



# Libero® SoC v2021.1

## Design Flow User Guide for PolarFire®

### Introduction

The Microchip Libero® System-on-Chip (SoC) design suite offers high productivity with its comprehensive, easy to learn, easy to adopt development tools for designing Resolved Microchip's power efficient flash [field-programmable gate arrays \(FPGAs\)](#), [SoC FPGAs](#), and [Rad-Tolerant FPGAs](#). The suite integrates industry-standard Synopsys [SynplifyPro®](#) synthesis and Mentor Graphics [ModelSim®](#) simulation with best-in-class constraints management, debug capabilities, and secure production programming support.

#### Supported Device Family

The following table lists the family of devices that Libero SoC supports. This guide covers all of these device families.

**Table 1. Device Family Supported by Libero SoC**

Device Family	Description
<a href="#">PolarFire</a>	PolarFire® FPGAs deliver the industry's lowest power at mid-range densities with exceptional security and reliability.
<a href="#">PolarFire SoC</a>	PolarFire SoC is the first SoC FPGA with a deterministic, coherent RISC-V CPU cluster and a deterministic L2 memory subsystem enabling Linux and real-time applications.

#### Helpful Links

- [Data sheets, tutorials, application notes, and silicon user guides](#)
- [Development boards and kits](#)
- [Libero SoC v12.0 and later](#)
- [Programming Solutions](#)
- [XLS-based power calculator estimators for device families](#)
- [Libero licensing](#)
- [Libero SoC PolarFire® product family](#)
- [Libero SoC PolarFire technology](#)
- [Libero SoC PolarFire documentation](#)
- [Libero SoC PolarFire software tools](#)

## Table of Contents

Introduction.....	1
1. Overview.....	5
1.1. Libero SoC Design Flow.....	5
1.2. Software IDE Integration.....	7
1.3. Software Tools.....	8
2. Managing Licenses.....	9
2.1. Microsemi License Utility.....	9
2.2. Selecting a Default License from a License List.....	10
2.3. Setting a Default License.....	12
2.4. Viewing License Details.....	14
2.5. Libero SoC Online Help.....	15
2.6. Libero SoC User Guides.....	15
3. Getting Started.....	17
3.1. Starting Libero SoC.....	17
3.2. Design Report.....	17
3.3. Creating a New Project.....	17
3.4. Opening a Project.....	24
4. Creating and Verifying Designs.....	26
4.1. Create SmartDesign.....	26
4.2. Create Core from HDL.....	35
4.3. Designing with HDL.....	37
4.4. Designing with Block Flow.....	39
4.5. Viewing Configured Components and SmartDesigns in a Project.....	39
4.6. Create a New SmartDesign Testbench.....	42
4.7. HDL Testbench.....	42
4.8. Import MSS.....	44
4.9. Verify Pre-Synthesized Design - RTL Simulation.....	44
5. Libero SoC Constraint Management.....	50
5.1. Invocation of Constraint Manager from the Design Flow Window.....	50
5.2. Libero SoC Design Flow.....	50
5.3. Introduction to Constraint Manager.....	52
5.4. Import a Constraint File.....	55
5.5. Constraint Types.....	59
5.6. Constraint Manager – I/O Attributes Tab.....	60
5.7. Constraint Manager – Timing Tab.....	62
5.8. Derived Constraints.....	64
5.9. Constraint Manager – Floor Planner Tab.....	65
5.10. Constraint Manager – Netlist Attributes Tab.....	66
6. Implementing Designs.....	68
6.1. Synthesize.....	68
6.2. Verifying Post-Synthesized Designs.....	74

6.3.	Compile Netlist.....	75
6.4.	Constraint Flow in Implementation.....	76
6.5.	Place and Route.....	81
6.6.	Multiple Pass Layout Configuration.....	85
6.7.	Post Layout Editing of I/O Signal Integrity and Delay Parameters.....	88
6.8.	Resource Usage.....	89
6.9.	Global Net Report.....	90
6.10.	Verify Post Layout Implementation.....	95
7.	Configure Hardware.....	113
7.1.	Programming Connectivity and Interface.....	113
7.2.	Select and Configure Actions and Procedures.....	115
7.3.	Programmer Settings.....	116
7.4.	Select Programmer.....	118
8.	Program Design.....	120
8.1.	Generating FPGA Array Data.....	120
8.2.	Initializing Design Blocks.....	120
8.3.	Generating Initialization Clients.....	156
8.4.	Configuring Actions and Procedures.....	156
8.5.	Configuring I/O States During JTAG Programming.....	160
8.6.	Configuring Programming Options.....	164
8.7.	Configuring Security.....	166
8.8.	Configure Permanent Locks for Production.....	176
8.9.	Configure Bitstream.....	179
8.10.	Generate Bitstream.....	180
8.11.	Run Programming Device Actions.....	181
8.12.	Program SPI Flash Image.....	187
9.	Debug Design.....	194
9.1.	Generate SmartDebug FPGA Array Data.....	194
9.2.	SmartDebug.....	194
9.3.	Identify Debug Design.....	195
10.	Handoff Design for Production.....	196
10.1.	Export Bitstream.....	196
10.2.	Export FlashPro Express Job.....	213
10.3.	Export Job Manager Data.....	217
10.4.	Export SPI Flash Image.....	218
10.5.	Export Pin Report.....	219
10.6.	Export BSDL File.....	220
10.7.	Export IBIS Model.....	220
11.	Export SmartDebug Data (Libero SoC).....	223
12.	References.....	225
12.1.	Archive Project Dialog Box.....	225
12.2.	Adding or Modifying Bus Interfaces in SmartDesign.....	226
12.3.	Catalog.....	228

12.4. Changing Output Port Capacitance.....	231
12.5. Core Manager.....	232
12.6. Configure Permanent Locks for Production.....	232
12.7. Importing Source Files – Copying Files Locally.....	233
12.8. Create Clock Constraint Dialog Box.....	233
12.9. Select Source Pins for Clock Constraint Dialog Box.....	234
12.10. Specifying Clock Constraints.....	235
12.11. Specifying Generated Clock Constraints.....	241
12.12. Design Hierarchy in the Design Explorer.....	244
12.13. Digest File.....	250
12.14. Design Rules Check.....	251
12.15. Editable Constraints Grid.....	252
12.16. Files Tab and File Types.....	253
12.17. Importing Files.....	253
12.18. Layout Error Message: layoutg4NoValidPlacement.....	254
12.19. Layout Error Message: layoutg4DesignHard.....	255
12.20. Save Project As Dialog Box.....	255
12.21. Project Settings Dialog Box.....	256
12.22. Global Include Files.....	265
12.23. Create and Configure Core Component.....	265
12.24. Search in Libero SoC.....	266
12.25. Organize Source Files Dialog Box – Synthesis.....	268
12.26. SmartDesign Testbench.....	269
12.27. Stimulus Hierarchy.....	269
12.28. Timing Exceptions Overview.....	270
12.29. Admin Profile Tool Dialog Box.....	270
12.30. Tool Profiles Dialog Box.....	271
12.31. User Preferences Dialog Box – Design Flow Preferences.....	272
12.32. IP Report.....	273
12.33. Force Update Design Flow.....	274
12.34. Generic/Parameter Report.....	276
13. Revision History.....	278
The Microchip Website.....	279
Product Change Notification Service.....	279
Customer Support.....	279
Microchip Devices Code Protection Feature.....	279
Legal Notice.....	280
Trademarks.....	280
Quality Management System.....	281
Worldwide Sales and Service.....	282



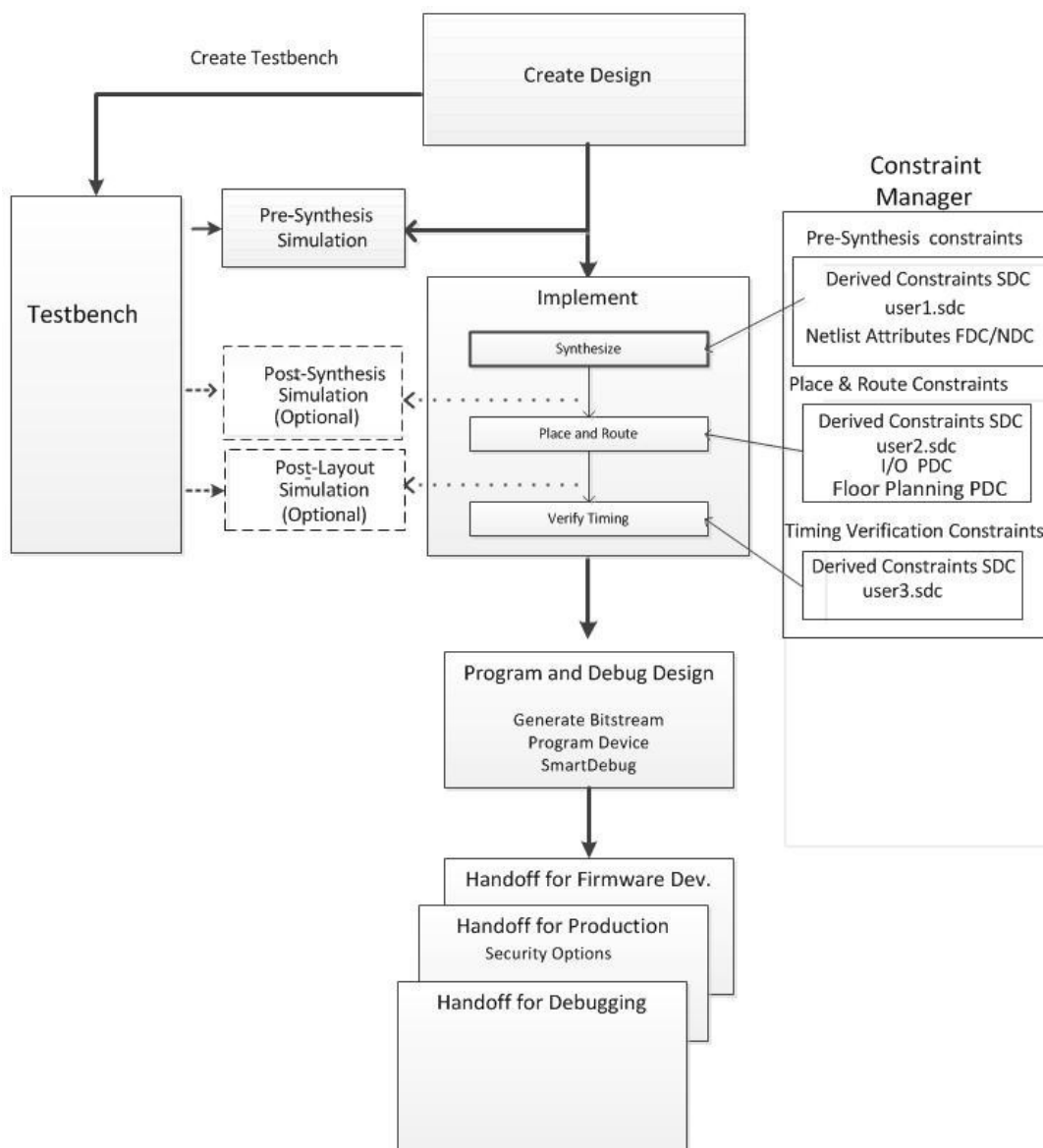
## 1. Overview

The following topics provide an overview of Libero SoC.

### 1.1 Libero SoC Design Flow

The following figure shows the Libero SoC design flow.

Figure 1-1. Libero SoC Design Flow




#### 1.1.1 Creating Your Design

Create your design with the following design capture tools:

- System Builder
- Create SmartDesign
- Create HDL

- Create SmartDesign Testbench (optional, for simulation only)
- Create HDL Testbench (optional, for simulation only)

After you create the design, invoke simulation for pre-synthesis verification.

You can also click the  button to invoke the Libero SoC software through Place and Route with default settings. However, doing so bypasses constraint management.

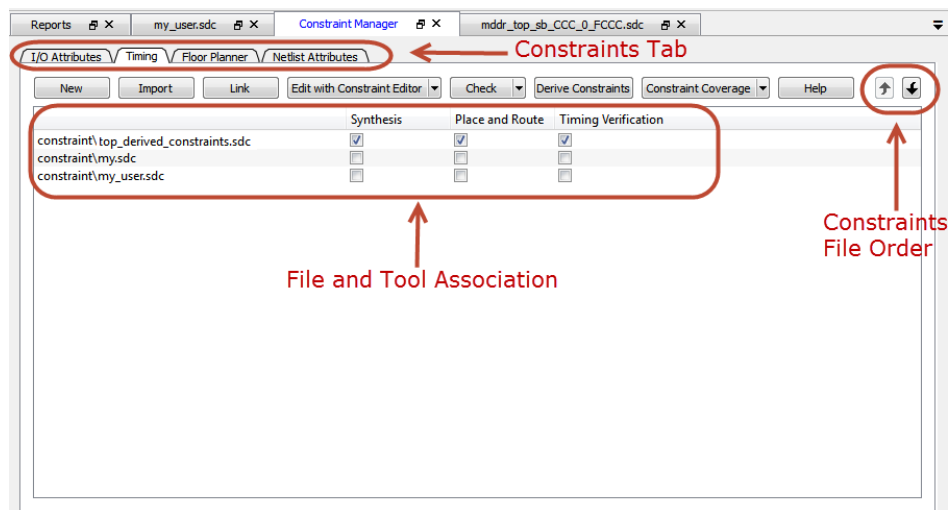
### 1.1.2 Working with Constraints

In the FPGA design world, constraint files are as important as design source files. Constraint files are used throughout the FPGA design process to guide FPGA tools to achieve the timing and power requirements of the design. For the synthesis step, SDC timing constraints set the performance goals whereas non-timing FDC constraints guide the synthesis tool for optimization. For the Place-and-Route step, SDC timing constraints guide the tool to achieve the timing requirements whereas Physical Design Constraints (PDC) guide the tool for optimized placement and routing (Floorplanning). For Static Timing Analysis, SDC timing constraints set the timing requirements and design-specific timing exceptions for static timing analysis.

Libero SoC provides the Constraint Manager to manage your design constraint needs. Constraint Manager is a single centralized graphical interface that allows you to create, import, link, check, delete, edit design constraints and associate the constraint files to design tools in the Libero SoC environment. Constraint Manager also allows you to manage constraints for SynplifyPro synthesis, Libero SoC Place-and-Route, and the SmartTime Timing Analysis throughout the design process.

After project creation, double-click **Manage Constraints** in the Design Flow window to open the **Constraint Manager**.


**Figure 1-2. Constraint Manager**



### 1.1.3 Implementing Your Design

The following guides and topics provide information about implementing your design.

- [Netlist Viewer \(User Guide\)](#)
- [6.1 Synthesize](#). This procedure runs synthesis on your design with the default settings, and passes to Synplify the constraints associated with Synthesis in the Constraint Manager.
- [6.2 Verifying Post-Synthesized Designs](#)
- [#unique\\_9](#)
- [6.5 Place and Route](#). Place and Route takes the design constraints from the Constraint Manager and uses

default settings. This is the last step in the push-button  design flow execution.

- [6.10 Verify Post Layout Implementation](#)

- [6.10.1 Generate Back Annotated Files](#)
- [6.10.2 Simulate - Opens ModelSim ME](#)
- [6.10.3 Verify Timing](#)
- [6.10.5 SmartTime](#)
- [6.10.6 Verify Power](#)
- [6.10.9 Simultaneous Switching Noise](#)

### 1.1.4 Programming and Debugging Your Design

The following topics provide information about programming and debugging your design.

#### Generating and Updating Data

- [Generate FPGA Array Data](#)
- [Update eNVM Memory Content](#)

#### Configuring Your Hardware

- [Programming Connectivity and Interface](#)
- [Configure Programmer](#)
- [Device I/O States During Programming - JTAG Mode Only](#)
- [Configure Programming Options](#)
- [Configure Security Policy Manager](#)

#### Programming Your Design

- [Generate Bitstream](#)
- [Run PROGRAM Action](#)

#### Debugging Your Design

- [Identify Debug Design](#)
- [SmartDebug \(User Guide\)](#)
- [Configure Permanent Locks for Production \(Configure OTP Security\)](#)

### 1.1.5 Handing Off Your Design to Production

The following topics provide information about handing off your design to production.

- [Export Bitstream](#)
- [Export FlashPro Express Job](#)
- [Export Job Manager Data](#)
- [Export Pin Report](#)
- [Export BSDL](#)
- [Export IBIS Model](#)
- [Handoff Design for Firmware Development](#)
- [Handoff Design for Debugging \(Export SmartDebug Data\)](#)

## 1.2 Software IDE Integration

Libero SoC simplifies the task of transitioning between designing your FPGA to developing your embedded firmware.

Libero SoC manages the firmware for your FPGA hardware design, including:

- Firmware hardware abstraction layers required for your processor.
- Firmware drivers for the processor peripherals that you use in your FPGA design.
- Sample application projects are available for drivers that illustrate the proper usage of the APIs.

To see which firmware drivers Libero SoC has found to be compatible with your design, open the [Firmware View](#). From this view, you can change the configuration of your firmware, change to a different version, read driver documentation, and generate sample projects for each driver.

Libero SoC manages the integration of your firmware with your preferred Software Development Environment, including SoftConsole, Keil, and IAR Embedded Workbench. The projects and workspace for your selected development environment are generated automatically with the proper settings and flags so you can write your applications immediately.

### 1.3 Software Tools

Libero SoC integrates design tools, streamlines design flow, manages design and log files, and passes design data between tools. The following table identifies the tools you can use to perform Libero SoC functions.

For more information about Libero SoC tools, visit <https://www.microsemi.com/products/fpga-soc/design-resources/design-software/libero-soc#overview>.

**Table 1-1. Matching Functions with Tools**

Function	Tool	Comments
Project Manager, HDL Editor, Core Generation	Libero SoC	Project Manager, HDL Editor targets the creation of HDL code. HDL Editor supports VHDL and Verilog with color, highlighting keywords for both HDL languages.
Synthesis	Synplify® Pro ME	SynplifyPro ME is integrated as part of the design package, enabling designers to target HDL code to specific devices.
Simulation	ModelSim® ME Pro	Allows source-level verification so designers can verify HDL code line by line. Designers can perform simulation at all levels: behavioral (or pre-synthesis), structural (or post-synthesis), and dynamic simulation.
Timing/Constraints, Power Analysis, Netlist Viewer, Floorplanning, Package Editing, Place-and-Route, Debugging	Libero SoC	This software package includes: <ul style="list-style-type: none"> <li>• ChipPlanner — displays I/O and logic macros in your design for floorplanning</li> <li>• Netlist Viewer — design schematic viewer.</li> <li>• SmartPower — power analysis tool.</li> <li>• SmartTime — static timing analysis and constraints editor.</li> </ul>

## 2. Managing Licenses

This chapter describes Libero SoC licensing.

### 2.1 Microsemi License Utility

The Microsemi License Utility allows you to check and update your license settings for the Libero SoC software. It displays your current license settings, the license host-id for the current host, and allows you to add a new license file to your settings.

#### Starting the Microsemi License Utility

Click **Start > All Programs > Microsemi Libero SoC vx.xx> Microsemi License Utility**.

**Note:** If you have more than one license available and have not selected a default license, the [Select License dialog box](#) appears.

#### Requesting a License

To request a license, click **Request License** to display the Microsemi license website. Then right-click and copy the disk volume value in the window, and paste the value into the Microsemi license web form.

The following table describes the available licenses.

**Table 2-1. Available Licenses**

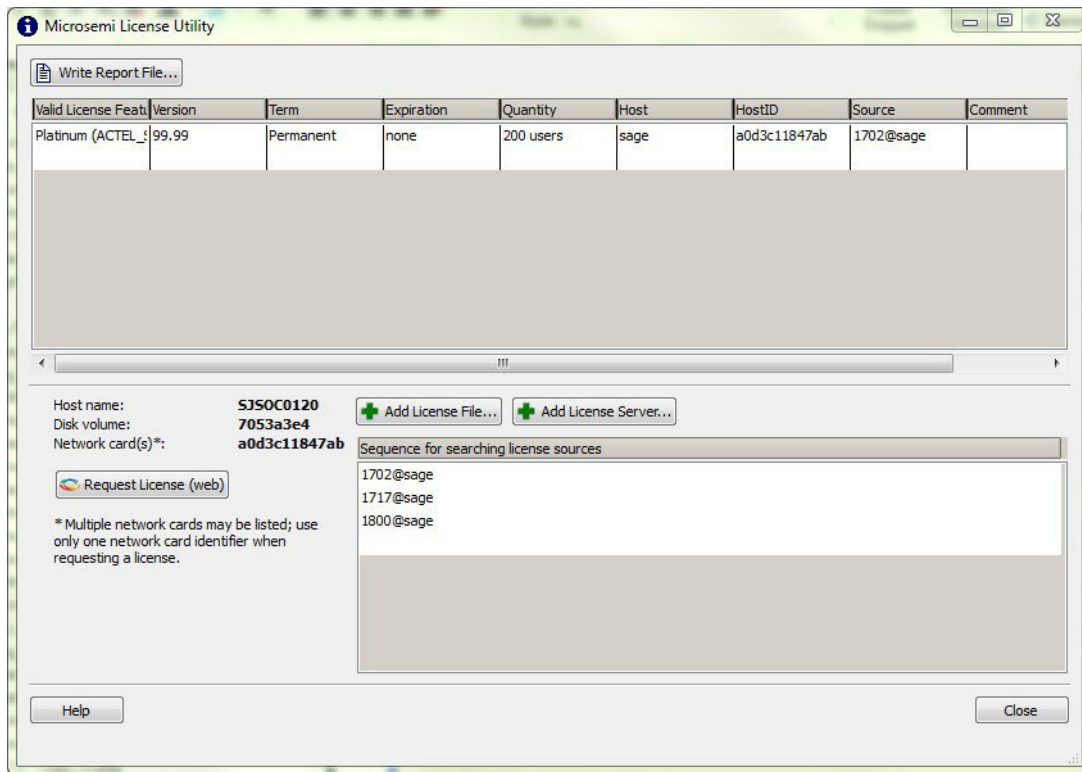
License	Description
1-year Platinum	Purchased license that supports all devices.
1-year Gold	Purchased license that supports a smaller set of devices than Platinum.
1-year Silver	Free license that supports a smaller set of devices than Gold.
30-day Evaluation	Free license that supports all devices, but disables programming.

When you receive your license file, follow the instructions that come with it and save the license to your local disk. In the Microsemi License Utility window, click **Add License File**. browse to the license file, and then select it. If you use a floating license, click **Add License Server** and enter the port number and name of the license server host.

Although the list of features for which you are licensed shows all versions, your license must have a version equal to or greater than your design tools release version for the `libero.exe` and `designer.exe` tools to run.

The list at the lower right shows the order in which the license files are read. The first file read appears at the top of the list.

Figure 2-1. Example of Microsemi License Utility



### Printing the Microsemi Licenses Report

Click **Write Report File** to print the Microsemi Tools Licenses Report or to save it as a .txt file.

### Related Information

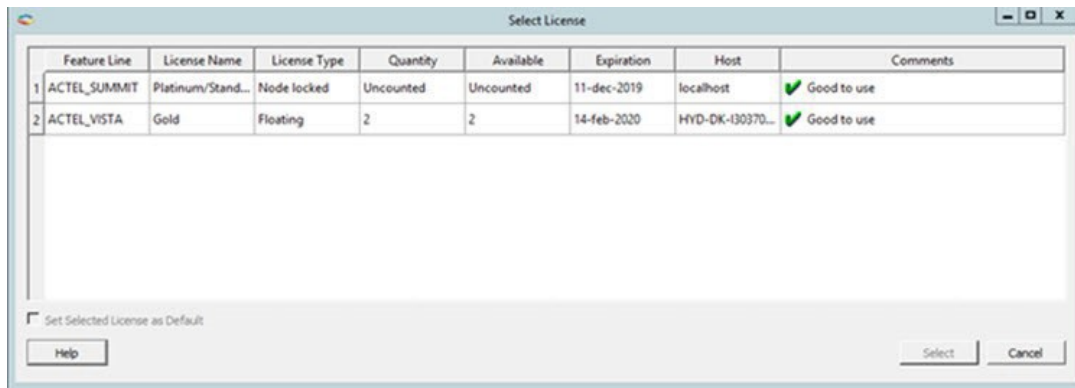
For more information about licensing, including links to troubleshooting and FAQ documents, see the [Microsemi Libero SoC License Information Web Page](#).

## 2.2 Selecting a Default License from a License List

If you have more than one license available and have not selected a default license, the Select License dialog box appears when you start the Microsemi License Utility. Select the feature license you want to use from the list of available licenses shown.

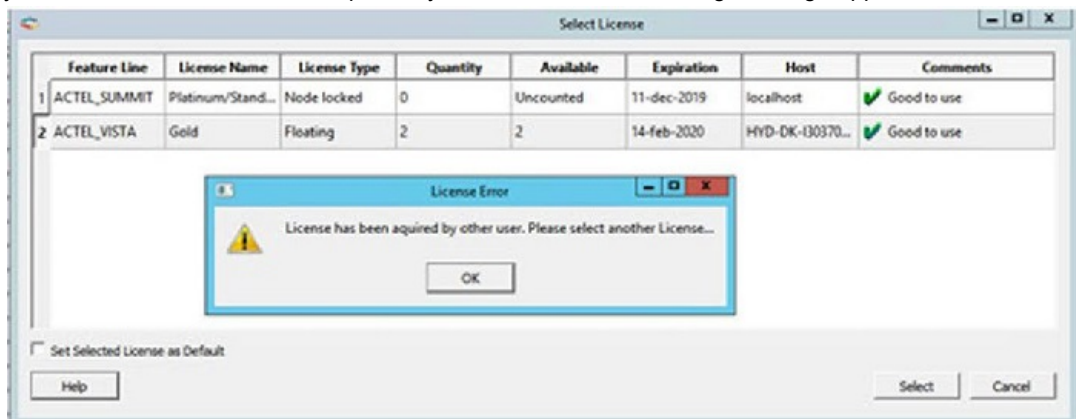
- The **Quantity** and **Available** columns show the total number of licenses and number of available licenses, respectively.
- The **License Type** column shows whether an available license is a Node Locked license or a Floating or Server-based license. Floating and Server-based Licenses can be used by multiple users, depending on the number of seats available.

**Figure 2-2. License Dialog Box**

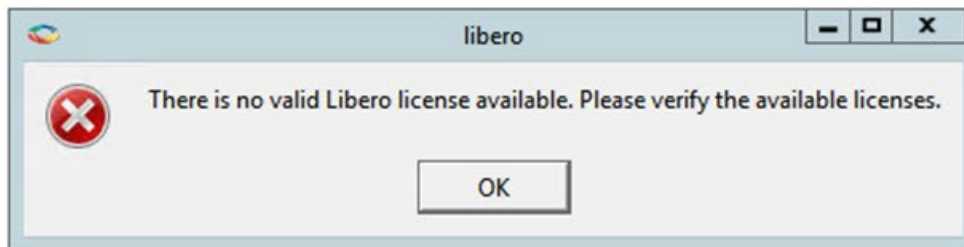


### Selecting a License to Use

1. In the Select License dialog box, click **Select** to activate Libero using the selected license. This button is disabled by default, and is enabled after you select a license.
2. Check the **Set Selected License as Default** check box to save the selected license as the default license to be used for future sessions. Selecting this option skips the license selection step for future sessions. Use this option if you want to use the same license features for future sessions.
  - If you select a license that was acquired by another user, the following message appears:



- If you select a license for which there are no valid licenses available, the following message appears:



To close the license selection window and exit Libero, click **Cancel**. To view the online help topic for License Selection, click **Help**.

### License Expiration

If a license will expire within 15 days, a warning appears in the **Comments** column of the dialog box and in the Log window (see the following examples).

Figure 2-3. Select License Dialog Box with License Expiration Warning Message

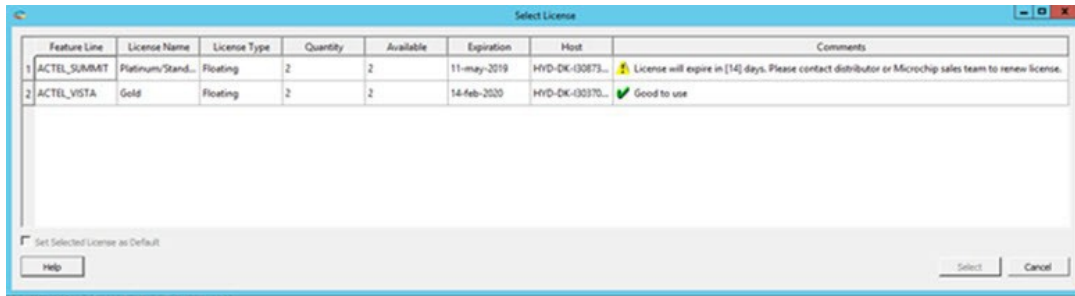
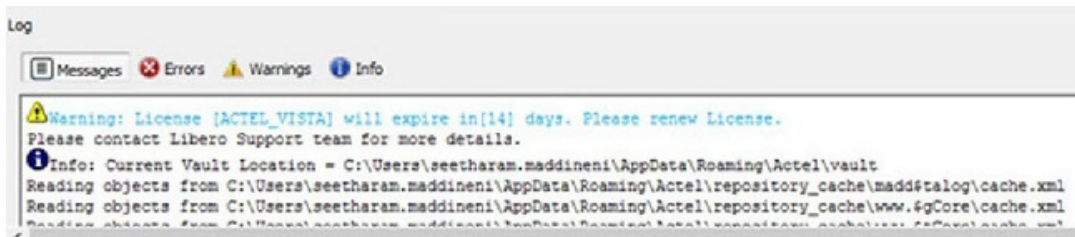


Figure 2-4. Log Window with License Expiration Warning Message

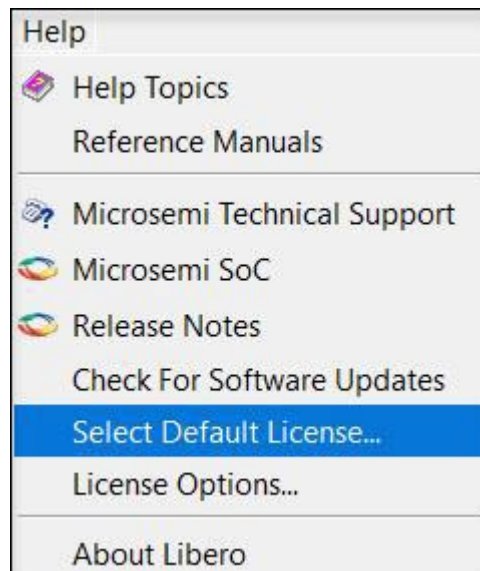


## 2.3 Setting a Default License

After you start Libero, you can select the default Libero license that will be used for future sessions. After you select a default Libero license, you can use this same procedure to change the default license.

Click **Help > Select Default License**.

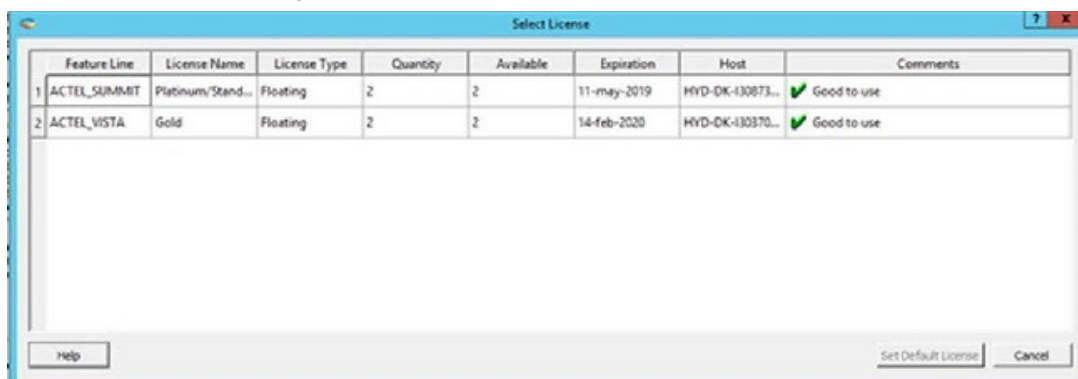
Figure 2-5. Selecting Help > Select Default License



When the following Select License dialog box appears, click a row, and then click **Set Default License** to specify the Libero license you selected as the default and close the dialog box.

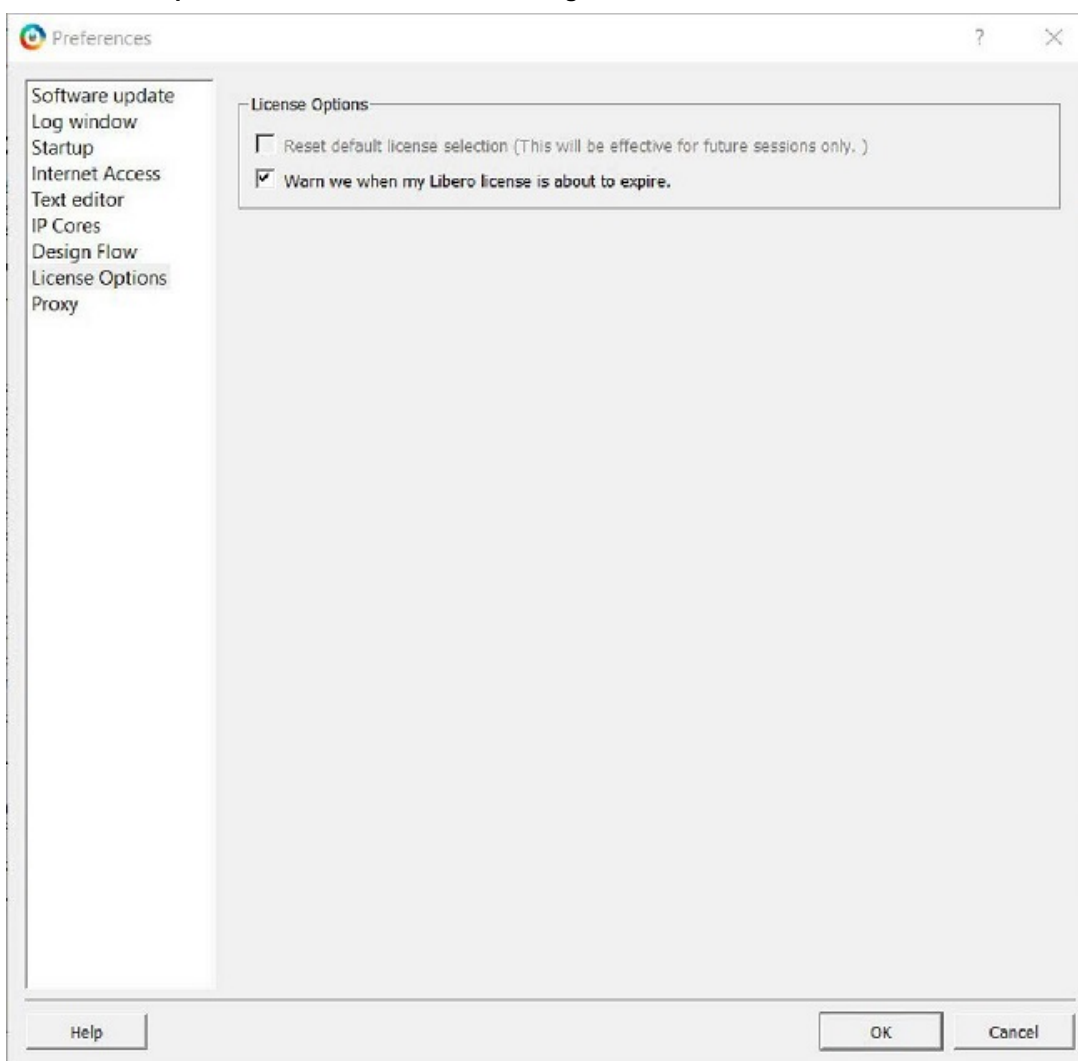


**Figure 2-6. Select License Dialog Box**



After you select a default Libero license, you can set license options in the Libero Preferences dialog box by clicking **Project > Preferences**. The following Preferences dialog box appears with the options described below.

**Figure 2-7. License Options in Libero Preferences Dialog Box**



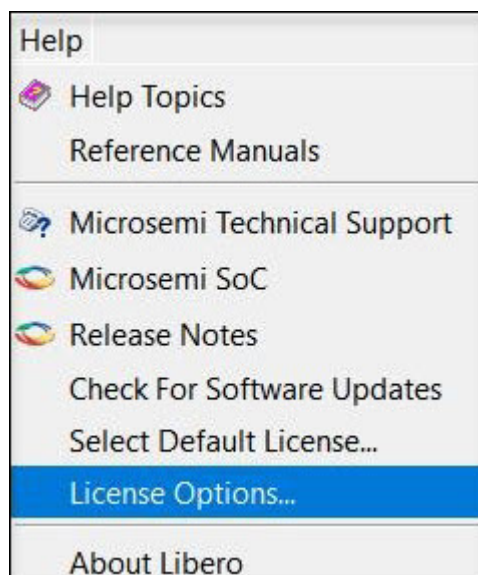
**Table 2-2. Preferences Dialog Box Options**

Option	Description
Reset Default License Selection	This option is selected when a default license is available. When checked, the default license is cleared and the check box is disabled.
Warn we when my Libero license is about to expire	Enables or disables Notification of License expiry. When checked, a popup message informs you about the impending expiration when the selected license's expiration date is within 15 days. Use this option only when the license's expiration date is longer than 5 days and shorter than 15 days.

## 2.4 Viewing License Details

The License Details dialog box shows detailed information about Libero SoC licenses and cores. To display this dialog box, click **Help > License Options**.

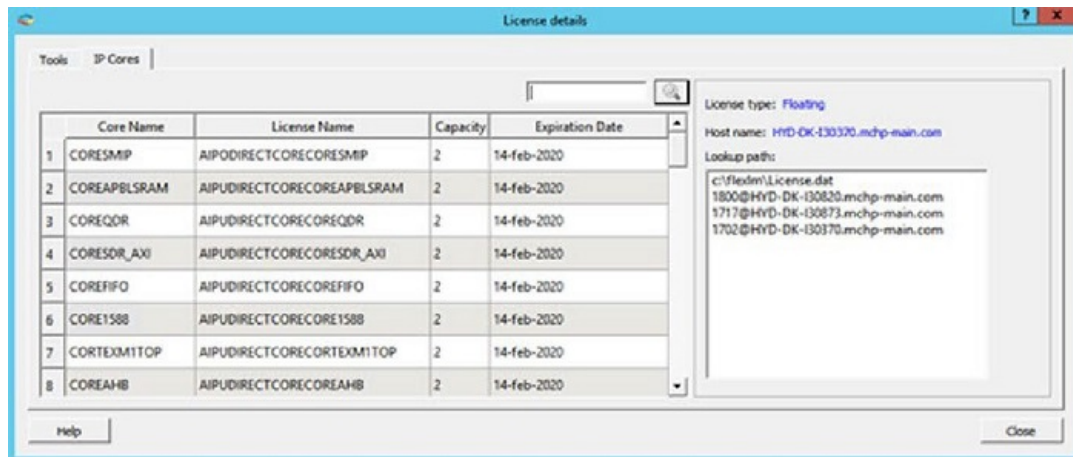
**Figure 2-8. Selecting Help > License Options**



The License Details dialog box has two tabs:

- **Tools** displays tool license details.
- **IP Cores** displays IP Cores license details, as shown in the following example.

Figure 2-9. IP Cores License Details



The following table describes the elements in the dialog box.

Table 2-3. Elements in the License Details Dialog Box

Element	Description
Close	Closes the dialog box.
Help	Displays the online help topic for License Selection.
Filter	Searches for the pattern entered in the text edit box. Filtered rows appear in the Cores table.
Lookup Path	Shows the list of License hosts included in the LM_LICENSE_FILE.

## 2.5 Libero SoC Online Help

The Libero SoC online help system is designed to open in the HTML Help Viewer – Microsoft's Help window for viewing compiled HTML Help. If you do not have the HTML Help Viewer components installed on your system, you can view the help using Microsoft Internet Explorer browser version 4.x or later.

The Libero SoC online help includes the following navigation tabs:

- **Contents.** The **Contents** tab shows books and pages that represent the categories of information in the online help system. When you click a closed book, it opens to display its content of sub-books and pages. When you click an open book, it closes. When you click pages, you select topics to view in the right-hand pane of the HTML Help viewer.
- **Search.** The **Search** tab allows you to find topics that contain key words. Full-text searching searches through every word in the online help to find matches. When the search is completed, a list of topics appears, so you can select a topic to view.

**Note:** Linux users might need to set the `LINUX_HTMLREADER` variable to enable an HTML viewer. For example:  
`setenv LINUX_HTMLREADER /usr/bin/firefox.` If you do not set this variable, HTML files, such as the online help, will not be available from within the software.

## 2.6 Libero SoC User Guides

Libero SoC includes online manuals that are in PDF format and are available from the Libero SoC Start menu. To access these guides, click **All Programs > Microsemi > Libero SoC > Libero SoC Reference Manuals**. You must have Adobe Acrobat Reader or similar PDF viewer to open and view the PDF user guides.

**Note:** Linux users may need to set their `LINUX_PDFREADER` variable to enable a PDF viewer. For example:  
`setenv LINUX_PDFREADER /usr/bin/kpdf`. If you do not set this variable, some PDF files will not be available from within software.

## 3. Getting Started

The following sections describe how to start using Libero SoC.

### 3.1 Starting Libero SoC

When you start Libero SoC, the Welcome screen appears. In the left pane, links under **Projects** allow you to create a new Libero SoC project or open an existing one.

- Clicking **New** starts the [New Project Creation Wizard](#). You use this wizard to create new Libero SoC projects.
- Clicking **Open** opens an [existing Libero SoC project](#).

### 3.2 Design Report

The **Design Report** tab lists the reports available for your design. Reports are added automatically as you move through design development. For example, Timing reports are added when you run timing analysis on your design. Reports are updated each time you run a timing analysis.

To display the **Design Report** tab, click **Design > Reports**.

If a report is not listed in the tab, you might have to create it manually. For example, you must invoke **Verify Power** manually before its report is available.

The following table lists the reports you can view in the **Design Report** tab.

**Table 3-1. Reports That Can be Viewed from the Design Report Tab**

Category	Report
Project Summary	<ul style="list-style-type: none"> <li><a href="#">Synthesize</a></li> <li><a href="#">Place and Route</a></li> <li><a href="#">Verify Timing</a></li> <li><a href="#">Verify Power</a></li> </ul>
Programming	<ul style="list-style-type: none"> <li><a href="#">Generate FPGA Array Data</a></li> <li><a href="#">Generate Bitstream</a></li> </ul>
Export	<ul style="list-style-type: none"> <li><a href="#">Export Pin Report</a></li> <li><a href="#">Export BSDL File</a></li> </ul>

### 3.3 Creating a New Project

To simplify project creation, Libero SoC provides a wizard that takes you step by step through the process of creating a new Libero SoC project.

To start a new Libero SoC project, click **Project > New**. The following table summarizes the screens in the wizard.

**Table 3-2. Screens in the New Project Wizard**

Screen	Description
<a href="#">Project Details</a>	Specify the name and location of your project, device family and parts, I/O standards, and HDL source files and design constraint files.
<a href="#">Device Selection</a>	Select a device for your project. After you select a device, or in any wizard screen that follows, you can click the <b>Finish</b> button to create the project and exit the wizard.

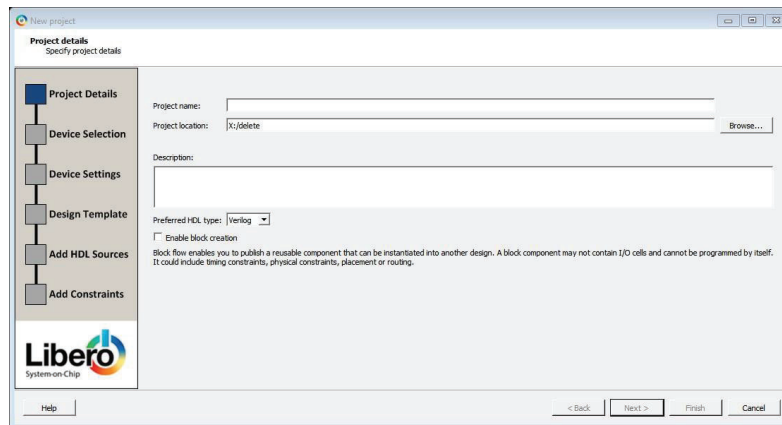
.....continued

Screen	Description
<a href="#">Device Settings</a>	Specify the device I/O technology and reserve pins for probes.
<a href="#">Add HDL Sources</a>	Add HDL design source files to your Libero SoC project.
<a href="#">Add Constraints</a>	Add timing and physical constraints files to your Libero SoC project.

### 3.3.1 New Project Creation Wizard – Project Details

Project Details is the first screen that appears in the New Project Creation wizard.

Figure 3-1. Libero SoC New Project Creation Wizard - Project Details



The following table describes the fields in the Project Details screen. After you complete the fields, click **Next** to go to [Device Selection](#).

Table 3-3. Fields in the Libero SoC New Project Creation Wizard - Project Details

Field	Description
<b>Project Name</b>	Identifies your project name. Do not use spaces or reserved Verilog or VHDL keywords.
<b>Project Location</b>	Identifies your project location on disk.
<b>Description</b>	General information about your design and project.
<b>Preferred HDL Type</b>	Sets your HDL type to one of the following: <ul style="list-style-type: none"> <li>Verilog</li> <li>VHDL</li> </ul> Libero-generated files (SmartDesigns, SmartGen cores, and so on) are created in the HDL type you specify. Libero SoC supports mixed HDL designs.
<b>Enable Block Creation</b>	Allows you to build blocks for your design. These blocks can be assembled in other designs, with partial layout, and been optimized for timing and power performance for a specific Microchip device. Once optimized, you can use the same blocks in multiple designs.

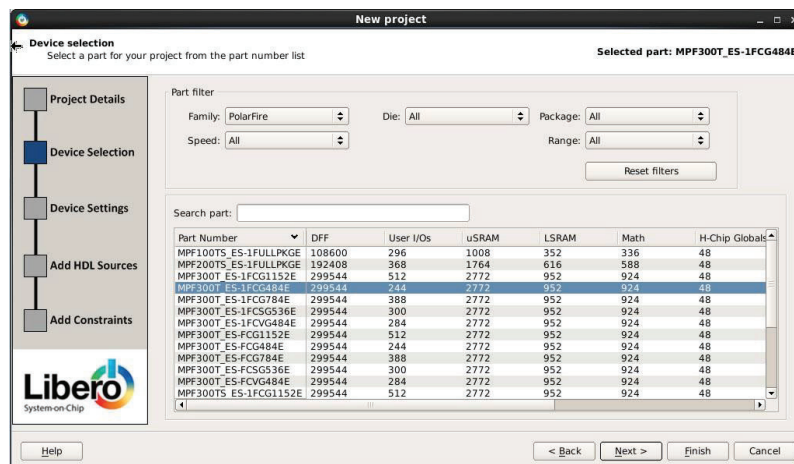
### 3.3.2 New Project Creation Wizard – Device Selection

The Device Selection screen is where you specify the Microchip device for your project. Use the filters and drop-down lists to refine your search for the right part to use for your design.

This screen contains a table of all the parts, with associated FPGA resource details generated based on a value you enter in a filter. When you select a filter value:

- The parts table is updated to reflect the result of the new filtered value.
- All other filters are updated, and only relevant items are available in the filter drop-down lists. For example, if you select **PolarFire** in the **Family** filter, the parts table includes only PolarFire parts and the **Die** filter includes only PolarFire dies in the **Die** drop-down list.

**Figure 3-2. Libero SoC New Project Creation Wizard - Device Selection**



The following table describes the fields in the Device Selection screen. After you complete the fields, click **Next** to go to the [Device Settings](#) screen or click **Finish** to complete new project creation with all remaining defaults.

**Table 3-4. Fields in the Libero SoC New Project Creation Wizard - Device Selection**

Field	Description
<b>Family</b>	Specify the Microchip device family. Only devices that belong to the family are listed in the parts table.
<b>Die/Package/Speed</b>	Select your device die, package, and speed grade. Use the Die/Package/Speed filters to view only the selections that interest you. The Die/Package/Speed grades available for selection depend on the level of <a href="#">Libero SoC license</a> you have (Evaluation, Silver, Gold, or Platinum). For more information, see the <a href="#">Libero SoC Licensing web page</a> .
<b>Range</b>	Defines the voltage and temperature ranges a device might encounter in your application. Tools such as SmartTime, SmartPower, timing-driven layout, power-driven layout, the timing report, and back-annotated simulation are affected by operating conditions. Select the appropriate option for your device. Supported operating condition ranges vary according to your device and package. To find your recommended temperature range, see your device datasheet. <ul style="list-style-type: none"> <li>• <b>All</b> – All ranges</li> <li>• <b>EXT</b> – Extended</li> <li>• <b>IND</b> – Industrial</li> <li>• <b>MIL (Military)</b></li> </ul>
<b>Reset Filters</b>	Reset all filters to the default ALL option except <b>Family</b> .

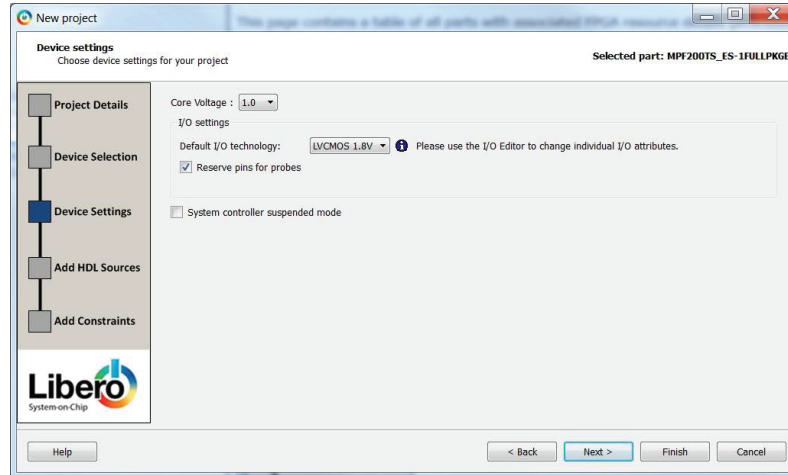
.....continued

Field	Description
<b>Search Parts</b>	Enter a character-by-character search for parts. Search results appear in the parts table.

### 3.3.3 New Project Creation Wizard – Device Settings

For PolarFire devices, the Device Settings page is where you set the device I/O technology and reserve pins for probes.

Figure 3-3. Libero SoC New Project Creation Wizard – Device Settings (PolarFire)



The following table describes the fields in the Device Settings screen. After you complete the fields:

- Click **Next** to go to the [Add HDL Sources](#) screen for PolarFire and PolarFire2 devices .  
**OR**
- Click **Finish** to complete new project creation with all remaining defaults.

Table 3-5. Fields in the Libero SoC New Project Creation Wizard - Device Settings

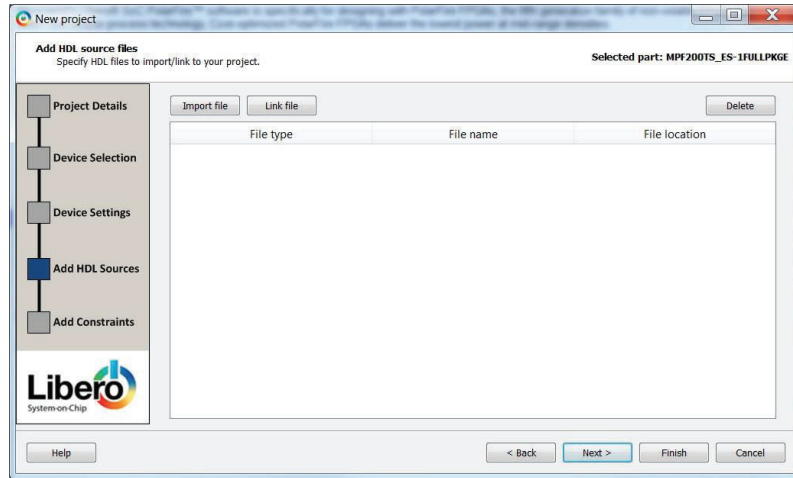
Field	Description
<b>Core Voltage</b>	Set the core voltage for your device.
<b>Default I/O Technology</b>	Set all your I/Os to a default value. You can change the values for individual I/Os in the I/O Attribute Editor. The I/O technology available is family-dependent.
<b>Reserve Pins for Probes</b>	Reserve your pins for probing if you intend to debug using SmartDebug.
<b>System controller suspended mode</b>	<p>PolarFire only. If the system controller suspended mode is enabled, the following operations will not be available during normal operation.</p> <ul style="list-style-type: none"> <li>• All System controller services requested after power-up is complete and the System controller is suspended</li> <li>• System controller-generated Tamper flags</li> <li>• Device reset and zeroization Tamper responses</li> <li>• SPI-Master In-Application Programming (IAP)</li> <li>• SPI-Slave programming mode</li> </ul>



### 3.3.4 New Project Creation Wizard – Add HDL Source Files

The Add HDL Source Files screen is where you add HDL design source files to your Libero SoC project. The HDL source files can be imported or linked to the Libero SoC project.

Figure 3-4. Libero SoC New Project Creation Wizard - Add HDL Source Files

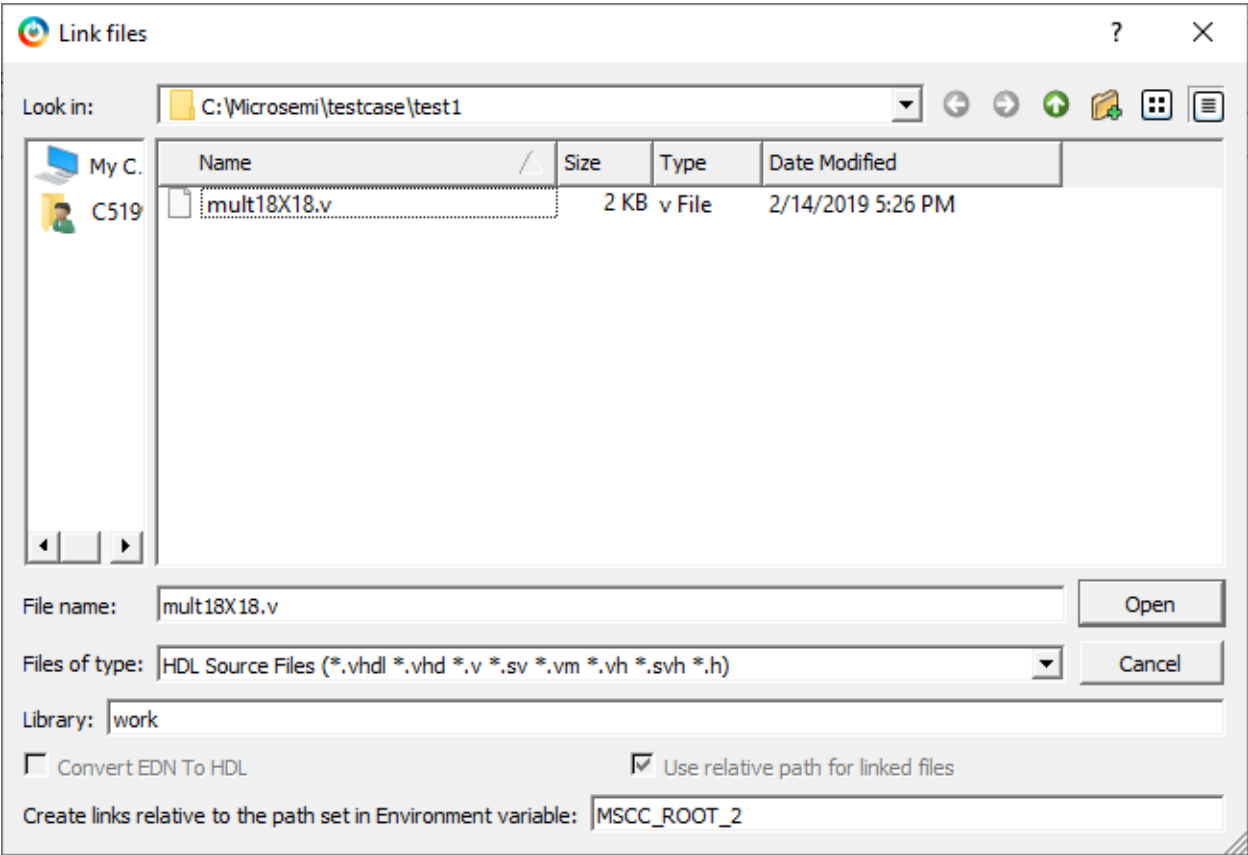


The following table describes the elements in the Add HDL Source Files screen. After you complete the fields, click **Next** to go to the [Add Constraints](#) screen or click **Finish** to complete new project creation with all remaining defaults.

Table 3-6. Elements in the Libero SoC New Project Creation Wizard - Add HDL Source Files

Element	Description
<b>Import File</b> button	Imports an HDL source file. When the dialog box appears, go to the location where the HDL source resides, select the HDL file, and click <b>Open</b> . The HDL file is copied to the <prj_folder>/hdl folder in your Libero project.
<b>Link File</b> button	Allows you to continue with an absolute or relative path for linked files. When the Link files dialog box appears (see the figure below), go to the location where the HDL source resides, select the HDL file, and click <b>Open</b> .
<b>Create links relative to the path set in Environment variable</b>	The HDL file is linked to the Libero project. Check this check box if the HDL source file is located and maintained outside the Libero project. This option requires you to specify an environment variable that has a relative path set to it. Links are created relative to the path set in the environment variable.  <b>Note:</b> If you select relative path and provide an environment variable for the relative path, you cannot switch to absolute path. After the environment variable is set, this option becomes read-only in all other link files dialog boxes.
<b>Delete</b> button	Deletes the selected HDL source file from your project. If the HDL source file is linked to the Libero project, the link will be removed.

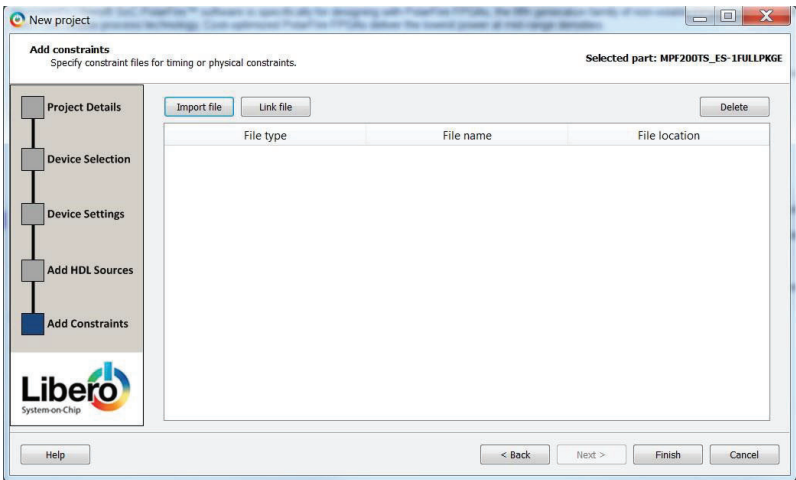
Figure 3-5. New Project Creation Wizard – Link Files Dialog Box



3.3.5 New Project Creation Wizard - Add Constraints

The Add Constraints screen is where you add timing constraints and physical constraints files to your Libero SoC project. The constraints file can be imported or linked to the Libero SoC project.

Figure 3-6. Libero SoC New Project Creation Wizard – Add Constraints

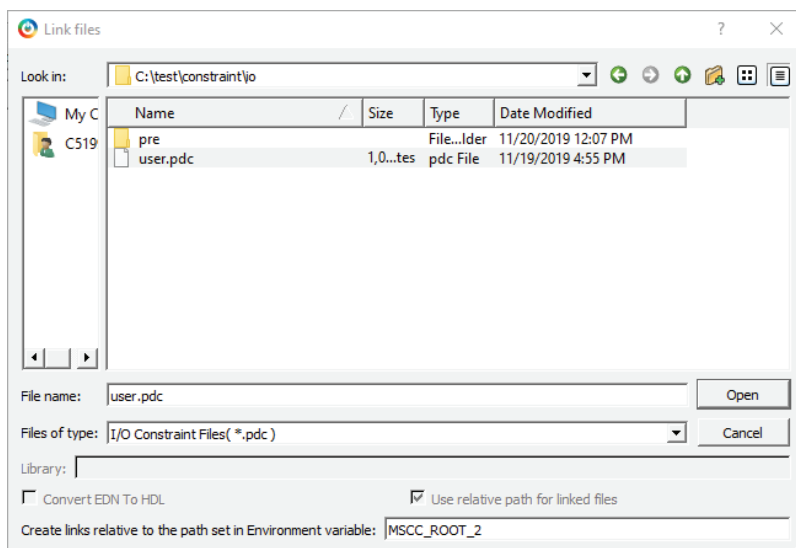


The following table describes the elements in the Add Constraints screen.

**Table 3-7. Elements in the Libero SoC New Project Creation Wizard - Add Constraints**

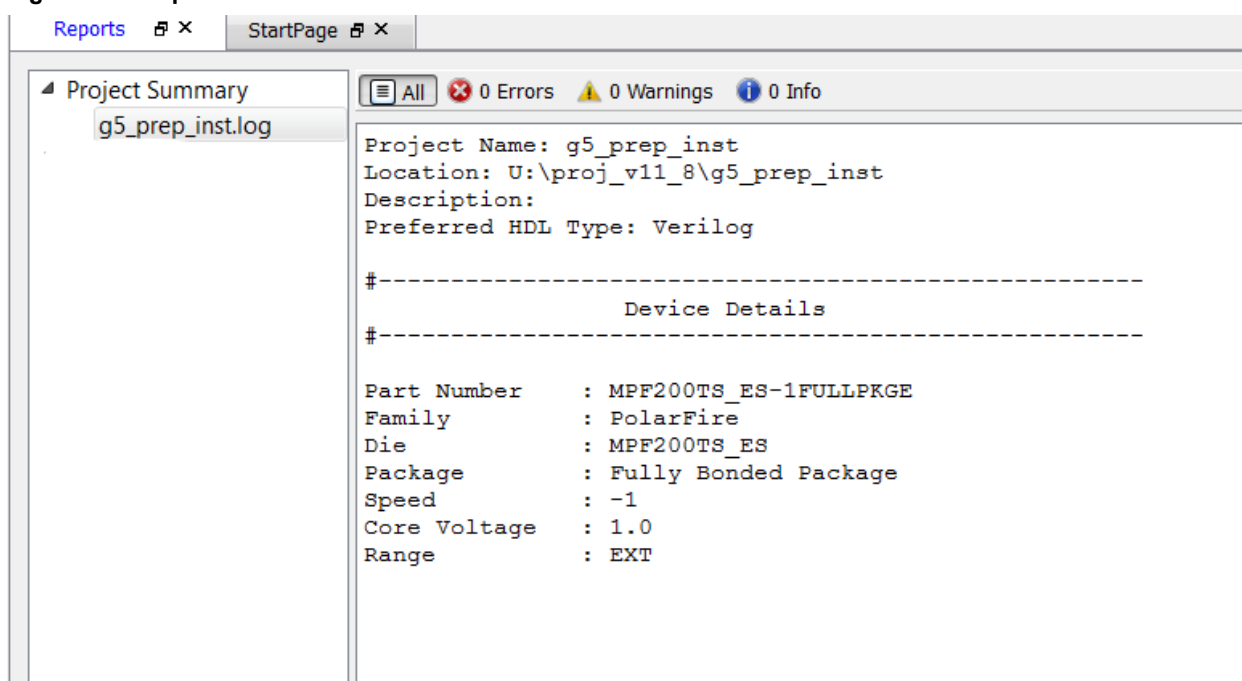
Element	Description
<b>Import File</b> button	Go to the location where the constraints file resides. Select the constraints file and click <b>Open</b> . The constraints file is copied to the <prj_folder>/constraint folder in your Libero project.
<b>Link File</b> button	Click this button if the constraint file is located and maintained outside the Libero project. When the Link files dialog box appears (see the figure below), specify an absolute path or choose a relative path for linked files. Go to the location where the constraints file resides. Select the constraints file and click <b>Open</b> . The constraints file is linked to the Libero project.
<b>Create links relative to the path set in Environment variable</b>	<p>The constraints file is linked to the Libero project. Check this check box if the constraints file is located and maintained outside the Libero project. This option requires you to specify an environment variable that has a relative path set to it. Links are created relative to the path set in the environment variable.</p> <p><b>Note:</b> If you select relative path and provide an environment variable for the relative path, you cannot switch to absolute path. After the environment variable is set, this option becomes read-only in all other link files dialog boxes.</p>
<b>Delete</b> button	Deletes the selected constraints file from your project. If the constraints file is linked to the Libero project, the link will be removed.

**Figure 3-7. New Project Creation Wizard – Link Files Dialog Box**



After you complete the fields, click **Finish** to complete new project creation. The **Reports** tab shows the result of your new project.

Figure 3-8. Reports Tab



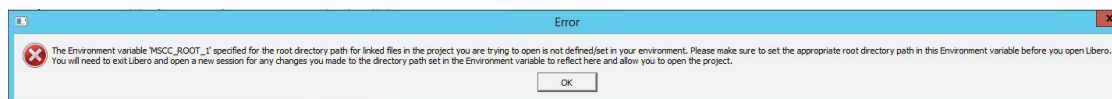
### 3.4 Opening a Project

To open a project:

1. From the **File** menu, choose **Open Project**.
2. Select the project file you want to open. The file ends in the extension `.prjx`.
3. Click **Open**.

**Note:** Opening a project created using a relative path for linked files displays the following error message if the environment variable does not exist or the path set in the environment variable is empty and cancels opening the project.

Figure 3-9. Error Message

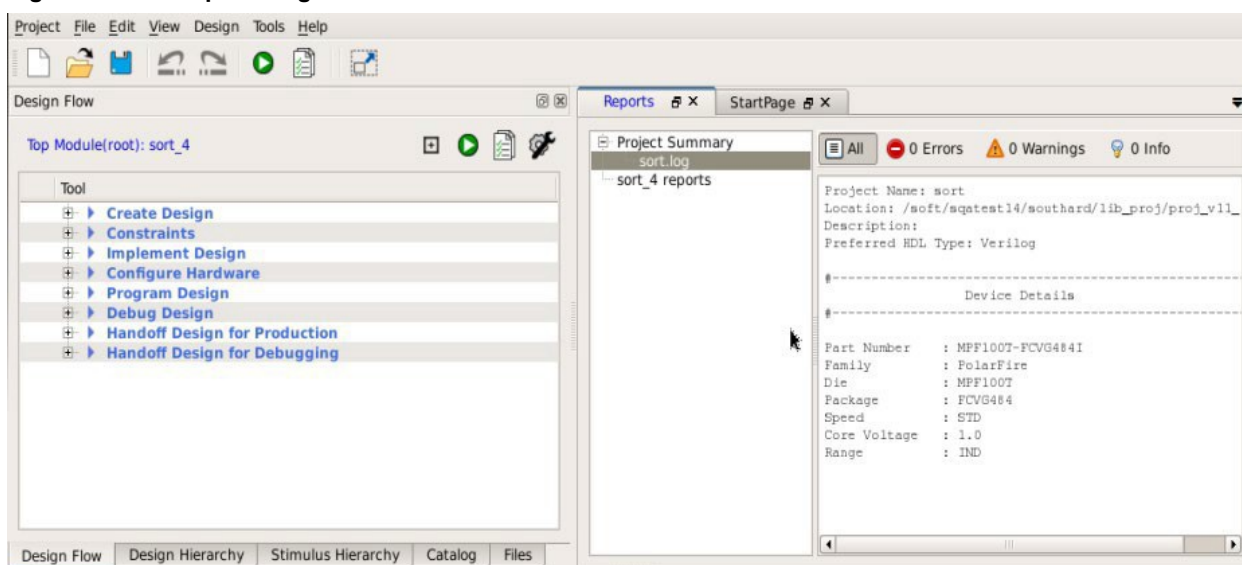


When you open an existing Libero SoC project:

- A Design Flow window appears on the left side.
- A log and message window appear at the bottom.
- Project information windows appear on the right side.

The following figure is an example of a newly created project, with only the top-level Design Flow window steps shown.

Figure 3-10. Sample Design Flow Window



The Design Flow window may be slightly different for each technology family. However, all flows include some version of the following design steps:

- Create Design
- Constraints
- Implement Design
- Configure Hardware
- Program Design
- Debug Design
- Handoff Design for Production
- Handoff Design for Debugging

## 4. Creating and Verifying Designs

Create your design with the following design capture tools:

- [Create SmartDesign](#)
- [Create HDL](#)
- [Create SmartDesign Testbench](#) (optional, for simulation only)
- [Create HDL Testbench](#) (optional, for simulation only)
- [Verify Pre-synthesized Design](#)

### 4.1 Create SmartDesign

SmartDesign is a visual block-based design creation and entry tool for instantiating, configuring, and connecting Microchip IPs, user-generated IPs, and custom/glue-logic HDL modules. This tool provides a canvas for instantiating and stitching together design objects. The final result from SmartDesign is a design-rule-checked and automatically abstracted synthesis-ready HDL file. A generated SmartDesign can be the entire FPGA design or a component subsystem to be re-used in a larger design.

The following design objects can be instantiated in the SmartDesign Canvas:

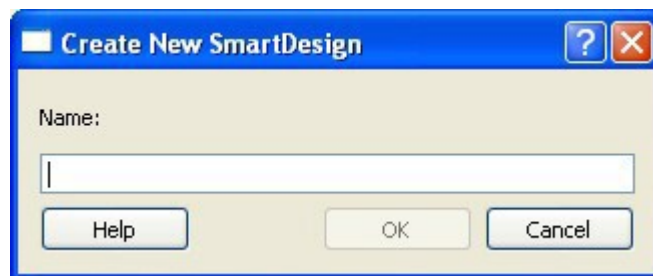
- Microchip IP Cores
- User-generated or third-party IP Cores
- HDL design files
- HDL + design files
- Basic macros
- Other SmartDesign components (\*.cxf files). These files can be generated from SmartDesign in the current Libero SoC project, or they can be imported from other Libero SoC projects.
- Re-usable design blocks (\*.cxyz files) published from Libero SoC. For more information, see the [SmartDesign User Guide](#).

#### 4.1.1 Create New SmartDesign

This SmartDesign component may be the top level of the design, or it can be used as a lower level SmartDesign component after successful generation in another design.

1. From the **File** menu, choose **New > SmartDesign** in the Design Flow window or double-click **Create SmartDesign**. The Create New SmartDesign dialog box appears.

**Figure 4-1. Create New SmartDesign Dialog Box**



2. Enter a name and click **OK**. The component appears in the **Design Hierarchy** tab of the Design Explorer.

**Note:** The component name you choose must be unique in your project. For more information, see the [SmartDesign User Guide](#).

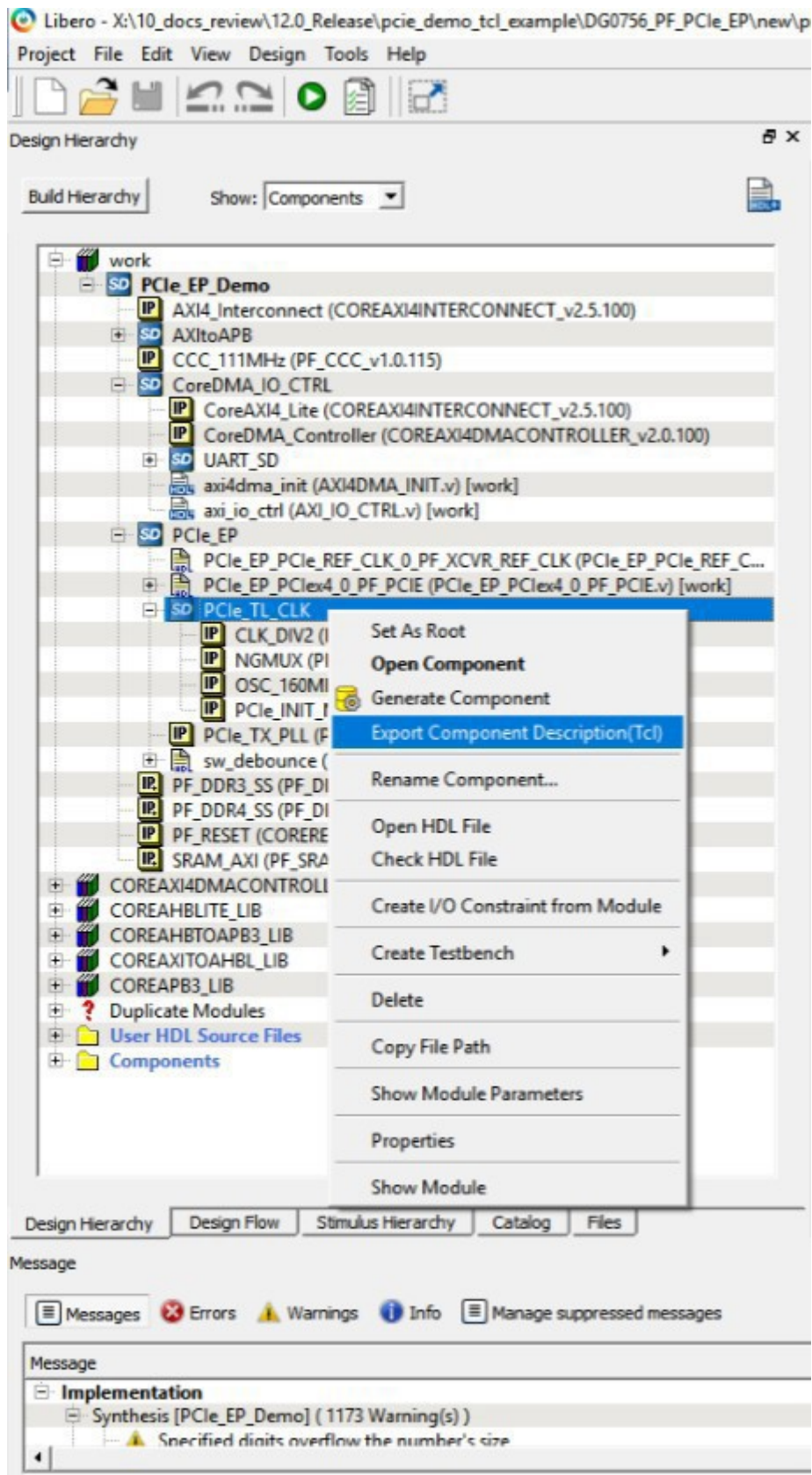
#### 4.1.2 Export Component Description (Tcl)

Using the Export Component Description option, you can export components such as SmartDesign components, configured cores, and HDL+ cores separately as Tcl.

To export a SmartDesign component:

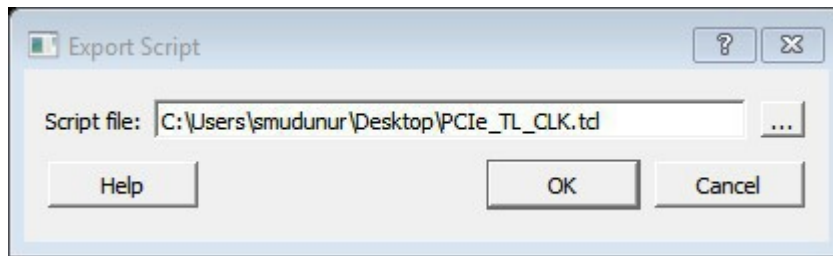
1. Right-click the component and choose **Export Component Description (Tcl)**.

Figure 4-2. Export as TCL Option for SmartDesign Component





**Figure 4-3. Export Script Dialog Box**



2. Click the **Browse** button to specify the location where you want to export the Tcl file, and then click **OK**.

### 4.1.3 Examples

The following is an example of an exported Tcl script for a SmartDesign Component (PCie\_TL\_CLK).

```
# Creating SmartDesign PCie_TL_CLK set sd_name {PCie_TL_CLK}
create_smartdesign -sd_name ${sd_name}
# Disable auto promotion of pins of type 'pad' auto_promote_pad_pins -promote_all 0
# Create top level Ports
sd_create_scalar_port -sd_name ${sd_name} -port_name {CLK_125MHz} -port_direction {IN}
sd_create_scalar_port -sd_name ${sd_name} -port_name {TL_CLK} -port_direction {OUT}
sd_create_scalar_port -sd_name ${sd_name} -port_name {DEVICE_INIT_DONE} -port_direction {OUT}
# Add CLK_DIV2_0 instance
sd_instantiate_component -sd_name ${sd_name} -component_name {CLK_DIV2} -instance_name {CLK_DIV2_0}
# Add NGMUX_0 instance
sd_instantiate_component -sd_name ${sd_name} -component_name {NGMUX} -instance_name {NGMUX_0}
# Add OSC_160MHz_0 instance
sd_instantiate_component -sd_name ${sd_name} -component_name {OSC_160MHz} -instance_name {OSC_160MHz_0}
# Add PCie_INIT_MONITOR_0 instance
sd_instantiate_component -sd_name ${sd_name} -component_name {PCie_INIT_MONITOR} -instance_name {PCie_INIT_MONITOR_0}
sd_mark_pins_unused -sd_name ${sd_name} -pin_names {PCie_INIT_MONITOR_0:FABRIC_POR_N}
sd_mark_pins_unused -sd_name ${sd_name} -pin_names {PCie_INIT_MONITOR_0:USRAM_INIT_DONE}
sd_mark_pins_unused -sd_name ${sd_name} -pin_names {PCie_INIT_MONITOR_0:SRAM_INIT_DONE}
sd_mark_pins_unused -sd_name ${sd_name} -pin_names {PCie_INIT_MONITOR_0:XCVR_INIT_DONE}
sd_mark_pins_unused -sd_name ${sd_name} -pin_names {PCie_INIT_MONITOR_0:USRAM_INIT_FROM_SNVM_DONE}
sd_mark_pins_unused -sd_name ${sd_name} -pin_names {PCie_INIT_MONITOR_0:USRAM_INIT_FROM_UPROM_DONE}
sd_mark_pins_unused -sd_name ${sd_name} -pin_names {PCie_INIT_MONITOR_0:USRAM_INIT_FROM_SPI_DONE}
sd_mark_pins_unused -sd_name ${sd_name} -pin_names {PCie_INIT_MONITOR_0:SRAM_INIT_FROM_SNVM_DONE}
sd_mark_pins_unused -sd_name ${sd_name} -pin_names {PCie_INIT_MONITOR_0:SRAM_INIT_FROM_UPROM_DONE}
sd_mark_pins_unused -sd_name ${sd_name} -pin_names {PCie_INIT_MONITOR_0:SRAM_INIT_FROM_SPI_DONE}
sd_mark_pins_unused -sd_name ${sd_name} -pin_names {PCie_INIT_MONITOR_0:AUTOCALIB_DONE}
# Add scalar net connections
sd_connect_pins -sd_name ${sd_name} -pin_names {"NGMUX_0:CLK1" "CLK_125MHz" } sd_connect_pins
-sd_name ${sd_name} -pin_names {"CLK_DIV2_0:CLK_OUT" "NGMUX_0:CLK0" }
sd_connect_pins -sd_name ${sd_name} -pin_names {"PCie_INIT_MONITOR_0:DEVICE_INIT_DONE"
"DEVICE_INIT_DONE" }
sd_connect_pins -sd_name ${sd_name} -pin_names {"CLK_DIV2_0:CLK_IN"
"OSC_160MHz_0:RCOSC_160MHz_CLK_DIV" }
sd_connect_pins -sd_name ${sd_name} -pin_names {"NGMUX_0:SEL"
"PCie_INIT_MONITOR_0:PCIE_INIT_DONE" }
sd_connect_pins -sd_name ${sd_name} -pin_names {"NGMUX_0:CLK_OUT" "TL_CLK" }
# Re-enable auto promotion of pins of type 'pad'
auto_promote_pad_pins -promote_all 1
# Save the smartDesign save_smartdesign -sd_name ${sd_name}
# Generate SmartDesign PCie_TL_CLK
generate_component -component_name ${sd_name}
```

The following is an example of an exported Tcl script for a System Builder Core (PF\_DDR3\_SS).

```
# Exporting core PF_DDR3_SS to TCL
# Create design TCL command for core PF_DDR3_SS
create_and_configure_core -core_vlnv {Actel:SystemBuilder:PF_DDR3:2.3.120} - component_name
{PF_DDR3_SS} -params {\
"ADDRESS_MIRROR:false" \
"ADDRESS_ORDERING:CHIP_ROW_BANK_COL" \
"AUTO_SELF_REFRESH:1" \
"AXI_ID_WIDTH:6" \
"AXI_WIDTH:64" \
"BANKSTATMODULES:4" \
"BANK_ADDR_WIDTH:3" \
"BURST_LENGTH:0" \
"CAS_ADDITIVE_LATENCY:0" \
"CAS_LATENCY:9" \
"CAS_WRITE_LATENCY:7" \
"CCC_PLL_CLOCK_MULTIPLIER:6" \
"CLOCK_DDR:666.666" \
"CLOCK_PLL_REFERENCE:111.111" \
"CLOCK_RATE:4" \
"CLOCK_USER:166.6665" \
"COL_ADDR_WIDTH:11" \
"DLL_ENABLE:1" \
"DM_MODE:DM" \
"DQ_DQS_GROUP_SIZE:8" \
"ENABLE_ECC:0" \
"ENABLE_INIT_INTERFACE:false" \
"ENABLE_LOOKAHEAD_PRECHARGE_ACTIVATE:false" \
"ENABLE_PAR_ALERT:false" \
"ENABLE_REINIT:false" \
"ENABLE_TAG_IF:false" \
"ENABLE_USER_ZQCALIB:false" \
"EXPOSE_TRAINING_DEBUG_IF:false" \
"FABRIC_INTERFACE:AXI4" \
"FAMILY:26" \
"MEMCTRLR_INST_NO:1" \
"MEMORY_FORMAT:COMPONENT" \
"MINIMUM_READ_IDLE:1" \
"ODT_ENABLE_RD_RNK0_ODT0:false" \
"ODT_ENABLE_RD_RNK0_ODT1:false" \
"ODT_ENABLE_RD_RNK1_ODT0:false" \
"ODT_ENABLE_RD_RNK1_ODT1:false" \
"ODT_ENABLE_WR_RNK0_ODT0:true" \
"ODT_ENABLE_WR_RNK0_ODT1:false" \
"ODT_ENABLE_WR_RNK1_ODT0:false" \
"ODT_ENABLE_WR_RNK1_ODT1:true" \
"ODT_RD_OFF_SHIFT:0" \
"ODT_RD_ON_SHIFT:0" \
"ODT_WR_OFF_SHIFT:0" \
"ODT_WR_ON_SHIFT:0" \
"OUTPUT_DRIVE_STRENGTH:RZQ6" \
"PARAM_IS_FALSE:false" \
"PARTIAL_ARRAY_SELF_REFRESH:FULL" \
"PHYONLY:false" \
"PIPELINE:false" \
"QOFF:0" \
"QUEUE_DEPTH:3" \
"RDIMM_LAT:0" \
"READ_BURST_TYPE:SEQUENTIAL" \
"ROW_ADDR_WIDTH:16" \
"RTT_NOM:DISABLED" \
"RTT_WR:OFF" \
"SDRAM_NB_RANKS:1" \
"SDRAM_NUM_CLK_OUTS:1" \
"SDRAM_TYPE:DDR3" \
"SELF_REFRESH_TEMPERATURE:NORMAL" \
"SHIELD_ENABLED:true" \
"SIMULATION_MODE:FAST" \
"TDQS_ENABLE:DISABLE" \
"TGIGEN_ADD_PRESET_WIDGET:true" \
"TIMING_DH:150" \
"TIMING_DQSCK:400" \
"TIMING_DQSQ:200" \
"TIMING_DQSS:0.25" \
"TIMING_DS:75" \
```

```
"TIMING_DSH:0.2" \
"TIMING_DSS:0.2" \
"TIMING_FAW:30" \
"TIMING_IH:275" \
"TIMING_INIT:200" \
"TIMING_IS:200" \
"TIMING_MODE:0" \
"TIMING_MRD:4" \
"TIMING_QH:0.38" \
"TIMING_QSH:0.38" \
"TIMING_RAS:36" \
"TIMING_RC:49.5" \
"TIMING_RCD:13.5" \
"TIMING_REFI:7.8" \
"TIMING_RFC:350" \
"TIMING_RP:13.5" \
"TIMING_RRD:7.5" \
"TIMING_RTP:7.5" \
"TIMING_WR:15" \
"TIMING_WTR:5" \
"TURNAROUND_RTR_DIFFRANK:1" \
"TURNAROUND_RTW_DIFFRANK:1" \
"TURNAROUND_WTR_DIFFRANK:1" \
"TURNAROUND_WTW_DIFFRANK:0" \
"USER_POWER_DOWN:false" \
"USER_SELF_REFRESH:false" \
"WIDTH:16" \
"WRITE_LEVELING:ENABLE" \
"WRITE_RECOVERY:5" \
"ZQ_CALIB_PERIOD:200" \
"ZQ_CALIB_TYPE:0" \
"ZQ_CALIB_TYPE_TEMP:0" \
"ZQ_CAL_INIT_TIME:512" \
"ZQ_CAL_L_TIME:256" \
"ZQ_CAL_S_TIME:64" } -inhibit_configurator 0
# Exporting core PF_DDR3_SS to TCL done
```

The following is an example of an exported Tcl script for a HDL+ core.

```
# Exporting core pattern_gen_checker to TCL
# Exporting Create HDL core command for module pattern_gen_checker create_hdl_core -file
{X:/10_docs_review/12.0_Release/pcie_demo_tcl_example/DG0756_PF_PCIE_EP/new/project/hdl/
PATTERN_GEN_CHECKER.v} -module {pattern_gen_checker} -library {work} -package {}
# Exporting BIF information of HDL core command for module pattern_gen_checker
```

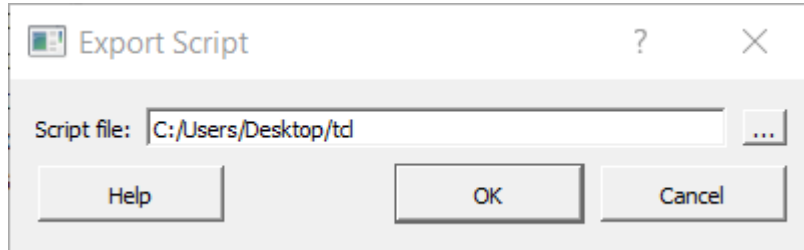
The following is an example of an exported Tcl script for a SgCore (PF\_TX\_PLL).

```
# Exporting core PCIe_TX_PLL to TCL
# Exporting Create design command for core PCIe_TX_PLL
create_and_configure_core -core_vlnv {Actel:SgCore:PF_TX_PLL:1.0.115} -component_name
{PCIe_TX_PLL} -params {
"CORE:PF_TX_PLL" \
"FAMILY:26" \
"INIT:0x0" \
"PARAM_IS_FALSE:false" \
"SD_EXPORT_HIDDEN_PORTS:false" \
"TxPLL_AUX_LOW_SEL:true" \
"TxPLL_AUX_OUT:125" \
"TxPLL_CLK_125_EN:true" \
"TxPLL_DYNAMIC_RECONFIG_INTERFACE_EN:false" \
"TxPLL_EXT_WAVE_SEL:0" \
"TxPLL_FAB_LOCK_EN:false" \
"TxPLL_FAB_REF:200" \
"TxPLL_JITTER_MODE_SEL:10G SyncE 32Bit" \
"TxPLL_MODE:NORMAL" \
"TxPLL_OUT:2500.000" \
"TxPLL_REF:100" \
"TxPLL_SOURCE:DEDICATED" \
"TxPLL_SSM_DEPTH:0" \
"TxPLL_SSM_DIVVAL:1" \
"TxPLL_SSM_DOWN_SPREAD:false" \
"TxPLL_SSM_FREQ:64" \
"TxPLL_SSM_RAND_PATTERN:0" \
"VCOFREQUENCY:1600" } -inhibit_configurator 1
# Exporting core PCIe_TX_PLL to TCL done
```

#### 4.1.4 Hierarchical Export Component Description (Tcl)

This option exports the complete design and its subcomponents to Tcl. When this option is executed on a SmartDesign, it iterates through all the instances and collects information about the pins, groups, and nets in the SmartDesign. All the TCL scripts generated are exported to a folder you select when the Hierarchical Export Component Description (Tcl) option executes.

**Figure 4-4. Export Script for Hierarchical Export Description (Tcl)**



This exported folder consists of the following files and subfolders:

**Table 4-1. Subfolders in the Exported File**

Subfolders	Description
HDL	Contains all of the imported HDL source files.
Stimulus	Contains all of the imported HDL stimulus files.
Components	Contains all of the Tcl files of the components used in the SmartDesign.

**Table 4-2. Files in the Exported File**

Files	Description
hdl_source.tcl	Contains the tcl for imported and linked files.
<component>_recursive.tcl	Top-level tcl used to recreate the design.
Un_Supported_Cores_List.txt	Contains all the cores for which the export function cannot be performed.

To run this option, right-click the desired component for which the information must be exported in Tcl in the Design/ Stimulus Hierarchy, and then choose **Hierarchical Export Component Description (Tcl)**.

Figure 4-5. Hierarchical Export Component Description (Tcl) Option

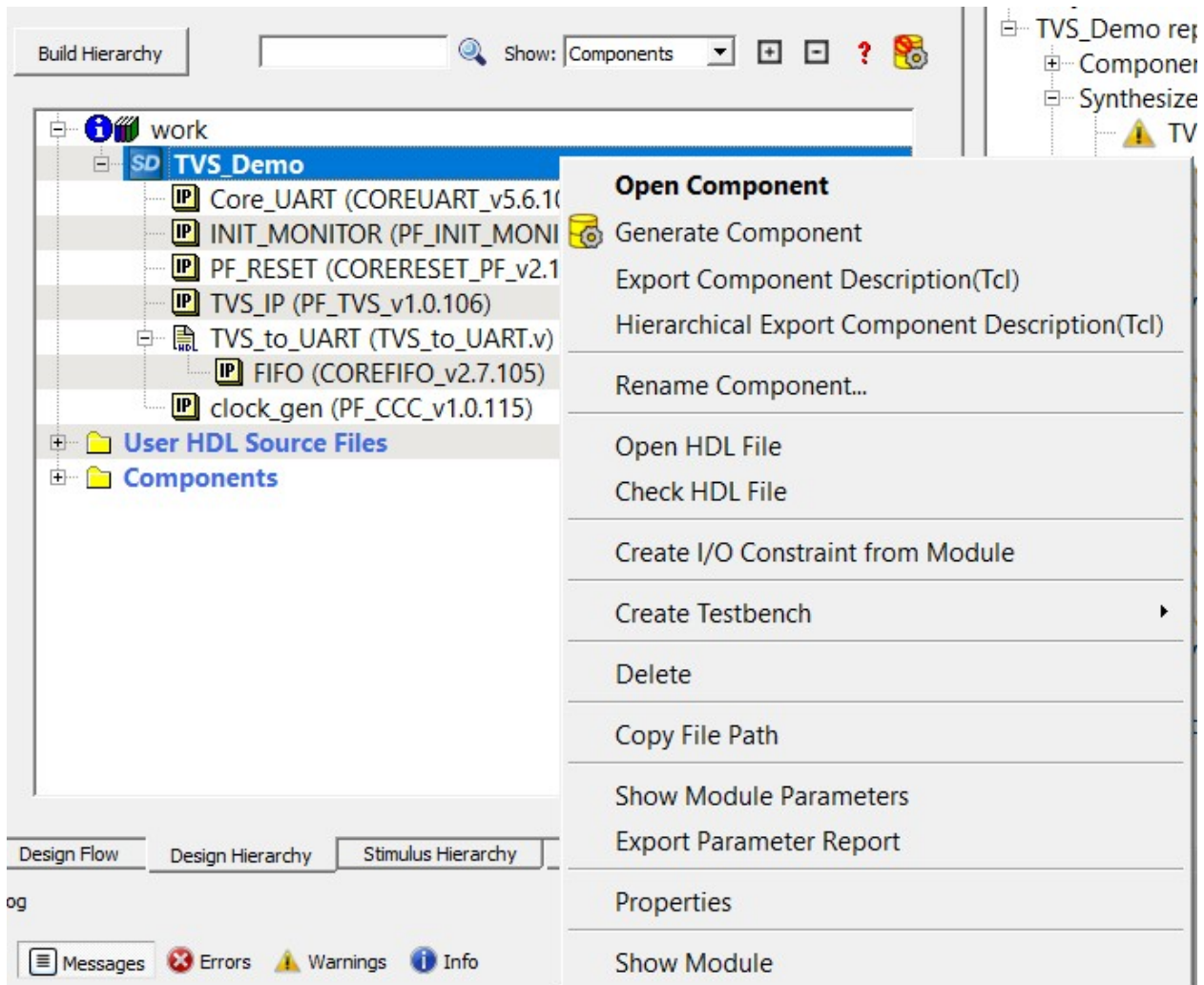
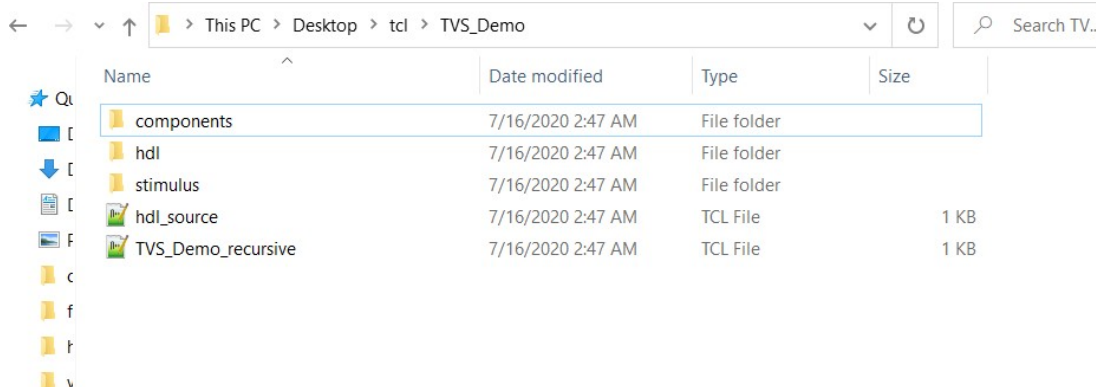
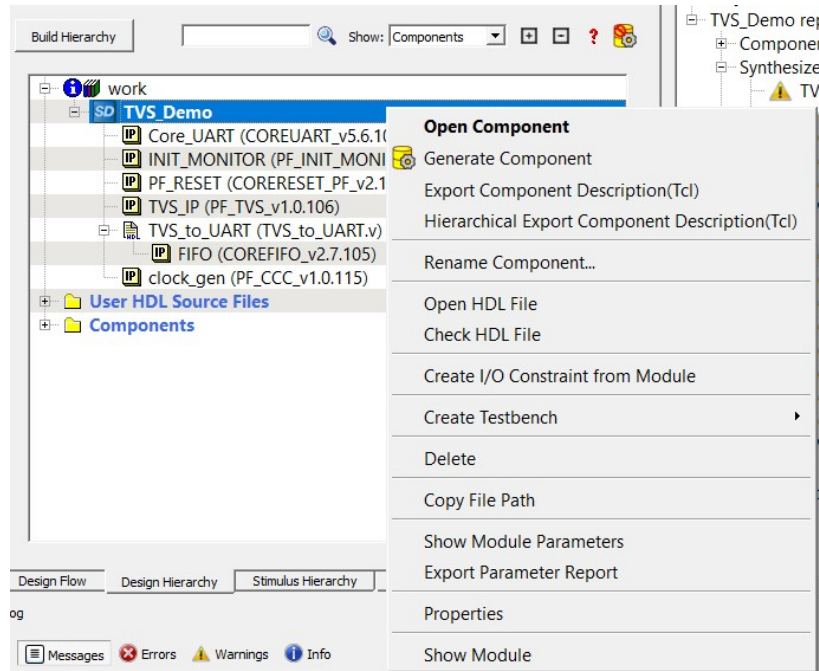


Figure 4-6. Hierarchical Export Component Description (Tcl) Option After Right-clicking a Component



The following figure shows an example of files that have been generated in the Exported folder.

Figure 4-7. Files and Subfolders Generated in the Exported Folder



### Limitations

Hierarchical Export Component Description (Tcl) support is not available for blocks.

### Messages

The following table lists the messages that the tool generates in the log window.

Table 4-3. Tool-generated Messages

Message	Description
Error: Please check the permission of the specified folder.	The folder you specified is not writable.
Error: Unable to Export Component 'top' to path	The export operation was not successful.
Info: Component 'top' exported successfully to path	The export operation was successful.

### 4.1.5 Generating a SmartDesign Component


Before your SmartDesign component can be used by downstream processes, such as synthesis and simulation, you must generate it.



Click the **Generate** (Generate button icon) button to generate a SmartDesign component. An HDL file is generated in the directory `<libero_project>/components/<library>/<yourdesign>`.

**Note:** The generated HDL file will be deleted when your SmartDesign design is modified and saved to ensure synchronization between your SmartDesign component and its generated HDL file.


Generating a SmartDesign component may fail if there are any [design rule checking \(DRC\) errors](#). DRC errors must be corrected before you generate your SmartDesign design.

If the ports of a sub-design change, the parent SmartDesign component is annotated with the icon  in the **Design Hierarchy** tab of the Design Explorer.

### Generate Recursively vs. Non-Recursively

You can generate a SmartDesign component recursively or nonrecursively. These options are set in the [Project Preference Dialog Box - Design Flow Preferences section](#).

**Table 4-4. Design Flow Preferences Options**

Option	Description
Recursive generation	Clicking the <b>Generate</b> button generates all sub-design SmartDesigns, depth first. The parent SmartDesign is generated only if all the sub-designs generate successfully.
Non-Recursive generation	Clicking the <b>Generate</b> button generates the specified SmartDesign only. The generation can be marked as successful even if a sub-design is “ungenerated” (either never attempted or unsuccessful). An ungenerated component is annotated with the icon  in the <b>Design Hierarchy</b> tab of the Design Explorer.

## 4.2 Create Core from HDL

You can instantiate any HDL module and connect it to other blocks inside SmartDesign. However, there are situations where you may want to extend your HDL module with more information before using it inside SmartDesign.

- If you have an HDL module that contains configurable parameters or generics.
- If your HDL module is intended to connect to a processor subsystem and has implemented the appropriate bus protocol, then you can add a bus interface to your HDL module so that it can easily connect to the bus inside of SmartDesign.

### 4.2.1 Creating a Core from Your HDL

To create a core from your HDL:

1. Import or create a new HDL source file; the HDL file appears in the Design Hierarchy.
2. Select the HDL file in the Design Hierarchy and click the HDL+ icon.

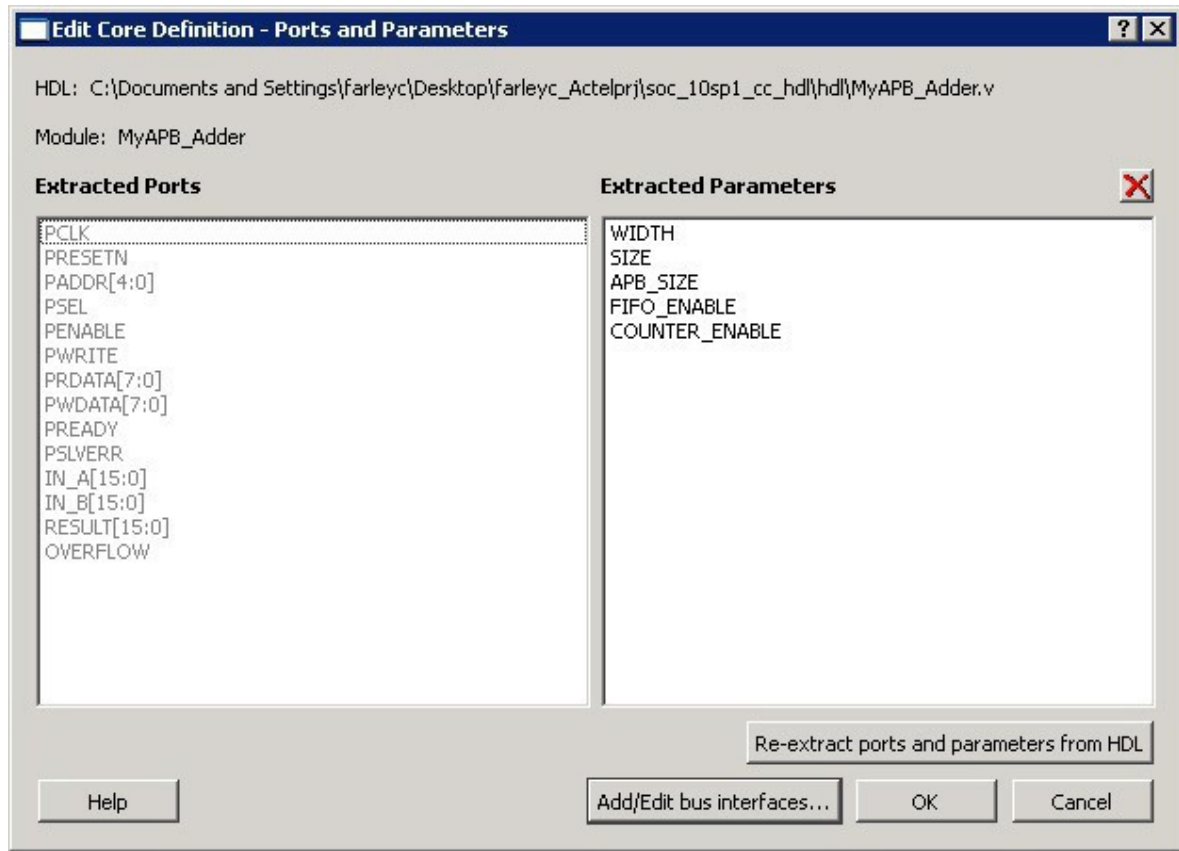
**OR**

Right-click the HDL file and choose **Create Core from HDL**.

The Edit Core Definition – Ports and Parameters dialog box shows the ports and parameters that were extracted from your HDL module.

3. Remove parameters that are not intended to be configurable by selecting them from the list and clicking the **X** icon. Remove parameters that are used for internal variables, such as state machine enumerations. If you removed a parameter by accident, click **Re-extract ports and parameters from HDL file** to reset the list so it matches your HDL module.

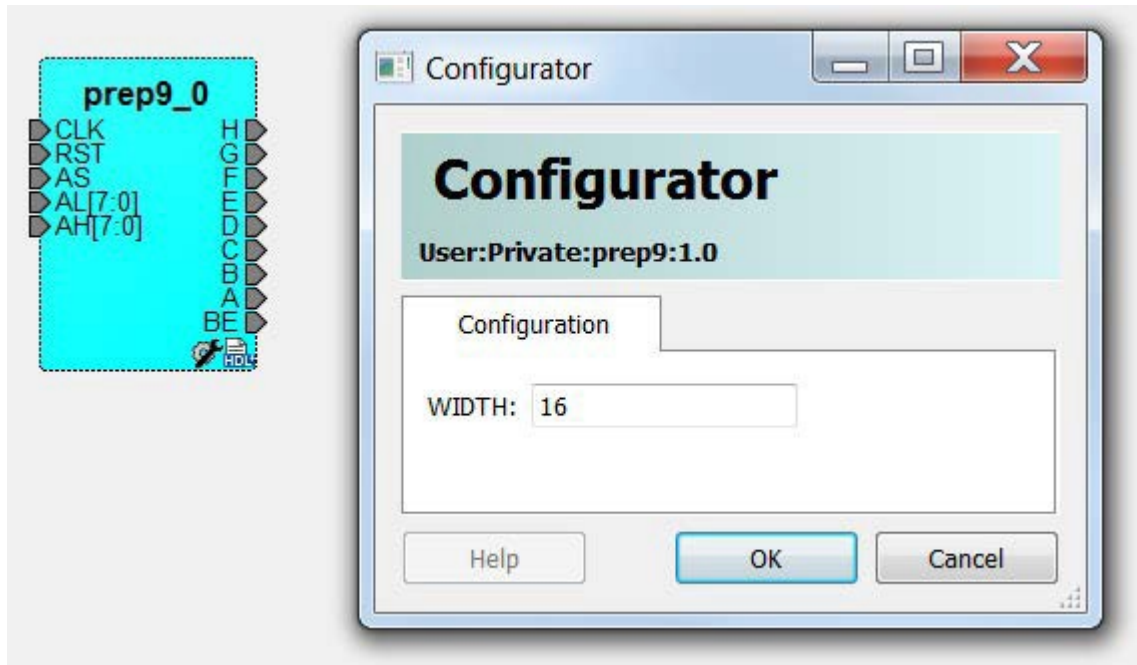
**Figure 4-8. Edit Core Definition - Ports and Parameters Dialog Box**



4. (Optional) To [add bus interfaces](#) to your core, click **Add/Edit Bus Interfaces**.
5. After you specify the information, your HDL turns into an HDL+ icon in the Design Hierarchy. Click and drag your HDL+ module from the Design Hierarchy to the **Canvas**.
6. If you added bus interfaces to your HDL+ core, it appears in your SmartDesign, with a bus interface pin you can use to connect to the appropriate bus IP core.
7. If your HDL+ has configurable parameters, double-clicking the object on the Canvas (or right-clicking and choosing **Configure**) displays a configuration dialog box that allows you to set these values. On generation, the specific configuration values per instance are written to the SmartDesign netlist.



Figure 4-9. HDL+ Instance and Configuration Dialog Box



8. To open the HDL file inside the text editor, right-click the instance and choose **Modify HDL**.

#### 4.2.2 Editing Core Definitions

After you create a core definition, you can edit it by selecting your HDL+ module in the Design Hierarchy and clicking the HDL+ icon.

#### 4.2.3 Removing Core Definitions

If you do not want the extended information on your HDL module, you can revert it to a regular HDL module.

1. Right-click the HDL+ in the Design Hierarchy and choose **Remove Core Definition**.
2. After removing your definition, update the instances in your SmartDesign that referenced the core you removed by right-clicking the instance and choosing **Replace Component for Instance**.

### 4.3 Designing with HDL

This section describes how to use HDL to implement designs using the HDL Editor.

#### 4.3.1 Create HDL

Create HDL opens the HDL editor with a new VHDL or Verilog file. Your new HDL file is saved to your `/hdl` directory and all modules created in the file appear in the Design Hierarchy.

You can use VHDL and Verilog to implement your design.

To create an HDL file:

1. In the Design Flow window, double-click **Create HDL**. The Create new HDL file dialog box opens.
2. Select your **HDL Type**. Choose whether to **Initialize file with standard template** to populate your file with default headers and footers. The HDL Editor workspace opens.
3. Enter a name. Do not enter a file extension because Libero SoC adds one for you. The filename must follow Verilog or VHDL file naming conventions.
4. Click **OK**.
5. After creating your HDL file, click the **Save** button to save your file to the project.

#### 4.3.2 Using the HDL Editor

The HDL Editor is a text editor for editing HDL source files. In addition to regular editing features, the editor provides keyword highlighting, line numbering, and a syntax checker.

You can have multiple files open at one time in the HDL Editor workspace. Click the tabs to move between files.

To start the editor:

1. Right-click inside the HDL Editor to show the **Edit** menu items. Available editing functions include the following. These functions are also available in the toolbar.
  - Cut, copy, and paste
  - Go to line
  - Comment and Uncomment Selection
  - Check HDL File
  - Word Wrap mode (disabled by default)
  - Font size changes. To increase or decrease the font size of text in the editor, right-click in the editor and choose **Increase Font** or **Decrease Font**.
2. Save your file to add it to your Libero SoC project by selecting **Save** from the **File** menu.  
**OR**  
Clicking the **Save** icon in the toolbar.
3. To print your project, select **Print** from the **File** menu or the toolbar.

**Note:** To avoid conflicts between changes made in your HDL files, use one editor for all of your HDL edits.

#### 4.3.3 HDL Syntax Checker

The HDL syntax checker parses through HDL files to identify typographical mistakes and syntactical errors.

To run the syntax checker:

1. From the **Files** list, double-click the HDL file to open it.
2. Right-click in the body of the HDL editor and choose **Check HDL File**. The syntax checker parses the selected HDL file and looks for typographical mistakes and syntactical errors. Warning and error messages for the HDL file appear in the Libero SoC Log window.

#### 4.3.4 Commenting Text

You can comment text as you type in the HDL Editor, or you can comment out blocks of text by selecting a group of text and applying the **Comment** command.

To comment or uncomment out text:

1. Type your text.
2. Select the text.
3. Right-click inside the editor and choose **Comment Selection** or **Uncomment Selection**.

#### 4.3.5 Find

You can search for a whole or partial word, with or without matching the case.

To find an entire or partial word:

1. From the **File** menu, choose **Find**. The Find dialog box appears below the Log/Message window.
2. Enter the text you want to find.
3. Use the options to match case, whole word, and/or regular expression.

**Note:** The editor also supports a Find to Replace function.

#### 4.3.6 Editing Columns

To select a column of text to edit, select the column, and then press ALT+click.

#### 4.3.7 Importing HDL Source Files

To import an HDL source file:

1. In the Design Flow window, right-click **Create HDL** and choose **Import Files**. The Import Files window appears.
  2. Go to the location where the HDL file is located.
  3. Select the file to import and click **Open**.
- Note:** You can import SystemVerilog (\*.sv), Verilog (\*.v), and VHDL (\*.vhd/\* .vhd1) files.

### 4.3.8 Mixed HDL Support in Libero SoC

To use mixed HDL in the Libero SoC, you require:

- ModelSim ME Pro
- SynplifyPro to synthesize a mixed HDL design

When you [create a project](#), you select a preferred language. The HDL files generated in the flow are created in the preferred language. If your preferred language is Verilog, the post-synthesis and post-layout netlists are in Verilog 2001.

The language used for simulation is the same language as the last compiled test bench. For example, if `tb_top` is in Verilog, `<Fam>.v` is compiled.

## 4.4 Designing with Block Flow

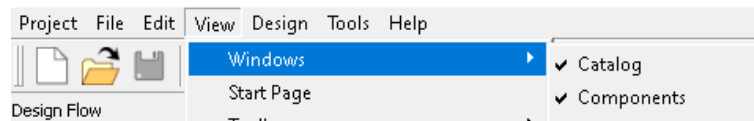
For information about designing with Block Flow, see [Designing with Blocks for Libero SoC Enhanced Constraint Flow](#).

## 4.5 Viewing Configured Components and SmartDesigns in a Project

Libero SoC supports the Components view that lists all the configured components and SmartDesigns in a project.

To open the view, click **View > Windows > Components**. Follow the same procedure to close the Components view.

**Figure 4-10. Opening the Components View**



When you open the Components view, it appears as a tab in the left-side area of the Libero SoC and lists all configured components and SmartDesigns in the project. If you open the Components view when a project is not open, the tab is displayed as blank.

**Figure 4-11. Example of the Components View**

Components

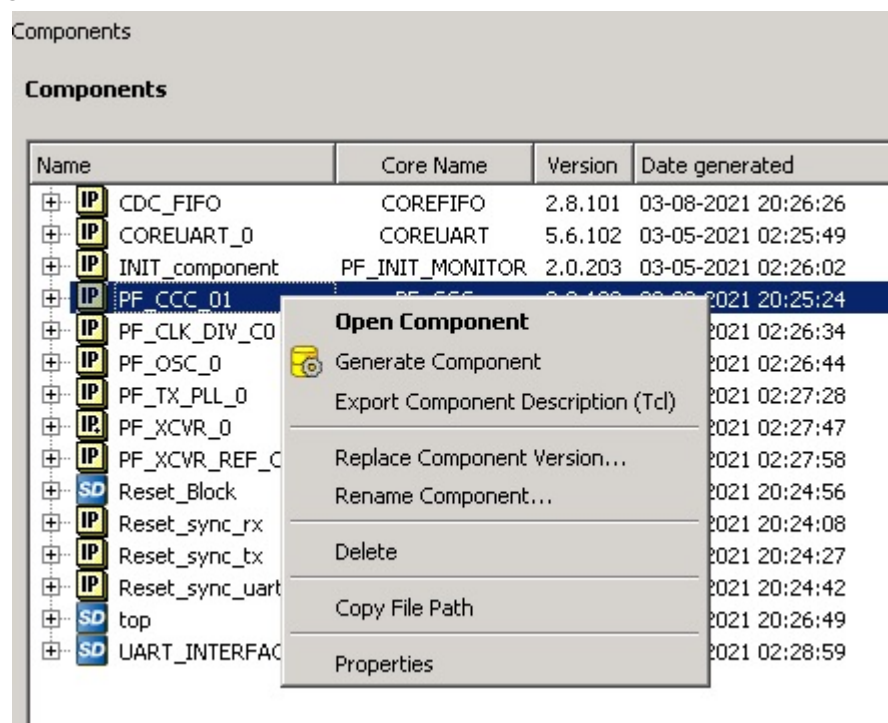
**Components**

Name	Core Name	Version	Date generated
+ IP CDC_FIFO	COREFIFO	2.8.101	03-08-2021 20:26:26
+ IP COREUART_0	COREUART	5.7.100	03-08-2021 20:34:05
+ IP INIT_component	PF_INIT_MONITOR	2.0.203	03-05-2021 02:26:02
+ IP PF_CCC_01	PF_CCC	2.2.100	03-08-2021 20:25:24
+ IP PF_CLK_DIV_C0	PF_CLK_DIV	1.0.103	03-05-2021 02:26:34
+ IP PF_OSC_0	PF_OSC	1.0.102	03-05-2021 02:26:44
+ IP PF_TX_PLL_0	PF_TX_PLL	2.0.300	03-05-2021 02:27:28
+ IP PF_XCVR_0	PF_XCVR_ERM	3.1.100	03-05-2021 02:27:47
+ IP PF_XCVR_REF_CLK_0	PF_XCVR_REF_CLK	1.0.103	03-05-2021 02:27:58
+ SD Reset_Block			03-08-2021 20:24:56
+ IP Reset_sync_rx	CORERESET_PF	2.2.107	03-08-2021 20:24:08
+ IP Reset_sync_tx	CORERESET_PF	2.2.107	03-08-2021 20:24:27
+ IP Reset_sync_uart	CORERESET_PF	2.2.107	03-08-2021 20:24:42
+ SD top			03-08-2021 20:26:49
+ SD UART_INTERFACE			03-05-2021 02:28:59

Design Flow   Design Hierarchy   Stimulus Hierarchy   **Components**   Catalog   Files

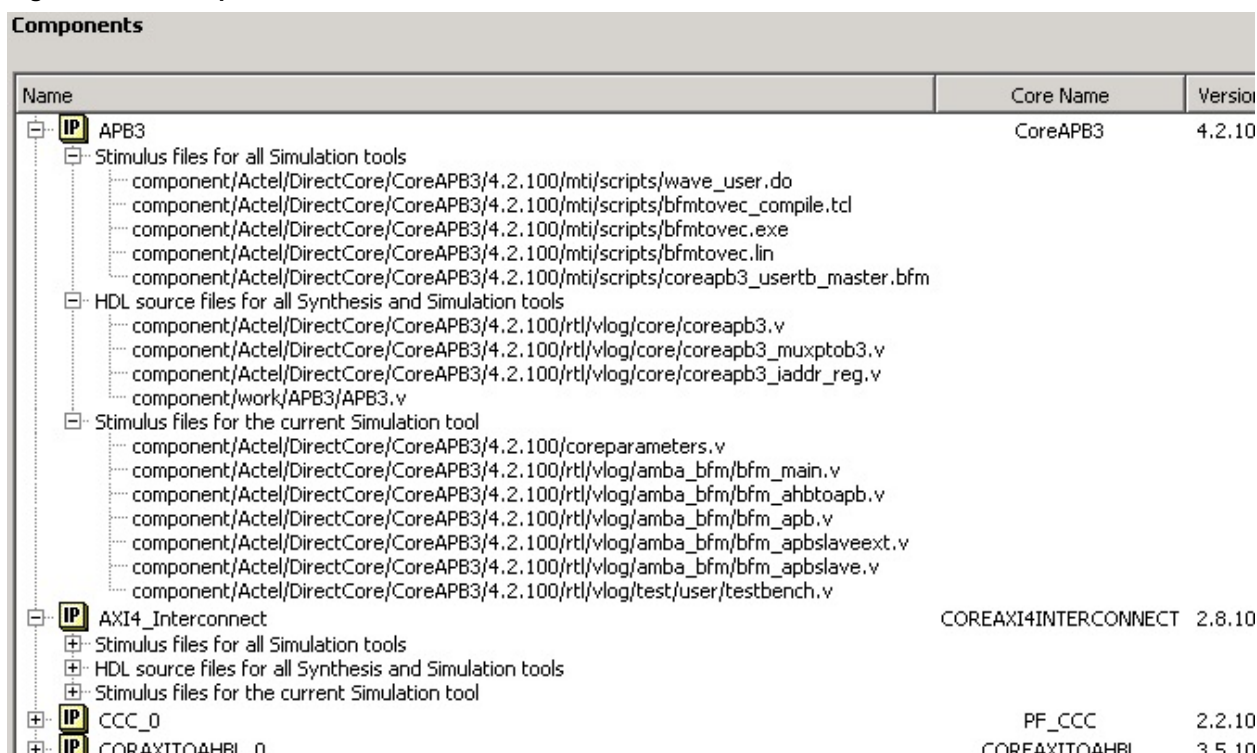
Right-clicking a component in the Components view tab displays a menu that is similar to the one that appears when you right-click entries in the Components section of the Design Hierarchy.

Figure 4-12. Right-Click Menu in Components View



HDL source files present at various levels in <project>/Component/work/<core\_name>/ appear under **HDL Source Files**.

Figure 4-13. Example of HDL Source Files



The **Vendor**, **Library**, **CoreName**, and **Version** columns show the appropriate information.

The timestamp shown for **Generation** appears in the **Date Generated** column and gets updated when the component gets regenerated.

## 4.6 Create a New SmartDesign Testbench

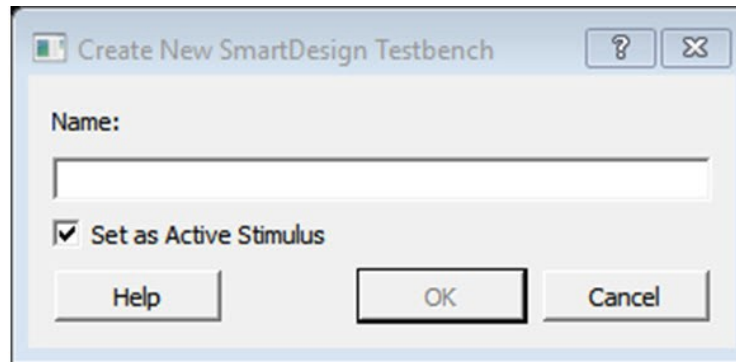
The SmartDesign Testbench component can be the top level of the design or, following a successful generation, it can be used as a lower level SmartDesign Testbench component in another design.

1. From the **File** menu, choose **New > SmartDesign Testbench**.  
**OR**

In the Design Flow window, double-click **Create SmartDesign Testbench**.

The Create New SmartDesign Testbench dialog box appears.

**Figure 4-14. Create New SmartDesign Testbench**



2. Enter a name.  
**Note:** The component name must be unique in your project.
3. To make this SmartDesign Testbench your active stimulus, check **Set as Active Stimulus**.
4. Click **OK**. The component appears in the **Stimulus Hierarchy** tab of the Design Explorer.

For more information, see the [SmartDesign User Guide](#).

## 4.7 HDL Testbench

To create an HDL testbench, right-click a SmartDesign in the Design Hierarchy and choose **Create Testbench > HDL**. The HDL testbench instantiates the selected SmartDesign into the component automatically.

To create a new testbench file, double-click **Create HDL Testbench** to display the Create New HDL Testbench dialog box. This dialog box allows you to create a new testbench file, with the option to include standard testbench content and your design data.

### 4.7.1 HDL Type

Set **HDL Type** to **Verilog** or **VHDL** for the testbench.

### 4.7.2 Name

Specify a testbench file name. A \*.v or a \*.vhd file is created and opened in the HDL Editor.

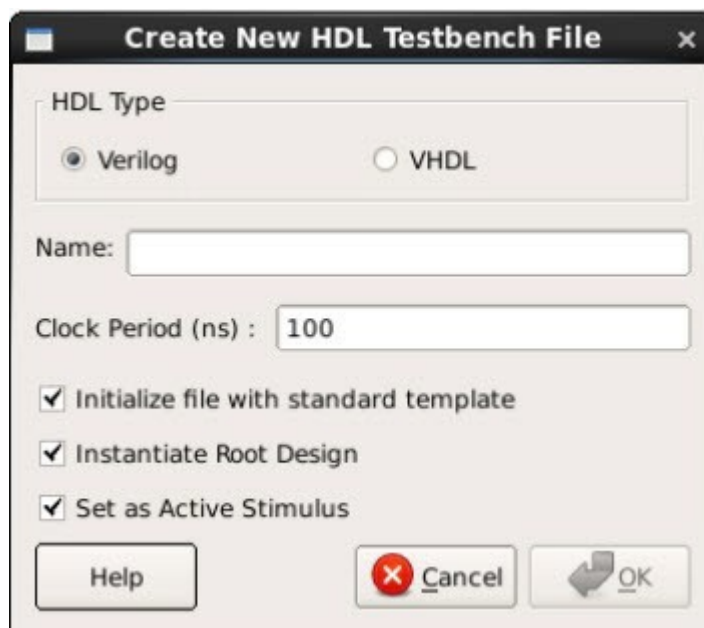
### 4.7.3 Clock Period (ns)

Enter a clock period in nanoseconds (ns) for the clock to drive the simulation. The default value is 100 ns (10 MHz). Libero creates in the testbench a SYSCLK signal with the specified frequency to drive the simulation.

**Table 4-5. Clock Period Options**

Option	Description
Set as Active Stimulus	Sets the HDL Testbench as the stimulus file to use for simulations. The active stimulus file/testbench is included in the <code>run.do</code> file that Libero generates to drive the simulation. Setting one testbench as the Active Stimulus is necessary when there are multiple testbenches in the stimulus hierarchy.
Initialize with Standard Template	Adds boilerplate for a minimal standard test module. This test module does not include an instantiation of the root module under test.
Instantiate Root Design	Creates a test module that includes an instance of the root module under test, and clocking logic in the test module that drives the base clock of the root module under test.

**Figure 4-15. Create New HDL Testbench File Dialog Box**





**Figure 4-16. HDL Testbench Example - VHDL, Standard Template, and Root Design Enabled**

```

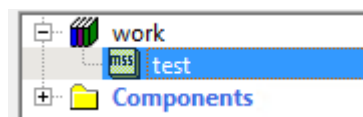
1  -----
2  -- Created by Microsemi SmartDesign Mon Mar 27 15:07:29 2017
3  -- Testbench Template
4  -- This is a basic testbench that instantiates your design with basic
5  -- clock and reset pins connected. If your design has special
6  -- clock/reset or testbench driver requirements then you should
7  -- copy this file and modify it.
8  -----
9
10 -----
11 -- Company: <Name>
12 --
13 -- File: counter_tb.vhd
14 -- File history:
15 --   <Revision number>: <Date>: <Comments>
16 --   <Revision number>: <Date>: <Comments>
17 --   <Revision number>: <Date>: <Comments>
18 --
19 -- Description:
20 --
21 -- <Description here>
22 --
23 -- Targeted device: <Family::PolarFire> <Die::MPF200TS_ES> <Package::Fully Bonded Package>
24 -- Author: <Name>
25 --
26 -----
27
28
29 library ieee;
30 use ieee.std_logic_1164.all;
31
32 entity counter_tb is
33 end counter_tb;
34
35 architecture behavioral of counter_tb is
36
37     constant SYSCLK_PERIOD : time := 100 ns; -- 10MHZ
38
39     signal SYSCLK : std_logic := '0';
40     signal NSYSRESET : std_logic := '0';
41
42     component count16

```

## 4.8 Import MSS

If you use a device from the PolarFire SoC family, a new tool Import MSS is added under **Create Design** in the design flow process. When you run this tool, an Import MSS Component dialog box allows you to select the MSS Component (\*.czz) files. After the selected \*.czz module file is imported, it appears in the Design Hierarchy window preceded by a new icon, as shown in the following figure.

**Figure 4-17. MSS Component File in Design Hierarchy Window**



## 4.9 Verify Pre-Synthesized Design - RTL Simulation

To perform pre-synthesis simulation, either:

- Double-click **Simulate** under **Verify Pre-Synthesized Design** in the Design Flow window.
- OR**
- In the Stimulus Hierarchy, right-click the testbench and choose **Simulate Pre-Synth Design > Run**

The default tool for RTL simulation in Libero SoC PolarFire is ModelSim™ ME Pro.

ModelSim ME Pro is a custom edition of ModelSim PE that is integrated into Libero SoC's design environment. ModelSim for Microchip is an OEM edition of Mentor Graphics ModelSim tools. ModelSim ME Pro supports mixed VHDL, Verilog, and SystemVerilog simulation. It works only with Microchip simulation libraries and is supported by Microchip.



Libero SoC supports other editions of ModelSim. To use other ModelSim editions, do not install ModelSim ME from the Libero SoC media.

**Note:** ModelSim for Microchip includes online help and documentation. After starting ModelSim, click the **Help** menu to display the help.

For more information about simulations in Libero SoC, see the following topics:

- [Simulation Options](#)
- [Selecting a Stimulus File for Simulation](#)
- [Selecting additional modules for simulation](#)
- [Performing Functional Simulation](#)

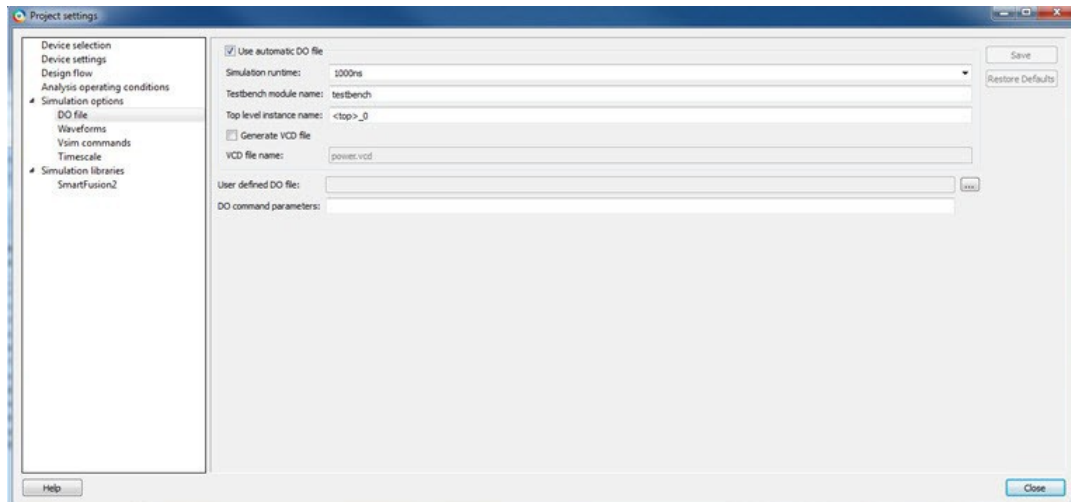
### 4.9.1 Project Settings: Simulation - Options and Libraries

The Project Settings dialog box allows you to change how Libero SoC handles Do files in simulation and imports your DO files. You can also set simulation run time, change the DUT name used in your simulation, and change your library mapping.

To access the Project Settings dialog box:

1. From the **Project** menu, choose **Project Settings**.
2. In the left pane, click **Simulation options** or **Simulation libraries** to expand the options.
3. For **Simulation options**, click the option you want to edit:
  - **DO file**
  - **Waveforms**
  - **Vsim commands**
  - **Timescale**
4. For **Simulation libraries**, click the library whose path you want to change.

**Figure 4-18. Project Settings: DO File**



### 4.9.2 DO file

**Table 4-6. DO File Options**

Option	Description
Use automatic DO file	Allows Project Manager to create a DO file automatically that will allow you to simulate your design.
Simulation Run Time	Specifies how long the simulation should run. If the value is 0 or if the field is empty, the run command is omitted from the <code>run.do</code> file.

.....continued	
Option	Description
Testbench module name	Specifies the name of your testbench entity name. Default is <code>testbench</code> , which is the value that WaveFormer Pro uses.
Top Level instance name	Default is <code>&lt;top_0&gt;</code> , the value that WaveFormer Pro uses. Project Manager replaces <code>&lt;top&gt;</code> with the top-level macro when you run simulation (presynth/postsynth/postlayout).
Generate VCD file	Clicking the check box generates a VCD file.
VCD file name	Specifies the name of your generated VCD file. The default is <code>power.vcd</code> . To change the name, click <b>power.vcd</b> and enter the new name.
User defined DO file	Enter the DO file name or click the <b>Browse</b> button to go to the file.
DO command parameters	Text in this field is added to the DO command.

### 4.9.3 Waveforms

**Table 4-7. Waveforms Options**

Option	Description
Include DO file	Including a DO file allows you to customize the set of signal waveforms displayed in ModelSim.
Display waveforms for	<p>You can display signal waveforms for the top-level testbench or for the design under test.</p> <ul style="list-style-type: none"> <li><b>top-level testbench</b> = Project Manager outputs the line <code>addwave/testbench/*</code> in the DO file <code>run.do</code>.</li> <li><b>DUT</b> = Project Manager outputs the line <code>add wave/testbench/DUT/*</code> in the <code>run.do</code> file.</li> </ul>
Log all signals in the design	Saves and logs all signals during simulation.

### 4.9.4 Vsim Commands

**Table 4-8. Vsim Command Options**

Option	Description
Resolution	The default is 1ps. Some custom simulation resolutions might not work with your simulation library. Consult your simulation help for information about how to work with your simulation library and detect infinite zero-delay loops caused by high resolution values.

.....continued	
Option	Description
Additional options	<p>Text entered in this field is added to the vsim command.</p> <ul style="list-style-type: none"> <li>• <b>SRAM ECC Simulation</b> = two options can be added to specify the simulated error and correction probabilities of all ECC SRAMs in the design: <ul style="list-style-type: none"> <li>– -gERROR_PROBABILITY=&lt;value&gt;, where 0 &lt;= value &lt;= 1</li> <li>– -gCORRECTION_PROBABILITY=&lt;value&gt;, where 0 &lt;= value &lt;= 1</li> </ul> </li> <li>• During Simulation, the SB_CORRECT and DB_DETECT flags on each SRAM block are raised based on generated random numbers that fall below the specified &lt;value&gt;s.</li> </ul> <p>When you run the Post-Layout Simulation Tool, a <code>run.do</code> file is created, which consists of information that must be sent to a simulator. To run a simulation on a corner, select an SDF corner along with the type of delay needed from one of the options in SDF timing delays section.</p> <ul style="list-style-type: none"> <li>• SDF Corners: <ul style="list-style-type: none"> <li>– Slow Process, Low Voltage and High Temperature</li> <li>– Slow Process, Low Voltage and Low Temperature</li> <li>– Fast Process, High Voltage and Low Temperature</li> </ul> </li> <li>• SDF Timing Delays <ul style="list-style-type: none"> <li>– Minimum</li> <li>– Typical</li> <li>– Maximum</li> </ul> </li> <li>• Disable pulse filtering during SDF based simulations: This option disables pulse filtering during SDF simulations.</li> <li>• After you select the corner, appropriate files for simulation are written in the <code>run.do</code> file, as shown in the following figure.</li> </ul>

**Figure 4-19. Files Written to the run.do File**

```
vlog -sv-work postlayout "${PROJECT_DIR}/designer/sd1/sd1_fast_hv_lt_ba.v"
vsim -L PolarFire -L postlayout -t lps -sdfmax
/sd1=${PROJECT_DIR}/designer/sd1/sd1_fast_hv_lt_ba.sdf +pulse_int_e/1
+pulse_int_r/1 +transport_int_delays postlayout.sdl
```

#### 4.9.5 Timescale

**Table 4-9. Timescale Options**

Option	Description
TimeUnit	Time base for each unit. Enter a value and select s, ms, us, ns, ps, or fs from the drop-down list. The default setting is ns.
Precision	Enter a value and select s, ms, us, ns, ps, or fs from the drop-down list. The default setting is ps.

## 4.9.6 Simulation Libraries

**Table 4-10. Simulation Libraries Options**

Option	Description
Restore Defaults	Sets the library path to default from your Libero SoC installation.
Library path	Allows you to change the mapping for your Verilog and VHDL simulation libraries. Enter the pathname or click the <b>Browse</b> button to go to your library directory.

## 4.9.7 Selecting a Stimulus File for Simulation

Before running simulation, associate a testbench. If you try to run simulation without an associated testbench, the Libero SoC Project Manager prompts you to associate a testbench or open ModelSim without a testbench.

To associate a stimulus:

1. Run the simulation.  
**OR**  
 In the Design Flow window, under **Verify Pre-Synthesized Design**, right-click **Simulate** and choose **Organize Input Files > Organize Stimulus Files**.  
 The Organize Stimulus Files dialog box appears.
2. Associate your testbenches. In the Organize Stimulus Files dialog box, all the stimulus files in the current project appear in **Source Files** in the **Project** list box. Files already associated with the block appear in the **Associated Source Files** list box.  
 In most cases, you will have one testbench associated with your block. However, if you want simultaneous association of multiple testbench files for one simulation session, as in the case of PCI cores, add multiple files to the **Associated Source Files** list.
  - To add a testbench: Select the testbench you want to associate with the block in **Source Files** in the **Project** list box and click **Add** to add it to the **Associated Source Files** list.
  - To remove a testbench or change the files in the **Associated Source Files** list box: Select the files and click **Remove**.
  - To order testbenches: Use the up and down arrows to arrange the order in which you want the testbenches compiled. The top level-entity should be at the bottom of the list.
3. When you are satisfied with the **Associated Source Files** list, click **OK**.

## 4.9.8 Selecting Additional Modules for Simulation

Libero SoC passes all the source files related to the top-level module to simulation.

If you need additional modules in simulation:

1. Right-click **Simulate** in the Design Flow window and choose **Organize Input Files > Organize Source Files**. The Organize Files for Simulation dialog box appears.
2. From the **Simulation Files** in the **Project** list, select the HDL modules you want to add and click **Add** to add them to the **Associated Stimulus Files** list.

## 4.9.9 Performing Functional Simulation

To perform functional simulation:

1. Create your testbench.
2. In the Design Flow window, select **Implement Design > Verify Post-Synthesis Implementation**.
3. Right-click **Simulate** and choose **Organize Input Files > Organize Simulation Files** from the right-click menu. In the Organize Files for Source dialog box, all the stimulus files in the current project appear under **Source Files** in the **Project** list box. Files associated with the block appear in the **Associated Source Files** list box.

4. In most cases, you will have one testbench associated with your block. However, if you want simultaneous association of multiple testbench files for one simulation session, as in the case of PCI cores, add multiple files to the **Associated Source Files** list.
  - To add a testbench: In the **Source Files** of the **Project** list box, select the testbench you want to associate with the block. Click **Add** to add it to the **Associated Source Files** list.
  - To remove a testbench or change the files in the **Associated Source Files** list box: Select the files and click **Remove**.
5. When you are satisfied with the **Associated Simulation Files** list, click **OK**.
6. To start ModelSim ME Pro, right-click **Simulate** in the Design Hierarchy window and choose **Open Interactively**. ModelSim starts and compiles the appropriate source files. When the compilation completes, the simulator runs for 1 microsecond and the Wave window shows the simulation results.
7. Scroll in the Wave window to verify that the logic of your design functions as intended. Use the zoom buttons to zoom in and out as necessary.
8. When finished, select **Quit** from the **File** menu.

## 5. Libero SoC Constraint Management

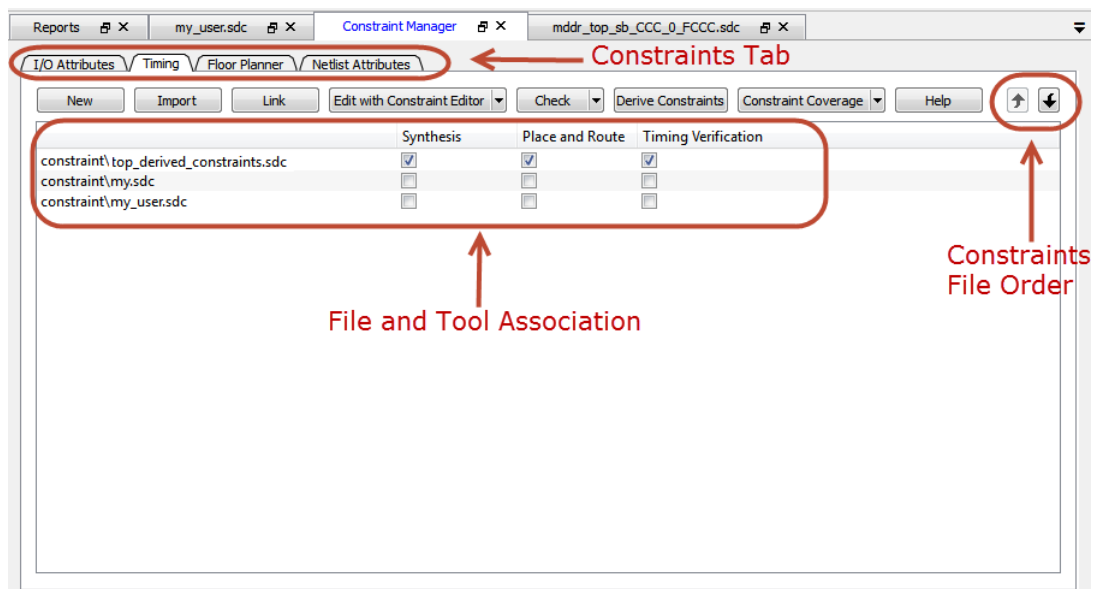
In the FPGA design world, constraint files are as important as design source files. Constraint files are used throughout the FPGA design process to guide FPGA tools to achieve the timing and power requirements of the design. For the synthesis step, SDC timing constraints set the performance goals whereas non-timing FDC constraints guide the synthesis tool for optimization. For the Place-and-Route step, SDC timing constraints guide the tool to achieve the timing requirements whereas Physical Design Constraints (PDC) guide the tool for optimized placement and routing (Floorplanning). For Static Timing Analysis, SDC timing constraints set the timing requirements and design-specific timing exceptions for static timing analysis.

Libero SoC provides the Constraint Manager as the cockpit to manage your design constraint needs. This is a single centralized graphical interface for you to create, import, link, check, delete, and edit design constraints and associate the constraint files to design tools in the Libero SoC environment. The Constraint Manager allows you to manage constraints for SynplifyPro synthesis, Libero SoC Place-and-Route and the SmartTime Timing Analysis throughout the design process.

### 5.1 Invocation of Constraint Manager from the Design Flow Window

After project creation, double-click **Manage Constraints** in the Design Flow window to open the Constraint Manager.

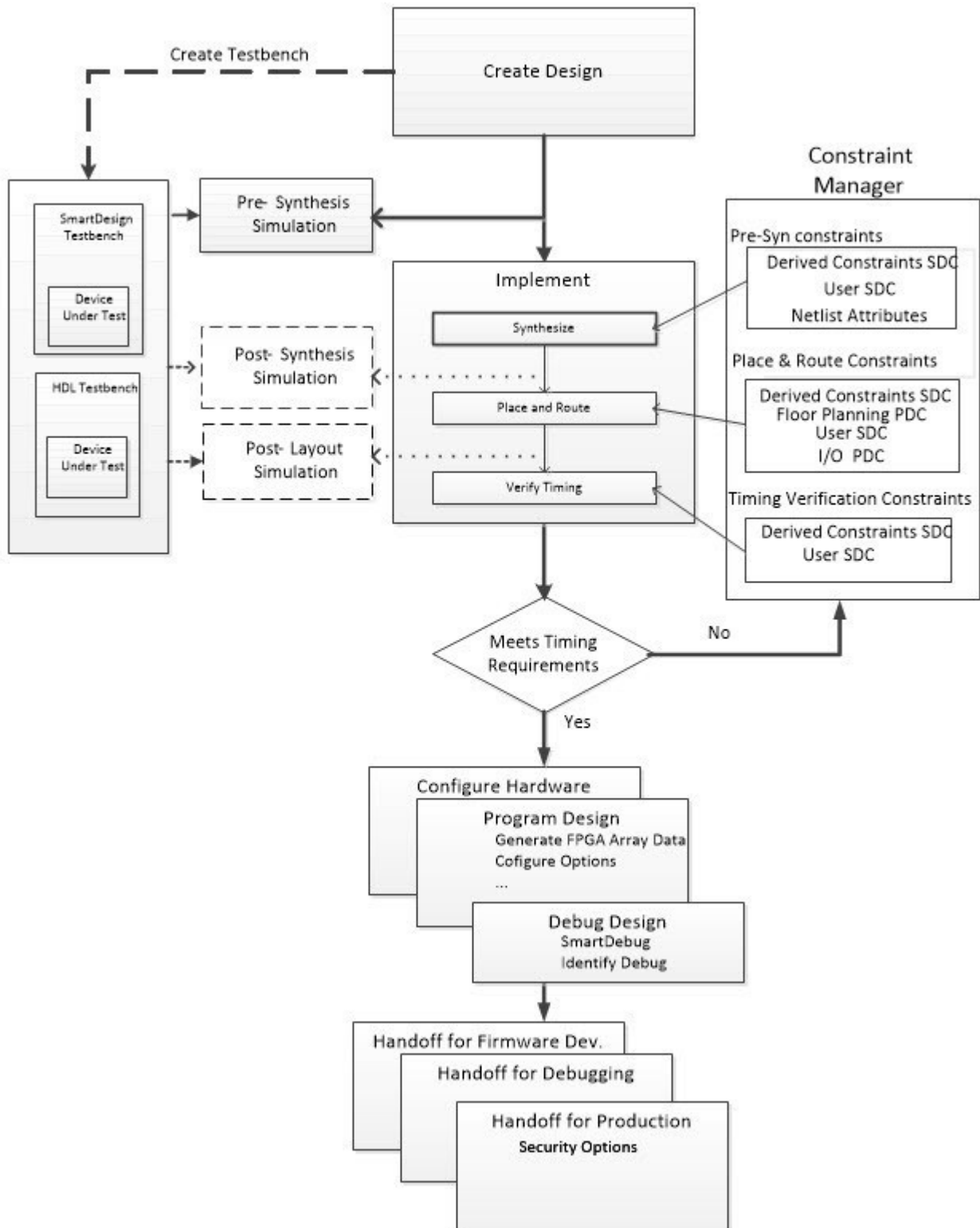
**Figure 5-1. Constraint Manager**



### 5.2 Libero SoC Design Flow

The Constraint Manager is Libero SoC's single centralized Graphical User Interface for managing constraints files in the design flow.

**Figure 5-2. Constraint Manager in Libero SoC Design Flow**



## 5.3 Introduction to Constraint Manager

The Constraint Manager manages these synthesis constraints and passes them to SynplifyPro:

- Synplify Netlist Constraint File (\*.fdc)
- Compile Netlist Constraint File (\*.ndc)
- SDC Timing Constraints (\*.sdc)
- Derived Timing Constraints (\*.sdc)

### 5.3.1 Synplify Netlist Constraints (\*.fdc)

These are non-timing constraints that help SynplifyPro optimize the netlist. From the Constraint Manager Netlist Attribute tab import (**Netlist Attributes > Import**) an existing FDC file or create a new FDC file in the Text Editor (**Netlist Attributes > New > Create New Synplify Netlist Constraint**). After the FDC file is created or imported, click the checkbox under synthesis to associate the FDC file with Synthesis.

### 5.3.2 Compile Netlist Constraints (\*.ndc)

These are non-timing constraints that help Libero SoC optimize the netlist by combining I/Os with registers. I/Os are combined with a register to achieve better clock-to-out or input-to-clock timing. From the Constraint Manager Netlist Attribute tab import (**Netlist Attributes > Import**) an existing NDC file or create a new NDC file in the Text Editor (**Netlist Attributes > New > Create New Compile Netlist Constraint**). After the NDC file is created or imported, click the checkbox under synthesis to associate the NDC file with Synthesis.

### 5.3.3 SDC Timing Constraints (\*.sdc)

These are timing constraints to guide SynplifyPro to optimize the netlist to meet the timing requirements of the design. From the Constraint Manager Timing tab, import (**Timing > Import**) or create in the Text Editor (**Timing > New**) a new SDC file. After the SDC file is created or imported, click the checkbox under synthesis to associate the SDC file with Synthesis.

After the synthesis step, you may click **Edit with Constraint Editor > Edit Synthesis Constraints** to edit existing constraints or add new SDC constraints.

### 5.3.4 Derived Timing Constraints (\*.sdc)

These are timing constraints Libero SoC generates for IP cores used in your design. These IP cores, available in the Catalog, are family/device-dependent. Once they are configured, generated, and instantiated in the design, the Constraint Manager can generate SDC timing constraints based on the configuration of the IP core and the component SDC. From the Constraint Manager Timing tab, click Derive Constraints to generate the Derived Timing Constraints (\*.sdc). Click the \*derived\_constraints.sdc file to associate it with synthesis.

### 5.3.5 Place and Route Constraints

The Constraint Manager manages these constraints for the Place-and-Route step:

- I/O PDC Constraints (\*io.pdc)
- Floorplanning PDC Constraints (\*fp.pdc)
- Timing SDC constraint file (\*.sdc)

### 5.3.6 I/O PDC Constraints

These are I/O Physical Design Constraints in an \*io.pdc file. From the Constraint Manager I/O Attribute tab, you may import (**I/O Attributes > Import**) or create in the Text Editor (**I/O Attributes > New**) an \*io.pdc file.

Click the checkbox under Place and Route to associate the file with Place and Route.

### 5.3.7 Floorplanning PDC Constraints

These are floorplanning Physical Design Constraints in a \*fp.pdc file. From the Constraint Manager Floor Planner tab, you may import (**Floor Planner > Import**) or create in the Text Editor (**Floor Planner > New**) a \*fp.pdc file. Click the checkbox under Place and Route to associate the file with Place and Route.



### 5.3.8 Timing SDC Constraint File (\*.sdc)

These are timing constraint SDC files for Timing-driven Place and Route. From the Constraint Manager Timing tab, you can import (**Timing > Import**) or create in the Text Editor (**Timing > New**) a timing SDC file. Click the check box under **Place and Route** to associate the SDC file with Place and Route. This file is passed to Timing-driven Place and Route (**Place and Route > Configure Options > Timing Driven**).

### 5.3.9 Timing Verifications Constraints

The Constraint Manager manages the SDC timing constraints for Libero SoC's SmartTime, which is a Timing Verifications/Static Timing analysis tool. SDC timing constraints provide the timing requirements (e.g. create\_clock and create\_generated\_clock) and design-specific timing exceptions (e.g. set\_false\_path and set\_multicycle\_path) for Timing Analysis.

From the Constraint Manager Timing tab, you may import (**Timing > Import**) or create in the Text Editor (**Timing New**) a SDC timing file. Click the checkbox under Timing Verifications to associate the SDC timing constraints file with Timing Verifications.

Note: You may have the same set of SDC Timing Constraints for Synthesis, Place and Route and Timing Verifications to start with in the first iteration of the design process. However, very often and particularly when the design is not meeting timing requirements you may find it useful in subsequent iterations to have different sets of Timing SDC files associated with different tools. Take for example; you may want to change/modify the set of SDC timing constraints for Synthesis or Place and Route to guide the tool to focus on a few critical paths. The set of SDC timing constraints associated with Timing Verifications can remain unchanged.

The Constraint Manager lets you associate/dis-associate the constraint files with the different tools with a mouse click.

### 5.3.10 Constraint Manager Components

The Constraint Manager has four tabs, each corresponding to a constraint type that Libero SoC supports:

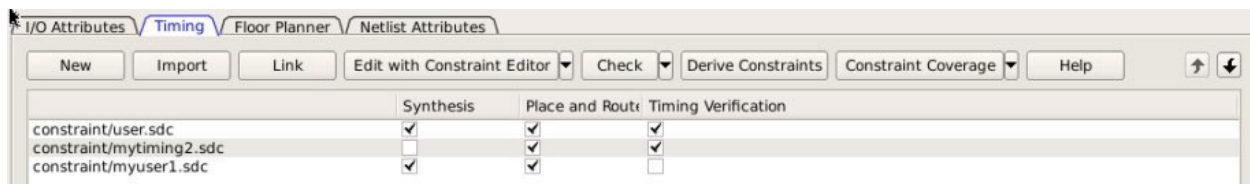
- I/O Attributes
- Timing
- Floor Planner
- Netlist Attribute


Clicking the tabs displays the constraint file of that type managed in the Libero SoC project.


### 5.3.11 Constraint File and Tool Association

Each constraint file can be associated or disassociated with a design tool by checking and unchecking the checkbox corresponding to the tool and the constraint file. When associated with a tool, the constraint file is passed to the tool for processing.

**Figure 5-3. Constraint File and Tool Association**



Libero SoC's Design Flow window displays the state the tool is in. A green check mark  indicates

successful completion. A warning icon  indicates invalidation of the state because the input files for the tool have changed since the last successful run. Association of a new constraint file with a tool or dis-association of an existing constraint file with a tool invalidates the state of the tool with which the constraint file is associated.

All Constraint files except Netlist Attributes can be opened, read, and edited by the following Interactive Tools invoked from the Constraint Manager directly:

- I/O Editor
- Chip Planner
- Constraint Editor

**Table 5-1. Constraint Types, File Extensions, Locations, and Tools**

Constraint Type	Constraint File Extension	Location Inside Project	Associated with Design Tool	Interactive Tool (for Editing)
I/O Attributes	PDC (*.pdc)	<proj>\constraints\io\*.pdc	Place and Route	I/O Editor
Floorplanning	PDC (*.pdc)	<proj>\constraints\fp\*.pdc	Place and Route	Chip Planner
Timing	SDC (*.sdc)	<proj>\constraints\*.sdc	Synthesis, Place and Route, Timing Verification	Constraint Editor
Netlist Attributes	FDC (*.fdc)	<proj>\constraints\*.fdc	Synthesis	n/a
	NDC (*.ndc)	<proj>\constraints\*.ndc	Synthesis	n/a

### 5.3.12 Derive Constraints in Timing Tab

The Constraint Manager can generate timing constraints for IP cores used in your design. These IP cores, available in the Catalog, are family- and device-dependent. After they are configured, generated, and instantiated in your design, the Constraint Manager can generate SDC timing constraints based on the configuration of the IP core and the component SDC. A typical example of an IP core for which the Constraint Manager can generate SDC timing constraints is the IP core for Clock Conditioning Circuitry (CCC).

### 5.3.13 Create New Constraints

From the Constraint Manager, create new constraints in one of two ways:

- Use the Text Editor
- Use Libero SoC's Interactive Tools

To create new constraints from the Constraint Manager using the Text Editor:

1. Select the Tab that corresponds to the type of constraint you want to create.
2. Click **New**.
3. When prompted, enter a file name to store the new constraint.
4. Enter the constraint in the Text Editor.
5. Click **OK**.

The Constraint file is saved and visible in the Constraint Manager in the tab you select:

- I/O Attributes constraint file (<proj>\io\\*.pdc) in the I/O Attributes tab
  - Floorplanning constraints (<proj>\fp\\*.pdc) in the Floor Planner tab
  - Timing constraints (<proj>\constraints\\*.sdc) in the Timing tab
6. (Optional) Double-click the constraint file in the Constraint Manager to open and add more constraints to the file.

**Note:** Netlist Attribute constraints cannot be created by an Interactive Tool. Netlist Attribute files can only be created with a Text Editor.

**Note:** Except for timing constraints for Synthesis, the design needs to be in the post-synthesis state to enable editing/creation of new constraints by the Interactive Tool.

**Note:** The \*.pdc or \*.sdc file the Constraint Manager creates is marked [Target]. This denotes that it is the target file. A target file receives and stores new constraints from the Interactive Tool. When you have multiple constraint files of the same type, you may select any one of them as target. When there are multiple constraint files but none of them is set as target, or there are zero constraint files, Libero SoC creates a new file and sets it as target to receive and store the new constraints created by the Interactive Tools.

To create new constraints from the Constraint Manager using Interactive Tools:

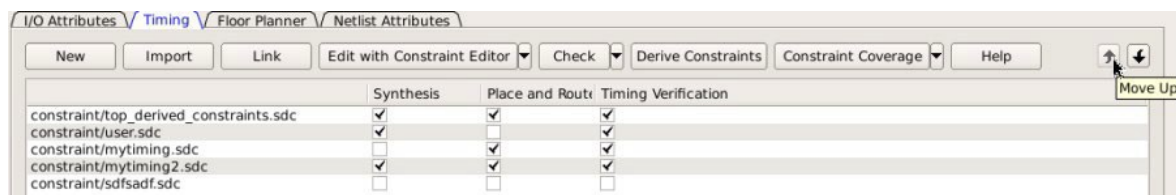
1. Select the Tab that corresponds to the type of constraint you want to create.
2. Click Edit to open the Interactive Tools. The Interactive Tool that Libero SoC opens varies with the constraint type:
  - I/O Editor to edit/create I/O Attribute Constraints. See [I/O Editor User Guide](#) for details.
  - Chip Planner to edit/create Floorplanning constraints. See [Chip Planner User Guide](#) for details.
  - Constraint Editor to edit/create Timing Constraints. See [Timing Constraints Editor User Guide](#) for details.
3. Create the Constraints in the Interactive Tool. Click **Commit and Save**.
4. Check that Libero SoC creates these files to store the new constraints:
  - Constraints\io\user.pdc file when I/O constraints are added and saved in I/O Editor.
  - Constraints\fp\user.pdc file when floorplanning constraints are added and saved in Chip Planner.
  - Constraints\user.sdc file when Timing Constraints are added and saved in Constraint Editor.

### 5.3.14 Constraint File Order

When there are multiple constraint files of the same type associated with the same tool, use the Up and Down arrow to arrange the order in which the constraint files pass to the associated tool. Constraint file order is important when there is a dependency between constraints files. When a floorplanning PDC file assigns a macro to a region, the region must first be created and defined. If the PDC command for region creation and macro assignment are in different PDC files, the order of the two PDC files is critical.

1. To move a constraint file up, select the file and click the Up arrow.
2. To move a constraint file down, select the file and click the Down arrow.

**Figure 5-4. Move constraint file Up or Down**



**Note:** Changing the order of the constraint files associated with the same tool invalidates the state of that tool.

## 5.4 Import a Constraint File

Use the Constraint Manager to import a constraint file into the Libero SoC project. When a constraint file is imported, a local copy of the constraint file is created in the Libero Project.

To import a constraint file:

1. Click the Tab corresponding to the type of constraint file you want to import.
2. Click **Import**.
3. Navigate to the location of the constraint file.
4. Select the constraint file and click **Open**. A copy of the file is created and appears in Constraint Manager in the tab you have selected.

### Link a Constraint File

Use the Constraint Manager to link a constraint file into the Libero SoC project. When a constraint file is linked, a file link rather than a copy is created from the Libero project to a constraint file physically located and maintained outside the Libero SoC project.

To link a constraint file:

1. Click the Tab corresponding to the type of constraint file you want to link.
2. Click **Link**.
3. Navigate to the location of the constraint file you want to link to.
4. Select the constraint file and click **Open**. A link of the file is created and appears in Constraint Manager under the tab you have selected. The full path location of the file (outside the Libero SoC project) is displayed.

### 5.4.1 Check a Constraint File

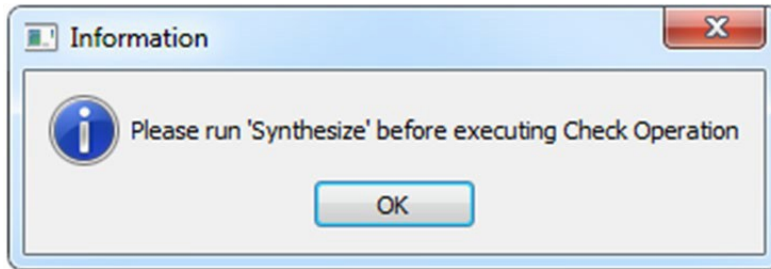
Use the Constraint Manager to check a constraint file.

To check a constraint file:

1. Select the tab for the constraint type to check.
2. Click **Check**.

**Note:** I/O constraints, Floorplanning constraints, Timing constraints, and Netlist Attributes can be checked only when the design is in the proper state. A pop-up message appears when the check is made and the design state is not proper for checking.

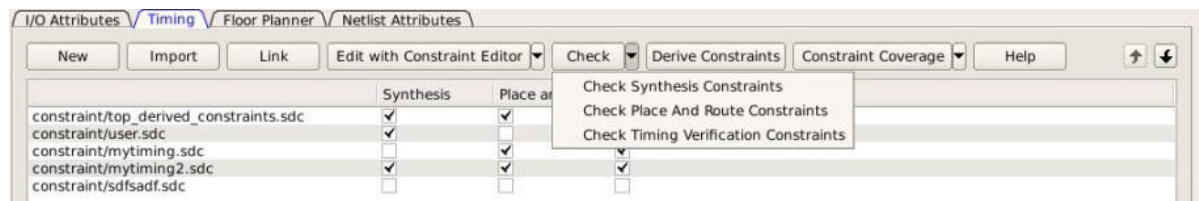
All constraint files associated with the tool are checked. Files not associated with a tool are not checked.



For Timing Constraints, select from the one of the following from the **Check** drop-down menu:

- Check Synthesis Constraints
- Check Place and Route Constraints
- Check Timing Verification Constraints

**Figure 5-5. Check Constraints**



**Check Synthesis Constraints** checks only the constraint files associated with the Synthesis.

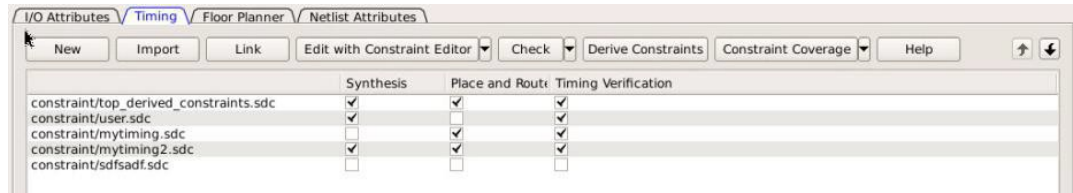
**Check Place and Route Constraints** checks only the constraint files associated with Place and Route.

**Check Timing Verification Constraints** checks only the Constraint Files associated with Timing Verification. For the constraint files and tool association shown in the *SDC file and Tool Association* figure below:

- **Check Synthesis Constraints** checks the following files:
  - top\_derived\_constraints.sdc
  - user.sdc
  - mytiming2.sdc
- **Check Place and Route Constraints** checks the following files:
  - top\_derived\_constraints.sdc
  - mytiming.sdc
  - mytiming2.sdc
- **Check Timing Verification Constraints** checks the following files:
  - top\_derived\_constraints.sdc
  - user.sdc
  - mytiming.sdc
  - mytiming2.sdc

**Note:** sdfasdf.sdc Constraint File is not checked because it is not associated with any tool.

**Figure 5-6. Timing Constraints SDC File and Tool**



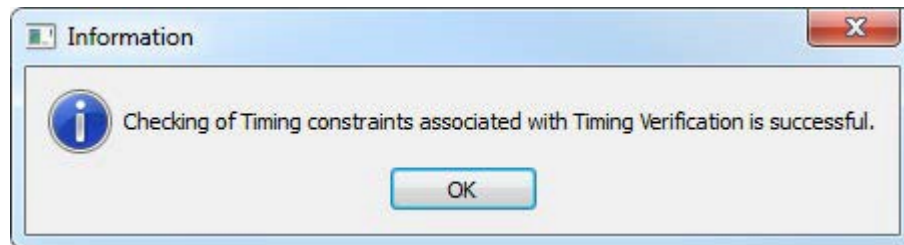
Association when a constraint file is checked, the Constraint Manager:

- Checks the SDC or PDC syntax.
- Compares the design objects (pins, cells, nets, ports) in the constraint file versus the design objects in the netlist (RTL or post-layout ADL netlist). Any discrepancy (e.g. constraints on a design object which does not exist in the netlist) are flagged as errors and reported in the \*.log file or message window.

#### 5.4.2 Check Result

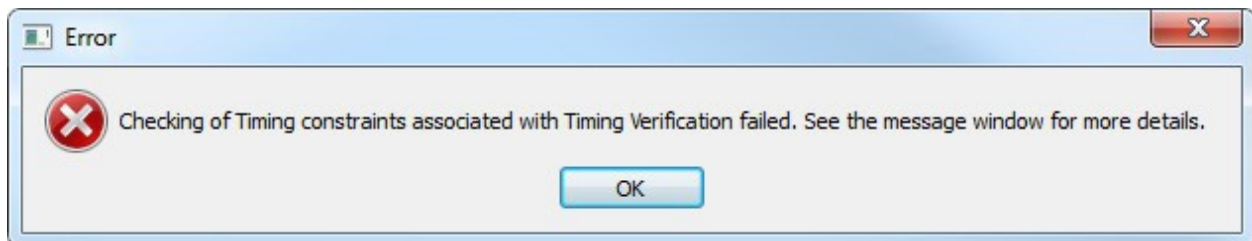
If the check is successful, the following message appears.

**Figure 5-7. Check Successful Message**



If the check fails, the following error message appears.

**Figure 5-8. Check Fails Message**



Constraint Type	Check for Tools	Required Design State Before Checks	Check Result Details
I/O Constraints	Place and Route	Post-Synthesis	Libero Message Window
Floorplanning Constraints	Place and Route	Post-Synthesis	Libero Message Window
Timing Constraints	Synthesis	Pre-Synthesis	synthesis_sdc_check.log
	Place and Route	Post-Synthesis	placer_sdc_check.log
	Timing Verifications	Post-Synthesis	timing_sdc_check.log
Netlist Attributes (*.fdc)	Synthesis	Pre-Synthesis	*cck.srr file
Netlist Attributes (*.ndc)	Synthesis	Pre-Synthesis	Libero Log Window

### 5.4.3 Edit a Constraint File

The Edit button in the Constraint Manager allows you to:

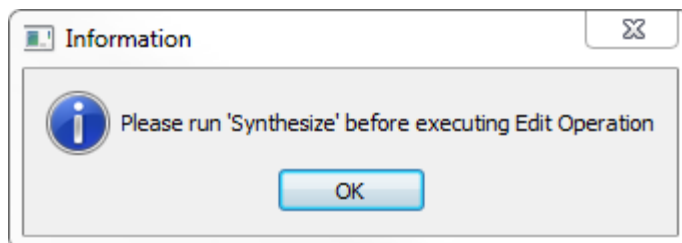
- Create new constraint files. See [5.3.13 Create New Constraints](#) for details.
- Edit existing constraint files.

#### To edit a constraint file

**Note:** Netlist Attributes cannot be edited by an Interactive Tool. Use the Text Editor to edit the Netlist Attribute constraint (\*.fdc and \*.ndc) files.

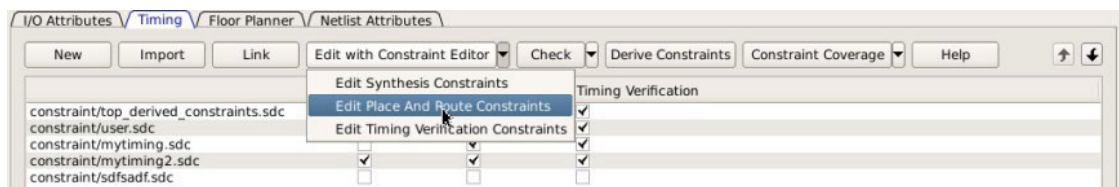
1. Select the tab for the constraint type to edit. An Interactive Tool is opened to make the edits.
  2. Click Edit.
    - All constraint files associated with the tool are edited. Files not associated with the tool are not edited.
    - When a constraint file is edited, the constraints in the file are read into the Interactive Tool.
    - Different Interactive Tools are used to edit different constraints/different files:
      - I/O Editor to edit I/O Attributes (<proj>\io\\*.pdc). For details, refer to the [I/O Editor User Guide](#)
      - Chip Planner to edit Floorplanning Constraints (<proj>\fp\\*.pdc). For details, refer to the [Chip Planner User Guide](#) (Chip Planner > Help > Reference Manuals)
      - Constraint Editor to edit Timing Constraints (constraints\\*.sdc). For details, refer to the [Timing Constraints Editor User Guide](#) (Help > Constraints Editor User's Guide)
- Note:** I/O constraints, Floorplanning constraints, Timing constraints can be edited only when the design is in the proper state. A message pops up if the file is edited when the design state is not proper for edits. If, for example, you open the Constraints Editor (Constraint Manager > Edit) to edit timing constraints when the design state is not post-synthesis, a pop-up message appears.

**Figure 5-9. Pop-up Message**



3. For Timing Constraints, click one of the following to edit from the Edit with Constraint Editor drop-down menu.
  - Edit Synthesis Constraints
  - Edit Place and Route Constraints
  - Edit Timing Verification Constraints

**Figure 5-10. Edit Drop-down Menu**

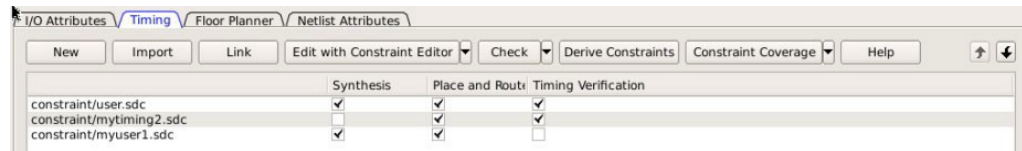


For the constraint files and tool association shown in the *Timing Constraint File and Tool Association* below:

- Edit Synthesis Constraints reads the following files into the Constraint Editor:
  - user.sdc
  - myuser1.sdc
- Edit Place and Route Constraints reads the following files into the Constraint Editor:
  - user.sdc
  - mytiming2.sdc
  - myuser1.sdc

- Edit Timing Verification Constraints reads the following files into the Constraint Editor:
  - user.sdc
  - mytiming2.sdc

**Figure 5-11. Timing Constraint File and Tool Association**



4. Edit the constraint in the Interactive Tool, save and exit.
5. The edited constraint is written back to the original constraint file when the tool exits.

Refer to the [Timing Constraints Editor User Guide](#) (Help > Constraints Editor User's Guide) for details on how to enter/modify timing constraints.

**Note:** When a constraint file is edited inside an Interactive Tool, the Constraint Manager is disabled until the Interactive Tool is closed.

**Note:** Making changes to a constraint file invalidates the state of the tool with which the constraint file is associated. For instance, if Place and Route has successfully completed with user.sdc as the associated constraint file, then making changes to user.sdc invalidates Place and Route. The green checkmark (denoting successful completion) next to Place and Route turns into a warning icon when the tool is invalidated.

## 5.5 Constraint Types

Libero SoC manages four types of constraints:

- **I/O Attributes Constraints:** Used to constrain placed I/Os in the design. Examples include setting I/O standards, I/O banks, and assignment to Package Pins, output drive, and so on. These constraints are used by Place and Route.
- **Timing Constraints:** Specific to the design set to meet the timing requirements of the design, such as clock constraints, timing exception constraints, and disabling certain timing arcs. These constraints are passed to Synthesis, Place and Route, and Timing Verification.
- **Floor Planner Constraints:** Non-timing floorplanning constraints created by the user or Chip Planner and passed to Place and Route to improve Quality of Routing.
- **Netlist Attributes:** Microchip-specific attributes that direct the Synthesis tool to synthesize/optimize the design, leveraging the architectural features of the Microchip devices. Examples include setting the fanout limits and specifying the implementation of a RAM. These constraints are passed to the Synthesis tool only.

The following table summarizes the features and specifics of each constraint type.

Constraint Type	File Location	File Ext.	User Actions	Constraints Edited By	Constraints Used By	Changes Invalidate Design State?
I/O Attributes	<proj>/constraints/io folder	*.pdc	Create New, Import, Link, Edit, Check	I/O Editor Or user editing the *.pdc file in Text Editor	Place and Route	YES



.....continued						
Constraint Type	File Location	File Ext.	User Actions	Constraints Edited By	Constraints Used By	Changes Invalidate Design State?
Timing Constraints	<proj>/constraints folder	*.sdc	Create New, Import, Link, Edit, Check	Constraint Editor Or user editing the *.sdc file in Text Editor	Synplify Place and Route Verify Timing (SmartTime)	YES
Floor Planner Constraints	<proj>/constraints/fp folder	*.pdc	Create New, Import, Link, Edit, Check	Chip Planner Or user Editing the *.pdc file in Text Editor	Place and Route	YES
Netlist Attributes	<proj>/constraints folder	*.fdc	Create New, Import, Link, Check	User to Open in Text Editor to Edit	Synplify	YES
Netlist Attributes	<proj>/constraints folder	*.ndc	Import, Link, Check	User to Open in Text Editor to Edit	Synplify	YES

## 5.6 Constraint Manager – I/O Attributes Tab

The I/O Attributes tab allows you to manage I/O attributes/constraints for your design's Inputs, Outputs, and Inouts. All I/O constraint files (PDC) have the \*.pdc file extension and are placed in the <Project\_location>/constraint/io folder. Available actions are:

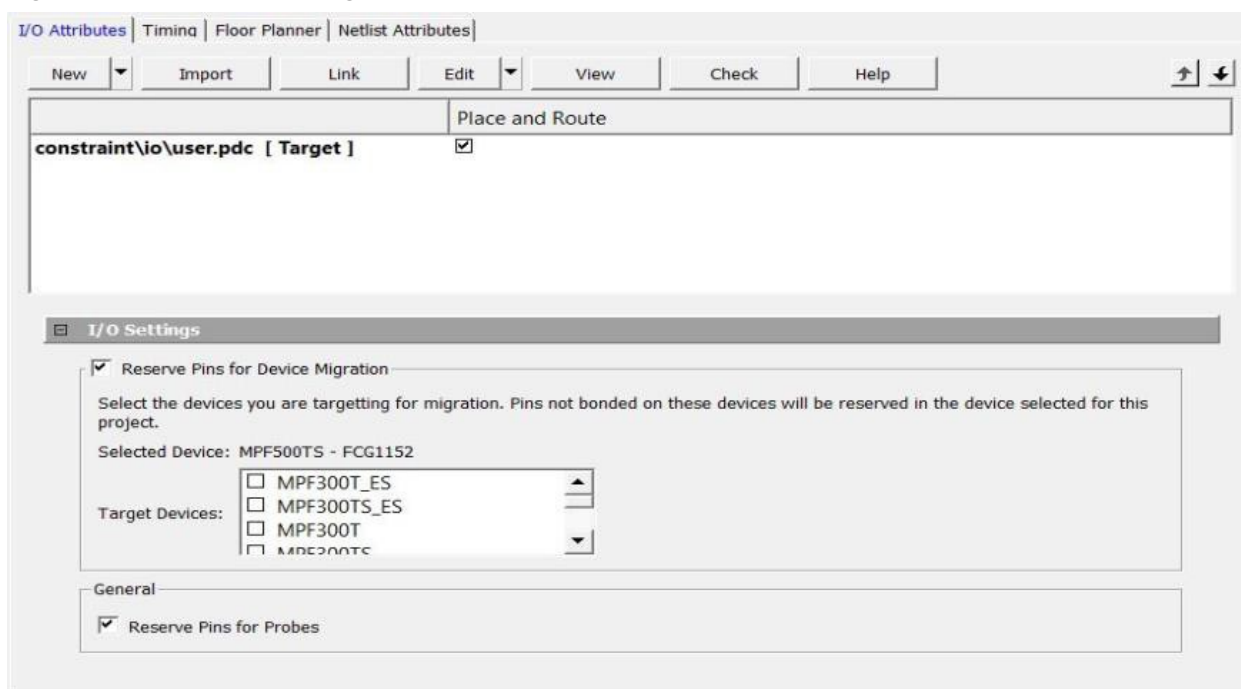
- **New:** Creates a new I/O PDC file and saves it into the <Project\_location>\constraint\io folder. There are two options:
  - Create New I/O Constraint
  - Create New I/O Constraint From Root Module -- This will pre-populate the PDC file with information from the Root Module
  - Having selected the create method:
    - When prompted, enter the name of the constraint file.
    - The file is initially opened in the text editor for user entry.
- **Import:** Imports an existing I/O PDC file into the Libero SoC project. The I/O PDC file is copied into the <Project\_location>\constraint\io folder.
- **Link:** Creates a link in the project's constraint folder to an existing I/O PDC file (located and maintained outside of the Libero SoC project).
- **Edit:** Opens the I/O Editor tool to modify the I/O PDC file(s) associated with the Place and Route tool.
- **View:** Opens the I/O Editor tool to view the I/O PDC file(s) associated with the Place and Route tool. You cannot save/commit any changes made to the constraints file. However, you can export the PDC file(s) using the I/O Editor.
- **Check:** Checks the legality of the PDC file(s) associated with the Place and Route tool against the gate level netlist.

When the I/O Editor tool is invoked or the constraint check is performed, all files associated with the Place and Route tool are being passed for processing.



When you save your edits in the I/O Editor tool, the I/O PDC files affected by the change will be updated to reflect the change you have made in the I/O Editor tool. New I/O constraints you add in the I/O Editor tool are written to the *Target* file (if a target file has been set) or written to a new PDC file (if no file is set as target) and stored in the <project>\constraint\io folder.

**Figure 5-12. Constraint Manager – I/O Attributes Tab**



Right-click the I/O PDC files to access the available actions:

- **Set/UnSet as Target:** Sets or clears the selected file as the target to store new constraints created in the I/O Editor tool. Newly created constraints only go into the target constraint file. Only one file can be set as target. This option is not available for linked files.
- **Open in Text Editor:** Opens the selected constraint file in the Libero Text Editor.
- **Clone:** Copies the file to a file with a different name. The original file name and its content remain intact. This option is not available for linked files.
- **Rename:** Renames the file to a different name. This option is not available for linked files.
- **Copy File Path:** Copies the file path to the clipboard.
- **Delete:** Deletes the file from the project and from the disk. This option is not available for linked files.
- **Unlink:** Removes the linked file from the project. The original file is untouched. This option is only available for linked files.
- **Unlink: Copy file locally:** Removes the link and copies the file into the <Project\_location>\constraint\io folder. This option is only available for linked files.

### 5.6.1 File and Tool Association

Each I/O constraint file can be associated or disassociated with the Place and Route tool. Click the checkbox under **Place and Route** to associate/disassociate the file from the tool.

### 5.6.2 I/O Settings

**Reserve Pins for Device Migration** – This option allows you to reserve pins in the currently selected device that are not bonded in a device or list of devices you may later decide to migrate your design to. Select the target device(s) you may migrate to later to ensure that there will be no device/package incompatibility if you migrate your design to that device.

**Reserve Pins for Probes** – Check this box if you plan to use live probes when debugging your design with SmartDebug.

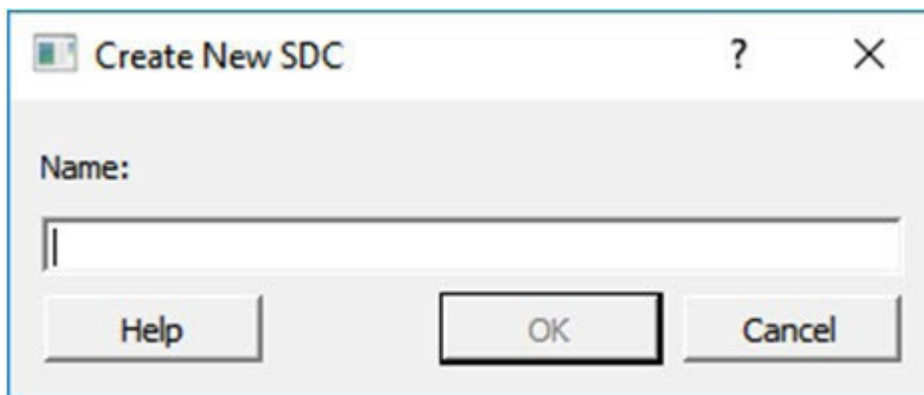
## 5.7 Constraint Manager – Timing Tab

The Timing tab allows you to manage timing constraints throughout the design process. Timing constraints files (SDC) have the \*.sdc file extension and are placed in the <Project\_location>\constraint folder.

Available actions are:

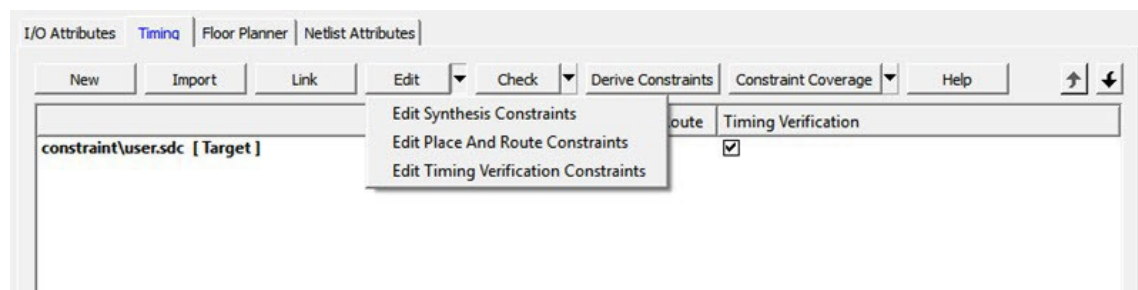
- **New:** Creates a new timing SDC file and saves it into the <Project\_location>\constraint folder.
  - When prompted, enter the name of the constraint file.
  - The file is initially opened in the text editor for user entry.

**Figure 5-13. Create New SDC Dialog Box**



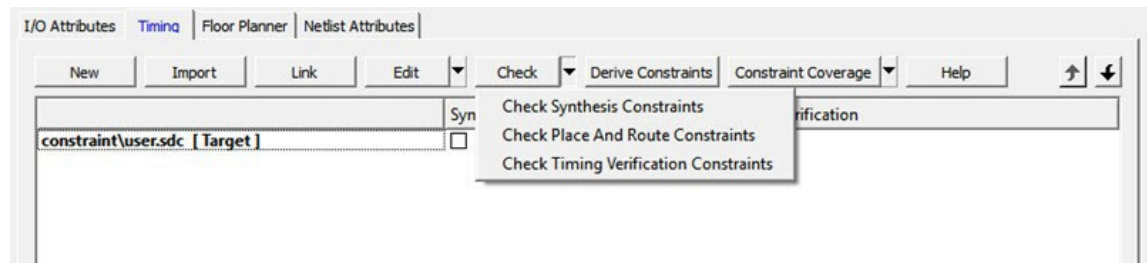
- **Import:** Imports an existing timing SDC file into the Libero SoC project. The timing SDC file is copied into the <Project\_location>\constraint folder.
- **Link:** Creates a link in the project's constraint folder to an existing timing SDC file (located and maintained outside of the Libero SoC project).
- **Edit:** Opens the Timing Constraints Editor (see [Timing Constraints Editor User Guide](#) for details) to modify the SDC file(s) associated with one of the three tools:
  - **Synthesis:** When selected, the timing SDC file(s) associated with the Synthesis tool is loaded in the constraints editor for editing.
  - **Place and Route:** When selected, the timing SDC file(s) associated with the Place and Route tool is loaded in the constraints editor for editing.
  - **Timing Verification:** When selected, the timing SDC file(s) associated with the Timing Verification tool is loaded in the constraints editor for editing.

**Figure 5-14. Timing Constraints Edit Options in Constraint Manager**



- **Check:** Check the legality of the SDC file(s) associated with one of the three tools described below:
  - **Synthesis:** The check is performed against the pre-synthesis HDL design.
  - **Place and Route:** The check is performed against the post-synthesis gate level netlist.
  - **Timing Verification:** The check is performed against the post-synthesis gate level netlist.

**Figure 5-15. Timing Constraints Check Options in Constraint Manager**

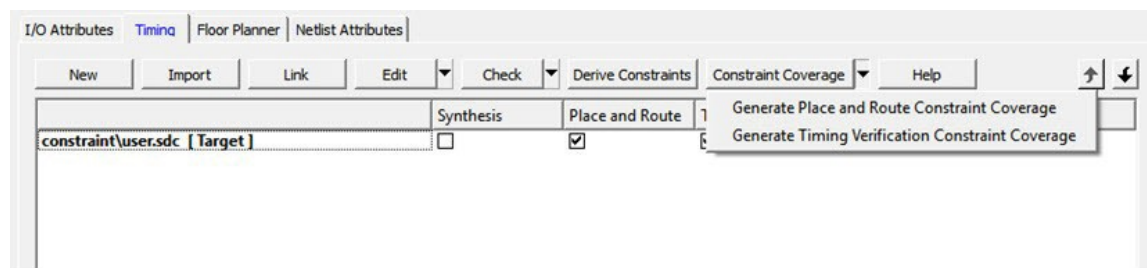


- **Derive Constraints:** When clicked, Libero generates a timing SDC file based on user configuration of IP core, components, and component SDC. It generates the create\_clock and create\_generated\_clock SDC timing constraints. This file is named <top\_level\_> derived\_constraints.sdc. The component SDC and the generated <root>\_derived\_constraint.sdc files are dependent on the IP cores and vary with the device family.

```
create -name {REF_CLK_PAD_0} -period 5 [ get_ports { REF_CLK_PAD_0 } ]
create_generated_clock -name {PF_TX_PLL_0/txpll_isnt_0/DIV_CLK} -divide by 2 - source
[ get_pins { PF_TX_PLL_0/txpll_isnt_0/REF_CLK_P } ] [ get_pins { PF_TX_PLL_0/txpll_isnt_0/
DIV_CLK } ]
```

- **Constraint Coverage:** When clicked, a pull-down list displays. Select the Constraint Coverage Reports you want:
  - Generate Place and Route Constraint Coverage Report
  - Generate Timing Verification Constraint Coverage Report

**Figure 5-16. Constraint Coverage Options for Timing Constraints in Constraint Manager**



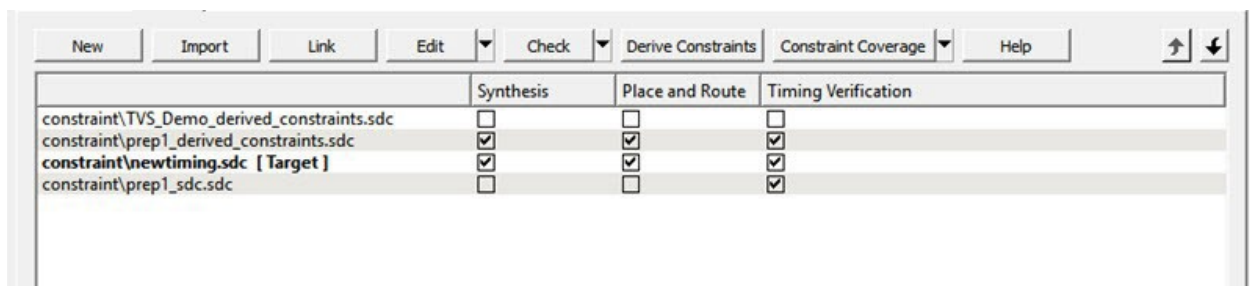
**Note:** Constraint Coverage Reports can be generated only after synthesis. A warning message appears if the design is not in the post-synthesis state when this button is clicked.

The generated report will be visible in the respective nodes of the report view (**Design > Reports**).

When the SmartTime Constraint Editor tool is invoked or the constraint check is performed, all the files associated with the targeted tool – Synthesis, Place and Route, Timing Verification – are being passed for processing.

When you save your edits in the SmartTime Constraint Editor tool, the timing SDC files affected by the change are updated to reflect the changes you have made in the SmartTime Constraints Editor tool. New timing constraints you add in the SmartTime Constraint Editor tool are written to the *Target* file (if a target file has been set) or written to a new SDC file (if no file is set as target) and stored in the <project>\constraint folder.

**Figure 5-17. Constraint Manager – Timing Tab**



Right-click the timing SDC files to access the available actions for each constraint file:

- **Set/Unset as Target:** Sets or clears the selected file as the target to store new constraints created in the SmartTime Constraint Editor tool. Newly created constraints only go into the target constraint file. Only one file can be set as target, and it must be a PDC or SDC file. This option is not available for the derived constraint SDC file. This option is not available for linked files.
- **Open in Text Editor:** Opens the selected constraint file in the Libero Text Editor.
- **Clone:** Copies the file to a file with a different name. The original file name and its content remain intact. This option is not available for linked files.
- **Rename:** Renames the file to a different name. This option is not available for linked files.
- **Copy File Path:** Copies the file path to the clipboard.
- **Delete:** Deletes the selected file from the project and from the disk. This option is not available for linked files.
- **Unlink:** Removes the linked file from the project. The original file is untouched. This option is only available for linked files.
- **Unlink: Copy file locally:** Removes the link and copies the file into the <Project\_location>\constraint folder. This option is only available for linked files.

### File and Tool Association

Each timing constraint file can be associated or disassociated with any one, two, or all three of the following tools:

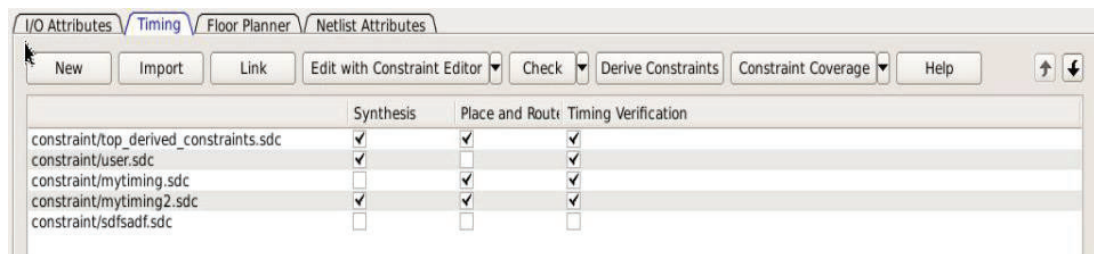
- Synthesis
- Place and route
- Timing Verification

Click the check box under **Synthesis**, **Place and Route**, or **Timing Verification** to associate/disassociate the file from the tool.

When a file is associated, Libero passes the file to the tool for processing.

### 5.7.1 Example

**Figure 5-18. File and Tool Association Example**



In the context of the graphic above, when Edit Synthesis Constraint is selected, user.sdc, top\_derived\_constraints.sdc, and mytiming2.sdc are read (because these three files are associated with Synthesis); mytiming.sdc and sdfsadf.sdc are not read (because they are not associated with Synthesis). When the SmartTime Constraint Editor opens for edit, the constraints from all the files except for sdfsadf.sdc are read and loaded into the Constraint Editor. Any changes you made and saved in the Constraint Editor are written back to the files.

**Note:** sdfsadf.sdc Constraint File is not checked because it is not associated with any tool.

## 5.8 Derived Constraints

Libero SoC is capable of generating SDC timing constraints for design components when the root of the design has been defined. Click **Derive Constraints** in the Constraint Manager's Timing tab to generate SDC timing constraints for your design's components.

The generated constraint file is named <root>\_derived.sdc and is created by instantiating component SDC files created by IP configurators (for example, CCC) and oscillators used in the design.

The <root>\_derived.sdc file is associated by default to the Synthesis, Place and Route and Timing Verification tool. You can change the file association in the Constraint Manager by checking or unchecking the checkbox under the tool.

To generate SDC timing constraints for IP cores:

1. Configure and generate the IP Core.
2. From the Constraint Manager's Timing tab, click Derive Constraints (**Constraint Manager > Timing > Derive Constraints**).  
The Constraint Manager generates the <root>\_derived\_constraints.sdc file and places it in the Timing Tab along with other user SDC constraint file.
3. When prompted for a **Yes** or **No** on whether you want the Constraint Manager to automatically associate the derived SDC file to Synthesis, Place and Route, and Timing Verification, click **Yes** to accept automatic association or **No** and then check or uncheck the appropriate checkbox for tool association.  
Microchip recommends the <root>\_derived\_constraints.sdc be always associated with all three tools: Synthesis, Place and Route, and Verify Timing. Before running SynplifyPro Synthesis, associate the <root>\_derived\_constraints.sdc file with Synthesis and Place and Route. This will ensure that the design objects (such as nets and cells) in the <root>\_derived\_constraints.sdc file are preserved during the synthesis step and the subsequent Place and Route step will not error out because of design object mismatches between the post-synthesis netlist and the <root>\_derived\_constraints.sdc file.

**Note:** Full hierarchical path names are used to identify design objects in the generated SDC file.

**Note:** The Derive Constraints button is available for HDL-based **and** SmartDesign-based design flows. It is not available for Netlist Designs (Project > Project Settings > Design Flow > Enable Synthesis [not checked]).

## 5.9 Constraint Manager – Floor Planner Tab

The Floor Planner tab allows you to manage floorplanning constraints. Floorplanning constraints files (PDC) have the \*.pdc file extension and are placed in the <Project\_location>\constraint\fp folder.

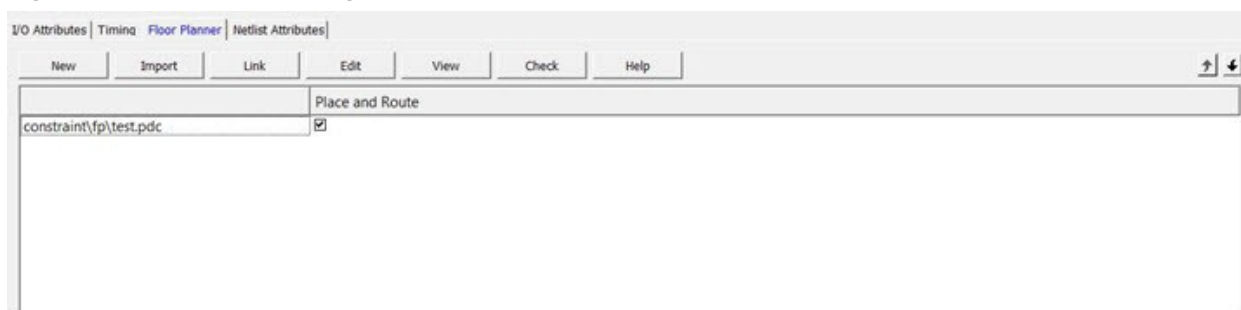
Available actions are:

- **New:** Creates a new floorplanning PDC file and saves it into the <Project\_location>\constraint\fp folder.
- **Import:** Imports an existing floorplanning PDC file into the Libero SoC project. The floorplanning PDC file is copied into the <Project\_location>\constraint\fp folder.
- **Link:** Creates a link in the project's constraint folder to an existing floorplanning PDC file (located and maintained outside of the Libero SoC project).
- **Edit:** Opens the [Chip Planner](#) tool to modify the floorplanning PDC file(s) associated with the Place and Route tool.
- **View:** Opens the Chip Planner tool to view the floorplanning PDC file(s) associated with the Place and Route tool. You cannot save/commit any changes made to the constraints file. However, you can export the PDC file(s) using Chip Planner.
- **Check:** Checks the legality of the PDC file(s) associated with the Place and Route tool against the gate level netlist.

When the Chip Planner tool is invoked or the constraint check is performed, all files associated with the Place and Route tool are passed for processing.

When you save your edits in the Chip Planner tool, the floorplanning PDC files affected by the change are updated to reflect the change you made in the Chip Planner tool. New floorplanning constraints that you add in the Chip Planner tool are written to the *Target* file (if a target file has been set) or written to a new PDC file (if no file is set as target) and stored in the <project>\constraint\fp folder.

**Figure 5-19. Constraint Manager – Floor Planner Tab**



Right-click the floorplanning PDC files to access the available actions:

- **Set/Unset as Target:** Sets or clears the selected file as the target to store new constraints created in the Chip Planner tool. Newly created constraints only go into the target constraint file. Only one file can be set as target. This option is not available for linked files.
- **Open in Text Editor:** Opens the selected constraint file in the Libero Text Editor.
- **Clone:** Copies the file to a file with a different name. The original file name and its content remain intact. This option is not available for linked files.
- **Rename:** Renames the file to a different name. This option is not available for linked files.
- **Copy File Path:** Copies the file path to the clipboard.
- **Delete:** Deletes the selected file from the project and from the disk. This option is not available for linked files.
- **Unlink:** Removes the linked file from the project. The original file is untouched. This option is only available for linked files.
- **Unlink: Copy file locally:** Removes the link and copies the file into the <Project\_location>\constraint\fp folder. This option is only available for linked files.

### 5.9.1 File and Tool Association

Each floorplanning constraint file can be associated or disassociated to the Place and Route tool. Click the checkbox under **Place and Route** to associate/disassociate the file from the tool.

When a file is associated, Libero passes the file to the tool for processing.

#### 5.9.1.1 See Also

[Chip Planner User Guide](#)

## 5.10 Constraint Manager – Netlist Attributes Tab

The Netlist Attributes tab allows you to manage netlist attribute constraints to optimize your design during the synthesis and/or compile process. Timing constraints should be entered using SDC files managed in the Timing tab. Netlist Attribute constraints files are placed in the <Project\_location>\constraint folder. Libero SoC manages two types of netlist attributes:

- FDC constraints are used to optimize the HDL design using Synopsys SynplifyPro synthesis engine and have the \*.fdc extension.
- NDC constraints are used to optimize the post-synthesis netlist with the Libero SoC compile engine and have the \*.ndc file extension.

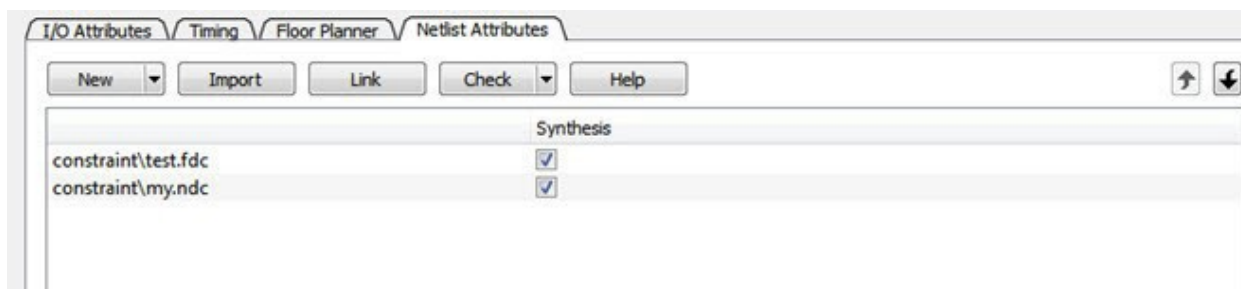
Available operations are:

- **New:** Creates a new FDC or NDC netlist attribute constraints file in the <Project\_location>\constraint folder.
- **Import:** Imports an existing FDC or NDC netlist attribute constraints file into the Libero SoC project. The FDC or NDC netlist attribute constraints file is copied into the <Project\_location>\constraint folder.
- **Link:** Creates a link in the project's constraint folder to an existing FDC or NDC netlist attribute constraints file (located and maintained outside of the Libero SoC project).
- **Check:** Checks the legality of the FDC and NDC file(s) associated with the Synthesis or Compile tools.



When the constraint check is performed, all files associated with the Synthesis or Compile tools are passed for processing.

**Figure 5-20. Constraint Manager – Netlist Attributes Tab**



Right-click the FDC or NDC files to access the available actions:

- **Open in Text Editor:** Opens the selected constraint file in the Libero SoC Text Editor.
- **Clone:** Copies the file to a file with a different name. The original file name and its content remain intact. This option is not available for linked files.
- **Rename:** Renames the file to a different name. This option is not available for linked files.
- **Copy File Path:** Copies the file path to the clipboard.
- **Delete:** Deletes the file from the project and from the disk. This option is not available for linked files.
- **Unlink:** Removes the linked file from the project. The original file is untouched. This option is only available for linked files.
- **Unlink: Copy file locally:** Removes the link and copies the file into the <Project\_location>\constraint folder. This option is only available for linked files.

### 5.10.1 File and Tool Association

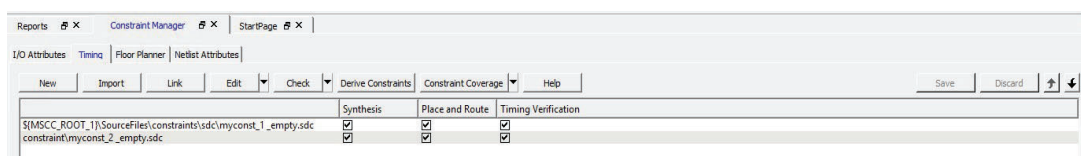
Each netlist attributes constraint file can be associated with or disassociated from the Synthesis tool.

Click the checkbox under **Synthesis** (Compile) to associate/disassociate the file from Synthesis (Compile). When a file is associated, Libero passes the file to Synthesis (Compile) for processing when Synthesis is run.

When Synthesis is ON (Project > Project Settings > Design Flow > Enable synthesis [checked]) for a project, the Design Flow Synthesis action runs both the synthesis engine and the post-synthesis compile engine.

When Synthesis is OFF (Project > Project Settings > Design Flow > Enable synthesis [not checked]) for a project, the Design Flow Synthesis action is replaced by the Compile action and runs the compile engine on the gate-level netlist (EDIF or Verilog) available in the project.

**Note:** Linked files in constraint manager are shown with relative path if relative option is set for linked files. When a constraint file is missing during environment variable change or a path in an environment variable changes, the constraint manager does show any broken links. Therefore, if a constraint file is missing, an error message appears in the log window if the user tries to access the file.



## 6. Implementing Designs

The following topics describe how to implement your design.

### 6.1 Synthesize

There are two ways to run synthesis using the synthesis tool:

- Run synthesis on your design with the default settings specified in the synthesis tool: Double-click **Synthesize**.
- Run the synthesis tool interactively: Right-click **Synthesize** and choose **Open Interactively**. If you open the tool interactively, you must complete synthesis from within the synthesis tool.

The default synthesis tool included with Libero SoC is SynplifyPro ME. If you want to use a different synthesis tool, change the settings in [Tool Profiles](#).

You can organize input synthesis source files using the [Organize Source Files](#) dialog box.

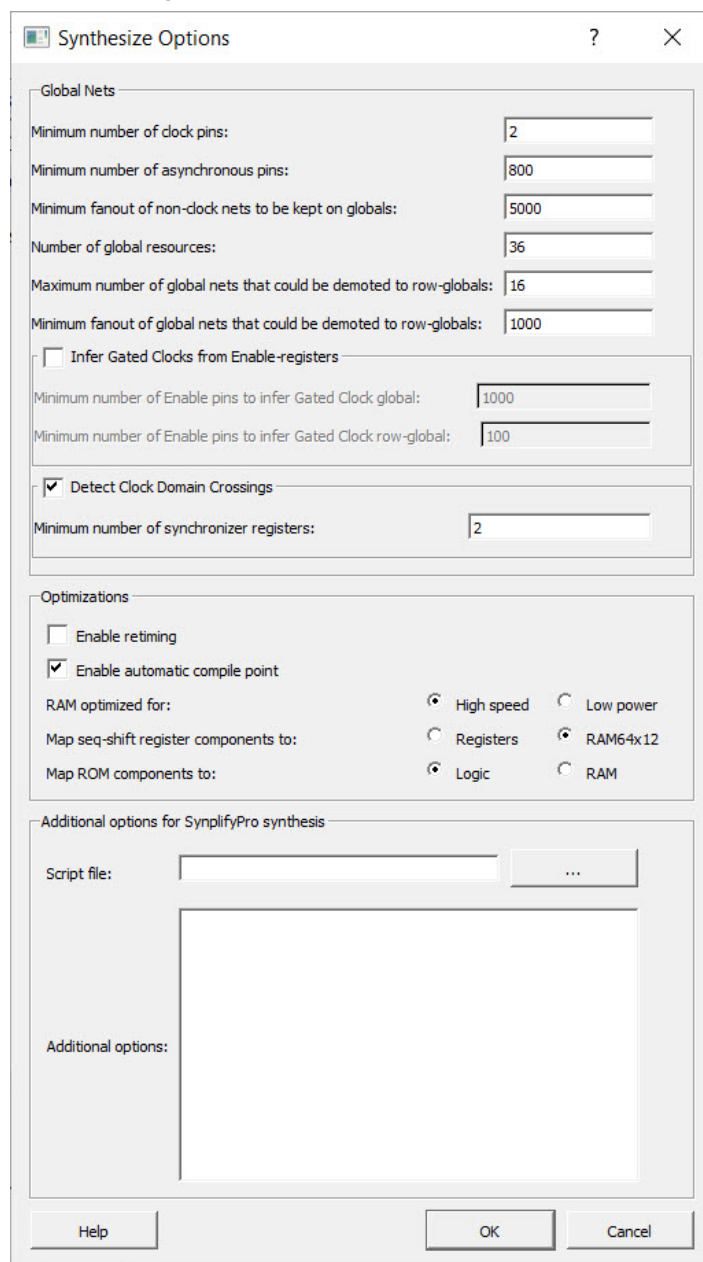
#### 6.1.1 Synthesize Options

You can set or change synthesis configuration options for your synthesis tool using the Synthesize Options dialog box.

To display this dialog box, expand **Implement Design** in the Design Flow window, right-click **Synthesize**, and choose **Configure Options**.



Figure 6-1. Synthesize Options Dialog Box



The dialog box is titled "Synthesize Options" and contains three main sections: Global Nets, Optimizations, and Additional options for SynplifyPro synthesis.

**Global Nets**

- Minimum number of clock pins: 2
- Minimum number of asynchronous pins: 800
- Minimum fanout of non-clock nets to be kept on globals: 5000
- Number of global resources: 36
- Maximum number of global nets that could be demoted to row-globals: 16
- Minimum fanout of global nets that could be demoted to row-globals: 1000
- ☐ Infer Gated Clocks from Enable-registers
  - Minimum number of Enable pins to infer Gated Clock global: 1000
  - Minimum number of Enable pins to infer Gated Clock row-global: 100
- ☒ Detect Clock Domain Crossings
  - Minimum number of synchronizer registers: 2

**Optimizations**

- ☐ Enable retiming
- ☒ Enable automatic compile point
- RAM optimized for:
  - ☒ High speed
  - ☐ Low power
- Map seq-shift register components to:
  - ☐ Registers
  - ☒ RAM64x12
- Map ROM components to:
  - ☒ Logic
  - ☐ RAM

**Additional options for SynplifyPro synthesis**

Script file: [Text Box] ...

Additional options: [Large Text Area]

Buttons: Help, OK, Cancel

### 6.1.2 Global Nets (Promotions and Demotions)

Use the following options to specify to the Synthesis tool the threshold value beyond which the Synthesis tool promotes the pins to globals:

**Note:** You cannot use these options to control hardwired connections to global resources, such as CCC hardwired connections to GB and I/O hardwired connections to GB.

Table 6-1. Global Nets Options

Option	Description
Minimum number of clock pins	Threshold value for Clock pin promotion. Default: 2

.....continued

Option	Description
Minimum number of asynchronous pins	Threshold value for Asynchronous pin promotion. Default: 800
Minimum fanout of non-clock nets to be kept on globals	Threshold value for data pin promotion to global resources. This value is the minimum fanout of non-clock (data) nets to be kept on globals (no demotion). Range: 1000 - 200000 Default: 5000
Number of global resources	Controls the number of Global resources you want to use in your design. Default: 36
Maximum number of global nets that could be demoted to row-globals	Maximum number of global nets that could be demoted to row-globals. Range: 0 - 50 Default: 16
Minimum fanout of global nets that could be demoted to row-globals	Minimum fanout of global nets that could be demoted to row-global. It is undesirable to have high fanout nets demoted using row globals because it may result in high skew. If you run out of global routing resources for your design, reduce this number to allow more globals to be demoted to Row Globals. Range: 25 - 5000 Default: 1000
Infer Gated Clocks from Enable-registers	Enable this option to infer gated clocks from enable registers. This option is unchecked by default. Additional sub-options are as follows: <ul style="list-style-type: none"> <li>Minimum number of Enable pins to Infer Gated Clock global - Enter the minimum number of enable pins to infer gated chip-level global. Default value is 1000.</li> <li>Minimum number of Enable pins to Infer Gated Clock row-global - Enter the minimum number of enable pins to infer gated clock row-global. Default value is 100.</li> </ul>
Detect Clock Domain Crossings	Enable this option to detect all clock domain crossings (CDC) in the RTL design that have paths either between two asynchronous clocks or two synchronous clocks but with a false path or max-delay constraint. For each crossing, analyze if the RTL design contains a control or data synchronizer circuit and report if it is considered "safe" according to the minimum requirements you specify below. The option is checked by default. <ul style="list-style-type: none"> <li>Minimum number of synchronizer registers - Default value is 2. Range of value can be from 2 - 9.</li> </ul>

The generated CDC report will not contain any synchronizer circuits formed with macros instantiated from the catalog. The report will contain all CDC inferred from the RTL design and explain the reason(s) of why a synchronizer is considered "unsafe":

- No synchronizer circuit detected
  - In case of control signals (reset,enable,set) on first and second of sync registers are not matching.
  - Combinational logic between the first and second register synchronizers.
  - Diversion between first and second register synchronizers (i.e., the fanout is greater than 1).
  - Only 1 register in the synchronizer circuit.
- Number of register levels in synchronizer logic is less than the specified threshold.
- Combinational logic detected at clock domain crossing.
  - Combinational logic is present between the source register(start instance) and the destination register(end instance) at the crossover.
- Divergence detected in the crossover path.
  - Source register (start instance) has fanout greater than 1 at the crossover.
- Enable signal for synchronizer registers does not have a safe crossing.
  - Enable of data synchronizer doesn't have a safe synchronizer circuit.
- Sources from different domains in fanin.
  - The destination is driven by multiple registers from different clock domains and are asynchronous to the destination register clock domain.
- Synchronizer registers have synchronous reset or set as control signal.
  - The Synchronizer registers have a synchronous set or reset even if shared by all. This is tagged as unsafe since the reset logic can move to the data path instead of connecting to the reset port of the register and hence lead to metastability.

### 6.1.3 Optimizations

The following table describes the Optimizations options

**Table 6-2.**


Option	Description
Enable retiming	Check this box to enable Retiming during synthesis. Retiming is the process of moving registers (register balancing) across combinational gates automatically to improve timing, while ensuring identical logic behavior. Default: Not checked (no retiming during synthesis)
Enable automatic compile point	Check this box to enable Automatic Compile Point during synthesis. Default: Checked (Automatic Compile Point enabled)
RAM optimized for	Guides the Synthesis tool to optimize RAMs to achieve your design goal. <ul style="list-style-type: none"> <li>• <b>High speed:</b> RAM is optimized for speed. The resulting synthesized design achieves better performance (higher speed) at the expense of more dynamic power. (<i>Default</i>)</li> <li>• <b>Low power:</b> RAM is optimized for power. RAMs are inferred and configured to ensure the lowest power consumption.</li> </ul>
Map seq-shift register components to:	Maps sequential shift registers. Choices are: <ul style="list-style-type: none"> <li>• <b>Registers:</b> Sequential shift logic in the RTL maps to registers.</li> <li>• <b>RAM64x12:</b> Sequential shift logic in the RTL maps to a 64x12 RAM block. This is the default setting.</li> </ul>

.....continued	
Option	Description
Map ROM components to:	Maps ROM components. Choices are: <ul style="list-style-type: none"> <li>• <b>Logic</b>: Maps ROM components to logic. (<i>Default</i>)</li> <li>• <b>RAM</b>: Maps ROM components to RAM.</li> </ul>

#### 6.1.4 Additional Options for SynplifyPro Synthesis

The following table describes additional options for SynplifyPro Synthesis.

**Table 6-3. Additional Options for SynplifyPro Synthesis**

Option	Description
Script File	Click the Browse (  ) button to navigate to a Synplify Tcl file that contains the SynplifyPro-specific options. Libero passes the options in the Tcl file to SynplifyPro for processing.
Additional Options	Enter additional Synplify options. Place each option on a separate line. Libero passes these additional options “as-is” to SynplifyPro for processing, without checking syntax. All of these options are set on the Active Implementation only. <b>Note:</b> Options from the Additional Options Editor have priority over Tcl Script file options if they are the same.

#### Recommended Synthesis Tcl Options

You can add or modify the following list of recommended Synthesis Tcl options in the Tcl Script File or Additional Options Editor.

```
set_option -use_fsm_explorer 0/1
set_option -frequency 200.000000
set_option -write_verilog 0/1
set_option -write_vhdl 0/1
set_option -resolve_multiple_driver 1/0
set_option -rw_check_on_ram 0/1
set_option -auto_constrain_io 0/1
set_option -run_prop_extract 1/0
set_option -default_enum_encoding default/onehot/sequential/gray
set_option -maxfan 30000
set_option -report_path 5000
set_option -update_models_cp 0/1
set_option -preserve_registers 1/0
set_option -continue_on_error 1/0
set_option -symbolic_fsm_compiler 1/0
set_option -compiler_compatible 0/1
set_option -resource_sharing 1/0
set_option -write_apr_constraint 1/0
set_option -dup 1/0
set_option -enable64bit 1/0
set_option -fanout_limit 50
set_option -frequency auto
set_option -hdl_define SLE_INIT=2
set_option -hdl_param -set "width=8"
set_option -looplimit 3000
set_option -fanout_guide 50
set_option -maxfan_hard 1/0
set_option -num_critical_paths 10
set_option -safe_case 0/1
```

#### Entering Additional Options

You can enter any additional options using the Script File or Additional Options Editor. All of these options can be added and modified outside of Libero through interactive SynplifyPro.

For more information about the options and supported families, see the *SynplifyPro Reference Manual*. The following options are set by Libero. Do not include them in the additional options field or Script File:

```
add_file <*>
impl <*>
project_folder <*>
add_folder <*>
constraint_file <*>
project <*>
project_file <*>
open_file <*>
set_option -part
set_option -package
set_option -speed_grade
set_option -top_module
set_option -technology
set_option -opcond
set_option -vlog_std
set_option -vhdl2008
set_option -disable_io_insertion
set_option -async_globalthreshold
set_option -clock_globalthreshold
set_option -globalthreshold
set_option -low_power_ram_decomp
set_option -retiming
set_option -automatic_compile_point
set_option -seqshift_to_uram
set_option -rom_map_logic
set_option -gclint_threshold
set_option -rgclint_threshold
set_option -low_power_gated_clock
set_option -report_preserve_cdc
set_option -min_cdc_sync_flops
```

### 6.1.5 SynplifyPro ME

SynplifyPro ME is the default synthesis tool for Libero SoC.

To run synthesis using SynplifyPro ME and its default settings, right-click **Synthesize** and choose **Run**.

To use custom settings, use the following procedure to run Synplify interactively.

1. If Synplify is your default synthesis tool, right-click **Synthesize** in the Libero SoC Design Flow window and choose **Open Interactively**. Synplify starts and loads the appropriate design files with preset default values.
2. From Synplify's **Project** menu, choose **Implementation Options**.
3. Set your specifications and click **OK**.
4. Deactivate synthesis of the defparam statement. The defparam statement is only for simulation tools and is not intended for synthesis. Embed the defparam statement between the **translate\_on** and **translate\_off** synthesis directives:

```
/* synthesis translate_off */
defparam M0.MEMORYFILE= "meminit.dat"
/*synthesis translate_on */
// rest of the code for synthesis
```

5. Click the **Run** button. Synplify compiles and synthesizes the design into an HDL netlist. The resulting \*.vnm files appear under **Synthesis Files** in the **Files** list.  
If errors appear after you click the **Run** button, use the Synplify editor to edit the file. Double-click the file name in the Synplify window showing the loaded design files. Your changes are saved to the original design file in your project.
6. To close Synplify, from the **File** menu, choose **Exit**. When prompted to save changes you made, click **Yes**.

**Note:** For a list of attributes related to Microchip devices, see the Microchip Attribute and Directive Summary in the Synplify online help.

**Note:** To add a clock constraint in Synplify, add **n:<net\_name>** in your SDC file. If you omit the **n:**, the constraint will not be added.

### 6.1.6 Identifying Debug Designs

Libero SoC integrates the Identify RTL debugger tool. Identify debugging software allows you to probe and debug your FPGA design directly in the source RTL. Use the software if the design behavior after programming is not in accordance with the simulation results.

The following list summarizes the Identify key features:

- Instrument and debug your FPGA directly from RTL source code.
- Internal design visibility at full speed.
- Incremental iteration - Design changes are made to the device from the Identify environment using incremental compile operations. This allows you to iterate in a fraction of the time it takes route the entire device.
- Debug and display results - You collect only the data you need using unique and complex triggering mechanisms.

To open the Identify RTL debugger, in the Design Flow window, under **Debug Design**, double-click **Instrument Design**.

The following procedure describes how to use the Identify Instrumentor and Debugger. To run the debugging flow described below, you must have the Identify RTL Debugger and the Identify Instrumentor.

1. Create your source file and run pre-synthesis simulation.
2. Optional: Perform an entire flow (Synthesis - Compile - Place and Route - Generate a Programming File) without starting Identify.
3. Right-click **Synthesize** and choose **Open Interactively** in Libero SoC to launch Synplify.
4. In Synplify, click **Options > Configure Identify Launch** to setup Identify.
5. In Synplify, create an Identify implementation by clicking **Project > New Identify Implementation**.
6. In the Implementations Options dialog box, make sure the **Implementation Results > Results Directory** points to a location under `<libero project>\synthesis\`; otherwise, Libero SoC will not detect your resulting Verilog Netlist file.
7. From the Instrumentor UI, specify the sample clock, the breakpoints, and other signals to probe. Synplify creates a new synthesis implementation. Synthesize the design.
8. In Libero SoC, run Synthesis, Place and Route and generate a programming file.  
**Note:** Libero SoC works from the edit netlist of the current active implementation, which is the implementation you created in Synplify for Identify debug.
9. In the Design Flow window, double-click **Identify Debug Design** to launch the Identify Debugger.

To work properly, the Identify RTL Debugger, Synplify, and FlashPro must be synchronized. For more information about which versions of the tools work together, see the [Release Notes](#).

## 6.2 Verifying Post-Synthesized Designs

The following topics describe how to verify designs that have been synthesized.

### 6.2.1 Generating Simulation Files

This step generates the post-synthesis Verilog or VHDL netlist for post-synthesis simulation.

- The post-synthesis Verilog netlist ends with a `*.v` extension.
- The VHDL netlist ends with a `*.vhd` extension.

Post-synthesis simulation verifies the post-synthesis implementation of the design.

The netlist file is located in the synthesis folder of the project. Libero SoC passes this file to the simulator for the post-synthesis simulation run.

**Note:** Before performing post-synthesis, the design must pass the synthesis process. If you have not run synthesis, generating Simulation Files initiates a synthesis run automatically.

### 6.2.2 Verifying Post-Synthesis Implementations - Simulate

The steps for performing [functional](#) (post-synthesis) and timing (post-layout) simulation are nearly identical.

- You perform functional simulation before place-and-route to simulate the functionality of the logic in the design.

- You perform timing simulation after the design completes place-and-route. This simulation uses timing information based on the delays in the placed-and-routed designs.

To perform functional simulation:

1. Back-annotate your design and create your testbench.
2. In the Design Flow window, click **Implement Design**.
3. Right-click **Simulate** and choose **Organize Input Files > Organize Stimulus Files**.  
In the Organize Files for Source dialog box, all stimulus files in the current project appear in **Source Files** in the **Project** list box. Files associated with the block appear in the **Associated Source Files** list box.  
  
In most cases, you will have one testbench associated with your block. However, if you want simultaneous association of multiple testbench files for one simulation session, as in the case of PCI cores, add multiple files to the **Associated Source Files** list.
4. To add a testbench, select the testbench you want to associate with the block in the **Source Files** in the **Project** list box and click **Add** to add it to the **Associated Source Files** list.
5. To remove a testbench or change the files in the **Associated Source Files** list box, select the files and click **Remove**.
6. To order testbenches, use the up and down arrows to define the order in which you want the testbenches compiled. The top level-entity should be at the bottom of the list.
7. When you are satisfied with the **Associated Simulation Files** list, click **OK**.
8. To start ModelSim ME, right-click **Simulate** in the Design Hierarchy window and choose **Open Interactively**. ModelSim starts and compiles the appropriate source files. When the compilation completes, the simulator runs for 1ps and the Wave window shows the simulation results.
9. Scroll in the Wave window to verify the logic works as intended. Use the cursor and zoom buttons to zoom in, zoom out, and measure timing delays.
10. When you are done, choose **Quit** from the **File** menu.

## 6.3 Compile Netlist

Compile contains functions that perform legality checking and basic netlist optimization. Compile checks for netlist errors (bad connections and fan-out problems) removes unused logic (gobbling) and combines functions to reduce logic count and improve performance. Compile also verifies that your selected device has sufficient resources to fit your design.

The Compile Netlist step appears in the Design Flow window after unchecking the **Enable Synthesis** option in the **Project > Project Settings > Design Flow** page. This option appears after importing or linking your HDL Netlist files into the project and building the design hierarchy.

To compile your device with default settings, right-click **Compile Netlist** in the Design Flow window and choose **Run** or double-click **Compile Netlist**.

To compile your design with custom settings, right-click **Compile Netlist** in the Design Flow window and choose **Configure Options**.

During compile, the Log window shows information about your design, including warnings and errors. Libero SoC issues warnings when your design violates recommended Microchip design rules. Microchip recommends that you address all warnings, if possible, by modifying your design before you continue.

If the design fails to compile due to errors, modify the design to remove the errors and re-Compile.

The Compile Netlist Options set the threshold value for global resource promotion and demotion when Place and Route is executed.

Figure 6-2. Compile Netlist Options Dialog Box

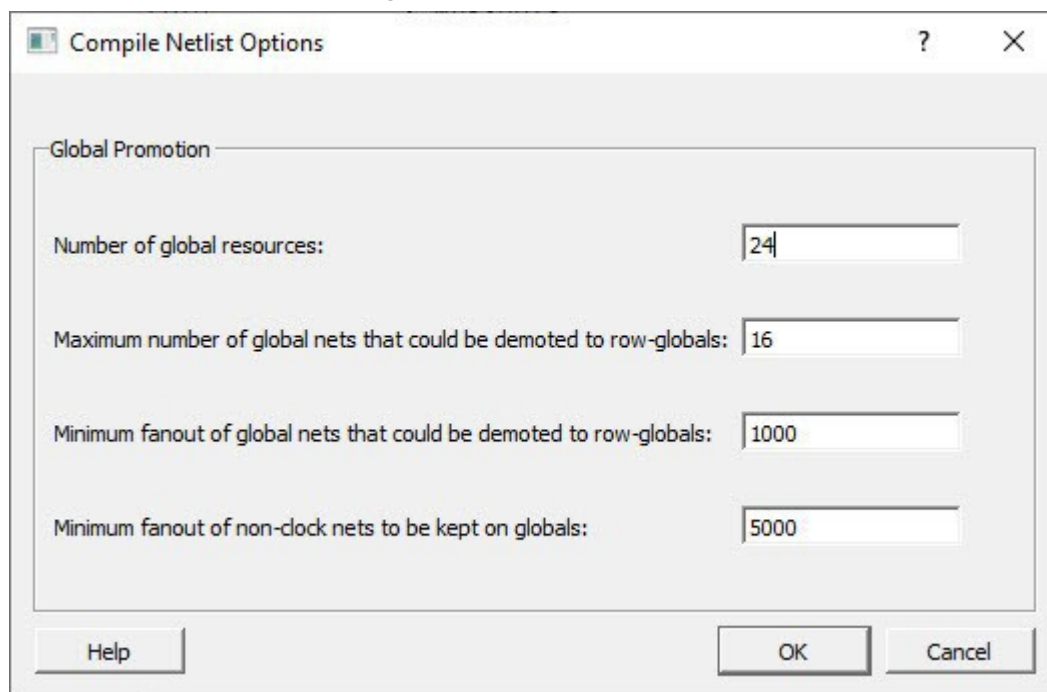


Table 6-4. Compile Netlist Options







Option	Description
Number of global resources	Number of available global resources for the die.
Maximum Number of global nets that could be demoted to row-globals	Maximum number of global nets that can be demoted to row-globals. Default: 16
Minimum fanout of global nets that could be demoted to row-globals	Minimum fanout of global nets that can be demoted to row-global. If you run out of global routing resources for your design, reduce this number to allow more globals to be demoted or select a larger die for your design. Default: 1000
Minimum fanout of non-clock nets to be kept on globals	Minimum fanout of non-clock (data) nets to be kept on globals (no demotion). If you run out of global routing resources for your design, increase this number. Range: 1000 - 200000 Default: 5000

## 6.4 Constraint Flow in Implementation

### 6.4.1 Design State Invalidation


The Libero SoC Design Flow window displays status icons to indicate the status of the design state. For any status other than a successful run, the status icon is identified with a tooltip to give you additional information.



Status Icon	Tooltip	Description	Possible Causes/Remedy
N/A	Tool has not run yet.	NEW state.	Tool has not run or it has been cleaned.
	Tool runs successfully.	Tool runs with no errors. PASS state.	N/A
	Tool forced by user to completed state.	Force updates the tools state to PASS state.	Only for Synthesize/Compile and Place and Route tools. The remaining tools do not change states
	Varies with the tool.	Tool runs but with Warnings.	Varies with the tool (e.g., for the Compile Netlist step, not all I/Os have been assigned and locked).
	Tool Fails.	Tool fails to run.	Invalid command options or switches, invalid design objects, invalid design constraints.
	Design State is Out of Date.	Tool state changes from PASS to OUT OF DATE.	Since the last successful run, design source design files, constraint files or constraint file/tool association, constraint files order, tool options, and/or project settings have changed.
	Timing Constraints have not been met.	Timing Verification runs successfully but the design fails to meet timing requirements.	Design fails Timing Analysis. Design has either set-up or hold time violations or both. See <a href="#">PolarFire FPGA Timing Constraints User Guide</a> on how to resolve the timing violations.

### 6.4.2 Constraints and Design Invalidation

A tool in the Design Flow changes from a PASS state (green check mark) to an OUT OF DATE state when a source file or setting affecting the outcome of that tool has changed.

The out-of-date design state is identified by the  icon in the Design Flow window. Sources and/or settings are defined as:

- HDL sources (for Synthesis), gate level netlist (for Compile), and Smart Design components
- Design Blocks (\*.cxz files) – low-level design units which may have completed Place and Route and re-used as components in a higher-level design
- Constraint files associated with a tool
- Upstream tools in the Design Flow:
  - If the tool state of a Design Flow tool changes from PASS to OUT OF DATE, the tool states of all the tools below it in the Design Flow, if already run and are in PASS state, also change to OUT OF DATE with appropriate tooltips. For example, if the Synthesis tool state changes from PASS to OUT OF DATE, the tool states of Place and Route tool as well as all the tools below it in the Design Flow change to OUT OF DATE.
    - If a Design Flow tool is CLEANED, the tool states of all the tools below it in the Design Flow, if already run, change from PASS to OUT OF DATE.
    - If a Design Flow tool is rerun, the tool states of all the tools below it in the Design Flow, if already run, are CLEANED.
  - Tool Options
    - If the configuration options of a Design Flow tool (right-click the tool and choose **Configure Options**) are modified, the tool states of that tool and all the other tools below it in the Design Flow, if already run, are changed to OUT OF DATE with appropriate tooltips.
  - Project Settings:
    - Device selection
    - Device settings
    - Design Flow
    - Analysis operating conditions

Setting Changed	Note	Design Flow Tools Affected	New State of the Affected Design Flow Tools
Die	Part# is changed	All	CLEANED/NEW
Package	Part# is changed	All	CLEANED/NEW
Speed	Part# is changed	All	CLEANED/NEW
Core Voltage	Part# is changed	All	CLEANED/NEW
Range	Part# is changed	All	CLEANED/NEW
Default I/O Technology		Synthesize, and all tools below it	OUT OF DATE
Reserve Pins for Probes		Place and Route, and all tools below it	OUT OF DATE
PLL Supply Voltage (V)		Verify Power, Generate FPGA Array Data and all other "Program and Debug Design" tools below it	OUT OF DATE
Power On Reset Delay		Generate FPGA Array Data and all other "Program and Debug Design" tools below it	OUT OF DATE
System controller suspended mode		Generate FPGA Array Data and all other "Program and Debug Design" tools below it	OUT OF DATE
Preferred Language		None	N/A
Enable synthesis		All	OUT OF DATE
Synthesis gate level netlist format		Synthesize	CLEANED/NEW
Reports(Maximum number of high fanout nets to be displayed)		None	N/A
Abort flow if errors are found in PDC		None	N/A
Abort flow if errors are found in SDC		None	N/A
Temperature range(C)		Verify Timing, Verify Power	OUT OF DATE
Core voltage range(V)		Verify Timing, Verify Power	OUT OF DATE
Default I/O voltage range		Verify Timing, Verify Power	OUT OF DATE

**Note:** Cleaning a tool means the output files from that tool are deleted including log and report files, and the tool's state is changed to NEW.

#### 6.4.3 Check Constraints

When a constraint file is checked, the Constraint Checker does the following:

- Checks the syntax

- Compares the design objects (pins, cells, nets, ports) in the constraint file versus the design objects in the netlist (RTL or post-layout ADL netlist). Any discrepancy (e.g., constraints on a design object which does not exist in the netlist) are flagged as errors and reported in the \*\_sdc.log file

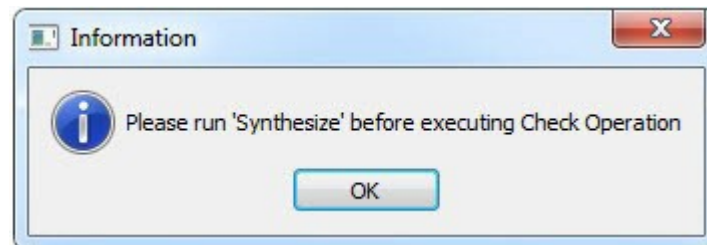
#### 6.4.4 Design State and Constraints Check

Constraints can be checked only when the design is in the right state.

Constraint Type	Check for Tools	Required Design State Before Checking	Netlist Used for Design Objects Checks	Check Result
I/O Constraints	Place and Route	Post-Synthesis	ADL Netlist	Reported in Libero Log Window
Floorplanning Constraints	Place and Route	Post-Synthesis	ADL Netlist	par_sdc.log
Timing Constraints	Synthesis	Pre-Synthesis	RTL Netlist	synthesis_sdc.log
	Place and Route	Post-Synthesis	ADL Netlist	par_sdc.log
	Timing Verification	Post-Synthesis	ADL Netlist	vt_sdc.log
Netlist Attributes	FDC Check	Pre-Synthesis	RTL Netlist	Libero Message Window
Netlist Attributes	NDC Check	Pre-Synthesis	RTL Netlist	Reported in Libero Log Window

A pop-up message appears when the check is made and the design flow has not reached the right state.

**Figure 6-3. Pop-Up Message: Design State Insufficient for Constraints Check Operation**



#### 6.4.5 Edit Constraints

Click the **Edit with I/O Editor/Chip Planner/Constraint Editor** button to edit existing and add new constraints. Except for the Netlist Attribute constraints (\*.fdc and \*.ndc) file, which cannot be edited by an interactive tool, all other constraint types can be edited with an Interactive Tool. The \*.fdc and \*.ndc files can be edited using the Libero SoC Text Editor.

The I/O Editor is the interactive tool to edit I/O Attributes, Chip Planner is the interactive tool to edit Floorplanning Constraints, and the Constraint Editor is the interactive tool to edit Timing Constraints.

For Timing Constraints that can be associated to Synthesis, Place and Route, and Timing Verification, you need to specify which group of constraint files you want the Constraint Editor to read and edit:

- **Edit Synthesis Constraints** - reads associated Synthesis constraints to edit.
- **Edit Place and Route Constraints** - reads only the Place and Route associated constraints.
- **Edit Timing Verification Constraints** - reads only the Timing Verification associated constraints.

For the three SDC constraints files (a.sdc, b.sdc, and c.sdc, each with Tool Association as shown in the table below) when the Constraint Editor opens, it reads the SDC file based on your selection and the constraint file/tool association.

	Synthesis	Place and Route	Timing Verification
a.sdc		X	X
b.sdc	X	X	
c.sdc [target]	X	X	X

- **Edit Synthesis Constraints** reads only the b.sdc and c.sdc when Constraint Editor opens.
- **Edit Place and Route Constraints** reads a.sdc, b.sdc and c.sdc when Constraint Editor opens.
- **Edit Timing Verification Constraints** reads a.sdc and c.sdc when Constraint Editor opens.

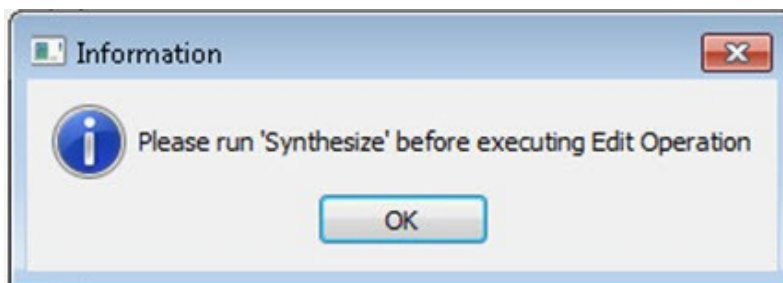
Constraints in the SDC constraint file that are read by the Constraint Editor and subsequently modified by you will be written back to the SDC file when you save the edits and close the Constraint Editor.

When you add a new SDC constraint in the Constraint Editor, the new constraint is added to the c.sdc file, because it is set as target. If no file is set as target, Libero SoC creates a new SDC file to store the new constraint.

### 6.4.6 Constraint Type and Interactive Tool

Constraint Type	Interactive Tool for Editing	Design Tool the Constraints File is Associated	Required Design State Before Interactive Tool Opens for Edit
I/O Constraints	I/O Editor	Place and Route Tool	Post-Synthesis
Floorplanning Constraints	Chip Planner	Place and Route Tool	Post-Synthesis
Timing Constraints	SmartTime Constraints Editor	Synthesis Tool Place and Route Timing Verification	Pre-Synthesis Post-Synthesis Post-Synthesis
Netlist Attributes Synplify Netlist Constraint (*.fdc)	Interactive Tool Not Available Open the Text Editor to edit.	Synthesis	Pre-Synthesis
Netlist Attributes Compile Netlist Constraint (*.ndc)	Interactive Tool Not Available Open the Text Editor to edit.	Synthesis	Pre-Synthesis

**Note:** If the design is not in the proper state when **Edit with <Interactive tool>** is invoked, a pop-up message appears.



**Note:** When an interactive tool is opened for editing, the Constraint Manager is disabled. Close the Interactive Tool to return to the Constraint Manager.

## 6.5 Place and Route

Double-click **Place and Route** to run Place and Route on your design with the default settings.

### 6.5.1 Place and Route Options

To change your place and route settings from the Design Flow window, expand **Implement Design**, right-click **Place and Route**, and choose **Configure Options**. When the Layout Options dialog box appears, specify your settings, and then click **OK**.

Figure 6-4. Layout Options Dialog Box

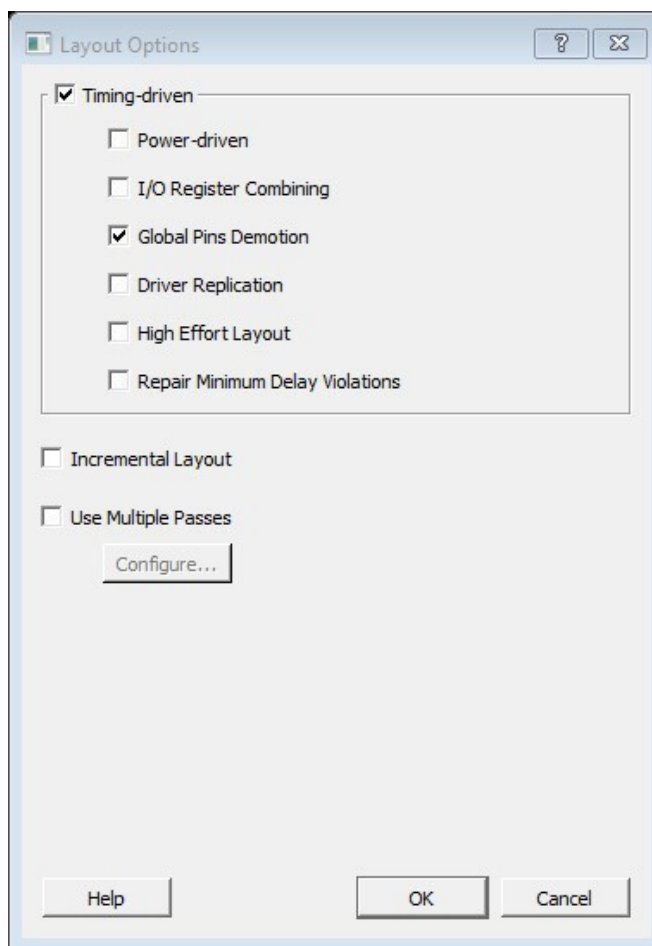
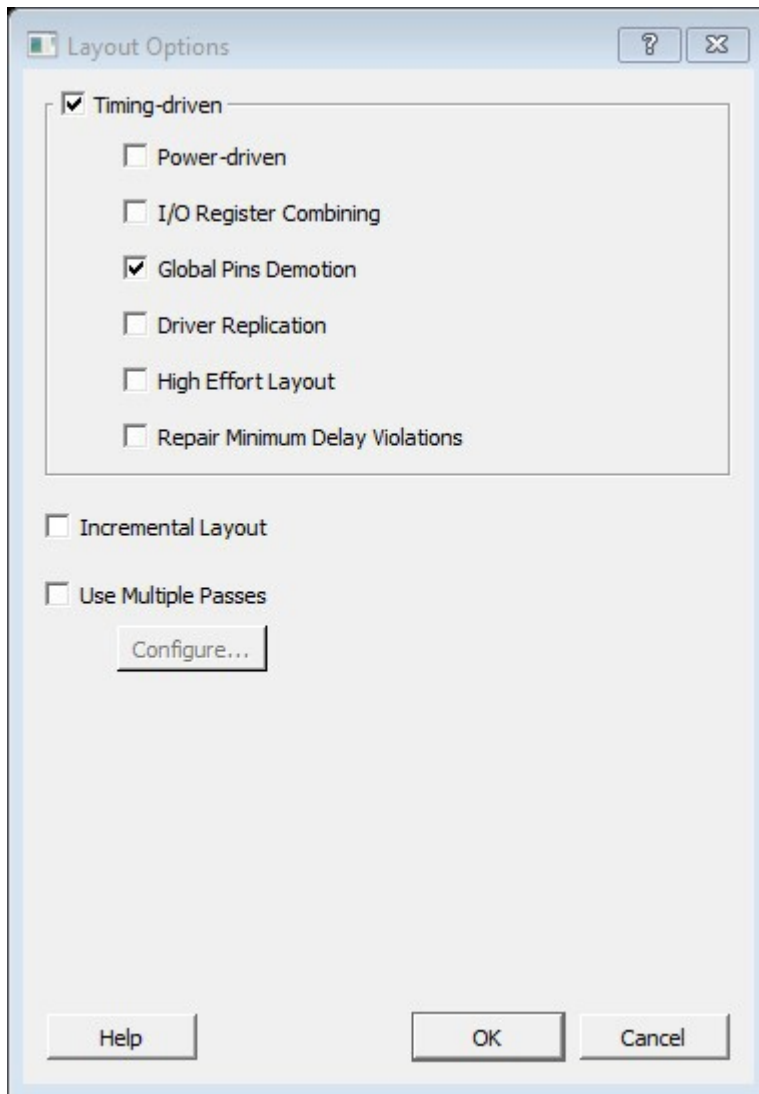


Figure 6-5. Layout Options Dialog Box with Block Flow Enabled



The following table describes the place and route options.

**Table 6-5. Place and Route Options**

Option	Description
Timing-Driven	Timing-driven Place and Route strives to meet timing constraints specified by you or generated automatically. Timing-driven Place and Route delivers better performance than Standard Place and Route. If you do not select this option, Libero SoC ignores timing constraints, although timing reports based on timing constraints can still be generated for the design.
Power-driven	Runs the Power-Driven layout. This layout reduces dynamic power while still maintaining timing constraints.

.....continued

Option	Description
I/O Register Combining	Combines any register directly connected to an I/O when it has a timing Constraint. If multiple registers are directly connected to a bi-directional I/O, select one register to combine in the following order: input-data, output-data, output-enable.
Global Pins Demotion	This option is selected when Timing-Driven Place and Route is enabled. This option allows the layout tool to select the most timing critical pins on any Global network and moves them to the source that drives the Global resource (proactively attempts to improve timing by putting timing-critical pins onto routed resources). If the driver for the global is a fabric register, the driver is replicated and the duplicate names are printed. Each set of names should be used in place of the original register in any specified timing constraint.
Driver Replication	Allows an algorithm to replicate critical net drivers to reduce timing violations. The algorithm prints the list of registers along with the duplicate names. Each set of names should be used in place of the original register in any specified timing constraint.
High Effort Layout	Improves layout success, but increases layout runtime and may impact timing performance.

.....continued

Option	Description
Repair Minimum Delay Violations	<p>Repairs Minimum Delay violations (Timing-Driven Place and Route option enabled) and performs an additional route. This is done by increasing the length of routing paths and inserting routing buffers to add delay to the top violating paths.</p> <p>If this option is enabled, the programmable delays through I/Os are adjusted to meet hold time requirements from input to registers. For register-to-register paths, buffers are inserted.</p> <p>The Repair tool analyzes paths iteratively with negative minimum delay slacks (hold time violations), and chooses suitable connections and locations to insert buffers. Not all paths can be repaired using this technique, but many common cases will benefit.</p> <p>Even when this option is enabled, it will not repair a connection or path that:</p> <ul style="list-style-type: none"> <li>• Is a hardwired, preserved, or global net.</li> <li>• Has a sink pin which is a clock pin.</li> <li>• Violates a maximum delay constraint (that is, the maximum delay slack for the pin is negative).</li> <li>• May cause the maximum delay requirement for the sink pin to be violated (setup violations).</li> </ul> <p>Typically, this option is enabled with the Incremental Layout option when a design's maximum delay requirements have been satisfied.</p> <p>Every effort is made to avoid creating max-delay timing violations on worst case paths.</p> <p>Min Delay Repair generates a report in the implementation directory that lists all of the paths that were considered.</p> <p>If your design continues to have internal hold time violations, rerun repair Minimum Delay Violations with Incremental Layout to analyze additional paths.</p>
Incremental Layout	<p>Uses previous placement data as the initial placement for the next run.</p> <p>To preserve portions of your design, use Compile Points, which are RTL partitions of the design that you define before synthesis. The synthesis tool treats each Compile Point as a block that allows you to preserve its structure and timing characteristics. By executing Layout in Incremental Mode, locations of previously placed cells and the routing of previously routed nets is preserved. Compile Points makes it easy to mark portions of a design as black boxes, and let you divide the design effort between designers or teams. for more information, see the <a href="#">Synopsys FPGA Synthesis Pro ME User Guide</a>.</p>
Use Multiple Pass	<p>Runs multiple passes of Place and Route to achieve the best layout result. Click <b>Configure</b> to specify the criteria you want to use to determine the best layout result.</p>



.....continued

Option	Description
Block Creation	This option is available only when the Block Creation option is turned on ( <b>Project &gt; Project Settings &gt; Design Flow &gt; Enable Block Creation</b> ). The value entered here limits the number of row-global resources available in every row-global region of the device. During Place and Route of the block, the tool will not exceed this capacity on any row-global region. The default value is the maximum number of row-globals. If you enter a value lower than the maximum capacity (the default), the layout of the block can integrate with the rest of the design if it consumes the remaining row-global capacity.

## 6.6 Multiple Pass Layout Configuration

Multiple Pass Layout attempts to improve layout quality by selecting results from a few number of Layout passes. This is done by running individual place and route multiple times with varying placement seeds and measuring the best results for the specified criteria.

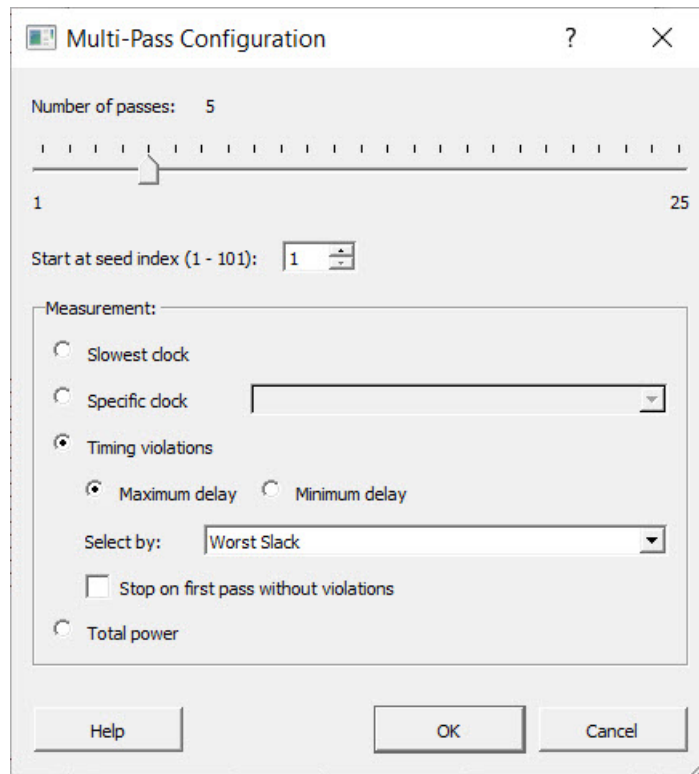
When using Multiple Pass Layout, observe the following guidelines:

- Multiple Pass Layout saves your design file with the pass that has the best layout results. If you want to preserve your existing design state, you should save your design file with a different name before proceeding. To do this, from the File menu, choose **Save As**.
- The following four types of reports for each pass are written to the working directory to assist you in later analysis: timing, maximum delay timing violations, minimum delay timing violations, and power.
  - <root\_module\_name>\_timing\_r<runNum>\_s<seedIndex>.rpt
  - <root\_module\_name>\_timing\_violations\_r<runNum>\_s<seedIndex>.rpt
  - <root\_module\_name>\_timing\_violations\_min\_r<runNum>\_s<seedIndex>.rpt
  - <root\_module\_name>\_power\_r<runNum>\_s<seedIndex>.rpt
  - <root\_module\_name>\_iteration\_summary.rpt provides additional details about the saved files.

To configure multiple pass options:

1. When running Layout, select the Use Multiple Passes in the Layout Options dialog box.
2. Click **Configure**. The Multi-Pass Configuration dialog box appears.

Figure 6-6. Multi-Pass Configuration Dialog Box



3. Set the options in the following table and click **OK**.

Table 6-6. Multi-Pass Configuration Options

Option	Description
Number of passes	Number of passes (iterations) using the slider. 1 is the minimum and 25 is the maximum. Default: 5
Start at seed index	Index into the array of random seeds that is to be the starting point for the passes. If not specified, the default behavior is to continue from the last seed index that was used.
Measurement	Measurement criteria against which you want to compare layout results.
Slowest clock	Uses the slowest clock frequency in the design in a given pass as the performance reference for the layout pass.
Specific clock	Uses a specific clock frequency as the performance reference for all layout passes.
Timing violations	Timing Violations to use the pass that best meets the slack or timing-violations constraints. This is the default. <b>Note:</b> You must enter your own timing constraints through SmartTime or SDC.
Maximum delay	Examines timing violations (slacks) obtained from maximum delay analysis. This is the default.

.....continued	
Option	Description
Minimum delay	Examines timing violations (slacks) obtained from minimum delay analysis.
Select by	Slack criteria. Choices are: <ul style="list-style-type: none"> <li>Worst Slack: Largest amount of negative slack (or least amount of positive slack if all constraints are met) for each pass is identified, and the largest value of all passes determines the best pass. <i>(Default)</i></li> <li>Total Negative Slack. Sum of negative slacks from the first 100 paths in the Timing Violations report for each pass is identified, and the largest value of all the passes determines the best pass. If no negative slacks exist for a pass, the worst slack is used to evaluate that pass.</li> <li>Stop on first pass without violations: Stops performing remaining passes if all timing constraints have been met when there are no negative slacks reported in the timing violations report.</li> </ul>
Total power	Determines the best pass to be the one that has the lowest total power (static + dynamic) of all layout passes.

### 6.6.1 Iteration Summary Report

The file <root\_module>\_iteration\_summary.rpt records a summary of whether the multiple pass run was invoked through the GUI or extended\_run\_lib Tcl script, with arguments for repeating each run. Each new run appears with its own header in the Iteration Summary Report with fields RUN\_NUMBER and INVOKED AS, followed by a table containing Seed Index, corresponding Seed value, Comparison data, Report Analyzed, and Saved Design information.

Figure 6-7. Iteration Summary Report

```

1 #-----
2 # RUN NUMBER: 1  DATE: 15:21:35 04-Mar-2015
3 # INVOKED AS: W:\po\11_5_7_4_cj\Designer\bin\libero.exe extended_run_lib.tcl -skip_open_project -root (C:\Actelprj\efusion2\designer\shift_reg32) -n 5 -starting_se
4 #-----
5 # Seed Index  Seed      Maximum Delay      Worst Slack  Report Analyzed      Saved Design
6 #-----
7 0 0 M/A Compare failed shift_reg32_timing_violations_max_r1_s1.rpt shift_reg32_r1_s1
8 1 86662958 M/A Layout failed shift_reg32_timing_violations_max_r1_s2.rpt shift_reg32_r1_s2
9 2 8988747 M/A Layout failed shift_reg32_timing_violations_max_r1_s3.rpt shift_reg32_r1_s3
10 3 51071856 M/A Layout failed shift_reg32_timing_violations_max_r1_s4.rpt shift_reg32_r1_s4
11 4 78381505 M/A Layout failed shift_reg32_timing_violations_max_r1_s5.rpt shift_reg32_r1_s5
12 #-----
13 # RUN NUMBER: 2  DATE: 15:28:27 04-Mar-2015
14 # INVOKED AS: W:\po\11_5_7_4_cj\Designer\bin\libero.exe extended_run_lib.tcl -skip_open_project -root (C:\Actelprj\efusion2\designer\shift_reg32) -n 5 -starting_se
15 #-----
16 # Seed Index  Seed      Maximum Delay      Worst Slack  Report Analyzed      Saved Design
17 #-----
18 5 82287664 M/A -3.359 shift_reg32_timing_violations_max_r2_s6.rpt shift_reg32_r2_s6
19 6 23702026 M/A -3.174 shift_reg32_timing_violations_max_r2_s7.rpt shift_reg32_r2_s7
20 7 51370950 M/A -3.413 shift_reg32_timing_violations_max_r2_s8.rpt shift_reg32_r2_s8
21 8 93189207 M/A -3.168 shift_reg32_timing_violations_max_r2_s9.rpt shift_reg32_r2_s9
22 9 17538078 M/A -3.354 shift_reg32_timing_violations_max_r2_s10.rpt shift_reg32_r2_s10
23 #-----

```

## 6.7 Post Layout Editing of I/O Signal Integrity and Delay Parameters

The **Edit Post Layout Design** tool in the Design Flow allows you to tune I/O signal integrity parameters and external timing without executing Place and Route again.

Input is provided using a PDC file. From the Libero user interface, double-click **Edit Post Layout Design** to open a file selection dialog box and selecting the input file. In the batch flow, you can issue the command `edit_post_layout_design <input.pdc>`.

The PDC file contains one or more invocations of two PDC commands:

- `edit_io`
- `edit_instance_delay`

For more information about these commands, see the *PDC Command User Guide - Polarfire*.

To assist you in knowing those instances on which this tool can update delays, the Place and Route tool generates a `<root>_delayinstance.rpt` report file. This report has an editable (**Editable?**) column with **Yes** and **No** values that indicate whether or not a delay parameter can be edited by this tool.

The batch command `edit_post_layout_design` fails when any commands in the input PDC file fail. The PDC commands fail if the syntax is incorrect, the referenced instances do not exist, or the values are out of legal ranges. If the batch command fails, the layout state of the design does not change. If the batch command succeeds:

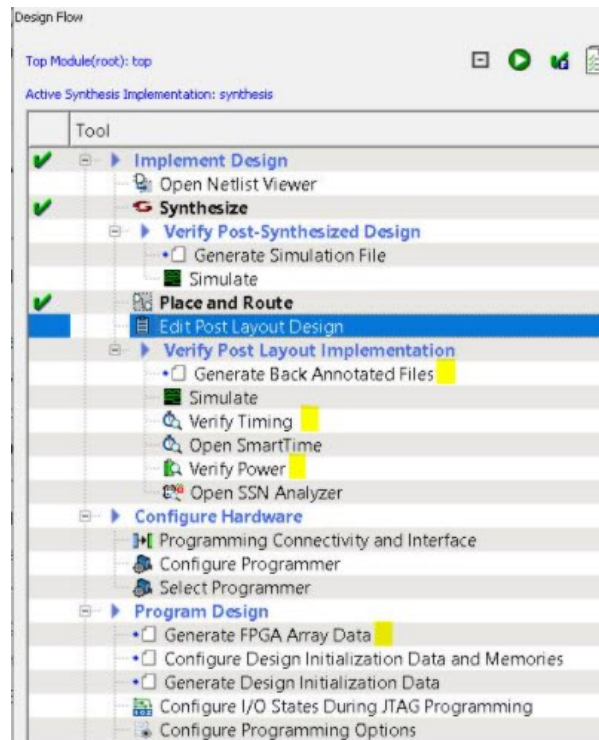
- Layout state changes to reflect the values in the PDC commands
- Pin report and delay instance report files are regenerated to reflect the latest values
- Downstream tools Verify Timing, Verify Power, Generate FPGA Array Data, and Generate Back Annotated Files are invalidated.

The batch command `edit_post_layout_design` generates the log file `<project>/designer/<root>/top_editpostlayout_log.log` to provide information about the run and which PDC file was used in the run.

This log gets appended with information from each run of the command. Each run's start message is prefixed by the time when the run executed. The log file can be viewed from the Reports tab (Design > Reports). The log file is removed when Place and Route is run or cleaned.

**Note:** In the following figure, downstream steps this tool invalidates are highlighted in yellow.

Figure 6-8. Edit Post Layout Design Tool in Design Flow



## 6.8 Resource Usage

After layout, you can check the resource usage of your design.

1. From the Design menu, choose **Design > Reports**.
2. Click **<design\_name>\_layout\_log.log** to open the log file.

The log file contains a Resource Usage report that lists the type and percentage of resource used for each resource type relative to the total resources available for the chip.

Table 6-7. Resource Usage

Type	Used	Total	Percentage
4LUT	400	86184	0.46
DFF	300	86184	0.34
I/O Register	0	795	0.00
Logic Element	473	86184	0.55

4LUTs are 4-input Look-up Tables that can implement any combinational logic functions with up to four inputs.

The Logic Element (LE) is a logic unit in the fabric. It may contain a 4LUT, a DFF, or both. The number of LEs in the report includes all LEs, regardless of whether they contain 4LUT only, DFF only, or both.

### 6.8.1 Overlapping of Resource Reporting

The number of 4LUTs in the report is the total number used for your design, regardless of whether they are combined with the DFFs. Similarly, the number of DFFs in the report is the total number used for your design, regardless of whether they are combined with 4LUT's.

In the report above, a total of 473 LEs are used for the design.

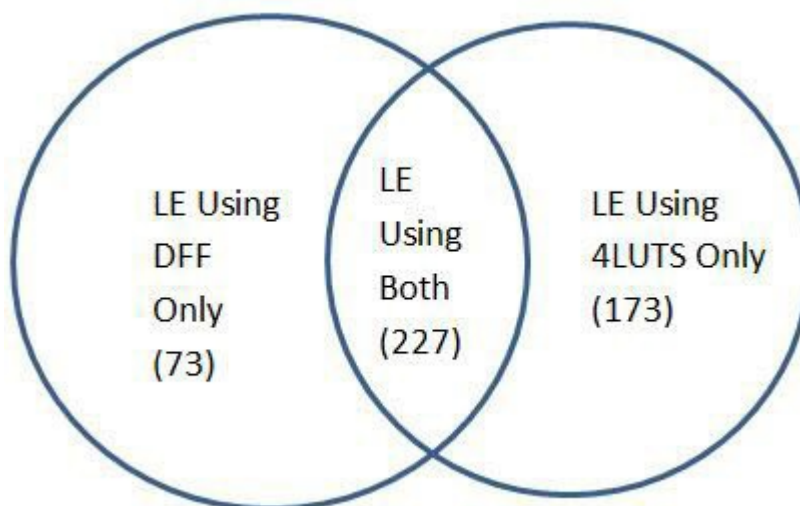
- 300 of the 473 LEs have DFFs inside, which means 173 (473-300) of them have no DFFs in them. These 173 LEs use only the 4LUTs portion of the LE.
- 400 of the 473 LEs have 4LUTs inside, which means 73 (473-400) of them have no 4LUTs in them. These 73 LEs use only the DFF portion of the LE.

**Table 6-8. Logical Element Calculations**

Logical Element Description	Calculated Logical Elements
LEs using DFF Only = $473 - 400 =$	73
LEs using 4LUTS only = $473 - 300 =$	173
	= 246 (Total of LEs using 4LUTS ONLY or DFF ONLY)
Report's Overlapped resource =	227 (LEs using both 4LUTS <i>and</i> DFF)
Total number of LEs used =	473

In the following figure, the area where the two circles overlap represents the overlapped resources in the Resource Usage report.

**Figure 6-9. Overlapped Resources**



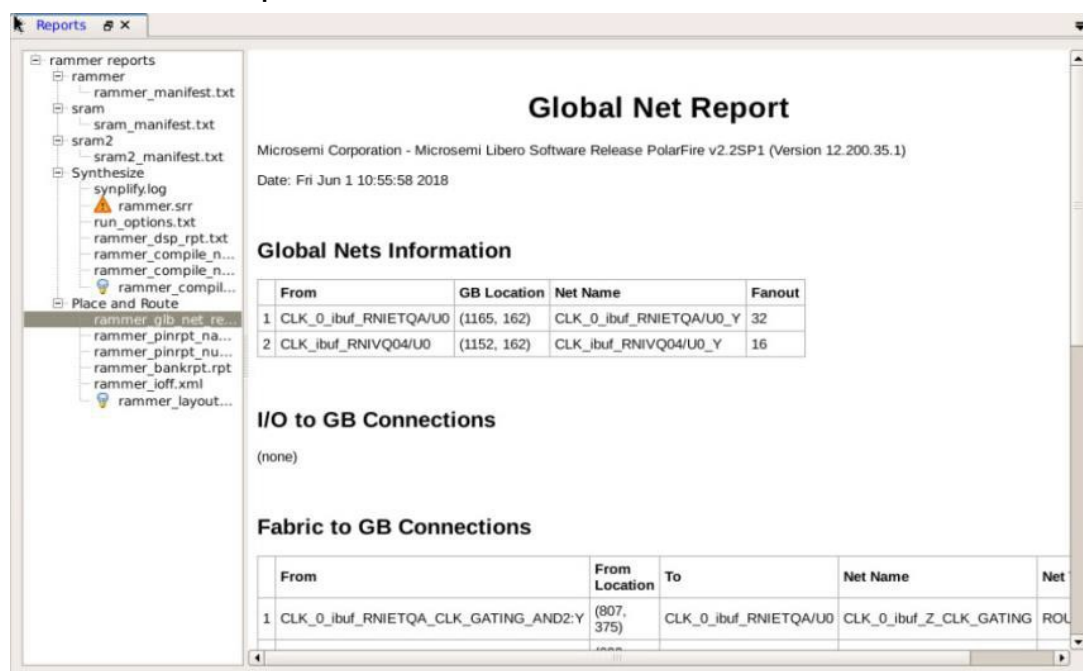
## 6.9 Global Net Report

The Global Net Report displays all the nets that use the global routing resources of the device. This report is generated after the Place and Route step and is available in XML format in the **Reports** tab (**Libero SoC > Design>Reports > <design\_name>\_glb\_net\_report.xml**).

The global routing resources in Microchip FPGA devices offer a low-skew network for effective distribution of high fanout nets including clock signals. Global routing resources include the following:

- Fabric CCC
- Global Buffers (GB)
- Row Global Buffers (RGB)

Figure 6-10. Global Net Report



The following topics describe the sections in the Global Net Report.

### 6.9.1 Global Nets Information

Under **Global Nets Information**, the **GB Location** column refers to the location of the global routing resource/instance name of the macro on the chip. The location is indicated by X-Y coordinates of the global resource macro.



Figure 6-11. Global Nets Information Section

## Global Nets Information

	From	GB Location	Net Name	Fanout
1	GB[4]	(726, 156)	reset_ctrl_i/R_core_reset_out_RNIFUQ6/U0_YWn	35701
2	GB[8]	(734, 156)	pll_i/GL0_INST/U0_YWn	35488
3	GB[15]	(741, 156)	pll_i/GL3_INST/U0_YWn	2006
4	GB[13]	(739, 156)	reset_ctrl_i/R_global_reset_RNIOT36/U0_YWn	1848
5	GB[14]	(740, 156)	pll_i/GL2_INST/U0_YWn_GEAs	1104
6	GB[2]	(724, 156)	serdes_i/SERDES_IF_0/EPCS_1_RX_CLK_keep_RNIEDL3/U0_YWn	514
7	GB[0]	(722, 156)	serdes_i/SERDES_IF_0/EPCS_0_RX_CLK_keep_RNID1J5/U0_YWn	513
8	GB[3]	(725, 156)	serdes_i/SERDES_IF_0/EPCS_2_RX_CLK_keep_RNIFPN1/U0_YWn	511
9	GB[7]	(729, 156)	serdes_i/pcs_gl_0_pcs_0/rx_rst_n_i_0_RNI9T3C/U0_YWn	312
10	GB[11]	(737, 156)	serdes_i/pcs_gl_1_pcs_0/rx_rst_n_i_0_RNIAGFA/U0_YWn	310
11	GB[1]	(723, 156)	serdes_i/SERDES_IF_0/EPCS_0_TX_CLK_keep_RNIFLD8/U0_YWn	178
12	GB[5]	(727, 156)	serdes_i/SERDES_IF_0/EPCS_2_TX_CLK_keep_RNIHD14/U0_YWn	178
13	GB[6]	(728, 156)	serdes_i/SERDES_IF_0/EPCS_1_TX_CLK_keep_RNIG1G6/U0_YWn	178
14	GB[10]	(736, 156)	SPI_SCLK_ibuf_RNIT4T6/U0_YWn_GEAs	149
15	GB[9]	(735, 156)	SRAM_CQ_ibuf_RNIB7IC/U0_YWn_GEAs	111
16	GB[12]	(738, 156)	SRAM_CQn_ibuf_RNIPMM6/U0_YWn_GEAs	18

## 6.9.2 I/O to GB Connections

The **I/O to GB Connections** section lists all the I/Os connected to the Global Resource/instance name of the macro.

Figure 6-12. I/O to GB Connections Section

### I/O to GB Connections

Port Name	Pin Number	I/O Function	From	From Location	To	Net Name	Net Type	Fanout
1 SPI_SCLK	D33	MSIO176NB18	SPI_SCLK_ibuf/U0/U_JOIN.Y	(6, 307)	GB[10]	SPI_SCLK_ibuf	ROUTED	1
2 SRAM_CQ	G16	DDRIO120PB2/MDDR_DQ_ECC0/CCC_NE1_CLK03	SRAM_CQ_ibuf/U0/U_JOIN.Y	North IO #7 (1005, 313)	GB[9]	SRAM_CQ_ibuf	ROUTED	1
3 SRAM_CQn	F16	DDRIO120PB2/MDDR_DQ_ECC1/GB12/CCC_NE1_CLK12	SRAM_CQn_ibuf/U0/U_JOIN.Y	North IO #6 (1002, 313)	GB[12]	SRAM_CQn_ibuf	HARDWIRED	1

Table 6-9. Columns in the I/O to GB Connections Section

Column	Description
I/O Function	<p>I/O connection details, such as the bank name, any hardwired GB or hardwired CCC connections, and any dedicated SERDES/DDR connections.</p> <ul style="list-style-type: none"> <li>For hardwired connections, the function name (DDRIO120PB2/MDDR_DQ_ECC1/GB12/CCC_NE1_CLK12) contains the GB index (GB12 in the example above) that matches the GB index in the <b>To</b> column (GBL[12] in the example above).</li> </ul> <p>For routed connections, the function name does not contain the proper GB index.</p>



.....continued

Column	Description
Net Type	<p>Routed or hardwired:</p> <ul style="list-style-type: none"> <li>Hardwired net types are dedicated wiring resources and have lower insertion delays.</li> <li>Routed net types are implemented using fabric routing resources and the insertion delay (generally higher than hardwired nets) and vary from iteration to iteration.</li> </ul>

### 6.9.3 Fabric to GB Connections

The **Fabric to GB Connections** section lists all the nets originating from the fabric to the Global Resources/Instance name of the macro. The **From Location** column refers to the X-Y coordinates of the driver pin of the net. The nets are routed nets, not hardwired.

Figure 6-13. Fabric to GB Connections Section

#### Fabric to GB Connections

	From	From Location	To	Net Name	Net Type	Fanout
1	reset_ctrl_vR_core_reset_out_Q	(720, 160)	GB[4]	reset_ctrl_vcore_reset_out	ROUTED	1
2	reset_ctrl_vR_global_reset_Q	(722, 160)	GB[13]	reset_ctrl_vglobal_reset_0	ROUTED	3
3	serdes_vSERDES_IF_0/SERDESIF_INST/INST_SERDESIF_IP.EPCS_RXCLK_1	(72, 2)	GB[2]	serdes_vSERDES_IF_0/EPCS_1_RX_CLK	ROUTED	1
4	serdes_vSERDES_IF_0/SERDESIF_INST/INST_SERDESIF_IP.EPCS_RXCLK_0	(72, 2)	GB[0]	serdes_vSERDES_IF_0/EPCS_0_RX_CLK	ROUTED	1
5	serdes_vSERDES_IF_0/SERDESIF_INST/INST_SERDESIF_IP.EPCS_RXCLK[0]	(72, 2)	GB[3]	serdes_vSERDES_IF_0/EPCS_2_RX_CLK	ROUTED	1
6	serdes_vpcs_gl_0_pcs_0/rx_rst_n_i_0_Y	(269, 54)	GB[7]	serdes_vpcs_gl_0_pcs_0/rx_rst_n_i_0_1	ROUTED	1
7	serdes_vpcs_gl_1_pcs_0/rx_rst_n_i_0_Y	(266, 96)	GB[11]	serdes_vpcs_gl_1_pcs_0/rx_rst_n_i_0	ROUTED	1
8	serdes_vSERDES_IF_0/SERDESIF_INST/INST_SERDESIF_IP.EPCS_TXCLK_0	(72, 2)	GB[1]	serdes_vSERDES_IF_0/EPCS_0_TX_CLK	ROUTED	1
9	serdes_vSERDES_IF_0/SERDESIF_INST/INST_SERDESIF_IP.EPCS_TXCLK[0]	(72, 2)	GB[5]	serdes_vSERDES_IF_0/EPCS_2_TX_CLK	ROUTED	1
10	serdes_vSERDES_IF_0/SERDESIF_INST/INST_SERDESIF_IP.EPCS_TXCLK_1	(72, 2)	GB[6]	serdes_vSERDES_IF_0/EPCS_1_TX_CLK	ROUTED	1

### 6.9.4 CCC to GB Connections

The **CCC to GB Connections** section lists the nets originating from the Clock Conditioning Circuitry (CCC) outputs (GLx) to the Global Resources/instance name of the macro. To minimize clock skew, CCC clock outputs usually are hardwired dedicated connections to Global resources (GB).

Figure 6-14. CCC to GB Connections Section

#### CCC to GB Connections

	From	From Location	To	Net Name	Net Type	Fanout
1	pll_i/CCC_INST/INST_CCC_IP:GL0	CCC-NE0 (1428, 302)	GB[8]	pll_i/GL0_net	HARDWIRED	1
2	pll_i/CCC_INST/INST_CCC_IP:GL3	CCC-NE0 (1428, 302)	GB[15]	pll_i/GL3_net	HARDWIRED	1
3	pll_i/CCC_INST/INST_CCC_IP:GL2	CCC-NE0 (1428, 302)	GB[14]	pll_i/GL2_net	HARDWIRED	1

### 6.9.5 CCC Input Connections

The **CCC Input Connections** section lists the nets from the I/O pins to the CCC inputs.

Net type can be routed or hardwired. Hardwired net types are dedicated wiring resources and have lower insertion delays. Routed net types are implemented using fabric routing resources and the insertion delay (generally higher than that of hardwired nets), varies from iteration to iteration.

Figure 6-15. CCC Input Connections Section

#### CCC Input Connections

Port Name	Pin Number	I/O Function	From	From Location	To (Pin Swapped for Back Annotation Only)	CCC Location	Net Name	Net Type	Fanout
1	FPGA_CLK_P V12	MSIO35PB5/CCC_NE0_CLK0	pll_vCLK0_PAD_INST/U_JOPADP_IOUT_P	East IO #0 (1455, 112)	pll_vCCC_INST/INST_CCC_IP.CLK0_PAD	CCC-NE0 (1428, 302)	fixed_clk	HARDWIRED	1

Table 6-10. Columns in the CCC Input Connections Section

Column	Description
I/O Function	I/O connection details, such as the bank name, any hardwired GB or hardwired CCC connections, and any dedicated SERDES/DDR connections. For hardwired connections, the I/O function name contains the CCC location (CCC_NE0 in the figure above).
To (Pin Swapped for Back Annotation Only)	For hardwired connections, input pin of the CCC in the back annotated netlist.

### 6.9.6 Local Clock Nets to RGB Connections

The **Local Clock Nets to RGB Connections** section lists the clock nets from the local clock nets to RGB (Row globals). RGBs are situated on the vertical stripes of the global network architecture inside the FPGA fabric. The global signals from the GBs are routed to the RGBs. Each RGB is independent and can be driven by fabric routing in addition to being driven by GBs. This facilitates using RGBs to drive regional clocks spanning a small fabric area, such as the clock network for SERDES.

Figure 6-16. Local Clock Nets to RGB Connections Section

Local Clock Nets to RGB Connections

	From	From Location	Net Name	Fanout		RGB Location	Local Fanout
1	serdes_i/pcs_gl_2_pcs_0/rx_rst_n_i_0_Y	(216, 111)	serdes_i/pcs_gl_2_pcs_0/rx_rst_n_i_0_0	310	1	(364, 72)	15
					2	(364, 75)	44
					3	(364, 78)	37
					4	(364, 81)	19
					5	(364, 84)	28
					6	(364, 87)	20
					7	(364, 90)	25
					8	(364, 93)	25
					9	(364, 96)	36
					10	(364, 99)	19
					11	(364, 102)	14
					12	(364, 111)	17
					13	(364, 114)	11

Table 6-11. Columns in the Local Clock Nets to RGB Connections Sections

Column	Description
From	Driver routed to different RGBs, each with different local fanout.
From Location	X-Y coordinates of the driver of the net.
Fanout	Total fanout of the net and the local fanout column gives the fanout at the local RGB only.
RGB Location	X-Y coordinates on the chip.
RGB Fanout	Fanout at the local RGB.

### 6.9.7 Global Clock Nets to RGB Connections

The **Global Clock Nets to RGB Connections** section lists all nets from Globals (GBs) to Row Globals (RGBs).

Figure 6-17. Global Clock Nets to RGB Connections Section

### Global Clock Nets to RGB Connections

	From	From Location	Net Name	Fanout		RGB Location	Local Fanout
1	GBR[16]	(736, 154)	clk16_ibuf_RNI69L/U0_Y	5003	1	(1166, 114)	7
					2	(1166, 120)	53
					3	(1166, 123)	40
					4	(1166, 129)	39
					5	(1166, 132)	48
					6	(1166, 135)	50
					7	(1166, 138)	77
					8	(1166, 141)	100
					9	(1166, 144)	64
					10	(1166, 147)	53
					11	(1166, 150)	63
					12	(1166, 156)	39
					13	(1166, 159)	128
					14	(1166, 162)	146
					15	(1166, 165)	146
					16	(1166, 168)	130
					17	(1166, 171)	139
					18	(1166, 174)	146
					19	(1166, 177)	164
					20	(1166, 180)	20
					21	(1166, 183)	145
					22	(1166, 186)	127
					23	(1166, 189)	121
					24	(1166, 192)	127
					25	(1166, 195)	110
					26	(1166, 198)	94
					27	(1166, 201)	30

Table 6-12. Columns in the Global Clock Nets to RGB Connections Section

Column	Description
From	Hardwired to different RGBs each with different local fanout.
From Location	X-Y coordinates on the chip. The <b>Fanout</b> column gives the total fanout of the net.
Local Fanout	Fanout local to RGB.

## 6.10 Verify Post Layout Implementation

The following sections describe how to verify post-implementations of your design.

### 6.10.1 Generate Back Annotated Files

The first step when verifying the post-layout implementation generates two files:

- `*ba.sdf`
- `*ba.v/.vhd`

The `*ba.sdf` file is a delay file in Standard Delay Format (SDF). It is used for back annotation to the simulator.

The `*ba.v/.vhd` file is a post-layout flattened netlist used for back-annotated timing simulation. The file can contain low-level macros to improve design performance.

This step allows you to select **Export Enhanced min delays for best case**. Checking this option exports your enhanced min delays to include the best-case timing results in your Back Annotated File.

### 6.10.2 Simulate - Opens ModelSim ME

The back-annotation functions are used to extract timing delays from your post layout data. These extracted delays are placed into a file for use by your CAE package's timing simulator. The default simulator for Libero SoC is ModelSim ME. You can change your default simulator in your [Tool Profile](#).

To perform [pre-layout simulation](#), in the Design Flow Window, under **Verify Pre-Synthesized design**, double-click **Simulate**.

To perform timing simulation:

1. Back-annotate your design and create your testbench.
2. Right-click **Simulate** in the Design Flow window (**Implement Design > Verify Post-Synthesis Implementation > Simulate**) and choose **Organize Input Files > Organize Simulation Files**.
  - In the Organize Files for Source dialog box, all the stimulus files in the current project appear in the Source Files in the Project list box. Files already associated with the block appear in the Associated Source Files list box.
  - In most cases, you will have one testbench associated with your block. However, if you want simultaneous association of multiple testbench files for one simulation session, as in the case of PCI cores, add multiple files to the **Associated Source Files** list.
  - To add a testbench: Select the testbench you want to associate with the block in the **Source Files** in the **Project** list box and click **Add** to add it to the Associated Source Files list.
  - To remove or change the files in the **Associated Source Files** list box: select the files and click **Remove**.
  - To order testbenches: Use the up and down arrows to define the order you want the testbenches compiled. The top level-entity should be at the bottom of the list.
3. When you are satisfied with the **Associated Simulation Files** list, click **OK**.
4. To start ModelSim ME, right-click **Simulate** in the Design Hierarchy window and choose **Open Interactively**. ModelSim starts and compiles the appropriate source files. When the compilation completes, the simulator runs for 1 microsecond and the Wave window opens to display the simulation results.
5. In the Wave window, scroll to verify the logic works as intended. Use the cursor and zoom buttons to zoom in and out and measure timing delays. If you did not create a testbench with WaveFormer Pro, you may get error messages with the vsim command if the instance names of your testbench do not follow the same conventions as WaveFormer Pro. Ignore the error message and type the correct vsim command.
6. When you are done, from the **File** menu, choose **Quit**.

### 6.10.3 Verify Timing

Using the Verify Timing Configuration dialog box, you can configure the Verify Timing tool to generate a timing constraint coverage report along with detailed static timing analysis and violation reports based on different combinations of process speed, operating voltage, and temperature.

The following figures show an example of the Verify Timing Configuration dialog box.

Figure 6-18. Verify Timing Configuration Dialog Box – Report Tab

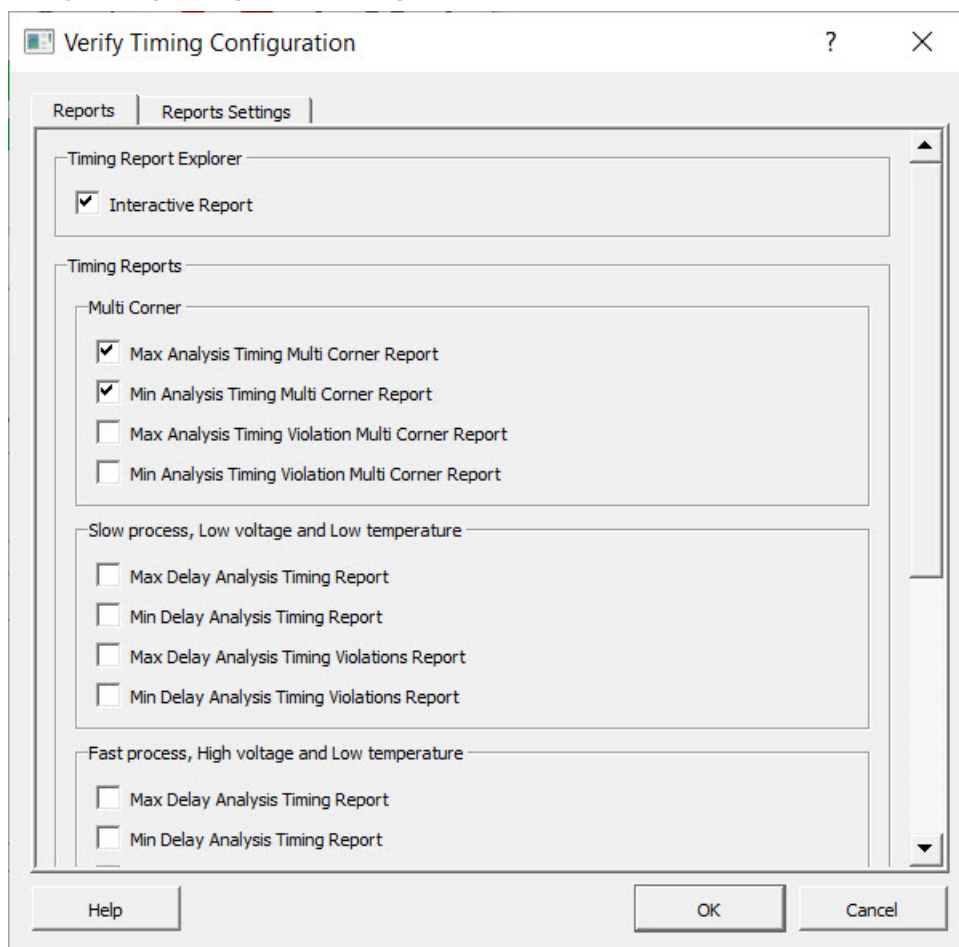
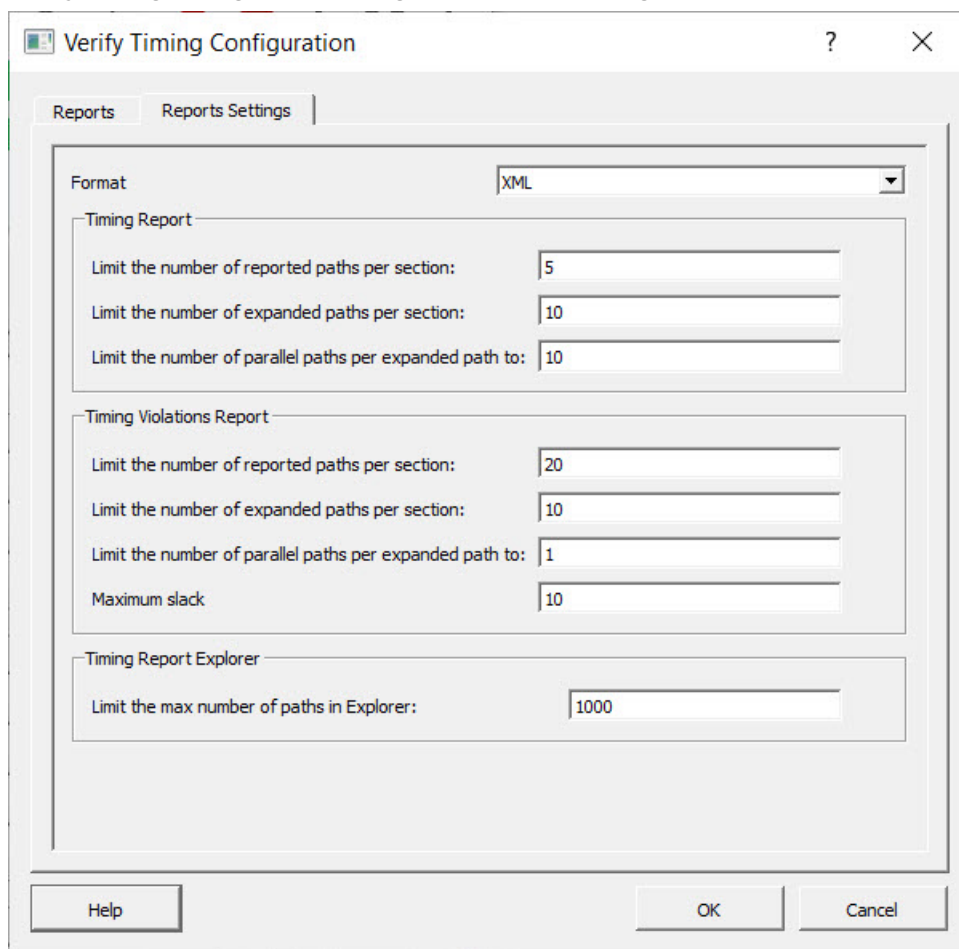


Figure 6-19. Verify Timing Configuration Dialog Box – Report Settings Tab



The reports can be generated in XML or text format. The following table describes the report settings.

**Note:** If any options are left blank in the **Report Settings** tab, a tool tip error icon appears as shown in the following figure.

Figure 6-20. Verify Timing Configuration Dialog Box - Tooltip Error

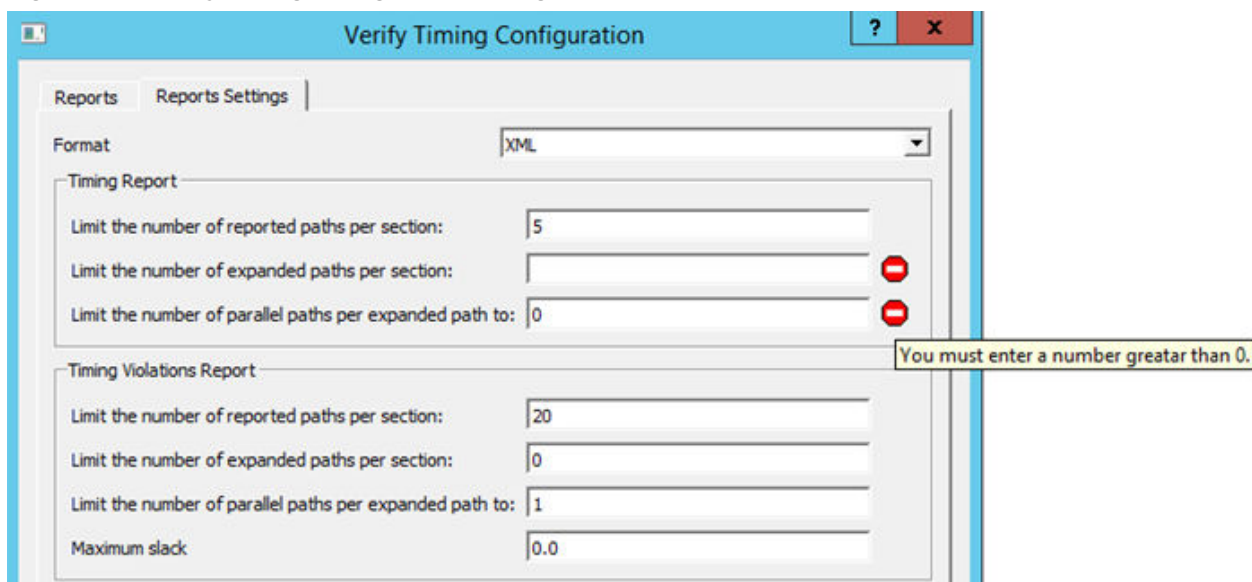
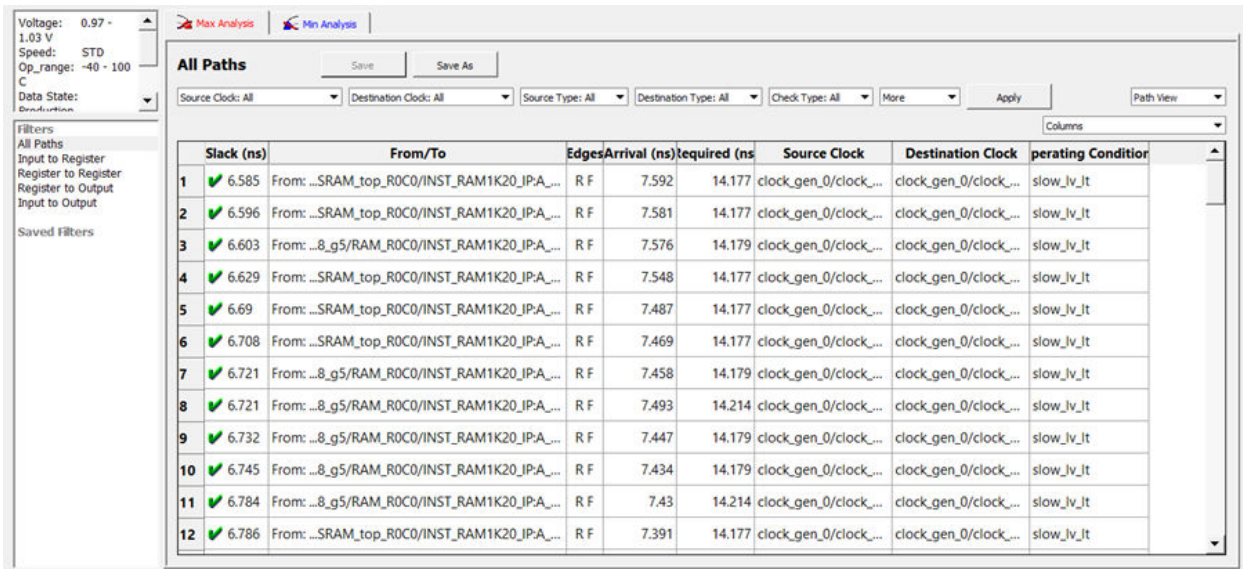


Table 6-13. Verify Timing Configuration Dialog Box Settings

Report	Setting	Description
Timing Report	Limit the number of reported paths per section	Number of reported paths under each section. Range: 1 - 20000
	Limit the number of expanded paths per section	Number of expanded paths under each section. Range: 1 - 20000
	Limit the number of parallel paths per expanded path to	Number of parallel paths for each expanded path. Range: 1 - 20000
Timing Violations Report	Limit the number of reported paths per section	Number of reported paths under each section. Range: 1 - 20000
	Limit the number of expanded paths per section	Number of expanded paths under each section. Range: 0 - 20000
	Limit the number of parallel paths per expanded path to	Number of parallel paths for each expanded path. Range: 1 - 20000
	Maximum Slack	Maximum slack threshold value in nanoseconds. Paths are filtered based on the slack threshold value in Timing Violation reports.
Timing Report Explorer	Limit the max number of paths in Explorer	Number of input paths in Timing Report Explorer. Range: 1 - 10000  Timing Report Explorer can also be opened from <b>Design Tab &gt; Timing Report Explorer</b> .  After generating a report, the Timing Report Explorer appears in Path View as shown in the following figure.

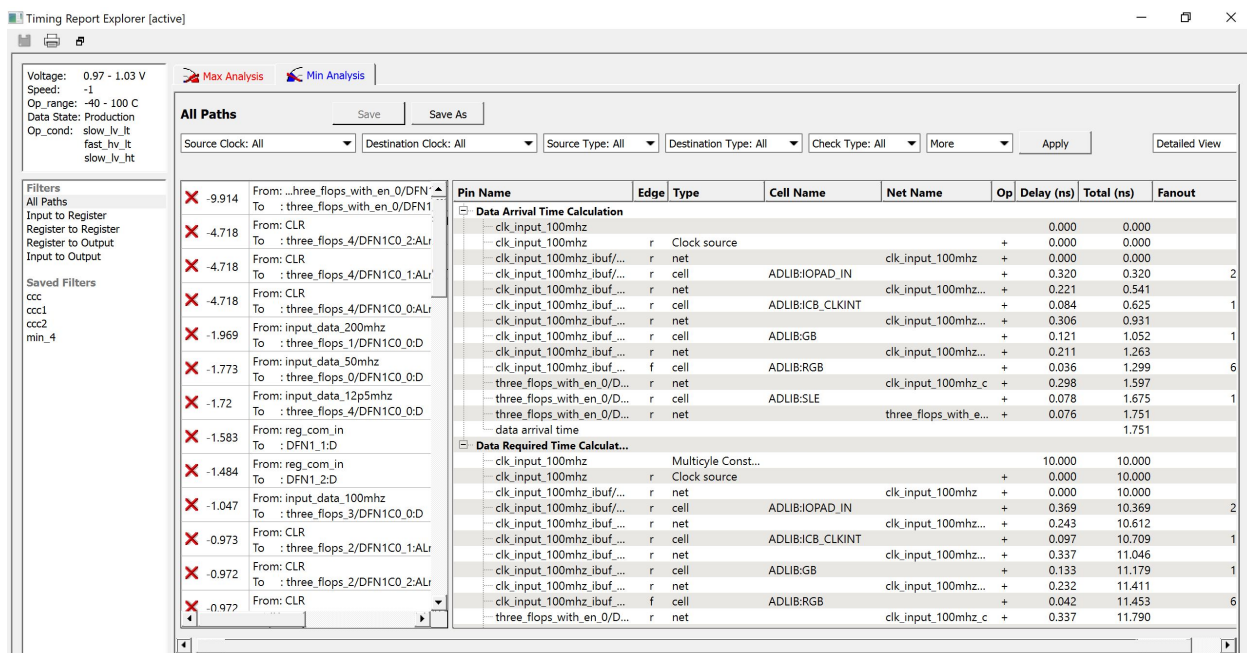


**Figure 6-21. Timing Report Explorer - Path View**



In the figure above, click any path and select the Detailed view from the settings drop down list to display a detailed view similar to the one in the following figure.

**Figure 6-22. Timing Report Explorer - Detailed View**



Observe the following guidelines:

- The following warning message appears if there are no paths for a selected filter: No paths were found to match your filter. Choose another filter or try modifying your search criteria.
- Cross Probing between Timing Report Explorer and Chip-Planner is supported for nets and cells.
- Mousing over the Source/Destination clock combo boxes displays tool tips similar to the following:



Figure 6-23. Example of Source Tool Tip

**All Paths** Save Save As

PF\_CCC\_C0\_0/PF\_CCC\_C0\_0 ▾ Destination Clock: All ▾ Source Type: All ▾

Source Clock: All  
PF\_CCC\_C0\_0/P...I inst 0/OUT0

	Slack (ns)	PF_CCC_C0_0/PF_CCC_C0_0/pll_inst_0/OUT0	Edges	A
1	✓ 17.733	From: ..._ROM_0/MACC_PHYS_0/INST_MACC_IP:CLK To : ...0/MACC_PHYS_0/INST_MACC_IP:CDIN[21]	R F	
2	✓ 17.739	From: ..._ROM_0/MACC_PHYS_0/INST_MACC_IP:CLK To : ...0/MACC_PHYS_0/INST_MACC_IP:CDIN[24]	R F	
3	✓ 17.742	From: ..._ROM_0/MACC_PHYS_0/INST_MACC_IP:CLK To : ...0/MACC_PHYS_0/INST_MACC_IP:CDIN[16]	R R	
4	✓ 17.751	From: ..._ROM_0/MACC_PHYS_0/INST_MACC_IP:CLK To : ...0/MACC_PHYS_0/INST_MACC_IP:CDIN[35]	R R	
5	✓ 17.758	From: ..._0/data_pipe_0/delayLine_CF2[1]:CLK To : ...Line seashift 0 0/R DATA 0 inst:SLn	R F	

Figure 6-24. Example of Destination Tool Tip

Max Analysis Min Analysis

**All Paths** Save Save As

Source Clock: All ▾ Destination Clock: All ▾ Source Type: All ▾ Destination Type: All ▾ Check Type: All ▾ More ▾ Apply Path View

Destination Clock: All  
PF\_CCC\_C0\_0/P...I inst 0/OUT0  
PF\_CCC\_C0\_0/P...I inst 0/OUT1

	Slack (ns)	From: ...three_flops_4/DFN1C0_2:ALn	Required (ns)	Source Clock	Destination Clock	Operating Conditions
1	✗ -9.914	From: ...three_flops_4/DFN1C0_2:ALn To : three_flops_4/DFN1C0_0:ALn	1.751	clk_input_100mhz	clk_input_100mhz	fast_hv_lt
2	✗ -4.718	From: CLR To : three_flops_4/DFN1C0_2:ALn	1.927	clk_input_100mhz	gen_clk_12p5mhz	slow_lv_ht
3	✗ -4.718	From: CLR To : three_flops_4/DFN1C0_0:ALn	1.927	clk_input_100mhz	gen_clk_12p5mhz	slow_lv_ht
4	✗ -4.718	From: CLR To : three_flops_4/DFN1C0_0:ALn	1.927	clk_input_100mhz	gen_clk_12p5mhz	slow_lv_ht
5	✗ -1.969	From: input_data_200mhz To : three_flops_0/DFN1C0_0:D	8.363	PF_CCC_C0_0/ PF_CCC_C0_0/pll_inst_0/...	PF_CCC_C0_0/ PF_CCC_C0_0/pll_inst_0/...	slow_lv_lt
6	✗ -1.773	From: input_data_50mhz To : three_flops_0/DFN1C0_0:D	8.558	PF_CCC_C0_0/ PF_CCC_C0_0/pll_inst_0/...	PF_CCC_C0_0/ PF_CCC_C0_0/pll_inst_0/...	slow_lv_lt
7	✗ -1.72	From: input_data_12p5mhz To : three_flops_0/DFN1C0_0:D	5.118	gen_clk_12p5mhz	gen_clk_12p5mhz	slow_lv_ht
8	✗ -1.583	From: reg_com_in To : DFN1_1:D	1.303	reg_com_clk_10mhz	reg_com_clk_10mhz	slow_lv_ht
9	✗ -1.484	From: reg_com_in To : DFN1_2:D	1.402	reg_com_clk_10mhz	reg_com_clk_10mhz	slow_lv_ht

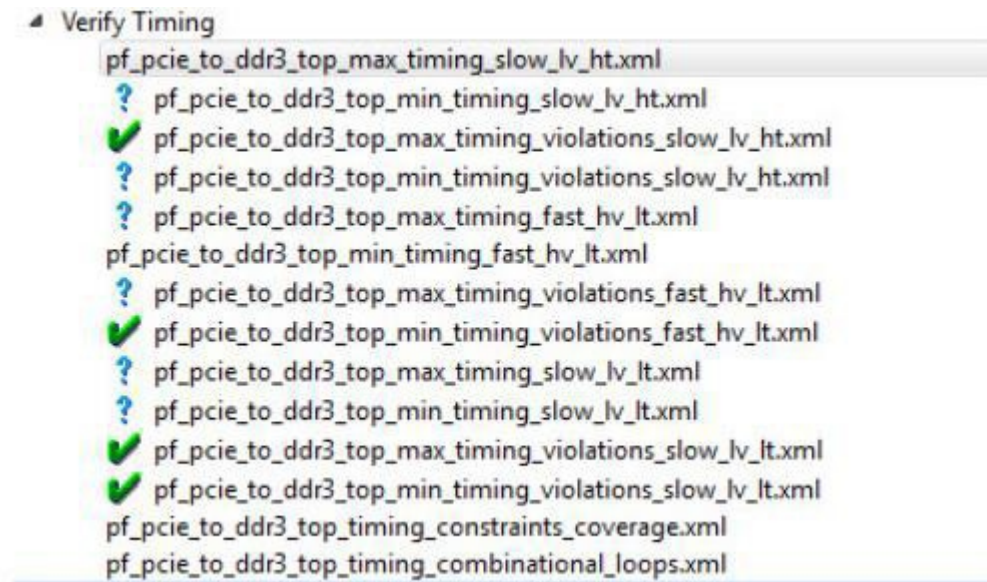
#### 6.10.4 Types of Timing Reports

From the **Design Flow window > Verify Timing**, you can generate the following reports. The following reports organize timing information by clock domain. Four types of timing reports are available. To configure which reports to generate, use the Verify Timing Configuration dialog box (**Design Flow > Verify Timing > Configure Options**).

**Table 6-14.**

Report	Description
Timing reports	<p>The following reports can be generated:</p> <ul style="list-style-type: none"> <li>• Max Delay Static Timing Analysis report based on Slow process, Low Voltage, and High Temperature operating conditions.</li> <li>• Min Delay Static Timing Analysis report based on Fast process, High Voltage, and Low Temperature operating conditions.</li> <li>• Max Delay Static Timing Analysis report based on Fast process, High Voltage, and Low Temperature operating conditions.</li> <li>• Min Delay Static Timing Analysis report based on Slow process, Low Voltage, and High Temperature operating conditions.</li> <li>• Max Delay Static Timing Analysis report based on Slow process, Low Voltage, and Low Temperature operating conditions.</li> <li>• Min Delay Static Timing Analysis report based on Slow process, Low Voltage, and Low Temperature operating conditions.</li> </ul>
Timing violations reports	<p>Organizes timing information by clock domain. Four types of timing violations reports are available. You can configure which reports to generate using the Verify Timing Configuration dialog box. The following reports can be generated:</p> <ul style="list-style-type: none"> <li>• Max Delay Analysis Timing Violation report based on Slow process, Low Voltage, and High Temperature operating conditions.</li> <li>• Min Delay Analysis Timing Violation report based on Fast process, High Voltage, and Low Temperature operating conditions.</li> <li>• Max Delay Analysis Timing Violation report based on Fast process, High Voltage, and Low Temperature operating conditions.</li> <li>• Min Delay Analysis Timing Violation report based on Slow process, Low Voltage, and High Temperature operating conditions.</li> <li>• Max Delay Analysis Timing Violation report based on Slow process, Low Voltage, and Low Temperature operating conditions.</li> <li>• Min Delay Analysis Timing Violation report based on Slow process, Low Voltage, and Low Temperature operating conditions.</li> </ul>
Constraints coverage report	<p>Displays the overall coverage of the timing constraints set on the current design.</p> <p>&lt;root&gt;_timing_constraints_coverage.xml (generated by default)</p>

Figure 6-25. Reports Example



The following table describes the icons associated with reports.

Table 6-15. Report Listing Icon Legend

Icon	Definition
	Timing requirement met for this report
	Timing requirement not met (violations) for this report
	Timing report available for generation but has not been selected/configured for generation

## 6.10.5 SmartTime

SmartTime is the Libero SoC gate-level static timing analysis tool. With SmartTime, you can perform complete timing analysis of your design to ensure that you meet all timing constraints and that your design operates at the desired speed, with the right amount of margin across all operating conditions.

For information about creating and editing timing constraints, see the [Timing Constraints Editor User Guide](#).

### 6.10.5.1 Static Timing Analysis (STA)

Static timing analysis (STA) identifies timing violations in your design and ensures it meets your timing requirements. You can communicate timing requirements and timing exceptions to the system by setting timing constraints. A static timing analysis tool then checks and reports setup and hold violations as well as violations on specific path requirements.

STA is well-suited for traditional synchronous designs. The main advantage of STA is that, unlike dynamic simulation, it does not require input vectors. It covers all possible paths in the design and does all the above with relatively low run-time requirements.

STA tools report all possible paths, including false paths. False paths are timing paths in the design that do not propagate a signal. Because STA tools do not automatically detect false paths in their algorithms, you need to identify false paths as false path constraints to the STA tool and exclude them from timing considerations to obtain a true and useful timing analysis.

#### Timing Constraints

SmartTime supports a range of timing constraints to provide useful analysis and efficient timing-driven layout.

### **Timing Analysis**

SmartTime provides analysis types that allow you to:

- Find the minimum clock period/highest frequency that does not result in a timing violations
- Identify paths with timing violations
- Analyze delays of paths that have no timing constraints
- Perform inter-clock domain timing verification
- Perform maximum and minimum delay analysis for setup and hold checks

To improve the accuracy of the results, SmartTime evaluates clock skew during timing analysis by computing individual clock insertion delays for each register.

SmartTime checks the timing requirements for violations, such as multicycle or false paths, while evaluating timing exceptions.

#### **6.10.5.2 SmartTime and Place and Route**

Timing constraints impact analysis the same way they affect place and route. As a result, adding and editing timing constraints in SmartTime is the best way to achieve optimum performance.

#### **6.10.5.3 SmartTime and Timing Reports**

The following report files can be generated from **SmartTime > Tools > Reports**:

- Timing Report (for both Max and Min Delay Analysis)
- Timing Violations Report (for both Max and Min Delay Analysis)
- Bottleneck Report
- Constraints Coverage Report
- Combinational Loop Report

#### **6.10.5.4 SmartTime and Cross-Probing into Chip Planner**

From SmartTime, you can select a design object and cross-probe the same design object in Chip Planner. Design objects that can be cross-probed from SmartTime to Chip Planner include:

- Ports
- Macros
- Timing Paths

For more information, see the *SmartTime User's Guide* (**Libero SoC > Help > Reference Manual > SmartTime User's Guide**).

#### **6.10.5.5 SmartTime and Cross-Probing into Constraint Editor**

From SmartTime, you can cross-probe into the Constraint Editor. Select a Timing Path in SmartTime's Analysis View and add one of the following Timing Exception Constraints:

- False Path
- Multicycle Path
- Max Delay
- Min Delay

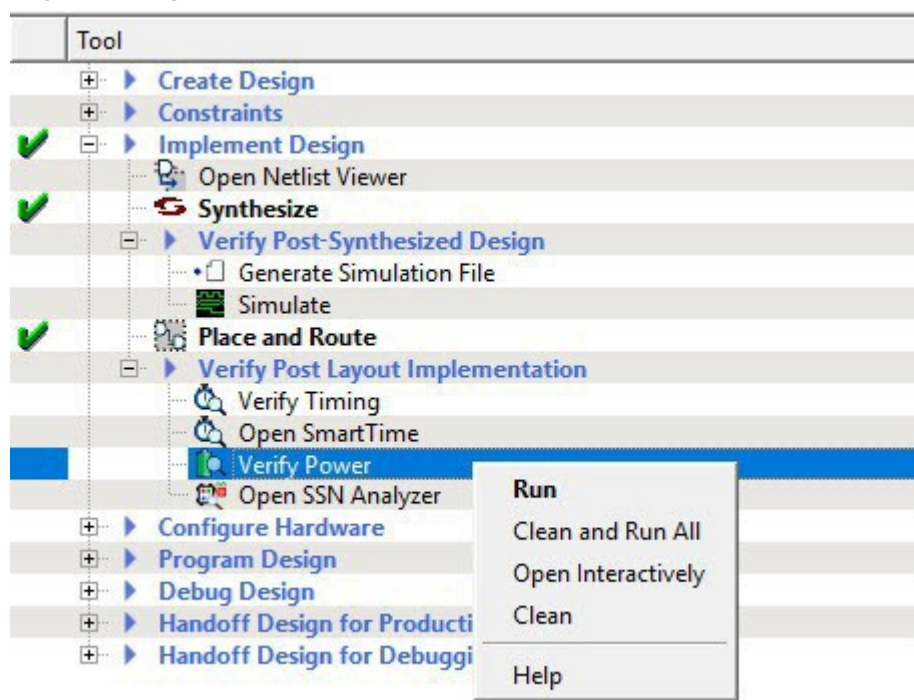
The Constraint Editor reflects the newly added timing exception constraint.

For more information, see the [SmartTime Static Timing Analyzer User Guide](#).

#### **6.10.6 Verify Power**


In the Design Flow window, right-click **Verify Power** to display following menu options.

Figure 6-26. Verify Power Right-click Menu



The following table describes the options.

Table 6-16.

Option	Description
Run	Runs the default power analysis and produces a power report. This option is functionally equivalent to double-clicking.
Clean and Run All	functionally equivalent to the sequence of Clean and Run (described below).
Open Interactively	Displays the SmartPower for Libero SoC tool (see below)
Clean	Clears the history of any previous default power analysis, including deletion of any reports. The flow task completion icon  will also be cleared.
Configuration Options	If there are options to configure, a dialog box displays technology-specific choices.
View Report	If a report is available, clicking this option adds the <b>Report</b> tab to the Libero SoC GUI window and <b>Power Report</b> will be selected.


### 6.10.7 Verify Power sub-commands

**Run** - Runs the default power analysis and produces a power report. This is also the behavior of a double-click to Verify Power.

**Clean and Run All** - Identical to the sequence of commands "Clean" (see below) and "Run"

**Open interactively** - Brings up the SmartPower for Libero SoC tool (see below)



**Clean** - Clears the history of any previous default power analysis, including deletion of any reports. The flow task completion icon  will also be cleared.

**Configure Options ...** - This sub-command is only available if there are options to configure, in which case a dialog box will pop-up presenting the user with technology-specific choices.

**View Report** - This sub-command is only available and visible if a report is available. When **View Report** is invoked, the Report tab will be added to the Libero SoC GUI window, and the Power Report will be selected and made visible.

### 6.10.8 SmartPower

SmartPower is the Microchip SoC state-of-the-art power analysis tool. SmartPower allows you to visualize power consumption and potential power consumption problems within your design globally and in-depth, so you can make adjustments to reduce power.

SmartPower provides a detailed and accurate way to analyze designs for Microchip SoC FPGAs. From a top-level summaries, you can drill down specific functions within the design, such as gates, nets, I/Os, memories, clock domains, blocks, and power supply rails.

You can analyze the hierarchy of block instances and specific instances within a hierarchy. Each analysis can be viewed in different ways to show the respective power consumption of the component pieces.

SmartPower allows you to analyze power by functional modes, such as Active, Flash\*Freeze, Shutdown, Sleep, or Static, depending on the specific FPGA family used. You can also create custom modes created in the design. Custom modes can also be used for testing "what if" potential operating modes.

SmartPower allows you to create test scenario profiles. Using a profile, you can create sets of operational modes to understand the average power consumed by a combination of functional modes, such as a combination of Active, Sleep, and Flash\*Freeze modes used over time in an application.

SmartPower generates detailed hierarchical reports of the power consumption of a design for easy evaluation. This allows you to find power consumption sources and take appropriate action to reduce power.

SmartPower supports the Value-Change Dump (VCD) file format, as specified in the IEEE 1364 standard, generated by the simulation runs. Support for this format allows you to generate switching activity information from ModelSim or other simulators, and then use the switching activity-over-time results to evaluate average and peak power consumption for your design.

For more information, see the [SmartPower User Guide](#).

### 6.10.9 Simultaneous Switching Noise

Simultaneous Switching Noise (SSN) is the Libero SoC voltage noise analysis tool. It provides a detailed analysis of the noise margin about each I/O pin in the design, based on the pin information and other active pins placed in the same I/O bank of the design. The tool computes the noise margin based on I/O standards, drive strength, and pin placement. The SSN Analyzer helps you achieve the desired voltage noise margin, resulting in improved signal integrity.

To open the SSN Analyzer, right-click **SSN Analyzer** in the Design Flow window and select **Open Interactively**.

### 6.10.10 Supported Dies and Packages

The following shows supported dies and packages. Dies and packages for which characterization data is unavailable are not supported.

**Note:** In the following table, 1 ns pulse width is supported for MPF300XT/FCG1152 only.

**Table 6-17. Supported Dies and Packages**

Family	Die	Package
PolarFire	MPF300XT	FCG1152
—	MPF100T	FCG484
—	MPF200T	FCG484

.....continued

Family	Die	Package
—	MPF300T	FCG484/FCG1152
—	MPF500T	FCG1152

### 6.10.11 Supported I/O Standard

The SSN Analyzer supports the following I/O standards:

- LVCMOS 3.3V
- LVCMOS 2.5V
- LVCMOS 1.8V
- LVCMOS 1.5V
- LVCMOS 1.2V
- LVTTTL

### 6.10.12 Supported I/O Types

The SSN Analyzer supports single-end I/Os only. Differential I/Os are not supported.

### 6.10.13 SSN Analyzer Tabs

The SSN Analyzer has three tabs:

- Noise Report
- Excluded IOs
- Summary

#### 6.10.13.1 Noise Report Tab

The **Noise Report** tab displays by default when the SSN Analyzer opens. This tab lists all of the design Output and Inout ports. Input I/Os are not supported.

Figure 6-27. SN Analyzer – Noise Report Tab

Bank Name/ Pin Number	Port Name	Instance Name	IO Standard	Drive Strength (mA)	Static	Don't Care	Noise Margin (%)	Within Guideline
Bank0 (1.5v)								
A16	wr	wr_obuf/U0/U_IOPAD	LVCMOS33	4	<input checked="" type="checkbox"/>	<input type="checkbox"/>	99.01	Yes
A15	datao[28]	datao_obuf[28]/U0/U_IOPAD	LVCMOS33	4	<input type="checkbox"/>	<input type="checkbox"/>	99.01	Yes
Bank1 (3.3v)								
J23	datao[29]	datao_obuf[29]/U0/U_IOPAD	LVTTTL	4	<input type="checkbox"/>	<input type="checkbox"/>	98.38	Yes
J29	datao[25]	datao_obuf[25]/U0/U_IOPAD	LVCMOS33	4	<input type="checkbox"/>	<input type="checkbox"/>	98.38	Yes
Bank2 (2.5v)								
M30	datao[4]	datao_obuf[4]/U0/U_IOPAD	LVCMOS25	12	<input type="checkbox"/>	<input type="checkbox"/>	-11.66	No
N26	address[7]	address_1_obuf[7]/U0/U_IOPAD	LVCMOS25	16	<input type="checkbox"/>	<input type="checkbox"/>	-69.11	No
M28	address[4]	address_1_obuf[4]/U0/U_IOPAD	LVCMOS25	16	<input type="checkbox"/>	<input type="checkbox"/>	-50.88	No
M30	address[3]	address_1_obuf[3]/U0/U_IOPAD	LVCMOS25	16	<input type="checkbox"/>	<input type="checkbox"/>	-11.66	No
N27	address[2]	address_1_obuf[2]/U0/U_IOPAD	LVCMOS25	16	<input type="checkbox"/>	<input type="checkbox"/>	-25.04	No
M27	address[25]	address_1_obuf[25]/U0/U_IOPAD	LVCMOS25	16	<input type="checkbox"/>	<input type="checkbox"/>	-74.95	No
L28	address[24]	address_1_obuf[24]/U0/U_IOPAD	LVCMOS25	16	<input type="checkbox"/>	<input type="checkbox"/>	-74.21	No
P23	address[23]	address_1_obuf[23]/U0/U_IOPAD	LVCMOS25	16	<input type="checkbox"/>	<input type="checkbox"/>	-72.10	No
M25	address[22]	address_1_obuf[22]/U0/U_IOPAD	LVCMOS25	16	<input type="checkbox"/>	<input type="checkbox"/>	-32.03	No
N25	address[21]	address_1_obuf[21]/U0/U_IOPAD	LVCMOS25	16	<input type="checkbox"/>	<input type="checkbox"/>	-68.80	No
K28	address[20]	address_1_obuf[20]/U0/U_IOPAD	LVCMOS25	16	<input type="checkbox"/>	<input type="checkbox"/>	-73.62	No
N24	address[1]	address_1_obuf[1]/U0/U_IOPAD	LVCMOS25	16	<input type="checkbox"/>	<input type="checkbox"/>	-49.89	No
P24	address[19]	address_1_obuf[19]/U0/U_IOPAD	LVCMOS25	16	<input type="checkbox"/>	<input type="checkbox"/>	-59.92	No
L29	address[15]	address_1_obuf[15]/U0/U_IOPAD	LVCMOS25	16	<input type="checkbox"/>	<input type="checkbox"/>	-76.35	No
J27	address[14]	address_1_obuf[14]/U0/U_IOPAD	LVCMOS25	16	<input type="checkbox"/>	<input type="checkbox"/>	-62.78	No
L30	address[13]	address_1_obuf[13]/U0/U_IOPAD	LVCMOS25	16	<input type="checkbox"/>	<input type="checkbox"/>	-75.03	No
P30	address[12]	address_1_obuf[12]/U0/U_IOPAD	LVCMOS25	16	<input type="checkbox"/>	<input type="checkbox"/>	100.00	Yes
Bank3 (2.5v)								
A829	mio	mio_obuf/U0/U_IOPAD	LVCMOS25	6	<input type="checkbox"/>	<input type="checkbox"/>	70.52	Yes
W29	datao[20]	datao_obuf[20]/U0/U_IOPAD	LVCMOS25	12	<input type="checkbox"/>	<input type="checkbox"/>	74.07	Yes
U25	datao[15]	datao_obuf[15]/U0/U_IOPAD	LVCMOS25	12	<input type="checkbox"/>	<input type="checkbox"/>	68.75	Yes
U23	datao[14]	datao_obuf[14]/U0/U_IOPAD	LVCMOS25	12	<input type="checkbox"/>	<input type="checkbox"/>	74.30	Yes

The following table describes the columns in the tab.

**Table 6-18. Columns in the Noise Report Tab**

Column	Description
Bank Name/Pin Number	Shows the bank number and package pin number of the port.
Port Name	Shows the name of the port.
Instance name	Shows the instance name of the port.
I/O Standard	Shows the I/O standards supported by SSN Analyzer. Supported standards are: <ul style="list-style-type: none"> <li>• LVCMOS 3.3V</li> <li>• LVCMOS 2.5V</li> <li>• LVCMOS 1.8V</li> <li>• LVCMOS 1.5V</li> <li>• LVCMOS 1.2V</li> <li>• LVTTTL</li> </ul>
Drive Strength (mA)	Drive Strength selections are available from 2 to 12.
Static	Checked: I/O is considered neither as an Aggressor nor as a Victim. It is excluded from SSN Analysis.
Don't Care	Checked: I/O is excluded from consideration as a Victim for Noise Margin computation. This is considered as an Aggressor for Noise Margin computation of other I/Os. <b>Note:</b> Static and Don't Care are mutually exclusive.
Noise Margin (%)	Noise margin number computed by the SSN Analyzer. A red negative number indicates that it is outside the guideline of SSN analysis.
Within Guideline	Yes (Positive Noise Margin) or No (Negative Noise Margin). The Yes (within guideline) or No (outside guideline) guideline is different for different I/O standards: <ul style="list-style-type: none"> <li>• LVTTTL/LVCMOS (3.3V): Yes (within guideline) is defined as follows: <ul style="list-style-type: none"> <li>– A ground bounce voltage less than or equal to 1.25V and a pulse width of less than or equal to 1 ns.</li> <li>– A VDD dip voltage greater than or equal to VIHmin and a pulse width of less than or equal to 1 ns.</li> </ul> </li> <li>• All other LVCMOS standards (2.5V, 1.8V, 1.5V, 1.2V): A Yes (within guideline) is defined as follows: <ul style="list-style-type: none"> <li>– A ground bounce voltage less than or equal to VILmax for ground bounce and a pulse width of less than or equal to 1 ns.</li> <li>– A VDD dip voltage greater than or equal to VIHmin and a pulse width of less than or equal to 1 ns.</li> </ul> </li> <li>• Noise margin violating the criteria for “Yes” is considered to fall outside the specified guidelines, and is reported as a “No.”</li> </ul>



The following table describes the menu items available when you right-click an I/O in the **Noise Report** tab. You can select multiple I/Os and then right-click to apply the menu items to all selected I/Os.

**Table 6-19. Noise Report Right-Click Menu**

Menu	Description
Show in IO Editor/Chip Planner	Allows you to cross-probe or reconfigure the selected I/Os in I/O Editor or Chip Planner. <b>Note:</b> This menu item is active when the <b>I/O Editor</b> is open.
Mark Selected Static	Marks the selected I/Os as static (excluded from Noise Analysis).
Unmark Selected Static	Unmarks the selected I/Os as static (included for Noise Analysis).
Mark Selected Don't Care	Marks the selected I/O as Don't Care (not to be considered as Victim).
Unmark Selected Don't Care	Unmarks the selected I/Os as Don't Care (to be considered as Victim).
Copy Selection	Copies the selected I/Os to the Clipboard for pasting into other applications.
Print Selection	Copies the selected I/Os and sends to the printer.
Sort by Package Die Pad Number	Sorts the pin number by the order of the I/O pad number. Use this option to find a pin and its neighboring pins. All used pins are arranged in order of geographical proximity.
Search and Filter	Filtering is available for Port Names. For example, if you enter the search pattern "DATA*" in the Port Name field and click <b>Search</b> , the list is populated with all I/O names beginning with DATA. Names that do not begin with DATA are excluded from the list. Filtering allows you to focus on I/Os in which you are interested for SSN analysis.
Pulse Width	<p>Settling time of the signal bounce. This is a threshold value that the signal bounce must exceed before the signal bounce is recognized for SSN calculation. Choices are:</p> <ul style="list-style-type: none"> <li>0ns: any signal bounce with a pulse width above 0ns is recognized for SSN calculation.</li> <li>1ns: only signal bounces with a pulse width at or above 1ns are recognized for SSN calculation.</li> </ul> <p>Changing this selection discards changes made for the current pulse width selection and triggers a re-analysis based on the new pulse width.</p> <p><b>Note:</b> 1 ns pulse width is supported for the MPF300XT/FCG1152 die/package only.</p>

.....continued	
Menu	Description
Run Analysis	<p>not active when SSN first opens. It is activated only when you have made changes in the Noise Report. These changes may include one or more of the following:</p> <ul style="list-style-type: none"> <li>• Checking/unchecking the <b>Don't Care</b> check box for one or more I/Os.</li> <li>• Checking/unchecking the <b>Static</b> check box for one or more I/Os.</li> </ul> <p>When you have made your changes, click <b>Run Analysis</b> to have SSN recompute the Noise Margin number.</p>
Save Report	<p>Click to save the Noise Report in one of the following formats:</p> <ul style="list-style-type: none"> <li>• Text: Text file with *.txt file extension.</li> <li>• CSV: Spreadsheet file with *.csv file extension.</li> <li>• XML: XML file with *.xml file extension.</li> </ul>

### 6.10.13.2 Excluded I/Os Tab

The **Excluded I/O** tab shows all I/Os excluded from Noise Analysis. Excluded I/Os include:

- I/Os on unsupported I/O standards
- I/Os marked as **Static** in the **Noise Analysis** tab
- JTAG I/Os for which noise analysis is irrelevant

Figure 6-28. SSN Analyzer – Excluded I/Os Tab

Bank Name/Pin Number	Port Name	Instance Name	I/O Standard	Comment
Bank8 (2.5v)				
Bank7 (2.5v)				
Bank6 (1.2v)				
Bank5 (1.8v)				
AK19	ast2	ast2_obuf/U0/U_IOPAD	LPDDR1	IO Standard is not supported
AE27	ast1	ast1_obuf/U0/U_IOPAD	LPDDR1	IO Standard is not supported
AE18	datao[30]	datao_obuf[30]/U0/U_IOPAD	SSTL18I	IO Standard is not supported
Bank3 (2.5v)				
Bank2 (2.5v)				
Bank1 (3.3v)				
Bank0 (1.5v)				
D15	datao[10]	datao_obuf[10]/U0/U_IOPAD	HSTLI	IO Standard is not supported
D14	datao[11]	datao_obuf[11]/U0/U_IOPAD	HSTLI	IO Standard is not supported

The following table describes the columns in the tab.

**Note:** You can right-click an I/O previously marked as static in the **Excluded I/Os** list and select **Unmarked Selected Static** to include it in Noise Report Analysis.

Table 6-20. Columns in the Excluded I/Os Tab

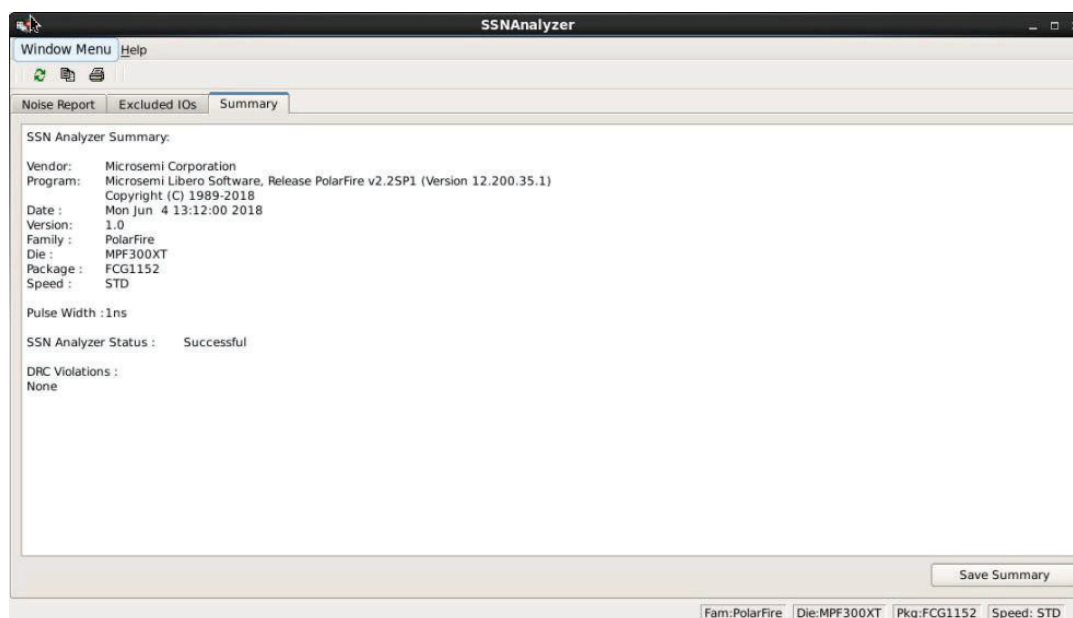
Bank Name/Pin Number	Shows the bank number and package pin number of the port.
----------------------	---

.....continued	
Port Name	Shows the name of the port.
Instance name	Shows the instance name of the port.
I/O Standard	Shows the I/O standards supported by SSN Analyzer. Supported standards are: <ul style="list-style-type: none"> <li>• LVCMOS 3.3V</li> <li>• LVCMOS 2.5V</li> <li>• LVCMOS 1.8V</li> <li>• LVCMOS 1.5V</li> <li>• LVCMOS 1.2V</li> <li>• LVTTTL</li> </ul>
Comment	Reason for exclusion (for example, unsupported I/O Standards or Marked as Static I/Os).

### 6.10.13.3 Summary Tab

The **Summary** tab summarizes the SSN Analyzer. Click **Save Summary** to save the summary in text, CSV, or XML format.

Figure 6-29. SSN Analyzer - Summary Tab



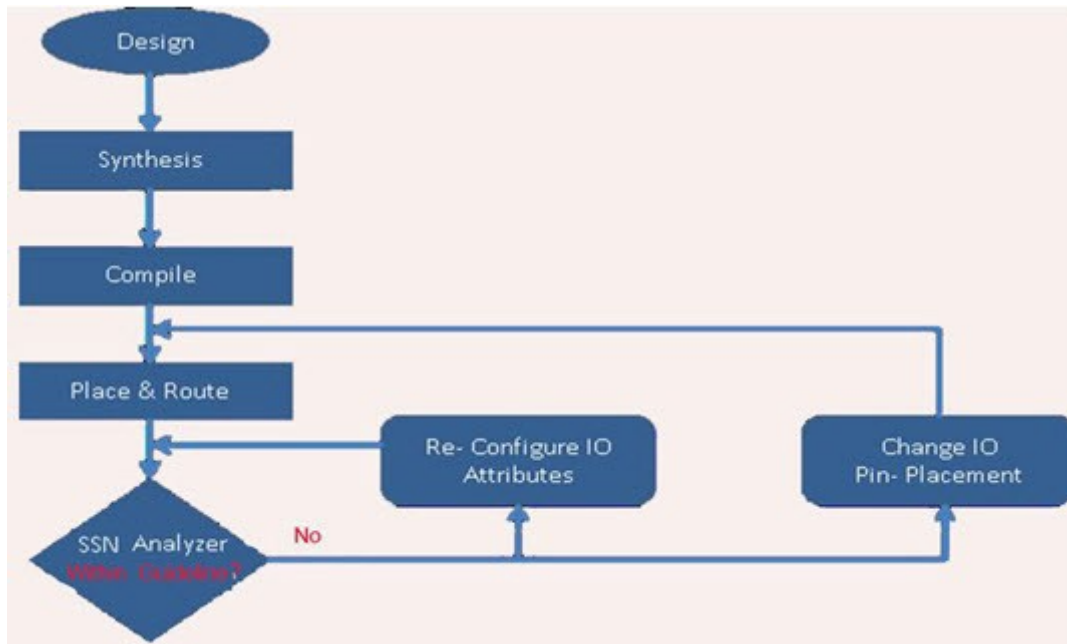
### 6.10.14 SSN Noise Analyzer Reports Failure Procedure

If the SSN Noise Analyzer reports poor noise margin or failure, perform the following procedure to improve the noise margin:

1. Change the I/O standard to one that has a lower noise impact for the failing I/O Bank.
2. Select the lower Drive-Strength to reduce the noise. Open the I/O Advisor to see the power/timing impact of the specific I/O cell.
3. Rerun the SSN Analyzer to see if the noise margin of the I/O Cell improves. In this scenario, Place and Route information remains intact.
4. If the improvement is not significant, open the Pin Attributes Editor and change the placement of the pin within the I/O bank to a location farther away from the noisy pins.
5. Spread the failing pins across multiple I/O banks. This reduces the number of aggressive outputs on the power system of the I/O bank.

6. Rerun Place and Route and rerun SSN Analyzer to check the Noise Report.

Figure 6-30. SSN Analyzer in the Design Flow



## 7. Configure Hardware

The following sections provide information about configuring the hardware for your designs.

### 7.1 Programming Connectivity and Interface

The Programming Connectivity and Interface window shows the physical chain from TDI to TDO or SPI Slave configuration. To open this window, expand **Configure Hardware** in the Libero SoC Design Flow window, and then double-click **Programming Connectivity and Interface**.

The Programming Connectivity and Interface view provides options for performing the following actions on non-target devices.

**Table 7-1. Programming Connectivity and Interface Options**

Option	Description
Select Programming Interface	Select JTAG or SPI Slave mode. SPI Slave mode is supported by FlashPro6 for PolarFire devices. JTAG is the default interface.
Construct Chain Automatically	Constructs the physical chain automatically.
Add Microsemi Device	Adds a Microsemi device to the chain.
Add Non-Microsemi Device	Adds a non-Microsemi device to the chain.
Add Microsemi Devices From Files	Adds a Microsemi device from a programming file.
Delete Selected Devices	Deletes selected devices in the grid.
Scan and Check Chain	Scans the physical chain connected to the programmer and check if it matches the chain constructed in the grid.
Zoom In	Zooms into the grid.
Zoom Out	Zooms out of the grid.

#### 7.1.1 Hover Information

If you hover your pointer over a device in the grid, the device tooltip shows the following device information.

**Table 7-2.**

Tooltip	Description
Name	User-specified device name. If you have two or more identical devices in your chain, use this field to give them unique names.
Device	Name of the device.
File	Path to the programming file.
Programming action	When a programming file is loaded, select a programming action for any device that is not a Libero design device.
IR	Length of the device instruction.
TCK	Maximum clock frequency, in Hz, to program a specific device; Libero uses this information to ensure the programmer operates at a frequency lower than the slowest device in the chain.

### 7.1.2 Device Chain Details

The device within the chain has the following details.

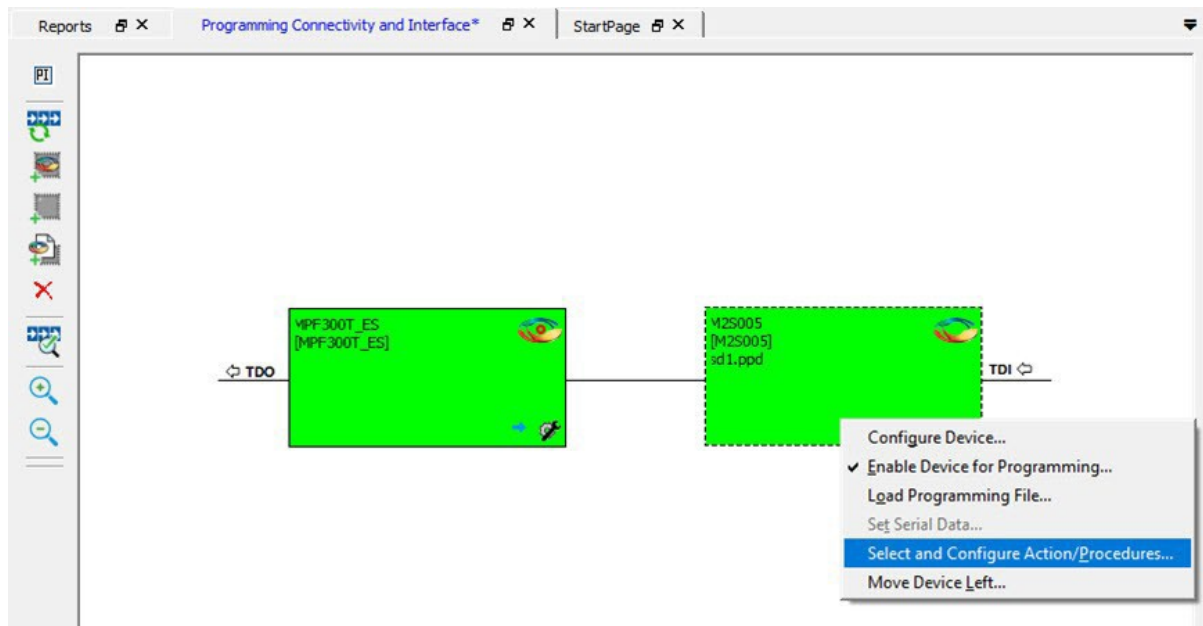
**Table 7-3. Device Chain Details**

Detail	Description
Libero design device	Red circle within Microsemi logo. Libero design device cannot be disabled.
Left/right arrow	Moves the device left or right according to the physical chain.
Enable device	Enables the device for programming. <ul style="list-style-type: none"> <li>Green: device is enabled.</li> <li>Gray: device is disabled.</li> </ul>
Name	Name of your specified device.
File	Path to the programming file.

### 7.1.3 Right-Click Options

The following figure shows the options that appear when you right-click your design.

**Figure 7-1. Right-click Properties**



Right-clicking a device displays the following options.

**Table 7-4. Right-Click Options**

Option	Description
Set as Libero Design Device	Sets the Libero design device when there are multiple identical Libero design devices in the chain.
Configure Device	Reconfigures the device. For a Libero SoC target device, the dialog box appears, but only the device name can be edited.

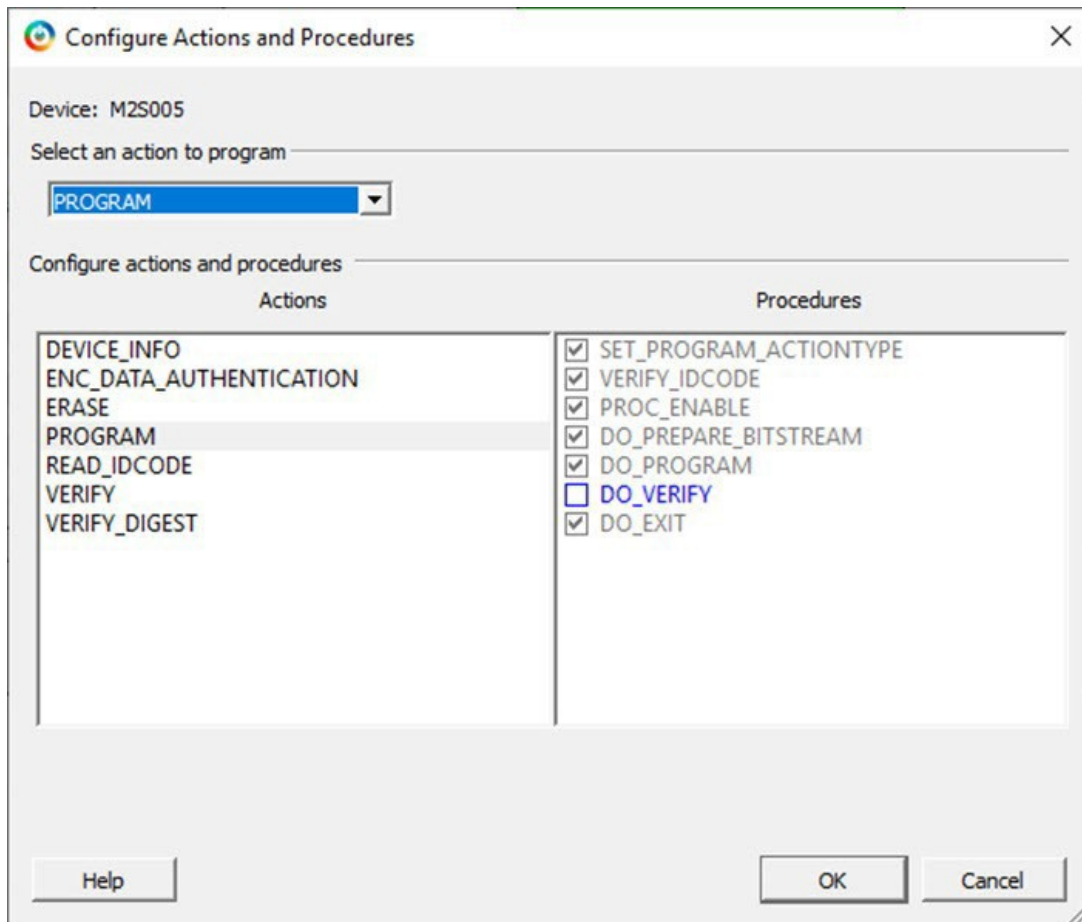
.....continued	
Option	Description
Enable Device for Programming	Enables the device for programming. <ul style="list-style-type: none"> <li>• Green: enabled devices.</li> <li>• Gray: disabled devices.</li> </ul>
Load Programming File	Loads the programming file for the selected device. This option is not supported for Libero SoC target design devices.
Set Serial Data	Displays the Serial Settings dialog box, where you can set your serialization data.
Select and Configure Action/Procedure	This option applies to devices other than the Libero SoC target design device. Choices are: <ul style="list-style-type: none"> <li>• Select an action to program: Selected action is programmed in the Libero environment and saved to an exported FlashPro Express job.</li> <li>• Configure actions and procedures: <ul style="list-style-type: none"> <li>– Actions: List of programming actions for your device.</li> <li>– Procedures: Advanced option that allows you to customize the list of recommended and optional procedures for an action.</li> </ul> </li> </ul>
Move Device Left/Right	Moves the device in the chain to left or right.

## 7.2 Select and Configure Actions and Procedures

The Libero SoC allows you to configure actions and procedures for devices other than the Libero SoC target design device and select an action to program in the Libero environment. You perform these actions using the Configure Actions and Procedures dialog box.

To display the Configure Actions and Procedures dialog box, in the Programming and connectivity interface, right-click a non-target device and choose **Select and Configure Actions/Procedures**.

Figure 7-2. Configure Actions and Procedures Dialog Box



The following table describes the elements in the Configure Actions and Procedures dialog box.

Table 7-5. Elements in the Configure Actions and Procedures Dialog Box

Element	Description
Select an action to program	The selected action will be programmed in the Libero environment.
Configure actions and procedures	<p>Configure actions and procedures:</p> <ul style="list-style-type: none"> <li>• Actions: List of programming actions for your device.</li> <li>• Procedures: Advanced option that allows you to customize the list of recommended and optional procedures for an action.</li> </ul> <p><b>Note:</b> You cannot select an action or configure actions in the chain view for the target device. To configure actions for Libero target devices, use the Configure Actions and Procedures tool.</p>

## 7.3 Programmer Settings

For the JTAG interface, you can set specific voltage and force TCK frequency values for your programmer. For the SPI Slave interface, you can set specific voltage and force SCK frequency values for your programmer. You perform these actions using the view the Programmer Settings dialog box.



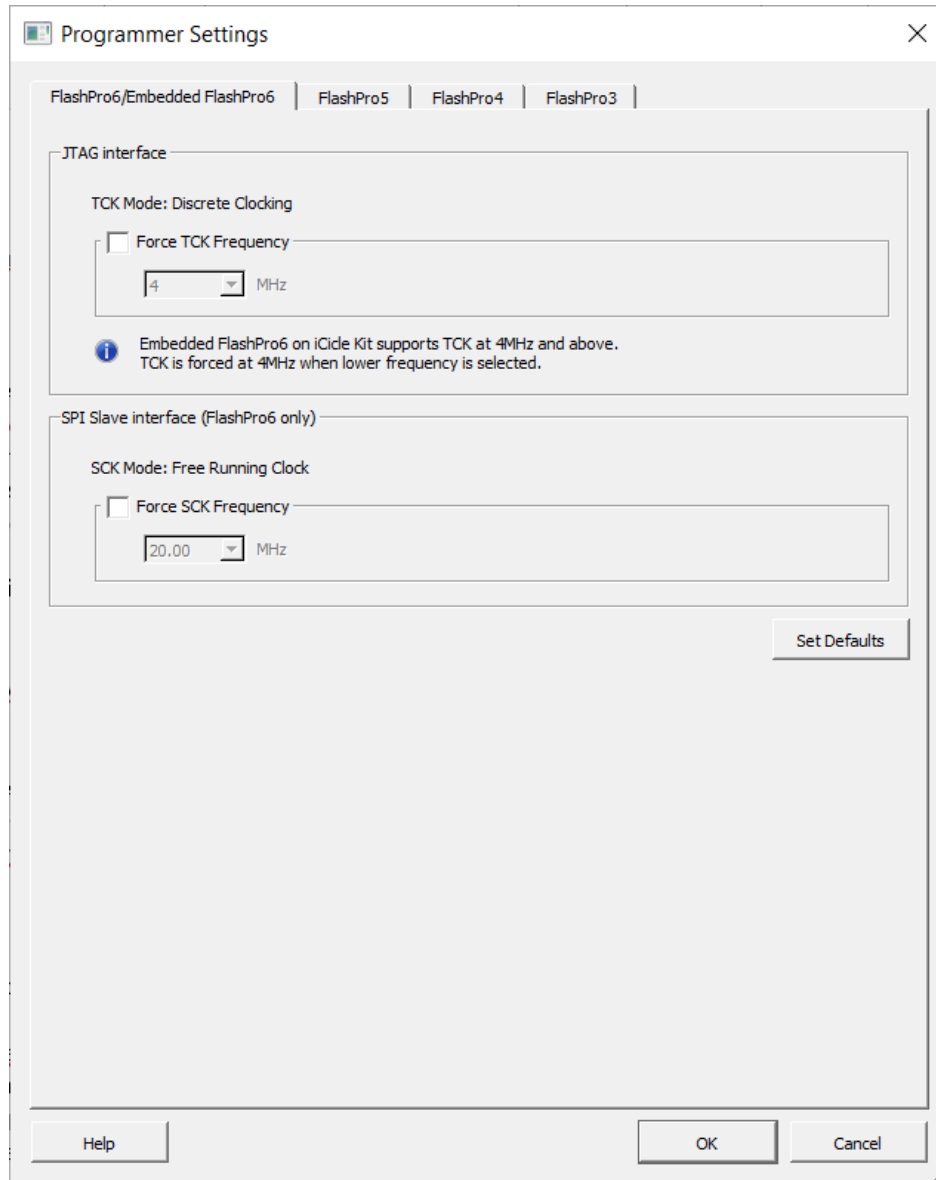
To display the Programmer Settings dialog box, in the Libero SoC Design Flow window, expand **Configure Hardware** and double-click **Configure Programmer**

OR

Right-click **Configure Programmer** and choose **Programmer Settings**.

**Note:** SPI Slave mode is supported by FlashPro6 for PolarFire devices.

**Figure 7-3. Programmer Settings Dialog Box (FlashPro6)**



The Programmer Settings dialog box has options for FlashPro6/5/4/3/3X. The following table lists the TCK frequency limitations for the selected programmer:

**Table 7-6. TCK Frequency Limitations**

Programmer	Limitations
FlashPro6	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20 MHz

.....continued	
Programmer	Limitations
FlashPro5	1, 2, 3, 4, 5, 6, 10, 15, 30 MHz
FlashPro4	1, 2, 3, 4, 5, 6 MHz
FlashPro3/3X	1, 2, 3, 4, 6 MHz

For information about TCK frequency limits by target device, see the target device data sheet.

During execution, the frequency set by the FREQUENCY statement in the PDB/STAPL file overrides the TCK frequency setting in the Programmer Settings dialog box. To prevent this override, check **Force TCK Frequency**.

The following list shows the SCK frequency limitations for the selected programmer:

- 1.00 MHz
- 2.00 MHz
- 2.50 MHz
- 3.33 MHz
- 4.00 MHz
- 5.00 MHz
- 6.67 MHz
- 8.00 MHz
- 10.00 MHz
- 13.33 MHz
- 20.00 MHz

### 7.3.1 FlashPro5/4/3/3X Programmer Settings

By default, **Force TCK Frequency** is not checked. This setting instructs the FlashPro5/4/3/3X to use the TCK frequency specified by the Frequency statement in the PDB/STAPL file(s). If you check **Force TCK Frequency**, select the appropriate MHz frequency.

For FlashPro4/3X settings, you can switch the TCK mode between **Free Running Clock** and **Discrete Clocking**. By default, **TCK Mode** is set to **Free Running Clock**. Use **Discrete Clocking** when there is a JTAG non-compliant device in a chain with Microchip devices.

After you make your selections, click **OK**.

**Note:** The **Set Vpump** check box has been removed. For older projects prior to Libero SoC v12.5, if **Set Vpump** was checked, the warning "Set Vpump parameter is obsolete. VPUMP will not be sensed or driven for all devices." appears in the log window when the design opens for the first time in Libero SoC v12.5.

### 7.3.2 TCK Setting (Force TCK Frequency)

If **Force TCK Frequency** is checked in **Programmer Setting**, the selected TCK value is set for the programmer and the Frequency statement in the PDB/STAPL file is ignored.

### 7.3.3 Default TCK Frequency

If the IPD/STAPL file or Chain does not exist, the default TCK frequency is set to 4 MHz. If more than one Microchip flash device is targeted in the chain, the FlashPro Express software passes through all the files and searches for the `freq` keyword and the MAX\_FREQ **Note** field. The FlashPro Express software uses the lowest value of all the TCK frequency settings and the MAX\_FREQ **Note** field values.

## 7.4 Select Programmer

The Select Programmer dialog box allows you to select the programmer you want to use.

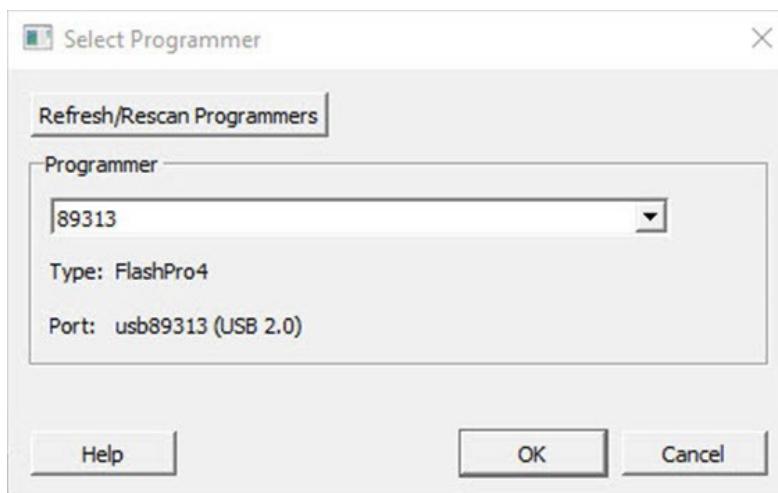
To display the Select Programmer dialog box, in the Libero SoC Design Flow window, expand **Configure Hardware** and double-click **Select Programmer**.

OR

Right-click **Select Programmer**.

Use the drop-down list to select the programmer you want to use. If no programmers are connected, connect a programmer without closing the dialog box, and then click **Refresh/Rescan Programmers** to display the connected programmer in the drop-down list.

**Figure 7-4. Select Programmer Dialog Box**



## 8. Program Design

The following topics provide program design considerations.

### 8.1 Generating FPGA Array Data

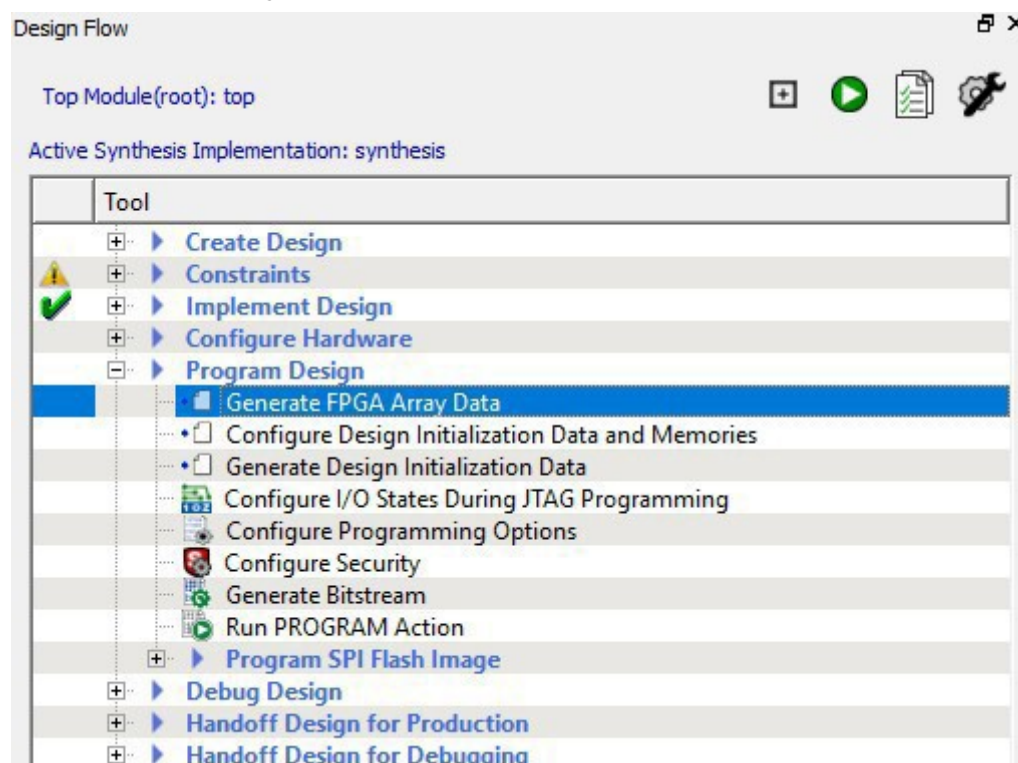
The Generate FPGA Array Data tool generates database files used in the following downstream tools:

- \*.map files used for Programming
- RAM structural information used in Configure Design Initialization and Memories tools

To generate FPGA array data:

1. Make sure the design completed the Place and Route step. If not, Libero SoC runs the upstream tools (Synthesis, Compile Netlist, and Place and Route) implicitly before it generates the FPGA Array Data.
2. Double-click **Generate FPGA Array Data** or right-click **Generate FPGA Array Data** in the Design Flow window and click **Run**

Figure 8-1. Generate FPGA Array Data



### 8.2 Initializing Design Blocks

The Configure Design Initialization Data and Memories tool allows you to initialize design blocks such as LSRAM, uSRAM, XCVR (transceivers), and PCIe using data stored in non-volatile uPROM, sNVM, or external SPI Flash storage memory. The tool has the following tabs for defining the specification of the design initialization sequence and the specification of the initialization clients:

- [Design Initialization tab](#)
- [uPROM tab](#)
- [sNVM tab](#)
- [SPI Flash tab](#)

- [eNVM tab](#) (applies to PolarFire SoC only)
- [Fabric RAMs tab](#)

**Note:** The Configure Design Initialization Data and Memories tool can be invoked only after completing the Generate FPGA Array Data step.

Use the tabs in the tool to configure the design initialization data and memories. If a tab title has an asterisk (\*) next to it, it means an item on that tab has been changed but not yet applied. The following table describes the buttons common to every tab.

**Table 8-1. Common Buttons on Every Tab**

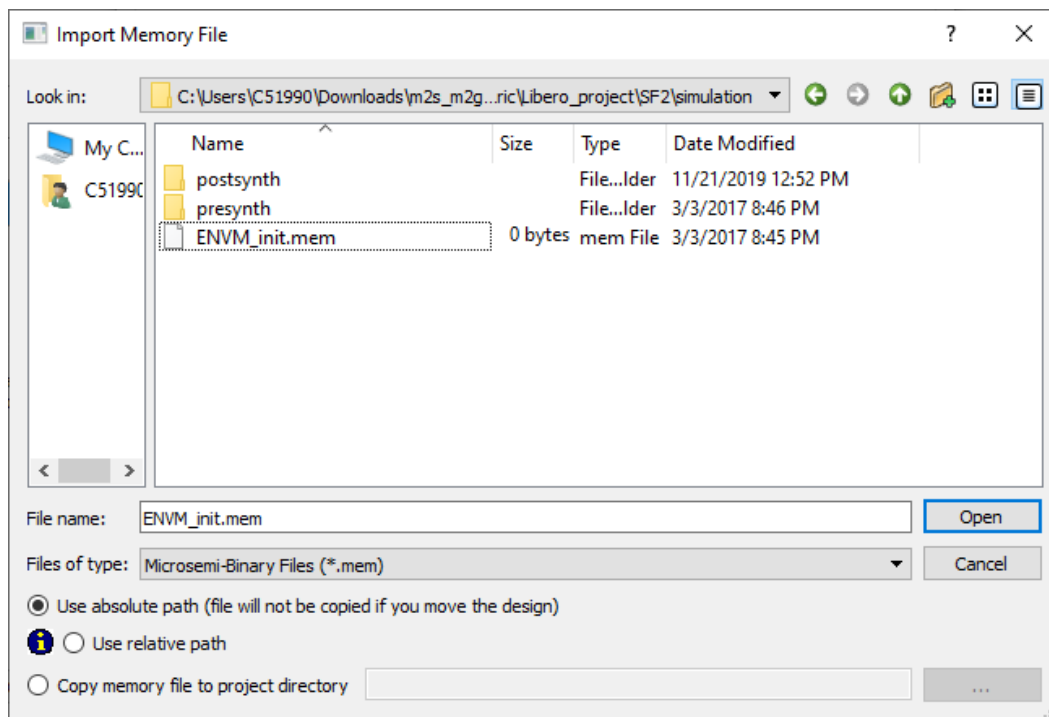
Button	Description
Apply	Click this button to save the changes made in a tab. The Apply button saves configuration changes only. For the initialization of the memory block to take effect, click <b>Generate Initialization Clients</b> on the <b>Device Initialization</b> tab.
Discard	Click this button to cancel any changes made in a tab.

After completing the configuration, perform the following steps to program the initialization data:

1. Generate initialization clients.
2. Generate or export the bitstream.
3. Program the device.

When importing memory files, note that the option **Use relative path** allows you to choose the relative to project or relative to environment variable, depending on the setting used in Libero. This option is extended to all memory files that are referenced in various configurators, as well as to sNVM/uPROM/SPI-Flash update tools.

**Figure 8-2. Import Memory File Dialog Box**



### 8.2.1 Design Initialization Tab

**Design Initialization** is the first tab in the Configure Design Initialization Data and Memories tool. The following topics describe the options in this tab.

**Figure 8-3. Design Initialization Tab**

**Design Initialization** | uPROM | sNVM | SPI-Flash | Fabric RAMs | eNVM

Apply | Discard | Help

In design initialization, user design blocks such as LSRAM, µSRAM, transceivers, and PCIe can be initialized as an option using data stored in the non-volatile storage memory. The initialization data can be stored in µPROM, sNVM, or an external SPI-Flash.

Follow the below steps to program the initialization data:

1. Set up your fabric RAMs initialization data, if any, using the 'Fabric RAMs' tab
2. Define the storage location of the initialization data
3. Generate the initialization clients
4. Generate or export the bitstream
5. Program the device

**Design initialization specification**

**First stage (sNVM)**

**In the first stage, the initialization sequence de-asserts FABRIC\_POR\_N.**

☒ Broadcast instructions to initialize RAMs to zeros

**Second stage (sNVM)**

**In the second stage, the initialization sequence initializes the PCIe and XCVR blocks present in the design.**

Start address for second stage initialization client: 0x 00000000

**Third stage (sNVM/uPROM/SPI-Flash)**

**In the third stage, the initialization sequence initializes the Fabric RAMs present in the design.**

To save the initialization instructions in sNVM/uPROM/SPI-Flash, please use 'Fabric RAMs' tab to make your selection for each RAM client.

☐ Start address for sNVM clients: 0x 00000000

☐ Start address for uPROM clients: 0x 00000000

☐ Start address for SPI-Flash clients: 0x 00000400

SPI-Flash Binding: SPI-Flash - No-binding Plaintext SPI Clock divider value: 2(40 MHz)

Time Out (s): 128

Auto Calibration Time Out (ms): 3000

Custom configuration file: ...

### 8.2.1.1 First Stage (sNVM)

In the first stage, the initialization sequence de-asserts the FABRIC\_POR\_N signal and starts the I/O calibration routine. The initialization client for this stage is always placed in sNVM. It uses the last two pages of the sNVM memory space. There is one configuration option for this stage.

**Figure 8-4. Design Initialization Tab - First Stage**

**Design Initialization** | uPROM | sNVM | SPI-Flash | Fabric RAMs | eNVM

Apply | Discard | Help

In design initialization, user design blocks such as LSRAM, µSRAM, transceivers, and PCIe can be initialized as an option using data stored in the non-volatile storage memory. The initialization data can be stored in µPROM, sNVM, or an external SPI-Flash.

Follow the below steps to program the initialization data:

1. Set up your fabric RAMs initialization data, if any, using the 'Fabric RAMs' tab
2. Define the storage location of the initialization data
3. Generate the initialization clients
4. Generate or export the bitstream
5. Program the device

**Design initialization specification**

**First stage (sNVM)**

**In the first stage, the initialization sequence de-asserts FABRIC\_POR\_N.**

☒ Broadcast instructions to initialize RAMs to zeros

**Second stage (sNVM)**

**In the second stage, the initialization sequence initializes the PCIe and XCVR blocks present in the design.**

Start address for second stage initialization client: 0x 00000000

**Third stage (sNVM/uPROM/SPI-Flash)**

**In the third stage, the initialization sequence initializes the Fabric RAMs present in the design.**

To save the initialization instructions in sNVM/uPROM/SPI-Flash, please use 'Fabric RAMs' tab to make your selection for each RAM client.

☐ Start address for sNVM clients: 0x 00000000

☐ Start address for uPROM clients: 0x 00000000

☐ Start address for SPI-Flash clients: 0x 00000400

SPI-Flash Binding: SPI-Flash - No-binding Plaintext SPI Clock divider value: 2(40 MHz)

Time Out (s): 128

Auto Calibration Time Out (ms): 3000

Custom configuration file: ...

Table 8-2. First Stage Configuration Option

Configuration Option	Description
Broadcast instructions to initialize RAMs to zeros	Affects all LSRAM and uSRAM blocks in the device. Selecting this option initializes all RAM blocks to zeros before FABRIC PoR is asserted. To accommodate the additional instructions, the sNVM start page will be 202. To initialize the individual logical RAM blocks to zeros without using this global option, select the <b>Content filled with 0s</b> option in the Add Data Storage Client dialog box, and then wait for the corresponding RAM INIT complete signal before accessing those RAMs (see the <a href="#">PolarFire FPGA Device Powerup and Reset User Guide</a> , which describes the INIT DONE/COMPLETE signals). If this global option is not selected, the sNVM start page will be 219.

### 8.2.1.2 Second Stage (sNVM)

In the second stage, the initialization sequence initializes the PCIe and XCVR blocks present in the design. This stage is grayed out if the design does not have PCIe or XCVR Blocks.

The initialization client for this stage is named INIT\_STAGE\_2\_SNVN\_CLIENT. It is always placed in sNVM at the start address of your choice. The start address can be at the start of an sNVM page (page boundary) only.

Each sNVM page is 256 bytes in size, so the valid start hexadecimal addresses are 0x0, 0x100, 0x200, and so on. Only the plain text non-authenticated client is supported for initialization.

Figure 8-5. Design Initialization Tab - Second Stage

Design Initialization | uPROM | sNVM | SPI Flash | Fabric RAMs | eNVM

Apply | Discard | Help

In design initialization, user design blocks such as LSRAM, uSRAM, transceivers, and PCIe can be initialized as an option using data stored in the non-volatile storage memory. The initialization data can be stored in uPROM, sNVM, or an external SPI Flash.

Follow the below steps to program the initialization data:

1. Set up your fabric RAMs initialization data, if any, using the 'Fabric RAMs' tab
2. Define the storage location of the initialization data
3. Generate the initialization clients
4. Generate or export the bitstream
5. Program the device

Design initialization specification

First stage (sNVM)

In the first stage, the initialization sequence de-asserts FABRIC\_POR\_II.

☒ Broadcast instructions to initialize RAMs to zeros

Second stage (sNVM)

In the second stage, the initialization sequence initializes the PCIe and XCVR blocks present in the design.

Start address for second stage initialization client: 0x 00000000

Third stage (sNVM/uPROM/SPI-Flash)

In the third stage, the initialization sequence initializes the Fabric RAMs present in the design.

To save the initialization instructions in sNVM/uPROM/SPI-Flash, please use 'Fabric RAMs' tab to make your selection for each RAM client.

☐ Start address for sNVM clients: 0x 00000000

☐ Start address for uPROM clients: 0x 00000000

☐ Start address for SPI-Flash clients: 0x 00000400

SPI-Flash Binding: SPI-Flash - No-binding Plaintext SPI Clock divider value: 2(40 MHz)

Time Out (s): 128

Auto Calibration Time Out (ms): 3000

Custom configuration file:

### 8.2.1.3 Third Stage (uPROM/sNVM/SPI Flash)

In the third stage, the initialization sequence initializes the Fabric RAMs present in the design. The initialization client for this stage is placed in the memory type of the user's choice (uPROM/sNVM/External SPI Flash). If the design does not have any Fabric RAMs, this stage of the initialization sequence is not needed and is grayed out. Each logical RAM block can be initialized from any of the three memory types. Use the Fabric RAMs configuration tab to assign the memory type to the logical RAM blocks.

Only the memory types used by the design, as defined in the Fabric RAMs configuration tab, are selected and enabled.

**Figure 8-6. Design Initialization Tab - Third Stage**

Design Initialization | uPROM | sNVM | SPI-Flash | Fabric RAMs | eNVM

Apply | Discard | Help

In design initialization, user design blocks such as LSRAM, uSRAM, transceivers, and PCIe can be initialized as an option using data stored in the non-volatile storage memory. The initialization data can be stored in uPROM, sNVM, or an external SPI Flash.

Follow the below steps to program the initialization data:

1. Set up your fabric RAMs initialization data, if any, using the 'Fabric RAMs' tab
2. Define the storage location of the initialization data
3. Generate the initialization clients
4. Generate or export the bitstream
5. Program the device

Design initialization specification

First stage (sNVM)

In the first stage, the initialization sequence de-asserts FABRIC\_POR\_N.

☒ Broadcast instructions to initialize RAMs to zeros

Second stage (sNVM)

In the second stage, the initialization sequence initializes the PCIe and XCVR blocks present in the design.

Start address for second stage initialization client: 0x 00000000

Third stage (sNVM/uPROM/SPI-Flash)

In the third stage, the initialization sequence initializes the Fabric RAMs present in the design.

To save the initialization instructions in sNVM/uPROM/SPI-Flash, please use 'Fabric RAMs' tab to make your selection for each RAM client.

☐ Start address for sNVM clients: 0x 00000000

☐ Start address for uPROM clients: 0x 00000000

☐ Start address for SPI-Flash clients: 0x 00000400

SPI-Flash Binding: SPI-Flash - No-binding Plaintext SPI Clock divider value: 2 (40 MHz)

Time Out (s): 128

Auto Calibration Time Out (ms): 3000

Custom configuration file: ...

**Table 8-3. Memory Types**

Memory Type	Description
uPROM	Name of the initialization client is INIT_STAGE_3_UPROM_CLIENT. Its start address is at the user's choice, subject to the limitation that the start address can only be at the start of a uPROM block. Each uPROM block is 256 words, so the allowed hexadecimal start addresses are 0x0, 0x100, 0x200, and so on.
sNVM	Name of the initialization client is INIT_STAGE_3_SNVM_CLIENT. Its start address is at the user's choice, subject to the limitation that the start address can only be at the start of an sNVM page (page boundary). Each sNVM page is 256 bytes long, so the allowed hexadecimal start addresses are 0x0, 0x100, 0x200, and so on.
SPI-Flash	Name of the initialization client is INIT_STAGE_3_SPIFLASH_CLIENT.



.....continued	
Memory Type	Description
SPI-Flash Binding	<p>The four Binding options that you can select are:</p> <ul style="list-style-type: none"> <li>• No Binding Plaintext: &lt;root&gt;_uic.bin file is a script file that can be opened to see readable text.</li> <li>• Binding Encrypted with Default Key: &lt;root&gt;_uic.bin file is encrypted with the default encryption key. The design version is displayed and can be modified from Configure Programming Options. If Default key is selected, you do not need to specify any other details.</li> <li>• Binding Encrypted with User Encryption Key 1 (UEK1): &lt;root&gt;_uic.bin file is encrypted with UEK1. The design version is displayed and can be modified from Configure Programming Options. You must configure SPM along with UEK1. If UEK1 is not specified, the Generate SPI Flash Image and Export SPI Flash Image steps cause an error. UEK1 can be configured using the Configure Security Tool.</li> <li>• Binding Encrypted with User Encryption Key 2 (UEK2): &lt;root&gt;_uic.bin file is encrypted with UEK2. The design version is displayed and can be modified from Configure Programming Options. You must configure SPM along with UEK2. If UEK2 is not specified, the Generate SPI Flash Image and Export SPI Flash Image steps cause an error. UEK2 can be configured using the Configure Security Tool.</li> </ul>
SPI Clock divider value	<p>Sets the clock divider value for the clock that is generated by the System Controller. Choose the value that meets the minimum clock width requirement of the external SPI Flash.</p> <p>Range: 1, 2, 4, 6</p> <p>Default: 2</p>

#### 8.2.1.4 Time Out (s)

Use the **Time Out (s)** drop-down list to select a time-out for completing all three stages of the initialization process. The default setting is 128, which is the maximum value.

**Figure 8-7. Design Initialization Tab - Time Out (s)**

Design Initialization | uPROM | sNVM | SPI Flash | Fabric RAMs | eNVM

Apply | Discard | Help

In design initialization, user design blocks such as LSRAM, µSRAM, transceivers, and PCIe can be initialized as an option using data stored in the non-volatile storage memory. The initialization data can be stored in µPROM, sNVM, or an external SPI Flash.

Follow the below steps to program the initialization data:

1. Set up your fabric RAMs initialization data, if any, using the 'Fabric RAMs' tab
2. Define the storage location of the initialization data
3. Generate the initialization clients
4. Generate or export the bitstream
5. Program the device

Design initialization specification

First stage (sNVM)

In the first stage, the initialization sequence de-asserts FABRIC\_POR\_N.

☒ Broadcast instructions to initialize RAMs to zeros

Second stage (sNVM)

In the second stage, the initialization sequence initializes the PCIe and XCVR blocks present in the design.

Start address for second stage initialization client: 0x 00000000

Third stage (sNVM/uPROM/SPI-Flash)

In the third stage, the initialization sequence initializes the Fabric RAMs present in the design.

To save the initialization instructions in sNVM/uPROM/SPI-Flash, please use 'Fabric RAMs' tab to make your selection for each RAM client.

☐ Start address for sNVM clients: 0x 00000000

☐ Start address for uPROM clients: 0x 00000000

☐ Start address for SPI-Flash clients: 0x 00000400

SPI-Flash Binding: SPI-Flash - No-binding Plaintext SPI Clock divider value: 2(40 MHz)

Time Out (s): 128

Auto Calibration Time Out (ms): 3000

Custom configuration file: ...

### 8.2.1.5 Auto Calibration Time Out

The **Auto Calibration Time Out** value specifies the time-out before which the I/O calibration instructions must be completed. The default value is 3000 milliseconds. This time-out value applies to MPF100T, MPF200T, MPF300T, and MPF500T devices.

**Figure 8-8. Design Initialization Tab - Auto Calibration Time Out**

Design Initialization | uPROM | sNVM | SPI Flash | Fabric RAMs | eNVM

Apply | Discard | Help

In design initialization, user design blocks such as LSRAM, µSRAM, transceivers, and PCIe can be initialized as an option using data stored in the non-volatile storage memory. The initialization data can be stored in µPROM, sNVM, or an external SPI Flash.

Follow the below steps to program the initialization data:

1. Set up your fabric RAMs initialization data, if any, using the 'Fabric RAMs' tab
2. Define the storage location of the initialization data
3. Generate the initialization clients
4. Generate or export the bitstream
5. Program the device

Design initialization specification

First stage (sNVM)

In the first stage, the initialization sequence de-asserts FABRIC\_POR\_N.

☒ Broadcast instructions to initialize RAMs to zeros

Second stage (sNVM)

In the second stage, the initialization sequence initializes the PCIe and XCVR blocks present in the design.

Start address for second stage initialization client: 0x 00000000

Third stage (sNVM/uPROM/SPI-Flash)

In the third stage, the initialization sequence initializes the Fabric RAMs present in the design.

To save the initialization instructions in sNVM/uPROM/SPI-Flash, please use 'Fabric RAMs' tab to make your selection for each RAM client.

☐ Start address for sNVM clients: 0x 00000000

☐ Start address for uPROM clients: 0x 00000000

☐ Start address for SPI-Flash clients: 0x 00000400

SPI-Flash Binding: SPI-Flash - No-binding Plaintext SPI Clock divider value: 2(40 MHz)

Time Out (s): 128

Auto Calibration Time Out (ms): 3000

Custom configuration file: ...

### 8.2.1.6 Custom Configuration File

The **Custom Configuration file** contains signal integrity parameters for Transceivers. Click the **Browse** button at the far right to navigate to and select a custom configuration file for Transceiver solutions. For more information, contact Microchip Technical Support.

**Figure 8-9. Design Initialization Tab - Custom Configuration File**

Design Initialization | uPROM | sNVM | SPI Flash | Fabric RAMs | eNVM

Apply | Discard | Help

In design initialization, user design blocks such as LSRAM, uSRAM, transceivers, and PCIe can be initialized as an option using data stored in the non-volatile storage memory. The initialization data can be stored in uPROM, sNVM, or an external SPI Flash.

Follow the below steps to program the initialization data:  
 1. Set up your fabric RAMs initialization data, if any, using the 'Fabric RAMs' tab  
 2. Define the storage location of the initialization data  
 3. Generate the initialization clients  
 4. Generate or export the bitstream  
 5. Program the device

Design initialization specification

First stage (sNVM)  
 In the first stage, the initialization sequence de-asserts FABRIC\_POR\_N.  
☒ Broadcast instructions to initialize RAMs to zeros

Second stage (sNVM)  
 In the second stage, the initialization sequence initializes the PCIe and XCVR blocks present in the design.  
 Start address for second stage initialization client: 0x 00000000

Third stage (sNVM/uPROM/SPI-Flash)  
 In the third stage, the initialization sequence initializes the Fabric RAMs present in the design.  
 To save the initialization instructions in sNVM/uPROM/SPI-Flash, please use 'Fabric RAMs' tab to make your selection for each RAM client.  
☐ Start address for sNVM clients: 0x 00000000  
☐ Start address for uPROM clients: 0x 00000000  
☐ Start address for SPI-Flash clients: 0x 00000400  
 SPI-Flash Binding: SPI-Flash - No-binding Plaintext SPI Clock divider value: 2(40 MHz)

Time Out (s): 128  
 Auto Calibration Time Out (ms): 3000

Custom configuration file:

## 8.2.2 uPROM Tab

**uPROM** is the second tab in the Configure Design Initialization Data and Memories tool. Use this tab to manage data clients targeted for uPROM memory.

**Figure 8-10. uPROM Tab**

Design Initialization | **uPROM** | sNVM | SPI Flash | Fabric RAMs | eNVM

Apply | Discard | Help

Usage statistics  
 Available memory(9-bit words): 52224  
 Used memory(9-bit words): 0  
 Free memory(9-bit words): 52224

Clients  
 Add ... | Edit ... | Delete | Load design configuration

Client Name	Start Address	9-bit words
-------------	---------------	-------------

The following table describes the elements in the **uPROM** tab.

**Table 8-4. Elements in the uPROM Tab**

Element	Description
Add button	Adds uPROM clients.
Edit button	Edits uPROM clients.
Delete button	Deletes uPROM clients.

.....continued	
Element	Description
Load Design Configuration button	<p>Loads the design's original uPROM configuration file into the &lt;project&gt;/component/work/UPROM.cfg file. This button is grayed out if the design does not have an original uPROM configuration file.</p> <p>This configuration changes when the design is updated in the design window. If changes are made to the design configuration after you click Apply, info icons appear next to the Load design configuration button and the title of the uPROM tab.</p> <p>The tool-tip for both icons contains the time-stamp information of the design configuration file. The icons disappear the next time you click Apply.</p>
Usage Statistics pie chart	Shows memory usage for the uPROM.

#### 8.2.2.1 Adding uPROM Clients

1. In the **uPROM** tab, click **Add**.
2. When the Add Data Storage Client dialog box appears, complete the fields (see the following figure and table).
3. Click **OK**.
4. Click the **Apply** button. The client is added to the [uPROM clients table](#).

Figure 8-11. Add Data Storage Client Dialog Box

Table 8-5. Add Data Storage Client Dialog Box

Field	Description
Client name	Name of the uPROM client to be added.
Content from file	Navigate to and select a file whose content will be used to fill the uPROM.
Format	Memory file types. Choices are: <ul style="list-style-type: none"> <li>• Micro Binary 9-bit (<i>default</i>)</li> <li>• Micro Binary 32-bit</li> <li>• Intel-Hex</li> <li>• Motorola-S</li> <li>• Simple-Hex</li> </ul>
Content filled with 0s	Populates the uPROM with zeros.
Start address	Start address, in hexadecimal notation, of the uPROM client. If there are multiple uPROM clients, the start address must not overlap; otherwise, a warning message appears. Range: 0 – CBFF (Hex)

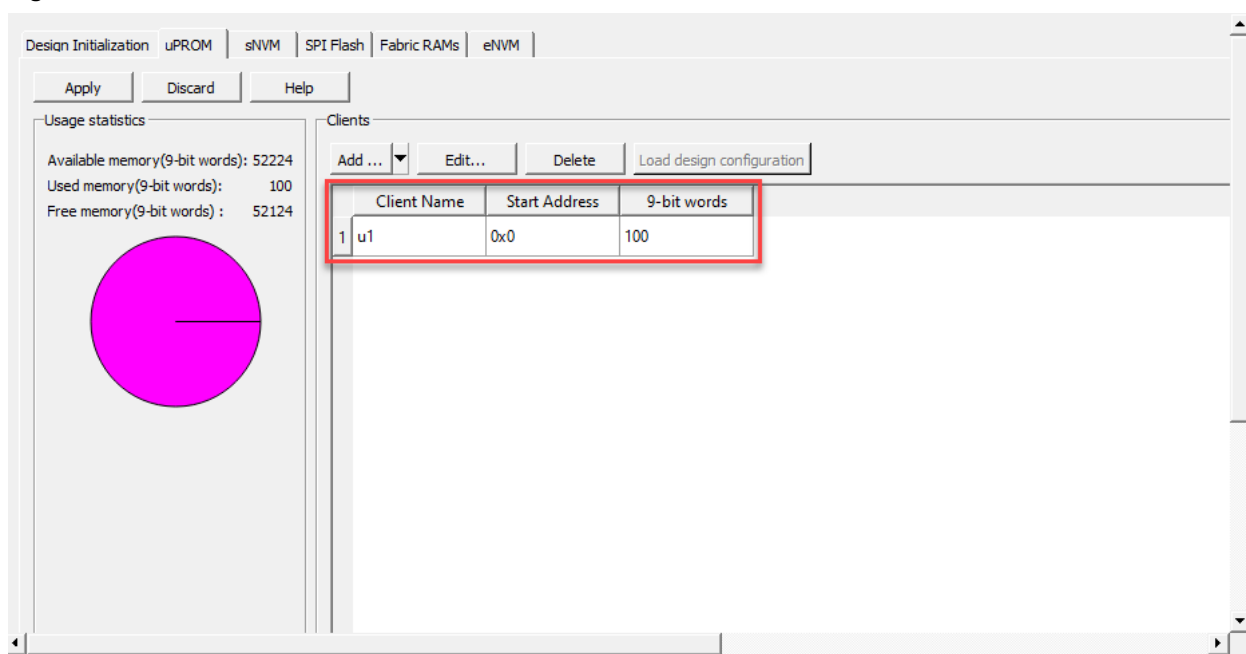
.....continued	
Field	Description
Number of 9-bit words	Number, in decimal notation, of 9-bit words to populate the uPROM. If the number of 9-bit words exceeds the memory size of the uPROM, an “out-of –bounds” warning message appears.
Use for initialization of RAMs	Disabled and unavailable.
Use content for simulation	Disabled and unavailable.

#### 8.2.2.1.1 uPROM Clients Table

The uPROM clients table shows the uPROM clients you add.

Each uPROM client appears on its own row. After you add a uPROM client, you can select it in this table to [edit](#) or [delete](#) the client.

**Figure 8-12. uPROM Clients Table**



**Table 8-6. Columns in the uPROM Clients Table**

Column	Description
Client Name	Name you gave to the client.
Start Address	Starting address you gave to the client.
9-bit words	Number of 9-bit words in the client.

#### 8.2.2.2 Editing uPROM Clients

If you need to change the settings for a uPROM client, you can edit the client.

To edit a uPROM client:

- In the table of the **uPROM** tab, perform one of the following steps:
  - Double-click the client you want to edit.
  - Click the client you want to edit, and then click the **Edit** button.
  - Right-click the client you want to edit, and select **Edit**.
- When the Edit Data Storage Client dialog box appears, complete the fields (see the following table).

3. Click **OK**.
4. Click the **Apply** button.

**Table 8-7. Edit Data Storage Client Dialog Box**

Field	Description
Client name	Read-only field that shows the name of the uPROM client.
Content from file	Navigate to and specify a file whose content will be used to fill the uPROM.
Content filled with 0s	Populates the uPROM with zeros.
Start address	Start address, in hexadecimal notation, of the uPROM client. If there are multiple uPROM clients, the start address must not overlap; otherwise, a warning message appears. Range: 0 - CBFF (Hex)
Number of 9-bit words	Number, in decimal notation, of 9-bit words to populate the uPROM. If the number of 9-bit words exceeds the memory size of the uPROM, an “out-of –bounds” warning message appears.
Use for initialization of RAMs	Disabled and unavailable.
Use content for simulation	Disabled and unavailable.

### 8.2.2.3 Deleting uPROM Clients

If you no longer need a uPROM client, you can delete the client.



A warning message does not appear before you delete a client. Therefore, be sure you no longer need a client before you delete it.

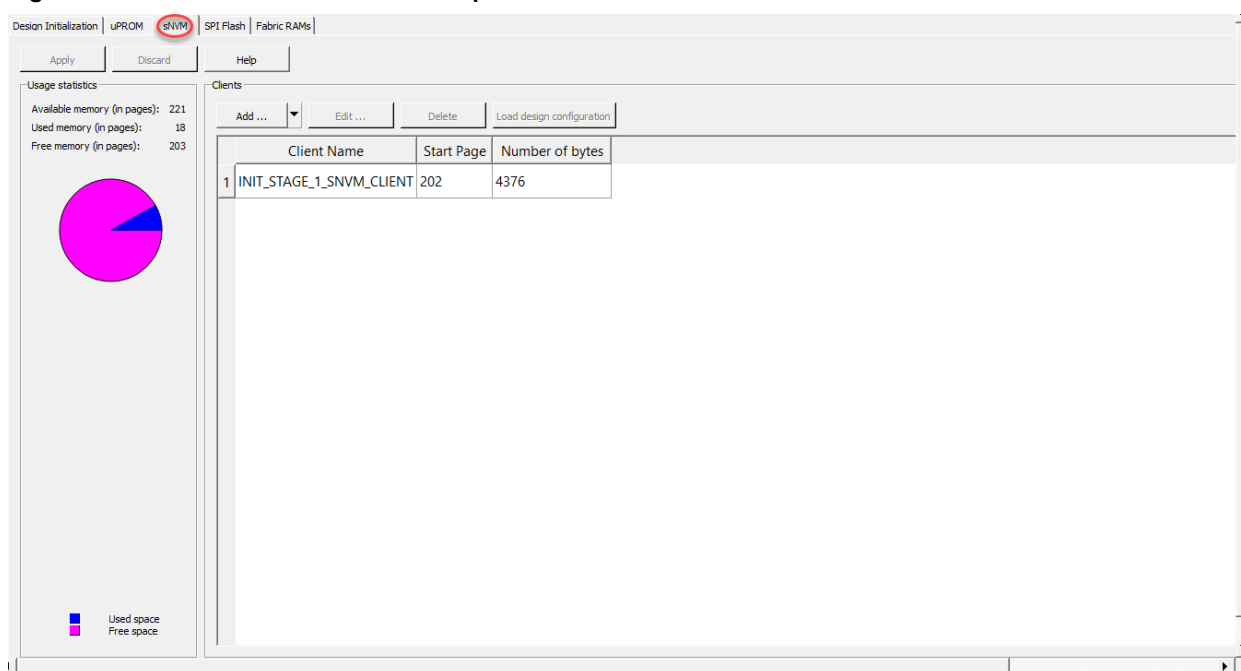
To delete a uPROM client:

1. In the table at the bottom of the **uPROM** tab, perform one of the following steps:
  - Click the client you want to delete, and then click the **Delete** button.
  - Right-click the client you want to delete, and select **Delete**.
2. Click the **Apply** button.

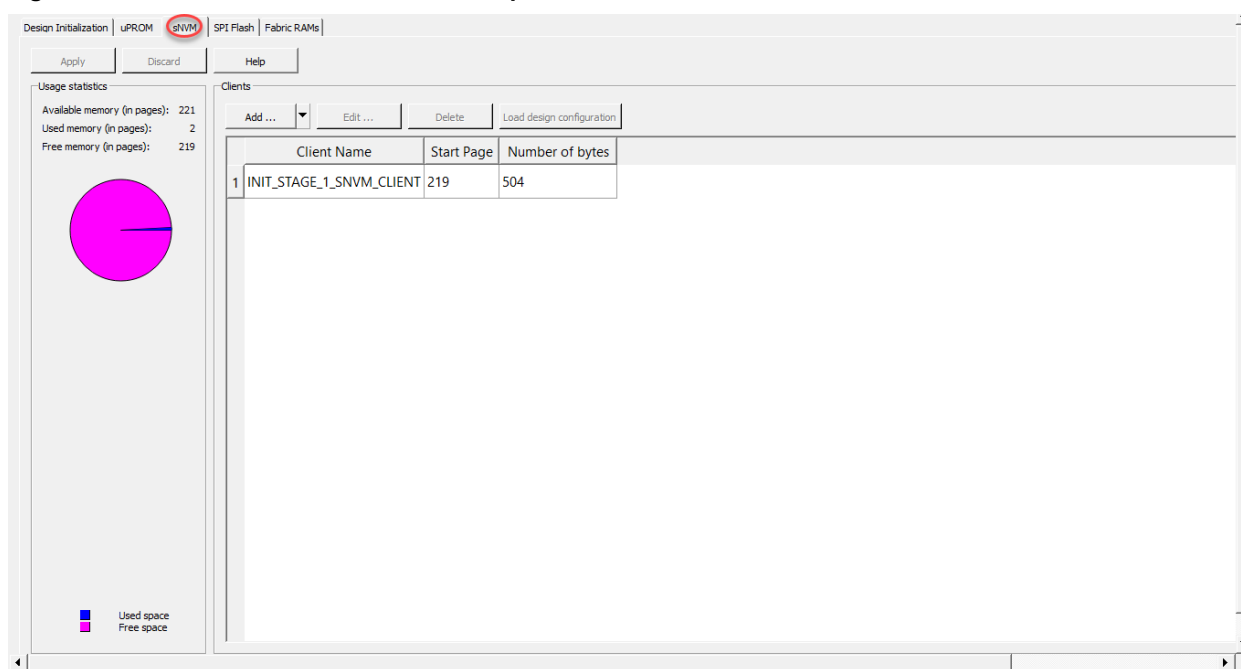
### 8.2.3 sNVM Tab

**sNVM** is the third tab in the Configure Design Initialization Data and Memories tool. Use this tab to manage data clients targeted for sNVM memory. The table in the tab is automatically populated if **Broadcast instructions to initialize RAM's to zero's** is checked in the **Design Initialization** tab.

**Figure 8-13. sNVM Tab with Broadcast Option Enabled**



**Figure 8-14. sNVM Tab without Broadcast Option Enabled**



The following table describes the elements in the **sNVM** tab.

**Table 8-8. Elements in the sNVM Tab**

Element	Description
Add button	Adds sNVM clients.
Edit button	Edits sNVM clients.
Delete button	Deletes sNVM clients.

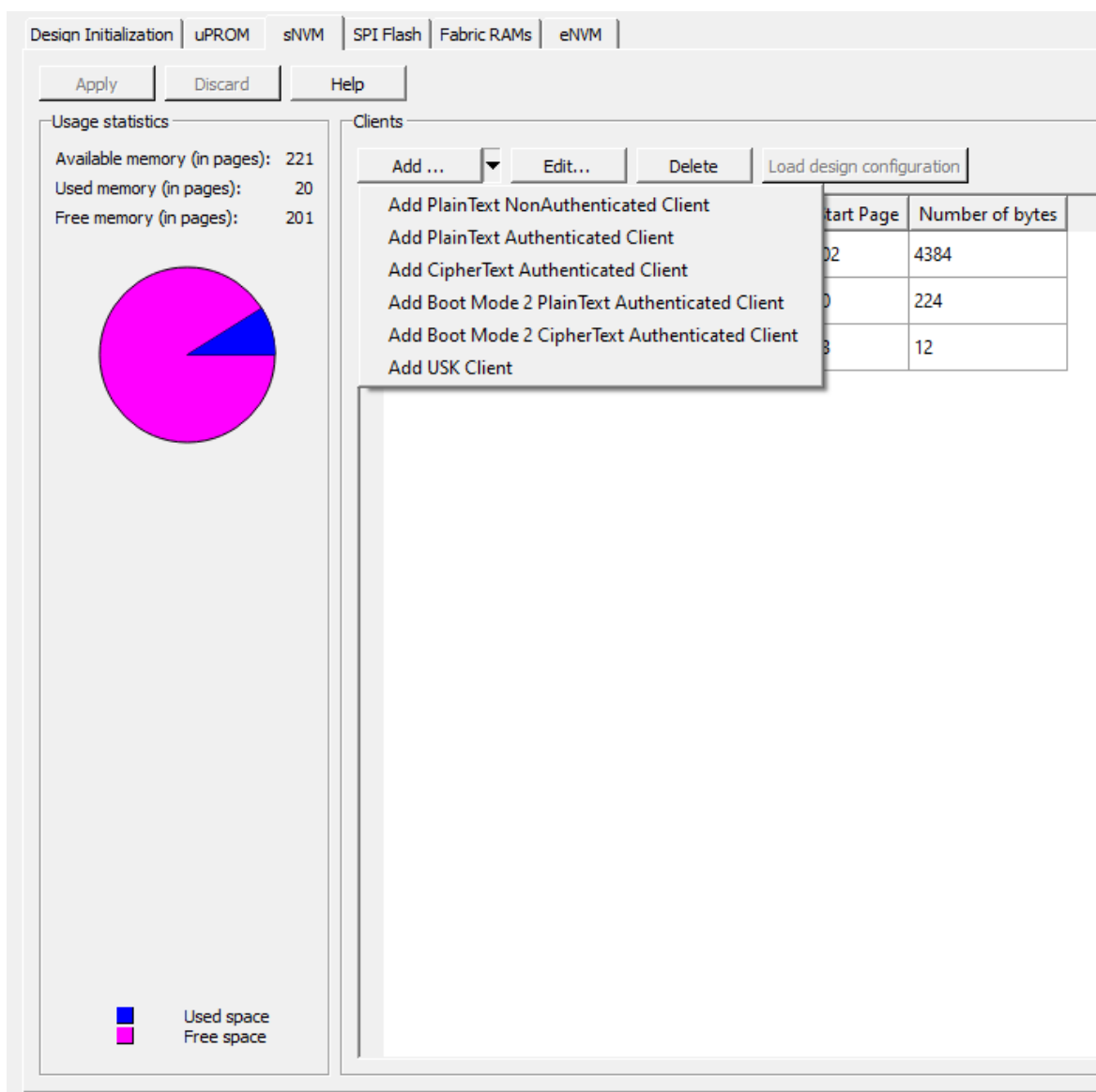


.....continued	
Element	Description
Load Design Configuration button	<p>Loads the design's original sNVM configuration file into the &lt;project&gt;/component/work/sNVM.cfg file. This button is grayed out if the design does not have an original sNVM configuration file.</p> <p>This configuration changes when the design is updated in the design window. If changes are made to the design configuration after you click Apply, info icons appear next to the Load design configuration button and the title of the sNVM tab.</p> <p>The tool-tip for both icons contains the time-stamp information of the design configuration file. The icons disappear the next time you click Apply.</p>
Usage Statistics pie chart	Shows available, used, and free memory, in pages, for all sNVM clients.

### 8.2.3.1 Adding sNVM Clients

1. In the **sNVM** tab, click the **Add** drop-down list, and then select the client you want to add (see the following figure).

**Figure 8-15. sNVM Client Selections**



2. Complete the fields in the dialog box, and then click **OK** (see the following sections).
3. Click the **Apply** button. The client is added to the [sNVM clients table](#).

#### 8.2.3.1.1 Settings for Add PlainText and Add CipherText Clients

In the **sNVM** tab, click the **Add** drop-down list, and then select the **Add Plain Text NonAuthenticated Client** to add the client to the [sNVM clients table](#).

Figure 8-16. Add Plain Text NonAuthenticated Client Dialog Box

Table 8-9. Fields in the Add PlainText NonAuthenticated Client Dialog Box

Field	Description
Client name	Name of the sNVM client to be added.
Content from file	Navigate to and specify a file whose content will be used to fill the sNVM. <b>Note:</b> If you select the format Intel-HEX, the Base address specified is subtracted from user address records. Intel-Hex files have Extended Linear and Extended Segment addresses. The Complete Starting address of the Linear or Segment address in the Hex file must be specified. For example, if the Intel-Hex file has the Extended Linear address 2022, specify the base address 20220000.
Content filled with 0s	Populates the sNVM with zero's.
No content	Client is a placeholder and will not be programmed.

.....continued	
Field	Description
Start page	Start page, in decimal notation, of the sNVM client. sNVM client address starts at page boundaries. If there are multiple sNVM clients, their start page cannot be the same; otherwise, a warning message appears. Range: 0 – 220 (decimal)
Number of bytes	Total number, in decimal notation, of bytes to populate the sNVM. If the number of bytes exceeds the memory size of the sNVM, an out-of-bounds warning message appears. Range: 1 – 47376
Use content for simulation	Check if this client should be loaded for the simulation run.
Fabric access	Allows you to read from Fabric, write to Fabric, or both.
MSS access	Allows you to read from MSS, write to MSS, or both.

#### 8.2.3.1.2 Settings for Boot Mode 2 Clients

In this boot mode, you specify the start page in SNVM. All authenticated/encrypted clients will share the same USK. If you add authenticated/encrypted clients, you must create a USK client to specify the USK.

Figure 8-17. Add PlainText Authenticated Boot Mode 2 Client Dialog Box

The dialog box is titled "Add PlainText Authenticated Boot Mode 2 client". It features a standard Windows-style title bar with a question mark icon and a close button (X). The main content area is divided into sections. The first section is labeled "Client name:" followed by a text input field containing the text "BOOT\_MODE\_2\_SNVM\_PLAIN\_TEXT\_AUTH\_CLIENT". Below this is a section labeled "sNVM" which contains several configuration fields:
 

- "Content from file:" with a text input field and a browse button (three dots).
- "Format:" with a dropdown menu currently showing "Intel-Hex".
- "Base Address:" with a blue information icon (i) and a text input field containing "0x00000000".
- "Start page (decimal):" with a text input field containing "0" and a blue "0x0" label to its right.
- "Number of bytes (decimal):" with a text input field containing "0" and a blue "0 page" label to its right.

 At the bottom of the dialog are three buttons: "Help", "OK", and "Close".

Figure 8-18. Add CipherText Authenticated Boot Mode 2 Client Dialog Box

Table 8-10. Fields in the Boot Mode 2 Client Dialog Box

Field	Description
Client name	Read-only field that shows the name of the sNVM client.
Content from file	Navigate to and specify a file whose content will be used to fill the sNVM. <b>Note:</b> If you select the format Intel-HEX, the Base address specified is subtracted from user address records. Intel-Hex files have Extended Linear and Extended Segment addresses. The Complete Starting address of the Linear or Segment address in the Hex file must be specified. For example, if the Intel-Hex file has the Extended Linear address 2022, specify the base address 20220000.
Format	Memory file types. Choice is Intel-Hex. The Intel-Hex file is generated using Soft Console.
Base Address	Base address that is subtracted from the user address records for Intel-Hex files.
Start page	Start page, in decimal notation, of the sNVM client. sNVM client address starts at page boundaries. If there are multiple sNVM clients, their start page cannot be the same; otherwise, a warning message appears. Range: 0 – 220 (decimal)

.....continued	
Field	Description
Number of bytes	Read-only field that shows the total number of bytes to populate the sNVM. The value is shown in decimal notation. If the number of bytes exceeds the memory size of the sNVM, an out-of-bounds warning message appears. Range: 1 – 47376

### 8.2.3.1.3 Settings for Add USK Clients

Figure 8-19. Add USK Client Dialog Box

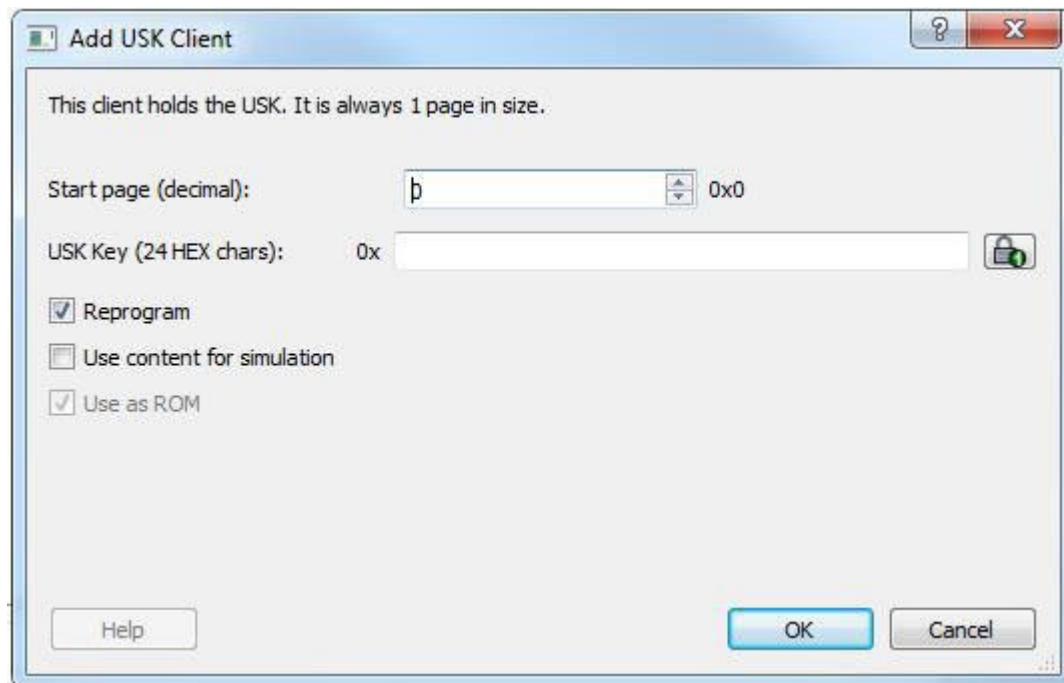


Table 8-11. Fields in the Add USK Client Dialog Box

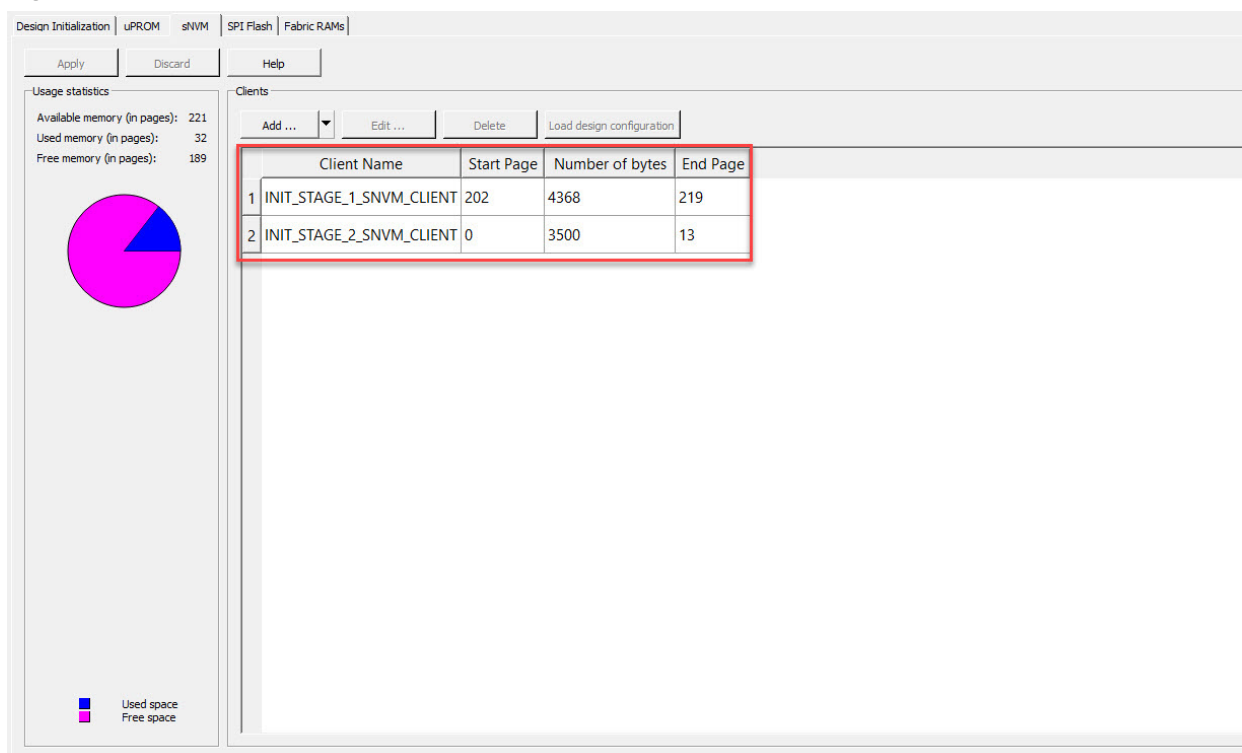
Field	Description
Start page	Start page can vary between 0 and 220.
USK Key	USK key (24 hexadecimal characters). A random key can be generated by clicking the padlock icon to the right of this field.
Reprogram	Check if this client should be programmed.
Use content for simulation	Check if this client should be loaded for the simulation run.
Use as ROM	Check if this client should be used as ROM.

### 8.2.3.1.4 sNVM Clients Table

The sNVM clients table shows the sNVM clients you add.

Each sNVM client appears on its own row. After you add an sNVM client, you can select it in this table to edit or delete the client.

Figure 8-20. sNVM Clients Table



The following table describes the columns in the sNVM clients table.

Table 8-12. Columns in the sNVM Clients Table

Column	Description
Client Name	Name you gave to the client.
Start Page	Starting page you gave to the client.
Number of bytes	Number of bytes in the client.
End Page	Ending page that Libero SoC ascertained based on the start page you provided.

### 8.2.3.2 Editing sNVM Clients

If you need to change the settings for an sNVM client, you can edit the client.

To edit an sNVM client:

- In the table of the **sNVM** tab, perform one of the following steps:
  - Double-click the client you want to edit.
  - Click the client you want to edit, and then click the **Edit** button.
  - Right-click the client you want to edit, and select **Edit**.
- When the dialog box appears, complete the fields (see [Settings for Add PlainText and Add CipherText Clients](#), [Settings for Boot Mode 2 Clients](#), or [Settings for Add USK Clients](#)).
- Click **OK**.
- Click the **Apply** button.

### 8.2.3.3 Deleting sNVM Clients

If you no longer need an sNVM client, you can delete the client.



A warning message does not appear before you delete a client. Therefore, be sure you no longer need a client before you delete it.

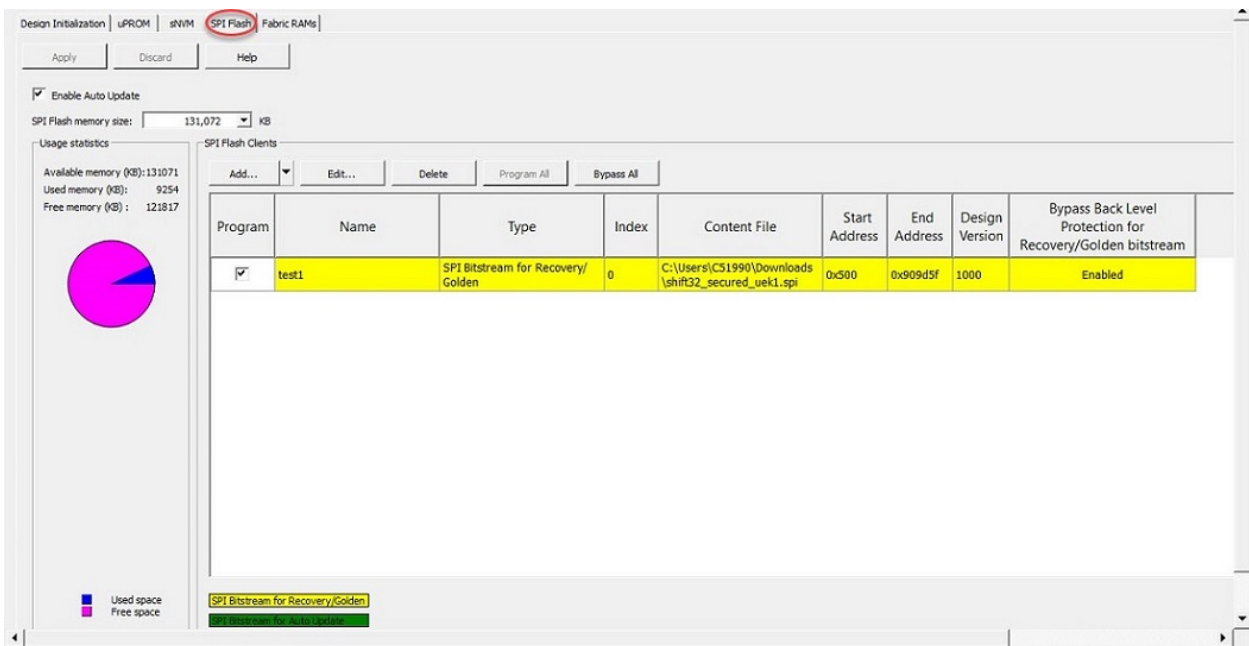
To delete an sNVM client:

1. In the table at the bottom of the **sNVM** tab, perform one of the following steps:
  - Click the client you want to delete, and then click the **Delete** button.
  - Right-click the client you want to delete, and select **Delete**.
2. Click the **Apply** button.

### 8.2.4 SPI Flash Tab

**SPI Flash** is the fourth tab in the Configure Design Initialization Data and Memories tool. Use this tab to enable auto update and select the SPI flash memory size in kilobytes. The configuration is saved in the `spiflash.cfg` file in the Libero design implementation folder.

**Figure 8-21. SPI Flash Tab**



The following table describes the elements in the **SPI Flash** tab.

**Table 8-13. Elements in the SPI Flash Tab**

Element	Description
Add button	Adds SPI Flash clients.
Edit button	Edits SPI Flash clients.
Delete button	Deletes SPI Flash clients.
Program All	This option selects all clients for programming at once. It is enabled when there is at least one unselected client.

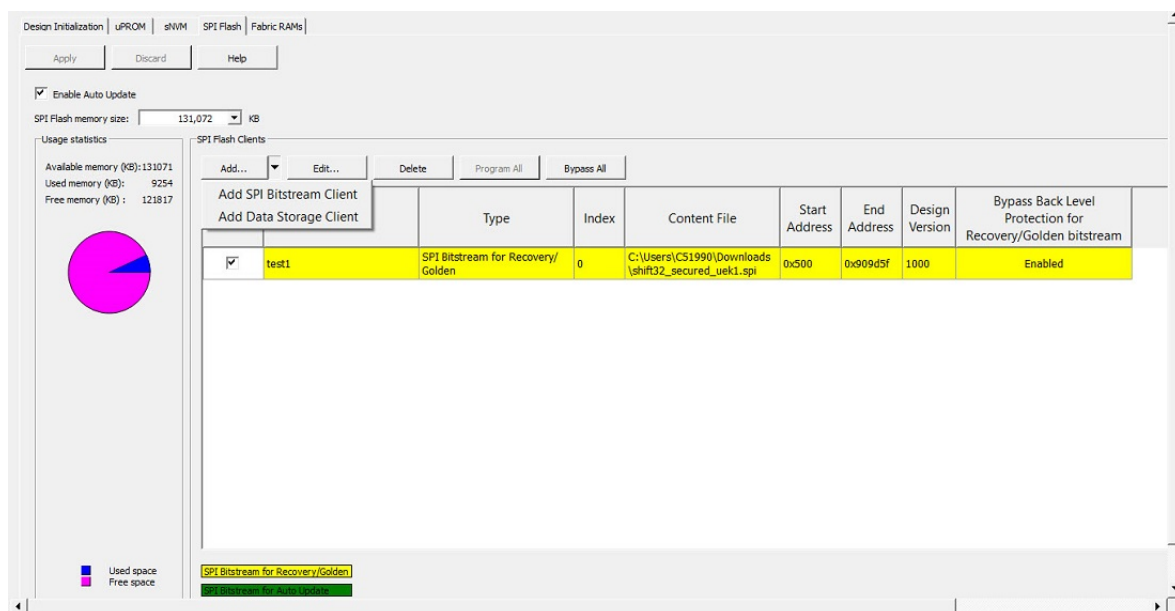


.....continued	
Element	Description
Bypass All	This option unselects all clients, except for STAGE 3 Initialization client. It is enabled if there is at least one client besides the STAGE 3 Initialization that is selected for programming (STAGE 3 Initialization client must always be programmed).
Enable Auto Update check box	<p>Enables auto update on the target device. The bitstream generated in Libero enables this feature.</p> <p>If you check this option, two SPI Bitstreams can be added.</p> <ul style="list-style-type: none"> <li>One for auto update.</li> <li>Another for recovery/golden.</li> </ul> <p>The tool enforces the recovery/golden bitstream to be at index 0 and the auto update bitstream to be at index 1. The auto update bitstream design version must be greater than the design version of the recovery/golden bitstream.</p>
SPI Flash memory size	Selects the memory size, in KB, for the SPI Flash.
Usage Statistics pie chart	Shows available, used, and free memory, in KB, for all SPI Flash clients.

#### 8.2.4.1 Adding SPI Flash Clients

1. In the **SPI Flash** tab, click the **Add** drop-down list, and then select the client you want to add (see the following figure).


**Figure 8-23. SPI Flash Client Selections**



2. Complete the fields in the dialog box, and then click **OK** (see the following sections).
3. Click the **Apply** button. The client is added to the [SPI Flash clients table](#).

##### 8.2.4.1.1 Settings for Add SPI Bitstream Client

Figure 8-24. Add SPI Bitstream Client Dialog Box


 Add SPI Bitstream Client

Name:

Type:

☒ SPI Bitstream file for IAP

☐ SPI Bitstream file for Recovery/Golden



☐ SPI Bitstream file for Auto Update

☐ Filled with 1s

Content file:

...

Design version:

Start address (HEX):

0x

400

Size in bytes (decimal):

0

Help

OK

Cancel

Table 8-14. Fields in the Add SPI Bitstream Client Dialog Box

Client	Guideline
Name	Name of the SPI bitstream client.

.....continued

Client	Guideline
Type	<p>Type of the SPI bitstream client. Choices are:</p> <ul style="list-style-type: none"> <li>• SPI Bitstream Client for IAP: Adds a SPI Client for IAP. The total number of SPI Bitstream Clients allowed including Recovery/Golden and Auto Update Clients is 255. Index range: 2 - 255</li> <li>• SPI Bitstream Client for Recovery/Golden: Highlighted in yellow in the client table in this tab. Required if a SPI Bitstream is added. There can be only one SPI Bitstream configured as Recovery/Golden. An error message appears if none is configured or more than one is configured. Index 0 is reserved for this client. <ul style="list-style-type: none"> <li>– If Auto Update is enabled, the SPI Bitstream Client for Recovery/Golden must have a Design Version smaller than the Design Version for the SPI Bitstream Client for Auto Update.</li> <li>– Do not use the master file for Recovery/Golden client with IAP.</li> <li>– If Back Level Protection is enabled in the Configure Security tool, Programming Recovery fails if the Back Level Version programmed in the device is greater than or equal to the Design Version of the SPI Bitstream Client for Recovery/Golden.</li> <li>– To allow for programming Recovery to pass, import a Bitstream that has been exported with the Bypass Back Level Protection option.</li> </ul> <p><b>Note:</b> Bypass Back Level protection feature is supported only for SPI Bitstream clients for Recovery/Golden.</p> </li> <li>• SPI Bitstream Client for Auto Update: Highlighted in green in the client table in this tab. To add a SPI Client for Auto Update, check the Enable Auto Update check box in this tab. This client is optional. The Design Version of this client must be greater than the Design Version for the SPI Bitstream Client for Recovery/Golden. Index 1 is reserved for this client. <p><b>Note:</b> The tool rejects a Bitstream file with Bypass Back Level Protection enabled for this type of client.</p> </li> <li>• Filled with 1s: Populates the SPI bitstream client with one's.</li> </ul>
Start address	Start address, in hexadecimal notation, of the SPI bitstream client. sNVM client address starts at page boundaries.
Size in bytes	Size, in bytes, of the SPI bitstream client.

#### 8.2.4.1.2 Settings for Add Data Storage Client

Figure 8-25. Add Data Storage Client Dialog Box

Table 8-15. Fields in the Add Data Storage Client Dialog Box

Client	Guideline
Name	Name of the SPI flash data storage client.
Content	Choices are: <ul style="list-style-type: none"> <li>Intel = Hex Files (*.hex, *.ihx)</li> <li>Filled with 1s: Populates the data storage client with one's.</li> </ul>
Start address	Start address, in hexadecimal notation, of the data storage client.
Size in bytes	Size of bytes, in decimal notation, of the data storage client.

#### 8.2.4.1.3 SPI Flash Clients Table

The SPI Flash clients table shows the SPI Flash clients you add.

Each SPI Flash client appears on its own row. After you add a SPI Flash client, you can select it in this table to [edit](#) or [delete](#) the client.

Figure 8-26. SPI Flash Clients Table

Table 8-16. Columns in the SPI Flash Clients Table

Column	Description
Program	Check boxes for selecting clients that will be enabled or disabled for programming. Clients whose content is filled with 1s cannot be enabled for programming.
Name	Name you gave to the client.
Type	Type of client. Choices are: <ul style="list-style-type: none"> <li>• SPI Bitstream for Recovery/Golden</li> <li>• SPI Bitstream for IAP</li> <li>• SPI Bitstream for Auto Update</li> <li>• Data Storage</li> </ul>
Index	<ul style="list-style-type: none"> <li>• Index 0 is reserved for SPI Bitstream for Recovery/Golden.</li> <li>• Index 1 is reserved for auto update.</li> </ul> <p>The index for an IAP client can be in the range of 2 - 255.</p>
Start Address	Starting address you gave to the client.
End Address	Ending address that Libero SoC ascertained based on the start address you provided.
Content	Choices are: <ul style="list-style-type: none"> <li>• .spi file: spi bitstream clients.</li> <li>• intel-hex(.hex or .ihx): data storage clients.</li> </ul>
Design Version	Client design version.
Bypass Back Level Protection for Recovery/Golden bitstream	This feature is enabled for only the SPI Bitstream clients for Recovery/Golden.

#### 8.2.4.2 Editing SPI Flash Clients

If you need to change the settings for a SPI Flash client, you can edit the client.

To edit a SPI Flash client:

1. In the table of the **SPI Flash** tab, perform one of the following steps:
  - Double-click the client you want to edit.
  - Click the client you want to edit, and then click the **Edit** button.
  - Right-click the client you want to edit, and select **Edit**.

2. When the dialog box appears, complete the fields (see [8.2.4.1.1 Settings for Add SPI Bitstream Client](#) or [8.2.4.1.2 Settings for Add Data Storage Client](#)).
3. Click **OK**.
4. Click the **Apply** button.

### 8.2.4.3 Deleting SPI Flash Clients

If you no longer need a SPI Flash client, you can delete the client.



A warning message does not appear before you delete a client. Therefore, be sure you no longer need a client before you delete it.

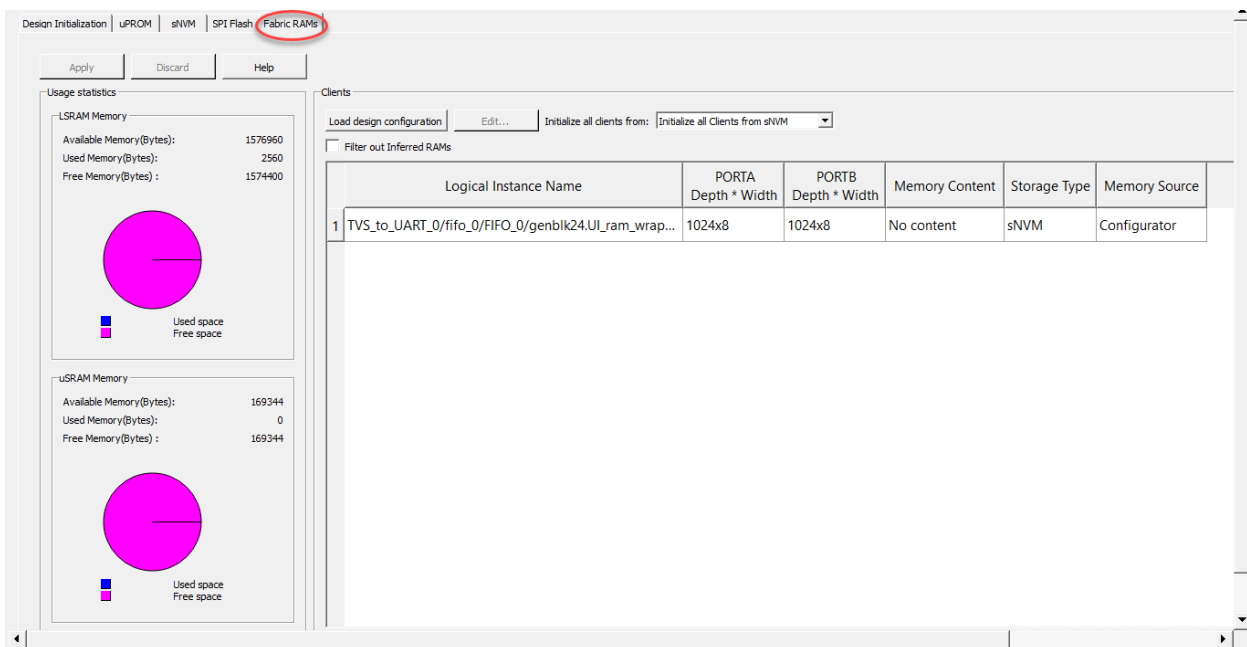
To delete a SPI Flash client:

1. In the table at the bottom of the **SPI Flash** tab, perform one of the following steps:
  - Click the client you want to delete, and then click the **Delete** button.
  - Right-click the client you want to delete, and select **Delete**.
2. Click the **Apply** button.

### 8.2.5 Fabric RAMs Tab

The **Fabric RAMs** tab allows you to select initialization options for Dual-Port SRAM, Two-Port SRAM, and µSRAM memory blocks in your design.

**Figure 8-27. Fabric RAMs Tab**



The following table describes the elements in the **Flash RAMs** tab.

**Table 8-17. Elements in the Flash RAMs Tab**

Element	Description
Load design configuration button	Resets all Fabric RAM clients to the initial configuration that was in effect the first time you clicked Apply. Clicking this button overrides any subsequent commands you applied and resets the Fabric RAM clients table.
Edit button	<a href="#">Edits Fabric RAMs clients.</a>

.....continued

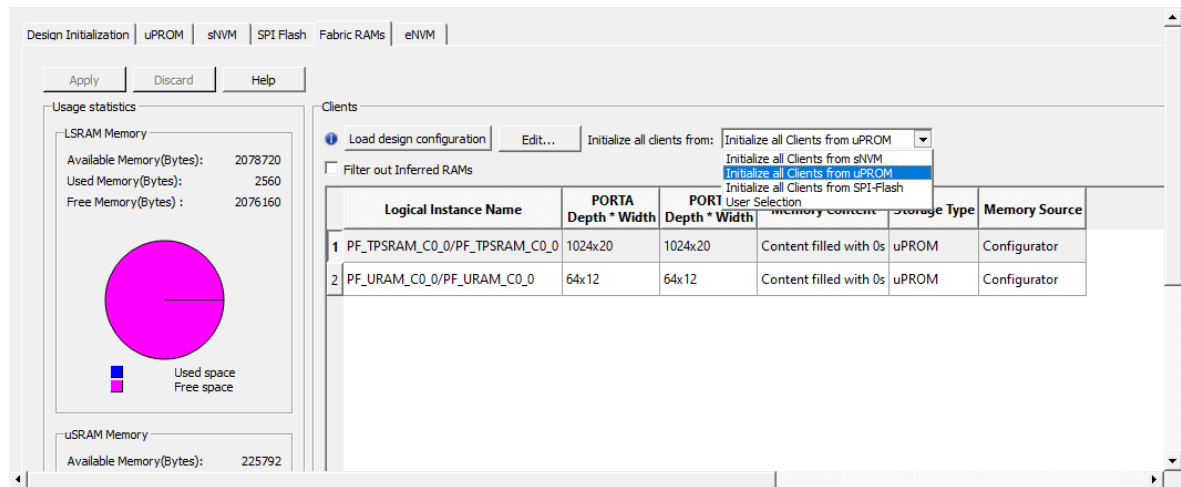
Element	Description
Initialize all clients from drop-down list	<a href="#">Initializes clients from SmartDesign.</a>
Filtered out Inferred RAMs check box	Filters out the inferred RAMs and shows only the RAMs that were generated using the Configurator.
Usage Statistics, LSRAM pie chart	Shows available, used, and free memory, in bytes, for large static random access memory (LSRAM).
Usage Statistics, uSRAM pie chart	Shows available, used, and free memory, in bytes, for micro SRAMs (uSRAM).

### 8.2.5.1 Initializing Fabric RAM Clients

1. In the **Fabric RAMs** tab, click the **Initialize all clients from** drop-down list, and then select the client you want to initialize (see the following figure). Your selection appears in the **Storage Type** column in the [Fabric RAMS client table](#).

**Note:** Selecting **User Selection** indicates that each Fabric RAM client will be configured separately.

**Figure 8-28. Fabric RAMs Client Selections**



#### 8.2.5.1.1 Fabric RAMs Clients Table

The Fabric RAMs clients table shows the Fabric RAM clients you initialize.

Each Fabric RAM client appears on its own row. After you add a Fabric RAM client, you can select it in this table to [edit the client](#).

Figure 8-29. Fabric RAMs Clients Table

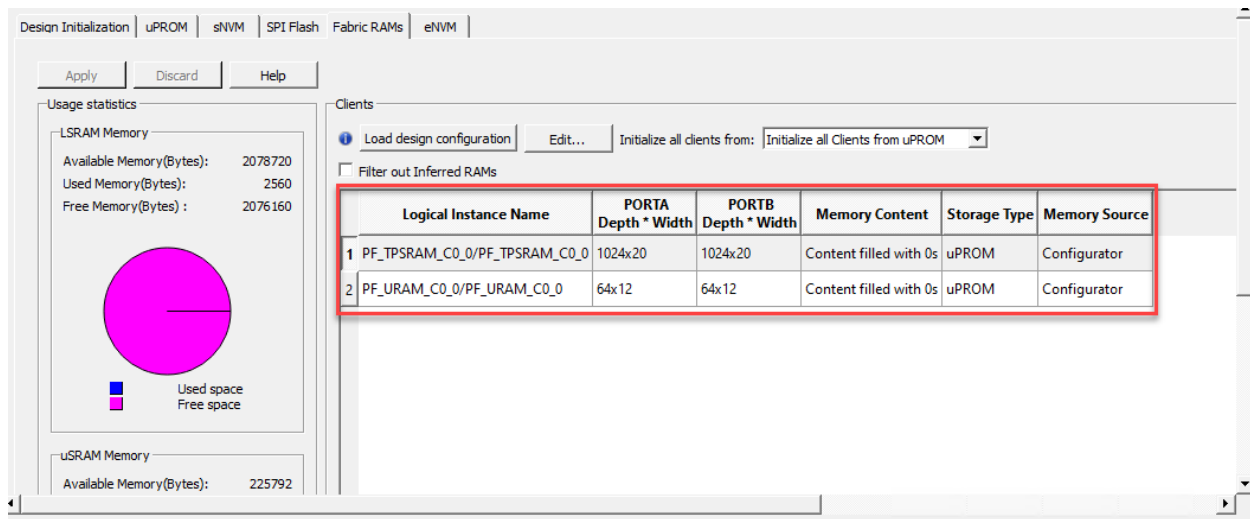


Table 8-18. Columns in the Fabric RAMs Clients Table

Column	Description
Logical Instance Name	Name of the logical instance that indicates the memory type.
PortA Depth*Width	Depth and width of Port A.
PortB Depth*Width	Depth and width of Port B.
Memory Content	Shows information about the content held in memory.
Storage Type	Storage type you selected for the client.
Memory Source	Shows the memory source (for example, Synthesis, Configurator, and so on).

### 8.2.5.2 Editing Fabric RAM Clients

If you need to change the settings for a Fabric RAM client, you can edit the client.

To edit a Fabric RAM client:

- In the table of the **Fabric RAM** tab, perform one of the following steps:
  - Double-click the client you want to edit.
  - Click the client you want to edit, and then click the **Edit** button.
  - Right-click the client you want to edit, and select **Edit**.
- When the Edit Fabric RAM Initialization dialog box appears, complete the fields (see the following figure and table).
- Click **OK**.
- Click the **Apply** button.



Figure 8-30. Edit Fabric RAM Initialization Client Dialog Box

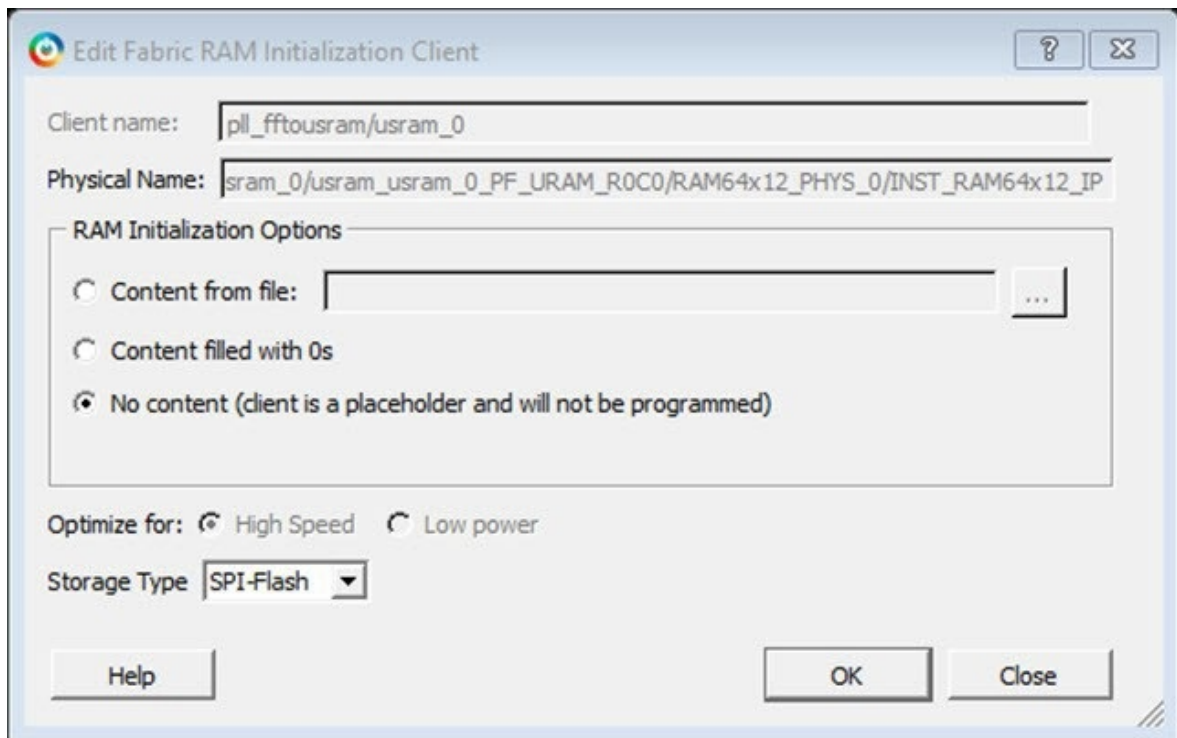


Table 8-19. Fields in the Edit Fabric RAM Initialization Client Dialog Box

Option	Description
Client name	Read-only field that shows the name of the client.
Physical Name	Read-only field that shows the physical name of the client.
RAM Initialization Options	<p>Choices are:</p> <ul style="list-style-type: none"> <li>Content from file: Click the Browse button to go to the location of a memory file, and then import the file to the memory block. By default, the same memory file specified in the memory configurator is used. Supported memory file formats are: <ul style="list-style-type: none"> <li>Intel-HEX (*.hex)</li> <li>Motorola (*.s)</li> <li>Simple-Hex (*.shx)</li> <li>Microsemi-Binary (*.mem)</li> </ul> </li> <li>Content Filled with 0s: Memory block is filled with zeros for initialization.</li> <li>No Content: Memory block is not initialized.</li> </ul>
Optimize for	Read-only field.

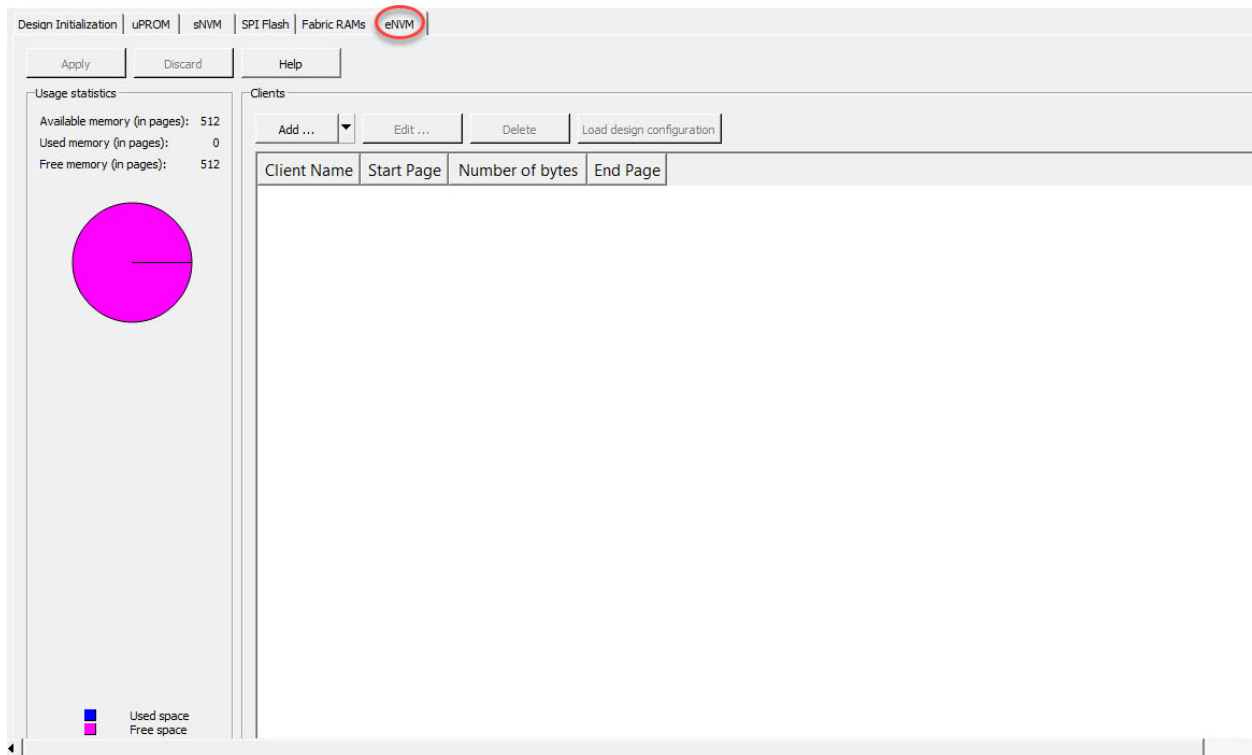
.....continued	
Option	Description
Storage Type	<p>If you change the storage type for a client to a selection other than the one previously chosen for all clients, the Initialize all clients from value also changes. Choices are:</p> <ul style="list-style-type: none"> <li>• sNVM</li> <li>• uPROM</li> <li>• SPI-Flash (<i>default</i>)</li> </ul>

### 8.2.6 eNVM Tab (PolarFire SoC Only)

PolarFire SoC users can use the **eNVM** tab to manage and configure eNVM clients. Libero SoC supports 512 pages.

The procedures for adding, editing, and deleting eNVM clients are similar as those for adding, editing, and deleting sNVM clients, except that you can add only Add Plaintext Boot Mode 1 and Add Plaintext Boot Mode 3.

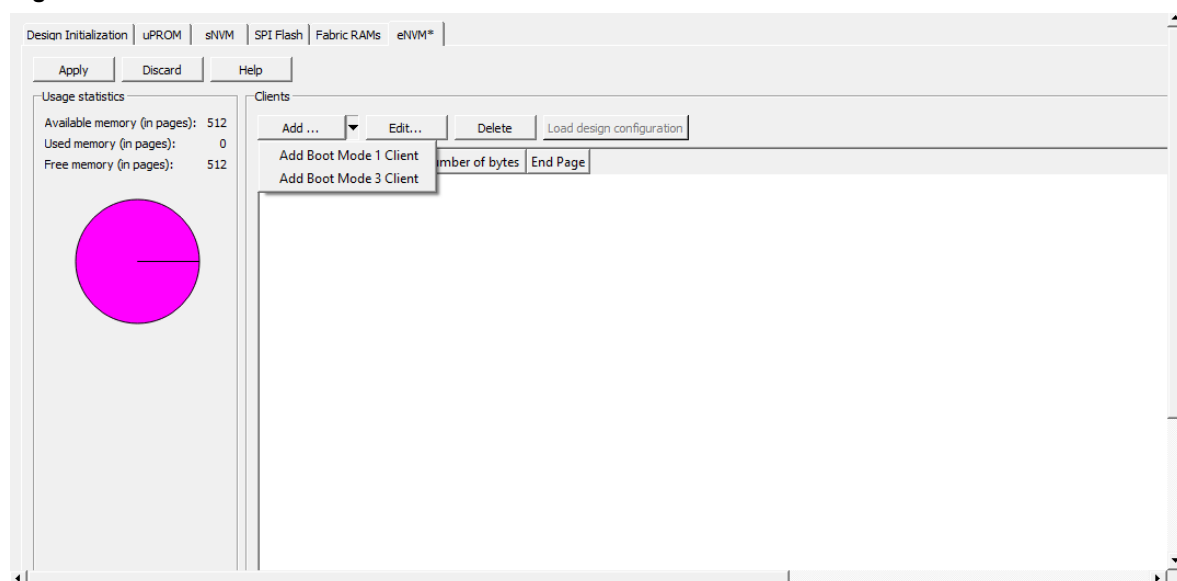
**Figure 8-31. eNVM Tab**



#### 8.2.6.1 Adding eNVM Clients

1. In the **eNVM** tab, click the **Add** drop-down list, and then select the client you want to add (see the following figure).

Figure 8-32. eNVM Client Selections



2. Complete the fields in the dialog box, and then click **OK** (see the following sections).
3. Click the **Apply** button. The client is added to the [eNVM clients table](#).

#### 8.2.6.1.1 Settings for Boot Mode 1 Clients

In this boot mode, the Core Complex boots directly from a specified address in eNVM with no authentication. The Start Page of of Boot Mode 1 client cannot be modified.

**Note:** eNVM memory can have one client only: Boot Mode 1 or [Boot Mode 3](#).

Figure 8-33. Boot Mode 1 Client Dialog Box

Table 8-20. Fields in the Boot Mode 1 Client Dialog Box

Field	Description
Client name	Read-only field that shows the name of the eNVM client.
Content from file	Navigate to and specify a file whose content will be used to fill the eNVM. <b>Note:</b> If you select the Intel-HEX format, the Base address specified is subtracted from user address records. Intel-Hex files have Extended Linear and Extended Segment addresses. The Complete Starting address of the Linear or Segment address in the Hex file must be specified. For example, if the Intel-Hex file has the Extended Linear address 2022, specify the base address 20220000.
Format	Memory file types. Choice is Intel-Hex. The Intel-Hex file is generated using Soft Console.
Base Address	Read-only field that shows the base address that is subtracted from the user address records for Intel-Hex files.

.....continued	
Field	Description
Start page	Read-only field that shows the start page, in decimal notation, of the eNVM client. eNVM client address starts at page boundaries. If there are multiple eNVM clients, their start page cannot be the same; otherwise, a warning message appears. Range: 0 – 220 (decimal)
Number of bytes	Read-only field that shows the total number of bytes to populate the eNVM. The value is shown in decimal notation. If the number of bytes exceeds the memory size of the eNVM, an out-of-bounds warning message appears. Range: 1 – 47376

#### 8.2.6.1.2 Settings for Boot Mode 3 Clients

In this boot mode, you specify the start page in eNVM. You must specify the public keys x and y, along with SBIC reset.

**Note:** An eNVM client can have one client only: [Boot Mode 1](#) or Boot Mode 3.

**Figure 8-34. Boot Mode 3 Client Dialog Box**

Table 8-21. Fields in the Boot Mode 3 Client Dialog Box

Field	Description
Client name	Read-only field that shows the name of the eNVM client.
Content from file	Navigate to and specify a file whose content will be used to fill the eNVM. <b>Note:</b> If you select the format Intel-HEX, the Base address specified is subtracted from user address records. Intel-Hex files have Extended Linear and Extended Segment addresses. The Complete Starting address of the Linear or Segment address in the Hex file must be specified. For example, if the Intel-Hex file has the Extended Linear address 2022, specify the base address 20220000.
Format	Memory file types. Choice is Intel-Hex. The Intel-Hex file is generated using Soft Console.
Base Address	Read-only field that shows the base address that is subtracted from the user address records for Intel-Hex files.
Start page	Read-only field that shows the start page, in decimal notation, of the eNVM client. eNVM client address starts at page boundaries. If there are multiple eNVM clients, their start page cannot be the same; otherwise, a warning message appears. Range: 0 – 220 (decimal)
Number of bytes	Read-only field that shows the total number of bytes to populate the eNVM. The value is shown in decimal notation. If the number of bytes exceeds the memory size of the eNVM, an out-of-bounds warning message appears. Range: 1 – 47376
Public Key X	A unique secret number is generated and known only to the generated person. For more information, see <a href="http://x9.org/">x9.org/</a> .
Public Key Y	A number that corresponds to a private key, but does not need to be kept secret. The public key can be used to determine whether a signature is genuine without requiring the private key to be divulged. For more information, see <a href="http://x9.org/">x9.org/</a> .

#### 8.2.6.1.3 eNVM Clients Table

The eNVM clients table shows the eNVM clients you initialize.

Each eNVM client appears on its own row. After you add an eNVM client, you can select it in this table to [edit the client](#).

Figure 8-35. eNVM Clients Table

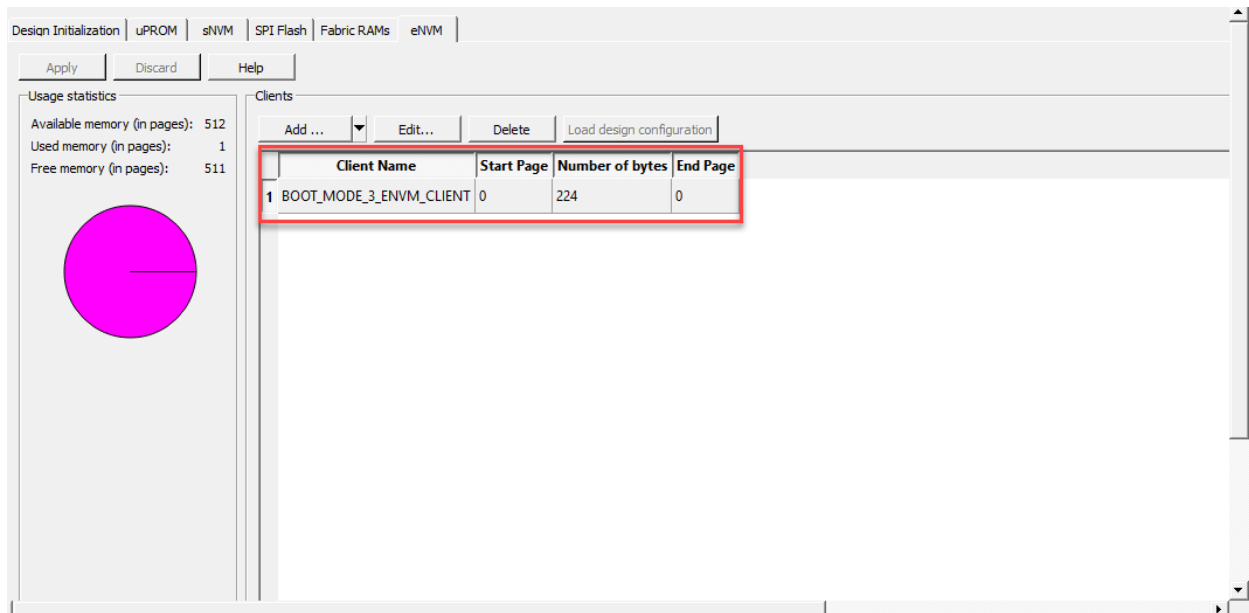


Table 8-22. Columns in the eNVM Clients Table

Column	Description
Client Name	Name you gave to the client.
Start Page	Starting page you gave to the client.
Number of bytes	Number of bytes in the client.
End page	Ending page that Libero SoC ascertained based on the start page you provided.

### 8.2.6.2 Editing eNVM Clients

If you need to change the settings for an eNVM client, you can edit the client.

To edit an eNVM client:

- In the table of the **eNVM** tab, perform one of the following steps:
  - Double-click the client you want to edit.
  - Click the client you want to edit, and then click the **Edit** button.
  - Right-click the client you want to edit, and select **Edit**.
- When the dialog box appears, complete the fields (see [8.2.6.1.1 Settings for Boot Mode 1 Clients](#) or [8.2.6.1.2 Settings for Boot Mode 3 Clients](#)).
- Click **OK**.
- Click the **Apply** button.

### 8.2.7 Deleting eNVM Clients

If you no longer need an eNVM client, you can delete the client.

**CAUTION** A warning message does not appear before you delete a client. Therefore, be sure you no longer need a client before you delete it.

To delete an eNVM client:

- In the table at the bottom of the **eNVM** tab, perform one of the following steps:

- Click the client you want to delete, and then click the **Delete** button.
  - Right-click the client you want to delete, and select **Delete**.
2. Click the **Apply** button.

### 8.3 Generating Initialization Clients

To generate the initialization clients, perform one of the following steps in the Design Flow window:

- Double-click **Generate Design Initialization Data**.  
OR
- Right-click **Generate Design Initialization Data** and choose **Run**.

Either step causes Libero SoC to perform the following actions:

- Generates memory files corresponding to the three stages of the initialization sequence.
- Removes all pre-existing initialization clients.
- Creates initialization clients for each stage and places them in their target memories (see the following table).  
For more information, see [8.2 Initializing Design Blocks](#).

**Table 8-23. Client Initialization Stages**

Stage	Description
First stage initialization client	Client is always placed in sNVM. The default start address is either 0xca00 (page 202), or 0xdb00 (page 219) if <b>Broadcast instructions to initialize RAM's to zero's</b> is disabled in the <b>Design Initialization</b> tab.
Second stage initialization client	Client is created when there are PCIe blocks or Transceiver blocks in the design. The client is always placed in sNVM at the start address you specified in the <b>Design Initialization</b> tab of the Configure Design Initialization Data and Memories tool.
Third stage initialization client	Client is created only when there are Fabric RAMs in the design. The client can be placed in any uPROM, sNVM, or SPI memories at a start address you specify. You specify the target memory and target start address in the <b>Design Initialization</b> tab of the Configure Design Initialization Data and Memories tool.

### 8.4 Configuring Actions and Procedures

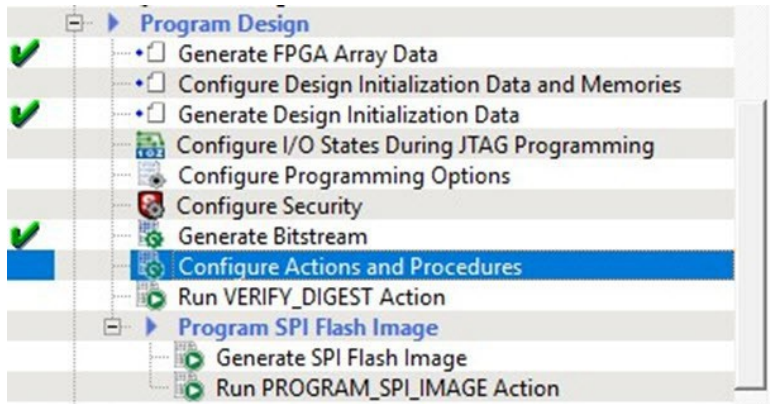
The Configure Actions and Procedures tool allows you to configure actions with optional or recommended procedures for a Libero target device. The information is saved and can be used by the Run Action tool.

Observe the following guidelines when using this tool:

- Available actions and their procedures depend on current bitstream components selected in the Generate Bitstream and Configure Options tools.
- Changing procedures for the action selected to run invalidates the Run Action tool state. Changing any other action does not affect the Run Action tool state.

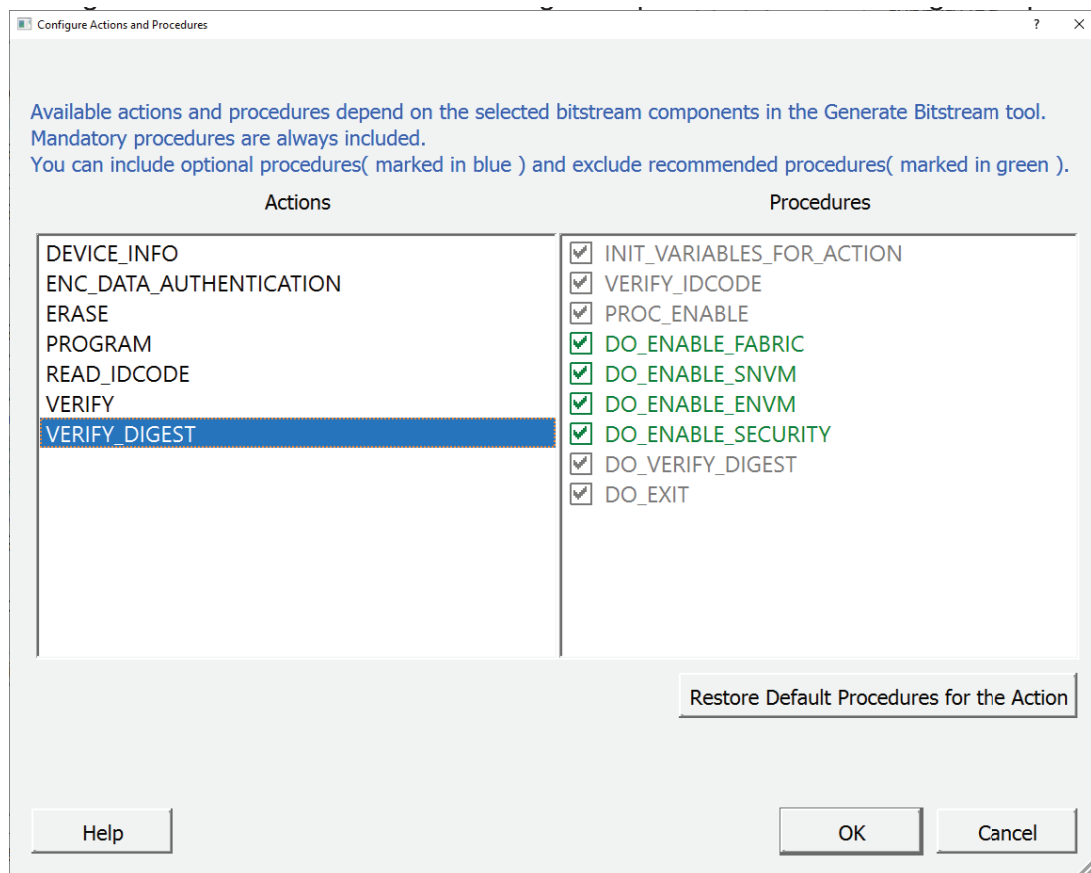
To run the Configure Actions and Procedures tool, from the Libero Design Flow window, expand **Program Design** and double-click **Configure Actions and Procedures**.





The Configure Actions and Procedures dialog box opens. The actions and procedures shown depend on the device family you use and the bitstream components selected in the Generate Bitstream and Configure Options tools.

**Figure 8-36. Configure Actions and Procedures Dialog Box**



#### 8.4.1 Programming File Actions and Supported Procedures

The following table lists programming file actions and supported procedures.

- Mandatory procedures are grayed out and not selectable, and must be performed.
- Recommended procedures are shown in green, and can be included or excluded.
- Optional procedures are shown in blue, and can be included or excluded.

**Table 8-24. Programming File Actions and Supported Procedures**

Action	Procedures
DEVICE_INFO	<ul style="list-style-type: none"> <li>• INIT_VARIABLES_FOR_ACTION</li> <li>• SET_DEVICE_INFO_ACTIONTYPE</li> <li>• VERIFY_IDCODE</li> <li>• DO_DEVICE_INFO</li> <li>• DO_EXIT</li> </ul>
ENC_DATA_AUTHENTICATION	<ul style="list-style-type: none"> <li>• INIT_VARIABLES_FOR_ACTION</li> <li>• SET_AUTHORIZATION_ACTIONTYPE</li> <li>• VERIFY_IDCODE</li> <li>• DO_AUTHENTICATION</li> <li>• DO_EXIT</li> </ul>
ERASE	<ul style="list-style-type: none"> <li>• INIT_VARIABLES_FOR_ACTION</li> <li>• SET_ERASE_ACTIONTYPE</li> <li>• VERIFY_IDCODE</li> <li>• PROC_ENABLE</li> <li>• DO ERASE DO_EXIT</li> </ul>
PROGRAM	<ul style="list-style-type: none"> <li>• INIT_VARIABLES_FOR_ACTION</li> <li>• SET_PROGRAM_ACTIONTYPE</li> <li>• VERIFY_IDCODE</li> <li>• PROC_ENABLE</li> <li>• DO_PROGRAM</li> <li>• DO_VERIFY (optional)</li> <li>• DO_EXIT</li> </ul>
READ_IDCODE	<ul style="list-style-type: none"> <li>• INIT_VARIABLES_FOR_ACTION</li> <li>• SET_READ_IDCODE</li> <li>• VERIFY_IDCODE</li> <li>• PRINT_IDCODE</li> <li>• DO_EXIT</li> </ul>
VERIFY	<ul style="list-style-type: none"> <li>• INIT_VARIABLES_FOR_ACTION</li> <li>• SET_VERIFY_ACTIONTYPE</li> <li>• VERIFY_IDCODE</li> <li>• PROC_ENABLE</li> <li>• DO_VERIFY</li> <li>• DO_EXIT</li> </ul>
VERIFY_DIGEST	<ul style="list-style-type: none"> <li>• INIT_VARIABLES_FOR_ACTION</li> <li>• VERIFY_IDCODE PROC_ENABLE</li> <li>• DO_ENABLE_FABRIC (recommended)</li> <li>• DO_ENABLE_SNVM (recommended)</li> <li>• DO_ENABLE_ENVM(recommended)</li> <li>• DO_ENABLE_SECURITY (recommended)</li> <li>• DO_VERIFY_DIGEST (recommended)</li> <li>• DO_EXIT</li> </ul>

.....continued	
Action	Procedures
ZEROIZE_LIKE_NEW	<ul style="list-style-type: none"> <li>INIT_VARIABLES_FOR_ACTION</li> <li>VERIFY_IDCODE</li> <li>DO_ZEROIZE_LIKE_NEW DO_EXIT</li> </ul>
ZEROIZE_UNRECOVERABLE	<ul style="list-style-type: none"> <li>INIT_VARIABLES_FOR_ACTION</li> <li>VERIFY_IDCODE</li> <li>DO_ZEROIZE_UNRECOVERABLE</li> <li>DO_EXIT</li> </ul>

#### 8.4.2 Programming File Actions and Descriptions

The following table lists programming file actions and descriptions.

**Table 8-25. Programming File Actions**

Action	Description
PROGRAM	Programs all selected family features: <ul style="list-style-type: none"> <li>FPGA Array</li> <li>Targeted sNVM clients</li> <li>Targeted eNVM clients</li> <li>Security settings</li> </ul>
ERASE	Erases the selected family features for: <ul style="list-style-type: none"> <li>FPGA Array</li> <li>Security settings</li> </ul>
VERIFY_DIGEST	Calculates the digests for the components (Custom Security, Fabric, or sNVM or eNVM) included in the bitstream and compares them against the programmed values.
VERIFY	Verifies all selected family features for: <ul style="list-style-type: none"> <li>FPGA Array</li> <li>Targeted sNVM clients</li> <li>Targeted eNVM clients</li> <li>Security settings</li> </ul>
ENC_DATA_AUTHENTICATION	Encrypted bitstream authentication data.
READ_IDCODE	Reads the device ID code from the device.
DEVICE_INFO	Displays the IDCODE, design name, checksum, and device security settings and programming environment information programmed into the device.

#### 8.4.3 Options for Specific Programming Actions

The following table lists the options available for specific programming actions.

To configure actions for other JTAG devices, use the [Programming Connectivity and Interface](#) tool.

**Note:** The eNVM feature is available for PolarFire SoC devices only.

Table 8-26. Programming File Actions - Options

Action	Option	Description
PROGRAM	DO_VERIFY	Enable or disable programming verification.
VERIFY_DIGEST	DO_ENABLE_FABRIC	Include Fabric and Fabric configuration in the digest check.
VERIFY_DIGEST	DO_ENABLE_SNVM	Include the sNVM in the digest check.
VERIFY_DIGEST	DO_ENABLE_ENVM	Include the eNVM in the digest check.
VERIFY_DIGEST	DO_ENABLE_SECURITY	Include security policy settings, and UPK1, UEK1, User Key Set 2 (UPK2 and UEK2), DPK, and SMK security segments in the digest check.

## 8.5 Configuring I/O States During JTAG Programming

You can specify I/O states either [prior to programming](#) or [during programming](#). The following topics describe both methods.

### 8.5.1 Specifying I/O States Prior to Programming

This feature is not supported for PolarFire SoC. The Libero SoC v12.5 hard-codes all I/Os to tri-state after you complete the layout.

In the Libero SoC Design Flow window, expand **Program Design** and double-click **Configure I/O States During JTAG Programming**. The default state for all I/Os is Tri-state.

### 8.5.2 Specifying I/O States During Programming

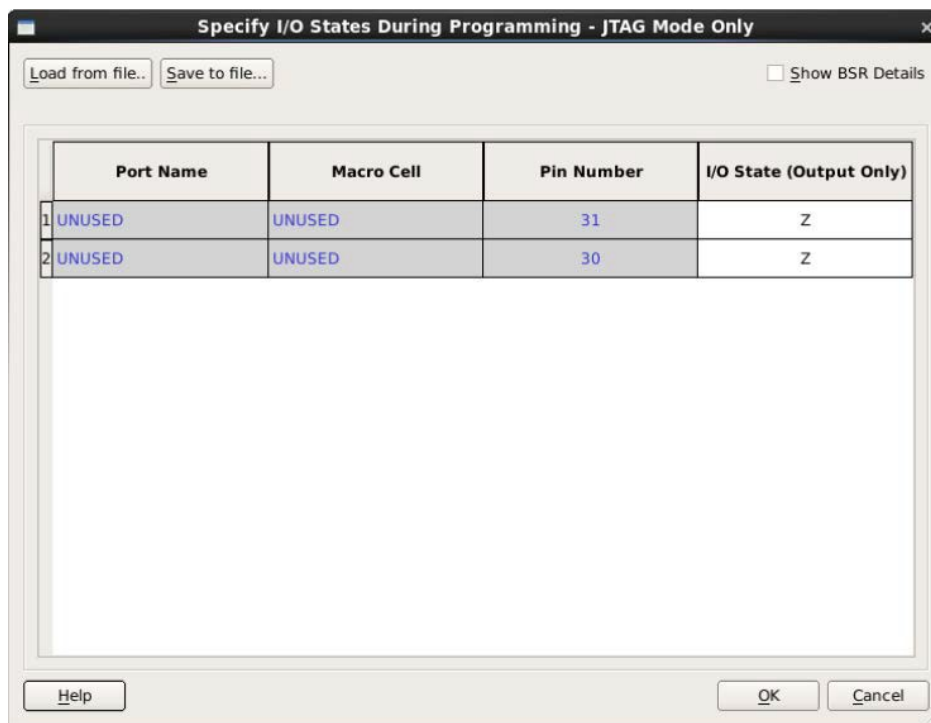
Use the following procedure to specify I/O states during programming or when exporting a bitstream.

1. Click a column header to sort the entries by that header, and then select the I/Os you want to modify.
2. Set the I/O output state using either basic I/O settings to accept default I/O settings for your pins (see the following table) or [custom I/O settings](#) to customize the settings for each pin.

Table 8-27. Basic I/O State Settings

Setting	Description
1	I/O is set to drive out logic High.
0	I/O is set to drive out logic Low.
Last Known State	I/O is set to the last value that was driven out prior to entering the programming mode, and then held at that value during programming.
Z - Tri-State	I/O is tristated with weak pull up (10k $\Omega$ ).

Figure 8-37. I/O States During Programming Window



- Click **OK** to save your settings.

#### 8.5.2.1 Configuring Custom Settings

The I/O States During Programming dialog box allows you to specify custom settings for I/Os in your programming file. This is useful if you want to set an I/O to drive out specific logic, or if you want to use a custom I/O state to manage settings for each Input, Output Enable, and Output associated with an I/O.

Figure 8-38. I/O States During Programming Dialog Box

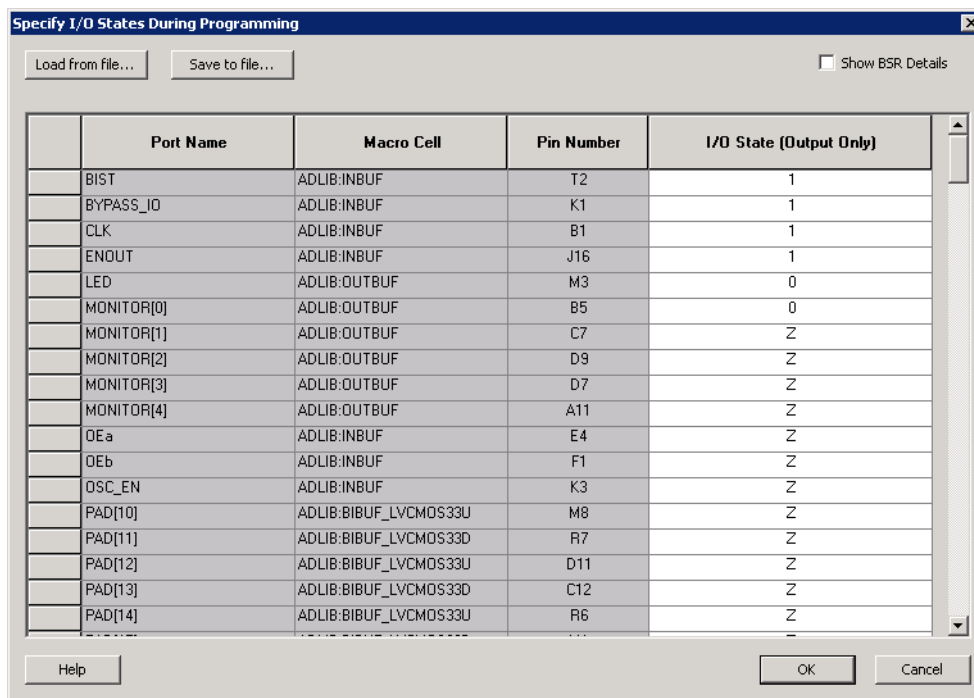


Table 8-28. Elements in the I/O States During Programming Dialog Box

Element	Description
Load from file button	<p>Loads an I/O settings (*.ios) file that you can use to import saved custom settings for your I/Os. The exported IOS file has the following format:</p> <ul style="list-style-type: none"> <li>Used I/Os have a file entry with the following format:  <pre>set_prog_io_state -portName {&lt;design_port_name&gt;} -input &lt;value&gt; - outputEnable &lt;value&gt; - output &lt;value&gt;</pre> </li> <li>Unused I/Os have a file entry with the following format:  <pre>set_prog_io_state -pinNumber {&lt;device_pinNumber&gt;} -input &lt;value&gt; - outputEnable &lt;value&gt; - output &lt;value&gt;</pre> </li> </ul> <p>In the above formats, &lt;value&gt; is one of the values shown in the key in the following table.</p>
Save to file button	Saves your I/O Settings File (*.ios) for future use. This is useful if you set custom states for your I/Os and want to use them again later with a PDC file.
Port Name column	Names of all the ports in your design.
Macro Cell column	I/O types, such as INBUF, OUTBUF, PLLs, and so on.
Pin Number column	Package pin associated with the I/O.
I/O State (Output Only) column	Sets your I/O states during programming (see the Default I/O Output Settings table below). This column header changes to Boundary Scan Register if you select the <b>Show BSR Details</b> check box.
Boundary Scan Registers - Enabled with Show BSR Details	<p>Sets your I/O state to a specific output value during programming and allows you to customize the values for the Boundary Scan Register (Input, Output Enable, and Output). You can change any Don't Care value in Boundary Scan Register States without changing the Output State of the pin (as shown in the BSR Details I/O Output Settings table below).</p> <p>Examples:</p> <ul style="list-style-type: none"> <li>To Tri-State a pin during programming, set Output Enable to 0; the Don't Care indicates that the other two values are immaterial.</li> <li>To have a pin drive a logic High and have a logic 1 stored in the Input Boundary scan cell during programming, set all the values to 1.</li> </ul>

Table 8-29. Default I/O Output Settings

Output State	Settings		
	Input	Control (Output Enable)	Output
Z (Tri-State)	1	0	0
0 (Low)	1	1	0

.....continued

Output State	Settings		
	Input	Control (Output Enable)	Output
1 (High)	0	1	1
Last_Known_State	Last_Known_State	Last_Known_State	Last_Known_State

**Table Key:**

- 1: High: I/Os are set to drive out logic High.
- 0: Low: I/Os are set to drive out logic Low.
- Last\_Known\_State: I/Os are set to the last value that was driven out prior to entering the programming mode, and then held at that value during programming.
- Z: Tri-State: I/O is tristated.

**Table 8-30. BSR Details I/O Output Settings**

Output State	Settings		
	Input	Output Enable	Output
Z (Tri-State)	Don't Care	0	Don't Care
0 (Low)	Don't Care	1	0
1 (High)	Don't Care	1	1
Last Known State	Last State	Last State	Last State

**Table Key:**

- 1 – High: I/Os are set to drive out logic High.
- 0 – Low: I/Os are set to drive out logic Low.
- Don't Care – Don't Care values have no impact on the other settings.
- Last\_Known\_State – Sampled value: I/Os are set to the last value that was driven out prior to entering the programming mode, and then held at that value during programming.
- Z: Tri-State: I/O is tristated.

The following figure shows an example of Boundary Scan Register settings.

Figure 8-39. I/O States During Programming Dialog Box

The dialog box titled "Specify I/O States During Programming" contains a table with the following data:

	Port Name	Macro Cell	Pin Number	Boundary Scan Registers		
				Input	Output Enable	Output
	BIST	ADLIB:INBUF	T2	0	1	1
	BYPASS_IO	ADLIB:INBUF	K1	0	1	1
	CLK	ADLIB:INBUF	B1	0	1	1
	ENOUT	ADLIB:INBUF	J16	0	1	1
	LED	ADLIB:OUTBUF	M3	1	1	0
	MONITOR[0]	ADLIB:OUTBUF	B5	1	1	0
	MONITOR[1]	ADLIB:OUTBUF	C7	1	0	0
	MONITOR[2]	ADLIB:OUTBUF	D9	1	0	0
	MONITOR[3]	ADLIB:OUTBUF	D7	1	0	0
	MONITOR[4]	ADLIB:OUTBUF	A11	1	0	0
	OEa	ADLIB:INBUF	E4	1	0	0
	OEb	ADLIB:INBUF	F1	1	0	0
	OSC_EN	ADLIB:INBUF	K3	1	0	0
	PAD[10]	ADLIB:BIBUF_LVCMOS33U	M8	1	0	0
	PAD[11]	ADLIB:BIBUF_LVCMOS33D	R7	1	0	0
	PAD[12]	ADLIB:BIBUF_LVCMOS33U	D11	1	0	0
	PAD[13]	ADLIB:BIBUF_LVCMOS33D	C12	1	0	0
	PAD[14]	ADLIB:BIBUF_LVCMOS33U	R6	1	0	0

The dialog box also includes buttons for "Load from file...", "Save to file...", "Show BSR Details" (checked), "Help", "OK", and "Cancel".

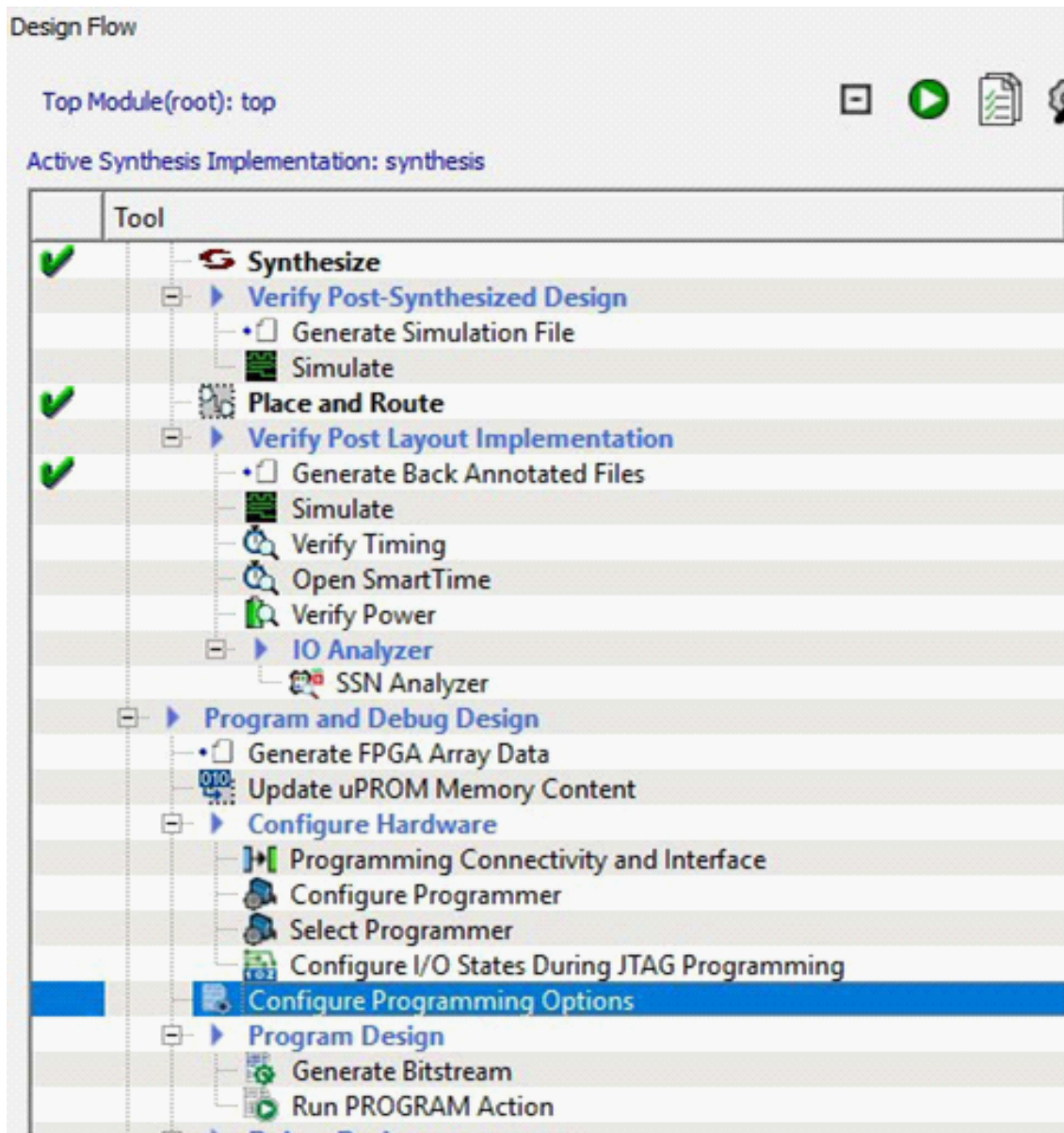
## 8.6 Configuring Programming Options

The program options you can configure depend on the device you are programming. The following topics describe the options available to all product families.

To configure programming options, from the Design Flow window, double-click **Configure Programming Options** or right-click it and choose **Configure Options**. The Configure Programming Options dialog box appears with the appropriate options (see the remaining topics in this section for more information).



Figure 8-40. Selecting Configure Programming Options



### 8.6.1 PolarFire Programming Options

The following figure shows the programming options for PolarFire. The table following the figure describes the options.

**Note:** SPI file programming for Auto Programming, Auto Update (IAP), and IAP/ISP Services can program security only one time with the master file. Update files cannot update the Security settings. In addition, Silicon signature and Tamper Macro can be programmed with the master file only and cannot be updated.

Figure 8-41. PolarFire Programming Options

The screenshot shows a 'Configure Programming Options' dialog box. It has a title bar with a close button. Inside, there are four rows of configuration options:

- Design name:** fftousram\_new (read-only text)
- Design version (number between 0 and 65535):** 3 (text input)
- Back Level version (number between 0 and 65535):** 3 (text input, preceded by a warning icon)
- Silicon signature (max length is 8 HEX chars):** 0x2 (text input)

At the bottom, there are three buttons: 'Help', 'OK', and 'Cancel'.

Table 8-31. PolarFire Programming Options

Option	Description
Design name	Read-only field that identifies your design.
Design version	Design version to be programmed into the device. This value is also used for Back Level protection in the <a href="#">Update Policy</a> step of the Configure Security tool.
Black Level version	Back Level version to be programmed to the device. This value must be less than or equal to the Design version number. This value is used for Back Level protection (if enabled) in the <a href="#">Update Policy</a> step of the Configure Security tool.
Silicon signature	32-bit user configurable silicon signature to be programmed into the device. This field can be read from the device using the JTAG (IEEE 1149-1) USERCODE instruction or by running the DEVICE_INFO programming action.

## 8.7 Configuring Security

The following topics describe how to configure custom security settings using the Configure Security Wizard, and provide answers to frequently asked questions about security.

### 8.7.1 Configure Security Wizard

The Configure Security Wizard is a GUI-based wizard that guides you through the procedure for configuring custom security settings. The wizard consists of the following five steps:

1. [User keys](#)
2. [Update Policy](#)
3. [Debug Policy](#)
4. [Microsemi Policy](#)
5. [JTAG/SPI Slave Commands](#)

The following table describes the elements in the Configure Security Wizard.

Figure 8-42. Configure Security Wizard

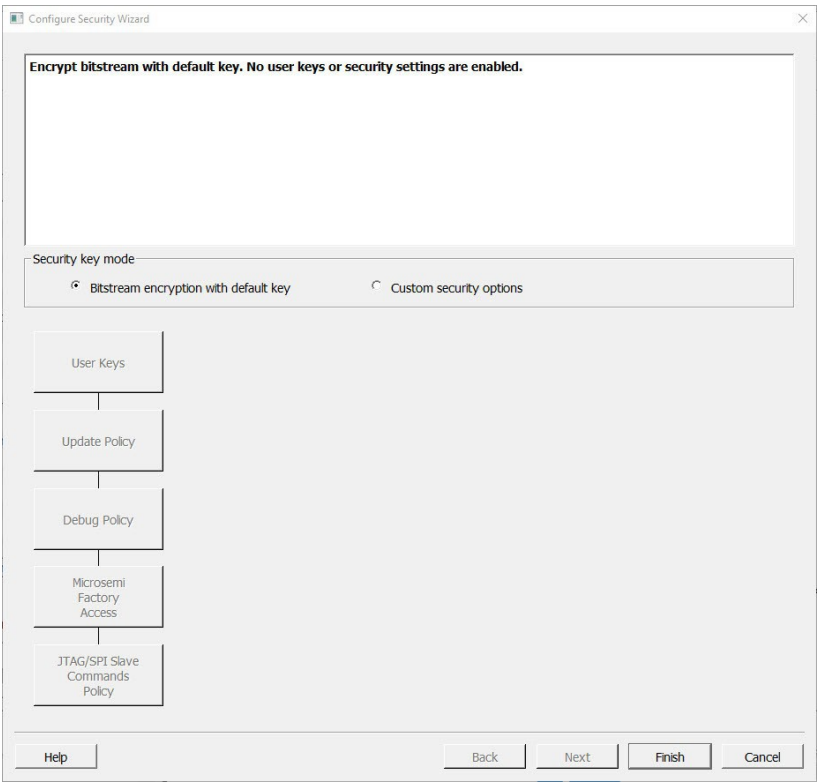


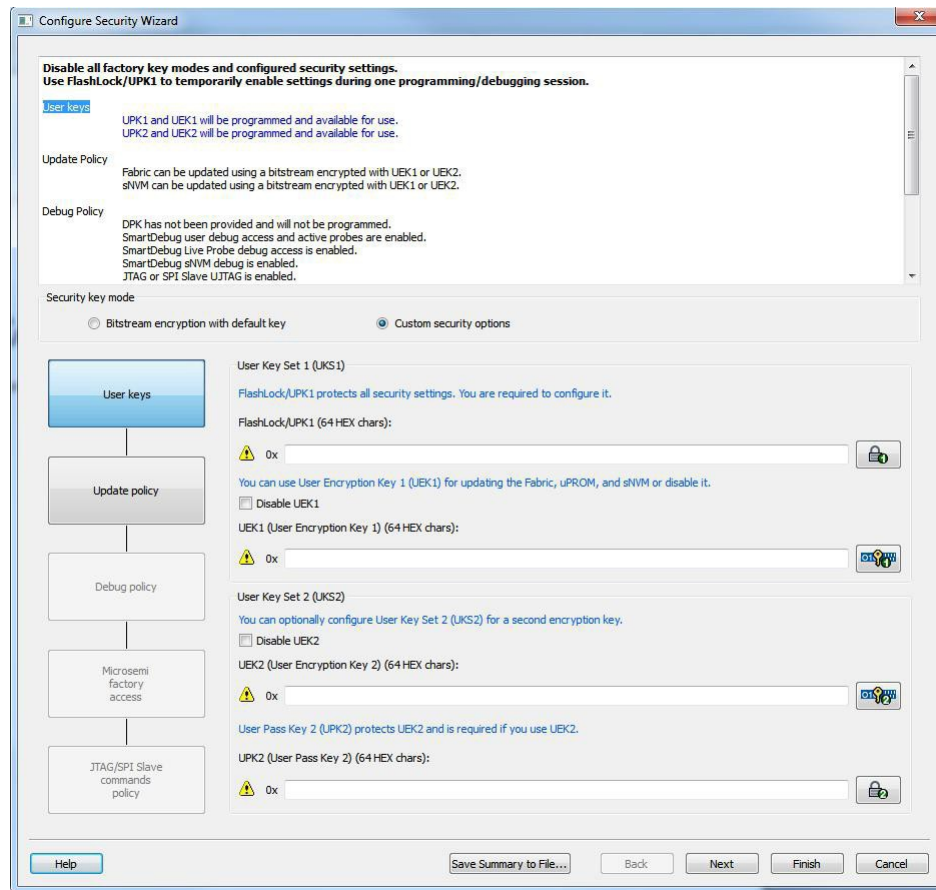
Table 8-32. Elements in the Configure Security Wizard

Element	Description
Summary window	Displays the summary of the current configuration settings. The window scrolls to the current page as you move from page to page.
Security key mode	Two security key modes are available: <ul style="list-style-type: none"><li>• Bitstream encryption with default key: Use the default encryption key for security. The Next and Back buttons are disabled. All steps are disabled. Custom User Keys and security settings are disabled.</li><li>• Custom security Mode: Configures custom security keys and settings. All steps are enabled, as are the Next and Back buttons.</li></ul>
Back button	Click to return to previous step.
Next button	Click to proceed to next step.
Finish button	Click to skip steps and complete the configuration.

8.7.2 Step 1: User Keys

In step 1 of the Configure Security Wizard, you configure user keys. All keys are 256 bits (64 HEX characters).

### Figure 8-43. Configure Security Wizard - User Keys



The following table describes the options in this step.

### Table 8-33. Options in the User Keys Step

Option	Description
FlashLock/UPK1	<p>Protects all security settings. This key is required and must be a string of 64 HEX characters.</p> <p>Enter the key or click the padlock icon at the far right to generate a random key.</p> <p>Default: enabled</p>
User Encryption Key 1 (UEK1)	<p>Used for updating the Fabric, uPROM, and sNVM. This key is required and must be a string of 64 HEX characters.</p> <p>Enter the key or click the padlock icon at the far right to generate a random key. To disable it, click Disable.</p> <p>Default: enabled</p>
User Encryption Key 2 (UEK2)	<p>Used as a second encryption key for updating the Fabric, uPROM, and sNVM. This key is required and must be a string of 64 HEX characters.</p> <p>Enter the key or click the padlock icon at the far right to generate a random key. To disable it, click Disable.</p> <p>Default: enabled</p>

.....continued	
Option	Description
User Pass Key 2 (UPK2)	UPK2 is required if UEK2 is enabled. Enter the key or click the padlock icon at the far right to generate a random key.

### 8.7.3 Step 2: Update Policy

In step 2 of the Configure Security Wizard, you disable field updates and specify field-update protection parameters. Field updates are enabled by default.

**Figure 8-44. Configure Security Wizard - Update Policy**

The following table describes the options in this step.

Table 8-34. Options in the Update Policy Step

Option	Description
Fabric/sNVM update protection	<p>Choices are:</p> <ul style="list-style-type: none"> <li>Disable Erase/Write operations.</li> </ul> <p><b>Note:</b> The field-update STAPL files _uek1 and _uek2 include plain-text FlashLock/UPK1.</p> <ul style="list-style-type: none"> <li>Updates allowed using user-defined encryption keys: FlashLock/UPK1 is not required for updates.</li> </ul>
eNVM Update Protection	(PolarFire SoC only) Updates allowed using user-defined encryption keys; FlashLock/UPK1 is not required for updates.
Enable Back Level protection	<p>When checked, a field update design being programmed must be a version higher than the Back Level version value in the programmed device. This safeguard prevents field update designs with back level versions less than or equal to the design version programmed in the device.</p> <ul style="list-style-type: none"> <li>Design version (number between 0 to 65535): Display the current Design version set in the Configure Programming Options tool.</li> <li>Back Level version (number between 0 to 65535): Display the current Back Level version set in the Configure Programming Options tool). Back Level version uses the Design version value to determine which bitstreams are allowed for programming. The Back Level version must be smaller than or equal to Design version.</li> </ul> <p><b>Note:</b> If Back Level Protection is disabled and Back Level version is greater than zero, Generate Bitstream and Export Bitstream tools error out.</p> <p>The examples in the following tables show Back Level protection enabled in the Configure Security tool.</p>
Disable Auto Programming and IAP Services	When this option is selected, Auto Programming, Auto Update, IAP Services, and Programming Recovery are disabled. FlashLock/UPK1 unlocking is only available for JTAG and SPI Slave interfaces.
Disable programming interfaces	<p>You can disable the following programming interfaces:</p> <ul style="list-style-type: none"> <li>JTAG</li> <li>SPI Slave</li> </ul> <p><b>Note:</b> You cannot disable Auto Programming and IAP Services and both the programming interfaces. If you try, an error message appears.</p>

.....continued	
Option	Description
Disable bitstream programming actions (JTAG/SPI Slave)	<p>Choices are:</p> <ul style="list-style-type: none"> <li>• Program</li> <li>• Authenticate</li> <li>• Verify</li> </ul> <p><b>Note:</b> The field-update STAPL files _uek1 and _uek2 include plain-text FlashLock/UPK1. The summary at the top of the wizard summarizes the result of the selection.</p>
Reset to default	Reset the options to default values. All options are unchecked.

**Table 8-35. Example 1: Programming the Back Level Version to the Same Version as the Design Version**

Step	Bitstream	Action	Bitstream Design version	Bitstream Back Level version	Device Back Level version	Result
1	_master	PROGRAM	1	1	1	Pass
2	_master	VERIFY	1	1	1	Pass if device has UPK1
3	_master	ERASE	1	1	1	Pass if device has UPK1
4	_master	AUTHENTICATE	1	1	1	Pass if device has UPK1
5	_master	DEVICE_INFO	1	1	1	Always passes
6	_uek1	PROGRAM	2	2	2	Pass
7	_master	PROGRAM	1	1	2	Fail due to back level protection

**Table 8-36. Example 2: Programming the Back Level Version Less Than the Design Version**

Step	Bitstream	Action	Bitstream Design version	Bitstream Back Level version	Device Back Level version	Result
1	_master	PROGRAM	2	1	1	Pass
2	_uek1	PROGRAM	3	1	1	Pass
3	_uek1_1	PROGRAM	4	1	1	Pass
4	_uek1_2	PROGRAM	5	4	4	Pass
5	_master	PROGRAM	2	1	4	Fail due to back level protection
6	_uek1	PROGRAM	3	1	4	Fail due to back level protection



.....continued

Step	Bitstream	Action	Bitstream Design version	Bitstream Back Level version	Device Back Level version	Result
7	_uek1_1	PROGRAM	4	1	1	Fail due to back level protection
8	uek1_2	VERIFY	5	4	4	Pass

#### 8.7.4 Step 3: Debug Policy

Debugging is enabled by default. Use this page to configure Debug Protections.

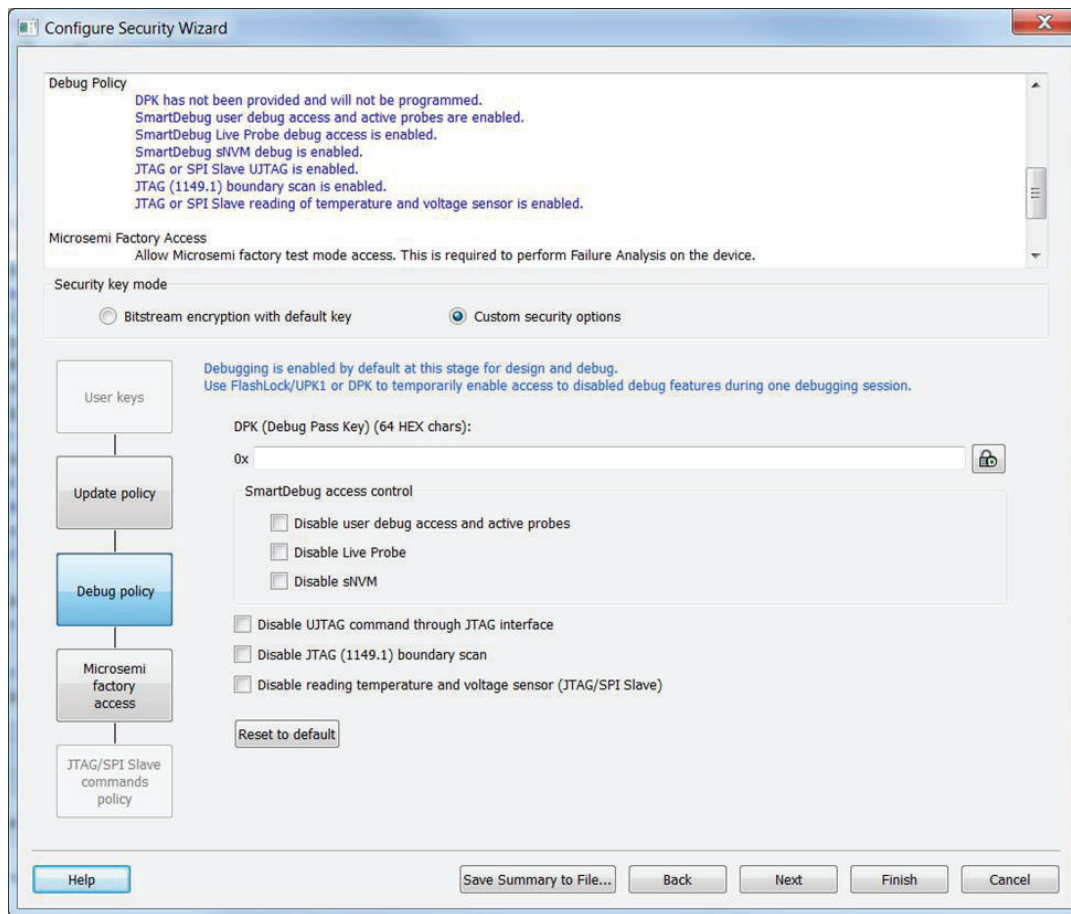



Table 8-37. Elements in Debug Policy Step

Element	Description
Debug with DPK (Debug Pass Key) - Optional	Protect Debug with a 256-bit (64-character HEX) Debug Pass Key. Enter the key in the field or click the padlock icon at the far right to generate a random key. This key is optional if you want a separate passkey to enable access to disabled debug features during one debugging session. If the DPK key is entered, then at least one option must be checked.



.....continued

Element	Description
SmartDebug Access Control	<p>All of the following are enabled by default for SmartDebug access. Check to disable access.</p> <ul style="list-style-type: none"> <li>• Disable User Debug Access and Active Probe</li> <li>• Disable Live Probe</li> <li>• Disable sNVM</li> </ul> <p> <b>WARNING</b> Leaving SmartDebug access control enabled on production devices will allow anyone to debug or access active probes, access Live Probe, or read the content of sNVM.</p> <p>Three additional options are:</p> <ul style="list-style-type: none"> <li>• Disable UJTAG command through JTAG Interface.</li> <li>• Disable JTAG (1149.1) boundary scan. Disables JTAG (1149.1) commands. The following JTAG commands will be disabled: HIGHZ, EXTEST, INTEST, CLAMP, SAMPLE, and PRELOAD. I/Os will be tri-stated when in JTAG programming mode and BSR control during programming is disabled. BYPASS, IDCODE, and USERCODE instructions will remain functional.</li> <li>• Disable reading temperature and voltage sensor (JTAG/SPI Slave). The summary at the top of the page displays the results of the selection.</li> </ul>

#### 8.7.5 Step 4: Microsemi Factory Access Policy

In step 4 of the Configure Security Wizard, you configure the policy for Microsemi test mode access. Field updates are enabled by default. Test mode access is required for failure analysis on the device.

Figure 8-45. Configure Security Wizard - Microsemi Access Policy

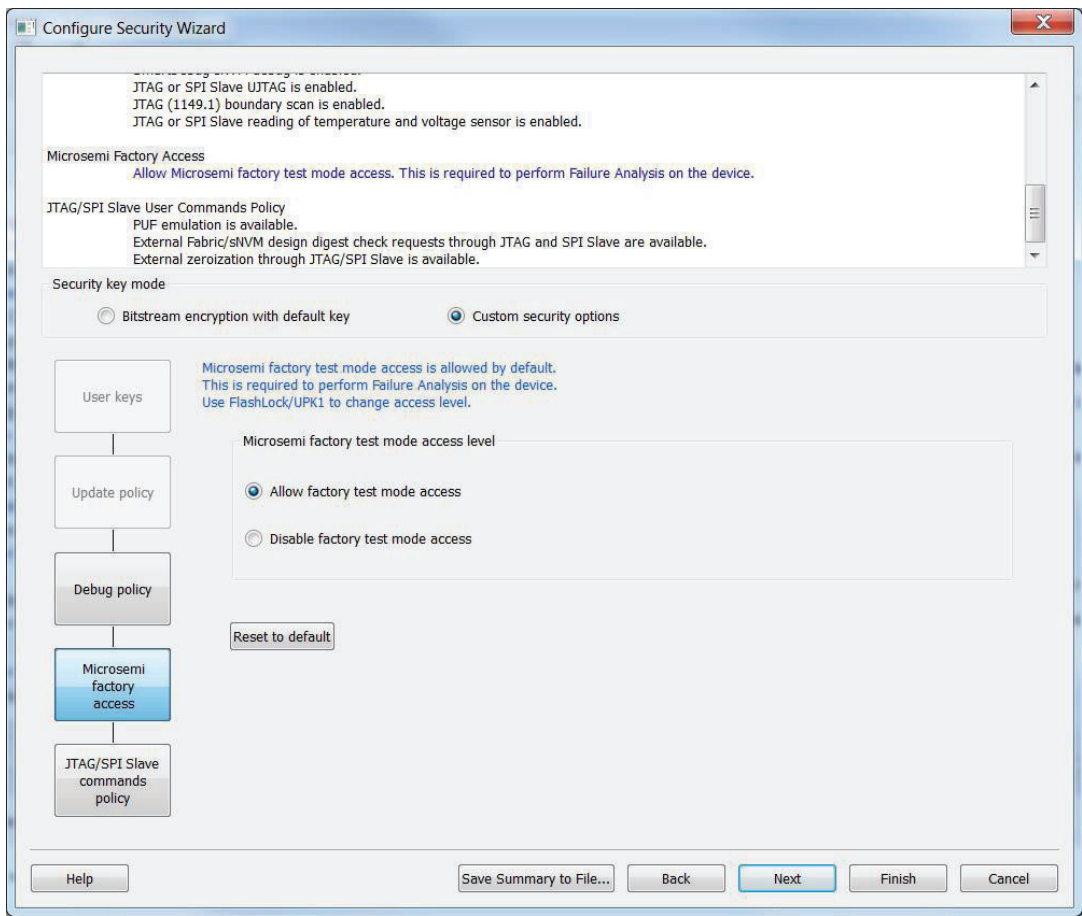


Table 8-38. Option in the Microsemi Access Policy Step

Option	Description
Microsemi factory access level	Choices are: <ul style="list-style-type: none"><li>Allow factory test mode access: <i>(default)</i> <b>Note:</b> Do not use this setting for production devices.</li><li>Disable factory test mode access.</li><li><b>Note:</b> Use FlashLock/UPK1 to change access level.</li></ul>

8.7.6 Step 5: JTAG/SPI Slave Command Policy

In step 5 of the Configure Security Wizard, you configure the policy for JTAG/SPI slave user commands.

Figure 8-46. Configure Security Wizard - JTAG/SPI Slave Commands Policy

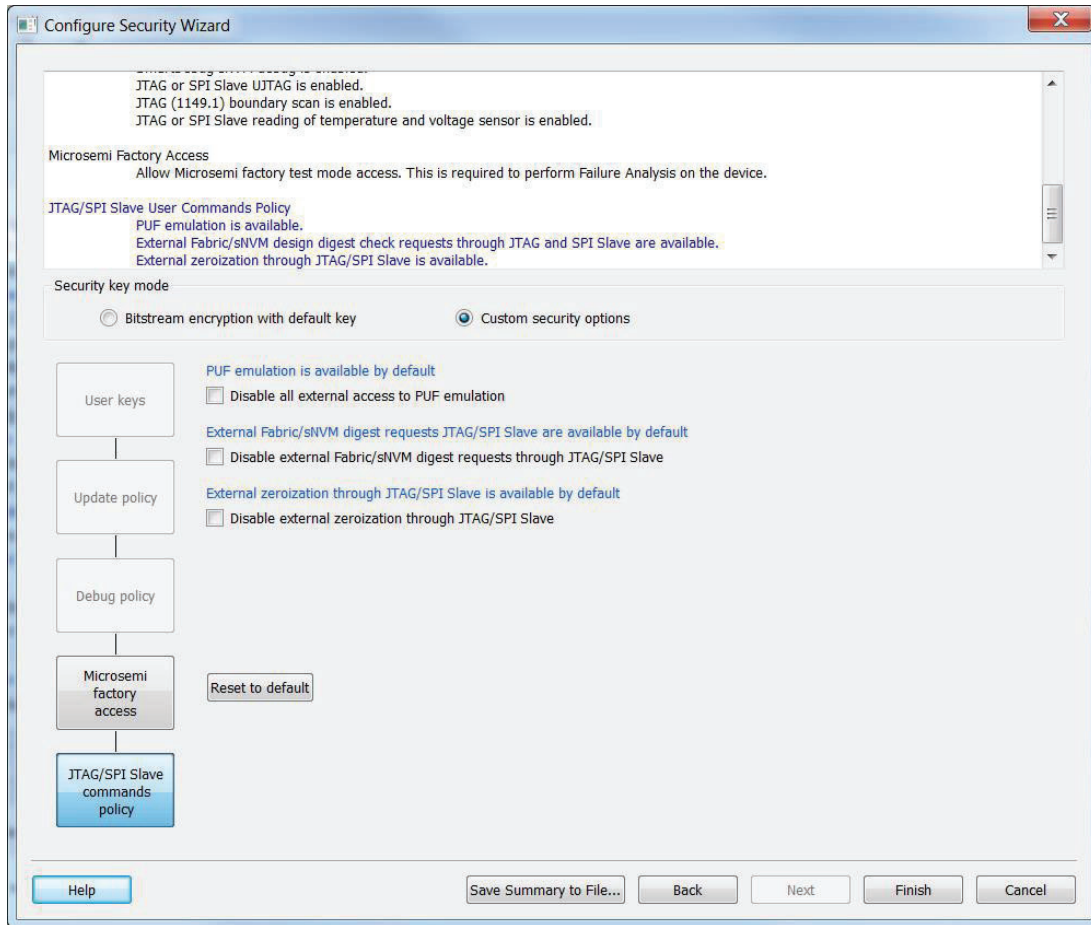




Table 8-39. Options in the JTAG/SPI Slave Command Policy Step

Option	Description
Disable all external access to PUF emulation	<p>Determines whether PUF emulation is available by default. Choices are:</p> <ul style="list-style-type: none"> <li>• Checked: Disable all external access to PUF emulation through the JTAG/SPI slave.</li> <li>• Not checked: Enable all external access to PUF emulation through the JTAG/SPI slave. (<i>default</i>)</li> </ul>
Disable external Fabric/sNVM digest requests through JTAG/SPI Slave	<p>Determines whether external Fabric/sNVM digest requests through the JTAG/SPI slave are available by default. Choices are:</p> <ul style="list-style-type: none"> <li>• Checked: Disable external Fabric/sNVM digest requests through JTAG/SPI slave.</li> <li>• Not checked: Enable external Fabric/sNVM digest requests through JTAG/SPI slave. (<i>default</i>)</li> </ul> <div>  <b>WARNING</b> Repeated external Fabric digest calculations can impact device reliability. For more information, see the datasheet. </div>

.....continued	
Option	Description
Disable external zeroization through JTAG/SPI Slave	<p>Determines whether external zeroization through the JTAG/SPI slave is available by default. Choices are:</p> <ul style="list-style-type: none"> <li>• Checked: Disable external zeroization through JTAG/SPI slave.</li> <li>• Not checked: Enable external zeroization through JTAG/SPI slave. (<i>default</i>)</li> </ul>
	<div style="display: flex; align-items: center;"> <div style="background-color: #ffcc00; padding: 2px 5px; margin-right: 5px;">  <b>WARNING</b> </div> <div>Do not enable zeroization for production devices.</div> </div>

### 8.7.7 Security Features Frequently Asked Questions

**I have configured the Security Wizard and enabled security in my design but I do not want to program my design with the Security Policy Manager features enabled. What do I do?**

Go to Configure Bitstream and uncheck Security.

**What is programmed when I click Program Device?**

All features configured in your design and enabled in the Configure Bitstream tool. Any features you have configured (such as **sNVM** or Security) are enabled for programming by default.

**When I click Program Device is the programming file encrypted?**

All programming files are encrypted. To generate programming files encrypted with UEK1 or UEK2, you must generate them from Export Bitstream for field updates.

**Note:** Once security is programmed, you must erase the security before attempting to reprogram the security.

**How do I generate encrypted programming files with User Encryption Key 1/2?**

- Configure the Security Wizard and specify User Key Set 1, User Key Set 2. Ensure the Security programming feature is enabled in Configure Bitstream; it is enabled by default once you configure the Security Policy Manager.
- Export Bitstream from Handoff Design for Production - <filename>\_uek1.(stp/spi/dat), <filename>\_uek2.(stp/spi/dat) files are encrypted with UEK1, UEK2, respectively. See Security Programming File Descriptions below for more information on programming files.

**What are Security Programming Files?**

See the [Security Programming Files topic](#) for more information.

## 8.8 Configure Permanent Locks for Production

Configure Permanent Locks for Production is a GUI-based tool that guides the user step by step on how to configure the Permanent Locks for Production. The wizard has six steps/pages executed in sequential order. One Time Programmable (OTP) settings in the Permanent Locks page are applied to configured Security settings from the Configure Security tool. The subsequent pages have read only fields, which will be affected by Permanent Lock settings. These settings can only be configured by the [Configure Security](#) tool.

If you configure any Permanent Lock settings, you will be forced to go through each page to review the Security settings to make sure they are as desired. The settings cannot be changed once they have been programmed.

**Note:** This feature is not supported for PolarFire SoC devices.

1. [Permanent Locks](#)
2. [User keys in Configure Security](#)
3. [Update Policy in Configure Security](#)

4. [Debug Policy in Configure Security](#)
5. [Microsemi Factory Access in Configure Security](#)
6. [JTAG/SPI Slave Commands Policy in Configure Security](#)

#### 8.8.1 Summary Window

The summary window displays the summary of the current page configuration settings. Based on the selection made in the first page, the summary for the subsequent pages change. The window will scroll to the current page as you move from page to page.

#### 8.8.2 Back

Click **Back** to return to the previous step.

#### 8.8.3 Next

Click **Next** to proceed to the next step.

#### 8.8.4 Finish

Click **Finish** to complete the configuration after executing the all the steps in sequential order.

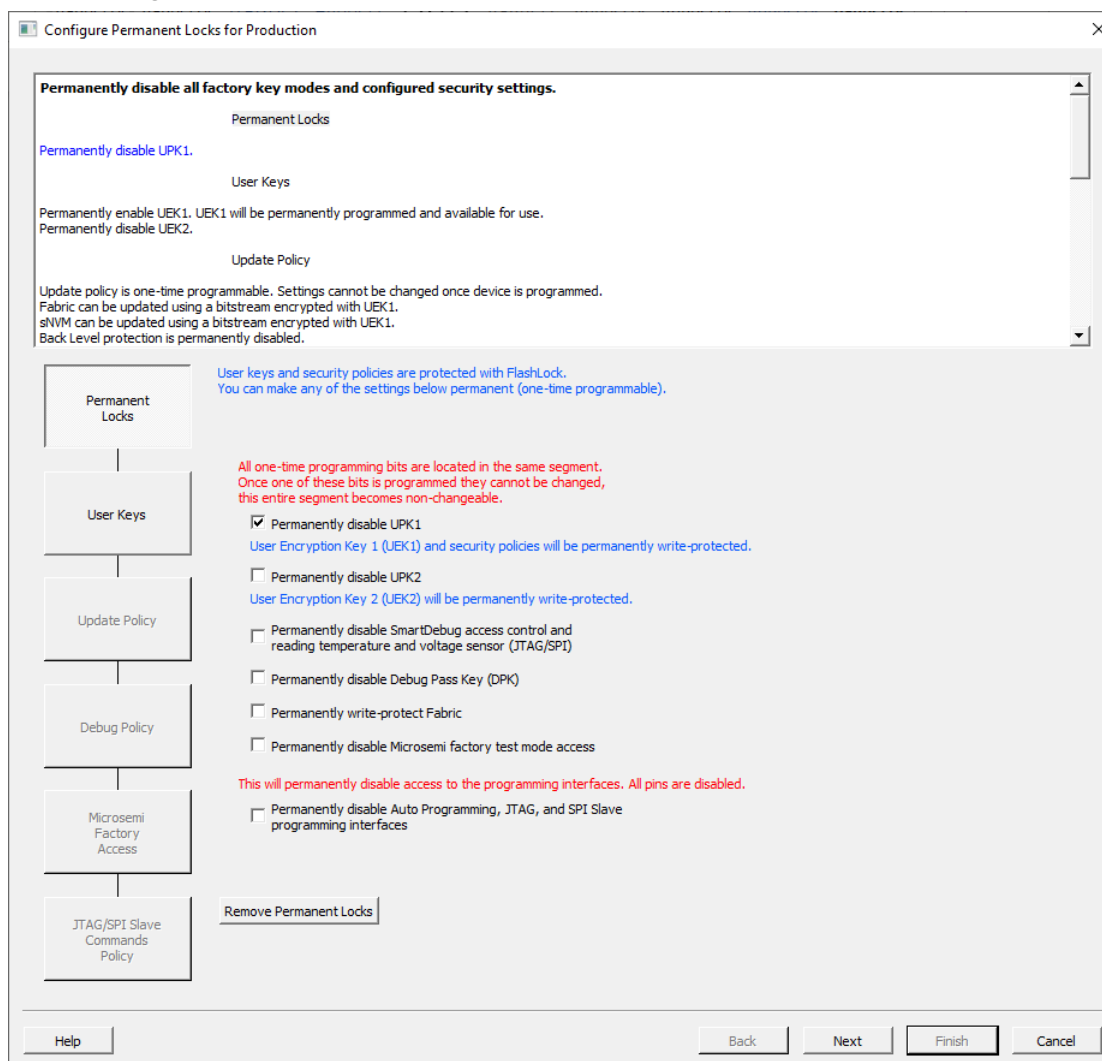
#### 8.8.5 Save Summary to File

Click **Save Summary to File** to save the display in the Summary field to a file.

#### 8.8.6 Permanent Locks

This page allows you to configure Permanent Locks for Production programming. Permanent Locks should be configured after the Design/Debug phase has been completed. The Permanent Lock settings will not be applied to programming within Libero. They are only applied to the Export tools used for Production programming. Once the Permanent Locks are programmed, they cannot be changed. Configuring the Permanent Locks will affect the settings on the subsequent pages and should be reviewed carefully. The settings cannot be changed once they are programmed.

Figure 8-47. Configure Permanent Lock for Production



All the user keys and security policies are protected with FlashLock and can be made One-Time Programmable, by configuring the Permanent Lock settings.

You can select one or more of the below options to be locked permanently:

- **Permanently disable UPK1** - This will permanently disable FlashLock/UPK1 from being able to be matched by the device. Any feature that is disabled will be permanently disabled. Any feature that is available will be permanently available.
- **Permanently disable UPK2** - This will permanently disable FlashLock/UPK2 from being able to be matched by the device. If UEK2 is enabled and selected for programming, then it cannot be changed.
- **Permanently disable SmartDebug access control and reading temperature and voltage sensor** - This will permanently disable SmartDebug access control for user debug and active probes, live probes and, sNVM along with the ability to read the temperature and voltage sensor.
- **Permanently disable Debug Pass Key (DPK)** - This will permanently disable the FlashLock/DPK from being able to be matched by the device. If DPK was programmed, then it can no longer be used for SmartDebug access.
- **Permanently write-protect Fabric** - This will make the Fabric One-Time Programmable. Verify of the Fabric will still be possible. Erase/Program of the Fabric is permanently disabled.
- **Permanently disable Microsemi factory test mode access** - This will permanently disable Microsemi factory test mode access. Microsemi will not be able to perform a Failure Analysis on this device.

- **Permanently disable Auto Programming, JTAG and SPI Slave programming interfaces** - This will permanently disable all programming interfaces. The actual JTAG and SPI Slave ports are disabled and you cannot access the device for any operations including reading the IDCODE of the device. The device will become One-Time Programmable and there will be no way to Erase/Program/Verify the device.

The summary window displays the summary of the current configuration set in Permanent Locks page. The summary can be exported to a file and is recommended to keep a record of the settings configured. It is important to review the summary carefully and ensure the settings and behaviors are as expected since they cannot be changed once the device is programmed.

### 8.8.7 Remove Permanent Locks

You can remove the Permanent Lock settings by either right-clicking on the tool in the Design Tree or by clicking the button in the UI.

When selected removes all the Permanent Locks selected and restores to initial security settings configured in Configure Security tool. This option is highlighted only when at least one of the Permanent Locks is enabled.

#### 8.8.7.1 See Also

[Configure Security](#)

[Tcl command for Permanent Locks remove\\_permanent\\_locks](#)

## 8.9 Configure Bitstream

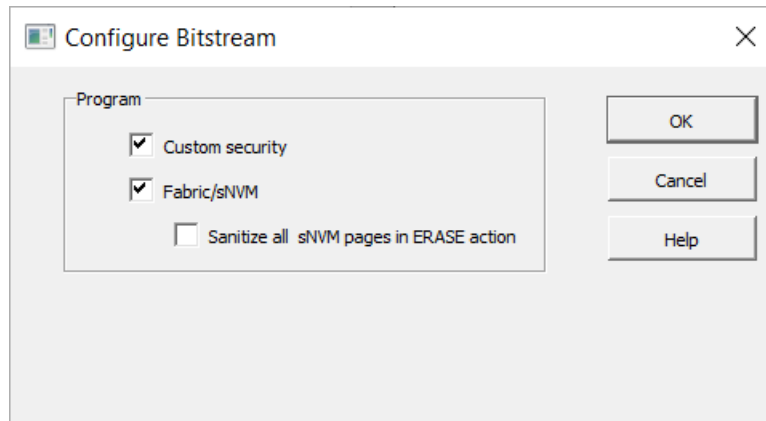
Right-click **Generate Bitstream** in the Design Flow window and choose **Configure Options** to open the Configure Bitstream dialog box.

The Configure Bitstream dialog box enables you to select which components you wish to program. Only features that have been added to your design are available for programming.

sNVM sanitize option is supported for PolarFire and PolarFire SoC. The sNVM sanitization option is enabled in the Configure Bitstream if Fabric/sNVM component is selected.

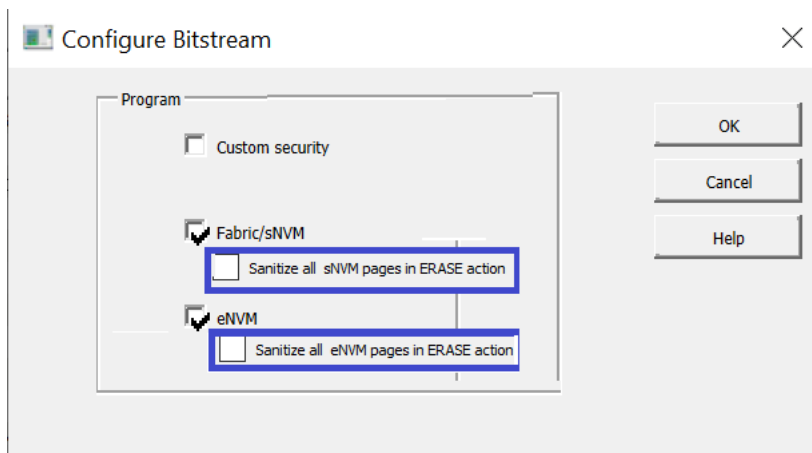
If the design includes uPROM, it will be included in the Fabric.

**Figure 8-48. Configure Bitstream**



eNVM option will be available only if eNVM clients are present and are being programmed and is available for PolarFire SoC devices only.

Figure 8-49. Configure Bitstream with eNVM option



Notes:

- Custom security is enabled if security was configured.
- All available features are selected by default.
- sNVM is always programmed with Fabric.

See Also

[Generate Bitstream](#)

## 8.10 Generate Bitstream

Generates the bitstream for use with the Run PROGRAM Action tool.

The tool incorporates the Fabric design, **sNVM** configuration, eNVM configuration (if configured) and custom security settings (if configured) to generate the bitstream file. You need to configure the bitstream before you generate the bitstream. Otherwise, default settings with all available features included will be used. Right-click **Generate Bitstream** and choose **Configure Options** to open the Configure Bitstream dialog box to select which components you wish to program. Only features that have been added to your design are available for programming.

If the design includes uPROM, it will be included in the Fabric.

eNVM option will available only if eNVM clients are present and are being programmed and is available for PolarFire SoC devices only.

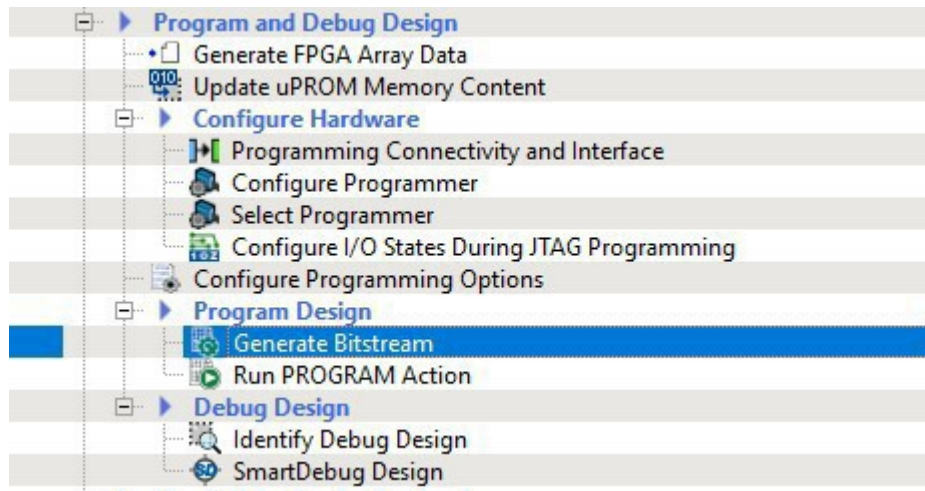
Modifications to the Fabric design, **sNVM** configuration, eNVM configuration or security settings will invalidate this tool and require regeneration of the bitstream file.

The Fabric programming data will only be regenerated if you make changes to the Fabric design, such as in the Create Design, Create Constraints and Implement Design sections of the Design Flow window.

When the process is complete a green check appears next to the operation in the Design Flow window (as shown in the figure below) and information messages appear in the Log window.



Figure 8-50. Generate Bitstream (Complete)



See Also

[Configure Bitstream Dialog Box](#)

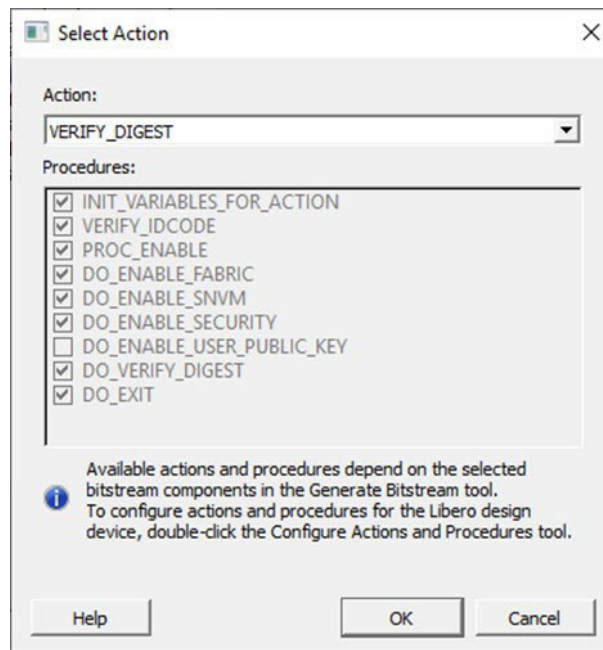
## 8.11 Run Programming Device Actions

If you have a device programmer connected, you can double-click **Run PROGRAM Action** to execute your programming in batch mode with default settings.

If your programmer is not connected, or if your default settings are invalid, the Reports view lists the error(s). To select a programming action to run:

1. Right-click **Run PROGRAM Action** and choose **Select Action**. The **Select Action** dialog box opens.

Figure 8-51. Select Action Dialog Box



2. Select a programming action from the drop-down list and click **OK**.

To configure programming actions, use the Configure Actions and Procedures tool.

See Also

[Configure Actions and Procedures](#)

### 8.11.1 Programming File Actions

Libero SoC enables you to program security settings, FPGA Array, **sNVM** and **eNVM** features.

**Note:** **eNVM** features are available for PolarFire SoC devices only.

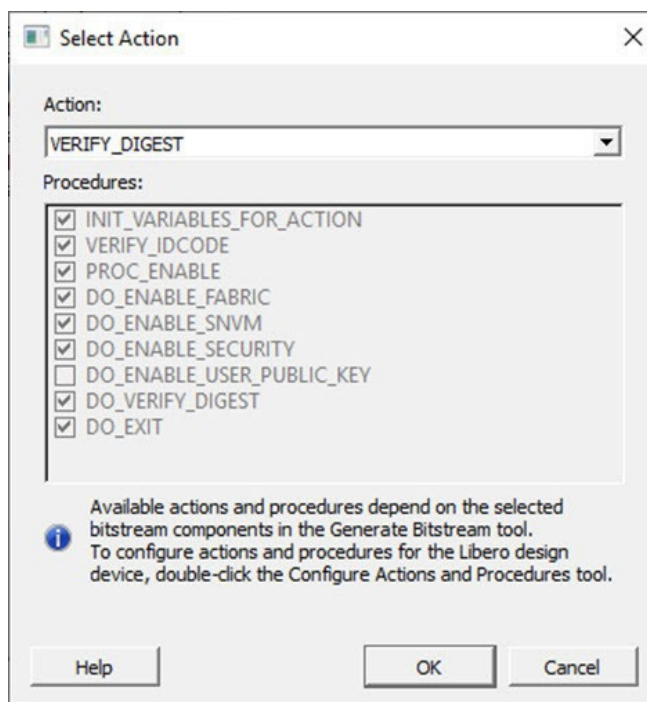
**Note:** If the design includes UPROM, it will be included in the Fabric.

You can program these features separately using different programming files or you can combine them into one programming file.

In the Design Flow window, expand **Program Design**, click **Run PROGRAM Action**, and right-click **Select Action**.

The Select Action dialog box opens.

**Figure 8-52. Select Action Dialog Box**



For details about configuring actions and procedures, see [Configure Actions and Procedures](#).

**Table 8-40. Exit Codes (PolarFire)**

Error Code	Exit Message	Exit Code	Possible Cause	Possible Solution
	Passed (no error)	0	—	—
0x8002	Failed to disable programming mode Failed to set programming mode	5	Unstable voltage level Signal integrity issues on JTAG pins	Monitor related power supplies that cause the issue during programming; check for transients outside of Microchip specifications. See your device datasheet for more information on transient specifications.  Monitor JTAG supply pins during programming; measure JTAG signals for noise or reflection.

.....continued

Error Code	Exit Message	Exit Code	Possible Cause	Possible Solution
0x8032	Device is busy	5	Unstable VDDIx voltage level	Monitor related power supplies that cause the issue during programming; check for transients outside of Microchip specifications. See your device datasheet for more information on transient specifications.
0x8003	Failed to enter programming mode	5	Unstable voltage level Signal integrity issues on JTAG pins DEVRST_N is tied to LOW	Monitor related power supplies that cause the issue during programming; check for transients outside of Microchip specifications. See your device datasheet for more information on transient specifications.  Monitor JTAG supply pins during programming; measure JTAG signals for noise or reflection.  Tie DEVRST_N to HIGH prior to programming the device.
0x8004	Failed to verify IDCODE	6	Incorrect programming file  Incorrect device in chain  Signal integrity issues on JTAG pins	Choose the correct programming file and select the correct device in the chain.  Measure JTAG pins and noise for reflection. If TRST is left floating then add pull-up to pin.  Reduce the length of Ground connection.
0x8005 0x8006 0x8007 0x8008 0x8009	Failed to verify FPGA Array  Failed to verify Fabric Configuration  Failed to verify Security Failed to verify sNVM Failed to verify eNVM	11	Device is programmed with a different design or the component is blank  Unstable voltage level  Signal integrity issues on JTAG pins	Verify the device is programmed with the correct data/design.  Monitor related power supplies that cause the issue during programming; check for transients outside of Microchip specifications. See your device datasheet for more information on transient specifications.  Monitor JTAG supply pins during programming; measure JTAG signals for noise or reflection.
0x8013	External digest check via JTAG/SPI Slave is disabled.	-18	External Digest check via JTAG/SPI Slave is disabled	Need to use a bitstream file which has a valid FlashLock/UPK1 to enable external digest check via JTAG/SPI Slave.
0x8015	FPGA Fabric digest verification: FAIL  Deselect procedure 'DO_ENABLE_FABRIC' to remove this digest check.	-20	FPGA Fabric is either erased or the data has been corrupted or tampered with	If the Fabric is erased, deselect procedure "DO_ENABLE_FABRIC" from action "VERIFY_DIGEST"

.....continued				
Error Code	Exit Message	Exit Code	Possible Cause	Possible Solution
0x8016	sNVM digest verification: FAIL  Deselect procedure 'DO_ENABLE_SNVN' to remove this digest check.	-20	sNVM is either erased or the data has been corrupted or tampered with	If the sNVM is erased, deselect procedure "DO_ENABLE_SNVN" from action "VERIFY_DIGEST"
0x8018	User security policies segment digest verification: FAIL  Deselect procedure 'DO_ENABLE_SECURITY' to remove this digest check.	-20	Security segment is either erased or the data has been corrupted or tampered with	If the security is erased, deselect procedure "DO_ENABLE_SECURITY" from action "VERIFY_DIGEST"
0x8019	UPK1 segment digest verification: FAIL  Deselect procedure 'DO_ENABLE_SECURITY' to remove this digest check.	-20	UPK1 segment is either erased or the data has been corrupted or tampered with	If the UPK1 is erased, deselect procedure "DO_ENABLE_SECURITY" from action "VERIFY_DIGEST"
0x801A	UPK2 segment digest verification: FAIL  Deselect procedure 'DO_ENABLE_UKS2' to remove this digest check.	-20	UPK2 segment is either erased or the data has been corrupted or tampered with	If the UPK2 is erased, deselect procedure "DO_ENABLE_UKS2" from action "VERIFY_DIGEST"
0x801B	Factory row and factory key segment digest verification: FAIL	-20	Factory row and factory key segment have been erased through zeroization or the data has been corrupted or tampered with	
0x801C	Fabric configuration segment digest verification: FAIL  Deselect procedure 'DO_ENABLE_FABRIC' to remove this digest check.	-20	Fabric configuration segment is either erased or has been corrupted or tampered with	If the Fabric configuration is erased, deselect procedure "DO_ENABLE_FABRIC" from action "VERIFY_DIGEST"

.....continued

Error Code	Exit Message	Exit Code	Possible Cause	Possible Solution
0x8052	UEK1 segment digest verification: FAIL  Deselect procedure 'DO_ENABLE_UEK1' to remove this digest check.	-20	UEK1 segment is either erased or the data has been corrupted or tampered with	If the UEK1 is erased, deselect procedure "DO_ENABLE_UEK1" from action "VERIFY_DIGEST"
0x8053	UEK2 segment digest verification: FAIL  Deselect procedure 'DO_ENABLE_UEK2' to remove this digest check.	-20	UEK2 segment is either erased or the data has been corrupted or tampered with	If the UEK2 is erased, deselect procedure "DO_ENABLE_UEK2" from action "VERIFY_DIGEST"
0x8054	DPK segment digest verification: FAIL  Deselect procedure 'DO_ENABLE_DPK' to remove this digest check.	-20	DPK segment is either erased or the data has been corrupted or tampered with	If the DPK is erased, deselect procedure "DO_ENABLE_DPK" from action "VERIFY_DIGEST"
0x8057	SMK segment digest verification: FAIL	-20	SMK segment is either erased or the data has been corrupted or tampered with	If the SMK is erased, deselect procedure "DO_ENABLE_SMK" from action "VERIFY_DIGEST"
0x8058	User Public Key segment digest verification: FAIL	-20	User Public Key segment is either erased or the data has been corrupted or tampered with	If the User Public Key is erased, deselect procedure "DO_ENABLE_USER_PUBLIC_KEY" from action "VERIFY_DIGEST"
0x801D	Device security prevented operation	-21	The device is protected with user pass key 1 and the bitstream file does not contain user pass key 1  User pass key 1 in the bitstream file does not match the device	Run DEVICE_INFO to view security features that are protected.  Provide a bitstream file with a user pass key 1 that matches the user pass key 1 programmed into the device.
0x801F	Programming Error.  Bitstream or data is corrupted or noisy	-22	Bitstream file has been corrupted or was incorrectly generated  Unstable voltage level  Signal integrity issues on JTAG pins	Regenerate bitstream file  Monitor related power supplies that cause the issue during programming; check for transients outside of Microchip specifications. See your device datasheet for more information on transient specifications. Monitor JTAG supply pins during programming; measure JTAG signals for noise or reflection.

.....continued

Error Code	Exit Message	Exit Code	Possible Cause	Possible Solution
0x8021	Programming Error. Invalid/Corrupted encryption key	-23	File contains an encrypted key that does not match the device  File contains user encryption key, but device has not been programmed with the user encryption key	Provide a programming file with an encryption key that matches that on the device  First program security with master programming file, then program with user encryption 1/2 field update programming files
0x8023	Programming Error. Back level not satisfied	-24	Design version is not higher than the back-level programmed device	Generate a programming file with a design version higher than the back level version
0x8001	Failure to read DSN	-24	Device is in System Controller Suspend Mode Check board connections	TRSTB should be driven High or disable "System Controller Suspend Mode".
0x8027	Programming Error. Insufficient device capabilities	-26	Device does not support the capabilities specified in programming file	Generate a programming file with the correct capabilities for the target device
0x8029	Programming Error. Incorrect DEVICEID	-27	Incorrect programming file  Incorrect device in chain  Signal integrity issues on JTAG pins	Choose the correct programming file and select the correct device in chain  Measure JTAG pins and noise or reflection. If TRST is left floating, then add pull-up to pin  Reduce the length of ground connection
0x802B	Programming Error. Programming file is out of date, please regenerate.	-28	Programming file version is out of date	Generate programming file with latest version of Libero SoC
0x8030	Programming Error Invalid or inaccessible Device Certificate	-31	FAB_RESET_N is tied to ground	FAB_RESET_N should be tied to HIGH
0x8032 0x8034 0x8036 0x8038	Instruction timed out	-32	Unstable voltage level  Signal integrity issues on JTAG pins	Monitor related power supplies that cause the issue during programming; check for transients outside of Microchip specifications. See your device datasheet for more information on transient specifications.  Monitor JTAG supply pins during programming; measure JTAG signals for noise or reflection.
0x8010	Failed to unlock user pass key 1	-35	Pass key in file does not match device	Provide a programming file with a pass key that matches pass key programmed into the device.

.....continued				
Error Code	Exit Message	Exit Code	Possible Cause	Possible Solution
0x8011	Failed to unlock user pass key 2	-35	Pass key in file does not match device	Provide a programming file with a pass key that matches pass key programmed into the device.
0x804F	Bitstream programming action is disabled	-38	Unstable voltage level Bitstream programming action has been disabled in Security Policy Manager	Monitor related power supplies that cause the issue during programming; check for transients outside of Microchip specifications. See your device datasheet for more information on transient specifications.  Need to use a bitstream file which has a valid FlashLock/UPK1 to enable the bitstream programming action.
0x805B	Error, security must be either programmed on a blank device or with the FPGA Fabric design	-42	Security only bitstream programming on a programmed device	Use this bitstream on a blank device or generate a new bitstream that contains the FPGA Fabric design along with the security
0x805C	eNVM digest verification: FAIL  Deselect procedure 'DO_ENABLE_ENVM' to remove this digest check	-20	eNVM is either erased or the data has been corrupted or tampered with.	If the eNVM is erased, deselect procedure 'DO_ENABLE_ENVM' from action 'VERIFY_DIGEST'

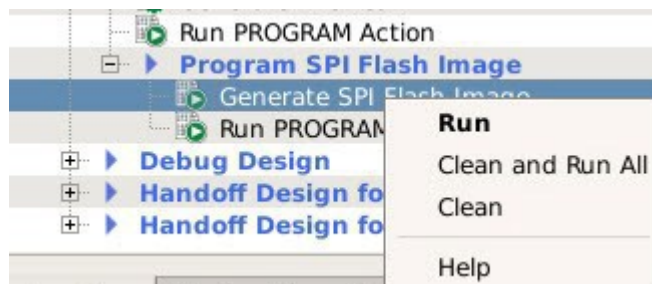
## 8.12 Program SPI Flash Image

### 8.12.1 Generate SPI Flash Image

This tool generates the <design>\_spi\_flash.bin file in the implementation folder.

To run this tool; under the **Program SPI Flash Image**, right-click **Generate SPI Flash Image** and choose **Run**.

**Figure 8-53. Selecting the Run Command**



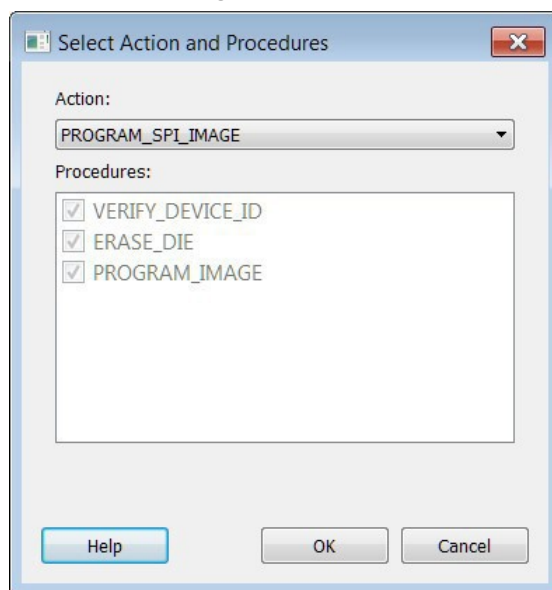
This tool depends on the Configure Design Initialization Data and Memories tool and the Generate Design Initialization Data tool. When running, the tool verifies that the SPI Flash configuration data is saved and valid; and that the SPI Flash initialization client was generated successfully (if required).

### 8.12.2 Configure SPI Flash Image Actions and Procedures

If SPI Flash is configured, you can select supported SPI Flash Image actions and procedures in the Select Action and Procedures dialog box. See the following example.



Figure 8-54. Select Action and Procedures Dialog Box



The following table lists the actions and procedures for the Run PROGRAM\_SPI\_Flash tool.

Action	Mandatory Procedures	Description
PROGRAM_SPI_IMAGE	VERIFY_DEVICE_ID ERASE_DIE PROGRAM_IMAGE	This action will erase the entire SPI flash then program the SPI image.
VERIFY_SPI_IMAGE	VERIFY_DEVICE_ID VERIFY_IMAGE	This action verifies the SPI Image on the SPI Flash.
READ_SPI_IMAGE	VERIFY_DEVICE_ID READ_IMAGE	This action reads the SPI Image from the SPI Flash.
ERASE_SPI_FLASH	VERIFY_DEVICE_ID ERASE_DIE	This action erases the entire SPI Flash.

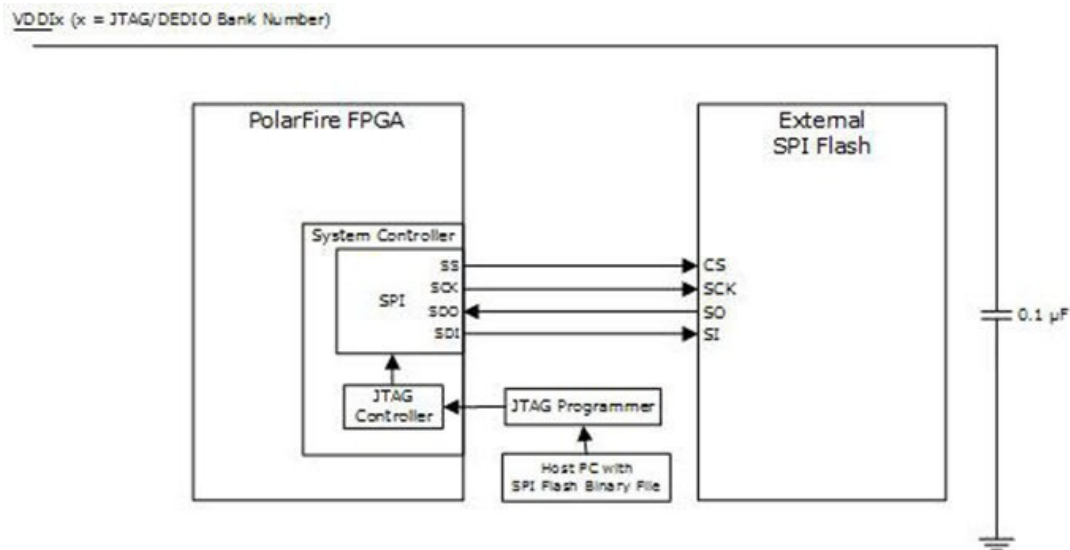
**Note:** If the device ID does not match when running any action, the action will fail.

### 8.12.3 Run Programming SPI Flash Actions

This tool allows the user to program the SPI Flash device connected to the PolarFire device through the JTAG programming interface. Currently, only the Micron 1Gb SPI flash is supported, and is included with the Evaluation Kit. This feature minimizes cost by not requiring a mux and external SPI pins on the board for SPI flash programming by another tool. This tool always erases the entire SPI flash prior to programming. Programming starts at address 0 of the SPI flash until the last client. Any gaps in the SPI flash are programmed with all 1's.

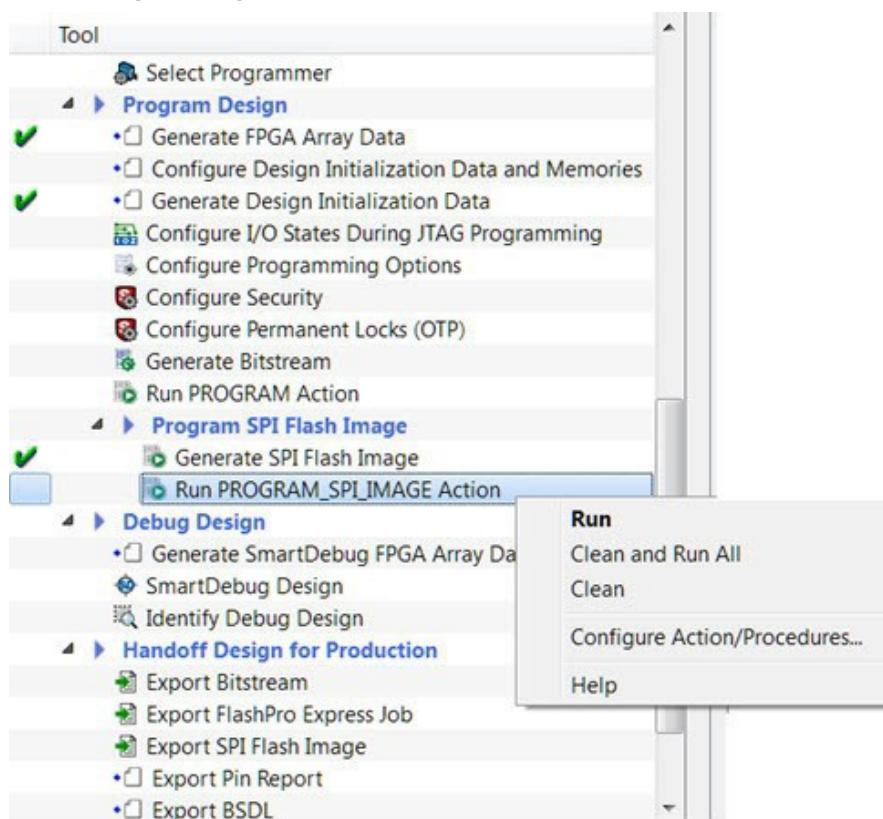
**Note:** This version of the programmer does not support SPI Flash security. Device security options such as "Hardware Write Protect" should be disabled for the External SPI Flash device.





**Note:** The SPI pins are controlled by the Boundary Scan Register one bit at a time.

**Figure 8-55. SPI Flash Programming with PolarFire Device**



The following table provides the expectations of programming the SPI flash with a FP5 programmer. Future programmers are planned, and should greatly improve programming times. All times are in hh:mm:ss.

SPI Size	ERASE	PROGRAM	VERIFY/READ	TCK	Programmer
1 MB	3:55	00:00:45	00:10:46	4MHz	FP5

.....continued					
SPI Size	ERASE	PROGRAM	VERIFY/READ	TCK	Programmer
1 MB	3:55	00:00:28	00:10:05	15MHz	FP5
9 MB	3:55	00:06:38	01:19:15	4MHz	FP5
9 MB	3:55	00:04:26	01:08:49	10MHz	FP5
18 MB	3:55	00:09:04	02:32:43	10MHz	FP5
128 MB	3:55	00:58:38	22:07:55	15MHz	FP5

Recommendations:

1. Since the verify time is currently not optimized, it is recommended to authenticate the SPI bitstreams with system services for quicker verification.
2. Since this tool erases the SPI flash prior to programming and currently does not support Data Storage clients for user data, it is recommended to program the SPI flash with Libero prior to programming other data on the SPI flash.
3. Since programming time is currently not optimized, it is recommended to not have huge gaps between clients in the SPI flash, since gaps are currently programmed with 1's.

If SPI Flash is configured, you can execute Run PROGRAM\_SPI\_IMAGE Action and select SPI Flash Image actions and procedures.

In the Design Flow window, under Program SPI Flash Image, right-click **Run PROGRAM\_SPI\_Image Action** and choose **Configure Action/Procedures**.

**Note:** In this release, SPI Flash programming is supported for MICRON devices only.

See [Configure SPI Flash Image Actions and Procedures](#) for information about supported actions and procedures.

## 8.12.4 Partial Programming Support

### eFP6/FP6 Partial Support

This feature allows you to program clients anywhere within the SPI-Flash memory space connected to PolarFire and PolarFire SoC devices.

Every client is programmed at a specified target Start Address, from the lowest address to the highest. The Libero generated lookup table is programmed first, followed by INIT\_STATGE\_3\_SPI\_CLIENT, SPI bitstream for IAP, Recovery/Golden, and Auto Update.

The following image is a sample list of clients:

Program	Name	Type	Index	Content File	Start Address	End Address
<input checked="" type="checkbox"/>	i1	SPI Bitstream for IAP	2	C:\Workspace\Projects\Libero\hex_files\hex_files\ai_no_bypas...	0x700	0x148f
<input checked="" type="checkbox"/>	g1	SPI Bitstream for Recovery/Golden	0	C:\Workspace\Projects\Libero\hex_files\hex_files\ai_golden.spi	0x1500	0x228f
<input checked="" type="checkbox"/>	u1	SPI Bitstream for Auto Update	1	C:\Workspace\Projects\Libero\hex_files\hex_files\ai_no_bypas...	0x2400	0x318f
<input checked="" type="checkbox"/>	INIT_STAGE_3_SPI_CLIENT	Design Initialization		designer\test\test_uic.bin	0x666	0x685
<input checked="" type="checkbox"/>	Client1	Data Storage		C:\Workspace\Projects\Libero\hex_files\hex_files\random_256k...	0xfffff	0x140000
<input checked="" type="checkbox"/>	Empty1	Data Storage			0x1affcd	0x1b00069
<input checked="" type="checkbox"/>	Middle	Data Storage		C:\Workspace\Projects\Libero\hex_files\hex_files\random_13b...	0x3fffffa	0x4000006

The memory regions between the clients are left intact. However, SPI-Flash erase erases data in the selected sector, subsector, or block if clients target Start Address is outside of sector, subsector, or block boundaries.

The software reports an error code along with messages that indicate whether original data has been lost.

The verify operation verifies only the data of the selected clients.

The following example is a sample log from FPExpress SPI Flash programming:

```

programmer '138000A' : Scan Chain...
programmer '138000A' : Scan and Check Chain PASSED.
programmer '138000A' : device 'MPF200T' : Executing action PROGRAM_SPI_IMAGE
programmer '138000A' : JTAG TCK frequency = 4 MHz
Performing SPI-Flash action. Please wait...
Warning: It is recommended that the target client addresses align to sector, subsector or
block boundaries. Else, during erase or program actions, the software will erase and restore
data that is partly outside of target memory regions. If restoring data operation fails, the
user must reprogram the original clients covering the affected failed areas.
Processing SPI-Flash Client 0: Target Address = 0x0. Size = 1024 Bytes
FP6 acceleration mode enabled with PPD file. Please wait...
Programmer '138000A' : JTAG TCK frequency = 4 MHz
FP6 Messages:
=====
JTAG DirectC Version: 5.1
Identifying FPGA device...
ActID = 0x0f8121cf
Micron device is found.
SPI-Flash IDCode = 0x21ba20
Device size (MBytes) = 128
Performing SPI Flash Program Action:
SPI-Flash memory target address = 0x00. Image byte size = 1024
SPI Flash memory region to erase: 0x00 - 0x03ff. Please wait...
Restoring data at address = 0x0400 - 0x0fff
Programming image from address = 0x00 - 0x03ff
Programming data at address: 0x00 - 0x03ff
Operation Status: Passed
=====
Processing SPI-Flash Client 1: Target Address = 0x666. Size = 32 Bytes
FP6 acceleration mode enabled with PPD file. Please wait...
Programmer '138000A' : JTAG TCK frequency = 4 MHz
FP6 Messages:
=====
JTAG DirectC Version: 5.1
Identifying FPGA device...
ActID = 0x0f8121cf
Micron device is found.
SPI-Flash IDCode = 0x21ba20
Device size (MBytes) = 128
Performing SPI Flash Program Action:
SPI-Flash memory target address = 0x0666. Image byte size = 32
SPI Flash memory region to erase: 0x0666 - 0x0685. Please wait...
Restoring data at address = 0x00 - 0x0665
Restoring data at address = 0x0686 - 0x0fff
Programming image from address = 0x0666 - 0x0685
Programming data at address: 0x0666 - 0x0685
Operation Status: Passed
=====

```

```
Processing SPI-Flash Client 6: Target Address = 0x1AFFFCd. Size = 157 Bytes
FP6 acceleration mode enabled with PPD file. Please wait...
Programmer '138000A' : JTAG TCK frequency = 4 MHz
FP6 Messages:
=====
JTAG DirectC Version: 5.1
Identifying FPGA device...
ActID = 0x0f8121cf
Micron device is found.
SPI-Flash IDCode = 0x21ba20
Device size (MBytes) = 128
Performing SPI Flash Image Erase Action:
SPI-Flash memory target address = 0x01afffcd. Image byte size = 157
SPI Flash memory region to erase: 0x01afffcd - 0x01b00069. Please wait...
Restoring data at address = 0x01aff000 - 0x01afffcc
Restoring data at address = 0x01b0006a - 0x01b00fff
Operation Status: Passed
```

### FlashPro3/FlashPro4/FlashPro5 Partial Support

The partial programming support described for eFP6/FP6 does not apply for FP3/FP4, or FP5 programmers. FP3/4/5 programmers deal with partial clients by generating on the fly one bitstream containing the data of all the clients, as shown in the previous example. Memory gaps between the clients are filled with 0xff. The entire SPI-Flash memory device is erased first, and then the generated bitstream is programmed, starting with address 0.

The verify operation verifies that the client data and the memory gaps between the clients are verified against 0xff.

The following example is a sample output for programming the previous clients using FP5 programmer.

```
programmer 'S2001KZSR7' : Scan Chain...
Programmer 'S2001KZSR7' : JTAG TCK / SPI SCK frequency = 1 MHz
programmer 'S2001KZSR7' : Check Chain...
programmer 'S2001KZSR7' : Scan and Check Chain PASSED.
programmer 'S2001KZSR7' : device 'MPF200T' : Executing action PROGRAM_SPI_IMAGE
Programmer 'S2001KZSR7' : JTAG TCK / SPI SCK frequency = 4 MHz
?Warning: FP3, FP4, and FP5 programmers do not support partial SPI-Flash programming. The
entire SPI-Flash device will be erased and programmed with the currently enabled clients.
ID: 00441021ba20
Erasing SPI flash die...
ERASE SPI Flash Finished : Fri Feb 19 15:31:05 2021 (Elapsed time 00:00:05)
Programming SPI image...
Program SPI image Finished : Fri Feb 19 15:31:07 2021 (Elapsed time 00:00:02)
programmer 'S2001KZSR7' : device 'MPF200T' : Executing action PROGRAM_SPI_IMAGE PASSED.
programmer 'S2001KZSR7' : Chain programming PASSED.
Chain Programming Finished: Fri Feb 19 15:31:07 2021 (Elapsed time 00:00:07)
```

### Notes:

- Backward Compatibility:

FP3/4/5/6 that use the software version 2021.1 release support all jobs created using older versions of the Libero software.

Do not use older FPEXpress versions of software with FPEXpress jobs created by Libero version 2021.1 and later releases. If used, the entire bitstream, including header contents, will be programmed into the SPI-Flash memory device, which is not the intended behavior.

- Programmers used:

Jobs programmed with FP6 but verified with FP3/4/5 programmers may fail using software version 2021.1 release. Such jobs fail because the memory gap between clients is skipped during verification but are verified against 0xff if FP3/4/5 programmers are used.

- SPI address table:

If only storage clients are created, Libero generates a 1024-byte SPI address lookup table automatically while adding any SPI-Flash client.

Erasing a client using eFP6/FP6 programmers also erases the SPI lookup address table. For FP3/4/5 programmers, erase operations erase the entire SPI-Flash memory device.

### 8.12.5 SPI-Flash Bitstream Format

For versions prior to Libero SoC v2021.1, the SPI-Flash Bitstream is the payload.

For Libero SoC v2021.1 and later, the SPI-Flash Bitstream format is as follows:

**Table 8-41. SPI-Flash Bitstream Format**

Number of Bytes	Value	Description
4	FF FF FF FF	Tag to indicate new version of bitstream.
2	XX XX	Major Version of bitstream.
2	XX XX	Minor Version of bitstream.
4	XX XX XX XX	Client0 Target Address.
4	FF FF FF FF	Client0 Byte Size.
1	XX	Flag. If set to 1, Payload will follow.  If set to 0, Payload is not present and memory region will be erased.
X	X	Payload based on previous flag
4	XX XX XX XX	Client1 Target Address
4	XX XX XX XX	Client1 Byte Size
1	XX	Flag. If set to 1, Payload will follow.  If set to 0, no Payload is present and memory region will be erased only
X	X	Payload based on previous flag
Repeat for other clients		
4	FF FF FF FF	End Tag

## 9. Debug Design

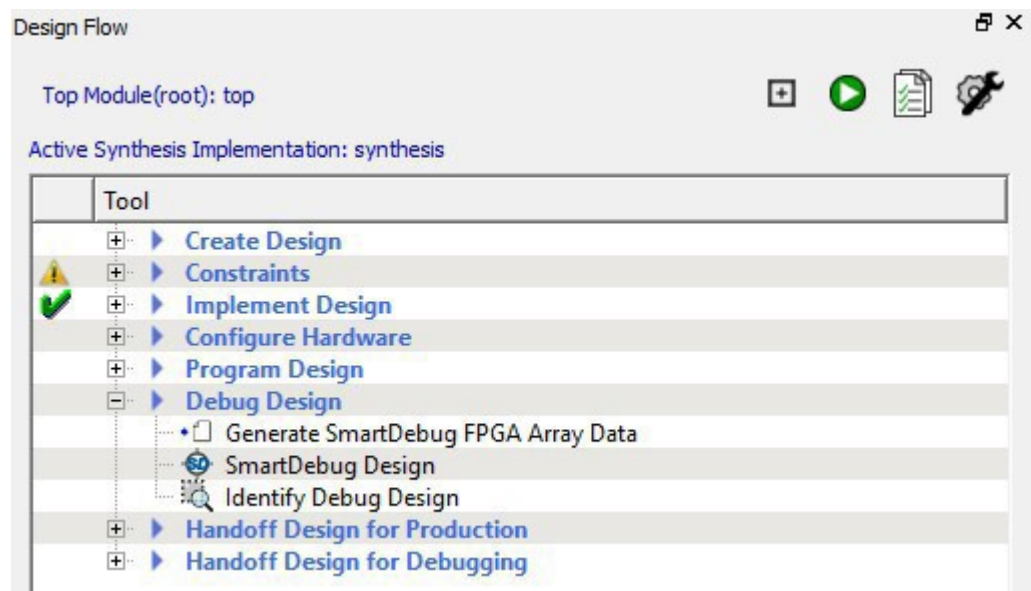
### 9.1 Generate SmartDebug FPGA Array Data

The Generate SmartDebug FPGA Array Data tool generates database files used in downstream tools:

- \*.db used for debugging FPGA Fabric in SmartDebug

Double-click **Generate SmartDebug FPGA Array Data** or right-click **Generate SmartDebug FPGA Array Data** in the Design Flow window and click **Run** to generate SmartDebug FPGA Array Data. Before running this tool, the design should have completed the Place and Route step. If not, Libero SoC runs implicitly the upstream tools (Synthesis, Compile Netlist, and Place and Route) before it generates the FPGA SmartDebug Array Data.

**Figure 9-1. Generate SmartDebug FPGA Array Data**



### 9.2 SmartDebug

Design debug is a critical phase of FPGA design flow. Microchip's SmartDebug tool complements design simulation by allowing verification and troubleshooting at the hardware level. SmartDebug can provide access to Microchip FPGA device's built-in probe logic, which enables designers to check the state of inputs and outputs in real-time without re-layout of the design.

SmartDebug can be run in two modes:

- Integrated mode from the Libero Design Flow
- Standalone mode

#### 9.2.1 Integrated Mode

When run in integrated mode from Libero, SmartDebug can access all design and programming hardware information. No extra setup step is required. In addition, the Probe Insertion feature is available in Debug FPGA Array.

To open SmartDebug in the Libero Design Flow window, expand **Debug Design** and double-click **SmartDebug Design**.

### 9.2.2 Standalone Mode

SmartDebug can be installed separately in the setup containing FlashPro Express and Job Manager. This provides a lean installation that includes all the programming and debug tools to be installed in a lab environment for debug. In this mode, SmartDebug is launched outside of the Libero Design Flow. Prior to launch of SmartDebug standalone mode, you must go through SmartDebug project creation and import a Design Debug Data Container (DDC) file, exported from Libero, to access all debug features in the supported devices.

**Note:** In standalone mode, the Probe Insertion feature is not available in FPGA Array Debug, as it requires incremental routing to connect the user net to the specified I/O.

### 9.2.3 See Also

[SmartDebug User Guide](#)

## 9.3 Identify Debug Design

Libero SoC integrates the Identify RTL debugger tool. It enables you to probe and debug your FPGA design directly in the source RTL. Use Identify software when the design behavior after programming is not in accordance with the simulation results.

To open the Identify RTL debugger, in the Design Flow window under Debug Design double-click **Instrument Design**.

Identify features:

- Instrument and debug your FPGA directly from RTL source code.
- Internal design visibility at full speed.
- Incremental iteration - Design changes are made to the device from the identify environment using incremental compile. You iterate in a fraction of the time it takes route the entire device.
- Debug and display results - You gather only the data you need using unique and complex triggering mechanisms.

You must have both the Identify RTL Debugger and the Identify Instrumentor to run the debugging flow outlined below.

To use the Identify Instrumentor and Debugger:

1. Create your source file (as usual) and run pre-synthesis simulation.
2. (Optional) Run through an entire flow (Synthesis - Compile - Place and Route - Generate a Programming File) without starting Identify.
3. Right-click **Synthesize** and choose **Open Interactively** in Libero SoC to launch Synplify.
4. In Synplify, click **Options > Configure Identify Launch** to setup Identify.
5. In Synplify, create an Identify implementation; to do so, click **Project > New IdentifyImplementation**.
6. In the Implementations Options dialog, make sure the Implementation Results > Results Directory points to a location under <libero project>\synthesis\, otherwise Libero SoC is unable to detect your resulting **Verilog** Netlist file.
7. From the Instrumentor UI specify the sample clock, the breakpoints, and other signals to probe. Synplify creates a new synthesis implementation. Synthesize the design.
8. In Libero SoC, run Synthesis, Place and Route and Generate a Programming File.  
**Note:** Libero SoC works from the edif netlist of the current active implementation, which is the implementation you created in Synplify for Identify debug.
9. Double-click **Identify Debug Design** in the Design Flow window to launch the IdentifyDebugger.

The Identify RTL Debugger, Synplify, and FlashPro must be synchronized in order to work properly. See the [Release Notes](#) for more information on which versions of the tools work together.

## 10. Handoff Design for Production

### 10.1 Export Bitstream

Export Bitstream enables you to export PPD, STAPL, DAT, and SPI programming files.

To export a bitstream file:

1. Under Handoff Design for Production, double-click **Export Bitstream**. The Export Bitstream dialog box opens. The dialog box options depend on your Custom Security settings and [Permanent Locks](#) for production settings:
  - [Export Bitstream tool when the device is configured with Bitstream Encryption with Default key option in the Configure Security tool](#)
  - [Export Bitstream tool when the device is configured with Custom Security option in the Configure Security tool](#)
  - [Export Bitstream tool when the device is configured with Permanent Locks for Production tool](#)
2. Enter your Bitstream file name and select the location to export the bitstream file.
3. Choose your options, such as DAT file if you wish to include support for Embedded ISP, SPI file if you need support for Auto programming, Auto Update and IAP services, PPD, or STAPL if you need support for ISP.
4. You can choose whether you want to export the ASCII HEX file for debugging when DAT bitstream file format is selected.
5. Select whether you want to export files for Microchip In House Programming (IHP).
6. Select to include the bitstream components, **Bypass Back Level protection option for Recovery/Golden Bitstream** and Include Pass Keys in Plaintext.
7. Choose the Zeroization Action that you want to use to reprogram the device.  
**Note:** Zeroization Action is not supported for XT and ES devices.  
  
**Note:** PolarFire SoC has same zeroization options in Export Bitstream tool and Programming Actions and Procedures as PolarFire.
8. Click OK to export the selected bitstream files.
9. The sanitization settings for ERASE action has two options. Depending on the device and configured features, the corresponding options are visible.
  - Sanitize all sNVM pages (This option is available for both PolarFire and PolarFire SoC families).
  - Sanitize all eNVM pages (This option is available for PolarFire SoC family only).

#### See Also

[Digest File](#)



### 10.1.1 Export Bitstream tool when the device is configured with Bitstream Encryption with Default Key in the Configure Security Tool

Figure 10-1. Export Bitstream with Default Key Dialog Box (For PolarFireSoC)

**Export Bitstream**

Design settings

Programming options set with the Configure Programming Options tool:  
Design version - 0    Back Level version - 0

Security options set with the Configure Security tool:  
[Encrypt bitstream with default key. No user keys or security settings are enabled.](#)

Bitstream file(s)

Name:     Location:

Existing files:  

This folder is empty.

Formats:

☒ PPD    Support for ISP (JTAG and SPI Slave)

☒ DAT    Support for Embedded ISP (JTAG and SPI Slave)    ☐ Export separate ASCII HEX file for debugging

☐ STAPL    Support for ISP

☐ Chain STAPL    Support for ISP, Single Microsemi device in a JTAG chain

☒ SPI    Support for Auto Programming, Auto Update, and IAP Services

Export files for Microsemi In House Programming (IHP): ☐ Yes ☒ No

File types:

	Fabric/sNVM	eNVM	Advanced Options (*.spi files only)
<input type="checkbox"/> File to program at trusted facility	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="button" value="Set..."/>

Zeroization actions:

☐ Like New (Erases all user data; device can be immediately reprogrammed by user)

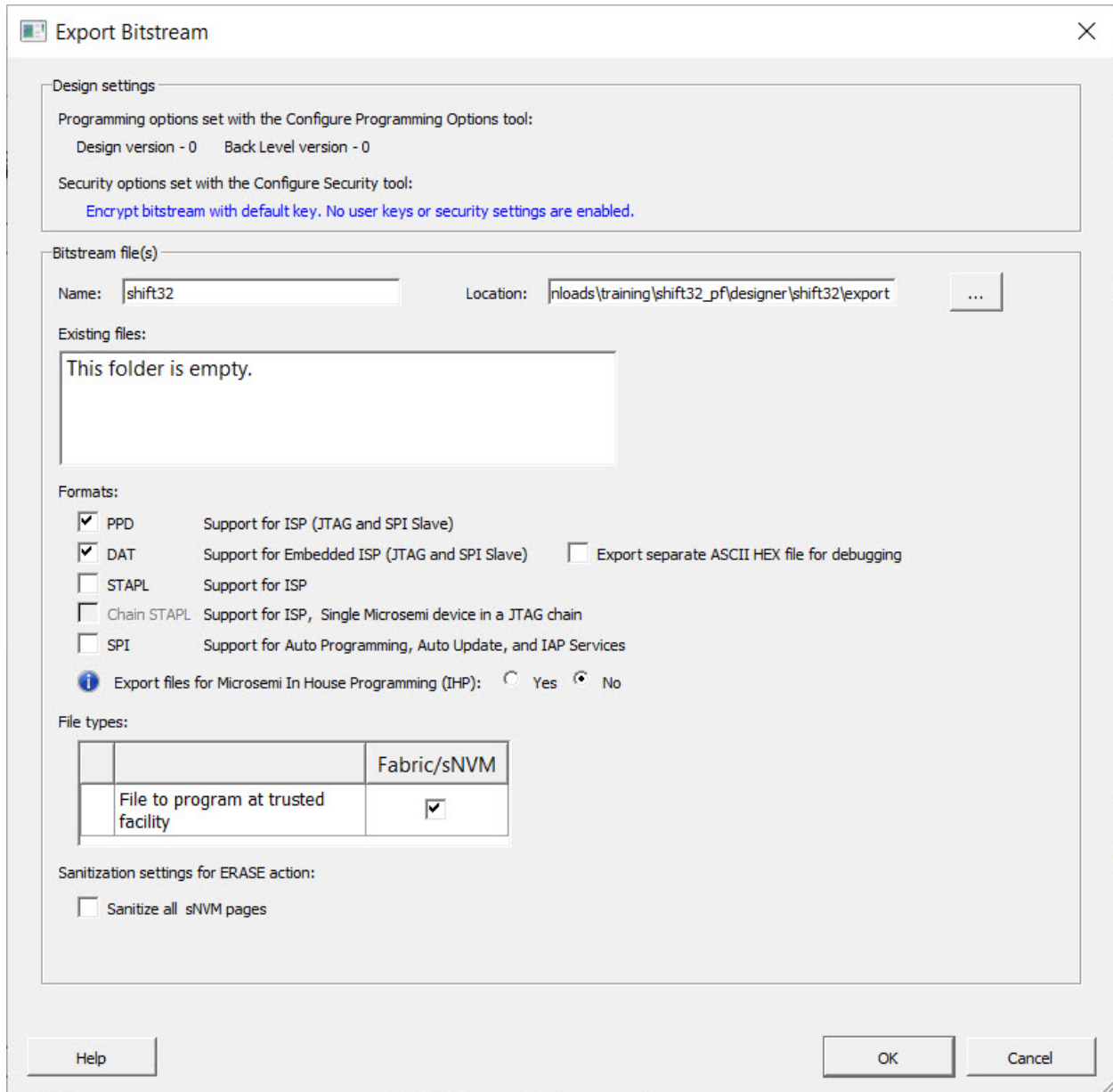
☐ Unrecoverable (Erases all data and destroys reprogrammability; device must be scrapped)

Sanitization settings for ERASE action:

☐ Sanitize all sNVM pages

☐ Sanitize all eNVM pages

Figure 10-2. Export Bitstream with Default Key Dialog Box (For PolarFire)



The dialog box is titled "Export Bitstream" and contains the following sections:

- Design settings:**
  - Programming options set with the Configure Programming Options tool:
    - Design version - 0    Back Level version - 0
  - Security options set with the Configure Security tool:
    - Encrypt bitstream with default key. No user keys or security settings are enabled.
- Bitstream file(s):**
  - Name:     Location:
  - Existing files:
 

This folder is empty.
- Formats:**
  - ☒ PPD    Support for ISP (JTAG and SPI Slave)
  - ☒ DAT    Support for Embedded ISP (JTAG and SPI Slave)    ☐ Export separate ASCII HEX file for debugging
  - ☐ STAPL    Support for ISP
  - ☐ Chain STAPL    Support for ISP, Single Microsemi device in a JTAG chain
  - ☐ SPI    Support for Auto Programming, Auto Update, and IAP Services
  - ☒ Export files for Microsemi In House Programming (IHP):    ☐ Yes    ☒ No
- File types:**

	Fabric/sNVM
File to program at trusted facility	<input checked="" type="checkbox"/>
- Sanitization settings for ERASE action:**
  - ☐ Sanitize all sNVM pages

Buttons:

#### Design Settings:

- **Programming Options set with Configure Programming Options tool-** Displays the Design version and Back Level version set in the Configure Programming Options tool. They are read-only options and cannot be modified.

**Note:** Info and warning messages appear based on the value set for Back Level version.

- **Security options set with the Configure Security tool-** Provides a brief description of current security options.

#### Bitstream file(s):

- **Name** - Sets the name of your bitstream file. The default name is the design name.
- **Location** - Location to save the exported file.
- **Existing files** - Lists bitstream files previously created.
- **Formats** - Select the Bitstream File format you want to export:
  - PPD

- DAT
- Export separate ASCII HEX file for debugging – Exports DAT file in HEX format. This option is active only when DAT file format is selected.
- STAPL
- SPI

PPD and DAT file formats are the default file formats. If Auto programming is disabled in SPM, neither master nor update files can be exported in SPI format.

For PolarFire, Auto programming is disabled if Fabric is disabled for updates or if Fabric is open but disable Auto programming option is selected. If Auto programming is enabled and an action is locked, plaintext options are disabled if only SPI format is selected and enabled otherwise.

For PolarFireSoC, Auto programming is disabled if Fabric and eNVM both are disabled for updates or disable Auto programming option is selected. If Fabric is disabled for updates but eNVM is not, then SPI bitstream file for update cannot include Fabric component. If Fabric is open and eNVM is locked then, SPI bitstream file for update cannot include eNVM.

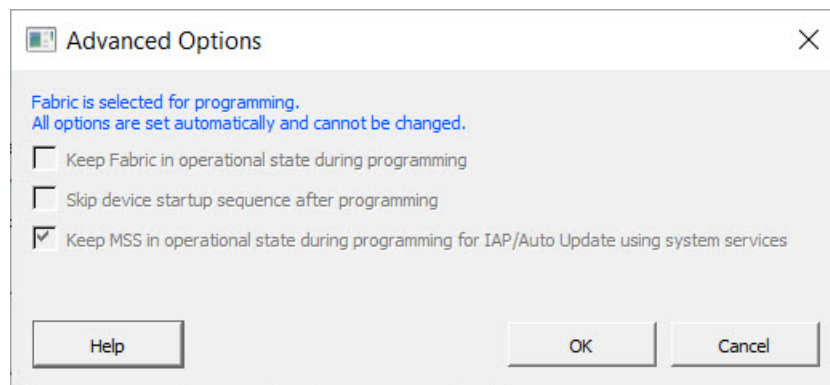
- Export files for Microsemi In House Programming (IHP) – Exports DAT and STP file formats if 'Yes' is selected.

- **File types:** Lists all the bitstream files to be exported

- **File to program at trusted facility** – Click to include Fabric /sNVM into the bitstream files to be programmed at a trusted facility.
- eNVM option is available for PolarFireSoC family only.
- Advanced options are available for PolarFireSoC devices only. The advanced options are always enabled if SPI file format is selected else the advanced options are disabled.

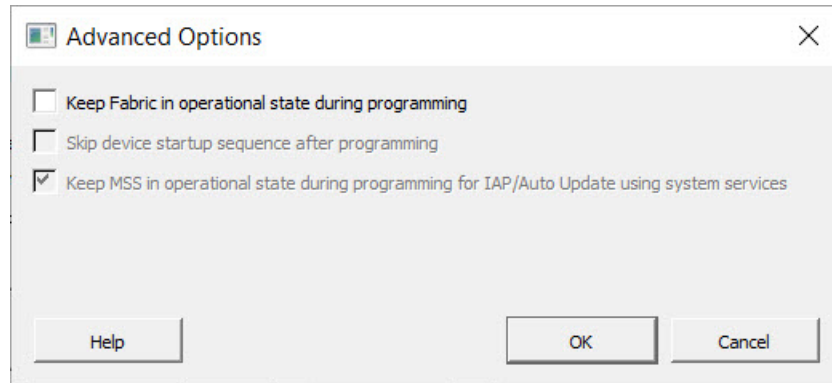
For a test case when fabric is being programmed, all options are set automatically and cannot be changed. The advanced options appear as in below figure:

**Figure 10-3. Advanced Options for a testcase without Custom Security (when fabric is being programmed)**



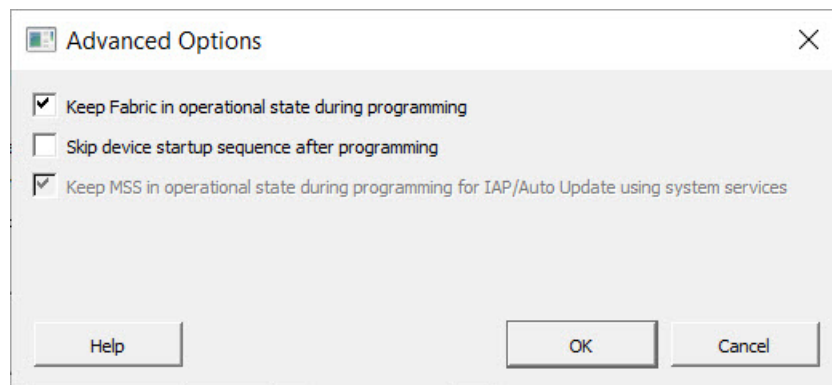
For a test case when fabric is NOT being programmed, you can choose to **Keep Fabric in operational state during programming**. The advanced options appear as in below figure:

**Figure 10-4. Advanced Options for a testcase without Custom Security (when fabric is NOT being programmed)**



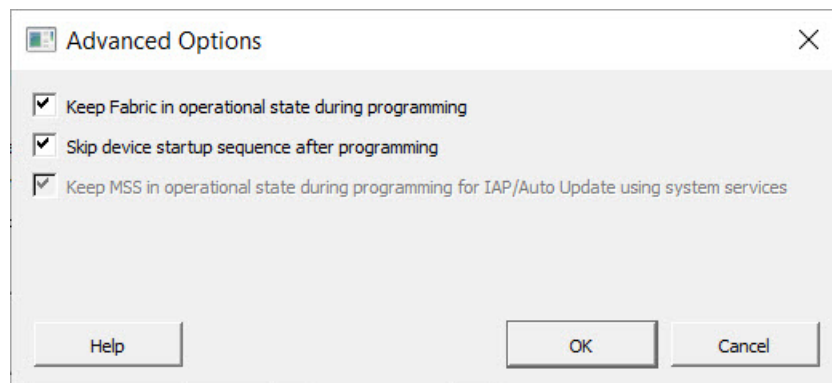
If **Keep Fabric in operational state during programming** option is selected, you can choose to **Skip device startup sequence after programming**. The advanced options appear as in below figure:

**Figure 10-5. Advanced Options for a testcase without custom security( when fabric is NOT being programmed) and "Keep Fabric in operational state during programming" option is selected**



If **Keep Fabric in operational state during programming** and **Skip device startup sequence after programming** options are selected, then the advanced options appear as in below figure:

**Figure 10-6. Advanced Options for a testcase without custom security( when fabric is NOT being programmed) and "Keep Fabric in operational state during programming" and "Skip device startup sequence after programming" options are selected**



- **Zeroization Actions** – Select the **Like New** checkbox to erase all user data to make the device like new and reprogram the device immediately, or select the **Unrecoverable** checkbox to erase all data and destroy reprogrammability (in this case, the device must be scrapped).
- **Sanitization settings for ERASE Action** -

The sNVM sanitization option is enabled if Fabric/sNVM component is selected for at least one file type (master or update).

The eNVM sanitization option is available if eNVM is configured and selected for at least one file type (master or update). This option is applicable for PolarFireSoC family only.

Default for both options when they are enabled is “off”.

The sanitization options do not depend on OTP security setting; ERASE action will fail if there is OTP security.

### 10.1.2 Export Bitstream tool when the Device is Configured with Custom Security option in the Configure Security Tool

**Note:** If Permanent Locks for Production are configured along with Custom Security settings, Export Bitstream Security options are set with Configure Permanent Lock for Production tool. Otherwise, they are set with the Configure Security tool. The Permanent Lock Security settings can be removed by either right-clicking on the tool in the Design Tree or by clicking the Remove Permanent Locks button in the UI.

**Figure 10-7. Export Bitstream with Custom Security Dialog Box (For PolarFireSoC)**

**Design settings**

Programming options set with the Configure Programming Options tool:  
 Design version - 0    Back Level version - 0

Security options set with the Configure Security tool:  
 Disable all factory key modes and configured security settings.  
 Use FlashLock/UPK1 to temporarily enable settings during one programming or debugging session.  
 SmartDebug access control is enabled. Internal data may be accessible. Anyone can debug or access active probes, access Live Probe, and read the content of sNVM.  
 Factory test mode is allowed. This will allow Microsemi to perform Failure Analysis.  
 Zeroization through JTAG/SPI Slave is enabled. This is not recommended for production devices.  
 External Fabric/sNVM design verify and read digest requests through JTAG/SPI Slave are enabled.  
 Repeated external Fabric digest calculations can impact its reliability. View Datasheet for additional information.

**Bitstream file(s)**

Name:     Location:     ...

**Existing files:**

This folder is empty.

**Formats:**

☒ PPD    Support for ISP (JTAG and SPI Slave)  
☒ DAT    Support for Embedded ISP (JTAG and SPI Slave)    ☐ Export separate ASCII HEX file for debugging  
☐ STAPL    Support for ISP  
☐ Chain STAPL    Support for ISP, Single Microsemi device in a JTAG chain  
☒ SPI    Support for Auto Programming, Auto Update, and IAP Services

☒ Export files for Microsemi In House Programming (IHP):    ☐ Yes    ☒ No

**File types:**

		Custom Security	Fabric/sNVM	eNVM	Include Plaintext Pass Keys (N/A for *.spi files)	Bypass Back Level protection for Recovery/Golden bitstream (*.spi files only)	Advanced Options (*.spi files only)
1	Master file	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="button" value="View..."/>
2	File encrypted with UEK1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="button" value="Set..."/>
3	File encrypted with UEK2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="button" value="Set..."/>

**Zeroization actions:**

☐ Like New (Erases all user data; device can be immediately reprogrammed by user)  
☐ Unrecoverable (Erases all data and destroys reprogrammability; device must be scrapped)

**Sanitization settings for ERASE action:**

☐ Sanitize all sNVM pages  
☐ Sanitize all eNVM pages

Figure 10-8. Export Bitstream with Custom Security Dialog Box (For PolarFire)

**Design settings**

Programming options set with the Configure Programming Options tool:  
Design version - 0    Back Level version - 0

Security options set with the Configure Security tool:  
Disable all factory key modes and configured security settings.  
Use FlashLock/UPK1 to temporarily enable settings during one programming or debugging session.  
SmartDebug access control is enabled. Internal data may be accessible. Anyone can debug or access active probes, access Live Probe, and read the content of sNVM.  
Factory test mode is allowed. This will allow Microsemi to perform Failure Analysis.  
Zeroization through JTAG/SPI Slave is enabled. This is not recommended for production devices.  
External Fabric/sNVM design verify and read digest requests through JTAG/SPI Slave are enabled.  
Repeated external Fabric digest calculations can impact its reliability. View Datasheet for additional information.

**Bitstream file(s)**

Name:     Location:  ...

Existing files:  

This folder is empty.

**Formats:**

☒ PPD    Support for ISP (JTAG and SPI Slave)  
☒ DAT    Support for Embedded ISP (JTAG and SPI Slave)    ☐ Export separate ASCII HEX file for debugging  
☐ STAPL    Support for ISP  
☐ Chain STAPL    Support for ISP, Single Microsemi device in a JTAG chain  
☐ SPI    Support for Auto Programming, Auto Update, and IAP Services

☒ Export files for Microsemi In House Programming (IHP):    ☐ Yes    ☒ No

**File types:**

		Custom Security	Fabric/sNVM	Include Plaintext Pass Keys (N/A for *.spi files)	Bypass Back Level protection for Recovery/Golden bitstream (*.spi files only)
1	Master file	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2	File encrypted with UEK1		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3	File encrypted with UEK2		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

**Zeroization actions:**

☐ Like New (Erases all user data; device can be immediately reprogrammed by user)  
☐ Unrecoverable (Erases all data and destroys reprogrammability; device must be scrapped)

**Sanitization settings for ERASE action:**

☐ Sanitize all sNVM pages

Help    OK    Cancel

Design Settings:

- **Programming Options set with Configure Programming Options tool**- Displays the Design version and Back Level version set in the Configure Programming Options tool. They are read-only options and cannot be modified.

**Note:** Info and warning messages appear based on the value set for Back Level version.

- **Security options set with the Configure Security tool**- Provides a brief description of current security settings.

Bitstream file(s):

- **Name** - Sets the name of your bitstream file. The default name is the design name.
- **Location** - Location to save the exported file.
- **Existing files** - Lists bitstream files previously created.
- **Formats** - Select the Bitstream File format you want to export:
  - PPD
  - DAT
  - Export separate ASCII HEX file for debugging – Exports DAT file in HEX format. This option is active only when DAT file format is selected.
  - STAPL
  - SPI

PPD and DAT file formats are the default file formats. If Auto programming is disabled in SPM, neither master nor update files can be exported in SPI format.

For PolarFire, Auto programming is disabled if Fabric is disabled for updates or if Fabric is open but disable Auto programming option is selected. If Auto programming is enabled and an action is locked, plaintext options are disabled if only SPI format is selected and enabled otherwise.

For PolarFireSoC, Auto programming is disabled if Fabric and eNVM both are disabled for updates or disable Auto programming option is selected. If Fabric is disabled for updates but eNVM is not, then SPI bitstream file for update cannot include Fabric component. If Fabric is open and eNVM is locked then, SPI bitstream file for update cannot include eNVM.

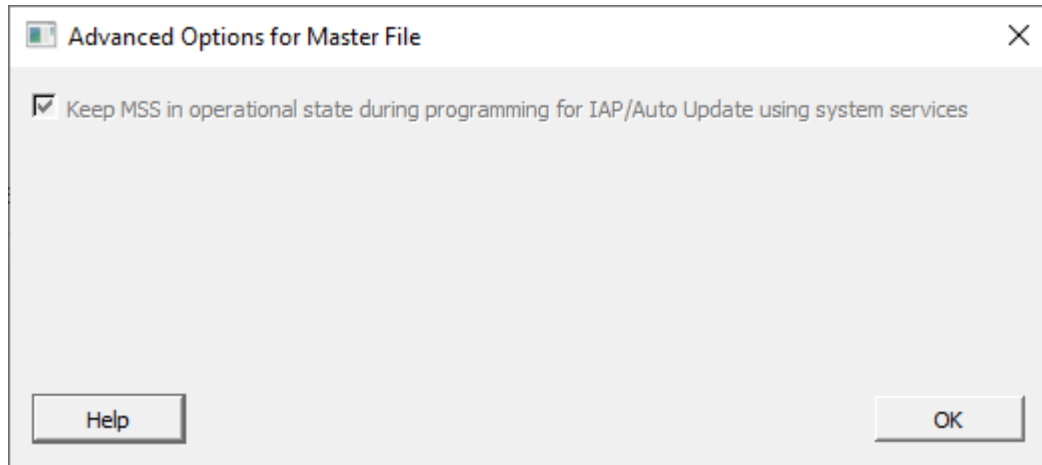
- Export files for Microsemi In House Programming (IHP) – Exports DAT and STP file formats if 'Yes' is selected.
- **File types** –  
**Note:** Refer to Include Plaintext Pass Keys to check the availability of Export Pass keys for Plaintext option for the following Bitstream files.  
**Note:** eNVM option is only visible when eNVM is used in the design. eNVM features are available for PolarFire SoC devices only.

Lists all the bitstream files to be exported.

- **Master file** – Click to include:
  - Fabric/sNVM.
  - eNVM (If the design does not have eNVM, but in Configure Security tool, the user locked eNVM for updates, the eNVM component is disabled and eNVM OTP warning is shown).
  - Bypass Back Level protection for Recovery/Golden bitstream (This option is available only if Back Level protection is enabled in the Configure Security tool and SPI file format is selected. For update files, this option is available if Back Level protection is enabled in Configure Security Tool, SPI file format is selected and Fabric is selected. By default, Back Level bypass is "off" for update files).
  - Include Plaintext Pass Keys. If UPK1 is permanently disabled, Include Plaintext Pass Keys option is disabled for Master file.
  - Advanced options are available for PolarFireSoC devices only. The advanced options are always enabled if SPI file format is selected else the advanced options are disabled.  
For a testcase with Custom Security, the advanced options for Master file can be viewed as in figure below.

**Figure 10-9. Advanced Options for Master File**





**Notes:**

1. Custom Security is always programmed in the Master file.
2. Security-only programming must be performed only on erased or new devices. If performed on a device with fabric programmed, the fabric will be disabled after performing security-only programming. You must reprogram the fabric to re- enable it.
3. Depending on the Permanent Locks configured in the Configure Permanent Locks for Production tool, some of the bitstream components for Master file are selected by default and are unavailable for users to change.
  - File encrypted with UEK1 to program at untrusted facility or for Broadcast field update - Click to include:
    - Fabric/sNVM.
    - eNVM
    - Bypass Back Level protection for Recovery/Golden bitstream.  
(This option is available only if Back Level protection is enabled in the Configure Security tool and SPI file format is selected.

For update files, this option is available if Back Level protection is enabled in Configure Security Tool, SPI file format is selected and Fabric is selected. By default, Back Level bypass is “off” for update files).

  - Include Plaintext Pass Keys is available if UPK1 is not permanently locked and the Update Policy Fabric /sNVM or eNVM (available in PolarFireSoC only) is passcode protected or any of program, authenticate or verify action is locked.
  - For a test case when fabric is being programmed for file encrypted with UEK1, all options are set automatically and cannot be changed. The advanced options appear as in below figure:

**Figure 10-10. Advanced options for File Encrypted with UEK1 (when fabric is being programmed)**





For a test case when fabric is NOT being programmed, users can choose to **Keep Fabric in operational state during programming**. The advanced options appear as in below figure:

**Figure 10-11. Advanced options for File Encrypted with UEK1 (when fabric is NOT being programmed)**



If **Keep Fabric in operational state during programming** option is selected, users can choose to **Skip device startup sequence after programming**. The advanced options appear as in below figure:

**Figure 10-12. Advanced Options for File Encrypted with UEK1 when Fabric is NOT being programmed and user selects "Keep Fabric in operational state during programming" option**



If **Keep Fabric in operational state during programming** and **Skip device startup sequence after programming** options are selected. The advanced options appear as in below figure:

**Figure 10-13. Advanced Options for File Encrypted with UEK1 when Fabric is NOT being programmed and user selects both "Keep Fabric in operational state during programming" and "Skip device startup sequence after programming" options**



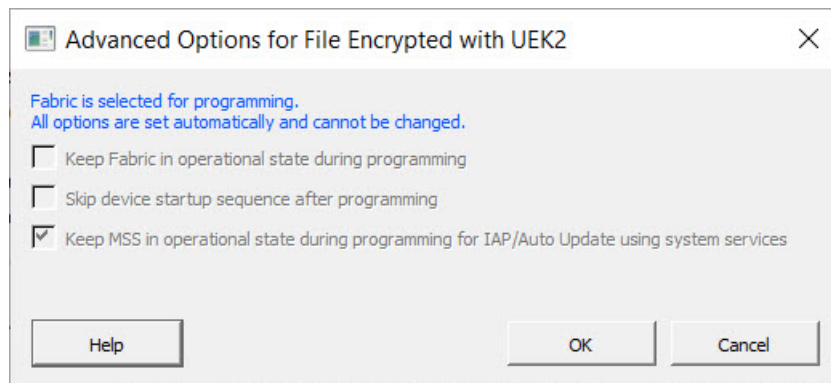
- File encrypted with UEK2 to program at untrusted facility or for Broadcast field update - Click to include:

- Fabric/sNVM.
- eNVM
- Bypass Back Level protection for Recovery/Golden bitstream.  
(This option is available only if Back Level protection is enabled in the Configure Security tool and SPI file format is selected.

For update files, this option is available if Back Level protection is enabled in Configure Security Tool, SPI file format is selected and Fabric is selected. By default, Back Level bypass is “off” for update files).

- Include Plaintext Pass Keys is available if UPK1 is not permanently locked and the Update Policy Fabric /sNVM or eNVM (available in PolarFireSoC only) is passcode protected or any of program, authenticate or verify action is locked.
- For a test case when fabric is being programmed for file encrypted with UEK2, all options are set automatically and cannot be changed. The advanced options appear as in below figure:

**Figure 10-14. Advanced options for File Encrypted with UEK2 (when fabric is being programmed)**



For a test case when fabric is NOT being programmed, users can choose to **Keep Fabric in operational state during programming**. The advanced options appear as in below figure:

**Figure 10-15. Advanced options for File Encrypted with UEK2 (when fabric is NOT being programmed)**



If **Keep Fabric in operational state during programming** option is selected, users can choose to **Skip device startup sequence after programming**. The advanced options appear as in below figure:

**Figure 10-16. Advanced Options for File Encrypted with UEK2 when Fabric is NOT being programmed and user selects "Keep Fabric in operational state during programming" option**



If **Keep Fabric in operational state during programming** and **Skip device startup sequence after programming** options are selected. The advanced options appear as in below figure:

**Figure 10-17. Advanced Options for File Encrypted with UEK2 when Fabric is NOT being programmed and user selects both "Keep Fabric in operational state during programming" and "Skip device startup sequence after programming" options**



**Note:** If sNVM/Fabric is One Time Programmable, it is precluded from bitstream encrypted with UEK1/2.

- **Zeroization Actions** – Select the Like New checkbox to erase all user data to make the device like new and reprogram the device immediately, or select the Unrecoverable checkbox to erase all data and destroy reprogrammability (in this case, the device must be scrapped).

- **Sanitization settings for ERASE Action -**

The sNVM sanitization option is enabled if Fabric/sNVM component is selected for at least one file type (master or update).

The eNVM sanitization option is available if eNVM is configured and selected for at least one file type (master or update). This option is applicable for PolarFireSoC family only.

Default for both options when they are enabled is “off”.

The sanitization options do not depend on OTP security setting; ERASE action will fail if there is OTP security.

**Bypass Back Level protection for Recovery/Golden bitstream-** If Back Level protection is enabled in the Configure Security tool, you can bypass the back level protection for SPI bitstreams to prevent Programming Recovery failures.

The following examples show the usage of Bypass Back Level Protection.

**Table 10-1. Example 1 – Fails without Bypass Back Level Protection**

Step	Bitstream	Action	Result	Design version	Back Level version	Device Back Level version
1	Golden/Recovery	Auto Programming	Pass	2	1	1
2	IAP/Update Bitstream	Auto Update/IAP	Pass	3	2	2
3	IAP/Update Bitstream	Auto Update/IAP	Fail, Attempt Programming Recovery	4	N/A	2

In the above example, the device will have Back Level version 2 after step 2. If you attempt to program with bitstream in step 3, it will fail. If the device attempts to reprogram, it will initiate Programming Recovery with the Golden/Recovery bitstream. Since the Golden/Recovery Bitstream has Design version 2, which is less than or equal to the Back Level version in the device, it will fail. If the Bypass Back Level version option is selected, this back level protection check will be bypassed for the Golden/Recovery Bitstream only and it will succeed.

**Table 10-2. Example 2 – Does not require Bypass Back Level Protection**

Step	Bitstream	Action	Result	Design version	Back Level version	Device Back Level version
1	Golden/Recovery	Auto Programming	Pass	2	1	1
2	IAP/Update Bitstream	Auto Update/IAP	Pass	3	1	1
3	IAP/Update Bitstream	Auto Update/IAP	Fail, Attempt Programming Recovery	4	N/A	1

In this example, the device will have Back Level version 1 after step 2. If you attempt to program with bitstream in step 3, it will succeed since the Golden/Recovery Design version is greater than the Back Level version on the device.

**Table 10-3. Example 3 – Requires Bypass Back Level Protection**

Step	Bitstream	Action	Result	Design version	Back Level version	Device Back Level version
1	Golden/Recovery	Auto Programming	Pass	1	1	1
2	IAP/Update Bitstream	Auto Update/IAP	Pass	2	1	1
3	IAP/Update Bitstream	Auto Update/IAP	Fail, Attempt Programming Recovery	3	N/A	1

In this example, the device will have Back Level version 1 after step 2. If you attempt to program with bitstream in step 3, it will fail. If the device attempts to reprogram, it will initiate Programming Recovery with the Golden/Recovery bitstream. Since the Golden/Recovery Bitstream has Design version 1, which is less than or equal to the Back Level version in the device, it will fail. If the Bypass Back Level version option is selected, this back level protection check will be bypassed for the Golden/Recovery Bitstream only and it will succeed.

### 10.1.3 Export Bitstream Tool when the Device is Configured with Permanent Locks for Production Tool

**Notes:**

- This feature is available for PolarFire family only. PolarFireSoC devices do not support permanent locks.
- If Permanent Locks for Production are configured along with Custom Security settings, Export Bitstream Security options are set with the Configure Permanent Lock for Production tool. Otherwise, they are set with the Configure Security tool. The Permanent Lock Security settings can be removed by either right-clicking on the tool in the Design Tree or by clicking the **Remove Permanent Locks** button in the UI.

**Figure 10-18. Export Bitstream Dialog Box with Security Options Set with the Configure Permanent Locks for Production Tool**

**Design settings**

Programming options set with the Configure Programming Options tool:  
Design version - 0    Back Level version - 0

Security options set with the Configure Permanent Locks for Production tool:  
 ⚠️ Disable all factory key modes and configured security settings.  
 Use FlashLock/UPK1 to temporarily enable settings during one programming or debugging session.  
 Factory test mode is allowed. This will allow Microsemi to perform Failure Analysis.  
 External Fabric/sNVM design verify and read digest requests through JTAG/SPI Slave are enabled.  
 Repeated external Fabric digest calculations can impact its reliability. View Datasheet for additional information.

**Bitstream file(s)**

Name:     Location:  ...

Existing files:  

This folder is empty.

**Formats:**

☒ PPD    Support for ISP (JTAG and SPI Slave)  
☒ DAT    Support for Embedded ISP (JTAG and SPI Slave)    ☐ Export separate ASCII HEX file for debugging  
☐ STAPL    Support for ISP  
☐ Chain STAPL    Support for ISP, Single Microsemi device in a JTAG chain  
☐ SPI    Support for Auto Programming, Auto Update, and IAP Services

Export files for Microsemi In House Programming (IHP): ☐ Yes ☒ No

**File types:**

		Custom Security	Fabric/sNVM	Include Plaintext Pass Keys (N/A for *.spi files)	Bypass Back Level protection for Recovery/Golden bitstream (*.spi files only)
1	Master file	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2	File encrypted with UEK1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3	File encrypted with UEK2	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Sanitization settings for ERASE action:  
☐ Sanitize all sNVM pages

Buttons: Help, OK, Cancel

**Design Settings:**

- **Programming Options set with the Configure Programming Options tool-** Displays the Design version and Back Level version set in the Configure Programming Options tool. They are read-only options and cannot be modified.

**Note:** Info and warning messages appear based on the value set for Back Level version.

- **Security options set with the Configure Permanent Locks for Production tool-** Provides a brief description of current security settings.

**Bitstream file(s):**

- **Name** - Sets the name of your bitstream file. The default name is the design name.
  - **Location** - Location to save the exported file.
  - **Existing files** - Lists bitstream files previously created.

**Formats** - Select the Bitstream File format you want to export:

- PPD
- DAT

- Export separate ASCII HEX file for debugging – Exports DAT file in HEX format. This option is active only when DAT file format is selected.
- STAPL
- SPI

PPD and DAT file formats are the default file formats. If Auto programming is disabled in SPM, neither master nor update files can be exported in SPI format.

For PolarFire, Auto programming is disabled if Fabric is disabled for updates or if Fabric is open but disable Auto programming option is selected. If Auto programming is enabled and an action is locked, plaintext options are disabled if only SPI format is selected and enabled otherwise.

- Export files for Microsemi In House Programming (IHP) – Exports DAT and STP file formats if 'Yes' is selected.

- **File types –**

**Note:** Refer to Include Plaintext Pass Keys to check the availability of Include Plaintext Pass keys option for the following Bitstream files.

Lists all the bitstream files to be exported

- **Master file** – Click to include:
  - Fabric/sNVM.
  - Bypass Back Level protection for Recovery/Golden bitstream  
(This option is available only if Back Level protection is enabled in the Configure Security tool and SPI file format is selected.  
  
For update files, this option is available if Back Level protection is enabled in Configure Security Tool, SPI file format is selected and Fabric is selected. By default, Back Level bypass is "off" for update files).
  - Include Plaintext Pass Keys is available if UPK1 is not permanently locked and the Update Policy Fabric / sNVM is passcode protected or any of program, authenticate or verify action is locked.

Notes:

1. Custom Security is always programmed in the Master file.
2. Security-only programming must be performed only on erased or new devices. If performed on a device with fabric programmed, the fabric will be disabled after performing security-only programming. You must reprogram the fabric to re-enable it.
3. Depending on the Permanent Locks configured in the Configure Permanent Locks for Production tool, some of the bitstream components for Master file are selected by default and are unavailable for users to change.
  - 3.1. File encrypted with UEK1 to program at untrusted facility or for Broadcast field update - Click to include:
    - Fabric/sNVM.
    - Bypass Back Level protection for Recovery/Golden bitstream (For master files, this option is available only if Back Level protection is enabled in the Configure Security tool and SPI file format is selected. By default, Back Level bypass is "on" for master file.  
For update files, this option is available if Back Level protection is enabled in Configure Security Tool, SPI file format is selected and Fabric is selected. By default, Back Level bypass is "off" for update files).
    - Include Plaintext Pass Keys is available if UPK1 is not permanently locked and the Update Policy Fabric /sNVM is passcode protected or any of program, authenticate or verify action is locked.
  - 3.2. File encrypted with UEK2 to program at untrusted facility or for Broadcast field update - Click to include:
    - Fabric/sNVM.
    - Bypass Back Level protection for Recovery/Golden bitstream (For master files, this option is available only if Back Level protection is enabled in the Configure Security tool and SPI file format is selected. By default, Back Level bypass is "on" for master file.  
For update files, this option is available if Back Level protection is enabled in Configure Security Tool, SPI file format is selected and Fabric is selected. By default, Back Level bypass is "off" for update files).

- Include Plaintext Pass Keys is available if UPK1 is not permanently locked and the Update Policy Fabric /sNVM is passcode protected or any of program, authenticate or verify action is locked.

**Note:** If sNVM/Fabric is One Time Programmable, it is precluded from bitstream encrypted with UEK1/2.

- **Zeroization Actions** – Select the Like New checkbox to erase all user data to make the device like new and reprogram the device immediately, or select the Unrecoverable checkbox to erase all data and destroy reprogrammability (in this case, the device must be scrapped).

- **Sanitization settings for ERASE Action** -

The sNVM sanitization option is enabled if Fabric/sNVM component is selected for at least one file type (master or update).

Default for both options when they are enabled is “off”.

The sanitization options do not depend on OTP security setting; ERASE action will fail if there is OTP security.

**Bypass Back Level protection for Recovery/Golden bitstream-** If Back Level protection is enabled in the Configure Security tool, you can bypass the back level protection for SPI bitstreams to prevent Programming Recovery failures.

The following examples show the usage of Bypass Back Level Protection.

**Table 10-4. Example 1 – Fails without Bypass Back Level Protection**

Step	Bitstream	Action	Result	Design version	Back Level version	Device Back Level version
1	Golden/Recovery	Auto Programming	Pass	2	1	1
2	IAP/Update Bitstream	Auto Update/IAP	Pass	3	2	2
3	IAP/Update Bitstream	Auto Update/IAP	Fail, Attempt Programming Recovery	4	N/A	2

In the above example, the device will have Back Level version 2 after step 2. If you attempt to program with bitstream in step 3, it will fail. If the device attempts to reprogram, it will initiate Programming Recovery with the Golden/Recovery bitstream. Since the Golden/Recovery Bitstream has Design version 2, which is less than or equal to the Back Level version in the device, it will fail. If the Bypass Back Level version option is selected, this back level protection check will be bypassed for the Golden/Recovery Bitstream only and it will succeed.

**Table 10-5. Example 2 – Does not Require Bypass Back Level Protection**

Step	Bitstream	Action	Result	Design version	Back Level version	Device Back Level version
1	Golden/Recovery	Auto Programming	Pass	2	1	1
2	IAP/Update Bitstream	Auto Update/IAP	Pass	3	1	1
3	IAP/Update Bitstream	Auto Update/IAP	Fail, Attempt Programming Recovery	4	N/A	1

In this example, the device will have Back Level version 1 after step 2. If you attempt to program with bitstream in step 3, it will succeed, since the Golden/Recovery Design version is greater than the Back Level version on the device.



**Table 10-6. Example 3 – Requires Bypass Back Level Protection**

Step	Bitstream	Action	Result	Design version	Back Level version	Device Back Level version
1	Golden/Recovery	Auto Programming	Pass	1	1	1
2	IAP/Update Bitstream	Auto Update/IAP	Pass	2	1	1
3	IAP/Update Bitstream	Auto Update/IAP	Fail, Attempt Programming Recovery	3	N/A	1

In this example, the device will have Back Level version 1 after step 2. If you attempt to program with bitstream in step 3, it will fail. If the device attempts to reprogram, it will initiate Programming Recovery with the Golden/Recovery bitstream. Since the Golden/Recovery Bitstream has Design version 1, which is less than or equal to the Back Level version in the device, it will fail. If the Bypass Back Level version option is selected, this back level protection check will be bypassed for the Golden/Recovery Bitstream only and it will succeed.

#### 10.1.4 Security Programming Files

Export Bitstream (expand Handoff Design for Production in the Design Flow window) creates the following files:

**<filename>\_master.(stp/spi/dat)** - Created when Enable custom security options is specified in the Security Wizard. This is the master programming file; it includes all programming features enabled, User Key Set 1, User Key Set 2 (optionally if specified), and your security policy settings.

**<filename>\_security\_only\_master.(stp/spi/dat)** – Created when Enable custom security options is specified in the Security Wizard. Master security programming file; includes User Key Set 1, User Key Set 2 (optionally if specified), and your security policy settings.

**<filename>\_uek1.(stp/spi/dat)** – Programming file encrypted with User Encryption Key 1 used for field updates; includes all your features for programming except security.

**<filename>\_uek2.(stp/spi/dat)** – Programming file encrypted with User Encryption Key 2 used for field updates; includes all your features for programming except security.

## 10.2 Export FlashPro Express Job

To program the design using standalone FlashPro Express tool on Linux or Windows, the user must export a FlashPro Express Job. The job file will include chain configuration, Programmer Settings and programming files loaded from Programming Connectivity and Interface.

**Note:** SPI Slave mode is supported by FlashPro6 for PolarFire devices. JTAG is the default interface.

**Note:** The Export FlashPro Express Job uses Permanent Locks for Production configuration, if Permanent Locks are configured. If Permanent Locks are not configured, it uses the Configure Security configuration.

Use the Configure Security tool before you export the programming job to add security. The Export FlashPro Express Job dialog box displays the Security Options you have configured through the Configure Security tool or Configure Permanent Locks for Production tool.

The Export FlashPro Express Job dialog box options vary depending on the device you are using and the Security Settings.

**Note:** The eNVM option is visible when eNVM is used in the design. eNVM features are available for PolarFire SoC devices only.

**Figure 10-19. Export FlashPro Express Job Dialog Box with Bitstream Encryption with Default Key**

**Export FlashPro Express Job**

**Design settings**

Programming options set with the Configure Programming Options tool:  
Design version - 0    Back Level version - 0

Security options set with the Configure Security tool:  
Encrypt bitstream with default key. No user keys or security settings are enabled.

**FlashPro Express Job file**

Name:     Location:  ...

**Existing files:**

This folder is empty.

**Program (JTAG programming interface):**

☒ Design    ☐ SPI Flash

**Design bitstream file format:**

**Design bitstream file type:**

	Fabric/sNVM	eNVM
File to program at trusted facility	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

**Design actions and procedures:**

**Zeroization actions:**

☐ Like New (Erases all user data; device can be immediately reprogrammed by user)

☐ Unrecoverable (Erases all data and destroys reprogrammability; device must be scrapped)

**Sanitization settings for ERASE action:**

☐ Sanitize all sNVM pages

☐ Sanitize all eNVM pages

**Design Settings:**

- **Programming Options set with Configure Programming Options tool-** Displays the Design version and Back Level version set in the Configure Programming Options tool. They are read-only options and cannot be modified.

**Note:** Info and warning messages appear based on the value set for Back Level version.

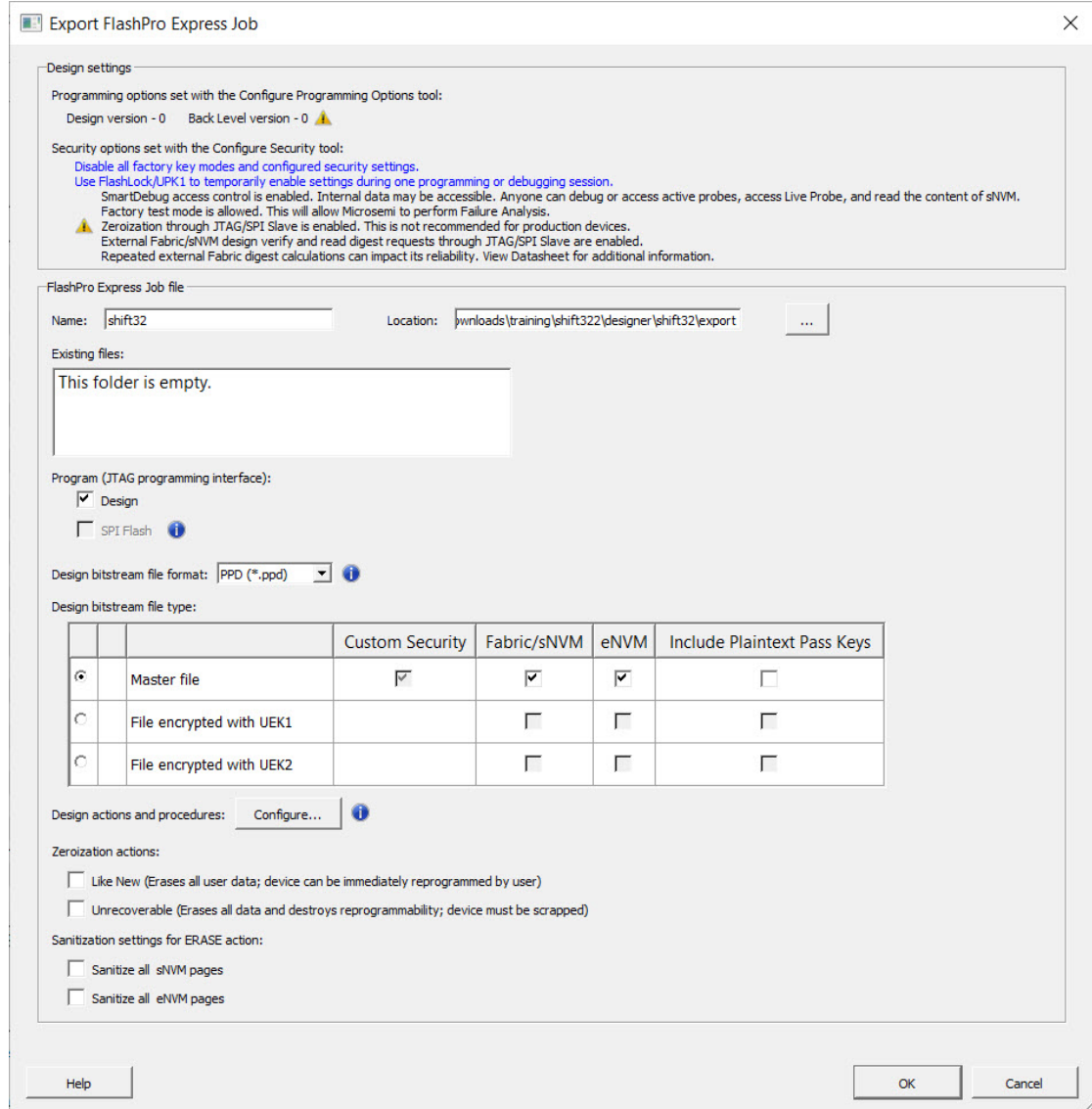
- **Security options set with the Configure Security tool-** Provides a brief description of current security settings.
- **Note:** eNVM option is only visible when eNVM is used in the design. eNVM features are available for PolarFireSoC devices only.

**FlashPro Express Job file:**

- **Name** - Sets the name of your bitstream file. The default name is the design name.
- **Location** - Location of the file to be exported.

- **Existing files** - Lists any existing programming job files at the selected location.
- **Program (JTAG Programming Interface) - Existing files** - Click **Design** or **SPI Flash** or both to include the configured device chain with bitstream files and/or SPI flash binary file. JTAG is the default programming interface for PolarFire Devices. SPI Flash option is only available if SPI Flash is configured in **Configure Design Initialization Data and Memories** tool. Otherwise it remains grayed out. Info message appears based on the configuration selected.
- **Design bitstream file format**- Lists all the available bitstream files, one of which will be included in the programming job for the current target device. The format of the Bitstream file can be selected from the **Format** drop down menu. PPD is the default bitstream file format.
- **Design bitstream file type**
  - File to Program at trusted facility(only available when the design is configured with Bitstream Encryption with Default Key in the [Configure Security tool](#)- Click to enable programming for Fabric/sNVM and eNVM bitstream components at a trusted facility.
  - Master file to program at trusted facility (available when the design is configured with Custom Security options in the Configure Security tool or when the design is configured with Permanent Locks in the Configure Permanent Locks for Production tool) – Click to include:
    - Fabric/sNVM.
    - eNVM
    - Include Plaintext Pass Keys into the bitstream files to be programmed at a trusted facility. It is available if UPK1 is not permanently locked and the Update Policy Fabric /sNVM or eNVM (available in PolarFire SoC only) is passcode protected or any of program, authenticate, or verify action is locked.

**Figure 10-20. Export FlashPro Express Job Dialog Box with Custom Security Options Set with the Configure Security Tool**



**Notes:**

- Custom Security is always programmed in the Master file.
- Security-only programming must be performed only on erased or new devices. If performed on a device with fabric programmed, the fabric will be disabled after performing security-only programming. You must reprogram the fabric to re-enable it.
- File encrypted with UEK1 to program at untrusted facility or for Broadcast field update – Click to include:
  - Fabric/sNVM.
  - eNVM
  - Include Plaintext Pass Keys into the bitstream files to be programmed is available if UPK1 is not permanently locked and the Update Policy Fabric /sNVM or eNVM (available in PolarFireSoC only) is passcode protected or any of program, authenticate or verify action is locked.
- File encrypted with UEK2 to program at untrusted facility or for Broadcast field update - Click to include:

- Fabric/sNVM.
- eNVM
- Include Plaintext Pass Keys into the bitstream files to be programmed is available if UPK1 is not permanently locked and the Update Policy Fabric /sNVM or eNVM (available in PolarFireSoC only) is passcode protected or any of program, authenticate or verify action is locked.

**Note:** Refer to Export Pass Keys for Plaintext to check the availability of Export Pass keys for Plaintext option for the following Bitstream files.

- **Zeroization Actions** – Click the **Like New** option to erase all user data to make the device like new and reprogram the device immediately, or click the **Unrecoverable** option to erase all data and destroy reprogrammability (in this case, the device must be scrapped).

#### **Prepare Design for Production Programming in FlashPro Express**

After you have exported a programming job you can handoff this programming job to the FlashPro Express tool for production programming. To do so:

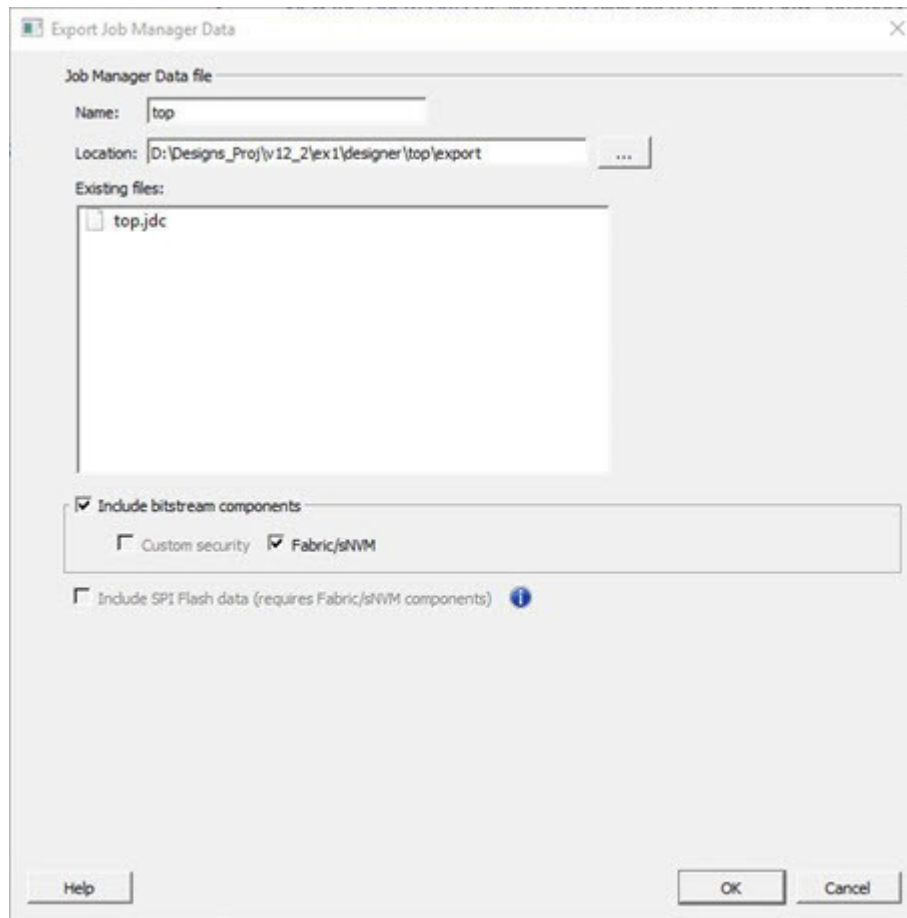
In FlashPro Express, from the **File** menu choose **Create Job Project From a Programming Job**. You will be prompted to specify the Programming Job location that you just exported from Libero and the location of where to store the Job Project. The Job Project name automatically uses the programming job name and cannot be changed. Click **OK** and a new Job Project will be created and opened for production programming.

### **10.3 Export Job Manager Data**

Job Manager is Microchip's HSM-based security software for job management.

As a part of the SPPS flow, the Export Job Manager Data dialog box allows a design engineer (user) to export Libero design data to Job Manager. Exported data is used by an operation engineer (OE) using Job Manager to prepare the manufacturing process for HSM or non-HSM flow.

Figure 10-21. Export Job Manager Data Dialog Box



**Programming data file** – export Job Data Container (JDC) file.

**Name** - All names use a prefix as shown in your software.

**Location** - Location of the file to be exported.

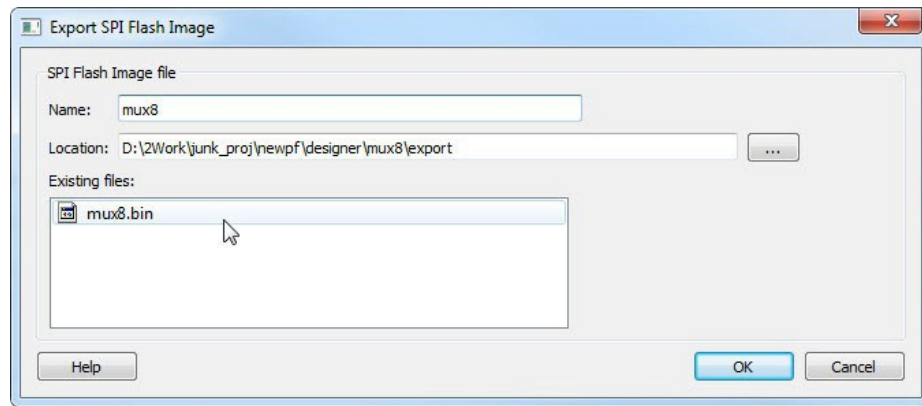
**Existing programming job files** - Lists any existing programming job files already in your project.

**Include bitstream components** - Lists the components of the design that can be saved to the file.

## 10.4 Export SPI Flash Image

This tool depends on the “Configure Design Initialization Data and Memories” tool. The SPI Flash configuration can be exported to a binary file here. Use this dialog to export a SPI Flash Image file.

Figure 10-22. Export SPI Flash Image



#### 10.4.1 Name

This is the top level design name by default. Use this field to change the default name. SPI Flash Image files are exported in binary format and have the \*.bin file extension and are named <design\_name>.bin.

#### 10.4.2 Location

The default location for the exported image file is <project\_folder>\designer\<top\_level\_design>\export. Use the browse button to navigate to and specify a different location for the exported SPI Flash Image file.

#### 10.4.3 Existing files

Existing SPI Flash Image files are listed.

### 10.5 Export Pin Report

In the Design Flow window, expand **Handoff Design for Production**. Right-click **Export Pin Report** to export a pin report.

The Export Pin Reports dialog box opens. Click **Browse** to navigate to a disk location where you want the pin report to be saved to.

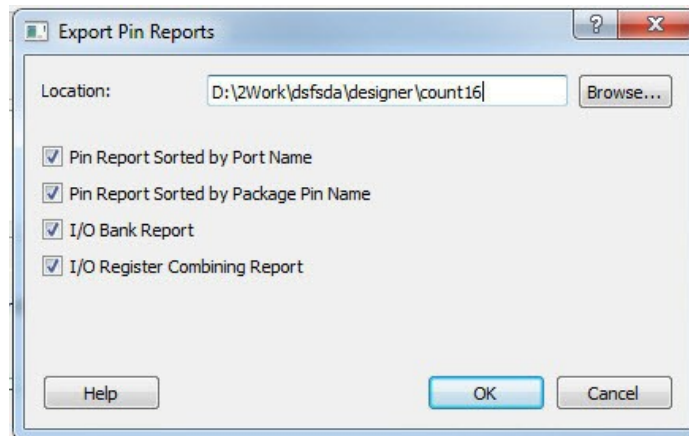
Check the checkbox to make your selections:

- Pin Report sorted by Port Name
- Pin Report sorted by Package Pin Name
- I/O Bank Report
- I/O Register Combining Report

The pin report lists the pins in your device sorted according to your preference: sort by Port Name or Sorted by Package Pin Name. The pin report generates two files:

- <design>\_pinrpt\_name.rpt - Pin report sorted by name.
- <design>\_pinrpt\_number.rpt - Pin report sorted by pin number. You must select at least one report. Export Pin Report generates a Bank Report by default; the filename is <design>-bankrpt.rpt. Export Pin Report also generates an I/O Register Combining Report listing the I/Os which have been combined into a Register for betting timing performance.

Figure 10-23. Export Pin Report Dialog Box



## 10.6 Export BSDL File

Double-click Export BSDL File (in the Libero SoC Design Flow window, **Handoff Design for Production > Export BSDL File**) to generate the BSDL File report to your [Design Report](#).

The BSDL file provides a standard file format for electronics testing using JTAG. It describes the boundary scan device package, pin description and boundary scan cell of the input and output pins. BSDL models are available as downloads for many Microchip SoC devices.

See the Microchip website for more information on BSDL Models.

## 10.7 Export IBIS Model

**Note:** Export Input Output Buffer Information Specification (IBIS) is supported for all the “T” devices of PolarFire Family except the “XT” device.

Export IBIS feature is supported in the following two stages:

- Pre-Layout Flow
- Post-Layout Flow

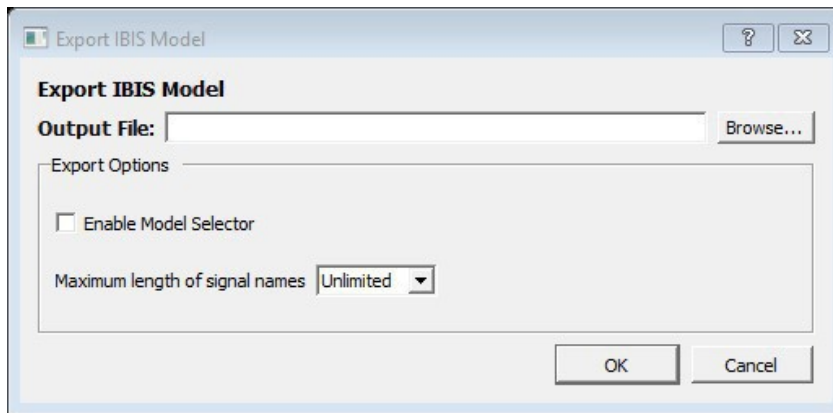
To export the IBIS Model in the Pre-Layout Flow, the design has to pass until the Synthesis/Compile stage. You need to provide the I/O placement and configuration details in the I/O PDC files to export the resultant IBIS files in Pre-Layout Flow.

To export the IBIS Model in the Post-Layout Flow, the design has to pass until the Place and Route stage. If a design has run through the Place and Route tool, the IBIS model for the Post-Layout Flow will be exported by default.

Double-click **Export IBIS Model** (in the Libero SoC Design Flow window, **Handoff Design for Production > Export IBIS Model**) to open the Export IBIS Model dialog box.



Figure 10-24. Export IBIS Model Dialog Box



### 10.7.1 Output File

Click **Browse** to specify the location to export the IBIS output file along with the output file name.

### 10.7.2 Export Options

The IBIS report \*.ibs file exported from Libero SoC can optionally support the [Model Selector] keyword as specified in the IBIS [5.0 Specifications](#). To generate Model Selector support in the exported IBIS file, check the **Enable Model Selector** box in the Export Options. Using this, the user can also optionally limit the maximum signal name to 40 characters.

Click **OK** to export the IBIS Model.

The IBIS model report provides an industry-standard file format for recording parameters like driver output impedance, rise/fall time, and input loading, which may then be used by software applications such as Signal Integrity tools or IBIS simulators.

The exported IBIS file has the file extension \*.ibs (named <root>.ibs) and is displayed in the Files tab.

In the [Pin] section of the IBIS \*.ibs file, listed under the model\_name are the Model Selector tag. The IBIS \*.ibs file has a [Model Selector] section that describes the model selector and its list of models. The Model Selector tag in the [Pin] section establishes the relationship between the pin and the [Model Selector].

Figure 10-25. Model Selector \*.ibs File

[Pin]	signal_name	model_name	R_pin	L_pin	C_pin
AE33	PAD_O<27>	LVCN0515_Bank15Group1_asiod	0.75142	5.68346e-009	2.14353e-012
E31	PAD_I<105>	LVCN0525_Bank1Group1_ddrio	0.609967	5.61206e-009	2.47835e-012
A2	PAD_O<33>	LVCN0525_Bank2Group1_ddrio	0.786904	7.52853e-009	2.87418e-012
<b>[Model Selector]</b>					
		LVCN0515_Bank15Group1_asiod			
		LVCN0515_6mA_MSIO			
		LVCN0515_2mA_MSIO			
		LVCN0515_4mA_MSIO			
<b>[Model Selector]</b>					
		LVCN0525_Bank1Group1_ddrio			
		LVCN0525_2mA_MEDFAST_DDRIO			
		LVCN0525_2mA_SLOW_DDRIO			
		LVCN0525_2mA_MED_DDRIO			
		LVCN0525_2mA_FAST_DDRIO			
		LVCN0525_4mA_MEDFAST_DDRIO			
		LVCN0525_4mA_SLOW_DDRIO			
		LVCN0525_4mA_MED_DDRIO			
		LVCN0525_4mA_FAST_DDRIO			
<b>[Model Selector]</b>					
		LVCN0525_Bank2Group1_ddrio			
		LVCN0525_4mA_SLOW_DDRIO			
		LVCN0525_4mA_MED_DDRIO			
		LVCN0525_4mA_MEDFAST_DDRIO			
		LVCN0525_4mA_FAST_DDRIO			
		LVCN0525_2mA_SLOW_DDRIO			
		LVCN0525_2mA_MED_DDRIO			
		LVCN0525_2mA_MEDFAST_DDRIO			
		LVCN0525_2mA_FAST_DDRIO			

Model Selector for Pin AE33

Model Selector for Pin E31

Model Selector for Pin A2

The advantage of Model Selector feature is that you can load the \*.ibs file from Libero SoC into Signal Integrity applications or IBIS simulators and switch the I/O to different models for individual I/Os on-the-fly in the tools. There is no need to go back to the Libero SoC I/O Attribute Editor to change the I/O settings and run Compile to switch to different I/O settings.

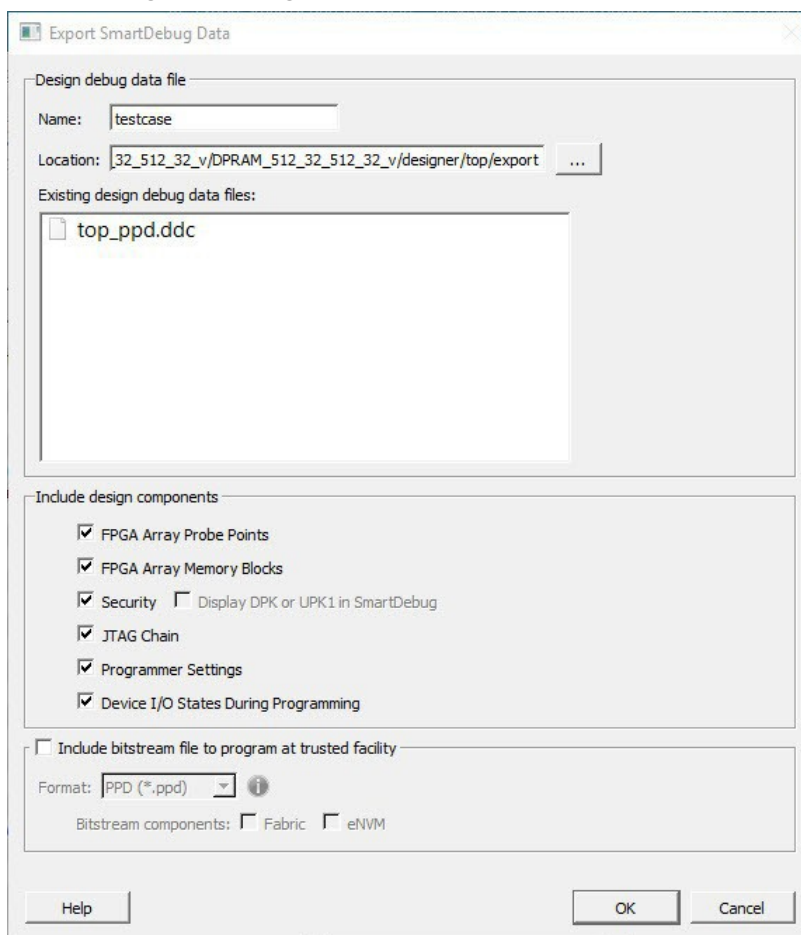
See the [Microchip Website for more information on IBIS Models](#).

## 11. Export SmartDebug Data (Libero SoC)

Export SmartDebug Data allows the export of SmartDebug Data from Libero to be handed off to the Standalone SmartDebug environment.

In the Libero SoC Design Flow window, expand **Handoff Design for Debugging**, right-click **Export SmartDebug Data** and click **Export** to open the Export SmartDebug Data dialog box. Specify the design debug data file (\*.ddc) to be exported. This file is also used as one of the ways to create a Standalone SmartDebug project. See the following figure for an example.

**Figure 11-1. Export SmartDebug Data Dialog Box**



**Note:** SmartDebug data can be exported without connecting the hardware.

### Design debug data file (\*.ddc)

- **Name:** The name of the design.
- **Location:** The location of the exported debug file. By default, the \*.ddc file is exported to the <project\_location>/designer/<design>/export folder and has the \*.ddc file extension.
- **Existing Design Debug Data Files:** The existing \*.ddc file, if any, in the export folder. SmartDebug data can be exported after you run Generate FPGA Array Data for the design in the Libero Design Flow. You can also directly export SmartDebug data after running Synthesize on the design. Other tools, such as Place and Route, Generate FPGA Array Data, and so forth) are implicitly run before the Export SmartDebug Data dialog box is displayed.
- **Include design components:** A DDC file can contain the following components:
  - **FPGA Array Probe Points** – When checked, Libero SoC exports Live and Active probes information (<design>\_probe.db file) into the \*.ddb container file.

- **FPGA Array Memory Blocks** – When checked, Libero SoC exports information about FPGA memories (<design>\_sii\_block.db) into the \*.ddb container file:
  - names and addresses of the memory blocks instantiated by the design
  - data formats selected by the user in the design
- **sNVM** – When checked, Libero SoC exports sNVM components.
- **Security** – This contains the security locks, keys, and security policy information needed for debug. This may be default or custom security (<design>.spm file). It is hidden if security is not supported for the device.
  - **Display DPK or UPK1 in SmartDebug** - This option is enabled only when custom security is provided, and is unchecked by default.
- **JTAG Chain** (device chain information configured using Programming Connectivity and Interface in Libero). When checked, Libero SoC exports chain data including devices, their programming files if loaded, device properties, and so on (<design>.pro file). If JTAG chain is unchecked, the default JTAG chain with Libero design device only is added to the \*.ddc file.
- **Programming Options set with Configure Programming Options tool**- Displays the Design version and Back Level version set in the Configure Programming Options Wizard. They are read only options and cannot be modified.
- Programming Options set with Configure Programming Options tool- Displays the Design version and Back Level version set in the Configure Programming Options Wizard. They are read only options and cannot be modified.
- **Device I/O States During Programming** (<design>.ios file) – This setting is used by some SmartDebug features, for example, for programming **sNVM**. It is NOT used during device programming in SmartDebug; programming files used to program devices already have I/O states data.

In addition, you can include bitstream file information, which can be used for programming the device in Standalone SmartDebug.

**Note:** Info and warning messages appear based on the value set for the Back Level version.

Include Bitstream file to program at trusted facility

- Bitstream components: Fabric
- Bitstream components: **sNVM**
- Bitstream components: Custom security (this option is only visible if Custom Security Option is selected in Configure Security wizard).

**Note:** If the Fabric Bitstream component is programmed, the sNVM Bitstream component must be programmed and it is selected by default. In this case, sNVM is disabled for the user to unselect. Otherwise, the sNVM component can either be enabled or disabled for programming.

The default location of the DDC file is: <Libero\_Project\_directory>/designer/<design\_name>/export.

The DDC file can be exported to any user-specified location if the location has read and write permission.

## 12. References

### 12.1 Archive Project Dialog Box

The Archive Project dialog box enables you to create an archive (\*.zip file) of your existing project and save it at the specified location. This is useful if you want to create a quick zip file of your project. Internally, **Archive Project** and **Save As Project** do the same task, but if you are saving your Libero application to the SVN repository, it is important to archive your project. When you do an SVN checkout after checkin, there is a chance that the timestamps will be outdated and the tool states will remain invalidated. Archive Project helps in retaining the tool states by preserving the database in compressed format.

To access this dialog box, choose **Archive Project** from the **Project** menu.

**Figure 12-1. Archive Project Dialog Box**



#### Project Location

Accept the default location or Browse to a new location where the archive (\*.zip) file is to be created and stored. The \*.zip file is named <project\_name>.zip.

#### Content

**Copy links locally** - Select this checkbox to copy the links from your current project into your archive. If you do not select this checkbox, the links will not be copied and you must add them manually.

**Files:** Specifies what kind of files are archived.

- **All** - Includes in the archive all your project and source files; the state of the project is retained.
- **Project files only** – Includes in the archive only the project-related information required to retain the state of the project.
- **Source files only** - Includes in the archive only the source files. This means the configuration of all the tools in the tool chain is retained but the states are not. Source files means constraint information and component information available in the component, hdl and smartgen directories.

Files are archived as shown in the table below for the different selections.

Folder Name	Files		
	All	Project Files Only	Source Files Only
component	All Files	All Files	All Files
constraint	All Files	All Files	All Files
hdl	All Files	All Files	All Files
stimulus	All Files	All Files	All Files
viewdraw	All Files	All Files	All Files
smartgen	All Files	All Files	All Files
firmware	All Files	All Files	All Files
CoreConsole	All Files	All Files	All Files
SoftConsole/Keil/IAR	All Files	All Files	All Files
simulation	All Files	*.ini, *.bfm, *.do., *.vec	*.ini, *.bfm, *.do., *.vec
synthesis	All Files	*.edn, *.vm, *.sdc, *.so, *.prj, *.srr, *.v, run_options.txt, synplify.log	*.prj files
Designer/impl1	All Files	All Files	*.ide_des files
Designer/<root>	All Files	All Files	Not archived
tooldata	All Files	All Files	All Files

**Note:** \*.edn files are not supported in PolarFire.

## 12.2 Adding or Modifying Bus Interfaces in SmartDesign

SmartDesign supports automatic creation of data driven configurators based on HDL generics/parameters. You can add a bus interface from your HDL module or you can add it from the Catalog.

To add a bus interface using your custom HDL block:

If your block has all the necessary signals to interface with the AMBA bus protocol (such as address, data, and control signals):

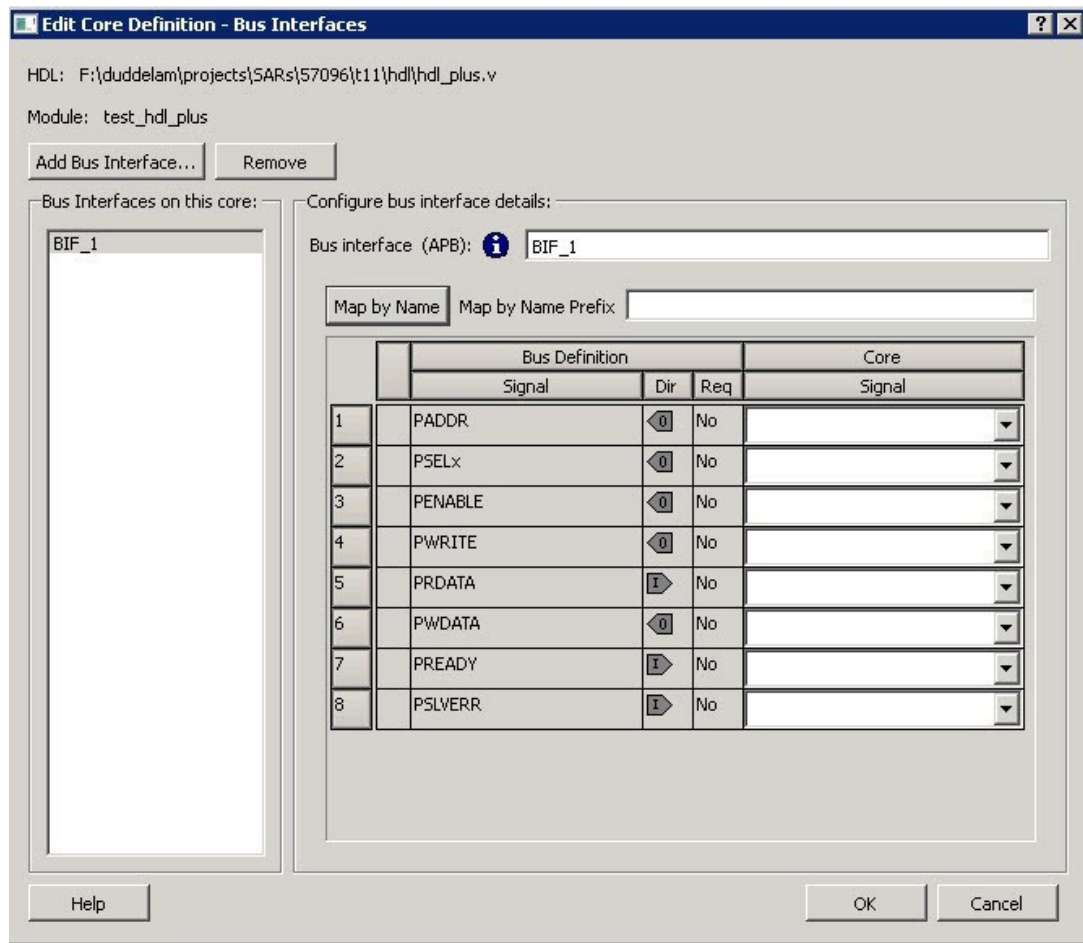
1. Right-click your custom HDL block and choose **Create Core from HDL**. The Libero SoC creates your core and asks if you want to add bus interfaces.
2. Click **Yes** to open the Edit Core Definition dialog box and add bus interfaces. Add the bus interfaces as necessary.
3. Click **OK** to continue.

Now your instance has a proper AMBA bus interface on it. You can manually connect it to the bus or let Auto Connect find a compatible connection.

To add (or modify) a bus interface to your Component:

1. Right-click your Component and choose **Edit Core Definition**. The Edit Core Definition dialog box opens, as shown in the figure below.

Figure 12-2. Edit Core Definition Dialog Box



2. Click **Add Bus Interface**. Select the bus interface you wish to add and click **OK**.
3. If necessary, edit the bus interface details.
4. Click **Map by Name** to map the signals automatically. Map By Name attempts to map any similar signal names between the bus definition and pin names on the instance. During mapping, bus definition signal names are prefixed with text entered in the **Map by Name Prefix** field.
5. Click **OK** to continue.

#### Bus Interface Details

- **Bus Interface:** Name of bus interface. Edit as necessary. **Bus Definition:** Specifies the name of the bus interface. **Role:** Identifies the bus role (master or slave).
- **Vendor:** Identifies the vendor for the bus interface.
- **Version:** Identifies the version for the bus interface.

#### Configuration Parameters

Certain bus definitions contain user configurable parameters.

- **Parameter:** Specifies the parameter name.
- **Value:** Specifies the value you define for the parameter.
- **Consistent:** Specifies whether a compatible bus interface must have the same value for this bus parameter. If the bus interface has a different value for any parameters that are marked with consistent set to **yes**, this bus interface will not be connectable.

#### Signal Map Definition

The signal map of the bus interface specifies the pins on the instance that correspond to the bus definition signals. The bus definition signals are shown on the left, under the **Bus Interface Definition**. This information includes the name, direction and required properties of the signal.

The pins for your instance are shown in the columns under the Component Instance. The signal element is a drop-down list of the pins that can be mapped for that definition signal.

If the Req field of the signal definition is Yes, you must map it to a pin on your instance for this bus interface to be considered legal. If it is No, you can leave it unmapped.

### 12.2.1 Bus Interfaces

When you add a bus interface the Edit Core Definition dialog box provides the following Microchip Libero SoC-specific bus interfaces:

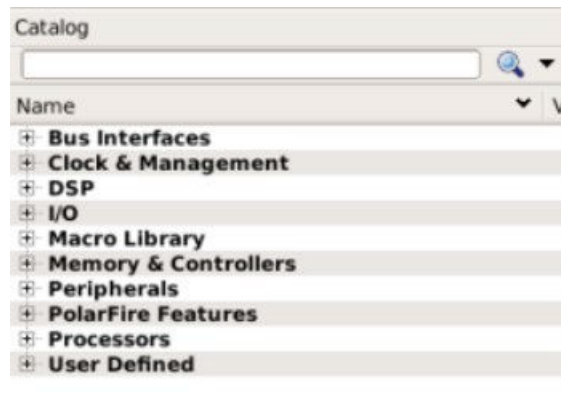
- AHB – Master, Slave, Mirrored Master, MirroredSlave
- APB – Master, Slave, Mirroredmaster, MirroredSlave
- AXI – Master, Slave, MirroredMaster, MirrorSlave, System
- AXI 4 - Master, Slave, MirroredMaster, MirrorSlave

## 12.3 Catalog

In the Libero SoC, from the **View** menu choose **Windows > Catalog**.

The Catalog displays a list of available cores, busses, and macros (see image below).

**Figure 12-3. Libero SoC Catalog**



From the Catalog, you can create a component from the list of available cores, add a processor or peripheral, [add a bus interface to your SmartDesign component](#), instantiate simulation cores or add a macro (Arithmetic, Basic Block, etc.) to your SmartDesign component.

Double-click a core to configure it and add it to your design. Configured cores are added to your list of Components/Modules in the Design Explorer.

Click the Simulation Mode checkbox to instantiate simulation cores in your [SmartDesign Testbench](#). Simulation cores are basic cores that are useful for stimulus, such as driving clocks, resets, and pulses.

#### Viewing Cores in the Catalog

The font indicates the status of the core:

- Plain text - In vault and available for use
- Asterisk after name (\*) - Newer version of the core (VLN) available for download
- *Italics* - Core is available for download but not in your vault
- ~~Strikethrough~~ - core is not valid for this version of Libero SoC

The colored icons indicate the license status. Blank means that the core is not license protected in any way. Colored icons mean that the core is license protected, with the following meanings:



**Green Key** - Fully licensed; supports the entire design flow.

**Yellow Key** - Has a limited or evaluation license only. Precompiled simulation libraries are provided, enabling the core to be instantiated and simulated within Libero SoC. Using the Evaluation version of the core it is possible to create and simulate the complete design in which the core is being included. The design is not synthesizable (RTL code is not provided). No license feature in the license.dat file is needed to run the core in evaluation mode. You can purchase a license to generate an obfuscated or RTL netlist.

**Yellow Key with Red Circle** - License is protected; you are not licensed to use this core.


Right-click any item in the Catalog and choose Show Details for a short summary of the core specifications. Choose Open Documentation for more information on the Core. Right-click and choose Configure Core to open the core generator.

Click the **Name** column heading to sort the cores alphabetically.

You can filter the cores according to the data in the Name and Description fields. Type the data into the filter field to view the cores that match the filter. You may find it helpful to set the Display setting in the Catalog Options to **List cores alphabetically** when using the filters to search for cores. By default the filter contains a beginning and ending '\*', so if you type 'controller' you get all cores with controller in the core name (case insensitive search) or in the core description. For example, to list all the Accumulator cores, in the filter field type:

accu

### Catalog Options

Click the Options button  (or the drop-down arrow next to it) to import a core, reload the Catalog, or modify the Catalog Options.


You may want to import a core from a file when:

- You do not have access to the internet and cannot download the core, or
- A core is not complete and has not been posted to the web (you have an evaluation core)

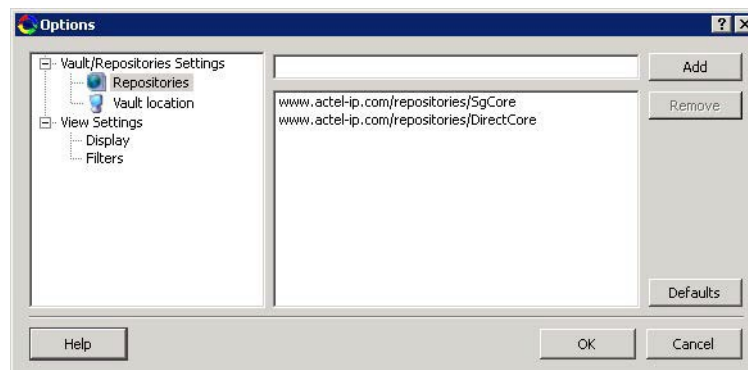
### Manually Downloading MegaVaults and Individual CPZ files

When Libero is used in an environment without automatic access to Microchip's online IP repositories via the Internet; see this article explaining [how to download MegaVaults and individual CPZ files](#).

## 12.3.1 Catalog Options Dialog Box

The Catalog Options dialog box (as shown below) enables you to customize your [Catalog](#). You can add a repository, set the location of your vault, and change the View Settings for the Catalog. To display this dialog box, click the Catalog Options button .

**Figure 12-4. Catalog Display Options Dialog Box**



- **Vault/Repositories Settings**
  - **Repositories:**  
A repository is a location on the web that contains cores that can be included in your design.

The Catalog Options dialog box enables you to specify which repositories you want to display in your Vault. The Vault displays a list of cores from all your repositories, and the [Catalog](#) displays all the cores in your Vault.

The default repository cannot be permanently deleted; it is restored each time you open the Manage Repositories dialog box.

Any cores stored in the repository are listed by name in your Vault and Catalog; repository cores displayed in your Catalog can be filtered like any other core.

Type in the address and click the **Add** button to add new repositories. Click the **Remove** button to remove a repository (and its contents) from your Vault and Catalog. Removing a repository from the list removes the repository contents from your Vault.

#### – **Vault location**

Use this option to choose a new vault location on your local network. Enter the full domain pathname in the Select new vault location field. Use the format:

```
\\server\share
```

and the cores in your Vault will be listed in the Catalog.

**Set ENV variable to set vault location** - In addition to setting the vault location using the Catalog dialog box, you can set the vault location using the environment variable `MSCC_IDE_VAULT_LOCATION`. Setting the vault through the environment variable takes precedence over all other options to set vault location.

To set the vault location on Linux, type the following command:

```
setenv MSCC_IDE_VAULT_LOCATION /home/temp_dir
```

To set the vault location on Windows:

Add a new environment variable `MSCC_IDE_VAULT_LOCATION` in System Properties and specify your vault location.

#### – **Read only vault**

In read only Mega Vault mode, you cannot download, add, or remove cores. However, you can configure and generate cores by creating a temporary extract location to extract the core. This temporary extract location can be set by setting the environment variable `MSCC_IDE_VAULT_EXTRACT_LOCATION`. By setting this environment variable, your configured cores are retained across sessions.

To set the extract location on Linux, type the following command:

```
setenv MSCC_IDE_VAULT_EXTRACT_LOCATION /home/vault_extract
```

To set the extract vault location on Windows:

Add a new environment variable `MSCC_IDE_VAULT_EXTRACT_LOCATION` in System Properties and specify your extract location.

If you do not specify the extract location, a temporary location will be created by Libero and it will be accessed only while the current session is active. If the session is no longer active, the temporary extract location will be cleaned up by Libero. If you specify the extract location, it will be available for any instance of libero on that machine, and it is your responsibility to clean up the extract location.

### • **View Settings**

#### – **Display**

**Group cores by function** - Displays a list of cores, sorted by function. Click any function to expand the list and view specific cores.

**List cores alphabetically** - Displays an expanded list of all cores, sorted alphabetically. Double click a core to configure it. This view is often the best option if you are using the filters to customize your display.

**Show core version** - Shows/hides the core version.

#### – **Filters**

**Filter field** - Type text in the Filter Field to display only cores that match the text in your filter. For example, to view cores that include 'sub' in the name, set the Filter Field to **Name** and type **sub**.

**Display only latest version of a core** - Shows/hides older versions of cores; this feature is useful if you are designing with an older family and wish to use an older core.

**Show all local and remote cores** - Displays all cores in your Catalog.

**Show local cores only** - Displays only the cores in your local vault in your Catalog; omits any remote cores.

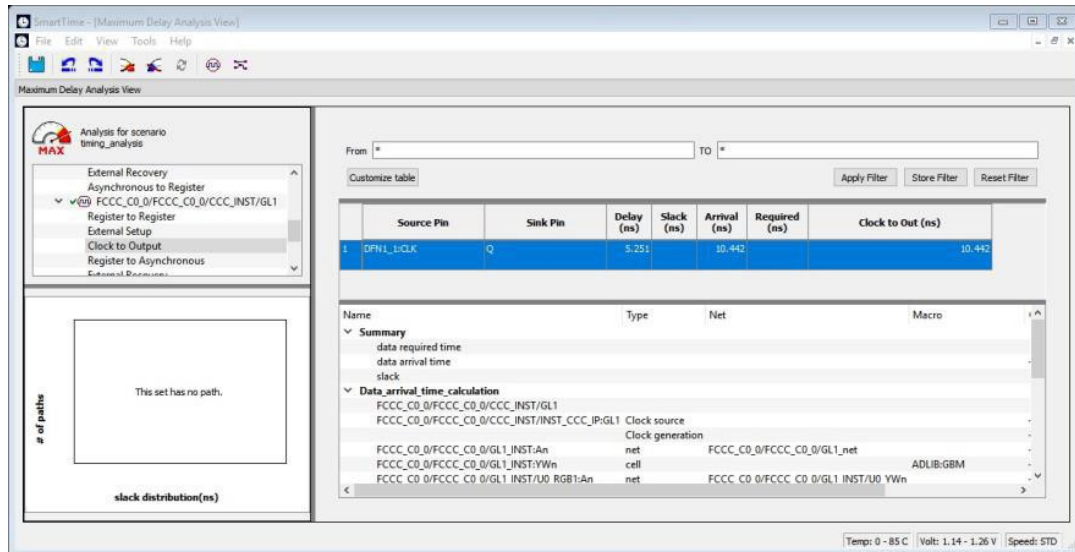
**Show remote cores that are not in my vault** - Displays remote cores that have not been added to your vault in your Catalog.

## 12.4 Changing Output Port Capacitance

Output propagation delay is affected by both the capacitive loading on the board and the I/O standard. The I/O Attribute Editor in Chip Planner provides a mechanism for setting the expected capacitance to improve the propagation delay model. SmartTime automatically uses the modified delay model for delay calculations.

To change the output port capacitance and view the effect of this change in SmartTime Timing Analyzer, refer to the following example. The figure below shows the delay from DFN1 to output port Q. It shows a delay of 6.603 ns based on the default loading of 5 pF.

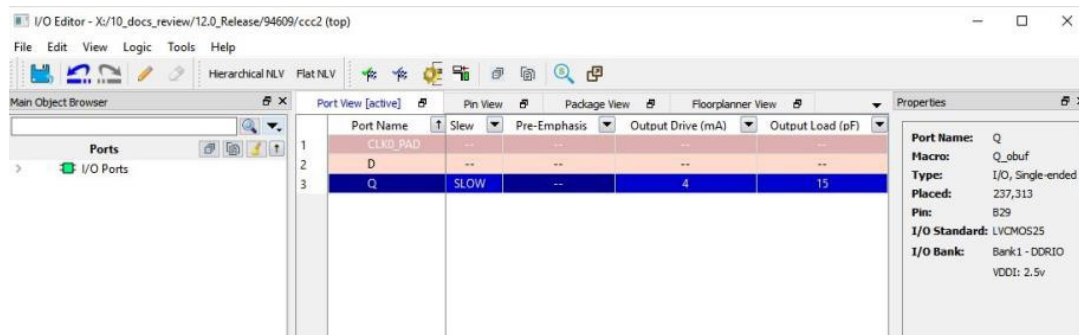
**Figure 12-5. Maximum Delay Analysis View**



If your board has output capacitance of 15 pf on Q, you must perform the following steps to update the timing number:

1. Open the I/O Attribute Editor and change the output load to 15 pf.

**Figure 12-6. I/O Attribute Editor View**



2. Select **File > Save**.
3. Select **File > Close**.
4. Open the SmartTime Timing Analyzer.

You can see that the Clock to Output delay changed to 5.952 ns.

## 12.5 Core Manager

The Core Manager only lists cores that are in your current project. If any of the cores in your current project are not in your vault, you can use the Core Manager to download them all at once.

For example, if you download a sample project and open it, you may not have all the cores in your local vault. In this instance you can use the Core Manager to view and download them with one click. Click **Download All** to add any missing cores to your vault. To add any individual core, click the green download button.

To view the Core Manager, from the **View** menu, choose **Windows > Cores**. The following table describes the column headings in the Core Manager.

**Table 12-1. Columns in the Core Manager**

Column	Description
Name	Name of the core.
Vendor	Source of the core.
Core Type	Core type.
Version	Version of the core used in your project. If the version is a later version than the one in your vault, click Download All to download the latest version.

## 12.6 Configure Permanent Locks for Production

Configure Permanent Locks for Production is a GUI-based tool that guides the user step by step on how to configure the Permanent Locks for Production. The wizard has six steps/pages executed in sequential order. One Time Programmable (OTP) settings in the Permanent Locks page are applied to configured Security settings from the Configure Security tool. The subsequent pages have read only fields, which will be affected by Permanent Lock settings. These settings can only be configured by the [Configure Security](#) tool.

If you configure any Permanent Lock settings, you will be forced to go through each page to review the Security settings to make sure they are as desired. The settings cannot be changed once they have been programmed.

1. [Permanent Locks](#)
2. [User keys in Configure Security](#)
3. [Update Policy in Configure Security](#)
4. [Debug Policy in Configure Security](#)
5. [Microsemi Factory Access in Configure Security](#)
6. [JTAG/SPI Slave Commands Policy in Configure Security](#)

### Summary Window

The summary window displays the summary of the current page configuration settings. Based on the selection made in the first page, the summary for the subsequent pages change. The window will scroll to the current page as you move from page to page.

### Back

Click **Back** to return to the previous step.

### Next

Click **Next** to proceed to the next step.

### Finish

Click **Finish** to complete the configuration after executing the all the steps in sequential order.

### Save Summary to File

Click **Save Summary to File** to save the display in the Summary field to a file.

## 12.7 Importing Source Files – Copying Files Locally

Designer in Libero SoC cannot import files from outside your project without copying them to your local project folder. You may import source files from other locations, but they are always copied to your local folder. Designer in Libero SoC always audits the local file after you import; it does not audit the original file.

When the Project Manager asks you if you want to copy files "locally", it means 'copy the files to your local project folder'. If you do not wish to copy the files to your local project folder, you cannot import them. Your local project folder contains [files](#) related to your Libero SoC project.

Files copied to your local folders are copied directly into their relevant directory: netlists are copied to the *synthesis* folder; source files are copied to *hdl* folder and constraint files to *constraint* folder, etc. The files are also added to the Libero SoC project; they appear in the Files tab.

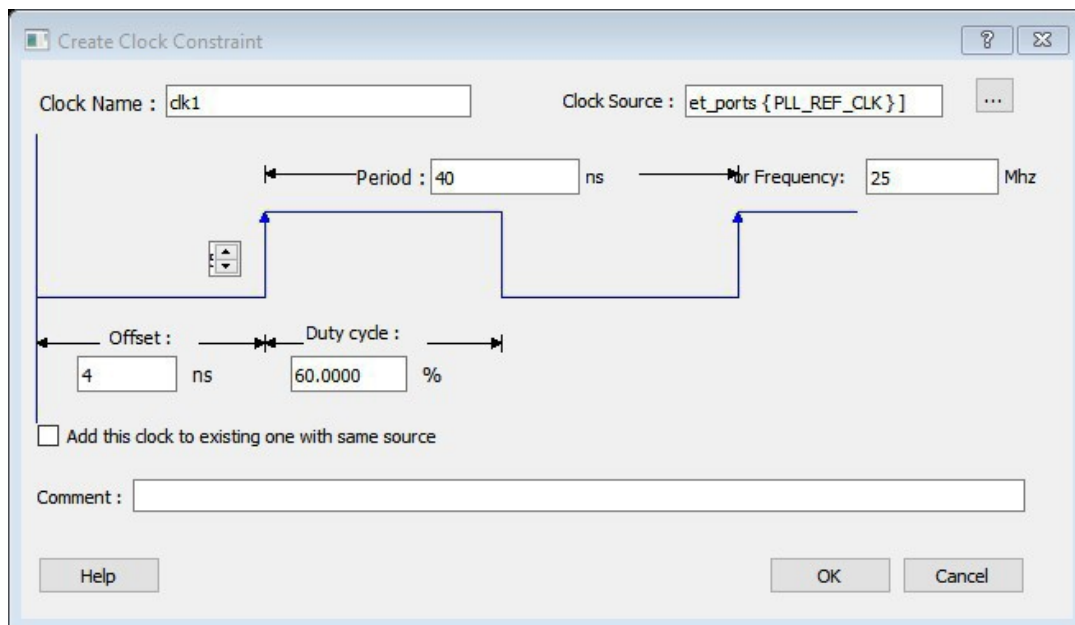
## 12.8 Create Clock Constraint Dialog Box

Use this dialog box to enter a clock constraint setting.

It displays a typical clock waveform with its associated clock information. You can enter or modify this information, and save the final settings as long as the constraint information is consistent and defines the clock waveform completely. The tool displays errors and warnings if information is missing or incorrect.

To open the Create Clock Constraint dialog box (shown below) from the SmartTime Constraints Editor, choose **Constraints > Clock**.

**Figure 12-7. Create Clock Constraint Dialog Box**



### Clock Source

Enables you to choose a pin from your design to use as the clock source.

The drop-down list is populated with all explicit clocks. You can also select the Browse button to access all potential clocks. The **Browse** button displays the [Select Source Pins for Clock Constraint Dialog Box](#).

### Clock Name

Specifies the name of the clock constraint. This field is required for virtual clocks when no clock source is provided.

### Period

---

When you edit the period, the tool automatically updates the frequency value. The period must be a positive real number. Accuracy is up to 3 decimal places.

**Frequency**

When you edit the frequency, the tool automatically updates the period value.

The frequency must be a positive real number. Accuracy is up to 3 decimal places.

**Starting Clock Edge Selector**

Click the Up or Down arrow to use the rising or falling edge as the starting edge for the created clock.

**Offset**

Indicates the shift (in nanoseconds) of the first clock edge with respect to instant zero common to all clocks in the design.

The offset value must be a positive real number. Accuracy is up to 2 decimal places. Default value is 0.

**Duty Cycle**

This number specifies the percentage of the overall period that the clock pulse is high.

The duty cycle must be a positive real number. Accuracy is up to 4 decimal places. Default value is 50%.

**Add this clock to existing one with same source**

Check this box if you want to add a new clock constraint on the same source without overwriting the existing clock constraint. The new clock constraint name must be different than the existing name. Otherwise, the new constraint will overwrite the existing one even if you check this box.

**Comment**

Enables you to save a single line of text that describes the clock constraints purpose.

**See Also**

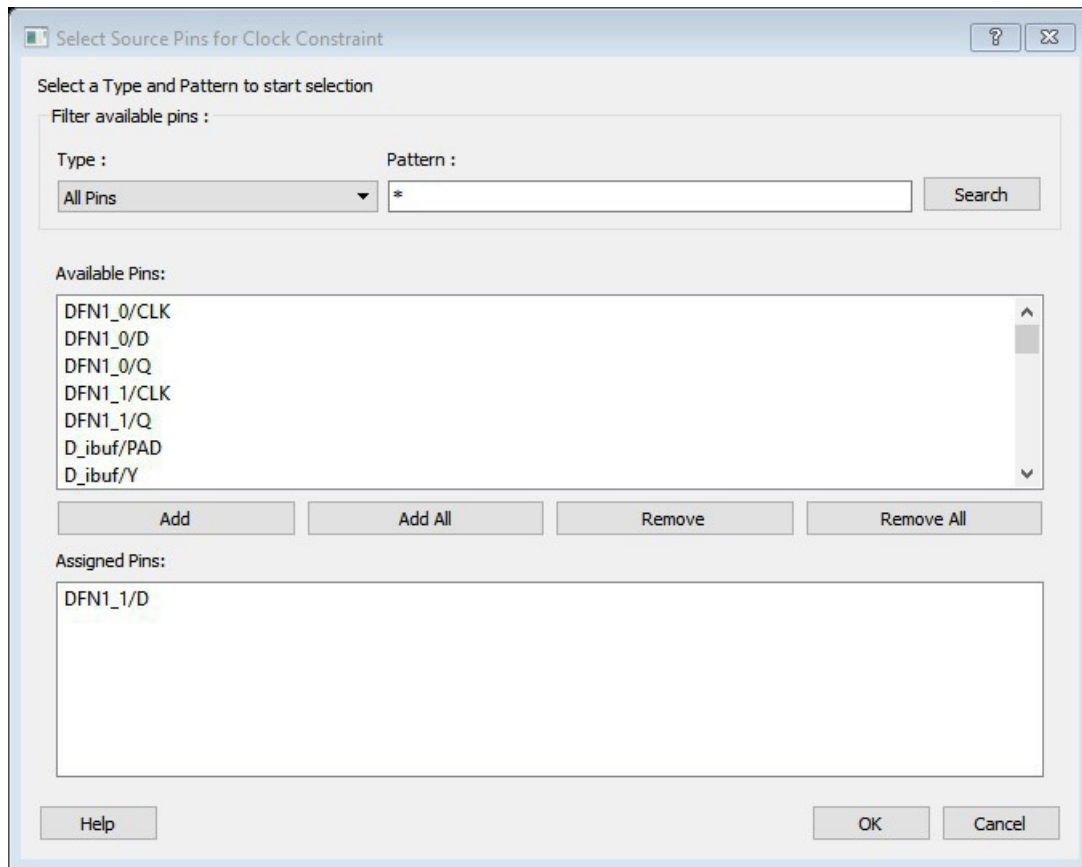
[Specifying Clock Constraints](#)

## 12.9 Select Source Pins for Clock Constraint Dialog Box

Use this dialog box to find and choose the clock source from the list of available pins.

To open the Select Source Pins for the Clock Constraint dialog box (shown below) from the SmartTime Constraints Editor, click the **Browse** button to the right of the Clock source field in the [Create Clock Constraint](#) dialog box.

Figure 12-8. Select Source Pins for Clock Constraint Dialog Box



#### Filter Available Pins

- **Type** – Displays the Type of the Available Pins in the design. The Pin Type options available for the Source are:
  - All Pins
  - Input Ports
  - All Nets
- **Pattern** – The default is \*, which is a wild-card match for all. You can specify any string value. Click **Search** to filter the available pins based on the specified pin Type and Pattern.

#### Available Pins

The list box displays the available pins. If you change the pattern value, the list box shows the available pins based on the filter.

Use **Add**, **Add All** to add the pins from the Available Pins list to Assigned Pins or **Remove**, **Remove All** to delete the pins from the Assigned Pins list.

#### Assigned Pins

Displays pins selected from the Available Pins list. Select Pins from this list and click **OK** to add the Source Pins for Clock Constraint.


#### See Also

[Specifying clock constraints](#)

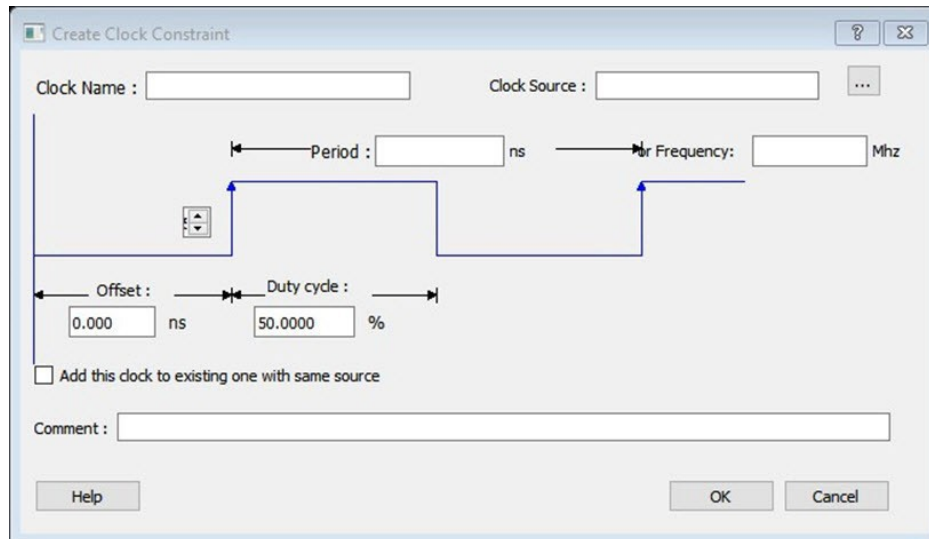
## 12.10 Specifying Clock Constraints

Specifying clock constraints is the most effective way to constrain and verify the timing behavior of a sequential design. Use clock constraints to meet your performance goals.

To specify a clock constraint:

1. Add the constraint in the [editable constraints grid](#) or open the [Create Clock Constraint](#) dialog box using one of the following methods:
  - Click the  icon in the Constraints Editor.
  - Right-click the **Clock** in the Constraint Browser and choose **Add Clock Constraint**.
  - Double-click **Clock** in the Constraint Browser.
  - Choose **Clock** from the Constraints drop-down menu (**Constraints > Clock**)  
The Create Clock Constraint dialog box appears (as shown below).

**Figure 12-9. Create Clock Constraint Dialog Box**



2. Select the pin to use as the clock source. You can click the **Browse** button to display the [12.9 Select Source Pins for Clock Constraint Dialog Box](#) (as shown below).

**Note:** Do not select a source pin when you specify a virtual clock. Virtual clocks can be used to define a clock outside the FPGA that is used to synchronize I/Os.

Use the Choose the Clock Source Pin dialog box to display a list of source pins from which you can choose. By default, it displays the explicit clock sources of the design. To choose other pins in the design as clock source pins, select **Filter available objects - Pin Type** as **Explicit clocks**, **Potential clocks**, **All Ports**, **All Pins**, **All Nets**, **Pins on clock network**, or **Nets in clock network**. To display a subset of the displayed clock source pins, you can create and apply a filter.

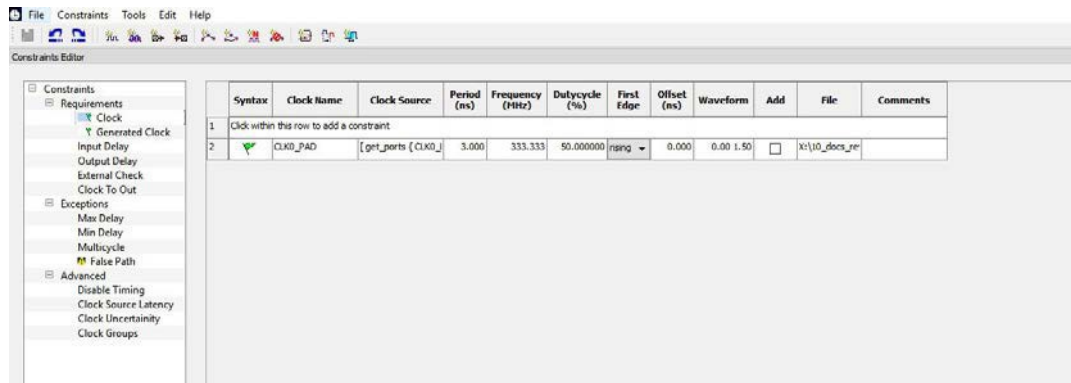
Multiple source pins can be specified for the same clock when a single clock is entering the FPGA using multiple inputs with different delays.

Click **OK** to save these dialog box settings.

3. Specify the **Period** in nanoseconds (ns) or **Frequency** in megahertz (MHz).
  4. Modify the **Clock Name**. The name of the first clock source is provided as default.
  5. Modify the **Duty cycle**, if needed.
  6. Modify the **Offset** of the clock, if needed.
  7. Modify the first edge direction of the clock, if needed.
  8. Select the check box for Add this clock to an existing one with the same source, if needed.
  9. Click **OK**. The new constraint appears in the Constraints List.
- Note:** When you choose **File > Save**, the Timing Constraints Editor saves the newly created constraint in the database.



Figure 12-10. Timing Constraint View



### 12.10.1 Create Generated Clock Constraint Dialog Box

Use this dialog box to specify generated clock constraint settings.

It displays a relationship between the clock source and its reference clock. You can enter or modify this information, and save the final settings as long as the constraint information is consistent. The tool displays errors and warnings if the information is missing or incorrect.

To open the Create Generated Clock Constraint dialog box (shown below) from the SmartTime Constraints Editor, choose **Constraints > Generated Clock**.

Figure 12-11. Create Generated Clock Constraint

Created Generated Clock Constraint

Reference Pin:

Clock Pin:

Clock Conditioning Circuitry

FPGA

Generated Clock Name :

☒ The generated frequency is such that :

$f(\text{clock}) = f(\text{reference}) * \frac{1}{1}$

☐ The generated clock edges are based on the reference edges :  Edges (example : 1 3 5)

☐ The edges are shifted by the following delays :  Edge Shift (example: 0.0 0.5 1.0)

The generated waveform is  the reference waveform.

☐ An External feedback is used to generate the clock.

PLL Output:

PLL Feedback:

☐ Phase shift is applied by PLL.

Phase shift :  degree

☐ Add this clock to existing one with same source

Master Clock:

Comment :

Help OK Cancel

### Clock Pin

Enables you to choose a pin from your design to use as a generated clock source.

The drop-down list is populated with all unconstrained explicit clocks. You can also select the Browse button to access all potential clocks and pins from the clock network. The Browse button displays the [Select Generated Clock Source](#) dialog box.

### Reference Pin

Enables you to choose a pin from your design to use as a generated reference pin. You can select the Browse button to access all the available reference pins. The Browse button displays the [Select Generated Clock Reference](#) dialog box.

### Generated Clock Name

Specifies the name of the Generated clock constraint. This field is required for virtual clocks when no clock source is provided.

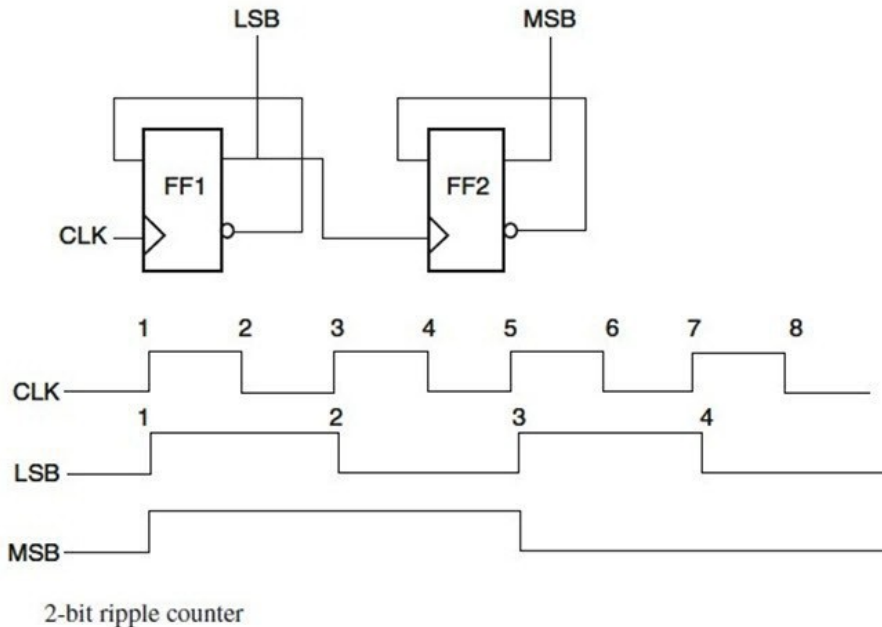
### Generated Frequency

Specify the values to calculate the generated frequency: a multiplication factor and/or division factor (must be positive integers) is applied to the reference clock to compute the generated clock.

### Generated Clock Edges

Frequency of the generated clock can also be specified by selecting the Generated Clock Edges option. Specify the integer values that represent the edges from the source clock that form the edges of the generated clock.

Three values must be specified to generate the clock. If you specify less than three, a tool tip indicates an error. The following example shows how to specify the clock edges.



If LSB is the generated clock from CLK clock source, the edge values must be [1 3 5].

If MSB is the generated clock from CLK clock source, the edge values must be [1 5 9].

### Edge Shift

Specify a list of three floating point numbers that represents the amount of shift, in library time units, that the specified edges are to undergo to yield the final generated clock waveform. These floating point values can be positive or negative. Positive value indicates a shift later in time, while negative indicates a shift earlier in time.

For example: An edge shift of {1 1 1} on the LSB generated clock, would shift each derived edge by 1 time unit. To create a 200MHz clock from a 100MHz clock, use edge {1 2 3} and edge shift {0 -2.5 -5.0}

### Generated Waveform

Specify whether the generated waveform is the same or inverted with respect to the reference waveform. Click **OK**.

### Phase

This field is primarily used to report the information captured from the CCC configuration process, and when constraint is auto-generated. Meaningful phase values are: 0, 45, 90, 135, 180, 225, 270, and 315. This field is used to report the information captured from the CCC configuration process, and when the constraint is auto-generated.

### PLL Output

This field refers to the CCC GL0/1/2/3 output that is fed back to the PLL (in the CCC). This field is primarily used to report the information captured from the CCC configuration process, and when constraint is auto-generated.

### PLL Feedback

This field refers to the manner in which the GL/0/1/2/3 output signal of the CCC is connected to the PLL's FBCLK input. This field is primarily used to report the information captured from the CCC configuration process, and when constraint is auto-generated.

#### Add Clock to Existing Clock

Specifies that the generated clock constraint is a new clock constraint in addition to the existing one at the same source. The name of the clock constraint should be different from the existing clock constraint. When this option is selected, master clock must be specified.

#### Master Clock

Specifies the master clock used for the generated clock when multiple clocks fan into the master pin. It can be selected from the drop-down menu. This option is used in conjunction with the add option of the generated clock.

#### Comment

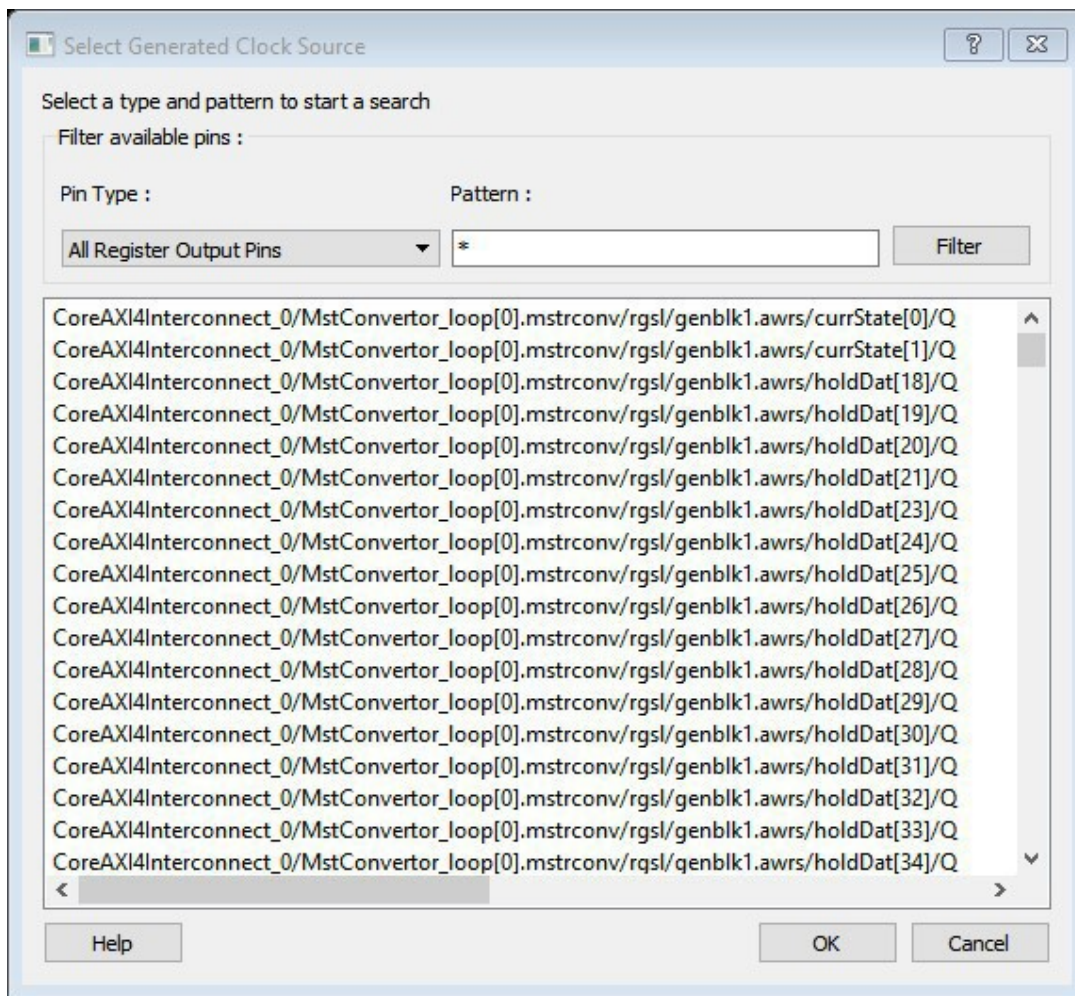
Enter a single line of text that describes the generated clock constraints purpose.

### 12.10.2 Select Generated Clock Source Dialog Box

Use this dialog box to find and choose the generated clock source from the list of available pins.

To open the Select Generated Clock Source dialog box (shown below) from the **Timing Constraints Editor**, open the [Create Generated Clock Constraint](#) dialog box and click the **Browse** button for the **Clock Pin**.

Figure 12-12. Select Generated Clock Source Dialog Box



#### Filter Available Pins

- **Pin type** – Displays the Available Pin types. The Pin Type options for Generated Clock Source are:

- 
- Output Ports
  - All Register Output Pins
  - All Pins
  - All Nets
  - Input Ports
  - **Pattern** – The default pattern is \*, which is a wild-card match for all. You can specify any string value.

Select **Filter** to filter the available pins based on the specified Pin Type and Pattern.

The list box displays the list of available pins based on the filter. Select the pins from the list and click **OK** to select the Generated Clock Source Pin.

## 12.11 Specifying Generated Clock Constraints

Specifying a generated clock constraint enables you to define an internally generated clock for your design and verify its timing behavior. Use generated clock constraints and [clock constraints](#) to meet your performance goals.

To specify a generated clock constraint:


1. Open the [Create Generated Clock Constraint](#) dialog box using one of the following methods:
  - Click the  icon.
  - Right-click the **Generated Clock** in the Constraint Browser and choose **Add Generated Clock**.
  - Double-click the Generated Clock Constraints grid. The Create Generated Clock Constraint dialog box appears (as shown below).

Figure 12-13. Create Generated Clock Constraint

Created Generated Clock Constraint

Reference Pin: [ ]

Clock Pin: [ ]

Clock Conditioning Circuitry

Clock Port

FPGA

Generated Clock Name : [ ]

☒ The generated frequency is such that :

$f(\text{dock}) = f(\text{reference}) * \frac{1}{1}$

☐ The generated dock edges are based on the reference edges : [ ] [ ] [ ] Edges (example : 1 3 5)

☐ The edges are shifted by the following delays : [ ] [ ] [ ] Edge Shift (example: 0.0 0.5 1.0)

The generated waveform is [ the same as ] the reference waveform.

☐ An External feedback is used to generate the dock.

PLL Output: [ ] [ ]

PLL Feedback: [ ] [ ]

☐ Phase shift is applied by PLL.

Phase shift : [ 0.00 ] degree

☐ Add this clock to existing one with same source

Master Clock: [ ]

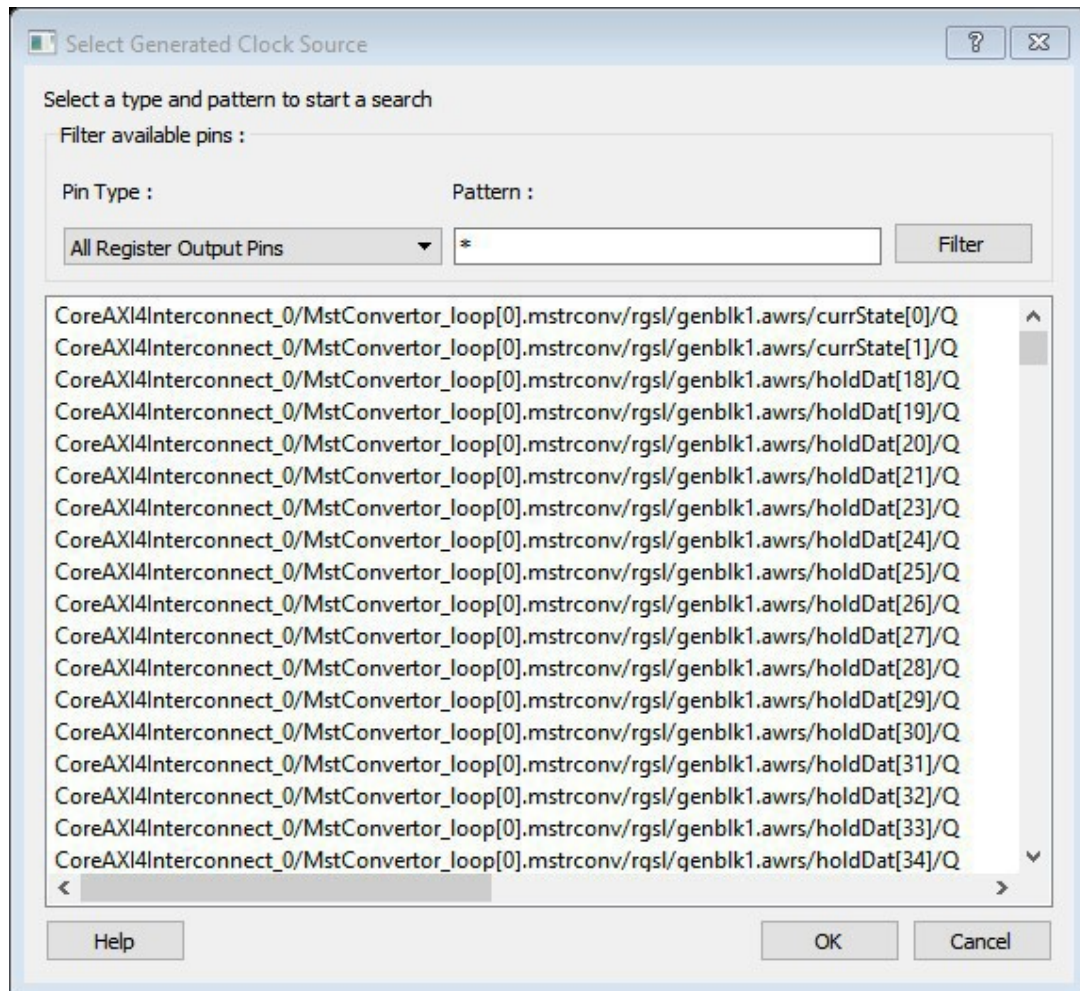
Comment : [ ]

Help OK Cancel

2. Select a **Clock Pin** to use as the generated clock source. To display a list of available generated clock source pins, click the **Browse** button. The [Select Generated Clock Source](#) dialog box appears (as shown below).



Figure 12-14. Select Generated Clock Source Dialog Box



3. Specify a **Reference Pin**. To display a list of available clock reference pins, click the **Browse** button. The Select Generated Clock Reference dialog box appears.
4. Specify the **Generated Clock Name**(optional).
5. Specify the values to calculate the generated frequency: a multiplication factor and/or a division factor (both positive integers).
6. Specify the orientation of the generated clock edges based on the reference edges by entering values for the edges and the edge shifts. This is optional.
7. Specify the first edge of the generated waveform either same as or inverted with respect to the reference waveform.
8. Specify the PLL output and PLL feedback pins, if an External feedback is used to generate the clock.
9. Specify the Phase shift applied by the PLL in degrees.
10. Specify the Master Clock, if you want to add this to an existing one with the same source.
11. Click **OK**. The new constraint appears in the Constraints List.

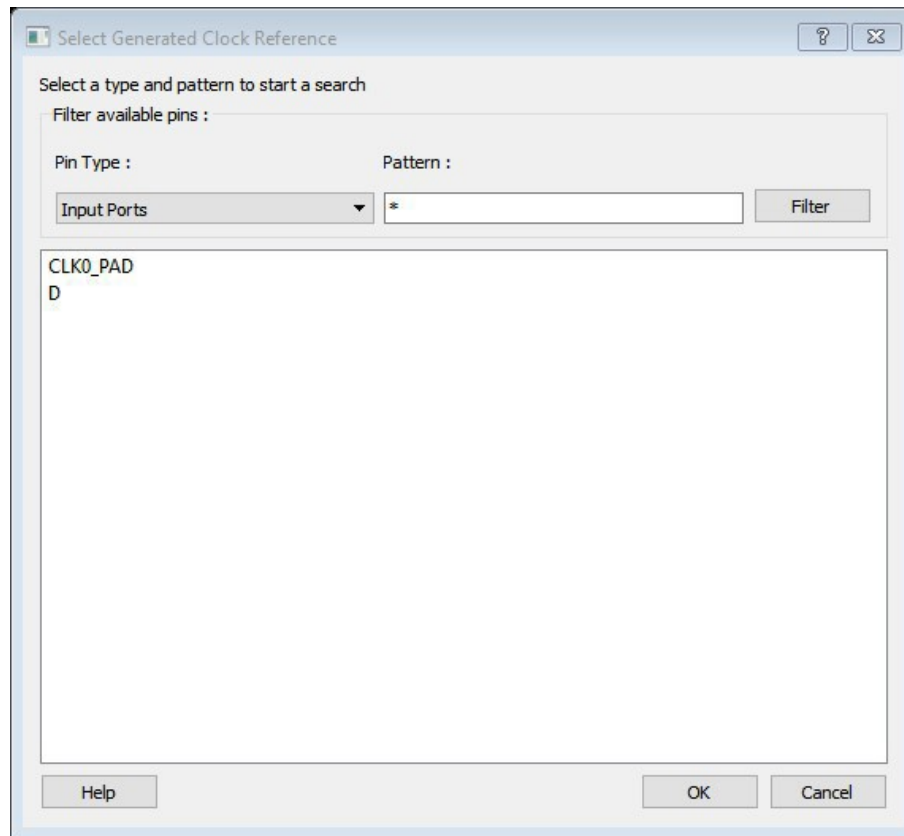
**Tip:** From the File menu, choose Save to save the newly created constraint in the database.

### 12.11.1 Select Generated Clock Reference Dialog Box

Use this dialog box to find and choose the generated clock reference pin from the list of available pins.

To open the Select Generated Clock Reference dialog box (shown below) from the SmartTime Constraints Editor, open the [Create Generated Clock Constraint Dialog Box](#) dialog box and click the **Browse** button for the **Clock Reference**.

Figure 12-15. Select Generated Clock Reference Dialog Box



#### Filter Available Pins

- **Pin type** – Displays the Available Pin types. The Pin Type options for Generated Clock Reference are:
  - Input Ports
  - All Pins
- **Pattern** – The default pattern is \*, which is a wild-card match for all. You can specify any string value. Select **Filter** to filter the available pins based on the specified Pin Type and Pattern.

The list box displays the list of available pins based on the filter. Select the pins from the list and click **OK** to select the Generated Clock Reference Pin.

#### See Also

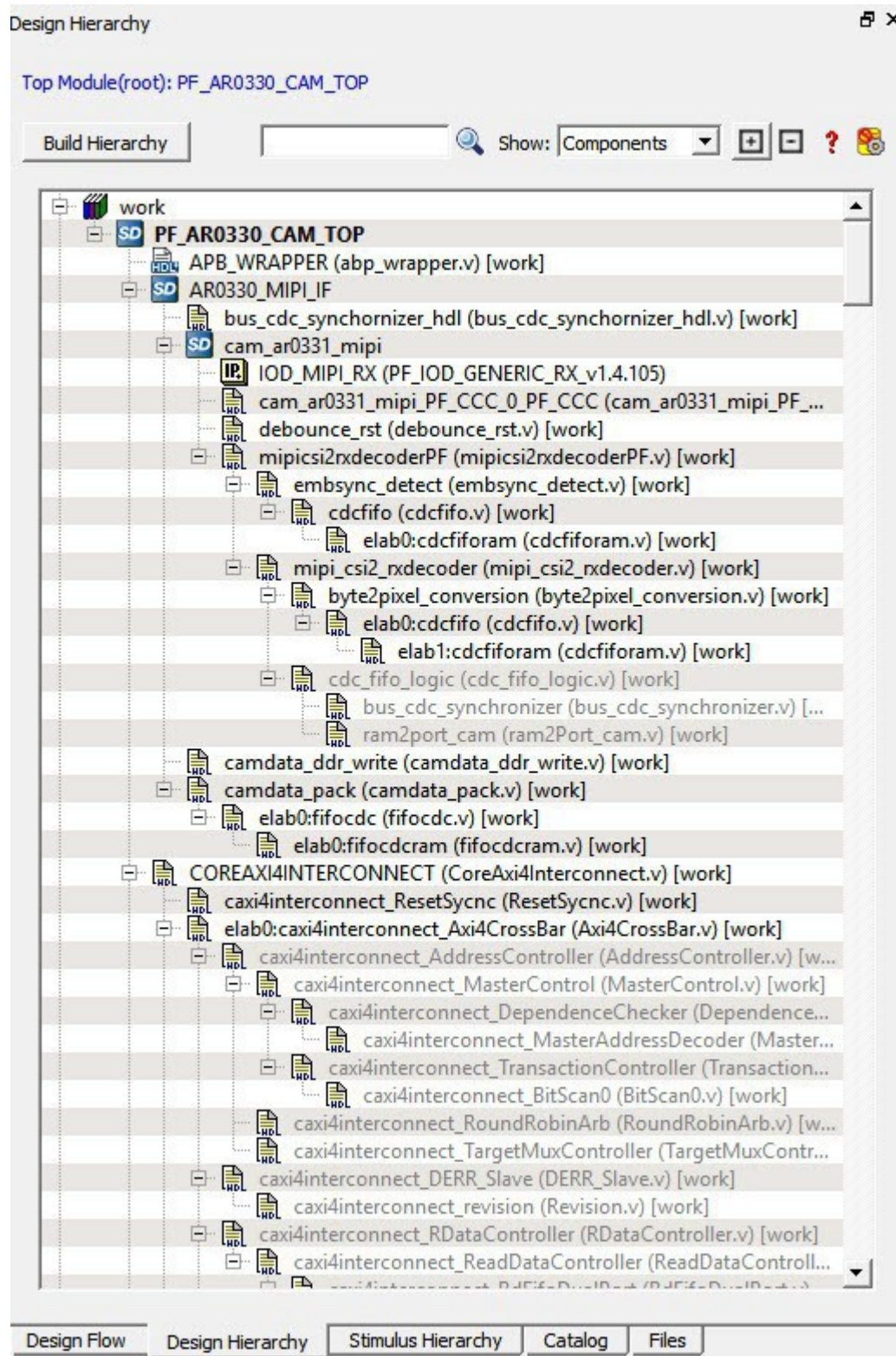
[Specifying generated clock constraints](#)

## 12.12 Design Hierarchy in the Design Explorer

The Design Hierarchy tab displays a hierarchical representation of the design based on the source files in the project. It also displays elaborated hierarchy constructed by propagating correct values for parameters and generics. The software continuously analyzes the source files and updates the content. The Design Hierarchy tab (see figure below) displays the structure of the modules and components as they relate to each other along with parameter/generic names and its values on the tool tip for which the module is instantiated. It also displays architecture name for a given entity and Configuration for VHDL modules.



Figure 12-16. Design Hierarchy



A module can have multiple elaborations depending on the different parameters/generics used in the instantiation of the module and all of these elaborated modules will be shown in the Design Hierarchy. The parameterized instantiated module will be shown as `elab<num>:<modulename>`.

Modules are instantiated with their actual names in the SmartDesign. If a module with elaborated name in the Design Hierarchy has to be instantiated in the SmartDesign, an instance of the original module is created in the SmartDesign. The following figure shows the design hierarchy with elaborated modules.

Figure 12-17. Design Hierarchy with Elaborated Modules (Verilog)

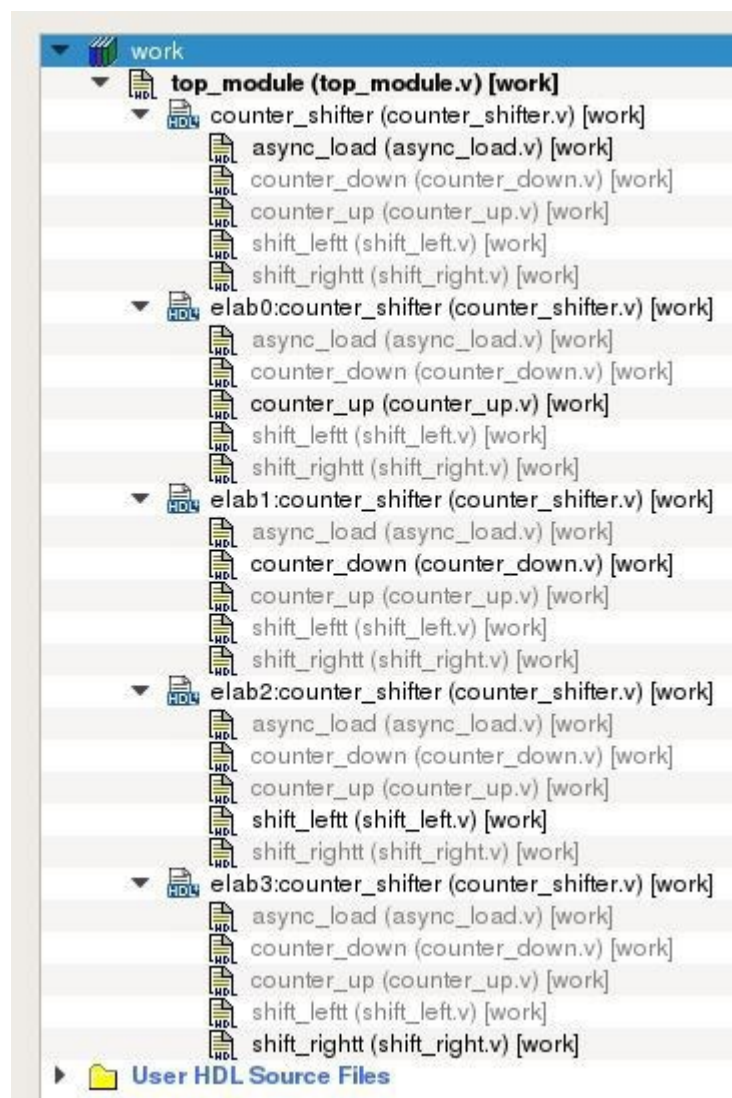
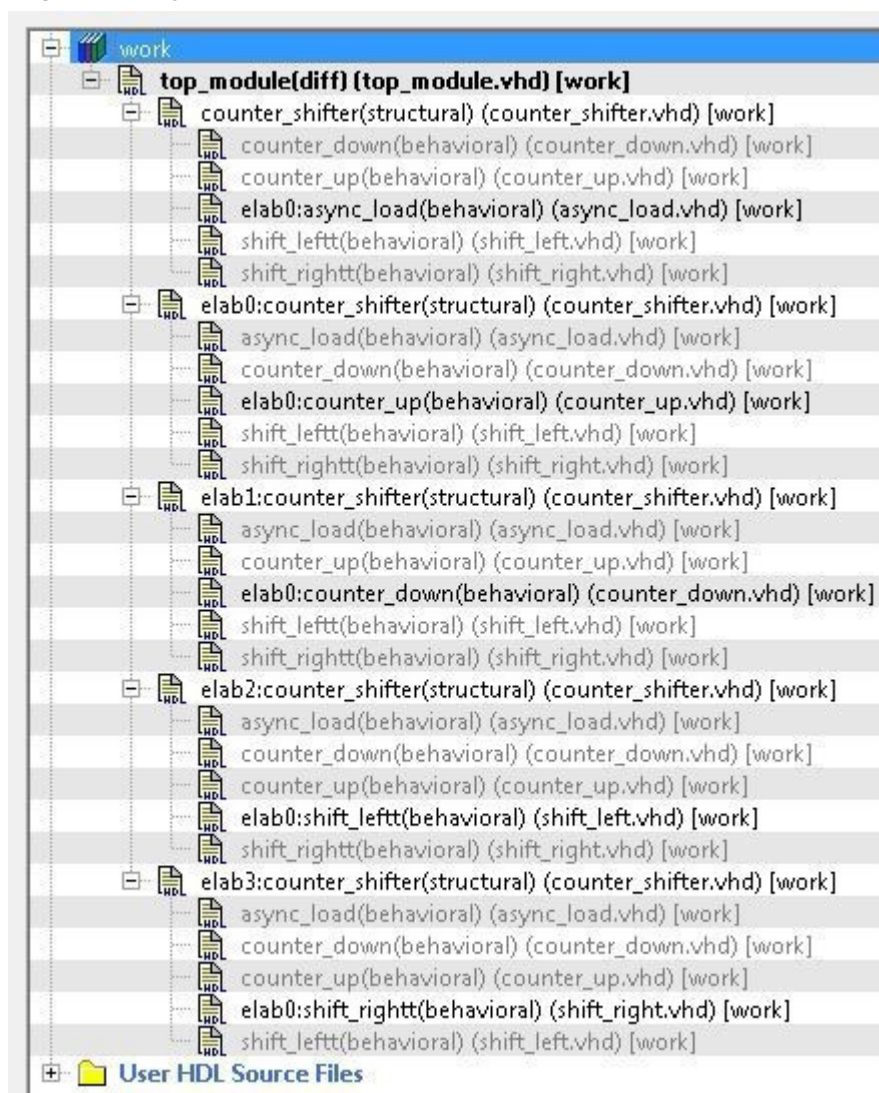


Figure 12-18. Design Hierarchy with Elaborated Modules (VHDL)



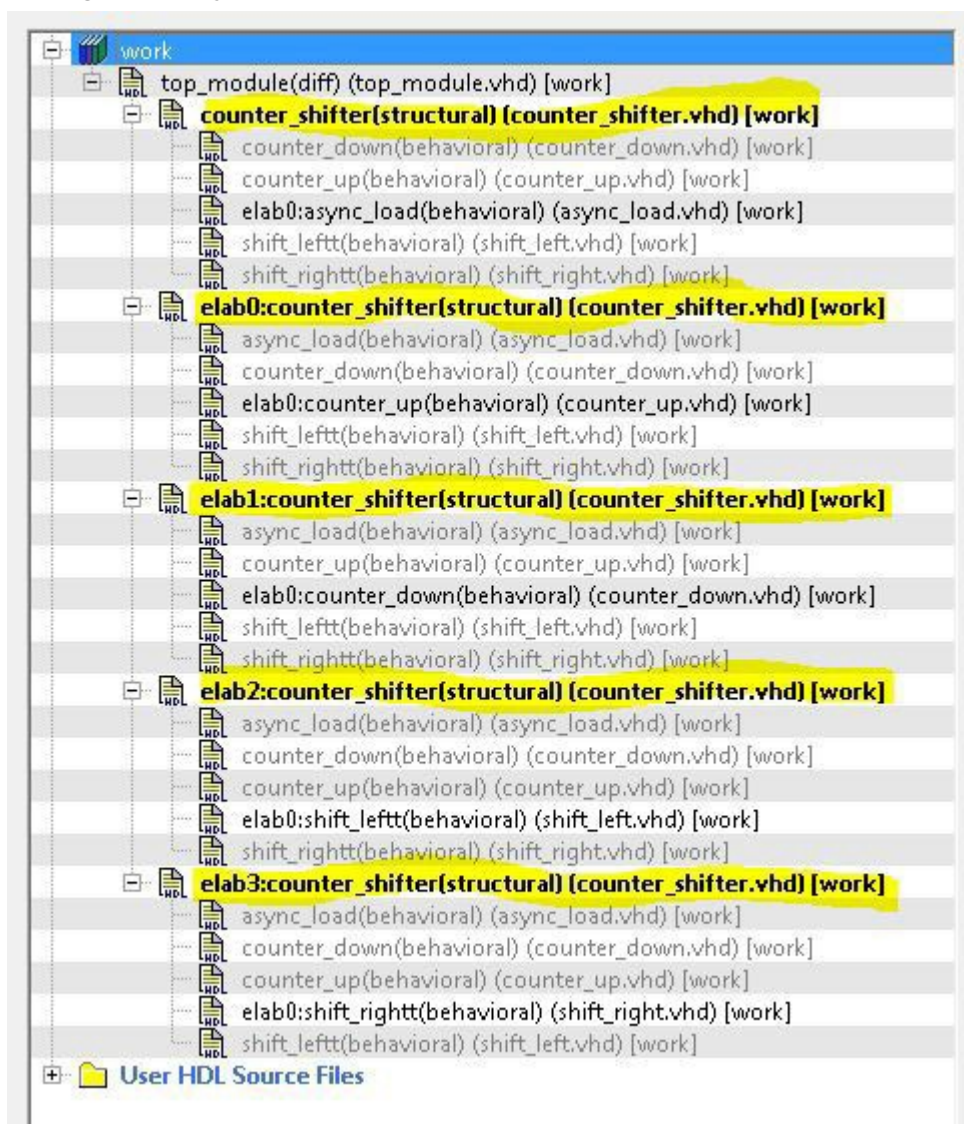
Modules which are not part of the elaboration will be shown in the complete hierarchy but they remain grayed out. When you create a core from a module, all the elaborated modules of that module will be shown as HDL+ core modules. You can get the parameter value of an elaborated module by selecting **Show Module parameters** on the right click menu of the elaborated module.

**Note:** A tool tip on each module shows all the parameters with their values for the instantiated module.

**Note:** Synthesis output will be the same for different elaborations of the same module i.e. **elab0:module1** and **elab1:module1** will have the same synthesis output. When one of the elaborated module is set as root, all the elaborations will be highlighted in the Design Hierarchy as shown in the below figure.




Figure 12-19. Design Hierarchy When one of the Elaborated Module is Set as Root



You can change the display mode of the Design Hierarchy by selecting **Components** or **Modules** from the **Show** drop-down list. The components view displays the entire design hierarchy; the modules view displays only schematic and HDL modules.

You can build the Design Hierarchy and Simulation Hierarchy by clicking the **Build Hierarchy** button.

A yellow icon  indicates that the Design Hierarchy is out of date (invalidated). Any change to the design sources/stimuli invalidates the Design Hierarchy/Stimulus Hierarchy. Click the **Build Hierarchy** button to rebuild the Design Hierarchy.

The file name (the file that defines the block) appears next to the block name in parentheses.

To view the location of a component, right-click and choose **Properties**. The Properties dialog box displays the path name, created date, and last modified date.

All integrated source editors are linked with the SoC software. If a source is modified and the modification changes the hierarchy of the design, the Build Hierarchy automatically updates to reflect the change.













If you want to update the Design Hierarchy, from the **View** menu, choose **Refresh Design Hierarchy**.

**To open a component:**

Double-click a component in the Design Hierarchy to open it. Depending on the block type and design state, several possible options are available from the right-click menu. You can instantiate a component from the Design Hierarchy to the SmartDesign Canvas. See the [SmartDesign User Guide](#) for more information.

Icons in the Hierarchy indicate the type of component and the state, as shown in the table below.

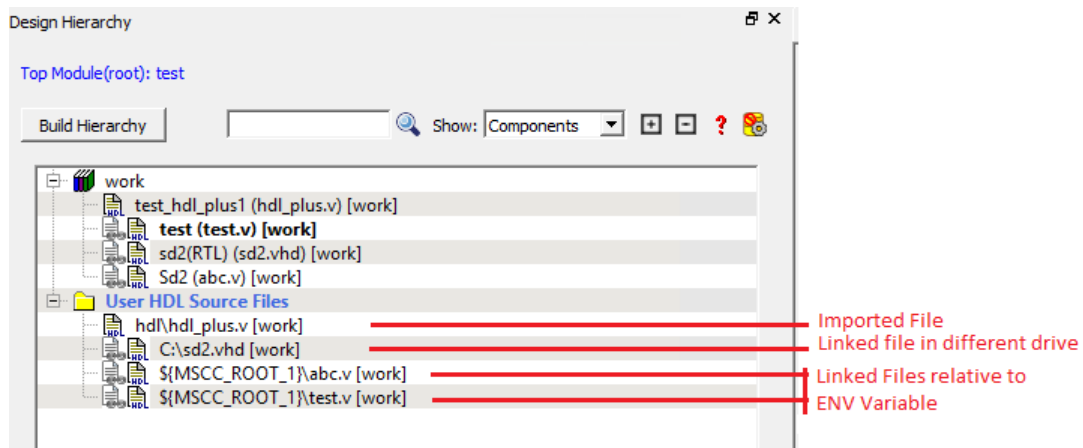
**Table 12-2. Design Hierarchy Icons**

Icon	Description
	SmartDesign component
	SmartDesign component with HDL netlist not generated
	IP core was instantiated into SmartDesign but the HDL netlist has not been generated
	Core
	Error during core validation
	Updated core available for download
	HDL netlist
	Shows ungenerated components
	Shows unknown modules
	Expands all the files and folders in the Design Hierarchy
	Collapses all the files and folders in the Design Hierarchy
	Finds the files in the Design Hierarchy

#### Linked Files with Relative Path in Design Hierarchy with Environment Variable

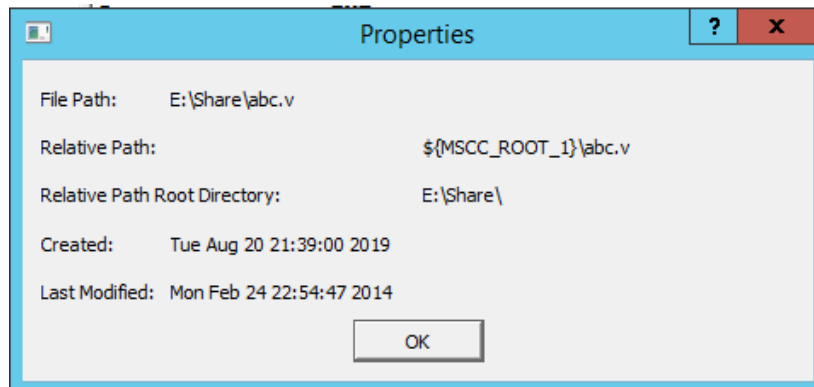
The following figure shows linked files that have a relative path in the design hierarchy with the Environment variable. Linked files that are in a drive different to the drive specified in Environment variable path are shown as absolute files in Windows.

**Figure 12-20. Linked Files with Relative Path in Design Hierarchy with Environment Variable**



Properties of file in Design Hierarchy shows Relative Path and Relative Path root directory

**Figure 12-21. File Path Properties in Design Hierarchy Representing Relative Path and Relative Path Root Directory**



## 12.13 Digest File

Users can verify which bitstream file was programmed onto their devices by running the VERIFY or VERIFY\_DIGEST actions on each device that was programmed. This is a costly and time-consuming process. To speed up the verification process, digests are printed during bitstream generation and bitstream programming.

These digests can be compared to verify that all of the devices were programmed with the correct bitstream file.

The bitstream file is divided into three major component sections: FPGA fabric, eNVM, and Security. A valid bitstream will contain a combination of any of the three primary bitstream components.

### Use Case

When a customer creates a design in Libero and then exports the STAPL file (for FlashPro) or programming job (for FlashPro Express), the digest of each of the primary components is printed in the Libero log window and saved in a digest file under the export folder. The digest file is a text file containing the bitstream component name with its corresponding digest. The name of the digest file will match the name of the STAPL/programming job exported, and will be appended with a ".digest" extension.

The customer then sends the STAPL/programming job to a production programming house. Now, when the devices are programmed, the digest of each of the primary components is printed in the log window. The production programming house saves the log files and sends the devices along with log files back to the customer. The customer can then verify that the correct design was programmed on the device by matching the digests in the log file with that in the \*.digest file under the Libero export folder.

### Note:

Digest printed during programming (same as in \*.digest file) is bitstream payload digest. It is meant for device to confirm that it receives the correct bitstream payload.


Digest exported from DEVICE\_INFO is the digest of the actual memory content. It does not have other metadata that is included in the encrypted bitstream payload, so it will be different than one generated during programming.

### See Also


[Export Bitstream](#)

## 12.14 Design Rules Check

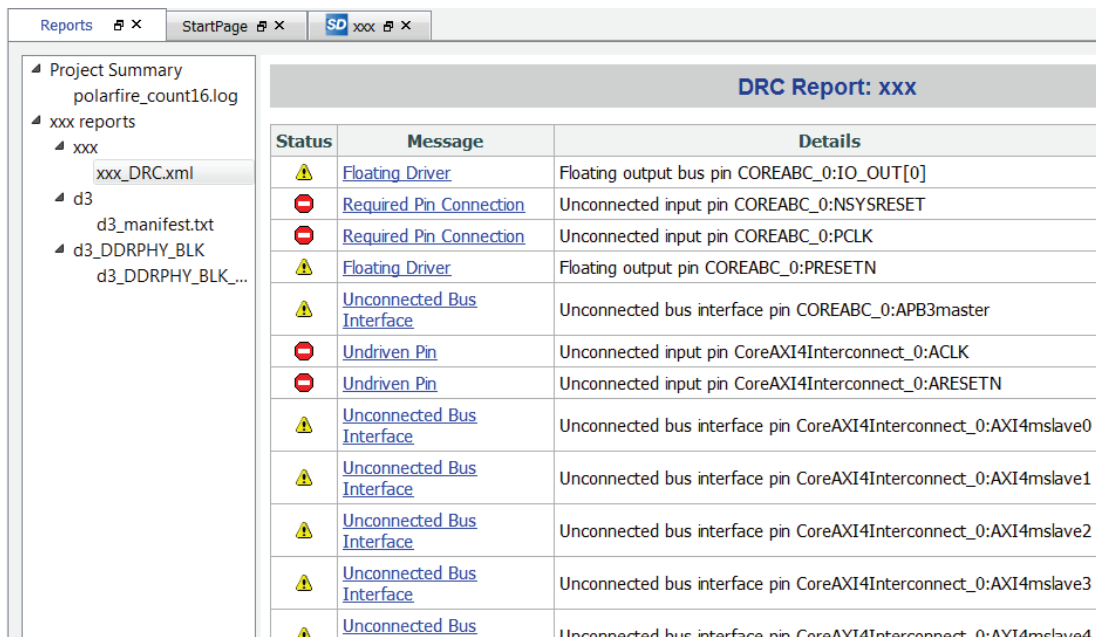
The Design Rules Check runs automatically when you generate your SmartDesign; the results appear in the Reports













tab. You can also initiate a Design Rules Check by clicking on the  button of the SmartDesign Canvas tab menu.

To view the results, from the **Design** menu, choose **Reports**.

- **Status** displays an icon to indicate if the message is an error or a warning (as shown in the figure below). Error messages are shown with a small red sign and warning messages with a yellow exclamation point. 
- **Message** identifies the specific error/warning (see list below); click any message to see where it appears on the Canvas.
- **Details** provides information related to the Message.

**Figure 12-22. Design Rules Check Results**



Status	Message	Details
	<a href="#">Floating Driver</a>	Floating output bus pin COREABC_0:IO_OUT[0]
	<a href="#">Required Pin Connection</a>	Unconnected input pin COREABC_0:NSYSRESET
	<a href="#">Required Pin Connection</a>	Unconnected input pin COREABC_0:PCLK
	<a href="#">Floating Driver</a>	Floating output pin COREABC_0:PRESETN
	<a href="#">Unconnected Bus Interface</a>	Unconnected bus interface pin COREABC_0:APB3master
	<a href="#">Undriven Pin</a>	Unconnected input pin CoreAXI4Interconnect_0:ACLK
	<a href="#">Undriven Pin</a>	Unconnected input pin CoreAXI4Interconnect_0:ARESETN
	<a href="#">Unconnected Bus Interface</a>	Unconnected bus interface pin CoreAXI4Interconnect_0:AXI4mslave0
	<a href="#">Unconnected Bus Interface</a>	Unconnected bus interface pin CoreAXI4Interconnect_0:AXI4mslave1
	<a href="#">Unconnected Bus Interface</a>	Unconnected bus interface pin CoreAXI4Interconnect_0:AXI4mslave2
	<a href="#">Unconnected Bus Interface</a>	Unconnected bus interface pin CoreAXI4Interconnect_0:AXI4mslave3
	<a href="#">Unconnected Bus Interface</a>	Unconnected bus interface pin CoreAXI4Interconnect_0:AXI4mslave4

### Message Types:

**Unused Instance** - You must remove this instance or connect at least one output pin to the rest of the design.

**Out-of-date Instance** - You must update the instance to reflect a change in the component referenced by this instance.

**Undriven Pin** - To correct the error you must connect the pin to a driver or change the state, i.e. tie low (GND) or tie high (VCC).

**Floating Driver** - You can mark the pin unused if it is not going to be used in the current design. Pins marked unused are ignored by the Design Rules Check.

**Unconnected Bus Interface** - You must connect this bus interface to a compatible port because it is required connection.

**Required Bus Interface Connection** – You must connect this bus interface before you can generate the design. These are typically silicon connection rules.

**Exceeded Allowable Instances for Core** – Some IP cores can only be instantiated a certain number of times for legal design because of silicon limitations. You must remove the extra instances.

**Incompatible Family Configuration** – The instance is not configured to work with this project's Family setting. Either it is not supported by this family or you need to re-instantiate the core.

**Incompatible Die Configuration** – The instance is not configured to work with this project's Die setting. Either it is not supported or you need to reconfigure the Die configuration.

**No RTL License, No Obfuscated License, No Evaluation License** – You do not have the proper license to generate this core. [Contact Microchip SoC](#) to obtain the necessary license.

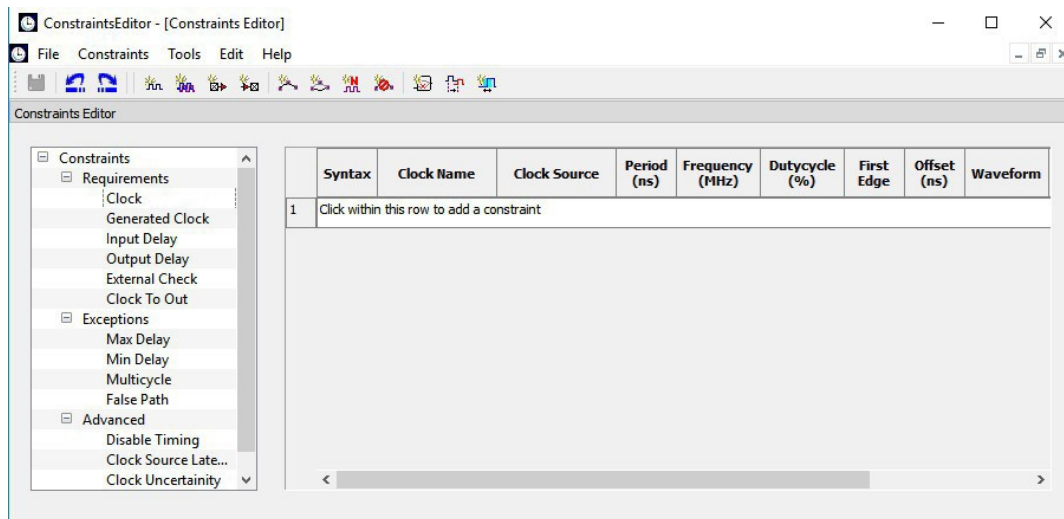
**No Top level Ports** - There are no ports on the top level. To auto-connect top-level ports, right-click the Canvas and choose Auto-connect.

**Self-Instantiation** - A component cannot instantiate itself-This is reported only in the Log/Message Window.

## 12.15 Editable Constraints Grid

The Constraints Editor enables you to add, edit, and delete.

**Figure 12-23. Constraints Editor**



To add a new constraint:

1. Select a constraint type from the constraint browser.
2. Enter the constraint values in the first row and click the green check mark to apply your changes. To cancel the changes, press the red cancel mark.
3. The new constraint is added to the Constraint List. The green syntax flag indicates that the constraint was successfully checked.

To edit a constraint:

1. Select a constraint type from the constraint browser.
2. Select the constraint, edit the values and click the green check mark to apply your changes. To cancel the changes, press the red cancel mark. The green syntax flag indicates that the constraint was successfully checked.



---

To delete a constraint:

1. Select a constraint type from the constraint browser.
2. Right-click the constraint you want to delete and choose **Delete Constraint**.

## 12.16 Files Tab and File Types

The Files tab displays all the files associated with your project, listed in the directories in which they appear.

Right-clicking a file in the Files tab provides a menu of available options specific to the file type. You can delete files from the project and the disk by selecting **Delete** from the right-click menu.

You can instantiate a component by dragging the component to a SmartDesign Canvas or by selecting **Instantiate in SmartDesign** from the right-click menu. See the [SmartDesign User Guide](#) for more details.

You can configure a component by double-clicking the component or by selecting **Open Component** from the right-click menu.

### File Types

When you create a new project in the Libero SoC it automatically creates new directories and project files. Your project directory contains all of your 'local' project files. **If you import files from outside your current project, the files must be copied into your local project folder (The Project Manager enables you to manage your files as you import them).**

Depending on your project preferences and the version of Libero SoC you installed, the software creates directories for your project.

The top level directory (<project\_name>) contains your PRJ file; only one PRJ file is enabled for each Libero SoC project.

**component** directory - Stores your SmartDesign components (SDB and CXF files) for your Libero SoC project.

**constraint** directory - All your constraint files (SDC, PDC).

**designer** directory - \*\_ba.sdf, \*\_ba.v(hd), **STP**, TCL (used to run designer), designer.log (logfile) **hdl** directory - all hdl sources. \*.vhd if VHDL, \*.v and \*.h if Verilog, \*.sv if SystemVerilog **simulation** directory - meminit.dat, modelsim.ini files.

**smartgen** directory - GEN files and LOG files from generated cores.

**stimulus** directory - BTIM and VHD stimulus files.

**synthesis** directory - \*.edn, \*\_syn.prj (Synplify log file), \*.srr (Synplify logfile), \*.tcl (used to run synthesis) and many other files generated by the tools (not managed by Libero SoC).

**tooldata** directory - includes the log file for your project with device details.

## 12.17 Importing Files

Anything that describes your design, or is needed to program the device, is a project source. These may include schematics, HDL files, simulation files, testbenches, etc. Import these source files.

To import a file:

1. From the **File** menu, choose **Import Files**.
2. In **Files of type**, choose the file type.
3. In **Look in**, navigate to the drive/folder where the file is located.
4. Select the file to import and click **Open**.  
**Note:** You cannot import a Verilog File into a VHDL project and vice versa.

Table 12-3. File Types for Import

File Type	File Extension
Behavioral and Structural VHDL; VHDL Package	*.vhd, *.vhdl
Design Block Core	*.gen
Verilog Include	*.h
Behavioral and Structural Verilog	*.v, *.sv
Netlist Verilog	*.vm
Stimulus	*.vhd, *.vhdl, *.v, *.sv
Memory file	*.mem
Components (Designer Blocks, Synplify DSP)	*.cxf
MSS Components	*.cxz

## 12.18 Layout Error Message: layoutg4NoValidPlacement

This is a generic error produced by the placer when it is unable to place a design. The most common cause for this failure is that the placer was unable to find a solution which could fit the design into the chip, either because the design is close to maximum utilization, or logic cannot be fit into user-defined region constraints.

If Libero is unable to find a legal placement, a list of unplaced cells will be provided in the log. The cells in this list may not be the cause of the placement problem; it is quite possible that some other constrained block of logic was placed first and now prohibits further placement. However, starting with the unplaced cell list is the easiest and most likely course:

- The simplest potential solution is to remove all placement constraints of the unplaced cells, and re-run Place and Route.

However, the cells in this list may not be the cause of the placement problem; it is quite possible that some other constrained block of logic which was placed first and now prohibits further placement. If removing the placement constraints on the unplaced cells does not succeed:

- Remove all region constraints and re-run Place & Route. Some designers make it a practice to put all their region constraints in a single, separate PDC file; in which case they need only disable that file.
  - If this Place & Route re-run still fails, there may be wider issues with the design's size and complexity that cannot be addressed by changes to P&R options.
  - If the unconstrained Place & Route re-run succeeds, then the user should add back constraints a few regions at a time in order of "simplicity". Usually, big regions with lots of free space are "simpler" for the placer, whereas tall/narrow regions with high utilization are "harder". Re-run Place & Route with each constraint restoration and repeat the process until the failing region(s) is identified. Depending on requirements, the failing region may be handled by removing or changing its constraints, or revising its design to use less resources.

The user may also re-run the Placer in high-effort mode. Applying high-effort mode to a design which is very full can incur additional runtime and may produce a placement solution which may not meet tight timing constraints, owing to the fact that the placer will aggressively attempt to fit the design. In practice, customers are **encouraged** to apply the previous suggestions first; and utilize high-effort mode only when other approaches have been exhausted.

## 12.19 Layout Error Message: layoutg4DesignHard

This design is very difficult to place, and high-effort techniques were required to make it fit. This may lead to increased layout runtime and diminished timing performance.

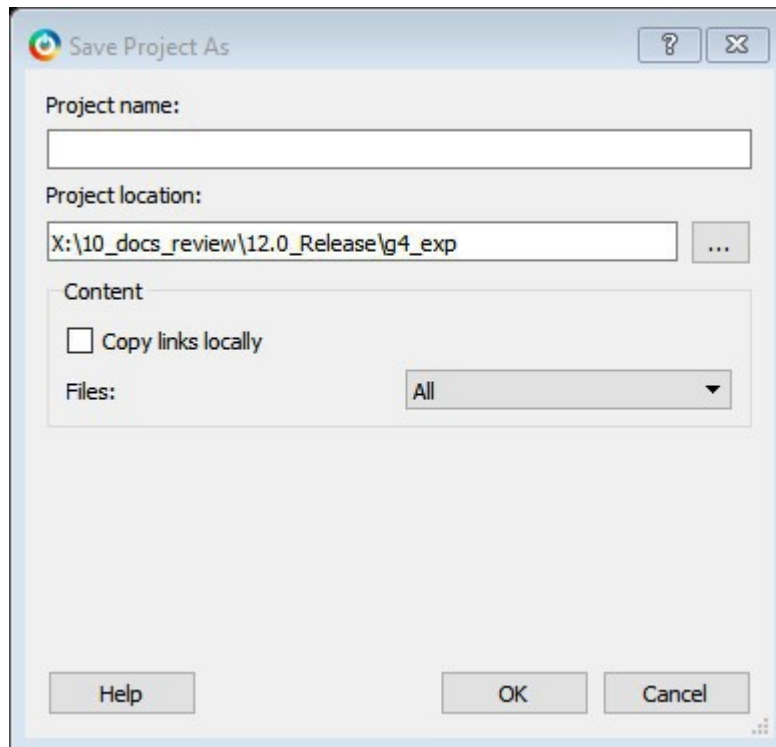
This message typically appears in designs with high utilization — a very full design, or a design with region constraints which are, themselves, very full. It can also occur in designs with moderate utilization but with numerous, long carry chains.

No immediate action is required on the user's part. However, if this notice is observed during Layout, the resultant performance of the design and the runtime of the Layout tools may not be optimal, and there is a strong possibility that reducing the size of the design, or relaxing region and floorplanning constraints, will help to improve timing closure and runtime.

## 12.20 Save Project As Dialog Box

The Save Project As dialog box enables you to save your entire project with a new name and location. To access this dialog box, choose **Save Project As** from the **Project** menu.

Figure 12-24. Save Project As Dialog Box



### Project Name

Type the project name for your modified project.

### Project Location

Accept the default location or **Browse** to the new location where you can save and store your project. All files for your project are saved in this directory.

### Content

**Copy links locally** - Select this checkbox to copy the links from your current project into your new project. If you do not select this checkbox, the links will not be copied and you must add them manually.

### Files

- **All** - Includes all your project and source files; the state of the project is retained.
- **Project files only** - Copies only the project-related information required to retain the state of the project.
- **Source files only** - Copies all the source files into the specified location. This means the configuration of all the tools in the tool chain is retained but the states are not. Source files means constraint information and component information available in the component, hdl and smartgen directories.

Files are saved as shown in the table below.

Folder Name	Files		
	All	Project	Source
component	All Files	All Files	All Files
constraint	All Files	All Files	All Files
hdl	All Files	All Files	All Files
stimulus	All Files	All Files	All Files
smartgen	All Files	All Files	All Files
firmware	All Files	All Files	All Files
CoreConsole	All Files	All Files	All Files
SoftConsole/Keil/IAR	All Files	All Files	All Files
Phy_Synthesis	All Files	All Files	Not Copied
simulation	All Files	*.ini, *.bfm, *.do., *.vec	*.ini, *.bfm, *.do., *.vec
synthesis	All Files	*.edn, *.vm, *.sdc, *.so, *.prj, *.srr, *.v, run_options.txt, synplify.log	*.prj files
Designer/impl1	All Files	All Files	*.ide_des files
Designer/<root>	All Files	All Files	Not Copied
tooldata	All Files	All Files	All Files

**Note:** \*.edn files are not supported in PolarFire.

## 12.21 Project Settings Dialog Box

The Project Settings dialog box enables you to modify your Device, HDL, and Design Flow settings and your Simulation Options. In Libero SoC, from the Project menu, click **Project Settings**.

The following figure shows an example of the Project Settings dialog box.

**Figure 12-25. Project Settings Dialog Box**



### Device Selection

Sets the device Die and Package for your project. See the [New Project Creation Wizard - Device Selection](#) page for a detailed description of the options.

### Device Settings

**Default I/O Technology** - Sets all your I/Os to a default value. You can change the values for individual I/Os in the I/O Attributes Editor.

**System controller suspended mode.** If the System controller suspended mode is enabled in any PolarFire design, the following operations will not be available during normal operation. For more information, refer to the System Services section in the PolarFire FPGA Security User Guide (UG0753).

- All System controller services requested after power-up is complete and the System controller is suspended
- System controller generated Tamper flags
- Device reset and zeroization Tamper responses
- SPI-Master In-Application Programming (IAP)
- SPI-Slave programming mode

For RT PolarFire devices, the System controller suspended mode is enabled by default for all new RT PolarFire projects. If the System controller suspended mode is disabled, it increases vulnerability to radiation single event effects in the System controller.

### Design Flow

See the [Project Settings: Design flow](#) topic for more information.

### Analysis Operating Conditions

Sets the Operating Temperature Range, the Core Voltage Range, and Default I/O Voltage Range from the pick list's provided. Typical values are COM/IND/MIL; but others are sometimes defined.

Only EXT and IND ranges are available for PolarFire at present.

Once the "Range" value is set, the Minimum/Typical/Maximum values for the selected range are displayed.

These settings are propagated to Verify Timing, Verify Power, and Backannotated Netlist for you to perform Timing/Power Analysis.

### Simulation Options and Simulation Libraries

Sets your simulation options. See the [Project Settings: Simulation Options](#) topic for more information.

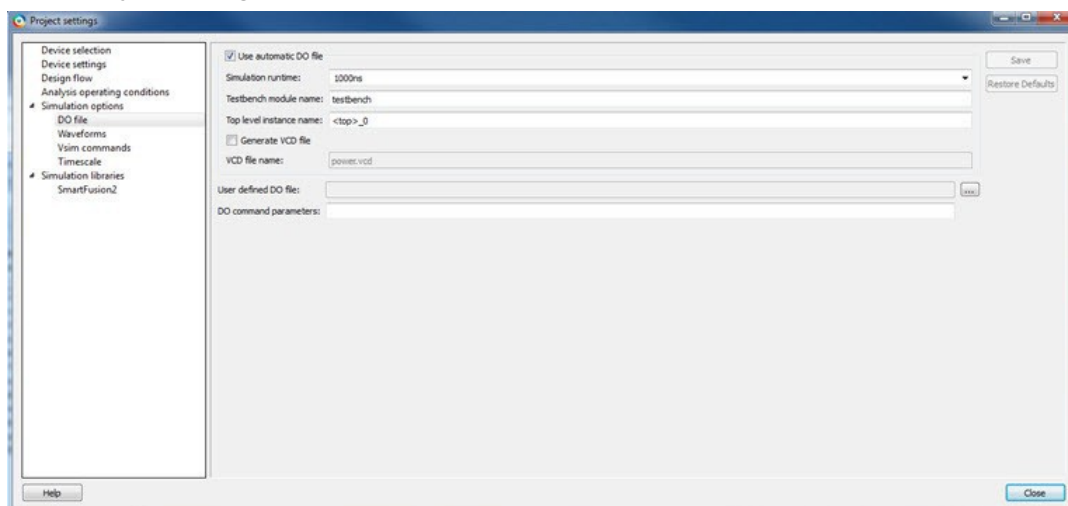
#### 12.21.1 Project Settings: Simulation - Options and Libraries

Using this dialog box, you can set change how Libero SoC handles Do files in simulation, import your own Do files, set simulation run time, and change the DUT name used in your simulation. You can also change your library mapping.

To access this dialog box, from the **Project** menu choose **Project Settings** and click to expand **Simulation options** or **Simulation libraries**.

For **Simulation options** click the option you wish to edit: **DO file**, **Waveforms**, **Vsim commands**, **Timescale**. For **Simulation libraries** click on the library you wish to change the path for.

Figure 12-26. Project Settings: DO File

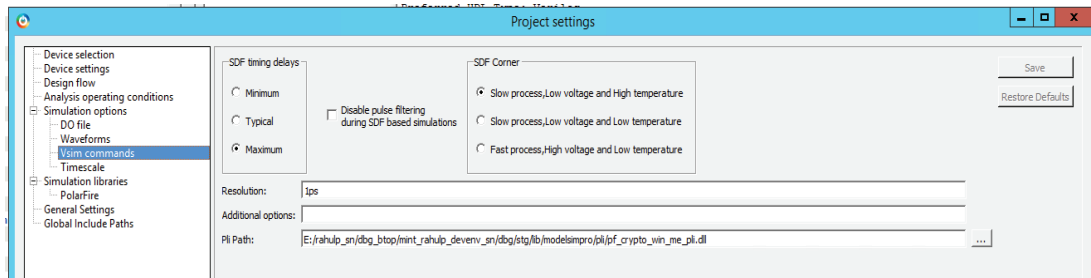


DO file:

- **Use automatic DO file** - Select if you want the Project Manager to automatically create a DO file that will enable you to simulate your design.
- **Simulation Run Time** - Specify how long the simulation should run. If the value is 0, or if the field is empty, there will not be a run command included in the run.do file.
- **Testbench module name** - Specify the name of your testbench entity name. Default is "testbench," the value used by WaveFormer Pro.
- **Top Level instance name** - Default is <top\_0>, the value used by WaveFormer Pro. The Project Manager replaces <top> with the actual top level macro when you runsimulation(presynth/postsynth/postlayout).
- **Generate VCD file** - Click the checkbox to generate a VCD file.
- **VCD file name** - Specifies the name of your generated VCD file. The default is power.vcd; click power.vcd and type to change the name.
- **User defined DO file** - Enter the DO file name or click the browse button to navigate to it.
- **DO command parameters** - Text in this field is added to the DO command.
- **Waveforms**
  - **Include DO file** - Including a DO file enables you to customize the set of signal waveforms that will be displayed in ModelSim.
  - **Display waveforms for** - You can display signal waveforms for either the top-level testbench or for the design under test. If you select **top-level testbench** then Project Manager outputs the line 'addwave /testbench/\*' in the DO file run.do. If you select **DUT** then Project Manager outputs the line 'add wave /testbench/DUT/\*' in the run.do file.
    - **Log all signals in the design** - Saves and logs all signals during simulation.

#### Vsim Commands

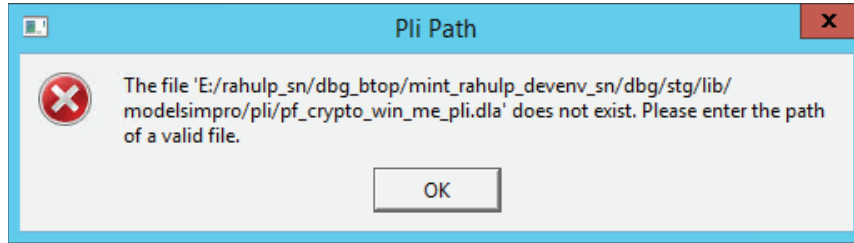
Figure 12-27. Vsim Commands



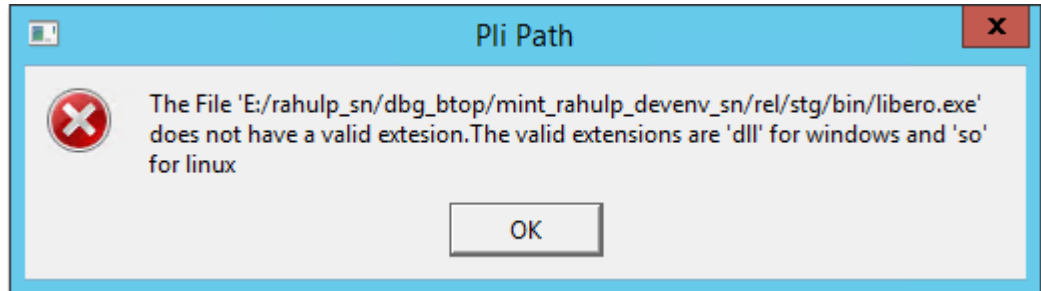
- **Resolution - The default is 1ps.** Some custom simulation resolutions may not work with your simulation library. Consult your simulation help for more information on how to work with your simulation library and detect infinite zero-delay loops caused by high resolution values.
- **Additional options** - Text entered in this field is added to the vsim command.
  - SRAM ECC Simulation -
    - Two options can be added to specify the simulated error and correction probabilities of all ECC SRAMs in the design.
    - -gERROR\_PROBABILITY=<value>, where 0 <= value <= 1
    - -gCORRECTION\_PROBABILITY=<value>, where 0 <= value <= 1
  - During Simulation, the SB\_CORRECT and DB\_DETECT flags on each SRAM block will be raised based on generated random numbers being below the specified<value>s.
  - When a user runs the Post-Layout Simulation Tool, a run.do file is created, which consists of information which needs to be sent to a simulator. To run a simulation on a corner, user should select one of the SDF corners along with the type of delay needed from one of the options in SDF timing delays section.
    - SDF Corners:
      - Slow Process, Low Voltage and High Temperature
      - Slow Process, Low Voltage and Low Temperature
      - Fast Process, High Voltage and Low Temperature
    - SDF Timing Delays:
      - Minimum
      - Typical
      - Maximum
    - Disable pulse filtering during SDF based simulations: This option disables the pulse filtering during SDF simulations.
    - After the user selects the corner, appropriate files needed for simulation are written in the run.do file as shown below.

```
vlog -sv -work postlayout "$ (PROJECT_DIR)/designer/sdl/sdl_fast_hv_lt_ba.v" vsim
-L PolarFire -L postlayout -t 1ps -sdfmax
/sdl=$(PROJECT_DIR)/designer/sdl/sdl_fast_hv_lt_ba.sdf +pulse_int_e/1
+pulse_int_r/1 +transport _int_delays postlayout.sdl
```

  - To run a simulation on a design having Crypto core, user needs to enter the PLI path in the PLI Path field. User can use the browse button to select the appropriate file. After the file is selected the PLI Path field is populated with the path and file. The user can also manually enter the path in the PLI path field.
    - Errors will be generated in case of wrong path or wrong file extension as follows:
      1. If the user enters a wrong PLI path and clicks on Save



2. If the user enters a wrong file extension.



- To compile all the files under System Verilog Multi-File Compilation Unit (MFCU) mode, group them based on the library. This applies to all the .hdl files, simulation files and core files. The code example below provides a general syntax to compile the files in System Verilog MFCU mode.

```
vlog "+incdir+<incdir1>" "+incdir+<incdir2>" -sv -mfcu -work library1 "file1"
"file2" "file3"
vlog "+incdir+<incdir1>" "+incdir+<incdir2>" -sv -mfcu -work library2 "file4"
"file5" "file6"
```

#### Timescale

- **TimeUnit** - Enter a value and select s, ms, us, ns, ps, or fs from the pull-down list, which is the time base for each unit. The default setting is ns.
- **Precision** - Enter a value and select s, ms, us, ns, ps, or fs from the pull-down list. The default setting is ps.

#### Simulation Libraries

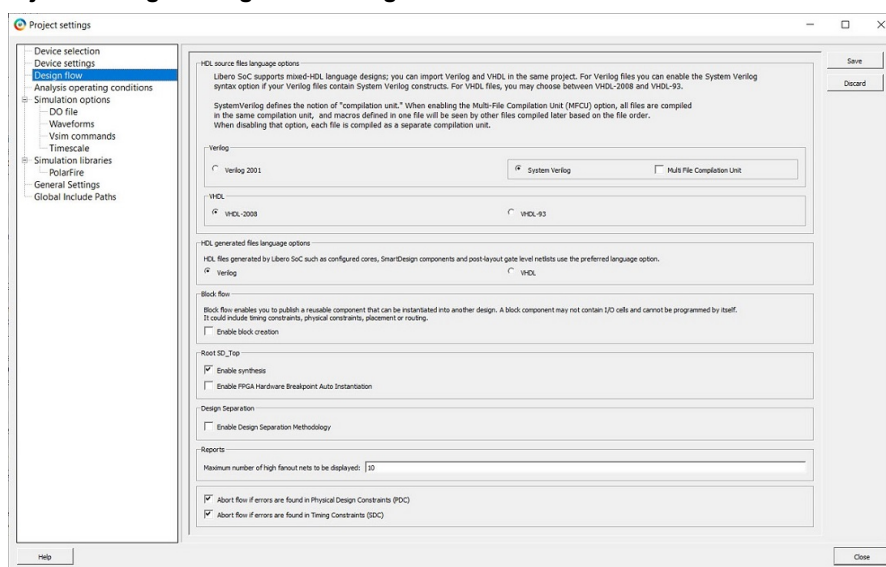
- **Restore Defaults**- Sets the library path to default from your Libero SoC installation.
- **Library path** - Enables you to change the mapping for your simulation library (both Verilog and VHDL). Type the pathname or click the Browse button to navigate to your library directory.

### 12.21.2 Project Settings: Design flow

To access the Design flow page, from the Project menu choose **Project settings**. From the **Project settings** window that appears, select the **Design flow** from the left pane.



**Figure 12-28. Project Settings Dialog Box – Design Flow Tab**



### HDL source files language options

Libero SoC supports mixed-HDL language designs. You can import Verilog and VHDL in the same project.

You can set HDL to VHDL or Verilog. For VHDL, choose VHDL-2008 or VHDL-93. For Verilog, choose System Verilog (if your Verilog files contain System Verilog constructs) or Verilog 2001.

SystemVerilog defines the notion of "compilation unit." Enabling the Multi-File Compilation Unit (MFCU) option, compiles all the files in the same compilation unit, and macros defined in one file will be seen by other files compiled later based on the file order. Disable the option to compile each file as a separate compilation unit.

When a New Project is created, **Multi-File Compilation Unit** option is enabled by default. This option will be disabled/greyed out if **Verilog-2001** option is selected.

**Notes:** Libero SoC supports the following Verilog and VHDL IEEE standards for ModelSim and SynplifyPro:

- Verilog 2005 (IEEE Standard 1364-2005)
- Verilog 2001 (IEEE Standard 1364-2001)
- Verilog 1995 (IEEE Standard 1364-1995)
- SystemVerilog 2012 (IEEE Standard 1800-2012)
- VHDL-2008 (IEEE Standard 1076-2008)
- VHDL-93 (IEEE Standard 1076-1993)

### HDL generated files language options

HDL files generated by Libero SoC can be set to use VHDL or Verilog. If there are no other considerations, it is generally recommended to use the same HDL language as you are using for HDL source files, as this may reduce the cost of simulation licenses.

### Block flow

Enable block creation - Enables you to create and publish design blocks (\*.cxz files) in Libero SoC. Design blocks are low-level components that may have completed the place-and-route step and met the timing and power requirements. These low-level design blocks can then be imported into a Libero SoC project and re-used as components in a higher level design. See [Designing with Designer Block Components](#) in Online Help for more information.

### Root <module\_name>

Enable synthesis - Option to enable or disable synthesis for your root file; useful if you wish to skip synthesis on your root file by default.

Enable FPGA Hardware Breakpoint Auto Instantiation - The FPGA Hardware Breakpoint (FHB) Auto Instantiation feature automatically instantiates an FHB instance per clock domain that is using gated clocks (GL0/GL1/GL2/GL3)

from an FCCC instance. The FHB instances gate the clock domain they are instantiated on. These instances can be used to force halt the design or halt the design through a live probe signal. Once a selected clock domain or all clock domains are halted, you can play or step on the clock domains, either selectively or all at once. FPGA Hardware Breakpoint controls in the SmartDebug UI provide control of the debugging cycle.

**Note:** The default format for Synthesis gate level netlist is Verilog for PolarFire devices.

### Reports

Maximum number of high fanout nets to be displayed - Enter the number of high fanout nets to be displayed. The default value is 10. This means the top 10 nets with the highest fanout will appear in the `<root>_compile_netlist_resource.xml` report.

### Abort Flow Conditions

Abort Flow if Errors are found in Physical Design Constraints (PDC) – Check this checkbox to abort Place and Route if the I/O or Floorplanning PDC constraint file contains errors.

Abort Flow if Errors are found in Timing Constraints (SDC) – Check this checkbox to abort Place and Route if the Timing Constraint SDC file contains errors.

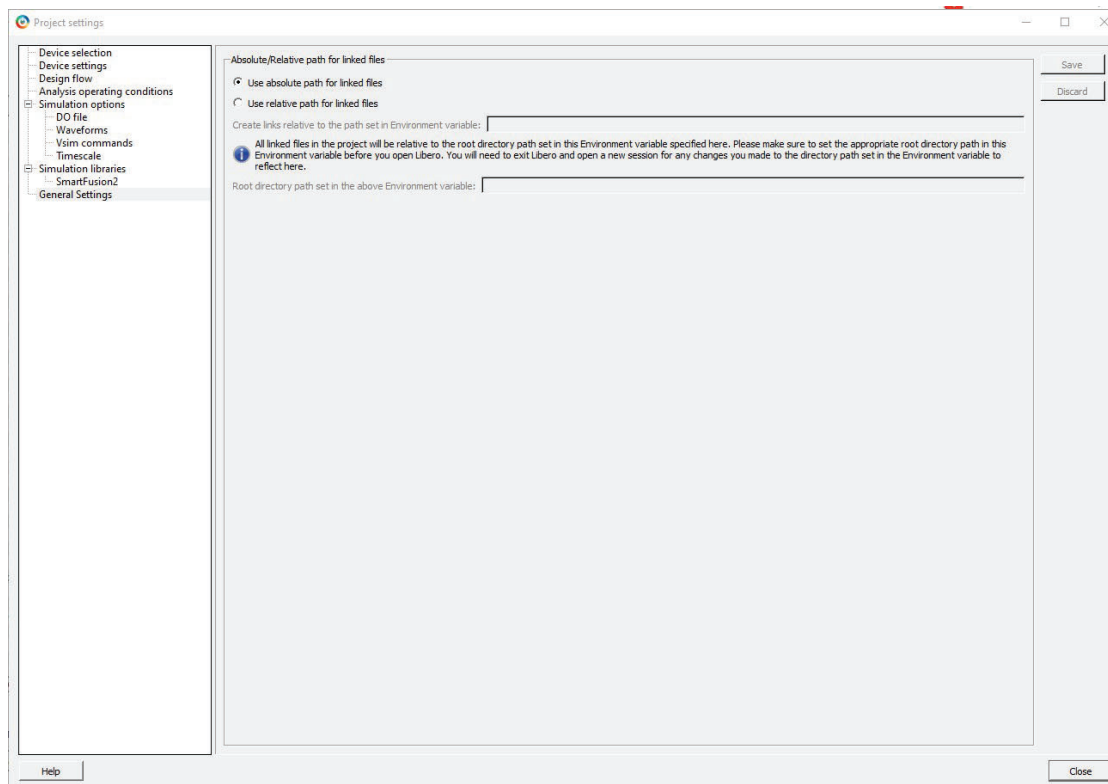
Abort flow if 3.3V I/Os are found in the design – Check this checkbox to abort Place and Route if 3.3 V I/Os are found in the design. This allows the tool to generate an error as shown below. However, if the checkbox is unchecked, the Place and Route tool will generate the same error message as a warning without interrupting the design flow process.

Error: I/Os at 3.3V are vulnerable to Single Event Latch-up at low LET levels. Please review RT PolarFire radiation data at Microsemi.com

## 12.21.3 Project Settings: General Settings

To access the General Settings page, from the Project menu choose **Project Settings** and click the General Settings tab.

**Figure 12-29. Project Settings Dialog Box – General Settings Tab**



### Absolute/Relative path for linked files

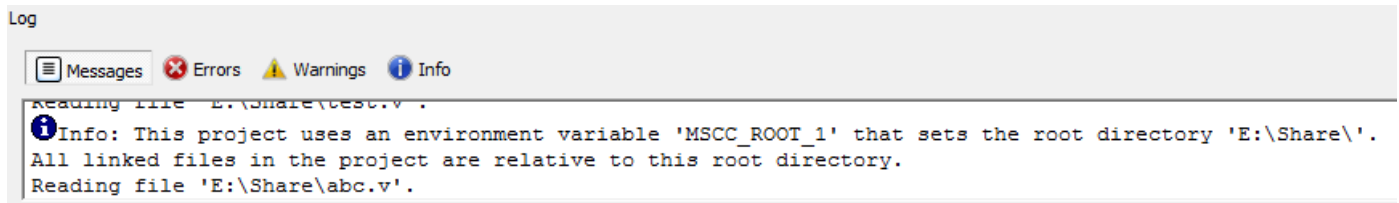
Users can choose to use either "absolute path for linked files" or "relative path for the linked files" for the files linked in the project. The path set in the environment variable is read only.

If the user selects "Use absolute path for linked files" option, all the linked files are stored with the absolute path.

If the user selects "Use relative path for linked files" option, the user is given an option to provide an environment variable that contains the root directory path that will be used as base location for the relative paths. All the files linked will be relative to the base location specified in environment variable by the user.

*After the user sets the relative path and provides an environment variable that has a valid path, the following message appears in the log window.*

**Figure 12-30. Message Displayed in Log Window After Environment Variable is Set**



This message can appear in the log window when the user:

- Opens a project that was created using relative path for linked files.
- Creates a project with relative path for linked files.
- Changes the environment variable from the project settings.

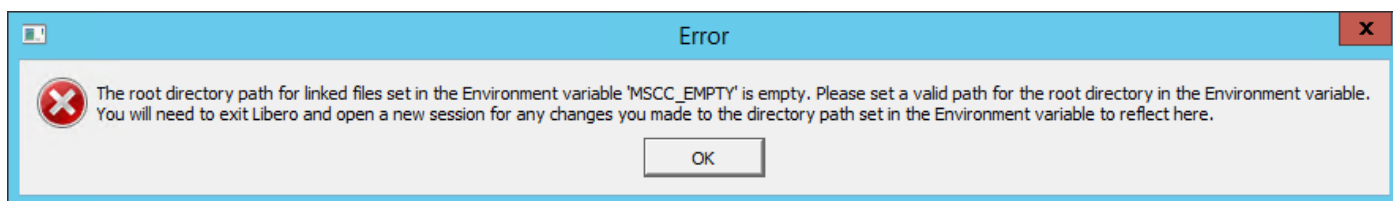
The environment variable set by the user is validated before proceeding. If the environment variable set by the user is not present or if it is set after invoking Libero, the tool issues the following error.



Error message: The Environment variable <env\_variable\_name> specified for the root directory path for linked files is not defined/set in your environment. Please make sure to set the appropriate root directory path in the Environment variable before you open Libero. You will need to exit Libero and open a new session for any changes you made to the directory path set in the Environment variable to reflect here.

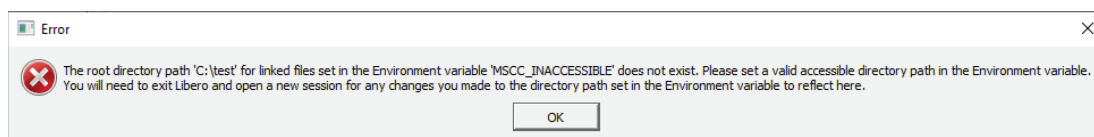
Other error messages that can occur are:

- Empty path in Environment variable



Error message: The root directory path for linked files set in the Environment variable <env\_variable\_name> is empty. Please set a valid path for the root directory in the Environment variable. You will need to exit Libero and open a new session for any changes you made to the directory path set in the Environment variable to reflect here.

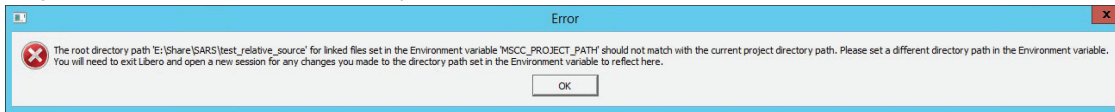
- Inaccessible path in environment variable



Error message: The root directory path <env\_variable\_directory> for linked files set in the Environment variable <env\_variable\_name> does not exist. Please set a valid accessible directory path in the Environment variable.

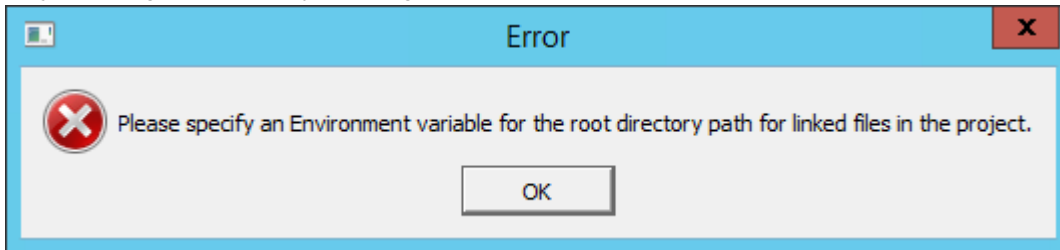
You will need to exit Libero and open a new session for any changes you made to the directory path set in the Environment variable to reflect here.

- Project location is set as root directory in environment variable



Error message: The root directory path <env\_variable\_path> for the linked files set in the Environment variable <env\_variable\_name> should not match with the current project directory path. Please set a different directory path in the Environment variable. You will need to exit Libero and open a new session for any changes you made to the directory path set in the Environment variable to reflect here.

- Project settings are saved by selecting relative path and the environment variable is not specified



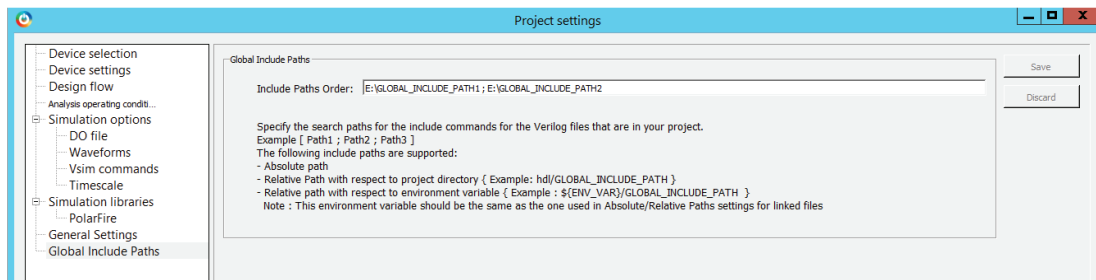
Error message: Please specify an Environment variable for the root directory path for linked files in the project.

**Note:** Users should set the Environment Variable before invoking Libero. Otherwise, Libero will not be able to get the path set in Environment Variable.

### 12.21.4 Project Settings: Global Include Paths

This setting facilitates the user with an option to provide Global Include paths in Libero and manage how the files are sent to downstream tools. It is the user's responsibility to provide the paths in order (Include Paths Order).

**Figure 12-31. Project settings with Global Include Paths Option**



#### Example:

If a user has a defines file which exists in all the three paths – source file path(default), global\_include\_path\_1 and global\_include\_path\_2; the order of priority of files will be as follows:

- Source file path(default)
- Global\_include\_path\_1
- Global\_include\_path\_2

#### Absolute and Relative paths in Global Include Paths

Users are given an option to specify both relative and absolute files in global include paths. Example -

Absolute Path : { E:/User/GLOBAL\_INCLUDE\_PATH\_1 } Relative Path: { ../hdl/GLOBAL\_INCLUDE\_PATH\_1 }

Relative Path w.r.t Environment Variable: { \$(ENV\_VAR)/GLOBAL\_INCLUDE\_PATH\_1 }

Here, the environment variable should be the one used in absolute and relative paths options only.

1. Changing the Global Include Paths will invalidate the Design Hierarchy but not the Design flow.
2. Change in content of Include Files present in Global Include Paths has impact as in cases below:
  - Case 1: If the Global Include Path is part of the project – Design flow will be invalidated.
  - Case 2: If the Global Include Path is outside the project – Design flow will not be invalidated.

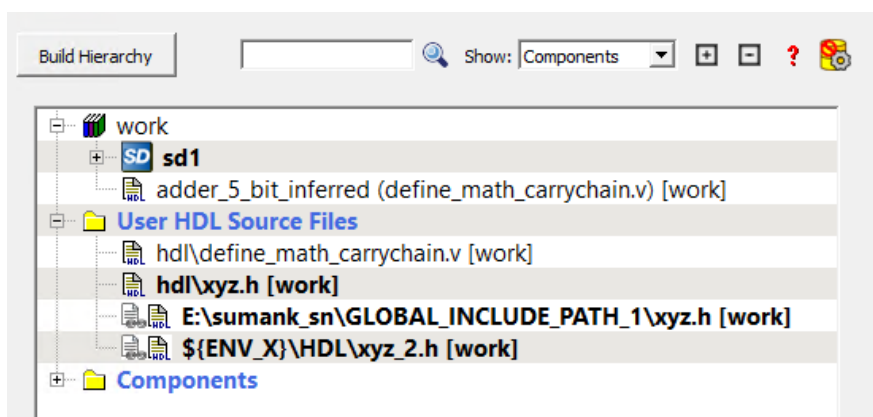
The files are audited only if they are part of the project.

## 12.22 Global Include Files

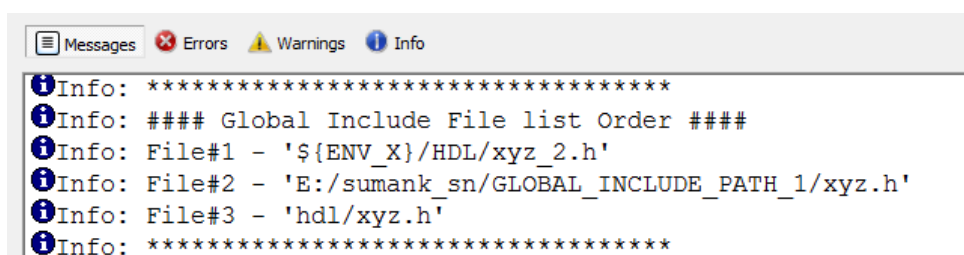
This option facilitates the users to provide global include file/s in Libero and manage how these files are sent to downstream tools occurring in Design Flow process.

Libero GUI facilitates user to provide Global Include Files. On right clicking a file in Design Hierarchy view, user can either set a file as global include by selecting “Set Global Include” option or unset a file as global include by selecting “Unset Global Include” option.

Linked files can also be set as Global Include files. The files that are set are highlighted in Design hierarchy view as shown below.



The selected order of global include files is shown in Info tab in log window as below:



### Note:

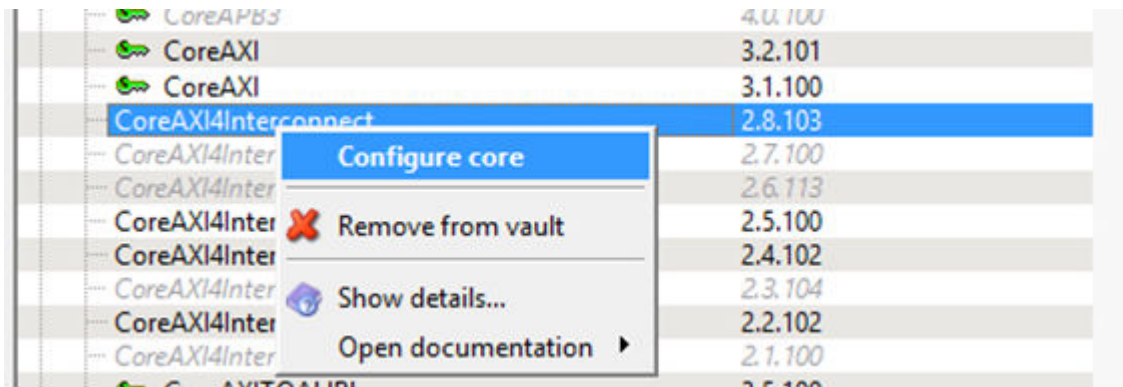
Setting Global Include File will use System Verilog MFCU mode internally. An info message will be generated in the log window as below:

Info: Global Include file requires System Verilog setting to be in MFCU mode.  
Enabling Multi File Compilation Unit internally.

## 12.23 Create and Configure Core Component

The user can create, configure, and generate a component for any core in the Libero SoC Catalog window. Select a core in the Catalog window by double clicking or right clicking on it and choose the option 'Configure core'. Specify a name to the core component to be created in the pop-up dialog box. The core's configurator window appears where the user can specify any desired configuration and click on the OK button to generate the core component.

Figure 12-32. Configure Core Option



Once the core component is created and generated, the component related files like the generated HDL netlist file are also generated to the directory <project directory>/component/work/<component\_name>. The generated HDL netlist file of the core component now contains the component description/configuration information in Tcl as comments. These comments also include Family Name and Part Number of the device used. The following figure shows a sample Tcl component description of the core COREAXI4INTERCONNECT.

Figure 12-33. Component Description of COREAXI4INTERCONNECT\_C0 Showing Family Name and Part Number

```
// Created by SmartDesign Tue Nov 24 00:05:58 2020
// Version: v12.6 12.900.20.21

`timescale 1ns / 100ps

// Component Description (Tcl)
/*
# Exporting Component Description of COREAXI4INTERCONNECT_C0 to TCL
# Family: PolarFire
# Part Number: MPF200TLS-FCG784I
# Create and Configure the core component COREAXI4INTERCONNECT_C0
create_and_configure_core -core_vlnv {Actel:DirectCore:COREAXI4INTERCONNECT:2.8.103} -component_name
{COREAXI4INTERCONNECT_C0} -params {
"ADDR_WIDTH:32" \
"CROSSBAR_MODE:1" \
"DATA_WIDTH:64" \
"DWC_ADDR_FIFO_DEPTH_CEILING:10" \

```

## 12.24 Search in Libero SoC

Search options vary depending on your search type.

To find a file:

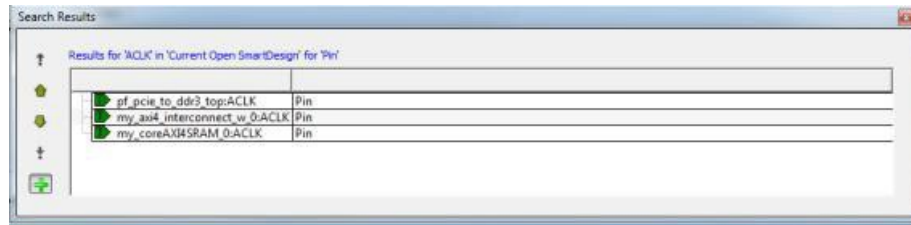
1. Use CTRL + F to open the Search window.
2. Enter the name or part of name of the object you wish to find in the Find field. '\*' indicates a wildcard, and [\*-\*] indicates a range, such as if you search for a1, a2, ... a5 with the string a[1-5].
3. Set the Options for your search (see below for list); options vary depending on your search type.
4. Click **Find All** (or **Next** if searching Text).  
Searching an open text file, Log window or Reports highlights search results in the file itself. All other results appear in the Search Results window (as shown in the figure below).

**Match case:** Select to search for case-sensitive occurrences of a word or phrase. This limits the search so it only locates text that matches the upper- and lowercase characters you enter.



**Match whole word:** Select to match the whole word only.

**Figure 12-34. Search Results**



### Current Open SmartDesign

Searches your open SmartDesign, returns results in the Search window.

**Type:** Choose Instance, Net or Pin to narrow your search.

**Query:** Query options vary according to Type.

Type	Query Option	Function
Instance	Get Pins	Search restricted to all pins
	Get Nets	Search restricted to all nets
	Get Unconnected Pins	Search restricted to all unconnected pins
Net	Get Instances	Searches all instances
	Get Pins	Search restricted to all pins
Pin	Get Connected Pins	Search restricted to all connected pins
	Get Associated Net	Search restricted to associated nets
	Get All Unconnected Pins	Search restricted to all unconnected pins

### Current Open Text Editor

Searches the open text file. If you have more than one text file open you must place the cursor in it and click CTRL + F to search it.

- **Find All:** Highlights all finds in the text file.
- **Next:** Proceed to next instance of found text.
- **Previous:** Proceed to previous instance of found text.
- **Replace with:** Replaces the text you searched with the contents of the field.
- **Replace:** Replaces a single instance.
- **Replace All:** Replaces all instances of the found text with the contents of the field.

### Design Hierarchy

Searches your Design Hierarchy; results appear in the Search window.

**Find All:** Displays all finds in the Search window.

### Stimulus Hierarchy

Searches your Stimulus Hierarchy; results appear in the Search window.

**Find All:** Displays all finds in the Search window.

### Log Window

Searches your Log window; results are highlighted in the Log window - they do not appear in the Search Results window.

**Find All:** Highlights all finds in the Log window.

**Next:** Proceed to next instance of found text.

**Previous:** Proceed to previous instance of found text.

### Reports

Searches your Reports; returns results in the Reports window.

**Find All:** Highlights all finds in the Reports window.

**Next:** Proceed to next instance of found text.

**Previous:** Proceed to previous instance of found text.

### Files

Searches your local project file names for the text in the Search field; returns results in the Search window.

**Find All:** Lists all search results in the Search window.

### Files on disk

Searches the files' content in the specified directory and subdirectories for the text in the Search field; returns results in the Search window.

**Find All:** Lists all finds in the Search window.

**File type:** Select a file type to limit your search to specific file extensions, or choose \*.\* to search all file types.

## 12.25 Organize Source Files Dialog Box – Synthesis

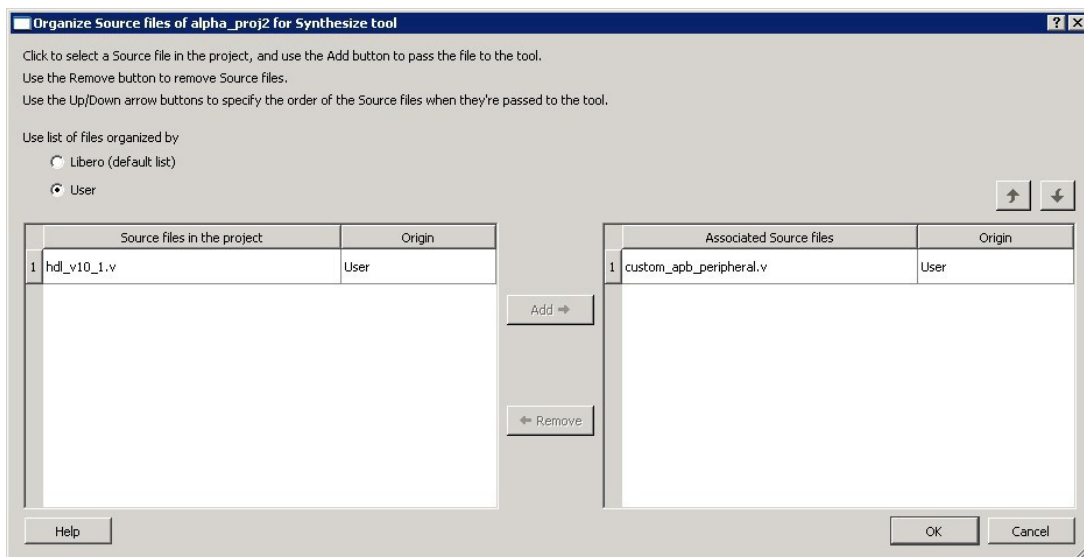
The Organize Source Files dialog box enables you to set the source file order in the Libero SoC.

Click the **Use list of files organized by User** radio button to Add/Remove source files for the selected tool.

To specify the file order:

1. In the Design Flow window under Implement Design, right-click **Synthesize** and choose **Organize Input Files > Organize Source Files**. The Organize Source Files dialog box appears.
2. Click the **Use list of files organized by User** radio button to Add/Remove source files for the selected tool.
3. Select a file and click the Add or Remove buttons as necessary. Use the Up and Down arrows to change the order of the Associated Source files.
4. Click **OK**.

Figure 12-35. Organize Source Files Dialog Box





## 12.26 SmartDesign Testbench

SmartDesign Testbench is a GUI-based tool that enables you to design your testbench hierarchy. Use SmartDesign Testbench to instantiate and connect stimulus cores or modules to drive your design.

You can create a SmartDesign Testbench by right-clicking a SmartDesign component in the Design Hierarchy and choosing **Create Testbench > SmartDesign**.

SmartDesign Testbench automatically instantiates the selected SmartDesign component into the Canvas.

You can also double-click **Create SmartDesign Testbench** in the Design Flow window to add a new SmartDesign testbench to your project.

New testbench files appear in the [Stimulus Hierarchy](#).

SmartDesign Testbench automatically instantiates your SmartDesign component into the Canvas.

You can instantiate your own stimulus HDL or simulation models into the SmartDesign Testbench Canvas and connect them to your DUT (design under test). You can also instantiate Simulation Cores from the [Catalog](#).

Simulation cores are simulation models (such as DDR memory simulation models) or basic cores that are useful for stimulus generation (such as Clock Generator, Pulse Generator, or Reset Generator).

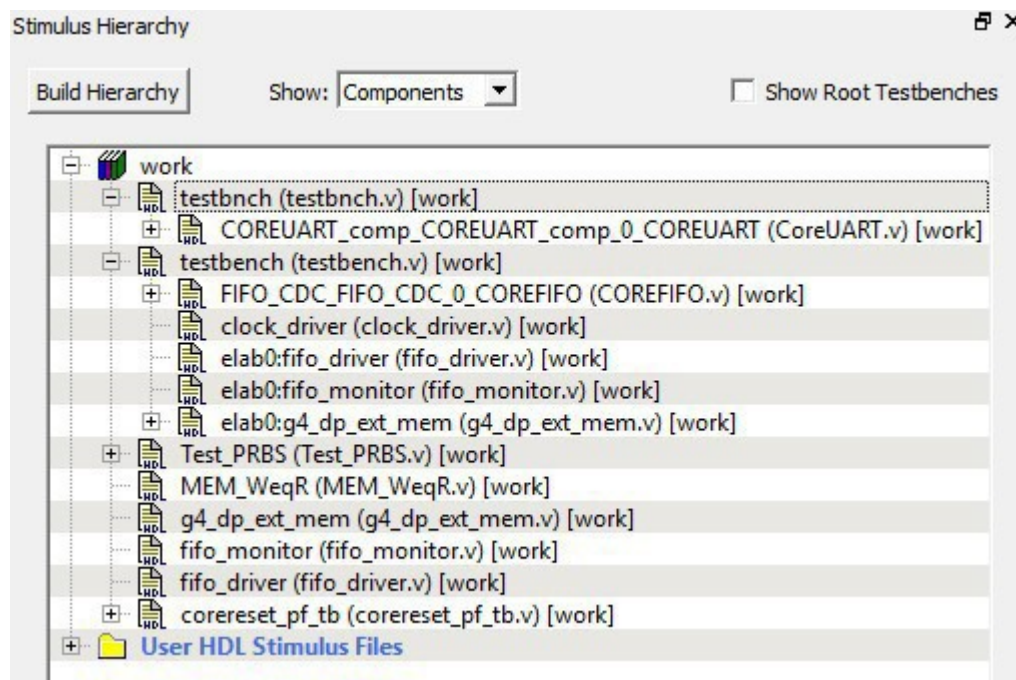
Click the Simulation Mode checkbox in the Catalog to view available simulation cores. Refer to the [SmartDesign User Guide](#) for more information.

## 12.27 Stimulus Hierarchy

To view the Stimulus Hierarchy, from the **View** menu choose **Windows > Stimulus Hierarchy**.

The Stimulus Hierarchy tab displays a hierarchical representation of the stimulus and simulation files in the project. The software continuously analyzes and updates files and content. The tab in the following figure displays the structure of the modules and component stimulus files as they relate to each other.

**Figure 12-36. Stimulus Hierarchy Dialog Box**



Expand the hierarchy to view stimulus and simulation files. Right-click an individual component and choose **Show Module** to view the module for only that component.

Select **Components**, instance or **Modules** from the **Show** drop-down list to change the display mode. The Components view displays the stimulus hierarchy; the modules view displays HDL modules and stimulus files.

The file name (the file that defines the module or component) appears in parentheses. Click **Show Root Testbenches** to view only the root-level testbenches in your design.

Right-click and choose **Properties**; the Properties dialog box displays the pathname, created date, and last modified date.







All integrated source editors are linked with the SoC software; if you modify a stimulus file the Stimulus Hierarchy automatically updates to reflect the change.

To open a stimulus file:

1. Double-click a stimulus file to open it in the HDL text editor.
2. Right-click and choose **Delete from Project** to delete the file from the project. Right-click and choose **Delete from Disk and Project** to remove the file from your disk.

The following table shows the icons in the hierarchy indicate the type of component and the state.

**Table 12-4. Design Hierarchy Icons**

Icon	Description
	SmartDesign component
	SmartDesign component with HDL netlist not generated
	SmartDesign testbench
	SmartDesign testbench with HDL netlist not generated
	IP core was instantiated into SmartDesign but the HDL netlist has not been generated
	HDL netlist

## 12.28 Timing Exceptions Overview

Use timing exceptions to overwrite the default behavior of the design path. Timing exceptions include:

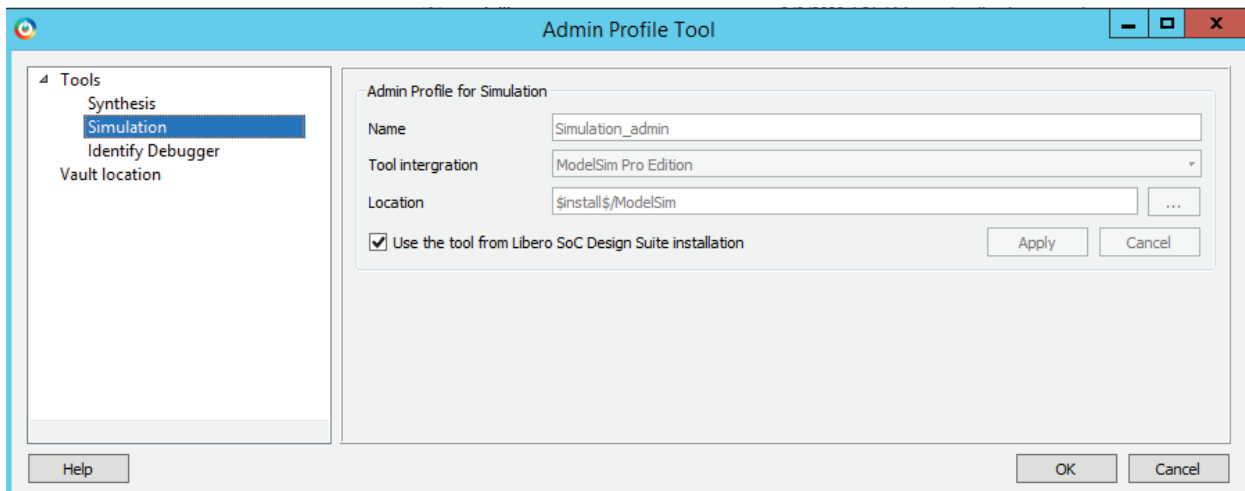
- Setting multicycle constraint to specify paths that (by design) will take more than one cycle.
- Setting a false path constraint to identify paths that must not be included in the timing analysis or the optimization flow.
- Setting a maximum/minimum delay constraint on specific paths to relax or to tighten the original clock constraint requirement.

## 12.29 Admin Profile Tool Dialog Box

This allows the users a way to use same set of tools and same vault location if Libero SoC is invoked from the same installed location. An admin (the person who generally takes care of Libero SoC installation) creates an admin profile. This profile will be common to all the users who are using Libero SoC from the same installed location. Individual users do not need to have to create the tool profile and vault location.

To create an admin profile, user needs to invoke `adminprofile.exe` from `<Libero installation directory>/Designer/bin` location in case of windows and from `<Libero installation directory>/bin64` location in case of Linux.

Figure 12-37. Admin Profile Tool Dialog Box



This admin profile will be common to all users who are using Libero SoC from the installed path, by providing the following values:

- Synplify location
- Modelsim location
- Vault location
- Any other tool mentioned in admin profile utility

**Note:** It is not mandatory for the user to create admin profile. If the user skips the creation of admin profile, the tool will continue to work normally.

If the user wishes to create admin profile, all the fields in the dialog box must be entered correctly. The vault location provided during admin profile creation will have precedence and will be used by all the other users.

#### Vaults and tools location

Vault location: Location of megavault consisting of all the cores for the current Libero SoC version must be mentioned.

Tools location: Correct tool executable location must be entered which is accessible by all users of Libero SoC.

Using admin profile, users will also have an option to use Libero SoC provided tools (which are a part of Libero SoC Design Suite) by selecting the **Use the tool from Libero SoC Design Suite installation** option. In this case, location of specific tools will be used from Libero SoC installation.

## 12.30 Tool Profiles Dialog Box

The Tool Profiles dialog box enables you to add, edit, or delete your project tool profiles.

Each Libero SoC project can have a different profile, enabling you to integrate different tools with different projects.

The following table shows the supported tool versions in this release.

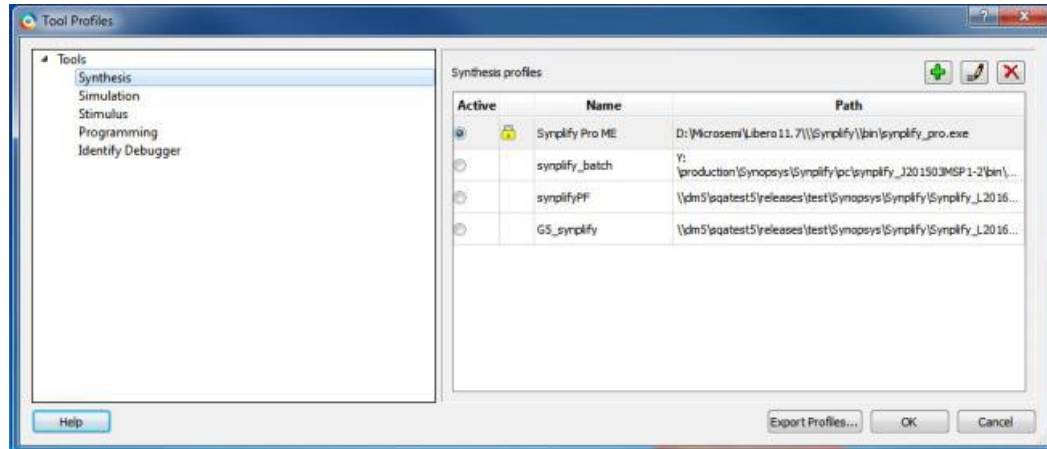
Table 12-5. Table for Supported Tool Versions

Tool	Supported Version
Modelsim ME	10.5c
Modelsim ME Pro	10.5c
SynplifyPro ME	N-2017.09M SP1-1
Identify ME	N-2017.09M SP1

**To set or change your tool profile:**

1. From the **Project** menu, choose **Tool Profiles**. Select the type of tool you wish to add.
  - **To add a tool:** Select the tool type and click the **Add** button. Fill out the tool profile and click **OK**.
  - **To change a tool profile:** After selecting the tool, click the **Edit** button to select another tool, change the tool name, or change the tool location.
  - **To remove a tool from the project:** After selecting a tool, click the **Remove** button.

**Figure 12-38. Libero SoC Tool Profiles Dialog Box**



2. When you are done, click **OK**.

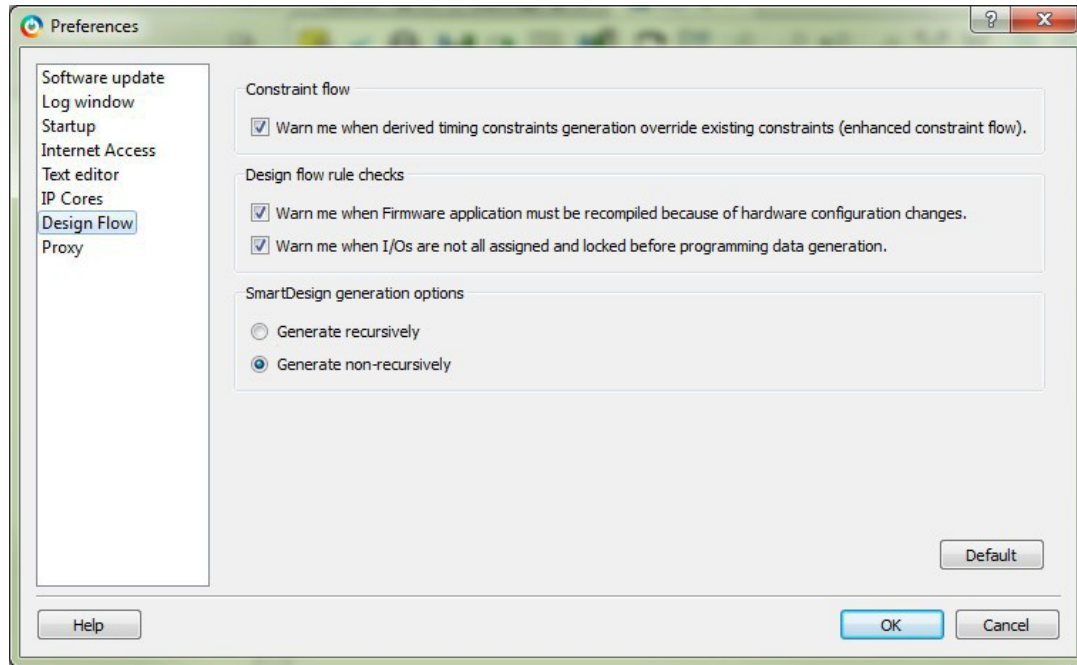
The tool profile with the padlock icon indicates that it is a pre-defined tool profile (the default tool that comes with the Libero SoC Installation).

To export the tool profile and save it for future use, click the **Export Tool Profiles** dialog box and save the tool profile file as a tool profile \*.ini file. The tool profile \*.ini file can be imported into a Libero SoC project (**File > Import > Others**) and select Tool Profiles (\*.ini) in the File Type pull-down list.

## 12.31 User Preferences Dialog Box – Design Flow Preferences

This dialog box allows you to set your personal preferences for how Libero SoC manages the design flow across the projects you create.

Figure 12-39. Preferences Dialog Box – Design Flow Preferences



#### Constraint Flow

- Warn me when derived timing constraints generation override existing constraints (enhanced constraint flow).  
Libero SoC can generate/derive timing constraints for known hardware blocks and IPs such as SerDes, CCC. Check this box to have Libero SoC pop up a warning message when the generated timing constraints for these blocks override the timing constraints you set for these blocks. This box is checked by default.

#### Design Flow Rule Checks

- Warn me when Firmware applications must be recompiled because of hardware configuration changes.  
Check this box if you want Libero SoC to display a warning message. This box is checked by default.
- Warn me when I/Os are not all assigned and locked before programming data generation.  
I/Os should always be assigned and locked before programming data generation. Check this box if you want Libero SoC to display a warning message. This box is checked by default.

#### SmartDesign Generation Options

- Generate recursively  
In this mode, all subdesigns must be successfully generated before a parent can be generated. An attempt to generate a SmartDesign results in an automatic attempt to generate all subdesigns.
- Generate non-recursively  
In this mode, the generation of only explicitly selected SmartDesigns is attempted. The generation of a design can be marked as successful even if a subdesign is ungenerated (either never attempted or unsuccessful).

**Note:** These preferences are stored on a per-user basis across multiple projects; they are not project-specific.

## 12.32 IP Report

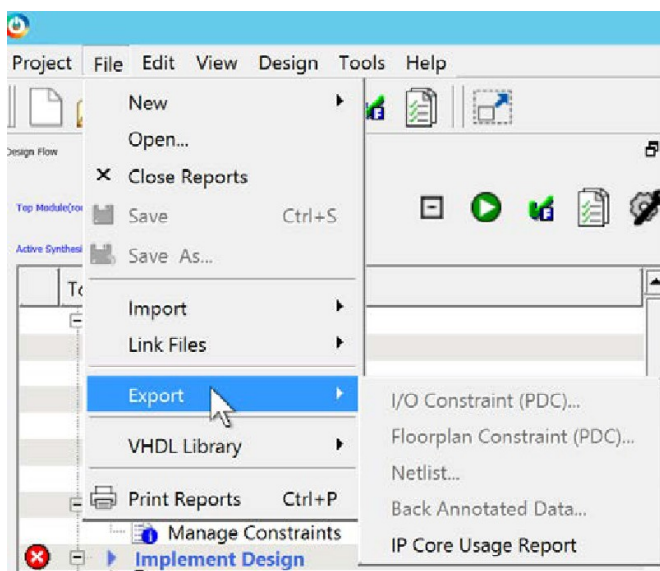
Libero tool generates a report listing all the IPs used in the design project.

The report consists of the following:

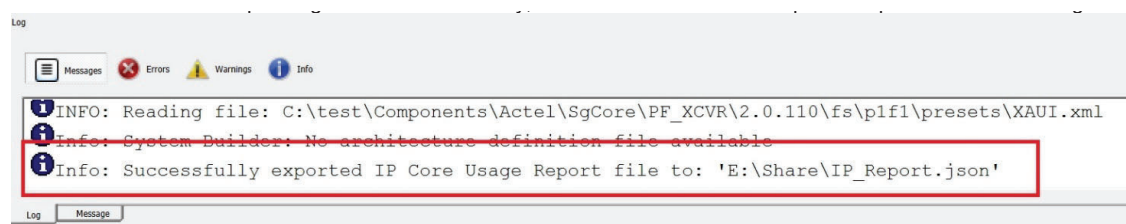
- Project Information
- List of components in the project with the below information:
  - Component name
  - Component State

- IP Core (Vendor,Library,Name,Version) used to create the component
- TCL Parameters of the core used to create the component
- Information of core instances used in the Smart Design.
  - Component Name – In the form of "smartdesign\_name>\_<instance\_name>"
  - Component State – State of the instance in the Smart Design
  - Vendor, Library, Name, and Version of the core used to create the instance
  - TCL Parameters of the core used to create the instance
- Single Page System Builder components
- Multi Page System Builder components without parameter list of latest core versions available for the cores used in the project. This is the list that is present at the end of open project.

In order to export the **IP CORE Usage Report**, go to File > Export > IP Core Usage Report. When clicked, a file dialog box launched asking for the location and file format to dump the report. The file can be generated in either "JSON" or "txt" format.



Once the report is generated successfully, the location to where the report is exported is shown in log window.

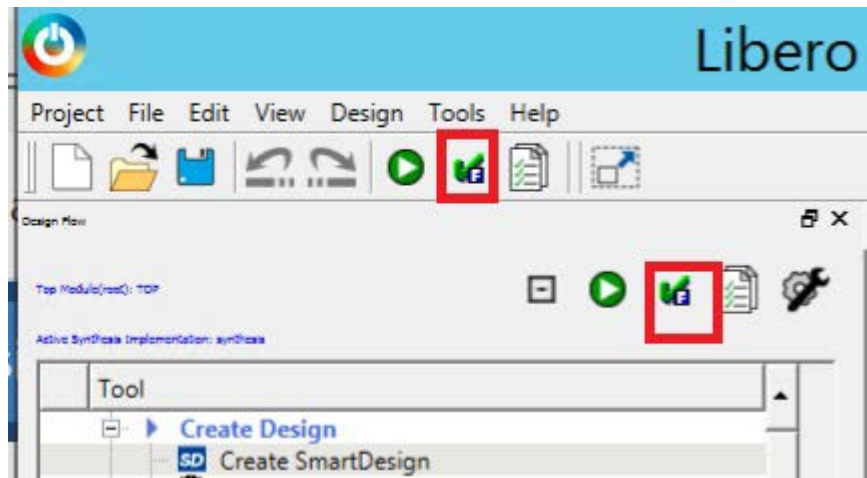


## 12.33 Force Update Design Flow

A global option to force update the design flow to Pass state has been added to Libero SoC v12.5. The tool's state is force updated to PASS state only for Synthesize/Compile and Place and Route tools. The remaining tools remain in the same state they were in before the force update action was performed.

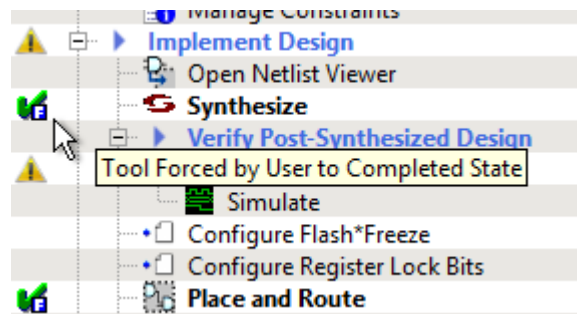


**Figure 12-40. Force Update Action Button at Top of Design Flow Window**



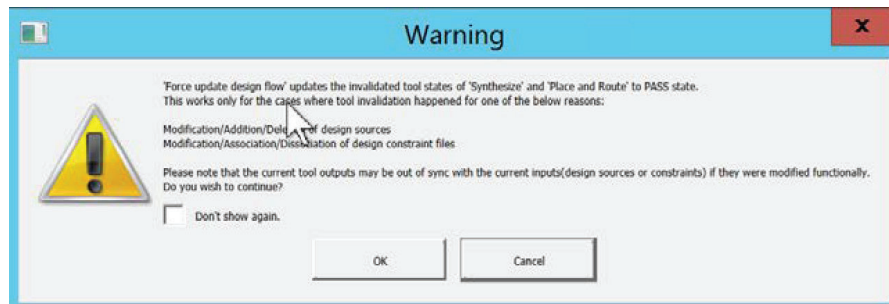
To differentiate between PASS state and FORCE UPDATE PASS state, the following icon and tooltip for the FORCE UPDATE PASS state (force update action) has been added.

**Figure 12-41. Force Update Action Icon with Tool Tip**



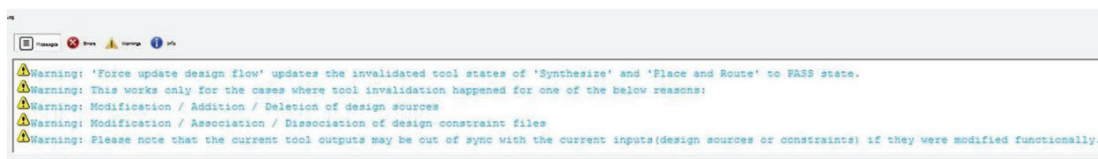
If you click the **Force update action** button, the following warning message appears. If you click **OK** in the message, the action proceeds. Otherwise, the action is canceled.

**Figure 12-42. Warning Message Pop-up**

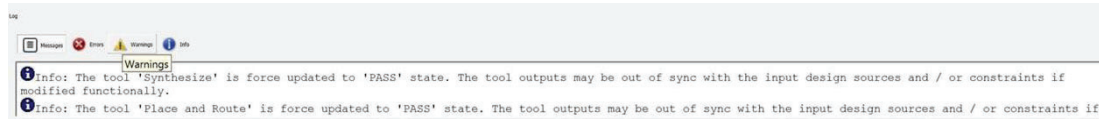


The message log window shows when the Force Update Action is performed.

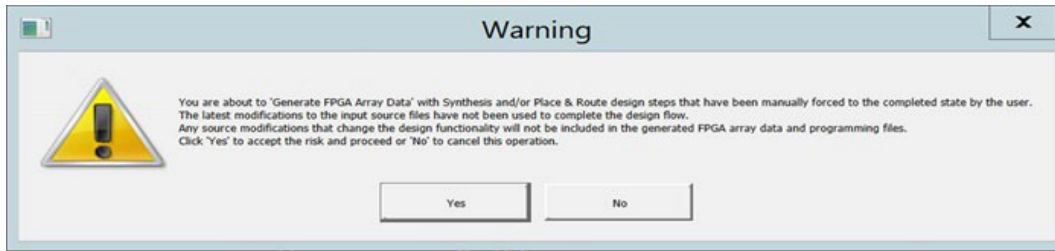
**Figure 12-43. Warning Message in Log Window**



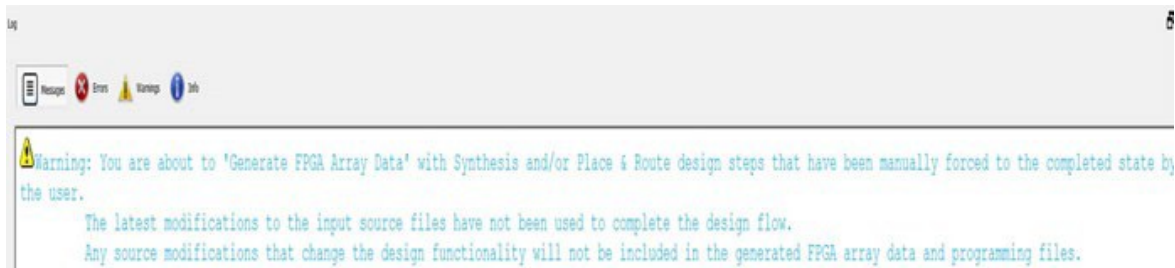
After the tool states are force updated to PASS state from Out of Date Design state, the following message appears in the log window.



On running programming tool – Generate FPGA Array Data, a warning message pops up if Synthesize, Place and Route, or both are in the Force Update Pass state.



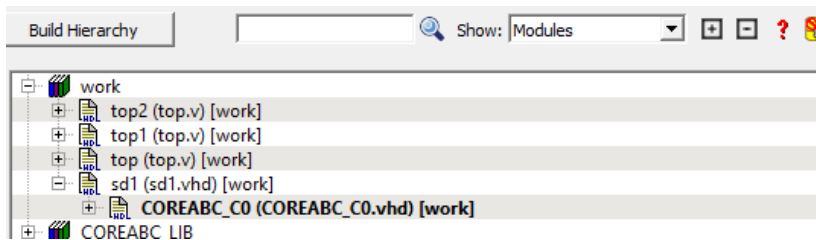
The log window shows the following message.



## 12.34 Generic/Parameter Report

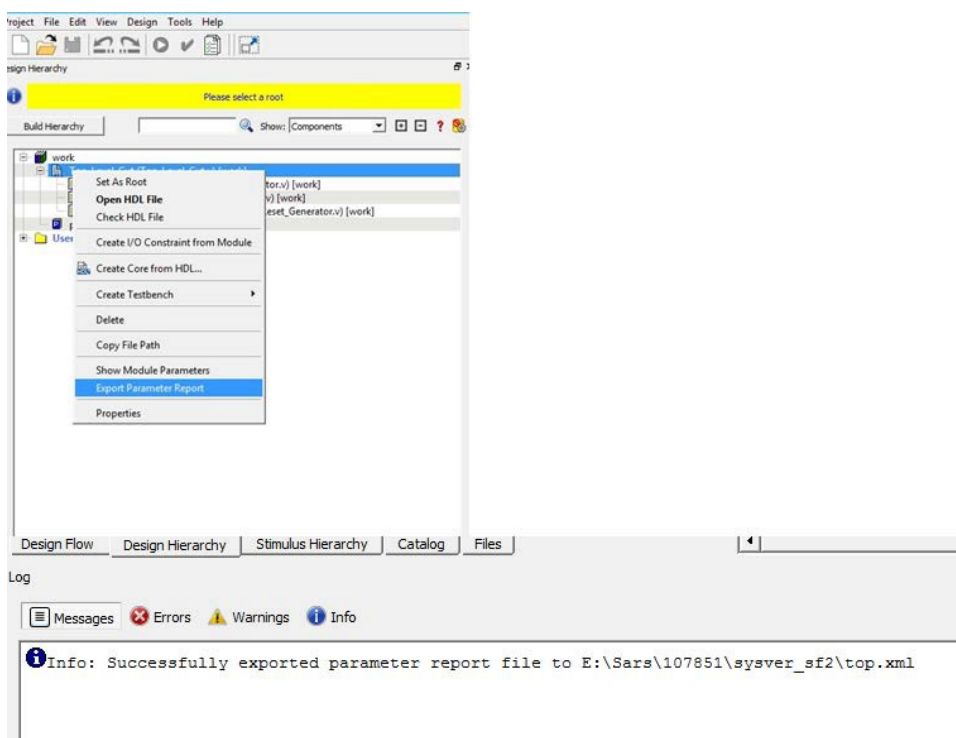
Libero tool generates hierarchical reports that show parameter/generic values used in instantiation of the modules. Report is generated for the active top module. Report will be generated for only that active top module which is not instantiated in any other modules. The report is generated in xml format.

In below example, report will not be generated for the active root(top) module **COREABC\_C0** because it is instantiated in sd1. The file sd1 needs to be set as a root to generate the report.



The report can be generated by right clicking on the active top module and selecting **Export Parameter Report** option in Design Hierarchy window. Once the report is generated successfully, the location to where the report is exported is shown in log window.





Error message is flagged in cases where the user tries to generate report for a module that is instantiated in another module.

"Error: Parameter/Generic report cannot be exported for a module that is instantiated in any other module. It can only be exported for top level modules. Set a top level module as the root or specify a top level module in the Tcl command 'export\_parameter\_report' to be able to export the report."

## 13. Revision History

Revision	Date	Description
B	4/2021	<p>The following changes are made in this revision:</p> <ul style="list-style-type: none"> <li>• <a href="#">4.5 Viewing Configured Components and SmartDesigns in a Project</a>: added this new topic.</li> <li>• <a href="#">6.1.2 Global Nets (Promotions and Demotions)</a>: added Clock Domain Crossing and effects of synchronizer.</li> <li>• <a href="#">6.7 Post Layout Editing of I/O Signal Integrity and Delay Parameters</a>: added this new topic.</li> <li>• <a href="#">8.2 Initializing Design Blocks</a>: reorganized and described Boot Mode 1 and Boot Mode 3.</li> <li>• <a href="#">8.7 Configuring Security</a>: added GUI screens and greater detail about security options.</li> <li>• <a href="#">Update Policy</a>: updated this section.</li> <li>• <a href="#">8.12.4 Partial Programming Support</a>: added this new section.</li> <li>• <a href="#">Export Bitstream</a>: added Advanced Options.</li> <li>• <a href="#">12. References</a>: removed commands.</li> <li>• <a href="#">12.21.2 Project Settings: Design flow</a>: added Multi-File Compilation Unit.</li> <li>• <a href="#">12.33 Force Update Design Flow</a>: added this new section.</li> </ul>
A	11/2020	Initial Revision

## The Microchip Website

---

Microchip provides online support via our website at [www.microchip.com/](http://www.microchip.com/). This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

## Product Change Notification Service

---

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to [www.microchip.com/pcn](http://www.microchip.com/pcn) and follow the registration instructions.

## Customer Support

---

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: [www.microchip.com/support](http://www.microchip.com/support)

## Microchip Devices Code Protection Feature

---

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods being used in attempts to breach the code protection features of the Microchip devices. We believe that these methods require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Attempts to breach these code protection features, most likely, cannot be accomplished without violating Microchip's intellectual property rights.
- Microchip is willing to work with any customer who is concerned about the integrity of its code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is "unbreakable." Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

## Legal Notice

Information contained in this publication is provided for the sole purpose of designing with and using Microchip products. Information regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications.

THIS INFORMATION IS PROVIDED BY MICROCHIP "AS IS". MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

## Trademarks

The Microchip name and logo, the Microchip logo, Adaptec, AnyRate, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PackeTime, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

AgileSwitch, APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, FlashTec, Hyper Speed Control, HyperLight Load, IntelliMOS, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, WinPath, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, Augmented Switching, BlueSky, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, Espresso T1S, EtherGREEN, IdealBridge, In-Circuit Serial Programming, ICSP, INICnet, Intelligent Paralleling, Inter-Chip Connectivity, JitterBlocker, maxCrypto, maxView, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICKit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, RTAX, RTG4, SAM-ICE, Serial Quad I/O, simpleMAP, SimpliPHY, SmartBuffer, SMART-I.S., storClad, SQL, SuperSwitcher, SuperSwitcher II, Switchtec, SynchroPHY, Total Endurance, TSHARC, USBCheck, VariSense, VectorBlox, VeriPHY, ViewSpan, WiperLock, XpressConnect, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, and Symmcom are registered trademarks of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2021, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

ISBN: 978-1-5224-7770-9

## Quality Management System

---

For information regarding Microchip's Quality Management Systems, please visit [www.microchip.com/quality](http://www.microchip.com/quality).

## Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
<b>Corporate Office</b> 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: <a href="http://www.microchip.com/support">www.microchip.com/support</a> Web Address: <a href="http://www.microchip.com">www.microchip.com</a>	<b>Australia - Sydney</b> Tel: 61-2-9868-6733 <b>China - Beijing</b> Tel: 86-10-8569-7000 <b>China - Chengdu</b> Tel: 86-28-8665-5511 <b>China - Chongqing</b> Tel: 86-23-8980-9588 <b>China - Dongguan</b> Tel: 86-769-8702-9880 <b>China - Guangzhou</b> Tel: 86-20-8755-8029 <b>China - Hangzhou</b> Tel: 86-571-8792-8115 <b>China - Hong Kong SAR</b> Tel: 852-2943-5100 <b>China - Nanjing</b> Tel: 86-25-8473-2460 <b>China - Qingdao</b> Tel: 86-532-8502-7355 <b>China - Shanghai</b> Tel: 86-21-3326-8000 <b>China - Shenyang</b> Tel: 86-24-2334-2829 <b>China - Shenzhen</b> Tel: 86-755-8864-2200 <b>China - Suzhou</b> Tel: 86-186-6233-1526 <b>China - Wuhan</b> Tel: 86-27-5980-5300 <b>China - Xian</b> Tel: 86-29-8833-7252 <b>China - Xiamen</b> Tel: 86-592-2388138 <b>China - Zhuhai</b> Tel: 86-756-3210040	<b>India - Bangalore</b> Tel: 91-80-3090-4444 <b>India - New Delhi</b> Tel: 91-11-4160-8631 <b>India - Pune</b> Tel: 91-20-4121-0141 <b>Japan - Osaka</b> Tel: 81-6-6152-7160 <b>Japan - Tokyo</b> Tel: 81-3-6880-3770 <b>Korea - Daegu</b> Tel: 82-53-744-4301 <b>Korea - Seoul</b> Tel: 82-2-554-7200 <b>Malaysia - Kuala Lumpur</b> Tel: 60-3-7651-7906 <b>Malaysia - Penang</b> Tel: 60-4-227-8870 <b>Philippines - Manila</b> Tel: 63-2-634-9065 <b>Singapore</b> Tel: 65-6334-8870 <b>Taiwan - Hsin Chu</b> Tel: 886-3-577-8366 <b>Taiwan - Kaohsiung</b> Tel: 886-7-213-7830 <b>Taiwan - Taipei</b> Tel: 886-2-2508-8600 <b>Thailand - Bangkok</b> Tel: 66-2-694-1351 <b>Vietnam - Ho Chi Minh</b> Tel: 84-28-5448-2100	<b>Austria - Wels</b> Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 <b>Denmark - Copenhagen</b> Tel: 45-4485-5910 Fax: 45-4485-2829 <b>Finland - Espoo</b> Tel: 358-9-4520-820 <b>France - Paris</b> Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 <b>Germany - Garching</b> Tel: 49-8931-9700 <b>Germany - Haan</b> Tel: 49-2129-3766400 <b>Germany - Heilbronn</b> Tel: 49-7131-72400 <b>Germany - Karlsruhe</b> Tel: 49-721-625370 <b>Germany - Munich</b> Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 <b>Germany - Rosenheim</b> Tel: 49-8031-354-560 <b>Israel - Ra'anana</b> Tel: 972-9-744-7705 <b>Italy - Milan</b> Tel: 39-0331-742611 Fax: 39-0331-466781 <b>Italy - Padova</b> Tel: 39-049-7625286 <b>Netherlands - Drunen</b> Tel: 31-416-690399 Fax: 31-416-690340 <b>Norway - Trondheim</b> Tel: 47-72884388 <b>Poland - Warsaw</b> Tel: 48-22-3325737 <b>Romania - Bucharest</b> Tel: 40-21-407-87-50 <b>Spain - Madrid</b> Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 <b>Sweden - Gothenberg</b> Tel: 46-31-704-60-40 <b>Sweden - Stockholm</b> Tel: 46-8-5090-4654 <b>UK - Wokingham</b> Tel: 44-118-921-5800 Fax: 44-118-921-5820