

Synopsys System Designer

User Guide

December 2009

SYNOPSYS®

:

<http://solvnet.synopsys.com>

Disclaimer of Warranty

Synopsys, Inc. makes no representations or warranties, either expressed or implied, by or with respect to anything in this manual, and shall not be liable for any implied warranties of merchantability or fitness for a particular purpose of for any indirect, special or consequential damages.

Copyright Notice

Copyright © 2009 Synopsys, Inc. All Rights Reserved.

Synopsys software products contain certain confidential information of Synopsys, Inc. Use of this copyright notice is precautionary and does not imply publication or disclosure. No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language in any form by any means without the prior written permission of Synopsys, Inc. While every precaution has been taken in the preparation of this book, Synopsys, Inc. assumes no responsibility for errors or omissions. This publication and the features described herein are subject to change without notice.

Trademarks

Registered Trademarks (®)

Synopsys, AMPS, Astro, Behavior Extracting Synthesis Technology, Cadabra, CATS, Certify, Design Compiler, DesignWare, Formality, HDL Analyst, HSPICE, Identify, iN-Phase, Leda, MAST, ModelTools, NanoSim, OpenVera, PathMill, Physical Compiler, PrimeTime, SiVL, SCOPE, Simply Better Results, SNUG, SolvNet, Synplicity, the Synplicity logo, Synplify, Synplify Pro, Synthesis Constraints Optimization Environment, TetraMAX, VCS, Vera, and YIELDirector are registered trademarks of Synopsys, Inc.

Trademarks (™)

AFGen, Apollo, Astro-Rail, Astro-Xtalk, Aurora, AvanWaves, BEST, Columbia, Columbia-CE, Cosmos, CosmosLE, CosmosScope, CRITIC, DC Expert, DC Professional, DC Ultra, Design Analyzer, Design Vision, Design-erHDL, DesignPower, Direct Silicon Access, Discovery, Eclipse, Encore, EPIC, Galaxy, HANEX, HAPS, HapsTrak, HDL Compiler, Hercules, Hierarchical Optimization Technology, High-performance ASIC Prototyping System, HSIM, HSIM^{plus}, i-Virtual Stepper, IICE, in-Sync, iN-Tandem, Jupiter, Jupiter-DP, JupiterXT, JupiterXT-ASIC, Liberty, Libra-Passport, Library Compiler, Magellan, Mars, Mars-Rail, Mars-Xtalk, Milkyway, ModelSource, Module Compiler, MultiPoint, Physical Analyst, Planet, Planet-PL, Polaris, Power Compiler, Raphael, Saturn, Scirocco, Scirocco-i, Star-RCXT, Star-SimXT, System Compiler, System Designer, Taurus, TotalRecall, TSUPREM-4, VCS Express, VCSi, VHDL Compiler, VirSim, and VMC are trademarks of Synopsys, Inc.

Service Marks (SM)

MAP-in, SVP Café, and TAP-in are service marks of Synopsys, Inc.

SystemC is a trademark of the Open SystemC Initiative and is used under license. ARM and AMBA are registered trademarks of ARM Limited. Saber is a registered trademark of SabreMark Limited Partnership and is used under license. All other product or company names may be trademarks of their respective owners.

Restricted Rights Legend

Government Users: Use, reproduction, release, modification, or disclosure of this commercial computer software, or of any related documentation of any kind, is restricted in accordance with FAR 12.212 and DFARS 227.7202, and further restricted by the Synopsys Software License and Maintenance Agreement. Synopsys, Inc., Synplicity Business Group, 600 West California Avenue, Sunnyvale, CA 94086, U. S. A.

Printed in the U.S.A
December 2009

:

Contents

Chapter 1: Getting Started with System Designer

About System Designer	1-10
Related Synopsys FPGA Tools	1-14
Accessing Help	1-15
Setting up the System Designer Tool	1-17

Chapter 2: Using System Designer

System Designer Design Flow	2-20
Setting up a System Designer Project	2-21
Opening a System Designer Project	2-21
Setting Global Language Options	2-24
Setting System Designer Preferences	2-24
Setting Preferences for Clearing the Console View	2-24
Setting View Preferences	2-25
Setting Validation Preferences	2-26
Working with Preference Files	2-27
Assembling Components	2-28
Loading IP Libraries	2-28
Downloading Evaluation IP	2-30
Adding IP Components	2-31
Configuring IP Cores	2-33
Making Interface-Level Connections	2-35
Connecting an Interface to a Bus	2-35
Directly Connecting Interfaces	2-37
Making Hierarchical Connections at the Interface Level	2-39
Removing Interface-Level Connections	2-40
Making Port-Level Connections	2-40
Making Port-to-Port Connections	2-41

Making Hierarchical Port Connections	2-42
Tying Ports to a Constant Value	2-43
Removing Port-Level Connections	2-44
Connecting and Configuring Automatically	2-45
Setting Auto-Connection Preferences	2-45
Automatically Connecting Component Instances	2-46
Automatically Configuring Addresses	2-47
Checking Constraints	2-48
Using Tcl Scripts	2-49
Generating Output Files for Synthesis	2-49
Verifying the Design with Simulation	2-50
Running Synthesis and P&R	2-50
Using System Designer with a HAPS Board	2-52
The System Designer-HAPS Board Design Flow	2-52
Connecting HAPS Peripherals	2-56
Assigning Traces for Designs with HAPS Boards	2-57

Chapter 3: Menus and Commands

The System Designer User Interface	3-62
Display in the Design View	3-67
Interface Connections View	3-72
Port Connections View	3-74
Edit Port Connection Command	3-76
Trace Assignment View	3-78
Project Menu Commands	3-81
New Project Command	3-81
Initialize Project Command	3-82
Library Menu Commands	3-84
Load IP Library Command	3-84
Actions Menu Commands	3-86
Generate Files Command	3-87
Open Design Configuration Command	3-87
Open Design Constraints Command	3-91
Preferences Command	3-95
Help Menu Commands	3-112

Context-Sensitive Command Menus	3-112
Popup Commands in the Design View	3-113
Popup Commands in the Console and Command Shell Windows	3-113
Popup Commands in the Memory Map Window	3-114

Chapter 4: Tcl Commands

Overview of Tcl Commands	4-116
Library Commands	4-117
Connectivity Commands	4-117
Design Commands	4-120
Output File Commands	4-121
Project Commands	4-122
Preference Commands	4-122

Appendix A: System Designer Tutorials

Tutorial Design Flows	A-132
Create a Synthesis Project	A-134
Create a System Designer Project	A-135
Set up the IP Library	A-138
Set Auto-Connection Preferences	A-138
Load the Library	A-138
Create an Embedded System (Basic)	A-141
Assemble the IP Core Components (Basic)	A-141
Connect the IP Core Components (Basic)	A-142
Create an Embedded System (Advanced)	A-144
Assemble the IP Core Components (Advanced)	A-145
Rename Instances (Advanced)	A-146
Connect the IP Core Components (Advanced)	A-147
Configure Instance Addresses (Advanced)	A-152
Configure Properties and Parameters (Advanced)	A-153
Create an Embedded System (Expert)	A-155
Assemble the IP Core Components (Expert)	A-156
Rename Instances (Expert)	A-157
Connect the IP Core Components (Expert)	A-159
Configure Instance Addresses (Expert)	A-161
Configure Properties and Parameters (Expert)	A-162

Run Synthesis and Place and Route A-163

Analyze the Design A-165

CHAPTER 1

Getting Started with System Designer

This section provides a brief overview of the Synopsys® FPGA System Designer tool. It describes the following:

- [About System Designer, on page 10](#)
- [Related Synopsys FPGA Tools, on page 14](#)
- [Accessing Help, on page 15](#)
- [Setting up the System Designer Tool, on page 17](#)

About System Designer

The Synopsys® FPGA System Designer™ tool is an implementation of the IP-XACT design environment as described in the IP-XACT specifications. This tool lets you select, configure, and assemble internal and third-party IP in the IP-XACT format, integrate that IP, and then easily implement it into a variety of FPGA vendor devices, including those from Actel, Altera, Lattice Semiconductor and Xilinx. Using IP and system-level blocks, the System Designer tool provides FPGA designers with a productive way to implement complex systems in FPGAs.

See the following for an overview of the tool:

- [System Designer and IP-XACT, on page 10](#)
- [System Designer and the FPGA Synthesis Tools, on page 12](#)
- [System Designer and the ReadyIP Design Flow, on page 12](#)

System Designer and IP-XACT

The System Designer tool is an implementation of the IP-XACT design environment as described in the IP-XACT specifications. IP-XACT is a standard specification maintained by The SPIRIT Consortium to enable the rapid, reliable deployment of IP into advanced design environments. Refer to The SPIRIT Consortium website (<http://www.spiritconsortium.org/home>) for details of the IP-XACT specification. The System Designer tool supports the following versions of the IP-XACT standard:

- IP-XACT version 1.1
- IP-XACT version 1.2
- IP-XACT version 1.4

The System Designer software provides an environment to accomplish the following design tasks described in the IP-XACT standard:

IP-XACT Task	Corresponding System Designer Features
Assembling IP cores	You can add or remove component instances.
Configuring IP cores	You can do the following: <ul style="list-style-type: none"> • Edit properties of component instances • Set parameters for components and bus interfaces • Set default signal values for components • Edit memory maps
Connecting IP cores	The tool lets you make the following kinds of connections: <ul style="list-style-type: none"> • Instance interface to a bus interface • Instance interface to another instance interface • Hierarchical connections between an instance interface and top-level design ports • Connections on a port-by-port basis (ad-hoc connections)
Generating RTL files	The tool generates top-level and component wrapper files in VHDL and Verilog.
Generating IP-XACT design file	Generates an IP-XACT file called <i><design>.xml</i> with the design information.
Generating a file for simulation	The tool generates a VCS script file (<i>vcs.ksh</i>) file for the system.

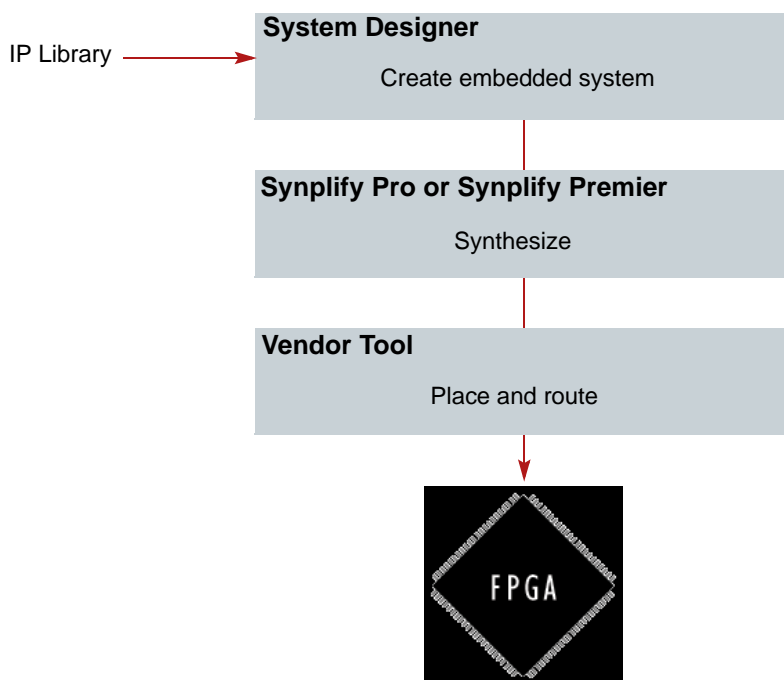
The System Designer tool does not currently support the following IP-XACT features:

- Transaction-level modeling (TLM)
- Tight generator interface (TGI)
- Design configuration and abstractor descriptions
- White box elements and monitor interfaces

System Designer and the FPGA Synthesis Tools

The System Designer tool is bundled with the Synplify Pro and Synplify Premier synthesis tools. Although it has a separate interface, it is only available from within the FPGA synthesis tools. It takes IP library input in the IP-XACT format from the SPIRIT consortium, and outputs top-level RTL for the IP and a project file that is ready for logic or physical synthesis.

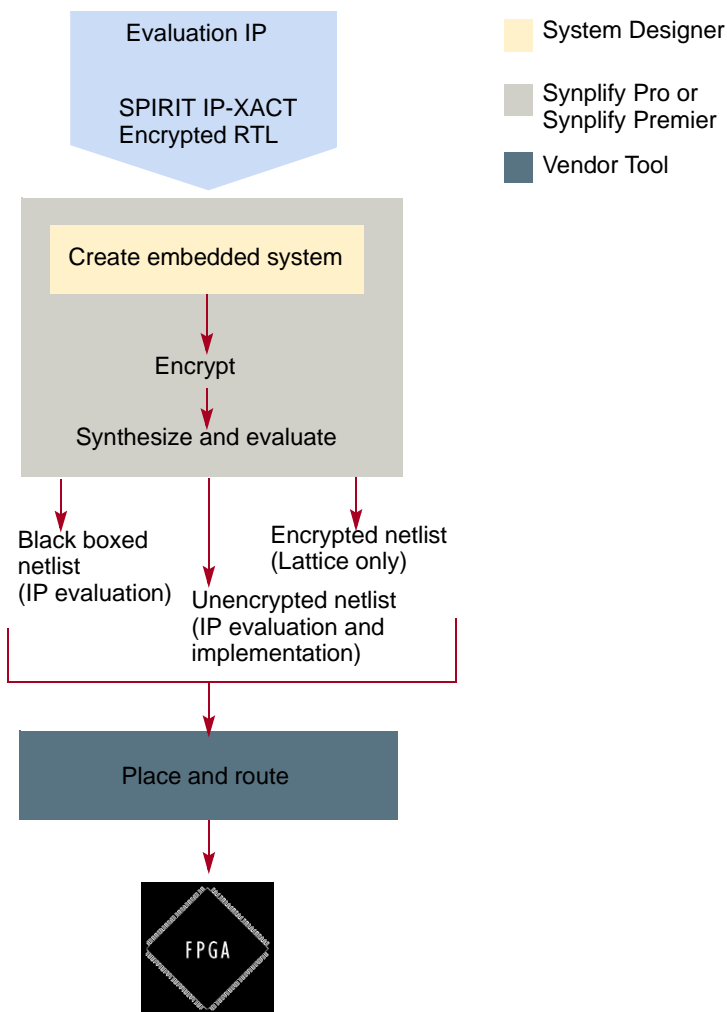
The following figure shows how the System Designer software works with the Synplify Pro or Synplify Premier tools:



System Designer and the ReadyIP Design Flow

The System Designer functionality is a key component of the Synopsys ReadyIP Initiative, a program that tries to simplify the access, evaluation and use of IP for FPGA-based system designs. It is a standards-based flow, and provides you with the choice of multiple FPGA targets on the back end, like Actel, Altera, Lattice Semiconductor, and Xilinx.

The following figure elaborates on how the System Designer tool is an integral part of the ReadyIP vision:



Related Synopsys FPGA Tools

This section briefly describes the following Synopsys® tools that you can use with or in parallel with the System Designer tool to enhance your design flow:

- [Synplify Pro and Synplify Premier Software, on page 14](#)
- [VCS Software, on page 14](#)
- [Identify Software, on page 15](#)
- [Synphony HLS Software, on page 15](#)

Synplify Pro and Synplify Premier Software

The Synplify Pro® tool is the most widely-used logic synthesis solution for FPGAs (Field Programmable Gate Arrays) and Complex PLDs (Programmable Logic Devices). The Synplify® Premier tool offers a push-button, graph-based approach to physical synthesis, improving overall device performance while simultaneously delivering tight correlation between pre-route timing estimates and final post place-and-route results. Both synthesis tools use a true timing-driven approach to synthesis, deliver the performance you need to meet the design timing requirements, and optimize the circuit for area, which significantly reduces chip costs.

You can only access and run the System Designer tool from within the Synplify Pro or the Synplify Premier interface.

VCS Software

VCS® is a Synopsys product that provides comprehensive RTL verification. It is based on advanced bug-finding technologies and includes a built-in debug and visualization environment and support for all popular design and verification languages including Verilog, VHDL, SystemVerilog and SystemC™. The tool includes full-featured Native Testbench, complete assertions and comprehensive code and functional coverage to find more bugs faster and easier. Additionally, the VCS Verification Library provides verification IP for the most popular bus standards. The System Designer tool automatically writes out a Tcl script that you can use as input for verification with the VCS software.

Identify Software

The Synopsys Identify® RTL Debugger software lets you probe and debug your FPGA design directly in the source RTL. You use the Identify software to verify your design in hardware as you would in simulation, only much faster and with in-system stimulus.

This tool allows you to navigate your design graphically and mark signals directly in RTL as probes or sample triggers. After synthesis, you view the results in the RTL source code or in waveform. Design iterations are rapidly done using incremental place and route. Identify software is closely integrated with synthesis and routing tools for a seamless development.

Symphony HLS Software

The Symphony HLS product is a high-level tool for hardware DSP design. It is an add-on to the Simulink® product from The MathWorks®, and provides the designer with an automated path from high-level design and simulation to an architecturally-optimized, synthesizable, system-level HDL implementation. This tool provides performance and productivity benefits for designers who are implementing DSP circuits into FPGA and ASIC devices. The software achieves significantly higher performance than alternative solutions and provides the designer with a mechanism to evaluate high-level area/performance tradeoffs. The output is synthesizable HDL code ready for use with the Synopsys® Synplify Pro® or ASIC synthesis software.

Accessing Help

The following describe how to access the online documentation in the System Designer interface and in the other Synopsys tools that work with the System Designer tool:

- [Viewing System Designer Online Help, on page 16](#)
- [Viewing System Designer PDFs, on page 16](#)
- [Viewing FPGA Synthesis Online Documentation, on page 16](#)

Viewing System Designer Online Help

The System Designer tool has a separate interface that includes its own online documentation. You can access the HTML online version of the documentation by first opening the System Designer interface and then selecting Help->Help Contents.

Viewing System Designer PDFs

The System Designer tool also includes a PDF version of the System Designer User Guide, located in `<synplify_install_dir>/doc`. You can also access this document from the System Designer interface, using the Help->Online Documents command.

Viewing FPGA Synthesis Online Documentation

Do the following to display the online documentation for the Synplify Pro and/or Synplify Premier tools:

1. Start the synthesis tool.
2. For the online help, select Help -> Help from the UI or press F1. You can then browse the contents.
3. For the PDF versions of the synthesis tool documentation, do the following:
 - Select Help -> Online Documents. A dialog box lists the PDFs available.
 - Double-click the document you want to open.

Setting up the System Designer Tool

The System Designer tool and is only available from within the synthesis tool interface. Do the following to use the tool:

1. Set up a project in the Synplify Pro or Synplify Premier synthesis tools, that will include the embedded system you create with the System Designer tool.

You must start with a synthesis project in the synthesis tool.

2. To launch System Designer, select Import->Launch System Designer in the synthesis tool, or click the System Designer icon in the toolbar.

This opens the System Designer tool. You are now ready to set up your project, as described in [Setting up a System Designer Project, on page 21](#).

CHAPTER 2

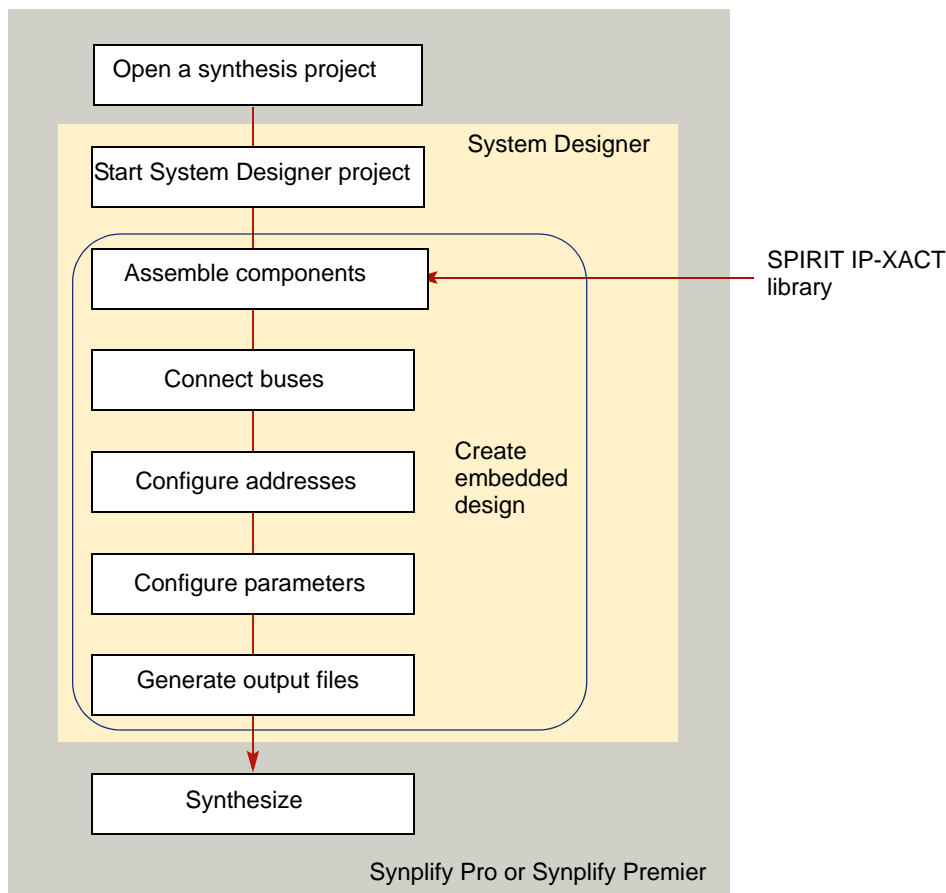
Using System Designer

The following describe the usual design flow for creating a design with the System Designer tool, and then describes individual steps in more detail. If you want to follow a step-by-step flow example, see [Appendix A, *System Designer Tutorials*](#).

- [System Designer Design Flow, on page 20](#)
- [Setting up a System Designer Project, on page 21](#)
- [Setting System Designer Preferences, on page 24](#)
- [Assembling Components, on page 28](#)
- [Configuring IP Cores, on page 33](#)
- [Making Interface-Level Connections, on page 35](#)
- [Making Port-Level Connections, on page 40](#)
- [Connecting and Configuring Automatically, on page 45](#)
- [Checking Constraints, on page 48](#)
- [Using Tcl Scripts, on page 49](#)
- [Generating Output Files for Synthesis, on page 49](#)
- [Verifying the Design with Simulation, on page 50](#)
- [Running Synthesis and P&R, on page 50](#)
- [Using System Designer with a HAPS Board, on page 52](#)

System Designer Design Flow

The System Designer tool provides an interactive mechanism to design an embedded IP core. The following figure shows the steps you would typically follow to create the embedded system in the System Designer tool. The figure also shows how the System Designer tool is embedded within the synthesis tool and fits into the synthesis design flow. You use the synthesis tool to synthesize the embedded system once it is assembled.



Setting up a System Designer Project

To set up a System Designer project, you execute the first two steps in the System Designer design flow (*System Designer Design Flow*, on page 20). See the following:

- [Opening a System Designer Project](#), on page 21
- [Setting Global Language Options](#), on page 24

Opening a System Designer Project

1. Create a synthesis project in the Synplify Pro or Synplify Premier tools.

If you already have a synthesis design open and then start System Designer, the System Designer design is added to the current synthesis project.

2. Start System Designer.

See [Setting up the System Designer Tool](#), on page 17 for details about setting up the tool.

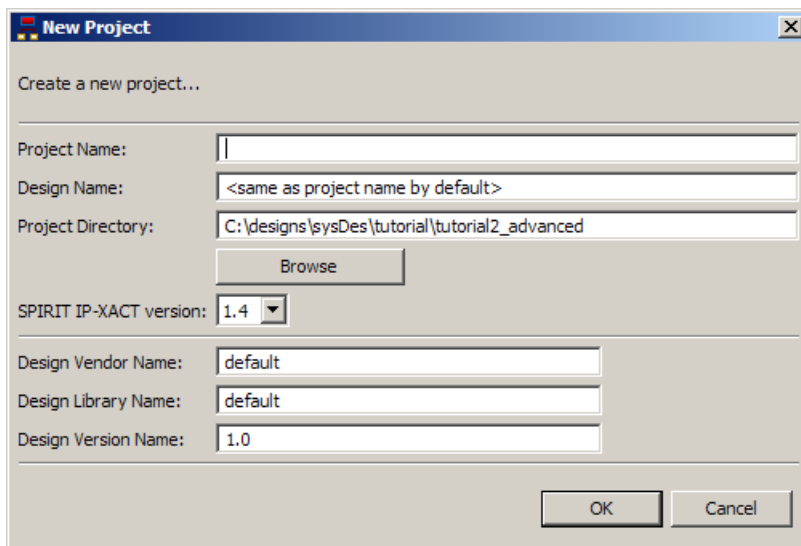
3. To open an existing System Designer project, do either of the following:
 - In the Initialize Project dialog box that comes up when you first launch the tool, select Open an Existing Project and click OK.
 - Alternatively, select Project-> Open Project from the System Designer interface. This displays the Select A Project File To Open window. Browse to the project file you want (*<project_name>.prx*), select it, and click Open button.

If you are just starting the tool, the UI opens with the System Designer project you specified. If you had already started the tool, the tool loads the project you selected.

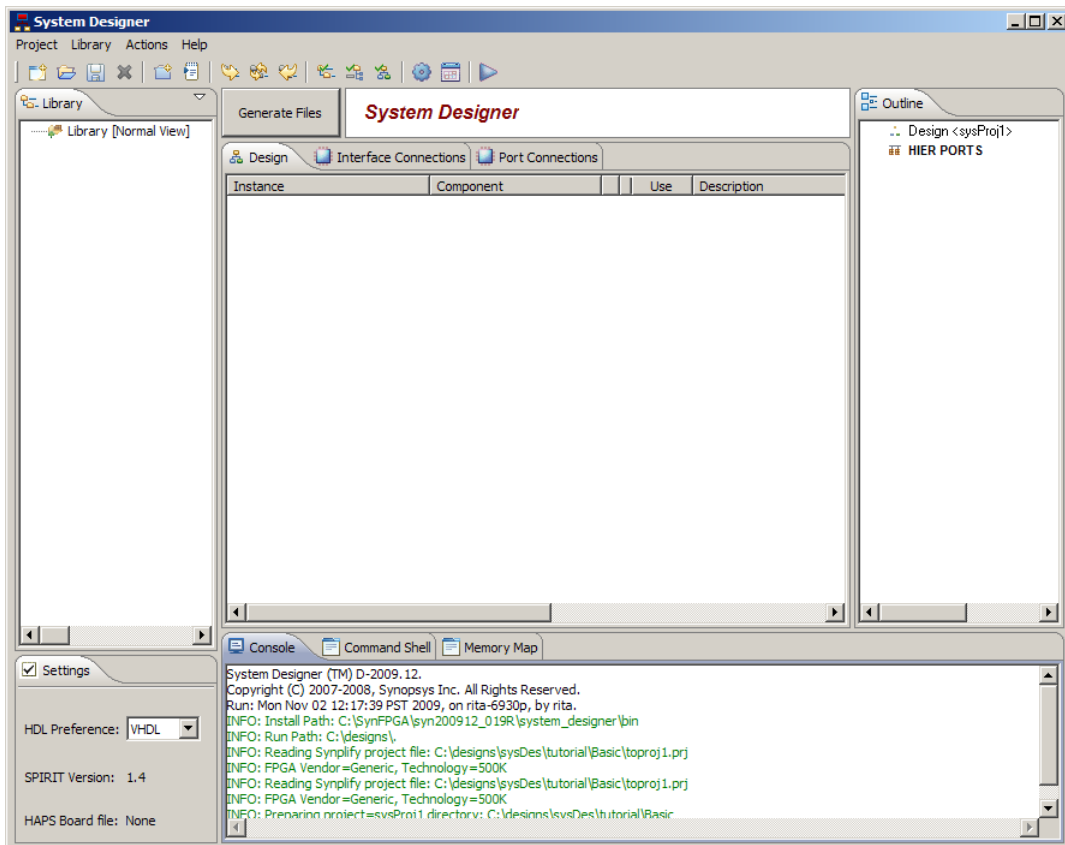
4. To create a new project, do either of the following:
 - In the Initialize Project dialog box that comes up when you first launch the tool, select Create a New Project, and click OK.
 - Alternatively, select Project-> New Project from the System Designer interface.

The New Project dialog box opens. Do the following in the dialog box:

- Specify a project name and a project directory.
- Specify a design name (by default this is the same as the project name).
- Specify the IP information like the design library, version, and vendor, or leave the defaults.
- Click OK.



If you are just starting the tool, the UI opens with the System Designer project you created. If you already started the tool, a new project opens.



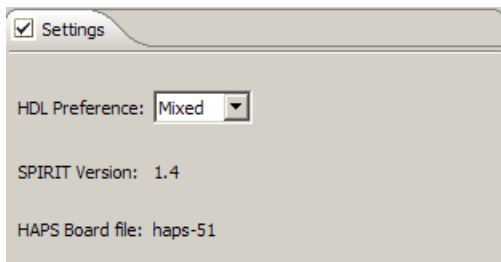
5. To close a project, select Project -> Close Project or click the Close Project button. You are prompted to save your design or close it without saving.
6. To save a project, select Project -> Save Project or click the Save Project button in UI.

If you created a new project, the software saves a project file (*<project_name>.prx*) and a design file (*<design_name>.xml*). If you are working with an existing project, the tool updates these files with the most recent information and saves them.

Setting Global Language Options

You can set global language options for the project from the main window:

1. Go to the **Settings** tab in the lower right corner of the main window.



2. Set HDL Preference to the format that matches the source files for your model views. You can set it to Verilog, VHDL or Mixed (for mixed language designs).
3. Save the design to save your project options.

Setting System Designer Preferences

The window that opens when you select **Actions -> Preferences** lets you set various preferences for working with the tool. The following show you how to set some preferences for working in the System Designer interface:

- [Setting Preferences for Clearing the Console View, on page 24](#)
- [Setting View Preferences, on page 25](#)
- [Setting Validation Preferences, on page 26](#)
- [Working with Preference Files, on page 27](#)

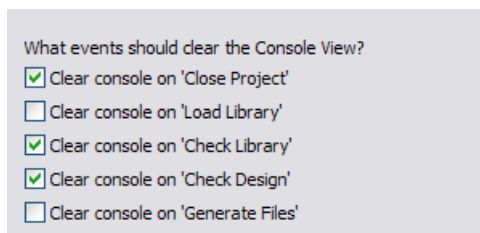
Setting Preferences for Clearing the Console View

You can configure your console to automatically clear, by following these steps.

1. Select Actions -> Preferences.

The Preferences window opens.

2. Select General on the left side of the window, if it is not already selected.
3. Specify when you want the console view to clear by enabling or disabling the appropriate options.



4. Click Apply or OK.

Setting View Preferences

You can configure your System Designer setup to display various views.

1. Select Actions -> Preferences.

The Preferences window opens.

2. Set the display options you want. The following table summarizes what you need to do in this window to display or hide different views:

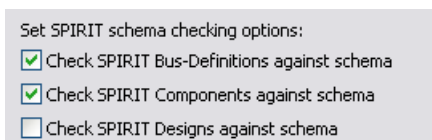
Library view	<ul style="list-style-type: none">• On the left side of the window, click Library View (under GUI Preferences).• Enable or disable Show Library View Description column, according to what you want to display.
Design view	<ul style="list-style-type: none">• On the left side of the window, click Design View (under GUI Preferences).• Enable or disable the Show Design View Description column option, according to what you want to display.
Outline view	<ul style="list-style-type: none">• On the left side of the window, click Outline View (under GUI Preferences).• Enable or disable the Show Outline View in the main GUI option, according to what you want to display.

3. Click Apply to apply the preferences you selected.
4. Click OK in the Preferences window to apply the preferences and close the window.

Setting Validation Preferences

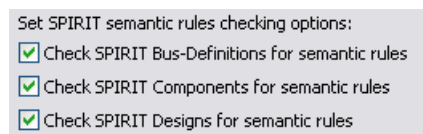
You can set various parameters to automatically check your components against the IP-XACT rules.

1. Select Actions -> Preferences.
2. To validate your library components against IP-XACT schema as it is loaded, do the following:
 - On the left side of the window, click SPIRIT Schema (under Checking).
 - Enable the components you want to check.



To validate a library that has already been loaded, select Actions -> Check Library Schema.

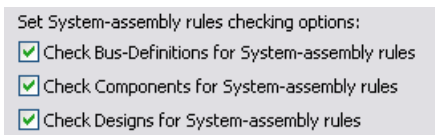
3. To validate your library components against IP-XACT semantic rules as it is loaded, do the following:
 - On the left side of the window, click Semantic Rules (under Checking).
 - Enable the components you want to check.



To validate a library that has already been loaded, select Actions -> Check Library Rules.

4. To validate your library components against IP-XACT system assembly rules as it is loaded, do the following:
 - On the left side of the window, click Assembly Rules (under Checking).

- Enable the components you want to check.



To validate a library that has already been loaded, select Actions -> Check Design Rule.

Working with Preference Files

By default, the tool saves your preference settings (Action -> Preferences) into a file in the installation directory: `data/system_designer.prefs`. Additionally, you can create your own preference files in your home directory and/or project directory.

1. Set preferences (see [Setting Preferences for Clearing the Console View, on page 24](#), [Setting View Preferences, on page 25](#), and [Setting Validation Preferences, on page 26](#)).
2. To save a preferences file in the project directory, do this:
 - Select Action -> Preferences.
 - From the menu on the left, select Save preferences in the project directory (under General).
3. To save a preferences file in the project directory, do this:
 - Select Action -> Preferences.
 - Select Save preferences in the project directory).
4. Click Apply or OK.

If you have multiple preference files, the preference file in the project directory takes precedence over the one in the installation directory.

Assembling Components

Once you have set up your System Designer project (see [Setting up a System Designer Project, on page 21](#)) you can load the IP library and assemble the components. The following describe the details:

- [Loading IP Libraries, on page 28](#)
- [Downloading Evaluation IP, on page 30](#)
- [Adding IP Components, on page 31](#)

Loading IP Libraries

For input, the System Designer tool accepts IP that complies with the IP-XACT standard for describing IP from the SPIRIT Consortium.

You can choose a set of hardware components such as processors, buses, and peripherals from a standard or customized library to build an embedded system. The System Designer tool includes library files for Spirit IP-XACT 1.2 and Spirit IP-XACT 1.4:

Spirit IP-XACT 1.2	<code><install_dir>/examples/ipcores/spirit1_2</code>
--------------------	---

Spirit IP-XACT 1.4	<code><install_dir>/examples/ipcores/spirit1_4</code>
--------------------	---

1. Download the libraries you want to use from the IP vendors and make sure you have access to them. Make sure to download any bus definitions you need.

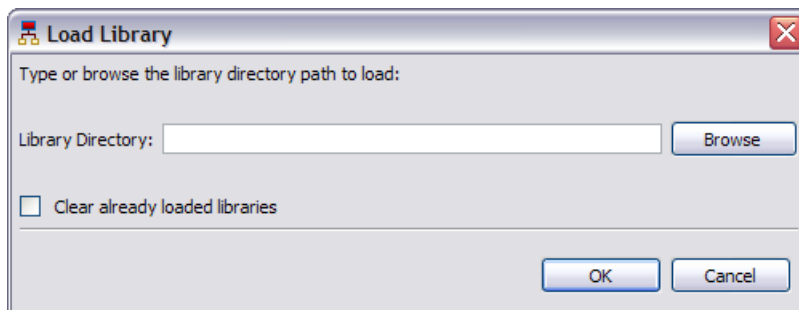
The IP library must follow the IP-XACT standard established in the SPIRIT Consortium.

The synthesis tools include easy access to some third-party IP vendor offerings. If you want to evaluate or use one of these libraries, follow the steps described in [Downloading Evaluation IP, on page 30](#).

2. If you are using an IP-XACT library, open the System Designer interface, select Actions->Preferences->General-> SPIRIT, and select the version that matches the library you want to use.

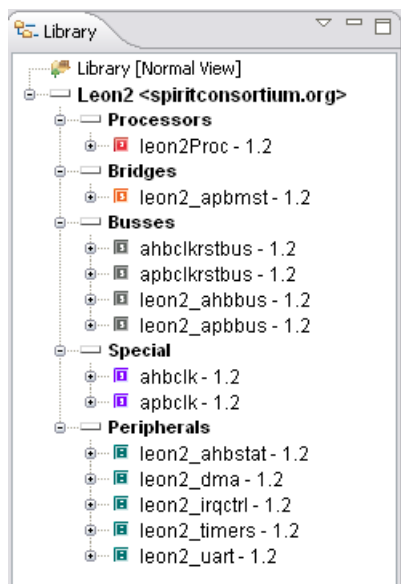
3. Select the library.

- Select Library -> Load Library, or click the Load Library button.
- Set Library Directory to the library version that matches the version you specified in the previous step. Use the Browse button if needed.



- Click OK.

This adds all library components to the System Designer project. You can view the contents in the Library tab on the left side of the UI. The Settings panel on the lower left reflects the library you selected.



After you have loaded your library, you can assemble the IP cores, as described in [Adding IP Components](#), on page 31.

4. To remove a library from the Library view, select Library -> Clear Library or click the Clear Library button.
5. To refresh a library from the Library view, select Library -> Refresh Library or click the Refresh Library button.

Downloading Evaluation IP

The Synplify Pro and Synplify Premier synthesis tools facilitate the connection between the IP vendor and the IP user by offering a mechanism through which you can evaluate and use available vendor IP in a synthesis project. To download IP into the System Designer tool for assembly and evaluation, use this procedure:

1. In the synthesis tool, select Import IP->Browse and Download ReadyIP.

The IP User Information dialog box opens.

The image shows a dialog box titled "IP User Information" with a standard Windows-style title bar (minimize, maximize, close buttons). The dialog contains several text input fields for user information: Name, Company, City, State/County/Province, Zip/Postal Code, Country, Phone Number, and E-mail. Below these fields is a message box that reads: "This information will be saved so that you don't have to enter it again when requesting IP." At the bottom right of the dialog are two buttons: "OK" and "Cancel".

2. Fill out your information, and click OK.

You are directed to a website with offerings from IP vendors.

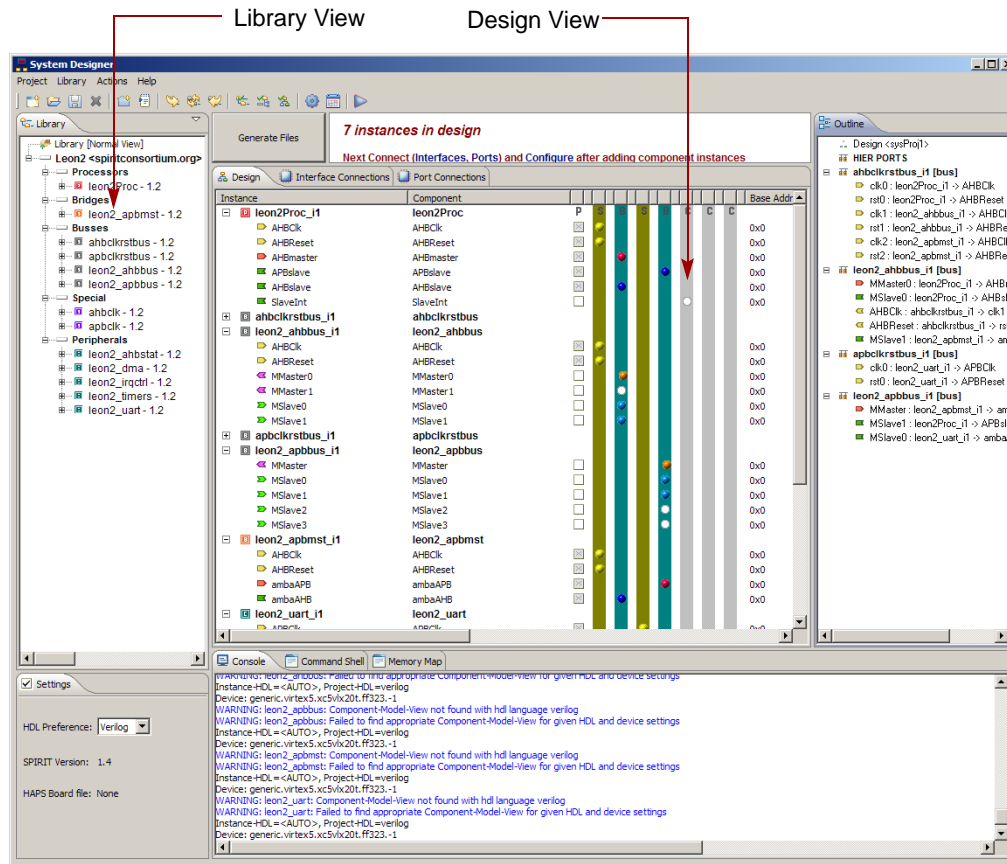
3. To download IP-XACT-based IP, do the following:
 - If you have not already done so, create or open an existing synthesis project that will include the IP.

- Start the System Designer tool (see [Setting up the System Designer Tool, on page 17](#) for details).
- In the tool, select the library version you want to use. See [Loading IP Libraries, on page 28](#) for details.
- Load the IP library as described in [Loading IP Libraries, on page 28](#).

After you have loaded your library, you can assemble the IP cores, as described in [Adding IP Components, on page 31](#).

Adding IP Components

Once you have loaded your library (see [Loading IP Libraries, on page 28](#)), you can begin to assemble your design from the library components.



1. Select the components you want to use in your embedded system.
 - Select the components in the Library view on the left of the interface.
 - Either double-click the components in Library view, or drag and drop the components from the Library view to the Design view.
2. To add an additional instance of a component, do either of the following:
 - Drag and drop the component again from the Library view to the Design view.
 - Right-click the added component and select Add another instance.


Both methods add another instance in the Design view. The instances have different number suffixes in the Instance column to distinguish them.

3. To rename an instance, do the following:
 - In the Design view, right-click the instance and select Rename Instance. This displays the Rename Instance window.
 - Edit the instance name and click OK.
4. To remove an instance from the Design view, right-click the instance to be removed and select Remove Instance.

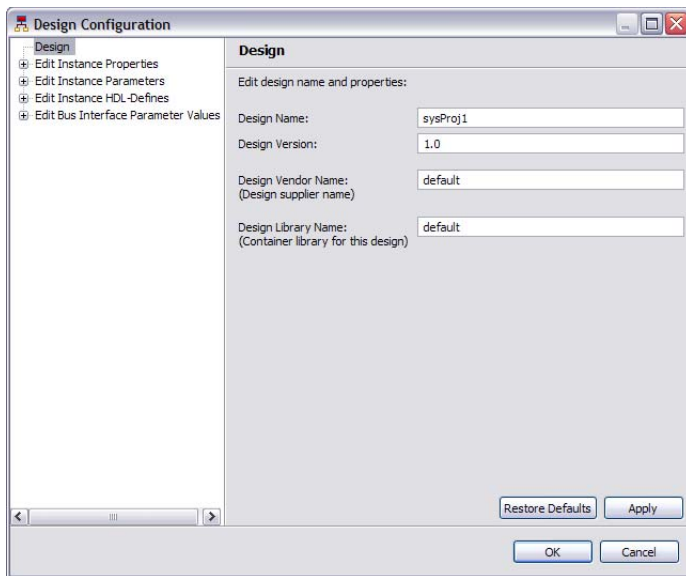
You can optionally edit and configure the components (see [Configuring IP Cores, on page 33](#)) before connecting your design (see [Making Interface-Level Connections, on page 35](#) and [Making Port-Level Connections, on page 40](#)).

Configuring IP Cores

The following procedure describes optional parameters for configuring components. Use the following procedure to set up parameters on properties on the bus interface from the Design Configuration window. This procedure shows you how to set various configuration parameters for individual components in your embedded system.

1. In the Design view, right-click on an instance and select Open configuration or click the Open Configuration icon ().

This opens the Design Configuration window.



2. To change parameters for the System Designer design, do the following:
 - Click **Design** on the left side of the window.
 - You can either leave the defaults or edit each of the design parameters. For example, to change the design name that was set when you created the project, type a new name in the field and click **Apply**.
3. To edit instance properties, do the following:
 - Click **Edit Instance Properties** on the left side of the window, and click the instance name. This displays the **Model-View** and **HDL (Hint)** tabs. The options in these tabs correspond to the options specified in the SPIRIT component design file.
 - Scroll to these tabs, edit the option you want, and click **Apply**.
4. To edit instance parameters, do the following:
 - Click **Edit Instance Parameter** on the left side of the window, and click the instance name. The parameters are displayed.
 - Edit the parameter you want.
 - Click **Apply**.

5. To edit bus parameters, do the following:
 - Click Bus Interface Parameter Values on the left side of the window, and click the instance name. The parameters are displayed.
 - Edit the parameter you want.
 - Click Apply.
6. Click OK in the Design Configuration window when you have finished configuring the components.

Making Interface-Level Connections

The SPIRIT IP-XACT standard allows the design to be connected at the interface level. An interface is a defined set of ports which can be connected to another compatible interface according to the SPIRIT IP-XACT specification. Interface-level compatibility is specified using bus definitions and abstraction definitions. Ports which are not part of interfaces or ports which are not connected at the interface level must be connected at the port level, as described in [Making Port-Level Connections, on page 40](#).

The System Designer tool allows the following types of connection at the interface level: component interface to bus, interface to interface without a bus (direct connection), and interface to top level (hierarchical connection). See the following for details:

- [Connecting an Interface to a Bus, on page 35](#)
- [Directly Connecting Interfaces, on page 37](#)
- [Making Hierarchical Connections at the Interface Level, on page 39](#)
- [Removing Interface-Level Connections, on page 40](#)

Connecting an Interface to a Bus

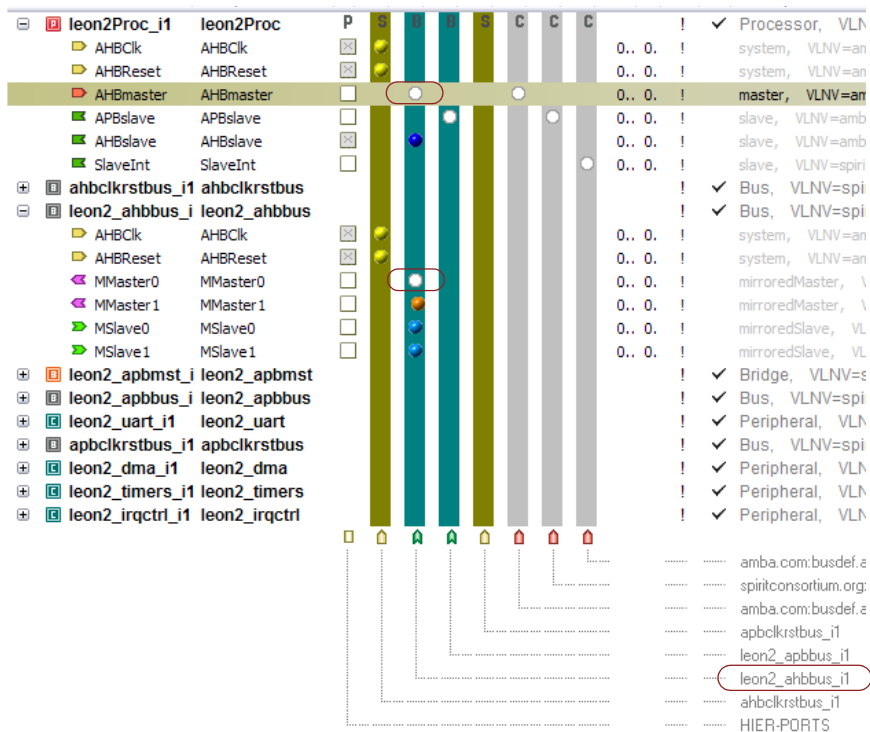
This procedure shows you how to connect an instance interface to a compatible interface on a bus component. You do this from the Design view.

1. In the Design view, go to the bus (dark teal bar) for the interface to be connected.

2. Click the white bubble on the bus.

Bubbles represent valid connection points that conform to the IP-XACT rules. A white bubble indicates that it is unconnected. When you click a white bubble, the tool connects the interface to the bus and changes its color to indicate that a connection was made. The colors vary according to the type of connection. See [Bus Connections, on page 71](#) for a description of how to interpret the symbols and colors in the Design view.

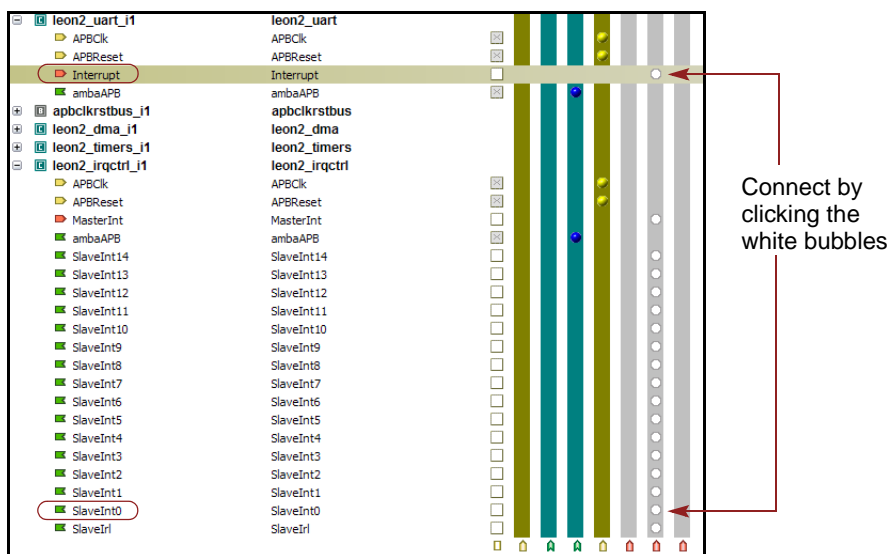
The following shows an unconnected white bubble for the AHBmaster interface on the `leon2_ahbbus_i1` bus, and the corresponding white bubble on the bus component, to which it will be connected after clicking.



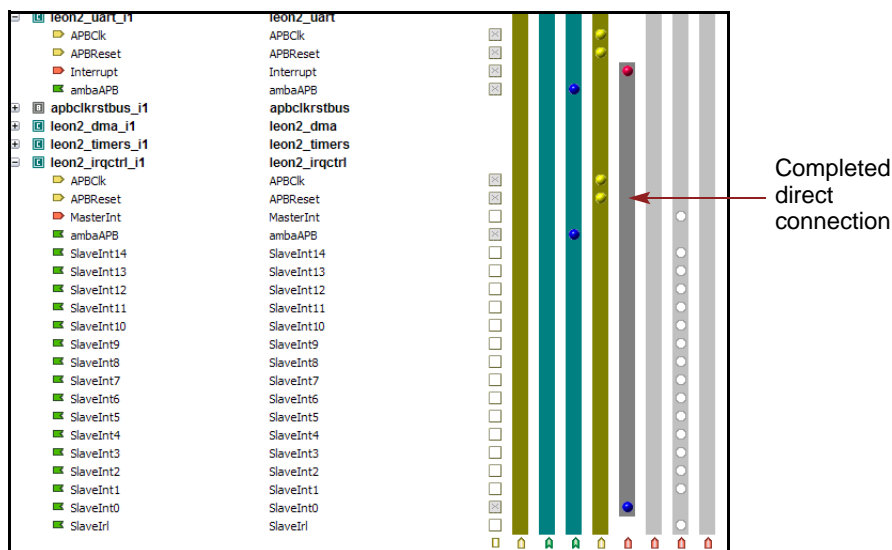
Directly Connecting Interfaces

Direct connections do not use a bus, but are made directly from the interface of one component to a compatible interface on another component. You can make interface-level direct connections from the Design view or the Interface Connections view, as described below. For information about direct connections between ports see [Making Port-to-Port Connections, on page 41](#).

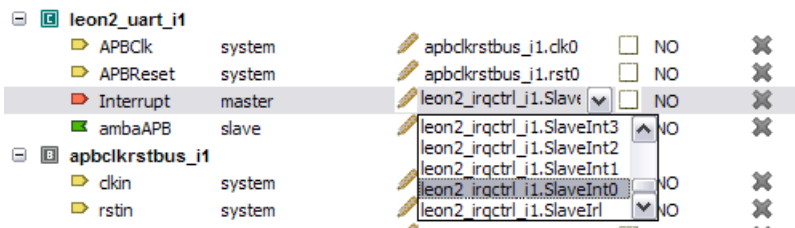
1. To make an interface-level direct connection between ports from the Design view, do the following:
 - If the grey vertical design connection bars are not displayed in the Design view, right-click in the Design view and enable Show Direct Connections to view the connection bars.
 - In the grey bar, locate and click the white bubble for one of the unconnected ports you want to connect.
 - Locate and click the white bubble for the second port, to which you want to connect.



The bubbles change color to indicate connection. The grey bar changes to a darker grey and now only runs between the two ports, to indicate the direct connection.



2. To make an interface-level direct connection between interfaces from the Interface Connections view, do the following:
 - In the Interface Connections view, go to the Connecting Interface column for one of the ports you want to connect.
 - Click in the column and select a compatible port to connect to from the pull-down menu.

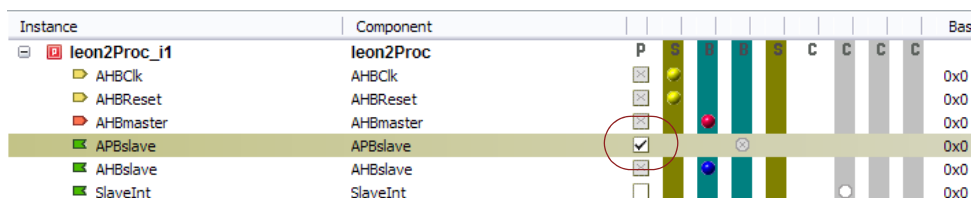


The tool directly connects the two interfaces.

Making Hierarchical Connections at the Interface Level

A hierarchical interface connection specifies that all ports of the interface be exported to the top level of the design. You can define hierarchical connections for interfaces in the Design view or the Interface Connections view. For information about defining hierarchical connections at the port level, see [Making Hierarchical Port Connections, on page 42](#).

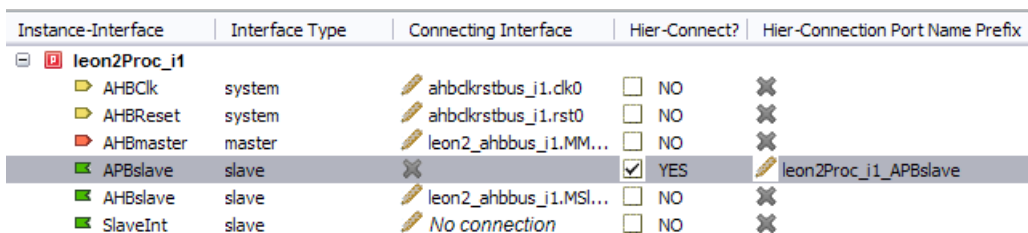
1. To make an interface-level direct connection between ports from the Design view, check the box in the P column for the port.



Instance	Component	P	A	S	C	C	C	C	Bas
leon2Proc_i1	leon2Proc								
AHBClk	AHBClk								0x0
AHBReset	AHBReset								0x0
AHBmaster	AHBmaster								0x0
APBslave	APBslave	<input checked="" type="checkbox"/>							0x0
AHBslave	AHBslave								0x0
SlaveInt	SlaveInt								0x0

The tool exports the port to the top level using the default `<instance_name>_<port_name>` as the port name at the top level.

2. To specify a hierarchical connection from the Interface Connections view, do the following:
 - In the Interface Connections view, go to the Hier-Connect column for the port, and click the box so that it reads Yes. A default prefix (`<instance_name>_<port_name>`) to be used for the port at the top level appears in Hier-connection Port Name Prefix.
 - If you like, edit the prefix and change the name.



Instance-Interface	Interface Type	Connecting Interface	Hier-Connect?	Hier-Connection Port Name Prefix
leon2Proc_i1				
AHBClk	system	ahbclkstbus_j1.clk0	<input type="checkbox"/> NO	✗
AHBReset	system	ahbclkstbus_j1.rst0	<input type="checkbox"/> NO	✗
AHBmaster	master	leon2_ahbbus_j1.MM...	<input type="checkbox"/> NO	✗
APBslave	slave	✗	<input checked="" type="checkbox"/> YES	leon2Proc_i1_APBslave
AHBslave	slave	leon2_ahbbus_j1.MSL...	<input type="checkbox"/> NO	✗
SlaveInt	slave	No connection	<input type="checkbox"/> NO	✗

Removing Interface-Level Connections

The following shows you how to remove interface-level connections you specified in the Design view and the Interface Connections view. For information about removing port-level connections in the Port Connections view, see [Removing Port-Level Connections, on page 44](#).

Design view	<ul style="list-style-type: none">• Port-to-port connections: Click one of the colored connection bubbles• Hierarchical connections: Uncheck the box in the P column.
Interface Connections view	<ul style="list-style-type: none">• Port-to-port connections: Click in the Connecting Interface column, and select No connection from the pull-down menu.• Hierarchical connections: Uncheck the box in the Hier-Connect column.

Making Port-Level Connections

Port-level connections are for ports that are not part of interfaces or ports which are not connected at the interface level. For information about interface-level connections, see [Making Interface-Level Connections, on page 35](#).

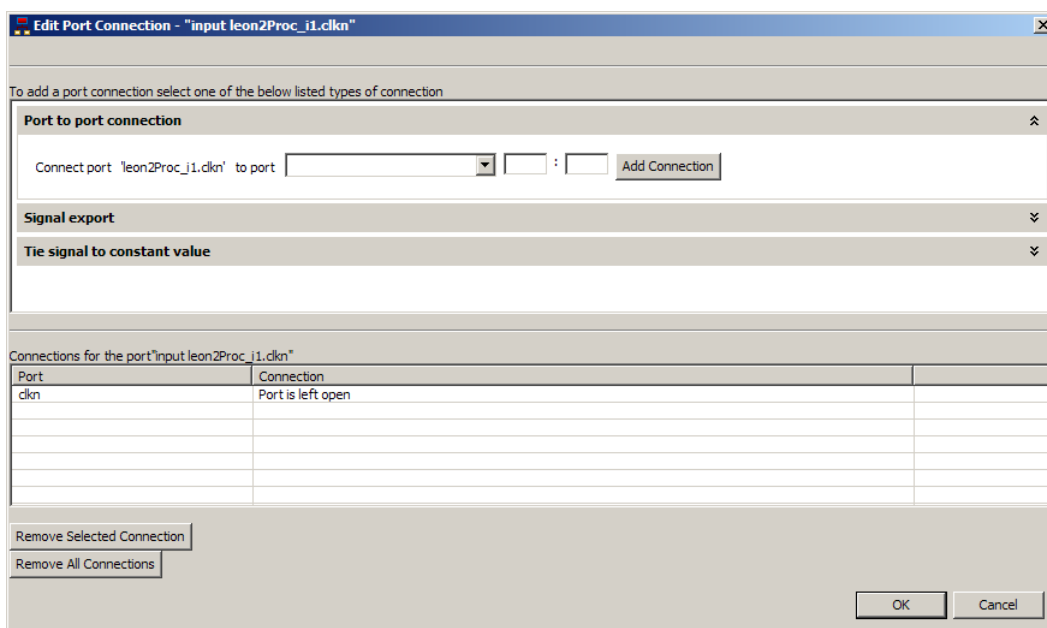
The System Designer tool lets you specify these port-level connections: port to port, port to top level (hierarchical connection), and port tied to a constant value. You specify all port-level connections in the Port Interface view. See the following for details:

- [Making Port-to-Port Connections, on page 41](#)
- [Making Hierarchical Port Connections, on page 42](#)
- [Tying Ports to a Constant Value, on page 43](#)
- [Removing Port-Level Connections, on page 44](#)

Making Port-to-Port Connections

The following procedure shows you how to make a direct connection from one port to a compatible port on another instance. For interface-level direct connections, see [Directly Connecting Interfaces, on page 37](#).

1. In the Port Connections view, click the icon in the Edit column for one of the ports you want to connect. This opens the Edit Port Connection dialog box.
2. Expand Port to Port connection in the dialog box, and select a compatible port from the pull-down menu.



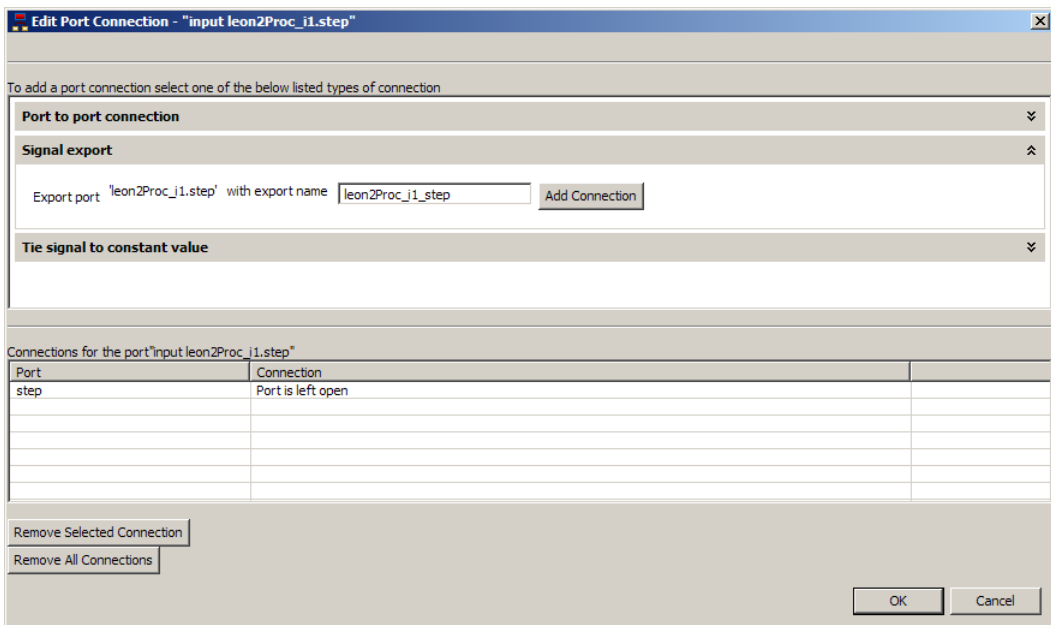
3. Optionally, specify a port range.
4. Click Add Connection, and click OK.
5. To remove a port-level connection, see [Removing Port-Level Connections, on page 44](#).

See [Connecting HAPS Peripherals, on page 56](#) for additional information for connections to HAPS peripherals.

Making Hierarchical Port Connections

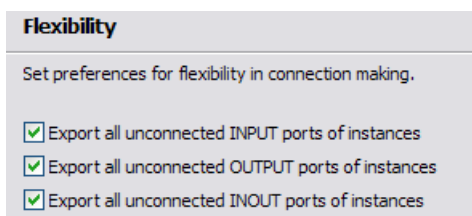
A port-level hierarchical connection exports the specified port to the top level of the design. For interface-level direct connections, see [Directly Connecting Interfaces, on page 37](#). You can make port-level hierarchical connections on an individual basis from the Port Connections view, or automatically export all the ports by setting a preference. The following describes the procedures.

1. To specify a hierarchical connection from the Port Connections view, do the following:
 - In the Port Connections view, click the icon in the Edit column for the port you want to connect. This opens the Edit Port Connection dialog box.
 - Click Signal export in the dialog box. You see a default name for the signal at the top level: `<instance_name>_<port_name>`.
 - If you want to specify a different name to be used for the port at the top level, type a new name in the field.



- Click Add Connection.
- Click OK.

2. To automatically connect the unconnected ports of an instance to the top-level design, do the following:
 - Select Action->Preferences.
 - In the Preferences dialog box, click Connectivity->Flexibility.
 - To connect all the unconnected input ports of the instance to the top-level design ports, enable Export all unconnected INPUT ports of instances.
 - To connect all the unconnected output ports of the instance to the top-level design ports, enable Export all unconnected OUTPUT ports of instances.
 - To connect all the unconnected in/out ports of the instance to the top-level design ports, enable Export all unconnected INOUT ports of instances.

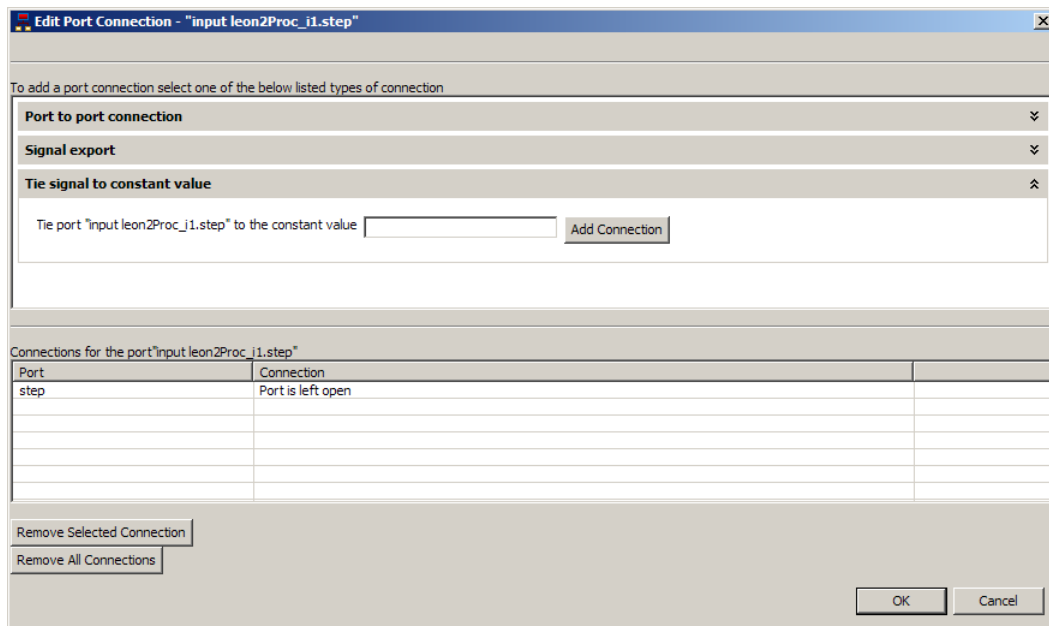


3. To remove a port-level connection, see [Removing Port-Level Connections, on page 44](#).

Tying Ports to a Constant Value

To assign a constant value to a port signal, you must use the Port Connections view as described below:

1. In the Port Connections view, click the Edit column icon for the port.
The Edit Port Connection dialog box opens.
2. In the dialog box, click Tie signal to constant value.
3. Specify a value for the constant in the field, and click Add Connection.



4. Click OK.
5. To remove a port-level connection, see [Removing Port-Level Connections, on page 44](#).

Removing Port-Level Connections

The following shows you how to remove previously-specified connections in the Port Connections view. For information about removing interface-level connections in the Design view or Interface Connections view, see [Removing Interface-Level Connections, on page 40](#).

1. In the Port Connections view, click the icon in the Edit column to open the Edit Port Connection dialog box.
2. Select the connection in the table.
3. Click Remove Selected Connection to remove the selected connection, or Remove All Connections to remove all connections.
4. Click OK.

Connecting and Configuring Automatically

The following procedures show you how to set up your design to automatically connect instances and configure the addresses. For information on how to manually connect and configure your design, see [Configuring IP Cores, on page 33](#).

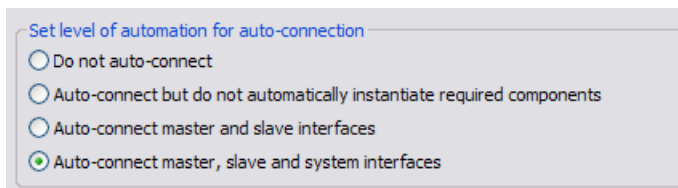
See the following for information on automatically configuring and connecting the design:

- [Setting Auto-Connection Preferences, on page 45](#)
- [Automatically Connecting Component Instances, on page 46](#)
- [Automatically Configuring Addresses, on page 47](#)

Setting Auto-Connection Preferences

You can set preferences that determine whether the tool automatically makes the connections.

1. Select Actions -> Preferences.
2. In the Preferences dialog box, expand Connectivity and click Flexibility.
3. Set a preference for automatic connections in the Set level of automation for auto-connection section:



- Select Do not auto-connect if you do not want to make any automatic connections.
- To make connections automatically but not automatically instantiate any required components from the library, select Auto-connect but do not automatically instantiate required components.

- To automatically connect and automatically instantiate any required master/slave interfaces from the component library, select Auto-connect master and slave interfaces.
- To automatically connect and automatically instantiate any required master/slave interfaces and system buses, select Auto-connect master, slave and system interfaces.

Automatically Connecting Component Instances

1. Select Actions -> Auto Connect Interfaces.

This displays a the message:

Auto-connect may connect all unconnected interfaces ...

2. Click Yes to automatically connect compatible instance interfaces with the bus.
3. You can check connectivity of compatible interfaces to buses by checking the Description column in the Design view.

Automatically Configuring Addresses

1. Select Actions -> Preferences, expand Configuration and click Flexibility.

The screenshot shows the 'Flexibility' dialog box with the title 'Flexibility'. The main instruction is 'Set preferences for flexibility in configuration.' Below this are several options:

- ☐ Allow instantiation of non-usable components
- ☐ Allow guessing if a model-view is synthesizable HDL view
- A group box titled 'Allow editing of parameters with resolve = <non-user> property' contains four radio buttons:
 - ☐ Allow editing only resolve=user
 - ☐ Allow editing of resolve=immediate
 - ☐ Allow editing of resolve=dependent
 - ☒ Allow editing of resolve=generated
- ☐ Allow changing of 'instance' HDL value if no matching model-view found
- ☐ Allow changing of 'project' HDL value if no matching model-view found
- A group box titled 'Tolerance level while choosing a model-view with mismatching info' contains four radio buttons:
 - ☐ Select model view only if vendor, technology, part, package and speed grade match
 - ☒ Select model view if vendor and technology match
 - ☐ Select model view if vendor matches
 - ☐ Select model view even if no elements match
- A text field labeled 'Minimum address-block size for auto-addressing' contains the value '4096'.

At the bottom right are two buttons: 'Restore Defaults' and 'Apply'.

2. Set a threshold value for the minimum address block size in Minimum address block size for auto-addressing and click OK.
3. Select Actions -> Auto Configure Addresses and click Yes to confirm and accept this message:

Auto-configure may change all unlocked address assignments ...

This automatically configures the addresses.

4. Verify the addresses in the Memory Map view.

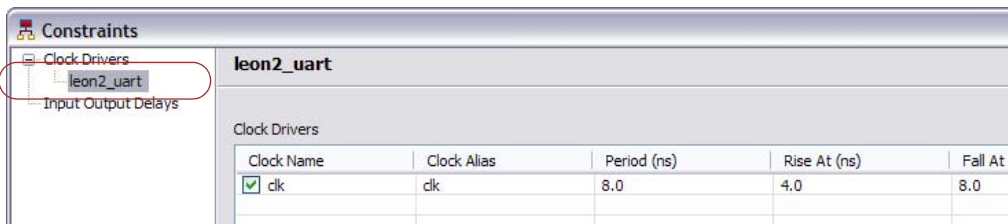
Checking Constraints

The System Designer tool maps the IP-XACT constraints specified for the design to constraints in the .sdc format that the synthesis tools to use. You can check the IP-XACT constraints that will be translated by following these steps. For details about how the tool handles the translation, see [Generating an .sdc File for Synthesis, on page 94](#).

1. Select Action->Open Design Constraints.

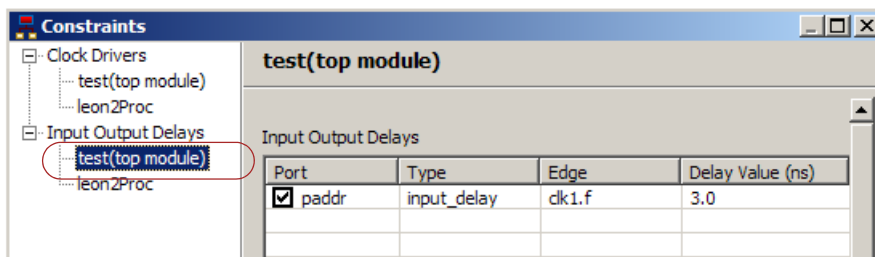
The Constraints dialog box opens. If you get a message alert instead, you are using an older version of SPIRIT IP-XACT. Switch to version 1.4 and proceed.

2. To check the clock driver constraints, do the following:
 - Select a component from the ones listed under Clock Drivers on the left.



- View the constraints defined for that component. These constraints are mapped to define_clock constraints in the .sdc file.

3. To check the input/output delay constraints, do the following:
 - Select a component from the ones listed under Input Output Delays on the left.



- View the constraints defined for that component. These constraints are mapped to define_input_delay and define_output_delay constraints in the .sdc file.

Using Tcl Scripts

1. Create a Tcl script using the commands described in [Chapter 4, Tcl Commands](#).
2. To run the script, select Project->Run Tcl Script.
3. Select the Tcl script to run. The script runs automatically.

Generating Output Files for Synthesis

After you have finished your design, you can generate output files to be used for synthesis. Do the following:

1. After you have created your design in the System Designer UI, select Actions-> Generate Files or click the Generate Files button.

The tool generates output files based on the design you created in System Designer.

2. Check that the following output files were created:
 - `synthesis.slib` in the `synplify` folder in the project directory
This is the Synplify Pro/Synplify Premier sub-library file. When you run synthesis, this file gets sourced in the `synthesis.prj` file.
 - `.v` or `.vhd` files in the `verilog` or `vhdl` folders in the project directory
These are the HDL files for the design. The file format is determined by the HDL setting you selected (under Settings in the Design Configuration window).
 - `.sdc` files in the project directory
These are the top-level and block-level constraint files for the design. The System Designer tool maps the IP-XACT constraints to `define_clock`, `define_input_delay`, and `define_output_delay` constraints, as described in [Generating an .sdc File for Synthesis, on page 94](#).
 - `vcs.ksh` file in the `vcs` folder
Use this file to run VCS simulation on the generated system. See [Verifying the Design with Simulation, on page 50](#).
 - `.do` file in the `modelsim` folder
Use this file to run Modelsim simulation on the generated system.

Verifying the Design with Simulation

You can verify your generated system design by running it through simulation. The following procedure describes how to simulate with the Synopsys VCS tool.

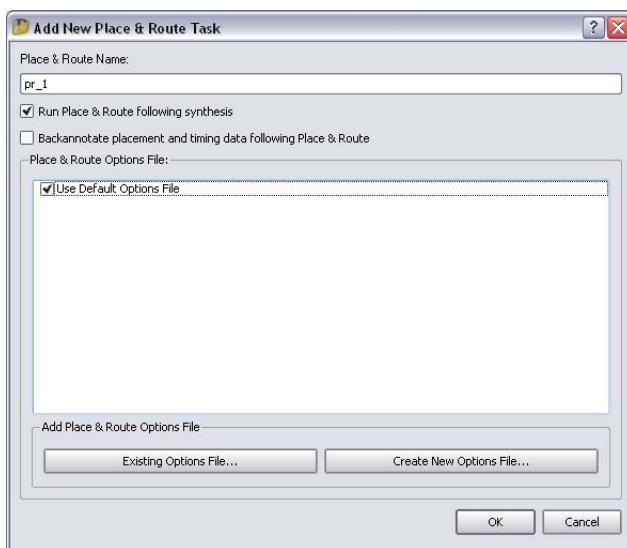
1. Generate your embedded system using System Designer.
The tool automatically creates script files for Modelsim and VCS.
2. To simulate with VCS, do the following:
 - Start the VCS software.
 - Use the `vcs.ksh` file generated by System Designer to run simulation. This file is in the `vcs` directory.
3. To simulate with Modelsim, do the following:
 - Start the Modelsim software.
 - Use the `.do` file generated by System Designer to run simulation. This file is in the `modelsim` directory.

Running Synthesis and P&R

The following steps show you how to synthesize, and then automatically place and route the embedded logic. Alternatively, you can choose to run P&R separately after synthesis.

1. Set up your target P&R tool, so it can run from the synthesis UI. See the FPGA documentation for details.
2. Open the System Designer synthesis project.
The empty synthesis project you created when you first started now contains the design you created in the System Designer tool.
3. Add a place-and-route implementation for this project.
 - Either right-click the current implementation and select Add P&R Job, or just click the Add P&R implementation button in the project UI.

- In the dialog box that opens, enable Run Place & Route After Synthesis. This option runs P&R automatically after synthesis is complete.



- Optionally, specify another name for the implementation or an options file for placement and routing.
 - Click OK. This creates the place and route implementation for the System Designer design, and specifies an automatic P&R run after synthesis.
4. Set other synthesis options and constraints as usual.
 5. Click Run.

The tool automatically synthesizes the design, including the embedded system. It then runs the place-and-route tool.

Using System Designer with a HAPS Board

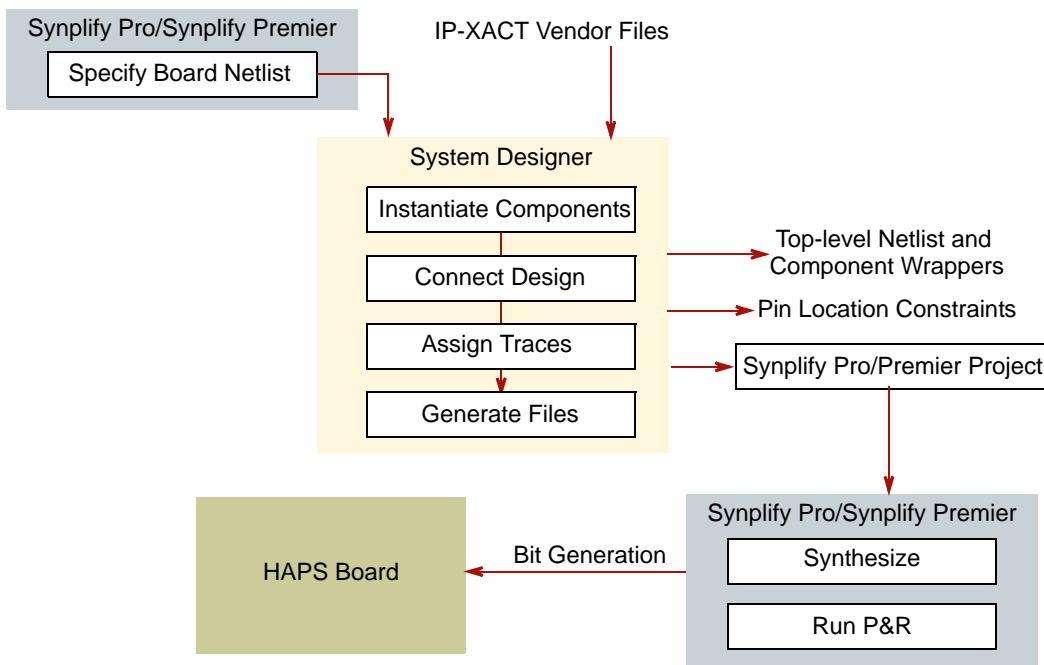
The System Designer tool lets you load a design created with System Designer on to a supported HAPS board so that you can verify it. This also allows you to include HAPS on-board peripherals and daughter cards in the design.

See the following for more information:

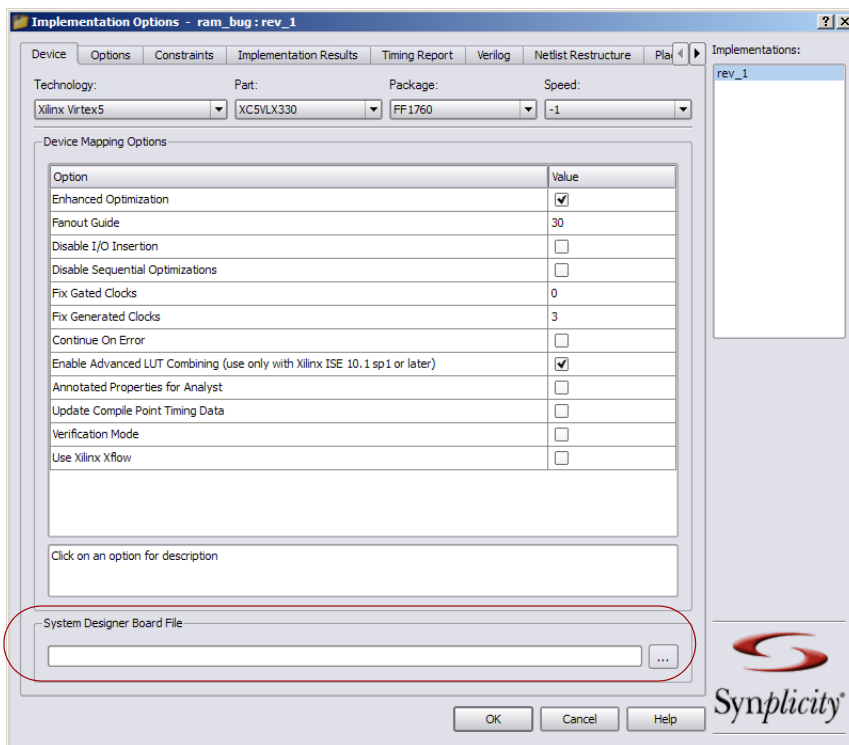
- [The System Designer-HAPS Board Design Flow, on page 52](#)
- [Connecting HAPS Peripherals, on page 56](#)
- [Assigning Traces for Designs with HAPS Boards, on page 57](#)

The System Designer-HAPS Board Design Flow

The following figure summarizes the flow. See the procedure for detailed information about each step.

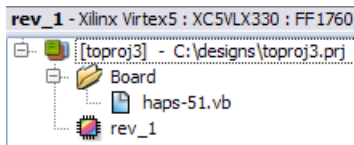


1. Open the Synplify Pro or Synplify Premier synthesis tool, and create a new project.
2. Specify the board file.
 - Click Implementation Options. Specify a .vb HAPS board file on the Device tab of the Implementation Options dialog box. Select one of the supported single-FPGA boards: haps-31, haps-51, or haps-51t. You cannot currently use this flow with multi-FPGA boards.



- If you want to use a daughter card, open the board definition file, and manually instantiate the daughter card in Verilog. You can use the following supported daughter cards: DDR2_1x2, DDR_1X1, FLASH_1x1, LCD_1x1, SDRAM_1x1, or SRAM_1x1.
- In the Implementation Options dialog box, set the Technology, Part, Package, and Speed options to match those on the board you selected.
- Click OK.

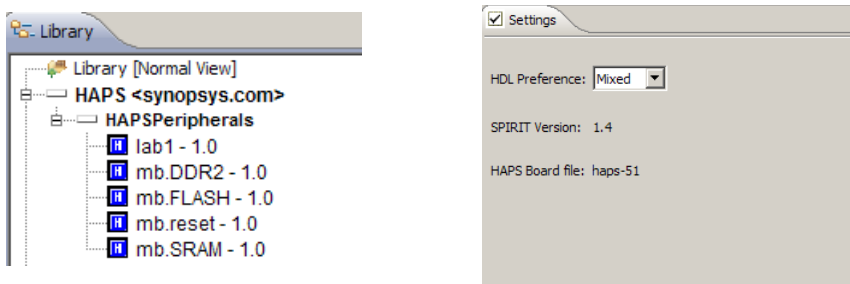
The Project view displays the board file, as shown below. The board file is only used by the System Designer tool, and is not used for synthesis.



If there is a mismatch between the part settings in the Implementation Options dialog box and the board settings, you get an error message. If you get an error message, go back to the Implementation Options dialog box and fix the mismatch before continuing.

3. Start the System Designer tool and set up the HAPS board project.

- Select Import->Launch System Designer or click the System Designer icon to start the System Designer tool.
- To use the board in an existing System Designer project, open an existing design. To use the board in a new System Designer project, open a new project in System Designer. In either case, the System Designer Library view displays the HAPS board and the peripherals, and the Settings panel lists the board you selected.



- Instantiate HAPS peripherals from the Library view. You can only instantiate a peripheral once in the design. Once instantiated, the peripheral is grayed out to indicate that it is not available.
 - If you are working with a new System Designer project instead of an existing one, you must now load an IP-XACT library, instantiate components, and connect them, as you would do normally.
4. Connect the HAPS board.
- Use the Port Connections view to connect the top-level ports of loaded IPs with HAPS peripherals, as described in [Connecting HAPS Peripherals, on page 56](#).
 - If you have interface-level definitions for the peripherals, you can also make interface-level connections from the Interface Connections view.
 - Click the Trace Assignments tab and assign traces, as described in [Assigning Traces for Designs with HAPS Boards, on page 57](#). The tool automatically assigns traces for daughter peripherals used in the design.

5. Generate the System Designer design by clicking Generate Files.

The System Designer tool generates the component wrappers, top-level file and pin location constraints for the design, and updates the Synplify Pro/Premier project file.

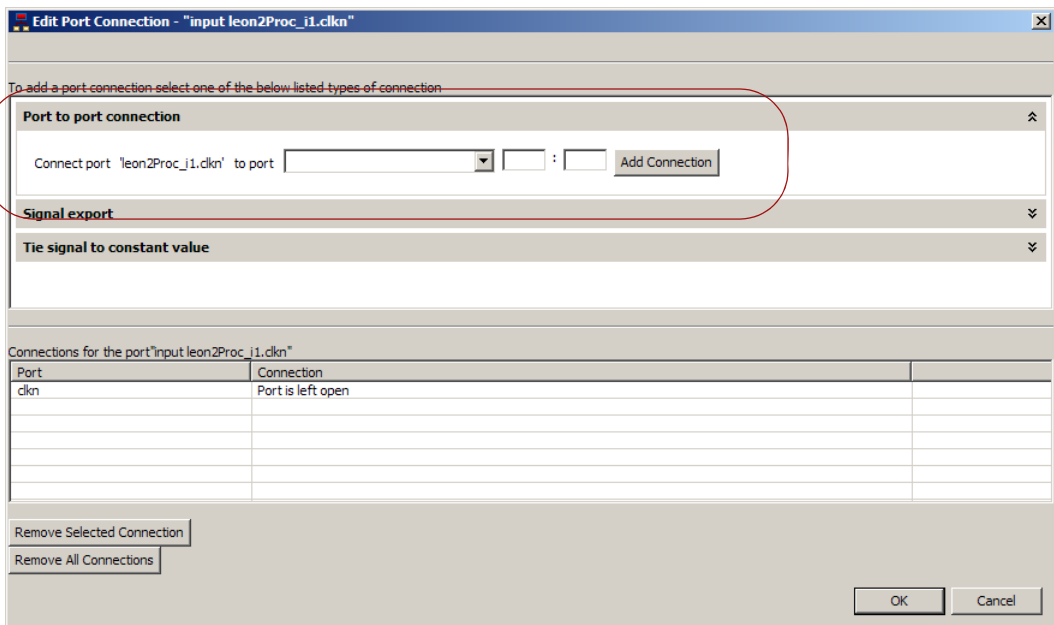
6. Return to the Synplify Pro/Premier interface and complete the flow.
 - Run synthesis, and place and route the design.
 - Generate a bitgen file and load the design on the HAPS board.
 - Verify the design.

Connecting HAPS Peripherals

You connect the HAPS board by making top-level connections between the HAPS peripherals and the IPs in your design. Do the following:

1. To connect top-level ports, use the procedure described in [Making Port-to-Port Connections, on page 41](#).

In general, you can connect an output port to multiple input ports, but for HAPS peripherals, you cannot share port connections.



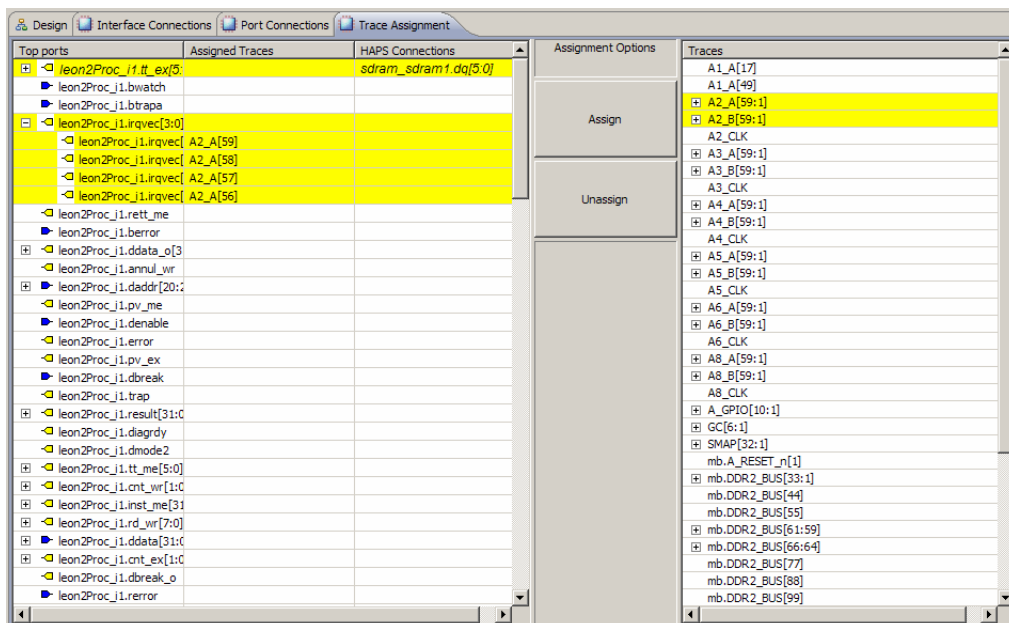
2. To make interface-level connections for HAPS peripherals, do the following:
 - Check that the HAPS peripherals have interface-level connections defined for them.
 - Click the Interface Connections tab to open the Interface Connections view.
 - Click in the Connecting Interface column and select a compatible interface from the pull-down list.

Assigning Traces for Designs with HAPS Boards

The following procedure shows you how to assign top-level logical nets to the physical traces on the HAPS board.

1. In the System Designer tool, select the Trace Assignments tab.

This opens the Trace Assignments view, which lists the top-level ports and available traces. It only displays the HAPS connectors that are connected to the FPGA or peripheral. See [Trace Assignment View, on page 78](#) for a description of this view.
2. To assign a trace to a net, do the following:
 - Select a top-level port in the Top Ports column.
 - Select a physical trace from the list in the Traces column on the right. Check the HAPS board documentation to make sure that you are making the right selection. The System Designer tool does not check compatibility. See [Trace Assignment Rules, on page 58](#).
 - Click the Assign button. The tool assigns the port to the selected trace as specified, without checking. Assigned traces have a yellow background. The assigned trace is also displayed in the Assigned Traces column.



When you generate the System Designer design, the tool generates pin location constraints based on the trace assignments. It saves the trace assignments in a file called `<design>.xml`.

3. To unassign a trace, do the following:
 - Select a top-level port in the Top Ports column. You cannot edit the assignments of ports connected to HAPS peripherals.
 - Select the assigned physical trace from the list in the Traces column on the right.
 - Click the Unassign button. The tool removes the trace assignment.

Trace Assignment Rules

The following are some basic rules for trace assignment. Consult the HAPS documentation for information about the traces.

- A port cannot have multiple trace assignments. If you assign a trace to a port, you cannot assign another trace to the same port.
- A trace cannot be assigned to multiple ports. Once you have assigned a trace, you cannot assign it to another port.

- If you select a mult-bit port or trace, the tool assigns all the individual bits.
- If you select an unequal number of ports or traces for assignment (more ports than traces, or more traces than ports), the tool assigns the traces but issues a warning message.
- You cannot assign traces for design ports that are connected to the ports of a HAPS peripheral, because these traces are automatically assigned and cannot be modified from this view. These traces are in italics, with a yellow background.

CHAPTER 3

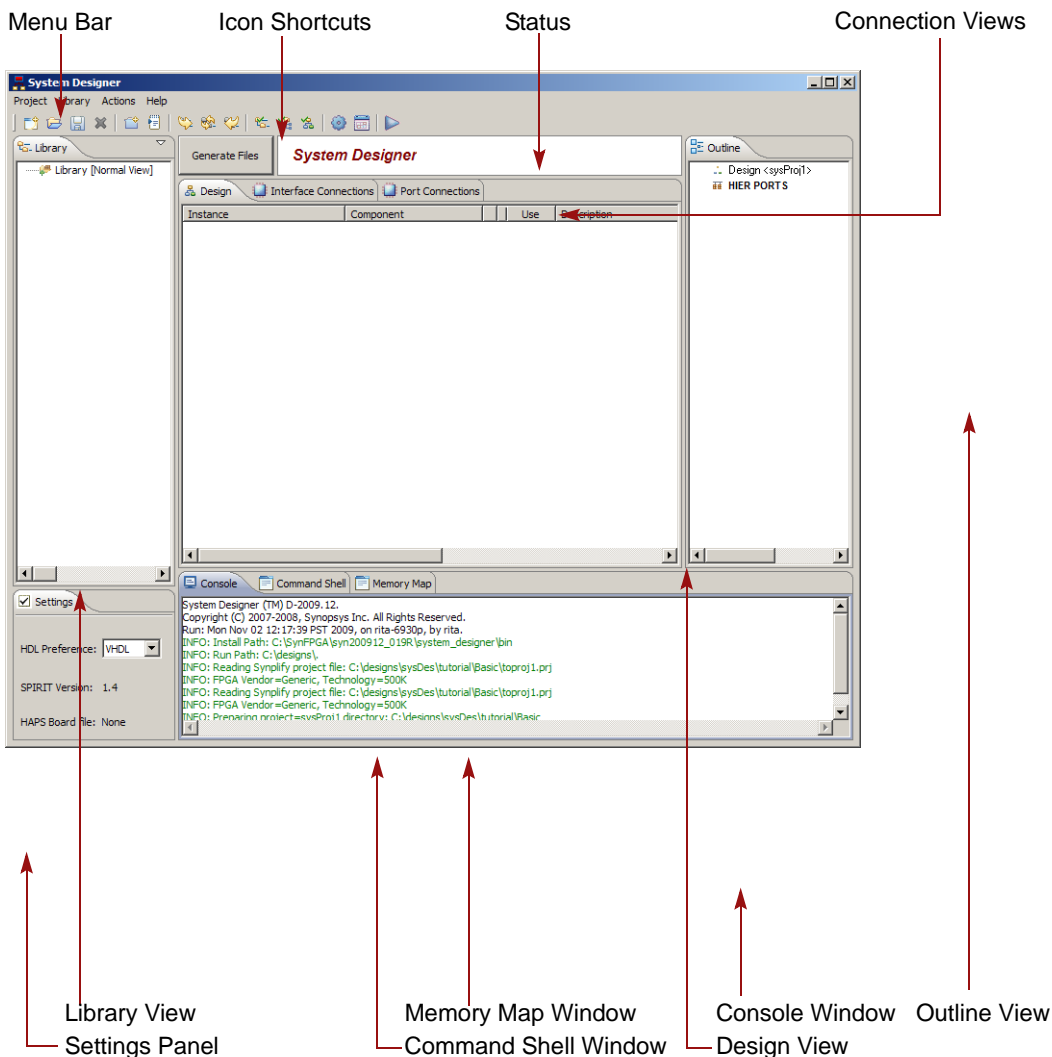
Menus and Commands

The System Designer tool relies on a graphical interface, and includes many commands. The following describe the interface and the commands by menu:

- [The System Designer User Interface, on page 62](#)
- [Display in the Design View, on page 67](#)
- [Interface Connections View, on page 72](#)
- [Port Connections View, on page 74](#)
- [Trace Assignment View, on page 78](#)
- [Project Menu Commands, on page 81](#)
- [Library Menu Commands, on page 84](#)
- [Actions Menu Commands, on page 86](#)
- [Help Menu Commands, on page 112](#)
- [Context-Sensitive Command Menus, on page 112](#)

The System Designer User Interface

The System Designer interface consists of a main project window, command menus, and other windows. The following figure shows the System Designer interface when it first opens, without a design loaded:

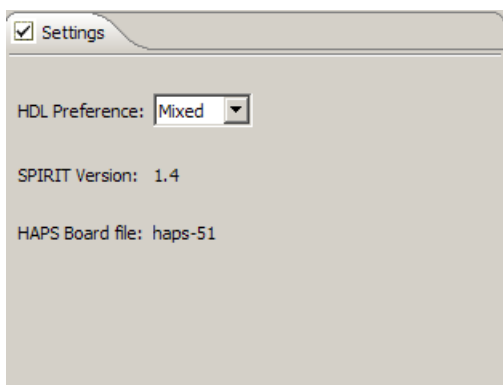


Design Window	Displays the design. This is the main window. When you load components into a design, it displays the components and connections. For a description of the objects and symbols used when you load components, see Display in the Design View, on page 67 for details. For a description of the popup commands available in this window, see Popup Commands in the Design View, on page 113 .
Menu Bar	<p>Lists the command menus. See the following for details of the commands available:</p> <ul style="list-style-type: none">• Project Menu Commands, on page 81• Library Menu Commands, on page 84• Actions Menu Commands, on page 86• Help Menu Commands, on page 112
Icon Shortcuts	Shortcuts to commonly-used commands that are usually accessed from the command menus. When you move the mouse over an icon, the tool displays the name of the command. The icons are also displayed in the menus, next to the corresponding command.
Status	Displays the current status of the tool based on the design context.
Interface Connections View	Displays the interface-level connections, and lets you make these connections interactively in this view. See Interface Connections View, on page 72 for a description.
Port Connections View	Displays the port-level connections. It displays the ports that are not interface-level ports, and lets you make the port-level connections interactively in this view. See Port Connections View, on page 74 for a description.
Library View	Displays the components in the libraries you loaded. See Display in the Design View, on page 67 for descriptions of the symbols used. The interface symbols are standard IP-XACT icons, but the other symbols are tool-specific.
Outline View	<p>Displays the outline, using standard IP-XACT symbols for interfaces. The display varies slightly, depending on what you select:</p> <ul style="list-style-type: none">• If you select a component in the Library view, the Outline view displays the IP-XACT description of the selected component.• If you select an interface or item in the Design view, the Outline view displays the interface connections in the design.

Trace Assignments View	The tool only displays this view if you specified a HAPS board file in the Synplify Pro or Synplify Premier project. It lets you specify physical traces for the ports in the design. See Trace Assignment View, on page 78 for a description.
Settings Panel	Lets you set some global options for your design. See Settings Panel, on page 65 for a description of the options.
Console Window	<p>Displays error, warning, and informational messages. Different colors are used:</p> <ul style="list-style-type: none">• Green text is for informational messages (INFO)• Black text is for configuration-related messages (CONFIG)• Blue text is for warnings• Red text is for errors <p>See Popup Commands in the Console and Command Shell Windows, on page 113 for a description of the popup commands available from this window.</p>
Command Shell Window	<p>Lets you type commands at the command line. Almost all the GUI commands have a command-line equivalent.</p> <p>See Popup Commands in the Console and Command Shell Windows, on page 113 for a description of the popup commands available from this window.</p>
Memory Map Window	The Memory Map window summarizes how components are mapped to memories. See Memory Map Window, on page 65 for a description of the mapping, and Popup Commands in the Memory Map Window, on page 114 for a description of the popup commands available in this window.

Settings Panel

The Settings panel lets you set the global language format for your design. It also lists some design settings.



HDL Preference	Specifies the format for the source files. You can set it to Verilog, VHDL, or Mixed.
SPIRIT Version	Lists the SPIRIT version for the library you selected. You cannot directly edit this field, but you can select another library with the Library->Load Library command.
HAPS Board File	Lists the HAPS board file you specified in the Synplify Pro or Premier project. You cannot directly edit this setting, but you can return to the synthesis tool and specify another file.

Memory Map Window

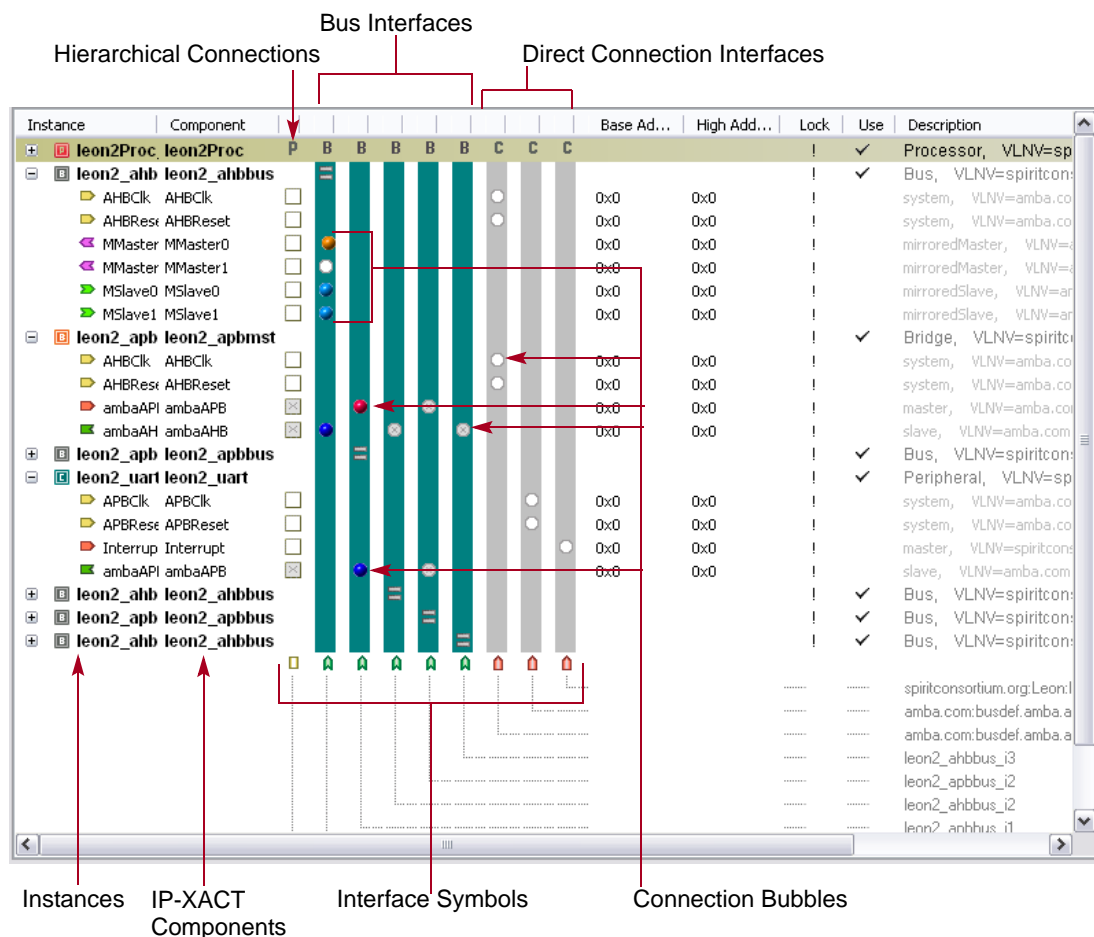
The Memory Map window summarizes how addresses are mapped to memory maps. When you map the address space of a master interface to the memory map of one or more slave components, the mapping information is displayed in this window. For details of different types of memory mapping, see the IP-XACT specifications.

You cannot edit the memory maps of master interfaces, but you can edit the memory maps of slave interfaces in this window.

Instance Name	Name of the instance
Base Address	Lists the base address for the memory. The base address represents the start address of the address range for the interface.
High Address	Lists the high address for the memory. The high address represents the end address of the address range for the interface.
Lock	Indicates whether the base address and the high address for the interface are locked. When an interface is locked, it cannot be addressed automatically.
Description	<p>Provides a brief description of the interface, including the following information:</p> <ul style="list-style-type: none">• Interface type• VLNV of the interface• Connection status <p>If a connection is required, the tool can make the connection. If a connection is explicit, you must explicitly specify the connection. The tool cannot automatically make the connection.</p>

Display in the Design View

When you load components in the Design view, the tool displays the instantiated components and their interface connections.



Reading from left to right, you see the following:

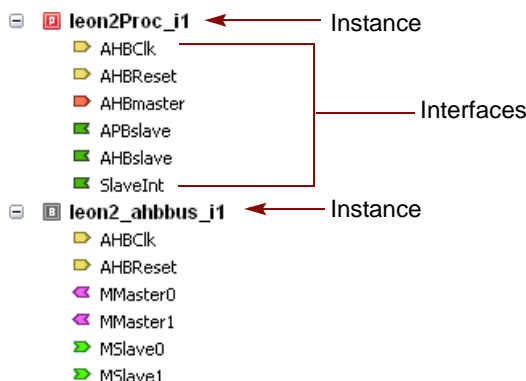
- The first two columns list the components and interfaces in a tree format.
- For hierarchical connections, there is a column with check boxes.
- Bus interfaces are represented by vertical bars.

- Interface connections are represented by bubbles on the vertical bars.
- Additional columns that you can choose to display or hide by setting your preferences.

The following table describes each of these more fully.

Instance	Lists the name of the component instance in the design. If you rename the instance, this column displays the name you set. The tool automatically appends an <i>_i<sequential_number></i> suffix to the end of the name to create a unique name for the instance. For example, <i>leon2_ahbbus_i1</i> , or <i>leon2_ahbbus_i2</i> . See Instance and Component Display, on page 69 for detailed descriptions of the icons.
Component	Lists the original component name from the library. See Instance and Component Display, on page 69 for detailed descriptions of the icons.
Interface Connections	Shows the interface connections. The letter at the top of the column and the colors of the vertical bars indicate the kind of interface. The column with the checkboxes is for hierarchical connections. The columns with colored vertical bars represent various bus interfaces. See Interface Display, on page 70 for detailed descriptions.
Base Address	Displays the base address for the component. This value is read from IP-XACT component files. If there is no value defined, the default value is 0.
High Address	Displays the high address for the component. This value is read from IP-XACT component files. If there is no value defined, the default value is 0.
Lock	Indicates whether the base address and the high address for the interface are locked. When an interface is locked, it cannot be addressed automatically.
Use	Unused attribute.
Description	Contains a brief description of the component or interface, including the following information: <ul style="list-style-type: none"> • Type of component or interface • VLVN (IP-XACT definition of vendor, library, name, and version) • Connection status <p>If a connection is required, the tool can make the connection.</p> <p>If a connection is explicit, you must explicitly specify the connection. The tool cannot automatically make the connection.</p>

Instance and Component Display



The Instance and Component columns in the Design view display tree views, with the instances as top-level entries and their interfaces under it.

Various symbols are used for the components and the interfaces. The interface symbols are standard IP-XACT symbols, but the others are specific to System Designer. The following table describes them:

Icon Description

Components

	Bridge component
	Bus component
	Processor component
	Peripheral component
	Special component

Interfaces

	Master interface
	Mirrored master interface
	Slave interface
	Mirrored slave interface
	System interface
	Mirrored system interface

Interface Display

See the following for descriptions of symbols and conventions used to display interfaces and connections in the Design view.

- [Interface Symbols, on page 70](#)
- [Bus Connections, on page 71](#)
- [Hierarchical Connection Interface, on page 72](#)





Interface Symbols

There is a column for hierarchical connections, and one for each bus interface. There are identifying letters at the top of each column and symbols at the bottom of each column that identify the kind of interface:

Identifying Letters (Top of Column)

B	Bus interface
C	Direct connection interface
P	Hierarchical connection (see Hierarchical Connection Interface, on page 72)
S	System interface

Interface Symbols (Bottom of Column)

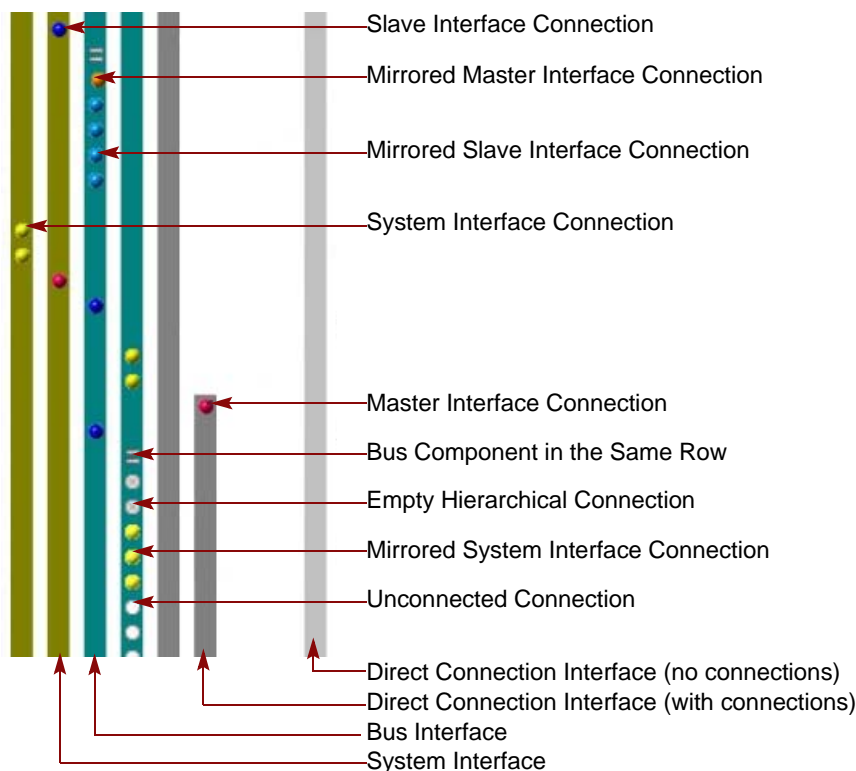
	Hierarchical connection (see Hierarchical Connection Interface, on page 72)
	Direct connection bus interface
	Mirrored connection bus interface
	System connection bus interface

Colored vertical bars are used to indicate bus interfaces. The colors indicate the type of bus interface. See [Bus Connections, on page 71](#) for an illustration of these interfaces.

Teal	Bus interface
Gold	System interface
Dark grey	Direct connection interface with connections
Light grey	Direct connection interface with no connections

Bus Connections

Connection points are indicated by bubbles on the vertical bars that represent the bus interfaces, as shown in the following figure. For a description of the hierarchical connection interface, see [Hierarchical Connection Interface](#), on page 72.



This table describes the different kinds of connections:

Red bubble	Master interface connection
Orange bubble	Mirrored master interface connection
Dark blue bubble	Slave interface connection
Light blue bubble	Mirrored slave interface connection
Dark yellow bubble	System interface connection

Bright yellow bubble	Mirrored system interface connection
White bubble	Empty connection
Grey bubble with X	Empty connection that cannot be connected because it is a hierarchical connection
Two bars (=)	Bus component in the same row

Hierarchical Connection Interface

The hierarchical connection interface consists of a column of check boxes. The letter P is at the top of the column. You make a hierarchical connection by checking the box. When the box is enabled, the software marks it as a hierarchical connection and takes the ports to the top level.

Empty check box	No hierarchical connection
Enabled check box	Hierarchical connection
Greyed out check box	Hierarchical connection is not available, because the interface has already been connected

Interface Connections View

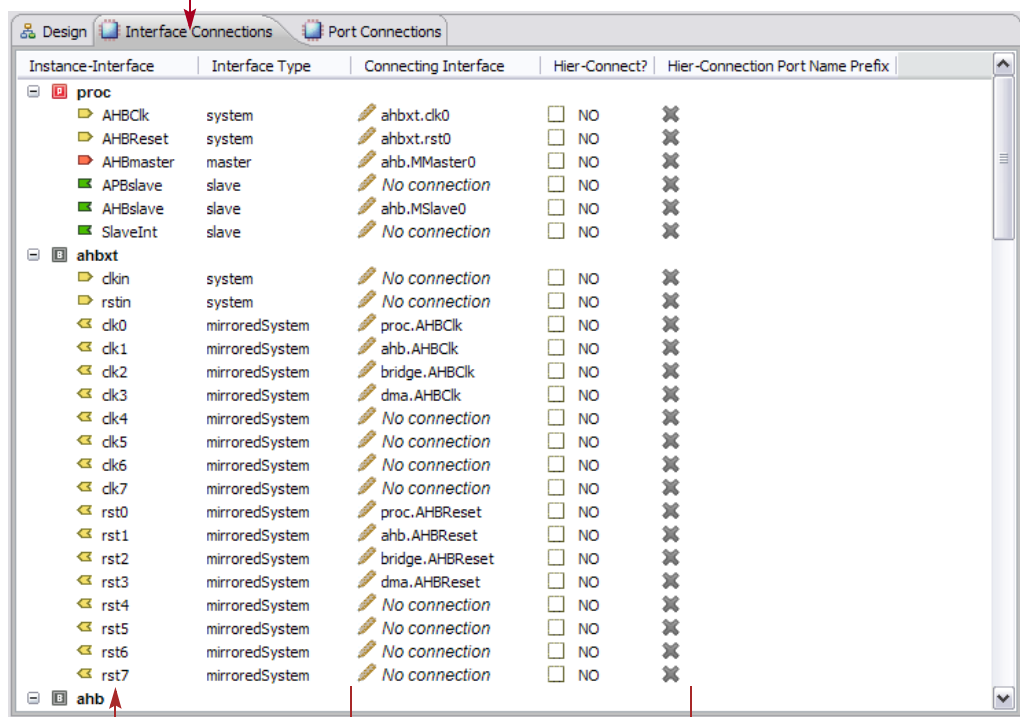
This view provides a convenient interface for making interface-level connections. This view lets you make two kinds of connections:

- Connections to compatible interfaces of other instances
- Hierarchical connections to the top level

You can also do this in the Design view, but the Interface Connections view additionally gives you a list of compatible connecting interfaces to select from and lets you specify port prefixes for hierarchical port connections that are exported to the top level.

To open this view, click the Interface Connections tab in the Design view.

Click to open the view



Components

Connection criteria

Instance-Interface

Lists the components in the design and their interfaces.

Interface Type

Displays the IP-XACT definition for the type of interface.

Connecting Interface

Displays the interface connection. No connection indicates that the interface is not connected. You can make a connection or edit one by clicking in this column and selecting from the pull-down menu of compatible interfaces for connection.

Hier-Connect



Lists whether an interface is a hierarchical connection to the top level. You can use this column to change the status or specify an interface as a hierarchical connection.

Hier-Connection Port Name Prefix

Displays the port name prefix that is used at the top level for hierarchical connections. You can edit the default prefix in this column. The default prefix is `<instance_name>.<interface_name>`.

Symbols in the Interface Connections View

The following describes the symbols in the Interface Connections view:

Component symbols	Same as the Design view. See Instance and Component Display, on page 69 for descriptions.
	Indicates that you can edit the description.
	Indicates a field that is not applicable, based on the type of connection.

Port Connections View

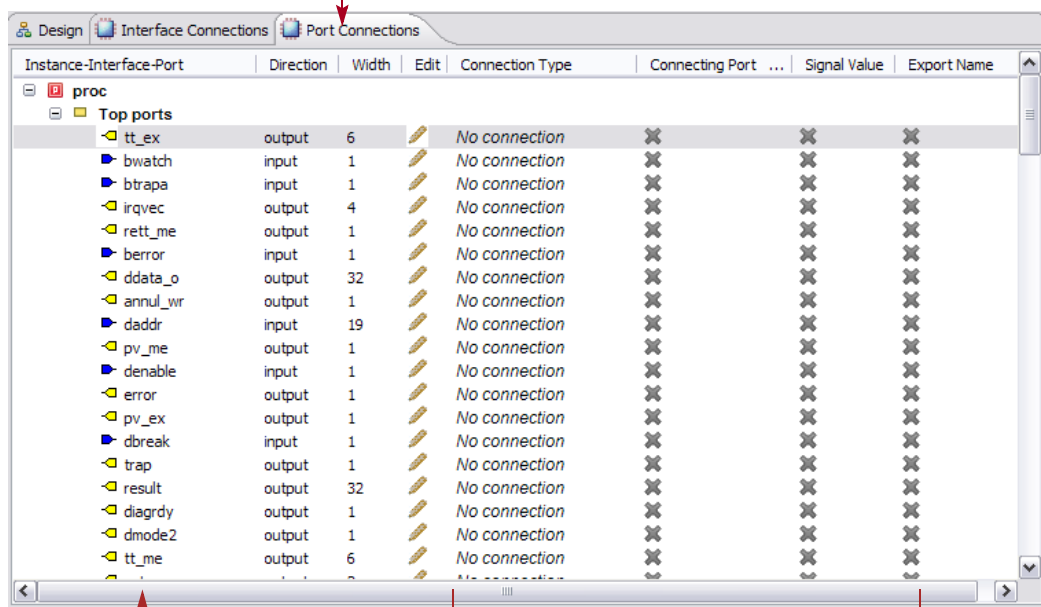
This view provides an interface for making port-level connections. This view includes ports that are part of interfaces as well as those that are not part of an interface. You can do the following in this view:

- Leave ports unconnected
- Connect ports to compatible ports on other instances
- Make hierarchical connections to the top level of the design
- Tie an input port to a constant value

See [Making Port-Level Connections, on page 40](#) for step-by-step procedures on accomplishing these tasks.


To open the Port Connections view, click the Port Connections tab in the Design view.

Click to open the view



Components

Connection criteria






Instance-Interface-Port	Lists the components in the design, their interfaces, and their ports that are not connected at the interface level
Direction	Displays the signal direction for the port.
Width	Displays the port width.
Edit	Displays the  icon to edit port connections. Clicking the icon opens the Edit Port Connection dialog box, where you can set the connections. See Edit Port Connection Command, on page 76 for a description.
Connection Type	Specifies how the port is connected. It can be one of the following: <ul style="list-style-type: none"> No connection is an unconnected port Port to port connection is a connection to a port on another instance Hier connection is a hierarchical connection to the top level Tied to constant value is an input port with a constant value

Connecting Port	Displays the connecting port (only for ports with the Port to port connection connection type).
Signal Value	Displays the constant value defined for the port (only for ports with the Tied to constant value connection type).
Export Name	Displays the export name for the port (only for ports with the Hier connection connection type). This name is used at the top level.

Symbols in the Port Connections View

The following describes the symbols in the Interface Connections view:

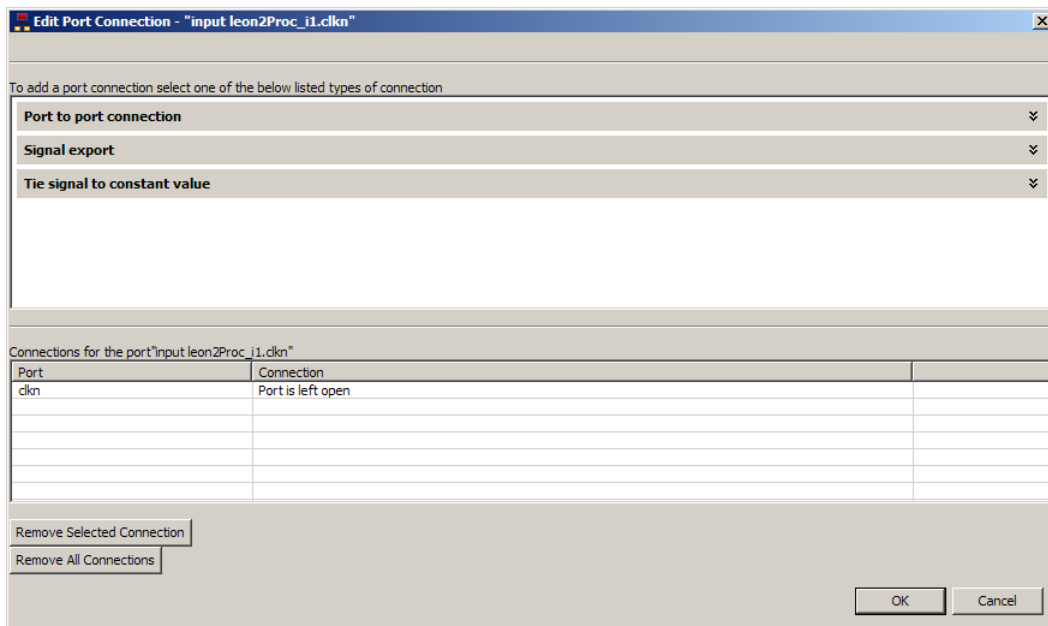
Component symbols	Same as the Design view. See Instance and Component Display, on page 69 for descriptions.
-------------------	---

	Indicates that you can edit the description.
	Indicates a field that is not applicable, based on the type of connection.
	Indicates an input port
	Indicates an output port.
	Indicates an inout port.

Edit Port Connection Command

To display the Edit Port Connection dialog box, click the Port Connections tab in the Design view, and then click the pen icon in the Edit column for the port you want to edit.

See [Making Port-Level Connections, on page 40](#) for detailed procedures on making port connections.



Port to port connection

Click Port to Port Connection. Lets you select a compatible port and an optional port range on another instance from a pull-down list. The name of this port appears in the Connecting Port column of the Port Connections view.

See [Making Port-to-Port Connections, on page 41](#) and [Connecting HAPS Peripherals, on page 56](#) for procedures.

Signal export

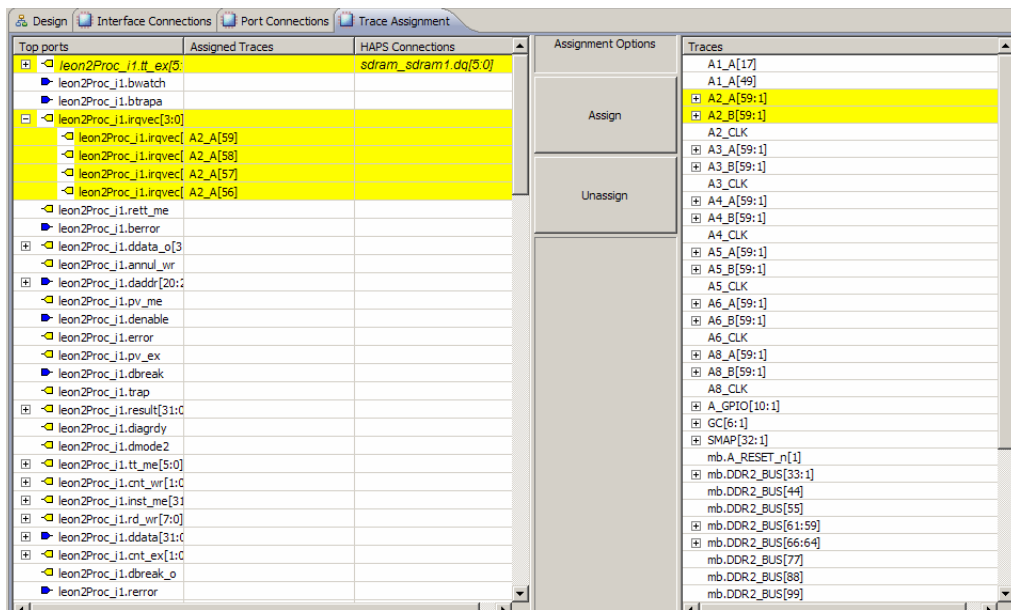
Click Signal Export. Lets you specify a name for the port at the top level. The default name is `<instance_name>_<port_name>`, but you can specify any name you want. This name appears in the Export Name column of the Port Connections view.

See [Making Hierarchical Port Connections, on page 42](#) for details.

Tie signal to constant value	<p>Click Tie Signal to Constant Value. Lets you specify a constant value for an input port. If a default value was defined, it displays, and you can edit this. The default value must be within the range of the port width. If the port width is 5, the valid range for the default value is 0 to (2^5-1).</p> <p>The value you specify here appears in the Signal Value column of the Port Connections view. See Tying Ports to a Constant Value, on page 43 for details.</p>
Remove selected connection	<p>Removes the selected connection. See Removing Port-Level Connections, on page 44 for more information.</p>
Remove all connections	<p>Removes all connections. See Removing Port-Level Connections, on page 44 for more information.</p>

Trace Assignment View

This view becomes available when you specify a HAPS board file, as described in [The System Designer-HAPS Board Design Flow, on page 52](#). It displays the top-level design ports and the available HAPS connector pins. It only displays HAPS pins that connect to the FPGA or a peripheral.



See [Assigning Traces for Designs with HAPS Boards](#), on page 57 for information on how to assign traces.




This view has the following columns

Top Ports	Lists the top-level ports in the design.
Assigned Traces	Lists the assigned trace for the top-level port.
HAPS Connections	Lists the corresponding HAPS peripheral port if the top-level port is connected to a HAPS peripheral.
Traces	Lists the physical traces available on the HAPS board.

The view contains two buttons for trace assignment:







Assign	Assigns the logical net for the selected port to the physical trace you selected.
Unassign	Removes the selected trace assignment.

The following display conventions are used in this view:

	Indicates an input port
	Indicates an output port.
	Indicates an inout port.
Yellow background	Assigned traces
Yellow background with italic lettering	Assigned traces that cannot be edited in this view, like design ports that are connected to the ports of a HAPS peripheral

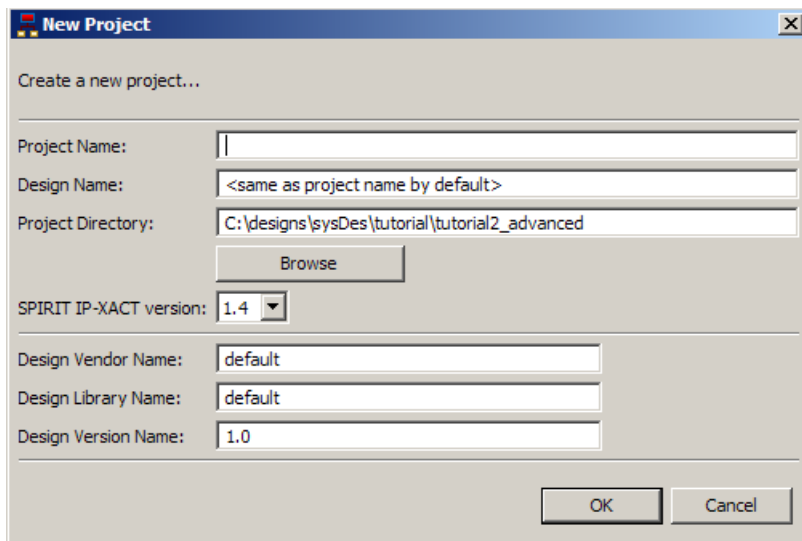
Project Menu Commands

Use the Project menu to open, create, save, and close projects. The following table describes the Project menu commands.

Command	Description
 New Project	Creates a new project. If a project is already open, it prompts you to save it before creating a new one. See New Project Command , on page 81.
 Open Project	Opens a project or file.
 Save Project	Saves a project or a file.
 Close Project	Closes the current project.
 Run Tcl Script	Lets you specify a Tcl script to drive System Designer.
 Initialize Project	Lets you start a new project or open an existing one. See Initialize Project Command , on page 82.
Exit	Ends the session and exits the tool.

New Project Command

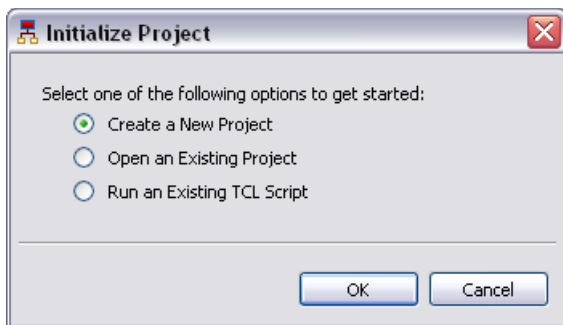
Select Project->New Project to display the New Project dialog box where you can specify options for a new project.



Field/Option	Description
Project Name	Specifies a name for the new project.
Design Name	Specifies a name for the IP design you are creating with the System Designer tool. The default name is the same name as the project name you specify.
Project Directory	Specifies a directory for the System Designer project.
Design Vendor Name	Specifies the name of the IP vendor.
Design Library Name	Specifies the name of the IP library.
Design Version Name	Specifies a version number for the System Designer design.

Initialize Project Command




Select Project->Initialize Project. You also see this dialog box when you first start the System Designer tool. It lets you set up a System Designer session by opening a project, creating a new project, or running a Tcl script. If you already have a project open, it prompts you to save it.



Field/Option	Description
Create a New Project	Opens the New Project dialog box (see New Project Command, on page 81), where you can set up a new project.
Open an Existing Project	Lets you browse and open an existing System Designer project.
Run an Existing Tcl Script	Lets you specify a Tcl script to run.

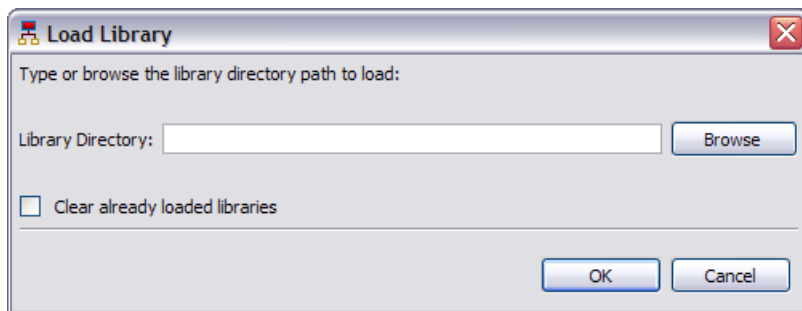
Library Menu Commands

Use the Library menu to load or remove libraries. The following table describes the Library menu commands.

Command	Description
 Load IP Library	Loads an IP library into a System Designer project. See Load IP Library Command , on page 84.
 Clear IP Library	Removes the specified IP library from the System Designer project.
 Refresh IP Library	Updates the specified IP library.

Load IP Library Command



This command lets you specify the IP library you want to include in your System Designer project.



Field/Option	Description
Library Directory	Specifies the library to be loaded. You can browse to find the library by clicking the Browse button.
Clear already loaded libraries	Determines whether existing libraries are removed from the project when the specified library is added. <ul style="list-style-type: none">• Enabled Removes existing libraries when the new library is added• Disabled Retains existing libraries in addition to the new one.

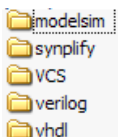
Actions Menu Commands

Use the Actions menu to configure your design, connect components, verify the design, and generate output files. The following table describes the Actions menu commands.

Command	Description
 Generate Files 	Generates System Designer output files that can be synthesized in the Synplify Pro or Synplify Premier interfaces. You can also access this command by clicking the Generate command in the UI. See Generate Files Command, on page 87 .
 Open Design Configuration	Specifies parameters for configuring instances and buses. See Open Design Configuration Command, on page 87 .
 Open Design Constraints	Opens a window for checking IP-XACT clock and delay constraints. See Open Design Constraints Command, on page 91 .
 Auto Connect Interfaces	Automatically connects all unconnected interfaces.
 Auto Connect Addresses	Automatically connects all unlocked addresses. To manually connect the interfaces, see Open Design Configuration Command, on page 87 .
 Auto Connect and Configure	Automatically connects all unconnected interfaces and all unlocked addresses. To manually connect the interfaces, see Open Design Configuration Command, on page 87 .
 Check IP Library Schema	Checks that the library conforms to the SPIRIT schema and prints the results in the Console window.
 Check IP Library Rules	Checks that the library conforms to the SPIRIT design rules and prints the results in the Console window.
 Check Design	Checks that the design conforms to the SPIRIT design rules and prints the results in the Console window.
Preferences	Specifies various design preferences. See Preferences Command, on page 95 for details.

Generate Files Command

When you run this command, the tool generates the following directories in the synthesis project directory:

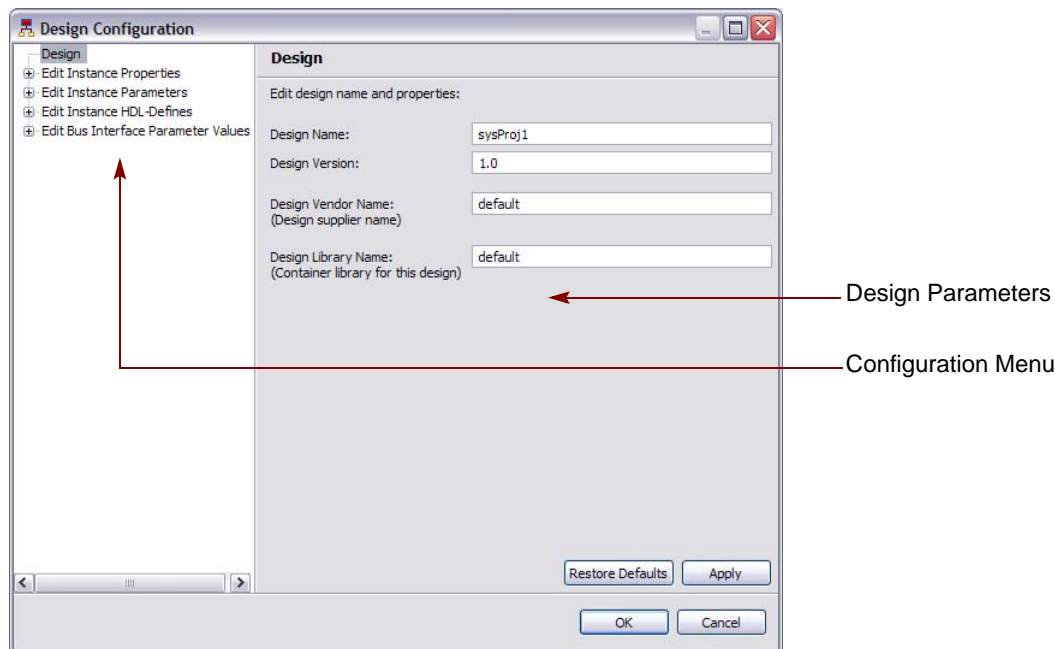


verilog	Contains Verilog wrapper files for the instantiated components and the top-level Verilog file with the component connection information.
vhdl	Contains VHDL wrapper files for the instantiated components and the top-level VHDL file with the component connection information.
vcs	Contains the <code>vcs.ksh</code> script to run VCS for the generated system.
synplify	Contains the <code>synthesis.slib</code> file. The synthesis project is updated to include this file.
modelsim	Contains a script to run VCS for the generated system.

Open Design Configuration Command

Select Actions->Open Design Configuration. This command lets you set or edit various design configuration parameters. To automatically set configuration parameters, use the Actions->Auto Connect Addresses or the Actions->Auto Connect and Configure commands.

The Design Configuration dialog box consists of a menu pane on the left. The configuration options for the selected menu item are displayed on the right.



The table describes the menu and common options shown here:

Design	See Design Configuration Parameters, on page 89 .
Edit Instance Properties	See Edit Instance Properties, on page 89 .
Edit Instance Parameters	See Edit Instance Parameters, on page 90 .
Edit Instance HDL-Defines	See Edit Instance HDL-Defines, on page 90 .
Edit Bus Interface Parameter Values	See Edit Bus Interface Parameter Values, on page 91 .
Restore Defaults	Restores the default values for the parameters you modified on that pane.

Design Configuration Parameters

This pane contains editable properties for the design.

Design Name	Specifies the name for the design.
Design Version	Specifies the version number for the design.
Design Vendor Name	Specifies the name of the vendor, as supplied by the vendor. You must supply a name. If you leave this set to default , the output files write out the vendor name as default .
Design Library Name	Specifies the name for the library from the vendor. You must supply a name. If you leave this set to default , the output files write out the library name as default .

Edit Instance Properties

Expand Edit Instance Properties in the menu, and select one of the instances from the list. This pane lists the properties for the selected instance.

leon2Proc_i1 (leon2Proc)

Edit instance properties:

Model-View: <AUTO>

HDL (Hint): <AUTO>

Model-view	Specifies the model view for the instance. If you leave this set to AUTO , the tool uses the model view in the format specified by HDL (Hint) .
HDL (Hint)	Specifies the preferred RTL format for the model view for this instance. If you specify a format that cannot be found, the tool uses whatever format is available. If you leave this set to AUTO , the tool looks for files Verilog files first, and then VHDL.

Edit Instance Parameters

Expand Edit Instance Parameters in the menu, and select one of the instances listed under it. This pane lists the editable parameters for the selected instance, like the start and end addresses shown below. This pane only lists the vendor-defined parameters for the IP-XACT component. It allows you to edit these parameters, but you cannot add or delete parameters.

leon2_ahbbus_i1 (leon2_ahbbus)

Edit instance parameter values:

1. Parameter: start_addr_slv0 (Prompt: Slave 0 Starting Address (Addr[31:28]));	0x0000
2. Parameter: end_addr_slv0 (Prompt: Slave 0 Ending Address (Addr[31:28]));	0x0111
3. Parameter: start_addr_slv1 (Prompt: Slave 1 Starting Address (Addr[31:28]));	0x1000
4. Parameter: end_addr_slv1 (Prompt: Slave 1 Ending Address (Addr[31:28]));	0x1111
5. Parameter: split_slv0 (Prompt: Slave 0 Splitted Transactions:)	false
6. Parameter: enable_slv0 (Prompt: Slave 0 Enabled:)	true
7. Parameter: split_slv1 (Prompt: Slave 1 Splitted Transactions:)	false
8. Parameter: enable_slv1 (Prompt: Slave 1 Enabled:)	true
9. Parameter: defmast (Prompt: Default master:)	0

Edit Instance HDL-Defines

Expand Edit Instance HDL-Defines in the menu, and select one of the instances listed under it. This pane lists the editable HDL parameters for the selected instance.

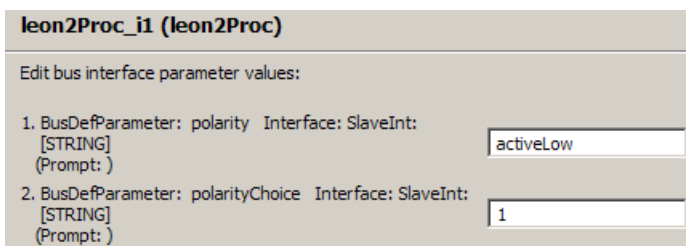
leon2_uart_i1 (leon2_uart)

Edit instance hdl-define values:

22. hdlsrc/leon2_Uart.v Defines net	67
-------------------------------------	----

Edit Bus Interface Parameter Values

Expand Edit Bus Interface Parameter Values in the menu, and select one of the components listed under it. This pane lists the bus interface parameters that are defined in the input libraries for the selected instance. You can edit the values.



Open Design Constraints Command

Select Actions->Open Design Constraints. This command displays the constraints in the IP-XACT component descriptions. It does not include constraints that are specified in the abstraction definition. The constraints shown are translated to the .sdc file format for later synthesis with the Synplify Pro or Synplify Premier tools. This command only works if you select a SPIRIT IP-XACT version 1.4 library; it does not work with the 1.2 version.

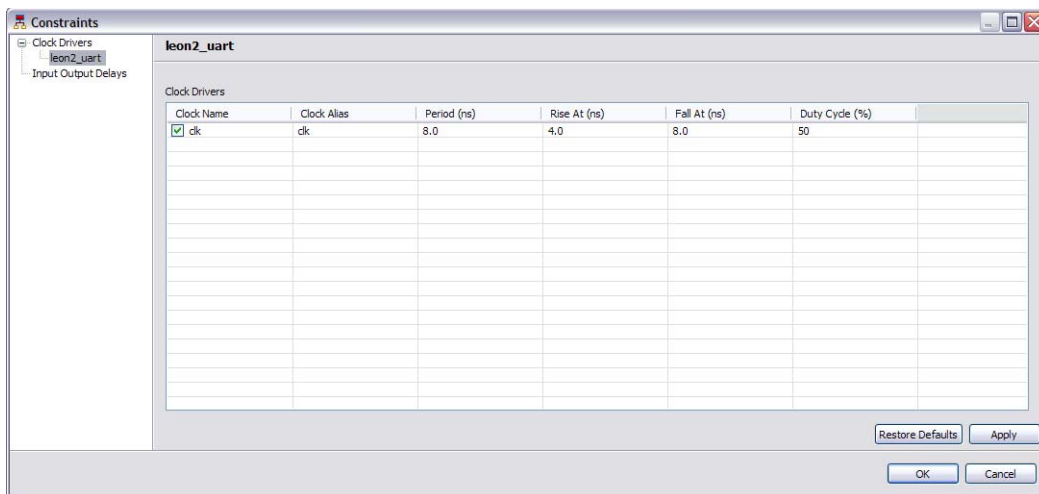


The Constraints dialog box displays the following types of constraints:

- Clock definitions for clock drivers. See *Clock Driver Constraints*, on page 92.
- Input/output delays for the specified ports. See *Input/Output Delay Constraints*, on page 93.

Clock Driver Constraints

When you select one of the components listed under Clock Drivers on the left, the command displays the relevant clock constraints. The components listed under Clock Drivers are the clock drivers for the top-level design and the components from which clock objects are specified in the input SPIRIT IP-XACT file.



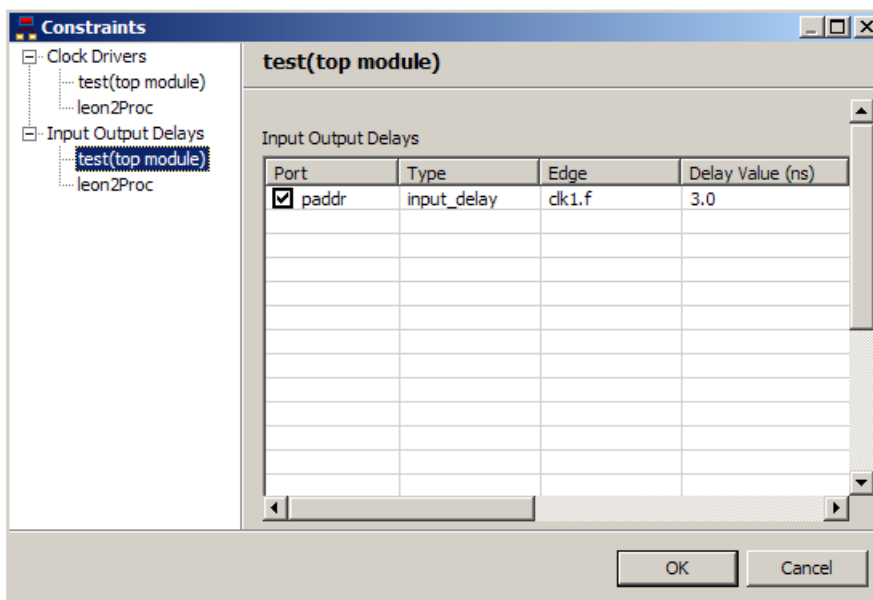
These constraints are mapped to Synplify Pro/Premier `define_clock` constraints.

Clock Name	Port name for the clock. See Clock Driver Constraints, on page 94 for information on how the tool handles clock names when it writes out the constraints to an .sdc file.
Clock Alias	The alias name for the clock. If no alias is specified, this is the same as Clock Name.
Period	Specifies the clock period in nanoseconds.

Rise At	Specifies the time in nanoseconds from the start of the cycle, at which the rising edge of the clock occurs.
Fall At	Specifies the time in nanoseconds from the start of the cycle, at which the falling edge of the clock occurs.
Duty Cycle	Specifies the duty cycle for the clock, as a percentage.

Input/Output Delay Constraints

When you select one of the components listed under Input Output Delays on the left, the command displays the relevant clock constraints. The components listed under Input Output Delays are the ones for which input/output delays are specified in the SPIRIT IP-XACT file.



These constraints are mapped to `define_input_delay` and `define_output_delay` constraints in the Synplify Pro or Synplify Premier synthesis tools.

Port	Port with the specified delay. See Hierarchical Port Constraints, on page 95 for information on how the tool writes out hierarchical port constraints.
Type	Specifies the type of delay: <code>input_delay</code> or <code>output_delay</code> .

Edge	Specifies the edge of the clock with respect to which the delay is defined.
Delay Value	Specifies the delay in nanoseconds at the given edge.

Generating an .sdc File for Synthesis

The System Designer tool takes the constraints defined in the SPIRIT IP-XACT timingConstraint element and converts it to the Synplify constraints file format (.sdc). It generates an sdc file and includes it in the Synplify project for later synthesis.

It generates the following constraint files:

<code><design name>.sdc</code>	Contains clock definitions and timing constraints for ports that are taken to the top level.
<code><component_name>.sdc</code>	Block-level constraint files for components for which clock definitions or timing constraints are specified.

Clock Driver Constraints

The tool handles clock names in clock drivers as follows:

- If no clock name (`spirit:clockName`) is defined for the clock driver, the name of the port for which the clock driver is defined is used as the clock name.
- If the defined clock name (`spirit:clockName`) for the clock driver is the name of another port in the same component, the System Designer tool does not use that name, but instead, uses the name of the port for which the clock driver is defined as the clock name.
- If there are multiple instances of a component and the component has a clock driver for a hierarchical port that is taken to the top level, the tool prefixes the clock name with the instance name:
`<instance_name>_<clock_name>`.
- If a `spirit:clockName` is defined in a clock driver (`port/wire/driver/clockDriver`), and there is an `input_delay` or `output_delay` constraint, the tool always uses the `spirit:clockName` for the clock driver, regardless of whether the timing constraints (`port/wire/constraintSets/constraintSet/timingConstraints`) refer to the clock port name or the `spirit:clockName` of the clock driver.

Hierarchical Port Constraints

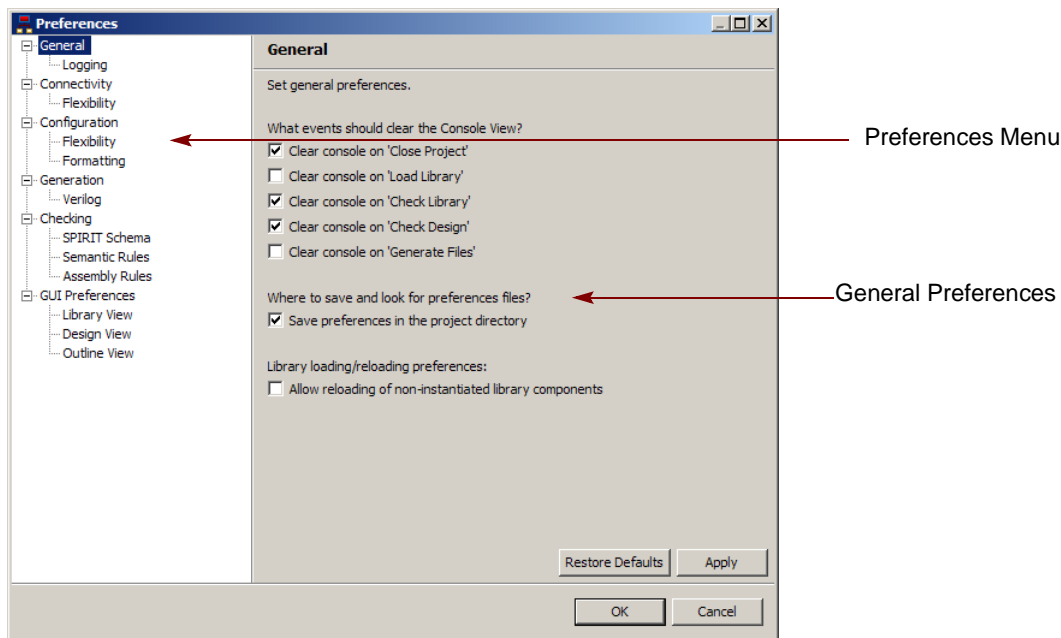
The following describe how the tool handles constraints on hierarchical ports:

- When a component port is taken to the top level, the tool writes the clock definitions and timing constraints for that port to the top-level constraints file as well as to the block-level constraints file for the component in which the port occurs.
- When ports of a component that are not part of any interface (default ports) are taken to the top level, the tool uses the `<instance_name>_<port_name>` scheme for the port name. This is how the System Designer tool writes out the port name in the top-level netlist file.
- When ports of a hierarchically connected interface of a component are taken to the top level, the tool uses the `<instance_name>_<interface_name>_<port_name>` convention to refer to the port in the top-level .sdc file. This is how the System Designer tool writes out the port name in the top-level netlist file.
- When a port for which a timing constraint is specified is taken to the top level but the clock referred to by the timing constraint is not taken to the top level, the System Designer tool does not apply that timing constraint to the design.

Preferences Command

Select Actions->Preferences. This command lets you set or edit display and other preferences. By default, the preferences are stored in this file in the installation directory: `data/system_designer.prefs`.

The Preferences dialog box consists of a menu pane on the left and specific parameters for the selected component on the right.



The following table describes the menu options:

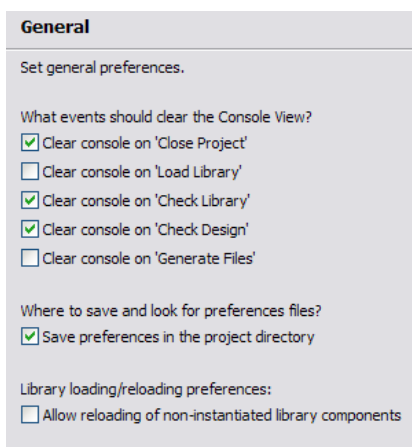
General	Sets general preferences. See General Pane, on page 97 .
Connectivity	Allows more flexibility in making connections by setting preferences for connections. See Connectivity Pane, on page 100 .
Configuration	Sets preferences for configuration. See Configuration Pane, on page 102 .
Generation	Sets preferences for running the System Designer tool. See Generation Pane, on page 105 .
Checking	Sets preferences for checking design rules. See Checking Pane, on page 106 .
GUI Preferences	Sets GUI preferences. See GUI Preferences Pane, on page 109 .
Restore Defaults	Resets the settings to the default value.

General Pane

Select Actions->Preferences. This pane lets you set general preferences ([General Pane - General Preferences, on page 97](#)), and also has a sub-entry for setting logging preferences ([General Pane - Logging Preferences, on page 98](#)).

General Pane - General Preferences

Select Actions->Preferences. If necessary, select General. This pane lets you set general System Designer preferences.



The following table describes the options:

Clear console on 'Close Project'	When enabled, clears the console when you close a project.
Clear console on 'Load Library'	When enabled, clears the console when you load a library.
Clear console on 'Check Library'	When enabled, clears the console when you check a library.
Clear console on 'Check Design'	When enabled, clears the console when you check the design.

Clear console on ‘Generate Files’	When enabled, clears the console when you generate the output files.
Save preferences in the project directory	<p>When enabled, saves a copy of your preferences in a <code>system_designer.prefs</code> file in your project directory. Note that the preferences file is only saved in your project directory if you update your preferences for that project.</p> <p>When disabled, the tool uses the default preferences file in <code><install_dir> /bin/data</code>.</p>
Allow reloading of non-instantiated library components	When enabled, the tool reloads any non-instantiated library components when you refresh the library.

General Pane - Logging Preferences

Select Actions->Preferences, and expand General->Logging from the menu on the left. These options determine the log messages you see in the log file and console.

Logging

Set Log messaging related preferences.

Set Logging Level for each log destination.

Set overall logging level for the logger

FINE

Set Console-view logging level for the logger

SEVERE

Set Shell-view logging level for the logger

SEVERE

Set Log-file logging level for the logger

FINE

The following table describes the options. You can set every option to one of the following: SEVERE, WARNING, INFO, CONFIG, FINE, FINER, or FINEST, which are described in the next table.

Set overall logging level for the logger	Sets the default level of all log messages in the log file, Command Shell or Console views.
--	---

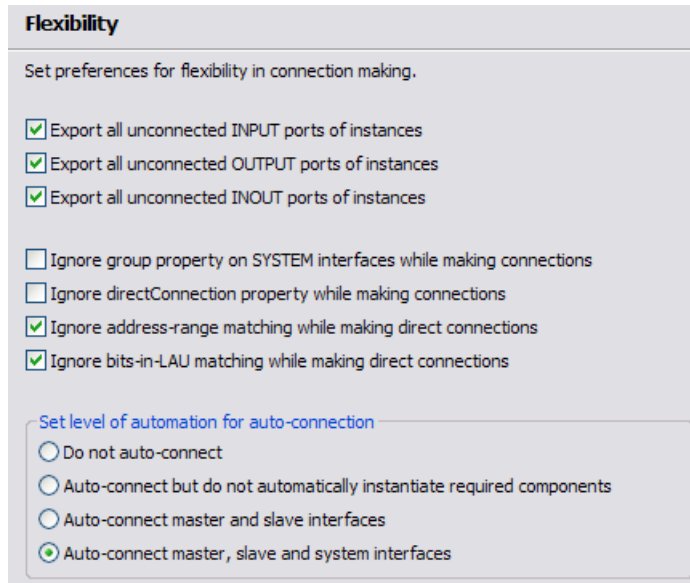
Set Console view logging level for the logger	Sets the level of all log messages displayed in the Console view. This setting overrides the overall logging setting.
Set Shell view logging level for the logger	Sets the level of all log messages displayed in the command shell window. This setting overrides the overall logging setting.
Set Log file logging level for the logger	Sets the level of all log messages displayed in the log file. This setting overrides the overall logging setting.

This table defines the logging levels. If you set a certain level, it automatically includes all levels above it in the table. For example, if you set the level to INFO, the tool prints INFO, WARNING, and SEVERE messages.

SEVERE	Prints messages only if the tool encounters an unexpected input or is unable to perform a requested operation.
WARNING	Prints messages only if the tool encounters a condition that is not the preferred ideal, but is not a severe deviation.
INFO	Prints information about the operations performed
CONFIG	Prints informational messages about tool configuration
FINE	Prints informational messages that are less significant than INFO
FINER	Prints informational messages that are less significant than FINE
FINEST	Prints informational messages that are less significant than FINER

Connectivity Pane

Select Actions->Preferences, and expand Connectivity in the menu on the left. Click Flexibility, under Connectivity. This pane lets you set how flexible you want to be when making connections.



Flexibility

Set preferences for flexibility in connection making.

- ☒ Export all unconnected INPUT ports of instances
- ☒ Export all unconnected OUTPUT ports of instances
- ☒ Export all unconnected INOUT ports of instances
- ☐ Ignore group property on SYSTEM interfaces while making connections
- ☐ Ignore directConnection property while making connections
- ☒ Ignore address-range matching while making direct connections
- ☒ Ignore bits-in-LAU matching while making direct connections

Set level of automation for auto-connection

- ☐ Do not auto-connect
- ☐ Auto-connect but do not automatically instantiate required components
- ☐ Auto-connect master and slave interfaces
- ☒ Auto-connect master, slave and system interfaces

The following table describes the options:

Export all unconnected INPUT ports or instances	Connects all unconnected input ports on the embedded system to the top-level design ports.
Export all unconnected OUTPUT ports or instances	Connects all unconnected output ports on the embedded system to the top-level design ports.
Export all unconnected INOUT ports or instances	Connects all unconnected inout ports on the embedded system to the top-level design ports.
Ignore group property on system interfaces while making connections	When enabled, the tool ignores the group property on system interfaces, and lets you connect system interfaces when the group names are not the same. This overrides the input library specifications.

Ignore directConnection property while making connections	<p>When enabled, the tool ignores the directConnection property, and lets you force connections for all interfaces. This overrides what the input libraries specify, which is the following:</p> <ul style="list-style-type: none"> • When the property is true, connections can only be made from master to slave. • When the property is false, connections can only be made between master and mirrored master, and slave and mirrored slave, through a bus.
Ignore address range matching while making direct connections	<p>When enabled, the tool ignores the IP-XACT address range matching rule. You can now connect a master interface directly to a slave interface even if the address range of the slave interface exceeds the address range of the master interface.</p>
Ignore bits-in-LAU while making direct connections	<p>Bits-in-LAU specifies the number of data bits addressable by the least significant bit in the bus interface. When this option is enabled, the tool ignores this value, and lets you connect a master interface directly to a slave interface, even if the address space of the master does not match the bits-in-LAU value in the slave memory map.</p>
Do not auto-connect	<p>When enabled, does not make any automatic connections.</p>
Auto-connect but do not automatically instantiate required components	<p>Makes auto-connections when enabled, but does not automatically instantiate the components required for the connections.</p>
Auto-connect master and slave interfaces	<p>When enabled, auto-connects master and slave interfaces, and automatically instantiates any components required for the connections.</p>
Auto-connect master, slave, and system interfaces	<p>When enabled, auto-connects master, slave, and system interfaces, and automatically instantiates any required components.</p>

Configuration Pane

Select Actions->Preferences, and expand Configuration in the menu on the left. You can select Flexibility or Formatting under Connectivity. These panes lets you set design configuration and formatting preferences.

Configuration Pane - Flexibility Preferences

These options determine how flexible your design configuration settings can be.

Flexibility

Set preferences for flexibility in configuration.

☐ Allow instantiation of non-usable components

☐ Allow guessing if a model-view is synthesizable HDL view

Allow editing of parameters with resolve=<non-user> property

☐ Allow editing only resolve=user

☐ Allow editing of resolve=immediate

☐ Allow editing of resolve=dependent

☒ Allow editing of resolve=generated

☐ Allow changing of 'instance' HDL value if no matching model-view found

☐ Allow changing of 'project' HDL value if no matching model-view found

Tolerance level while choosing a model-view with mismatching info

☐ Select model view only if vendor, technology, part, package and speed grade match

☒ Select model view if vendor and technology match

☐ Select model view if vendor matches

☐ Select model view even if no elements match

Minimum address-block size for auto-addressing

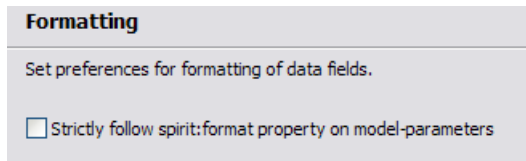
The following table describes the options:

Allow instantiation of non-usable components	When enabled, it lets you instantiate non-usable components in your design.
Allow guessing if a model view is a synthesizable HDL view	Allows a model view with tool-specific information (<code>spirit:envidentifier</code> tag) to be considered a generic model view.
Allow editing only <code>resolve=user</code>	Allows parameter to be edited only if the <code>resolve</code> property is set to <code>user</code> . By default, the <code>user</code> setting (where the value is specified as user input) is the only one that is editable.
Allow editing of <code>resolve=immediate</code>	Allows editing of <code>resolve=immediate</code> parameter even if <code>resolve</code> is not set to <code>user</code> . By default, the <code>user</code> setting (where the value is specified as user input) is the only one that is editable. The value is specified in the containing element.
Allow editing of <code>resolve=dependent</code>	Allows editing of <code>resolve=immediate</code> parameter even if <code>resolve</code> is not set to <code>user</code> . By default, the <code>user</code> setting (where the value is specified as user input) is the only one that is editable. This setting indicates that the value is defined by an XPATH equation.
Allow editing of <code>resolve=generated</code>	Allows editing of <code>resolve=immediate</code> parameter even if <code>resolve</code> is not set to <code>user</code> . By default, the <code>user</code> setting (where the value is specified as user input) is the only one that is editable. This setting indicates that the value is set by a generator and then stored in a design or design configuration.
Allow changing of 'instance' HDL value if no matching model view found	Overrides the default and lets you set any HDL type for the design in the Design Connections dialog box.
Allow changing of 'project' HDL value if no matching model view found	Overrides the default and lets you set a HDL type for the design in the Settings panel.

Select model view only if vendor, technology, part, package and speed grade match	When enabled, selects the model view only if all the elements match those in the IP-XACT spirit:envidentifier VendorSpecific field. This field includes additional processing information that may be required by the model in a particular environment. The System Designer tool uses this information to match FPGA devices: <code><vendor>.<technology>.<part>.<package>.<speed_grade></code>
Select model view if vendor and technology, match	When enabled, selects model view if vendor and technology match.
Select model view if vendor matches	When enabled, selects model view if vendor matches.
Select model view even if no element match	When enabled, selects model view even if nothing matches.
Minimum address block size for auto-addressing	Specifies the minimum size of blocks to be allocated while auto-addressing. The value must be a multiple of the minimum number of addressable units in the system. You must determine this value; the tool does not validate the range of the value you enter.

Configuration Pane - Formatting Preferences

These options determine how strictly you follow the SPIRIT format for data fields:



The following table describes the options:

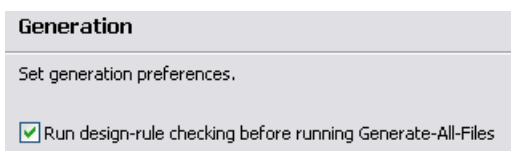
Strictly follow spirit:format property on model parameters	When enabled, the tool strictly follows the SPIRIT format defined for model parameters.
--	---

Generation Pane

Select Actions->Preferences, and click Generation in the menu on the left. On this pane, you can set preferences for the System Designer run.

Generation Pane->General Preferences

Select Actions->Preferences, and click Generation in the menu on the left. On this pane, you can decide whether to check design rule correctness before a System Designer run.

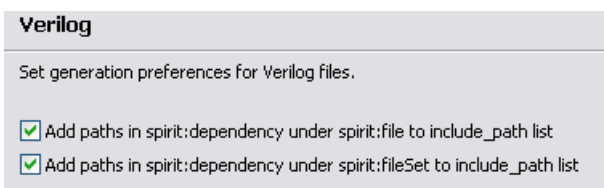


Run design-rule checking before running Generate All Files

When enabled, checks design rule before running System Designer and generating output files.

Generation Pane->Verilog

Select Actions->Preferences, and expand Generation in the menu on the left. Click Verilog, under Generation. The following options let you set Verilog preferences:



Add paths in spirit:dependency under spirit:file to include_path list

When enabled, adds the files specified in spirit:dependency as an include directory in the synthesis project.

Add paths in spirit:dependency under spirit:fileSet to include_path list

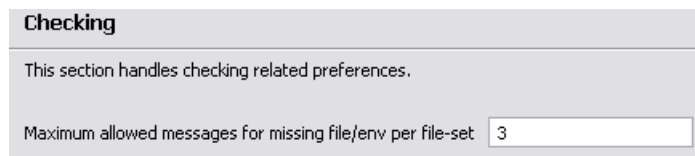
When enabled, adds the files specified in spirit:dependency for the file set as an include directory in the synthesis project.

Checking Pane

Select Actions->Preferences, and click Checking in the menu on the left. You can select sub-entries under Checking, where you can set design rule checking preferences to conform with SPIRIT standards.

Checking Pane - General Preferences

Select Actions->Preferences, and click Checking. When HDL file paths in `spirit:fileSet` are specified as an environment variable, there might be error messages because the environment variable is not defined on your computer. This option determines how many of these messages you see.

The screenshot shows a dialog box titled "Checking". Below the title bar, there is a text area that says "This section handles checking related preferences." At the bottom, there is a label "Maximum allowed messages for missing file/env per file-set" followed by a text input field containing the number "3".

Checking	
This section handles checking related preferences.	
Maximum allowed messages for missing file/env per file-set	3

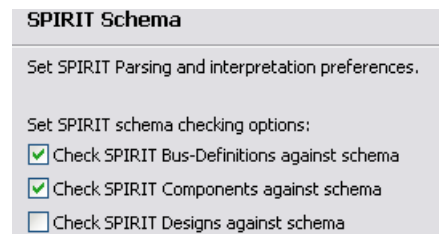
Maximum allowed messages for missing file/env per file-set

Determines how many error messages you see because HDL paths are specified as environment variables in `spirit:fileSet`, and that variable is not defined on your computer.

Note that the tool does not check the range for this preference.

Checking Pane - SPIRIT Schema

Select Actions->Preferences, expand Checking in the menu on the left, and click SPIRIT Schema. These options set preferences for checking SPIRIT schema:

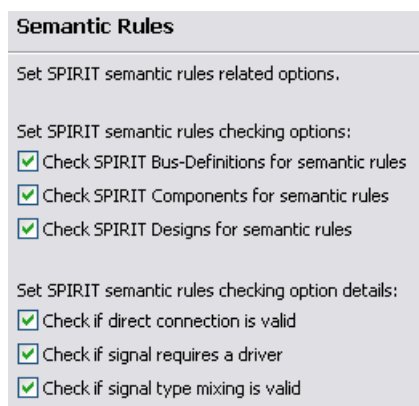
The screenshot shows a dialog box titled "SPIRIT Schema". Below the title bar, there is a text area that says "Set SPIRIT Parsing and interpretation preferences." Below that, there is a section titled "Set SPIRIT schema checking options:" followed by three checked checkboxes: "Check SPIRIT Bus-Definitions against schema", "Check SPIRIT Components against schema", and "Check SPIRIT Designs against schema".

SPIRIT Schema	
Set SPIRIT Parsing and interpretation preferences.	
Set SPIRIT schema checking options:	
<input checked="" type="checkbox"/>	Check SPIRIT Bus-Definitions against schema
<input checked="" type="checkbox"/>	Check SPIRIT Components against schema
<input checked="" type="checkbox"/>	Check SPIRIT Designs against schema

Check SPIRIT bus definitions against schema	When enabled, checks the SPIRIT bus definitions in the design against the IP-XACT schema file that specifies the structure of the XML bus definitions.
Check SPIRIT components against schema	When enabled, checks the SPIRIT components in the design against the IP-XACT schema file that specifies the structure of the XML SPIRIT component file.
Check SPIRIT designs against schema	When enabled, checks the SPIRIT design definitions against the IP-XACT schema file that specifies the structure of the XML design definitions.
Check SPIRIT Abstraction definitions against schema	When enabled, checks the SPIRIT abstraction definitions in the design against the IP-XACT schema file that specifies the structure of the XML abstract definitions. This option is only available when you select SPIRIT IP-XACT version 1.4.

Checking Pane - Semantic Rules

Select Actions->Preferences, expand Checking in the menu on the left, and click Semantic Rules. These options set preferences for checking SPIRIT semantic rules when the library is loaded and checked against the IP-XACT schema specifications:



Check SPIRIT bus definitions for semantic rules	When enabled, checks that the SPIRIT bus definitions follow the semantic rules defined by the IP-XACT specifications. The rules that are checked vary, depending on which IP-XACT version you are using. If it finds errors, the tool prints an error message in the Console window, along with the IP-XACT rule number for that failure.
Check SPIRIT components for semantic rules	When enabled, checks that the SPIRIT components follow the semantic rules. The rules that are checked vary, depending on which IP-XACT version you are using. If it finds errors, the tool prints an error message in the Console window, along with the IP-XACT rule number for that failure.
Check SPIRIT designs for semantic rules	When enabled, checks that the SPIRIT design definition follows the semantic rules. The rules that are checked vary, depending on which IP-XACT version you are using. If it finds errors, the tool prints an error message in the Console window, along with the IP-XACT rule number for that failure.
Check SPIRIT abstraction definitions for semantic rules	When enabled, checks that the SPIRIT abstract definition follows the semantic rules. The rules that are checked vary, depending on which IP-XACT version you are using. This option is only available if you selected SPIRIT version 1.4. If it finds errors, the tool prints an error message in the Console window, along with the IP-XACT rule number for that failure.
Check if direct connection is valid	When enabled, checks that the direct connection between a master and slave has compatible signal directions for the connecting interfaces.
Check if signal requires a driver	When enabled, checks that a port that requires a driver (spirit:requiresDriver = true) has a compatible signal.

Checking Pane - Assembly Rules

Select Actions->Preferences, expand Checking in the menu on the left, and click Assembly Rules. These options set preferences for checking that the design conforms to SPIRIT system assembly rules, so that the output files are correct:

Assembly Rules

Set System-assembly rules related preferences.

Set System-assembly rules checking options:

- ☒ Check Bus-Definitions for System-assembly rules
- ☒ Check Components for System-assembly rules
- ☒ Check Designs for System-assembly rules

Check bus definitions for system assembly rules

When enabled, checks that the SPIRIT bus definitions follow the rules for system assembly. When disabled, the tool does not check these rules after generating the output files.

Check components for system assembly rules

When enabled, checks that the SPIRIT components follow the rules for system assembly. When disabled, the tool does not check these rules after generating the output files.

Check designs for system assembly rules

When enabled, checks that the SPIRIT design definition follows the rules for system assembly. When disabled, the tool does not check these rules after generating the output files.

GUI Preferences Pane

Select Actions->Preferences, and click GUI Preferences in the menu on the left. You can select other entries under GUI Preferences. These panes let you set GUI display preferences.

GUI Preferences Pane - Library View

Select Actions->Preferences, expand GUI Preferences, and click Library View. This option lets you set display preferences for the Library view. Enable it to display the VLN descriptions of the library components in the library view.

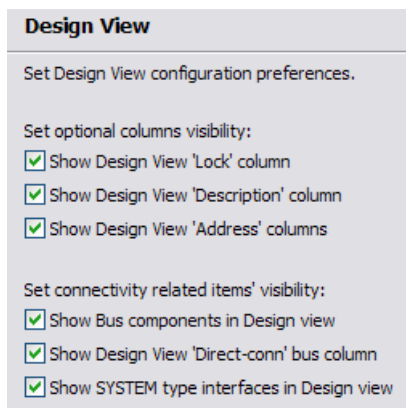
Library View

Set Library View configuration preferences.

- ☒ Show Library View 'Description' column

GUI Preferences Pane - Design View

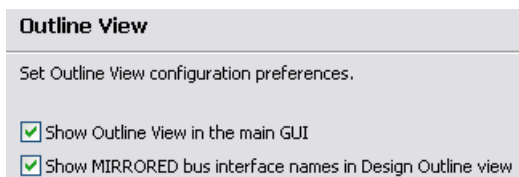
Select Actions->Preferences, expand GUI Preferences, and click Design View. These option lets you set display preferences for the Design view.




Show Design view 'Lock' column	When enabled, the Design view displays the Lock column, where you can set your preference for auto-addressing.
Show Design view 'Description' column	When enabled, the Design view displays the Description column. This column describes the kind of component or interface, its VLNV, and the connection type. If the connection type is defined as required , the interface can be automatically connected. If the connection type is explicit , it cannot be automatically connected.
Show Design view 'Address' column	When enabled, the Design view displays the Address columns, which show the base and high addresses for the interface.
Show Bus components in Design view	When enabled, displays the bus components in the Design view. When disabled, bus components are not displayed.
Show Design view 'Direct-conn' bus column	When enabled, the Design view display direct-connections between master and slave interfaces. They are displayed as vertical gray bars.
Show SYSTEM type interfaces in Design view	When enabled displays system interfaces. If it is disabled, the displays only shows master and slave interfaces.

GUI Preferences Pane - Outline View

Select Actions->Preferences, expand GUI Preferences, and click Outline View. These options let you set display preferences for the Outline view.



Show Outline view in the main GUI

When enabled, the System Designer window displays the Outline view on the right side of the window. When disabled, only the icon for the window is displayed on the right side. 

Show MIRRORED bus interface names in Design Outline view

When enabled, the Outline view displays the MIRRORED bus interface names.

Help Menu Commands

The following commands are available from the System Designer Help menu:

Command	Description
Help Contents	Opens the built-in help for the System Designer tool. If this is the first time you are opening Help, click Table of Contents in the help window to browse through a list of topics.
Online Documents	Lists the PDF versions of the System Designer documentation.
License Agreement	Opens a window with the license agreement.
About System Designer	Opens a window that displays the current System Designer version, as well as details about the plug-ins and configuration.

Context-Sensitive Command Menus

The System Designer tool also provides popup menus that list the most useful commands for specific situations. You access a popup menu by right-clicking. Many commands on the popup menus are also available from the menu bar at the top of the tool window. See the following for details:

- [Popup Commands in the Design View, on page 113](#)
- [Popup Commands in the Console and Command Shell Windows, on page 113](#)
- [Popup Commands in the Memory Map Window, on page 114](#)

Popup Commands in the Design View

You can access the following commands by right-clicking in the Design view.

Open Configuration	Opens the Design Configuration dialog box to the Instance Parameters page, where you can edit parameters for the selected instance.
Rename Instance	Opens a dialog box where you can rename the selected component.
Remove Instance	Deletes the selected component from the Design view.
Add another Instance	Adds another instance of the selected component to the Design view.
Refresh View	Refreshes the Design view.
Show Bus Components	Toggles on/off the display of bus components in the Design view.
Show Direct Connections	Toggles on/off the display of direct connections between master and slave interfaces (grey columns) in the Design view.
Show System Interfaces	Toggles on/off the display of system interfaces in the Design view.

Popup Commands in the Console and Command Shell Windows

You can access the following commands by clicking the triangle (View menu) in the upper right of the Console or Command Shell windows.

Copy	Copies the selected text.
Paste	Pastes the previously copied text. This command is not available in the Console window, but is available in the Command Shell window.
Refresh	Refreshes the view.
Clear View	Clears the view.

Popup Commands in the Memory Map Window

You can access the following commands by right-clicking in the Memory Map window.

Refresh View	Refreshes the view.
Show Only Typical Interfaces	Toggles the display on/off. When toggled on, the window only displays the memory maps of master and slave interfaces and excludes bridged and mirrored interfaces. When toggled off it shows all defined interfaces.
Auto Address Selected Interfaces	Auto-addresses the selected interfaces.

CHAPTER 4

Tcl Commands

The System Designer tool lets you use Tcl commands and scripts. See the following for details:

- [Overview of Tcl Commands, on page 116](#)
- [Library Commands, on page 117](#)
- [Connectivity Commands, on page 117](#)
- [Design Commands, on page 120](#)
- [Output File Commands, on page 121](#)
- [Project Commands, on page 122](#)
- [Preference Commands, on page 122](#)

Overview of Tcl Commands

You can enter TCL commands in the Console window or specify them in a script file.

Tcl Commands

The System Designer Tcl commands are grouped into the following categories:

library	Loads and unloads input IPs. See Library Commands, on page 117 .
connect	Controls connectivity between component instances or between component instances and the buses. See Connectivity Commands, on page 117 .
design	Controls design setting. See Design Commands, on page 120 .
generate	Generates project files and Verilog or VHDL wrappers. See Output File Commands, on page 121 .
project	Commands that control the System Designer project file. See Project Commands, on page 122 .
preferences	Sets preferences. See Preference Commands, on page 122 .

Tcl Syntax Conventions

The following conventions are used to specify the syntax for System Designer Tcl commands:

<>	Angle brackets indicate mandatory information
{ }	Elements separated by a pipe and enclosed in curly braces indicate a set of choices
[]	Information inside square brackets is optional.

Library Commands

Use the library command to specify how to load and unload IP files. The following table lists the options to the command:

Command	Description
library -list { <path> <comp> }	When <path> is specified, lists the library path. When <comp> is specified, lists the loaded components.
library -clear	Clears the loaded libraries, and the corresponding views.
library -load [<i>library_path</i>]	Loads the libraries from the specified <i>library_path</i> . If the path is not specified, the tool loads libraries from the paths that are currently set.
library -set < <i>library_path</i> >	Sets <i>library_path</i> as the path of the libraries to be loaded.
library -add < <i>library_path</i> >	Adds <i>library_path</i> as a path for loading libraries.
library -check {semantics schema}	When semantics is specified, checks library semantic rules When schema is specified, checks that library schema conform.

Connectivity Commands

The connect and disconnect commands control connectivity. You can connect component instances or connect component instances to buses.

Connect Commands

Command	Description
connect < <i>instance_name</i> > -interface < <i>interface_name</i> > -bus < <i>bus_number</i> >	Connects the specified interface <i>interface_name</i> in the instance <i>instance_name</i> to the bus specified by <i>bus_number</i> . The bus number refers to the index of the bus column displayed in the Design view.

Command	Description
<code>connect <instance_name>. <interface_name> -bus <bus_number></code>	Connects the specified interface <i><interface_name></i> in the instance <i><instance_name></i> to the bus specified by <i><bus_number></i> .
<code>connect <instance_name>. <interface_name> -hier <port_prefix></code>	Hierarchically connects the interface in <i><interface_name></i> to the instance <i><instance_name></i> using the hier connection port prefix name <i><port_prefix></i> .
<code>connect <instance_name1>. <interface_name1> <instance_name2>. <interface_name2></code>	Interconnects <i><interface_name1></i> in <i><instance_name1></i> and <i><interface_name2></i> in <i><instance_name2></i> .
<code>connect <instance_name1>. <interface_name1> <instance_name2></code>	Connects <i><interface_name1></i> in <i><instance_name1></i> and a compatible interface in <i><instance_name2></i> .
<code>connect -adhoc <instance_name1>.<port_name1> <instance_name2>. <port_name2></code>	Connects port <i><port_name1></i> of <i><instance_name1></i> to <i><port_name2></i> of <i><instance_name2></i> . You get an error if the specified ports are incompatible.
<code>connect -adhoc <instance_name1>.<port_name1> [left:right] <instance_name2>. <port_name2> [left:right]</code>	Connects port <i><port_name1></i> of <i><instance_name1></i> to <i><port_name2></i> of <i><instance_name2></i> . If you specify port ranges in <i>[left:right]</i> , only the specified range is connected. If no range is specified, the entire port is connected. You get an error in the following cases: <ul style="list-style-type: none"> • The specified ports do not have compatible widths • The specified ports do not have compatible port directions • One of the specified ports is an input port that is already connected.
<code>connect -adhoc <instance_name1>.<port_name1> [left:right] { <instance_name2>. <port_name2> [left:right] [,open[left:right]] <instance_name3>. <port_name3> [left:right], ... } }</code>	Connects multiple ports to a port. It connects <i><port_name1></i> of <i><instance_name1></i> to the ports listed within the curly braces { }. This is similar to Verilog concatenation. Use the <i>open[left:right]</i> syntax to specify those parts of a port that you want to leave unconnected. Note that the total width of the ports listed within the curly braces must be equal to the width of <i><port_name1></i> .

Command	Description
connect -const_value <instance_name>.<port_name> <constant_value>	Ties the port <port_name> of <instance_name> to the the constant value defined in <constant_value>. You get an error if the port has an adhoc connection already defined for it, or if the port is exported to the top level.
connect -signal_export <instance_name>.<port_name> <export_name>	Exports port <port_name> of <instance_name> to the top level, using <export_name> as the name for the port at the top level. You get an error if the port has another connection already defined for it.
connect -signal_export <instance_name>.<port_name> [left:right] <export_name>	Exports port <port_name> of <instance_name> to the top level, using <export_name> as the name for the port at the top level. You get an error if the port has an adhoc connection already defined for it.

Disconnect Commands

Command	Description
disconnect -adhoc <instance_name1>.<port_name1> <instance_name2>. <port_name2>[left:right]	Disconnects the connection between <port_name1> of <instance_name1> and <port_name2> of <instance_name2>. You get an error if the specified connection does not exist.
disconnect -adhoc <instance_name1>.<port_name1> [left:right] <instance_name2>. <port_name2> [left:right]	Disconnects the connection between <port_name1> of <instance_name1> and <port_name2> of <instance_name2>. If you specified [left:right] port ranges, only those port ranges are disconnected. If you do not specify a range, the entire port connection is disconnected. You get an error message if the specified connection does not exist.
disconnect -adhoc <instance_name1>.<port_name1> [left:right]	Disconnects all connections to <port_name1> of <instance_name1>. If you specify a [left:right] port range, only that port ranges is disconnected. If you do not specify a range, all port connections are disconnected.
disconnect -const_value <instance_name>.<port_name>	Removes the constant value attached to the port <port_name> of <instance_name>. You get an error if the port has another connection defined for it, or if the port is not tied to a constant value.

Command	Description
<code>disconnect -const_value <instance_name>.<port_name></code>	Removes the constant value attached to the port <code><port_name></code> of <code><instance_name></code> . You get an error if the port has another connection defined for it, or if the port is not tied to a constant value.
<code>disconnect -signal_export <instance_name>.<port_name></code>	Does not export the port <code><port_name></code> of <code><interface_name></code> to the top level. You get an error if there is no export connection defined for it.
<code>disconnect -signal_export <instance_name>.<port_name> [left:right]</code>	Does not export the port <code><port_name></code> of <code><interface_name></code> to the top level. You get an error if there is no export connection defined for it.

Design Commands

You can use the design Tcl commands to work with the design.

Command	Description
<code>design -add <component_name> [- iname <instance_name>]</code>	Adds components to the design. If you specify <code>-iname</code> , the tool adds the instance with the name <code><instance_name></code> . If you do not specify it, the tool uses the naming convention <code><component_name>_i<n></code> ; where <code>n</code> is a unique integer starting from 1. For example, instances of <code>leon2Proc</code> are added as <code>leon2Proc_i1</code> , <code>leon2Proc_i2</code> , etc.
<code>design -remove <instance_name> [-component <component_name>]</code>	Removes the instance from the design. When you specify <code>-component</code> , all instances of the specified component are removed.
<code>design -connect <instance_name> -interface <interface_name> -bus <bus_number></code>	Connects the interface <code><interface_name></code> in the instance <code><instance_name></code> to the bus whose index is specified by <code><bus_number></code> . The bus number is the index of the bus displayed in the Design view.

Command	Description
design -disconnect <instance_name> -interface <interface_name> -bus <bus_number>	Disconnects the interface <interface_name> in the instance <instance_name> from the bus whose index is specified by <bus_number>.
design -configure <instance_name> -interface <interface_name> { -name - description -baseaddress - highaddress -locked -used - expanded } <value>	Configures or modifies the information displayed in the Design view. You can specify the instance and interface names. The following determine what is displayed in the Design view: <ul style="list-style-type: none"> • -name specifies a new name for the instance • -description specifies a new description for the instance • -baseaddress and highaddress specify the addresses for the interfaces • -locked and used specify the status of these two properties
design -check	Checks the validity of the design

Output File Commands

Use the generate Tcl command to specify how output files are generated. See the following table for a description of the options for this command.

Command	Description
generate -project [-all]	Generates a project file for synthesis with Synplify Pro or Premier. If you specify -all, the tool generates Verilog netlist and wrappers, VHDL netlist and wrappers, and the synthesis project file.
generate -{vhdl verilog} -wrapper -instance <instance_name>	Generates a VHDL or Verilog wrapper for the specified instance.
generate -{vhdl verilog} {- wrapper netlist all}	Generates netlists or wrappers for VHDL or Verilog. If you specify all, the tool generates both netlists and wrappers in the format you specified.

Project Commands

The project Tcl command lets you set options for creating, saving and closing the System Designer project.

Command	Description
<code>project -new <project_file_name> -path <path> [-design <design_name>]</code>	Creates a new project file at the specified <i><path></i> . You can specify the design name in <i><design_name></i> . If you do not specify a name, the default name is the project name.
<code>project -save</code>	Saves the currently opened project.
<code>project -close [-nosave]</code>	Closes the currently opened project. If you specify <i>-nosave</i> , the tool closes the project without saving.

Preference Commands

Use the preference command to set preferences.

Command	Description
<code>preferences -get <preference_name> [-type <datatype>]</code>	Prints the value of the preference specified in <i><preference_name></i>
<code>preferences -set <preference_name> <preference_value> [-type <datatype>]</code>	Sets the value of the preference in <i><preference_name></i> .

You can have different values for *<preference_name>* and *<preference_value>*. The following describe the values for different types of preferences:

- [General Preferences, on page 123](#)
- [Logging Preferences, on page 124](#)
- [Configuration Preferences, on page 124](#)
- [Connectivity Preferences, on page 126](#)
- [Output File Generation Preferences, on page 127](#)

- [Design Checking Preferences, on page 127](#)
- [GUI Preferences, on page 128](#)

General Preferences

The following *<preference_name>* values for the preference Tcl command let you set general preferences.

Preference Name	Description	Value
SAVE_PREFS_IN_PROJ_DIR	Saves preferences in the project directory	true/false
ALLOW_LIBRARY_COMP_RELOAD	Allows non-instantiated library components to be reloaded	true/false
CLEAR_CONSOLE_ON_CLOSE_PROJECT	Clears console on Close Project	true/false
CLEAR_CONSOLE_ON_LOAD_LIBRARY	Clears console on Load Library	true/false
CLEAR_CONSOLE_ON_CHECK_LIBRARY	Clears console on Check Library	true/false
CLEAR_CONSOLE_ON_CHECK_DESIGN	Clears console on Check Design	true/false
CLEAR_CONSOLE_ON_GENERATE_FILES	Clears console on Generate Files	true/false
SUPPORT_SPIRIT_VERSION	SPIRIT IP-XACT version to support	spirit11 spirit12 spirit14

Logging Preferences

The following *<preference_name>* values for the preference Tcl command let you specify what is reported in the log file, Command shell, and Console windows.

Preference Name	Description	Values
LOGGING_LEVEL_OVERALL	Set overall (default) logging level	Use any of the following: SEVERE
LOGGING_LEVEL_FOR_FILE	Set logging level for the log file	WARNING
LOGGING_LEVEL_FOR_SHELL	Set logging level Command Shell window	INFO CONFIG
LOGGING_LEVEL_FOR_CONSOLE	Set logging level for Console view	FINE FINER FINEST

Configuration Preferences

The following *<preference_name>* values for the preference Tcl command let you set preferences for configuring the design.

Name	Description	Values
ALLOW_INVALID_COMPONENT_INSTANCE	Allows instantiation of non-usable components	true/false
ALLOW_RTL_MODEL_VIEW_GUESSING	Allows the tool to guessing if a model-view is synthesizable HDL	true/false
ALLOW_CONFIG_PARAMS_RESOLVE_USER	Allows the editing of parameters that have the <code>resolve=<non-user></code> property	0 - 3 (Resolve Property, on page 125)
AUTO_ADDRESS_MIN_BLOCK_SIZE	Sets minimum address-block size for auto-addressing	Integer (Address Block Size, on page 125)
INSTANCE_HDL_TYPE_IS_FLEXIBLE	Allows instance HDL value to be changed if no matching model-view is found	true/false
PROJECT_HDL_TYPE_IS_FLEXIBLE	Allows project HDL value to be changed if no matching model-view is found	true/false

Name	Description	Values
PROJECT_DEVICE_TYPE_FLEXIBILITY	Lets you choose a model view even if the last N parts of the device have mismatched information	0 - 5 (Model View Matching, on page 125)

Resolve Property

The different values of the resolve property are mapped to values from 0 -3 for ALLOW_CONFIG_PARAMS_RESOLVE_USER:

0,1	Allow editing parameters with resolve=immediate
2	Allow editing parameters with resolve=dependent
3	Allow editing parameters with resolve=generated

Address Block Size

The AUTO_ADDRESS_MIN_BLOCK_SIZE preference specifies the minimum size of blocks to be allocated when the tool auto-allocates addresses to memory maps. The range of the entered value is not validated. The basic rule is that this value must be a multiple of the minimum addressable units of the resultant system, and it is up to the user to ensure this.

Model View Matching

This specifies how much flexibility to allow when matching the FPGA device in model-view. Valid range is 0 to 5.

0	Select model view if vendor, technology, part, package & speed grade match
1	Select model view if vendor, technology, part & package match
2	Select model view if vendor, technology & part match
3	Select model view if vendor & technology match
4	Select model view if vendor match
5	Select model view even if none match

Connectivity Preferences

The following *<preference_name>* values for the preference Tcl command let you set general preferences.

Preference Name	Description	Values
IGNORE_SYSTEM_IFACE_GROUP_NAMES_FOR_CONN	Ignores group property on system interfaces while making connections	true/false
IGNORE_DIRECT_CONNECTION_PROP_FOR_CONN	Ignores directConnection property while making connections	true/false
IGNORE_ADDRESS_RANGE_MATCHING_FOR_CONN	Ignores address-range matching while making direct connections	true/false
IGNORE_BITS_IN_LAU_MATCHING_FOR_CONN	Ignores bits-in-LAU matching while making direct connections	true/false
AUTO_CONNECT_SETTING_LEVEL	Sets level of automation during auto-connection	0 - 3 (Auto-Connection Settings, on page 126)

Auto-Connection Settings

The values for the auto-connecting levels are listed here:

0	Do not auto-connect
1	Auto-connect, but do not auto-instantiate
2	Auto-connect and auto-instantiate everything, except for system interfaces
3	Auto-connect and auto-instantiate everything

Output File Generation Preferences

The following *<preference_name>* values for the preference Tcl command let you set preferences for generating output files.

Name	Description	Values
CHECK_DESIGN_BEFORE_GENERATION	Runs design-rule checking before generating output files	true/false
ADD_FILE_DEPENDENCY_TO_INCLUDE_PATHS	Adds paths in <code>spirit:dependency</code> under <code>spirit:file</code> to <code>include_path</code> list	true/false
ADD_FILESET_DEPENDENCY_TO_INCLUDE_PATHS	Adds paths in <code>spirit:dependency</code> under <code>spirit:fileSet</code> to <code>include_path</code> list	true/false

Design Checking Preferences

The following *<preference_name>* values for the preference Tcl command let you set preferences for design checks.

Name	Description	Values
CHECK_BUSDEF_SCHEMA_RULES	Check SPIRIT bus definitions against schema	true/false
CHECK_COMPONENT_SCHEMA_RULES	Checks SPIRIT components against schema	true/false
CHECK_ABSTRACTIONDEF_SCHEMA_RULES	Checks SPIRIT abstraction definitions against schema	true/false
CHECK_DESIGN_SCHEMA_RULES	Checks SPIRIT designs against schema	true/false
CHECK_BUSDEF_SEMANTIC_RULES	Checks SPIRIT bus definitions for semantic rules	true/false
CHECK_COMPONENT_SEMANTIC_RULES	Checks SPIRIT components for semantic rules	true/false
CHECK_ABSTRACTIONDEF_SEMANTIC_RULES	Check SPIRIT abstraction definitions for semantic rules	true/false
CHECK_BUSDEF_ASSEMBLY_RULES	Checks bus definitions for system assembly rules	true/false

Name	Description	Values
CHECK_COMPONENT_ASSEMBLY_RULES	Checks components for system assembly rules	true/false
CHECK_DESIGN_ASSEMBLY_RULES	Checks designs for system assembly rules	true/false
CHECK_BUSDEF_DIRECT_CONNECTION_VALID	Checks if direct connections are valid	true/false
CHECK_BUSDEF_SIGNAL_REQUIRES_DRIVER	Checks if signals require drivers	true/false

GUI Preferences

The following *<preference_name>* values for the preference Tcl command let you set preferences for GUI display.

Preference Name	Description	Values
SHOW_LIBRARY_VIEW_DESCRIPTION_COLUMN	Displays Description column in the Library view	true/false
SHOW_DESIGN_VIEW_LOCK_COLUMN	Displays Lock column in the Design view	true/false
SHOW_DESIGN_VIEW_DESCRIPTION_COLUMN	Displays Description column in the Design view	true/false
SHOW_DESIGN_VIEW_RANGE_COLUMN	Displays Address columns in the Design view	true/false
SHOW_DESIGN_VIEW_DIRECT_CONNS_BUS_COLUMN	Displays Direct Conns column in the Design view	true/false
SHOW_DESIGN_VIEW_SYSTEM_IFACES	Displays system interfaces in the Design view	true/false
SHOW_OUTLINE_VIEW_FULL_DESIGN	Displays Outline view independent of the Design view preferences	true/false
SHOW_OUTLINE_VIEW_SYSTEM_IFACES	Displays system interfaces in the Outline view	true/false
SHOW_OUTLINE_VIEW_HIER_CONNS	Displays Hier Ports section in the Outline view	true/false

Preference Name	Description	Values
SHOW_OUTLINE_VIEW_DIRECT_CONNS	Displays Direct Connection Interfaces in the Outline view	true/false
SHOW_OUTLINE_VIEW_MIRRORED_CONNS	Displays Mirrored Connection Interfaces in the Outline view	true/false
SHOW_OUTLINE_VIEW	Displays Outline view in the GUI	true/false

APPENDIX A

System Designer Tutorials

This chapter includes three ways to work through the System Designer design flow. They all follow the same basic methodology, but they vary in complexity. You can work through any or all of these tutorials:

Basic	Describes the basic methodology and minimum required to work through the System Designer design flow
Advanced	Starts with the Basic flow, but also includes some additional steps that illustrate how to use some additional options in the System Designer design flow
Expert	Starts with the Advanced flow, but uses a more complex design to illustrate other options in the System Designer flow.

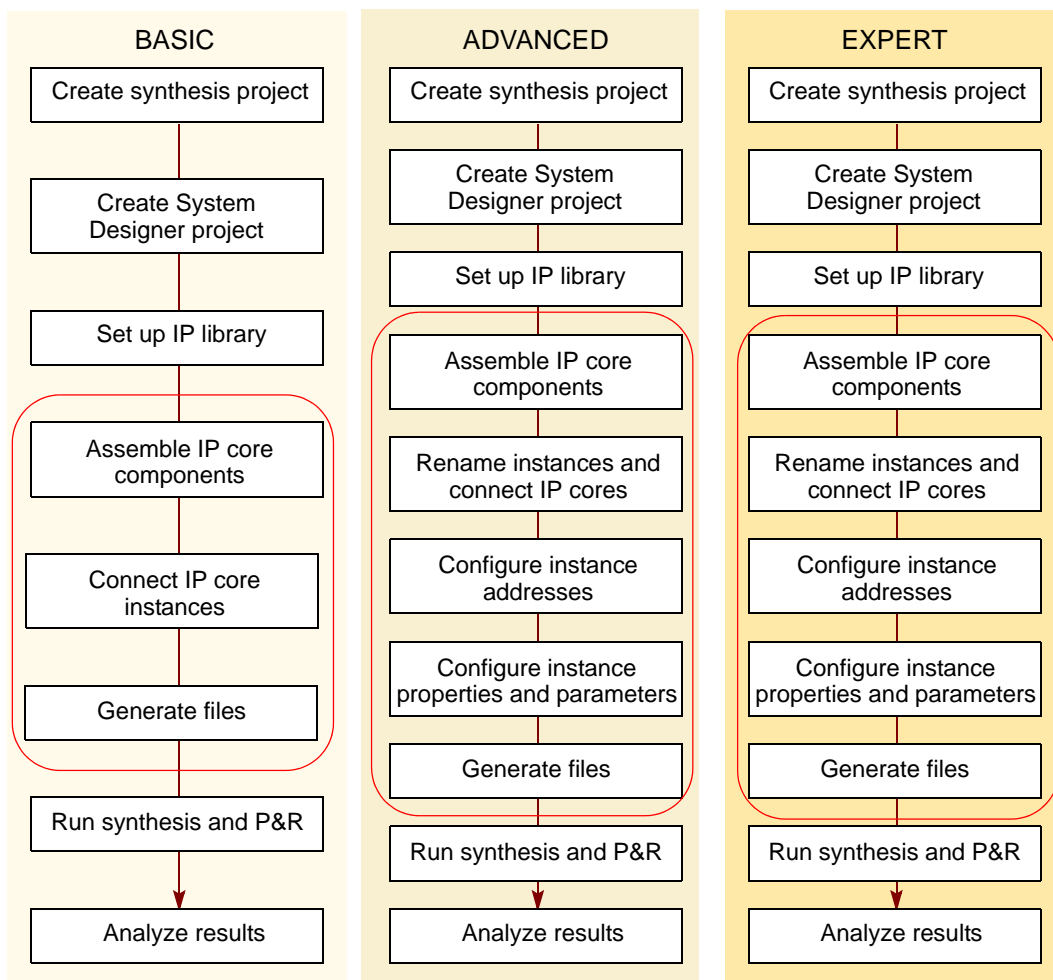
The following describe the design flows and step-by-step procedures for the tutorials. Unless specified otherwise, the steps apply to all three tutorial flows.


- [Tutorial Design Flows, on page 132](#)
- [Create a System Designer Project, on page 135](#)
- [Create a Synthesis Project, on page 134](#)
- [Set up the IP Library, on page 138](#)
- [Create an Embedded System \(Basic\), on page 141](#)
- [Create an Embedded System \(Advanced\), on page 144](#)
- [Create an Embedded System \(Expert\), on page 155](#)

- [Run Synthesis and Place and Route, on page 163](#)
- [Analyze the Design, on page 165](#)

Tutorial Design Flows

This figure summarizes the three System Designer tutorials. The examples are progressively more complex, with the Advanced and Expert tutorials elaborating and building on the Basic procedure. Note however that they all follow the same methodology. The rest of the chapter describes each step in detail, pointing out where steps are specific to the more complex examples. You can either follow the specifics to build any of the three examples, or use the procedures as a guide to building your own design.



 Create an embedded system

Create a Synthesis Project

The following steps guide you through the tutorial steps for creating a synthesis project in the FPGA synthesis tool.

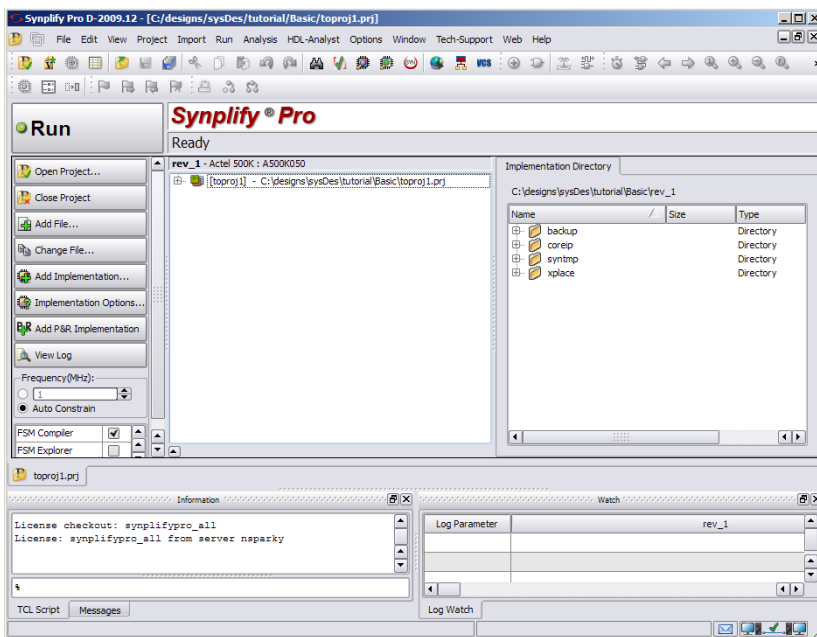
1. Start the Synplify Pro or Synplify Premier tool.
2. Click File -> New Project.

The tool creates a new project in the synthesis tool window.

3. Select File -> Save As and specify a project name. Use the name appropriate for the tutorial you are following:

Basic	toproj1
Advanced	toproj2
Expert	toproj3

- Click OK. This figure shows the Synplify Pro UI for the Basic tutorial.



The scope of the tutorial does not cover this, but if you were using a HAPS board for verification, you would specify it in the synthesis tool UI at this point. See [The System Designer-HAPS Board Design Flow, on page 52](#) for details about the procedure.

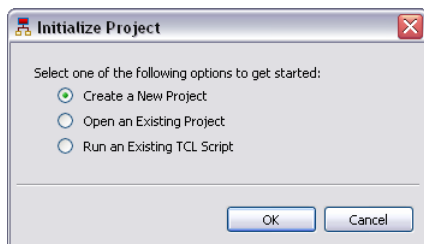
You can now create a System Designer project ([Create a System Designer Project, on page 135](#)).

Create a System Designer Project

After you have created a synthesis project, the next step in the tutorial is to create a System Designer project.

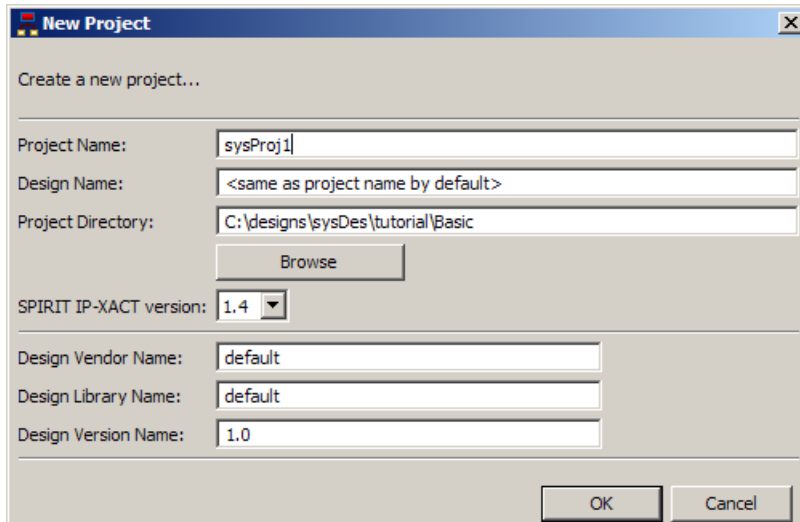
1. Start System Designer by selecting Import IP-> Launch System Designer or the System Designer icon () from the FPGA synthesis tool.

The System Designer UI opens with an initial dialog box.



2. In the dialog box that opens, select Create New Project.
3. In the New Project dialog box that opens, do the following:
 - Set Project Name and Design name as shown in this table:

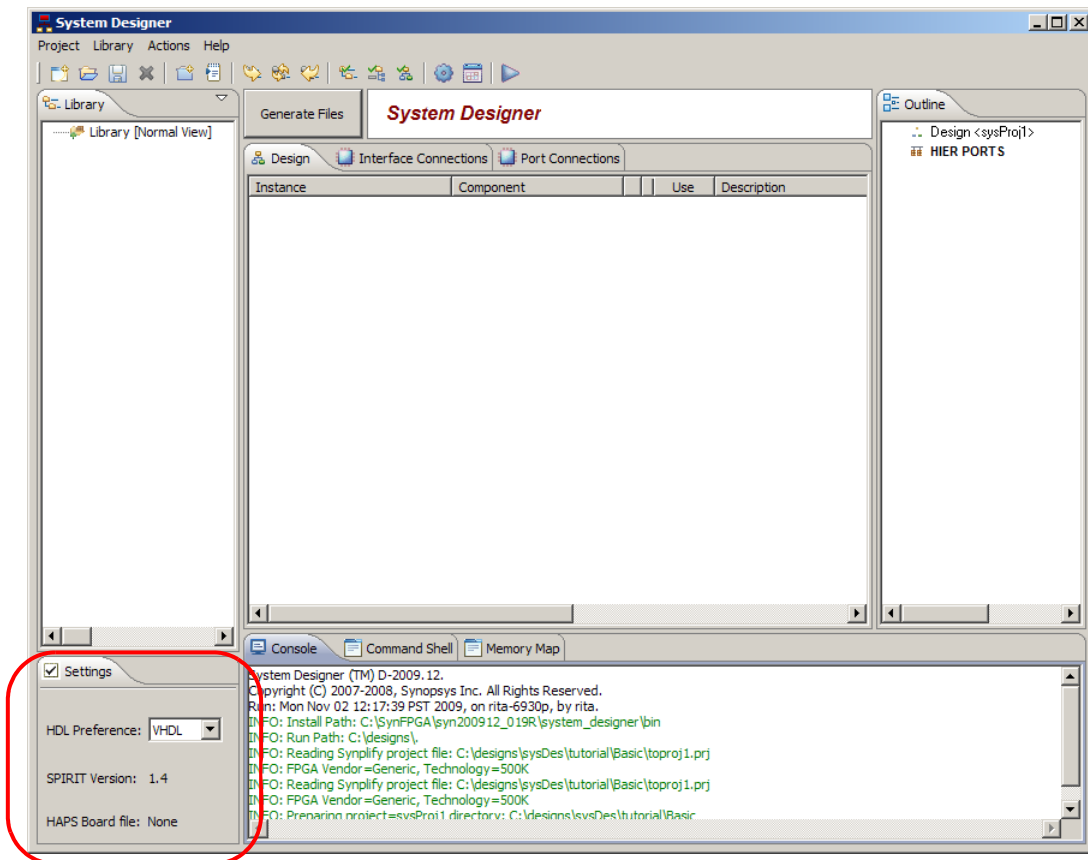
	Project Name	Design Name
Basic	sysproj1	sysdesign1
Advanced	sysproj2	sysdesign2
Expert	sysproj3	sysdesign3



- Set SPIRIT IP-XACT version to 1.4 for this tutorial.
 - If you want to change it, set the Project directory to *<your_results_directory>*, or use the Browse button to select a location. Otherwise leave it as is.
 - Leave the defaults for Design Vendor Name, Design Library Name, and Design Version Number.
 - Click OK.
4. Go to the Settings panel located in the lower left corner of the UI, and set HDL Preference to VHDL.

This sets the language default for the design.

Note that this panel also reflects the SPIRIT version you selected in the previous step.



5. Save the design.

After you have set up the synthesis project and the System Designer project you are ready to create an embedded system using the tool.

Set up the IP Library

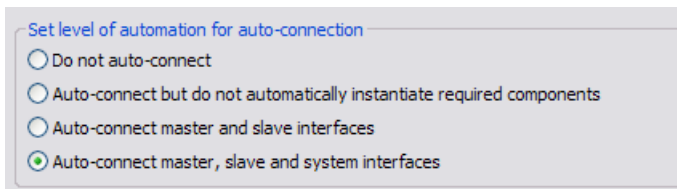
Do the following to set up the IP library:

- [Set Auto-Connection Preferences, on page 138](#)
- [Load the Library, on page 138](#)

Set Auto-Connection Preferences

For the purposes of these tutorials, you now set auto-connection preferences to ensure that connections are made automatically.

1. Select Actions->Preferences. From the menu on the left, select Connectivity->Flexibility.
2. Select Auto-connect master, slave and system interfaces in the Set level of automation for auto-connection section.



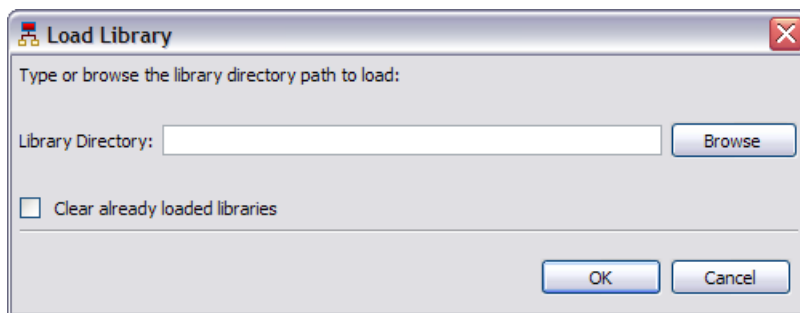
3. Click OK.

You can now load the library, as described in [Load the Library, on page 138](#).

Load the Library

To create an embedded system, you must load the appropriate library. The System Designer software includes example libraries for SPIRIT IP-XACT versions 1.2 and 1.4 . You use one of them in this tutorial.

1. Select Library -> Load IP Library or click the Load Library icon in the UI.



2. Click Browse, and select the appropriate library from the hierarchy and click OK.

For this tutorial, set it to `install_dir/system_designer/examples/ipcores/spirit1_4/Leon2`.

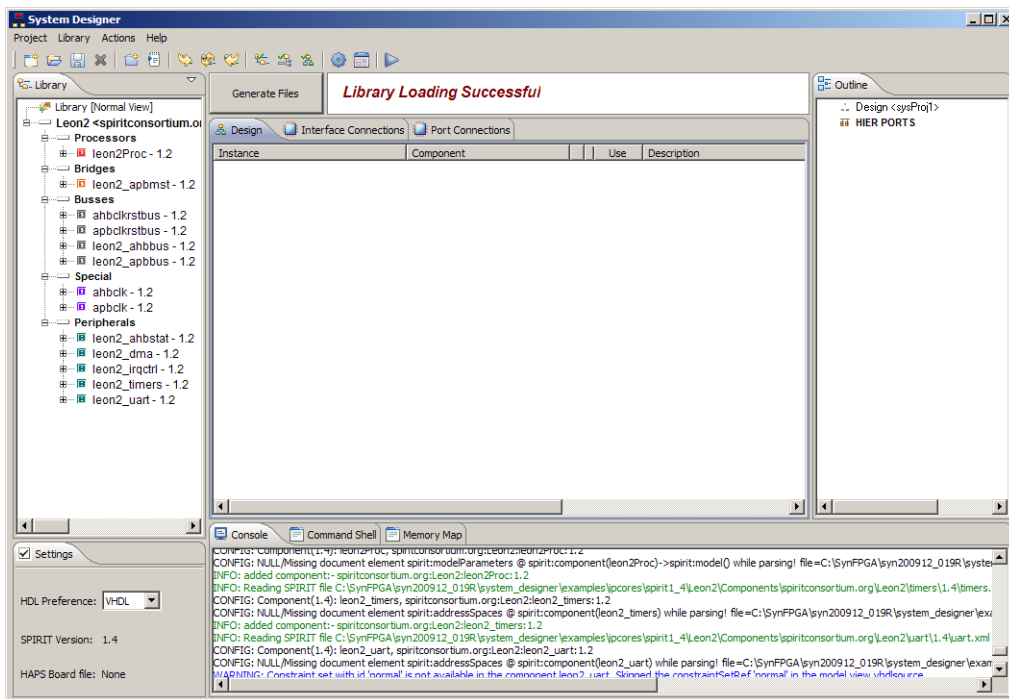
3. Click OK in the Load Library dialog box.

This loads the components and bus definitions from the appropriate locations in the `examples/ipcores` directory:

Components**Bus Definitions**

<code>spirit1_4/Leon2/Components</code>	<code>spirit1_4/Leon2/BusDefs</code>
---	--------------------------------------

This figure shows the Library view on the left side of the window populated with the components and buses from the library you loaded.

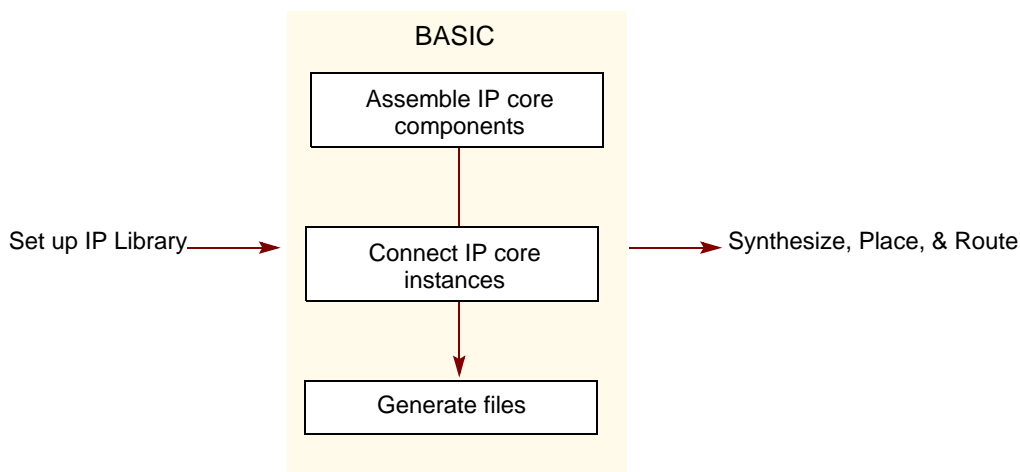


You are now ready to create the embedded system. Go to the procedure that corresponds to the tutorial you are following:

- [Create an Embedded System \(Basic\), on page 141](#)
- [Create an Embedded System \(Advanced\), on page 144](#)
- [Create an Embedded System \(Expert\), on page 155](#)

Create an Embedded System (Basic)

The following figure shows the steps required to create an embedded system for the Basic tutorial. This tutorial shows the minimum you must do to create an embedded system.



The following procedures provide details:

- [Assemble the IP Core Components \(Basic\), on page 141](#)
- [Connect the IP Core Components \(Basic\), on page 142](#)

Assemble the IP Core Components (Basic)

After you have loaded the component and bus definitions from the libraries, you can assemble the core components.

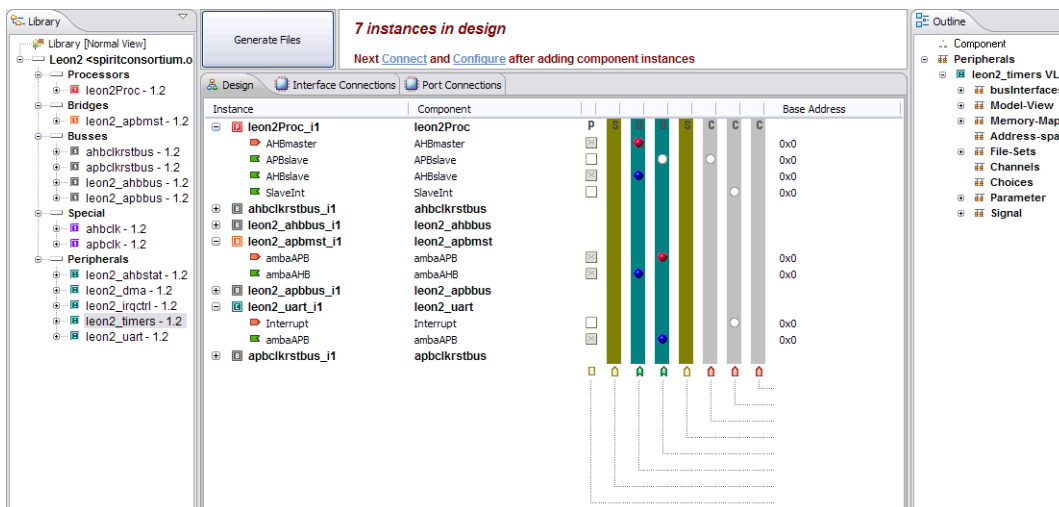
1. Instantiate the components by going to the Library view and double-clicking the following components. Alternatively you can drag and drop them from the Library view to the Design view:

Processors Leon2Proc 1.2

Bridges Leon2_apbmst 1.2

Peripherals Leon2_uart 1.2

The components are instantiated in the Design view. Buses are automatically instantiated. If you do not see the buses displayed, right-click in the Design view and enable Show Bus Components. The following figure shows the components. You can now connect the cores, as described in [Connect the IP Core Components \(Basic\)](#), on page 142.



Connect the IP Core Components (Basic)

This procedure shows you how to connect cores for the Basic tutorial example. In this tutorial, you only make bus interface connections. For information on making other connections, work through the Advanced or Expert tutorials

1. Check the bus interface connections that the tool made automatically.

The tool automatically connects some interfaces, and indicates them by filled colored bubbles. For a description of what the different bubbles mean, see [Bus Connections, on page 71](#). Empty white bubbles indicate unconnected interfaces. For instance, you see that the APBSlave interface of the Leon2_proc_i1 instance was not connected automatically.

Instance	Component										Base ...	Hig...	Lock	Use	Des
leon2Proc_i1	leon2Proc												!	✓	Pro
AHBmaster	AHBmaster										0x0	0x0	!		mas
APBslave	APBslave										0x0	0x0	!		slavi
AHBslave	AHBslave										0x0	0x0	!		slavi
SlaveInt	SlaveInt										0x0	0x0	!		slavi

2. Check the Description column for the APBslave interface.

- Scroll right to view the Description column, which is to the the right of the vertical interface bars. It lists the interface type and VLVN of the component or interface.
- Click in the description to scroll to the end of the description.
- You see that the ConnectionRequired property is set to false. With this setting, the tool does not automatically connect the interface. You must connect it manually.

3. Manually connect the interface.

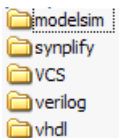
Click on the white bubble on the teal bar for bus 2 (Leon2_apbbus) to connect the APBslave interface to Leon2_apbbus_i1. This connects the Leon2_proc_i1, Leon2_apbmst_i1, and Leon2_uart_i1 instances with Leon2_ahpbus_i1 and Leon2_apbbus_i1.

The bubble changes color to indicate that the connection was made. It is now blue instead of white. See [Bus Connections, on page 71](#) for a description of the kinds of connections.

4. Save the project.

5. Generate the output files by selecting Actions-> Generate Files or clicking the Generate button.

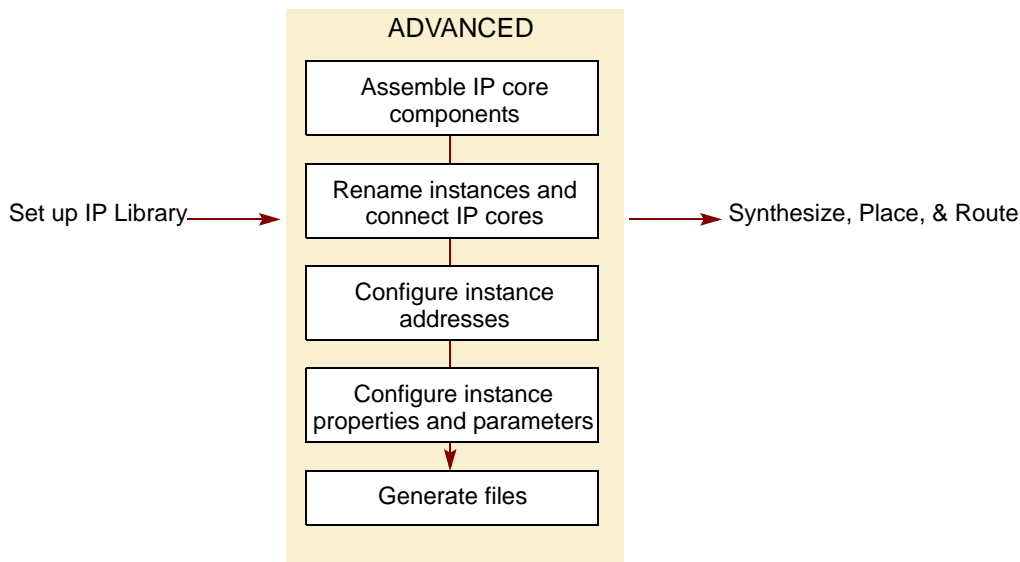
You see the following directories generated in the synthesis project directory:



You might see error messages in the Console view. This is because you did not make all the required connections in this tutorial. You can ignore them, and go on to synthesize the design. Go to [Run Synthesis and Place and Route](#), on [page 163](#) for the next step.

Create an Embedded System (Advanced)

The following figure shows the steps required to create an embedded system for the three tutorial examples. The Basic example shows the minimum you must do to create an embedded system; the additional steps in the Advanced and Expert examples demonstrate more options with more complex designs.



See the following for detailed procedures.

- [Assemble the IP Core Components \(Advanced\)](#), on page 145
- [Rename Instances \(Advanced\)](#), on page 146
- [Connect the IP Core Components \(Advanced\)](#), on page 147
- [Configure Instance Addresses \(Advanced\)](#), on page 152
- [Configure Properties and Parameters \(Advanced\)](#), on page 153

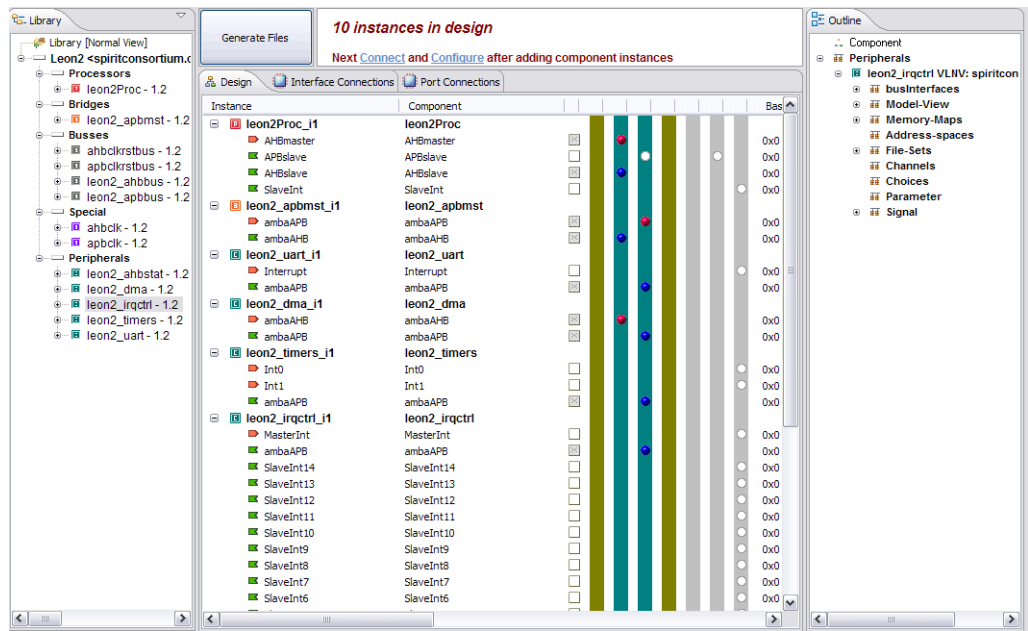
Assemble the IP Core Components (Advanced)

After you have loaded the component and bus definitions from the libraries, you can assemble the core components.

1. Instantiate the following components by double-clicking them in the Library view: Alternatively you can drag and drop them from the Library view to the Design view:

Processors	Leon2Proc 1.2
Bridges	Leon2_apbmst 1.2
Peripherals	Leon2_uart 1.2 Leon2_dma 1.2 Leon2_timers 1.2 Leon2_irqctrl 1.2






The Design view shows the instantiated components. Buses are instantiated automatically. If you do not see the buses displayed, right-click in the Design view and enable Show Bus Components. The next step in this tutorial is to rename the instances. See [Rename Instances \(Advanced\)](#), on page 146.



Rename Instances (Advanced)

This procedure shows you how to change the default names of the components you have instantiated. You might want to do this to simplify the instance names.

- 1. To rename an instance do the following:
 - In the Instance column, right-click the instance and select Rename Instance.
 - Enter the new name, then click OK. See the next step for details of the renaming.

Instance	Component
 proc	leon2Proc
 AHBmaster	AHBmaster
 APBslave	APBslave
 AHBslave	AHBslave
 SlaveInt	SlaveInt

2. Make the following name changes:

Change...	To...
ahbclkstbus_i1	ahbxt
apbclkstbus_i1	apbxt
leon2_ahbbus_i1	ahb
leon2_apbbus_i1	apb
leon2_apbmst_i1	bridge
leon2_dma_i1	dma
leon2_irqctrl_i1	irqctrl
leon2_timers_i1	timers1
leon2_uart_i1	uart1
leon2Proc_i1	proc

Once you have made the changes, you can connect the cores, as described in [Connect the IP Core Components \(Advanced\)](#), on page 147.

Connect the IP Core Components (Advanced)

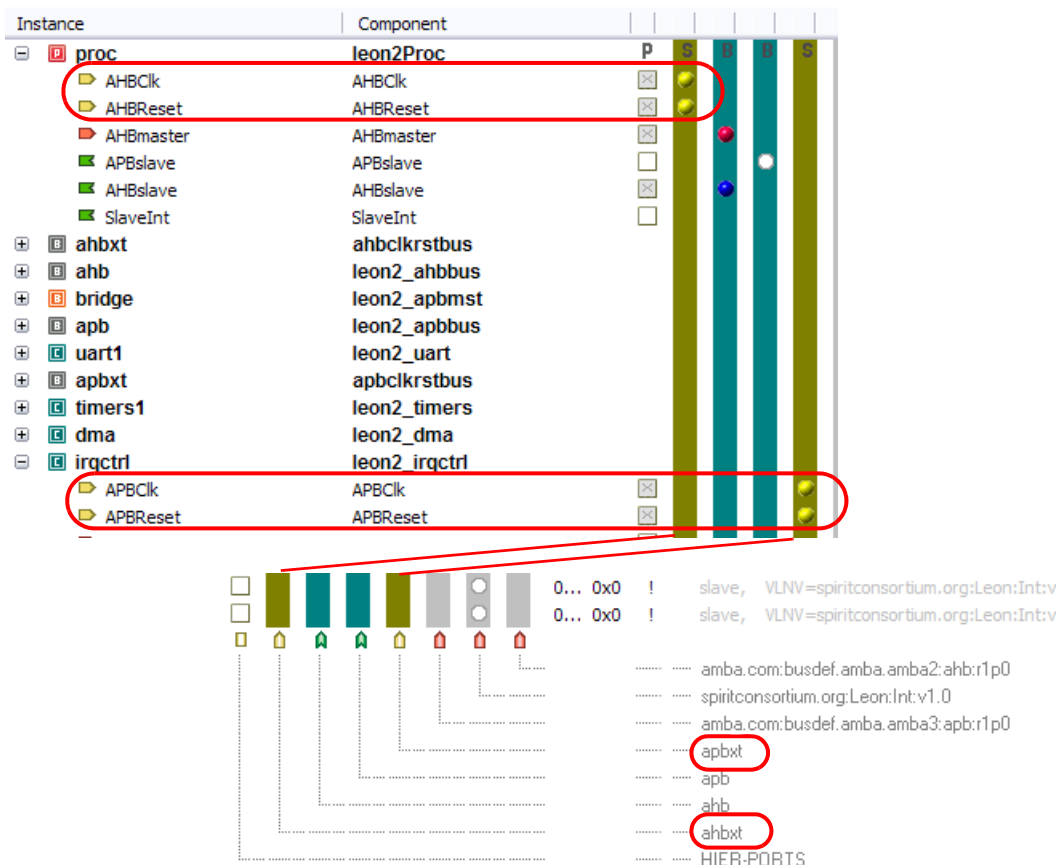
The following procedure shows you how to connect IP cores for the Advanced tutorial. It shows you how to make bus interface connections, system interface connections, and hierarchical connections from the Design view. The Expert tutorial shows you how to make these connections using a dialog box.

1. In the Advanced design, check the connections that the tool made automatically.

The tool automatically connects some interfaces, and indicates them by filled colored bubbles. For a description of what the different bubbles mean, see [Bus Connections](#), on page 71. Empty white bubbles indicate unconnected interfaces. For instance, you see that the APBSlave interface of the Leon2_proc_i1 instance was not connected automatically.

Instance	Component										Base Address
proc	leon2Proc										
AHBmaster	AHBmaster										0x0
APBslave	APBslave										0x0
AHBslave	AHBslave										0x0
APBslave	APBslave										0x0

2. Check the Description column for the APBslave interface.
 - Scroll to the Description column, to the right of the vertical interface bars. It lists the interface type and VLNV.
 - Click in the description to scroll to the end.
 - You see that the ConnectionRequired property is set to false. This is why the interface is not automatically connected.
3. Check the system interfaces to ensure that the clock and reset signals are properly connected.
 - If your design does not display the system interfaces, (yellow symbols under the components) display them by right-clicking in the Design view and selecting Show System Interfaces. The Design view now displays the system interface symbols. You must display the system interfaces before you can connect them.
 - Check that AHBCLK and AHBReset are connected to ahbxt. You see that the gold columns have yellow bubbles, indicating that they are connected.

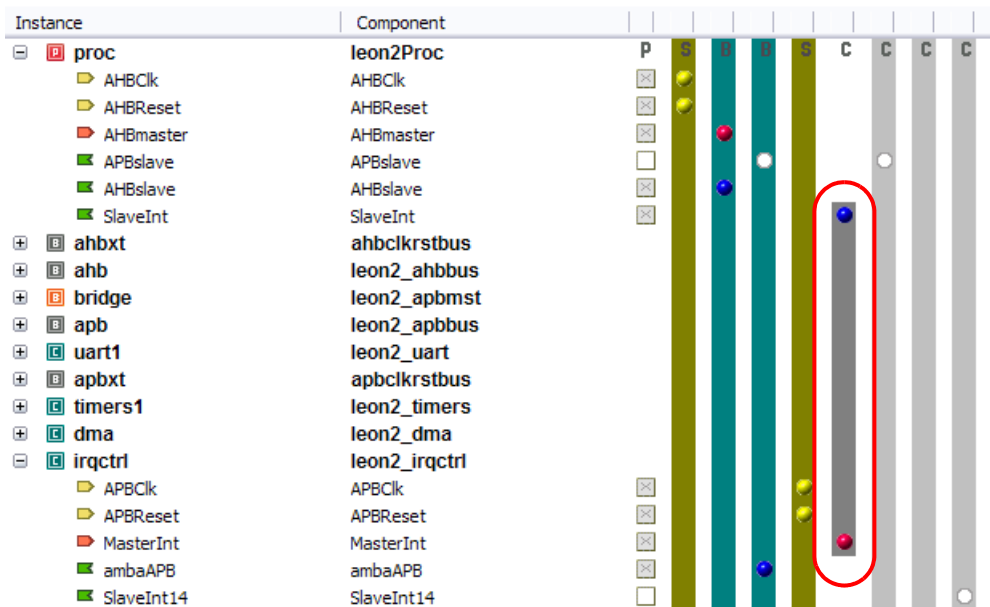


4. Make a direct connection from the interrupt interfaces of each component to the irqctrl instance.

Direct connections do not have to go through a bus. To make these connections, click a bubble next to the interrupt interface and the bubble next to the connecting irqctrl instance, as described below:

- In the grey direct connections column, click the white unconnected bubble for the SlaveInt interface of proc, and then click the white bubble

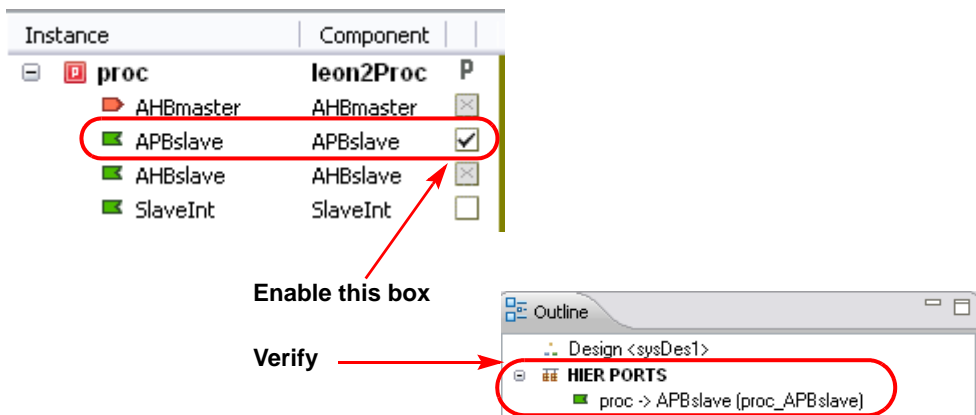
for the MasterInt interface of irqctrl in the same column. This makes a direct connection between the two, as shown below.



- Use the same technique to connect the Interrupt interface of uart1 to the SlaveInt0 slave interface of the irqctrl instance.
- Make a direct connection between the Int0 interface of the timers1 instance and the SlaveInt1 slave interface of the irqctrl instance.
- Make a direct connection between the Int1 interface of the timers1 instance and the SlaveInt2 slave interface of the irqctrl instance.

5. Make hierarchical connections.

- Make APBslave of proc a hierarchical connection by clicking on the corresponding check box in the P column.



- Use the same technique to make hierarchical connections for the following interfaces:

Instance	Interfaces
ahbxt	clkin
	rstin
apbxt	clkin
	rstin

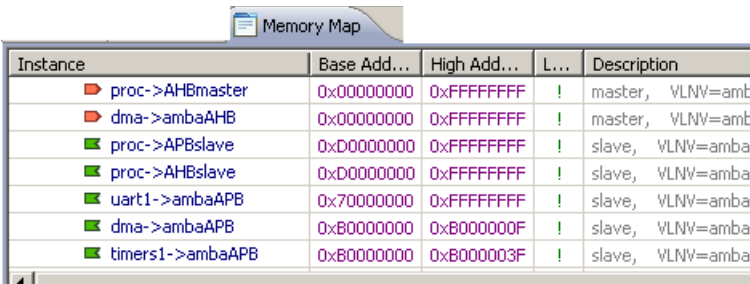
6. Verify that the connections were made by checking the Outline view.

Now that you have connected the cores in the Advanced design, the next step is to configure instance addresses, as described in [Configure Instance Addresses \(Advanced\)](#), on page 152.

Configure Instance Addresses (Advanced)

This procedure shows you how to modify instance addresses.

- 1. Click on the Memory Map tab to display the instance addresses.



Instance	Base Add...	High Add...	L...	Description
proc->AHBmaster	0x00000000	0xFFFFFFFF	!	master, VLNv=amt
dma->ambaAHB	0x00000000	0xFFFFFFFF	!	master, VLNv=amt
proc->APBslave	0xD0000000	0xFFFFFFFF	!	slave, VLNv=amba
proc->AHBslave	0xD0000000	0xFFFFFFFF	!	slave, VLNv=amba
uart1->ambaAPB	0x70000000	0xFFFFFFFF	!	slave, VLNv=amba
dma->ambaAPB	0xB0000000	0xB000000F	!	slave, VLNv=amba
timers1->ambaAPB	0xB0000000	0xB000003F	!	slave, VLNv=amba

- 2. Edit the addresses.
 - To edit an address, click on the appropriate Base Address or High Address column, and type in the new value.
 - For the Advanced tutorial, make the following changes:

proc: APBslave	Base Address: 0x10000000 High Address: 0x2FFFFFFFFF
uart1:ambaAPB	Base Address: 0x70000000 High Address: 0x8FFFFFFFFF
timers1: ambaAPB	Base Address: 0xB0000000 High Address: 0xCFFFFFFFFF

The next step is to configure instance parameters, as described in [Configure Instance Addresses \(Advanced\)](#), on page 152.

Configure Properties and Parameters (Advanced)

This procedure shows you how to configure the design to display instance properties and parameters.

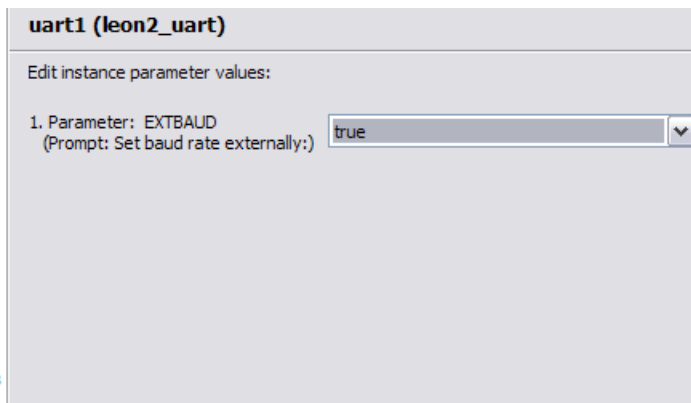
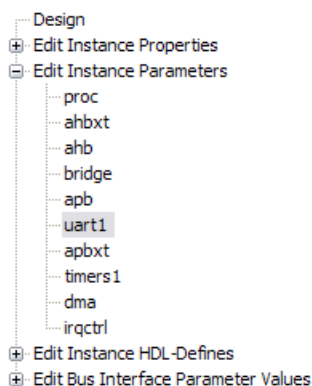
1. Select Actions -> Open Configuration.
2. View the following default settings:
 - Design Name: mydesign2. You can choose to change the design name.
 - Design Version: 1.0
 - Design Vendor Name: default
 - Design Library Name: default
3. Expand Edit Instance Properties, select an instance from the list and view the default settings.
 - Model-View: <AUTO>
 - HDL (Hint): <AUTO>

This tab lets you change parameter values. If the default value is <AUTO>, the tool automatically chooses the property.

4. Expand Edit Instance Parameters and edit some of the configurable parameters.

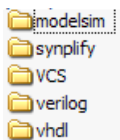
Configurable parameters are specified in the SPIRIT IP-XACT component files. For this tutorial, you modify some of the values specified by the IP vendor.

- Select instance `uart1` and set Parameter `EXTBAUD` to `true`. Click Apply.



- Select instance timers1 and set Parameter TPRESK to 100. Click Apply.
 - Click OK.
5. Select Project -> Save Project or click on the Save Project icon in GUI.
 6. Select Actions-> Generate Files or click on the Generate Files icon in the GUI.

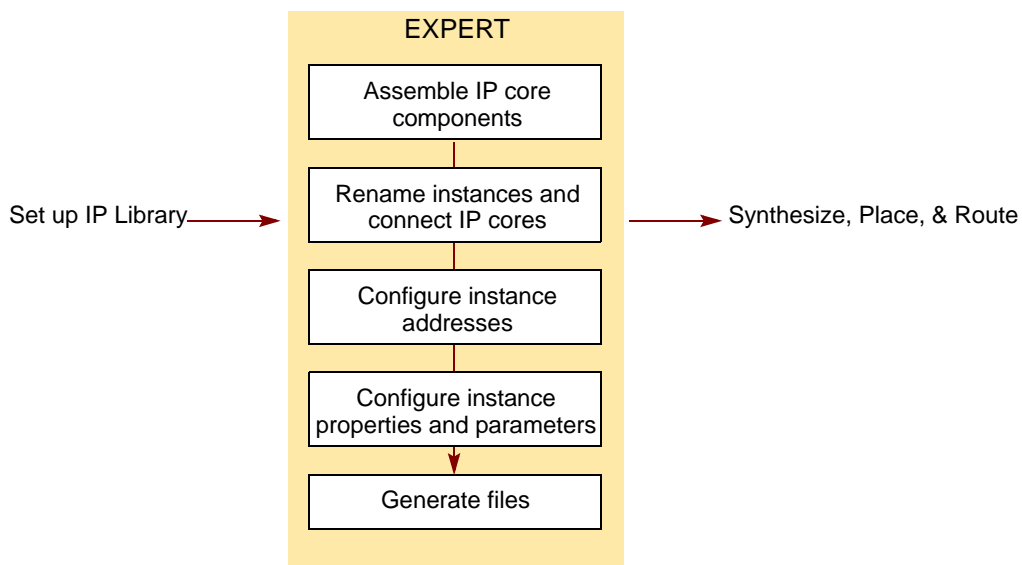
You see the following directories generated in the synthesis project directory:



You might see some design rule check errors reported in the Console window. This is because not all the interfaces were connected. You can ignore the messages and go on to synthesize your design. See [Run Synthesis and Place and Route, on page 163](#) for the next step.

Create an Embedded System (Expert)

The following figure shows the steps required to create an embedded system for the three tutorial examples. The Basic example shows the minimum you must do to create an embedded system; the additional steps in the Advanced and Expert examples demonstrate more options with more complex designs.



See the following for detailed procedures.

- [Assemble the IP Core Components \(Expert\), on page 156](#)
- [Rename Instances \(Expert\), on page 157](#)
- [Connect the IP Core Components \(Expert\), on page 159](#)
- [Configure Instance Addresses \(Expert\), on page 161](#)
- [Configure Properties and Parameters \(Expert\), on page 162](#)

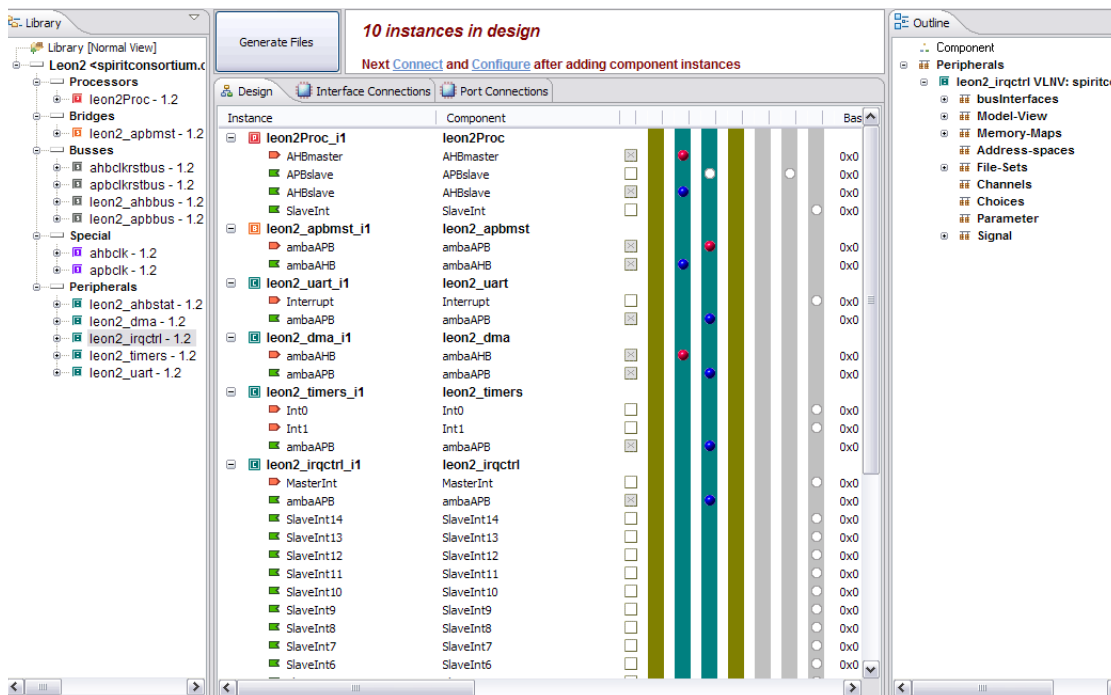
Assemble the IP Core Components (Expert)

After you have loaded the component and bus definitions from the libraries, you can assemble the core components.

1. Instantiate the following components by double-clicking them in the Library view:

Processors	Leon2Proc 1.2
Bridges	Leon2_apbmst 1.2
Peripherals	Leon2_uart 1.2 Leon2_dma 1.2 Leon2_timers 1.2 Leon2_irqctrl 1.2








The Design view shows the instantiated components. Buses are instantiated automatically. The next step in this tutorial is to rename the instances. See [Rename Instances \(Expert\)](#), on page 157.



Rename Instances (Expert)

This procedure shows you how to change the default names of the components you have instantiated. For example, you might want to do to simplify the instance names.

- 1. To rename an instance do the following:
 - In the Instance column, right-click the instance and select Rename Instance.
 - Enter the new name, then click OK. See the next step for details.

Instance	Component
 proc	leon2Proc
 AHBmaster	AHBmaster
 APBslave	APBslave
 AHBslave	AHBslave
 SlaveInt	SlaveInt
 ahbxt	ahbclkrstbus
 clk_in	clk_in

- 2. Make the following name changes:

Change...	To...
leon2Proc_i1	proc
leon2_ahbbus_i1	ahb
leon2_apbbus_i1	apb
leon2_apbmst_i1	bridge
leon2_uart_i1	uart1
leon2_dma_i1	dma
leon2_irqctrl_i1	irqctrl
leon2_timers_i1	timers1
apbclkrstbus_i1	apbxt
ahbclkrstbus_i1	ahbxt

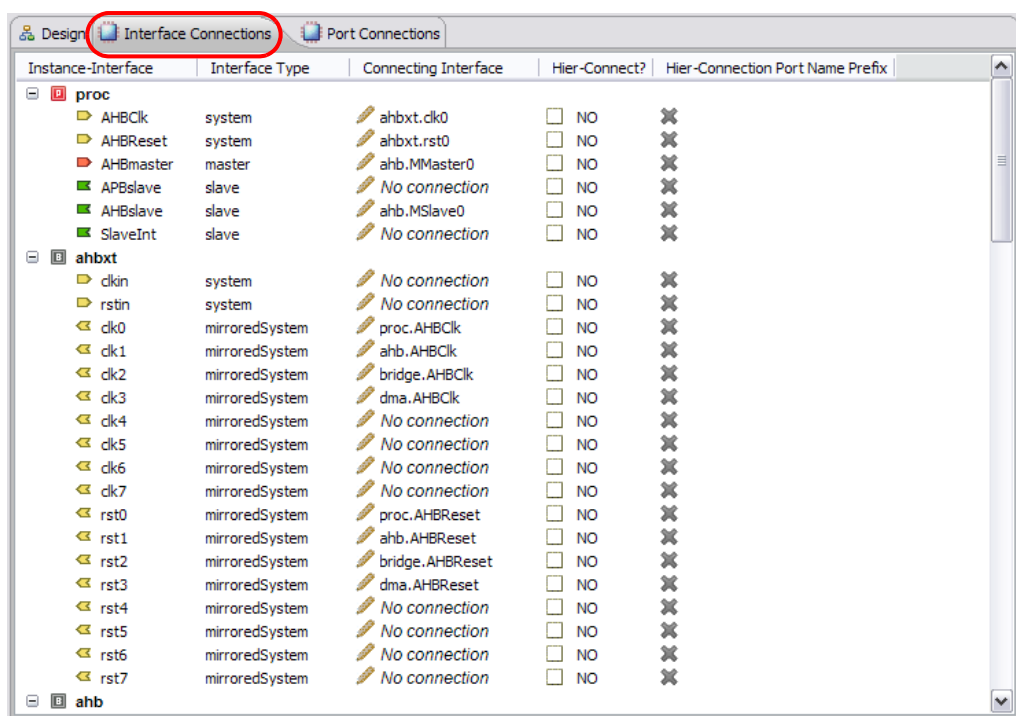
Once you have made the changes, you can connect the cores. Go on to [Connect the IP Core Components \(Expert\)](#), on page 159.

Connect the IP Core Components (Expert)

The Expert tutorial shows you how to connect the IP cores using the Interface Connections tab instead of the graphic display in the Design view. This differs from the other two tutorials. The Expert tutorial makes bus interface, system interface and hierarchical connections using this tab.

1. Click the Interface Connections tab in the Design view.

This tab displays the components and the interface connections. When you check the Connecting Interface column, you see unconnected interfaces classified as No connection.



2. Make direct connections between instances.
 - Click in the Connecting Interface column for the SlaveInt interface of the proc instance, which is currently listed as No connection. An arrow appears for a pull-down menu.

- Select irqCtrl.MasterInt from the pull-down menu of available connections. The tool shows the new connection in the Connecting Interface column.

Instance-Interface	Interface Type	Connecting Interface	Hier-Connect?	Hier
proc				
AHBClk	system	ahbxt.clk0	<input type="checkbox"/> NO	✕
AHBReset	system	ahbxt.rst0	<input type="checkbox"/> NO	✕
AHBmaster	master	ahb.MMaster0	<input type="checkbox"/> NO	✕
APBslave	slave	No connection	<input type="checkbox"/> NO	✕
AHBslave	slave	ahb.MSlave0	<input type="checkbox"/> NO	✕
SlaveInt	slave	irqctrl.MasterInt	<input type="checkbox"/> NO	✕
ahbxt				
clkIn	system	No connection	<input type="checkbox"/> NO	✕

- Use the same technique to connect the Interrupt interface of uart1 to irqctrl.SlaveInt0 (SlaveInt0 interface of irqctrl).
 - Make another direct connection from the Int0 interface of timers1 to irqctrl.SlaveInt2 (SlaveInt2 interface of irqctrl).
 - Make one more direct connection from the Int1 interface of timers1 to irqctrl.SlaveInt2 (SlaveInt3 interface of irqctrl).
3. Make hierarchical connections.
 - Go to the clkIn interface for ahbxt and check the box in the Hier-Connect column. It changes from No to Yes, and the default prefix (`<instance_name>.<interface_name>`) appears in the Hier-Connection Port Name Prefix column. This prefix is used for the port names at the top level.
 - Click in the Hier-Connection Port Name Prefix column and change the prefix to ahbclk.
 - Repeat the previous steps to make a hierarchical connection for the rstIn interface of ahbxt. Change the port prefix to ahbrst.
 - Make a hierarchical connection for the clkIn interface of apbxt. Rename the port prefix apbclk.
 - Make a hierarchical connection for the rstIn interface of apbxt. Rename the port prefix apbrst.

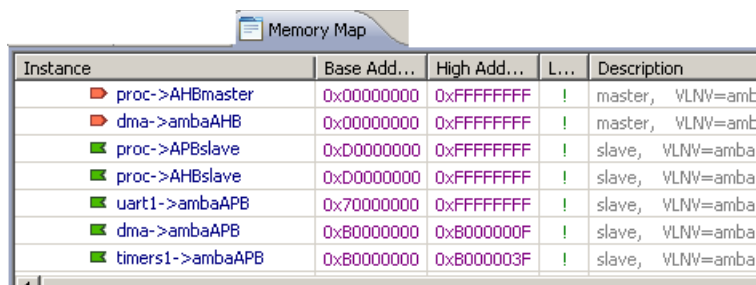
4. Save the design.

You can now configure instance addresses, as described in [Configure Instance Addresses \(Expert\)](#), on page 161.

Configure Instance Addresses (Expert)

This procedure shows you how to modify instance addresses.

1. Click on the Memory Map tab to display the instance addresses.



Instance	Base Add...	High Add...	L...	Description
proc->AHBmaster	0x00000000	0xFFFFFFFF	!	master, VLNV=amt
dma->ambaAHB	0x00000000	0xFFFFFFFF	!	master, VLNV=amt
proc->APBslave	0xD0000000	0xFFFFFFFF	!	slave, VLNV=amba
proc->AHBslave	0xD0000000	0xFFFFFFFF	!	slave, VLNV=amba
uart1->ambaAPB	0x70000000	0xFFFFFFFF	!	slave, VLNV=amba
dma->ambaAPB	0xB0000000	0xB000000F	!	slave, VLNV=amba
timers1->ambaAPB	0xB0000000	0xB000003F	!	slave, VLNV=amba

2. Edit the addresses.

- To edit an address, click on the appropriate Base Address or High Address column, and type in the new value.
- For the Advanced tutorial, make the following changes:

proc: APBslave	Base Address: 0x1
	High Address: 0x2

bridge:ambaAHB	Base Address: 0x000
	High Address: 0x0FF

uart1:ambaAPB	Base Address: 0x100
	High Address: 0x1FF

timers1: ambaAPB	Base Address: 0x300
	High Address: 0x3FF

The next step in the Expert tutorial is to configure instance parameters, as described in [Configure Properties and Parameters \(Expert\)](#), on page 162.

Configure Properties and Parameters (Expert)

This procedure shows you how to configure the design to display instance properties and parameters.

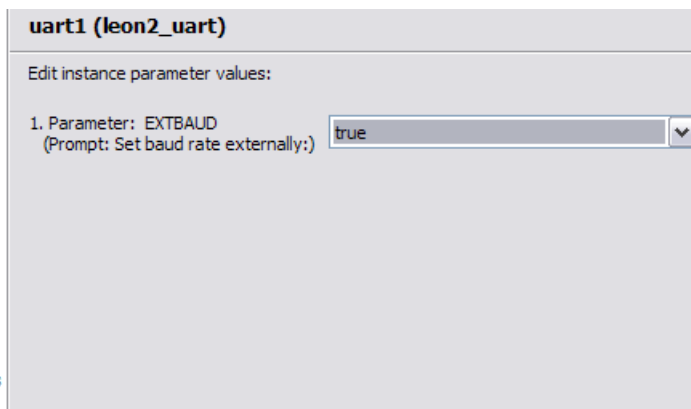
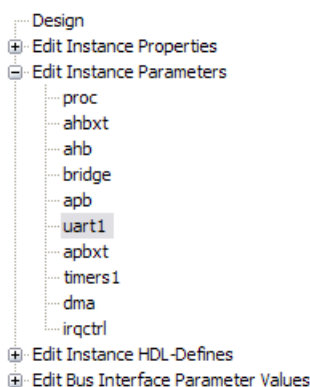
1. Select Actions -> Open Design Configuration.
2. View the following default settings:
 - Design Name: mydesign3. You can choose to change the design name.
 - Design Version: 1.0
 - Design Vendor Name: default
 - Design Library Name: default
3. Click Edit Instance Properties and view the default settings.
 - Model-View: <AUTO>
 - HDL (Hint): <AUTO>

This tab lets you change parameter values. If the default value is <AUTO>, the tool automatically chooses the property.

4. Click Edit Instance Parameters and edit some of the configurable parameters.

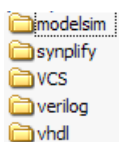
Configurable parameters are specified in the SPIRIT IP-XACT component files. For this tutorial, you modify some of the values specified by the IP vendor.

- Select instance `uart1` and set Parameter `EXTBAUD` to `true`. Click Apply.



- Select instance timers1 and set Parameter TPRESO to 100. Click Apply.
 - Click OK.
5. Select Project -> Save Project or click on the Save Project icon in GUI.
 6. Select Actions-> Generate Files or click on the Generate Files icon in the GUI.

At this point you have completed your embedded design and can go on to synthesis. See [Run Synthesis and Place and Route, on page 163](#) for the next step. You see the following directories generated in the synthesis project directory.



Run Synthesis and Place and Route

This section shows you how to synthesize your design and then run place and route. The synthesis tool can be either Synplify Pro or Synplify Premier.

1. Make sure you have generated the output files in System Designer.

When you select Actions->Generate Files, the System Designer tool updates the synthesis project file you created originally (see [Create a Synthesis Project, on page 134](#)).

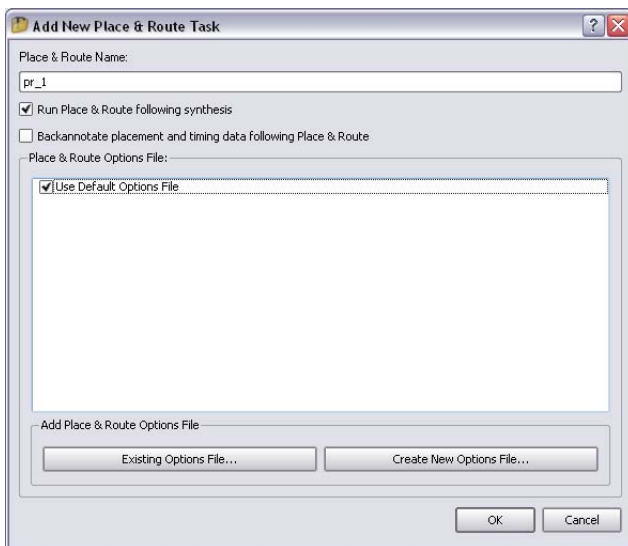
2. Go back to the Synplify Pro or Synplify Premier UI from which you invoked the System Designer tool.
 - You are prompted to reload the synthesis project you created earlier. Depending on which tutorial you are following, the name of the project will differ:

Basic	toproj1.prj
Advanced	toproj2.prj
Expert	toproj3.prj

- Click Yes to reload the project.

The synthesis project opens with the System Designer design. The synthesis tool uses the information in the `synthesis.slib` file that is generated by System Designer. You can now synthesize the design as usual.

3. Set up a P&R implementation to run after synthesis.
 - Set up your place and route tool to run automatically from the synthesis tool.
 - Either right-click the current implementation and select Add P&R Job, or just click the Add P&R implementation button in the project UI.
 - In the dialog box that opens, enable Run Place & Route After Synthesis. This option runs P&R automatically after synthesis is complete.



- Optionally, specify another name for the implementation or an options file for placement and routing.
 - Click OK. This creates the place and route implementation for the System Designer design, and specifies an automatic P&R run after synthesis.
4. Click the Run button to synthesize the design and automatically run place-and-route after synthesis is complete.

Once it is done, you can check your results and analyze them, as described in [Analyze the Design, on page 165](#).

Analyze the Design

To analyze your design, you can check it in either the synthesis tool or return to System Designer, depending on what errors you find. For the purposes of the tutorials, you will check both the synthesis tool and System Designer.

1. To check the design in the synthesis tool, do the following:
 - In the synthesis tool, open the project, and check the Technology view (HDL Analyst->Technology View). For the Basic and Advanced tutorials, you might see that some of the instances are optimized away. This is because these tutorials did not completely connect and configure the design.
 - You can also check the log file and the Message window for details of the run.
2. To review the design in System Designer, do the following:
 - Open System Designer, and select File -> Open Project or click the Open Project icon in the GUI.
 - Browse to the appropriate project file and click Open. Depending on which tutorial you are following, the name of the project will differ:

Basic	sysproj1.prx
Advanced	sysproj2.prx
Expert	sysproj3.prx

Index

Symbols

.do file [49](#)
.sdc file [94](#)
.vb file, specifying [53](#)

A

About System Designer command [112](#)
addresses
 configuring automatically [47](#)
Auto Connect Addresses command [86](#)
Auto Connect and Configure
 command [86](#)
auto-connection [45](#)
 configuring addresses [47](#)
 connecting cores [46](#)
 preferences [45](#)
auto-connection preferences [138](#)

B

board files. *See* HAPS boards
bus connections
 Design view symbols [71](#)
buses
 Tcl connect command [117, 122](#)

C

Check IP Library Schema command [86](#)
Clear IP Library command [84](#)
Close Project command [81](#)
Command Shell window
 commands [113](#)
 logging level [99](#)
Command shell window [64](#)
commands
 popup [112](#)

components
 assembling [28, 31](#)
 Design view descriptions [69](#)
 Tcl connect command [117, 122](#)
connect Tcl command [117, 122](#)
connections
 Design view symbols [71](#)
 using Interface Connections (Expert
 tutorial) [159](#)
Console window [64](#)
 commands [113](#)
 logging level [99](#)
 preferences [24](#)
constants for port signals [43](#)
constraints
 clock driver [92](#)
 hierarchical ports [95](#)
 input/output [93](#)
 mapping to .sdc file [94](#)
constraints, checking [48](#)
cores
 configuring [33](#)

D

design flow [20](#)
design Tcl command [120](#)
Design view
 component display [69](#)
 direct connections [37](#)
 disconnecting ports [40](#)
 display [67](#)
 hierarchical ports (interface level) [39](#)
 interface display [70](#)
 popup commands [113](#)
 preferences [110](#)
designs
 analyzing in synthesis tool [165](#)
 placing and routing [50](#)
 reopening in System Designer [165](#)
 synthesizing [50](#)

- Tcl design command [120](#)
- direct connections
 - Design view [37](#)
 - in the Design view (Advanced) [149](#)
 - Interface Connections view [38](#)
 - port-level [41](#)

E

- Edit Port Connection command [76](#)
- Exit command [81](#)

F

- flows
 - System Designer [20](#)

G

- Generate Files command [87](#)
- generate Tcl command [121](#)

H

- HAPS boards
 - daughter cards [54](#)
 - design flow [52](#)
 - supported boards [53](#)
 - using [52](#)
- HDL files [49](#)
- HDL format, setting [24](#)
- help [15](#)
- Help Contents command [112](#)
- hierarchical connections [43](#)
 - in the Design view (Advanced) [150](#)
 - interface [72](#)
 - port-level [42](#)
 - using Interface Connections (Expert tutorial) [160](#)
- hierarchical ports
 - constraints [95](#)

I

- Identify, introduction [15](#)
- Initialize Project command [82](#)
- instance addresses, configuring [152](#)

- instances
 - Design view descriptions [69](#)
 - editing bus paramaters [35](#)
 - editing parameters [34](#)
 - editing properties [34](#)
 - renaming [146](#)
 - Tcl connect command [117, 122](#)

- Interface Connections view
 - direct connections [38](#)
 - disconnecting ports [40](#)
 - hierarchical ports (interface level) [39](#)

- interfaces
 - definition [35](#)
 - Design view symbols [70](#)
 - disconnecting [40](#)

- IP
 - evaluating directly from the synthesis tool [30](#)

- IP-XACT
 - constraints [48](#)
 - mapping constraints to Synplify Pro constraints [92](#)
 - schema [26](#)
 - semantic rules [26](#)
 - support [10](#)
 - supported versions [10](#)
 - system assembly rules [26](#)
 - version [138](#)

L

- libraries
 - clearing [84](#)
 - downloading evaluation IP [30](#)
 - loading [28](#)
 - Tcl command [117](#)
- Library menu commands [84](#)
- library Tcl command [117](#)
- Library view [63](#)
 - preferences [109](#)
- License Agreement command [112](#)
- Load IP Library command [84](#)
- log file [99](#)
- logging levels [99](#)

M

Memory Map window 65
 commands 114

Modelsim 50

N

New Project command 81

O

Online Documents command 112

Open Design Configuration
 command 87

Open Design Constraints command 91

Open Project command 81

Outline view 63
 preferences 111

output files
 generating 49
 preferences 105
 Tcl generate command 121

P

popup commands 112

Port Connections view
 connecting ports 41
 hierarchical ports 42

port-level connections 40

ports
 connecting at the port level 40
 constant value 43
 disconnecting 44

preference files 27

preferences
 configuration 102
 configuration formatting 104
 configuration, Tcl command 124
 connectivity 100
 connectivity, Tcl command 126
 Console 24
 design checking, Tcl command 127
 general 97
 general, Tcl command 123, 128
 GUI 109

logging 98
 logging, Tcl command 124
 output file generation, Tcl
 command 127
 output files 105
 validation checks 26, 106
 views 25

Preferences command 95

project files
 Tcl generate command 121
 Tcl project command 122

Project menu commands 81, 86

project Tcl command 122

projects
 closing 23
 creating 21
 opening 21
 saving 23
 setting global options 24

R

ReadyIP design flow 12

Refresh IP Library command 84

Run Tcl Script command 81

S

Save Project command 81

Settings panel 65

simulation 50

SPIRIT
 schema preferences 106
 semantic rules 107
 system assembly rules, 108

symbols
 bus connections 71
 Design view display 69

Symphony HLS, introduction 15

Synplify Premier
 documentation 16
 flow with System Designer 12
 introduction 14

Synplify Pro
 documentation 16
 flow with System Designer 12
 introduction 14

- Synplify Pro/Premier define_clock constraints [92](#)
- Synplify Pro/Premier define_input_delay constraint [93](#)
- Synplify Pro/Premier define_output_delay constraint [93](#)
- synthesis projects
 - for System Designer [21](#)
- synthesis.slib [49](#)
- System Designer
 - introduction [10](#)
- system interface connections
 - Design view (Advanced) [148](#)

T

- Tcl
 - commands [116](#)
 - connectivity commands [117, 122](#)
 - design commands [120](#)
 - generate command [121](#)
 - library commands [117](#)
 - project commands [122](#)
 - syntax conventions [116](#)
- Tcl scripts, running [49](#)
- The SPIRIT Consortium [10](#)
- top-level connections [43](#)
- trace assignment file [58](#)
- Trace Assignment view
 - description [78](#)
- Trace Assignments view
 - using [57](#)
- traces, assigning [57](#)
 - rules [58](#)
- tutorials
 - assembling components (Advanced) [145](#)
 - assembling components (Expert) [156](#)
 - assembling IP cores (Basic) [141](#)
 - autoconnection [138](#)
 - configuring instance addresses (Advanced) [152](#)
 - configuring instance addresses (Expert) [161](#)
 - configuring properties (Advanced) [153](#)
 - configuring properties (Expert) [162](#)

- connecting cores (Advanced) [147](#)
- connecting cores (Expert) [159](#)
- connecting IP (Basic) [142](#)
- creating embedded systems (Advanced) [144](#)
- creating embedded systems (Basic) [141](#)
- creating embedded systems (Expert) [155](#)
- creating synthesis project [134](#)
- creating System Designer project [135](#)
- differences [132](#)
- flows [132](#)
- loading libraries [138](#)
- place and route [163](#)
- renaming instances (Advanced) [146](#)
- renaming instances (Expert) [157](#)
- selecting IP version [138](#)
- synthesis [163](#)

U

- user interface [62](#)

V

- VCS [50](#)
 - introduction [14](#)
- vcs.ksh [49](#)
- Verilog
 - preferences [105](#)
- verilog
 - Tcl generate command [121](#)
- vhdl
 - Tcl generate command [121](#)
- views
 - preferences [25](#)