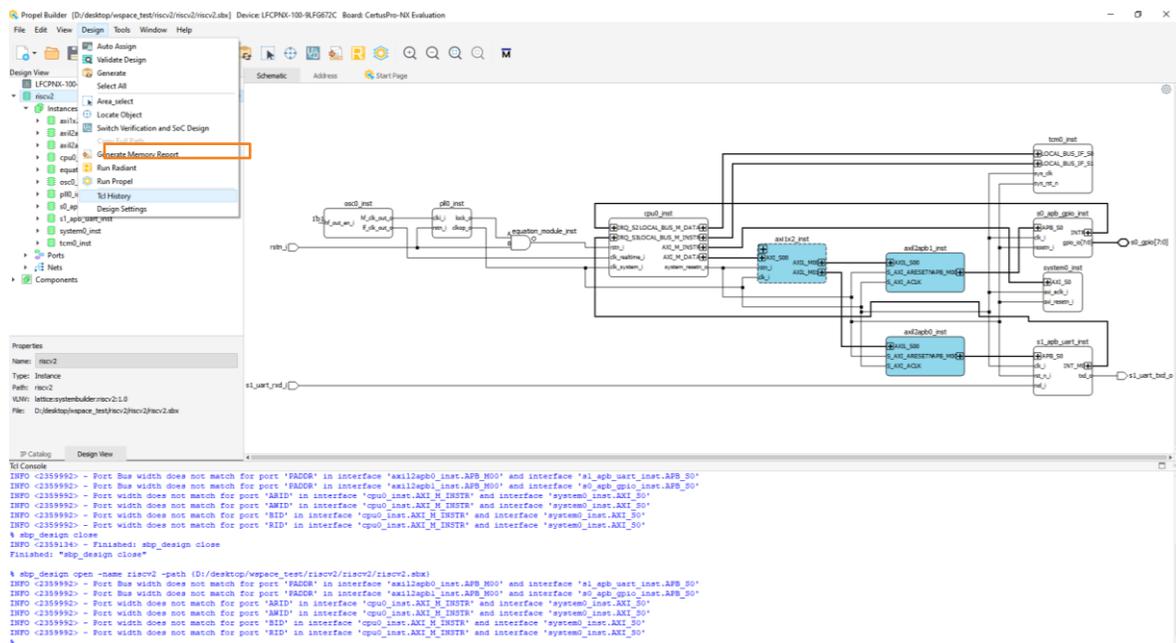


Figure 2.76. Open TCL History

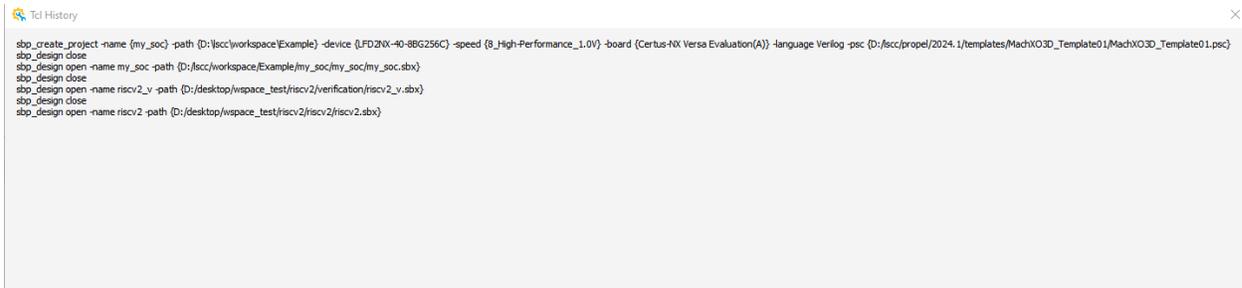


Figure 2.77. Tcl History Window

- e. PortBus MSB, LSB and direction can be modified. Click on PortBus and then use the drop-down menu to choose the desired direction (Figure 2.78).

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

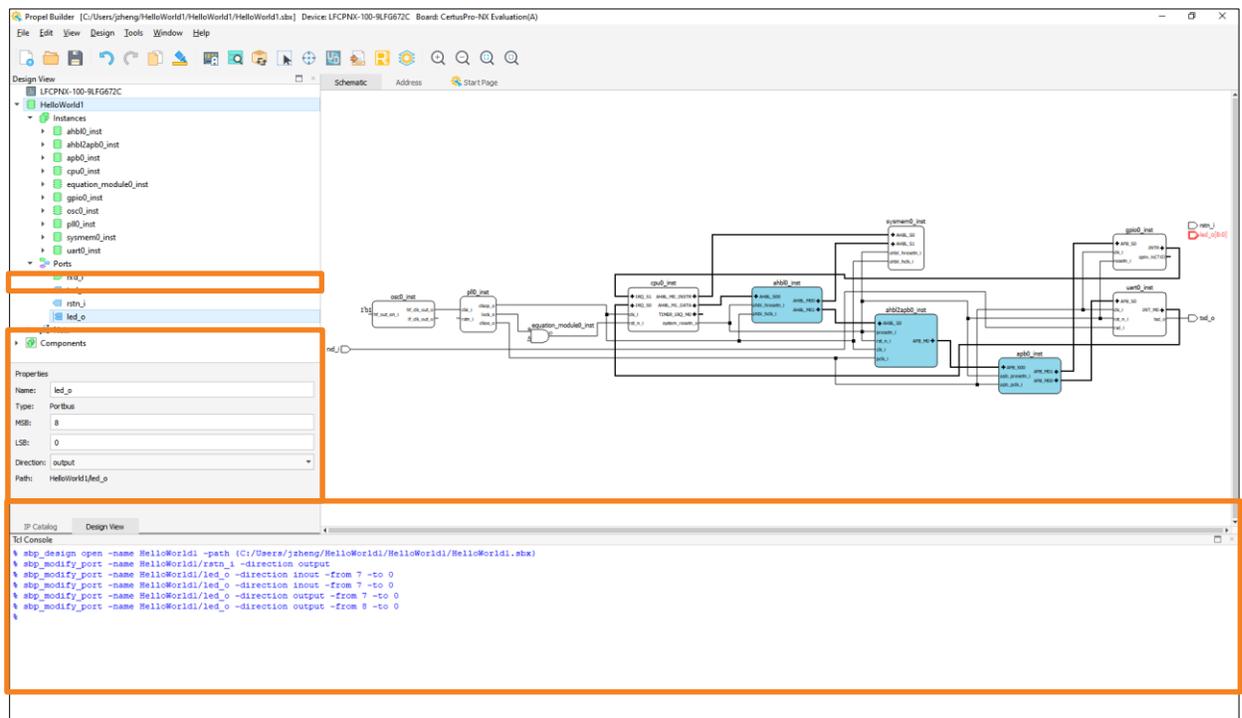


Figure 2.78. PortBus Direction

## 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 subordinate 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 subordinate 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 2.79) shows.

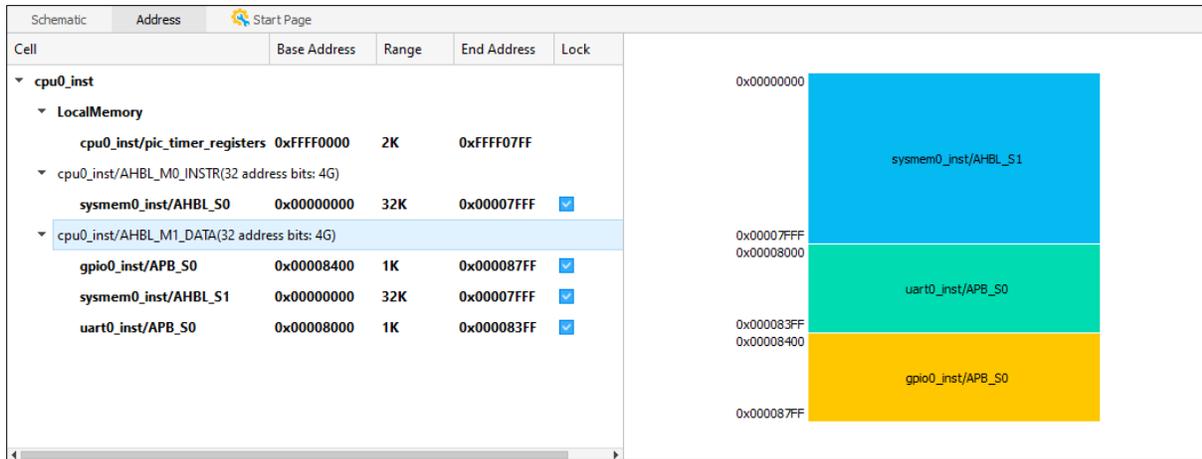


Figure 2.79. Address View

- (Optional) Set or clear the Lock options as desired in Address view.
- (Optional) Double-click the base address in Address view (Figure 2.80). Type the new value and press **Enter**.  
**Note:** 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.

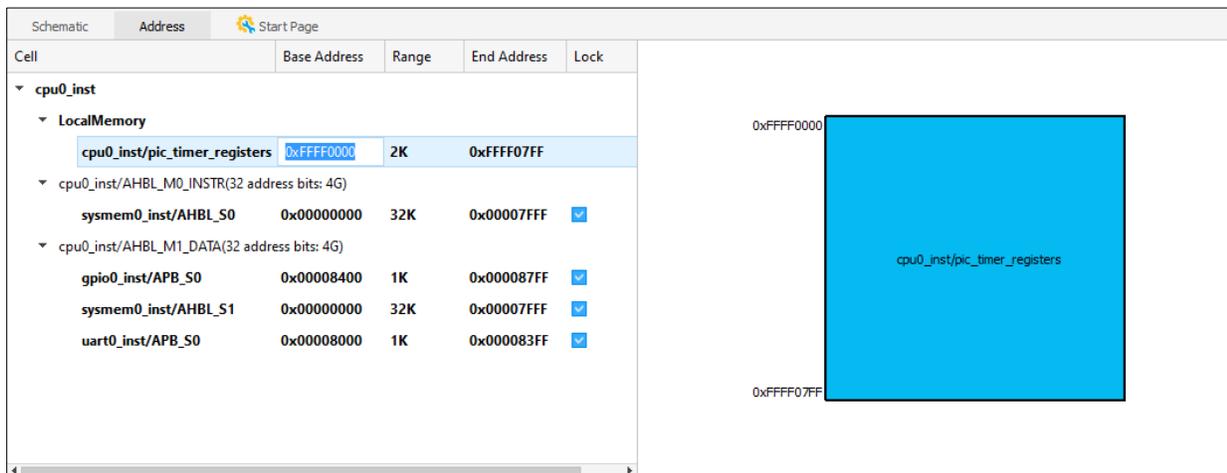


Figure 2.80. Edit Base Address

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

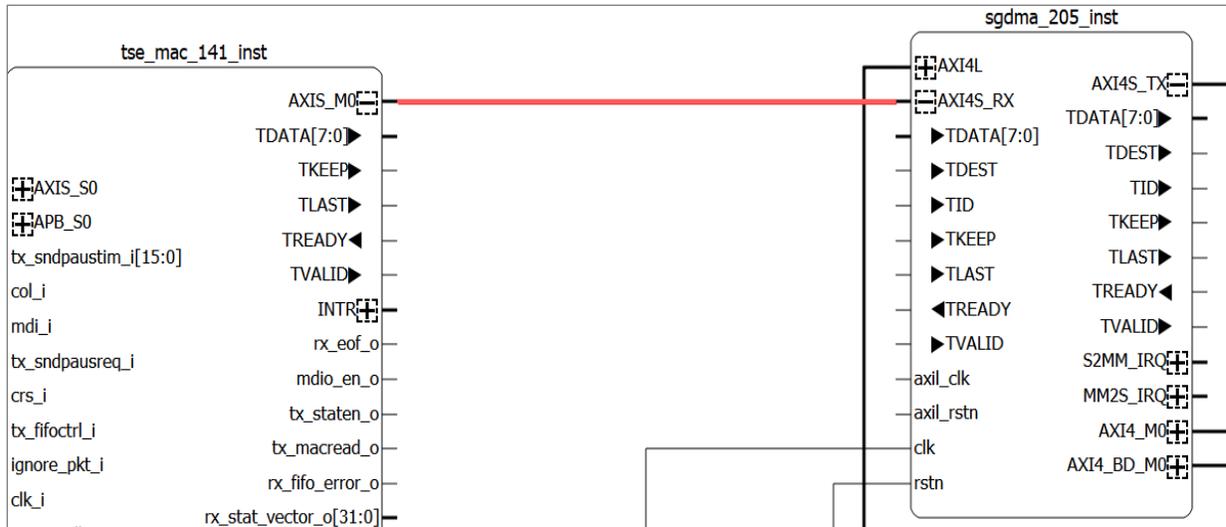
### 2.2.9. Validating the Design

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: Click the **Validate Design** icon  on the toolbar of the Propel Builder main window. The DRC results appear in the TCL Console.

**Note:** Optional bus ports handling

Optional input ports on Propel IP bus interface, such as AXI4 Stream, are left dangling when the driver side connected bus do not support these ports. This may be IP dependent if the dangling input signals affect the functionality.



**Figure 2.81. Optional Bus Ports Handling**

The default values of these ports are defined in abstraction Definition of AMBA Bus. Builder can detect the dangling ports and then tie the default values to them. (Figure 2.81)

### 2.2.10. Generating the Propel Design

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

To generate Propel design:

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) and can generate some RTL files. The RTL files include a top RTL file that is the top module for the entire Propel design. The RTL files for IPs are regenerated when you click this button. If any change is made on an IP, you need to use this icon to finish updating that IP.

## 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 2.82). To see more about address space, refer to the [Adjusting Address Spaces](#) section.

<p>SOC Memory Map for SOC : HelloWorld1</p> <p>Table of Contents</p> <p><a href="#">Project Summary</a></p> <p><a href="#">Detailed Memory Map</a></p> <ul style="list-style-type: none"> <li>- <a href="#">cpu0_inst</a></li> <li>- <a href="#">gpio0_inst</a></li> <li>- <a href="#">system0_inst</a></li> <li>- <a href="#">uart0_inst</a></li> </ul>	<p>Project Summary Page Generated by Lattice Propel Software SOC Verification Engine Copyright (C) 1992-2023 Lattice Semiconductor Corporation. All Rights Reserved.</p> <p>Project Summary</p> <table border="1"> <thead> <tr> <th>Project Name</th> <th>Family</th> <th>Device</th> <th>PartName</th> <th>Speed</th> <th>Language</th> </tr> </thead> <tbody> <tr> <td>HelloWorld1</td> <td>LFCPNX</td> <td>LFCPNX-100</td> <td>LFCPNX-100-9LFG672C</td> <td>9_High-Performance_1.0V</td> <td>verilog</td> </tr> </tbody> </table>	Project Name	Family	Device	PartName	Speed	Language	HelloWorld1	LFCPNX	LFCPNX-100	LFCPNX-100-9LFG672C	9_High-Performance_1.0V	verilog
Project Name	Family	Device	PartName	Speed	Language								
HelloWorld1	LFCPNX	LFCPNX-100	LFCPNX-100-9LFG672C	9_High-Performance_1.0V	verilog								

Figure 2.82. Memory Report

Click on the links in the left side pane, you can see more information on Detailed Memory Map (Figure 2.83) and each of the instances (Figure 2.84).

<p>SOC Memory Map for SOC : HelloWorld1</p> <p>Table of Contents</p> <p><a href="#">Project Summary</a></p> <p><a href="#">Detailed Memory Map</a></p> <ul style="list-style-type: none"> <li>- <a href="#">cpu0_inst</a></li> <li>- <a href="#">gpio0_inst</a></li> <li>- <a href="#">system0_inst</a></li> <li>- <a href="#">uart0_inst</a></li> </ul>	<p>Detailed Memory Map Summary Page Generated by Lattice Propel Software SOC Verification Engine Copyright (C) 1992-2023 Lattice Semiconductor Corporation. All Rights Reserved.</p> <p>Instance Summary</p> <table border="1"> <thead> <tr> <th>Instance Name</th> <th>Start Addr</th> <th>End Addr</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><a href="#">cpu0_inst</a></td> <td>0xFFFF0000</td> <td>0xFFFF07FF</td> <td></td> </tr> <tr> <td><a href="#">gpio0_inst</a></td> <td>0x00008400</td> <td>0x000087FF</td> <td></td> </tr> <tr> <td><a href="#">system0_inst</a></td> <td>0x00000000</td> <td>0x00007FFF</td> <td></td> </tr> <tr> <td><a href="#">uart0_inst</a></td> <td>0x00008000</td> <td>0x000083FF</td> <td></td> </tr> </tbody> </table>	Instance Name	Start Addr	End Addr	Description	<a href="#">cpu0_inst</a>	0xFFFF0000	0xFFFF07FF		<a href="#">gpio0_inst</a>	0x00008400	0x000087FF		<a href="#">system0_inst</a>	0x00000000	0x00007FFF		<a href="#">uart0_inst</a>	0x00008000	0x000083FF	
Instance Name	Start Addr	End Addr	Description																		
<a href="#">cpu0_inst</a>	0xFFFF0000	0xFFFF07FF																			
<a href="#">gpio0_inst</a>	0x00008400	0x000087FF																			
<a href="#">system0_inst</a>	0x00000000	0x00007FFF																			
<a href="#">uart0_inst</a>	0x00008000	0x000083FF																			

Figure 2.83. Memory Report

<p>SOC Memory Map for SOC : HelloWorld1</p> <p>Table of Contents</p> <p><a href="#">Project Summary</a></p> <p><a href="#">Detailed Memory Map</a></p> <ul style="list-style-type: none"> <li>- <a href="#">cpu0_inst</a></li> <li>- <a href="#">cpu10_inst</a></li> <li>- <a href="#">system0_inst</a></li> <li>- <a href="#">uart0_inst</a></li> </ul>	<p>Details for Instance - <code>cpu0_inst</code></p> <p>Generated by Lattice Propel Software SOC Verification Engine Copyright (C) 1992-2023 Lattice Semiconductor Corporation. All Rights Reserved.</p> <p><b>cpu0_inst Overview</b></p> <table border="1"> <thead> <tr> <th>Start Addr</th> <th>End Addr</th> <th>Actual Range</th> <th>Register Width</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0xFFFF0000</td> <td>0xFFFF07FF</td> <td>0x00000800</td> <td>32</td> <td></td> </tr> </tbody> </table> <p><b>cpu0_inst Register Summary</b></p> <table border="1"> <thead> <tr> <th>Offset</th> <th>Register Name</th> <th>Access</th> <th>Reset Value</th> <th>Width</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0000</td> <td><a href="#">PIC_STATUS</a></td> <td>"RW"</td> <td></td> <td>2</td> <td>Interrupt Status Register</td> </tr> <tr> <td>0x0004</td> <td><a href="#">PIC_ENABLE</a></td> <td>"RW"</td> <td></td> <td>2</td> <td>Interrupt Enable Register</td> </tr> <tr> <td>0x0008</td> <td><a href="#">PIC_SET</a></td> <td>"WO"</td> <td></td> <td>2</td> <td>Interrupt Set Register</td> </tr> <tr> <td>0x000C</td> <td><a href="#">PIC_POL</a></td> <td>"RW"</td> <td></td> <td>2</td> <td>Interrupt Polarity Register</td> </tr> <tr> <td>0x0400</td> <td><a href="#">TIMER_CNT_L</a></td> <td>"RW"</td> <td></td> <td>32</td> <td>Timer Counter Lower Register</td> </tr> <tr> <td>0x0404</td> <td><a href="#">TIMER_CNT_H</a></td> <td>"RW"</td> <td></td> <td>32</td> <td>Timer Counter Higher Register</td> </tr> <tr> <td>0x0410</td> <td><a href="#">TIMER_CMP_L</a></td> <td>"RW"</td> <td></td> <td>32</td> <td>Timer Compare Lower Register</td> </tr> <tr> <td>0x0414</td> <td><a href="#">TIMER_CMP_H</a></td> <td>"RW"</td> <td></td> <td>32</td> <td>Timer Compare Higher Register</td> </tr> </tbody> </table> <hr/> <p><b>Field Summary for Register <a href="#">PIC_STATUS</a></b></p> <table border="1"> <thead> <tr> <th>Bit</th> <th>Field Name</th> <th>Access</th> <th>Reset Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>[1:0]</td> <td>status</td> <td>"RW"</td> <td>0</td> <td>Interrupt Status</td> </tr> </tbody> </table> <p><b>Field Summary for Register <a href="#">PIC_ENABLE</a></b></p> <table border="1"> <thead> <tr> <th>Bit</th> <th>Field Name</th> <th>Access</th> <th>Reset Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>[1:0]</td> <td>enable</td> <td>"RW"</td> <td>0</td> <td>Interrupt Enable</td> </tr> </tbody> </table> <p><b>Field Summary for Register <a href="#">PIC_SET</a></b></p> <table border="1"> <thead> <tr> <th>Bit</th> <th>Field Name</th> <th>Access</th> <th>Reset Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>[1:0]</td> <td>set</td> <td>"WO"</td> <td>0</td> <td>Interrupt Set</td> </tr> </tbody> </table> <p><b>Field Summary for Register <a href="#">PIC_POL</a></b></p> <table border="1"> <thead> <tr> <th>Bit</th> <th>Field Name</th> <th>Access</th> <th>Reset Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>[1:0]</td> <td>pol</td> <td>"RW"</td> <td>0</td> <td>Interrupt Polarity</td> </tr> </tbody> </table> <p><b>Field Summary for Register <a href="#">TIMER_CNT_L</a></b></p> <table border="1"> <thead> <tr> <th>Bit</th> <th>Field Name</th> <th>Access</th> <th>Reset Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>[31:0]</td> <td>cnt_l</td> <td>"RW"</td> <td>0</td> <td>Lower 32 bits of 64-bit timer counter register</td> </tr> </tbody> </table> <p><b>Field Summary for Register <a href="#">TIMER_CNT_H</a></b></p> <table border="1"> <thead> <tr> <th>Bit</th> <th>Field Name</th> <th>Access</th> <th>Reset Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>[31:0]</td> <td>cnt_h</td> <td>"RW"</td> <td>0</td> <td>Higher 32 bits of 64-bit timer counter register</td> </tr> </tbody> </table> <p><b>Field Summary for Register <a href="#">TIMER_CMP_L</a></b></p> <table border="1"> <thead> <tr> <th>Bit</th> <th>Field Name</th> <th>Access</th> <th>Reset Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>[31:0]</td> <td>cmp_l</td> <td>"RW"</td> <td>0</td> <td>Lower 32 bits of 64-bit timer compare register</td> </tr> </tbody> </table>	Start Addr	End Addr	Actual Range	Register Width	Description	0xFFFF0000	0xFFFF07FF	0x00000800	32		Offset	Register Name	Access	Reset Value	Width	Description	0x0000	<a href="#">PIC_STATUS</a>	"RW"		2	Interrupt Status Register	0x0004	<a href="#">PIC_ENABLE</a>	"RW"		2	Interrupt Enable Register	0x0008	<a href="#">PIC_SET</a>	"WO"		2	Interrupt Set Register	0x000C	<a href="#">PIC_POL</a>	"RW"		2	Interrupt Polarity Register	0x0400	<a href="#">TIMER_CNT_L</a>	"RW"		32	Timer Counter Lower Register	0x0404	<a href="#">TIMER_CNT_H</a>	"RW"		32	Timer Counter Higher Register	0x0410	<a href="#">TIMER_CMP_L</a>	"RW"		32	Timer Compare Lower Register	0x0414	<a href="#">TIMER_CMP_H</a>	"RW"		32	Timer Compare Higher Register	Bit	Field Name	Access	Reset Value	Description	[1:0]	status	"RW"	0	Interrupt Status	Bit	Field Name	Access	Reset Value	Description	[1:0]	enable	"RW"	0	Interrupt Enable	Bit	Field Name	Access	Reset Value	Description	[1:0]	set	"WO"	0	Interrupt Set	Bit	Field Name	Access	Reset Value	Description	[1:0]	pol	"RW"	0	Interrupt Polarity	Bit	Field Name	Access	Reset Value	Description	[31:0]	cnt_l	"RW"	0	Lower 32 bits of 64-bit timer counter register	Bit	Field Name	Access	Reset Value	Description	[31:0]	cnt_h	"RW"	0	Higher 32 bits of 64-bit timer counter register	Bit	Field Name	Access	Reset Value	Description	[31:0]	cmp_l	"RW"	0	Lower 32 bits of 64-bit timer compare register
Start Addr	End Addr	Actual Range	Register Width	Description																																																																																																																																			
0xFFFF0000	0xFFFF07FF	0x00000800	32																																																																																																																																				
Offset	Register Name	Access	Reset Value	Width	Description																																																																																																																																		
0x0000	<a href="#">PIC_STATUS</a>	"RW"		2	Interrupt Status Register																																																																																																																																		
0x0004	<a href="#">PIC_ENABLE</a>	"RW"		2	Interrupt Enable Register																																																																																																																																		
0x0008	<a href="#">PIC_SET</a>	"WO"		2	Interrupt Set Register																																																																																																																																		
0x000C	<a href="#">PIC_POL</a>	"RW"		2	Interrupt Polarity Register																																																																																																																																		
0x0400	<a href="#">TIMER_CNT_L</a>	"RW"		32	Timer Counter Lower Register																																																																																																																																		
0x0404	<a href="#">TIMER_CNT_H</a>	"RW"		32	Timer Counter Higher Register																																																																																																																																		
0x0410	<a href="#">TIMER_CMP_L</a>	"RW"		32	Timer Compare Lower Register																																																																																																																																		
0x0414	<a href="#">TIMER_CMP_H</a>	"RW"		32	Timer Compare Higher Register																																																																																																																																		
Bit	Field Name	Access	Reset Value	Description																																																																																																																																			
[1:0]	status	"RW"	0	Interrupt Status																																																																																																																																			
Bit	Field Name	Access	Reset Value	Description																																																																																																																																			
[1:0]	enable	"RW"	0	Interrupt Enable																																																																																																																																			
Bit	Field Name	Access	Reset Value	Description																																																																																																																																			
[1:0]	set	"WO"	0	Interrupt Set																																																																																																																																			
Bit	Field Name	Access	Reset Value	Description																																																																																																																																			
[1:0]	pol	"RW"	0	Interrupt Polarity																																																																																																																																			
Bit	Field Name	Access	Reset Value	Description																																																																																																																																			
[31:0]	cnt_l	"RW"	0	Lower 32 bits of 64-bit timer counter register																																																																																																																																			
Bit	Field Name	Access	Reset Value	Description																																																																																																																																			
[31:0]	cnt_h	"RW"	0	Higher 32 bits of 64-bit timer counter register																																																																																																																																			
Bit	Field Name	Access	Reset Value	Description																																																																																																																																			
[31:0]	cmp_l	"RW"	0	Lower 32 bits of 64-bit timer compare register																																																																																																																																			

Figure 2.84. Memory Report of Each Instance

### 2.2.12. Launch Project in Diamond or Radiant Software

In a complete SoC project, a Diamond or Radiant project can be created upon a Propel Builder design. 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.

**Notes:**

- Different device families are supported and available in Radiant/Diamond software, 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
	LAV-AT Evaluation	LAV-AT

Software	Project Name	Part Number
Diamond	MachXO2	LCMXO2
	MachXO3D	LCMXO3D
	MachXO3L	LCMXO3DL
	MachXO3LF	LCMXO3DLF
	Mach-NX	LFMNX

- The installation path of Diamond or Radiant should be set using **Tools > Options > Directories**. See [Tools Options](#) section for more details. Or, Diamond or Radiant software cannot be invoked.

### 2.2.12.1. Launch Diamond Software

- Click the Diamond icon  from the Propel Builder toolbar. Diamond software is launched.

A `diamond_setup_template.tcl` is generated in project path when you click the icon and is automatically invoked in Diamond. Refer to the [Lattice Diamond Help](#) for the meaning and usage of the TCL commands.

In the meantime, there are also some files exported in project path, such as all RTL from IP, as well as the top-level module.

IP encryption is converted to blowfish when exported to Diamond software. There are two sets of files generated for each IP for Diamond. You can use the blowfish file in case you need to add it again later.

- Lattice Diamond is launched with a Diamond project generated for SoC at the background ([Figure 2.85](#)).

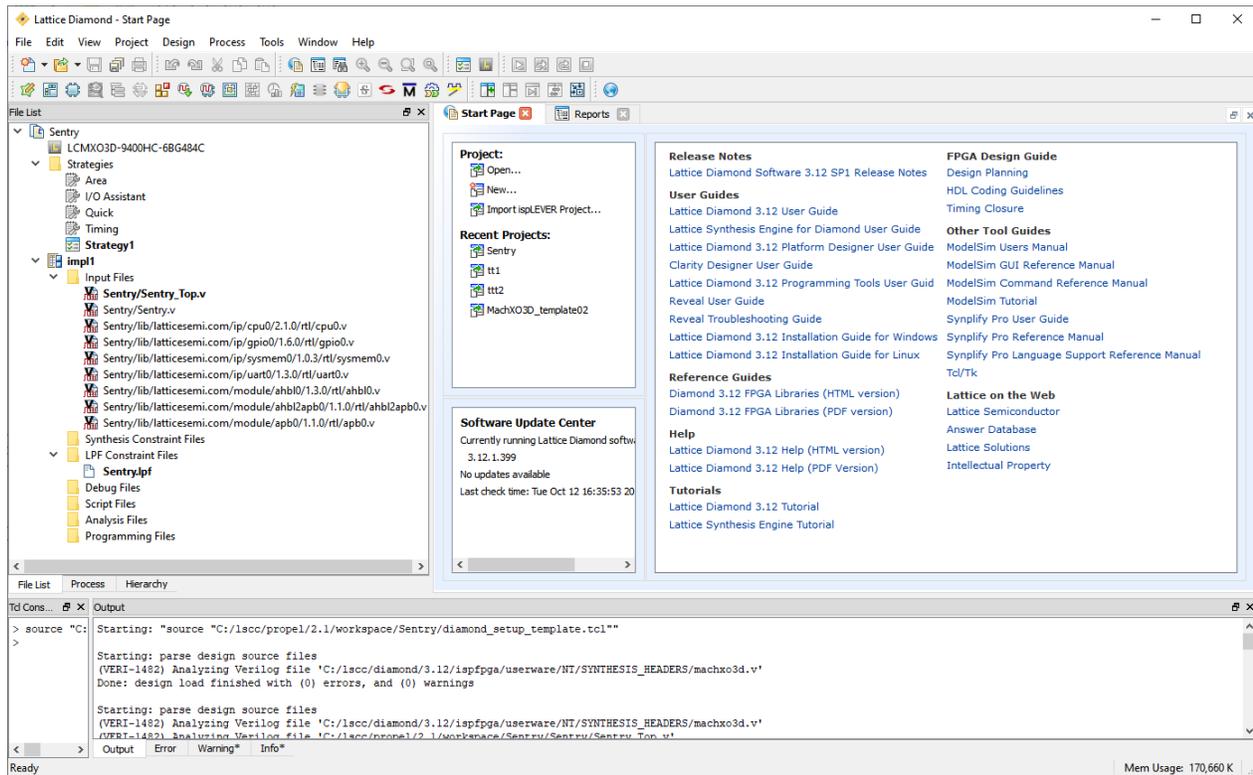


Figure 2.85. Diamond Project

- (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 2.86). Make sure at least one programming file (bitstream File or JEDEC file) is selected in the **Export Files** section. Available programming files can be different upon specific device included.

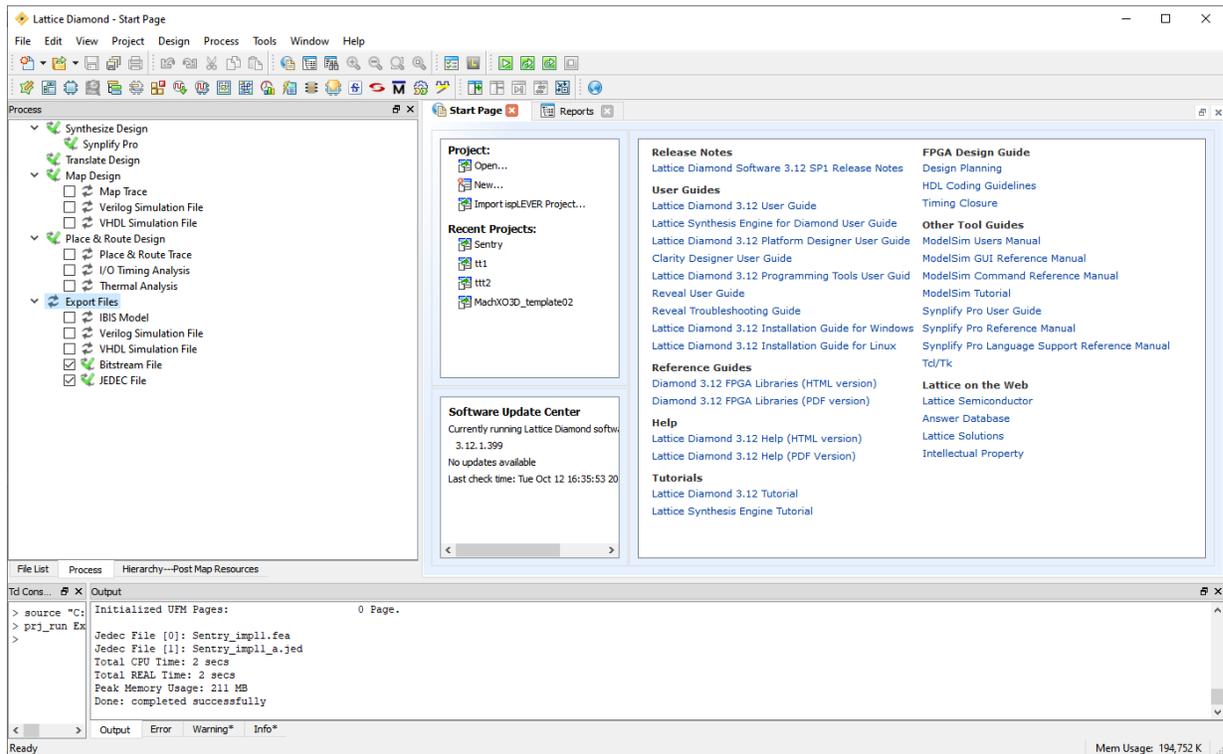


Figure 2.86. Generate Programming Files

- 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 [Lattice Diamond Help](#) for more details of the Diamond project. Re-launch the Diamond project, if the project settings are modified in Propel 2024.1 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. A radiant\_setup\_template.tcl is generated in project path when you click this icon and is automatically invoked in **Radiant**. Refer to the [Lattice Radiant Help](#) for meaning and usage of the TCL commands. In the meantime, there will also be some file exported in project path, like all RTL from IP, as well as the top-level module, and all files are listed in flist\_design.f.
2. Lattice Radiant is launched with a Radiant project generated for SoC at the background (Figure 2.87).

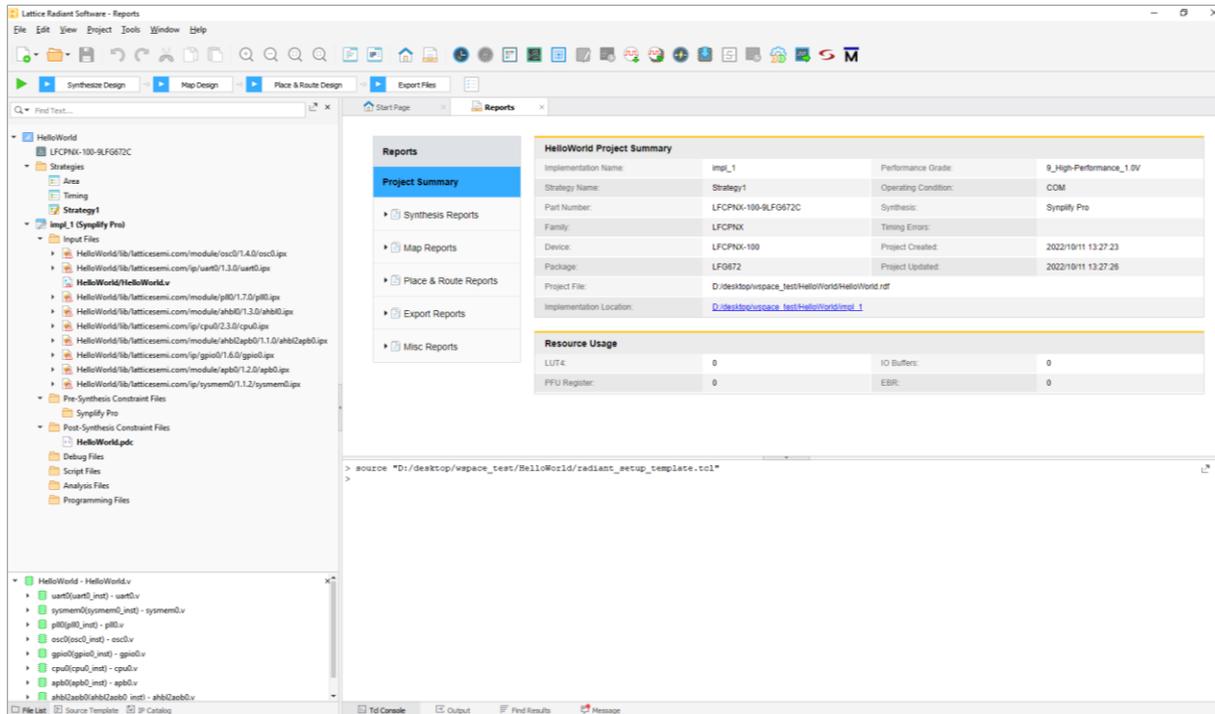


Figure 2.87. Radiant Project

- 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 [Lattice Radiant 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 2024.1. For project created before Propel Builder 2024.1, use a different method for exporting files (Figure 2.88). If you want to use ipx flow in these old projects, open them in Propel Builder 2024.1 and regenerate, then export to Radiant to change how it is exported to Radiant. (Figure 2.89).

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.

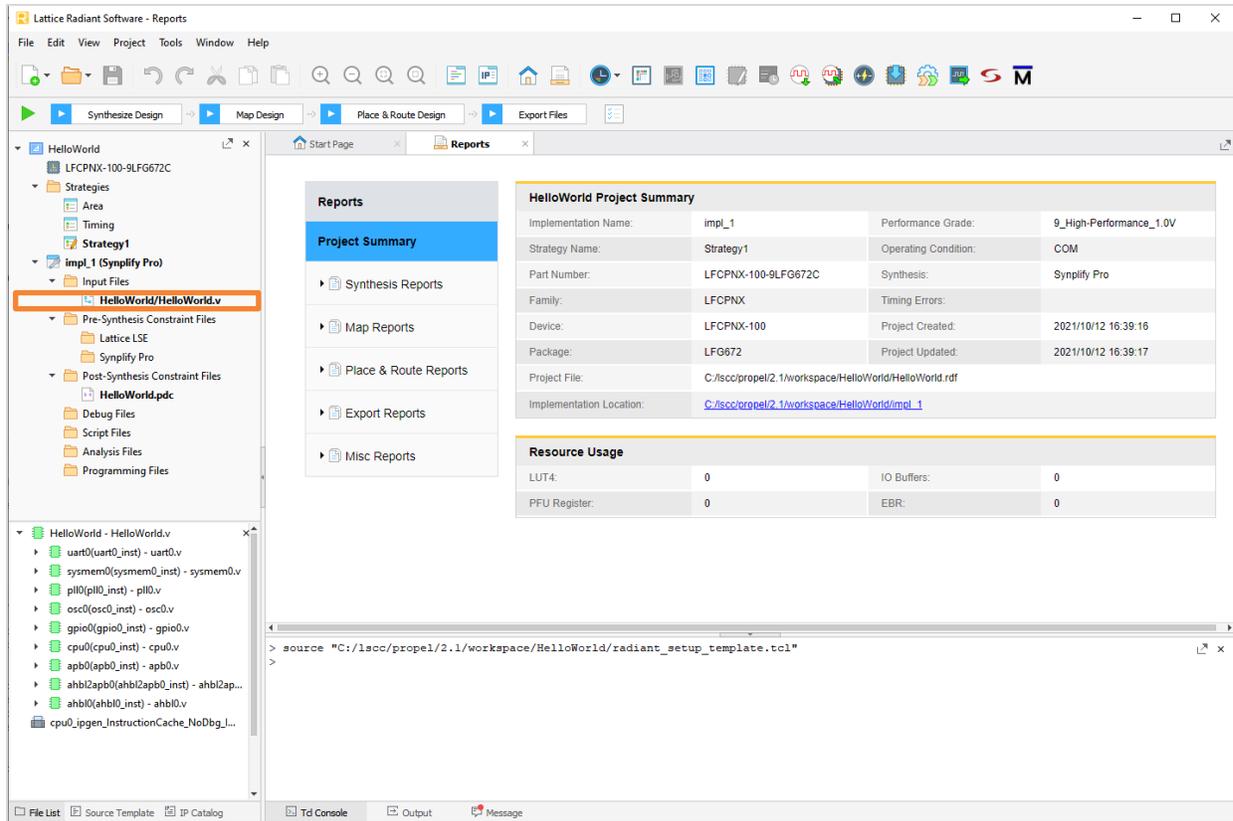


Figure 2.88. Radiant Project for Project before Propel Builder 2024.1

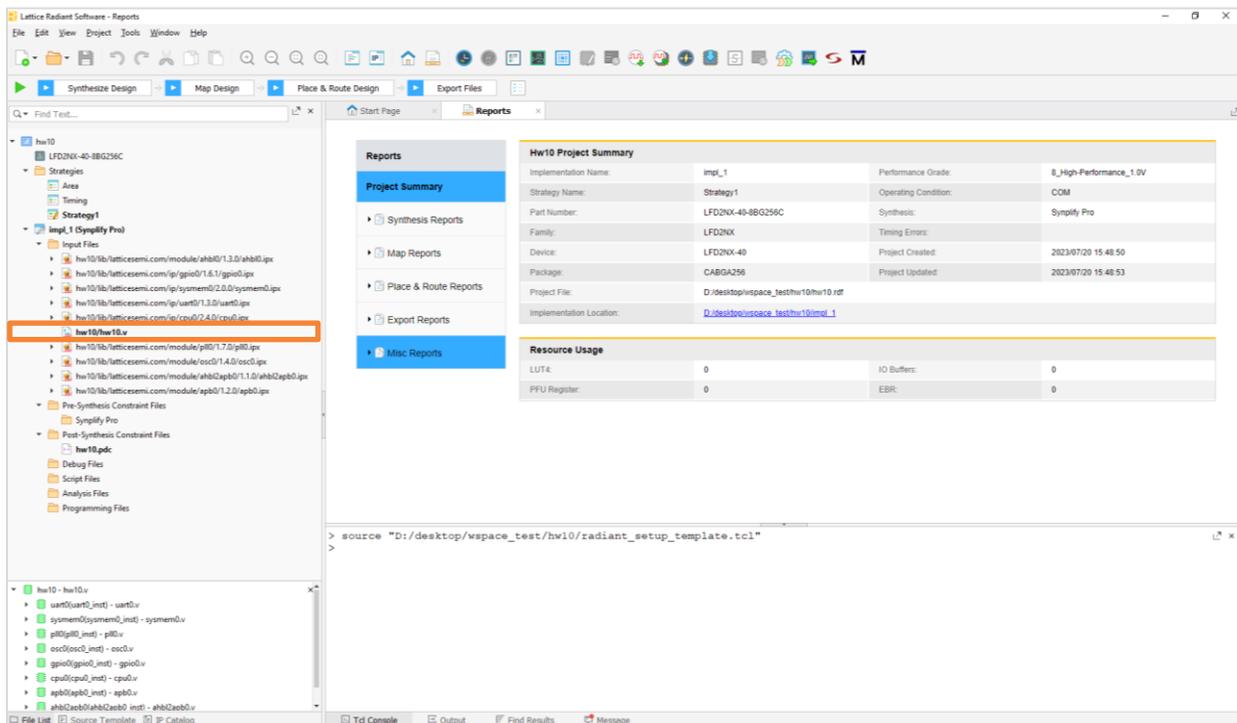


Figure 2.89. Radiant Project for Project Since Propel Builder 2024.1

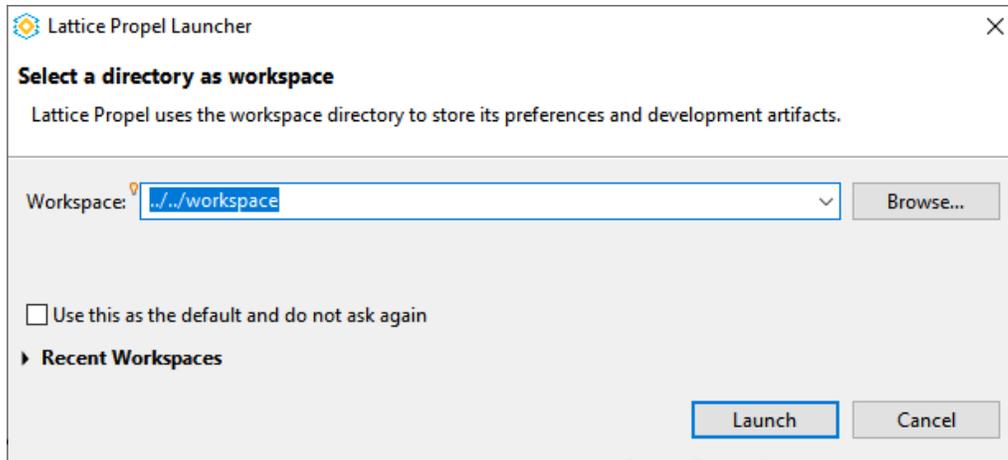
### 2.2.13. Launching SDK and Generate Mem File

After a SoC design is completed, Propel SDK can be launched in Propel Builder GUI for further software development in C project. For information about Propel SDK, refer to [Lattice Propel 2024.1 SDK User Guide \(FPGA-UG-02211\)](#).

Here is an example on how to generate a C project memory in Propel SDK for simulation use in verification project of Propel Builder. See [Generating Simulation Environment](#) section for more details on simulation.

For a template SoC project, you can launch SDK directly by clicking **Run Propel** icon from Propel Builder toolbar if a template SoC project is created.

1. Click the **Run Propel** icon  from Propel Builder GUI Toolbar. Lattice Propel Launcher ([Figure 2.90](#)) opens.



**Figure 2.90. Lattice Propel Launcher Wizard**

2. Click **Launch**. Propel SDK GUI opens. This C Project wizard is for loading the system and the board support package (BSP) to create a C/C++ project ([Figure 2.91](#)).  
Enter project name in the **Project Name** field, and then Click **Next**.

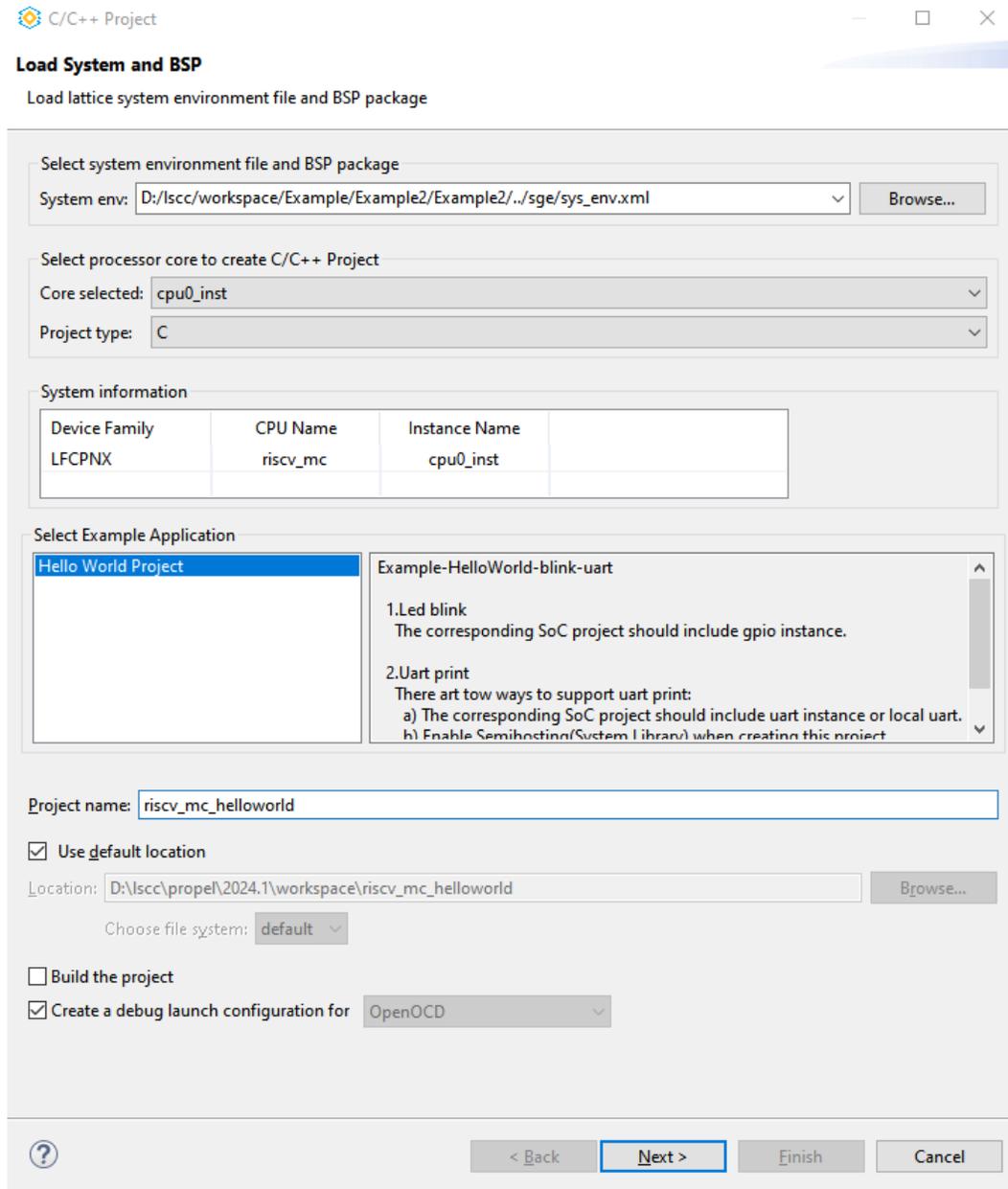
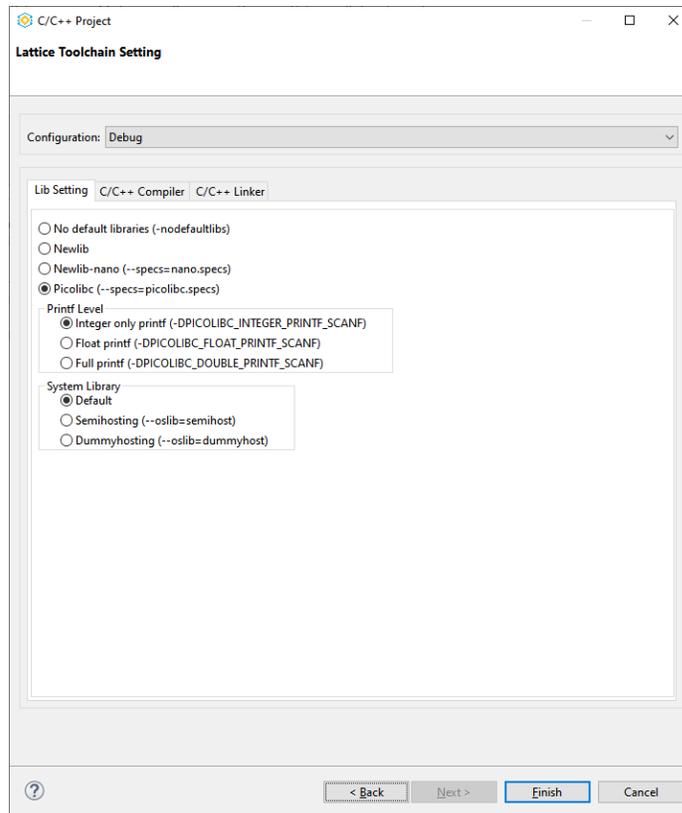


Figure 2.91. Propel SDK GUI and C/C++ Project Wizard

3. Click **Next**. The C/C++ project dialog (Figure 2.92) opens.



**Figure 2.92. 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.
5. Generate mem file in Propel SDK for simulation use:  
To use simulation, first we need to generate the initialization file of System Memory instance (mem file) in Propel SDK for simulation program to run on in verification project of Propel Builder.
6. In Propel SDK, right click on Project and then click **Build Project** (Figure 2.106). A mem file is generated in **Debug** folder (Figure 2.94).

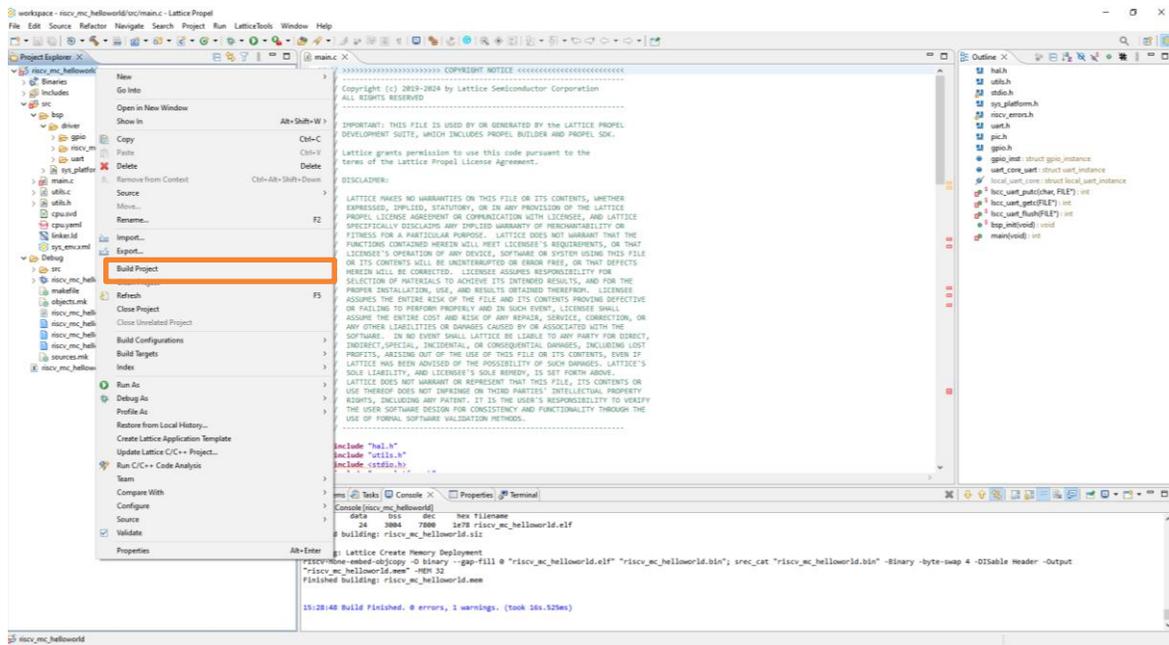


Figure 2.93. Build C/C++ Project

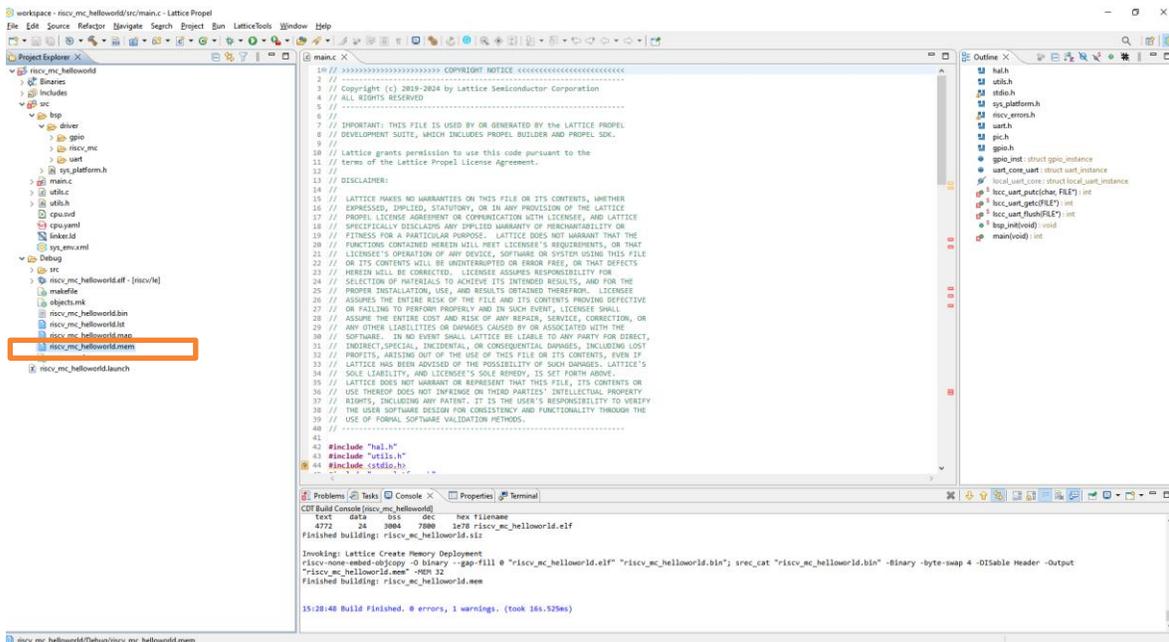


Figure 2.94. Mem File for Simulation

Right click on the mem file and choose **Show in > System Explorer** to get this file (Figure 2.95).

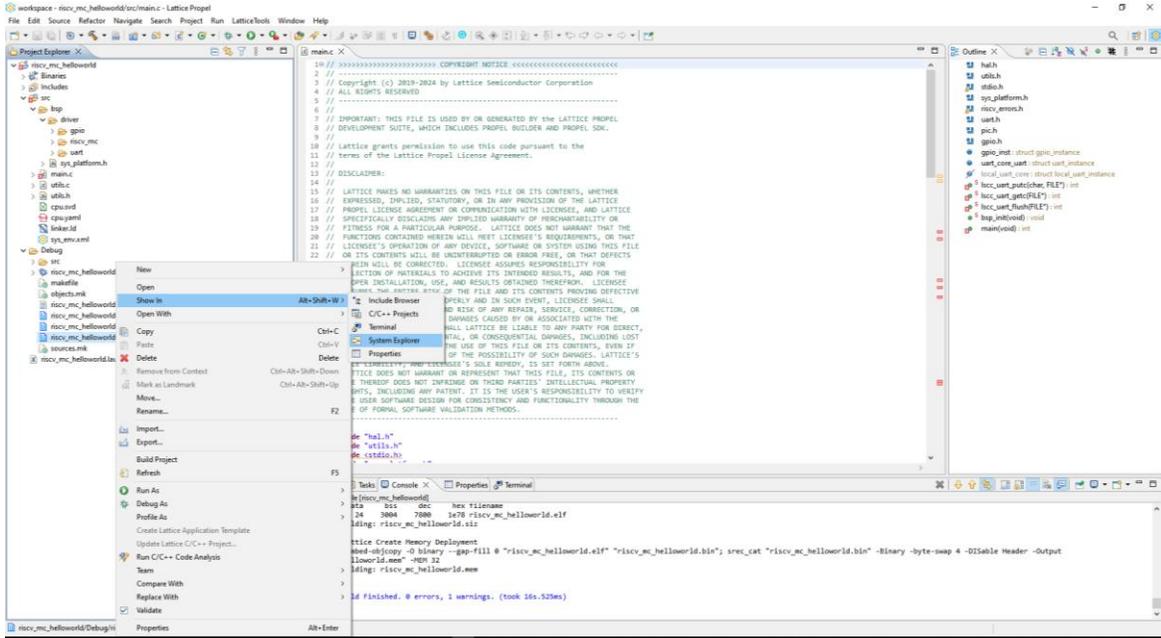


Figure 2.95. Mem File Location

7. Initialize memory for simulation:

After generating mem file in Propel SDK, you need to initialize memory for System Memory IP in Propel Builder. In Propel Builder schematic view, **double-click** on system memory IP to open **Module/IP Block Wizard**. Click on **'none'** in the Initialization File area and then click on **'...'** to select mem file from project (Figure 2.96).

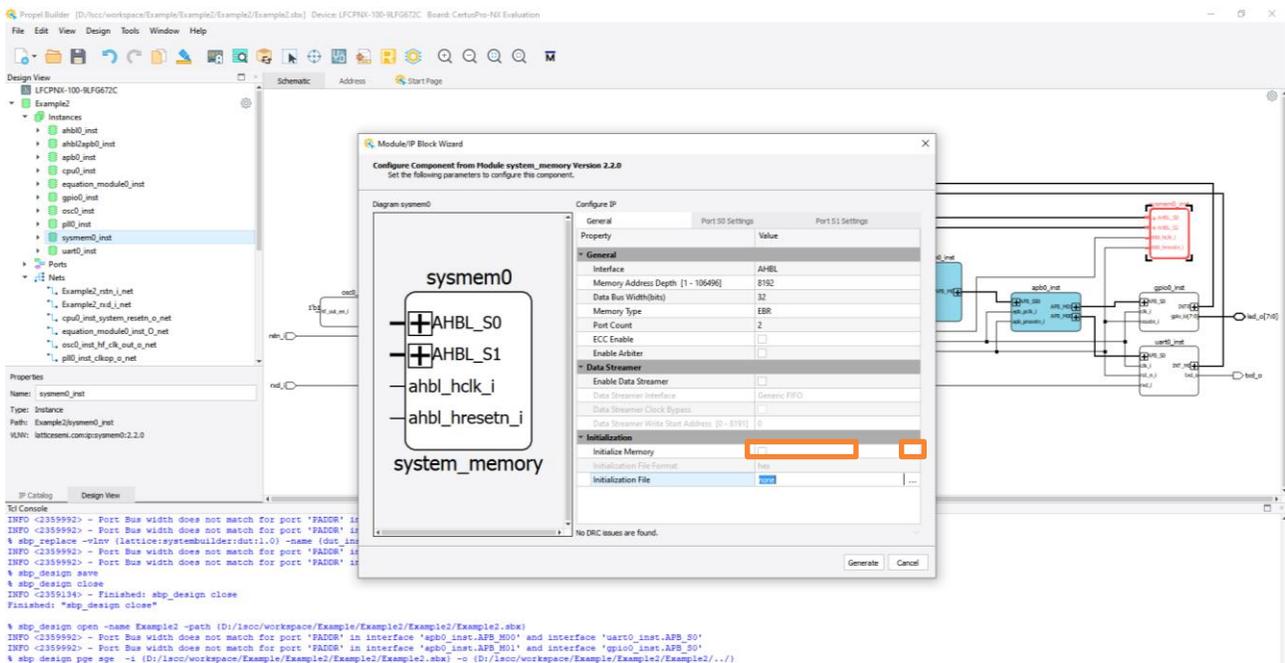
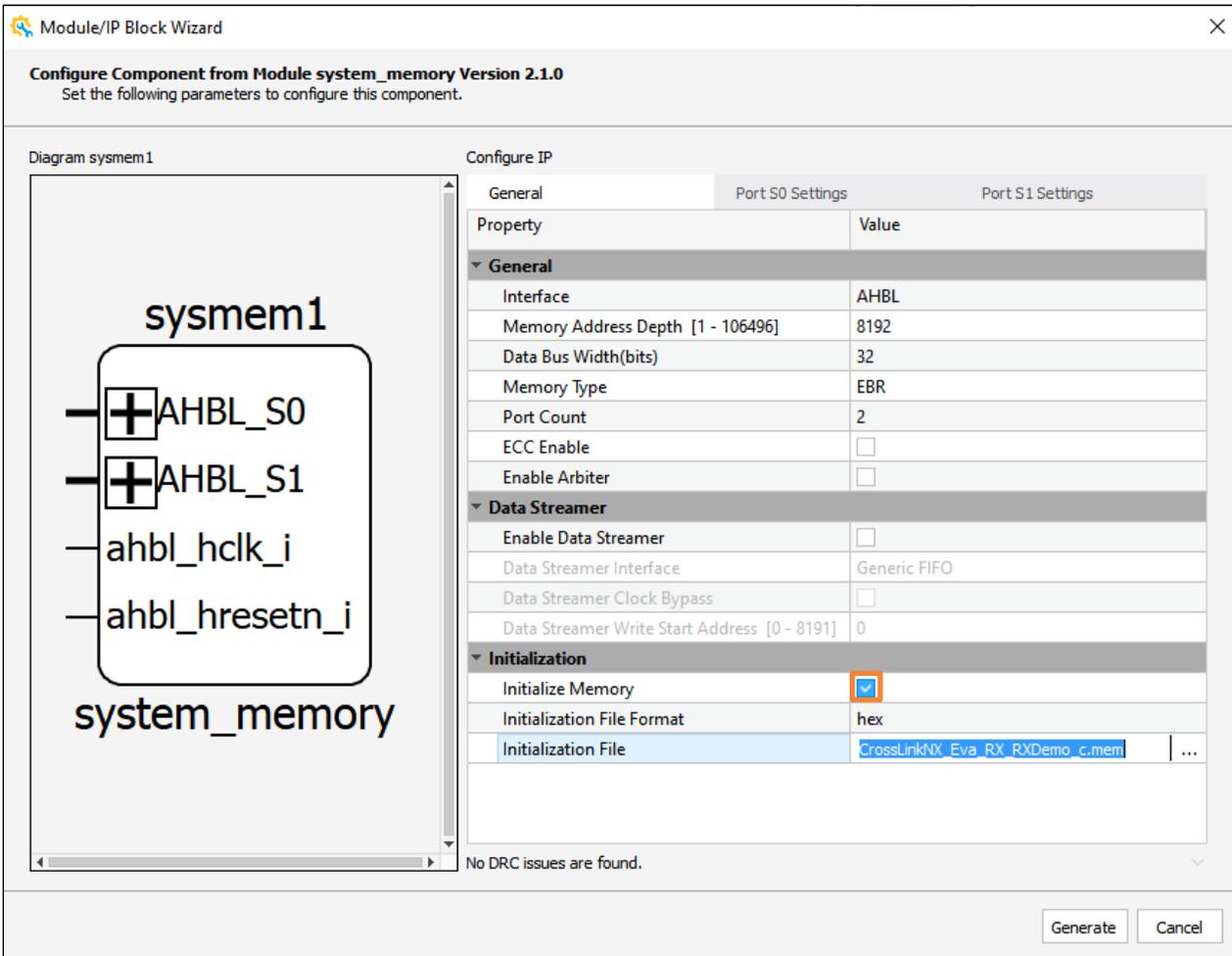


Figure 2.96. Initialize Memory for System Memory Module (1)

After the mem file is selected, check **Initialize Memory** (Figure 2.97).



**Figure 2.97. Initialize Memory for System Memory Module (2)**

For a SoC design, if the initialization file of System Memory instance (mem file) is built again in C project, system memory instance should always be re-configured by double clicking on it in GUI .

Or, use the ECO flow to update the information.

ECO Editor in Radiant enables you to safely make changes to an implemented design without having to rerun the entire process flow. ECOs are requests for small changes to be made to your design after it has been placed and routed. Choose **Tools > ECO Editor** or click the **ECO Editor** button on the toolbar to open the ECO Editor. Choose the right memory component and upload the memory file (Figure 2.98).

Refer to the [Lattice Diamond Help/Lattice Radiant Help](#) for more info on how to use an ECO flow.

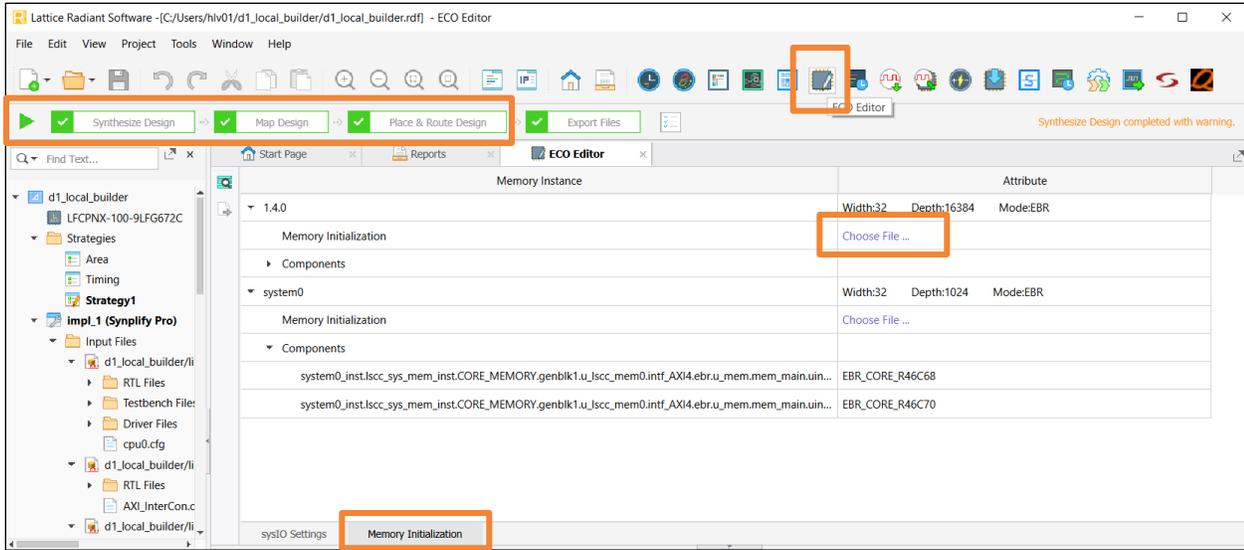


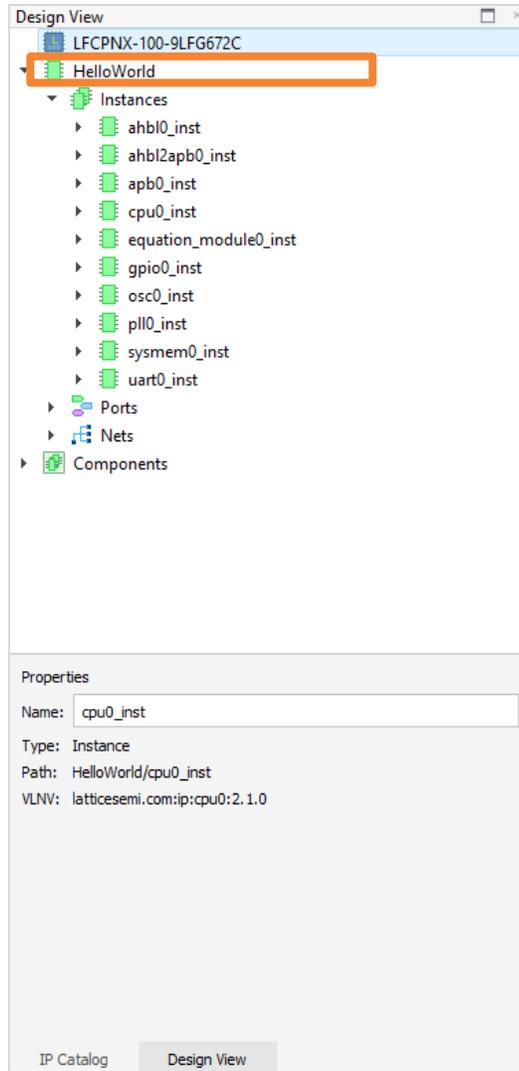
Figure 2.98. ECO flow in Radiant

## 2.2.14. Others

### 2.2.14.1. Modifying the Project Settings

Propel 2024.1 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 2.99), the Modify Device Info wizard pops up (Figure 2.100).



**Figure 2.99. 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 2.100). Use the drop-down menu to change the package, speed and operating condition in **Package, Speed, Operating Condition** filed.

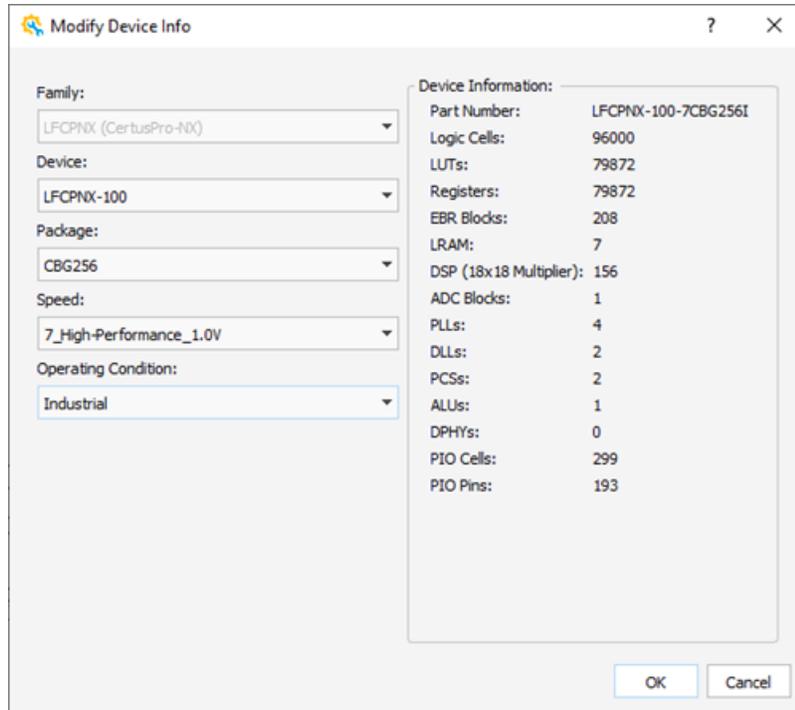


Figure 2.100. Modify Device Info

3. Click **OK**. The System Builder dialog (Figure 2.101) pops up showing a message of configuring all IPs.

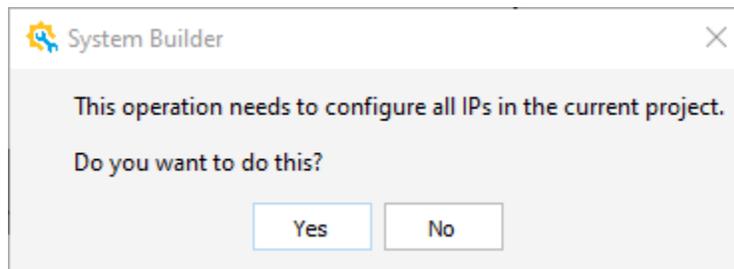
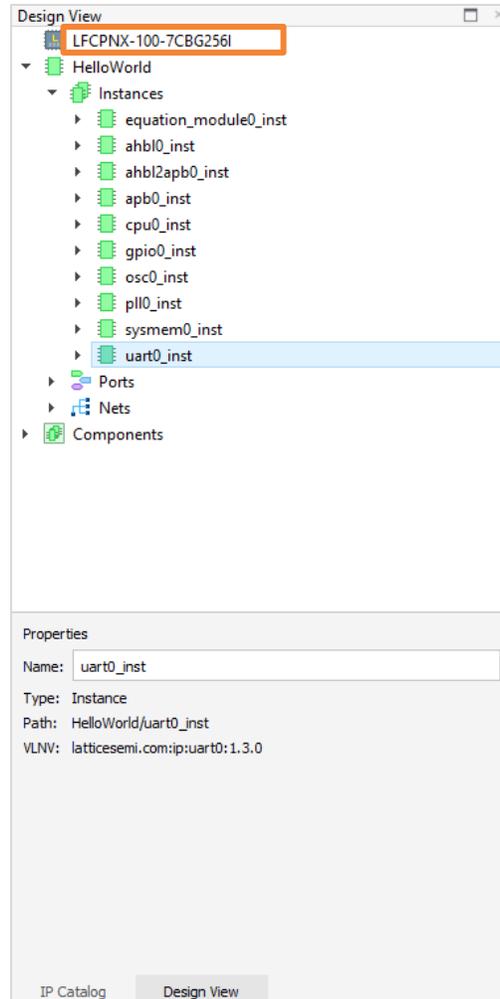


Figure 2.101. System Builder Dialog

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 2.102).



**Figure 2.102. Design View of Device Part Number**

### 2.2.14.2. Undo/Redo

Propel 2024.1 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 2024.1 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 2.103).

- Themes

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

**Note:** A restart of Propel Builder is required to take effect of the theme change (Figure 2.103).

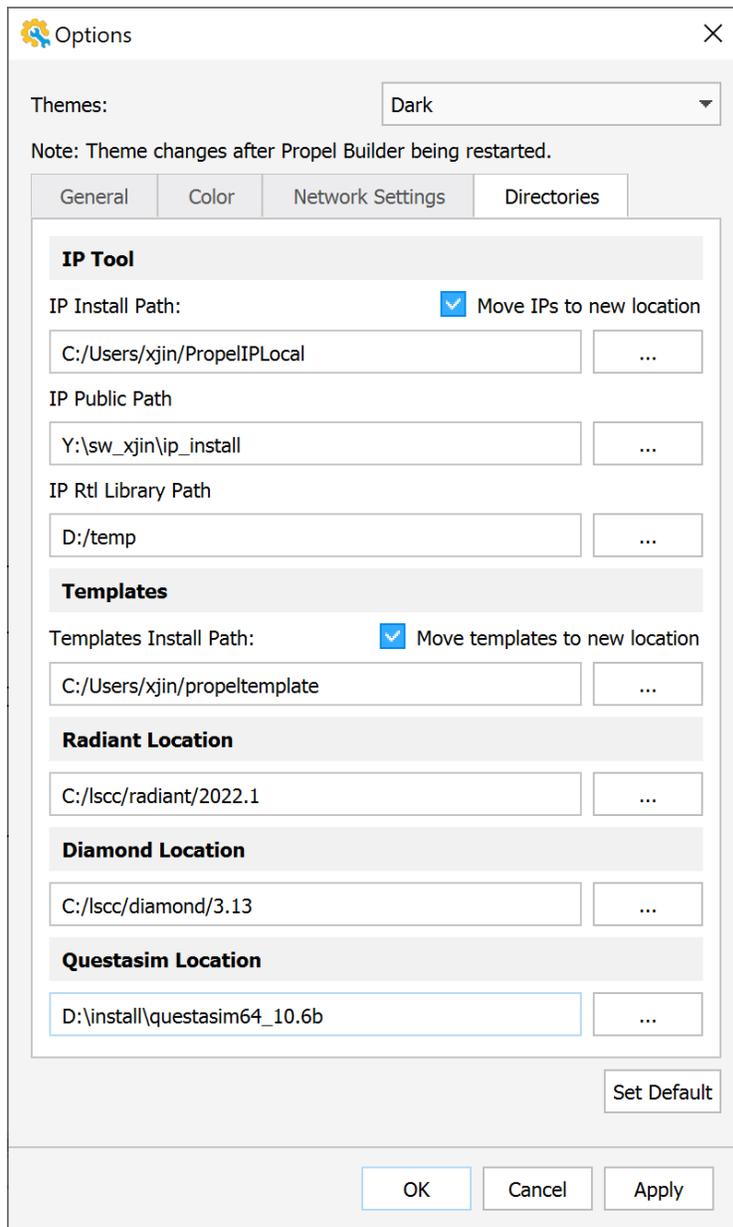
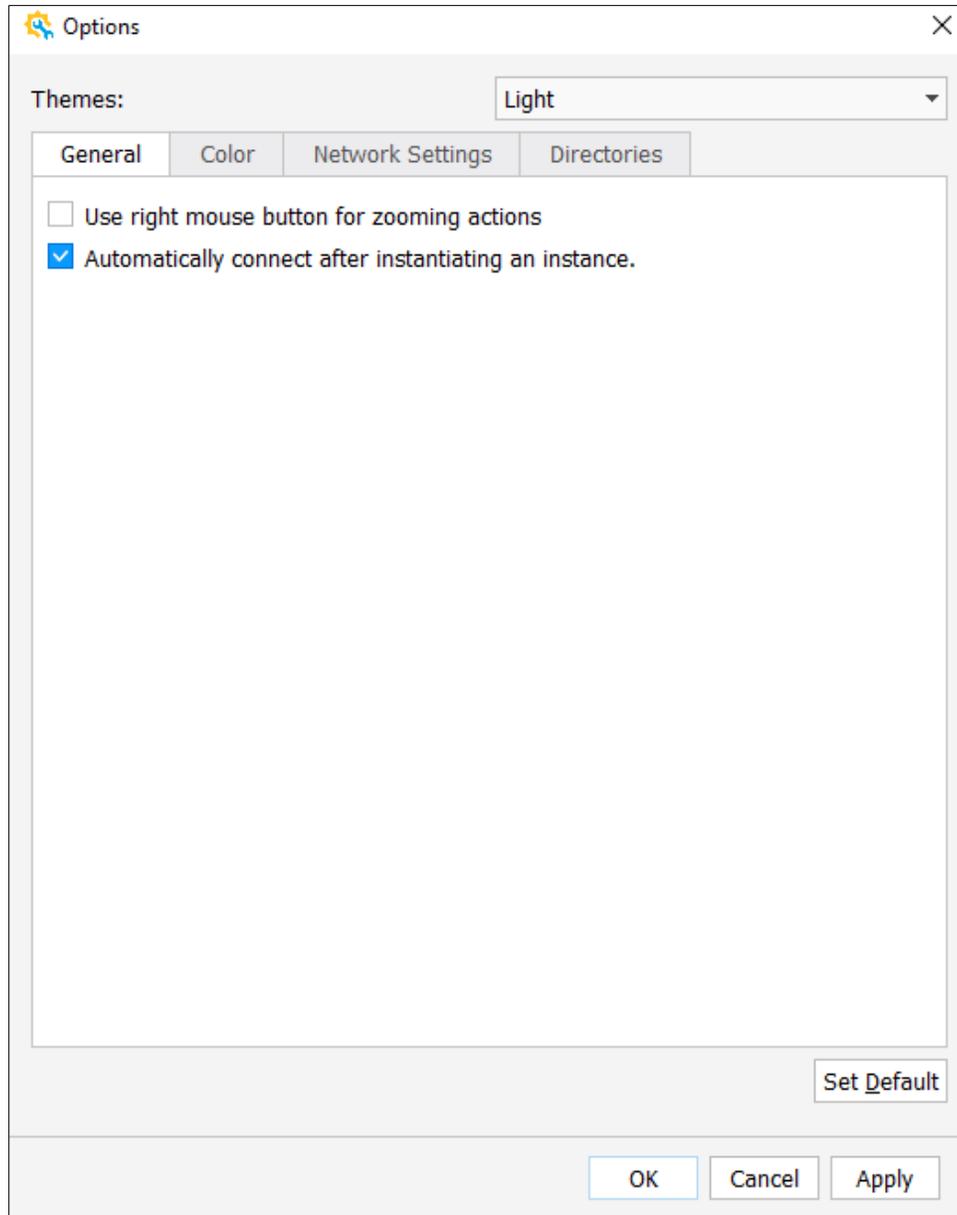


Figure 2.103. Theme Change

- General (Figure 2.104)

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 2.104. 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 2.105](#)), an addition dialog box pops up with some suggested Clock/Reset connection for the ports on this IP ([Figure 2.106](#)). Check/Uncheck to make the desired connections.

For functional use, refer to [Connect Pins by Auto Connect](#) for more details.

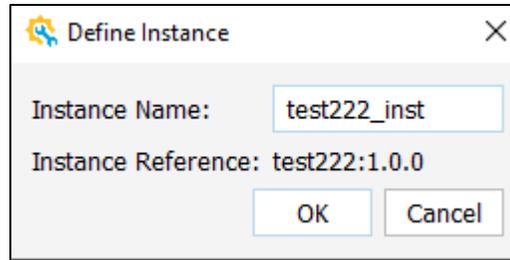


Figure 2.105. Define Instance Dialog

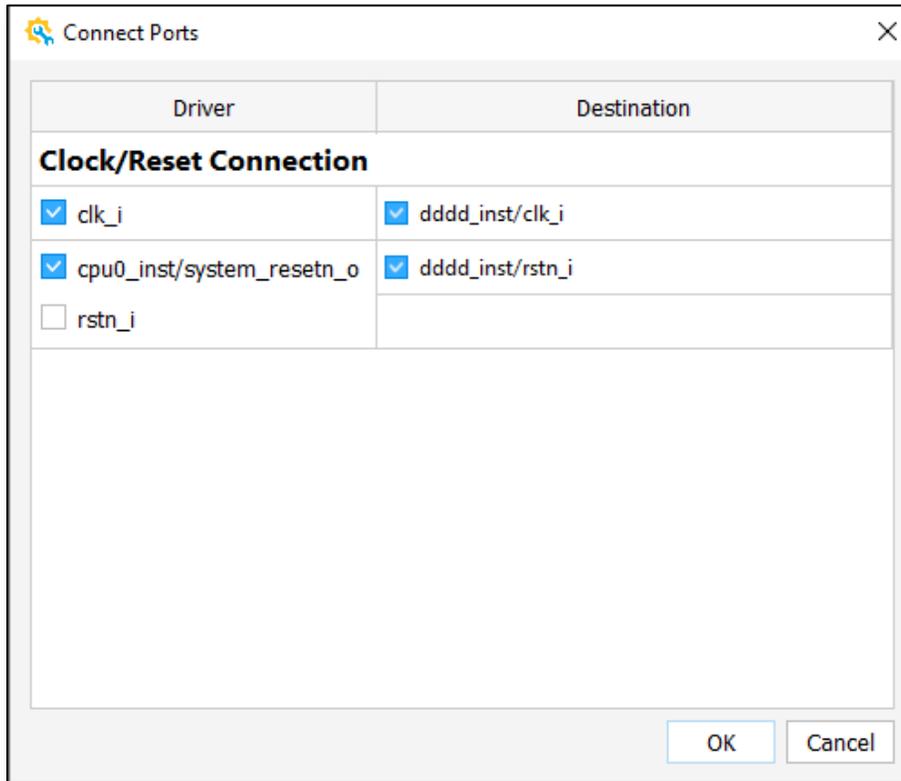


Figure 2.106. Connect Ports Dialog

- Color (Figure 2.107)

User can customize color for elements in Propel Builder, such as: the color of **Error/Warning/Info Message**, schematic background, highlight.

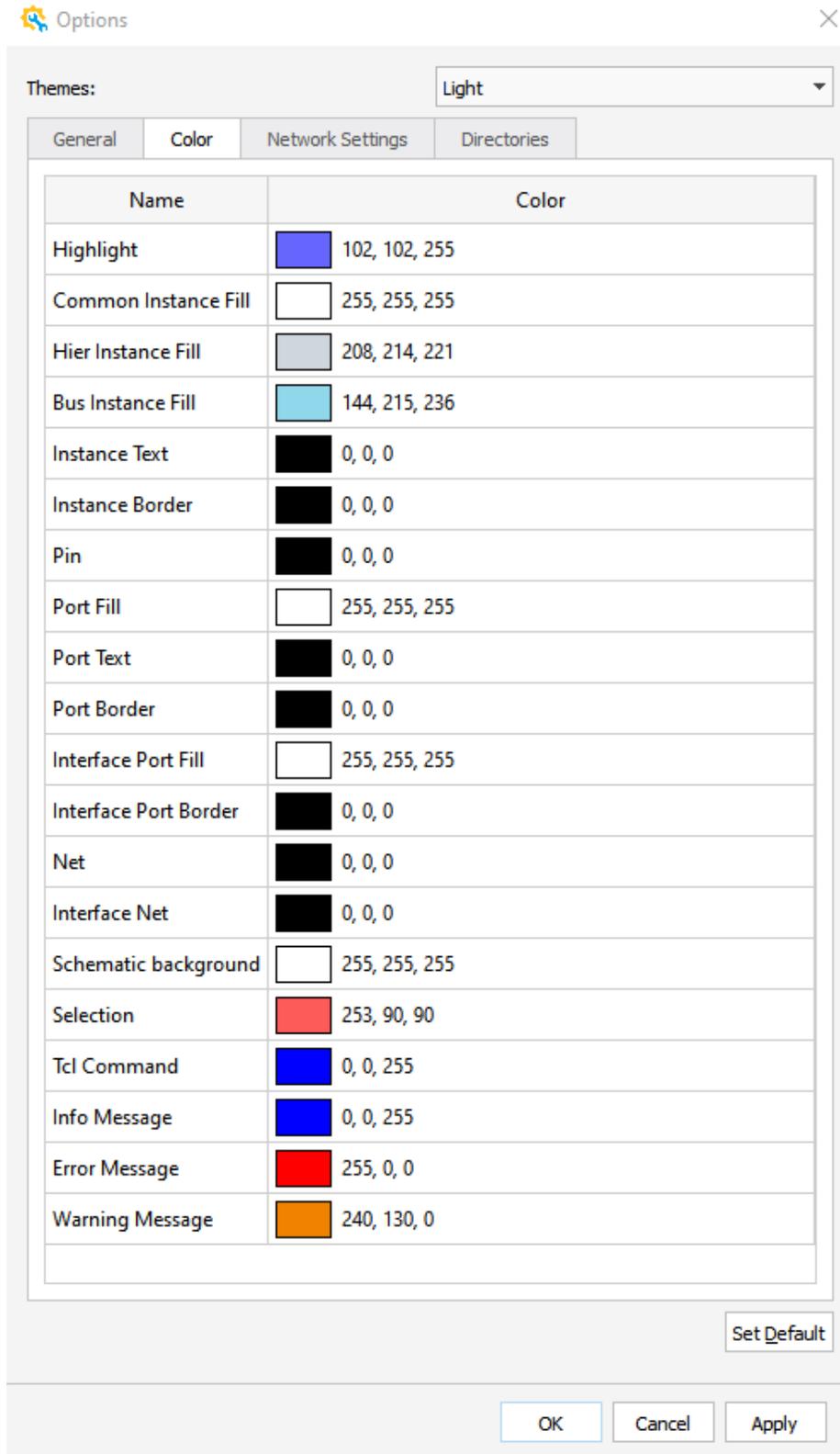
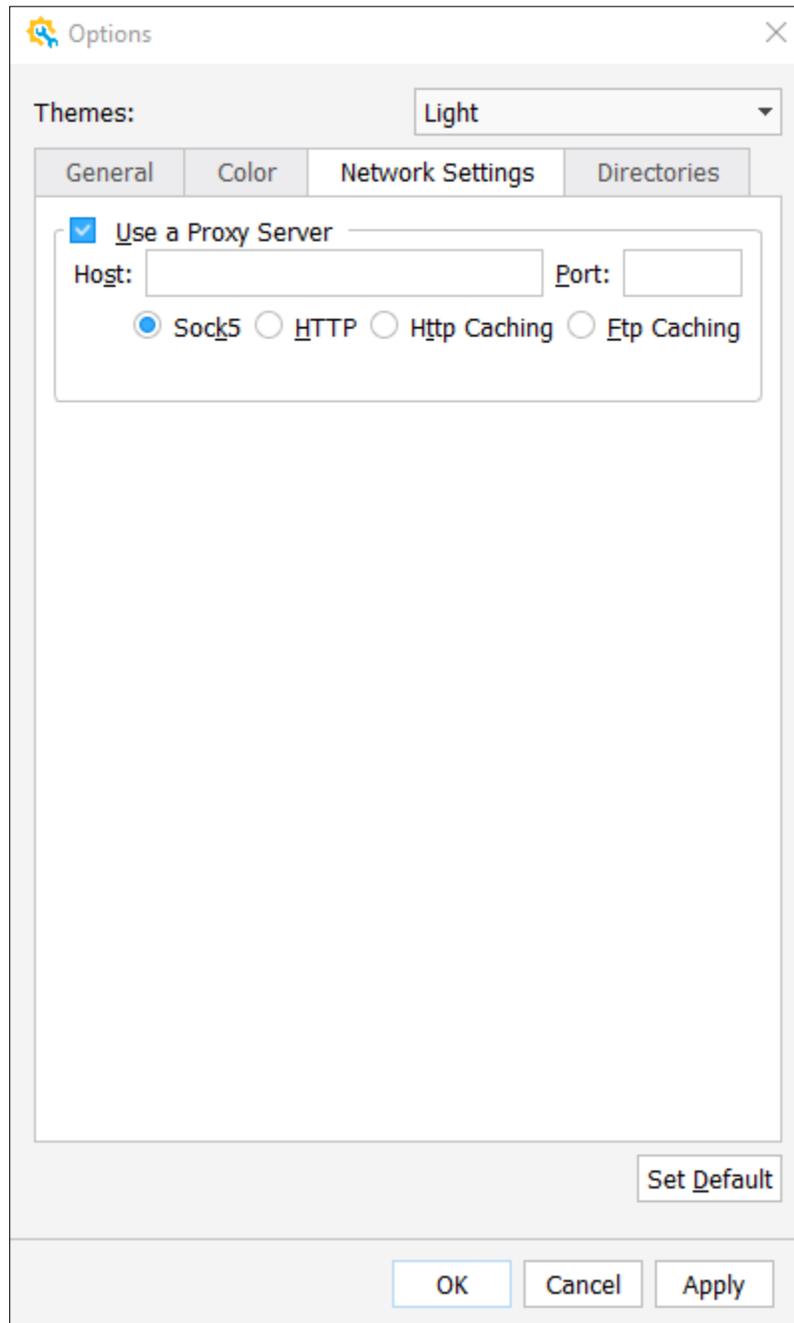


Figure 2.107. Color

- Network Settings (Figure 2.108)

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



**Figure 2.108. Network Settings**

- Directories (Figure 2.109)

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

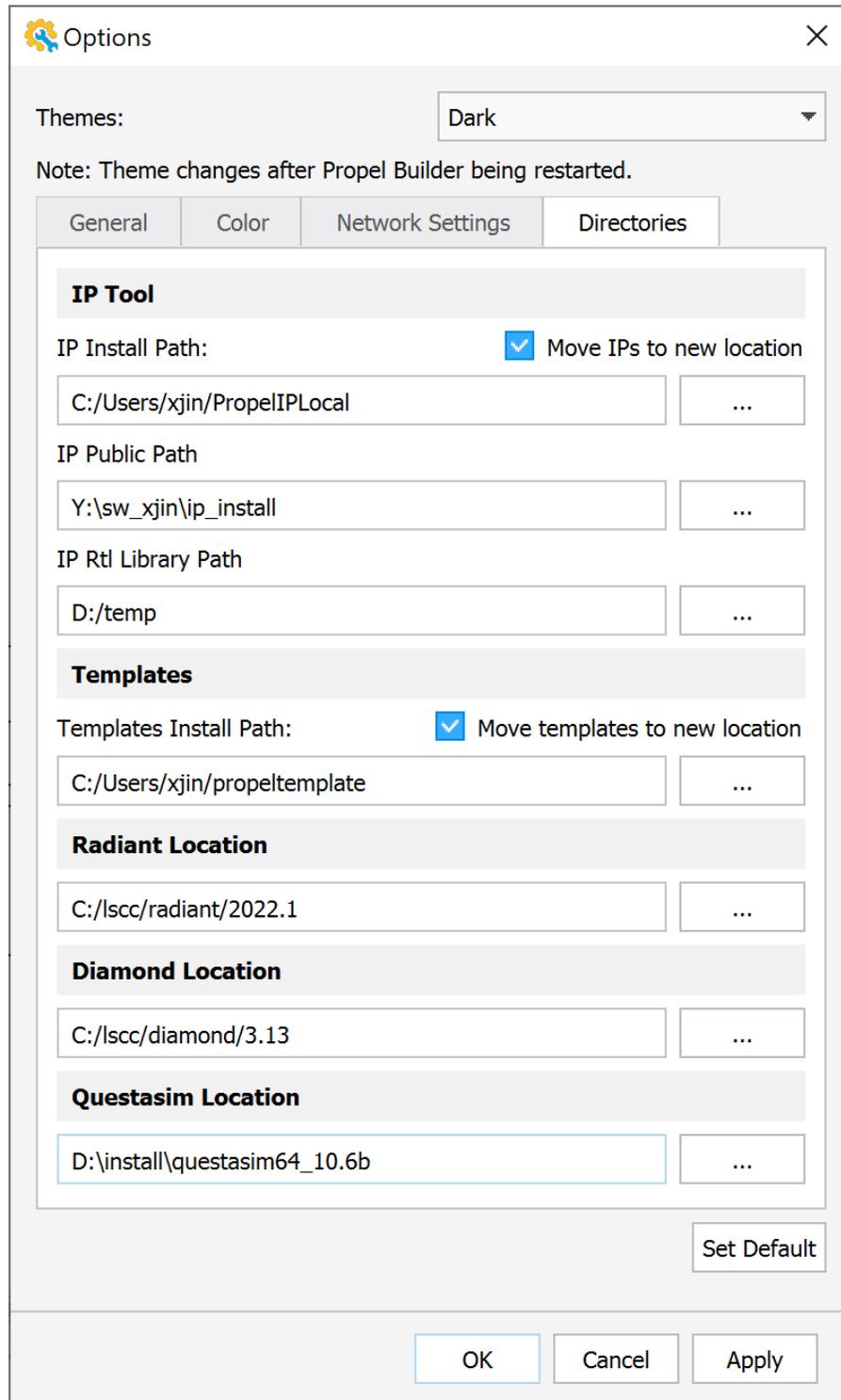


Figure 2.109. Directories

IP Tool (Figure 2.109):

- IP Installed Path (Figure 2.109): 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 2.110):

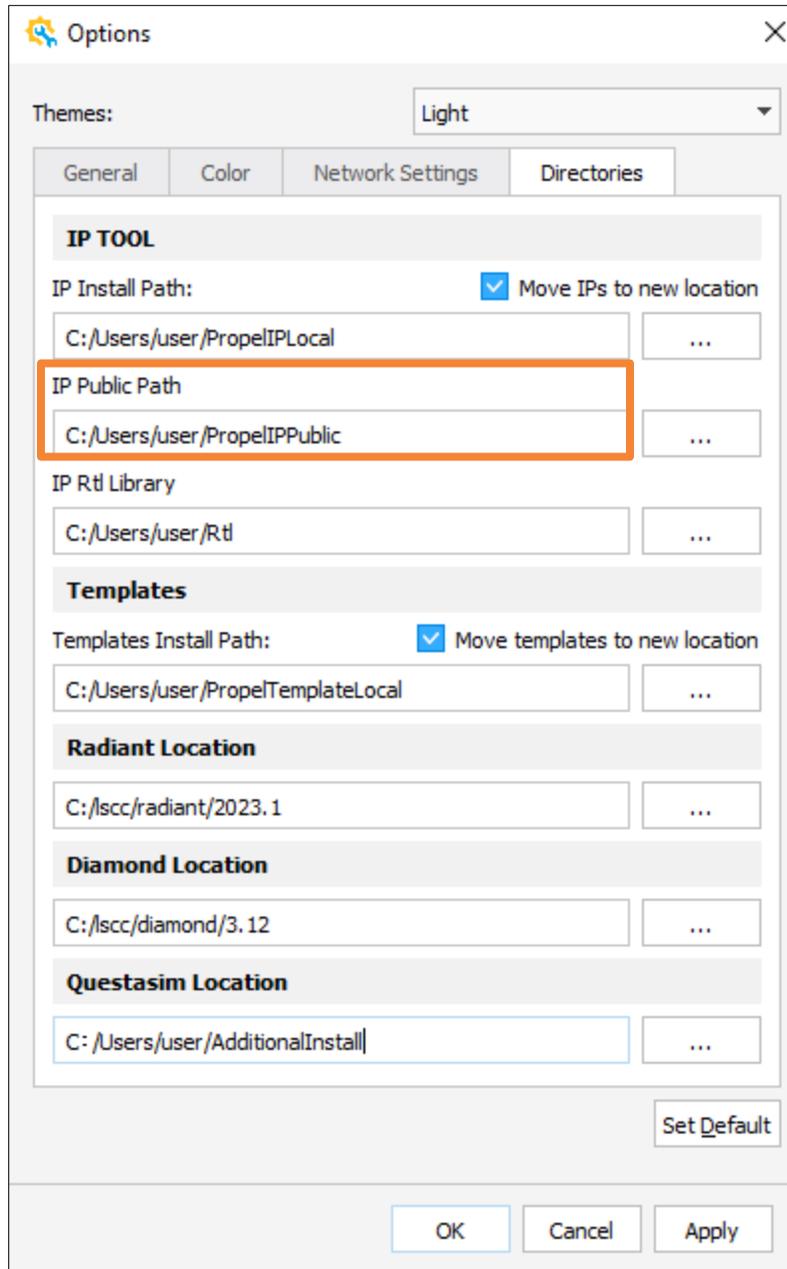


Figure 2.110. IP Public Path

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 in **IP catalog**.
- 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 2.111).

**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 2.112).

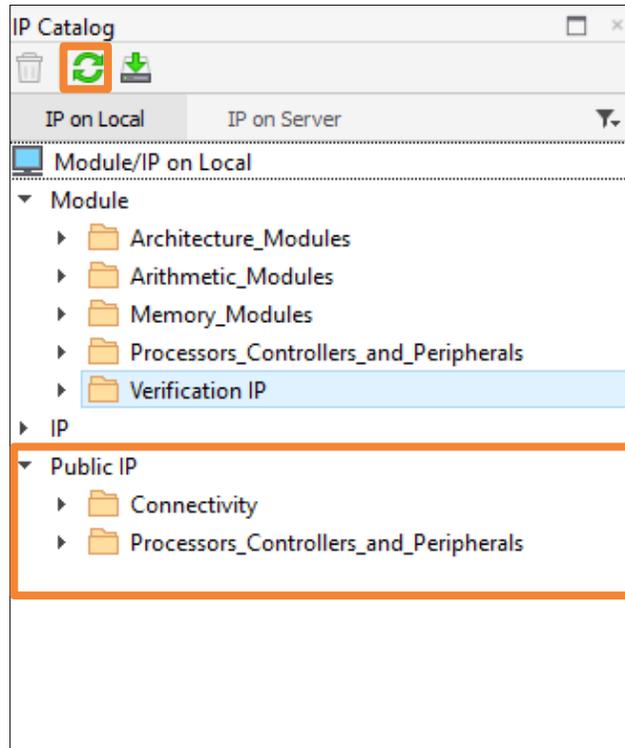


Figure 2.111. IP Catalog

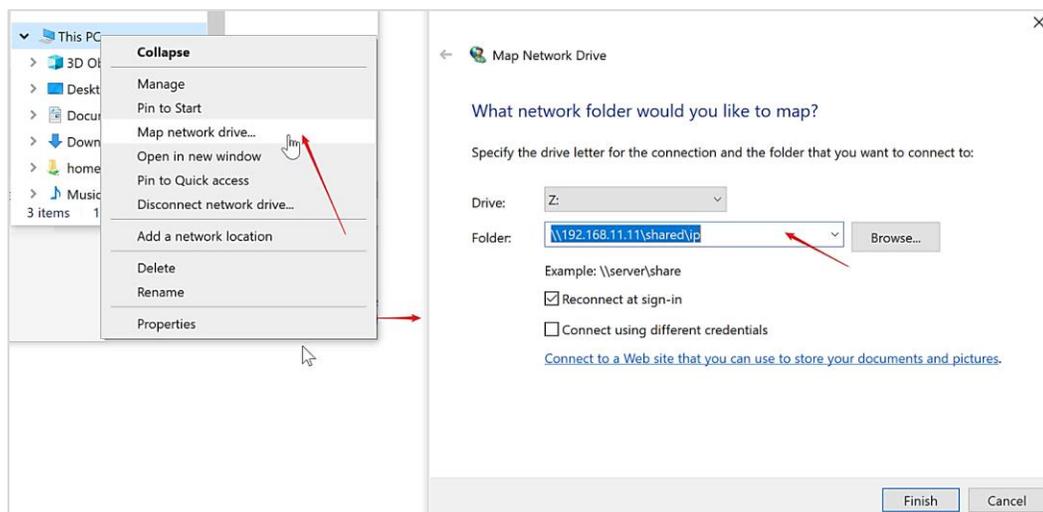


Figure 2.112. Map Network Device

- IP RTL Library (Figure 2.109): Folders of the IP RTL library. Refer to [Lattice Propel 2024.1 IP Package User Guide \(FPGA-UG-02213\)](#) for more details of the IP RTL Library.

Templates (Figure 2.109):

- Templates Install Path: Folders where the templates installed. See the [Define Custom Template](#) section for more details.
  - Radiant Location (Figure 2.109): Folder where Radiant software installed.
  - Diamond Location (Figure 2.109): Folder where Diamond software installed.
  - Questasim Location (Figure 2.109): Folder where Questasim software installed (see [Launching Simulation](#) section for more information).

## 2.3. Verification Project 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 2.113).

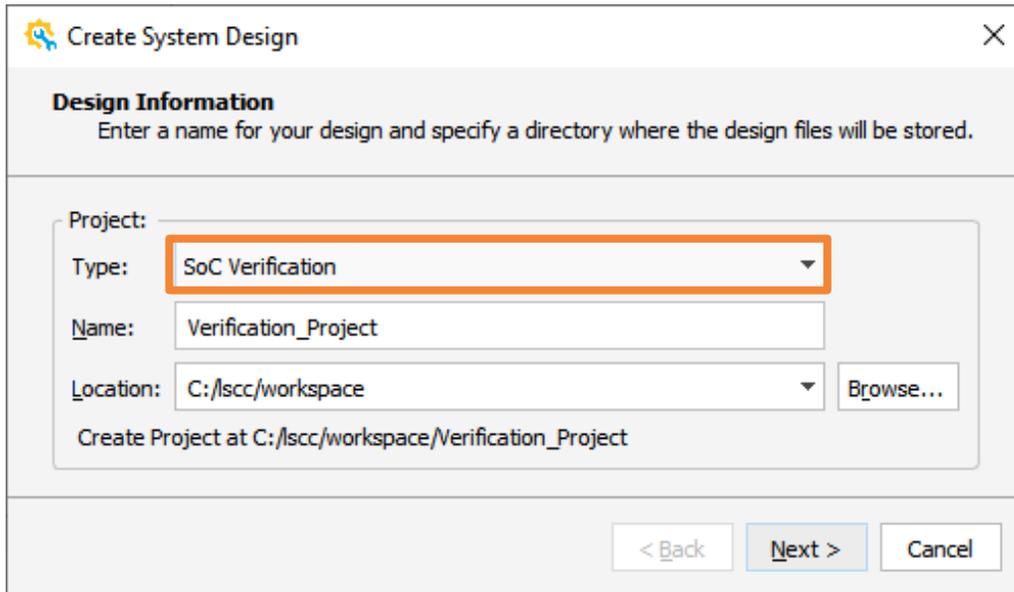


Figure 2.113. Create System Design – Design Information Wizard

2. Choose SoC Verification from the **Type** field.
3. Enter a project name in the **Name** field.
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 2.114](#).

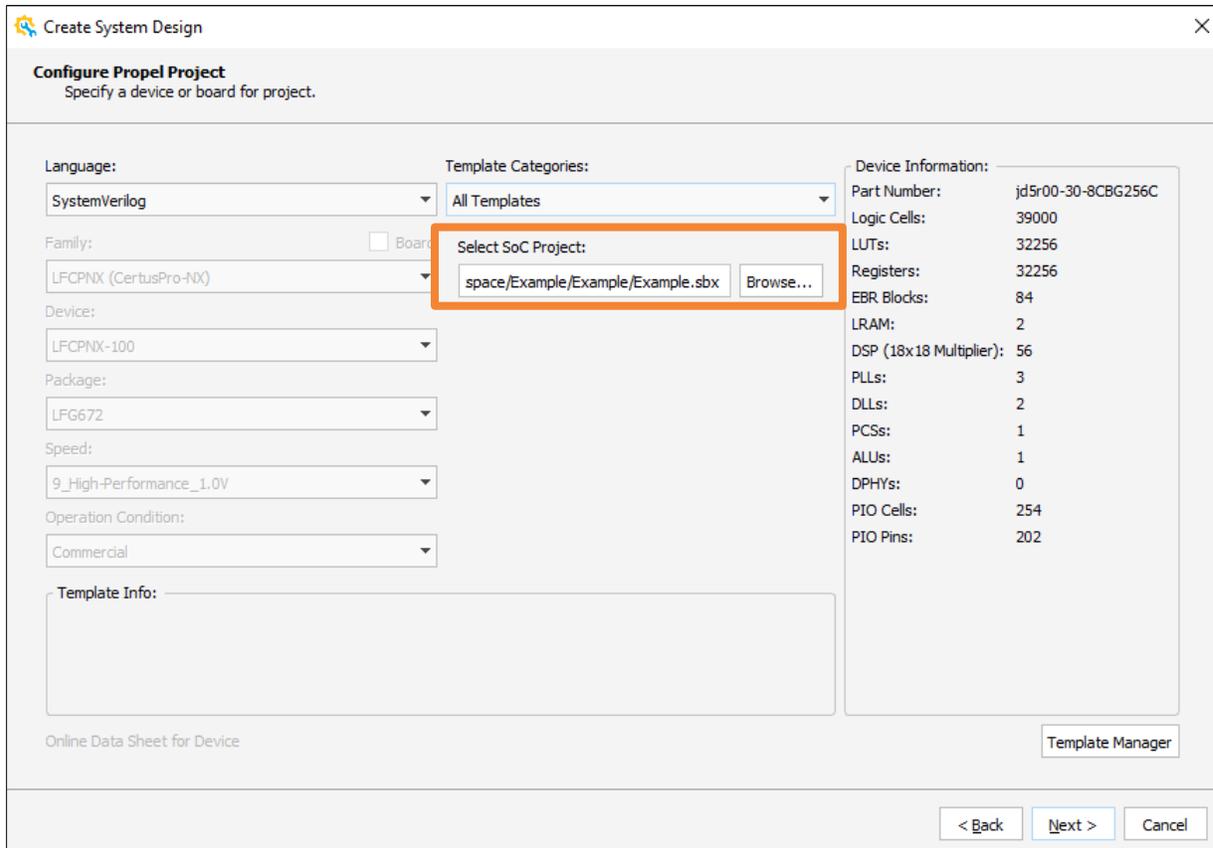


Figure 2.114. Create System Design – Propel Project Configure Wizard

- (Optional) The default Hardware Description Language (HDL) is displayed in the Language area. Use the drop-down menu to change the default language.
- Select an existing SBX design by using **Browse** to choose an SBX file.
- Click **Next**. The Project Information wizard opens.
- Click **Finish**. The dut\_inst of SoC project is shown in the Schematic view (Figure 2.115).

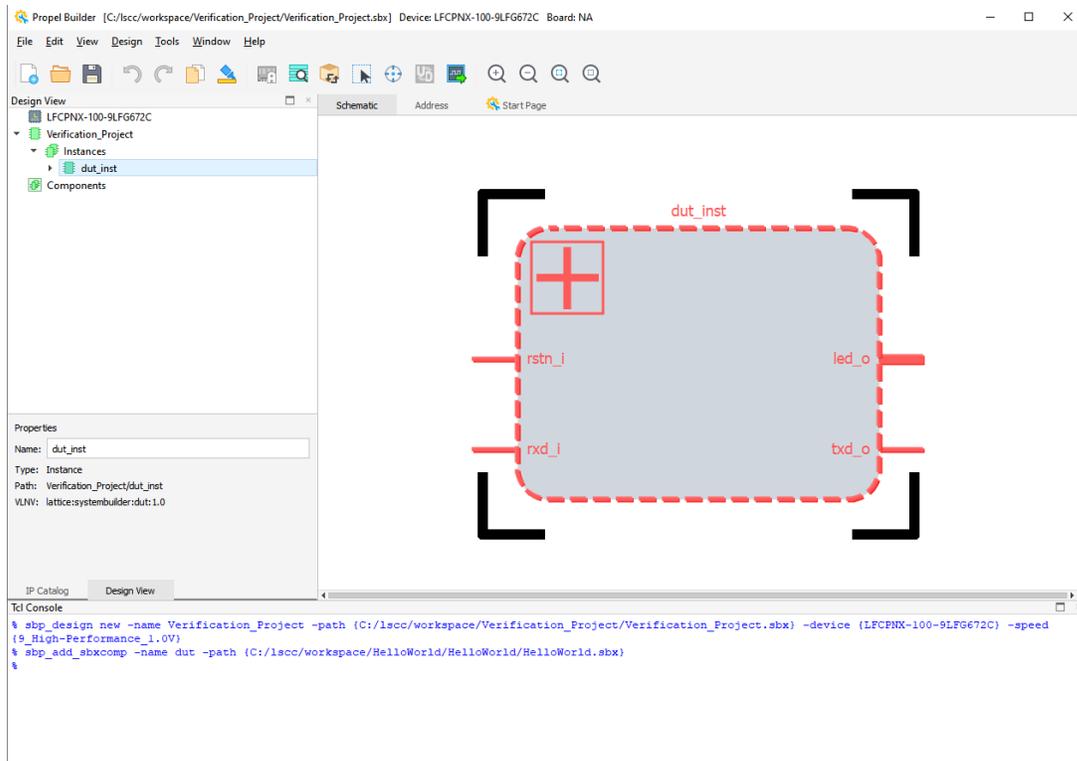


Figure 2.115. Verification Project

**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.

- Click the plus sign  of this `dut_inst` DUT block. You can see the whole SoC Design (Figure 2.116). Click the negative sign  to close the expanded bus.

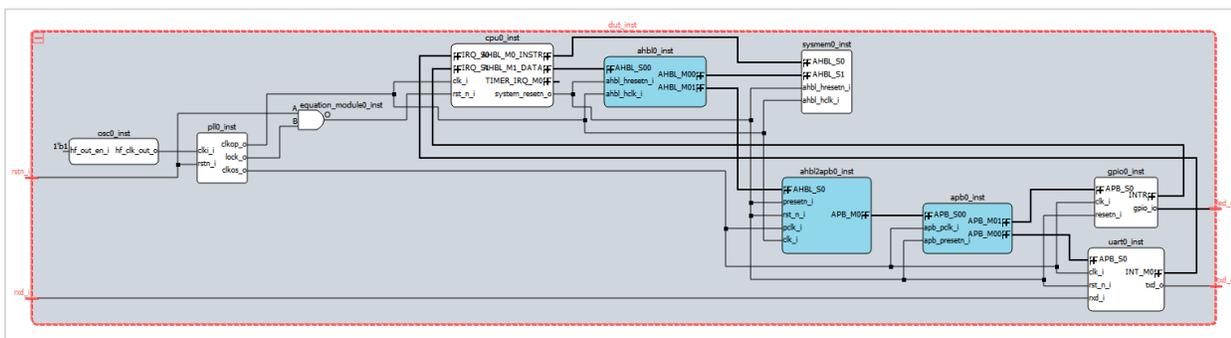


Figure 2.116. Whole SoC Design

### 2.3.2. Switching from SoC Design Project to Verification Project

Propel Builder also support switching from SoC project to verification project when a SoC project is already created.

- Create a SoC design project. Refer to the previous [Creating SoC Project](#) section for more details on this .

- From design project window, click **V/D button** . A dialog box opens prompting you to save the changes, if the current design project is not saved yet (Figure 2.117).

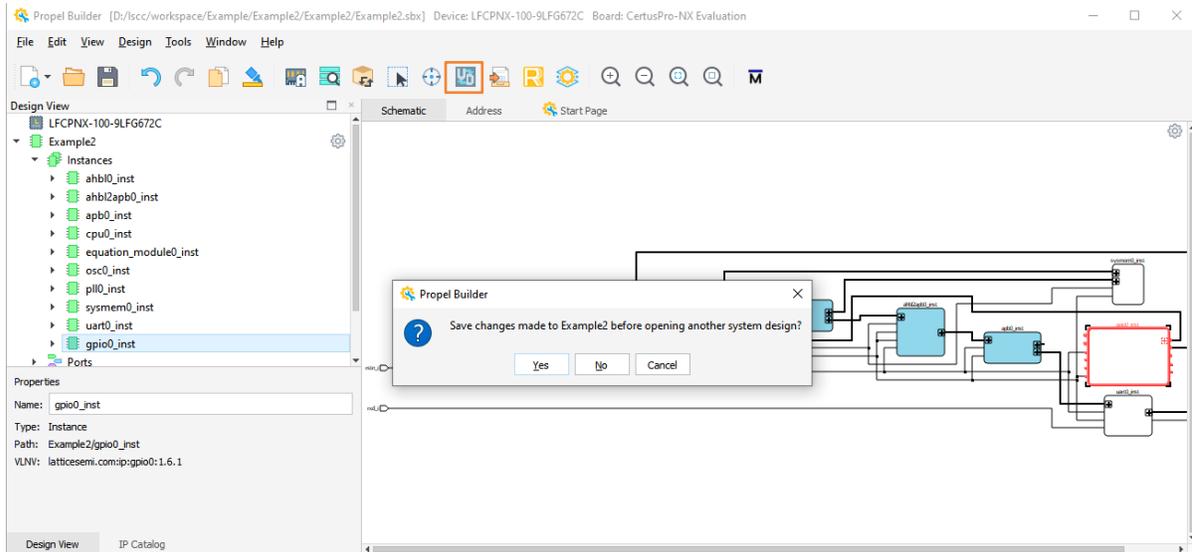


Figure 2.117. SoC Design Project

**Notes:**

- SoC design project is called **\*\*\*.sbx**. Verification project is called **\*\*\*\_v.sbx** (Figure 2.117).
  - **V/D button** is used to switch between Verification project and SoC Design project. When you create a SoC project, a related verification project is automatically created in the design project. This is how Verification/Design switch works, because they are auto-linked during project creation. Check **.soc** project file under the top project directory to see these details.
  - For some template SoC project, they may include a pre-developed functioning verification project.
- Click **Yes** to save the current design project. The Propel Builder GUI switches to verification project (Figure 2.118).

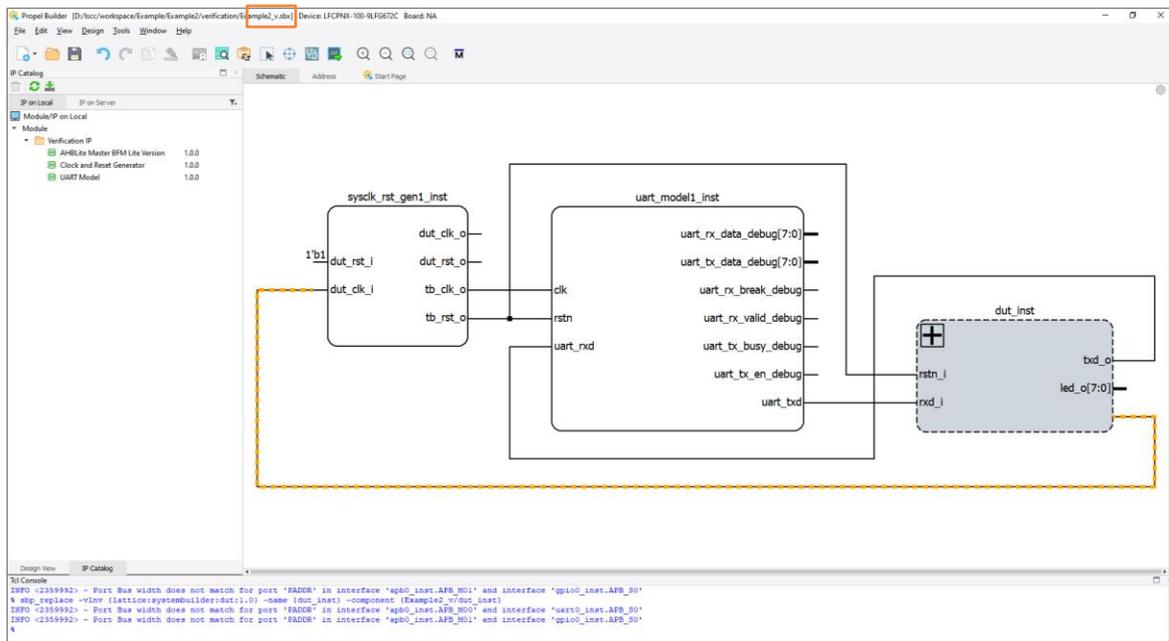
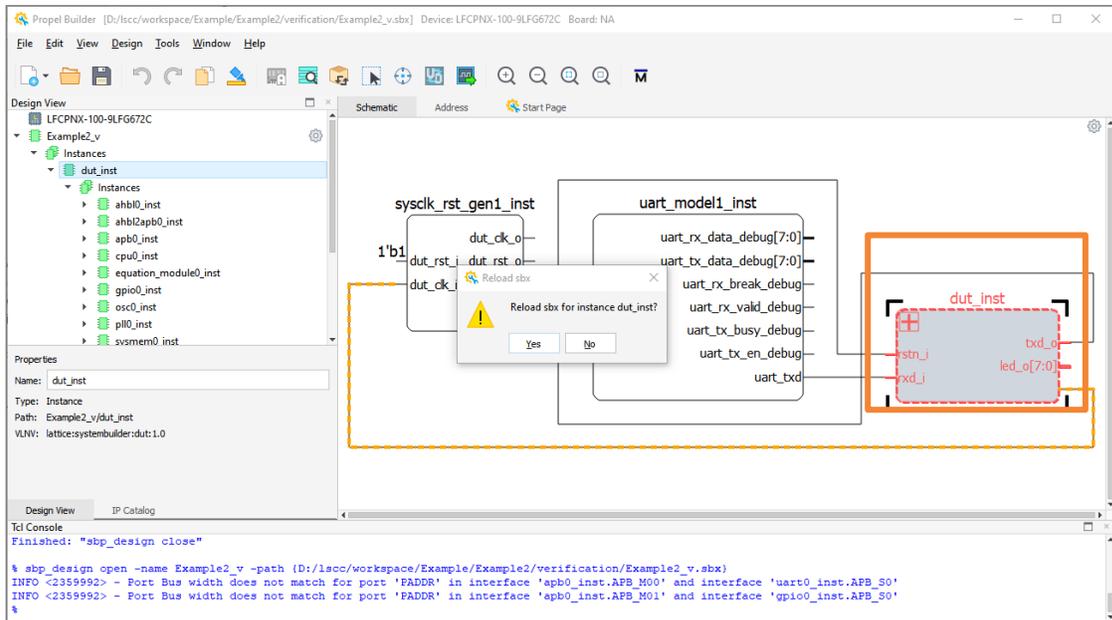


Figure 2.118. Verification Project

**Note:**

- If you want to go back and design more on SoC project, click **V/D** button  on the verification project window to go back to the SoC design project. (Figure 2.118)
- Each time you switch from SoC design project to Verification project, remember to:
  - a. **Save** SoC Design Project first.
  - b. Click on **V/D** button to switch to Verification project.
  - c. **Reload** dut\_inst to synchronize the SoC design change to verification project. Double click on dut\_inst to do the reload (Figure 2.119).



**Figure 2.119. Reload dut\_inst after Each SoC Project Design Change**

### 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. Viewing Address Maps

The Address view shows the read-only address maps of the soc design (Figure 2.120).

Schematic					Address					Start Page				
Cell	Base Address	Range	End Address	Lock										
▼ <b>cpu0_inst</b>														
▼ <b>LocalMemory</b>														
cpu0_inst/pic_timer_registers	0xFFFF0000	2K	0xFFFF07FF											
▼ test_v/cpu0_inst/riscv_ahbl_m_instr_Address_Space(32 address bits: 4G)														
system0_inst/AHBL_S0	0x00000000	32K	0x00007FFF	<input checked="" type="checkbox"/>										
▼ test_v/cpu0_inst/riscv_ahbl_m_data_Address_Space(32 address bits: 4G)														
gpio0_inst/APB_S0	0x00008400	1K	0x000087FF	<input checked="" type="checkbox"/>										
system0_inst/AHBL_S1	0x00000000	32K	0x00007FFF	<input checked="" type="checkbox"/>										
uart0_inst/APB_S0	0x00008000	1K	0x000083FF	<input checked="" type="checkbox"/>										

Figure 2.120. Address Maps

### 2.3.8. Monitoring DUT

This feature is available to 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 2.121. Only pin and pin bus are supported in the current release of Propel Builder.

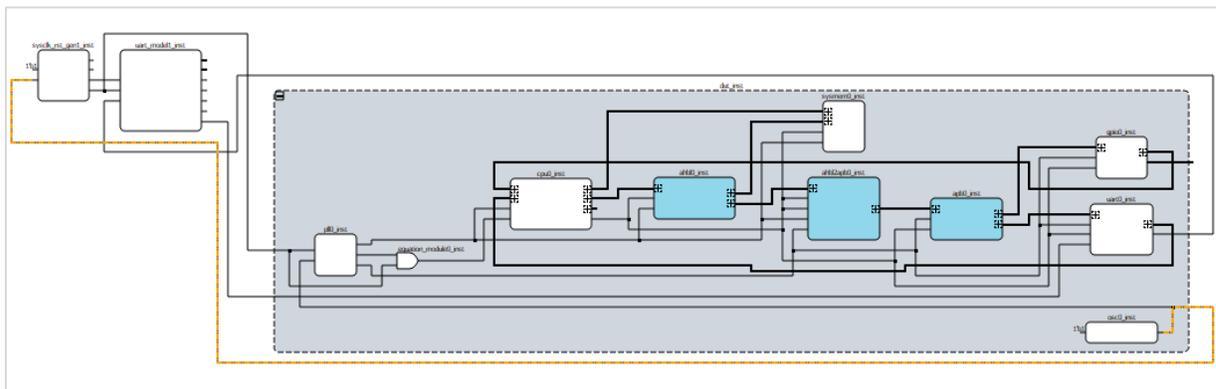


Figure 2.121. Monitoring DUT

The testbench generated for this Verification project is shown in Figure 2.122. This testbench file can be used for simulation. For testbench generation, see [Generating Simulation Environment](#).

```

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 2.122. Testbench of the Verification Project

### 2.3.9. Generating Simulation Environment

1. (Optional) In design project view, initialize mem file in System Memory IP instance. See [Launching SDK and Generate Mem File](#) section for how to generate mem file. If you want to launch simulation without the actual program running, which is not a functioning simulation environment, you can skip this step.

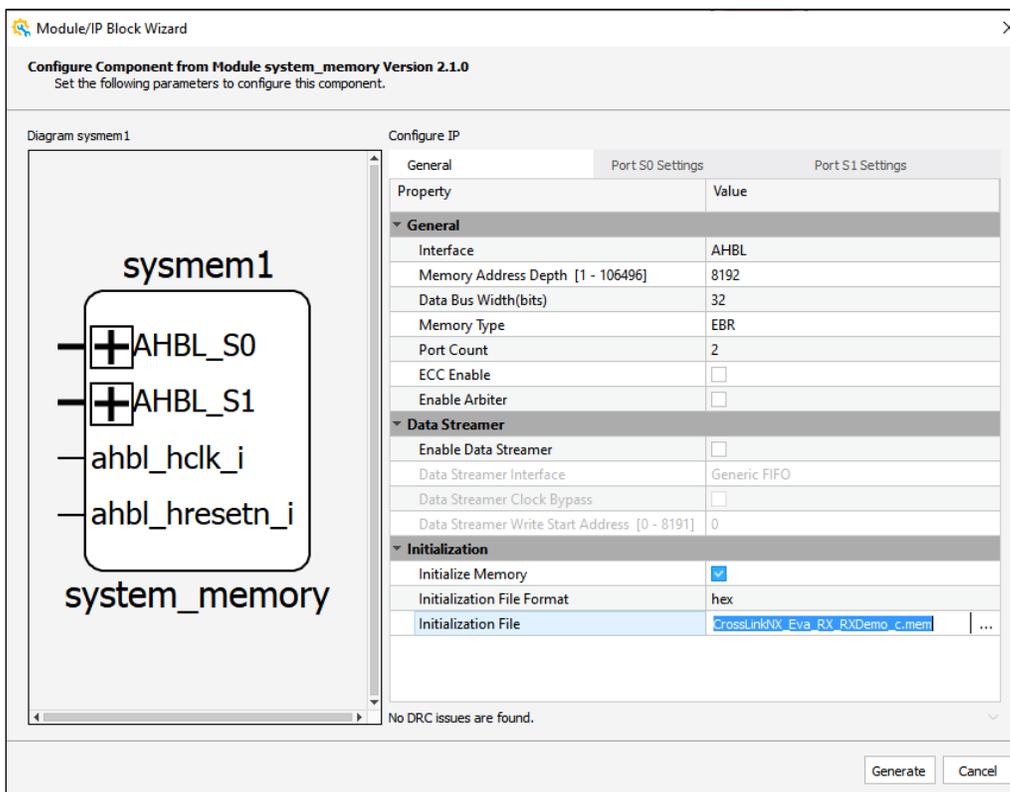


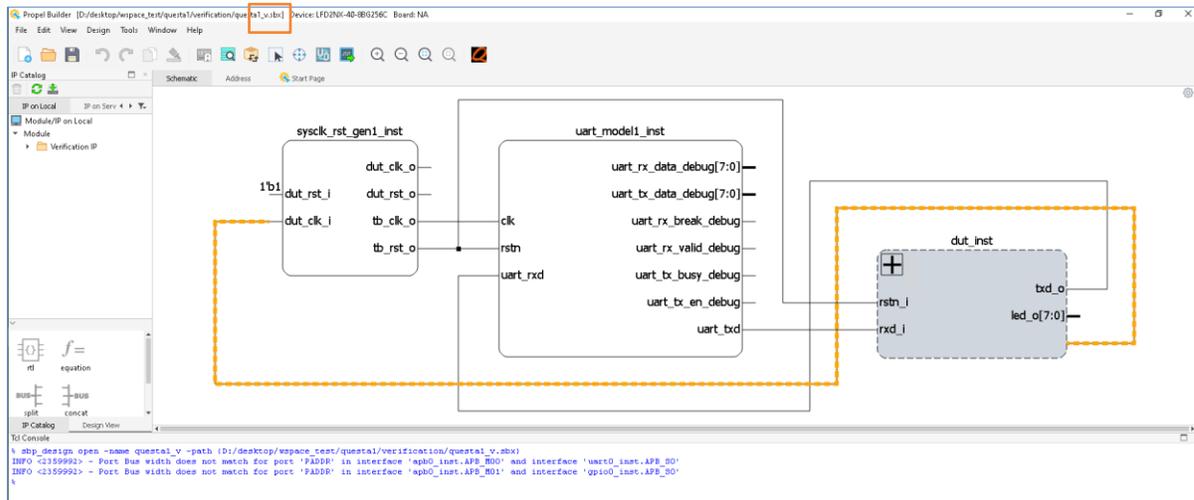
Figure 2.123. Initialization of Mem File

- In design project view, click **V/D** button to switch to verification project. Check the [Switching from SoC Design Project to Verification Project](#) section for more notes on this operation. The verification project is located at `./<project_name>/verification`.

- In verification project view, click **Generate** icon  in the Propel Builder Toolbar ([Figure 2.124](#)). If prompted with 'INFO: The simulation environment has been created' in TCL console, then the testbench is generated in: `<project_name>/verification/sim`.

**Note:**

- SoC design project is called `***.sbx`. Verification project is called `***_v.sbx` ([Figure 2.124](#)).



**Figure 2.124. Generate Simulation Environment on Verification Project**

The testbench file structure is shown as below ([Figure 2.125](#)):

```

+--- [sim]                                -- Generated simulation environment folder
| +--- [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
| +--- flist_sim.f                         -- File list for all files used in simulation
| +--- qsim.do                             -- Do script for simulator,
| |                                         qsim.do: 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>_v.sv                 -- Top testbench, it's SystemVerilog based.
    
```

**Figure 2.125. Testbench File Structure**

- If there is any change to the design project, make sure to **save** the design in design project view, switching from design project to verification project using **V/D** button. Reload `dut_inst` and generate simulation in verification project again. See [Switching from SoC Design Project to Verification Project](#) section for more details.
- Unlike those HDLs generated in the SOC design project directory, the testbench generation in the Verification project is just a start point for you to work with, not necessarily a full functioning verification project.
- If there is a simulation tool change, make sure to generate simulation environment again. Otherwise, a dialog box pops up prompting you to make a choice of Yes or No ([Figure 2.126](#)).

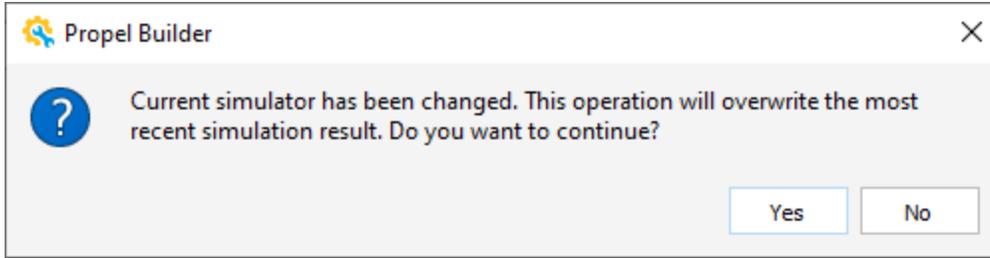


Figure 2.126. Remind for Simulation Tool Change

### 2.3.10. Launching Simulation

1. After [Generating Simulation Environment](#), simulation project is generated in `<project_name>/verification/sim`.



2. Click **Launch Simulation** icon in verification project view to launch simulation.

**Note:** SoC design project is called `***.sbx`. Verification project is called `***_v.sbx` (Figure 2.127).

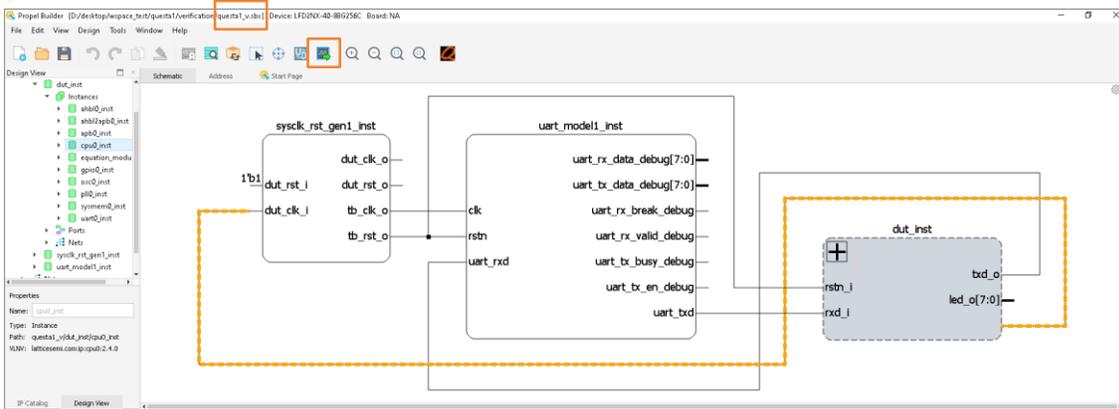


Figure 2.127. Launch Simulation in Verification Project

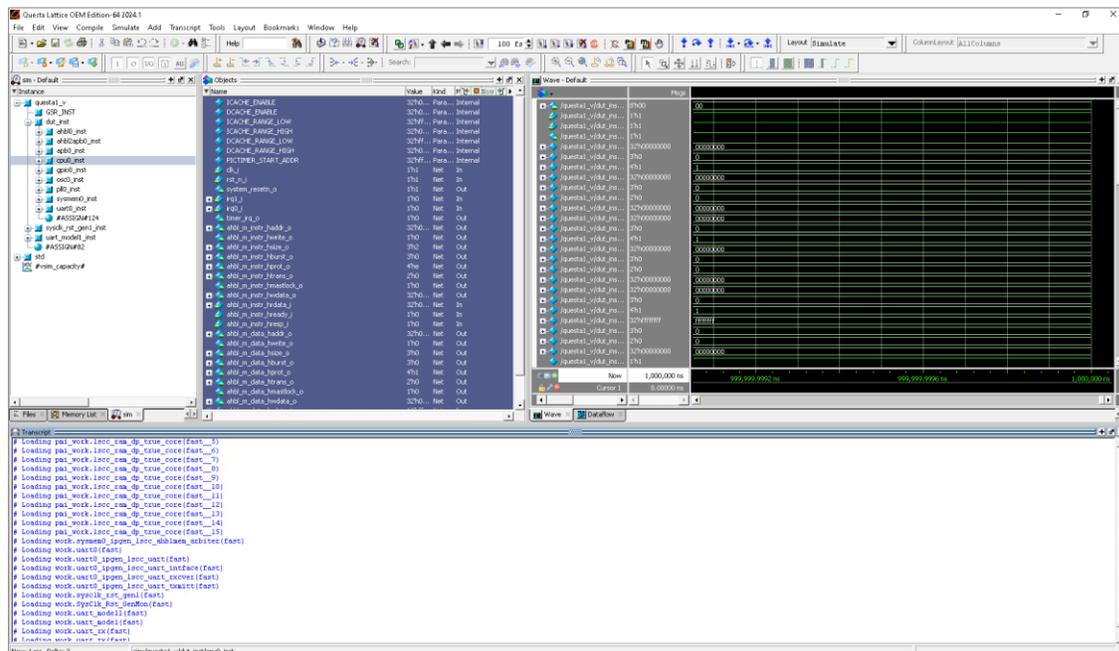
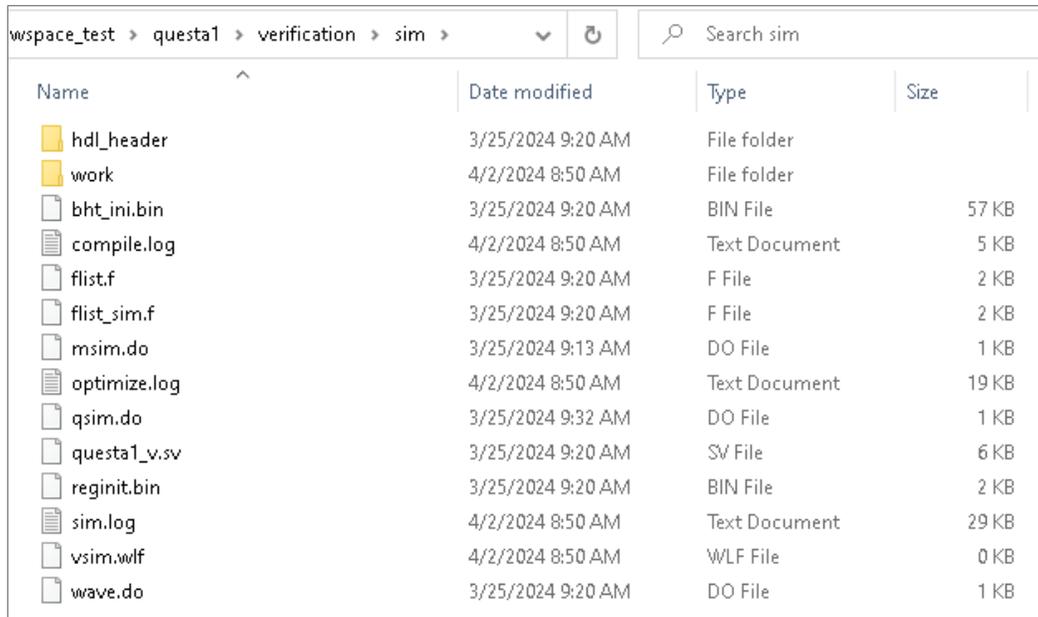


Figure 2.128. Questa Simulation GUI

3. If there is no simulation tool specified in **Tools > Options > Directories**, the default built-in simulation tool, OEM version of QuestaSim, opens (Figure 2.128). If specified, the specified simulation tool opens.
  - For OEM version QuestaSim, there are pre-compiled simulation libraries.
  - For non-OEM version simulation tool, the script under simulation project directory can detect whether or not there are existing simulation libraries under the `ovi_<device_family_name>` folder. If yes, simulation uses the existing library; otherwise, the device library can be created and compiled once into the `ovi` folder for further use, and this compiling process may take some time.
4. See Figure 2.129 for simulation file structure.



Name	Date modified	Type	Size
hdl_header	3/25/2024 9:20 AM	File folder	
work	4/2/2024 8:50 AM	File folder	
bht_ini.bin	3/25/2024 9:20 AM	BIN File	57 KB
compile.log	4/2/2024 8:50 AM	Text Document	5 KB
flist.f	3/25/2024 9:20 AM	F File	2 KB
flist_sim.f	3/25/2024 9:20 AM	F File	2 KB
msim.do	3/25/2024 9:13 AM	DO File	1 KB
optimize.log	4/2/2024 8:50 AM	Text Document	19 KB
qsim.do	3/25/2024 9:32 AM	DO File	1 KB
questa1_v.sv	3/25/2024 9:20 AM	SV File	6 KB
reginit.bin	3/25/2024 9:20 AM	BIN File	2 KB
sim.log	4/2/2024 8:50 AM	Text Document	29 KB
vsim.wlf	4/2/2024 8:50 AM	WLF File	0 KB
wave.do	3/25/2024 9:20 AM	DO File	1 KB

**Figure 2.129. Testbench File Structure in Simulation Project**

## 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 2024.1 User Guide \(FPGA-UG-02213\)](#) for detailed usage of these interfaces in the IP Packager.

### 2.4.2. Supported Language

Propel 2024.1 Builder supports the following languages:

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

Currently, verification projects and RTL module (see RTL section) 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 2.130](#)) to show the detailed design scope of this IP ([Figure 2.131](#)).

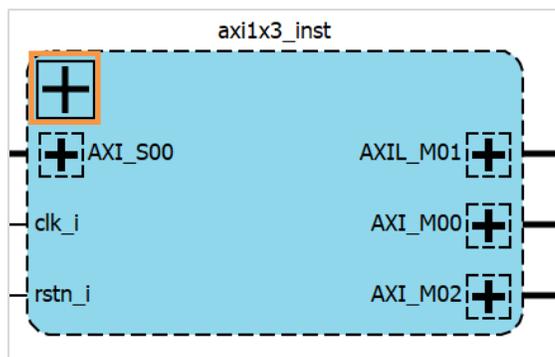


Figure 2.130. Hierarchical IP

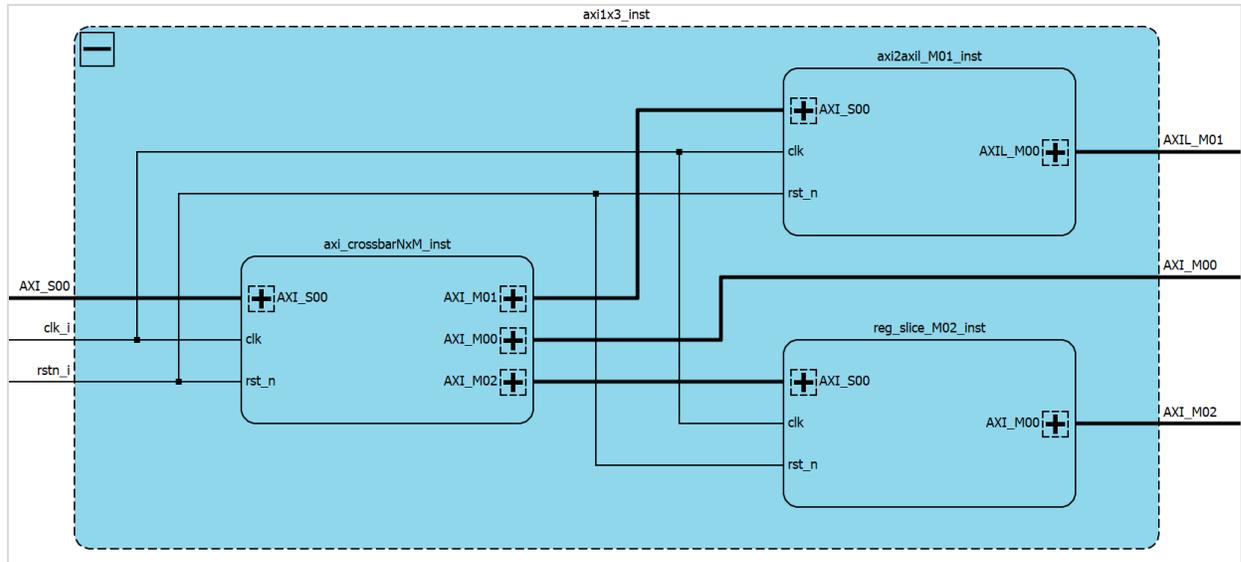


Figure 2.131. Hierarchical IP Scope in Detail

### 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 information when the feedthrough module is configured as major, or to hold memory map information when the feedthrough module is configured as subordinate. 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 2.132](#).

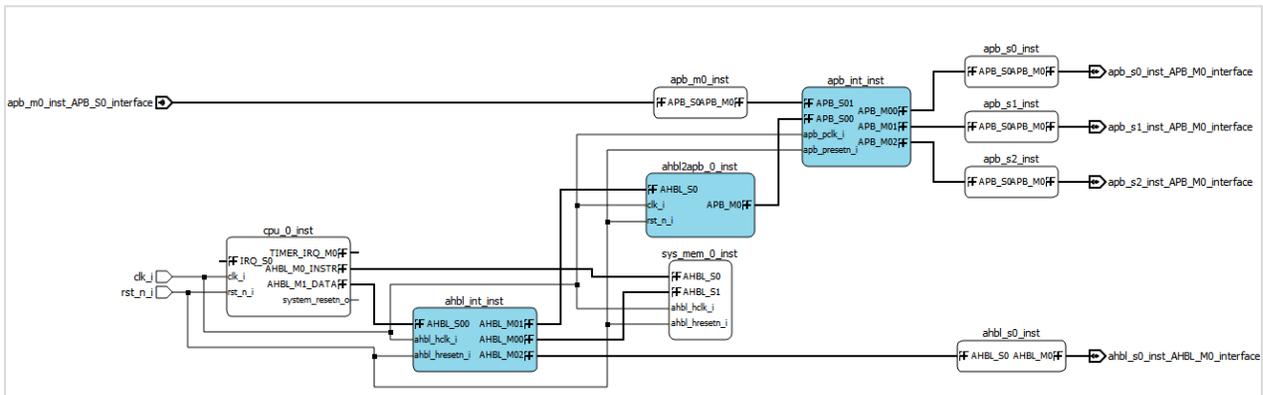


Figure 2.132. Export Interface Example

In the example above, to export interface `ahbl_int_inst.AHBL_M02`:

1. Instantiate an AHB-Lite feedthrough module from IP catalog, and configure it as subordinate. 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.AHBL_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 2.133](#)).

Schematic		Address		Start Page	
Cell	Base Address	Range	End Address	Lock	
▼ <b>apb_m0_inst</b>					
▼ apb_m0_inst/APB_M0(32 address bits: 64K)					
apb_s0_inst/APB_S0	0x00010800	1K	0x00010BFF	<input checked="" type="checkbox"/>	
apb_s1_inst/APB_S0	0x00010C00	1K	0x00010FFF	<input checked="" type="checkbox"/>	
apb_s2_inst/APB_S0	0x00011000	2K	0x000117FF	<input checked="" type="checkbox"/>	
▼ <b>cpu_0_inst</b>					
▼ LocalMemory					
cpu_0_inst/pic_timer_registers	0xFFFF0000	2K	0xFFFF07FF		
▼ cpu_0_inst/AHBL_M0_INSTR(32 address bits: 4G)					
sys_mem_0_inst/AHBL_S0	0x00000000	64K	0x0000FFFF	<input checked="" type="checkbox"/>	
▼ cpu_0_inst/AHBL_M1_DATA(32 address bits: 4G)					
ahbl_s0_inst/AHBL_S0	0x00010000	2K	0x000107FF	<input checked="" type="checkbox"/>	
apb_s0_inst/APB_S0	0x00010800	1K	0x00010BFF	<input checked="" type="checkbox"/>	
apb_s1_inst/APB_S0	0x00010C00	1K	0x00010FFF	<input checked="" type="checkbox"/>	
apb_s2_inst/APB_S0	0x00011000	2K	0x000117FF	<input checked="" type="checkbox"/>	
sys_mem_0_inst/AHBL_S1	0x00000000	64K	0x0000FFFF	<input checked="" type="checkbox"/>	

Figure 2.133. 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.

### Notes:

- The GUI entry of this function takes effect from Propel Builder 2022.1, so the design created in earlier Propel Builder versions cannot define its custom template.
- Refer to [Lattice Propel 2024.1 SDK User Guide \(FPGA-UG-02211\)](#) for more details on how to use custom template in Propel SDK.

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

- Access during project creation. See the [Creating 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 2.134](#)) opens.

Template Configuration
✕

---

**General**

Name:  Processor:  RISC-V MC  
 RISC-V SM  
 RISC-V RX

Vendor:

Version:

Board Only:

Device Family:  LAV-AT  
 LFCPNX  
 LFD2NX  
 LFMXO5  
 LIFCL  
 LFMNX  
 MachXO2  
 MachXO3D

Description:

---

**Supported Boards**

	Name	Version	PartName	Speed	Constraint File
1	CertusPro-NX Evaluation	A	LFCPNX-100-9LFG672C	9_High-Performance_1.0V	./Example2.pdc

---

**Template Resources**

1  ./Example2/application

---

Share Template

**Figure 2.134. 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 2.135). 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 2.136). 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 2.134).

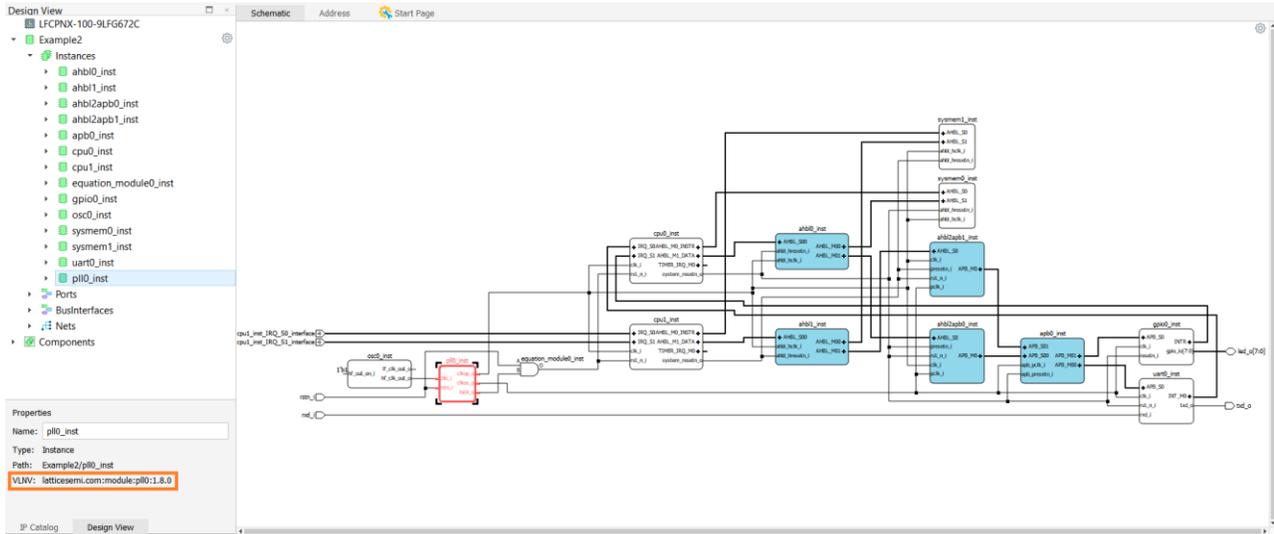


Figure 2.135. Check IP Name in Design View

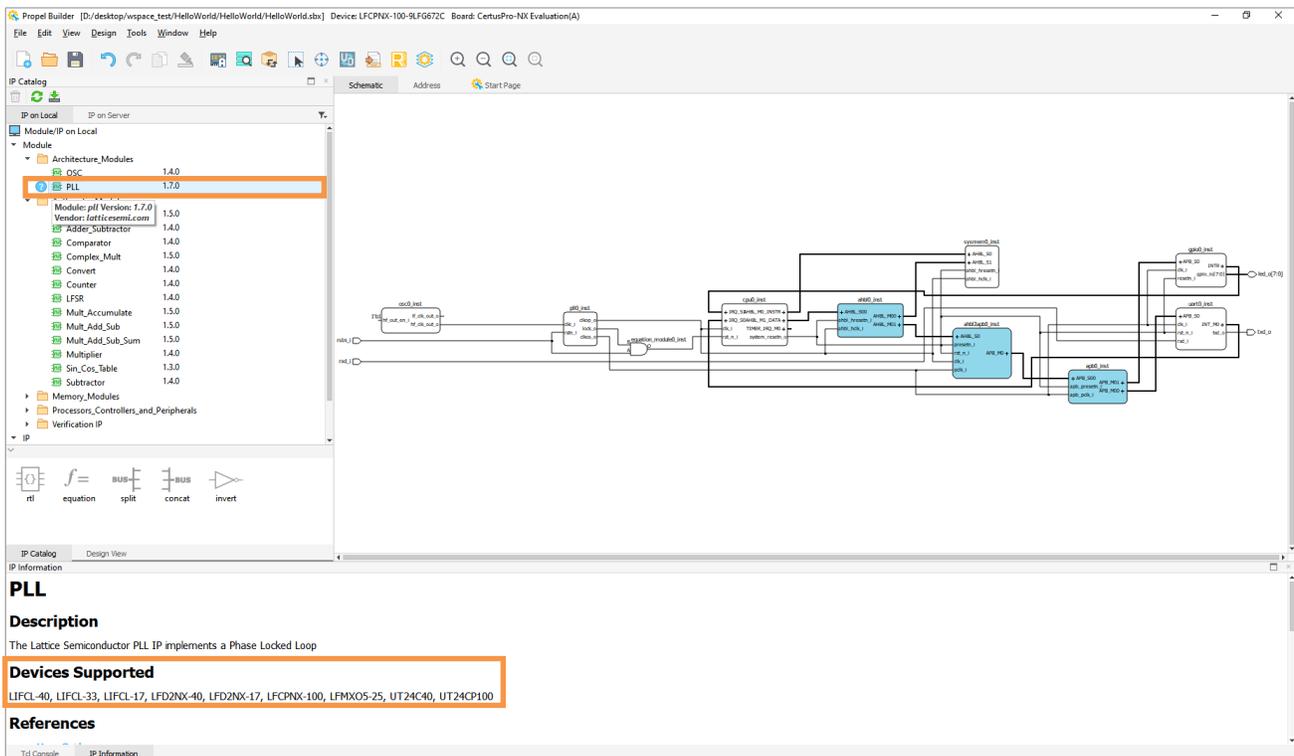
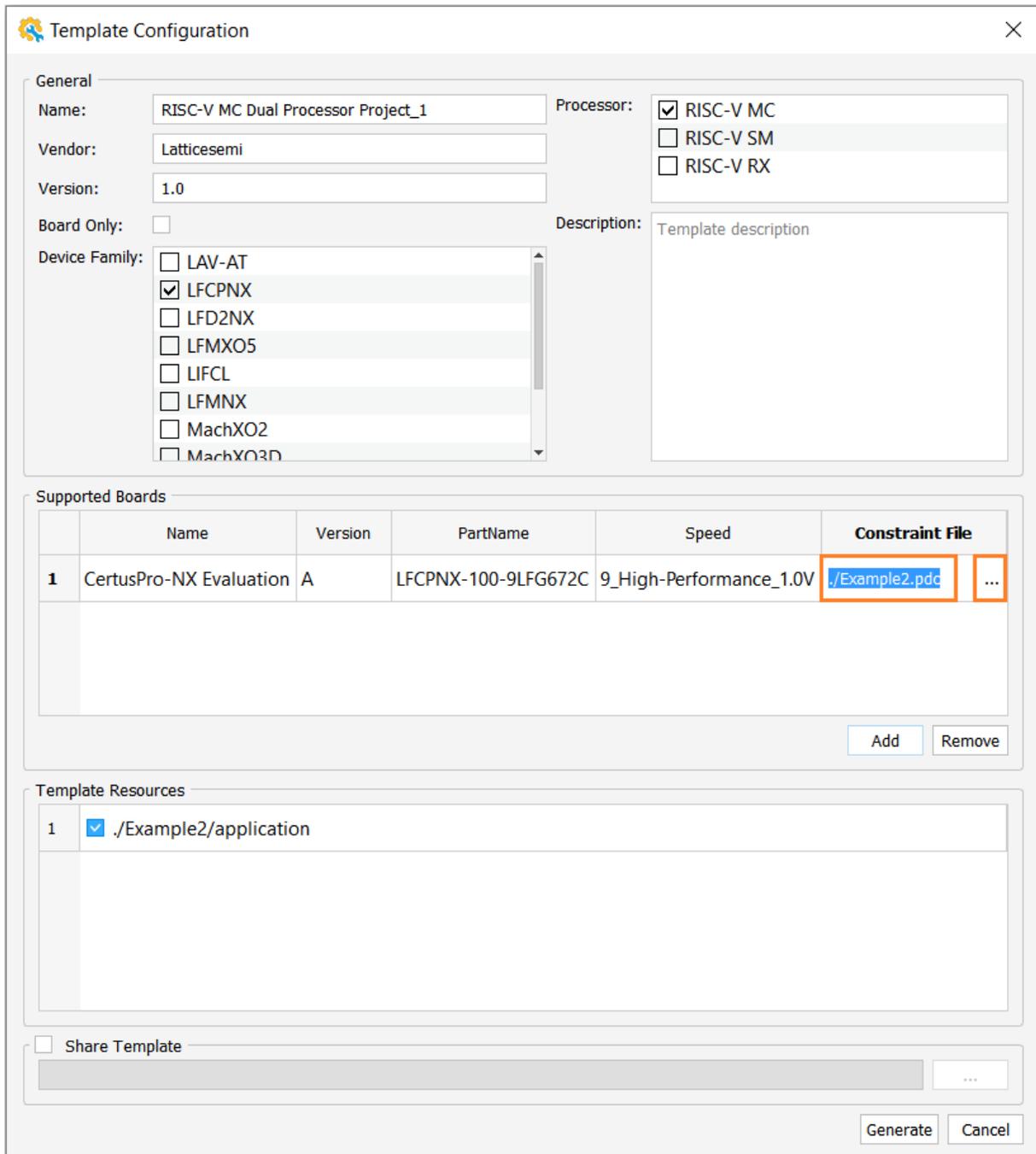


Figure 2.136. Check Device Supported in IP Information

- You can add the board information manually in **Supported Boards** (refer to Figure 2.5) for board information). You must ensure the correctness of the board information and the design can work on this board.
- Board **must** contain a Constraint File. **Double click** on the constraint file name (./Example2.pdc), and then click ‘...’ button to select the constraint file (Figure 2.137).



**Figure 2.137. Double Click on the Constraint File Name**

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 2.138).
- 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 2.134), then the application under current project is also exported to destination (Figure 2.138). Other files under this option works the same way.

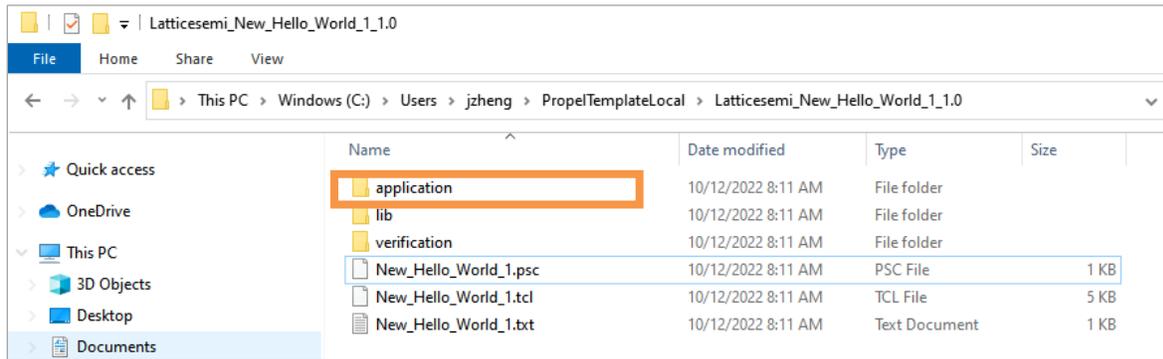


Figure 2.138. Export Template to PropelTemplateLocal Folder

- 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. You can export a template to share later using template manager.
- Template manager page for custom templates.
- You can import/delete/export a selected template in **Template Manager** when creating a new system design (Figure 2.139 and Figure 2.140).

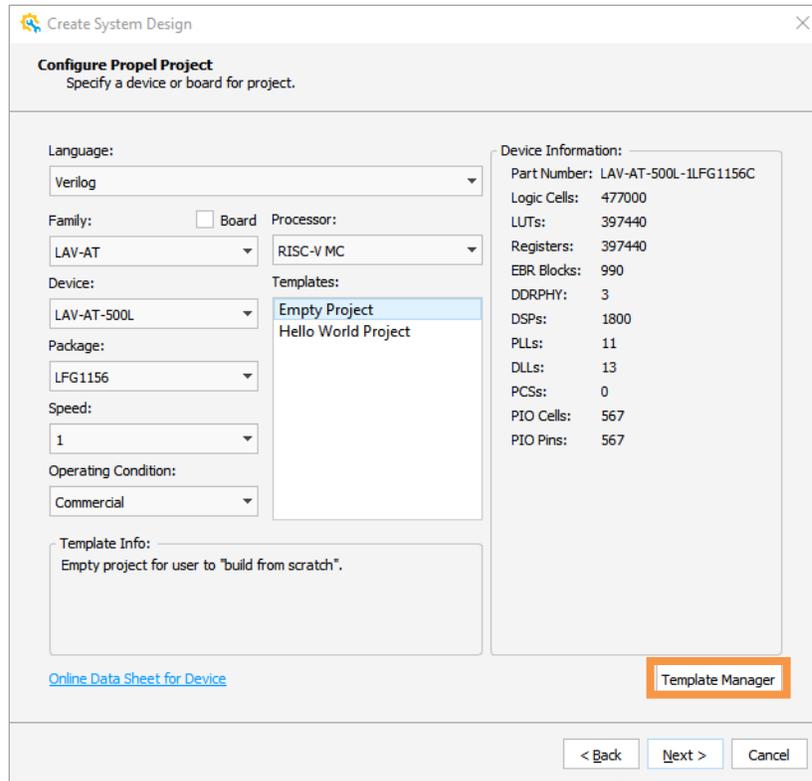


Figure 2.139. Template Manager Entry

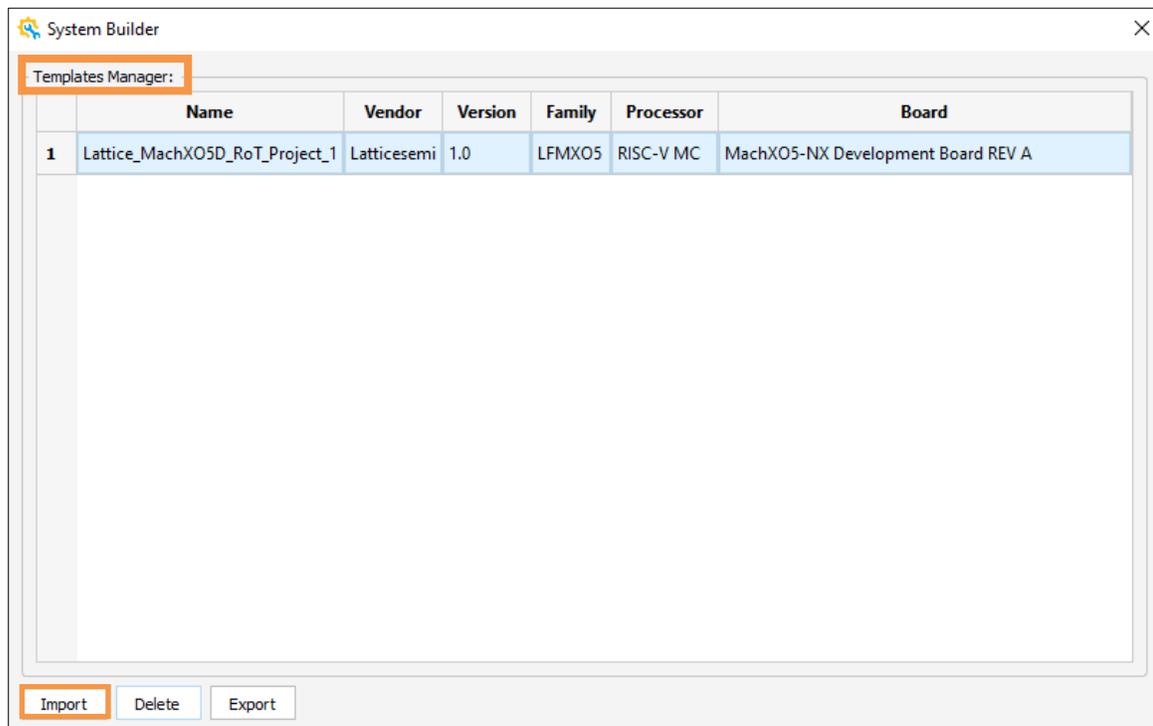
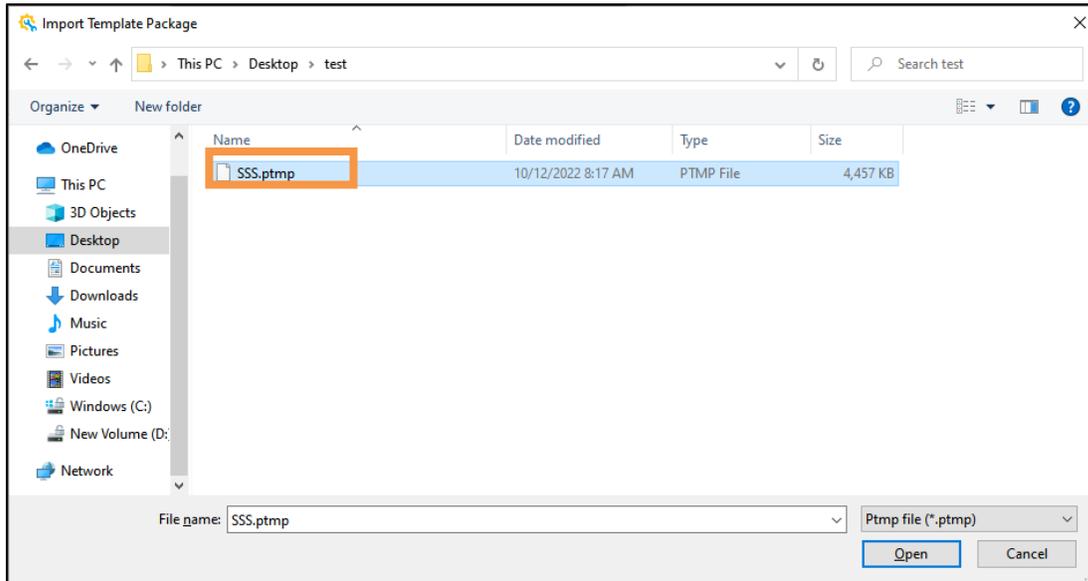


Figure 2.140. Templates Manager Page

### 2.4.5.2. Import a Template

1. Click **Import** (Figure 2.140), and then choose the desired .ptmp file (Figure 2.141).



**Figure 2.141. Import a .ptmp File**

2. Custom template installation path.

Custom templates are installed in \$HOME/PropelTemplateLocal directory by default (Figure 2.142).

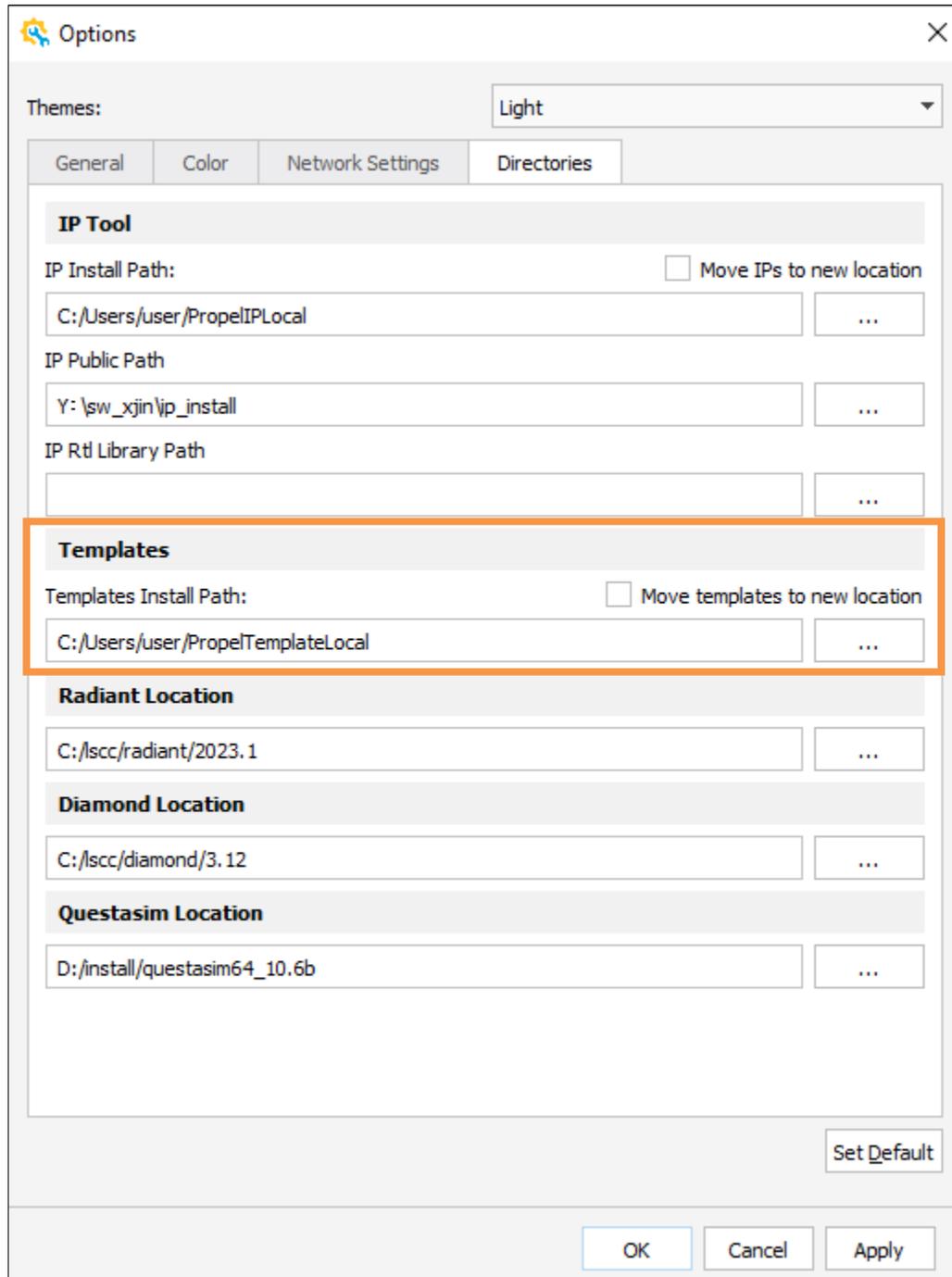


Figure 2.142. Template Manager Page

You can move it to another location by checking the *Move templates to new location* option in Directories tab of the **Options** dialog (Figure 2.143).

For custom templates usage in Propel SDK, refer to [Lattice Propel 2024.1 SDK User Guide \(FPGA-UG-02211\)](#).

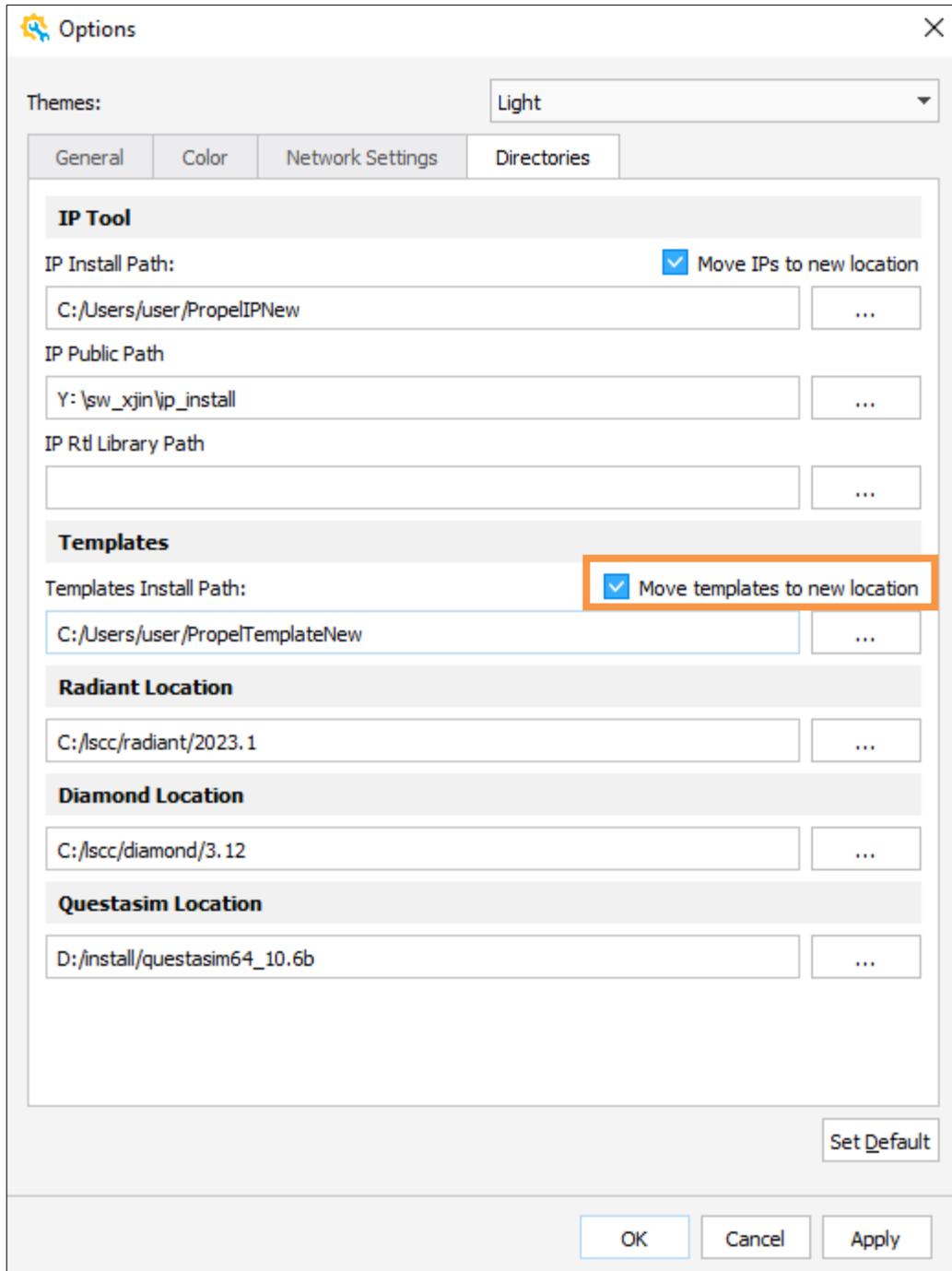


Figure 2.143. Template Manager Page – Change to New Location

### 2.4.5.3. Delete a Template

Click on the custom template you want to delete from the **Template Manager** Page (Figure 2.144), and then click **Delete**. The corresponding template folder under **Template Install Path** will be deleted.

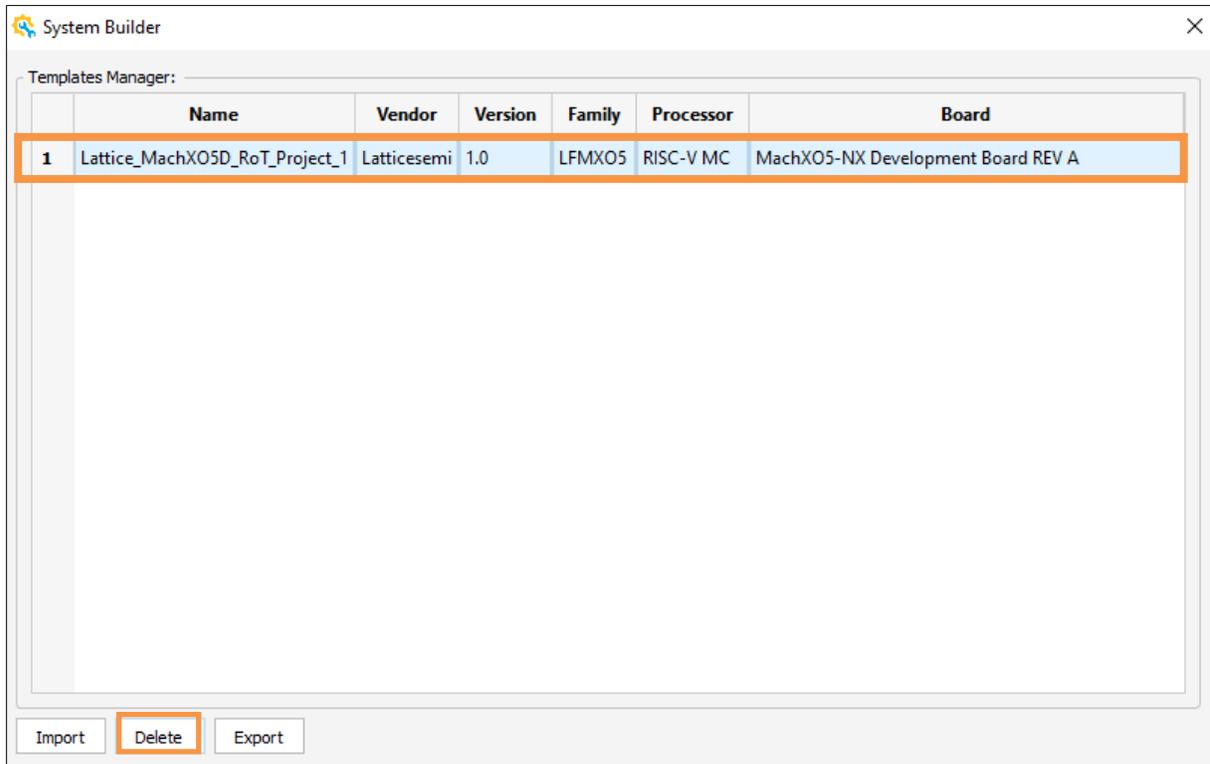


Figure 2.144. Templates Manager Page – Delete a Template

### 2.4.6. Include Sub Sbx File

Right Click on **Schematic View** and choose **Add Sbx Instance** (Figure 2.145), and then chose the sbx file you want to input (Figure 2.146).

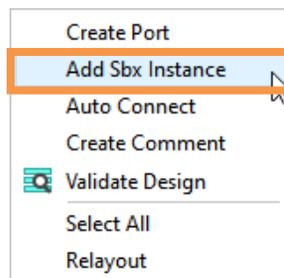


Figure 2.145. Right Click on Schematic View

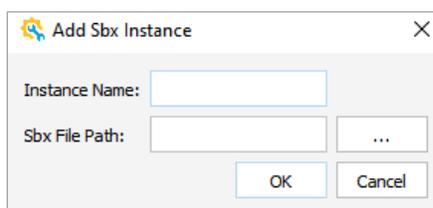


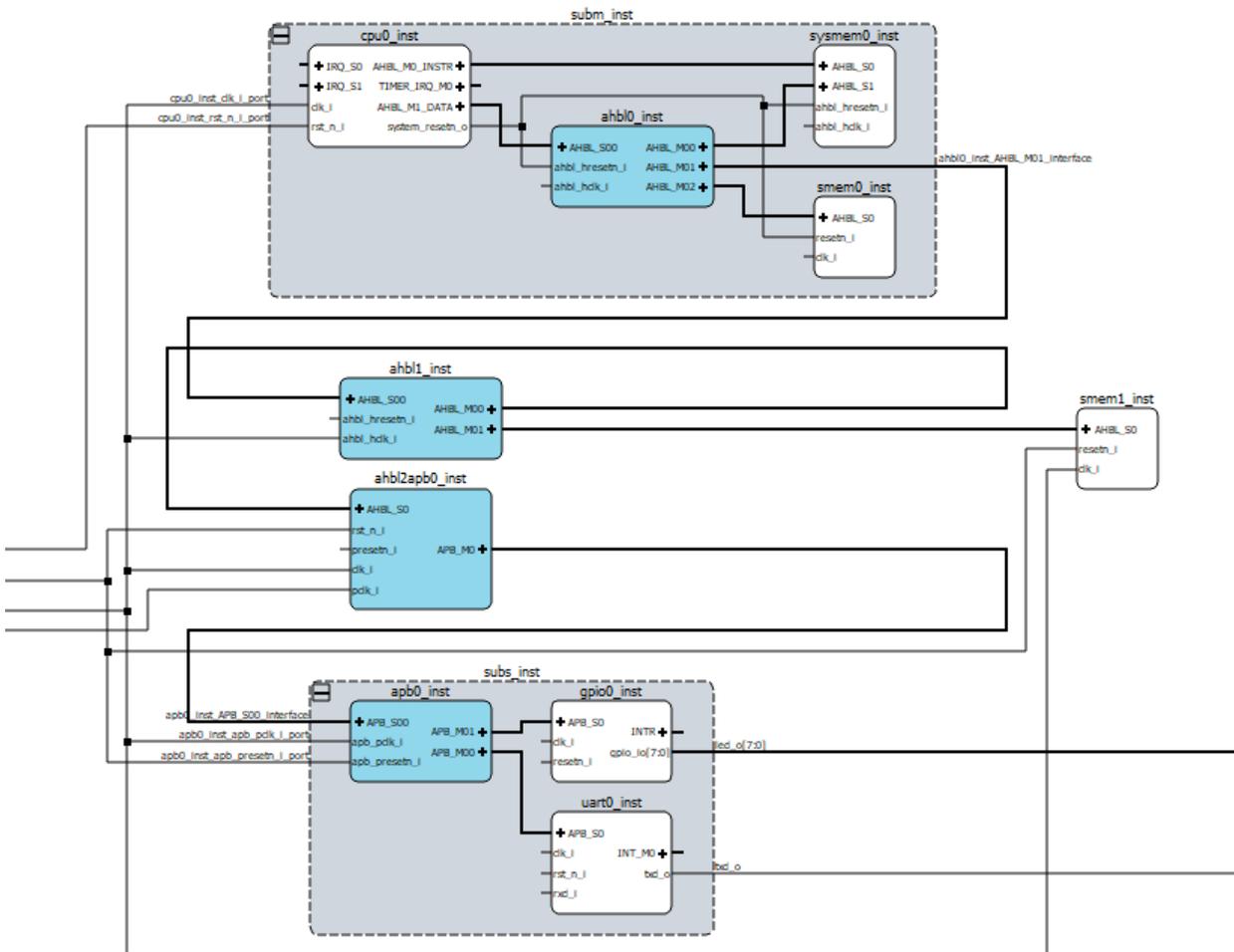
Figure 2.146. Input sbx File and Instance Name

**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 that of the component in the top sbx file.

- Make sure the top-level ports of the sub sbx are the in/outs of the component in the top-level design. You need to make the ports you intend on connecting in the top sbx at the top-level in the sub sbx by exporting these ports in sub sbx.
- Only one-level of hierarchy is supported. A sub sbx file cannot have hierarchical components.
- Support verification project design flow for sub-system sbx. Only one level of sbx component instantiation is supported.

Auto memory assignment considers memory map for a subordinate or address space of a manager in the sub system sbx, if there is interconnect bus to connect them. As shown in [Figure 2.147](#), subm\_inst and subs\_inst are the sub sbx instances.



**Figure 2.147. Sub major sbx and sub subordinate sbx**

The memory information in sub system sbx is displayed in top-level address tab ([Figure 2.148](#)).

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	<input checked="" type="checkbox"/>
▼ design/cpu0_inst/riscv_ahbl_m_data_Address_Space(32 address bits: 4G)				
gpio0_inst/APB_S0	0x00008400	1K	0x000087FF	<input checked="" type="checkbox"/>
smem0_inst/AHBL_S0	0x00008000	1K	0x000083FF	<input checked="" type="checkbox"/>
smem1_inst/AHBL_S0	0x00008C00	1K	0x00008FFF	<input checked="" type="checkbox"/>
sysmem0_inst/AHBL_S1	0x00000000	32K	0x00007FFF	<input checked="" type="checkbox"/>
uart0_inst/APB_S0	0x00008800	1K	0x00008BFF	<input checked="" type="checkbox"/>

Figure 2.148. 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.

```
Tcl Console

% sbp_design close
INFO - Finished: sbp_design close
Finished: "sbp_design close"

% sbp_design open -name hello_v -path {G:/Lattice/project/Raptor/tool/hw/hello_v/hello_v/hello_v.sbx}
% sbp_design close
INFO - Finished: sbp_design close
Finished: "sbp_design close"
```

Figure 3.1. TCL Console

### 3.1. sbp\_design Commands

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.

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

#### 3.1.2. Close

Closes a Propel Builder design currently opened.

<b>Usage</b>	sbp_design close
--------------	------------------

#### 3.1.3. New

Creates a new Propel Builder design.

<b>Usage</b>	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).

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

#### 3.1.5. Drc

Runs the design rule check (DRC) for the Propel Builder design file.

<b>Usage</b>	sbp_design drc
--------------	----------------

### 3.1.6. Generate

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

<b>Usage</b>	<code>sbp_design generate</code>
--------------	----------------------------------

### 3.1.7. Auto Assign Addresses

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

<b>Usage</b>	<code>sbp_design auto_assign_addresses</code>
--------------	---

### 3.1.8. Verify

Launches the SOC verification engine to verify design.

<b>Usage</b>	<pre>sbp_design verify [-h] [--workfolder &lt;workfolder_path&gt;] [--sbx_file &lt;sbx_file_path&gt;] {sim_gen,pfr_pack,regmap_gen,auto_run}</pre> <p>positional arguments: {sim_gen,pfr_pack,regmap_gen,auto_run} specify the application</p> <pre>sim_gen      Simulation Generation Flow pfr_pack     PFR Security Engine Packager Flow regmap_gen   Memory Map Generation Flow auto_run     Automation Flow</pre> <p>optional arguments:</p> <pre>-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</pre>
<b>Example</b>	<code>sbp_design verify --workfolder ./mem_map --sbx_file D:/XO3D_Initial/XO3D_Initial.sbx regmap_gen</code>

### 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.

<b>Usage</b>	<pre>sbp_design pge sge -l &lt;input path&gt; [-o &lt;output path&gt;] -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.</pre>
--------------	--

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

<b>Usage</b>	<pre>sbp_design pge dge -i &lt;input path&gt; [-o &lt;output path&gt;] [-diamond -radiant]</pre> <p>-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.</p>
--------------	--

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

<b>Usage</b>	<pre>sbp_design pge gen_signature -cfile &lt;c binary file&gt; -dfile &lt;d binary file&gt; -output &lt;output binary file&gt;</pre>
--------------	--

### 3.1.10. Undo

Undo operation. Currently, software only supports single undo.

<b>Usage</b>	<pre>sbp_design undo</pre>
--------------	----------------------------

### 3.1.11. Redo

Redo operation. Currently, software only supports single redo.

<b>Usage</b>	<pre>sbp_design redo</pre>
--------------	----------------------------

### 3.1.12. Set Device

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

<b>Usage</b>	<pre>sbp_design set_device -device &lt;device name&gt; -speed &lt;speed value&gt; -package &lt;package value&gt; -operating &lt;operating condition&gt;</pre>
--------------	---

## 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.

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

### 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.

<b>Usage</b>	<pre>sbp_add_component -vlv &lt;VLNV&gt; -name &lt;instance_name&gt;</pre>
<b>Example</b>	<pre>sbp_add_component -vlv lattice:ip:ahbl_interconnect:1.0 -name ahblite_interconnect</pre>

### 3.2.3. sbp\_add\_sbxcomp

Instantiates a hierarchical component from sbx file into the system.

<b>Usage</b>	<code>sbp_add_sbxcomp -name &lt;hname&gt; -path &lt;sbx_file_absolute_path_name&gt;</code>
<b>Example</b>	<code>sbp_add_sbxcomp -name sim_comp -path "C:/test/test.sbx"</code>

### 3.2.4. sbp\_config\_ip

args:

- `-vlnv`: component vlnv, consisted with vendor, library, name, version. When use this TCL command, vendor, library, version of component must be the same as `meta_vlnv`.
- `-meta_vlnv`: Location of IP configuration file by vendor library name and version, use TCL "ip\_catalog\_list" to get it. Usage: 'vendor:library:name:version'.
- `-cfg_value`: IP configuration value: '{id:value, id1:value1}'.
- `-meta_loc`: Location of IP module package.
- `-cfg`: Location of IP configuration file.

**Note:** This TCL command must contains one of `-meta_vlnv` and `-meta_loc`, also `-cfg_value` and `-cfg`.

<b>Usage1</b>	<code>sbp_config_ip -vlnv &lt;component vlnv&gt; -meta_vlnv {IP vlnv} [-cfg_value &lt;IP configuration value&gt;]</code>
<b>Example1</b>	<code>sbp_config_ip -vlnv {latticesemi.com:ip:gpio0:1.6.1} -meta_vlnv {latticesemi.com:ip:gpio:1.6.1} -cfg_value {DIRECTION_DEF_VAL_INPUT:FF,IO_LINES_COUNT:8,OUT_RESET_VAL_INPUT:FF}</code>
<b>Usage2</b>	<code>sbp_config_ip -vlnv &lt;component vlnv&gt; -meta_loc {location} -cfg &lt;IP configuration file&gt;</code>
<b>Example2</b>	<code>sbp_config_ip -vlnv {latticesemi.com:module:osc1:2.0.A} -meta_loc {/home/user/work/osc} -cfg {/home/user/work/tt1/tt1/.lib/latticesemi.com/module/osc1/2.0.A/osc1.cfg}</code>

### 3.2.5. ip\_catalog\_list

Return a list of IP, local IP contains the IP from Propel install package, IP server and customer's packaged IP, server IP contains the IP in server. Only get local IP by default.

args:

`-server`: Get IP list in server.

<b>Usage</b>	<code>ip_catalog_list [-name &lt;IP wild name&gt;] [-library &lt;IP wild library&gt;] [-vendor &lt;IP wild vendor&gt;] [-version &lt;IP wild version&gt;] [-server]</code>
<b>Example</b>	<code>ip_catalog_list -name adder*</code>

### 3.2.6. ip\_catalog\_install

Install one IP from local ipk file or one IP from server.

args:

- `-vlnv`: Location of IP configuration file by vendor library name and version, use tcl 'ip\_catalog\_list' to get it. Usage: 'vendor:library:name:version'.
- `-file`: ipk file on user local path.

<b>Usage</b>	<code>ip_catalog_install [-vlnv &lt;IP vlnv&gt;] [-file &lt;ipk file name&gt;]</code>
<b>Example</b>	<code>ip_catalog_install -vlnv latticesemi.com:ip:watchdog_timer:1.1.0 -file &lt;local path&gt;/watchdog_timer.ipk</code>

### 3.2.7. ip\_catalog\_uninstall

Uninstall one IP from local IPs, ip\_catalog\_list can list all installed local IPs.

args:

- `-vlnv`: Location of IP configuration file by vendor library name and version, use tcl 'ip\_catalog\_list' to get it. Usage: 'vendor:library:name:version'.

<b>Usage</b>	<code>ip_catalog_uninstall [-vlnv &lt;IP vlnv&gt;]</code>
<b>Example</b>	<code>ip_catalog_uninstall -vlnv latticesemi.com:ip:watchdog_timer:1.1.0</code>

### 3.2.8. sbp\_upgrade\_component

Reconfig component with latest IP version.

<b>Usage</b>	<code>sbp_upgrade_component -all/-component &lt;component name&gt; [-force_update] [-version &lt;version number&gt;]</code>
<b>Example</b>	<code>sbp_upgrade_component -component gpio0_inst -version 2.5.0</code>

### 3.2.9. sbp\_add\_gluelogic

Instantiates a gluelogic component into the system.

<b>Usage</b>	<code>sbp_add_gluelogic -name &lt;instance_name&gt; -logicinfo &lt;logic json info from sbp_create_glue_logic&gt;</code>
<b>Example</b>	<code>sbp_add_gluelogic -name equation_module_inst -logicinfo [sbp_create_glue_logic equation_module {"{"x": "A", "y": "B", "module": "equation_module"}]}</code>

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

### 3.2.10. 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.

<b>Usage</b>	<code>sbp_create_glue_logic &lt;type&gt; &lt;module_name&gt; &lt;rtl_path&gt; &lt;json cfg data&gt;</code>
<b>Example</b>	<code>sbp_create_glue_logic equation_module_inst -logicinfo [sbp_create_glue_logic equation_module {"{"x": "A", "y": "B", "module": "equation_module"}]}</code>

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

### 3.2.11. sbp\_reconfig\_gluelogic

Allow user to reconfig gluelogic component.

<b>Usage</b>	<code>sbp_reconfig_gluelogic -name &lt;instance_name&gt; -logicinfo &lt;logic json info from sbp_create_glue_logic&gt;</code>
<b>Example</b>	<code>sbp_reconfig_gluelogic equation_module0_inst -logicinfo [sbp_create_glue_logic equation_module0 {"{"expr": "A &amp; B", "module_name": "equation_module0"}]}</code>

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

### 3.2.12. sbp\_add\_port

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

<b>Usage</b>	<code>sbp_add_port [-from &lt;bit number&gt;] [-to &lt;bit number&gt;] -direction &lt;in/out/inout&gt; &lt;port_name&gt;</code>
--------------	---

### 3.2.13. sbp\_modify\_port

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

<b>Usage</b>	<code>sbp_modify_port -name &lt;port_name&gt; [-from &lt;bit number&gt;] [-to &lt;bit number&gt;] -direction &lt;dir.&gt;</code>
--------------	--

### 3.2.14. 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`.

<b>Usage</b>	<code>sbp_connect_net [-name &lt;net name&gt;] &lt;pin/port&gt; &lt;pin/port&gt;</code>
<b>Example</b>	<code>sbp_connect_net [sbp_get_pins clk] {CLOCK_IN}</code>

### 3.2.15. 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.

<b>Usage</b>	<code>sbp_connect_interface_net &lt;pin/port&gt; &lt;pin/port&gt;</code>
--------------	--

### 3.2.16. 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.

<b>Usage</b>	<code>sbp_connect_constant -constant &lt;integer&gt; &lt;pin/pin bus/ port/portbus&gt; &lt;pin/pin bus/ port/portbus&gt;...</code>
<b>Example</b>	<code>sbp_connect_constant -constant 1 {test/i2c_mst_apb/rst_n_i} {test/riscv/clk_i}</code>

### 3.2.17. sbp\_connect\_whitebox

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

<b>Usage</b>	<code>sbp_connect_whitebox &lt;design_name/vip_inst_name/pin_name&gt; &lt;design_name/uit_inst_name/port_name&gt;</code>
--------------	--

### 3.2.18. sbp\_connect\_group

Allow user to connect a group of signals.

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

### 3.2.19. sbp\_disconnect\_whitebox

Removes the connection cross the hierarchy boundary.

<b>Usage</b>	<code>sbp_disconnect_whitebox &lt;design_name/vip_inst_name/pin_name&gt; &lt;design_name/uit_inst_name/port_name&gt;</code>
--------------	---

### 3.2.20. 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.

<b>Usage</b>	<code>sbp_disconnect_interface_net &lt;pin/port&gt; &lt;pin/port&gt;</code>
--------------	---

### 3.2.21. 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.

<b>Usage</b>	sbp_disconnect_net <pin0> <pin1> <port2>
--------------	--

### 3.2.22. sbp\_assign\_addr\_seg

Assigns a memory map between a pair of manager and subordinate 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

<b>Usage</b>	sbp_assign_addr_seg -offset <offset> < subordinate connection name>
<b>Example</b>	sbp_assign_addr_seg -offset 32'h00001000 simple/riscv/AHBL_S00

### 3.2.23. 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.

<b>Usage</b>	sbp_unassign_addr_seg < subordinate interface name>
<b>Example</b>	sbp_unassign_addr_seg simple/spi/AHB_S00

### 3.2.24. sbp\_assign\_local\_memory

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

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

### 3.2.25. 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.

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

### 3.2.26. 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.

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

### 3.2.27. 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.

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

### 3.2.28. 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.

<b>Usage</b>	<code>sbp_replace -vlv &lt;VLNV&gt; -name &lt;instance&gt; -component &lt;component name&gt;</code>
<b>Example</b>	<code>sbp_replace -vlv lattice:ip:ahblite_bus_0:1.1 -name ahbl_bus_0 -component simple/ahblite</code>

### 3.2.29. 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.

<b>Usage</b>	<code>sbp_copy -postfix &lt;postfixString&gt; objects</code>
<b>Example</b>	<code>sbp_copy -postfix X \$selected_objs</code>

### 3.2.30. 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.

<b>Usage</b>	<code>Sbp_delete objects -type &lt;type name&gt;</code>
<b>Example 1</b>	<code>sbp_delete [sbp_get_ports &lt;clock&gt;] -type port</code>
<b>Example 2</b>	<code>sbp_delete {ddr3} -type component</code>

### 3.2.31. 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.

<b>Usage</b>	<code>sbp_get_pins [-from &lt;object Name&gt;] [pattern]</code>
<b>Example</b>	<code>sbp_get_pins -from ahblite_interconnect clk</code>

### 3.2.32. 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 subordinate interface pins from the interconnect IP.

<b>Usage:</b>	<code>sbp_get_interface_pins [-from &lt;objectName&gt;] [pattern]</code>
<b>Example:</b>	<code>sbp_get_interface_pins -from ahblite_interconnect S*_AHB</code>

### 3.2.33. 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.

<b>Usage</b>	<code>sbp_get_ports [-from &lt;objectName&gt;] [pattern]</code>
--------------	---

### 3.2.34. 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.

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

### 3.2.35. 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.

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

### 3.2.36. 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.

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

### 3.2.37. 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 manager interfaces to two and number of subordinate interface to three on AHB-Lite interconnect block named “ahbl\_interconnect”.

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

### 3.2.38. sbp\_get\_property

Gets the property of the object. The example below is to get the number of subordinate interfaces of AHB-Lite interconnect block named “ahbl\_interconnect\_0”.

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

### 3.2.39. sbp\_report\_properties

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

<b>Usage</b>	sbp_report_properties object
<b>Example</b>	sbp_report_properties ahbl_interconnect
<b>Outputs</b>	Name: ahbl_interconnect NUM_SI: 1 NUM_MI: 2 DATA_WIDTH: 32 ADDRESS_WIDTH: 32

### 3.2.40. 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.

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

### 3.2.41. sbp\_design set\_prj\_option

Change the output language (Verilog or VHDL) after the project was created.

<b>Usage</b>	sbp_design set_prj_option -language <Verilog or VHDL>
<b>Example</b>	sbp_design set_prj_option -language Verilog

### 3.2.42. sbp\_design gen\_tcl

Generate TCL file for design project in project folder.

<b>Usage</b>	sbp_design gen_tcl
<b>Example</b>	sbp_design gen_tcl

## References

- [Lattice Propel 2024.1 SDK User Guide \(FPGA-UG-02211\)](#)
- [Lattice IP Packager 2024.1 User Guide \(FPGA-UG-02213\)](#)
- [Lattice Propel Revision Control User Guide \(FPGA-UG-02214\)](#)
- [Lattice Propel 2024.1 Installation for Windows User Guide \(FPGA-AN-02088\)](#)
- [Lattice Propel 2024.1 Installation for Linux User Guide \(FPGA-AN-02089\)](#)
- [Lattice Propel 2024.1 Release Notes \(FPGA-AN-02090\)](#)
- [Lattice Diamond Help](#)
- [Lattice Radiant Help](#)

## Technical Support Assistance

Submit a technical support case through [www.latticesemi.com/techsupport](http://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 2024

Section	Change Summary
All	Production release.



[www.latticesemi.com](http://www.latticesemi.com)