Lattice Propel Builder 1.1

User Guide

FPGA-UG-02116-1.0

November 2020

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

Contents

Figures

Glossary

A list of words or terms specialized in this document.

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

1. Introduction

Lattice Propel Builder 1.1 is a graphical tool used to assemble complex System-on-Chip (SoC) modules which can be used in the supported Lattice FPGA devices. These module IPs can be assembled and connected easily by simply dragging and dropping the IPs 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 Builder 1.1 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, LIFCL, LFD2NX and LFMNX devices. The technical guidelines assume readers have expertise in the embedded system area and FPGA technologies.

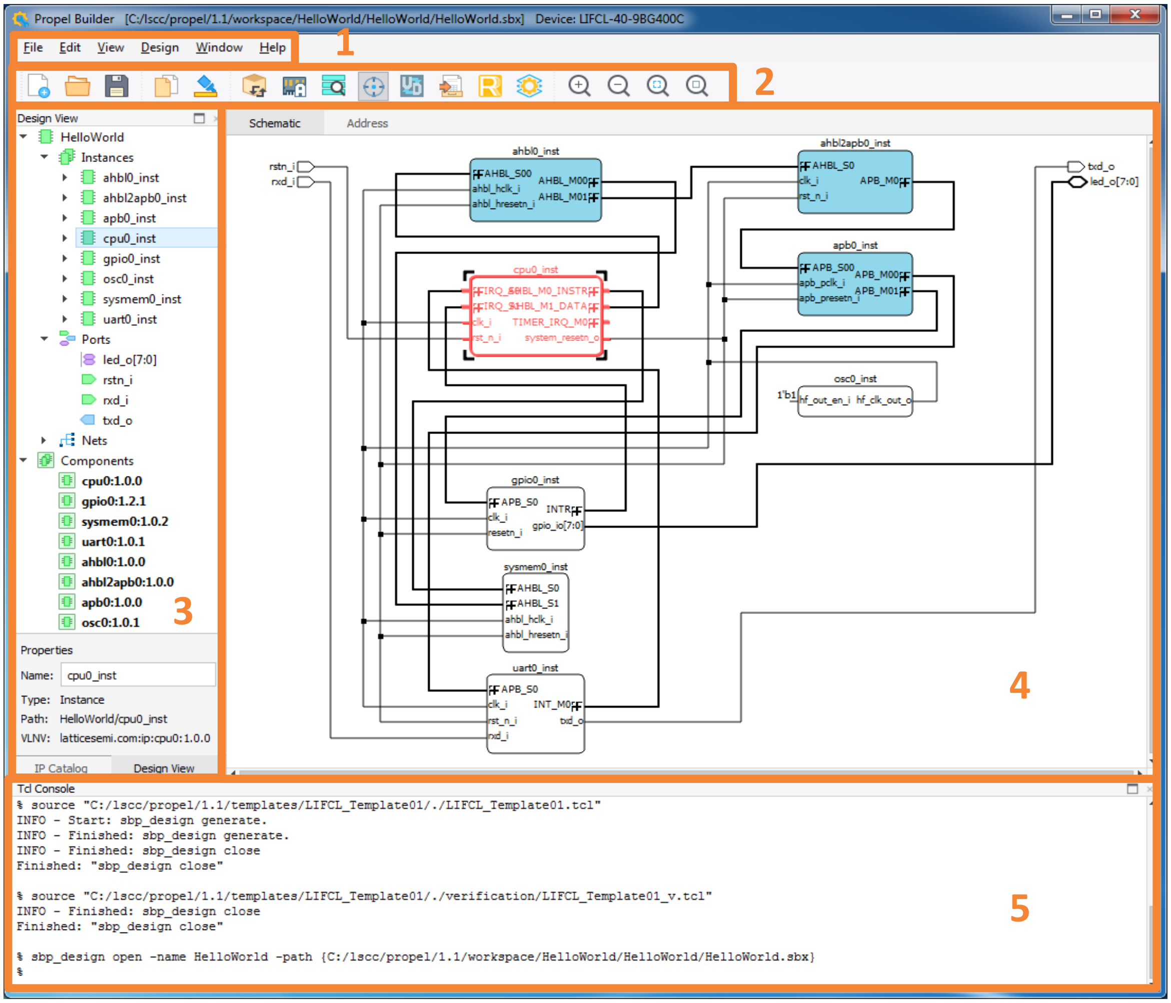
2. Lattice Propel Builder Design Flows

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

2.1. Builder Environment

After Propel 1.1 is installed, you can launch the stand-alone Propel Builder by double-clicking the Builder icon to launch Builder. Refer to the Lattice Propel Installation User Guide (FPGA-AN-02030) for details on the installation. After the Propel Builder is launched, a single workbench window is displayed. The workbench contains Menu, Toolbar, Design Info, IP catalog, Schematic view, address mapping, and TCL console. Figure 2.1 shows the workbench with opening a project.

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



Name: cpu0_inst
Type: Instance
Path: HelloWorld/cpu0_inst
VLNV: latticesemi.com:ip:cpu0:1.0.0

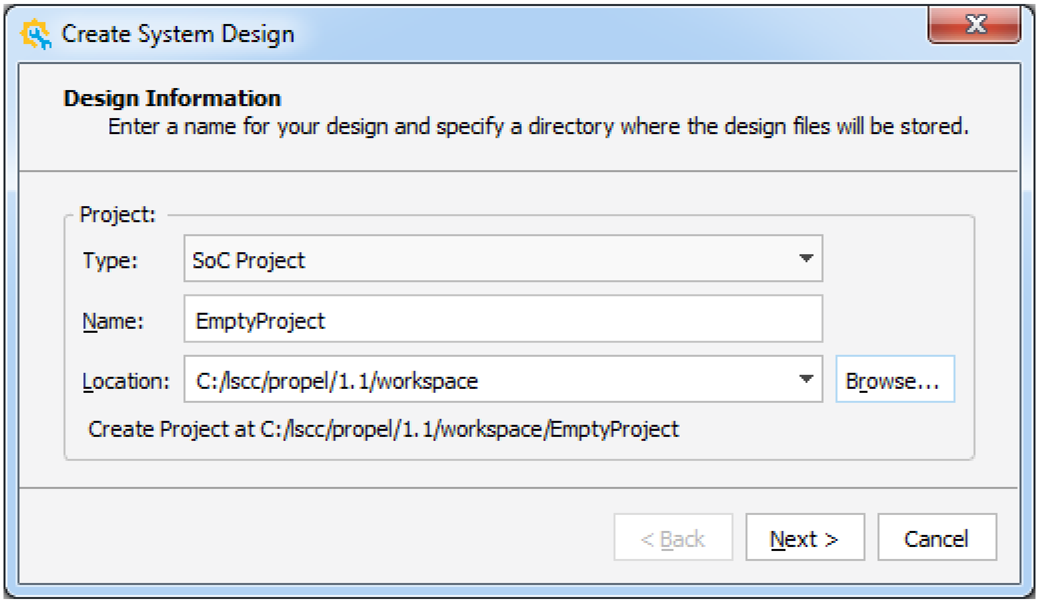
```
% source "C:/lscc/propel/1.1/templates/LIFCL_Template01/./LIFCL_Template01.tcl"
INFO - Start: sbp_design generate.
INFO - Finished: sbp_design generate.
INFO - Finished: sbp_design close
Finished: "sbp_design close"

% source "C:/lscc/propel/1.1/templates/LIFCL_Template01/./verification/LIFCL_Template01_v.tcl"
INFO - Finished: sbp_design close
Finished: "sbp_design close"

% sbp_design open -name HelloWorld -path {C:/lscc/propel/1.1/workspace/HelloWorld/HelloWorld/HelloWorld.sbx}
%
```

Figure 2.1. Propel Builder Workbench Window

2.2. Project Design Flow

2.2.1. Creating a New SoC Project

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



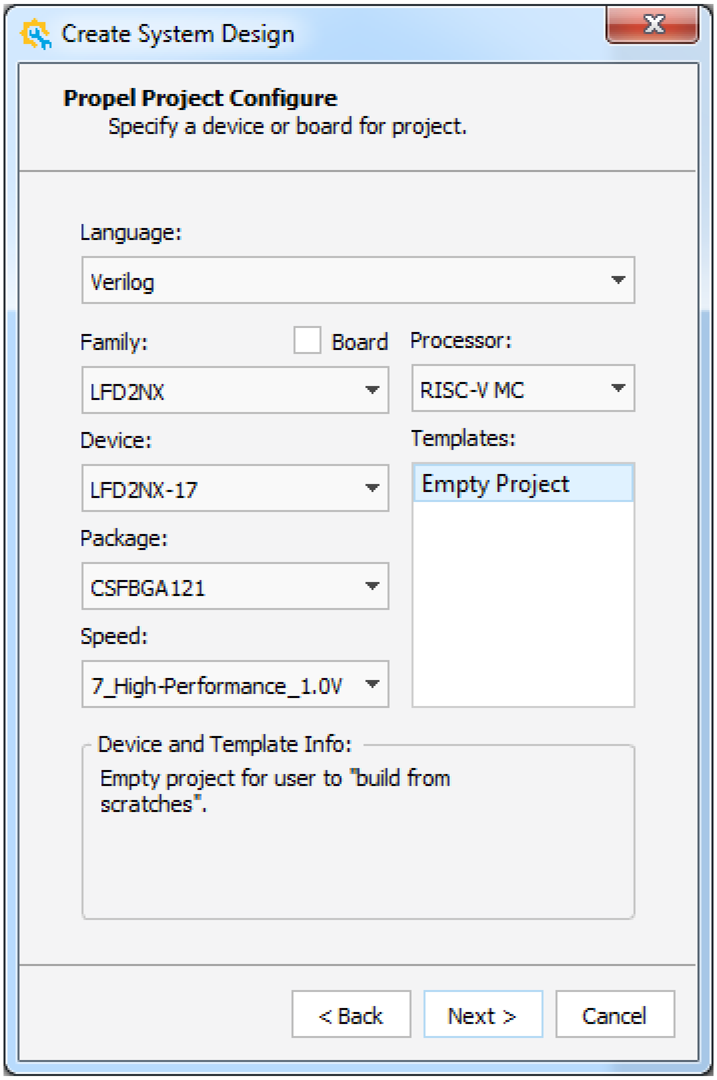
Create System Design

Design Information
Enter a name for your design and specify a directory where the design files will be stored.

Project:
Type: SoC Project
Name: EmptyProject
Location: C:/lscc/propel/1.1/workspace
Create Project at C:/lscc/propel/1.1/workspace/EmptyProject

Figure 2.2. Create System Design – Design Information Wizard

2. The default Project Type is displayed in the "**Type"** field. Use the drop-down menu to choose SoC Verification if creating verification project.
3. Enter the desired project name in the "**Name"** filed.
4. (Optional) The default location is shown in the "**Location"** field. Use the "Browse…" option to change the project location.
5. Click **Next**. The Create System Design - Propel Project Configure wizard opens (Figure 2.3).



Propel Project Configure
Specify a device or board for project.

Language: Verilog

Family: LFD2NX □ Board

Processor: RISC-V MC

Device: LFD2NX-17

Templates: Empty Project

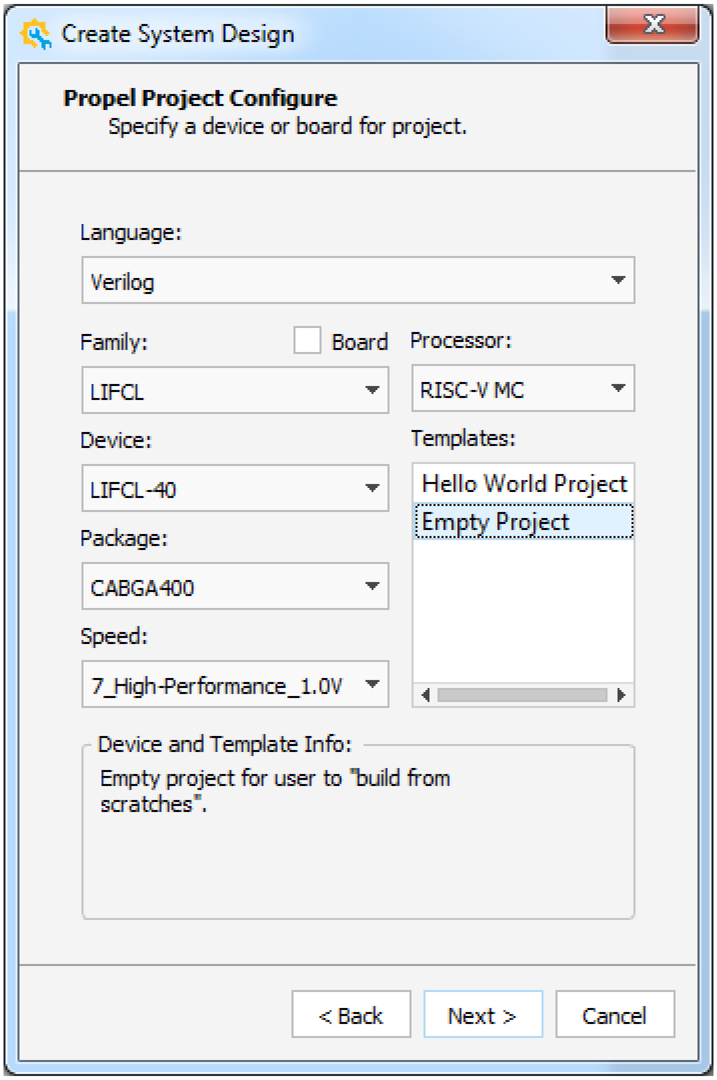
Package: CSFBGA121

Speed: 7_High-Performance_1.0V

Device and Template Info:
Empty project for user to "build from scratches".

Figure 2.3. Create System Design - Propel Project Configure Wizard

6. (Optional) The default Hardware Description Language (HDL) is displayed in the **Language** field. Use the drop-down menu to change the default language.
7. Specify a device or board for project.
   - Use the drop-down menu to select desired device information (Family, Device, Package and Speed), and select Empty Project in the **Templates** section (Figure 2.4).



Create System Design

Propel Project Configure
Specify a device or board for project.

Language: Verilog

Family: LIFCL

□ Board

Processor: RISC-V MC

Device: LIFCL-40

Templates:
Hello World Project
Empty Project

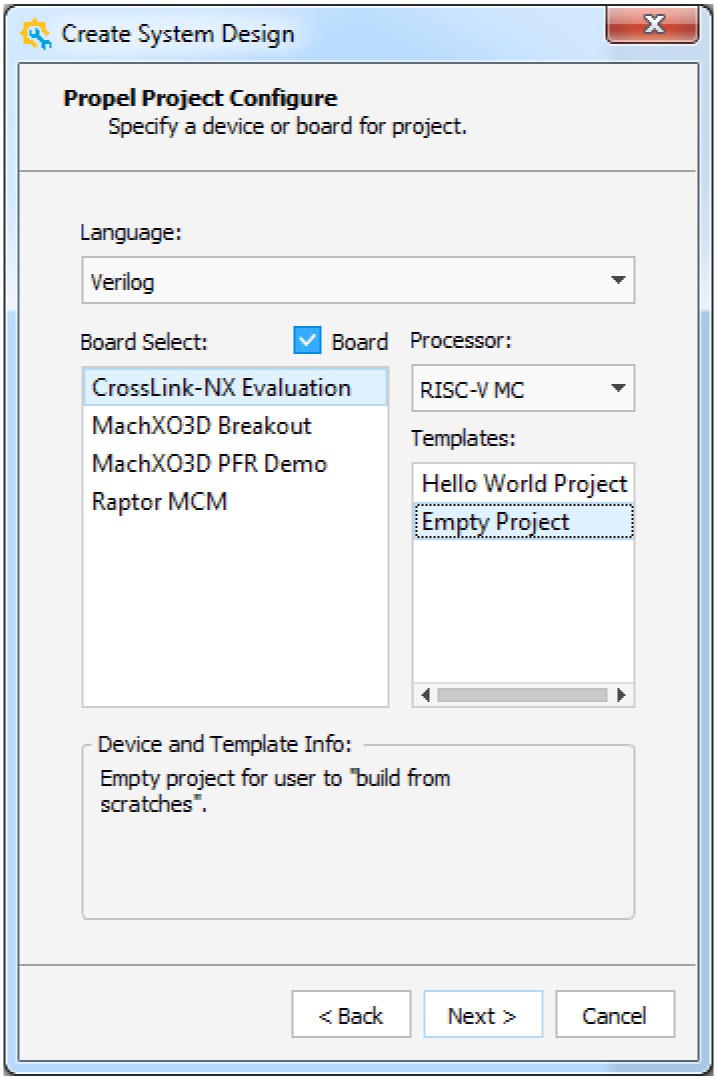
Package: CABGA400

Speed: 7_High-Performance_1.0V

Device and Template Info:
Empty project for user to "build from scratches".

Figure 2.4. Specify a Device for Project

- Click **Board**, Board Select area shows in Propel Project Configure Wizard (Figure 2.5), select desired board such as Crosslink-NX Evaluation.



Create System Design

Propel Project Configure
Specify a device or board for project.

Language: Verilog

Board Select: ☑ Board

- CrossLink-NX Evaluation
- MachXO3D Breakout
- MachXO3D PFR Demo
- Raptor MCM

Processor: RISC-V MC

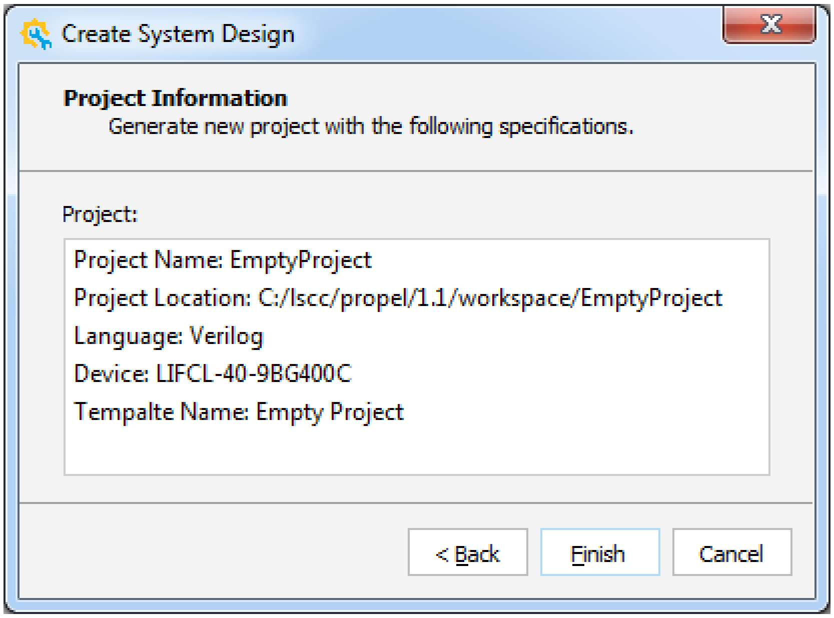
Templates:

- Hello World Project
- Empty Project

Device and Template Info:
Empty project for user to "build from scratches".

Figure 2.5 Specify a Board for Project

8. Click **Next**. The Project Information wizard opens (Figure 2.6).



Create System Design

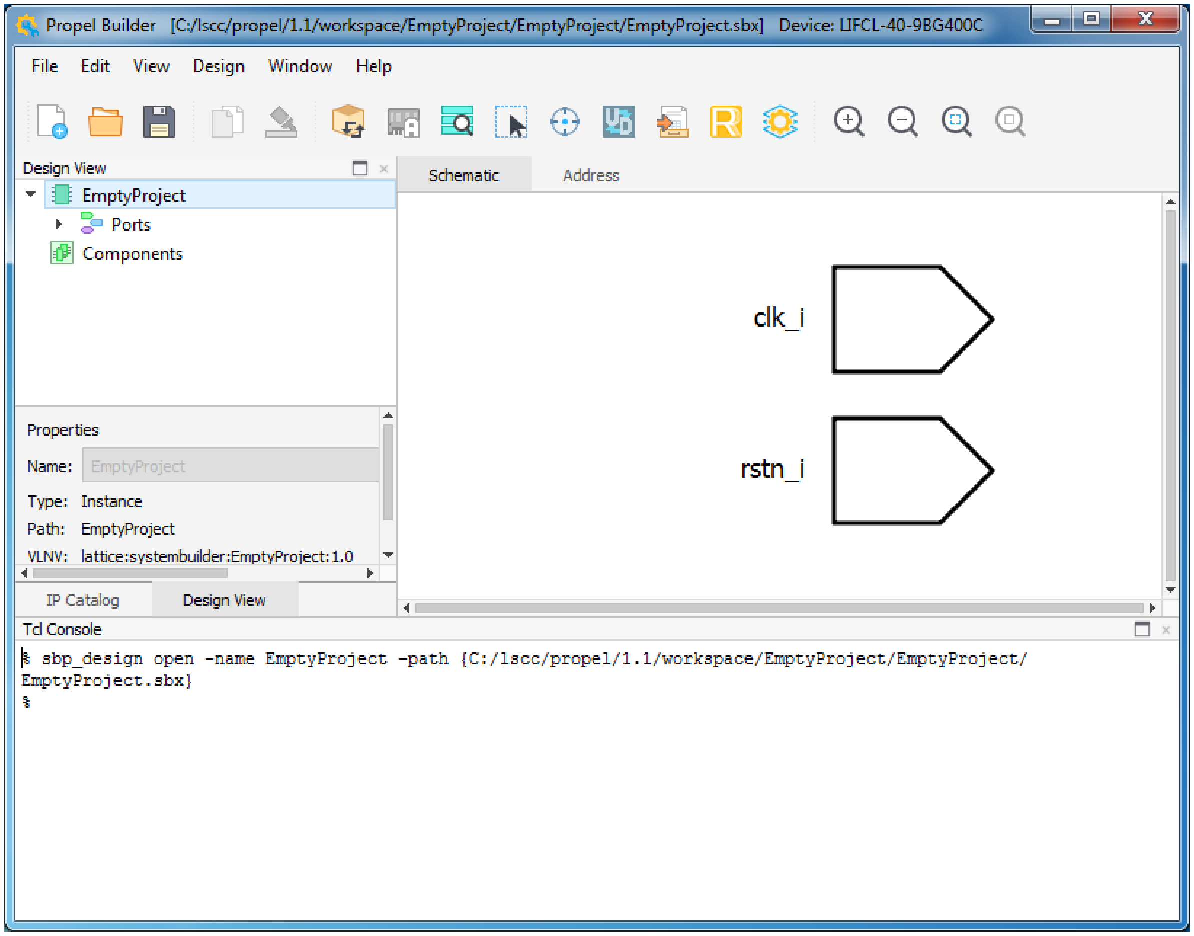
Project Information
Generate new project with the following specifications.

Project:

Project Name: EmptyProject
Project Location: C:/lscc/propel/1.1/workspace/EmptyProject
Language: Verilog
Device: LIFCL-40-9BG400C
Tempalte Name: Empty Project

Figure 2.6. Create System Design - Project Information Wizard

9. Click **Finish**. Propel Builder GUI opens (Figure 2.7).



Propel Builder [C:/lscc/propel/1.1/workspace/EmptyProject/EmptyProject/EmptyProject.sbx] Device: LIFCL-40-9BG400C

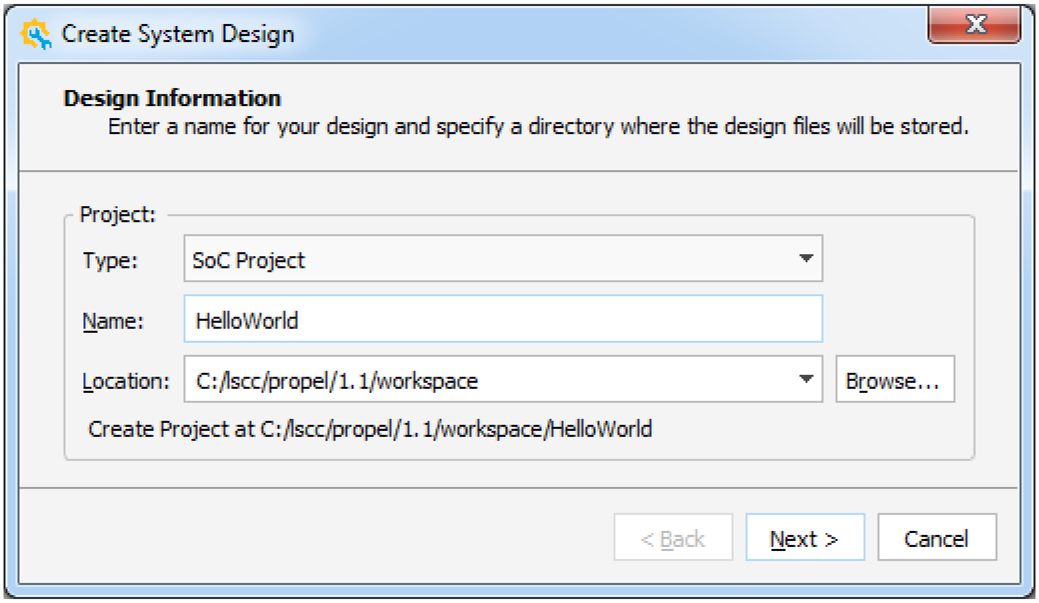
Properties
Name: EmptyProject
Type: Instance
Path: EmptyProject
VLNV: lattice:systembuilder:EmptyProject:1.0

```
% sbp_design open -name EmptyProject -path {C:/lscc/propel/1.1/workspace/EmptyProject/EmptyProject/
EmptyProject.sbx}
%
```

Figure 2.7. Propel Builder GUI Shows Empty Project

2.2.2. Creating Template SoC Project

1. Choose **File** > **New Design** from the Lattice Propel Builder Menu bar. The Create System Design wizard opens (Figure 2.8).



Create System Design

Design Information

Enter a name for your design and specify a directory where the design files will be stored.

Project:

Type: SoC Project

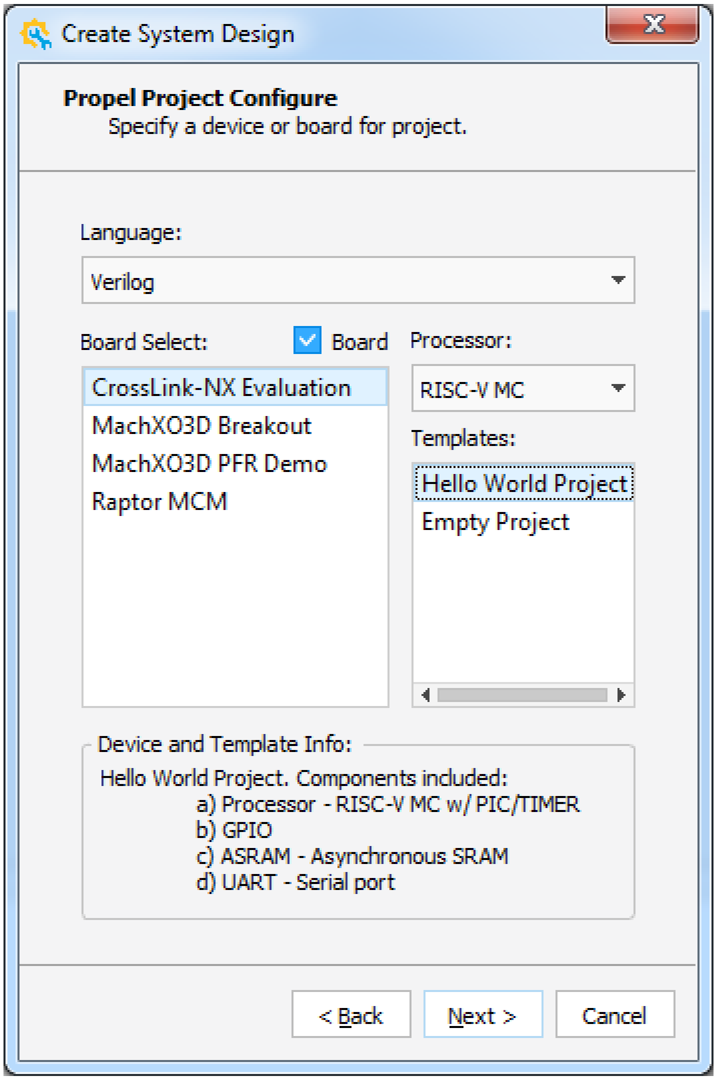
Name: HelloWorld

Location: C:/lscc/propel/1.1/workspace

Create Project at C:/lscc/propel/1.1/workspace/HelloWorld

Figure 2.8. Create System Design – Design Information Wizard

2. The default Project Type is displayed in the "**Type"** field. Use the drop-down menu to choose SoC Verification if creating verification project.
3. Enter a project name in the "**Name"** field, such as HelloWorld.
4. (Optional) The default location is shown in the **"Location"** field. Use the "Browse…" option to change the project location.
5. Click **Next**. The Propel Project Configure wizard opens (Figure 2.3).
6. Click **Board**, Board Select area shows in Propel Project Configure Wizard, Select a desired Board such as Crosslink-NX Evaluation and select desired template in the **Templates** section, such as Hello World Project (Figure 2.9).



Create System Design

Propel Project Configure

Specify a device or board for project.

Language: Verilog

Board Select: ☑ Board

CrossLink-NX Evaluation
MachXO3D Breakout
MachXO3D PFR Demo
Raptor MCM

Processor: RISC-V MC

Templates:

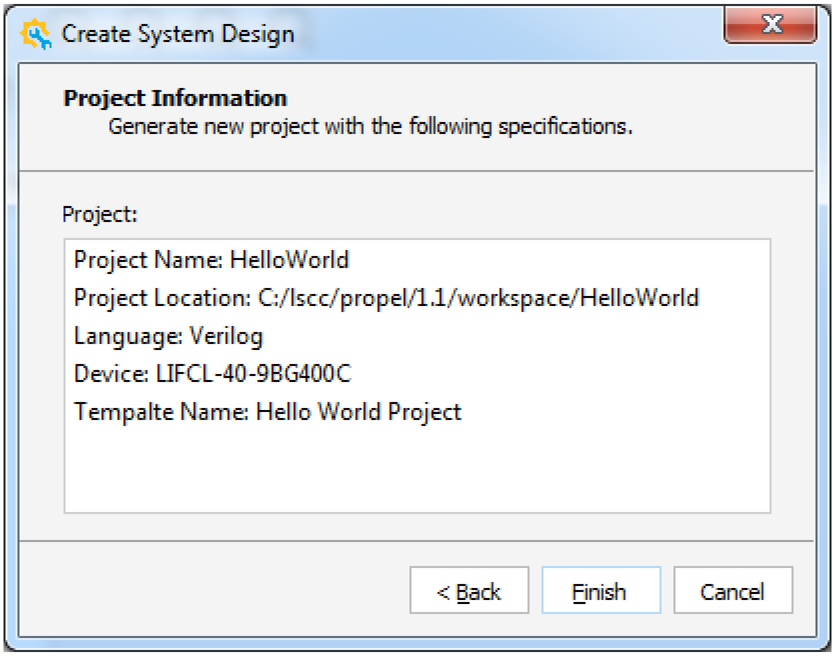
Hello World Project
Empty Project

Device and Template Info:

Hello World Project. Components included:
a) Processor - RISC-V MC w/ PIC/TIMER
b) GPIO
c) ASRAM - Asynchronous SRAM
d) UART - Serial port

Figure 2.9. Specify a Board for Project

7. Click **Next**. The Project Information wizard opens (Figure 2.10).



Project Information

Generate new project with the following specifications.

Project:

Project Name: HelloWorld

Project Location: C:/lscc/propel/1.1/workspace/HelloWorld

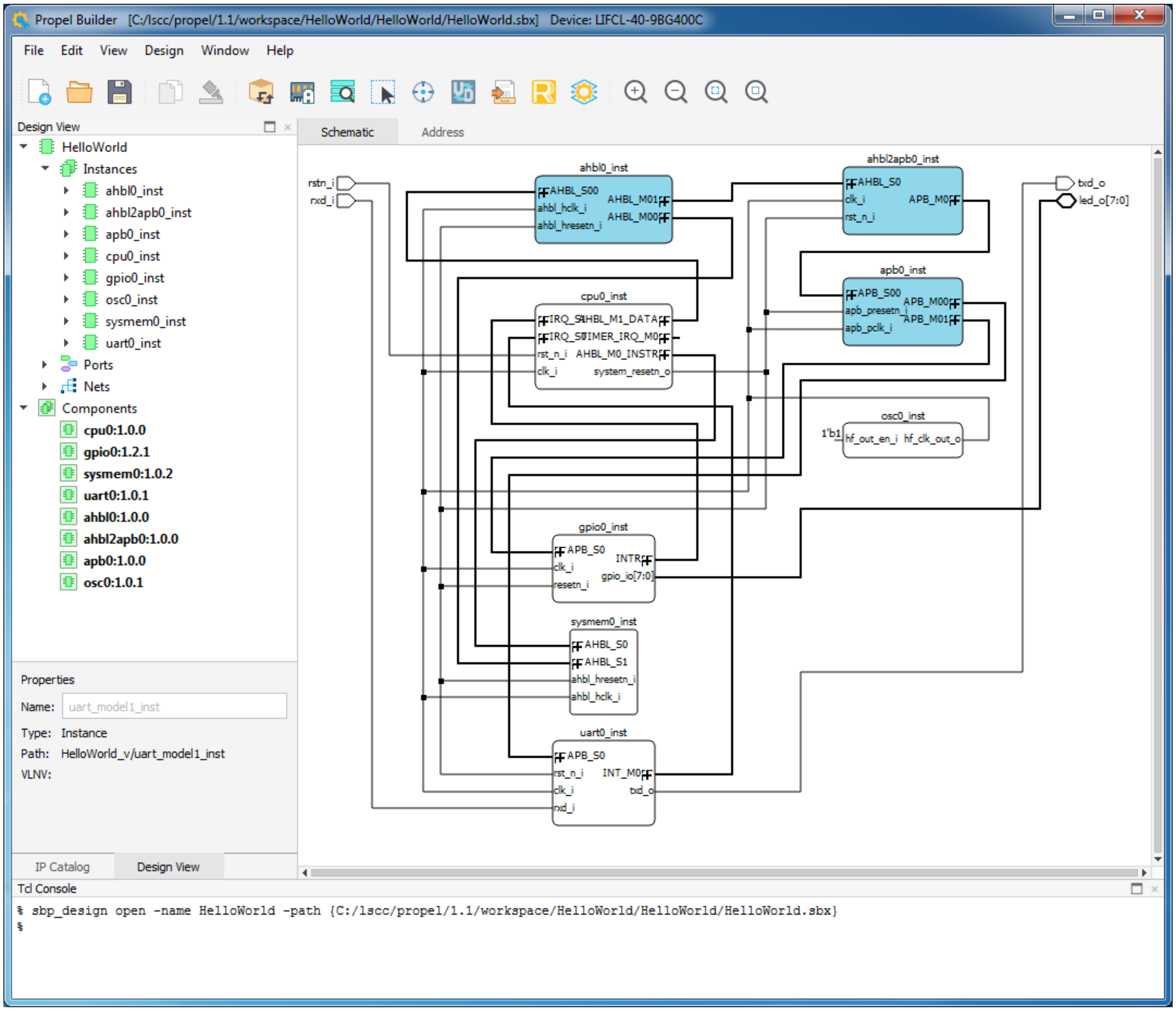
Language: Verilog

Device: LIFCL-40-9BG400C

Tempalte Name: Hello World Project

Figure 2.10. Project Information Wizard

8. Click **Finish**. Propel Builder GUI opens (Figure 2.11).



Properties

Name: uart_model1_inst

Type: Instance

Path: HelloWorld_v/uart_model1_inst

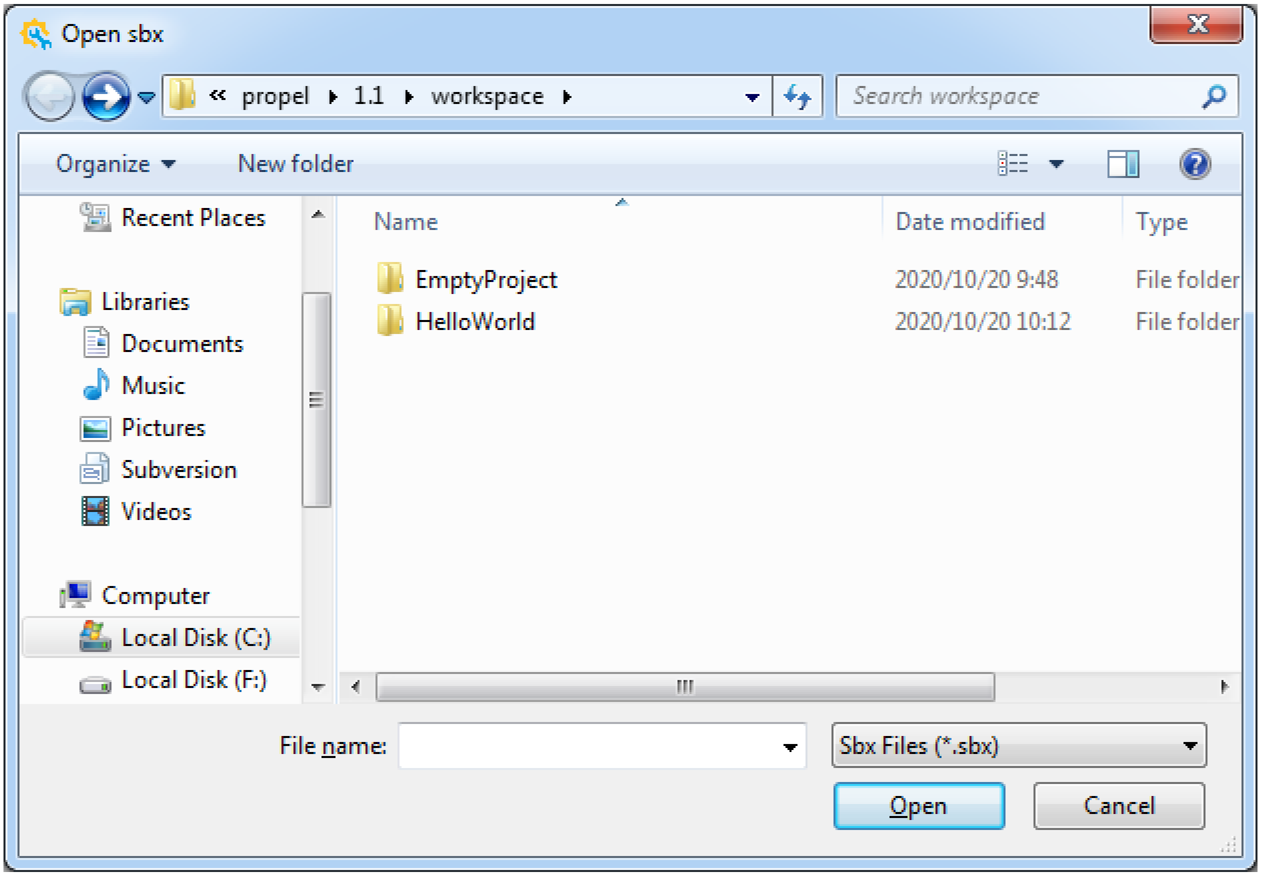
VLNV:

```
% sbp_design open -name HelloWorld -path {C:/lscc/propel/1.1/workspace/HelloWorld/HelloWorld/HelloWorld.sbx}
%
```

Figure 2.11. Propel Builder GUI shows Template Project

2.2.3. Opening an SoC Project

1. Choose **File** > 📂 **Open Design** from Propel Builder GUI Menu or Click **Open Design** icon 📂 from Propel Builder GUI Toolbar. The Open sbx dialog opens (Figure 2.12)



Name	Date modified	Type
EmptyProject	2020/10/20 9:48	File folder
HelloWorld	2020/10/20 10:12	File folder

File name:

Sbx Files (*.sbx)

Figure 2.12. Open Sbx Dialog

2. Enter the sbx file path of an existing SoC Project such as HelloWorld (Figure 2.13).

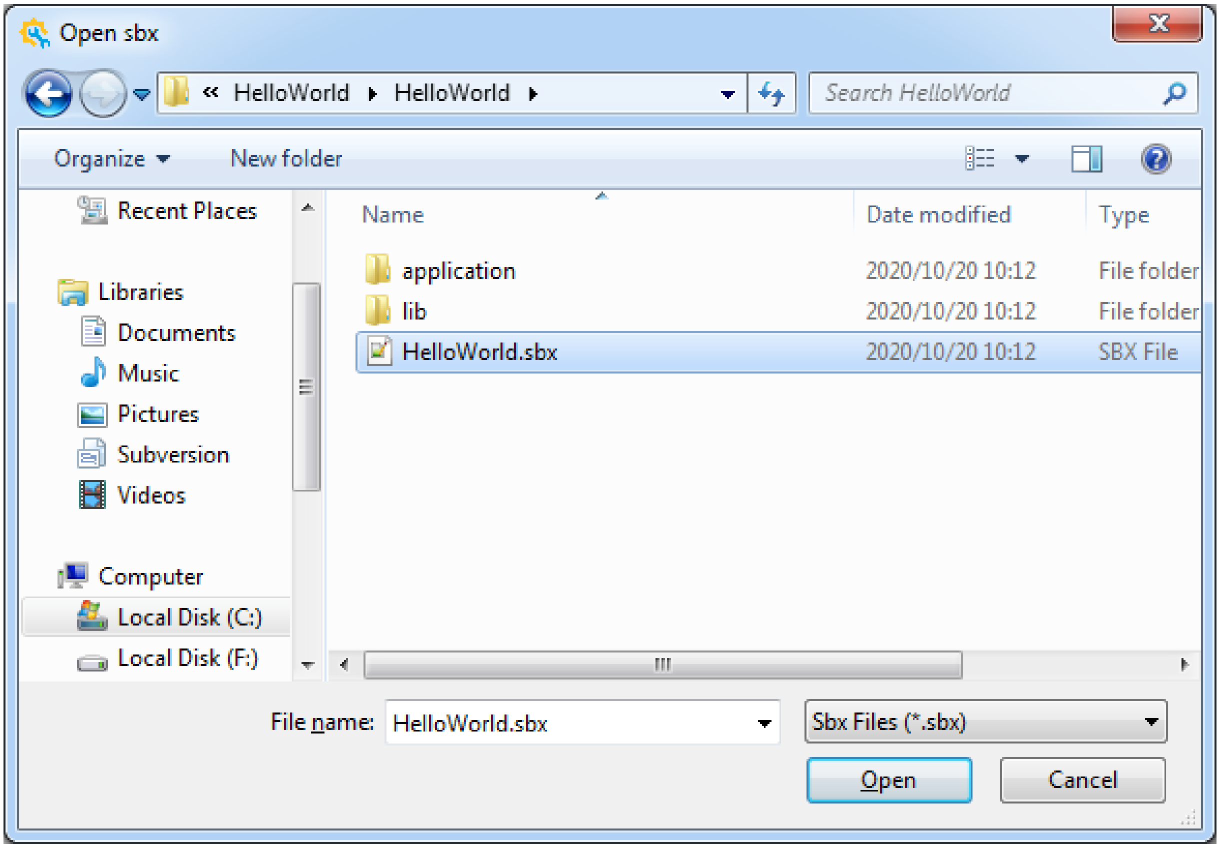


Figure 2.13. Open HelloWorld Project

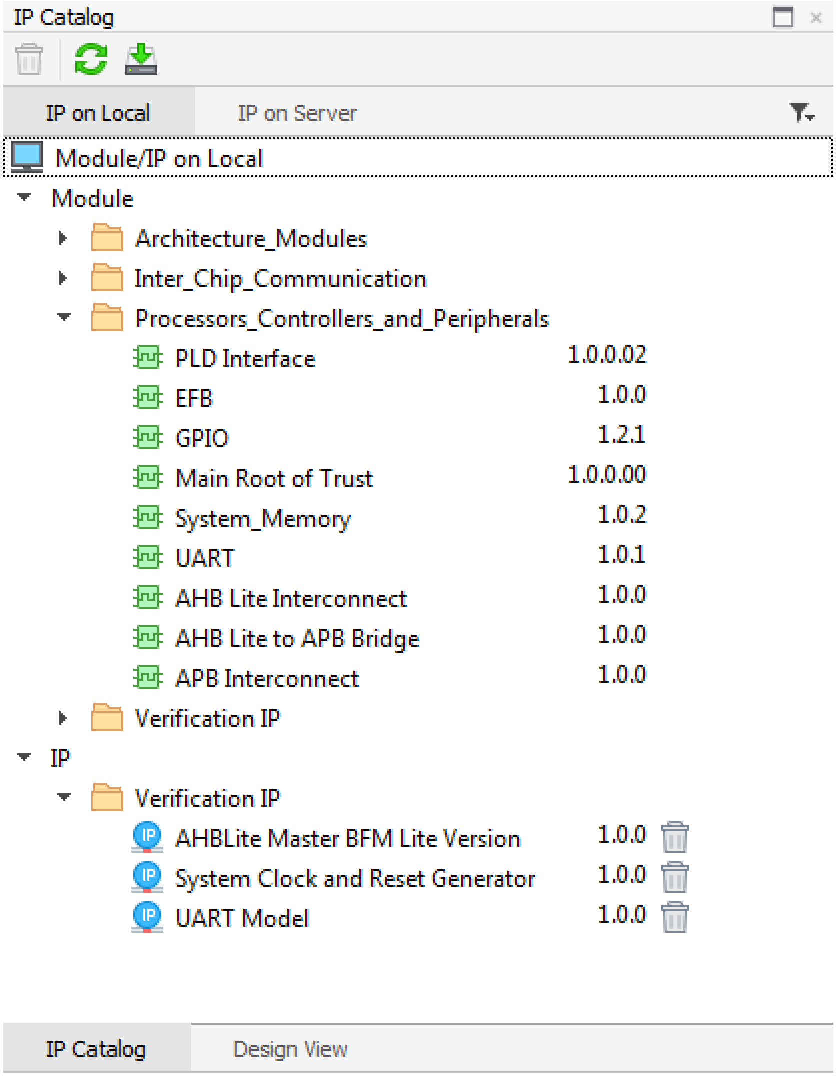
3. Click **Open**. The Builder GUI shows the SoC design.

Note: Choose **File** > **Recent Designs** from Propel Builder GUI Menu, can quickly open a recent closed project.

2.2.4. Adding Modules

After starting a Propel Builder project, you can add modules by dragging them from the IP Catalog view to Schematic view. The IP Catalog view comes with a large variety of commonly used modules. These are under the IP on Local tab. Click the IP on Server tab to see more specialized modules that you can download.

1. (Optional) From the Propel Builder GUI **IP Catalog** area (Figure 2.14), choose the **IP on Server** view. Select a desired IP. Click the ⬇ **Install** button. After the IP is installed successfully, the new IP can be shown in the **IP on Local** tab.



Module/IP on Local

- Module
  - Architecture_Modules
  - Inter_Chip_Communication
  - Processors_Controllers_and_Peripherals

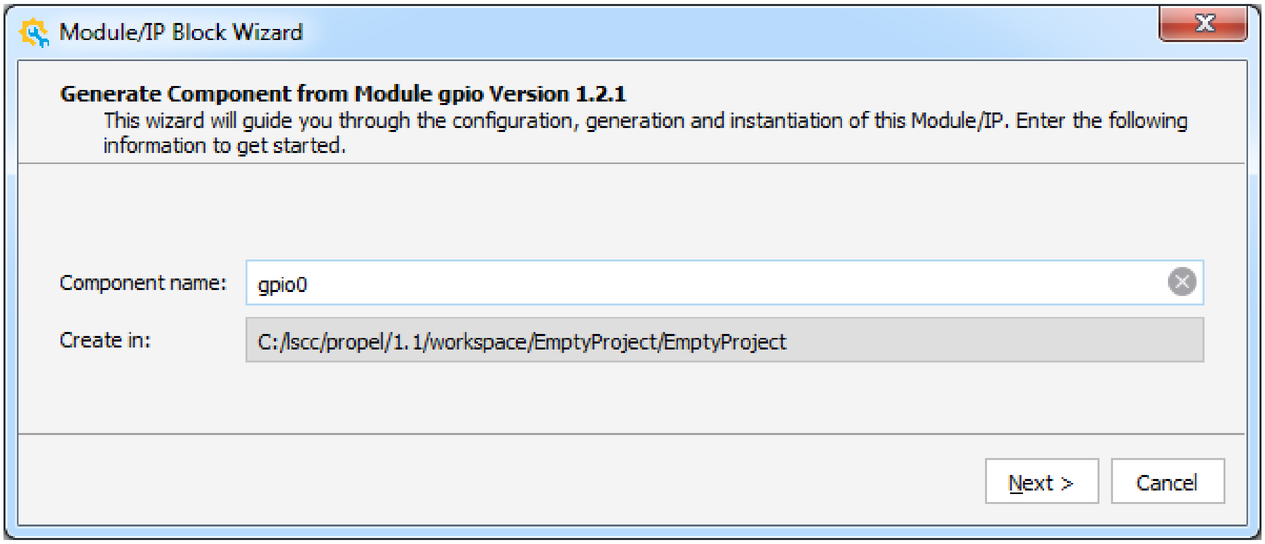
PLD Interface	1.0.0.02
EFB	1.0.0
GPIO	1.2.1
Main Root of Trust	1.0.0.00
System_Memory	1.0.2
UART	1.0.1
AHB Lite Interconnect	1.0.0
AHB Lite to APB Bridge	1.0.0
APB Interconnect	1.0.0

  - Verification IP
- IP
  - Verification IP

AHBLite Master BFM Lite Version	1.0.0
System Clock and Reset Generator	1.0.0
UART Model	1.0.0

Figure 2.14. IP Catalog

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



Generate Component from Module gpio Version 1.2.1

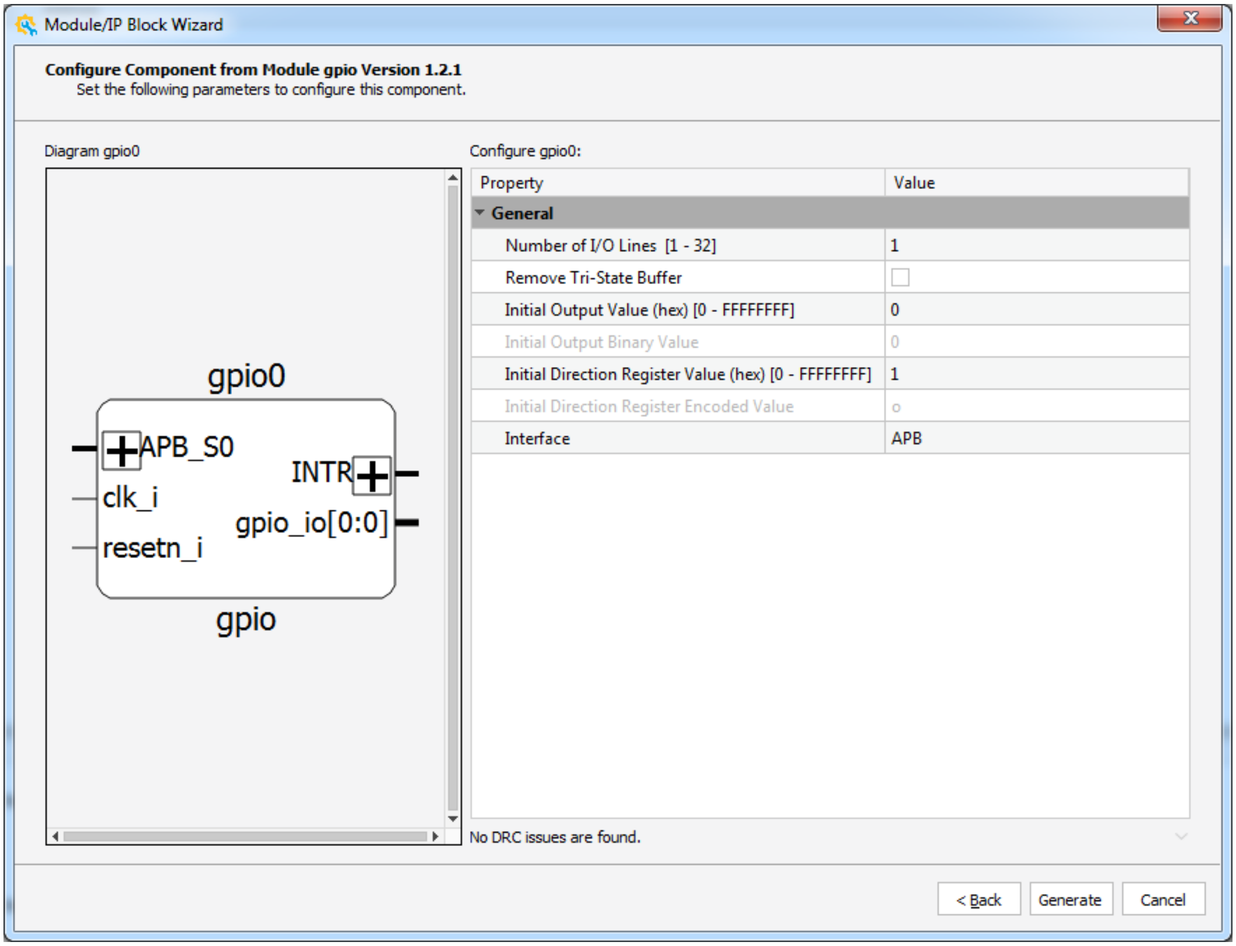
This wizard will guide you through the configuration, generation and instantiation of this Module/IP. Enter the following information to get started.

Component name: gpio0

Create in: C:/lscc/propel/1.1/workspace/EmptyProject/EmptyProject

Figure 2.15. Module/IP Block Wizard – Generate Component from Module gpio Version 1.2.1

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



Configure Component from Module gpio Version 1.2.1

Set the following parameters to configure this component.

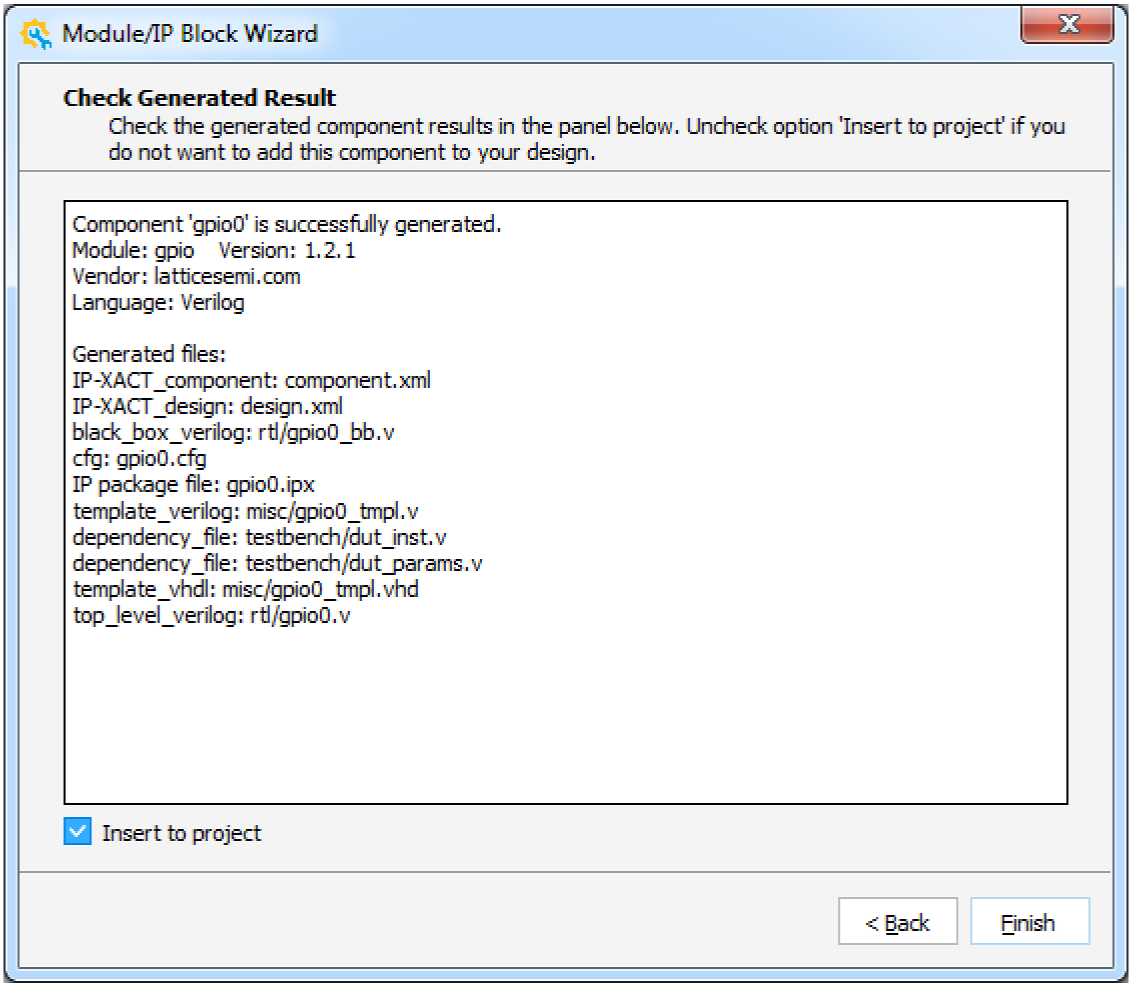
Configure gpio0:

Property	Value
General	
Number of I/O Lines [1 - 32]	1
Remove Tri-State Buffer	□
Initial Output Value (hex) [0 - FFFFFFFF]	0
Initial Output Binary Value	0
Initial Direction Register Value (hex) [0 - FFFFFFFF]	1
Initial Direction Register Encoded Value	o
Interface	APB

No DRC issues are found.

Figure 2.16. Module/IP Block Wizard - Configure Component from Module Gpio Version 1.2.1

4. (Optional)Configure the component to set the property value at General form. The property in gray font is non-configurable.
5. Click **Generate**. Module/IP Block wizard shows the Check Generated Result page (Figure 2.17).



Check Generated Result

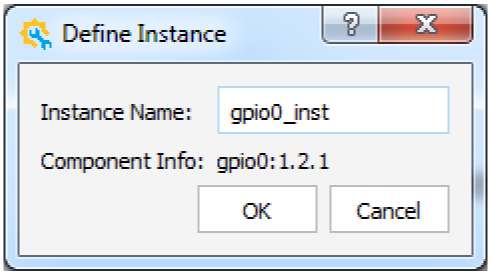
Check the generated component results in the panel below. Uncheck option 'Insert to project' if you do not want to add this component to your design.

```
Component 'gpio0' is successfully generated.
Module: gpio    Version: 1.2.1
Vendor: latticesemi.com
Language: Verilog

Generated files:
IP-XACT_component: component.xml
IP-XACT_design: design.xml
black_box_verilog: rtl/gpio0_bb.v
cfg: gpio0.cfg
IP package file: gpio0.ipx
template_verilog: misc/gpio0_tmpl.v
dependency_file: testbench/dut_inst.v
dependency_file: testbench/dut_params.v
template_vhdl: misc/gpio0_tmpl.vhd
top_level_verilog: rtl/gpio0.v
```

☑ Insert to project

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

6. Select **Insert to project** at Check Generate Result wizard.
7. Click **Finish**. The **Define Instance** dialog box opens (Figure 2.18).

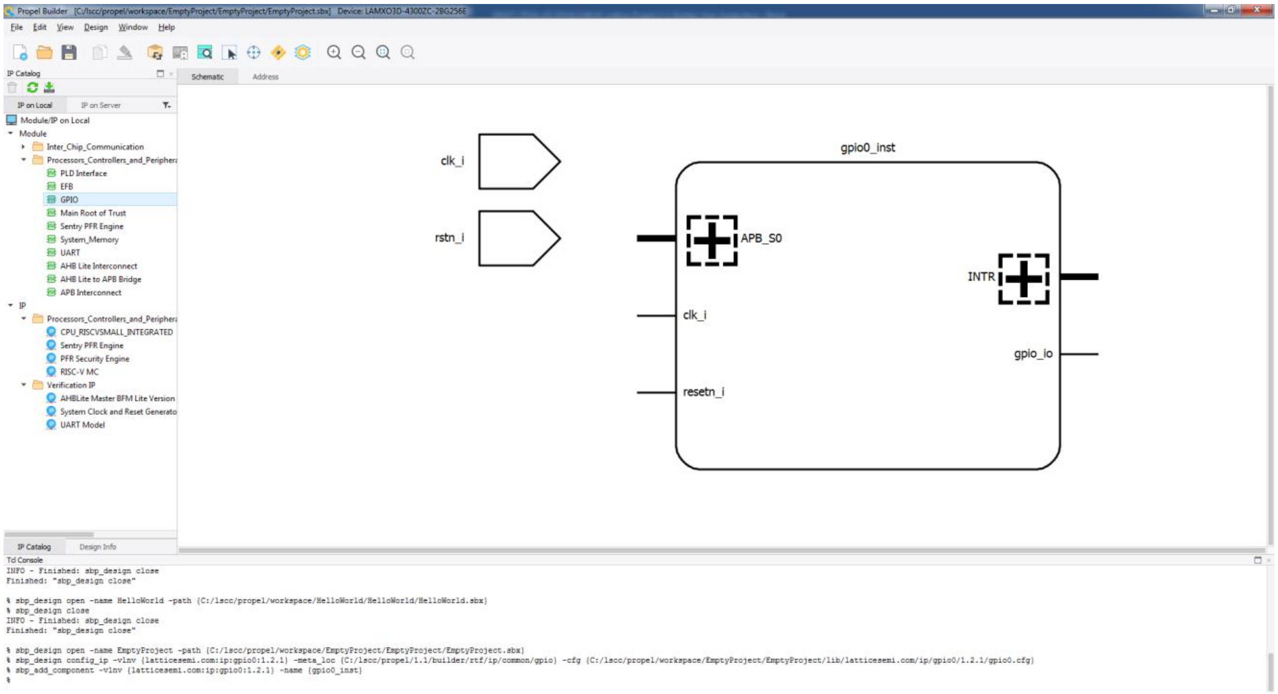


Instance Name: gpio0_inst

Component Info: gpio0:1.2.1

Figure 2.18. Design Instance Dialog Box

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



```
INFO - Finished: sbp_design close
Finished: "sbp_design close"

% sbp_design open -name HelloWorld -path {C:/lscc/propel/workspace/HelloWorld/HelloWorld/HelloWorld.sbx}
% sbp_design close
INFO - Finished: sbp_design close
Finished: "sbp_design close"

% sbp_design open -name EmptyProject -path {C:/lscc/propel/workspace/EmptyProject/EmptyProject/EmptyProject.sbx}
% sbp_design config_ip -vlnv {latticesemi.com:ip:gpio0:1.2.1} -meta_loc {C:/lscc/propel/1.1/builder/rtf/ip/common/gpio} -cfg {C:/lscc/propel/workspace/EmptyProject/EmptyProject/lib/latticesemi.com/ip/gpio0/1.2.1/gpio0.cfg}
% sbp_add_component -vlnv {latticesemi.com:ip:gpio0:1.2.1} -name {gpio0_inst}
%
```

Figure 2.19. Propel Builder Schematic View Shows Module Instance

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.

- To view signal list of block:

  Click the plus sign ⊞ of the desired block to see what signals it contains. The plus sign changes to a negative sign ⊟ and shows the signal list (Figure 2.20). Click the negative sign ⊟ to close the expanded bus. The schematic returns to its previous form.

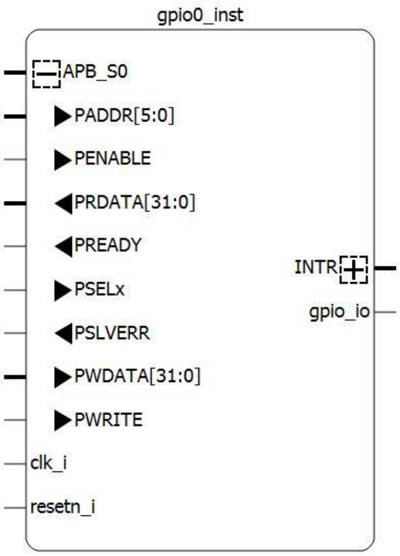
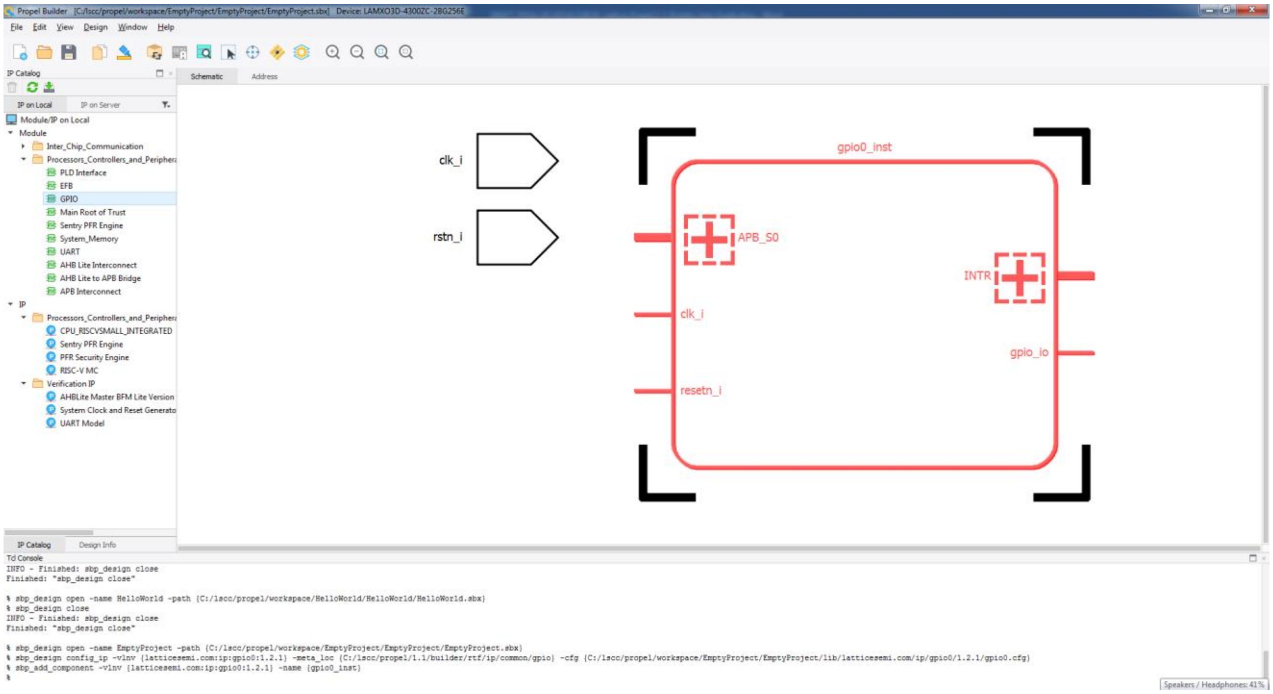


Figure 2.20. Signal List of Modules

- To 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 2.21).



```
INFO - Finished: sbp_design close
Finished: "sbp_design close"

% sbp_design open -name HelloWorld -path {C:/lscc/propel/workspace/HelloWorld/HelloWorld/HelloWorld.sbx}
% sbp_design close
INFO - Finished: sbp_design close
Finished: "sbp_design close"

% sbp_design open -name EmptyProject -path {C:/lscc/propel/workspace/EmptyProject/EmptyProject/EmptyProject.sbx}
% sbp_design config_ip -vlnv {latticesemi.com:ip:gpio0:1.2.1} -meta_loc {C:/lscc/propel/1.1/builder/rtf/ip/common/gpio} -cfg {C:/lscc/propel/workspace/EmptyProject/EmptyProject/lib/latticesemi.com/ip/gpio0/1.2.1/gpio0.cfg}
% sbp_add_component -vlnv {latticesemi.com:ip:gpio0:1.2.1} -name {gpio0_inst}
%
```

Figure 2.21. Select Object

  - Ctrl-click or Shift-click in the Schematic view to select more than one objects.
  - Click the **Area_select** button . Click and drag to draw a selection rectangle around the modules and ports in the Schematic view. Click the button 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 of 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.

- To re-arrange the schematic:

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

  a. 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).

  b. Click on the selected items and drag it to the desired location.

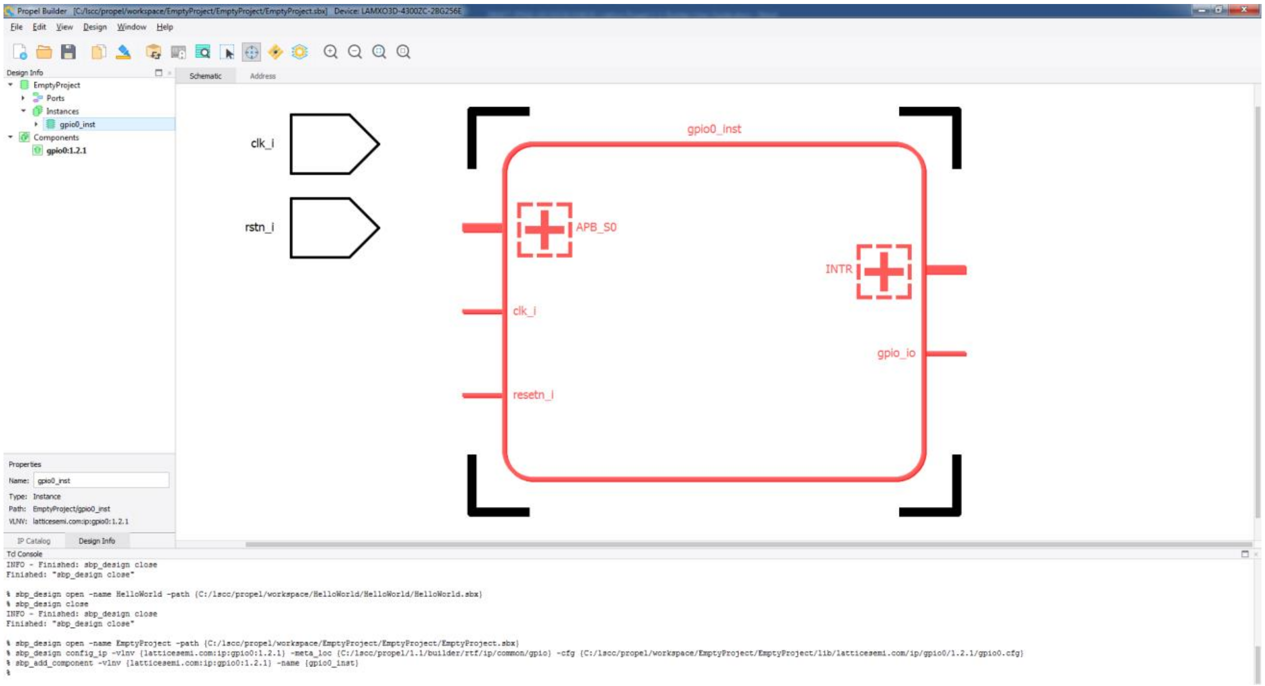
  c. Release the mouse button.

  Note: The selected objects can be moved to the specified location or as near as the rules allow. Other objects in the schematic can also be moved to accommodate the selected objects' new location.

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

  a. Click the Locate Object button so the background turns to dark gray.

  b. Select the object in the List view of Design Info. The selected object is in the center of the Schematic View (Figure 2.22).



Properties

Name: gpio0_inst

Type: Instance

Path: EmptyProject/gpio0_inst

VLNV: latticesemi.com:ip:gpio0:1.2.1

```
INFO - Finished: sbp_design close
Finished: "sbp_design close"

% sbp_design open -name HelloWorld -path {C:/lscc/propel/workspace/HelloWorld/HelloWorld/HelloWorld.sbx}
% sbp_design close
INFO - Finished: sbp_design close
Finished: "sbp_design close"

% sbp_design open -name EmptyProject -path {C:/lscc/propel/workspace/EmptyProject/EmptyProject/EmptyProject.sbx}
% sbp_design config_ip -vlnv {latticesemi.com:ip:gpio0:1.2.1} -meta_loc {C:/lscc/propel/1.1/builder/rtf/ip/common/gpio} -cfg {C:/lscc/propel/workspace/EmptyProject/EmptyProject/lib/latticesemi.com/ip/gpio0/1.2.1/gpio0.cfg}
% sbp_add_component -vlnv {latticesemi.com:ip:gpio0:1.2.1} -name {gpio0_inst}
%
```

Figure 2.22. Locate Objects

- To automatically simplify the layout:
  - Right-click anywhere in the Schematic view and choose **Relayout**.
- To create duplicate of 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 2.23).

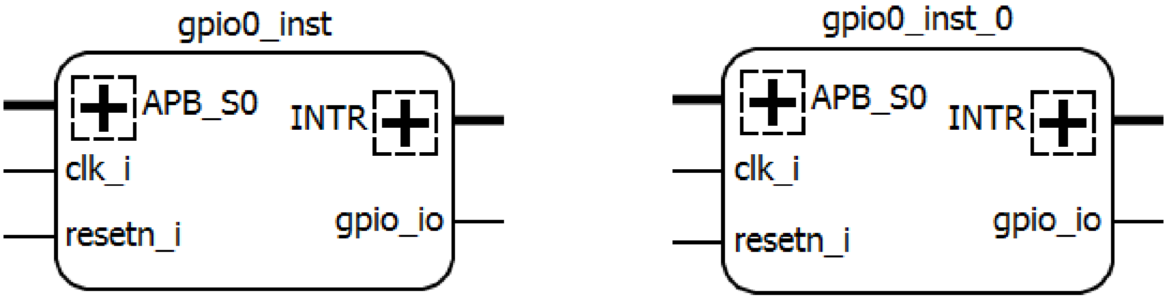
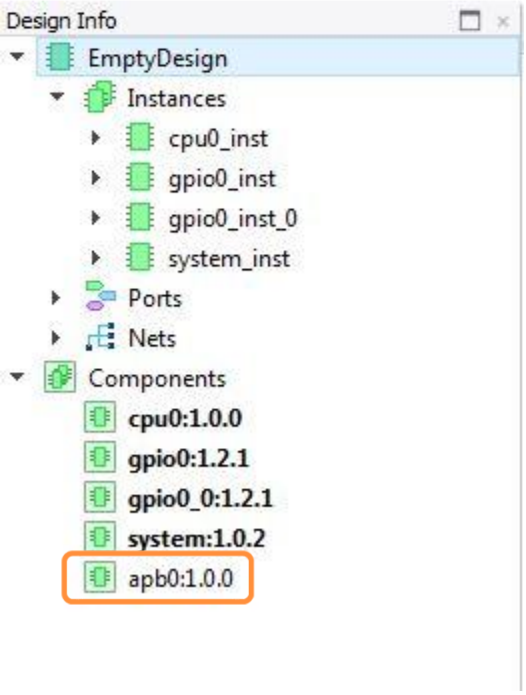


Figure 2.23. Create Duplicate of a Module

- To restore a deleted module:

  a. In the list view of Design Info (Figure 2.24), go to the Components folder. The deleted module can still be found in the List view in plain text. Those that are not deleted are in bold-faced text.

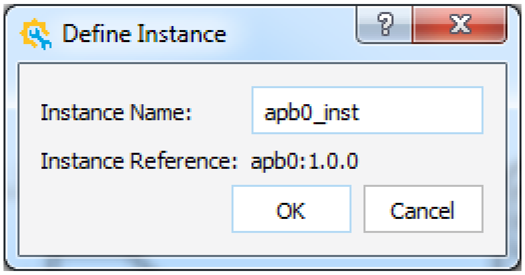


Design Info

- EmptyDesign
  - Instances
    - cpu0_inst
    - gpio0_inst
    - gpio0_inst_0
    - system_inst
  - Ports
  - Nets
- Components
  - **cpu0:1.0.0**
  - **gpio0:1.2.1**
  - **gpio0_0:1.2.1**
  - **system:1.0.2**
  - apb0:1.0.0

Figure 2.24. Design Info View

b. Right-click on the component and choose **Instantiate**. The Define Instance dialog box (Figure 2.25) open.



Define Instance

Instance Name: apb0_inst

Instance Reference: apb0:1.0.0

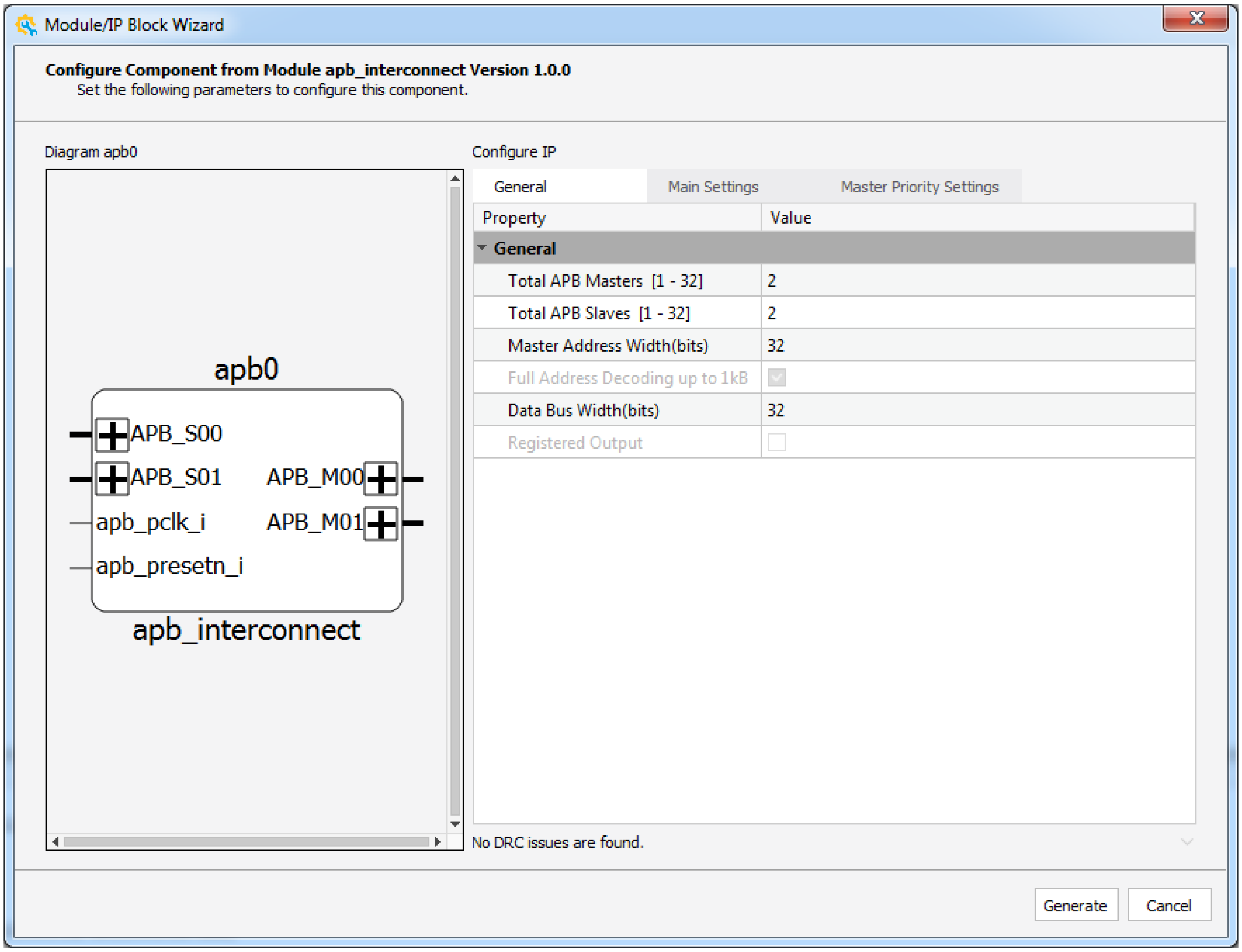
Figure 2.25. Define Instance Dialog Box

c. Enter a name for the new instance.

d. Click **OK**. The module appears in the Schematic view and List view of Design Info, and the component name is bold-faced.

- To reconfigure a module:

a. Double-click the module, or right-click the module. Choose **Reconfig**. The Module/IP Block wizard (Figure 2.26) opens.



Module/IP Block Wizard

Configure Component from Module apb_interconnect Version 1.0.0
Set the following parameters to configure this component.

Diagram apb0

Configure IP

General | Main Settings | Master Priority Settings

Property	Value
General	
Total APB Masters [1 - 32]	2
Total APB Slaves [1 - 32]	2
Master Address Width(bits)	32
Full Address Decoding up to 1kB	☑
Data Bus Width(bits)	32
Registered Output	☐

No DRC issues are found.

Figure 2.26. Module/IP Block Wizard – Configure Component

b. 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.

Note: For SoC design, if System Memory module instance update the init file, the system memory module instance should be re-configured.

- To resize module blocks:

a. Select the desired module block. The block is highlighted in red with black corners (Figure 2.27).

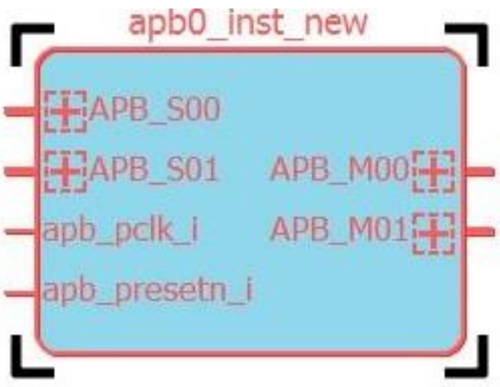


Figure 2.27. Select Module

b. Click and drag one of the corners to change the size and shape of the block.

c. 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.

- 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 on the toolbar.

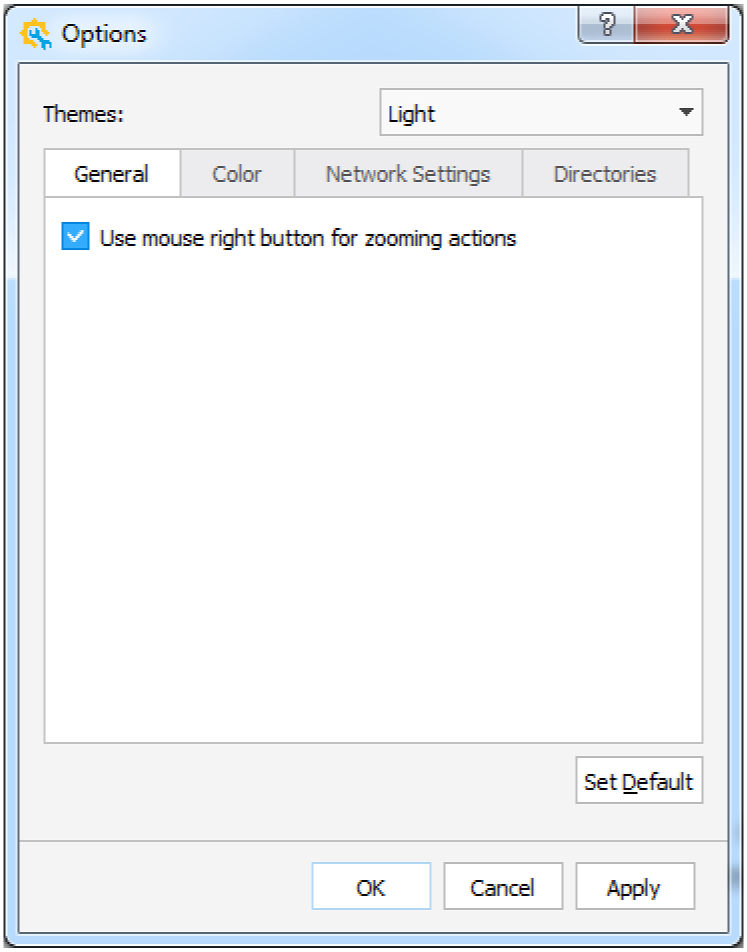
  - Zoom In (Ctrl++) — enlarges the view of the entire layout.
  - Zoom Out (Ctrl+-) — reduces the view of the entire layout.
  - Zoom Fit — reduces or enlarges the entire layout so that it fits inside the window.
  - Zoom To — enlarges the size of one or more selected objects on the layout and fills the 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** button is not selected. If you need to use Area_select and zoom by dragging frequently, choose **Design** > **Options** from Propel Builder menu bar. The Options Dialog opens (Figure 2.28). Select **Use mouse right button for zooming actions** and click **OK**. Then you can select instance using the left mouse button, zoom in or zoom out using the right mouse button, and the **Area_select** button disappears in the Builder tool bar.



Themes: Light

☑ Use mouse right button for zooming actions

Figure 2.28. Options Dialog

- To move the 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.
- To 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 2.29).

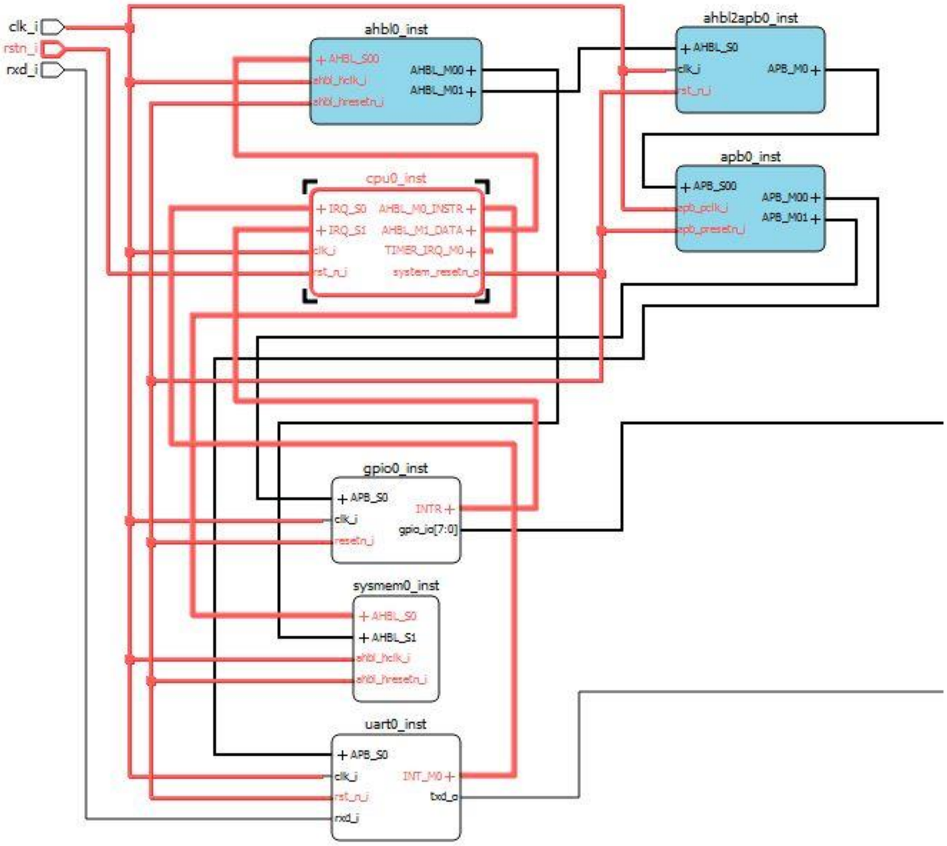


Figure 2.29. Show Connectivity of the Module

- To highlight an object: select an object and click **Highlight** , or, right-click the object and choose **Highlight** . The object is highlighted in blue (Figure 2.30). Click **Highlight** again. You can remove highlighting.

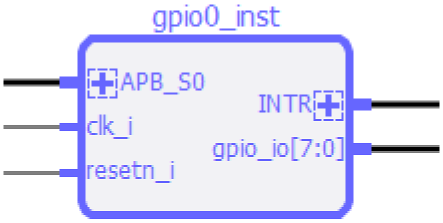
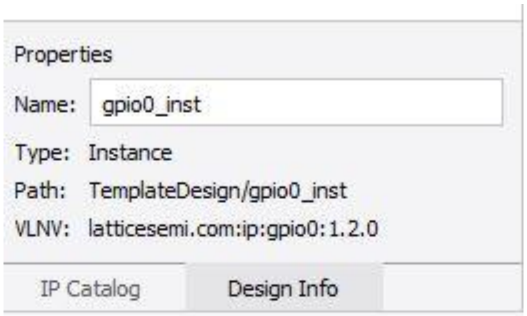


Figure 2.30. Highlight an Object

- To change the name of an object:
  - Select the object from the List view of Design Info. Information of the selected object is shown in the Properties area (Figure 2.31).



Properties

Name: gpio0_inst

Type: Instance

Path: TemplateDesign/gpio0_inst

VLNV: latticesemi.com:ip:gpio0:1.2.0

Figure 2.31. Object Properties

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

- To print a schematic:

a. Choose **File** > **Print Preview**. The Print Preview window (Figure 2.32) opens.

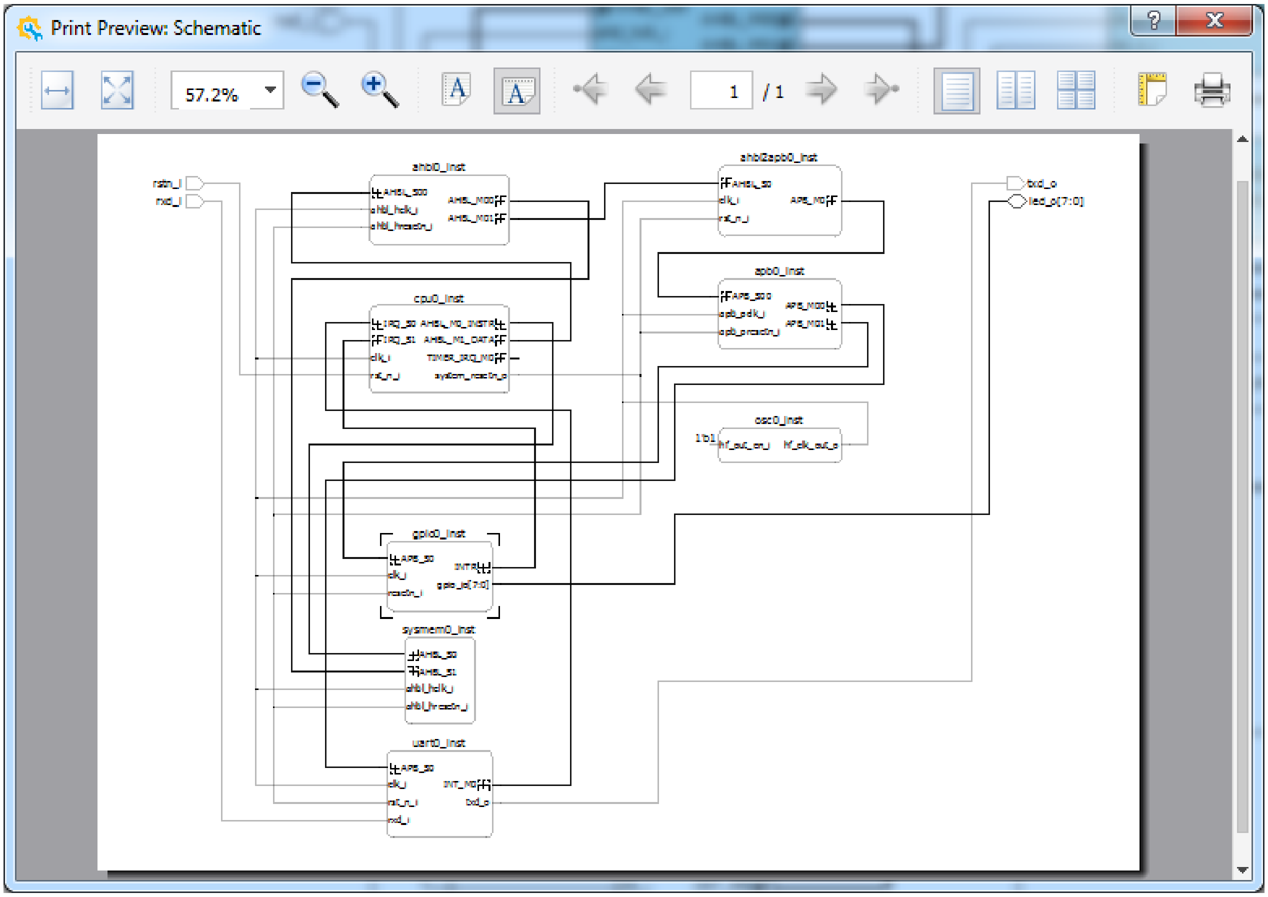


Figure 2.32. Print Preview

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

c. Click the **Page Setup** button and adjust the paper size and margins, if necessary.

d. Click the **Print** button.

e. 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.

- To connect modules by drawing:

  a. 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 check-mark when you hover over it (Figure 2.33). An allowed net becomes bold when you hover over it (Figure 2.34).

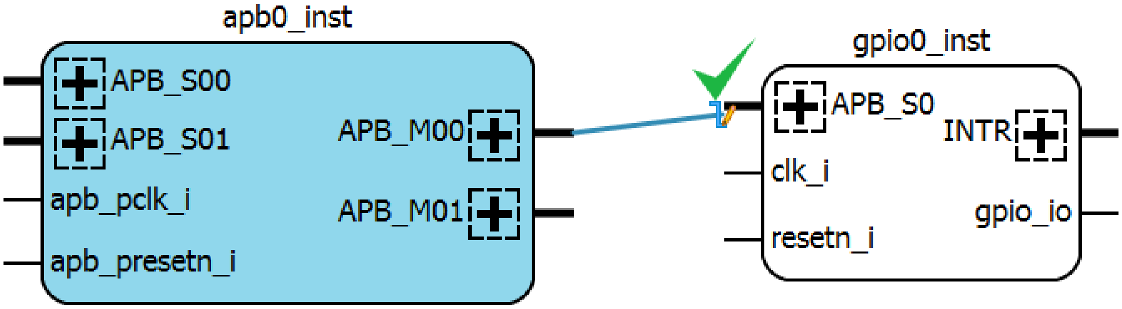


Figure 2.33. Draw a Pin or Port

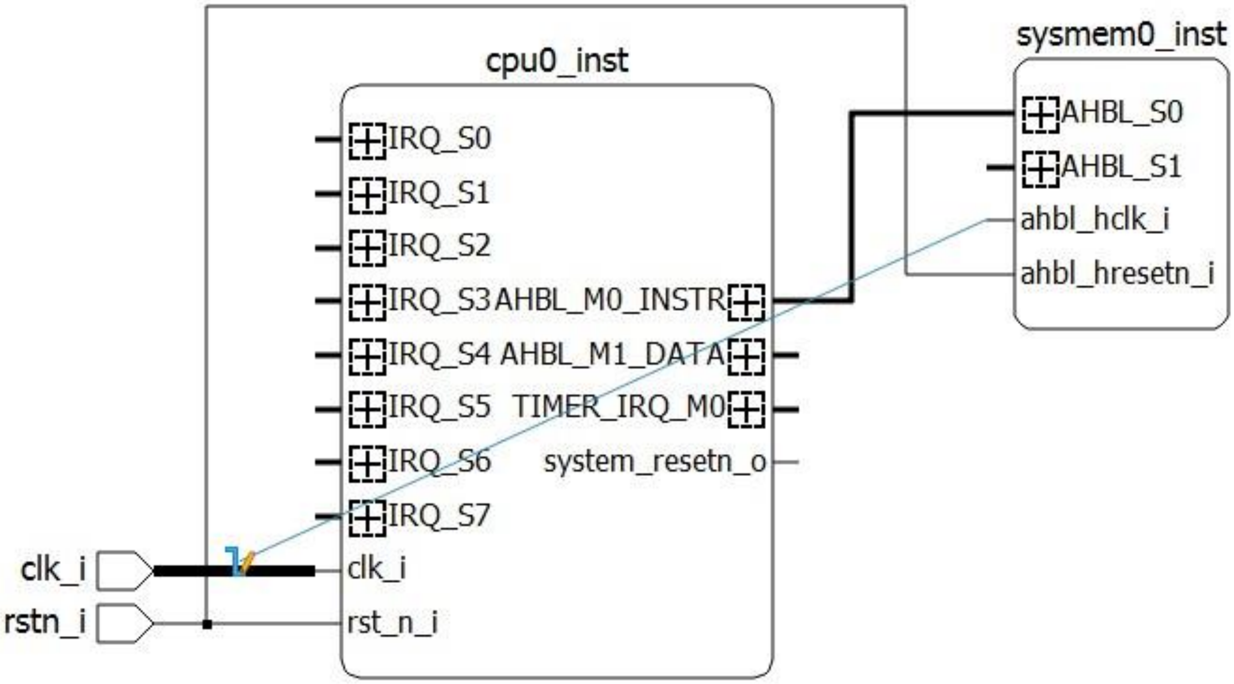


Figure 2.34. Draw Nets

  b. 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.

c. User can connect multiple ports once, when more than one port is selected (Figure 2.35), click **connect** in the right click menu to implement it.

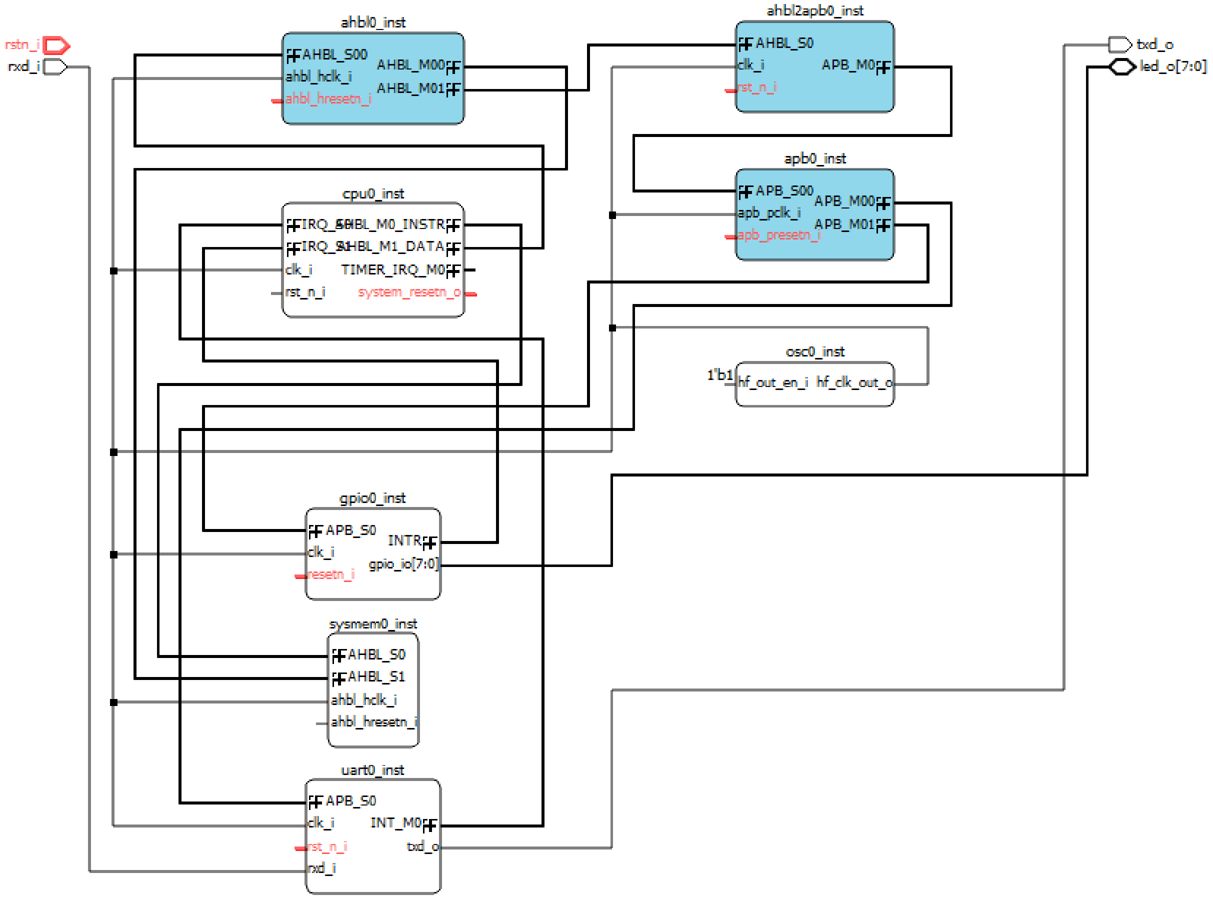


Figure 2.35. Select More Than One Ports

d. When the net is complete, right-click to leave drawing mode.

- To connect modules by selecting points:

a. Select the pins, ports, and nets that you want to connect.

b. Right-click one of the selected objects. Choose **Connect** .

- To assign a constant value to an input pin:

a. Select the desired pin and right-click the pin. Choose **Assign Constant Value** (Figure 2.36). A small dialog box appears (Figure 2.37).

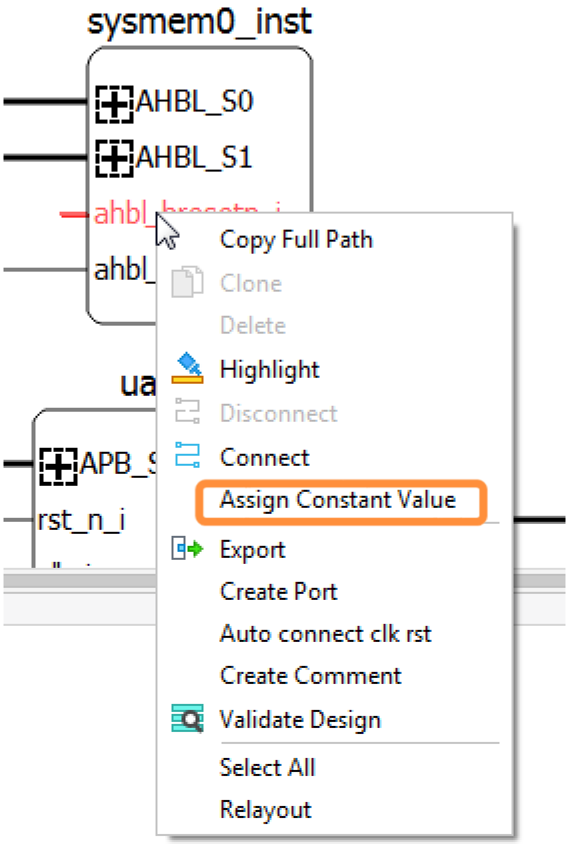


Figure 2.36.Show Action Menu of Right-clicking an Input Pin

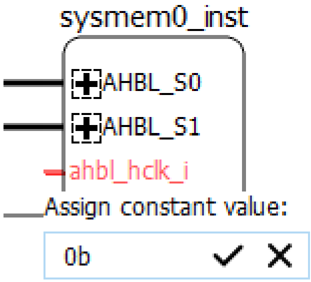


Figure 2.37. Dialog Box of Assigning Constant Value to an Input Pin

b. 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.

- Usually, disconnecting modules just 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

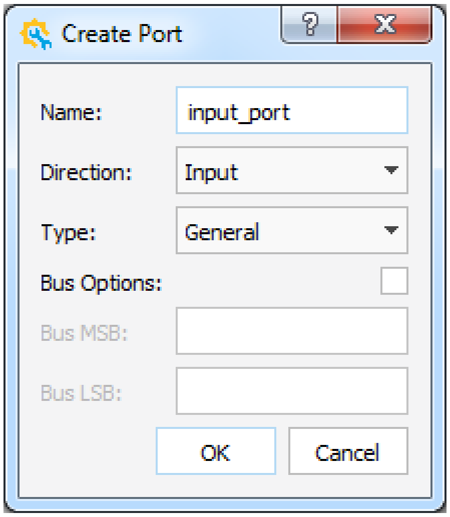
With multiple modules in a Propel Builder project, top-level ports need to be created for a complete Propel Builder module. You can create a top-level port either manually or automatically.

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 2.38).



Create Port

Name: input_port

Direction: Input

Type: General

Bus Options: □

Bus MSB:

Bus LSB:

Figure 2.38. 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 2.39 shows an input port, port name of which is on the left. Figure 2.40 shows an output port and an inout port, port names of which are on the right.

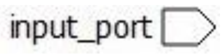


Figure 2.39. Input Port

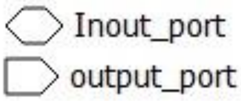


Figure 2.40. Output Port and Inout Port

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

- To create ports automatically:

a. Select the pins that are to be connected to the top-level ports. All the pins in a module can be selected by clicking its block. Any pins that are already connected to a net or a constant value are skipped.

b. Right-click on one of the selected pins, interfaces or modules, and choose **Export**. The selected pins are extended by lines to new top-level port symbols. The names of the ports and nets are added to the List view. Zoom out or scroll the image in the Schematic view to see the new ports.

c. Port or net names can be changed, if needed.

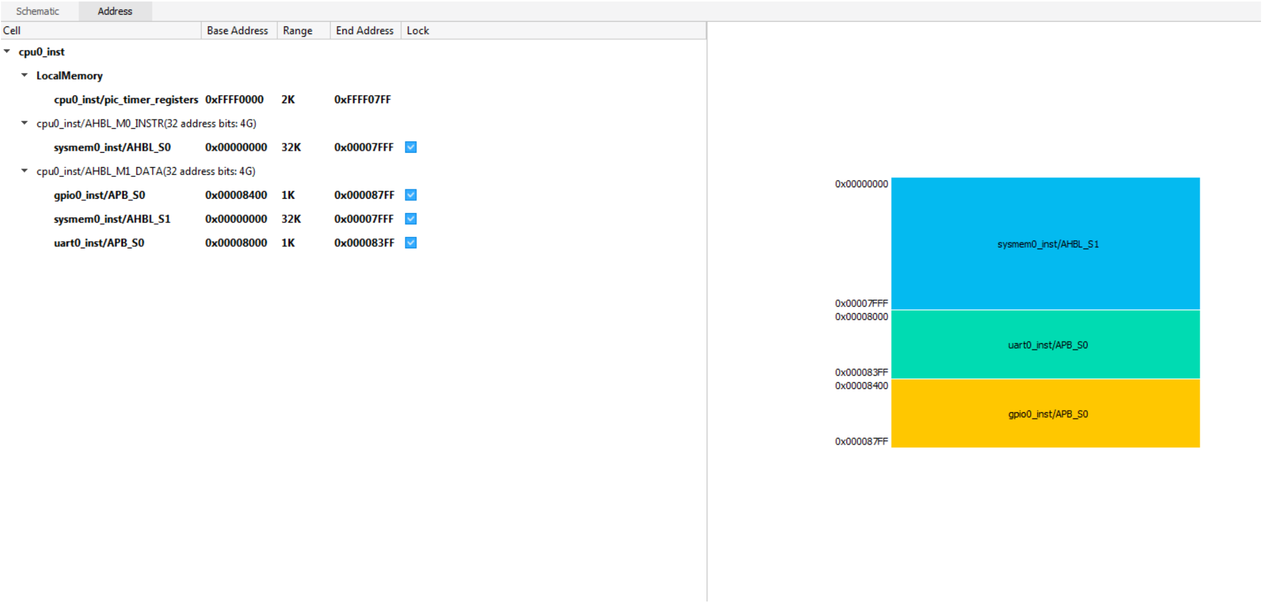
2.2.8. Adjusting Address Spaces

The Address view shows the base address, size of the address segment, and the end address for each leaf memory-mapped slave connection in the Propel Builder project. Propel Builder can automatically assign address values, while the base address value can be changed manually. The 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 2.41) shows.



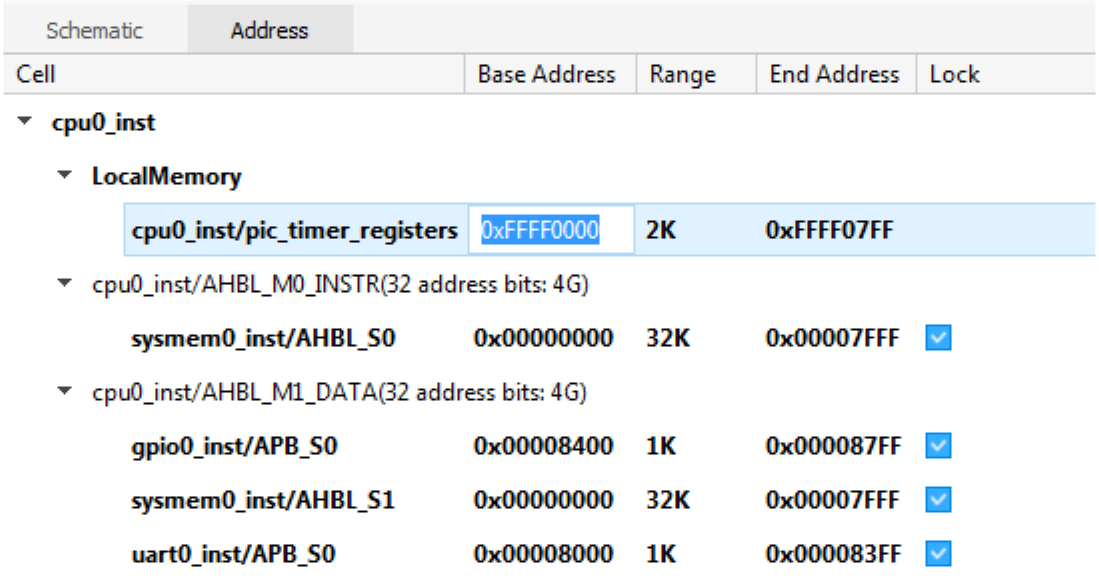
Cell	Base Address	Range	End Address	Lock
cpu0_inst				
LocalMemory				
cpu0_inst/pic_timer_registers	0xFFFF0000	2K	0xFFFF07FF	
cpu0_inst/AHBL_M0_INSTR(32 address bits: 4G)				
sysmem0_inst/AHBL_S0	0x00000000	32K	0x00007FFF	☑
cpu0_inst/AHBL_M1_DATA(32 address bits: 4G)				
gpio0_inst/APB_S0	0x00008400	1K	0x000087FF	☑
sysmem0_inst/AHBL_S1	0x00000000	32K	0x00007FFF	☑
uart0_inst/APB_S0	0x00008000	1K	0x000083FF	☑

Figure 2.41. Address View

2. (Optional) Set or clear the Lock options as desired in Address view.

3. In the Propel Builder toolbar, click **Auto Assign** . Default address values appear in Address view.

4. (Optional) Double-click the base address in Address view (Figure 2.42). Type the new value and press **Enter**. Values must align with 1K boundaries, such as 0x00000400, 0x00000800, or 0x00000C00. The end address value changes based on the new value. The Lock option is selected automatically at this time.



Cell	Base Address	Range	End Address	Lock
cpu0_inst				
LocalMemory				
cpu0_inst/pic_timer_registers	0xFFFF0000	2K	0xFFFF07FF	
cpu0_inst/AHBL_M0_INSTR(32 address bits: 4G)				
sysmem0_inst/AHBL_S0	0x00000000	32K	0x00007FFF	☑
cpu0_inst/AHBL_M1_DATA(32 address bits: 4G)				
gpio0_inst/APB_S0	0x00008400	1K	0x000087FF	☑
sysmem0_inst/AHBL_S1	0x00000000	32K	0x00007FFF	☑
uart0_inst/APB_S0	0x00008000	1K	0x000083FF	☑

Figure 2.42. 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. This checks whether or not there is any illegal connection or overlapping address space.

To perform the design rule check:

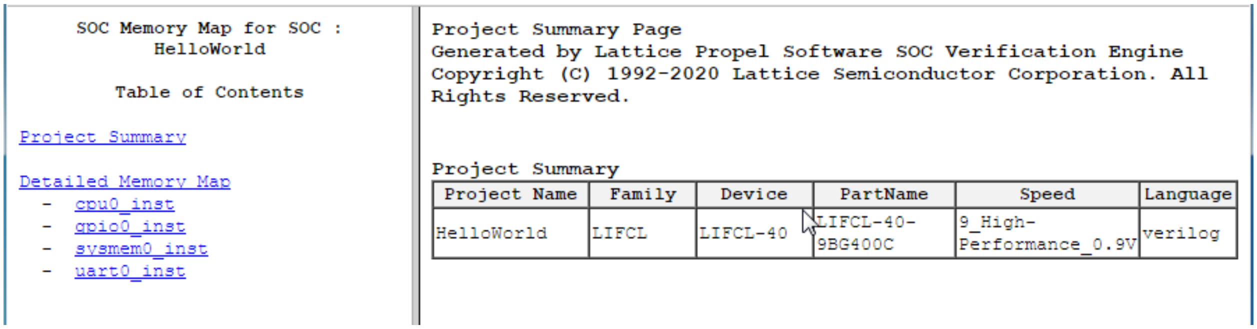
1. From the toolbar of the Propel Builder main window, click the **Validate Design** icon . The DRC results appear in the Tcl Console.

2.2.10. Generating the RTL file

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

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

- 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 2.43).



SOC Memory Map for SOC :
HelloWorld

Table of Contents

Project Summary

Detailed Memory Map
- cpu0_inst
- gpio0_inst
- sysmem0_inst
- uart0_inst

Project Summary Page
Generated by Lattice Propel Software SOC Verification Engine
Copyright (C) 1992-2020 Lattice Semiconductor Corporation. All Rights Reserved.

Project Summary

Project Name	Family	Device	PartName	Speed	Language
HelloWorld	LIFCL	LIFCL-40	LIFCL-40-9BG400C	9_High-Performance_0.9V	verilog

Figure 2.43. Memory Report

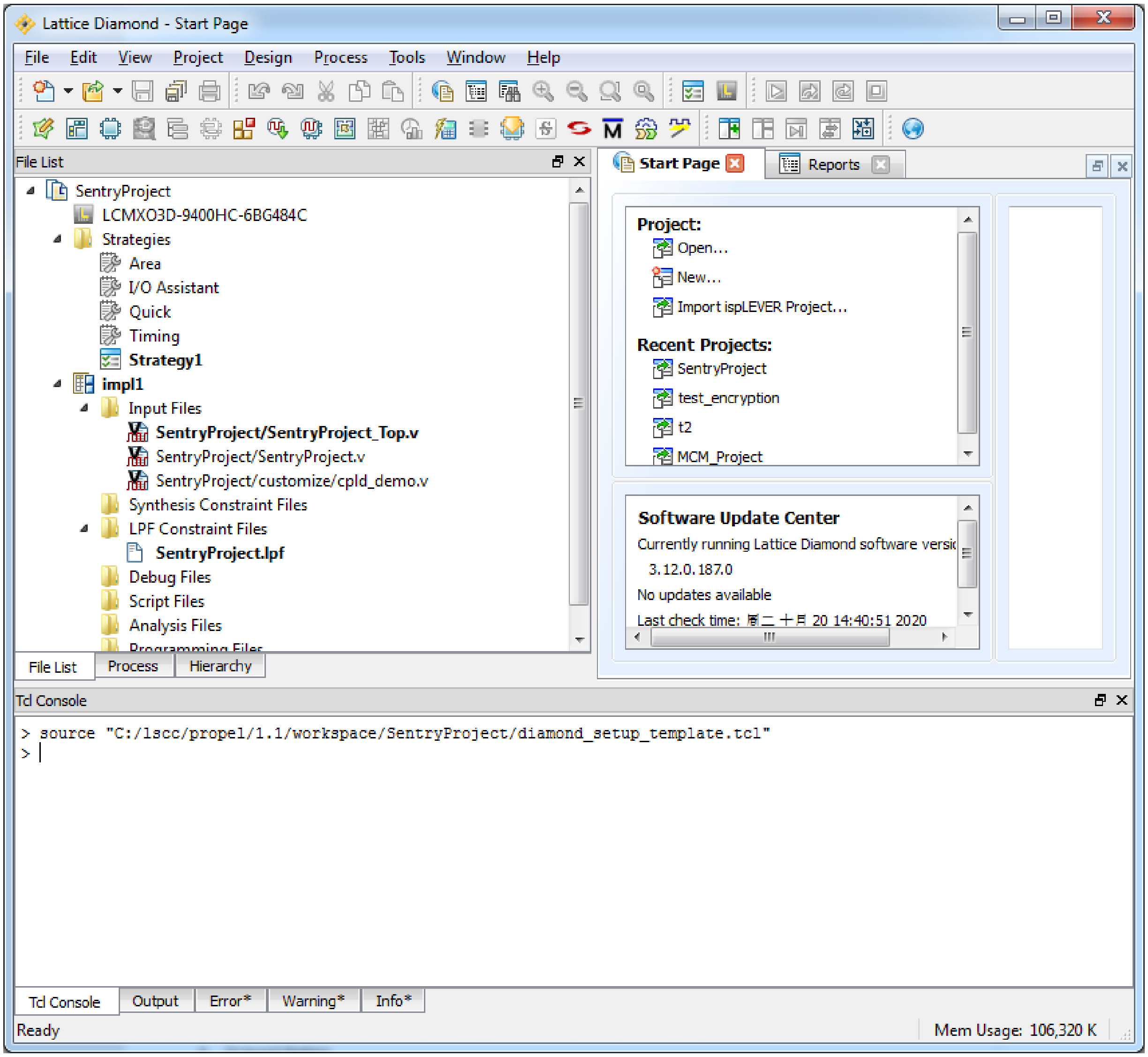
2.2.11. Opening Project in Lattice Diamond® or Lattice Radiant™

In a complete SoC project, a Diamond or Radiant project can be created including a Propel Builder design, and after that the SoC project can be opened in Diamond or Radiant software. According to device in the SoC project, Propel Builder can launch Diamond or Radiant software accordingly. If the device is LCMXO3D or LFMNX, the **Diamond** icon ◆ is shown in the Builder GUI Toolbar. If the device is LIFCL or LFD2NX, the **Radiant** icon R is shown in the Builder GUI Toolbar. For a template SoC project, you can launch **Diamond** or **Radiant** directly after template SoC project is created.

- To launch Diamond:

  a. Click Diamond icon ◆ from the Propel Builder toolbar.

  b. Lattice Diamond is launched with a Diamond project generated for SoC at background (Figure 2.44).



SentryProject
LCMXO3D-9400HC-6BG484C
Strategies
Area
I/O Assistant
Quick
Timing
Strategy1
impl1
Input Files
SentryProject/SentryProject_Top.v
SentryProject/SentryProject.v
SentryProject/customize/cpld_demo.v
Synthesis Constraint Files
LPF Constraint Files
SentryProject.lpf
Debug Files
Script Files
Analysis Files
Programming Files

Project:
Open…
New…
Import ispLEVER Project…

Recent Projects:
SentryProject
test_encryption
t2
MCM_Project

Software Update Center
Currently running Lattice Diamond software versi
3.12.0.187.0
No updates available
Last check time: 周二 十月 20 14:40:51 2020

```
> source "C:/lscc/propel/1.1/workspace/SentryProject/diamond_setup_template.tcl"
> 
```

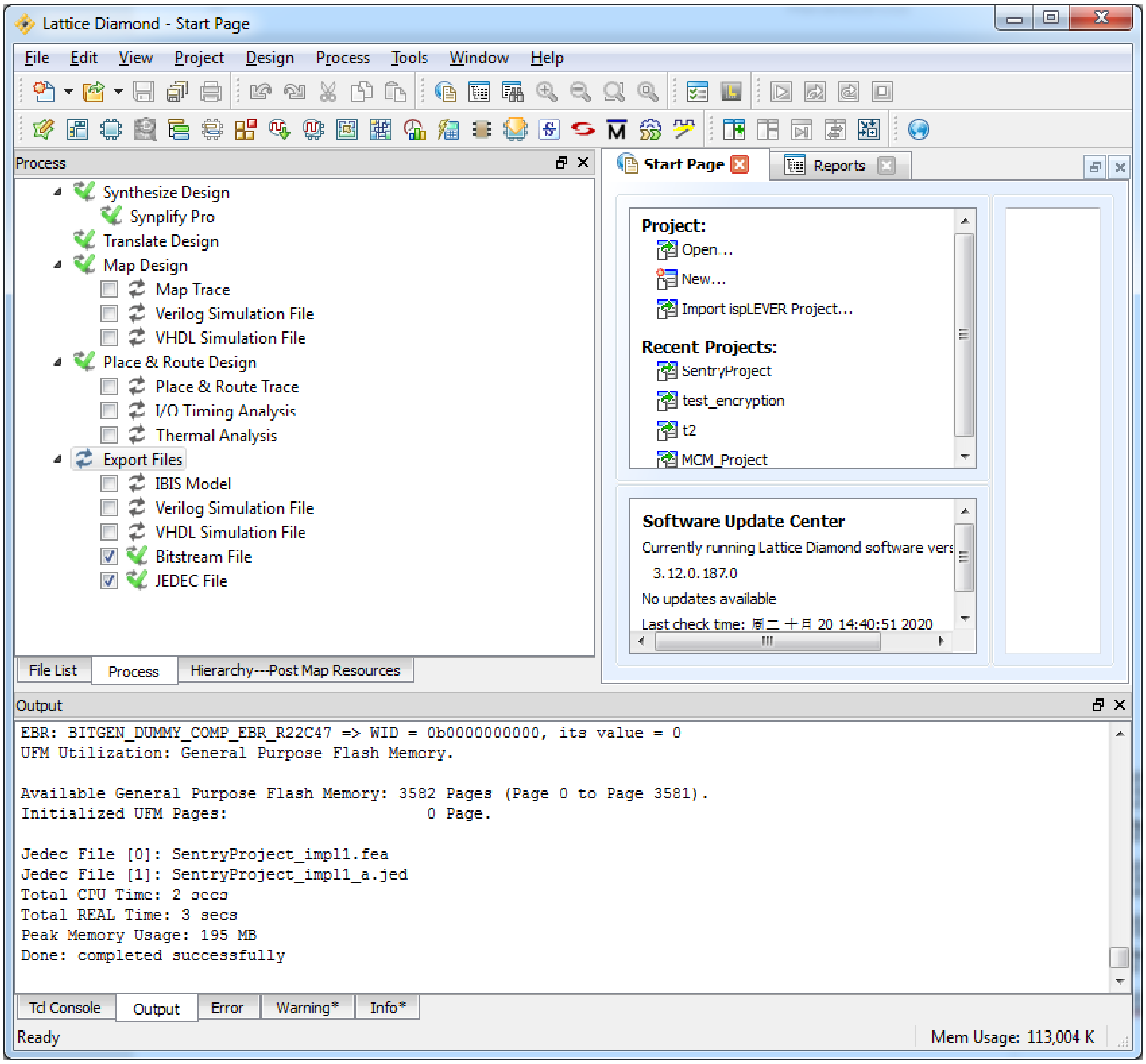
Figure 2.44. Diamond Project

c. (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.

d. (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.

e. Switch to **Process** view of the Diamond project (Figure 2.45). Make sure at least one programming file (IBIS Model, Verilog Simulation File, VHDL Simulation File, Bitstream File, or JEDEC file) is selected in the **Export Files** section. Available programming files can be different upon specific device included.



Currently running Lattice Diamond software vers
3.12.0.187.0
No updates available
Last check time: 周二 十月 20 14:40:51 2020

```
EBR: BITGEN_DUMMY_COMP_EBR_R22C47 => WID = 0b0000000000, its value = 0
UFM Utilization: General Purpose Flash Memory.

Available General Purpose Flash Memory: 3582 Pages (Page 0 to Page 3581).
Initialized UFM Pages:                     0 Page.

Jedec File [0]: SentryProject_impl1.fea
Jedec File [1]: SentryProject_impl1_a.jed
Total CPU Time: 2 secs
Total REAL Time: 3 secs
Peak Memory Usage: 195 MB
Done: completed successfully
```

Figure 2.45. Generate Programming Files

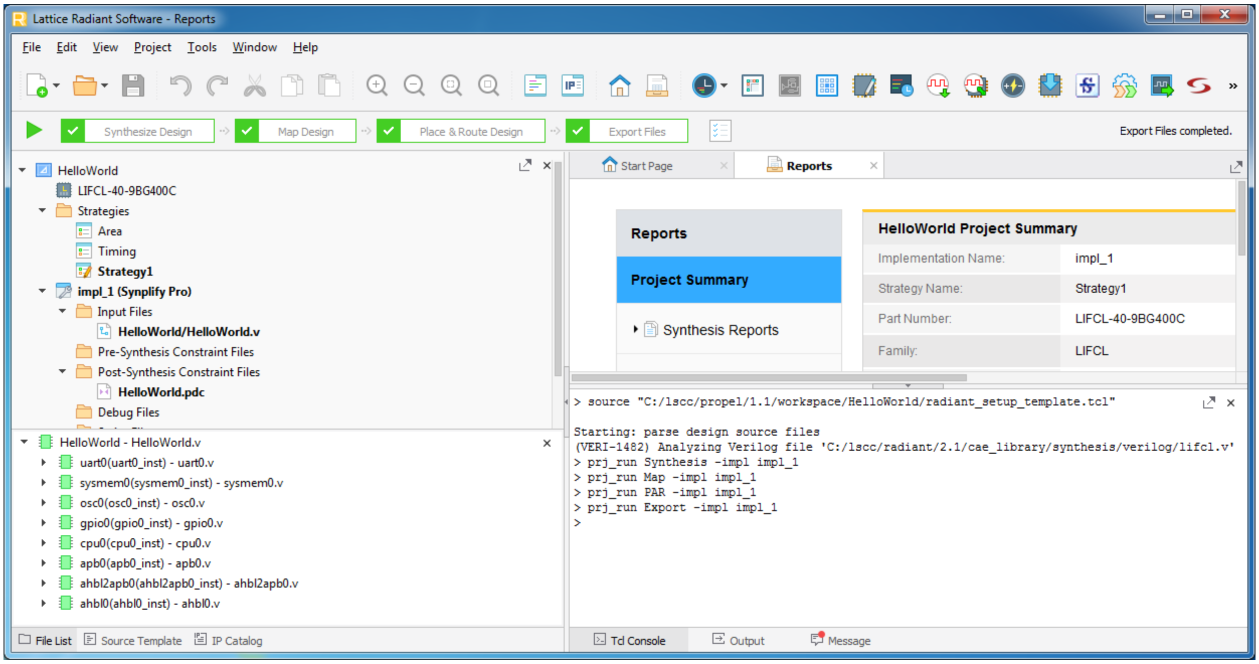
f. Right-click **Export Files**, and choose **Run** . Programming file is generated. The programming file can be used in the Diamond Programmer.

Note: Refer to the Diamond Help for more details of the Diamond project.

- To launch Radiant

a. Click the Radiant icon from the Propel Builder toolbar.

b. Lattice Radiant is launched with a Radiant project generated for SoC at background (Figure 2.46).



HelloWorld Project Summary

Implementation Name:	impl_1
Strategy Name:	Strategy1
Part Number:	LIFCL-40-9BG400C
Family:	LIFCL

```
> source "C:/lscc/propel/1.1/workspace/HelloWorld/radiant_setup_template.tcl"

Starting: parse design source files
(VERI-1482) Analyzing Verilog file 'C:/lscc/radiant/2.1/cae_library/synthesis/verilog/lifcl.v'
> prj_run Synthesis -impl impl_1
> prj_run Map -impl impl_1
> prj_run PAR -impl impl_1
> prj_run Export -impl impl_1
>
```

Figure 2.46. Radiant Project

c. (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.

d. (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.

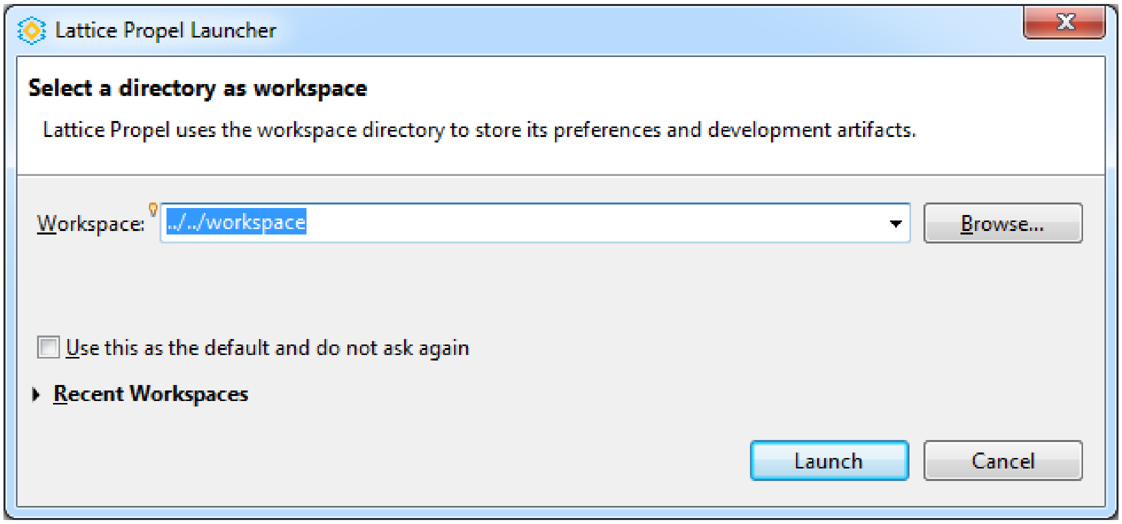
e. Click ▶ **Run all**. The Programming file is generated. It can be used in the Radiant Programmer.

Note: Refer to the Radiant Help for more details of the Radiant project.

2.2.12. Launching SDK

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

1. Click **Run Propel** icon from the Propel Builder GUI Toolbar. Lattice Propel Launcher (Figure 2.47) opens.



Lattice Propel Launcher

Select a directory as workspace

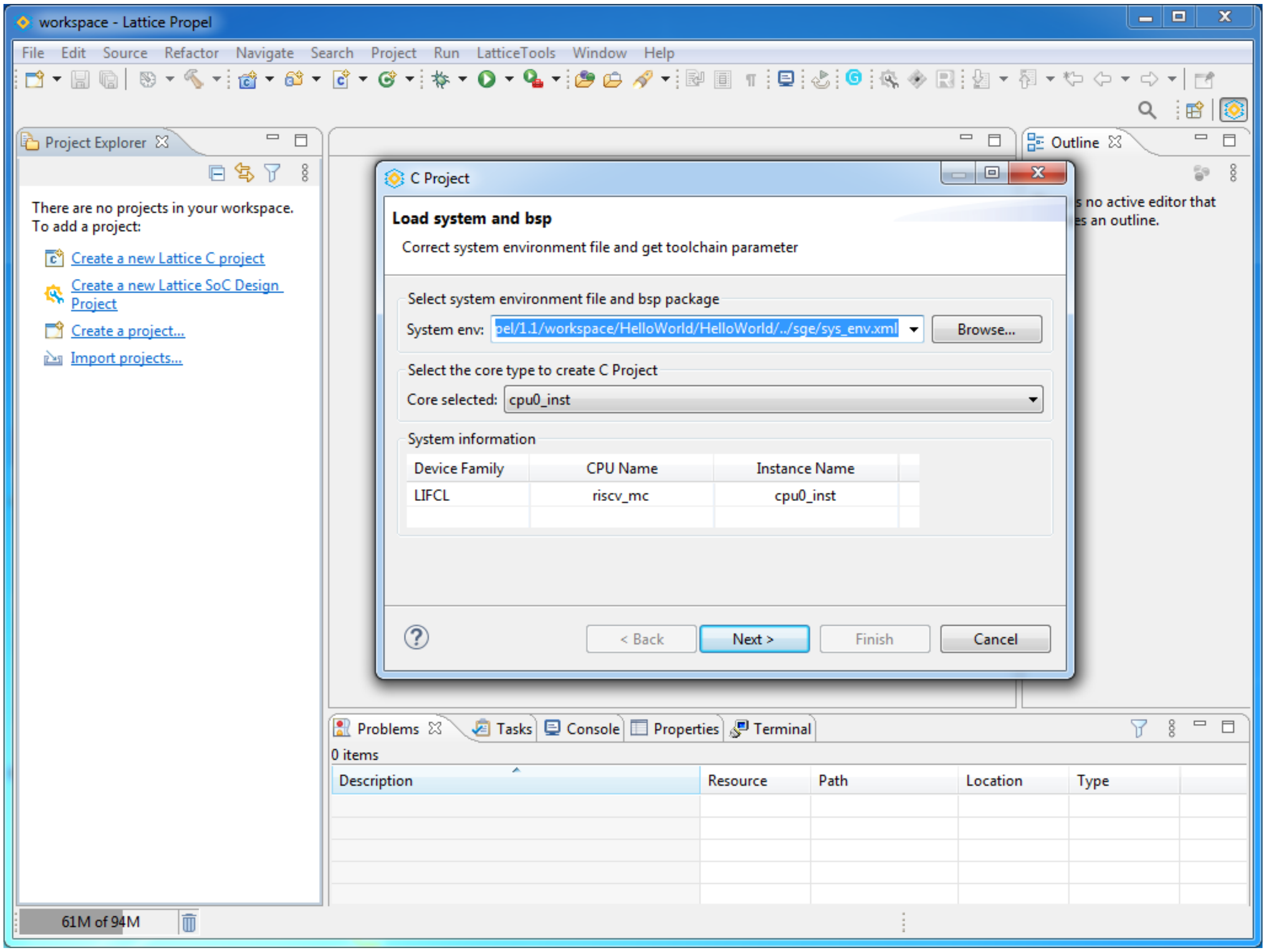
Lattice Propel uses the workspace directory to store its preferences and development artifacts.

Workspace: ../../workspace

☐ Use this as the default and do not ask again

Figure 2.47. Lattice Propel Launcher Wizard

2. Click **Launch**. Propel SDK GUI opens and the C Project wizard (Figure 2.48) pops up for loading the system and the board support package (BSP) to create a C project.



workspace - Lattice Propel

There are no projects in your workspace.
To add a project:

- Create a new Lattice C project
- Create a new Lattice SoC Design Project
- Create a project...
- Import projects...

C Project

Load system and bsp

Correct system environment file and get toolchain parameter

Select system environment file and bsp package

System env: pel/1.1/workspace/HelloWorld/HelloWorld/../sge/sys_env.xml

Select the core type to create C Project

Core selected: cpu0_inst

System information

Device Family	CPU Name	Instance Name
LIFCL	riscv_mc	cpu0_inst

s no active editor that
es an outline.

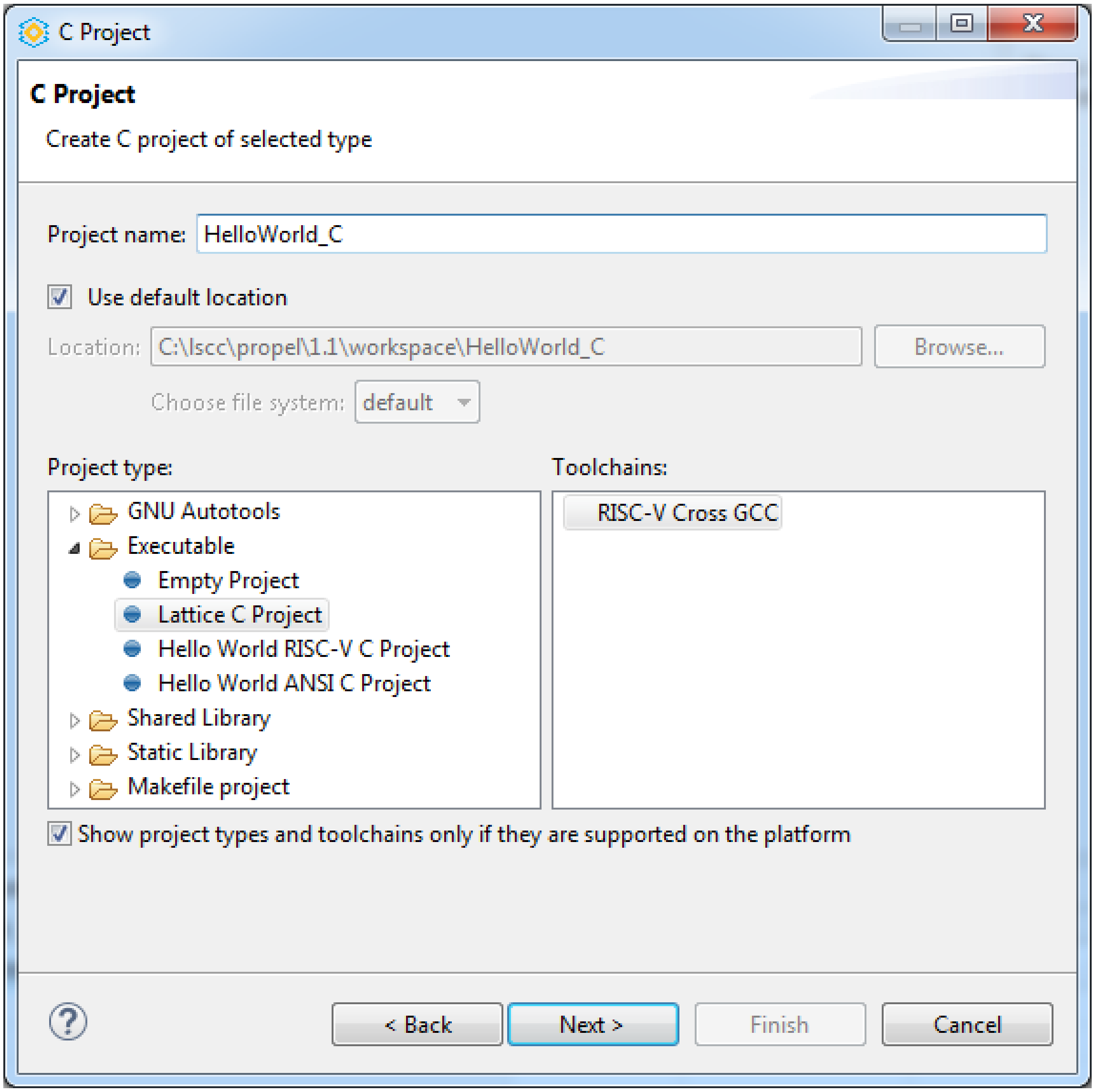
0 items

Description	Resource	Path	Location	Type

61M of 94M

Figure 2.48. Propel SDK GUI and C Project Wizard

3. Click **Next**. The C project dialog (Figure 2.49) opens.



C Project

C Project

Create C project of selected type

Project name: HelloWorld_C

☑ Use default location

Location: C:\lscc\propel\1.1\workspace\HelloWorld_C

Choose file system: default

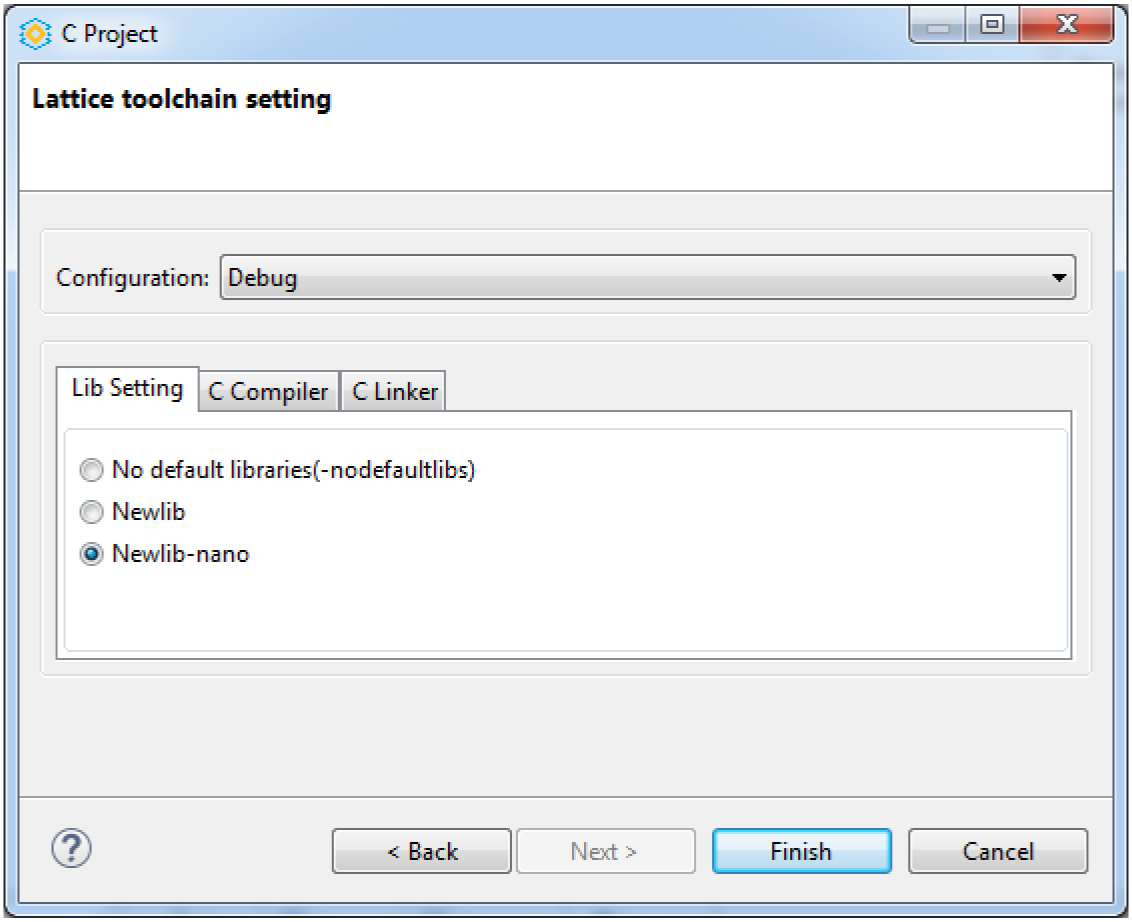
Project type:

Toolchains: RISC-V Cross GCC

☑ Show project types and toolchains only if they are supported on the platform

Figure 2.49. Create C project

4. Enter a project name in the “**Project Name”** field, such as HelloWorld_C. Click **Next**. The Lattice toolchain setting dialog opens
(Figure 2.50).



Lattice toolchain setting

Configuration: Debug

○ No default libraries(-nodefaultlibs)

○ Newlib

● Newlib-nano

Figure 2.50. Lattice Toolchain Setting Dialog

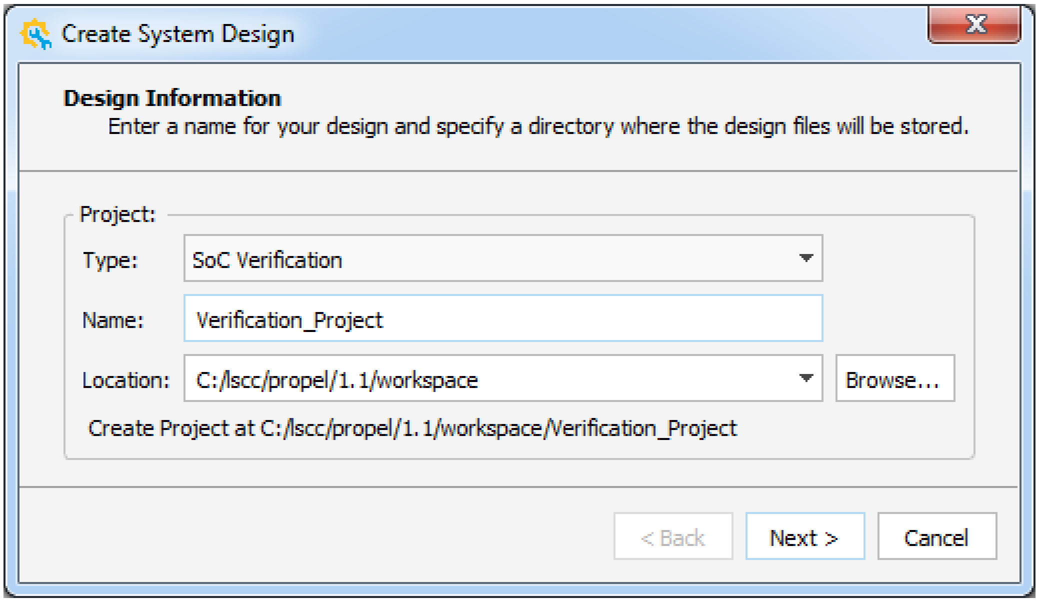
5. Click **Finish**. The C project is created and is displayed using the C/C++ perspective. A perspective is a collection of tool views for a particular purpose. The C/C++ perspective is for creating C/C++ programs.

Note: Refer to the Propel SDK User Guide for more details on how to create a C project and develop the C project in Propel SDK.

2.3. Verification Project Design Flow

2.3.1. Creating a Verification Project

1. Choose **File** > **New Design** from the Lattice Propel Builder Menu Bar. The Create System Design wizard opens (Figure 2.51).



Create System Design

Design Information

Enter a name for your design and specify a directory where the design files will be stored.

Project:

Type: SoC Verification

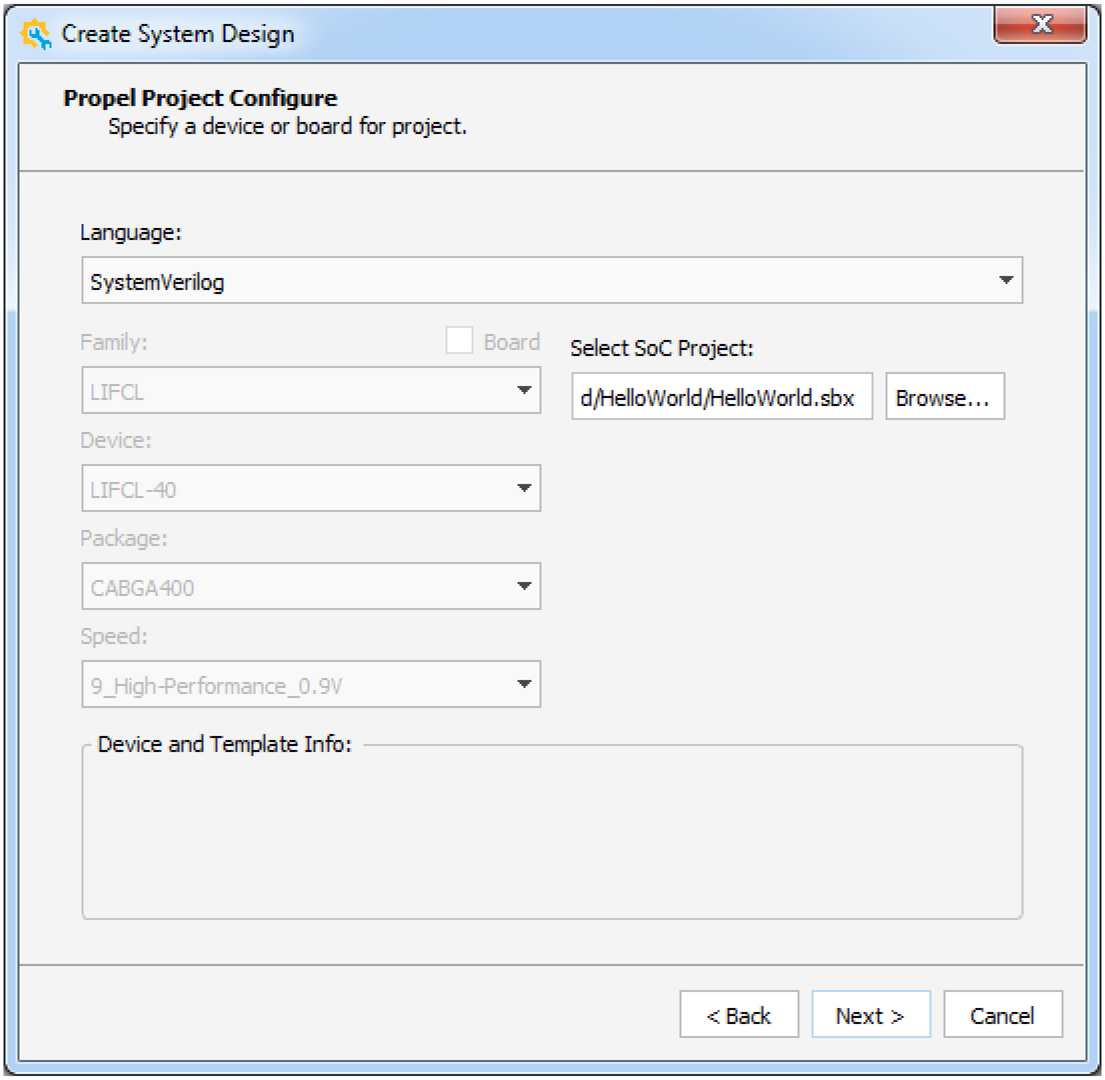
Name: Verification_Project

Location: C:/lscc/propel/1.1/workspace

Create Project at C:/lscc/propel/1.1/workspace/Verification_Project

Figure 2.51. Create System Design – Design Information wizard

2. Choose SoC Verification from the "**Type"** field.
3. Enter a project name in the "**Name"** field, such as Verification_Project.
4. (Optional)Browse to change the project location in the "**Location"** field, if needed.
5. Click **Next**. The Propel Project Configure wizard is shown as Figure 2.52.



Create System Design

Propel Project Configure

Specify a device or board for project.

Language: SystemVerilog

Family: LIFCL

□ Board

Select SoC Project: d/HelloWorld/HelloWorld.sbx

Device: LIFCL-40

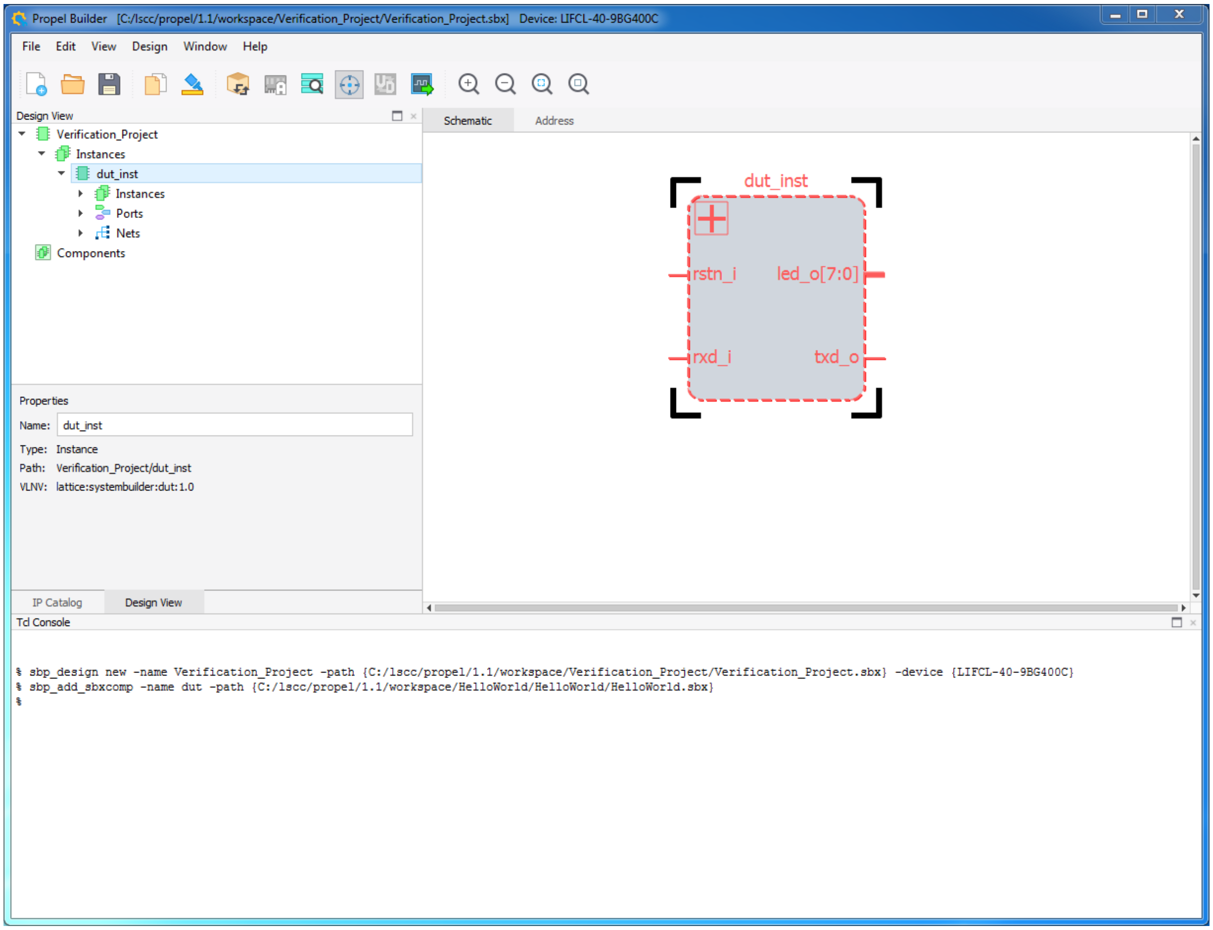
Package: CABGA400

Speed: 9_High-Performance_0.9V

Device and Template Info:

Figure 2.52. Create System Design - Propel Project Configure Wizard

6. (Optional) The default Hardware Description Language (HDL) is displayed in the Language area. Use the drop-down menu to change the default language.
7. Select an existing SBX design by Browsing to choose SBX file such as HelloWorld.
8. Click **Next**. The Project Information wizard opens.
9. Click **Finish**. A dut_inst of SoC project can be seen in the Schematic view (Figure 2.53).



Properties

Name: dut_inst

Type: Instance

Path: Verification_Project/dut_inst

VLNV: lattice:systembuilder:dut:1.0

```
% sbp_design new -name Verification_Project -path {C:/lscc/propel/1.1/workspace/Verification_Project/Verification_Project.sbx} -device {LIFCL-40-9BG400C}
% sbp_add_sbxcomp -name dut -path {C:/lscc/propel/1.1/workspace/HelloWorld/HelloWorld/HelloWorld.sbx}
%
```

Figure 2.53. Verification Project

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

10. Click the plus sign ⊞ of this dut_inst DUT block. You can see the whole SoC Design (Figure 2.54). Click the negative sign ⊟ to close the expanded bus.

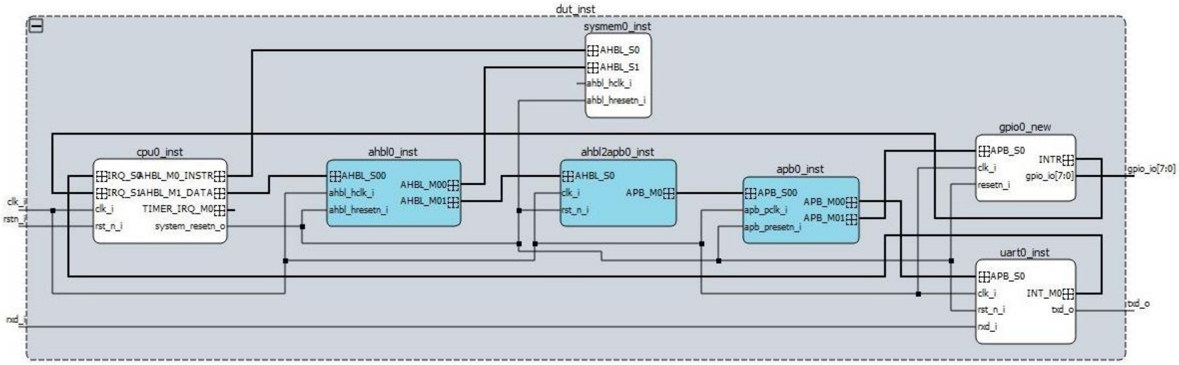


Figure 2.54. Whole SoC Design

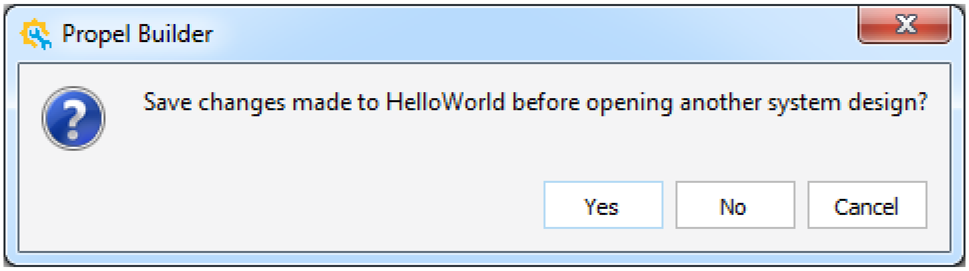
2.3.2. Switching SoC Project to Verification Project

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

1. Create SoC Project refer to Creating Template SoC Project such as HelloWorld project.

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

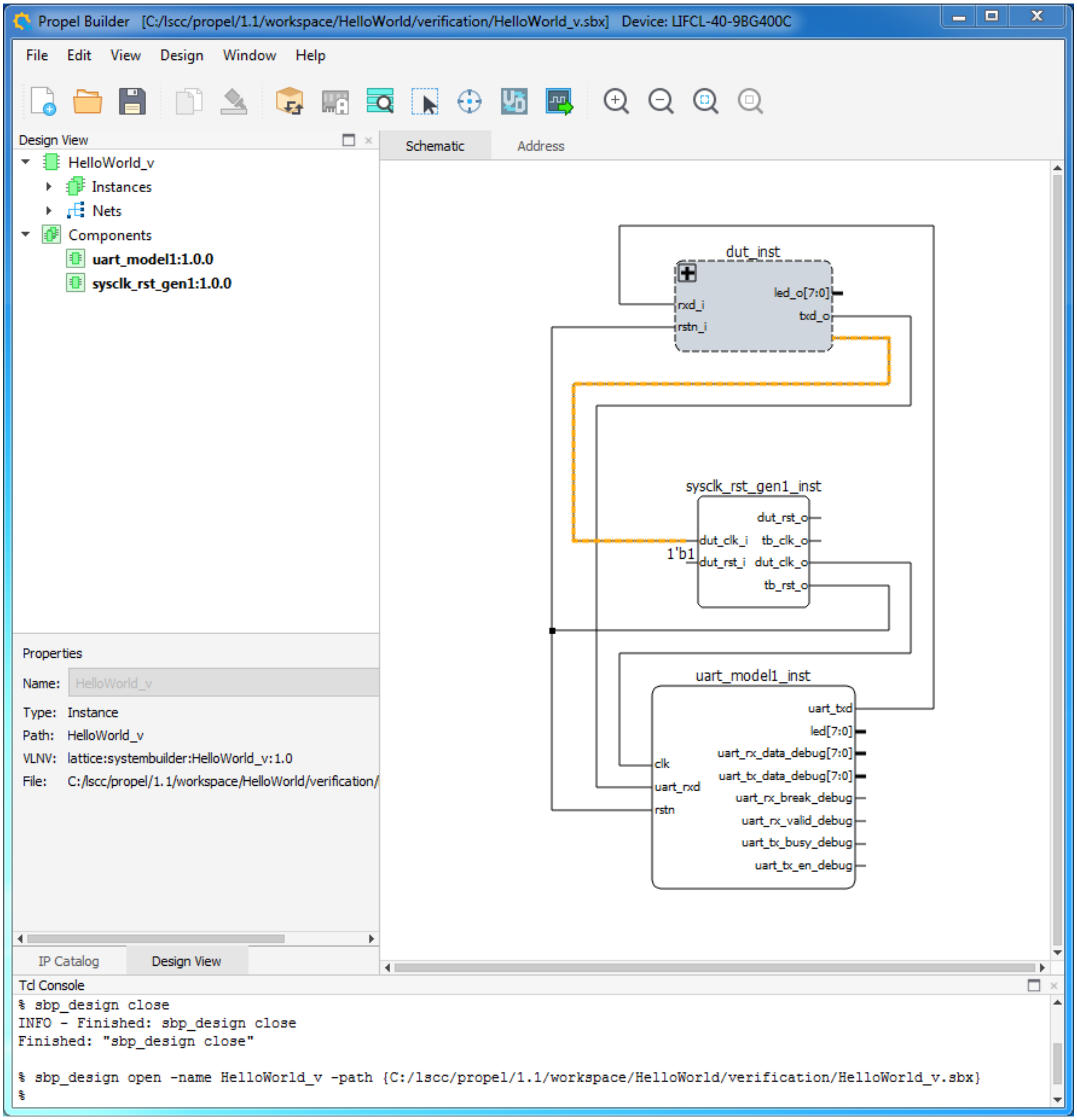
2. Click **Switch Verification and Soc Design** icon from Propel Builder GUI Toolbar. Propel Builder dialog box opens (Figure 2.55), if project is not saved.



Save changes made to HelloWorld before opening another system design?

Figure 2.55. Propel Builder Dialog Box

3. Click **Yes** button at Propel Builder dialog box. The Propel Builder GUI switches to verifying the project (Figure 2.56).



Properties

Name: HelloWorld_v

Type: Instance

Path: HelloWorld_v

VLNV: lattice:systembuilder:HelloWorld_v:1.0

File: C:/lscc/propel/1.1/workspace/HelloWorld/verification/

```
$ sbp_design close
INFO - Finished: sbp_design close
Finished: "sbp_design close"

$ sbp_design open -name HelloWorld_v -path {C:/lscc/propel/1.1/workspace/HelloWorld/verification/HelloWorld_v.sbx}
$
```

Figure 2.56. Switching to Project Verification

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

2.3.3. Opening a Verification Project

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

2.3.4. Adding Modules, IPs and VIPs

Refer to the Adding Modules section for procedure in detail.

2.3.5. Working with the Schematic View

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

2.3.6. Connecting Modules

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

2.3.7. Monitoring DUT

This feature is available to an SOC Verification project only. In Propel Builder, the pin inside DUT can be connected to the input pin of a VIP. The connected pins can be found at both ends of the orange line, as shown in Figure 2.57. Only pin, pin bus are supported in the current release of Propel Builder.

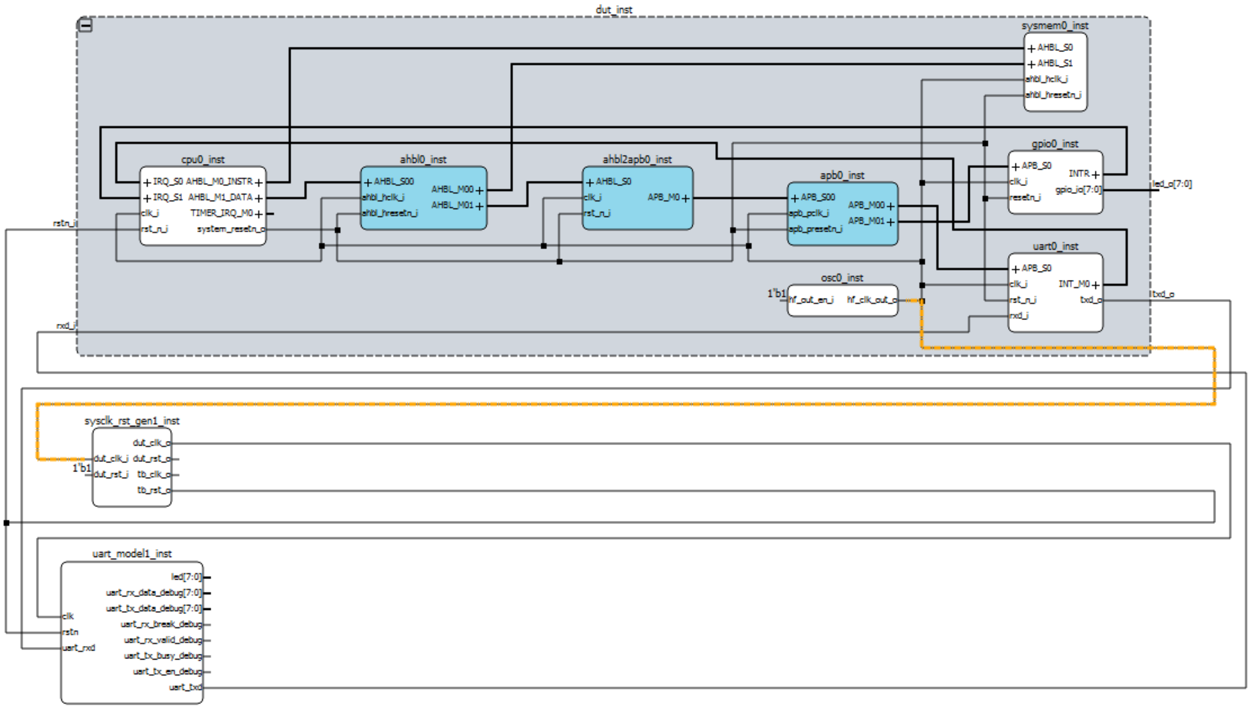
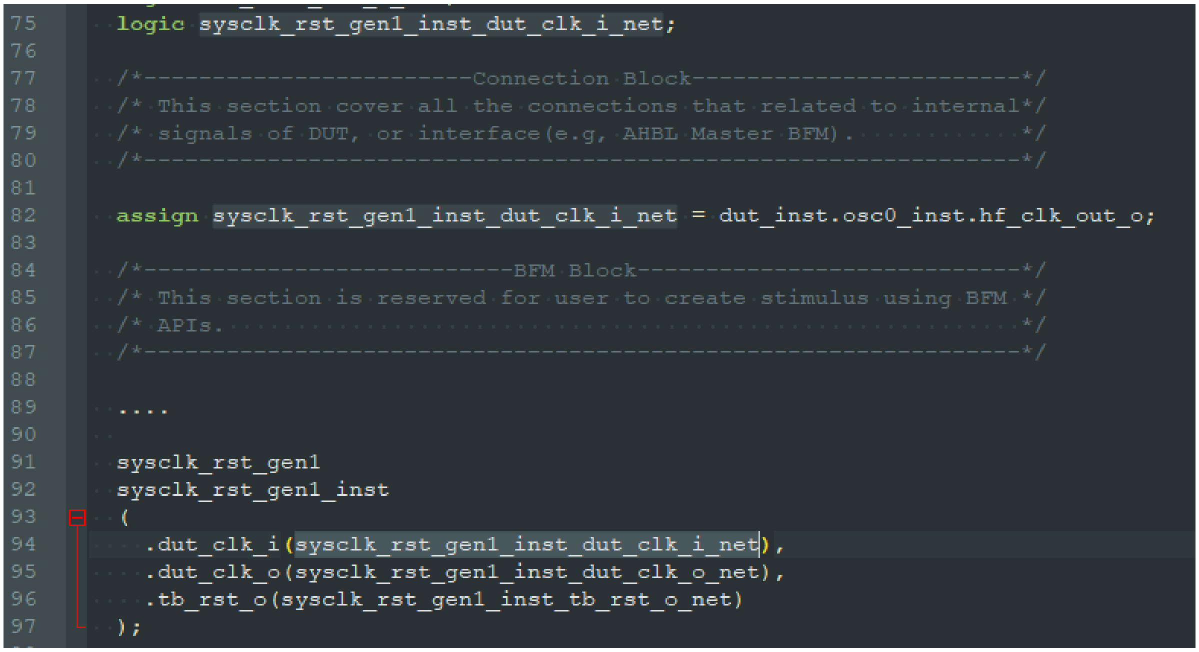


Figure 2.57. Monitoring DUT

The testbench generated for this Verification project is shown in Figure 2.58. This testbench file can be used for simulation.



```
  logic sysclk_rst_gen1_inst_dut_clk_i_net;

  /*-----------------------Connection Block-----------------------*/
  /* This section cover all the connections that related to internal*/
  /* signals of DUT, or interface(e.g, AHBL Master BFM).          */
  /*---------------------------------------------------------------*/

  assign sysclk_rst_gen1_inst_dut_clk_i_net = dut_inst.osc0_inst.hf_clk_out_o;

  /*--------------------------BFM Block---------------------------*/
  /* This section is reserved for user to create stimulus using BFM */
  /* APIs.                                                         */
  /*---------------------------------------------------------------*/

  ....

  sysclk_rst_gen1
  sysclk_rst_gen1_inst
  (
    .dut_clk_i(sysclk_rst_gen1_inst_dut_clk_i_net),
    .dut_clk_o(sysclk_rst_gen1_inst_dut_clk_o_net),
    .tb_rst_o(sysclk_rst_gen1_inst_tb_rst_o_net)
  );
```

Figure 2.58. Testbench of the Verification Project

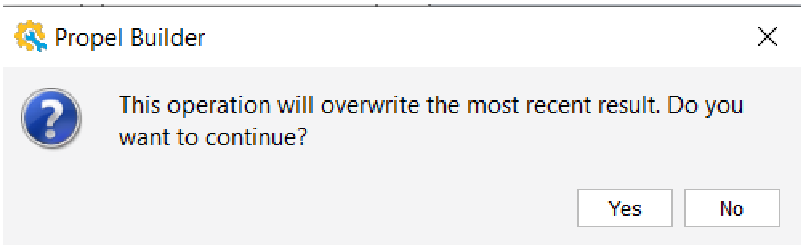
2.3.8. Generating Simulation Environment

1. Click the **Generate** icon from the Propel Builder Toolbar to generate a testbench file including the scripts for the chosen simulator, file list for HDLs, and some other files. The testbench file structure is shown in Figure 2.59.

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

Figure 2.59. Testbench File Structure

Unlike those HDLs generated in the SOC design, the testbench generation in the Verification project is just a start point for you to work with. Make sure that you generate a new testbench, if there is an existing simulation environment. A dialog box pops up prompting you to make a choice of Yes or No (Figure 2.60).

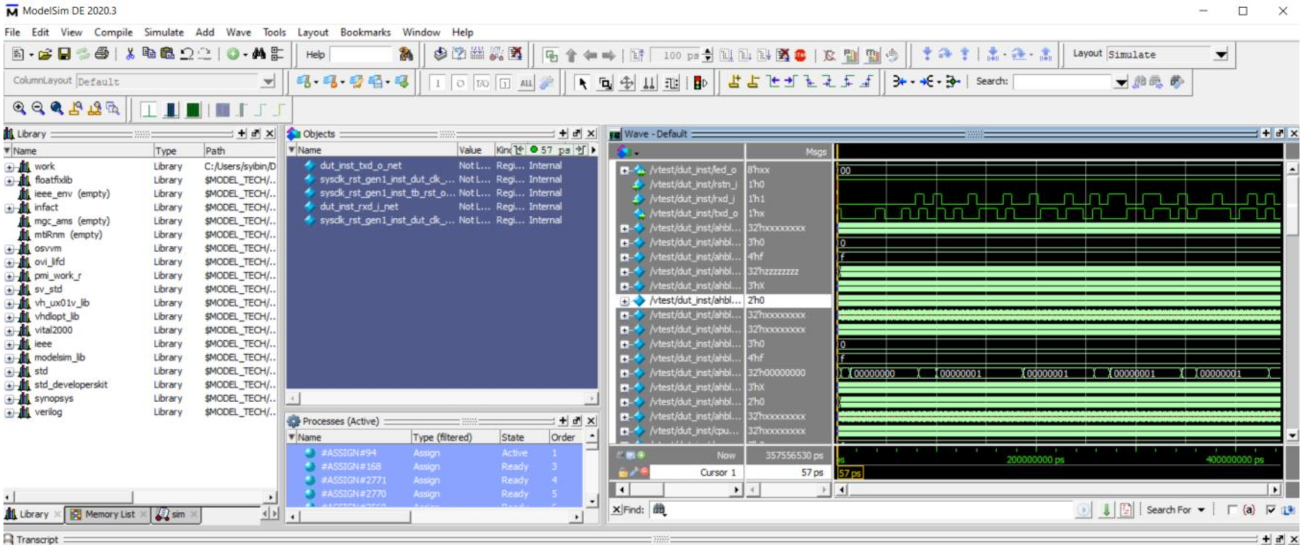


This operation will overwrite the most recent result. Do you want to continue?

Figure 2.60. Propel Builder – A Reminding Dialog Box

2.3.9. Launching Simulation

1. Click **Launch Simulation** icon from the Propel Builder Toolbar to launch simulation.



Name	Type	Path
work	Library	C:/Users/syibin/D
floatfixlib	Library	$MODEL_TECH/..
ieee_env (empty)	Library	$MODEL_TECH/..
infact	Library	$MODEL_TECH/..
mgc_ams (empty)	Library	$MODEL_TECH/..
mtiRnm (empty)	Library	$MODEL_TECH/..
osvvm	Library	$MODEL_TECH/..
ovi_lifd	Library	$MODEL_TECH/..
pmi_work_r	Library	$MODEL_TECH/..
sv_std	Library	$MODEL_TECH/..
vh_ux01v_lib	Library	$MODEL_TECH/..
vhdlopt_lib	Library	$MODEL_TECH/..
vital2000	Library	$MODEL_TECH/..
ieee	Library	$MODEL_TECH/..
modelsim_lib	Library	$MODEL_TECH/..
std	Library	$MODEL_TECH/..
std_developerskit	Library	$MODEL_TECH/..
synopsys	Library	$MODEL_TECH/..
verilog	Library	$MODEL_TECH/..

Name	Value	Kind	
dut_inst_txd_o_net	Not L...	Regi...	Internal
sysclk_rst_gen1_inst_dut_clk_...	Not L...	Regi...	Internal
sysclk_rst_gen1_inst_tb_rst_o...	Not L...	Regi...	Internal
dut_inst_rxd_i_net	Not L...	Regi...	Internal
sysclk_rst_gen1_inst_dut_clk_...	Not L...	Regi...	Internal

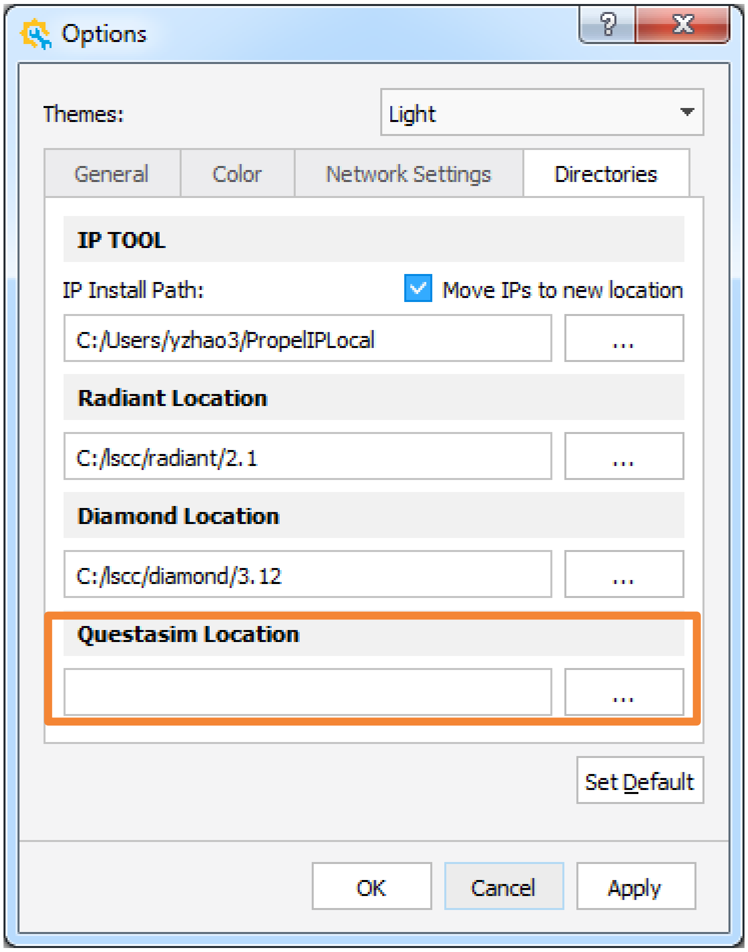
Name	Type (filtered)	State	Order
#ASSIGN#94	Assign	Active	1
#ASSIGN#168	Assign	Ready	3
#ASSIGN#2771	Assign	Ready	4
#ASSIGN#2770	Assign	Ready	5

	Msgs
/vtest/dut_inst/led_o	8'hxx
/vtest/dut_inst/rstn_i	1'h0
/vtest/dut_inst/rxd_i	1'h1
/vtest/dut_inst/txd_o	1'hx
/vtest/dut_inst/ahbl...	32'hxxxxxxxx
/vtest/dut_inst/ahbl...	3'h0
/vtest/dut_inst/ahbl...	4'hf
/vtest/dut_inst/ahbl...	32'hzzzzzzzz
/vtest/dut_inst/ahbl...	3'hX
/vtest/dut_inst/ahbl...	2'h0
/vtest/dut_inst/ahbl...	32'hxxxxxxxx
/vtest/dut_inst/ahbl...	32'hxxxxxxxx
/vtest/dut_inst/ahbl...	3'h0
/vtest/dut_inst/ahbl...	4'hf
/vtest/dut_inst/ahbl...	32'h00000000
/vtest/dut_inst/ahbl...	3'hX
/vtest/dut_inst/ahbl...	2'h0
/vtest/dut_inst/ahbl...	32'hxxxxxxxx
/vtest/dut_inst/cpu...	32'hxxxxxxxx
Now	357556530 ps
Cursor 1	57 ps

Figure 2.61. Simulation GUI

2. The default simulation tool, ModelSim, OEM version, opens (Figure 2.61).

   Note: Before launching simulation, you can change Simulator to Questasim by clicking **Design** > **Options** from Builder Menu bar. Builder Options Wizard opens (Figure 2.62). Click **Directories** to set desired Questasim Location.



Themes: Light

IP TOOL

IP Install Path: ☑ Move IPs to new location

C:/Users/yzhao3/PropelIPLocal

Radiant Location

C:/lscc/radiant/2.1

Diamond Location

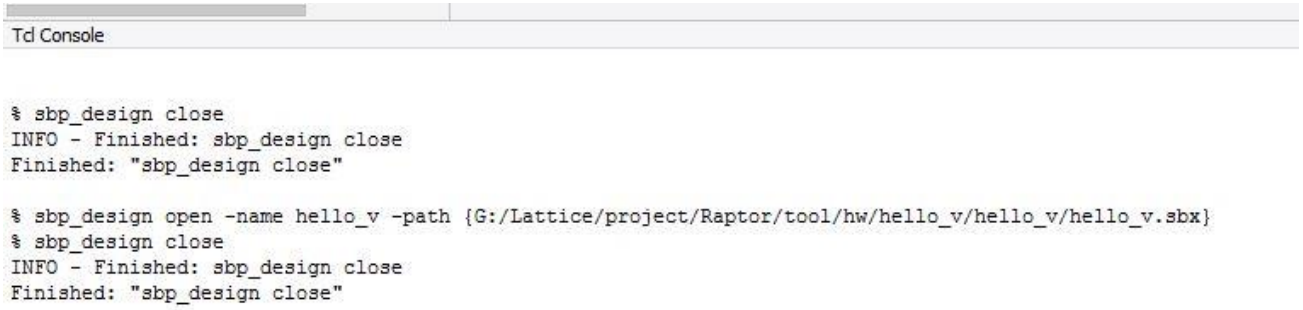
C:/lscc/diamond/3.12

Questasim Location

Figure 2.62. Builder Options Wizard

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

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

3.1.1. Open

Opens an existing Propel Builder design for modification.

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

3.1.2. Close

Closes a Propel Builder design currently opened.

Usage	sbp_design close

3.1.3. New

Creates a new Propel Builder design.

Usage	sbp_design new -name <new design name> -path <new design path>

3.1.4. Save

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

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

3.1.5. Drc

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

Usage	sbp_design drc

3.1.6. Generate

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

Usage	sbp_design generate

3.1.7. auto_assign_addresses

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

Usage	sbp_design auto_assign_addresses

3.1.8. Launch SoC Verification Engine

Launches the verification engine to verify design.

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

3.1.9. Execute PGE Engine

Runs command with different parameters to call SGE function, DGE function, and Signature.

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

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

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

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

- Run command to generate signature.

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

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

3.2. Other TCL Commands

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

3.2.1. sbp_add_component

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

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

3.2.2. sbp_add_sbx_component

Instantiates a hierarchical component from sbx file into the system.

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

3.2.3. sbp_add port

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

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

3.2.4. sbp_connect_net

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

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

3.2.5. sbp_connect_interface_net

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

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

3.2.6. sbp_connect_constant

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

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

3.2.7. sbp_connect_whitebox

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

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

3.2.8. disconnect whitebox connection

Removes the connection cross the hierarchy boundary

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

3.2.9. sbp_disconnect_interface_net

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

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

3.2.10. sbp_disconnect_net

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

Usage	sbp_disconnect_net <pin0> <pin1> <port2>

3.2.11. sbp_assign_addr_seg

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

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

3.2.12. sbp_unassign_addr_seg

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

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

3.2.13. sbp_assign_local_memory

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

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

3.2.14. sbp_export_pins

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

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

3.2.15. sbp_export_interface

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

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

3.2.16. sbp_rename

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

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

3.2.17. sbp_replace

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

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

3.2.18. sbp_copy

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

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

3.2.19. sbp_delete

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

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

3.2.20. sbp_get_components

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

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

3.2.21. sbp_get_pins

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

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

3.2.22. sbp_get_interface_pins

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

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

3.2.23. sbp_get_ports

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

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

3.2.24. sbp_get_interface_ports

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

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

3.2.25. sbp_get_nets

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

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

3.2.26. sbp_get_interface_nets

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

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

3.2.27. sbp_set_property

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

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

3.2.28. sbp_get_property

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

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

3.2.29. sbp_report_properties

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

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

References

- Lattice Propel SDK 1.1 Usage Guide (FPGA-UG-02115)
- Lattice Propel 1.1 Installation Notice for Windows (FPGA-AN-02030)
- Lattice Diamond Help
- Lattice Radiant Help

Technical Support Assistance

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

Revision History

Revision 1.0, November 2020

Section	Change Summary
All	Initial release.

LATTICE SEMICONDUCTOR®

www.latticesemi.com