

Introduction (Ask a Question)

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 with Microchip's power efficient flash [Field-Programmable Gate Arrays \(FPGAs\)](#), [SoC FPGAs](#), and [Radiation-Tolerant FPGAs](#). The suite integrates industry-standard Synopsys [SynplifyPro®](#) synthesis and Siemens [QuestaSim®](#) ME Pro simulation with best-in-class constraints management, debug capabilities, and secure production programming support.

Supported Device Families

The following table lists the family of devices that Libero SoC supports. This guide covers all these device families. However, some information in this guide might apply to certain device families only. In this case, such information is clearly identified.

Table 1. Device Families Supported by Libero SoC

Device Family	Description
PolarFire®	PolarFire FPGAs deliver the industry's lowest power at mid-range densities with exceptional security and reliability.
PolarFire SoC	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.
SmartFusion® 2	SmartFusion 2 addresses fundamental requirements for advanced security, high reliability, and low power in critical industrial, military, aviation, communications, and medical applications.
IGLOO® 2	IGLOO 2 is a low-power mixed-signal programmable solution.
RTG4™	RTG4 is Microchip's family of Radiation-Tolerant FPGAs.

Helpful Links

- [Datasheets, tutorials, application notes, and silicon user guides](#)
- [Development boards and kits](#)
- [Libero SoC v12.0 and later](#)
- [Programming Solutions](#)
- [Libero licensing](#)
- [Libero SoC PolarFire product family](#)
- [Libero SoC PolarFire technology](#)
- [Libero SoC PolarFire documentation](#)
- [Libero SoC PolarFire software tools](#)

Microchip also provides power calculator estimators for device families. For more information, navigate to the [Libero SoC Design Suite Versions page](#).



Important: This document is updated frequently. The latest version of this document is available at this location: [Libero SoC Design Suite Documentation](#).

Table of Contents

Introduction.....	1
1. Overview.....	6
1.1. Libero SoC Design Flow.....	6
1.2. File Types in Libero SoC.....	11
1.3. Software Tools.....	12
1.4. Software IDE Integration.....	13
2. Managing Licenses.....	14
2.1. Microchip License Utility.....	14
2.2. Selecting a Default License from a License List.....	15
2.3. Setting a Default License.....	17
2.4. Viewing License Details.....	18
2.5. Libero SoC Online Help.....	20
2.6. Libero SoC User Guides.....	20
3. Getting Started.....	21
3.1. Starting Libero SoC.....	21
3.2. Design Report.....	21
3.3. Creating a New Project.....	21
3.4. Opening a Project.....	32
4. Creating and Verifying Designs.....	34
4.1. SmartFusion 2 and IGLOO 2 Tools.....	34
4.2. Create SmartDesign.....	36
4.3. Create Core from HDL.....	45
4.4. Designing with HDL.....	47
4.5. HDL Testbench.....	49
4.6. Designing with Block Flow.....	51
4.7. Viewing Configured Components and SmartDesigns in a Project.....	51
4.8. Create a New SmartDesign Testbench.....	54
4.9. Import MSS.....	54
4.10. Verify Pre-Synthesized Design - RTL Simulation.....	55
5. Libero SoC Constraint Management.....	61
5.1. Opening Constraint Manager.....	61
5.2. Libero SoC Design Flow.....	61
5.3. Introduction to Constraint Manager.....	62
5.4. Import a Constraint File.....	67
5.5. Constraint Types.....	71
5.6. Constraint Manager – I/O Attributes Tab.....	71
5.7. IO Advisor (SmartFusion 2, IGLOO 2, and RTG4).....	73
5.8. Constraint Manager: Timing Tab.....	80
5.9. Derived Constraints.....	84
5.10. Automatic Clock Jitter Constraint.....	85
5.11. Constraint Manager: Floor Planner Tab.....	86
5.12. Constraint Manager: Netlist Attributes Tab.....	87

6. Implementing Designs.....	90
6.1. Synthesize.....	90
6.2. Verifying Post-Synthesized Designs.....	99
6.3. Compile Netlist.....	100
6.4. Configure Flash*Freeze (SmartFusion 2 and IGLOO 2).....	101
6.5. Configure Register Lock Bits.....	101
6.6. Constraint Flow in Implementation.....	103
6.7. Place and Route.....	107
6.8. Multiple Pass Layout Configuration.....	112
6.9. Post Layout Editing of I/O Signal Integrity and Delay Parameters.....	114
6.10. Resource Usage.....	115
6.11. Global Net Report.....	118
6.12. Verify Post Layout Implementation.....	124
7. Configure Hardware.....	147
7.1. Programming Connectivity and Interface.....	147
7.2. Programmer Settings.....	150
7.3. Working with Local and Remote Programmers.....	151
7.4. Configuring and Managing Remote Servers.....	152
7.5. Remote Programming Server (RPS) Installation Instructions.....	153
7.6. Remote Programmer Troubleshooting.....	154
8. Program Design.....	156
8.1. Generating FPGA Array Data.....	156
8.2. Initializing Design Blocks (PolarFire and PolarFire SoC).....	156
8.3. Generating Initialization Clients (PolarFire).....	194
8.4. Update uPROM Memory Content (RTG4).....	195
8.5. Update eNVM Memory Content (SmartFusion 2 and IGLOO 2).....	198
8.6. Modify Data Storage Client (SmartFusion 2 and IGLOO 2).....	198
8.7. Modify Serialization Client (SmartFusion 2 and IGLOO 2).....	199
8.8. Configuring I/O States During JTAG Programming.....	200
8.9. Configuring Programming Options.....	204
8.10. Configuring Security.....	215
8.11. Configuring Bitstreams.....	234
8.12. Generating the Bitstream.....	236
8.13. Configuring Actions and Procedures.....	237
8.14. Running Programming Device Actions.....	241
8.15. Programming SPI Flash Image (PolarFire and PolarFire SoC).....	246
9. Debug Design.....	252
9.1. Generating SmartDebug FPGA Array Data (PolarFire).....	252
9.2. Using the SmartDebug Tool.....	252
9.3. Identifying the Debug Design.....	253
10. Handoff Design for Production.....	254
10.1. Configuring Security Locks (and One-Way Passcode for PolarFire SoC) for Production.....	254
10.2. Exporting Bitstreams.....	257
10.3. Exporting FlashPro Express Jobs.....	277
10.4. Exporting Job Manager Data.....	284
10.5. Export a SPI Flash Image (PolarFire and PolarFire SoC).....	285

10.6. Exporting Pin Reports.....	287
10.7. Exporting BSDL Files.....	287
10.8. Exporting IBIS Models.....	288
10.9. Export Initialization Data and Memory Report (PolarFire and PolarFire SoC).....	289
10.10. Exporting µPROM Reports (RTG4).....	290
11. Handoff Design for Firmware Development (SmartFusion 2 and IGLOO 2).....	291
11.1. Software IDE Integration (SmartFusion 2 and IGLOO 2).....	291
11.2. Viewing/Configuring Firmware Cores (SmartFusion 2 and IGLOO 2).....	291
11.3. Exporting Firmware (SmartFusion 2 and IGLOO 2).....	293
12. Handoff Design for Debugging.....	296
12.1. Exporting SmartDebug Data.....	296
13. References.....	300
13.1. Archive Project Dialog Box.....	300
13.2. Adding or Modifying Bus Interfaces in SmartDesign.....	301
13.3. Catalog.....	303
13.4. Changing Output Port Capacitance.....	306
13.5. Core Manager.....	307
13.6. Configure Security Locks for Production.....	308
13.7. Importing Source Files by Copying Files Locally.....	308
13.8. Create Clock Constraint Dialog Box.....	309
13.9. Select Source Pins for Clock Constraint Dialog Box.....	310
13.10. Specifying Clock Constraints.....	312
13.11. Specifying Generated Clock Constraints.....	317
13.12. Design Hierarchy in the Design Explorer.....	320
13.13. Digest File.....	326
13.14. Design Rules Check.....	327
13.15. Editable Constraints Grid.....	328
13.16. Files Tab and File Types.....	329
13.17. Importing Files.....	329
13.18. Layout Error Message: layoutg4NoValidPlacement.....	330
13.19. Layout Error Message: layoutg4DesignHard.....	331
13.20. Save Project As Dialog Box.....	331
13.21. Project Settings Dialog Box.....	332
13.22. Global Include Files.....	342
13.23. Create and Configure Core Component.....	343
13.24. Search in Libero SoC.....	344
13.25. Organize Source Files Dialog Box – Synthesis.....	346
13.26. SmartDesign Testbench.....	347
13.27. Stimulus Hierarchy.....	347
13.28. Timing Exceptions Overview.....	349
13.29. Admin Profile Tool Dialog Box.....	349
13.30. Tool Profiles Dialog Box.....	350
13.31. User Preferences Dialog Box – Design Flow Preferences.....	351
13.32. IP Report.....	352
13.33. Force Update Design Flow.....	353
13.34. Generic/Parameter Report.....	355

13.35. Export Back Annotated Files.....	356
13.36. Absolute and Relative Path Support for MSS.....	357
14. Revision History.....	359
Microchip FPGA Support.....	364
Microchip Information.....	365
Trademarks.....	365
Legal Notice.....	365
Microchip Devices Code Protection Feature.....	365

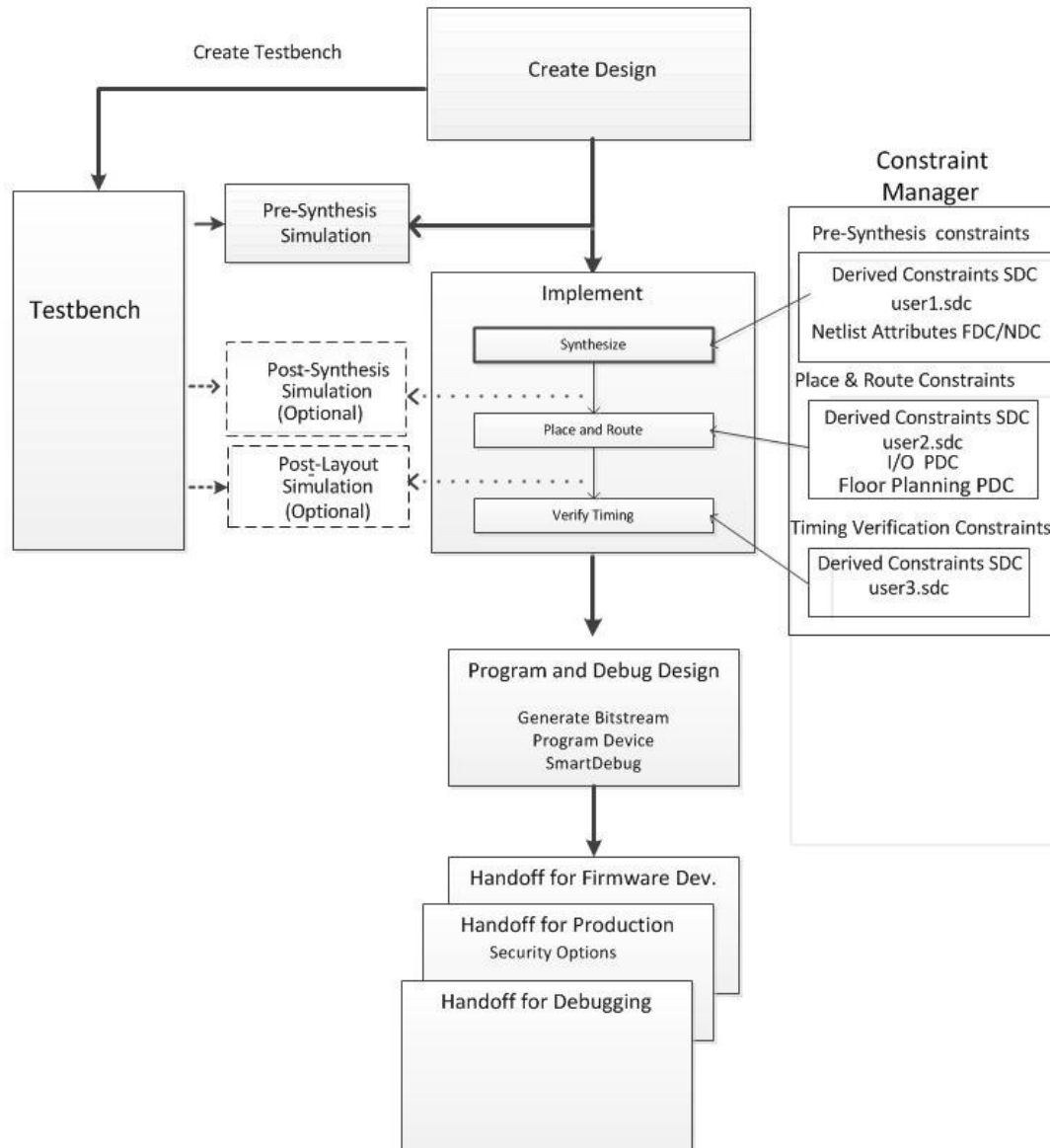
1. Overview [\(Ask a Question\)](#)

The following topics provide an overview of Libero SoC.

1.1. Libero SoC Design Flow [\(Ask a Question\)](#)

The following figure shows the Libero SoC design flow.

Figure 1-1. Libero SoC Design Flow



1.1.1. Creating the Design [\(Ask a Question\)](#)

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, start the simulation for pre-synthesis verification.



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

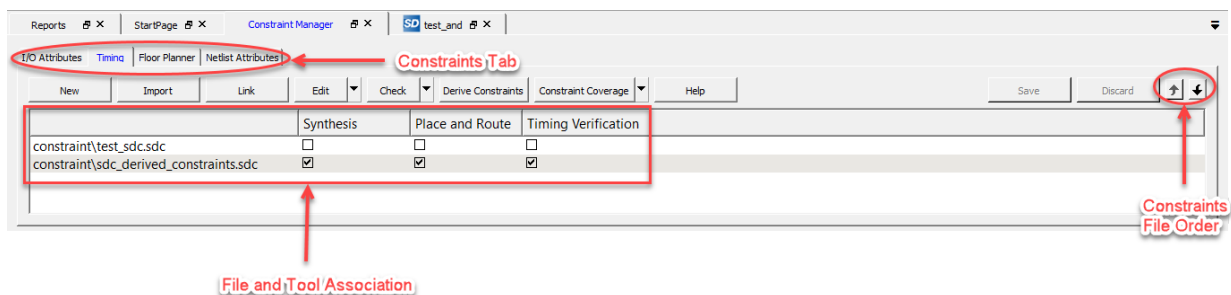
1.1.2. Working with Constraints [\(Ask a Question\)](#)

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, and 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



1.1.2.1. Constraint Flow and Design Sources [\(Ask a Question\)](#)

The Constraint flow supports HDL and Netlist design sources. The Libero SoC Design Flow window and the Constraint Manager are context-sensitive according to whether the design source is HDL or Netlist.

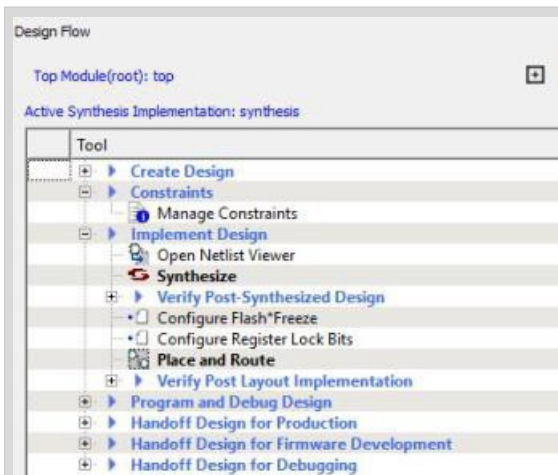
1.1.2.2. Constraint Flow for VM Netlist Designs [\(Ask a Question\)](#)

When the design source is a Netlist, the Design Flow window displays Compile Netlist as a design step. Timing constraints can be passed to Place and Route and Timing Verification only.

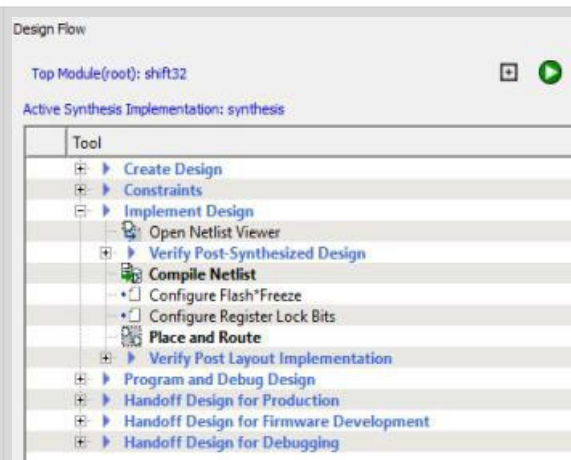
The options to promote or demote global resources of the chip are set in the Compile Netlist options. The following figure compares and contrasts the HDL flow versus the Netlist flow.

Table 1-1. HDL Flow vs. Netlist Flow

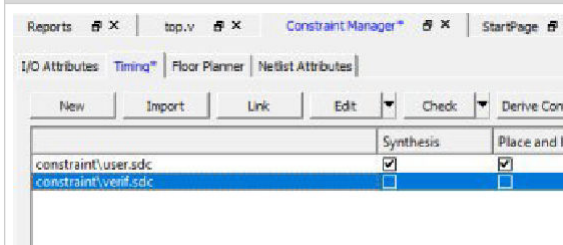
HDL Flow	Netlist Flow
----------	--------------



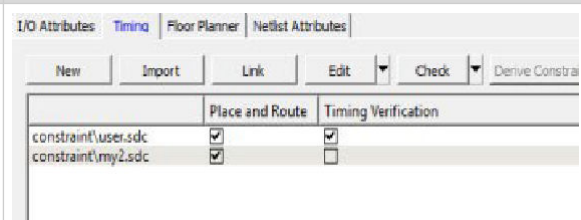
Design Flow Window



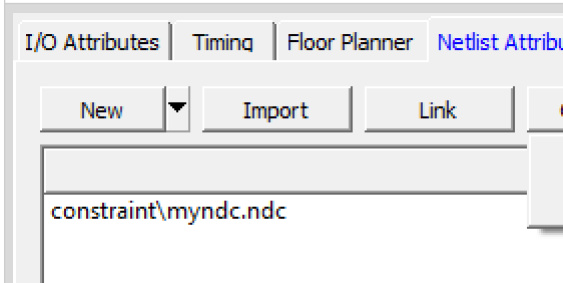
Design Flow Window



Constraint Manager



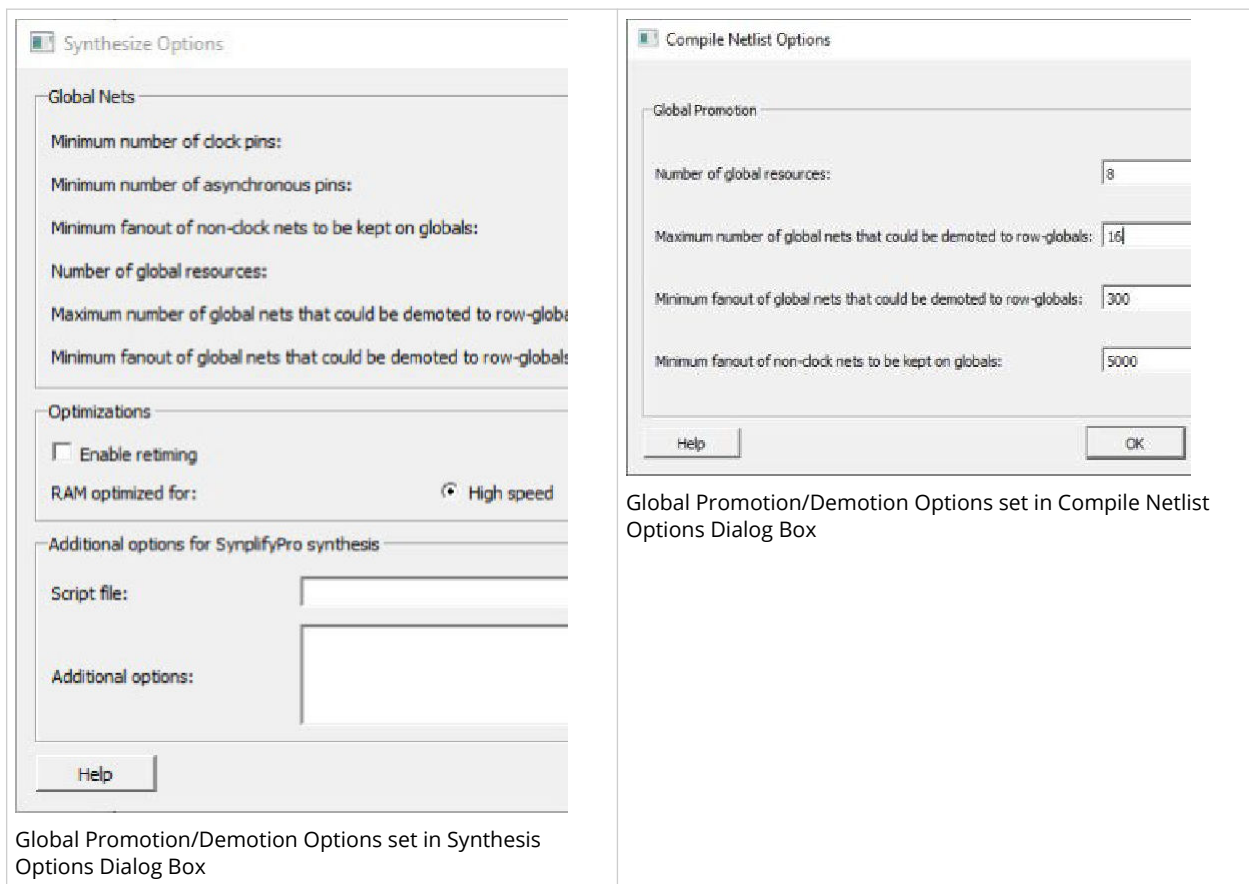
Constraint Manager



Constraint Manager - Check *.fdc and *.ndc



Constraint Manager - Check *.ndc only



1.1.2.3. Constraint Flow for HDL Designs [\(Ask a Question\)](#)

When the design source is HDL, the Design Flow window displays Synthesis as a design step. The Constraint Manager also makes available Synthesis as a target to receive timing constraints and netlist attribute constraints. The options to promote or demote global resources of the chip are set in the Synthesis options.

1.1.3. Implementing the Design [\(Ask a Question\)](#)

The following references provide information about implementing your design.

- [Netlist Viewer User Guide](#)
- [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.
- [Verifying Post-Synthesized Designs](#)
- [Configure Flash*Freeze \(SmartFusion 2 and IGLOO 2\)](#)
- [Configure Register Lock Bits](#)
- [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.
- [Verify Post Layout Implementation](#)
 - [Generate Back Annotated Files](#)
 - [Simulate - Opens QuestaSim ME Pro](#)
 - [Verify Timing](#)
 - [SmartTime](#)
 - [Verify Power](#)

- [Simultaneous Switching Noise](#)

1.1.4. **Programming and Debugging the Design** [\(Ask a Question\)](#)

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

Generating and Updating Data

- [Generating FPGA Array Data](#)
- [Initializing Design Blocks \(PolarFire and PolarFire SoC\)](#)
- [Generating Initialization Clients \(PolarFire\)](#)

Configuring the Hardware

- [Configuring Actions and Procedures](#)
- [Configuring I/O States During JTAG Programming](#)
- [Configuring Programming Options](#)

Programming the Design

- [Generating FPGA Array Data](#)
- [Initializing Design Blocks \(PolarFire and PolarFire SoC\)](#)
- [Generating Initialization Clients \(PolarFire\)](#)
- [Configuring Actions and Procedures](#)
- [Configuring I/O States During JTAG Programming](#)
- [Configuring Programming Options](#)
- [Configuring Security](#)
- [Configure Security Locks for Production](#)
- [Configuring Bitstreams](#)
- [Generating the Bitstream](#)
- [Running Programming Device Actions](#)
- [Programming SPI Flash Image \(PolarFire and PolarFire SoC\)](#)

Debugging the Design

- [Generating SmartDebug FPGA Array Data \(PolarFire\)](#)
- [Using the SmartDebug Tool](#)
- [Identifying the Debug Design](#)

1.1.5. **Handing Off the Design to Production** [\(Ask a Question\)](#)

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

- [Exporting Bitstreams](#)
- [Exporting FlashPro Express Jobs](#)
- [Exporting Job Manager Data](#)
- [Export a SPI Flash Image \(PolarFire and PolarFire SoC\)](#)
- [Exporting Pin Reports](#)
- [Exporting BSDL Files](#)
- [Exporting IBIS Models](#)
- [Export Initialization Data and Memory Report \(PolarFire and PolarFire SoC\)](#)

- [Exporting µPROM Reports \(RTG4\)](#)
- [Exporting SmartDebug Data](#)

1.1.6. Handing Off Design for Firmware (SmartFusion® 2, IGLOO® 2, and RTG4™) [\(Ask a Question\)](#)

The following topics provide information about handing off your design for firmware. These topics apply to SmartFusion 2, IGLOO 2, and RTG4.

- [Software IDE Integration \(SmartFusion 2 and IGLOO 2\)](#)
- [Viewing/Configuring Firmware Cores \(SmartFusion 2 and IGLOO 2\)](#)
- [Exporting Firmware \(SmartFusion 2 and IGLOO 2\)](#)

1.2. File Types in Libero SoC [\(Ask a Question\)](#)

Creating a new project in Libero SoC creates new directories and project files automatically. The project directory contains your local project files. When you import files from outside your current project, the files are copied into your local project folder.

The Project Manager allows you to manage your files as you import them. For example, to store and maintain your design source files and design constraint files in a central location outside the project location, you can link them to the Libero project folders when you created the project. These files are linked, not copied, to the project folder.

Depending on your project preferences and the installed version of Libero SoC, the software creates directories for your project. The following table describes these directories.

Table 1-2. Libero SoC Directories

<project_name>	This is the top-level directory. It contains the *.prjx file. Only one *.prjx file is enabled for each Libero SoC project. If you associate Libero SoC as the default program with the *.prjx file (Project > Preferences > Startup > Check the default file association (.prjx) at startup), you can open the project with Libero SoC by double-clicking the *.prjx file.
component	Contains SmartDesign components (.sdb and .cxf files) and the *_manifest.txt file for each design component in your Libero SoC project. To run synthesis, simulation, and firmware development with your own point tools outside the Libero SoC environment, see the *_manifest.txt file. For each design component, Libero SoC generates a <component_name>_manifest.txt file that stores the file name and location of: • HDL source files to be used for synthesis and simulations • Stimulus files and configuration files for simulation • Firmware files for software IDE tools • Configuration files for programming • Configuration files for power analysis To run synthesis, simulation, firmware development, programming, and power analysis outside the Libero SoC environment, see the <i>SmartFusion 2/IGLOO 2 Custom Flow User Guide</i>. Note: When importing components, make sure that .sdb and .cxf files reside in the same directory.
constraint	Contains all the constraint files: • SDC timing constraint files • Floorplanning PDC files • I/O PDC files • Netlist Attributes NDC files

designer	Contains the following files: - For Silicon Explorer: *_ba.sdf, *_ba.v(hd), STP, and PRB - To run designer: TCL - Local project file relative to revision: impl.prj_des - Log file: designer.log
hdl	Contains all HDL sources: - VHDL: *.vhd files - Verilog: *.v and *.h files
simulation	Contains the following files for simulation: - meminit.dat - modelsim.ini *.bfm *.vec file - run.do
smartgen	Contains GEN files and LOG files from generated cores.
stimulus	Contains BTIM, Verilog, and VHDL stimulus files.
synthesis	Contains files generated by the tools (not managed by Libero SoC), including: *.vm file - Synplify log file: *_syn.prj - Precision project file: *.psp - Synplify log file: *.srr - Precision log file: precision.log - To run synthesis: *.tcl
viewdraw	Contains viewdraw.ini files.

1.2.1. Internal Files [\(Ask a Question\)](#)

Libero SoC generates the following internal files, some of which are encrypted. These files are for Libero SoC housekeeping and are not intended for you to use.

File	File Extension	Remarks
Routing Segmentation File	*.seg	None
Combiner Info	*.cob	None
Hierarchical Netlist	*.adl	None
Flattened Netlist	*.afl	None
Location file	*.loc	None
map file	*.map	Fabric Programming File
tieoffs.txt	*.txt	RTG4 devices only

1.3. Software Tools [\(Ask a Question\)](#)

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 www.microchip.com/en-us/products/fpgas-and-plds/fpga-and-soc-design-tools/fpga/libero-software-later-versions#overview.

Table 1-3. 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	Synplify Pro ME is integrated as part of the design package, enabling designers to target HDL code to specific devices.
Simulation	QuestaSim® ME Pro	Allows source-level verification so designers can verify HDL code. 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: • ChipPlanner: displays I/O and logic macros in your design for floorplanning • Netlist Viewer: design schematic viewer. • SmartPower: power analysis tool. • SmartTime: static timing analysis and constraints editor.

1.4. Software IDE Integration [\(Ask a Question\)](#)

Libero SoC simplifies the task of transitioning between designing your FPGA and developing your embedded firmware by managing the firmware for your FPGA hardware design. This includes managing:

- Firmware hardware abstraction layers required for your processor.
- Firmware drivers for the processor peripherals in your FPGA design.
- Sample application projects available for drivers that show how to use the APIs properly.

To see which firmware drivers Libero SoC 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.

2. Managing Licenses [\(Ask a Question\)](#)

This chapter describes Libero SoC licensing.

2.1. Microchip License Utility [\(Ask a Question\)](#)

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

Starting the Microchip License Utility

Click **<Installation Directory>:\Microchip\Libero SoC_vXXXX_X\Designer\bin** > **actel_lutil.exe**.

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, navigate to the Microchip [Licensing page](#), and then click one of the Purchase License options or click **Register a Free License**. You are redirected to the microchipDIRECT website to generate a license.

The following table describes the available licenses.

Table 2-1. Available Licenses

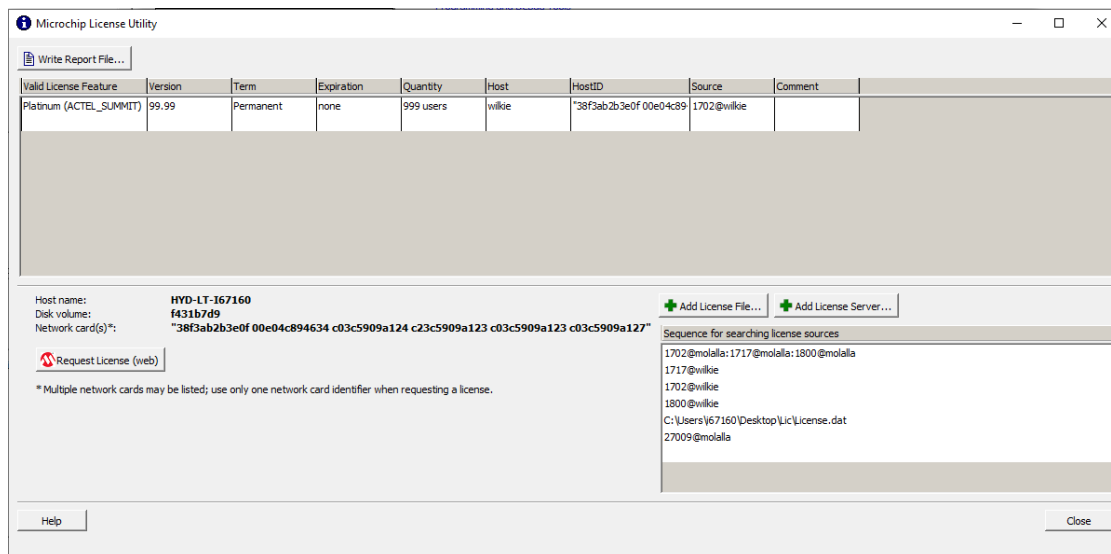
License	Description
1 year Platinum	Purchased license that supports all devices.
1 year Gold	Purchased license that supports a smaller set of devices than Platinum.
1 year Silver	Free license that supports a smaller set of devices than Gold.
30-day Evaluation	Free license that supports all devices, but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



Printing the Licenses Report

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

Related Information

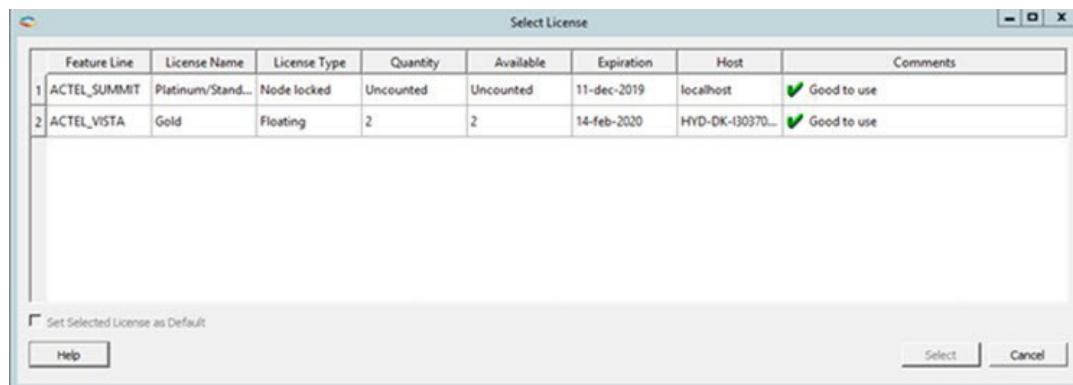
For more information about licensing, including links to troubleshooting and FAQ documents, see the [Microchip Libero SoC License Information Web Page](#).

2.2. Selecting a Default License from a License List [\(Ask a Question\)](#)

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 Microchip 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, Floating license, or Server-based license. Floating and Server-based Licenses can be used by multiple users, depending on the number of seats available.

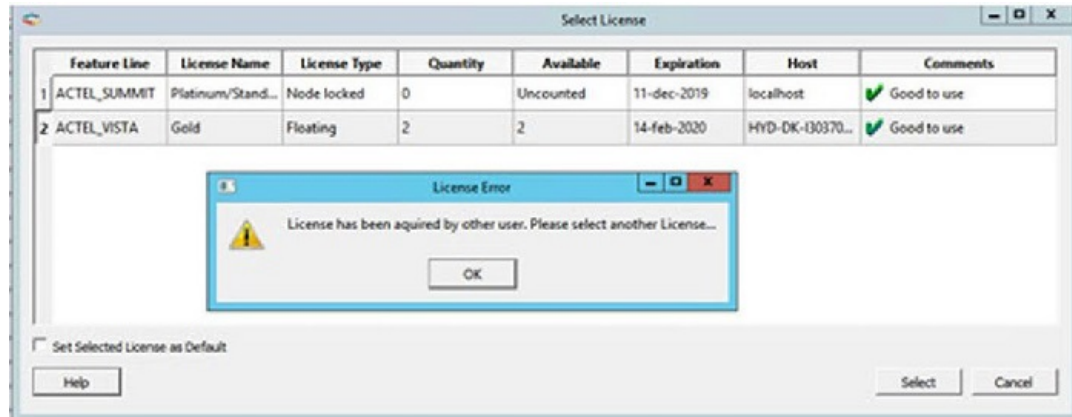
Figure 2-2. Select License Dialog Box



Selecting a License to Use

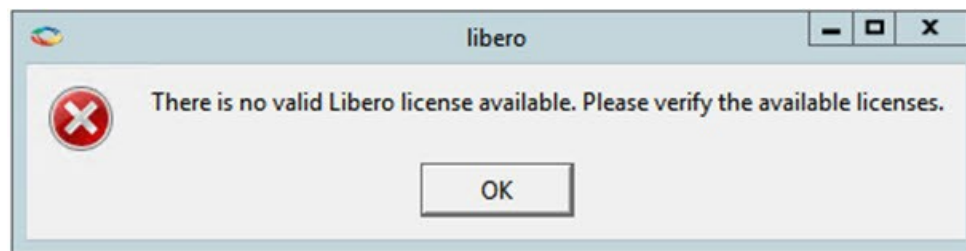
1. In the Select License dialog box, click **Select** to activate Libero using the selected license. This button is disabled by default and is enabled after you select a license.
2. Check the **Set Selected License as Default** check box to save the selected license as the default license to be used 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:

Figure 2-3. Message when a License was Acquired by Another User



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

Figure 2-4. Message when No Valid Licenses are Available



3. 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 2-5. Select License Dialog Box with License Expiration Warning Message

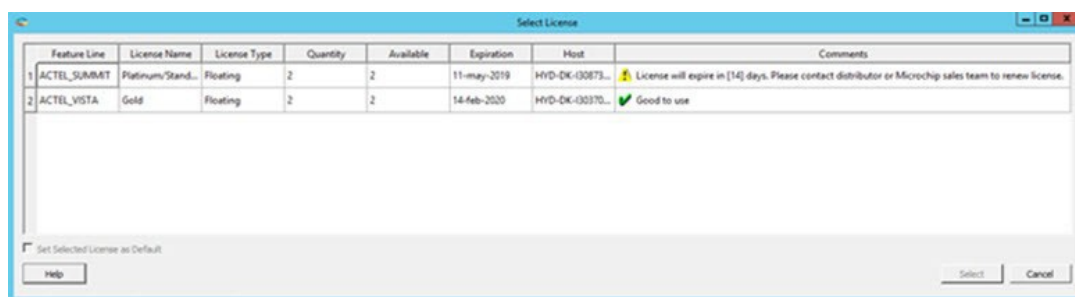
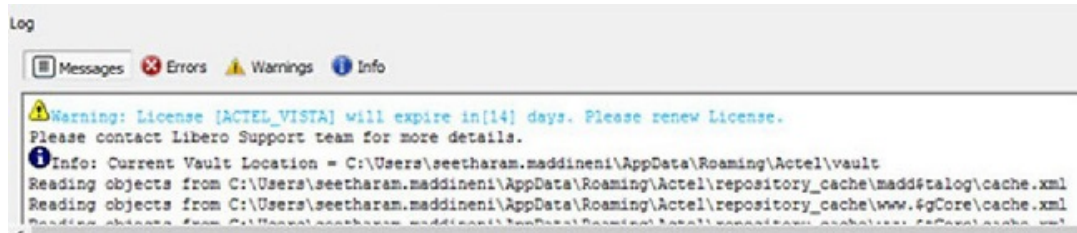


Figure 2-6. Log Window with License Expiration Warning Message

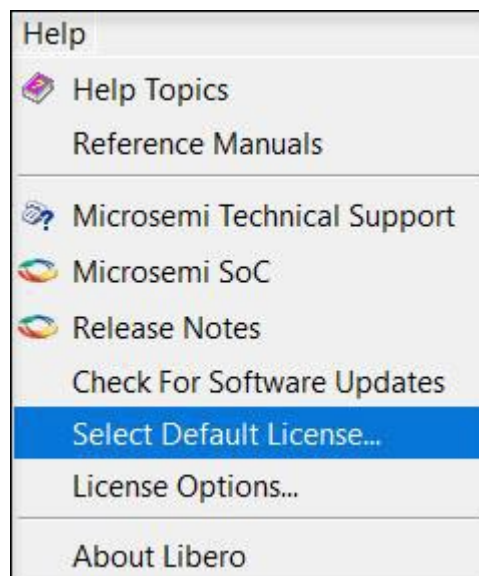


2.3. Setting a Default License [\(Ask a Question\)](#)

After you start Libero, use the following procedure to select the default Libero license that will be used for future sessions.

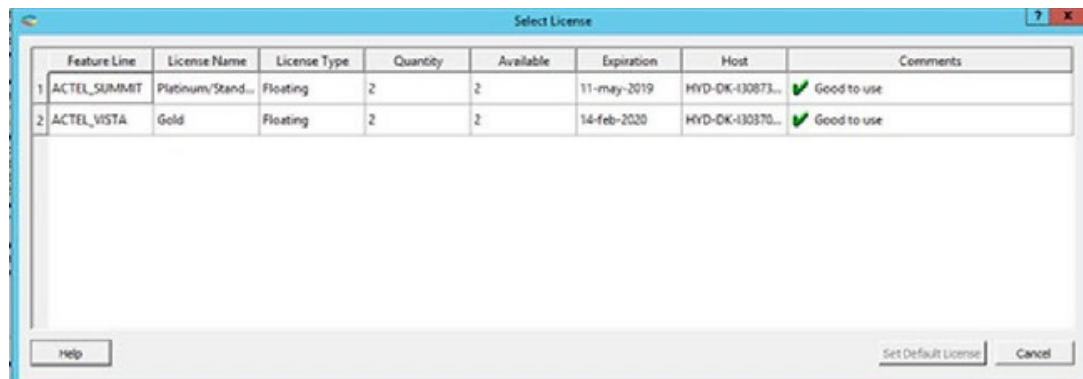
1. Click **Help > Select Default License** to change the default license.

Figure 2-7. Selecting Help > Select Default License



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

Figure 2-8. Select License Dialog Box



3. Close the dialog box.

- After you select a default Libero license, click **Project > Preferences** to set license options in the Libero Preferences dialog box. The following table describes the options.

Figure 2-9. License Options in Libero Preferences Dialog Box

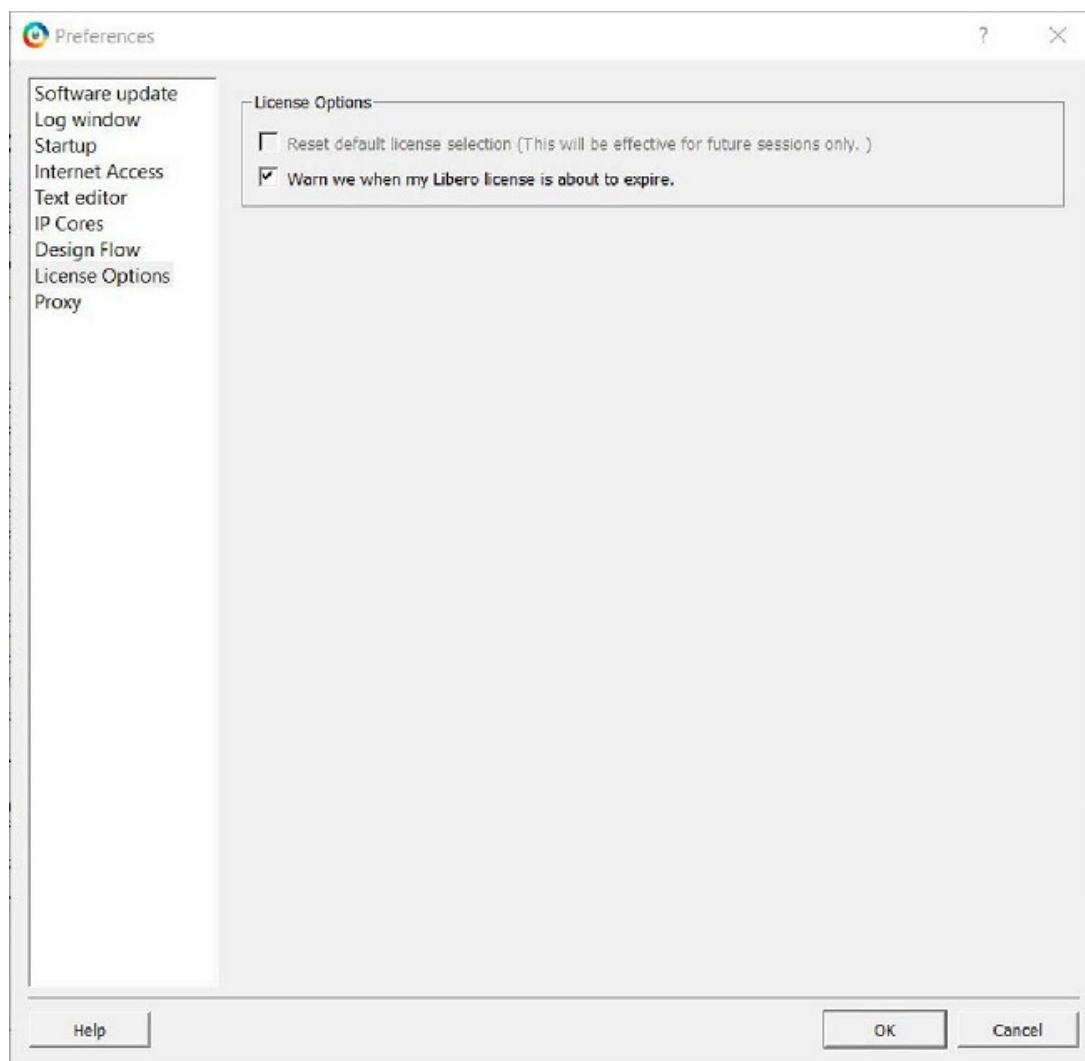


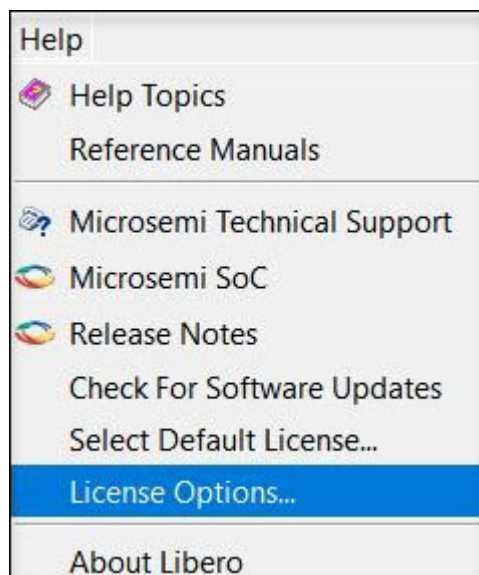
Table 2-2. Preferences Dialog Box Options

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 me when my Libero license is about to expire	Enables or disables Notification of License expiry. When checked, a message appears 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 [\(Ask a Question\)](#)

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

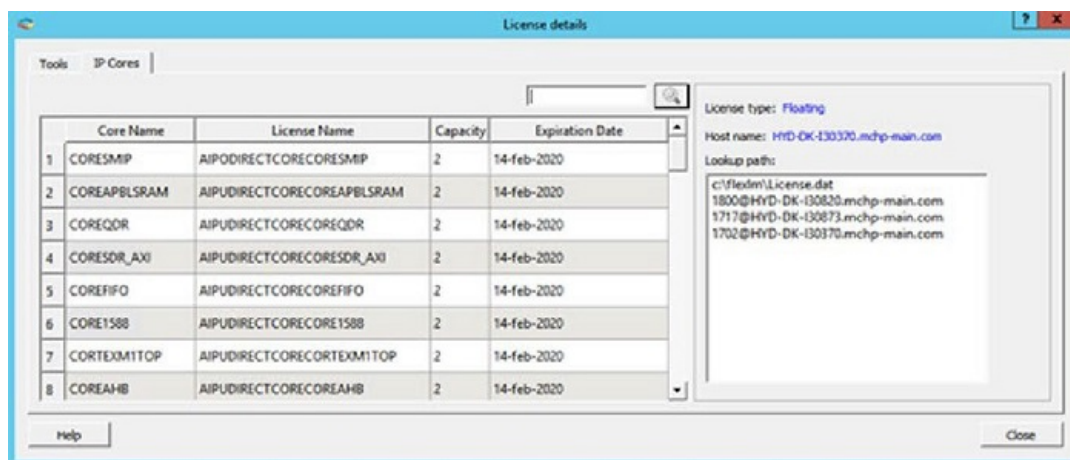
Figure 2-10. Selecting Help > License Options



The License Details dialog box has the following two tabs:

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

Figure 2-11. 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 [\(Ask a Question\)](#)

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:

- 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.
- 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 [\(Ask a Question\)](#)

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 > Microchip > 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 might 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 the software.

3. Getting Started [\(Ask a Question\)](#)

The following sections describe how to start using Libero SoC.

3.1. Starting Libero SoC [\(Ask a Question\)](#)

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](#). Use this wizard to create new Libero SoC projects.
- Clicking **Open** opens an [existing Libero SoC project](#).

3.2. Design Report [\(Ask a Question\)](#)

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 start **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 in the Design Report Tab

Category	Report
Project Summary	• Synthesize • Place and Route • Verify Timing • Verify Power
Programming	• Generate FPGA Array Data • Generate Bitstream
Export	• Export Pin Report • Export BSDL File

3.3. Creating a New Project [\(Ask a Question\)](#)

To simplify project creation, Libero SoC provides a wizard that takes you 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
Project Details	Specify the name and location of your project, device family and parts, I/O standards, and HDL source files and design constraint files.
Device Selection	Select a device for your project. After you select a device, or in any wizard screen that follows, you can click the Finish button to create the project and exit the wizard.
SmartFusion 2 and IGLOO 2: Device Settings	Specify the device I/O technology and reserve pins for probes.

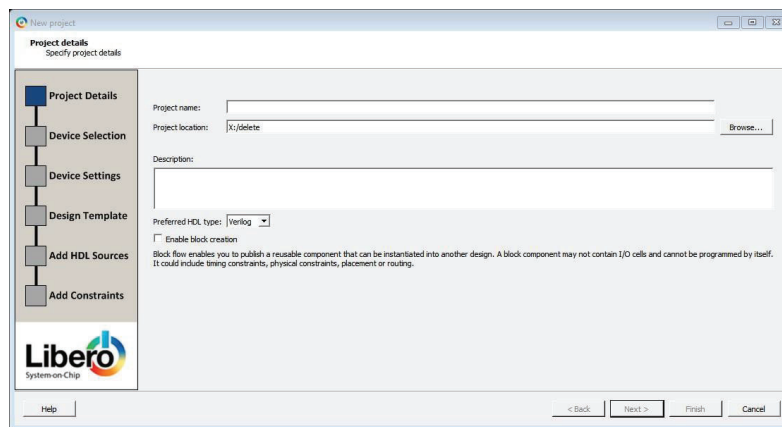
Table 3-2. Screens in the New Project Wizard (continued)

Screen	Description
Design Template	This dialog box might not be available if there are no design templates for the chosen technology.
Add HDL Sources	Add HDL design source files to your Libero SoC project.
Add Constraints	Add timing and physical constraints files to your Libero SoC project.

3.3.1. New Project Creation Wizard: Project Details [\(Ask a Question\)](#)

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
Project Name	Identifies your project name. Do not use spaces or reserved Verilog or VHDL keywords.
Project Location	Identifies your project location on disk.
Description	General information about your design and project.
Preferred HDL Type	Sets your HDL type to one of the following: - Verilog - VHDL Libero-generated files (SmartDesigns, SmartGen cores, and so on) are created in the HDL type you specify. Libero SoC supports mixed HDL designs.
Enable Block Creation	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.

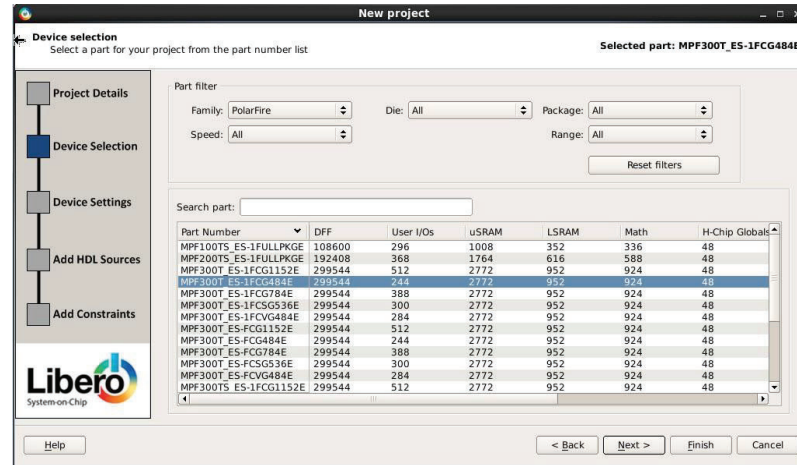
3.3.2. New Project Creation Wizard: Device Selection [\(Ask a Question\)](#)

The Device Selection screen is where you can 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 create the new project by accepting all of the remaining default settings.

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

Field	Description
Family	Microchip device family. Only devices that belong to the family appear in the parts table.
Die/Package/Speed	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 Libero SoC license you have (Evaluation, Silver, Gold, or Platinum). For more information, see the Libero SoC Licensing web page.
Core Voltage	Core voltage for your device. Two numbers separated by a “~” are shown if a wide range voltage is supported. For example, 1.2~1.5 means that the device core voltage can vary between 1.2 and 1.5 volts.
Range (PolarFire)	Voltage and temperature range 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. Choices are: • All: All ranges • EXT: Extended • IND: Industrial • MIL: Military

Table 3-4. Fields in the Libero SoC New Project Creation Wizard—Device Selection (continued)

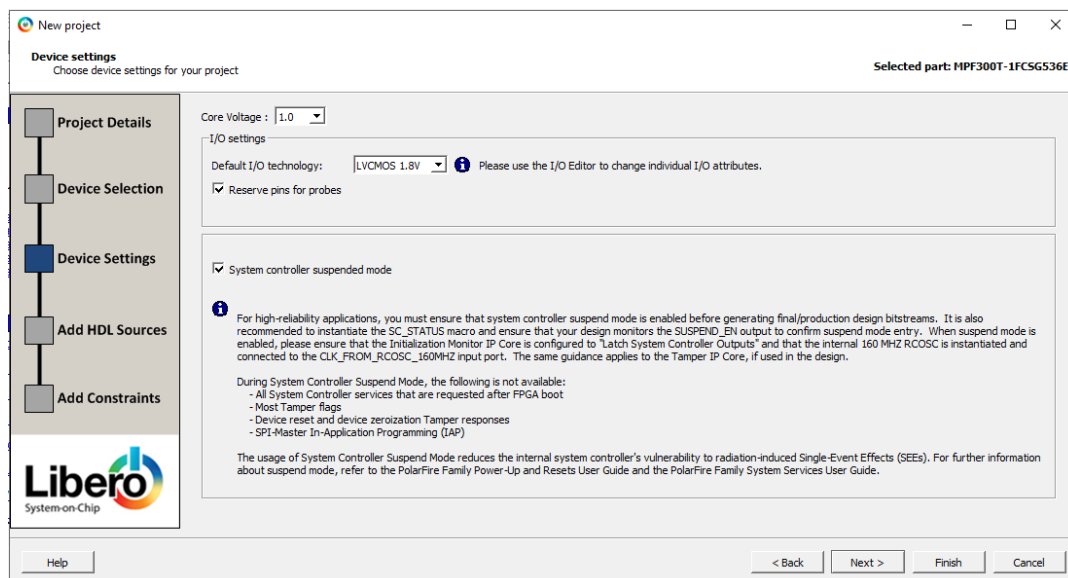
Field	Description
Range (SmartFusion 2, IGLOO 2, and RTG4)	Temperature ranges a device can encounter in your application. Junction temperature is a function of ambient temperature, air flow, and power consumption. Tools such as SmartTime, SmartPower, timing-driven layout, power-driven layout, the timing report, and back-annotated simulation are affected by operating conditions. Choices are: • ALL: All ranges • EXT: Extended • COM: Commercial (not available for RTG4 devices) • IND: Industrial • TGrade1: Automotive (not available for RTG4 devices) • TGrade2: Automotive (not available for RTG4 devices) • MIL: Military Supported operating condition ranges vary according to your device and package. Refer to the device datasheet to find your recommended temperature range. The temperature range corresponding to the value selected from the pick list can also be found by checking Project Settings > Analysis operating conditions.
Reset Filters	Resets all filters to the default ALL option except Family.
Search Parts	Character-by-character search for parts. Search results appear in the parts table.

3.3.3. New Project Creation Wizard: Device Settings [\(Ask a Question\)](#)

Device settings vary by device family.

3.3.3.1. PolarFire Device Settings [\(Ask a Question\)](#)

For PolarFire, the Device Settings page is where you can set the core voltage, default I/O technology, and enable reserve pins for probes.

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

The following table describes the fields in the Device Settings screen. After you complete the fields, click **Next** to go to the next screen or click **Finish** to create the new project by accepting all of the remaining default settings.

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

Field	Description
Core Voltage	Set the core voltage for your device.
Default I/O technology	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.
Reserve pins for probes	Reserve your pins for probing if you intend to debug using SmartDebug. If unchecked, the I/Os can be used as General Purpose I/Os.
System controller suspend mode	Suspends operation of the system controller. Checking this box places the system controller in a reset state when the device is powered up. This suspends all system services from being performed. For a list of system services for PolarFire, see the <i>System Controller User's Guide</i> for your device.

3.3.3.2. SmartFusion 2 and IGLOO 2 Device Settings [\(Ask a Question\)](#)

For SmartFusion 2 and IGLOO 2, the Device Settings page is where you can set the device I/O technology, enable reserve pins for probes, set power supplies, and enable system controller suspend mode (SCSM).

Figure 3-4. New Project Creation Wizard – Device Settings Page (SmartFusion 2 and IGLOO 2)

The following table describes the fields in the Device Settings screen for SmartFusion 2 and IGLOO 2. After you complete the fields, click **Next** to go to the next screen or click **Finish** to create the new project by accepting all of the remaining default settings.

Table 3-6. Fields in the Libero SoC New Project Creation Wizard—Device Settings (SmartFusion 2 and IGLOO 2)

Field	Description
Default I/O technology	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.

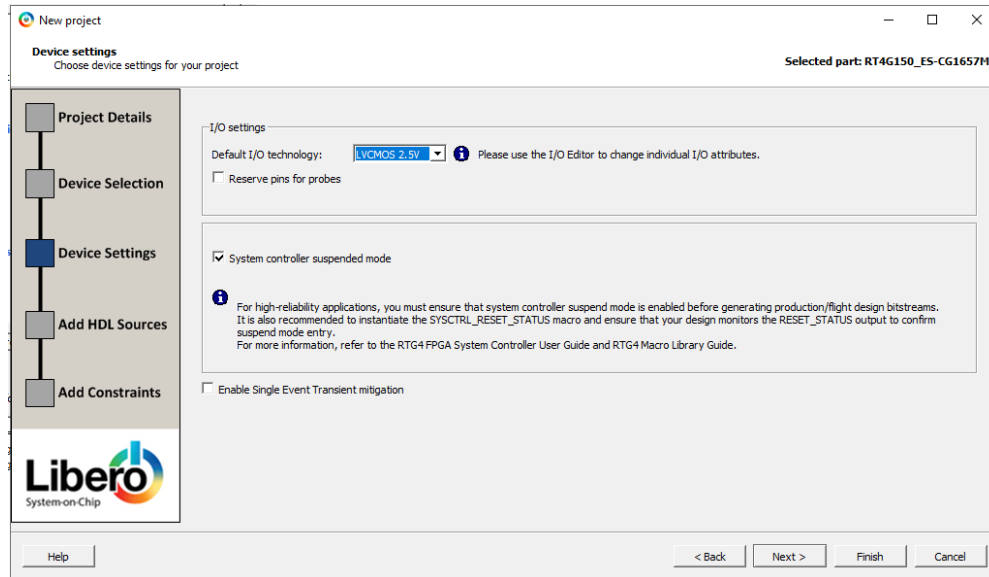
Table 3-6. Fields in the Libero SoC New Project Creation Wizard—Device Settings (SmartFusion 2 and IGLOO 2) (continued)

Field	Description
Reserve pins for probes	Reserve your pins for probing if you intend to debug using SmartDebug. If unchecked, the I/Os can be used as General Purpose I/Os.
PLL supply voltage (V)	Set the voltage for the power supply that you plan to connect to all the PLLs in your design, such as MDDR, FDDR, SERDES, and FCCC.
VDD Supply Ramp Time	Power-up management circuitry is designed into every SmartFusion 2 and IGLOO 2 FPGA. These circuits ensure easy transition from the powered-off state to the powered-up state of the device. The SmartFusion 2, IGLOO 2, and RTG4 system controller is responsible for systematic power-on reset whenever the device is powered on or reset. All I/Os are held in a high-impedance state by the system controller until all power supplies are at their required levels and the system controller has completed the reset sequence. The power-on reset circuitry in SmartFusion 2 and IGLOO 2 devices requires the V_{DD} and V_{PP} supplies to ramp monotonically from 0 V to the minimum recommended operating voltage within a predefined time. There is no sequencing requirement on VDD and VPP. Four ramp rate options are available during design generation: 50 μs, 1 ms, 10 ms, and 100 ms. Each selection represents the maximum ramp rate to apply to VDD and VPP. Device information (such as Die, Package, and Speed) can be modified later in the Project Settings dialog box.
System controller suspend mode	Suspends operation of the system controller. Checking this box places the system controller in a reset state when the device is powered up. This suspends all system services from being performed. For a list of system services for SmartFusion 2 and IGLOO 2, see the <i>System Controller User's Guide</i> for your device.

3.3.3.3. RTG4 Device Settings ([Ask a Question](#))

For RTG4 devices, the Device Settings page allows you to configure the default I/O technology, reserve pins for probes, enable single-event transient mitigation, and activate SCSM.

Figure 3-5. New Project Creation Wizard – Device Settings Page (RTG4)



The following table describes the fields in the Device Settings screen for RTG4. After you complete the fields, click **Next** to go to the next screen or click **Finish** to create the new project by accepting all of the remaining default settings.

Table 3-7. Fields in the Libero SoC New Project Creation Wizard - Device Settings (RTG4)

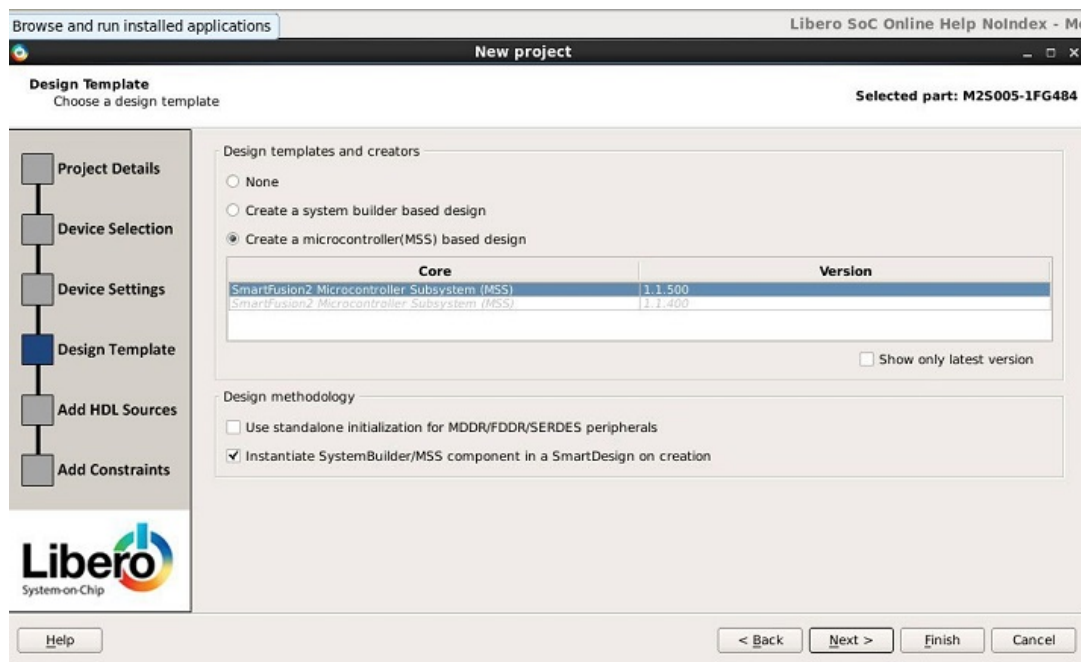
Field	Description
Default I/O technology	Set all your I/Os to a default value. You can change the values for individual I/Os in the I/O Attribute Editor. The I/O Technology available is family-dependent.
System controller suspend mode	Enables or disables SCSM. Selecting this option: - Disables the JTAG interface, Fabric Erase/Write, and Fabric Verify when TRSTB is low. To enable them, drive TRSTB high during device power-up or DEVRST_N deassertion. - Disables Probe Read/Write when TRSTB is low during device power-up. To enable Probe Read/Write, drive TRSTB high during device power-up or DEVRST_N deassertion. - Disables all Fabric read operations, such as verify digest, read digest, or when reading design or programming information in DEVICE_INFO when TRSTB is low during device power-up. To enable Digest Check, drive TRSTB high during device power-up or DEVRST_N deassertion.
Reserve pins for probes	Reserve your pins for probing if you intend to debug using SmartDebug. If unchecked, the I/Os can be used as General Purpose I/Os.
Enable Single Event Transient mitigation	Controls the mitigation of Single Event Transient (SET) in the FPGA fabric. When this box is checked, SET filters are turned on globally to help mitigate radiation-induced transients. By default, this box is not checked.

3.3.4. New Project Creation Wizard: Design Template (SmartFusion 2 and IGLOO 2) [\(Ask a Question\)](#)

The Design Template page is where you can use Libero SoC's built-in template to automate your SmartFusion 2 or IGLOO 2 design process. The template uses the System Builder tool for system-

level design or the Microcontroller Subsystem (MSS) in your design. Both will speed up the design process.

Figure 3-6. New Project Creation Wizard – Design Template Page



The following table describes the fields in the Device Template screen. After you complete the fields, click **Next** to go to the Add HDL Sources page or click **Finish** to create the new project by accepting all of the remaining default settings.

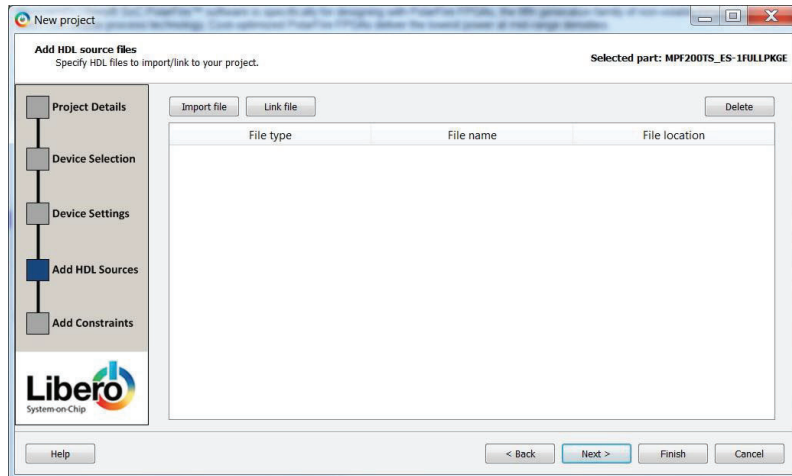
Table 3-8. Fields in the Libero SoC New Project Creation Wizard - Device Template

Field	Description
None	Select if you do not want to use a design template.
Create a System Builder based design	Use System Builder to generate your top-level design.
Create a Microcontroller (MSS) based design	Instantiate a Microcontroller (MSS) in your design. The version of the MSS cores available in your vault is displayed. Select the version you desire.
Use Standalone Initialization for MDDR/FDDR/SERDES Peripherals	Check if you want to create your own peripheral initialization logic in SmartDesign for each design peripheral (MDDR/ FDDR/SERDES). When checked, System Builder does not build the peripherals initialization logic for you. Stand-alone initialization is useful if you want to make the initialization logic of each peripheral separate from and independent of each other.
Instantiate System Builder/MSS component in a SmartDesign on creation	Uncheck if you are using this project to create System Builder or MSS components and do not plan on using them in a SmartDesign based design. This is especially useful for design flows where the System Builder or MSS components are stitched in a design using HDL.

3.3.5. New Project Creation Wizard: Add HDL Source Files [\(Ask a Question\)](#)

The Add HDL Source Files screen is where you can 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 3-7. Libero SoC New Project Creation Wizard - Add HDL Source Files

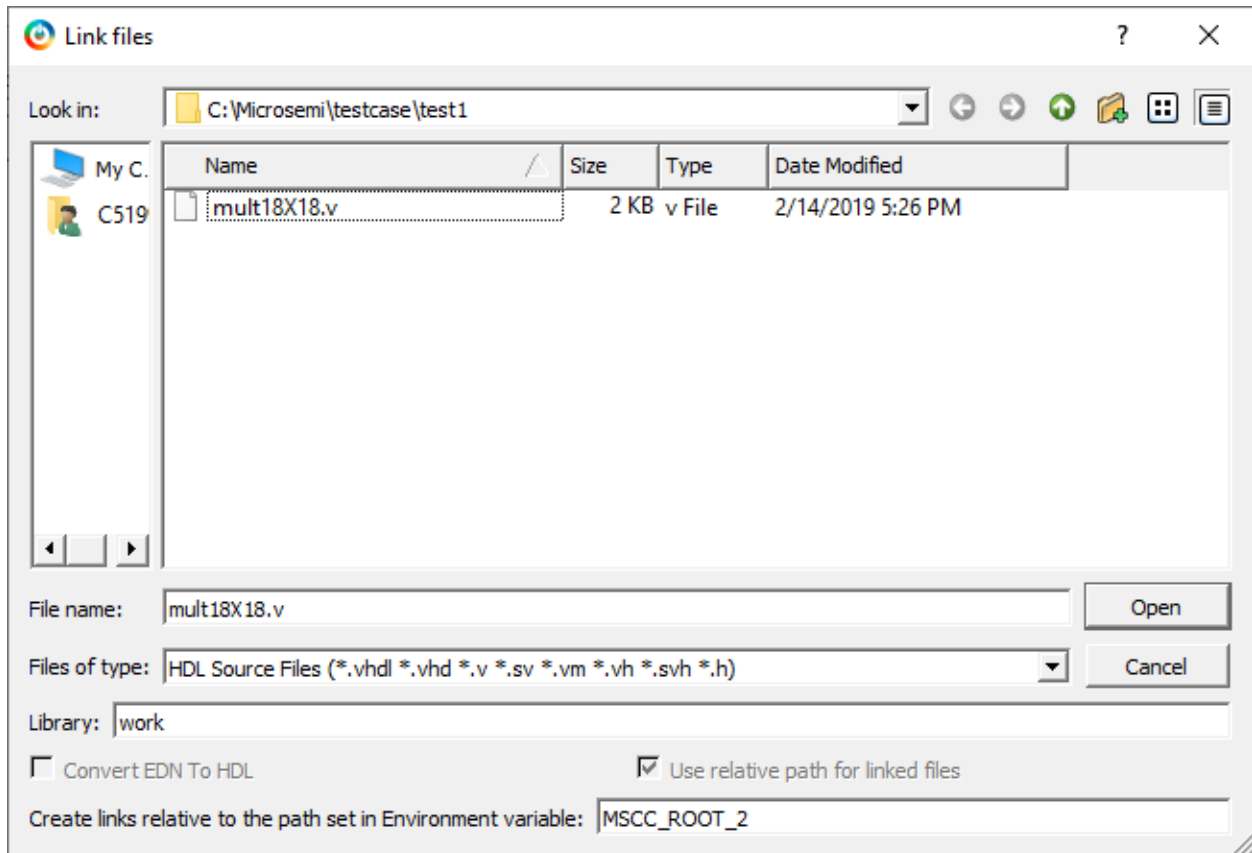


The following table describes the elements in the Add HDL Source Files screen. After you complete the fields, click **Next** to go to the [Add Constraints](#) screen or click **Finish** to create the new project by accepting all of the remaining default settings.

Table 3-9. Elements in the Libero SoC New Project Creation Wizard - Add HDL Source Files

Element	Description
Import File 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 Open . The HDL file is copied to the <prj_folder>/hdl folder in your Libero project.
Link File button	Allows you to continue with an absolute or relative path for linked files. When the Link files dialog box appears (see the following figure), go to the location where the HDL source resides, select the HDL file, and click Open .
Create links relative to the path set in Environment variable	Available when you click the Link File button. 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. Note: 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.
Delete 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.

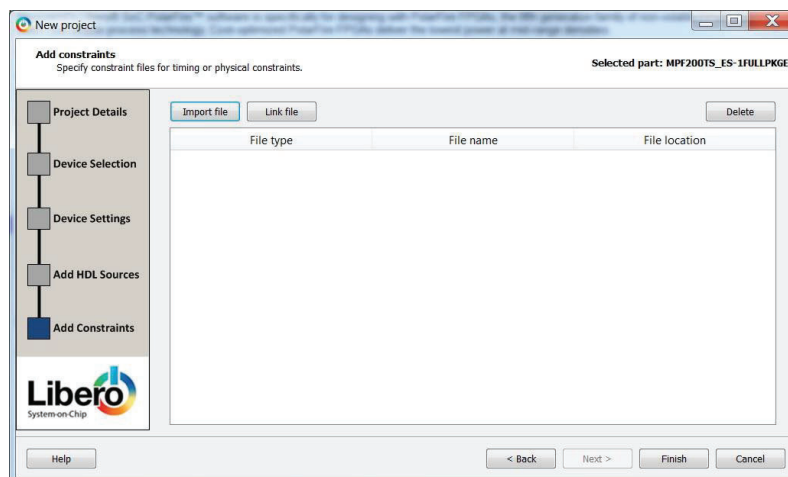
Figure 3-8. New Project Creation Wizard – Link Files Dialog Box



3.3.6. New Project Creation Wizard: Add Constraints [\(Ask a Question\)](#)

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 3-9. Libero SoC New Project Creation Wizard – Add Constraints

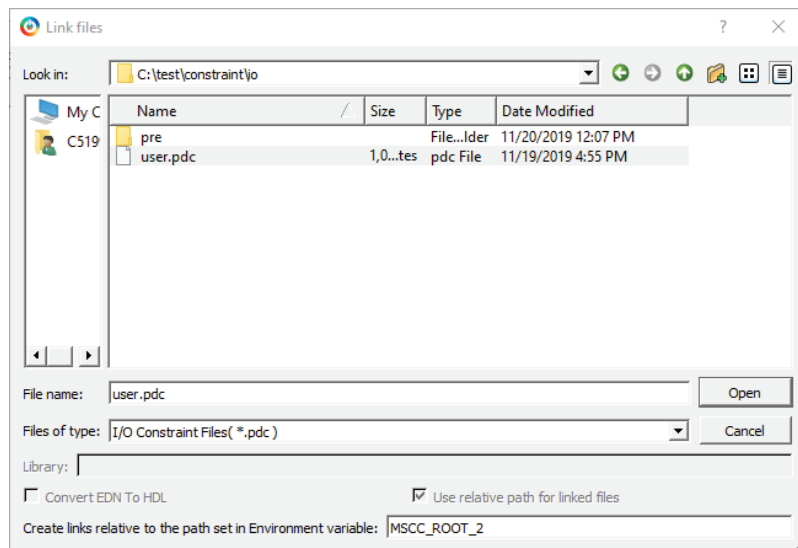


The following table describes the elements in the Add Constraints screen.

Table 3-10. Elements in the Libero SoC New Project Creation Wizard - Add Constraints

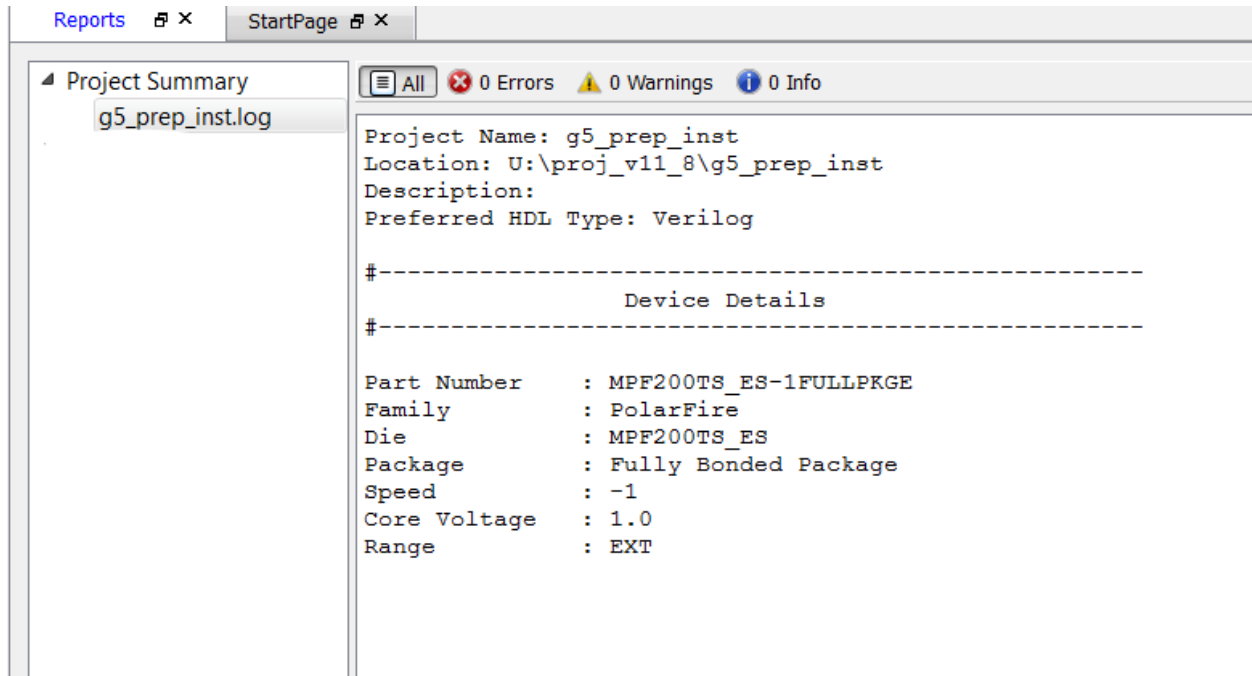
Element	Description
Import File button	Go to the location where the constraints file resides. Select the constraints file and click Open . The constraints file is copied to the <prj_folder>/constraint folder in your Libero project.
Link File button	Click this button if the constraint file is located and maintained outside the Libero project. When the Link files dialog box appears (see the following figure), 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 Open . The constraints file is linked to the Libero project.
Create links relative to the path set in Environment variable	Available when you click the Link File button. 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. Note: 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.
Delete 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 3-10. 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 3-11. Reports Tab



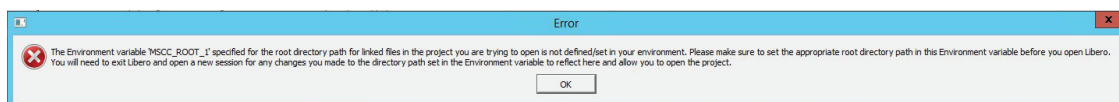
3.4. Opening a Project [\(Ask a Question\)](#)

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 3-12. Error Message

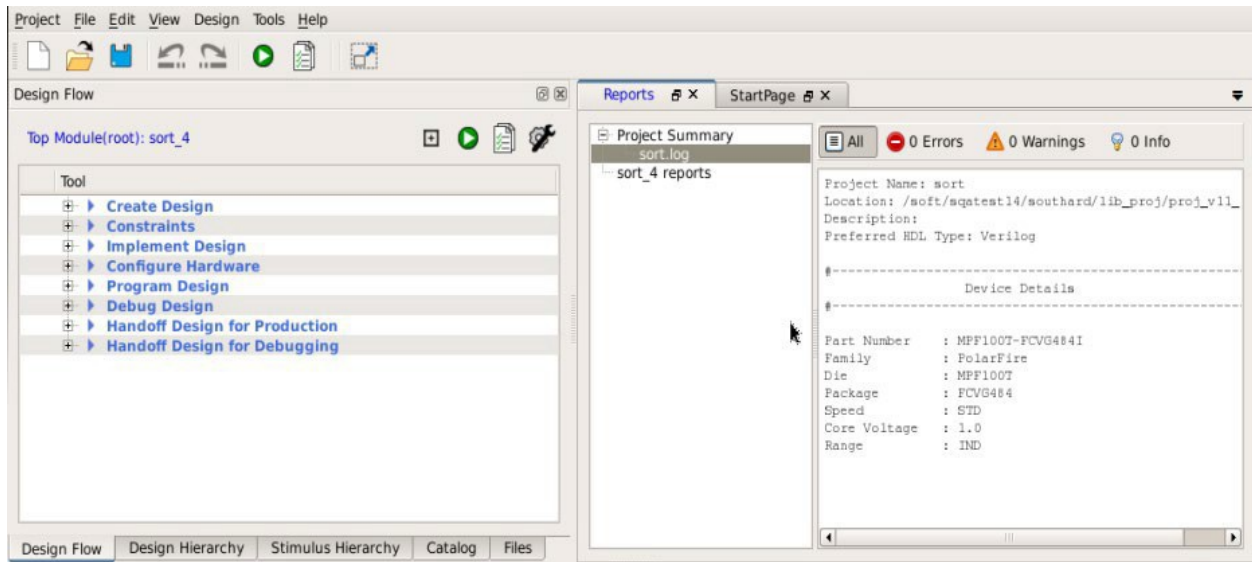


When you open an existing Libero SoC project:

- 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 3-13. Sample Design Flow Window



The Design Flow window might appear 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 [\(Ask a Question\)](#)

Create your design with the following design capture tools:

- [Create SmartDesign](#)
- [System Builder](#) (SmartFusion 2 and IGLOO 2 only)
- [Create HDL](#)
- [Create SmartDesign Testbench](#) (optional, for simulation only)
- [Create HDL Testbench](#) (optional, for simulation only)
- [Verify Pre-synthesized Design](#)

4.1. SmartFusion 2 and IGLOO 2 Tools [\(Ask a Question\)](#)

The following topics describe System Builder and how to use the MSS in your SmartFusion 2 designs.

4.1.1. System Builder [\(Ask a Question\)](#)

System Builder is a graphical design wizard that allows you to enter high-level design specifications for SmartFusion 2 or IGLOO 2.

System Builder takes you through the following steps:

- Asks basic questions about your system architecture and peripherals
- Builds a correct-by-design complete system

To start System Builder:

1. In the Design Flow window, click **System Builder > Run**.
2. In the Enter a name for your system dialog box, enter a name for the system you want to build.
3. Click **OK**. The System Builder Device Features page appears.

System Builder automatically configures the silicon features you select. To complete the design, add your custom logic or IP and connect them to your System Builder-generated design.

For a complete family-specific explanation of the tool, see the [SmartFusion 2 System Builder documentation](#) or the [IGLOO 2 System Builder documentation](#).

4.1.2. Using the MSS in Your SmartFusion 2 Designs [\(Ask a Question\)](#)

The following topics describe how to use the MSS in your SmartFusion 2 designs.

Note: MSS is valid for SmartFusion 2 devices only.

4.1.2.1. Instantiating a SmartFusion 2 MSS in Your Design [\(Ask a Question\)](#)

You can configure peripherals within the SmartFusion 2 MSS to suit your requirements. Examples of peripherals you can configure include:

- Arm® Cortex®-M3
- Embedded nonvolatile memory (eNVM)
- Ethernet MAC
- Timer
- UART
- SPI

The MSS operates stand-alone, without any dependencies on other logic within the device; however, designs that require functionality beyond a standalone MSS are handled by using SmartDesign to add user logic in the SmartFusion 2 SoC FPGA fabric.

You can instantiate an MSS into your design from the New Project Creation Wizard when you start a new SmartFusion 2 project, or from the Design Flow window after you have created a new project.

Instantiating a SmartFusion 2 MSS from the New Project Creation Wizard

1. Enable **Use Design Tool** (under **Design Templates and Creators**) and click to select **SmartFusion 2 Microcontroller Subsystem (MSS)** from the list.

Not Using a Design Tool when You Create a Project

If you decide not to use a Design Tool when you created your project:

1. Expand **Create Design** in the Design Flow window and double-click **Configure MSS**.
2. When the Add Microcontroller Subsystem dialog box appears, enter your design name and click **OK**. A SmartDesign Canvas appears, with the MSS added to your project. Double-click the MSS to view and configure MSS components.

4.1.2.2. Configure the SmartFusion 2 MSS [\(Ask a Question\)](#)

Documents for specific SmartFusion 2 MSS peripherals are available on the [Peripheral Documents web page](#).

The SmartFusion 2 MSS Configurator in the following figure contains the following elements. Double-click an element in the MSS to configure it. If a check box is available, check it to enable the element in your design or uncheck it to disable the element in your design.

MSS ARM Cortex-M3 Peripherals

- MSS CAN
- MSS Peripheral DMA (PDMA)
- MSS GPIO
- MSS I2C
- MSS Ethernet MAC
- MSS DDR Controller (MDDR)
- MSS MMUART
- MSS Real Time Counter (RTC)
- MSS Embedded Nonvolatile Memory (eNVM)
- MSS SPI
- MSS USB
- MSS Watchdog Timer

Fabric Interfaces

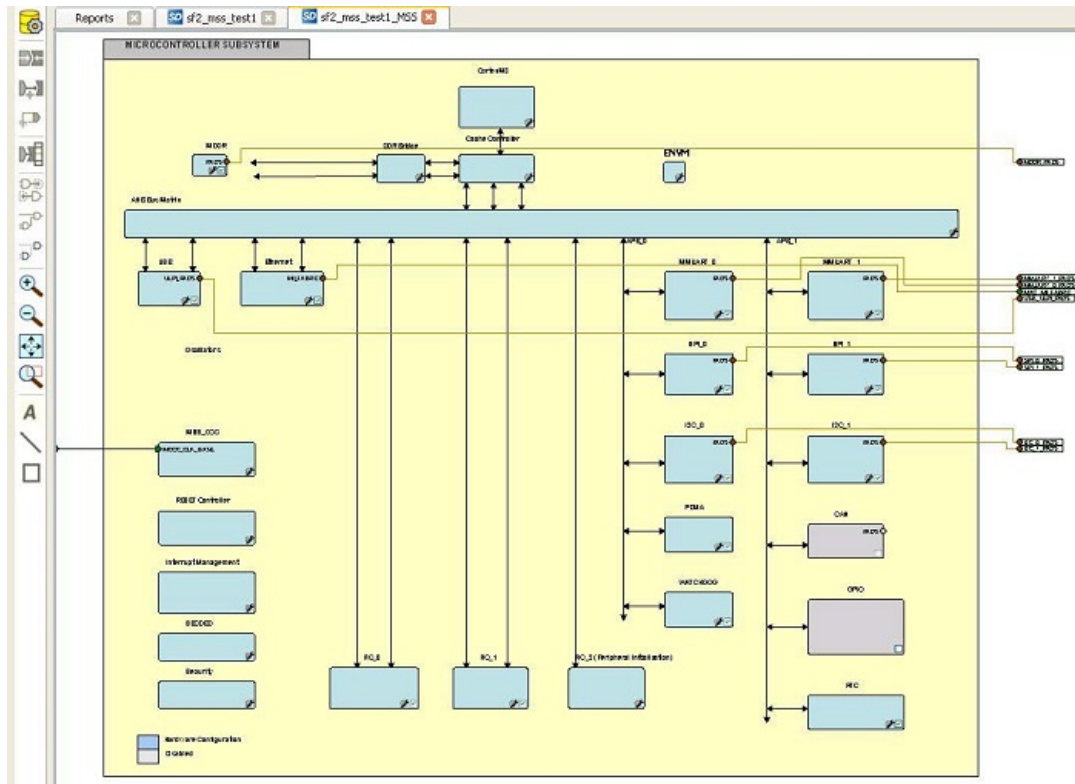
- MSS Fabric Interface Controllers (FICs)

Additional Information

- MSS Cache Controller
- MSS DDR Bridge Controller
- MSS AHB Bus Matrix
- MSS Clocks Configurator (MSS CCC)
- MSS Interrupts Controller
- MSS Reset Controller
- MSS SECEDED Configurator
- MSS Security Configurator

The MSS generates a component that is instantiated into your top-level design.

Figure 4-1. Microcontroller Subsystem Configurator



4.1.2.3. Generate SmartFusion 2 MSS Files [\(Ask a Question\)](#)

See the MSS Configurator help for more information on generating SmartFusion 2 MSS files.

Click the **Generate Component** button  to create your SmartFusion 2 MSS files. The MSS Configurator generates the following files:

- HDL files for the MSS components, including timing shells for synthesis - HDL files are automatically managed by the Libero SoC and passed to the Synthesis and Simulation tools.
- EFC File: Contains your eNVM client data - The EFC content is included in your final programming file.
- Firmware drivers and memory maps are exported into the <project>\firmware\ directory - Libero SoC automatically generates a Software IDE project that includes your Firmware drivers. If you are not using a software project automatically created by Libero, you can import this directory into your Software IDE project.
- Testbench HDL and BFM script for the MSS design: These files are managed by Libero SoC and automatically passed to the Simulation tool.
- PDC files for the MSS and the top-level design: These files are managed by Libero SoC and automatically integrated during Compile and Layout.

4.2. Create SmartDesign [\(Ask a Question\)](#)

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 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 (*.cdf 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 (*.cdx files) published from Libero SoC. For more information, see the [SmartDesign User Guide](#).

Note: The following table lists the direct cores that cannot be generated in Libero using Batch mode on Linux machines only via TCL if the DISPLAY variable is not set.

Table 4-1. Direct Cores That Cannot be Used in Batch Mode when DISPLAY Variable is not Set

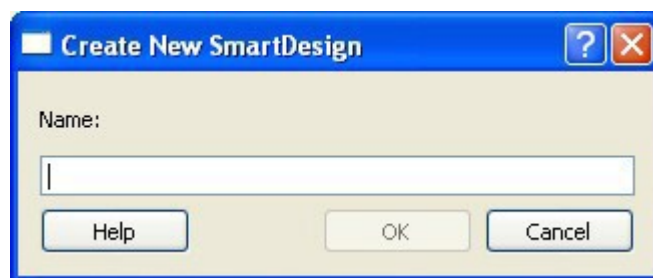
Direct Cores That Cannot be Used in Batch Mode	Direct Cores That Cannot be Used in Batch Mode
CoreAXI4SRAM	CoreFIFO
CoreCIC	CoreFFT
CoreCordic	CoreFIR_PF
CoreABC	CoreRSDEC
CoreEDAC	CoreRENC
CoreDDS	

4.2.1. Create New SmartDesign [\(Ask a Question\)](#)

This SmartDesign component might 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 4-2. 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](#).

4.2.2. Export Component Description (Tcl) [\(Ask a Question\)](#)

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 4-3. Export as TCL Option for SmartDesign Component

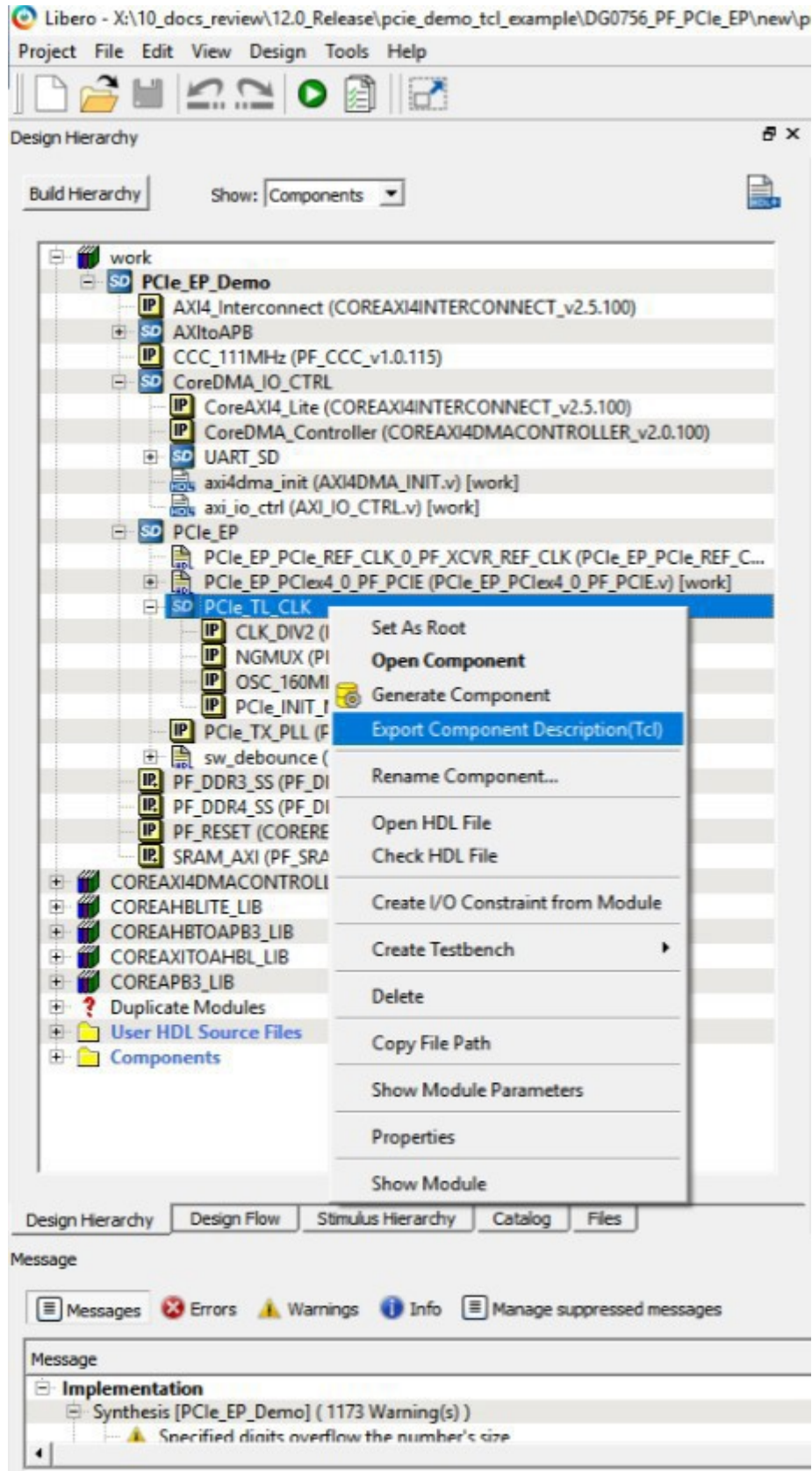


Figure 4-4. Export Script Dialog Box



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

4.2.3. Examples [\(Ask a Question\)](#)

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_DQCK: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 an 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
```

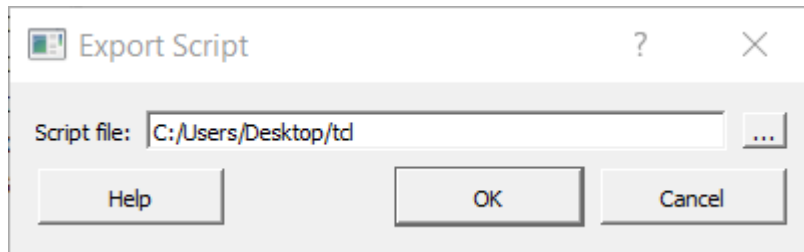
The following is an example of an exported Tcl script for a SgCore (PF_TX_PLL).

```
# 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
```

4.2.4. Hierarchical Export Component Description (Tcl) [\(Ask a Question\)](#)

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 4-5. Export Script for Hierarchical Export Description (Tcl)



This exported folder consists of the following files and subfolders:

Table 4-2. Subfolders in the Exported File

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

Table 4-3. 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 4-6. Hierarchical Export Component Description (Tcl) Option

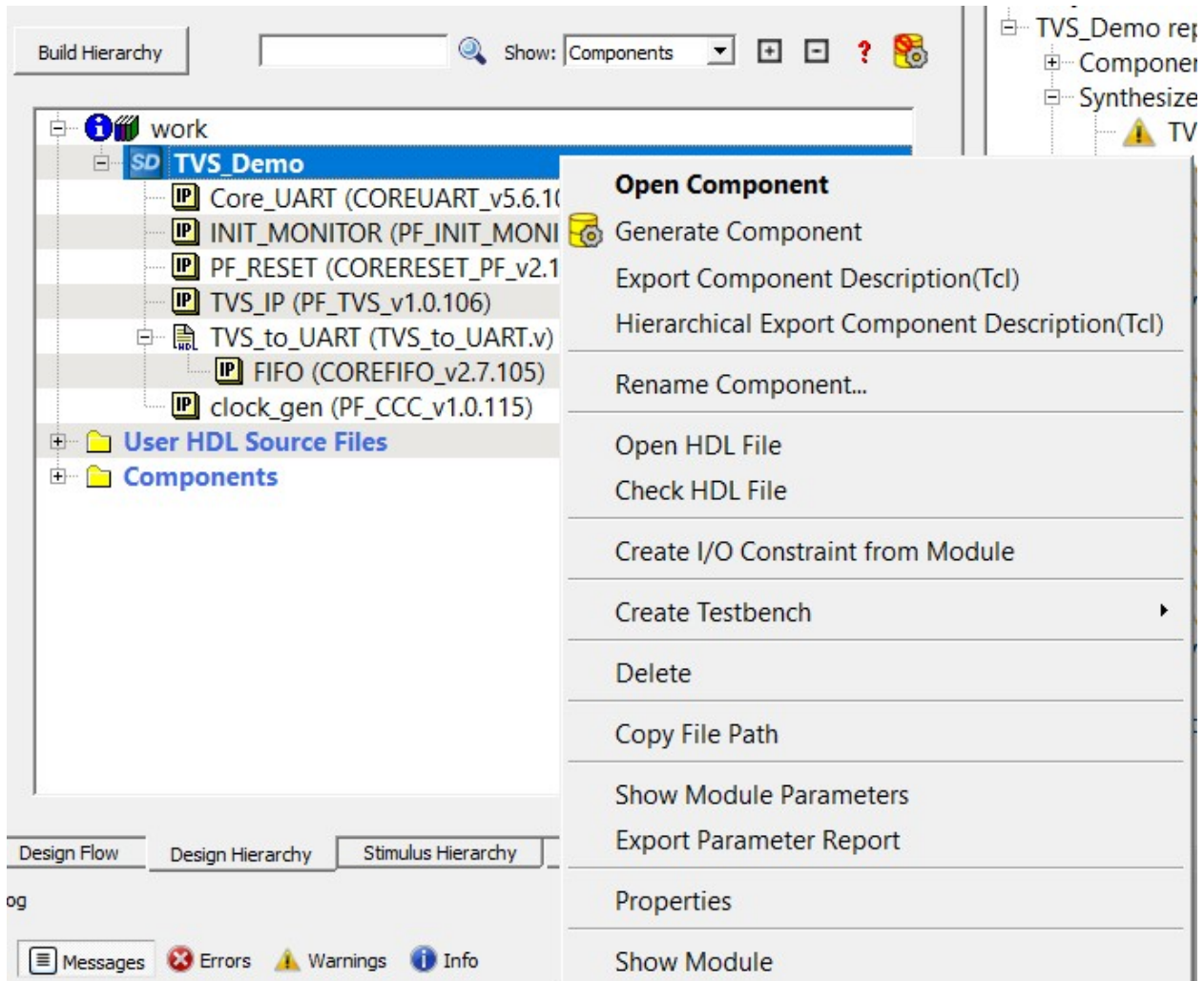
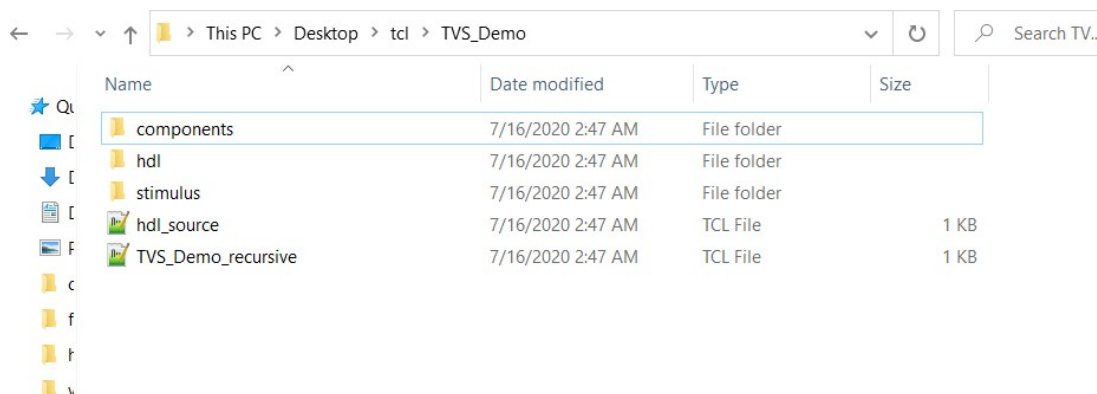


Figure 4-7. Hierarchical Export Component Description (Tcl) Option After Right-clicking a Component



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 4-4. 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.

4.2.5. Generating a SmartDesign Component [\(Ask a Question\)](#)

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



Click the **Generate** 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 might 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 4-5. Design Flow Preferences Options

Option	Description
Recursive generation	Clicking the Generate 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 Generate 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 Design Hierarchy tab of the Design Explorer.

4.3. Create Core from HDL [\(Ask a Question\)](#)

You can instantiate any HDL module and connect it to other blocks inside SmartDesign. However, there are situations where you might 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.

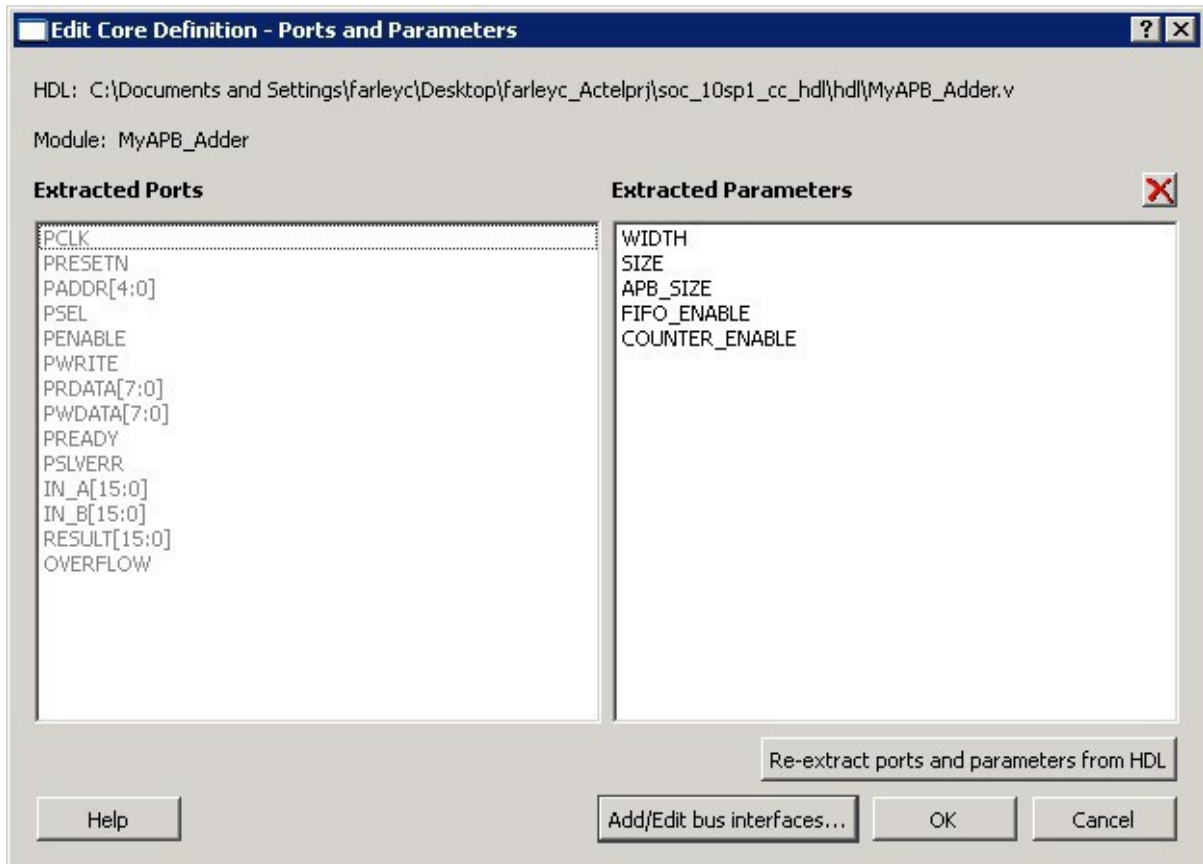
4.3.1. Creating a Core from Your HDL [\(Ask a Question\)](#)

To create a core from your HDL, perform the following steps:

1. Import or create a new HDL source file; build the Design Hierarchy by clicking **Build Hierarchy**. The HDL file appears in the Design Hierarchy.

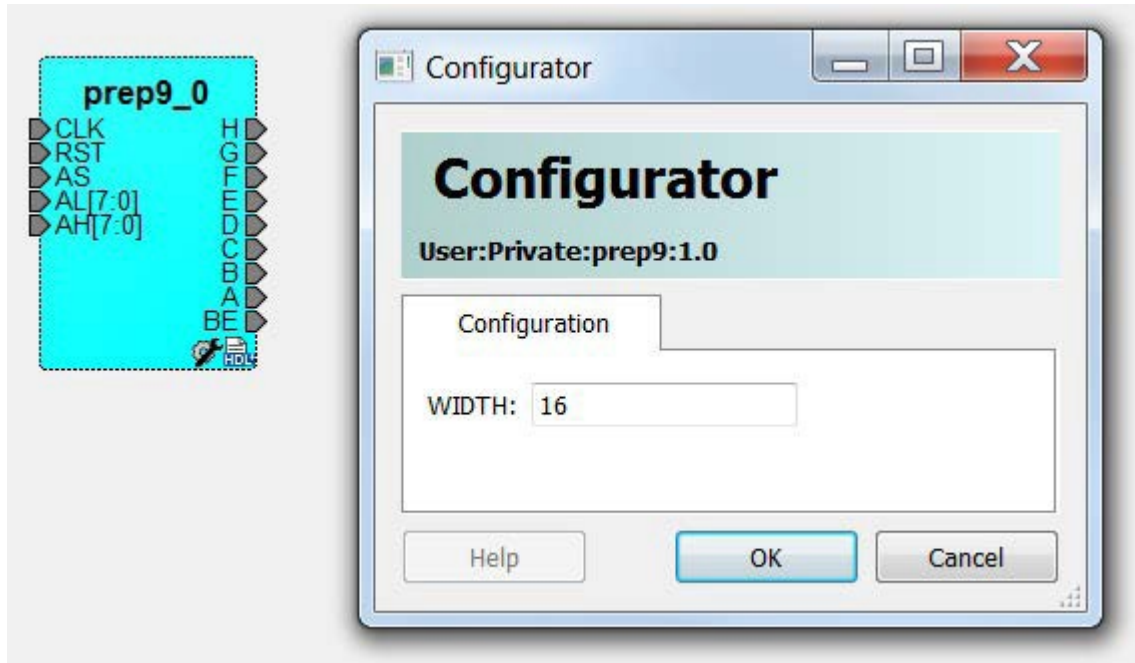
2. In the Design Hierarchy, select the HDL file and click the **HDL+** icon.
or
Right-click the HDL file and select the **Create Core from HDL** option.
The **Edit Core Definition – Ports and Parameters** dialog box shows the ports and parameters that were extracted from your HDL module.
3. Remove parameters that are not intended to configure. To remove such parameters, select them from the list and click the **X** icon. Remove parameters that are used for internal variables, such as state machine enumerations. If you remove a parameter unintentionally, 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-click the object on the Canvas (or right-click and choose **Configure**) to set the 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 select **Modify HDL**.

4.3.2. Editing Core Definitions [\(Ask a Question\)](#)

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

4.3.3. Removing Core Definitions [\(Ask a Question\)](#)

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**.

4.4. Designing with HDL [\(Ask a Question\)](#)

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

4.4.1. Create HDL [\(Ask a Question\)](#)

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.

4.4.2. Using the HDL Editor [\(Ask a Question\)](#)

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 your HDL edits.

4.4.3. HDL Syntax Checker [\(Ask a Question\)](#)

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. Warnings and error messages for the HDL file appear in the Libero SoC Log window.

4.4.4. Commenting Text [\(Ask a Question\)](#)

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**.

4.4.5. Find [\(Ask a Question\)](#)

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.

4.4.6. Editing Columns [\(Ask a Question\)](#)

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

4.4.7. Importing HDL Source Files [\(Ask a Question\)](#)

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.

4.4.8. Mixed HDL Support in Libero SoC [\(Ask a Question\)](#)

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

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

When you [create a new 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 testbench. For example, if `tb_top` is in Verilog, `<fam>.v` is compiled.

4.5. HDL Testbench [\(Ask a Question\)](#)

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.

4.5.1. HDL Type [\(Ask a Question\)](#)

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

4.5.2. Name [\(Ask a Question\)](#)

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

4.5.3. Clock Period (ns) [\(Ask a Question\)](#)

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 4-6. 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. Note: Setting one testbench as the Active Stimulus is necessary when there are multiple test benches in the stimulus hierarchy. When you select the testbench ensure that the Active Stimulus matches the DUT (from the Design Hierarchy view) that you want to stimulate.
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 4-10. Create New HDL Testbench File Dialog Box

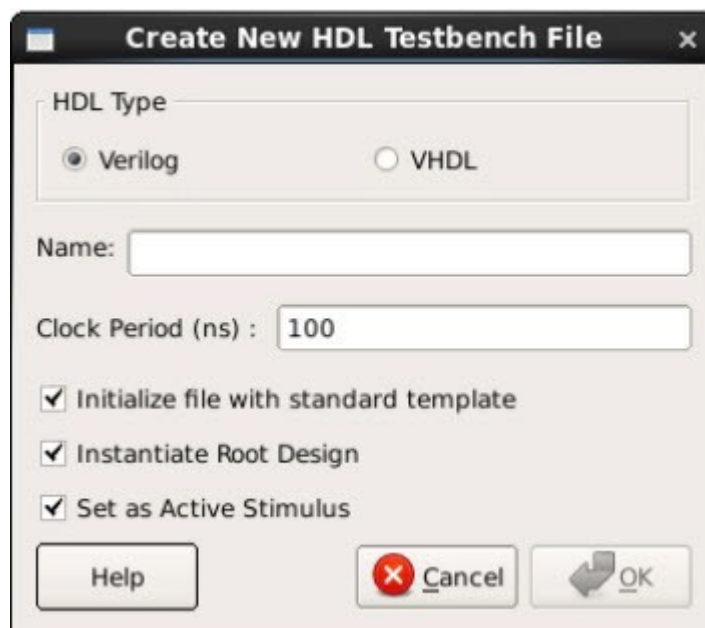


Figure 4-11. HDL Testbench Example - VHDL, Standard Template, and Root Design Enabled

```

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

```

4.6. Designing with Block Flow [\(Ask a Question\)](#)

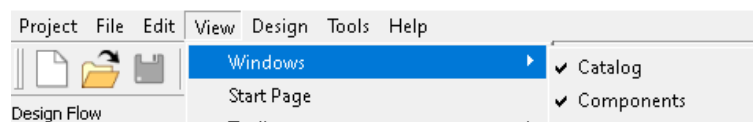
For information about designing with Block Flow, see the [PolarFire Block Flow User Guide](#).

4.7. Viewing Configured Components and SmartDesigns in a Project [\(Ask a Question\)](#)

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 4-12. 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 empty.

Figure 4-13. Example of the Components View

Components

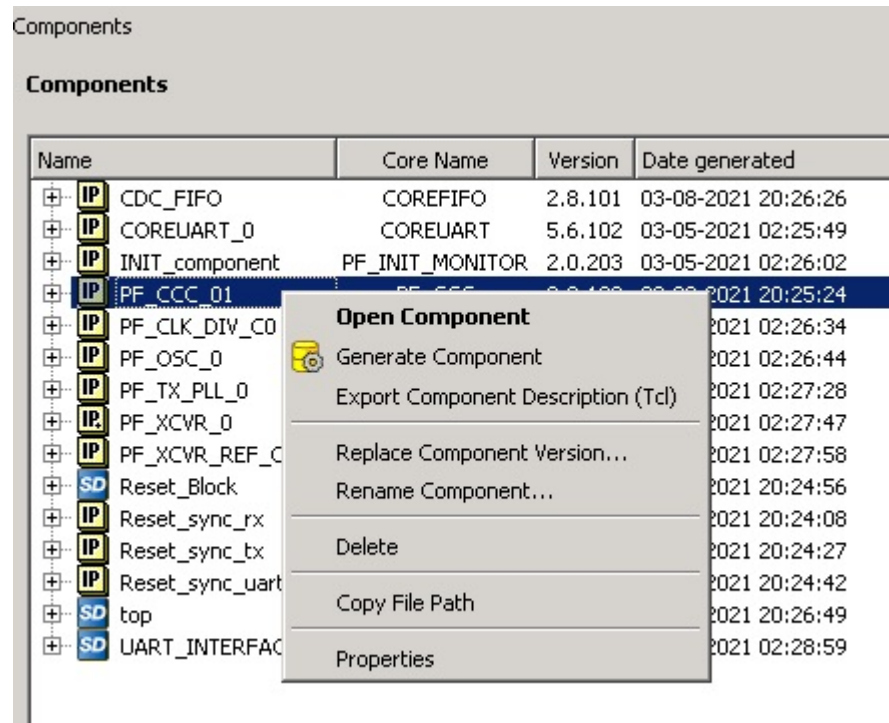
Components

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

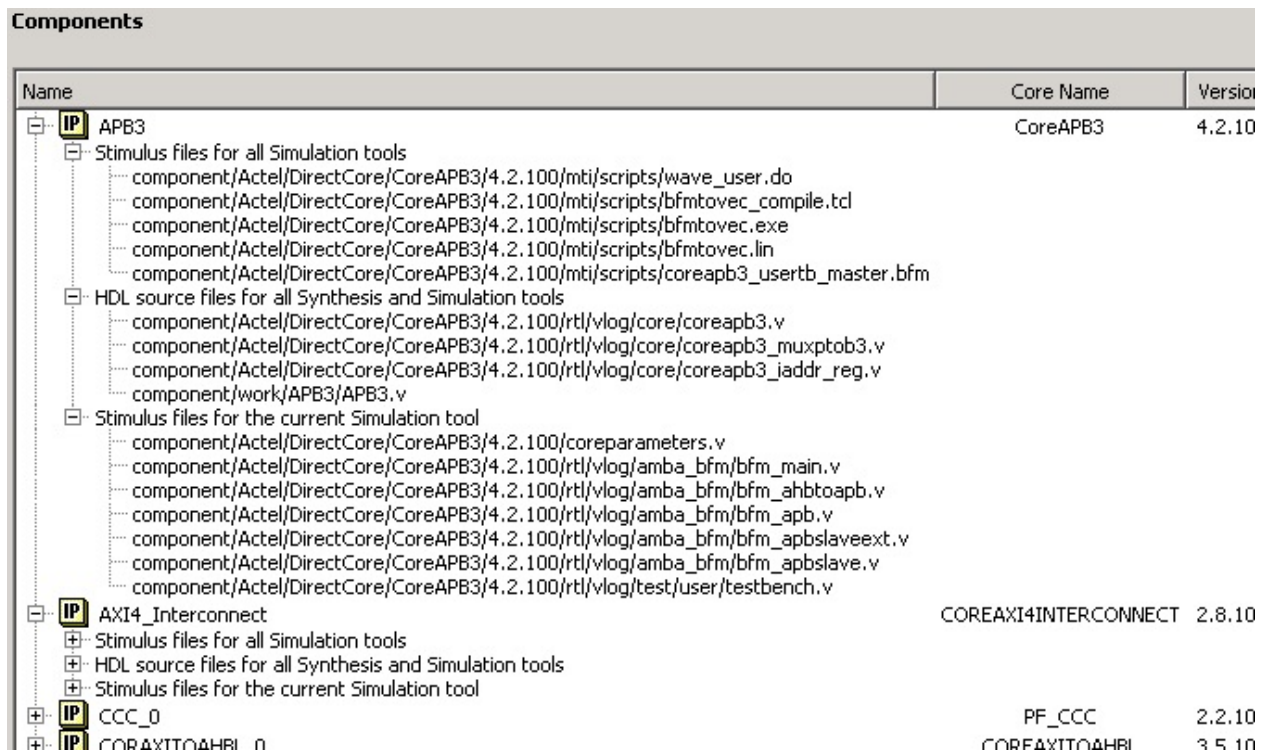
Right-clicking a component in the Components view tab displays a menu that is similar to the one that appears when you right-click entries in the Components section of the Design Hierarchy.

Figure 4-14. 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 4-15. Example of HDL Source Files



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

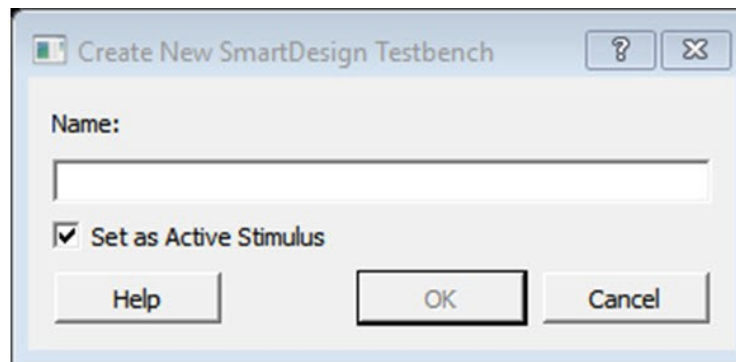
The timestamp shown for **Generation** appears in the **Date Generated** column and gets updated when the component gets regenerated.

4.8. Create a New SmartDesign Testbench [\(Ask a Question\)](#)

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

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 4-16. Create New SmartDesign Testbench



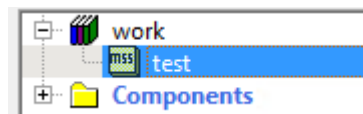
2. Enter a name.
Note: The component name must be unique in your project.
3. To make this SmartDesign Testbench your active stimulus, check the **Set as Active Stimulus** check box.
Note: Setting one testbench as the Active Stimulus is necessary when there are multiple test benches in the stimulus hierarchy. When you select the test bench, make sure the Active Stimulus matches the DUT (from the **Design Hierarchy** view) that you want to stimulate.
4. Click **OK**. The component appears in the **Stimulus Hierarchy** tab of the Design Explorer.

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

4.9. Import MSS [\(Ask a Question\)](#)

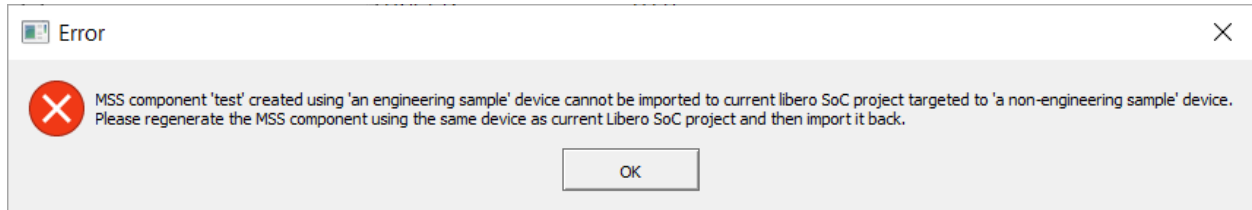
If you use a device from the PolarFire SoC family, a new tool called **Import MSS** is added under **Create Design** in the design flow process. When you run this tool, an Import MSS Component dialog box allows you to select the MSS Component (*.cxz) files. After importing the selected *.cxz module file, 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



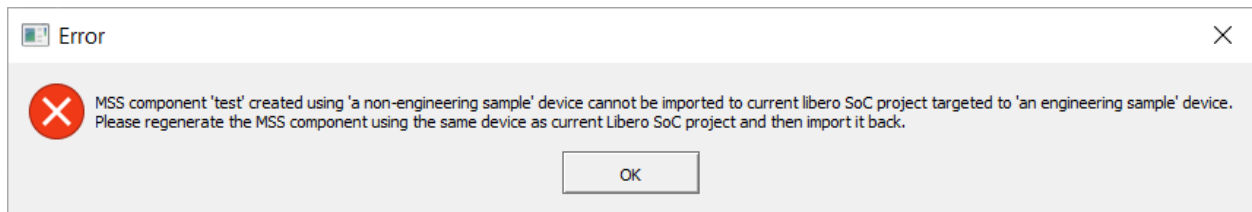
When importing an MSS block created using an ES device in a Libero project created using a non-ES device, the following error message appears:

Figure 4-18. Error Message Generated on Importing an ES Device to a Libero Project Created Using a Non-ES Device



When importing an MSS block created using an non-ES device, in a Libero project created using ES device, the following error message appears:

Figure 4-19. Error Message Generated on Importing a Non-ES Device to a Libero Project Created Using an ES Device



On right-clicking the MSS component and choosing **Open Component**, PolarFire SoC MSS Configurator opens up in read-only mode. If the MSS component is instantiated in your SmartDesign, you can also open the PolarFire SoC MSS component in the read-only mode on double-clicking the MSS Component instance in the SmartDesign Canvas. You will not be able to regenerate the design files from the PolarFire SoC MSS Configurator in the read-only mode.

You can also launch the PolarFire SoC MSS Configurator from the Libero tool as follows:

```
pfsoc_mss.exe \
-LIBERO_READ_ONLY_MODE \
-CONFIGURATION_FILE:<location_to_file>/<file_name>.cfg
```

Both the arguments are mandatory. This opens the PolarFire SoC MSS Configurator from the Libero tool in read-only mode. You must mention the path to the configuration file.

4.10. Verify Pre-Synthesized Design - RTL Simulation [\(Ask a Question\)](#)

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 QuestaSim ME Pro.

QuestaSim ME Pro is a custom edition of QuestaSim PE that is integrated into Libero SoC's design environment. QuestaSim for Microchip is an OEM edition of Siemens QuestaSim tools. QuestaSim 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 QuestaSim. To use other QuestaSim editions, do not install QuestaSim ME Pro from the Libero SoC media.

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

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

- Simulation Options and Simulation Libraries in the [Project Settings Dialog Box](#)

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

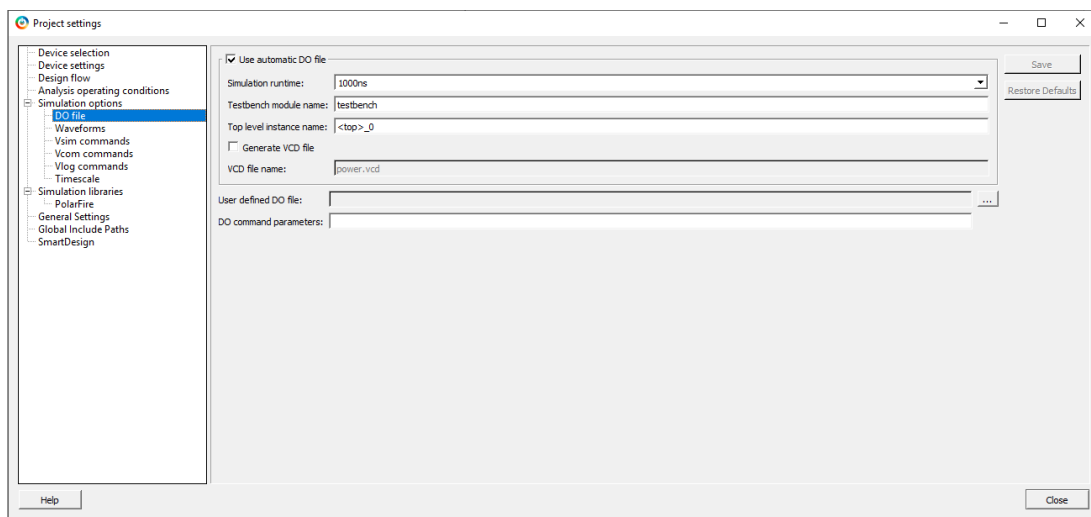
4.10.1. Project Settings: Simulation - Options and Libraries [\(Ask a Question\)](#)

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. In the **Simulation options**, select the library whose path you want to change:
 - **DO file**
 - **Waveforms**
 - **Vsim commands**
 - **Vcom commands**
 - **Vlog commands**
 - **Timescale**
4. For **Simulation libraries**, click the library whose path you want to change.

Figure 4-20. Project Settings: Simulation Options



4.10.2. DO file [\(Ask a Question\)](#)

Table 4-7. 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	Available when Use automatic DO file is checked. Specifies how long the simulation must run. If the value is 0 or if the field is empty, the run command is omitted from the <code>run.do</code> file.

Table 4-7. DO File Options (continued)

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

4.10.3. Waveforms [\(Ask a Question\)](#)

Table 4-8. Waveforms Options

Option	Description
Include DO file	Allows you to customize the set of signal waveforms displayed in QuestaSim® ME Pro.
Display waveforms for	Displays signal waveforms for the top-level testbench or for the design under test. - top-level testbench: Project Manager outputs the line <code>addwave/testbench/*</code> in the DO file <code>run.do</code>. - DUT: Project Manager outputs the line <code>add wave/testbench/DUT/*</code> in the <code>run.do</code> file.
Log all signals in the design	Saves and logs all signals during simulation.

4.10.4. Vsim Commands [\(Ask a Question\)](#)

Table 4-9. Vsim Command Options

Option	Description
Resolution	Default is 1 ps. 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.

Table 4-9. Vsim Command Options (continued)

Option	Description
Additional options	Text entered in this field is added to the <code>vsim</code> command. - SRAM ECC Simulation: Two options can be added to specify the simulated error and correction probabilities of all ECC SRAMs in the design: <code>-gERROR_PROBABILITY=<value></code>, where $0 \leq \text{value} \leq 1$. <code>-gCORRECTION_PROBABILITY=<value></code>, where $0 \leq \text{value} \leq 1$. - During Simulation: Raises SB_CORRECT and DB_DETECT flags on each SRAM block based on generated random numbers that fall below the specified <code><value>S</code>. When you run the Post-Layout Simulation Tool, a <code>run.do</code> file is created that consists of information that must be sent to a simulator. To run a simulation on a corner, select an SDF corner and 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 pulse filtering during SDF simulations. - After you select the corner, appropriate files for simulation are written in the <code>run.do</code> file, as shown in the following figure.

Figure 4-21. Files Written to the run.do File

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

4.10.5. VCOM Command [\(Ask a Question\)](#)

The `VCOM` command is used for VHDL compilation. It is used to compile .vhd and .vhdl files into a simulation library. It also handles VHDL dependencies (packages first, and then entities and architectures).

Table 4-10. VCOM Command Option

Option	Description
Additional options	Text entered in this field is added to the <code>VCOM</code> command.

Figure 4-22. Files Written to the run.do File

```
vcom -2008 -explicit +cover -work postsynth "${PROJECT_DIR}/hdl/led_pwm_pkg.vhd"
vcom -2008 -explicit +cover -work postsynth "${PROJECT_DIR}/stimulus/sd_tb.vhd"
```

4.10.6. VLOG Command [\(Ask a Question\)](#)

The `VLOG` command is used for System and Verilog compilation. It contains commands that invoke the simulator's `VLOG` compiler, and is used to compile .v and .sv files into a simulation library. It also handles macros, include paths, and optimization flags and ensures files are compiled in the correct order.

Table 4-11. VLOG Command Option

Option	Description
Additional options	Text entered in this field is added to the <code>VLOG</code> command.

Figure 4-23. Files Written to the run.do File

```
vlog +cover -sv -work postsynth "${PROJECT_DIR}/synthesis/led_brightness_top.v"
```

4.10.7. Timescale [\(Ask a Question\)](#)

Table 4-12. 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. Default: ns
Precision	Enter a value and select s, ms, us, ns, ps, or fs from the drop-down list. Default: isps

4.10.8. Simulation Libraries [\(Ask a Question\)](#)

Table 4-13. 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 Browse button to go to your library directory.

4.10.9. Selecting a Stimulus File for Simulation [\(Ask a Question\)](#)

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 QuestaSim ME Pro without a testbench.

To associate a stimulus:

- 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.
- 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 must be at the bottom of the list.

3. When you are satisfied with the **Associated Source Files** list, click **OK**.

4.10.10. Selecting Additional Modules for Simulation [\(Ask a Question\)](#)

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.

4.10.11. Performing Functional Simulation [\(Ask a Question\)](#)

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 QuestaSim ME Pro, right-click **Simulate** in the Design Hierarchy window and choose **Open Interactively**. QuestaSim ME Pro starts and compiles the appropriate source files. When the compilation completes, the simulator runs for 1 ms 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 [\(Ask a Question\)](#)

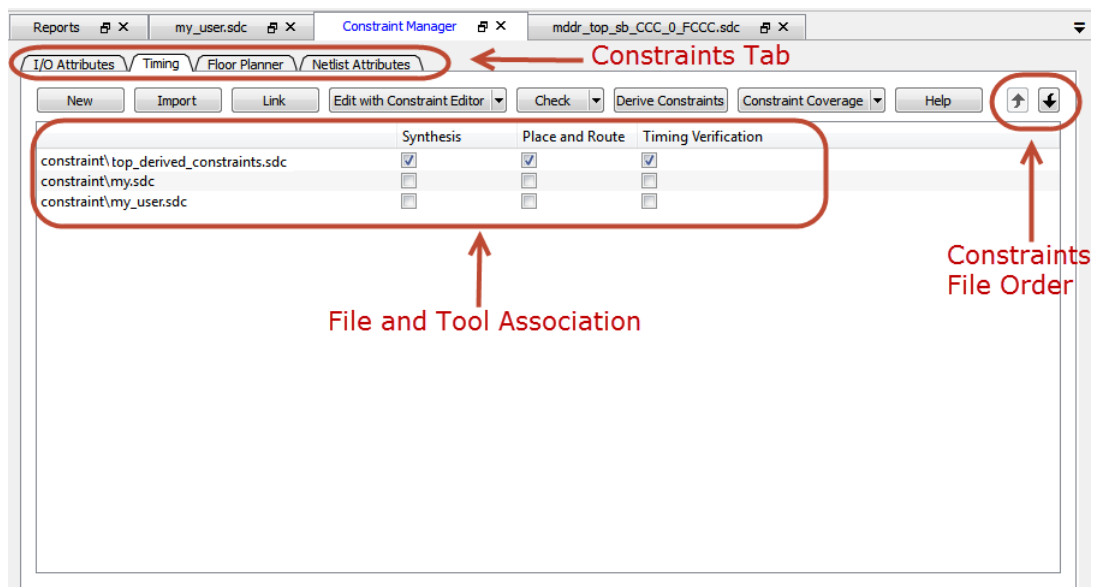
In the FPGA design world, constraint files are as important as design source files. Constraint files are used throughout the FPGA design process to guide FPGA tools to achieve the timing and power requirements of the design. For the synthesis step, SDC timing constraints set the performance goals whereas 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. Opening Constraint Manager [\(Ask a Question\)](#)

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

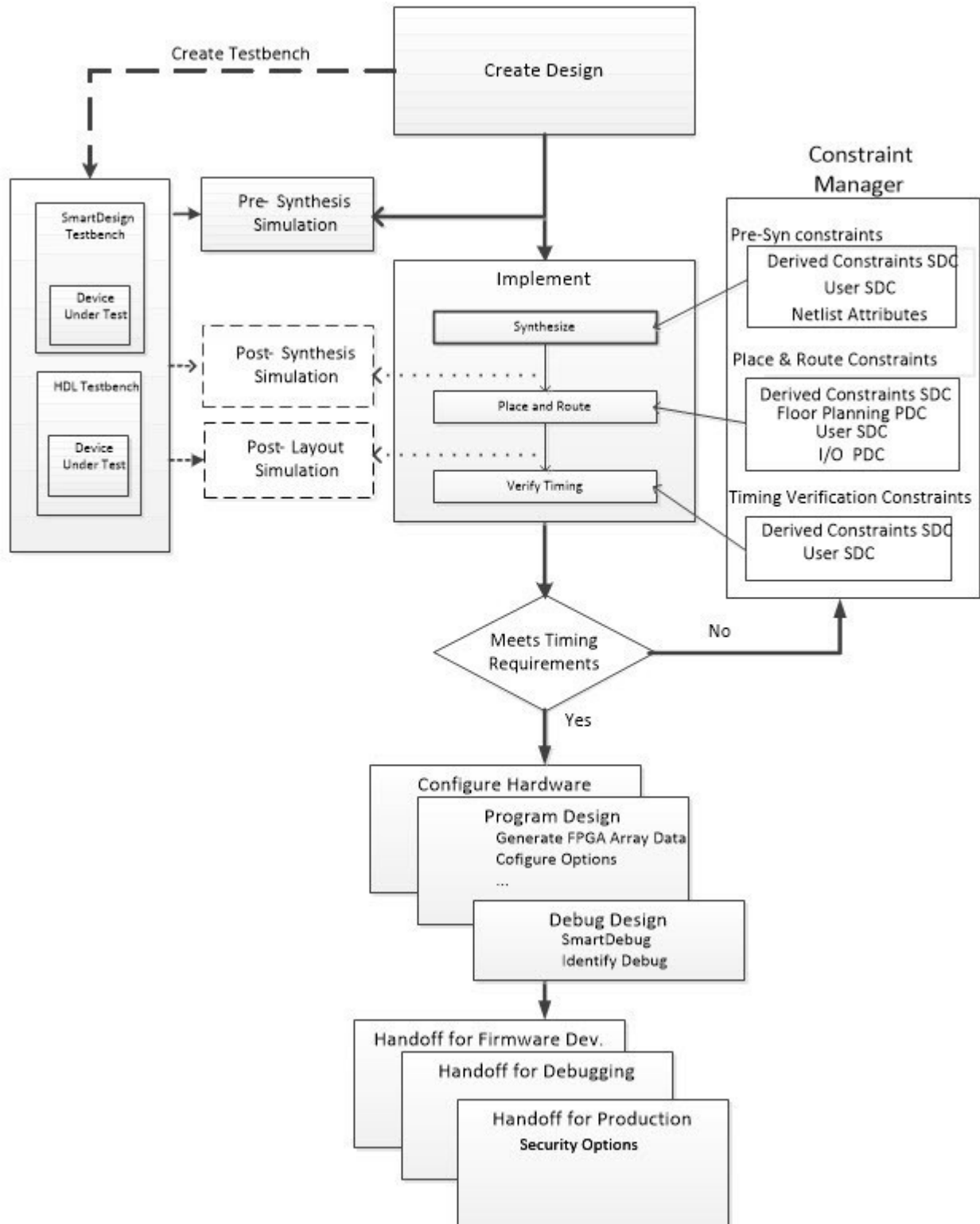
Figure 5-1. Constraint Manager



5.2. Libero SoC Design Flow [\(Ask a Question\)](#)

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 [\(Ask a Question\)](#)

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) [\(Ask a Question\)](#)

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, check the check box under synthesis to associate the FDC file with Synthesis.

5.3.2. Compile Netlist Constraints (*.ndc) [\(Ask a Question\)](#)

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, check the check box under synthesis to associate the NDC file with Synthesis.

5.3.3. SDC Timing Constraints (*.sdc) [\(Ask a Question\)](#)

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, check the check box under **Synthesis** to associate the SDC file with Synthesis.

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

5.3.4. Derived Timing Constraints (*.sdc) [\(Ask a Question\)](#)

These are timing constraints Libero SoC generates for IP cores used in your design. These IP cores are available in the Catalog and 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 [\(Ask a Question\)](#)

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 [\(Ask a Question\)](#)

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

Check the check box under **Place and Route** to associate the file with Place and Route.

5.3.7. Floorplanning PDC Constraints [\(Ask a Question\)](#)

These are floorplanning Physical Design Constraints in a `*fp.pdc` file. From the Constraint Manager **Floor Planner** tab, you can import (**Floor Planner > Import**) or create in the Text Editor (**Floor**

Planner > New) a *.fp.pdc file. Check the check box under **Place and Route** to associate the file with Place and Route.

5.3.8. Timing SDC Constraint File (*.sdc) [\(Ask a Question\)](#)

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. Check 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 [\(Ask a Question\)](#)

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 (for example, `create_clock` and `create_generated_clock`) and design-specific timing exceptions (for example, `set_false_path` and `set_multicycle_path`) for Timing Analysis.

From the Constraint Manager **Timing** tab, you can import (**Timing > Import**) or create an SDC timing file in the Text Editor (**Timing New**). Check the check box under **Timing Verifications** to associate the SDC timing constraints file with Timing Verifications.

Note: You might 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, if the design does not meet timing requirements, it might be useful in subsequent iterations to have different sets of Timing SDC files associated with different tools. For example, you might want to change 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 allows you to associate and disassociate constraint files with the different tools.

5.3.10. Constraint Manager Components [\(Ask a Question\)](#)

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

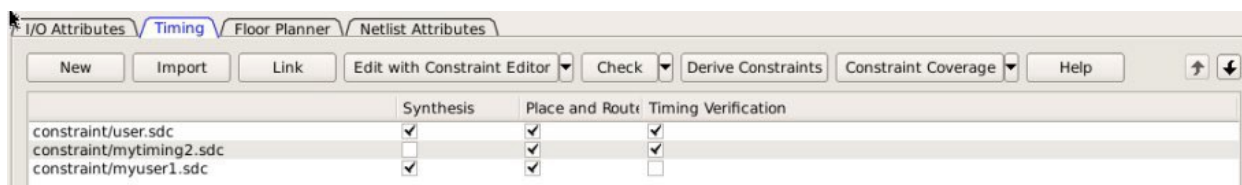
- I/O Attributes
- Timing
- Floor Planner
- Netlist Attribute

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

5.3.11. Constraint File and Tool Association [\(Ask a Question\)](#)

Each constraint file can be associated or disassociated with a design tool by checking or unchecking the check box 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



The Libero SoC Design Flow window displays the state of the tool. A green check mark



indicates that the tool has run successfully. 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 using the following Interactive Tools that you start from the Constraint Manager:

- 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 [\(Ask a Question\)](#)

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 [\(Ask a Question\)](#)

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

- (Optional) Double-click the constraint file in the Constraint Manager to add more constraints to the file.

Observe the following guidelines:

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

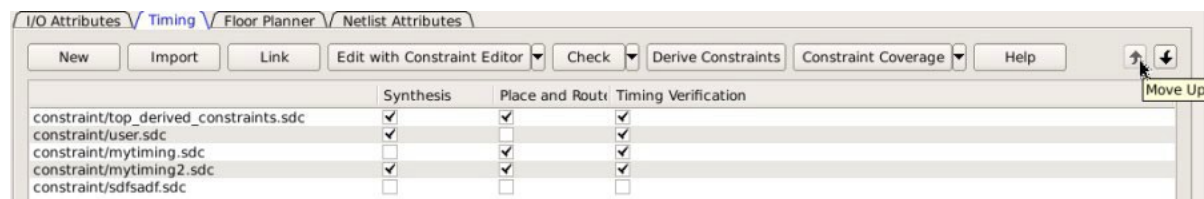
- Select the tab that corresponds to the type of constraint you want to create.
- Click **Edit** to open the Interactive Tools. The Interactive Tool that Libero SoC opens varies with the constraint type:
 - I/O Editor to edit/create I/O Attribute Constraints. See the [I/O Editor User Guide](#) for details.
 - Chip Planner to edit/create Floorplanning constraints. See the [Chip Planner User Guide](#) for details.
 - Constraint Editor to edit/create Timing Constraints. See the [Timing Constraints Editor User Guide](#) for details.
- Create the Constraints in the Interactive Tool. Click **Commit and Save**.
- Check that Libero SoC creates the following 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 [\(Ask a Question\)](#)

When there are multiple constraint files of the same type associated with the same tool, use the Up and Down arrows to sort 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.

- To move a constraint file up, select the file and click the Up arrow.
- 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 [\(Ask a Question\)](#)

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 that corresponds 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 that corresponds 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 [\(Ask a Question\)](#)

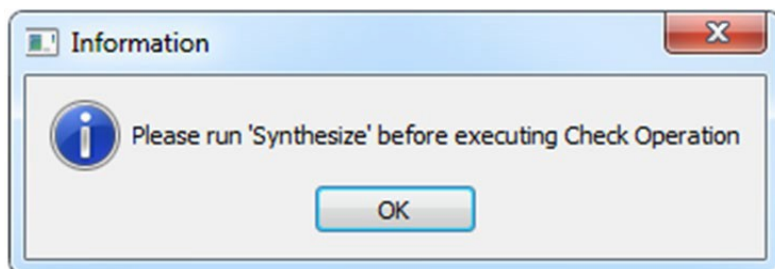
Use the Constraint Manager to check a constraint file.

To check a constraint file:

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

Note: You can check I/O constraints, Floorplanning constraints, Timing constraints, and Netlist Attributes only when the design is in the appropriate state. When checked, a pop-up message appears and the design state cannot be checked.

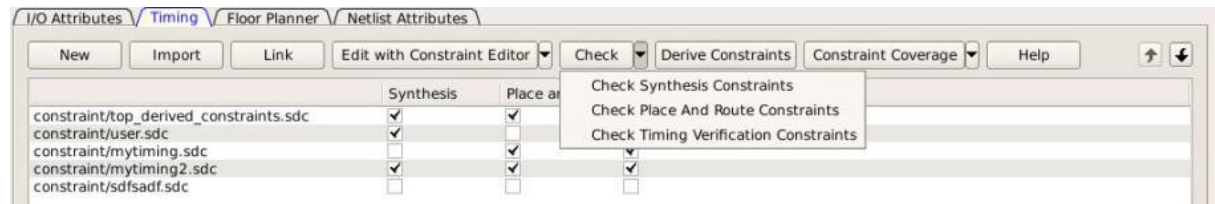
All constraint files associated with the tool are checked. Files not associated with a tool are not checked.



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

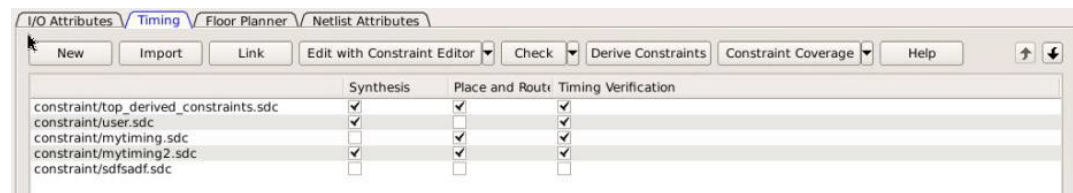
- **Check Synthesis Constraints.** Checks only the constraint files associated with the Synthesis. This constraint checks the following files: `top_derived_constraints.sdc`, `user.sdc`, and `mytiming2.sdc`.
- **Check Place and Route Constraints.** Checks only the constraint files associated with Place and Route. This constraint checks the following files: `top_derived_constraints.sdc`, `mytiming.sdc`, and `mytiming2.sdc`.
- **Check Timing Verification Constraints.** Checks only the Constraint Files associated with Timing Verification. For the constraint files and tool association shown in the following figure. This constraint checks the following files: `top_derived_constraints.sdc`, `user.sdc`, `mytiming.sdc`, and `mytiming2.sdc`.

Figure 5-5. Check Constraints



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

Figure 5-6. Timing Constraints SDC File and Tool



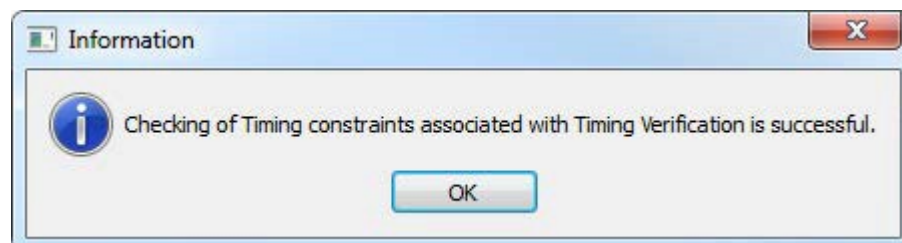
Association when a constraint file is checked, the Constraint Manager:

- 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 (for example, constraints on a design object that does not exist in the netlist) are flagged as errors and reported in the `*.log` file or message window.

5.4.2. Check Result [\(Ask a Question\)](#)

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 Failure Message

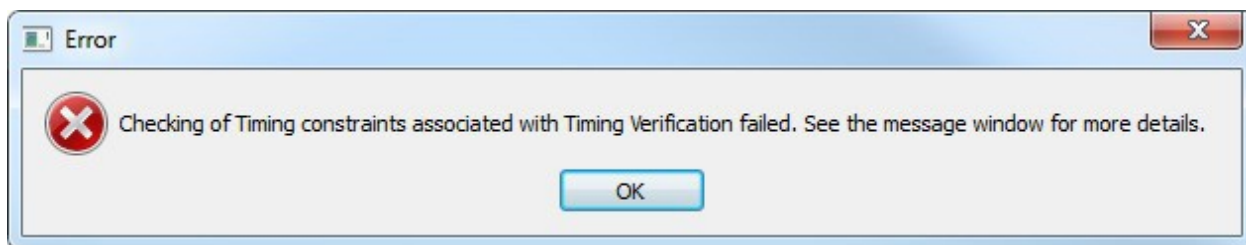


Table 5-2. Constraint File Check Results

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

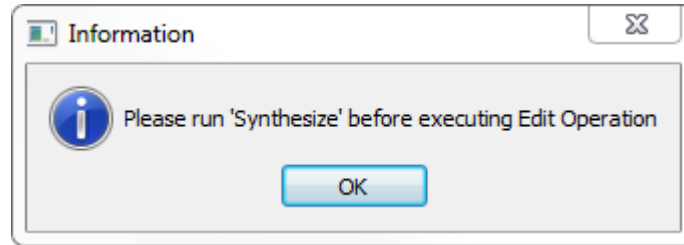
5.4.3. Edit a Constraint File [\(Ask a Question\)](#)

The **Edit** button in the Constraint Manager allows you to create new constraint files and edit existing constraint files (see [Create New Constraints](#)).

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

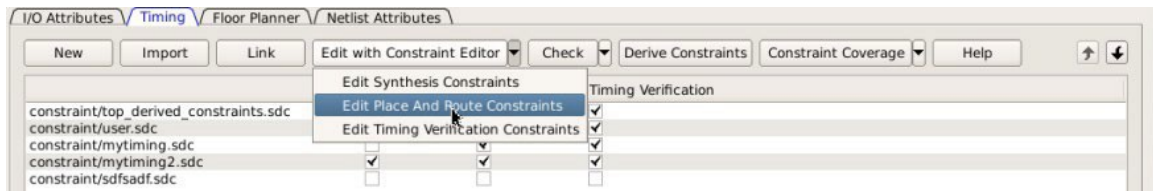
1. Select the tab for the constraint type to edit. An Interactive Tool opens to make the edits.
 2. Click **Edit**.
 - All constraint files associated with the tool are edited. Files not associated with the tool are not edited.
 - When a constraint file is edited, the constraints in the file are read 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, see the [I/O Editor User Guide](#).
 - Chip Planner to edit Floorplanning Constraints (<proj>\fp*.pdc). For details, see the [Chip Planner User Guide \(Chip Planner > Help > Reference Manuals\)](#)
 - Constraint Editor to edit Timing Constraints (constraints*.sdc). For details, see 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

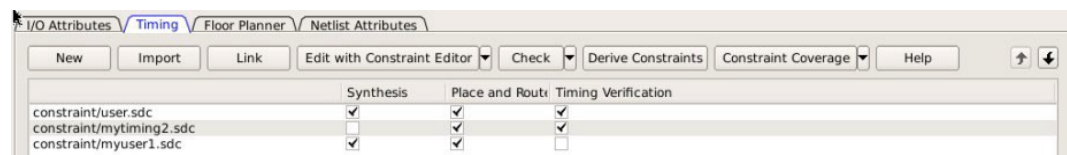
Figure 5-10. Edit Drop-down Menu



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

- 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.

See 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: Changing a constraint file invalidates the state of the tool with which the constraint file is associated. For instance, if Place and Route completed successfully with `user.sdc` as the associated constraint file, changing `user.sdc` invalidates Place and Route. Next to Place and Route, the green check mark that denotes successful completion changes to a warning icon when the tool is invalidated.

5.5. Constraint Types [\(Ask a Question\)](#)

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 for each constraint type.

Table 5-3. Constraint Type Features

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
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 [\(Ask a Question\)](#)

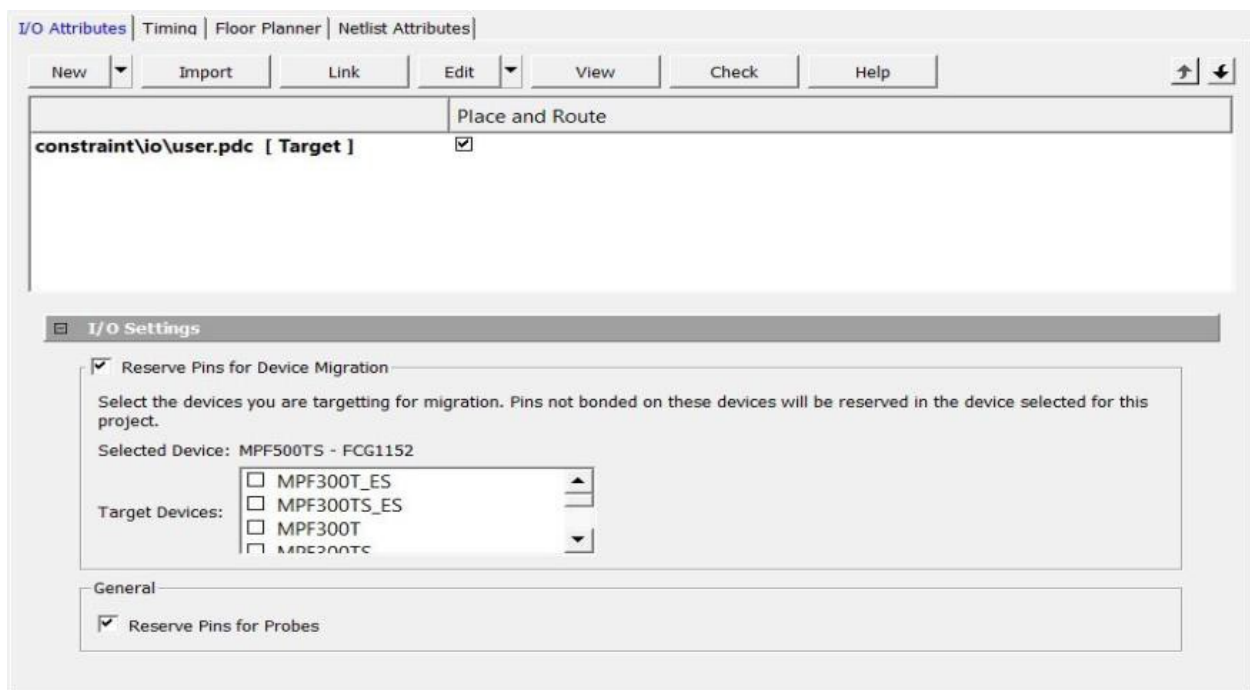
The I/O Attributes tab allows you to manage I/O attributes/constraints for your design's Inputs, Outputs, and Inouts. All I/O constraint files (PDC) have the *.pdc file extension and are placed in the <Project_location>/constraint/io folder. Available actions are:

- **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 started 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 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 [\(Ask a Question\)](#)

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

5.6.2. I/O Settings [\(Ask a Question\)](#)

Table 5-4. I/O Settings

Setting	Description
Reserve Pins for Device Migration	Reserves pins in the currently selected device that are not bonded in a device or list of devices you might later decide to migrate your design to. Select the target device(s) you might 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	• Checked = live probes can be used when debugging your design with SmartDebug. • Not checked = I/Os can be used as General Purpose I/Os.

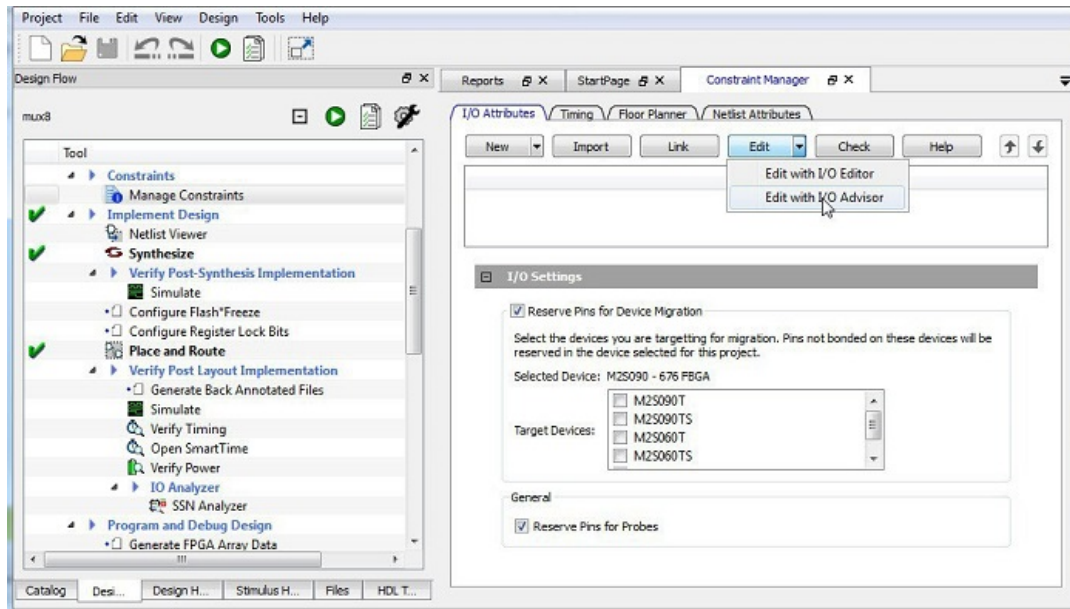
5.7. IO Advisor (SmartFusion 2, IGLOO 2, and RTG4) [\(Ask a Question\)](#)

The IO Advisor allows you to balance the timing and power consumption of the IOs in your design. For output I/Os, it offers suggestions about Output Drive and Slew values that meet (or get as close as possible to) the timing requirements and generates the lowest power consumption. For Input I/Os, it offers suggestions on On-Die Termination (ODT) Impedance values (when the ODT Static is ON) that meet (or get as close as possible to) the timing requirements and generates the lowest power consumption.

Timing data information is obtained from the Primary analysis scenario in SmartTime. Power data is obtained from the Active Mode in SmartPower.

From the Design Flow window, select **Manage Constraints > Open Manage Constraints View > I/O Attributes > Edit > Edit with I/O Advisor**.

Figure 5-13. I/O Advisor



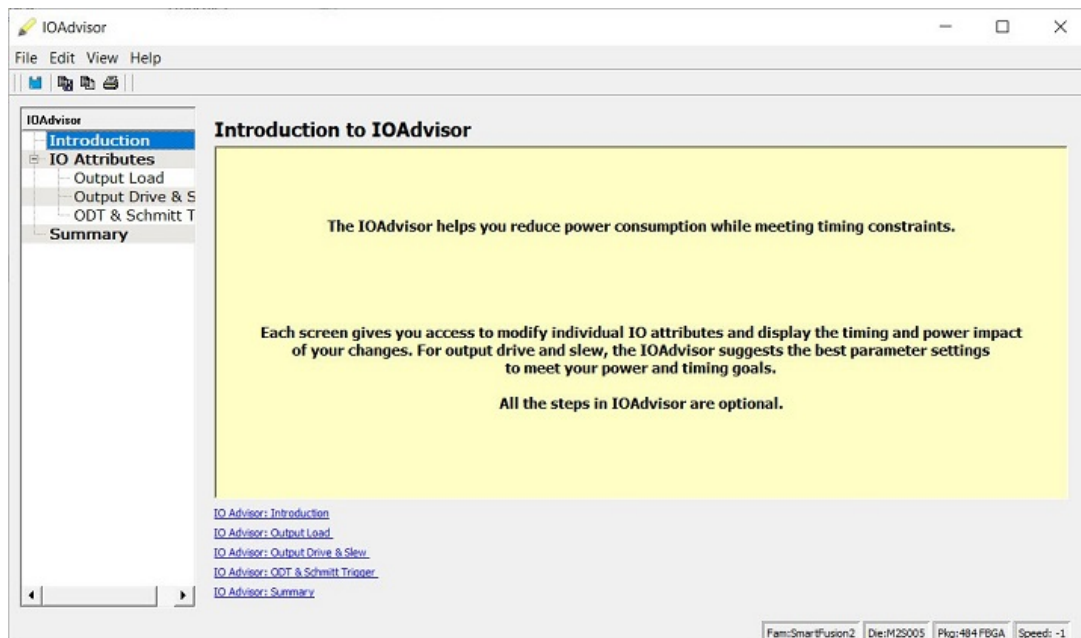
5.7.1. Introduction [\(Ask a Question\)](#)

The Introduction screen provides general information about the IO Advisor. Navigational panels allow you to navigate the following panels:

- **Output Load.** Displays the IO load Power and Delay values for Outputs and Inouts.
- **Output Drive and Slew.** Displays the Output Drive and Slew for Outputs and Inouts.
- **ODT & Schmitt Trigger.** Displays the ODT Static (On/Off), the ODT Impedance value (Ohms) for Inputs and Inouts and the Schmitt Trigger (ON/OFF).

All steps in the IO Advisor are optional.

Figure 5-14. IO Advisor - Introduction



5.7.2. Output Load [\(Ask a Question\)](#)

The Output Load panel displays the load of all output/inout ports in your design.

The display is sorted by Initial or Current value and is selectable in the **Sort By** drop-down menu.

Tooltips are available for each cell of the Table. For output and inout ports, the tooltip displays the Port Name, Macro Name, Instance Name, and Package Pin. Inout ports are identified by a blue bubble icon.

Figure 5-15. IO Advisor - Output Load Panel

5.7.3. Search and Regular Expressions [\(Ask a Question\)](#)

To search for a specific Port, enter the Port Name in the Port Name Search field and click Search. Regular expressions are accepted for the search. All Port Names matching the regular expression are displayed. The regular expression "FDDR*", for example, results in all the output ports beginning with FDDR in the Port Name appearing in the display.

Figure 5-16. Search Field and Regular Expressions

5.7.4. Status Column [\(Ask a Question\)](#)

The icon in the Status column displays the status of the output port.

Table 5-5. Description of Status Column

Icon	Status	Description
	OK	The I/O attributes match the suggestion in the Output Drive and Slew table.
	Error	The Timing constraints for this I/O are not met in the Output Drive and Slew table.
	Information	To improve the power and/or timing of the I/O, apply the suggestion in the Output Drive and Slew table.

5.7.5. Column Display and Sorting [\(Ask a Question\)](#)

To hide or unhide a column, click on the drop-down menu of a column header and select **Hide Column** or **Unhide All Columns**.

To sort the contents of a column, select the column header, and from the right-click menu, select **Sort A to Z**, **Sort Z to A**, **Sort Min to Max**, or **Sort Max to Min** as appropriate.

5.7.6. Set Output Load [\(Ask a Question\)](#)

To set the output load of a port, click the Port and click **Set Output Load** or edit the value in the Current Output Load cell. Initial value remains unchanged.

5.7.7. Restore Initial Value [\(Ask a Question\)](#)

To restore a Port's output load to the initial value, select the output port and click **Restore Initial Value**. The current value changes to become the same value as the initial value.

5.7.8. Output Drive and Slew [\(Ask a Question\)](#)

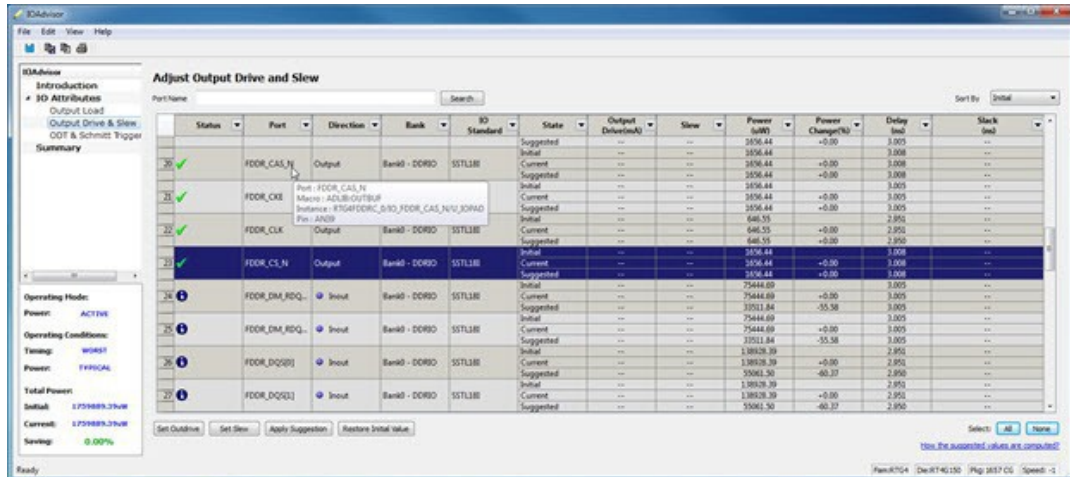
The Output Drive and Slew page displays the Output Drive and Slew of all output/inout ports of your design.

The display can be sorted according to the initial current or suggested values. To change the sorting, use the **Sort By** drop-down list.

Three values are displayed for Output Drive and Slew of each IO output/inout port:

- **Initial.** This is the initial value when the IO Advisor is launched.
- **Current.** This is the current value which reflects any changes you have made, including suggestions you have accepted from the IO Advisor.
- **Suggested.** This is the suggested value from the IO Advisor for optimum power and timing performance.

Figure 5-17. IO Advisor – Output Drive and Slew



5.7.9. How the Suggested Values Are Computed [\(Ask a Question\)](#)

The IO Advisor provides suggestions for output drive and slew values according to the following criteria:

- If you have set no output delay constraint for the port, the IO Advisor suggests IO attribute values that generate the lowest power consumption.
- If you set an output delay constraint on the port, the IO Advisor suggests IO attribute values that generates the lowest power consumption and positive timing slacks. If the slacks of all attribute combinations are negative, the IO Advisor suggests an attribute combination (Drive strength and slew) that generates the least negative slack.

In this screen, you can change the drive strength and slew of the design output I/Os.

1. Select the out drive and/or the slew current value cell.
2. Click the cell to open the combo box.
3. Choose the value you want from the set of valid values.

To restore the initial values, click **Restore Initial Value**.

To change multiple I/Os, select multiple I/Os (CTRL+Click), click **Set Slew** or **Set Outdrive**, select the value, and click **OK**.

5.7.10. Apply Suggestion [\(Ask a Question\)](#)

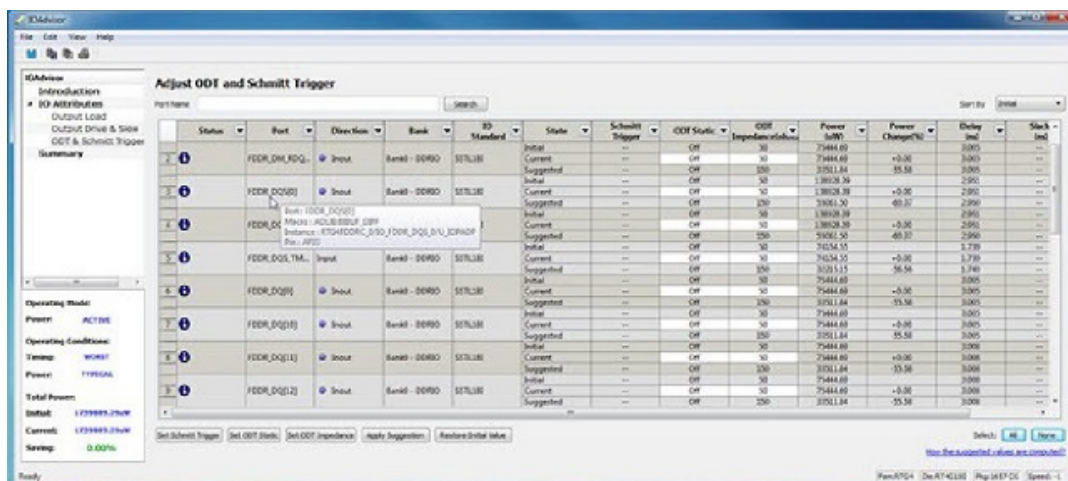
To apply the suggested value to a single output port, select the output port and click **Apply Suggestion**.

To apply the suggested values to multiple ports, select multiple ports (CTRL+Click) and click **Apply Suggestion**.

5.7.11. Adjust ODT and Schmitt Trigger [\(Ask a Question\)](#)

This page allows you to set the Schmitt Trigger setting (ON/OFF), On-Die Termination (ODT) Static setting (ON/OFF), and the ODT Impedance (in Ohms) to valid values for all Input/Inout IOs of your design. The IO Advisor page instantly gives you the Power (in uW) and Delay (in ns) values when you make changes. If the suggested values meet your design's power and/or timing requirements, you can accept the suggestions and continue with your design process.

Figure 5-18. IO Advisor – Adjust ODT and Schmitt Trigger

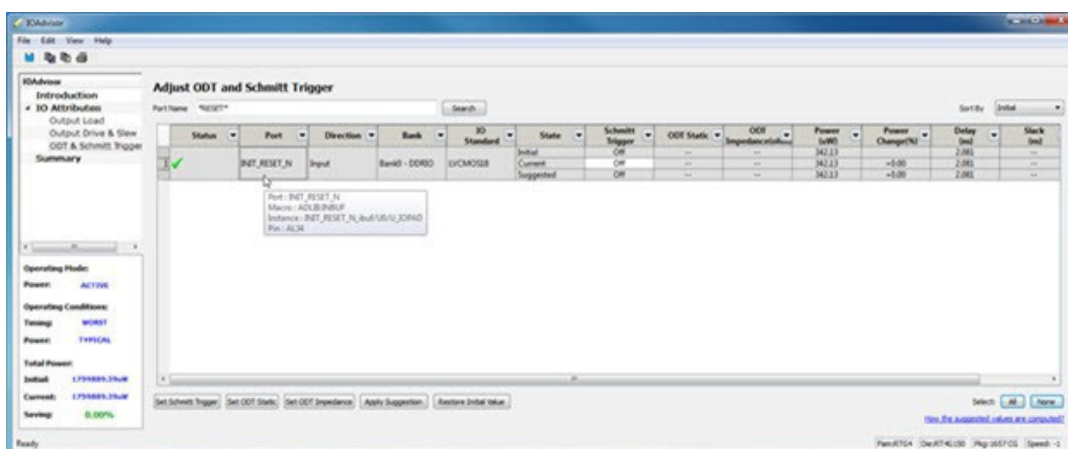


Note: ODT is not allowed for 2.5V or higher single-ended signals. It is allowed for differential signals.

5.7.12. Search and Regular Expressions [\(Ask a Question\)](#)

To search for a specific Port, enter the Port Name in the **Port Name Search** field and click **Search**. Regular expressions are accepted for the search. All Port Names matching the regular expression are displayed. For example, the regular expression **RESET*** results in the input/inout ports with the port name beginning with **RESET** appearing in the display.

Figure 5-19. Search Field and Regular Expressions



5.7.13. Status Column [\(Ask a Question\)](#)

The icon in the Status Column displays the status of the input/inout ports.

Table 5-6. Description of Status Column

Icon	Status	Description
	OK	The I/O attributes match the suggestion in the Adjust ODT and Schmitt Trigger table.
	Error	The Timing constraints for this I/O are not met in the Adjust ODT and Schmitt Trigger table.

Table 5-6. Description of Status Column (continued)

Icon	Status	Description
	Information	To improve the power and/or timing of the I/O, apply the suggestion in the Adjust ODT and Schmitt Trigger table.

5.7.14. Column Display and Sorting ([Ask a Question](#))

To hide or unhide a column, click on the drop-down menu of a column header and select **Hide Column** or **Unhide All Columns**.

To sort the contents of a column, select the column header, and from the right-click menu, select **Sort A to Z**, **Sort Z to A**, **Sort Min to Max**, or **Sort Max to Min** as appropriate.

5.7.15. Set Schmitt Trigger ([Ask a Question](#))

For IO Standards that support the Schmitt Trigger, you can turn the Schmitt Trigger ON or OFF. Select the I/O and click **Set Schmitt Trigger** to toggle between ON and OFF. Your setting is displayed in the **Schmitt Trigger** column for the I/O.

5.7.16. Set ODT Static ([Ask a Question](#))

For IO standards that support ODT static settings, you can turn the ODT Static ON or OFF according to your board layout or design needs:

- **ON.** The Termination resistor for impedance matching is located inside the chip.
- **OFF.** The Terminator resistor for impedance matching is located on the printed circuit board.

To turn the ODT Static ON or OFF, click to select the input/inout port and from the pull-down menu, toggle between ON and OFF. To turn ODT Static ON or OFF, click **Set ODT Static** and toggle between ON and OFF.

5.7.17. Set ODT Impedance (Ω) ([Ask a Question](#))

For each input/inout in your design, valid ODT Impedance values (in Ohms) are displayed for you to choose from. Click to select the input/inout port and select one of the valid ODT impedance values from the pull-down list in the ODT Impedance column. You can also click **Set ODT Impedance** to choose one of the valid ODT impedance values. The Power and Delay values might vary when you change the ODT Impedance (Ω).

Note: When `ODT_static` is set to OFF, changing the `ODT_Impedance` value has no effect on the Power and Delay values. The Power and Delay values change with `ODT_Impedance` value changes only when `ODT_static` is set to ON.

5.7.18. Apply Suggestion ([Ask a Question](#))

To apply the suggested value to a single input/inout port, select the port and click **Apply Suggestion**. To apply the suggested values to multiple ports, select the multiple ports (Control-click) and click **Apply Suggestion**.

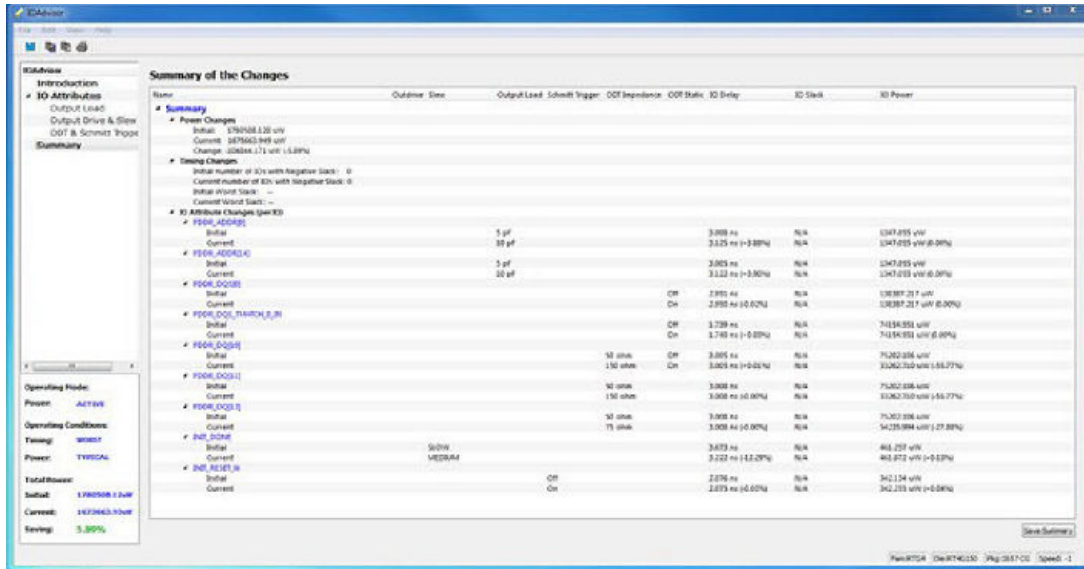
5.7.19. Restore Initial Value ([Ask a Question](#))

To restore an input/inout port's attribute values to the initial values, select the port and click **Restore Initial Value**. The current value changes to the same value as the initial value.

5.7.20. Summary of Changes ([Ask a Question](#))

This screen provides a summary of the timing and power changes you made in the IO Advisor.

Figure 5-20. IO Advisor – Summary



To save the summary, click **Save Summary**, select a save format (text or CSV), and click **OK**.

To commit IO Attribute changes you to the database (the *io_pdc file), choose **Save** from the **File** menu (**File > Save**), an then click **OK** in the dialog that appears.

Note: After saving the changes into the pdc file and database, the summary refreshes automatically and shows the latest data as per the latest database.

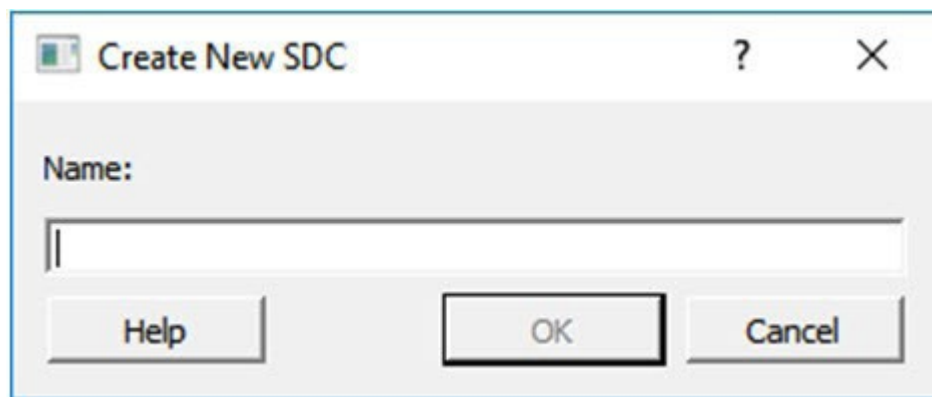
5.8. Constraint Manager: Timing Tab [\(Ask a Question\)](#)

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 options are:

- **New.** Creates a new timing SDC file and saves it in 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.

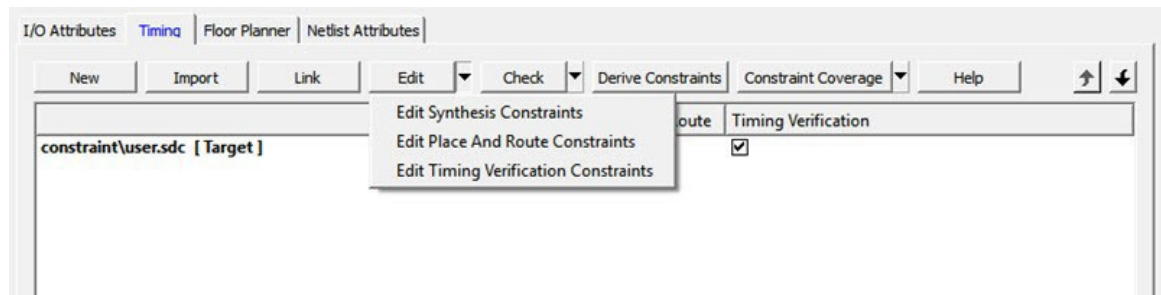
Figure 5-21. Create New SDC Dialog Box



- **Import.** Imports an existing timing SDC file into the Libero SoC project. The timing SDC file is copied into the <Project_location>\constraint folder.

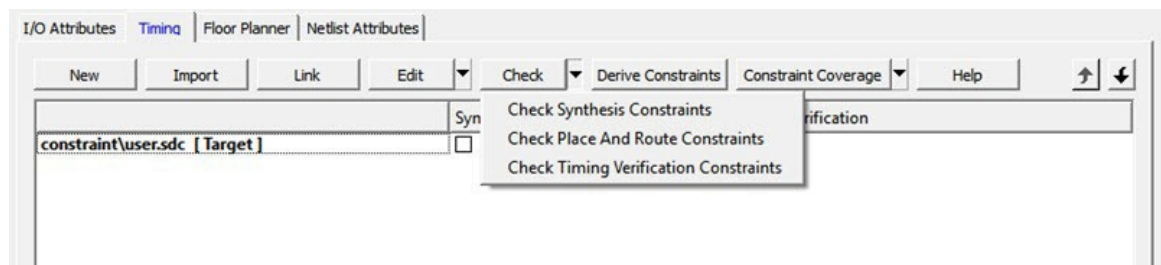
- **Link.** Creates a link in the project's constraint folder to an existing timing SDC file (located and maintained outside of the Libero SoC project).
- **Edit.** Opens the Timing Constraints Editor (see the [Timing Constraints Editor User Guide](#) for details) to modify the SDC file(s) associated with one of the three tools:
 - **Synthesis.** Loads the timing SDC file(s) associated with the Synthesis tool into the constraints editor for editing.
 - **Place and Route.** Loads the timing SDC file(s) associated with the Place and Route tool into the constraints editor for editing.
 - **Timing Verification.** Loads the timing SDC file(s) associated with the Timing Verification tool into the constraints editor for editing.

Figure 5-22. Timing Constraints Edit Options in Constraint Manager



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

Figure 5-23. Timing Constraints Check Options in Constraint Manager

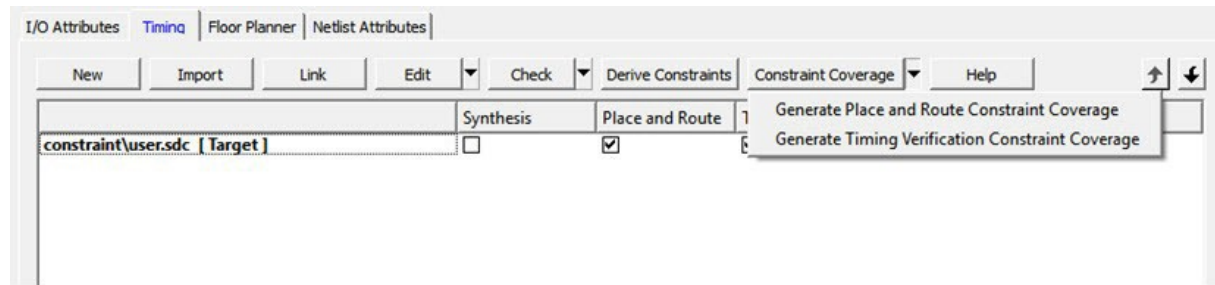


- **Derive Constraints.** Generates a timing SDC file based on your configuration of the IP core, components, and component SDC, which includes `create_clock` and `create_generated_clock` SDC timing constraints. This file is named `<top_level>_derived_constraints.sdc`. The component SDC and the generated `<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.** Displays a pull-down list for selecting the Generate Place and Route Constraint Coverage report and Generate Timing Verification Constraint Coverage report.

Figure 5-24. Constraint Coverage Options for Timing Constraints in Constraint Manager



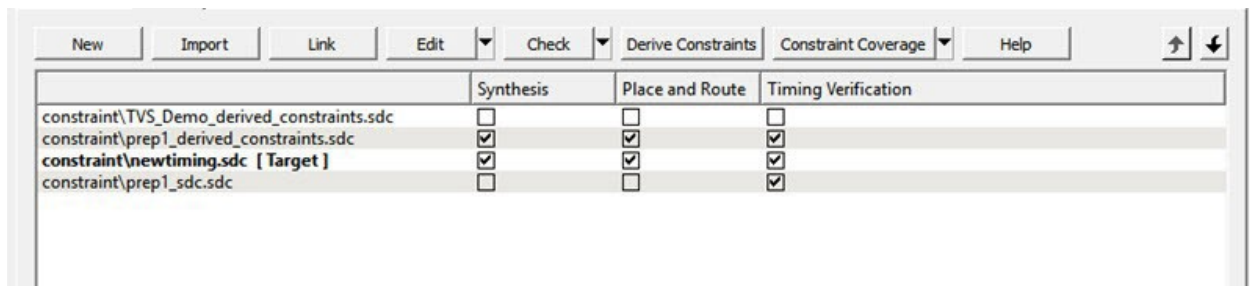
Note: Constraint Coverage Reports can be generated only after synthesis. A warning message appears if the design is not in the post-synthesis state when this button is clicked.

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

If the SmartTime Constraint Editor tool is started 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.

If 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 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 5-25. Constraint Manager – Timing Tab



Right-click the timing SDC files to access the available options 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 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 available for linked files only.

File and Tool Association

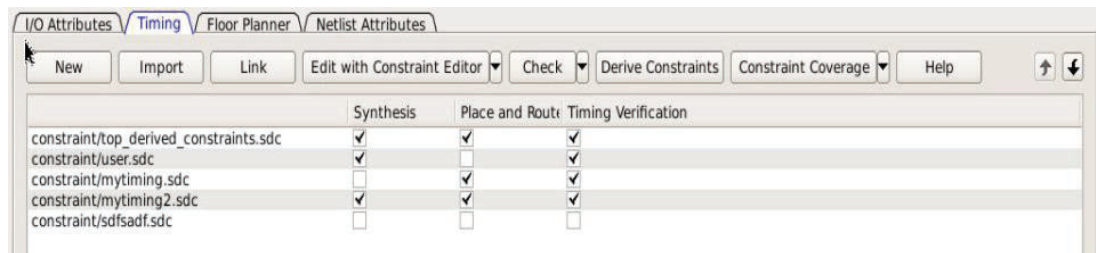
Each timing constraint file can be associated or disassociated with any one, two, or all three of the following tools:

- Synthesis
- Place and route
- Timing Verification

Check the check box under **Synthesis**, **Place and Route**, or **Timing Verification** to associate the file with or disassociate the file from the tool. When a file is associated, Libero passes the file to the tool for processing.

5.8.1. Example (PolarFire) [\(Ask a Question\)](#)

Figure 5-26. File and Tool Association Example (PolarFire)



In the preceding figure, 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. The files `mytiming.sdc` and `sdfsadf.sdc` are not read because they are not associated with Synthesis. When the SmartTime Constraint Editor opens for editing, the constraints from all the files except `sdfsadf.sdc` are read and loaded into the Constraint Editor. Any changes you make and save in the Constraint Editor are written back to the files.

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

5.8.2. Example 1: RTG4, SmartFusion 2, and IGLOO 2 [\(Ask a Question\)](#)

When **Edit Synthesis Constraint** is selected, as shown in the following figure, the files `prepl_derived_constraint.sdc` and `newtiming.sdc` are read because they are associated with Synthesis. The files `TVS_Demo_derived_constraints.sdc` and `prepl_sdc.sdc` are not read because they are not associated with Synthesis. When the SmartTime Constraints Editor opens for edit, the files `prepl_derived_constraint.sdc` and `newtiming.sdc` are read and loaded into the Constraints Editor. Any changes made to the constraints in these two files and saved in the Constraints Editor are written back to the two files.

When **Edit Synthesis Constraint** is selected, as shown in the following figure, the files **user.sdc**, **top_derived_constraints.sdc**, and **mytiming2.sdc** are read because they are associated with Synthesis. The files `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 `sdfsadf.sdc` are read and loaded into the Constraint Editor. Any changes made and saved in the Constraint Editor are written back to the files.

Figure 5-27. Example 1: Edit Synthesis Constraint

New Import Link Edit Check Derive Constraints Constraint Coverage Help ↑ ↓			
	Synthesis	Place and Route	Timing Verification
constraint\TVS_Demo_derived_constraints.sdc	☐	☐	☐
constraint\prep1_derived_constraints.sdc	☒	☒	☒
constraint\newtiming.sdc [Target]	☒	☒	☒
constraint\prep1_sdc.sdc	☐	☐	☒

5.8.3. Example 2: RTG4, SmartFusion 2, and IGLOO 2 [\(Ask a Question\)](#)

When **Check Synthesis Constraint** is selected, as shown in the following figure, the files `prep1_derived_constraints.sdc` and `newtiming.sdc` are checked because these two files are associated with Synthesis and the files `TVS_Demo_derived_constraints.sdc` and `prep1_sdc.sdc` are not checked because they are not associated with Synthesis.

When **Check for Timing Verification** is selected, the files `prep1_derived_constraints.sdc`, `newtiming.sdc`, and `prep1_sdc.sdc` are checked because they are associated with Timing Verification. The file `TVS_Demo_derived_constraints.sdc` is not checked because it is not associated with Timing Verification.

Figure 5-28. Example 2: Check Synthesis Constraint and Check for Timing Verification

New Import Link Edit Check Derive Constraints Constraint Coverage Help			
	Synthesis	Place and Route	Timing Verification
constraint\TVS_Demo_derived_constraints.sdc	☐	☐	☐
constraint\prep1_derived_constraints.sdc	☒	☒	☒
constraint\newtiming.sdc [Target]	☒	☒	☒
constraint\prep1_sdc.sdc	☐	☐	☒

5.9. Derived Constraints [\(Ask a Question\)](#)

Libero SoC can generate SDC timing constraints for design components when the root of the design is 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 check box under the tool.

Use the following procedure to generate SDC timing constraints for IP cores.

Tip: To derive constraints accurately, Microchip recommends you use the IP configurators to generate your design's components when applicable (for example, CCC).

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 files.

- When prompted about 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 check box 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: The complete hierarchical path names are used to identify design objects in the generated SDC file. 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]).

5.10. Automatic Clock Jitter Constraint [\(Ask a Question\)](#)

Libero SoC calculates a design's "system jitter" automatically based on the device's global net clock jitter of the clock domain with the highest load. Libero assumes that the average toggle rate in each domain is 30%. For cases where 30% avg. toggle rate is not appropriate, see the device datasheet global clock jitter specifications to estimate the global net clock jitter at different effective toggle rates. After the global clock jitter is ascertained for the design's highest loaded clock domain, you can override the automatically calculated system jitter using the `set_system_jitter` SDC constraint (see the [Timing Constraints Editor User Guide](#)).

Libero SoC also calculates the clock generation path contribution to the clock jitter based on the device datasheet specifications for blocks such as CCCs, internal RC Oscillators, and XCVRs (PolarFire, PolarFire SoC, RT PolarFire, and RTG4).

You are responsible for specifying input clock jitter on externally sourced clocks (see the `set_input_jitter` SDC constraint in the [Timing Constraints Editor User Guide](#)).

For RTG4, you should account for the device datasheet Input Buffer clock jitter specification when specifying the input jitter on external clock inputs. This guidance also applies to CCC/PLL reference clocks sourced from I/O buffers. In addition, RTG4 users should account for the device datasheet specification on effective recovered clock jitter from the CCC SpaceWire Clock Recovery circuits by applying the `set_input_jitter` constraint to the clock object created on the SpaceWire STROBE input.

For each clock domain, Libero SoC creates a clock uncertainty constraint whose value is the larger of the system jitter or the clock's individual clock generation/input path. The project's `/designer/<root>/` subfolder will contain `place_route.sdc` and `verify_timing.sdc` that contains the automatic clock uncertainty constraints used for the Place and Route and Verify Timing design flow steps. These tool-generated files are not to be modified by end-users.

After Place and Route and Verify Timing, Libero SoC generates jitter report files called `place_and_route_jitter_report.txt` and `timing_analysis_jitter_report.txt`, respectively. This report appears in the Libero SoC reports tab and is stored on disk in the project's `/designer/<root>/` subfolder.

If you have separate SDC files for Place and Route versus Verify Timing, make sure the desired `set_input_jitter` and `set_system_jitter` constraints exist in both files.

To obtain more clock jitter margin, you can also specify `set_clock_uncertainty` constraints in your SDC input files for Place and Route and Verify Timing. The user clock uncertainty will be added

to the Libero SoC automatically derived clock uncertainty values, with the sum reflected in the `place_route.sdc` and `verify_timing.sdc` files, respectively. Microchip recommends reviewing the applied clock jitter values in the respective `jitter_report.txt` files.

When determining a clock domain's system jitter, Libero SoC considers the impact of switching activity from loads in synchronous clock domains. This means that loads on synchronous domains are counted together when determining the largest clock domain to use for the design's global net system jitter calculation. By default, external clock inputs are considered to be asynchronous to each other for the purposes of calculating the system jitter.

During timing analysis and when viewing expanded path reports, a **clock jitter** line item appears in the calculation corresponding to the applied clock uncertainty. By default, hold time (min-delay) calculations on a single clock edge of a single clock domain do not have clock jitter constraints applied. In contrast, paths involving multiple clock domains, multiple clock edges, or external interfaces include clock jitter in the timing analysis.

5.11. Constraint Manager: Floor Planner Tab [\(Ask a Question\)](#)

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.

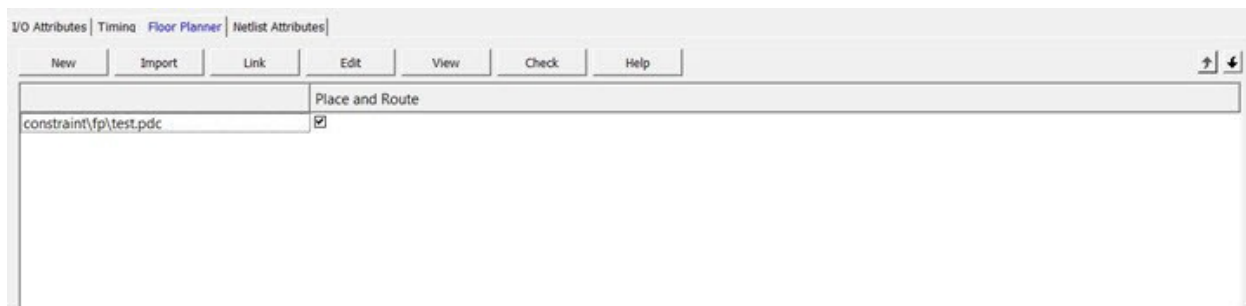
Available options are:

- **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 started 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 5-29. Constraint Manager – Floor Planner Tab



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

- **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 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 linked files only.

5.11.1. File and Tool Association [\(Ask a Question\)](#)

Each floorplanning constraint file can be associated or disassociated to the Place and Route tool. Check the check box under **Place and Route** to associate or disassociate the file from the tool.

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

5.12. Constraint Manager: Netlist Attributes Tab [\(Ask a Question\)](#)

The **Netlist Attributes** tab allows you to manage netlist attribute constraints to optimize your design during the synthesis and/or compile process. Timing constraints must 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.

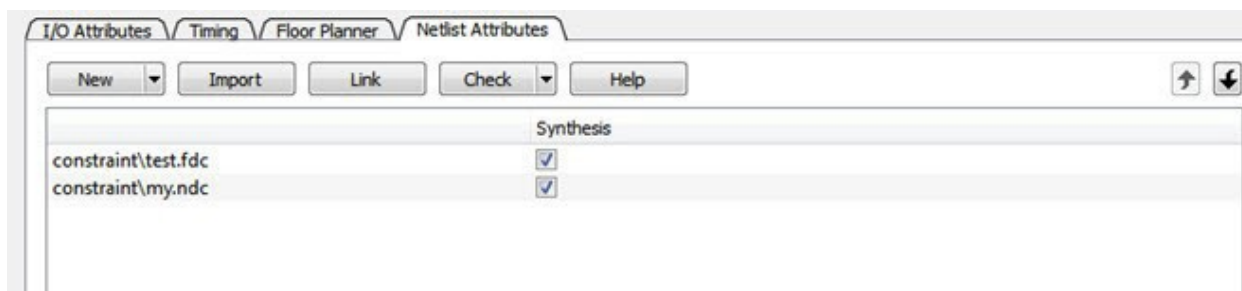
Available options are:

- **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 5-30. Constraint Manager – Netlist Attributes Tab



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

- **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 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.

5.12.1. File and Tool Association [\(Ask a Question\)](#)

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

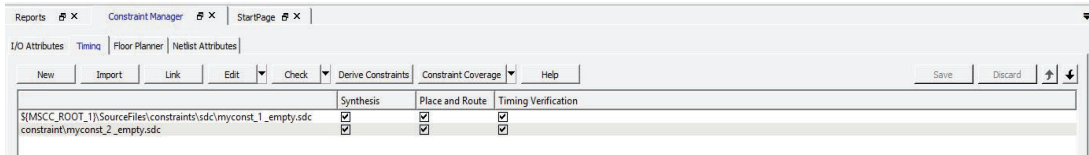
Check the check box under **Synthesis** (Compile) to associate/disassociate the file from Synthesis (Compile). When a file is associated, Libero passes the file to Synthesis (Compile) for processing when Synthesis is run.

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 as a relative path if the relative option is set for linked files. If a constraint file is missing during an environment variable change or if 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 you try to access the file.

Figure 5-31. Netlist Attributes Constraint Files Associated with the Synthesis Tool



6. Implementing Designs [\(Ask a Question\)](#)

The following topics describe how to implement the design.

Notes: Libero uses multiple cores automatically. For this reason, Libero does not provide options for setting multi-cores manually. The following scenarios describe when Libero uses multiple cores. Optimal runtime improvement is achieved when your system has at least four cores. Additional cores will deliver only marginal runtime improvements.

- [Synthesis](#) uses up to four cores when using Libero (only one Synplify Pro license is needed for four cores) if automatic or manual compile points are used.
- [Place and Route](#) uses up to four cores.
- [Static timing analysis](#) uses as many cores available.

6.1. Synthesize [\(Ask a Question\)](#)

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.

Notes: Libero uses multiple cores automatically. For this reason, Libero does not provide options for setting multi-cores manually. The following scenarios describe when Libero uses multiple cores. For maximum runtime improvements, use four cores. Adding more cores delivers marginal improvements.

- Synthesis uses up to four cores when using Libero (only one Synplify Pro license is needed for four cores).
- Automatic or manual compile points.
- Place and Route uses up to four cores.
- Static timing analysis uses as many cores available.

6.1.1. Synthesize Options [\(Ask a Question\)](#)

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

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

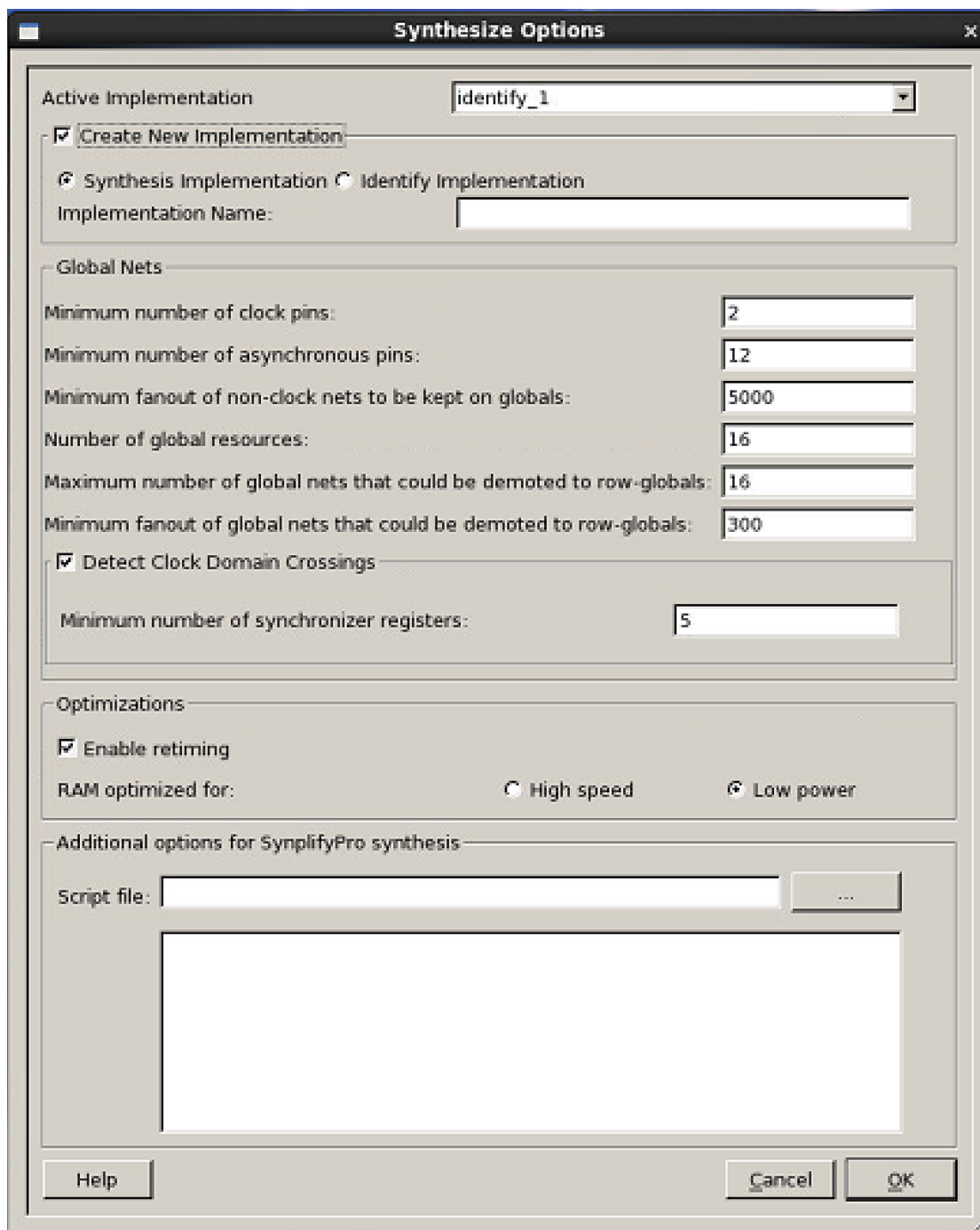
Figure 6-1. Synthesize Options Dialog Box (PolarFire)

The image shows the 'Synthesize Options' dialog box for PolarFire. It has a title bar with a close button. The dialog is organized into several sections:

- Active Implementation:** A dropdown menu showing 'synthesis_2'.
- Create New Implementation:** A checked checkbox.
- Implementation Selection:** Two radio buttons: 'Synthesis Implementation' (selected) and 'Identify Implementation'. Below them is an empty text field for 'Implementation Name'.
- Global Nets:** A section with several numeric input fields:
 - 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:** An unchecked checkbox.
- Detect Clock Domain Crossings:** A checked checkbox.
- Minimum number of synchronizer registers:** An input field with the value 2.
- Optimizations:**
 - Enable retiming: Unchecked checkbox.
 - Enable automatic compile point: Checked checkbox.
 - RAM optimized for: Two radio buttons, 'High speed' (selected) and 'Low power'.
 - Map seq-shift register components to: Two radio buttons, 'Registers' (selected) and 'RAM64x12'.
 - Map ROM components to: Two radio buttons, 'Logic' (selected) and 'RAM'.
- Additional options for SynplifyPro synthesis:**
 - Script file: A text field followed by a browse button ('...').
 - A large empty text area below the script file field.

At the bottom of the dialog are three buttons: 'Help', 'Cancel', and 'OK'.

Figure 6-2. Synthesize Options Dialog Box (SmartFusion 2, IGLOO 2, and RTG4)



The image shows the 'Synthesize Options' dialog box. It has a title bar with a close button. The main area is divided into several sections:

- Active Implementation:** A dropdown menu showing 'identify_1'.
- Create New Implementation:** A checked checkbox.
- Implementation Type:** Two radio buttons: 'Synthesis Implementation' (selected) and 'Identify Implementation'.
- Implementation Name:** An empty text field.
- Global Nets:** A section with several numeric input fields:
 - Minimum number of clock pins: 2
 - Minimum number of asynchronous pins: 12
 - Minimum fanout of non-clock nets to be kept on globals: 5000
 - Number of global resources: 16
 - 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: 300
- Detect Clock Domain Crossings:** A checked checkbox.
- Minimum number of synchronizer registers:** A text field containing '5'.
- Optimizations:**
 - Enable retiming:** A checked checkbox.
 - RAM optimized for:** Two radio buttons: 'High speed' and 'Low power' (selected).
- Additional options for SynplifyPro synthesis:**
 - Script file:** A text field with a browse button ('...').
 - A large empty text area below the script file field.

At the bottom, there are three buttons: 'Help', 'Cancel', and 'OK'.

6.1.1.1. Active Implementation [\(Ask a Question\)](#)

The **Active Implementation** option allows you to select an active Synthesis or Identify implementation from a list of existing implementations. A check box is provided for creating a new Synthesis or Identify implementation. If you check this box, the options in the following table appear.

Table 6-1. Active Implementation Options

Option	Description
Synthesis Implementation	Click to specify the new implementation as a Synthesis Implementation.
Identify Implementation	Click to specify the new implementation as an Identify Implementation.
Implementation Name	Assign a name to the new Synthesis or Identify implementation.

6.1.1.2. Global Nets (Promotions and Demotions) [\(Ask a Question\)](#)

Use the following options to specify 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 6-2. Global Nets Options

Option	Description
Minimum number of clock pins	Threshold value for Clock pin promotion. Default: 2
Minimum number of asynchronous pins	Threshold value for asynchronous pin promotion. Default for PolarFire® and PolarFire SoC: 800 Default for SmartFusion® 2, IGLOO® 2, and RTG4™: 12 This option is not available for the RT4G150_ES device.
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 for PolarFire and PolarFire SoC: 36 Default for SmartFusion 2, IGLOO 2, and RTG4: Number of available global resources for the die you selected for the project. This number varies with different die sizes.
Maximum number of global nets that could be demoted to row-globals	Maximum number of global nets that can 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 can be demoted to row-global. It is undesirable to have high fanout nets demoted using row-globals because it can 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 or select a bigger die for your design. Range: 25–5000 Default for PolarFire and PolarFire SoC: 1000 Default for SmartFusion 2, IGLOO 2, and RTG4: 300

Table 6-2. Global Nets Options (continued)

Option	Description
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: - Minimum number of Enable pins to Infer Gated Clock global: Minimum number of enable pins to infer gated chip-level global. Default: 1000 - Minimum number of Enable pins to Infer Gated Clock row-global: Minimum number of enable pins to infer gated clock row-global. Default: 100
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. Minimum number of synchronizer registers. Range: 2–9. Default: 2

The generated CDC report will not contain any synchronizer circuits formed with macros instantiated from the catalog. The generated report, with the name `<root_name>_cdc.csv`, will be visible in the respective Synthesis node of the report view (**Design > Reports**). The report will contain all CDC inferred from the RTL design and explain the reason(s) why a synchronizer is considered unsafe.

Table 6-3. Reasons Why a Synchronizer Might be Considered Unsafe

Reason	Description
No synchronizer circuit detected	- If the control signals <code>reset</code>, <code>enable</code>, and <code>set</code> on the first and second sync registers do not match. - Combinational logic between the first and second register synchronizers. - Diversion between first and second register synchronizers (that is, 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.	Register levels in synchronizer logic are lower than a certain threshold value.
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 signal of data synchronizer does not 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.

6.1.1.3. Optimizations [\(Ask a Question\)](#)

The following table describes the Optimizations options.

Table 6-4. Optimizations Options

Option	Description
Enable retiming	Check 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 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. - High speed: RAM is optimized for speed. The resulting synthesized design achieves better performance (higher speed) at the expense of more dynamic power. <i>(Default)</i> - Low power: RAM is optimized for power. RAMs are inferred and configured to ensure the lowest power consumption.
Map seq-shift register components to:	Maps sequential shift registers. Choices are: - Registers: Sequential shift logic in the RTL maps to registers. - RAM64x12: Sequential shift logic in the RTL maps to a 64x12 RAM block. This is the default setting.
Map ROM components to:	Maps ROM components. Choices are: - Logic: Maps ROM components to logic. <i>(Default)</i> - RAM: Maps ROM components to RAM.

6.1.1.4. Additional Options for SynplifyPro Synthesis [\(Ask a Question\)](#)

The following table describes additional options for SynplifyPro Synthesis.

Table 6-5. 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. These options are set on the Active Implementation only. Note: Options from the Additional Options Editor have priority over Tcl Script file options if they are the same.

Recommended Synthesis Tcl Options (PolarFire)

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 (PolarFire)

Any additional options can be entered through the Script File or Additional Options Editor. All these options can be added and modified outside of Libero through interactive SynplifyPro.

Refer to the *SynplifyPro Reference Manual* for detailed information about the options and supported families. The following options are already 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 -gclkint_threshold
set_option -rgclkint_threshold
set_option -low_power_gated_clock
set_option -report_preserve_cdc
set_option -min_cdc_sync_flops
```

Recommended Synthesis Tcl Options (SmartFusion 2, IGLOO 2, and RTG4)

```
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 (SmartFusion 2, IGLOO 2, and RTG4)

Any additional options can be entered through the Script File or Additional Options Editor. All these options can be added and modified outside of Libero through interactive SynplifyPro.

Refer to the *SynplifyPro Reference Manual* for detailed information about the options and supported families. The following options are already 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
```

6.1.2. SynplifyPro ME [\(Ask a Question\)](#)

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**.

For 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 *.vm 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.

6.1.3. Identifying Debug Designs [\(Ask a Question\)](#)

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 permit design changes 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 allow you to 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 following debugging flow, 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 that 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 [\(Ask a Question\)](#)

The following sections describe how to verify designs after they are synthesized.

6.2.1. Generating Simulation Files [\(Ask a Question\)](#)

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

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

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

The netlist file is 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 [\(Ask a Question\)](#)

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

- Perform functional simulation before Place and Route to simulate the functionality of the logic in the design.
- Perform timing simulation after the design completes Place and Route. This simulation uses timing information based on the delays in the Place and Route 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 must be at the bottom of the list.
7. When you are satisfied with the **Associated Simulation Files** list, click **OK**.
8. To start QuestaSim ME Pro, right-click **Simulate** in the Design Hierarchy window and choose **Open Interactively**. QuestaSim ME Pro 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, click **Quit** from the **File** menu.

6.3. Compile Netlist [\(Ask a Question\)](#)

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 the 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 6-3. Compile Netlist Options Dialog Box

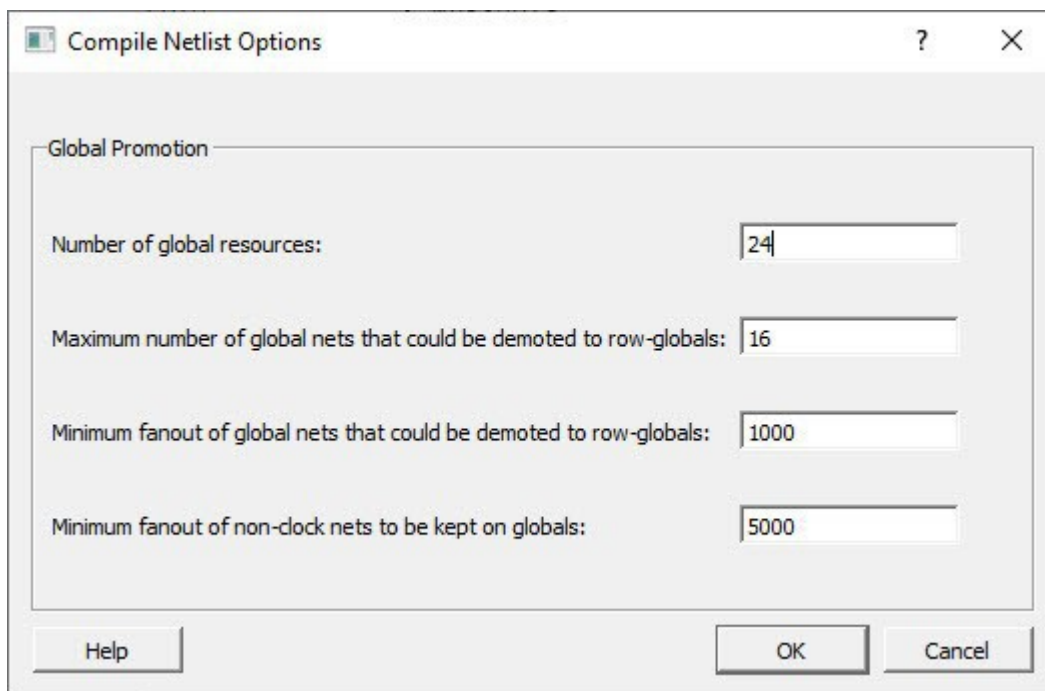


Table 6-6. Compile Netlist Options

Option	Description
Number of global resources	Number of available global resources for the die.

Table 6-6. Compile Netlist Options (continued)

Option	Description
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. Configure Flash*Freeze (SmartFusion 2 and IGLOO 2) [\(Ask a Question\)](#)

SmartFusion 2 SoC and IGLOO 2 FPGAs support Flash*Freeze technology for implementing low-power solutions.

Flash*Freeze mode is an ultra-low power static mode with the lowest standby power of 1.92 mW. It allows easy entry and exit from ultra-low power static mode while retaining SRAM content, I/O state, and register data, significantly reducing power.

For more information about Flash*Freeze mode, see the [UG0444: SmartFusion2 and IGLOO2 FPGA Low-Power Design User Guide](#).

Fabric SRAMs — both the Large SRAM (LSRAM) instances of RAM1xK18 and the Micro SRAM (μSRAM) instances of RAM64x18 — can be placed into Suspend mode or Sleep mode. These SRAMs are grouped in rows in Libero SoC devices.

6.4.1. uRAM/LSRAM State [\(Ask a Question\)](#)

The following table describes the uRAM/LSRAM State options.

Table 6-7. uRAM/LSRAM State Options

Option	Description
Sleep	Selects Sleep mode. LSRAM and uSRAM contents are not retained.
Suspend	Selects Suspend mode. LSRAM and uSRAM contents are retained.

6.4.2. MSS Clock Source [\(Ask a Question\)](#)

The lower the frequency, the lower the power. However, some peripherals, such as SPI or MMUART, can remain active. In these cases, you might need a higher MSS clock frequency (for example, to meet the baud rate for MMUART).

Table 6-8. MSS Clock Source Options

Option	Description
On-Chip 1 MHz RC Oscillator	Uses the on-chip 1 MHz RC oscillator block.
On-Chip 50 MHz RC Oscillator	Uses the on-chip 50 MHz RC oscillator block.

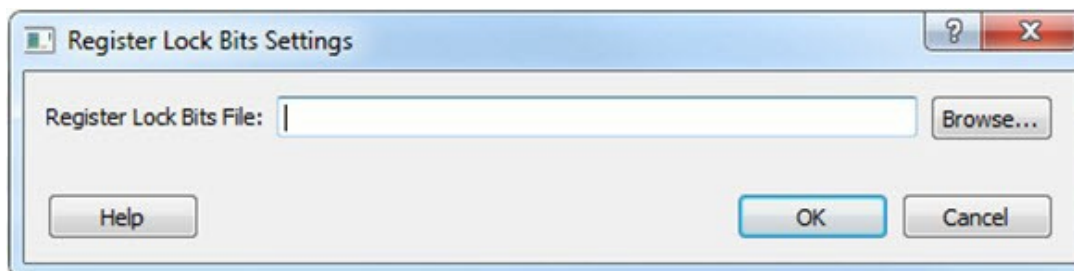
6.5. Configure Register Lock Bits [\(Ask a Question\)](#)

For SmartFusion 2, IGLOO 2, and RTG4 devices, use the Register Lock Bits Configuration tool to lock MSS, SERDES, and FDDR configuration registers to prevent them from being overwritten by initiators

that access these registers. The register lock bits are set in a text file (*.txt) and imported into a SmartFusion 2/IGLOO 2 project.

1. From the Design Flow window, click **Configure Register Lock Bits** to display the Register Lock Bits Setting dialog box.

Figure 6-4. Register Lock Bits Settings Dialog Box



2. Click the **Browse** button to navigate to a text (*.txt) file that contains the Register Lock Bit settings.

6.5.1. Register Lock Bit Text File Template [\(Ask a Question\)](#)

An initial Configuration Lock Bit file can be generated from the Design Flow window by clicking **Generate FPGA Array Data**.

The initial (default) file is named <proj_location>/designer/<root>/<root>_init_config_lock_bits.txt. Edit this file to ensure that the lock bits are set to "0" for all register bits you want to lock. After editing the file, save it as a *.txt file with a different name, and then import the file into the project using the Register Lock Bit Settings dialog box (**Design Flow window > Configure Register Lock Bits**).

6.5.2. Register Lock Bit File Syntax [\(Ask a Question\)](#)

A valid entry in the Lock Bit Configuration file is defined as a <lock_parameters> <lock bit value> pair.

- If the lock bit is for a register, the parameter name is defined as: <Physical block name>_<register name>_LOCK
- If the lock bit is for a field, the parameter name is defined as: <Physical block name>_<register name>_<field name>_LOCK

The physical block name can vary with device family and device (see the following table).

Table 6-9. Physical Block Names

Family	Name
SmartFusion 2 and IGLOO 2 devices	• MSS • FDDR • SERDES_IF_x (where x is 0, 1, 2, or 3 to indicate the physical SERDES location) for SmartFusion 2 and IGLOO 2 010/025/050/150 devices. • SERDES_IF2 060/090 devices (only one SERDES block per device for SmartFusion 2 and IGLOO 2 devices).
RTG4 devices	• FDDR_E • FDDR_W • PCIE_x (where x is 0,1 to indicate the physical SERDES location) • NPSSx (where x is 0,1,2,3 to indicate the physical SERDES location)

Setting the lock bit value to '1' indicates that the register can be written. Setting it to '0' indicates that the register cannot be written (locked). Lines starting with '#' or ';' are comments. Empty lines are permitted in the Lock Bit Configuration file.

Figure 6-5. Lock Bit Configuration File

```
# Register Lock Bits Configuration File for MSS, SERDES(s) and Fabric DOR
# Microsemi Corporation - Microsemi Libero Software Release v11.7 SP1 (Version 11.7.1.2)
# Date: Tue Mar 29 13:24:54 2016

# sb_sb_0/sb_sb_MSS_0/MSS_ADLIB_INST/INST_MSS_050_IP
MSS_ESRAM_CONFIG_LOCK      0
MSS_ESRAM_MAX_LAT_LOCK     1
MSS_DDR_CONFIG_LOCK        1
MSS_ENVM_CONFIG_LOCK       0
MSS_ENVM_REMAP_BASE_LOCK   1
MSS_ENVM_FAB_REMAP_LOCK    1
MSS_CC_CONFIG_LOCK         0
MSS_CC_CACHEREGION_LOCK    1
MSS_CC_LOCKBASEADDR_LOCK   1
MSS_CC_FLUSHINDX_LOCK      0
MSS_DDRB_BUF_TIMER_LOCK    1
MSS_DDRB_NB_ADR_LOCK       1
MSS_DDRB_NB_SIZE_LOCK      0
MSS_DDRB_CONFIG_LOCK       1
MSS_EDAC_ENABLE_LOCK       1
MSS_MASTER_WEIGHT_CONFIG0_LOCK 1
MSS_MASTER_WEIGHT_CONFIG1_LOCK 1
MSS_SOFT_INTERRUPT_LOCK    1
MSS_SOFTRESET_ENVM0_SOFTRESET_LOCK 1
MSS_SOFTRESET_ENVM1_SOFTRESET_LOCK 1
MSS_SOFTRESET_ESRAM0_SOFTRESET_LOCK 1
MSS_SOFTRESET_ESRAM1_SOFTRESET_LOCK 1
MSS_SOFTRESET_MAC_SOFTRESET_LOCK 1
MSS_SOFTRESET_PDMA_SOFTRESET_LOCK 1
MSS_SOFTRESET_TIMER_SOFTRESET_LOCK 1
MSS_SOFTRESET_MMUART0_SOFTRESET_LOCK 1
MSS_SOFTRESET_MMUART1_SOFTRESET_LOCK 1
MSS_SOFTRESET_G4SPI0_SOFTRESET_LOCK 1
MSS_SOFTRESET_G4SPI1_SOFTRESET_LOCK 1
MSS_SOFTRESET_I2C0_SOFTRESET_LOCK 1
MSS_SOFTRESET_I2C1_SOFTRESET_LOCK 1
MSS_SOFTRESET_CAN_SOFTRESET_LOCK 1
MSS_SOFTRESET_USB_SOFTRESET_LOCK 1
MSS_SOFTRESET_COMBLK_SOFTRESET_LOCK 1
MSS_SOFTRESET_FPGA_SOFTRESET_LOCK 1
MSS_SOFTRESET_HPDMA_SOFTRESET_LOCK 1
MSS_SOFTRESET_FIC32_0_SOFTRESET_LOCK 1
MSS_SOFTRESET_FIC32_1_SOFTRESET_LOCK 1
MSS_SOFTRESET_MSS_GPIO_SOFTRESET_LOCK 1
MSS_SOFTRESET_MSS_GPOUT_7_0_SOFT_RESET_LOCK 1
MSS_SOFTRESET_MSS_GPOUT_15_8_SOFT_RESET_LOCK 1
MSS_SOFTRESET_MSS_GPOUT_23_16_SOFT_RESET_LOCK 1
MSS_SOFTRESET_MSS_GPOUT_31_24_SOFT_RESET_LOCK 1
MSS_SOFTRESET_MDDR_CTLR_SOFTRESET_LOCK 1
MSS_SOFTRESET_MDDR_FIC64_SOFTRESET_LOCK 1
MSS_M2_FABRIC_LOCK         1
```

6.5.3. Validating the Register Lock Bits Configuration File [\(Ask a Question\)](#)

During Map File generation (**Design Flow window > Generate FPGA Array Data**), Libero SoC validates the Register Lock Bit Configuration file and displays the following error messages if it encounters an issue.

Table 6-10. Error Messages

Error Message	Description
Error: Invalid parameter name '<param>' while reading register lock bits configuration file <file name>	Libero SoC found an invalid parameter name in the file specified.
Error: Invalid value '<value>' for parameter '<param>' while reading register lock bits configuration file <file name>	Libero SoC found an invalid parameter value in the file specified.
Error: Parameter '<param>' cannot be set to '1', while reading register lock bits configuration file <file name>	The value of SERDES register fields *K_BRIDGE_SPEED is set by the SERDES Configurator to "0" and cannot be changed. It is illegal to change the value to "1".
Error: Parameter '<param>' cannot be set to '1', while reading register lock bits configuration file <file name>	The value of SERDES register fields *K_BRIDGE_SPEED is set by the SERDES Configurator to "0" and cannot be changed. It is illegal to change the value to "1".

6.6. Constraint Flow in Implementation [\(Ask a Question\)](#)

6.6.1. Design State Invalidation [\(Ask a Question\)](#)

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.

Table 6-11. Design Flow Window Status Icons and Tooltips

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 complete 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 (For example, for the Compile Netlist step, not all I/Os are 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 PolarFire FPGA Timing Constraints User Guide on how to resolve the timing violations.

6.6.2. Constraints and Design Invalidation [\(Ask a Question\)](#)

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 (* .cxx files) – low-level design units that might 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

Table 6-12. Design Flow Project Settings

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 suspend 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.

6.6.3. Check Constraints [\(Ask a Question\)](#)

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 (For example, constraints

on a design object which does not exist in the netlist) are flagged as errors and reported in the *_sdc.log file.

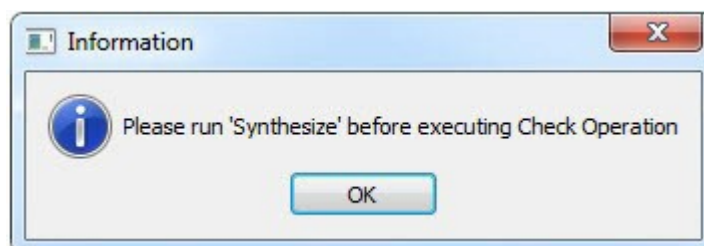
6.6.4. Design State and Constraints Check [\(Ask a Question\)](#)

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



6.6.5. Edit Constraints [\(Ask a Question\)](#)

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 following table) 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.

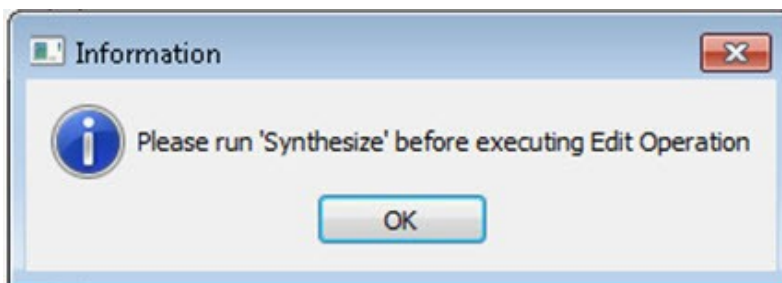
Constraints in the SDC constraint file that are read by the Constraint Editor and subsequently modified by you will be written back to the SDC file when you save the edits and close the Constraint Editor.

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.

6.6.6. Constraint Type and Interactive Tool [\(Ask a Question\)](#)

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 starts, 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.

6.7. Place and Route [\(Ask a Question\)](#)

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

6.7.1. Place and Route Options [\(Ask a Question\)](#)

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 6-7. Layout Options Dialog Box

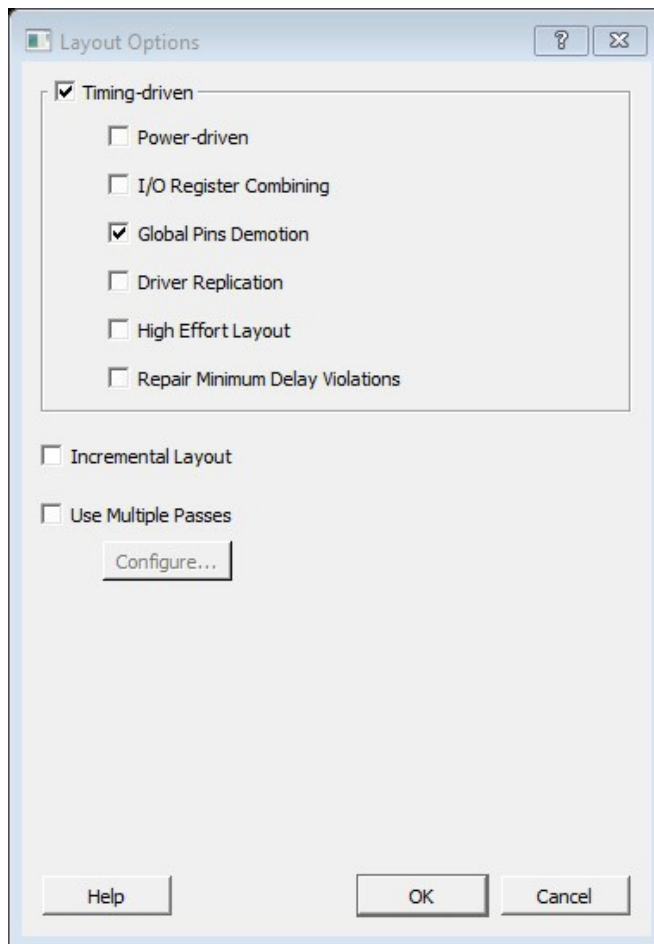
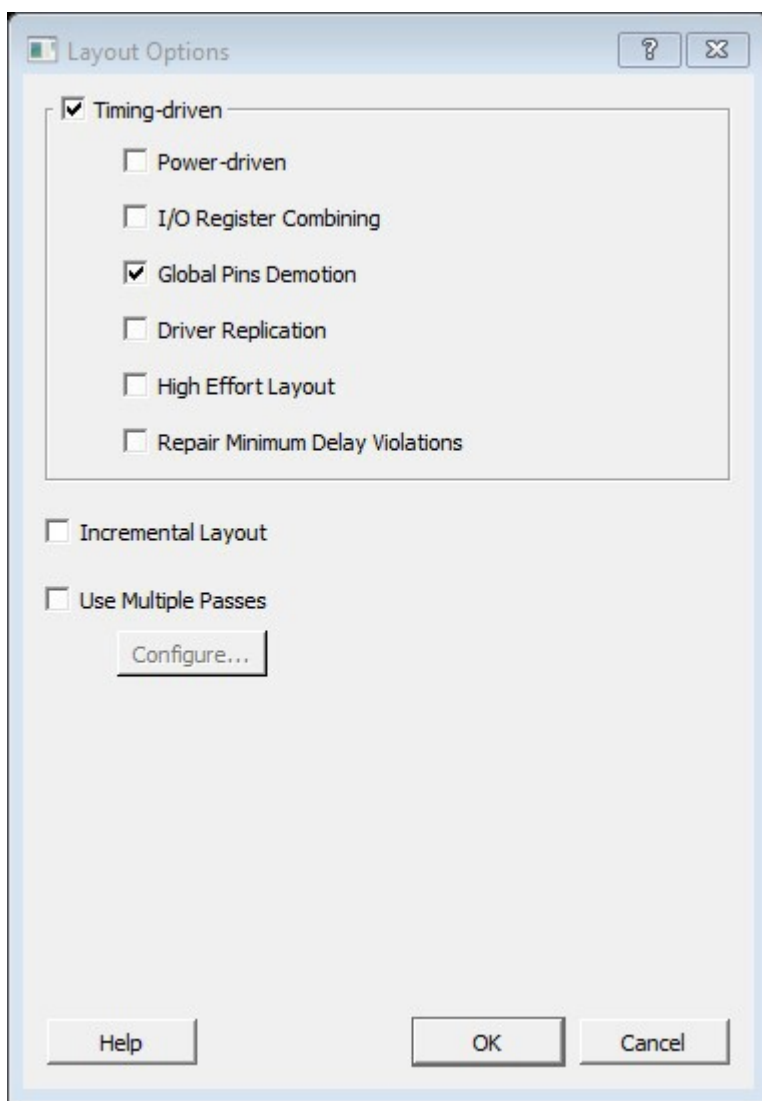


Figure 6-8. Layout Options Dialog Box with Block Flow Enabled



The following table describes the place and route options.

Table 6-13. 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	This option is available when Timing-Driven is checked. This option runs the Power-Driven layout. This layout reduces dynamic power while still maintaining timing constraints.

Table 6-13. Place and Route Options (continued)

Option	Description
I/O Register Combining	This option is available when Timing-Driven is checked. This option 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 available when Timing-Driven is checked. 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 must be used in place of the original register in any specified timing constraint.
Driver Replication	This option is available when Timing-Driven is checked. This option 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 must be used in place of the original register in any specified timing constraint.
High Effort Layout	This option is available when Timing-Driven is checked. This option improves layout success, but increases layout runtime and can impact timing performance.

Table 6-13. Place and Route Options (continued)

Option	Description
Repair Minimum Delay Violations	This option is available when Timing-Driven is checked. This option 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. 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. 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. Even when this option is enabled, it will not repair a connection or path that: • Is a hardwired, preserved, or global net. • Has a sink pin, which is a clock pin. • Violates a maximum delay constraint (that is, the maximum delay slack for the pin is negative). • Can cause the maximum delay requirement for the sink pin to be violated (setup violations). Typically, this option is enabled with the Incremental Layout option when a design's maximum delay requirements are satisfied. Every effort is made to avoid creating max-delay timing violations on worst case paths. Min Delay Repair generates a report in the implementation directory that lists all the paths that were considered. If your design continues to have internal hold time violations, rerun repair Minimum Delay Violations with Incremental Layout to analyze additional paths.
Incremental Layout	Uses previous placement data as the initial placement for the next run. 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 Synopsys FPGA Synthesis Pro ME User Guide.
Use Multiple Pass	Runs multiple passes of Place and Route to achieve the best layout result. Click Configure to specify the criteria you want to use to determine the best layout result.

Table 6-13. Place and Route Options (continued)

Option	Description
Block Creation	This option is available only when the Block Creation option is turned on (Project > Project Settings > Design Flow > Enable Block Creation). 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.

6.8. Multiple Pass Layout Configuration [\(Ask a Question\)](#)

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.

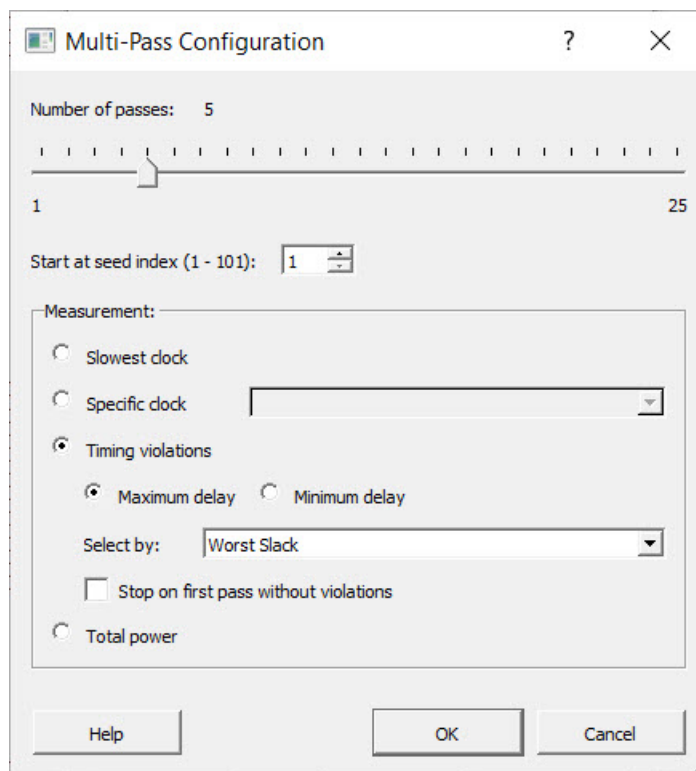
When using Multiple Pass Layout, observe the following guidelines:

- 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 must save your design file with a different name before proceeding. To do this, select **File > Save As**.
- The following 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 UI control in the Layout Options dialog box.
2. Click **Configure**. The Multi-Pass Configuration dialog box appears.

Figure 6-9. Multi-Pass Configuration Dialog Box



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

Table 6-14. 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 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 each 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. Note: 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.
Minimum delay	Examines timing violations (slacks) obtained from minimum delay analysis.

Table 6-14. Multi-Pass Configuration Options (continued)

Option	Description
Select by	Slack criteria. Choices are: - 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>) - 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. - Stop on first pass without violations: Stops performing remaining passes if all timing constraints are met when there are no negative slacks reported in the timing violations report.
Total power	Determines the best pass to be the one that has the lowest total power (static + dynamic) of all layout passes.

6.8.1. Iteration Summary Report [\(Ask a Question\)](#)

The file `<root_module>_iteration_summary.rpt` records a summary of whether the multiple pass run was started through the GUI or the `extended_run_lib` Tcl script, with arguments for repeating each run. Each new run appears with its own header in the Iteration Summary Report along with the 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 6-10. Iteration Summary Report

```

1  #
2  # RUN NUMBER: 1  DATE: 15:21:35 04-Mar-2015
3  # INVOKED AS: W:\po\11_5_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 N/A Compare failed shift_reg32_timing_violations_max_r1_s1.rpt shift_reg32_r1_s1
8  1 86662958 N/A Layout failed shift_reg32_timing_violations_max_r1_s2.rpt shift_reg32_r1_s2
9  2 8988747 N/A Layout failed shift_reg32_timing_violations_max_r1_s3.rpt shift_reg32_r1_s3
10 3 51071856 N/A Layout failed shift_reg32_timing_violations_max_r1_s4.rpt shift_reg32_r1_s4
11 4 78381505 N/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_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 N/A -3.359 shift_reg32_timing_violations_max_r2_s6.rpt shift_reg32_r2_s6
19 6 23702026 N/A -3.174 shift_reg32_timing_violations_max_r2_s7.rpt shift_reg32_r2_s7
20 7 51370950 N/A -3.413 shift_reg32_timing_violations_max_r2_s8.rpt shift_reg32_r2_s8
21 8 93189207 N/A -3.168 shift_reg32_timing_violations_max_r2_s9.rpt shift_reg32_r2_s9
22 9 17538078 N/A -3.354 shift_reg32_timing_violations_max_r2_s10.rpt shift_reg32_r2_s10
23 #

```

6.9. Post Layout Editing of I/O Signal Integrity and Delay Parameters [\(Ask a Question\)](#)

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 select 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 Commands User Guide for PolarFire FPGA](#).

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 a delay parameter can be edited by this tool.

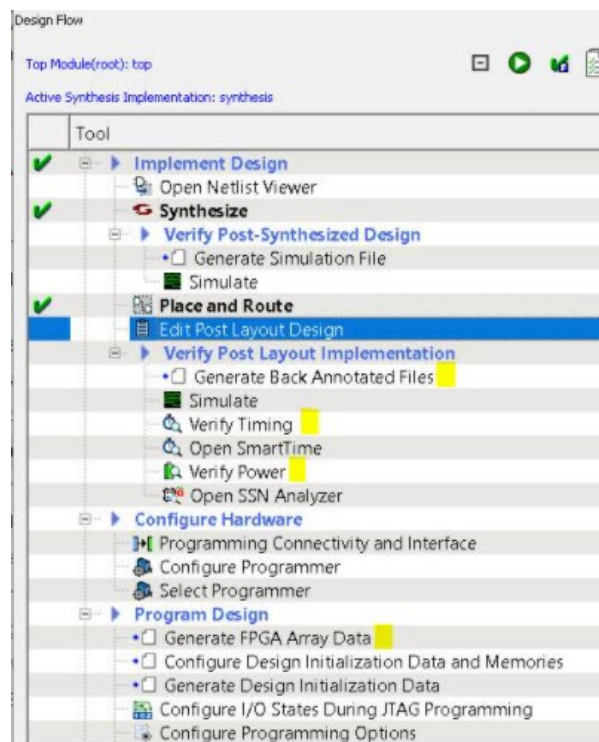
The batch command `edit_post_layout_design` fails when any commands in the input PDC file fail. The PDC commands fail if the syntax is incorrect, the referenced instances do not exist, or the values are out of legal ranges. If the batch command fails, the layout state of the design does not change. If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

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

Figure 6-11. Edit Post Layout Design Tool in Design Flow



6.10. Resource Usage [\(Ask a Question\)](#)

After completing the layout of your design, you can assess the resource usage to understand how efficiently your design utilizes the available resources on the chip. The **Resource Usage** report provides an overview of the types and percentages of resources used relative to the total resources available on the chip in a tabular format.

To access the report:

1. Navigate to **Design > Reports**.
2. Choose `<design_name>_layout_log.log` file.



Important: Numbers in the [Resource Usage Example Report](#) table are generated as an example for metric demonstration only.

Table 6-15. Resource Usage Example Report

Type	Used	Total	Percentage
4LUT	0	299544	0.00
DFF	0	299544	0.00
Logic Element	0	299544	0.00
I/Os using I/O Registers	263	512	51.37
I/O Register Flip-Flops	789	1536	51.37
-- Input I/O Flip-Flops	263	512	51.37
-- Output I/O Flip-Flops	263	512	51.37
-- Enable I/O Flip-Flops	263	512	51.37

Type Specifies the category of the resource being reported. It includes the following:

4LUT 4-input Look-up Table for executing any logic function involving up to four inputs. In the report, the number of 4-input lookup tables (4LUTs) indicates how many are used in the design, regardless of whether they are combined with D Flip-Flops (DFFs).

DFF D Flip-Flop stores individual bits of data. The report shows the total number of DFFs used in the design, irrespective of their combination with 4LUTs.

Logic Element A logic element (LE) is a unit in the fabric that might contain a 4LUT, a DFF, or both. The report includes all LEs, whether they contain only 4LUTs, only DFFs, or both.

I/Os using I/O Registers Refers to the input/output resources that utilize I/O registers. These registers manage data flow between the internal logic and the external environment.

I/O Register Flip-Flops Flip-flops specifically used in I/O operations. They are part of the I/O registers and help in managing the data flow for I/O operations.

Input I/O Flip-Flops Flip-flops are used for input operations such as capturing data from external sources.

Output I/O Flip-Flops Flip-flops are used for output operations such as driving data to external destinations.

Enable I/O Flip-Flops Flip-flops enable or disable I/O operations. They control the activation of input/output operations, providing fine-tuned management of data flow to and from the chip.

Used Indicates the number of resources of each type that are currently utilized in the design. It provides a direct count of how many 4LUTs, DFFs, LEs, or I/O resources are used.

Total Shows the total number of each resource type available on the chip. It represents the maximum capacity of the selected device for each resource type highlighting the limits within which your design must fit.

Percentage Calculates the percentage of each resource type that is used relative to the total available resources. It is computed as $(\text{Used}/\text{Total}) \times 100$.

6.10.1. Overlapping of Resource Reporting [\(Ask a Question\)](#)

The [Resource Usage](#) report contains the total number of used 4LUTs and DFFs, regardless of whether they are combined within LEs. The **Overlapping of Resource Reporting** provides detailed breakdown of LEs' utilization.

General Formulas for Logic Element Utilization

This section provides a set of general formulas that can be applied to calculate the usage of LEs in any design. Use these formulas to analyze how different components, such as 4LUTs and DFFs, are distributed across the available LEs.

Total LEs Used	Total LEs Used = LEs using both 4LUTs and DFFs + LEs using 4LUTs only or DFFs only
LEs Using 4LUT Only	LEs Using 4LUT Only = Total LEs used – LEs containing DFFs This formula calculates the number of LEs that contain only 4LUTs, excluding the LEs that also contain DFFs.
LEs Using DFF Only	LEs Using DFF Only = Total LEs used – LEs containing 4LUTs This formula calculates the number of LEs that contain only DFFs, excluding the LEs that also contain 4LUTs.
LEs Using Both 4LUT and DFF	LEs Using Both 4LUTs and DFFs = (LEs containing 4LUTs + LEs containing DFFs) – Total LEs used
LEs Using 4LUTs Only or DFFs Only	LEs Using 4LUTs only and DFFs only = LEs using DFFs only + LEs using 4LUTs only

These formulas are applied to different reports to calculate the usage of 4LUTs and DFFs resources within Logic Elements.

Example Application of Logic Element Utilization

To illustrate the practical use of the general formulas for Logic Element utilization, this section presents a detailed example.

 **Important:** Numbers for 4LUTs, DFFs and LEs in **Overlapping of Resource Reporting** are generated as an example to demonstrate the calculations only.

The following example report provides a breakdown of how many of each type of resource is used in the design:

Used 4LUT = 400

400 out of the 473 LEs have 4LUTs inside which is the total number of 4LUTs used in the design, irrespective of whether they are combined with DFFs. This means 73 (473 - 400) of them have no 4LUTs.

DFF = 300

300 out of the 473 LEs have DFFs inside which is the total number of DFFs used, regardless of their combination with 4LUTs. This means 173 (473 - 300) of them have no DFFs.

Logic Elements = 473

It details the number of LEs used, whether they contain only 4LUTs, only DFFs, or both.

The following table summarizes the calculations for this example:

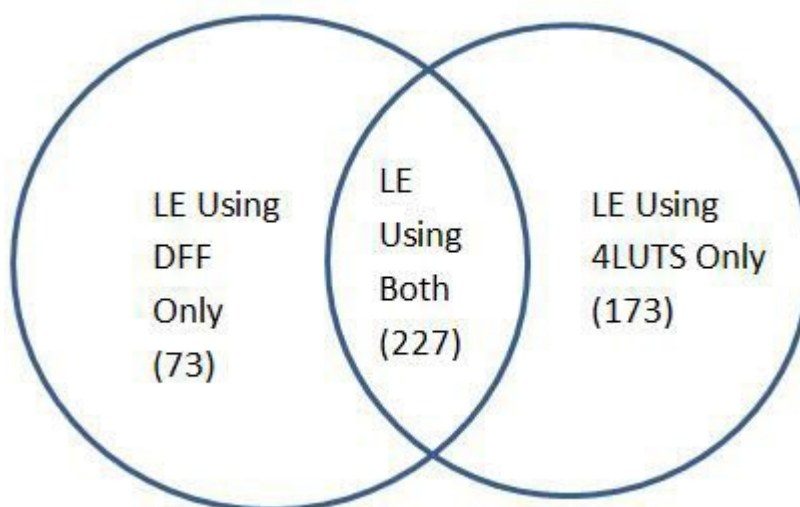
Table 6-16. Logic Element Calculations Summary

Logic Element Description	Calculated Logic Elements
LEs using DFF only	Calculated as the used LEs minus those containing 4LUTs: 473 - 400 = 73

Table 6-16. Logic Element Calculations Summary (continued)

Logic Element Description	Calculated Logic Elements
LEs using 4LUTs only	Calculated as the used LEs minus those containing DFFs: $473 - 300 = 173$
LEs using 4LUTs only or DFFs only	Calculated as the sum of LEs using only one type of resource: $73 + 173 = 246$
LEs using both 4LUTs and DFFs	The overlap between LEs containing 4LUTs and DFFs, calculated as the sum of used 4LUT and used DFF minus the total used LEs: $(400 + 300) - 473 = 227$
Total LEs used	The total number of Logic Elements used: 473

In the following figure, the overlapping area in a Venn diagram represents the LEs that utilize both 4LUTs and DFFs within the design.

Figure 6-12. Resource Utilization in Logic Elements

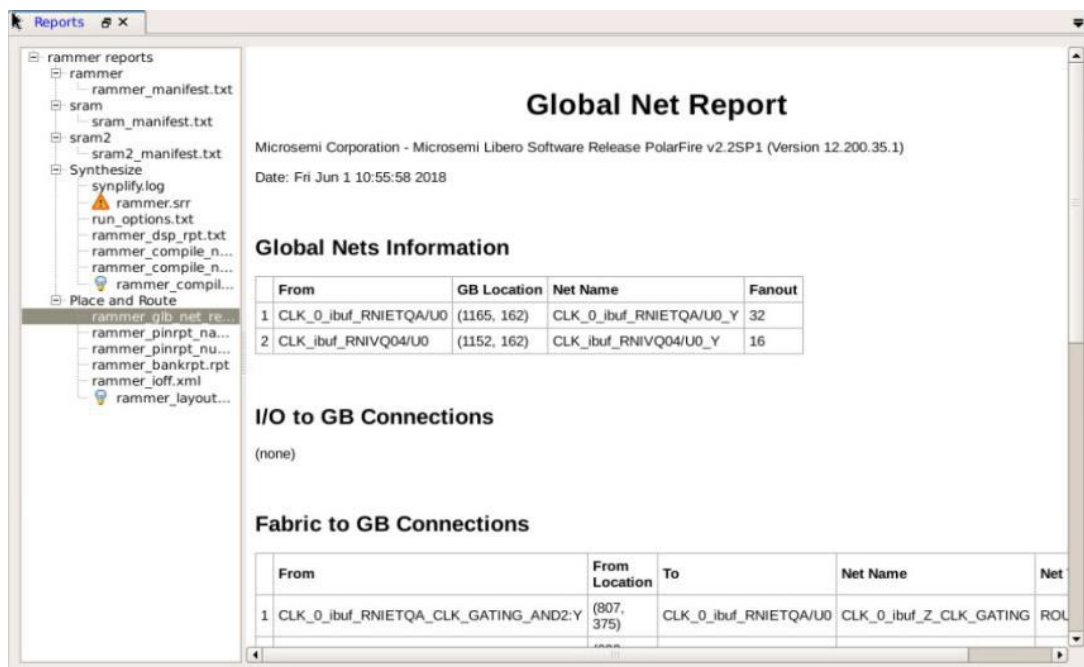
6.11. Global Net Report [\(Ask a Question\)](#)

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 6-13. Global Net Report



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

6.11.1. Global Nets Information [\(Ask a Question\)](#)

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 6-14. 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_GEast	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_RNIHDI4/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_GEast	149
15	GB[9]	(735, 156)	SRAM_CQ_ibuf_RNIB7IC/U0_YWn_GEast	111
16	GB[12]	(738, 156)	SRAM_CQn_ibuf_RNIPMM6/U0_YWn_GEast	18

6.11.2. I/O to GB Connections [\(Ask a Question\)](#)

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

Figure 6-15. 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_CLKI3	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_CLKI2	SRAM_CQn_ibuf/U0/U_JOIN.Y	North IO #6 (1002, 313)	GB[12]	SRAM_CQn_ibuf	HARDWIRED	1

Table 6-17. Columns in the I/O to GB 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 function name (DDRIO120PB2/MDDR_DQ_ECC1/GB12/CCC_NE1_CLKI2) contains the GB index (GB12 in the preceding example) that matches the GB index in the To column (GBL[12] in the preceding example). - For routed connections, the function name does not contain the proper GB index.

Table 6-17. Columns in the I/O to GB Connections Section (continued)

Column	Description
Net Type	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 hardwired nets) and vary from iteration to iteration.

6.11.3. Fabric to GB Connections [\(Ask a Question\)](#)

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.

6.11.4. CCC to GB Connections [\(Ask a Question\)](#)

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).

6.11.5. CCC Input Connections [\(Ask a Question\)](#)

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.

Table 6-18. Columns in the CCC Input Connections Section

Column	Description
I/O Function	I/O connection details, such as 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 preceding figure).
To (Pin Swapped for Back Annotation Only)	For hardwired connections, input pin of the CCC in the back annotated netlist.

6.11.6. Local Clock Nets to RGB Connections [\(Ask a Question\)](#)

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 6-19. 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.

6.11.7. Global Clock Nets to RGB Connections [\(Ask a Question\)](#)

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 6-20. 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 Fanout column gives the total fanout of the net.
Local Fanout	Fanout local to RGB.

6.11.8. Warnings (RTG4 only) [\(Ask a Question\)](#)

The Warning section is available in RTG4 devices only. It gives warnings about clock or reset nets which are not radiation protected and recommends ways to protect the nets from radiation. Some warning examples are:

- Clocks or reset nets that are routed are not radiation protected.

- Hardwired connections from DDRIO bank are not radiation protected.
- For radiation protection, Microchip recommends using dedicated global clocks that come with built-in radiation protection.

Figure 6-18. Warning Example 1

The following clocks or resets are driven by fabric-generated or local nets and are not radiation protected:

From	From Location	I/O Bank	To	Net Name	Fanout
1 clk16_ibuf/U0/U_JOIN.Y	(3, 151)	MSIOD	GBR[16]	clk16_ibuf_Z	1
2 clk16out_ibuf/U0/U_JOIN.Y	(8, 151)	MSIOD	GBR[14]	clk16out_ibuf_Z	1
3 clk16_a_c_ibuf/U0/U_JOIN.Y	(1524, 145)	MSIOD	GBR[15]	clk16_a_c_ibuf_Z	2

- Local resets that are not driven by three separate logic cones are not radiation protected.

Figure 6-19. Warning Example 2

The following local resets are not driven by three separate logic cones and are not radiation protected:

From	From Location	To	Net Name	Fanout
1 I_ARRAY_CORE_ABC/I_DATABUS/I_DB_CONTROL/un1_resetfifol.Y	(1313, 201)	RGRESET	I_ARRAY_CORE_ABC/un1_resetfifol_i	10
2 I_ARRAY_CORE_DEF/I_DATABUS/I_DB_CONTROL/un1_resetfifol.Y	(1394, 228)	RGRESET	I_ARRAY_CORE_DEF/un1_resetfifol_i	10
3 I_ARRAY_CORE_ABC/I_TGSLICE_CMP/I_C1_RESIDUE_FIFO/comb_reset.Y	(1176, 300)	RGRESET	I_ARRAY_CORE_ABC/I_TGSLICE_CMP/I_C1_RESIDUE_FIFO/comb_reset_Z	1
4 I_ARRAY_CORE_ABC/I_TGSLICE2/I_D2_RESIDUE_FIFO/comb_reset.Y	(1179, 291)	RGRESET	I_ARRAY_CORE_ABC/I_TGSLICE2/I_D2_RESIDUE_FIFO/comb_reset_Z	1
5 I_ARRAY_CORE_ABC/I_TGSLICE2/I_MATCH_FIFO/un1_resetoutl.Y	(1312, 201)	RGRESET	I_ARRAY_CORE_ABC/I_TGSLICE2/I_MATCH_FIFO/un1_resetoutl_i	1
6 I_ARRAY_CORE_ABC/I_TGSLICE/I_MATCH_FIFO/un1_resetoutl.Y	(1314, 201)	RGRESET	I_ARRAY_CORE_ABC/I_TGSLICE/I_MATCH_FIFO/un1_resetoutl_i	1
7 I_ARRAY_CORE_ABC/I_TGSLICE_CMP/I_MATCH_FIFO_CMP/un1_resetoutl.Y	(1309, 201)	RGRESET	I_ARRAY_CORE_ABC/I_TGSLICE_CMP/I_MATCH_FIFO_CMP/un1_resetoutl_i	1
8 I_ARRAY_CORE_ABC/I_OUTPUT_FIFO_WR_CTL/un1_resetsynci.Y	(1191, 240)	RGRESET	I_ARRAY_CORE_ABC/I_OUTPUT_FIFO_WR_CTL/un1_resetsynci_i	1
9 I_ARRAY_CORE_DEF/I_TGSLICE_CMP/I_C1_RESIDUE_FIFO/comb_reset.Y	(1191, 288)	RGRESET	I_ARRAY_CORE_DEF/I_TGSLICE_CMP/I_C1_RESIDUE_FIFO/comb_reset_Z	1
10 I_ARRAY_CORE_DEF/I_TGSLICE/I_MATCH_FIFO/un1_resetoutl.Y	(1178, 252)	RGRESET	I_ARRAY_CORE_DEF/I_TGSLICE/I_MATCH_FIFO/un1_resetoutl_i	1
11 I_ARRAY_CORE_DEF/I_TGSLICE_CMP/I_MATCH_FIFO_CMP/un1_resetoutl.Y	(1179, 252)	RGRESET	I_ARRAY_CORE_DEF/I_TGSLICE_CMP/I_MATCH_FIFO_CMP/un1_resetoutl_i	1
12 I_ARRAY_CORE_DEF/I_TGSLICE2/I_MATCH_FIFO/un1_resetoutl.Y	(1181, 252)	RGRESET	I_ARRAY_CORE_DEF/I_TGSLICE2/I_MATCH_FIFO/un1_resetoutl_i	1
13 I_ARRAY_CORE_DEF/I_OUTPUT_FIFO_WR_CTL/un1_resetsynci.Y	(1081, 252)	RGRESET	I_ARRAY_CORE_DEF/I_OUTPUT_FIFO_WR_CTL/un1_resetsynci_i	1
14 I_ARRAY_CORE_DEF/I_TGSLICE2/I_D2_RESIDUE_FIFO/comb_reset.Y	(1179, 255)	RGRESET	I_ARRAY_CORE_DEF/I_TGSLICE2/I_D2_RESIDUE_FIFO/comb_reset_Z	1

For radiation protection, Microsemi recommends that each of the three inputs of every RGRESET be driven by three separate logic cones replicating the paths from the source registers.

- For radiation protection, Microchip recommends that each of the three inputs of every RGRESET be driven by three separate logic cones replicating the paths from the source registers. See the descriptions of RGRESET macro in the [Macro Library User Guide for RTG4](#).

6.12. Verify Post Layout Implementation [\(Ask a Question\)](#)

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

6.12.1. Generate Back Annotated Files [\(Ask a Question\)](#)

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 minimum delays to include the best-case timing results in your back annotated file.

6.12.2. Simulate - Opens QuestaSim ME Pro [\(Ask a Question\)](#)

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 QuestaSim ME Pro. 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 arrange testbenches, use the up and down arrows to define the order you want the testbenches compiled. The top-level entity must be at the bottom of the list.
3. When you are satisfied with the **Associated Simulation Files** list, click **OK**.
4. To start QuestaSim ME Pro, right-click **Simulate** in the Design Hierarchy window and choose **Open Interactively**. QuestaSim ME Pro 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 might receive 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. On completion, from the **File** menu, click **Quit**.

6.12.3. Verify Timing [\(Ask a Question\)](#)

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. For multi-corner analysis, the following operation conditioning corners are considered:

- Max analysis at BEST and WORST operating conditions
- Min analysis at BEST and WORST operating conditions

Although the analysis is done for all corners, Multi Corner Timing Analysis reports show only two conditions. To cross-check this, run the command with the tool.

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

Figure 6-20. Verify Timing Configuration Dialog Box – Report Tab

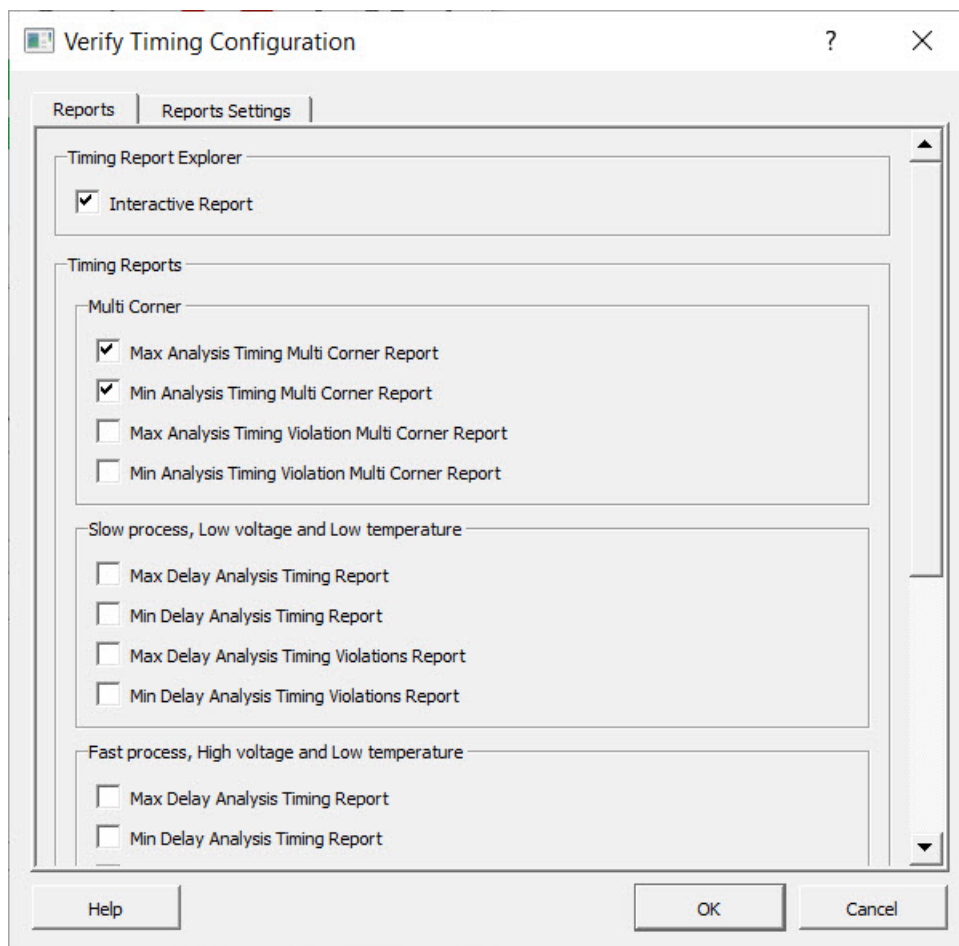
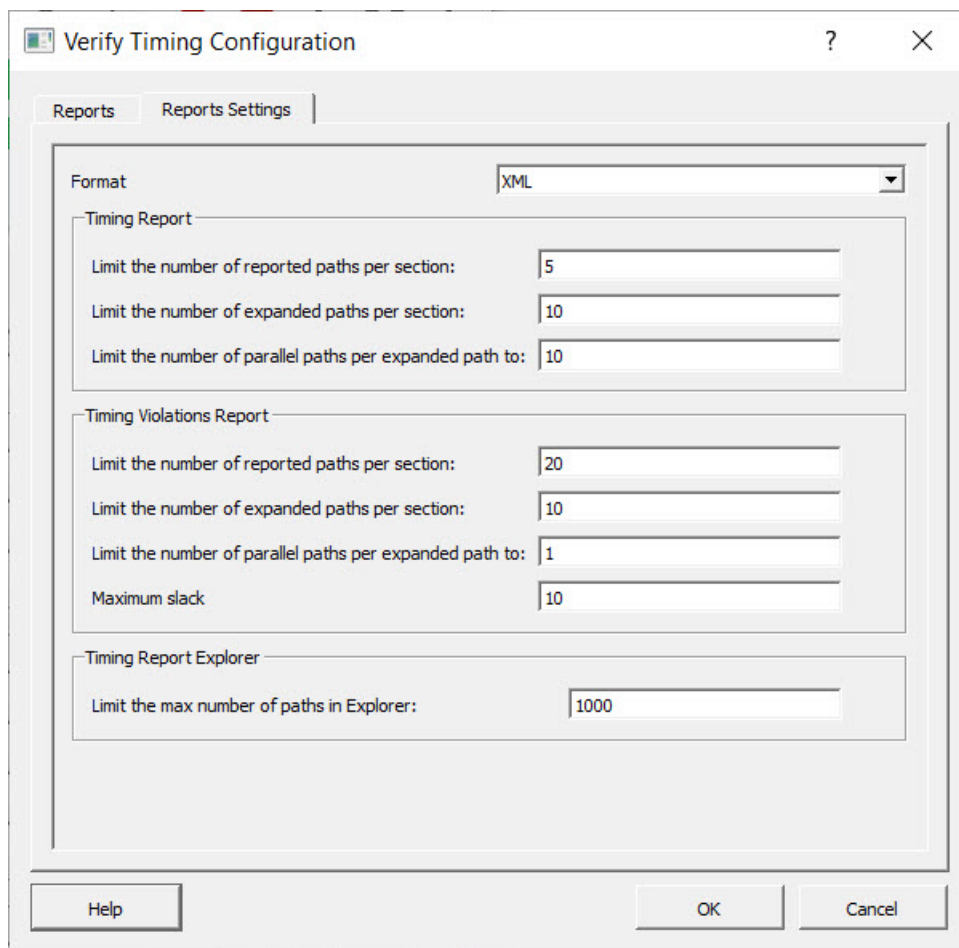


Figure 6-21. Verify Timing Configuration Dialog Box – Report Settings Tab



The image shows a screenshot of the 'Verify Timing Configuration' dialog box, specifically the 'Reports Settings' tab. The dialog has a title bar with a question mark and a close button. Below the title bar are two tabs: 'Reports' and 'Reports Settings', with 'Reports Settings' being the active tab. The main content area is divided into three sections: 'Format', 'Timing Report', and 'Timing Violations Report'. The 'Format' section has a dropdown menu set to 'XML'. The 'Timing Report' section contains three input fields: 'Limit the number of reported paths per section:' (5), 'Limit the number of expanded paths per section:' (10), and 'Limit the number of parallel paths per expanded path to:' (10). The 'Timing Violations Report' section contains four input fields: 'Limit the number of reported paths per section:' (20), 'Limit the number of expanded paths per section:' (10), 'Limit the number of parallel paths per expanded path to:' (1), and 'Maximum slack:' (10). At the bottom of the dialog is a 'Timing Report Explorer' section with one input field: 'Limit the max number of paths in Explorer:' (1000). At the very bottom of the dialog are three buttons: 'Help', 'OK', and 'Cancel'.

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

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

Figure 6-22. Verify Timing Configuration Dialog Box - Tooltip Error

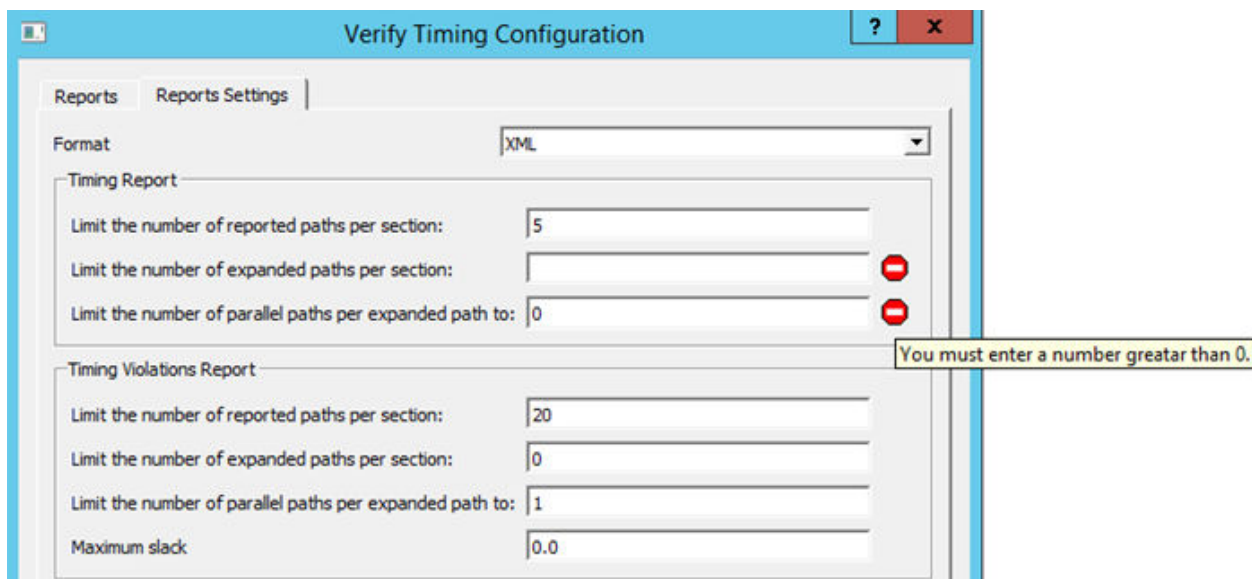
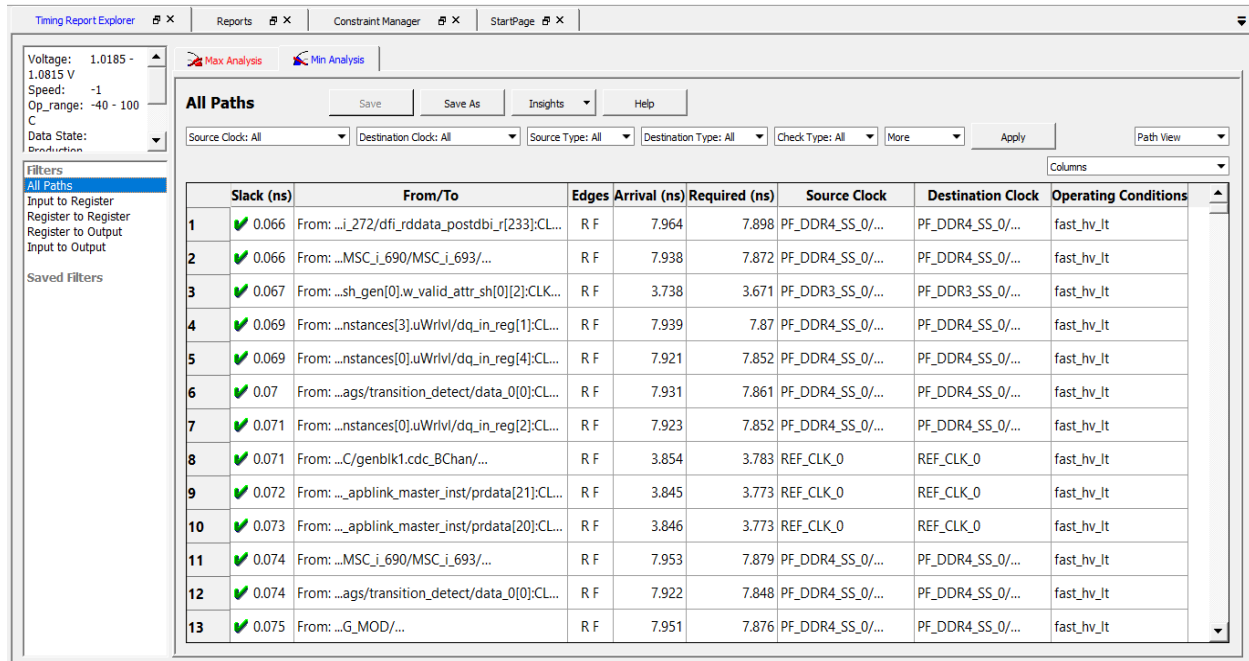


Table 6-21. Verify Timing Configuration Dialog Box Settings

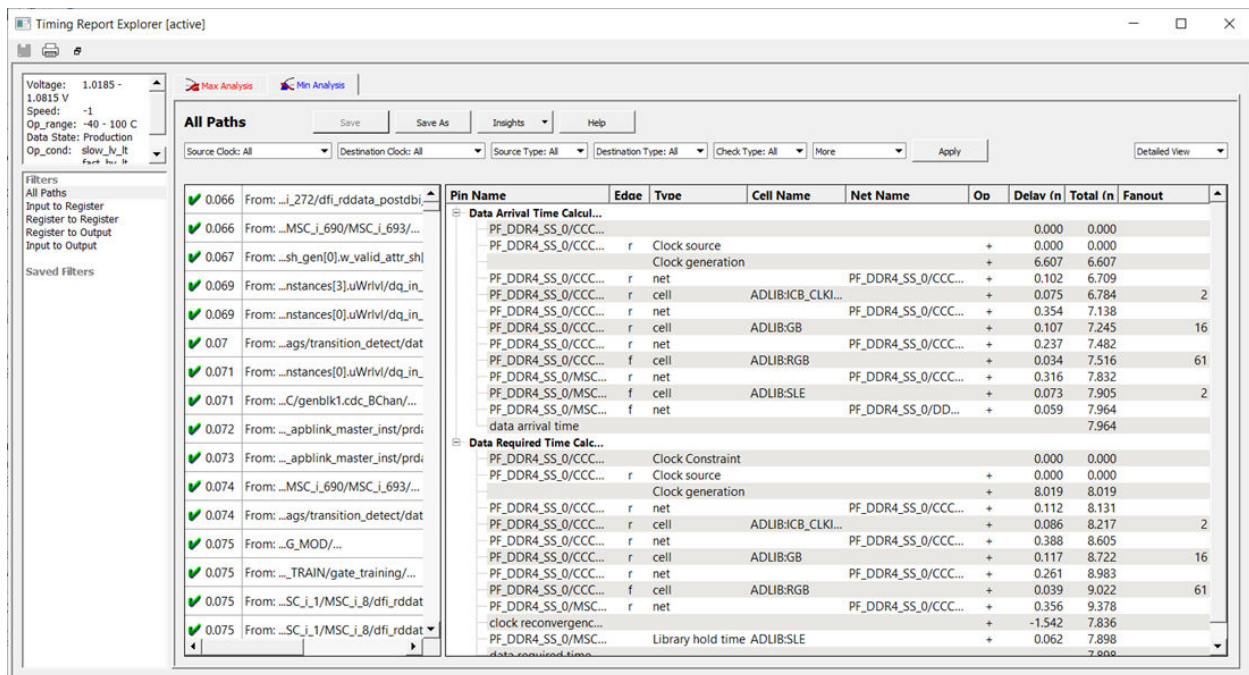
Report	Setting	Description
Timing Report	Limit the number of reported paths per section	Number of reported paths under each section. Range: 1–20000
	Limit the number of expanded paths per section	Number of expanded paths under each section. Range: 1–20000
	Limit the number of parallel paths per expanded 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 Design Tab > Timing Report Explorer . After generating a report, the Timing Report Explorer appears in Path View as shown in the following figure.

Figure 6-23. Timing Report Explorer - Path View



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

Figure 6-24. 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 tooltips as shown in the following figures:

Figure 6-25. Example of Source Tooltip

Slack (ns)	Path	Source Clock	Destination Clock	Source Type	Destination Type
0.066	From: ...i_272/dfi_rddata_postdbi_r[233]:CL...	PF_DDR3_SS_0/OUT1	PF_DDR3_SS_0/OUT1	R	F
0.066	From: ...MSC_i_690/MSC_i_693/...	PF_DDR3_SS_0/OUT1	PF_DDR3_SS_0/OUT1	R	F
0.067	From: ...sh_gen[0].w_valid_attr_sh[0][2]:CLK...	PF_DDR3_SS_0/OUT1	PF_DDR3_SS_0/OUT1	R	F
0.069	From: ...nstances[3].uWr1vl/dq_in_reg[1]:CL...	PF_DDR3_SS_0/OUT1	PF_DDR3_SS_0/OUT1	R	F
0.069	From: ...nstances[0].uWr1vl/dq_in_reg[4]:CL...	PF_DDR3_SS_0/OUT1	PF_DDR3_SS_0/OUT1	R	F
0.07	From: ...ags/transition_detect/data_0[0]:CL...	PF_DDR3_SS_0/OUT1	PF_DDR3_SS_0/OUT1	R	F
0.071	From: ...nstances[0].uWr1vl/dq_in_reg[2]:CL...	PF_DDR3_SS_0/OUT1	PF_DDR3_SS_0/OUT1	R	F

Figure 6-26. Example of Destination Tooltip

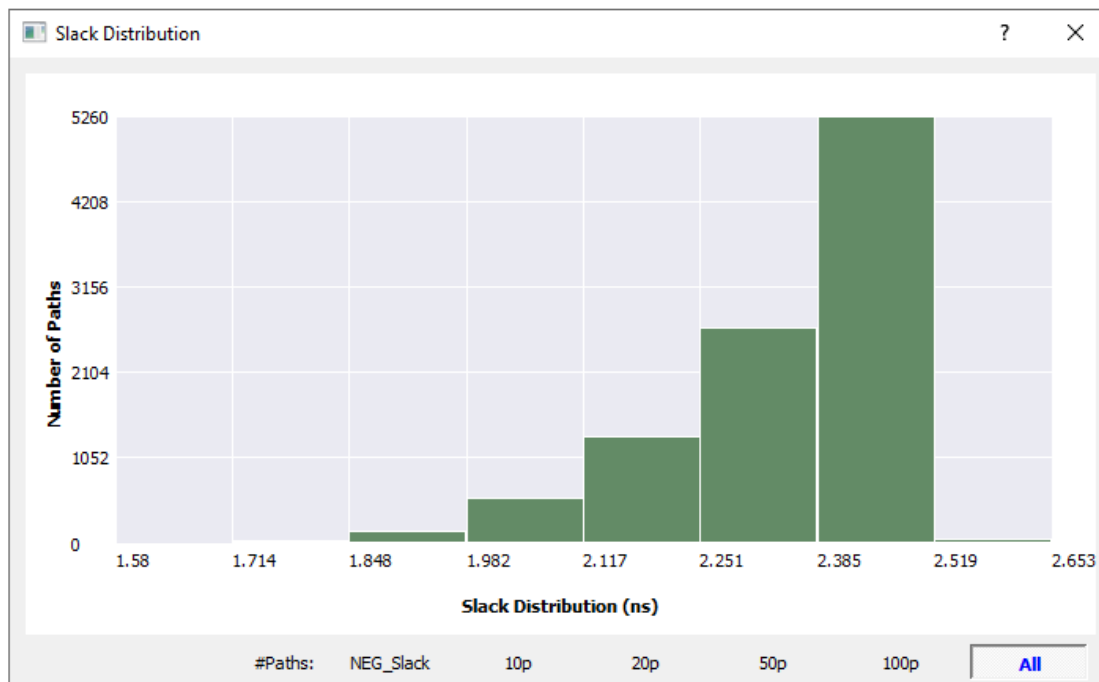
Slack (ns)	Path	Source Clock	Destination Clock	Operating Conditions
0.066	From: ...i_272/dfi_rddata_postdbi_r[233]:CL...	PF_DDR4_SS_0/...	PF_DDR4_SS_0/...	fast_hv_1t
0.066	From: ...MSC_i_690/MSC_i_693/...	PF_DDR4_SS_0/...	PF_DDR4_SS_0/...	fast_hv_1t
0.067	From: ...sh_gen[0].w_valid_attr_sh[0][2]:CLK...	PF_DDR3_SS_0/...	PF_DDR3_SS_0/...	fast_hv_1t
0.069	From: ...nstances[3].uWr1vl/dq_in_reg[1]:CL...	PF_DDR4_SS_0/...	PF_DDR4_SS_0/...	fast_hv_1t
0.069	From: ...nstances[0].uWr1vl/dq_in_reg[4]:CL...	PF_DDR4_SS_0/...	PF_DDR4_SS_0/...	fast_hv_1t
0.07	From: ...ags/transition_detect/data_0[0]:CL...	PF_DDR4_SS_0/...	PF_DDR4_SS_0/...	fast_hv_1t
0.071	From: ...nstances[0].uWr1vl/dq_in_reg[2]:CL...	PF_DDR4_SS_0/...	PF_DDR4_SS_0/...	fast_hv_1t
0.071	From: ...C/genblk1.cdc_BChan/...	REF_CLK_0	REF_CLK_0	fast_hv_1t
0.072	From: ...apblink_master_inst/prdata[21]:CL...	REF_CLK_0	REF_CLK_0	fast_hv_1t
0.073	From: ...apblink_master_inst/prdata[20]:CL...	REF_CLK_0	REF_CLK_0	fast_hv_1t
0.074	From: ...MSC_i_690/MSC_i_693/...	PF_DDR4_SS_0/...	PF_DDR4_SS_0/...	fast_hv_1t
0.074	From: ...ags/transition_detect/data_0[0]:CL...	PF_DDR4_SS_0/...	PF_DDR4_SS_0/...	fast_hv_1t
0.075	From: ...G_MOD/...	PF_DDR4_SS_0/...	PF_DDR4_SS_0/...	fast_hv_1t
0.075	From: ...TRAIN/gate_training/...	PF_DDR4_SS_0/...	PF_DDR4_SS_0/...	fast_hv_1t
0.075	From: ...SC_i_1/MSC_i_8/dfi_rddata_r[72]:CL...	PF_DDR3_SS_0/...	PF_DDR3_SS_0/...	fast_hv_1t

Insights: This option provides an insight into the graphical view of slack distribution across the selected filter, cell vs. net delays, and logic level distribution.

Slack Distribution

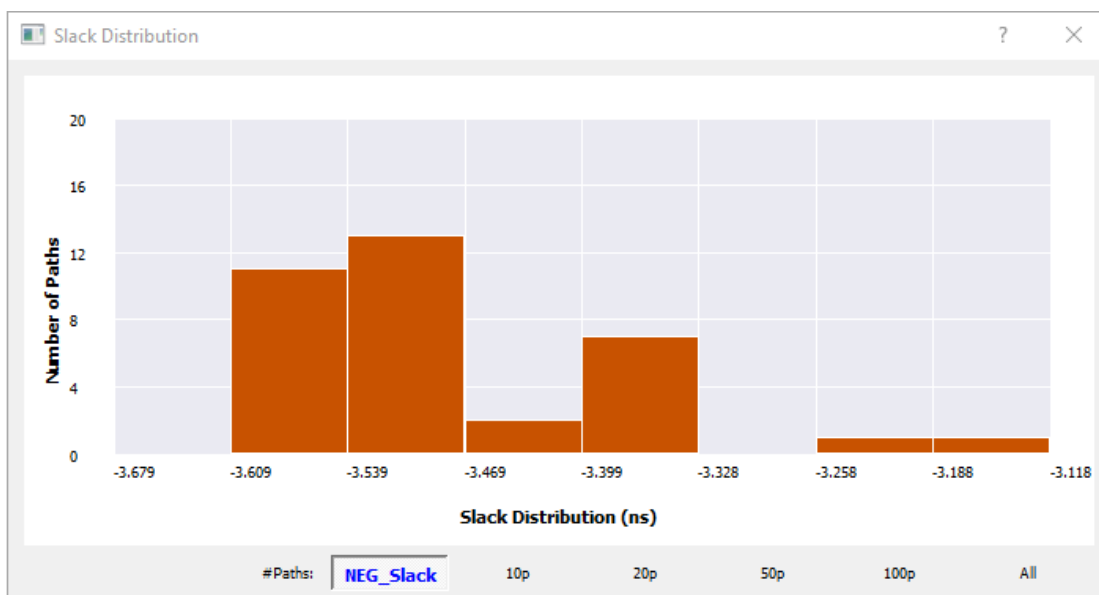
Click **Insights > Slack Distribution** to view the slack distribution. The following figure shows an example of positive slack distribution.

Figure 6-27. Positive Slack Distribution



The following figure shows an example of negative slack distribution.

Figure 6-28. Negative Slack Distribution



Cell vs Net Delays

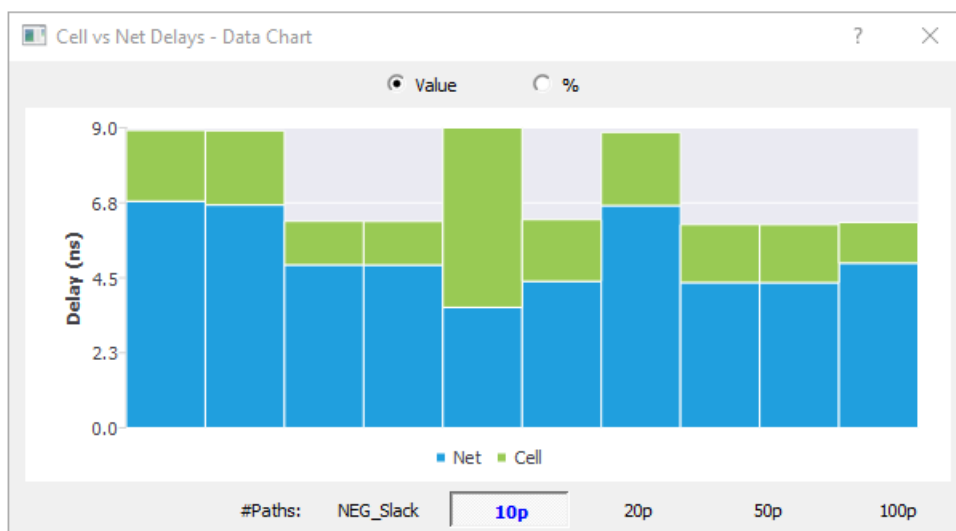
Click **Insights > Cell vs Net Delays** to view a delay breakdown of the top 100 paths in the form of a stacked bar chart, percentage chart, and pie chart.

Stacked Bar Chart

The stacked bar chart shows the net and cell delay for a path. Radio buttons at the top of the chart allows you to toggle between value and percentage (%) views. Buttons at the bottom of the chart

allow you to select the number of paths from the path range: **NEG_Slack**, **10p**, **20p**, **50p**, and **100p** (default is **10p**). Hovering over the bar of a data chart displays a tooltip with From, To, Net Delay, and Cell Delay information.

Figure 6-29. Example of Stacked Bar Chart (Value View)



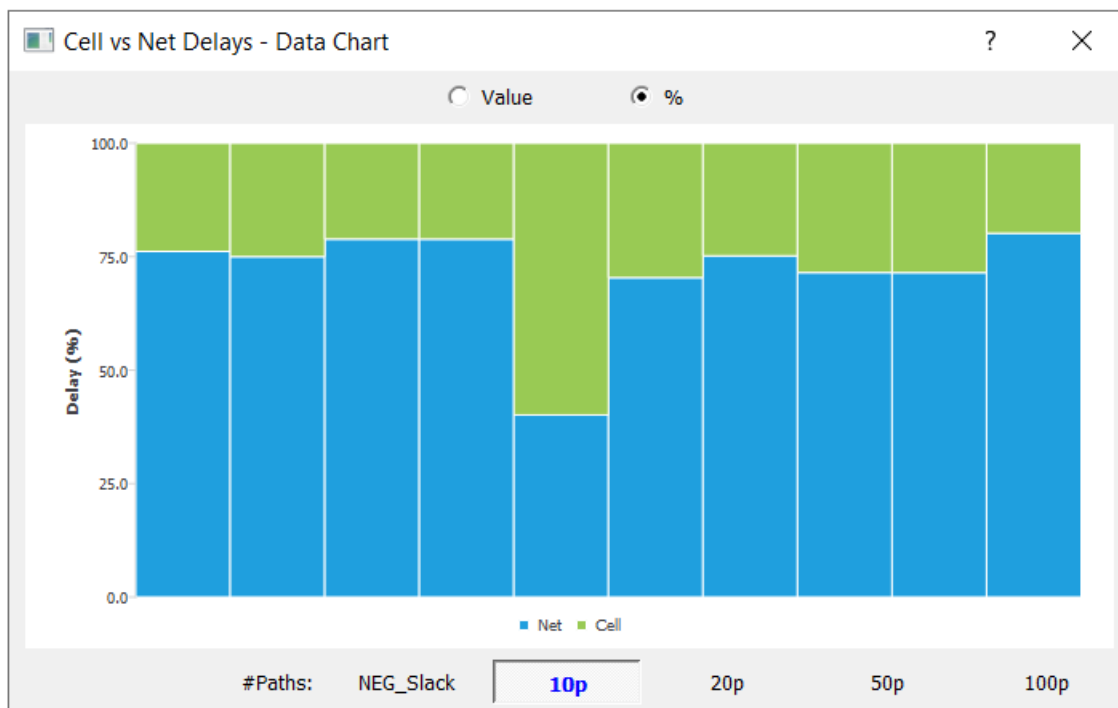
Clicking a bar in the chart displays a pop-up chart with expanded path information, similar to the following.

Figure 6-30. Example of Expanded Path Popup

Percentage (%) View

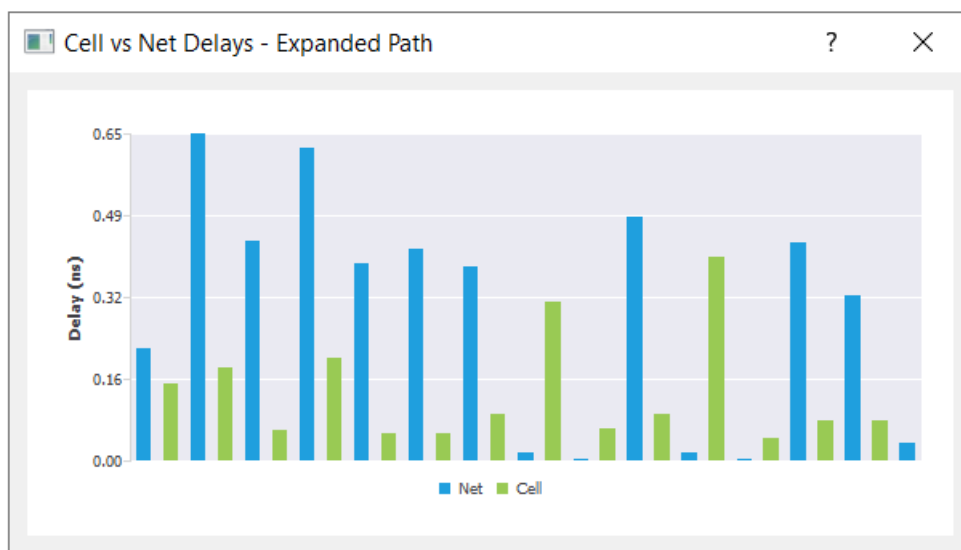
Every stack bar in the percentage chart represents net and cell delay for a path in terms of percentage. Buttons at the bottom of the chart allow you to select the number of paths from the path range: **NEG_Slack**, **10p**, **20p**, **50p**, and **100p** (default is **10p**).

Figure 6-31. Example of Stacked Bar Chart (Percentage View)



Clicking a bar in the chart displays a pop-up chart with expanded path information, similar to the following.

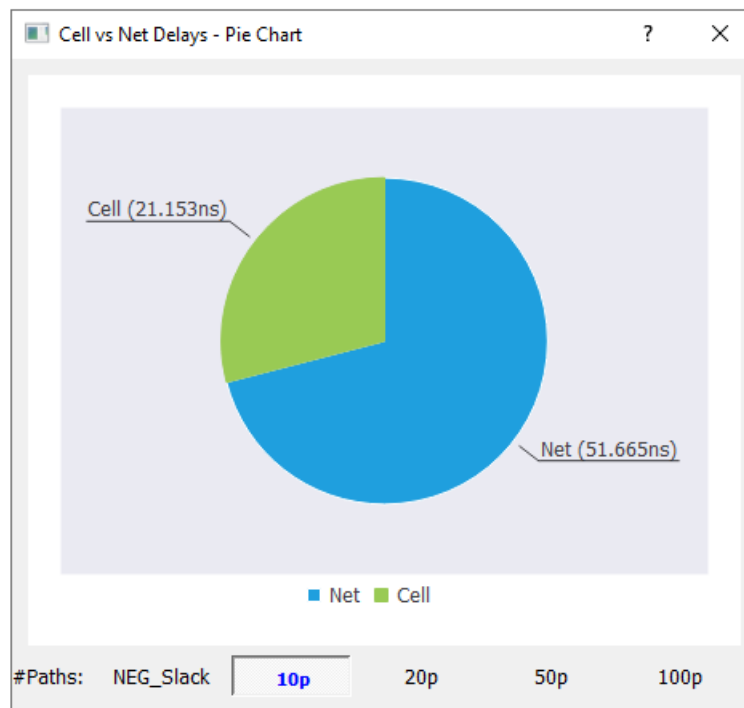
Figure 6-32. Example of Expanded Path Popup



Pie Chart

The pie chart shows the sum of total net delay and cell delay for a selected path range. Buttons at the bottom of the chart allow you to select the number of paths from the path range: **NEG_Slack**, **10p**, **20p**, **50p**, and **100p** (default is **10p**).

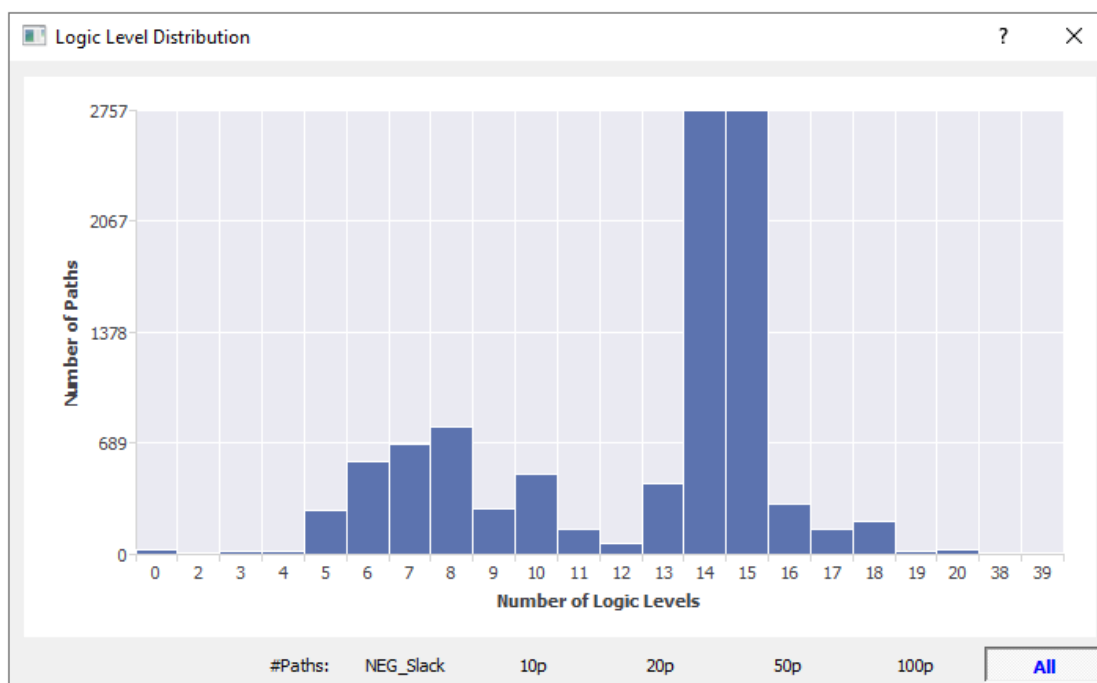
Figure 6-33. Example of Pie Chart



Logic Level Distribution

Click **Insights > Logic Level Distribution** to view a distribution chart that represents the number of paths having a logic count equal to the bar number. Hovering over a bar in the distribution chart displays a tooltip that shows the total number of path information. The following figure shows an example of logic level distribution.

Figure 6-34. Logic Level Distribution



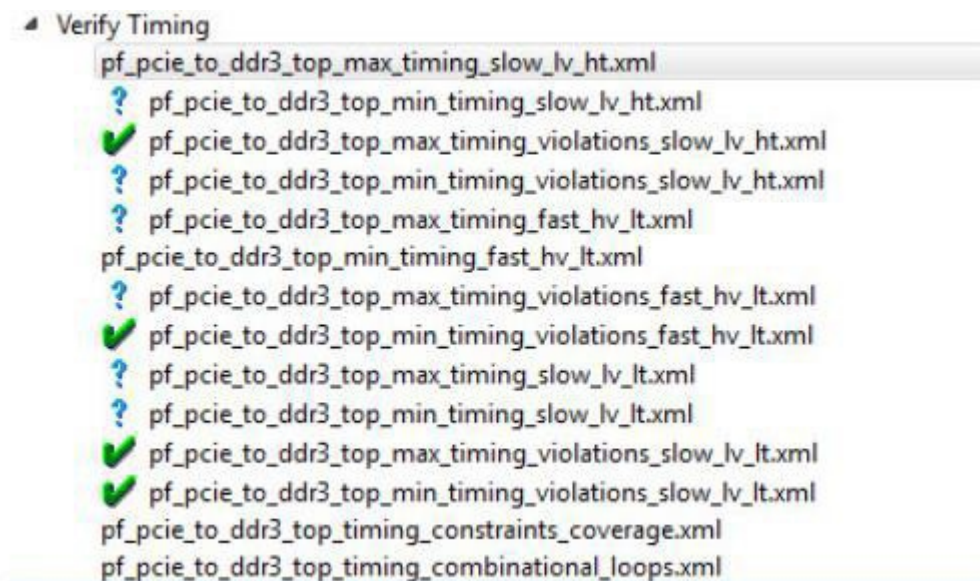
6.12.4. Types of Timing Reports [\(Ask a Question\)](#)

From the **Design Flow > 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 6-22. Types of Timing Reports

Report	Description
Timing reports	The following reports can be generated: • Max Delay Static Timing Analysis report based on Slow process, Low Voltage, and High Temperature operating conditions. • Min Delay Static Timing Analysis report based on Fast process, High Voltage, and Low Temperature operating conditions. • Max Delay Static Timing Analysis report based on Fast process, High Voltage, and Low Temperature operating conditions. • Min Delay Static Timing Analysis report based on Slow process, Low Voltage, and High Temperature operating conditions. • Max Delay Static Timing Analysis report based on Slow process, Low Voltage, and Low Temperature operating conditions. • Min Delay Static Timing Analysis report based on Slow process, Low Voltage, and Low Temperature operating conditions.
Timing violations reports	Organizes timing information by clock domain. You can configure which reports to generate using the Verify Timing Configuration dialog box. The following reports can be generated: • Max Delay Analysis Timing Violation report based on Slow process, Low Voltage, and High Temperature operating conditions. • Min Delay Analysis Timing Violation report based on Fast process, High Voltage, and Low Temperature operating conditions. • Max Delay Analysis Timing Violation report based on Fast process, High Voltage, and Low Temperature operating conditions. • Min Delay Analysis Timing Violation report based on Slow process, Low Voltage, and High Temperature operating conditions. • Max Delay Analysis Timing Violation report based on Slow process, Low Voltage, and Low Temperature operating conditions. • Min Delay Analysis Timing Violation report based on Slow process, Low Voltage, and Low Temperature operating conditions.
Constraints coverage report	Displays the overall coverage of the timing constraints set on the current design. <root>_timing_constraints_coverage.xml (generated by default)

Figure 6-35. Reports Example



The following table describes the icons associated with reports.

Table 6-23. Report Listing Icon Legend

Icon	Definition
	Timing requirements met for this report.
	Timing requirements not met (violations) for this report.
	Timing report is available but not selected/configured for generation.

6.12.5. SmartTime [\(Ask a Question\)](#)

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](#).

6.12.5.1. Static Timing Analysis (STA) [\(Ask a Question\)](#)

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.

6.12.5.2. SmartTime and Place and Route [\(Ask a Question\)](#)

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.

6.12.5.3. SmartTime and Timing Reports [\(Ask a Question\)](#)

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

6.12.5.4. SmartTime and Cross-Probing into Chip Planner [\(Ask a Question\)](#)

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**).

6.12.5.5. SmartTime and Cross-Probing into Constraint Editor [\(Ask a Question\)](#)

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

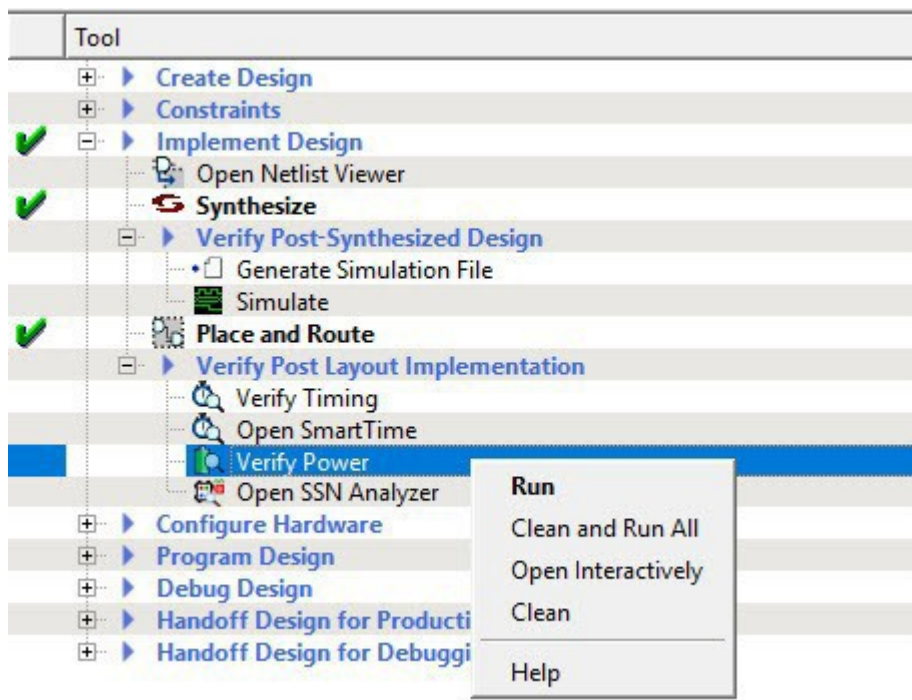
The Constraint Editor reflects the newly added timing exception constraint.

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

6.12.6. Verify Power [\(Ask a Question\)](#)

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

Figure 6-36. Verify Power Right-Click Menu



The following table describes the options.

Table 6-24. Verify Power Options

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.
Open Interactively	Displays the SmartPower for Libero SoC tool.
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 Report tab to the Libero SoC GUI window and Power Report will be selected.

6.12.7. Verify Power Sub-Commands [\(Ask a Question\)](#)

The following table describes the Verify Power sub-commands.

Table 6-25. Verify Power Sub-Commands

Sub-command	Description
Run	Runs the default power analysis and produces a power report. This sub-command is functionally equivalent to double-clicking Verify Power.
Clean and Run All	Functionally equivalent to clicking Clean and Run.
Open Interactively	Starts the SmartPower for Libero SoC tool.

Table 6-25. Verify Power Sub-Commands (continued)

Sub-command	Description
Clean	Clears the history of any previous default power analysis, including deletion of any reports, and clears the flow task completion icon  .
Configure Options	If there are options to configure, a dialog box appears with technology-specific choices.
View Report	Available if a report is available. Selecting this sub-command adds the Report tab to the Libero SoC GUI and selects and displays the Power Report.

6.12.8. SmartPower ([Ask a Question](#))

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 adjust to reduce power consumption.

SmartPower provides a detailed and accurate way to analyze designs for Microchip SoC FPGAs. From a top-level summary, you can analyze 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 QuestaSim ME Pro 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](#).

6.12.9. Simultaneous Switching Noise ([Ask a Question](#))

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**.

6.12.9.1. Supported Dies and Packages [\(Ask a Question\)](#)

The following table lists 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 6-26. Supported Dies and Packages

Family	Die	Package
PolarFire®	RTPF500ZT	CG1509
	MPF050T	FCSG325/FCVG484
	MPF300XT	FCG1152
	MPF100T	FCG484
	MPF200T	FCG484/FCG784
	MPF300T	FCG484/FCG784 /FCG1152
	MPF500T	FCG784/FCG1152
SmartFusion®2 / IGLOO®2	M2S/M2GL150	FC1152
	M2S/M2GL090	FC676
	M2S/M2GL060	FG896
	M2S/M2GL050	FG896
	M2S/M2GL025	FG484/VF400
	M2S/M2GL010	FG484
RTG4™ FPGA	RTG150	CG1657

6.12.9.2. Supported I/O Standard [\(Ask a Question\)](#)

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

6.12.9.3. Supported I/O Types [\(Ask a Question\)](#)

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

6.12.9.4. SSN Analyzer Tabs [\(Ask a Question\)](#)

The SSN Analyzer has three tabs:

- Noise Report
- Excluded IOs
- Summary

6.12.9.4.1. Noise Report Tab [\(Ask a Question\)](#)

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

Figure 6-37. SSN Analyzer – Noise Report Tab

Bank Name/ Pin Number	Port Name	Instance Name	I/O Standard	Drive Strength (mA)	Static	Don't Care	Noise Margin (%)	Within Guideline
Bank0 (1.5v)								
A16	wr	wr_obuf/U0/U_IOPAD	LVCN0515	4	☒	☐	99.01	Yes
A15	datao[28]	datao_obuf[28]/U0/U_IOPAD	LVCN0515	4	☐	☐	99.01	Yes
Bank1 (3.3v)								
J28	datao[29]	datao_obuf[29]/U0/U_IOPAD	LVTTL	4	☐	☐	98.38	Yes
J29	datao[25]	datao_obuf[25]/U0/U_IOPAD	LVCN0533	4	☐	☐	98.38	Yes
Bank2 (2.5v)								
M30	datao[4]	datao_obuf[4]/U0/U_IOPAD	LVCN0525	12	☐	☐	-11.66	No
N26	address[7]	address_1_obuf[7]/U0/U_IOP	LVCN0525	16	☐	☐	-69.11	No
M28	address[4]	address_1_obuf[4]/U0/U_IOP	LVCN0525	16	☐	☐	-50.88	No
K30	address[3]	address_1_obuf[3]/U0/U_IOP	LVCN0525	16	☐	☐	-11.66	No
N27	address[2]	address_1_obuf[2]/U0/U_IOP	LVCN0525	16	☐	☐	-25.04	No
M27	address[25]	address_1_obuf[25]/U0/U_IOP	LVCN0525	16	☐	☐	-74.95	No
L28	address[24]	address_1_obuf[24]/U0/U_IOP	LVCN0525	16	☐	☐	-74.21	No
P23	address[23]	address_1_obuf[23]/U0/U_IOP	LVCN0525	16	☐	☐	-72.10	No
M26	address[22]	address_1_obuf[22]/U0/U_IOP	LVCN0525	16	☐	☐	-32.03	No
N25	address[21]	address_1_obuf[21]/U0/U_IOP	LVCN0525	16	☐	☐	-68.80	No
K28	address[20]	address_1_obuf[20]/U0/U_IOP	LVCN0525	16	☐	☐	-73.62	No
N24	address[1]	address_1_obuf[1]/U0/U_IOP	LVCN0525	16	☐	☐	-49.89	No
P24	address[19]	address_1_obuf[19]/U0/U_IOP	LVCN0525	16	☐	☐	-59.92	No
L29	address[15]	address_1_obuf[15]/U0/U_IOP	LVCN0525	16	☐	☐	-76.35	No
J27	address[14]	address_1_obuf[14]/U0/U_IOP	LVCN0525	16	☐	☐	-62.78	No
L30	address[13]	address_1_obuf[13]/U0/U_IOP	LVCN0525	16	☐	☐	-75.03	No
P30	address[12]	address_1_obuf[12]/U0/U_IOP	LVCN0525	16	☐	☐	100.00	Yes
Bank3 (2.5v)								
A29	mio	mio_obuf/U0/U_IOPAD	LVCN0525	6	☐	☐	70.52	Yes
W29	datao[20]	datao_obuf[20]/U0/U_IOPAD	LVCN0525	12	☐	☐	74.07	Yes
U25	datao[15]	datao_obuf[15]/U0/U_IOPAD	LVCN0525	12	☐	☐	68.75	Yes
V23	datao[14]	datao_obuf[14]/U0/U_IOPAD	LVCN0525	12	☐	☐	74.07	Yes

The following table describes the columns in the tab.

Table 6-27. Columns in the Noise Report Tab

Column	Description
Bank Name/Pin Number	The bank number and package pin number of the port.
Port Name	Name of the port.
Instance name	Instance name of the port.
I/O Standard	I/O standards supported by SSN Analyzer. Supported standards are: - LVCN053.3V - LVCN052.5V - LVCN051.8V - LVCN051.5V - LVCN051.2V - LVTTL
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. Note: 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.

Table 6-27. Columns in the Noise Report Tab (continued)

Column	Description
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: • LVTTTL/LVCMOS (3.3 V): Yes (within guideline) is defined as follows: – A ground bounce voltage less than or equal to 1.25 V and a pulse width of less than or equal to 1 ns. – A V_{DD} dip voltage greater than or equal to V_{IHmin} and a pulse width of less than or equal to 1 ns. • All other LVCMOS standards (2.5 V, 1.8 V, 1.5 V, 1.2 V): A Yes (within guideline) is defined as follows: – A ground bounce voltage less than or equal to V_{ILmax} for ground bounce and a pulse width of less than or equal to 1 ns. – A V_{DD} dip voltage greater than or equal to V_{IHmin} and a pulse width of less than or equal to 1 ns. • Noise margin violating the criteria for Yes is considered to fall outside the specified guidelines, and is reported as a No.

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 6-28. 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. Note: This menu item is active when the I/O Editor 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 Search, 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 in SSN analysis.

Table 6-28. Noise Report Right-Click Menu (continued)

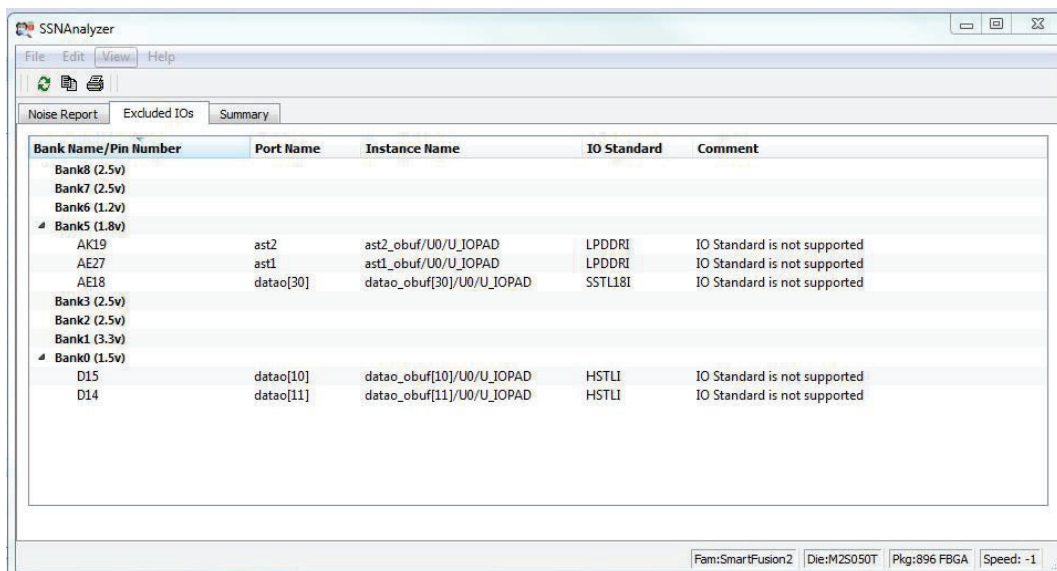
Menu	Description
Pulse Width	Settling time of the signal bounce. This is a threshold value that the signal bounce must exceed before the signal bounce is recognized for SSN calculation. Choices are: 0 ns: any signal bounce with a pulse width above 0 ns is recognized for SSN calculation. 1 ns: only signal bounces with a pulse width at or above 1 ns are recognized for SSN calculation. Changing this selection discards changes made for the current pulse width selection and triggers a re-analysis based on the new pulse width. Note: 1 ns pulse width is supported for the MPF300XT/FCG1152 die/package only.
Run Analysis	Not active when SSN first opens. It is activated only when you have made changes in the Noise Report. These changes might include one or more of the following: - Checking/unchecking the Don't Care check box for one or more I/Os. - Checking/unchecking the Static check box for one or more I/Os. When you make your changes, click Run Analysis to enable SSN recompute the Noise Margin number.
Save Report	Click to save the Noise Report in one of the following formats: - Text: Text file with *.txt file extension. - CSV: Spreadsheet file with *.csv file extension. - XML: XML file with *.xml file extension.

6.12.9.4.2. Excluded I/Os Tab [\(Ask a Question\)](#)

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 6-38. 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.

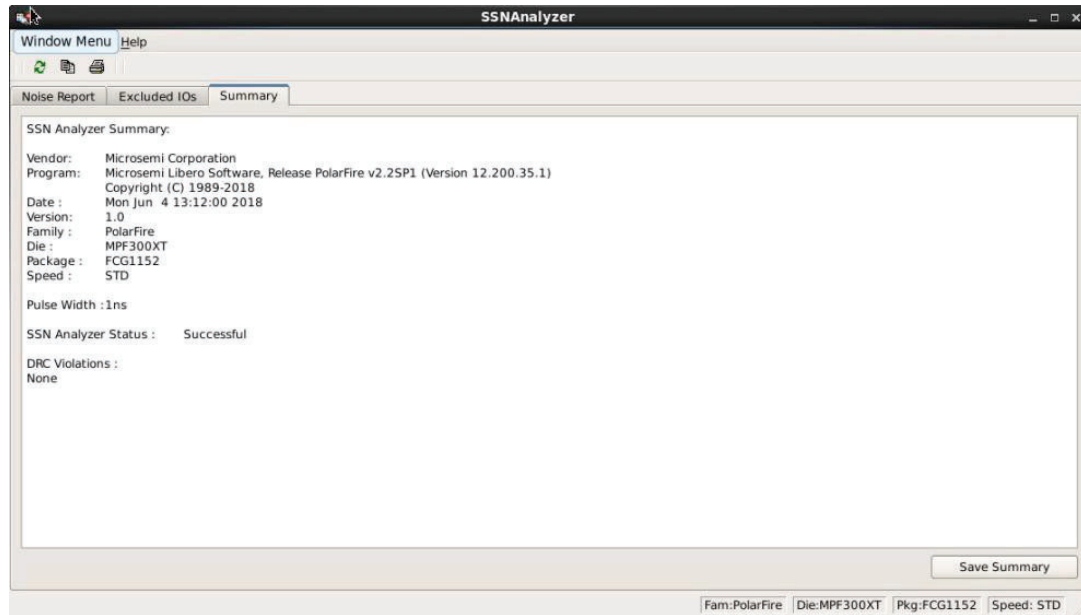
Table 6-29. Columns in the Excluded I/Os Tab

Options	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: • LVCMOS 3.3V • LVCMOS 2.5V • LVCMOS 1.8V • LVCMOS 1.5V • LVCMOS 1.2V • LVTTTL
Comment	Reason for exclusion (for example, unsupported I/O Standards or Marked as Static I/Os).

6.12.9.4.3. Summary Tab [\(Ask a Question\)](#)

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

Figure 6-39. SSN Analyzer - Summary Tab

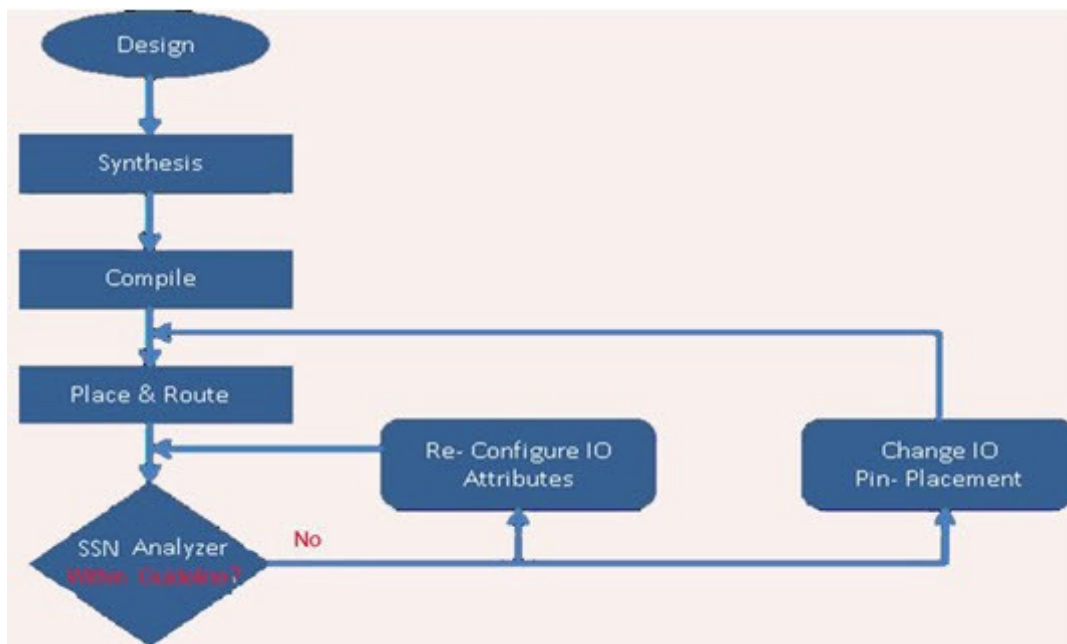


6.12.9.5. SSN Noise Analyzer Reports Failure Procedure [\(Ask a Question\)](#)

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 6-40. SSN Analyzer in the Design Flow



7. Configure Hardware [\(Ask a Question\)](#)

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

7.1. Programming Connectivity and Interface [\(Ask a Question\)](#)

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.

Note: If any FlashPro6 programmers are out of date, a message prompts you to update them. Do not disconnect your programmers during the update.

Figure 7-1. Programming Connectivity and Interface Window

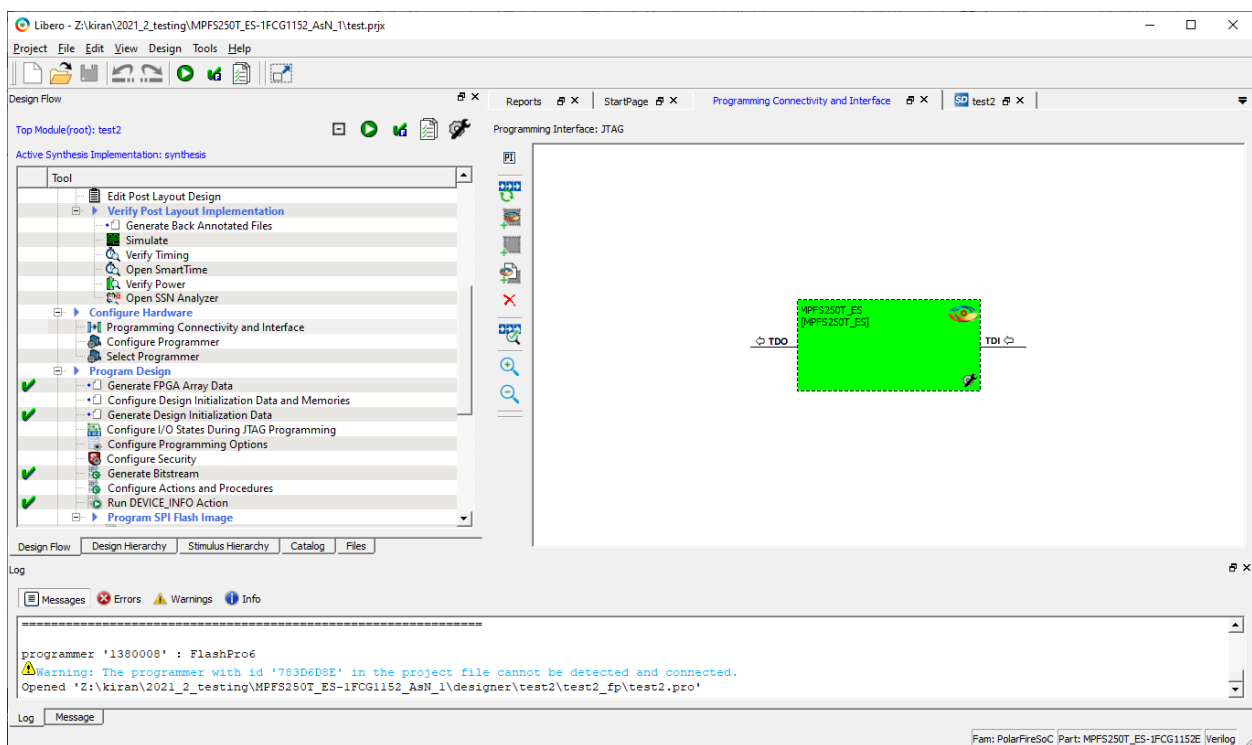


Table 7-1. Programming Connectivity and Interface Options and Icons

Option	Icon	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 Microchip device to the chain.
Add Non-Microsemi Device		Adds a non-Microchip device to the chain.

Table 7-1. Programming Connectivity and Interface Options and Icons (continued)

Option	Icon	Description
Add Microsemi Devices From Files		Adds a Microchip 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 [\(Ask a Question\)](#)

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

Table 7-2. Device Tooltips

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 that the programmer operates at a frequency lower than the slowest device in the chain.

7.1.2. Device Chain Details [\(Ask a Question\)](#)

The device within the chain contains the following details.

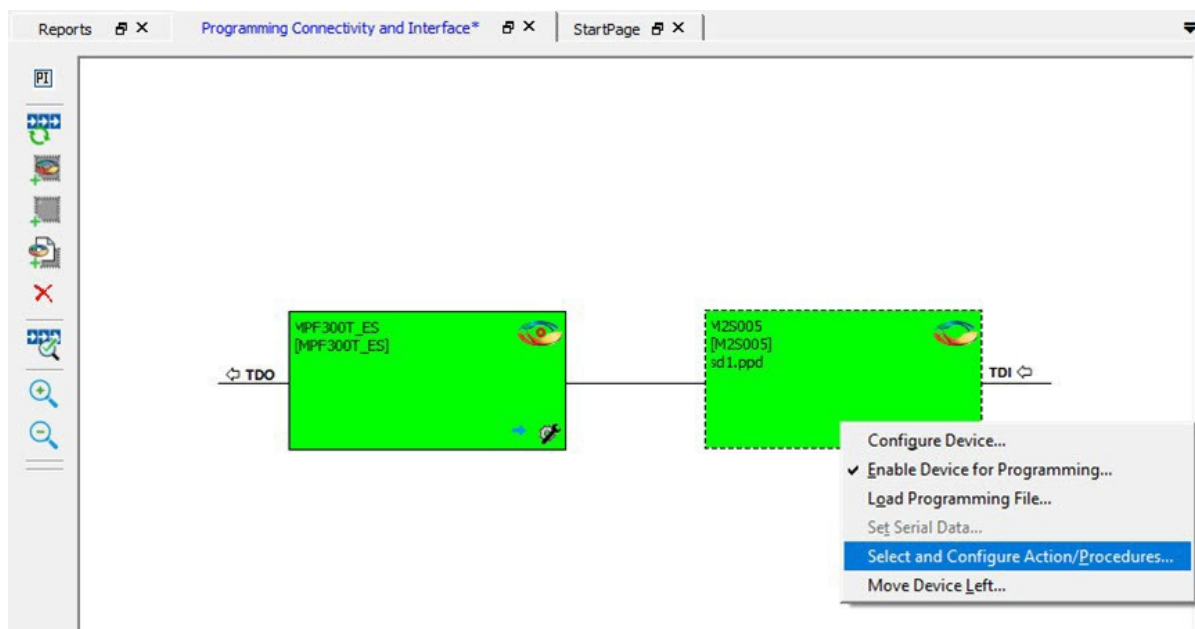
Table 7-3. Device Chain Details

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. - Green: device is enabled. - Gray: device is disabled.
Name	Name of your specified device.
File	Path to the programming file.
Set as Libero Design Device	Sets the Libero design device when there are multiple identical Libero design devices in the chain.

7.1.3. Right-Click Options [\(Ask a Question\)](#)

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

Figure 7-2. 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.
Enable Device for Programming	Enables the device for programming. - Green: enabled devices. - Gray: disabled devices.
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: - Select an action to program: Selected action is programmed in the Libero environment and saved to an exported FlashPro Express job. - Configure actions and procedures: - Actions: List of programming actions for your device. - Procedures: Advanced option that allows you to customize the list of recommended and optional procedures for an action.
Move Device Left/Right	Moves the device in the chain to left or right.

7.2. Programmer Settings [\(Ask a Question\)](#)

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 Programmer Settings dialog box.

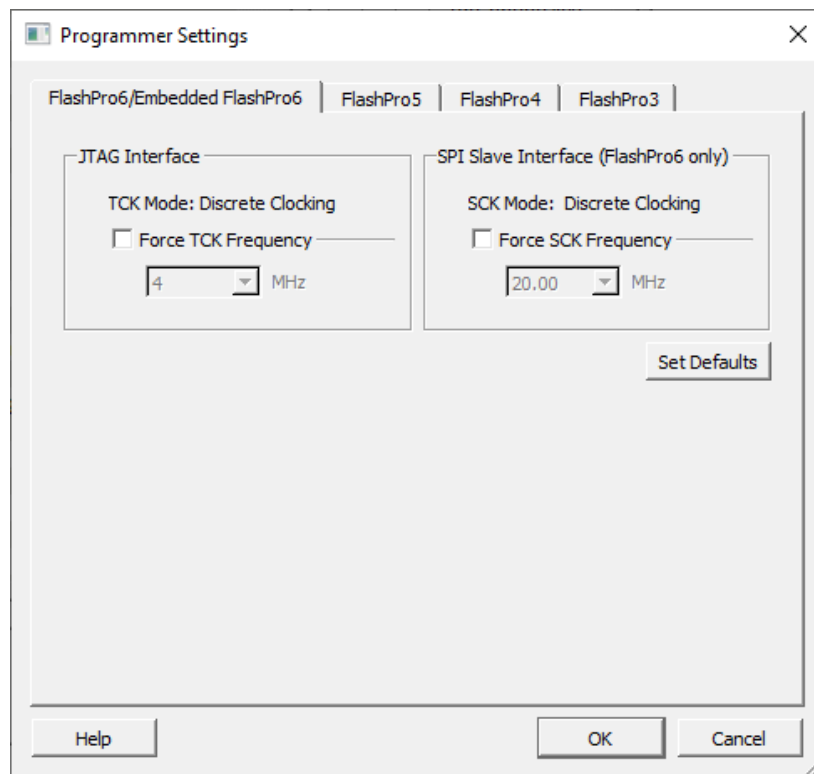
To display the Programmer Settings dialog box, in the Libero SoC Design Flow window, expand **Configure Hardware** and double-click **Configure Programmer**

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 7-5. 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
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 datasheet.

During execution, the frequency set by the FREQUENCY statement in the Program-Debug Data Base (PDB)/Standard Test and Programming Language (STAPL) file overrides the TCK frequency setting in the Programmer Settings dialog box. To prevent this override, check **Force TCK Frequency**

The following list provides 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

7.2.1. FlashPro5/4/3/3X Programmer Settings [\(Ask a Question\)](#)

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 is 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.

7.2.2. TCK Setting (Force TCK Frequency) [\(Ask a Question\)](#)

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.

7.2.3. Default TCK Frequency [\(Ask a Question\)](#)

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.

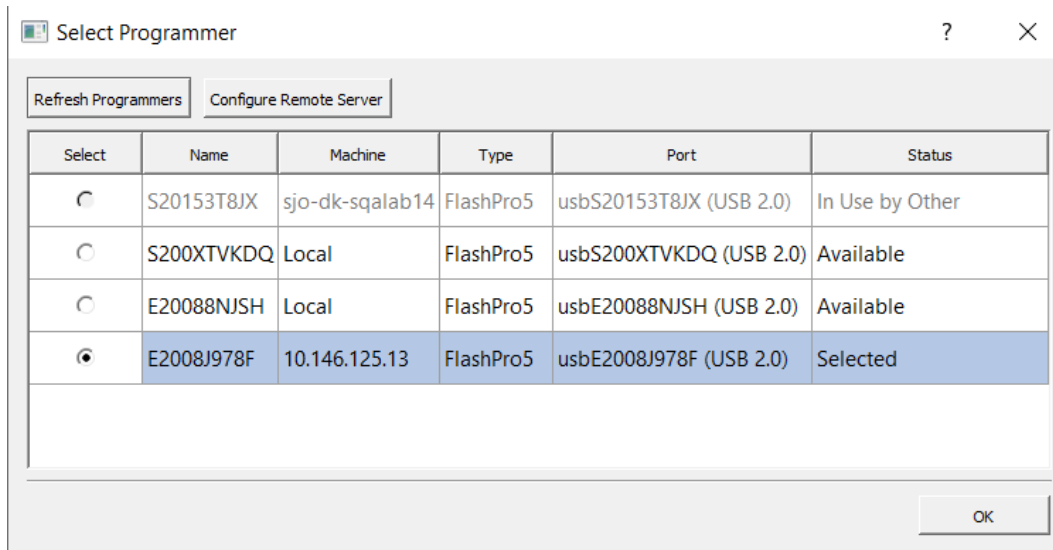
7.3. Working with Local and Remote Programmers [\(Ask a Question\)](#)

Libero® SoC supports programming Microchip® FPGAs by connecting multiple programmers to your computer. Additionally, you can program Microchip FPGAs over a network without requiring the programmer hardware to be physically connected to your local computer. When a programmer is connected to a remote machine, you can reserve and use it from your local machine as if it were attached locally.

In Libero SoC, new functionality is added in the **Design Flow > Configure Hardware > Select Programmer** dialog box.

The following figure illustrates the Select Programmer dialog box:

Figure 7-4. Select Programmer Dialog Box



The following buttons appear at the top of the dialog box:

- **Refresh Programmers:** Requests all available programmers from each connected host, including programmers connected directly to your workstation.
- **Configure Remote Server:** Opens the **Configure Remote Server** dialog box, which allows you to add and manage connections to remote programmer server hosts.
- **Select:** Selects the programmer to use.
- **Name:** Displays the programmer name.
- **Machine:** Displays the host name or IP address of the machine to which the programmer is connected. If the programmer is directly connected to your workstation, this field shows **Local**.
- **Type:** Specifies the programmer model.
- **Port:** Specifies the hardware port identifier (USB or COM) on the host machine to which the programmer is connected.
- **Status:** Indicates the current programmer availability: **Available**, **Selected** or **In Use by Other**.

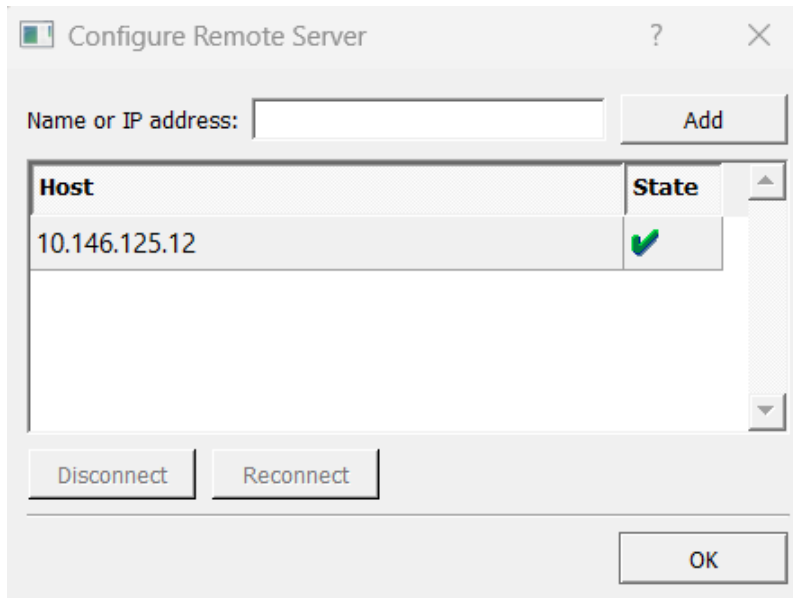
7.4. Configuring and Managing Remote Servers [\(Ask a Question\)](#)

The **Configure Remote Server** dialog box allows you to connect to remote programmer server hosts and select a programmer from any connected host.

Note: Remote access requires the host and client machines to be on the same network. The connection needs to be either directly in the network or within the VPN.

The following figure illustrates the Configure Remote Server dialog box:

Figure 7-5. Configure Remote Server



The dialog box contains the following fields and actions:

- Name or IP address: Specifies the name or IP address of the remote programmer server to connect to.
- Add: Connects to the server specified in the Name or IP address field and adds it to the Host list.
- Host: Displays the name or IP address of each remote server you have added.
- State: Indicates the connection status of the remote server. A green check mark confirms a successful connection. A red cross indicates that the connection has failed.
- Disconnect: Releases all reserved programmers and disconnects from the selected host.
- Reconnect: Reconnects to a host that has a failed connection.
- OK: Reserves the selected programmer, making it unavailable to other users, and releases all other programmers. Closes the **Configure Remote Server** dialog box.

Note: All tools started from Libero® SoC automatically use the same remote programmer.

7.5. Remote Programming Server (RPS) Installation Instructions [\(Ask a Question\)](#)

You can make your local machine act as a remote server. This allows other users within the same network to use FlashPro programmers physically connected to it.

The RPS is installed directly with the [Libero® SoC Media Installer](#). Complete the following instructions in the **Libero SoC Setup** wizard:

1. Follow the instructions in the wizard from the **Welcome to Installation Folder** windows.
2. In the **Select Installation Type** window, choose the **Custom** option.
3. In the **Select Components** window, with other desired components, select the RPS checkbox to include the functionality in the installer.
4. In the **RPS Server Configuration** window, specify the desired port numbers for programming requests.
 - The default ports to accept external connections are 57606 and 57605, which can be changed based on your requirements. These ports must be permitted through the firewall to enable remote access. The Media Installer completes this step for you during the installation.

7.6. Remote Programmer Troubleshooting [\(Ask a Question\)](#)

To recover from connection, detection, or reservation problems in Remote Programmer functionality, follow the instructions in this section to verify network connectivity, confirm the programming server is running on the remote machine, and resolve any remaining application issues.

Note: User-facing log files are not generated for the remote-side processes.



Attention: RPS is designed for trusted lab and corporate network environments. The client machine and remote machine must be on the same trusted network or connected through an approved VPN. The RPS service requires an open network port on the remote machine. Do not expose this port to the public internet. In the current release, RPS relies on the customer's network security controls, firewall rules, and VPN infrastructure to protect access to the remote machine.

7.6.1. Configuring Network Ports [\(Ask a Question\)](#)

If a connection fails to a remote machine from the **Configure Remote Hosts** dialog box even though the hostname or IP address is correct, you must verify that the required TCP ports are open on the remote machine.

Note: Ports 57605 and 57606 are the defaults assigned at installation. If the remote Host was installed with a different port configuration, check the actual ports configured on that machine and use them instead.

Perform the following steps on the remote machine (the machine where the programmer is physically connected).

- AlmaLinux OS/Red Hat® Linux®:
 - a. Open the terminal.
 - b. Run the following command:

```
sudo firewall-cmd --permanent --add-port 57605-57606/tcp
```
- Ubuntu:
 - a. Open the terminal.
 - b. Run the following command:

```
sudo ufw allow 57605:57606/tcp
```
- Windows® OS:
 - a. From the **Start Menu**, search for **Windows Defender Firewall with Advanced Security**, right-click, and select **Run as administrator**.
 - b. Click **Inbound Rules**, then click **New Rule**.
 - c. In the **New Inbound Rule Wizard**, you are prompted to select the type of rule you want to create. Choose **Port** to specify that this rule controls traffic based on network port numbers, and click **Next**.
 - d. Select **TCP** as the protocol, since the remote programmer requires TCP communication.
 - e. Select **Specific local ports** and enter 57605, 57606 to limit TCP communication access to these two ports. Click **Next**.
 - f. Select **Allow the connection** and click **Next**.
 - g. Select all three network profile options: **Domain**, **Private**, and **Public**. This ensures that the rule is active regardless of the type of network your computer is connected to. Click **Next**.
 - h. Enter a name for the rule and click **Finish**.

7.6.2. Configuring the RPS [\(Ask a Question\)](#)

When the remote server is reachable on the network but no programmers are detected, refer to the [Libero® SoC Media Installer](#) to verify the post-installation checks.

If the issue has not been resolved, uninstall the RPS from the Media Installer and install again.

7.6.3. Application Issues [\(Ask a Question\)](#)

In critical cases where normal behavior cannot be achieved, run the script in administrative mode from the bin folder to clean up any stuck processes and memory.

Note: The following solution needs to be considered cautiously and must be used as a last resort.

In case of a critical crash and abnormal behavior from the remote programmer functionality, run the bin/CleanUpRPS script in the RPS's bin folder or command shell.

For a manual equivalent, complete the following instructions to resolve the issue:

- Linux®: Enter the `rm /dev/shm/*` command.
- Windows® OS: Navigate to the `C:/ProgramData/boost_interprocess/` folder and remove all the content.
- Unified Solution: Depending on your operating system, complete the following steps:

1. Find the following processes:

- prgsrvmn
- prgservapp
- dbgserverapp

2. Kill all the processes.

Locally Connected Programmer Not Detected in Windows® OS

If everything appears to be working but your locally connected programmer is still not recognized, your antivirus or firewall may be blocking the programming software from operating correctly. To resolve this issue, follow these steps to add exceptions for the required applications:

Add firewall rules for the daemon and related processes by performing the following steps:

1. Open the command prompt as administrator.
2. Enter each of the following commands one at a time with the correct installation path. Press the Enter key after entering each command:

```
New-NetFirewallRule -DisplayName "Programming and Debug Server" -Direction Inbound
-Program "<installation path>\Libero_SoC_<version>\Libero_SoC\Designer\bin\prgsrvmn.exe"
-Action Allow
```

```
New-NetFirewallRule -DisplayName "Programming Server" -Direction Inbound -Program
"<installation path>\Libero_SoC_<version>\Libero_SoC\Designer\bin\prgservapp.exe" -Action
Allow
```

```
New-NetFirewallRule -DisplayName "Debug Server" -Direction Inbound -Program "<installation
path>\Libero_SoC_<version>\Libero_SoC\Designer\bin\dbgserverapp.exe" -Action Allow
```

8. Program Design [\(Ask a Question\)](#)

The following topics provide program design considerations.

8.1. Generating FPGA Array Data [\(Ask a Question\)](#)

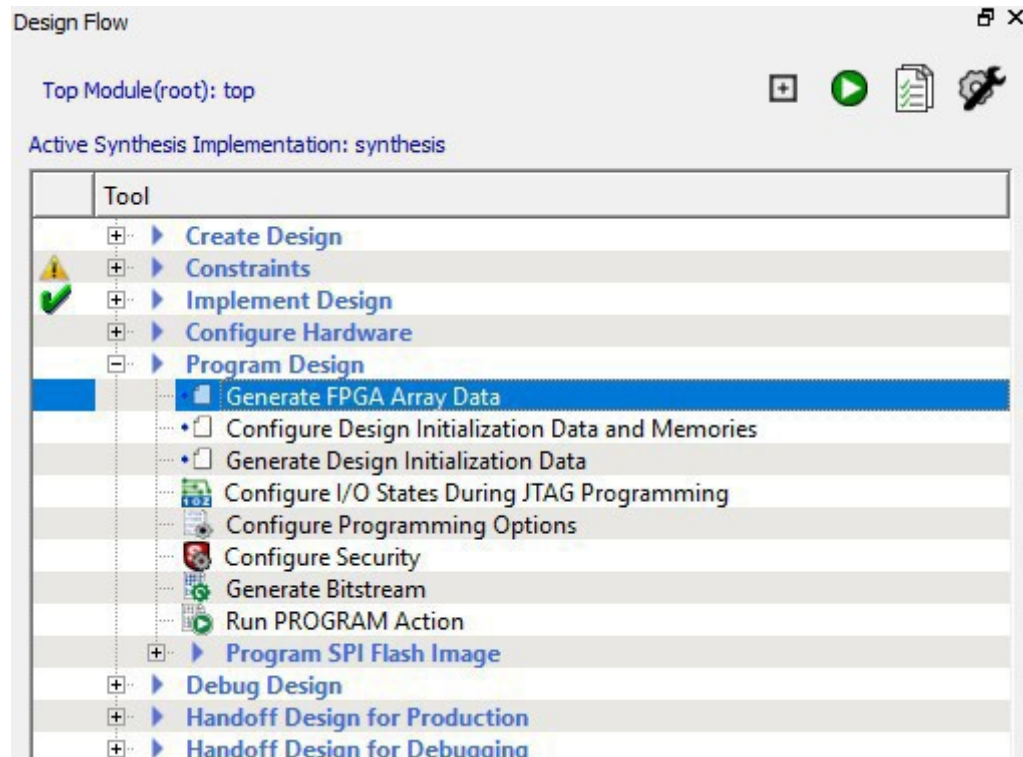
The Generate FPGA Array Data tool generates database files used in the following downstream tools:

- *.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 (PolarFire and PolarFire SoC) [\(Ask a Question\)](#)

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 nonvolatile 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 started only after completing the Generate FPGA Array Data step.

Use 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

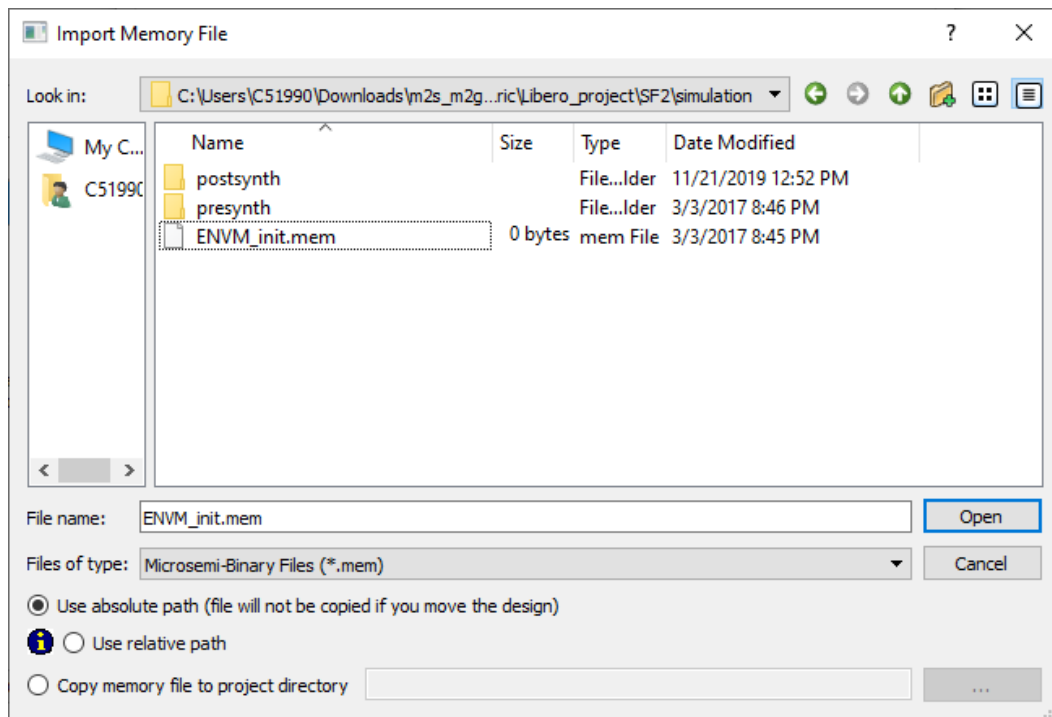
Button	Description
Apply	Click this button to save the changes made in a tab. The Apply button saves configuration changes only. For the initialization of the memory block to take effect, click Generate Initialization Clients on the Device Initialization tab.
Discard	Click this button to cancel any changes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

While importing memory files, note that the option **Use relative path** allows you to choose the path 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 [\(Ask a Question\)](#)

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, μ SRAM, 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 μ PROM, 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) [\(Ask a Question\)](#)

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, μ SRAM, 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 μ PROM, 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 Content filled with 0s 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 PolarFire Family Power-Up and Resets User Guide , 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) [\(Ask a Question\)](#)

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_SNVM_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 plaintext 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_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:

8.2.1.3. Third Stage (uPROM/sNVM/SPI Flash) [\(Ask a Question\)](#)

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 your 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, μ SRAM, 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 μ PROM, 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_I**.

☒ 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: ...

The following table describes the types of memory.

Table 8-3. Memory Types

Memory Type	Description
μ PROM	Name of the initialization client is INIT_STAGE_3_UPROM_CLIENT. Its start address is your choice, subject to the limitation that the start address can only be at the start of a uPROM block. Each μ PROM block is 256 words, so the allowed hexadecimal start addresses are 0x0, 0x100, 0x200, and so on.
sNVM	If there are no PCIe or XCVR blocks used in the design, the name of the sNVM initialization client for this stage is INIT_STAGE_3_SNVM_CLIENT. If there are PCIe or XCVR blocks used in the design along with Fabric RAMs, the name of the sNVM initialization client for this stage is INIT_STAGE_2_3_SNVM_CLIENT, which has the initialization sequence/instructions for the PCIe or XCVR blocks followed by Fabric RAMs. Its start address is your 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.

Table 8-3. Memory Types (continued)

Memory Type	Description
SPI-Flash Binding	Four binding options are available: - No Binding Plaintext: <root>_uic.bin file is a script file that can be opened to see readable text. - Binding Encrypted with Default Key: <root>_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. - Binding Encrypted with User Encryption Key 1 (UEK1): <root>_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. - Binding Encrypted with User Encryption Key 2 (UEK2): <root>_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. See the following table for various combinations of bindings that are valid or invalid in relation to security options.
SPI Clock divider value	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. Range: 1, 2, 4, 6 Default: 2

The following table lists combinations of bitstream types and stage 3 client binding.

Table 8-4. Combinations of Bitstream Types and Stage 3 Client Binding

Exported SPI Bitstream	Stage 3 Client Binding	Security	Security		Security		Security	
		Default	UEK1	UEK2	UEK1	UEK2	UEK1	UEK2
		Set	Set	Unset	Set	Set	Unset	Set
Trusted Facility	No binding	Valid	—		—		—	
	Default Security Binding	Valid	—		—		—	
	UEK1 binding	Invalid	—		—		—	
	UEK2 binding	Invalid	—		—		—	
Update UEK1	No binding	—	Valid		Valid		—	
	Default Security Binding	—	Invalid		Invalid		—	
	UEK1 binding	—	Valid		Valid		—	
	UEK2 binding	—	Invalid		Valid		—	

Table 8-4. Combinations of Bitstream Types and Stage 3 Client Binding (continued)

Exported SPI Bitstream	Stage 3 Client Binding	Security	Security		Security		Security	
		Default	UEK1	UEK2	UEK1	UEK2	UEK1	UEK2
		Set	Set	Unset	Set	Set	Unset	Set
Update UEK2	No binding	—	—	—	Valid	Valid	Valid	Valid
	Default Security Binding	—	—	—	Invalid	Invalid	Invalid	Invalid
	UEK1 binding	—	—	—	Valid	Invalid	Invalid	Invalid
	UEK2 binding	—	—	—	Valid	Valid	Valid	Valid

8.2.1.4. Time Out(s) [\(Ask a Question\)](#)

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)

The screenshot shows the 'Design Initialization' tab with several sub-tabs: uPROM, sNVM, SPI Flash, Fabric RAMs, and eNVM. The 'sNVM' sub-tab is active. Below the sub-tabs are buttons for 'Apply', 'Discard', and 'Help'. A text block explains that design blocks like LSRAM, pSRAM, transceivers, and PCIe can be initialized using data from non-volatile storage memory. A list of steps follows: 1. Set up your fabric RAMs initialization data, 2. Define the storage location, 3. Generate the initialization clients, 4. Generate or export the bitstream, 5. Program the device. The 'Design initialization specification' section contains three stages: First stage (sNVM) with a checkbox for 'Broadcast instructions to initialize RAMs to zeros'; Second stage (sNVM) with a text field for 'Start address for second stage initialization client'; Third stage (sNVM/uPROM/SPI-Flash) with checkboxes for 'Start address for sNVM clients', 'Start address for uPROM clients', and 'Start address for SPI-Flash clients'. At the bottom, the 'Time Out (s)' field is set to 128, 'Auto Calibration Time Out (ms)' is 3000, and 'Custom configuration file' is empty.

8.2.1.5. Auto Calibration Time Out [\(Ask a Question\)](#)

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, μ SRAM, 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 μ PROM, 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_IL.

☒ 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 [\(Ask a Question\)](#)

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.

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, μ SRAM, 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 μ PROM, 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_IL.

☒ 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)

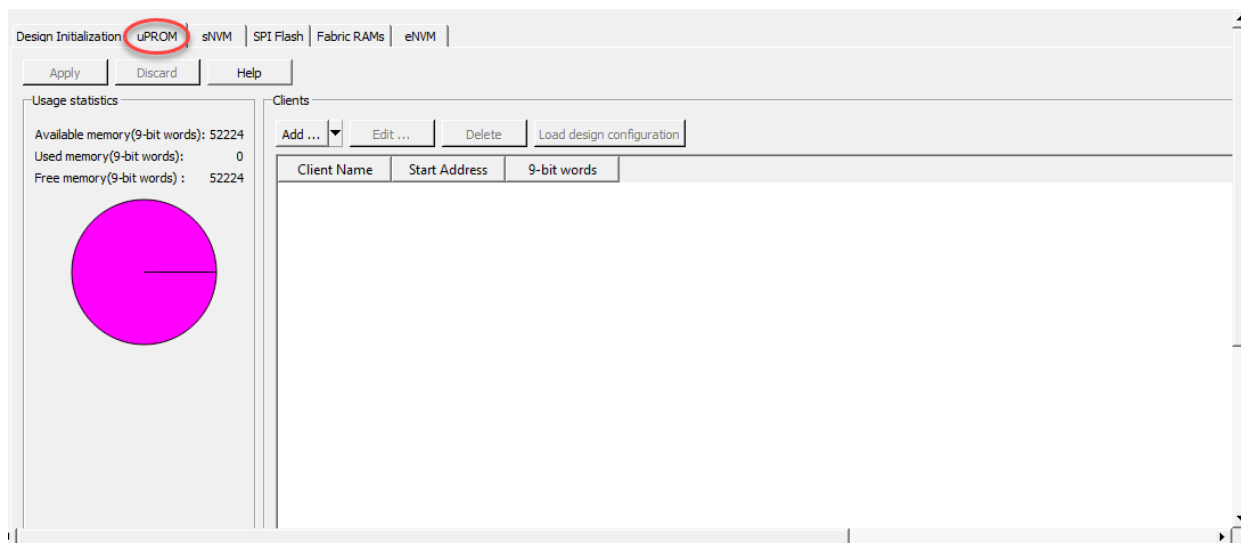
Time Out (s): 128

Auto Calibration Time Out (ms): 3000

Custom configuration file: ...

8.2.2. μ PROM Tab [\(Ask a Question\)](#)

μ PROM is the second tab in the Configure Design Initialization Data and Memories tool. Use this tab to manage data clients targeted for μ PROM memory.

Figure 8-10. uPROM Tab

The following table describes the elements in the **μPROM** tab.

Table 8-5. Elements in the uPROM Tab

Element	Description
Add	Adds uPROM clients.
Edit	Edits uPROM clients.
Delete	Deletes uPROM clients.
Load Design Configuration button	Loads the design's original μPROM configuration file into the <project>/component/work/UPROM.cfg file. This button is grayed out if the design does not have an original μPROM configuration file. 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 μPROM tab. The tooltip for both icons contains the time-stamp information of the design configuration file. The icons disappear the next time you click Apply.
Usage Statistics pie chart	Shows memory usage for the μPROM.

8.2.2.1. Adding μPROM Clients [\(Ask a Question\)](#)

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 8-6. 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: • Micro Binary 9-bit (<i>default</i>) • Micro Binary 32-bit • Intel-Hex • Motorola-S • Simple-Hex
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.

Table 8-6. Add Data Storage Client Dialog Box (continued)

Field	Description
Use content for simulation	Disabled and unavailable.

8.2.2.1.1. uPROM Clients Table [\(Ask a Question\)](#)

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

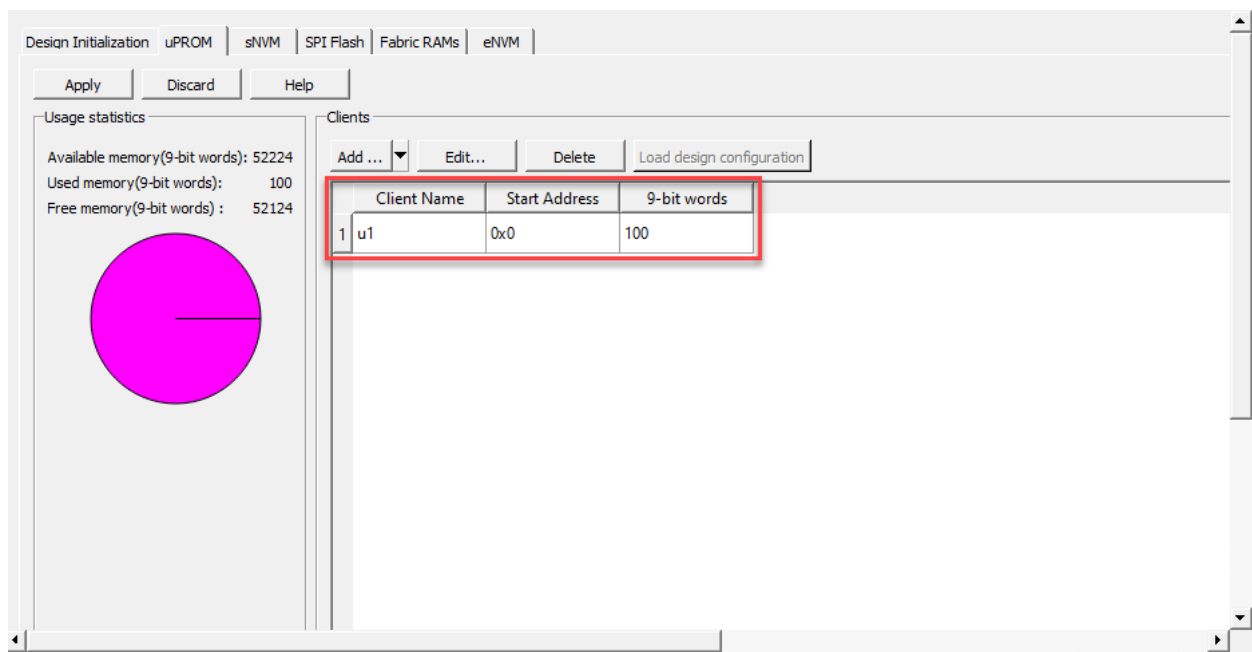


Table 8-7. Columns in the uPROM Clients Table

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 [\(Ask a Question\)](#)

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 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).
- Click **OK**.
- Click the **Apply** button.

Table 8-8. 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 [\(Ask a Question\)](#)

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, ensure 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 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 [\(Ask a Question\)](#)

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

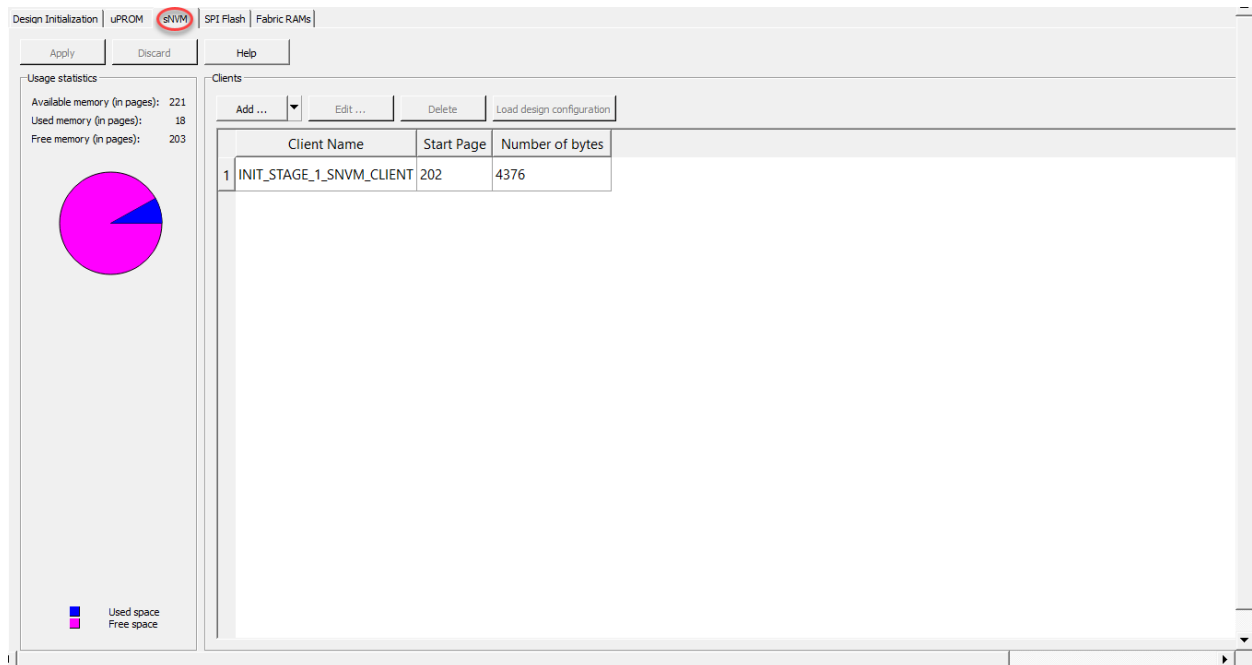
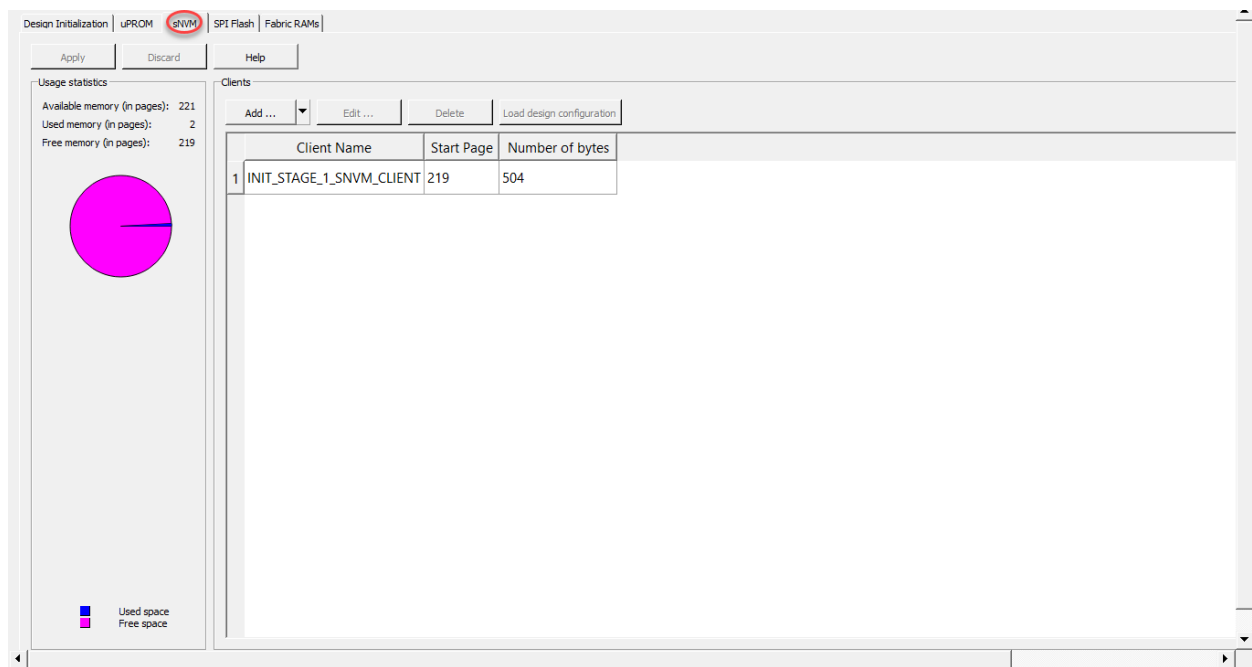


Figure 8-14. sNVM Tab without Broadcast Option Enabled



The following table describes the elements in the **sNVM** tab.

Table 8-9. Elements in the sNVM Tab

Element	Description
Add	Adds sNVM clients.
Edit	Edits sNVM clients.
Delete	Deletes sNVM clients.

Table 8-9. Elements in the sNVM Tab (continued)

Element	Description
Load Design Configuration	Loads the design's original sNVM configuration file into the <project>/component/work/sNVM.cfg file. This button is grayed out if the design does not have an original sNVM configuration file. 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. The tooltip for both icons contains the time-stamp information of the design configuration file. The icons disappear the next time you click Apply.
Usage Statistics pie chart	Shows available, used, and free memory, in pages, for all sNVM clients.

8.2.3.1. Adding sNVM Clients [\(Ask a Question\)](#)

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

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

Apply | Discard | Help

Usage statistics

Available memory (in pages): 221
Used memory (in pages): 20
Free memory (in pages): 201

Legend:
■ Used space
■ Free space

Clients

Add ... Edit... Delete Load design configuration

Add PlainText NonAuthenticated Client
Add PlainText Authenticated Client
Add CipherText Authenticated Client
Add Boot Mode 2 PlainText Authenticated Client
Add Boot Mode 2 CipherText Authenticated Client
Add USK Client

Start Page	Number of bytes
02	4384
0	224
3	12

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 [\(Ask a Question\)](#)

In the **sNVM** tab, from **Add** drop-down list, select **Add PlainText NonAuthenticated Client** to add the client to the [sNVM clients table](#).

Figure 8-16. Add PlainText NonAuthenticated Client Dialog Box

Table 8-10. 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. Note: 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.
Content filled with 0s	Populates the sNVM with zero.
No content	Client is a placeholder and will not be programmed.
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)

Table 8-10. Fields in the Add PlainText NonAuthenticated Client Dialog Box (continued)

Field	Description
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 must 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 (PolarFire SoC) [\(Ask a Question\)](#)

PolarFire SoC supports Boot Mode 2. 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

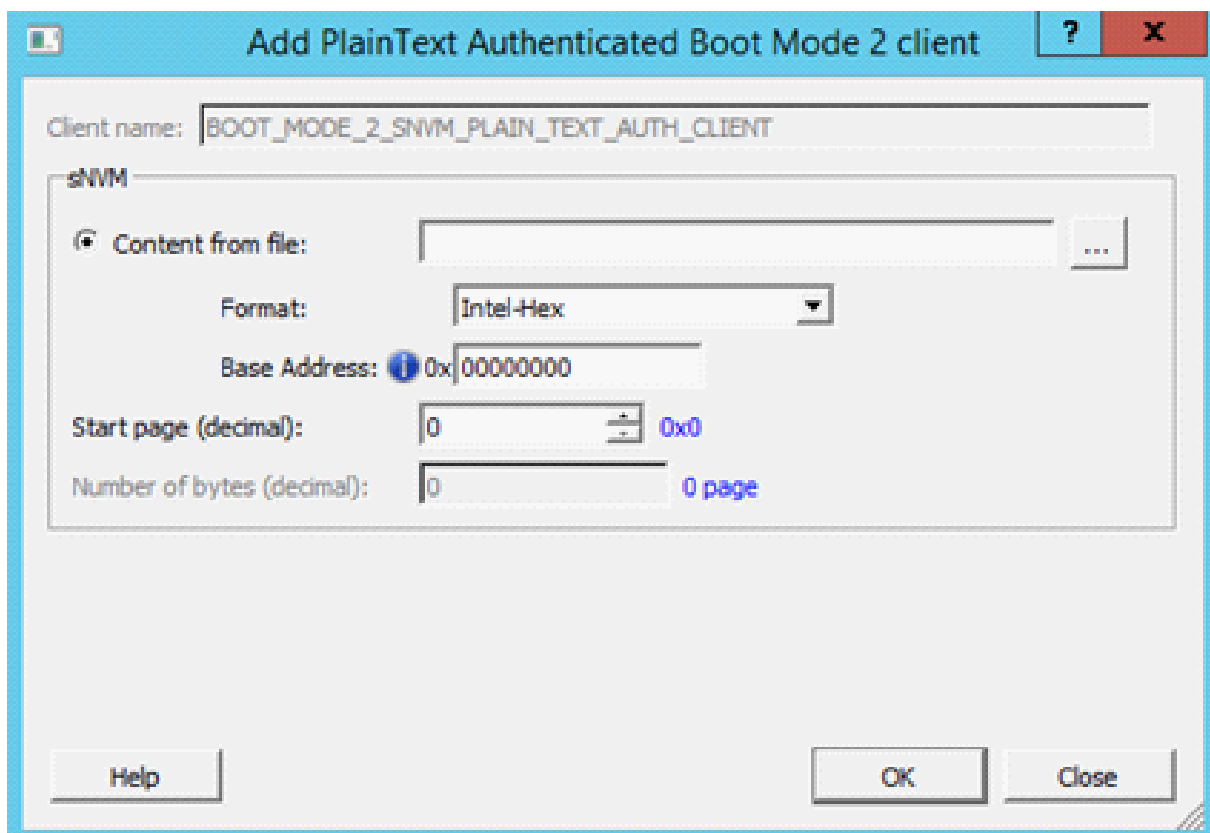


Figure 8-18. Add CipherText Authenticated Boot Mode 2 Client Dialog Box

Table 8-11. 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. Note: 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	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)
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 [\(Ask a Question\)](#)

Figure 8-19. Add USK Client Dialog Box

Table 8-12. 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 must be programmed.
Use content for simulation	Check if this client must be loaded for the simulation run.
Use as ROM	Check if this client must be used as ROM.
Fabric Access	Allows you to read from Fabric.
MSS Access	Allows you to read from MSS.

Note:

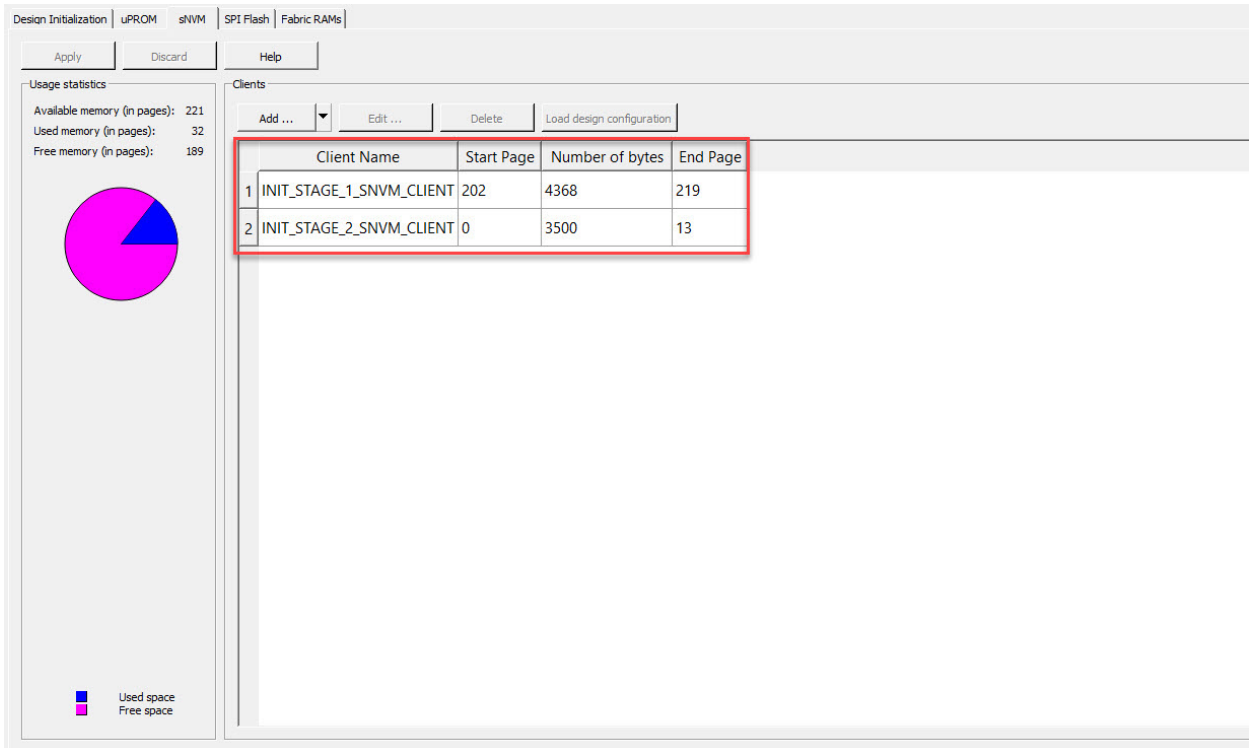
Atleast one of Fabric or MSS Access should be selected for Read, else a DRC error will be generated as "**Atleast Fabric or MSS should be selected for Read**".

8.2.3.1.4. sNVM Clients Table [\(Ask a Question\)](#)

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 client's table.

Table 8-13. 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 determined based on the start page you provided.

8.2.3.2. Editing sNVM Clients [\(Ask a Question\)](#)

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 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 [\(Ask a Question\)](#)

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, ensure 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 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 [\(Ask a Question\)](#)

SPI Flash is the fourth tab in the Configure Design Initialization Data and Memories tool. Use this tab to configure SPI Flash, select the memory size, and enable auto updating for parts of the SPI Flash configuration. 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 8-14. 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.

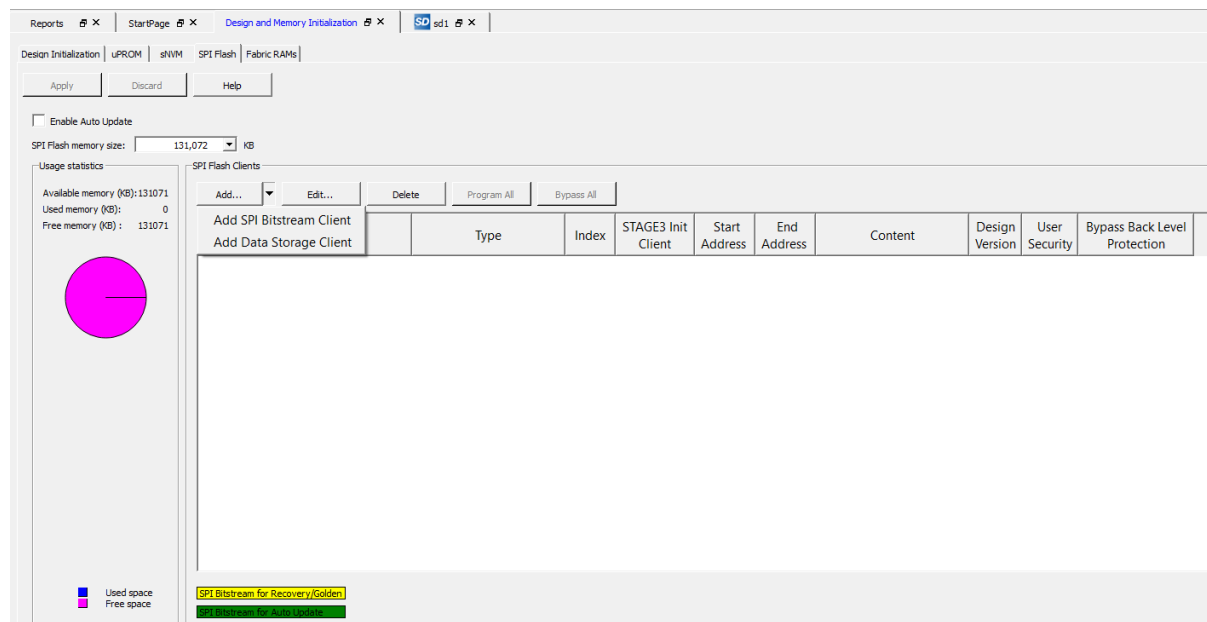
Table 8-14. Elements in the SPI Flash Tab (continued)

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	Enables auto update on the target device. The bitstream generated in Libero enables this feature. If you check this option, the SPI Bitstream for the Auto update client can be added. Auto update is set to index 1 automatically.
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 [\(Ask a Question\)](#)

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 (see the following sections).
3. Click **OK** to add the client to the [SPI Flash clients table](#).
4. Click the **Apply** button to save the configuration.

8.2.4.1.1. SPI Flash Clients Table [\(Ask a Question\)](#)

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.



Attention: If the SPI bitstream client is not present, you can create SPI data storage clients at any address, including those below 0x400. When the SPI bitstream client is present, however, the 0x000–0x3FF address range is reserved for the SPI directory and cannot be allocated to any other client. In this configuration, the usage statistics also account for the reserved SPI-Flash directory space.

Figure 8-24. SPI Flash Clients Table

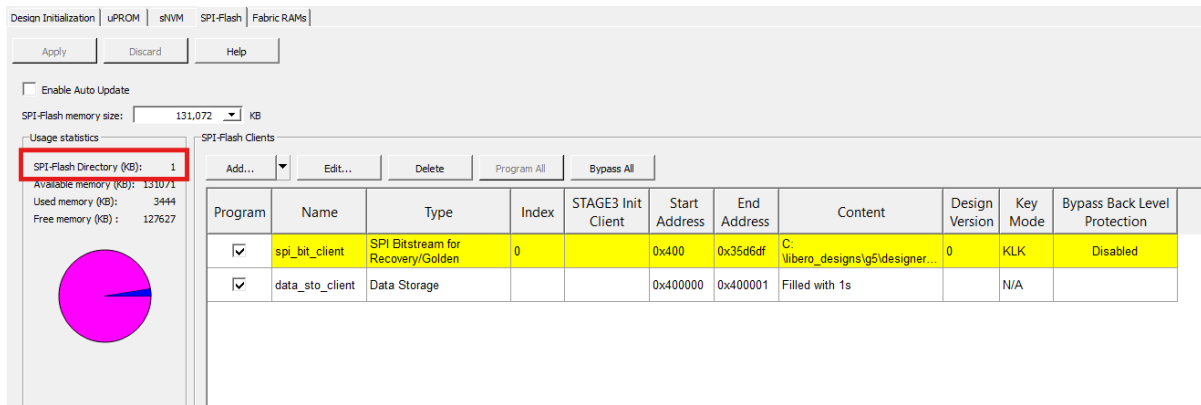


Table 8-15. Columns in the SPI Flash Clients Table

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: • SPI Bitstream for Recovery/Golden • SPI Bitstream for IAP • SPI Bitstream for Auto Update • Data Storage • Design Initialization
Index	• Index 0 is reserved for SPI Bitstream for Recovery/Golden. • Index 1 is reserved for auto update. The index for an IAP client can be in the range of 2 - 255.
STAGE3 INIT Client	If SPI Flash has a STAGE3 Initialization client, this columns shows the name of the client.
Start Address	Starting address, you gave to the client.
End Address	Ending address that Libero SoC determined based on the start address you provided.
Content	Choices are: • .spi file: SPI bitstream clients. • intel-hex (.hex or .ihx): Data storage clients. • Binary (*.bin): Data storage or design initialization client.
Design Version	Client design version.

Table 8-15. Columns in the SPI Flash Clients Table (continued)

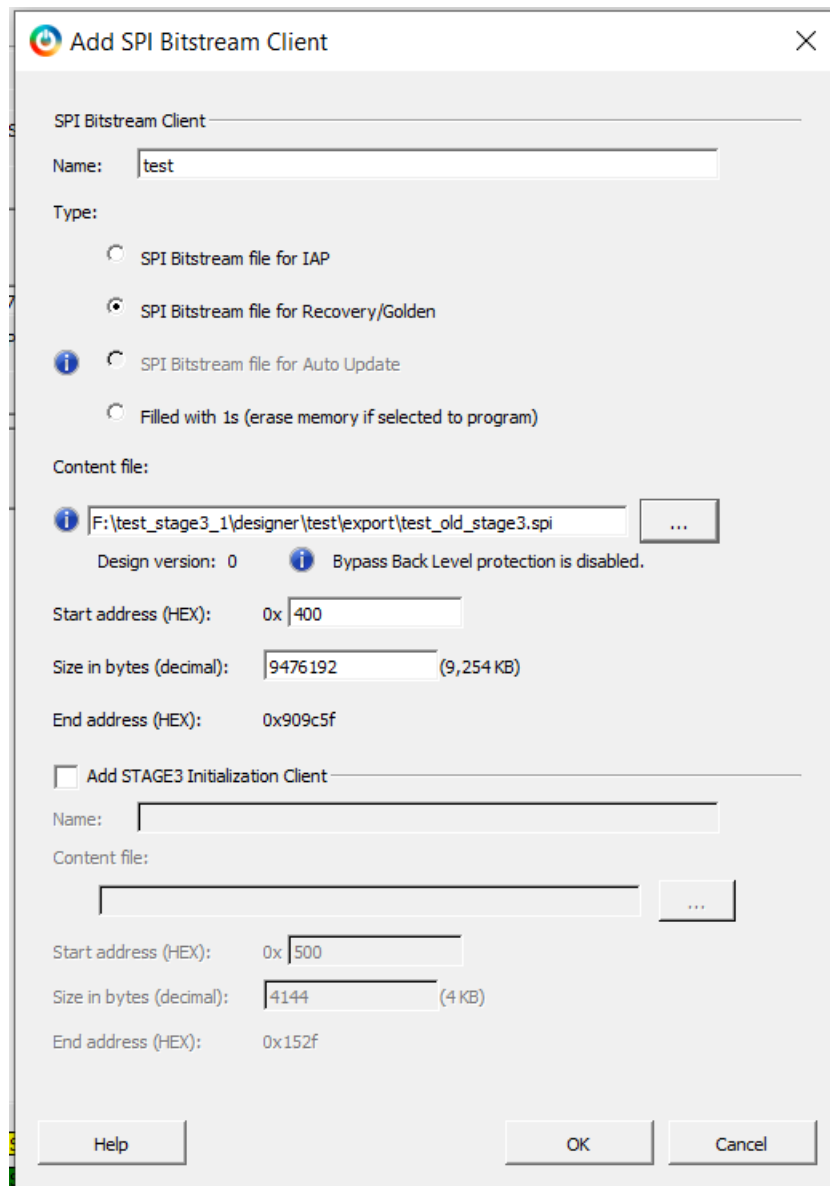
Column	Description
Key Mode	Key mode used to encrypt the SPI file (default is Key/UEK1/UEK2).
Bypass Back Level Protection for Recovery/Golden bitstream	This feature is enabled for only the SPI Bitstream clients for Recovery/Golden.

8.2.4.1.2. Settings for Add SPI Bitstream Client [\(Ask a Question\)](#)

Observe the following guidelines while using a SPI file generated with Libero 2022.2 or later version to create or edit a SPI bitstream client:

- If you use a SPI file with custom security to create or edit the client, an error message tells you that the SPI file cannot contain security.
- The SPI bitstream file used for SPI bitstream client must match the Libero target device. A SPI bitstream file from one family or device cannot be used to create or edit a client for another family or device. If you try to do so, an error message tells you that your current device does not match the SPI file.

Figure 8-25. Add SPI Bitstream Client Dialog Box Using Content File Generated with Libero Versions Earlier Than v2021.2



The dialog box is titled "Add SPI Bitstream Client" and contains the following fields and options:

- 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 (erase memory if selected to program)
 - Content file:
 -
 - Design version: 0 Bypass Back Level protection is disabled.
 - Start address (HEX): 0x
 - Size in bytes (decimal): (9,254 KB)
 - End address (HEX): 0x909c5f
- ☐ Add STAGE3 Initialization Client
 - Name:
 - Content file:
 - Start address (HEX): 0x
 - Size in bytes (decimal): (4 KB)
 - End address (HEX): 0x152f

Buttons:

Figure 8-26. Add SPI Bitstream Client Dialog Box Using Content File Generated with Libero v2021.2

Add SPI Bitstream Client

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 (erase memory if selected to program)

Content file:

...

Design version: ⓘ Bypass Back Level protection is disabled.

Start address (HEX):

Size in bytes (decimal): (9,254 KB)

End address (HEX):

☒ Add STAGE3 Initialization Client

Name:

Content file: ...

Start address (HEX):

Size in bytes (decimal): (4 KB)

End address (HEX):

Table 8-16. Fields in Add SPI Bitstream Client Dialog Box

Client	Guideline
Name	Name of the SPI bitstream client.

Table 8-16. Fields in Add SPI Bitstream Client Dialog Box (continued)

Client	Guideline
Type	Type of the SPI bitstream client. The following are the choices. - 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 - 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. - If Auto Update is enabled, then 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. - Do not use the master file for Recovery/Golden client with IAP. - If Back Level Protection is enabled in the Configure Security tool, then 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. - To allow for programming Recovery to pass, import a Bitstream that has been exported with the Bypass Back Level Protection option. Note: Bypass Back Level protection feature is supported only for SPI Bitstream clients for Recovery/Golden. - SPI Bitstream Client for Auto Update: In this tab, highlighted in green in the client table. 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. Note: The tool rejects a Bitstream file with Bypass Back Level Protection enabled for this type of client. - Filled with 1s: Populates the SPI bitstream client with one's.
Content file	Choice is spi file: SPI Bitstream clients.
Start address	Start address, in hexadecimal notation, of the SPI bitstream client.
Size in bytes	Size (in bytes) of the SPI bitstream client.
End address	Read-only field that shows the end address in hexadecimal notation.
Add STAGE3 Initialization Client	If the SPI file was generated with Libero® v2021.2, then this check box is checked or unchecked automatically, based on data from the SPI Bitstream content file. If a SPI file was generated with Libero versions earlier than v2021.2, you can add STAGE3 init data manually by checking this option.
Name	If Add STAGE3 Initialization Client is checked, enter a name for the STAGE3 init client.
Content file	If Add STAGE3 Initialization Client is checked, select the STAGE3 Initialization client content file. The file must be in .bin format.
Start address	If Add STAGE3 Initialization Client is checked, and the SPI file was generated with Libero v2021.2, the start address, in hexadecimal notation, of the STAGE3 Initialization client is populated automatically based on the data from the SPI file. Otherwise, you must enter the start address manually. Note: To prevent conflicts, the tool checks that no other SPI Flash clients have the same start address.
Size in bytes	Shown automatically after the content file is loaded. The size is based on the size of the content file. You can increase this value if desired.
End address	If Add STAGE3 Initialization Client is checked, then this read-only field shows the end address in hexadecimal notation of the STAGE3 Initialization client.

8.2.4.1.3. Settings for Add Data Storage Client [\(Ask a Question\)](#)

Figure 8-27. Add Data Storage Client Dialog Box

Add SPI Flash Data Storage Client

Name:

Content

☒ Memory file:

Format:

☐ Filled with 1s

Start address (HEX): 0x

Size in bytes (decimal):

End address (HEX):

Table 8-17. Fields in Add Data Storage Client Dialog Box

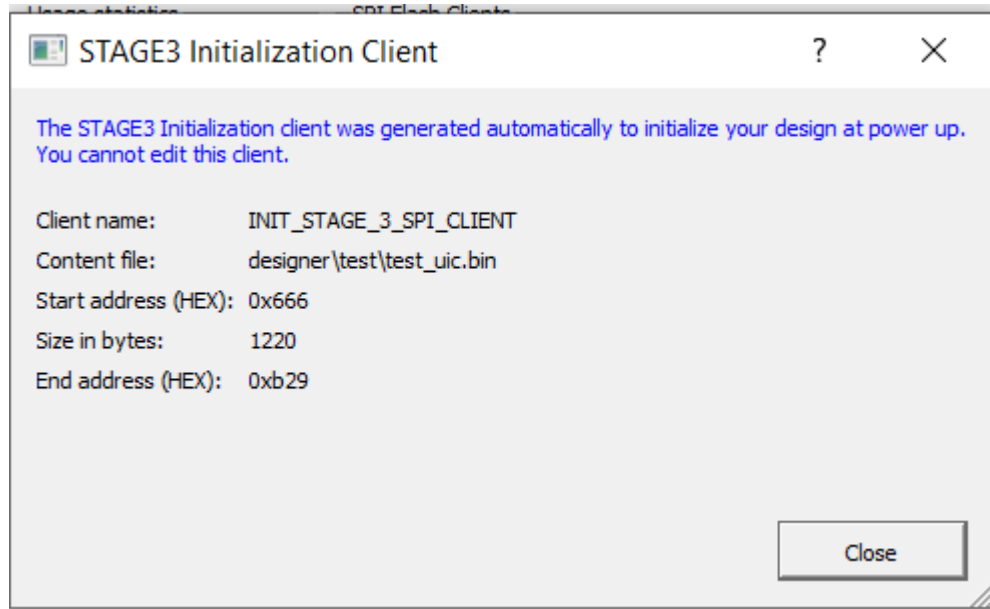
Client	Guideline
Name	Name of the SPI flash data storage client
Content	The following are the choices. - Intel: Hex Files (*.hex *.ihx). - Binary (*.bin). - Filled with 1s: Populates the data storage client with one's.
Start address	Start address, in hexadecimal notation, of the data storage client.
Size in bytes	Size of byte in decimal notation, of the data storage client.
End Address	Shows the end address of the client based on the value entered in the Size in Bytes field.

8.2.4.1.4. Automatically Generated Design Initialization Client [\(Ask a Question\)](#)

The Generate Design Initialization tool adds a Design Initialization client automatically to the **SPI Flash** tab. Double clicking this client displays the dialog box shown in the following figure.

For more information about the Initialization clients that can be generated, see [Generating Initialization Clients \(PolarFire\)](#).

Figure 8-28. Example of a STAGE3 Initialization Client Dialog Box



8.2.4.2. Editing SPI Flash Clients [\(Ask a Question\)](#)

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 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 [Settings for Add SPI Bitstream Client](#) or [Settings for Add Data Storage Client](#)).
3. Click **OK**.
4. Click the **Apply** button.

8.2.4.3. Deleting SPI Flash Clients [\(Ask a Question\)](#)

If you no longer need a SPI Flash client, you can delete the client.



CAUTION A warning message does not appear before you delete a client. Therefore, ensure you no longer need a client before you delete it.

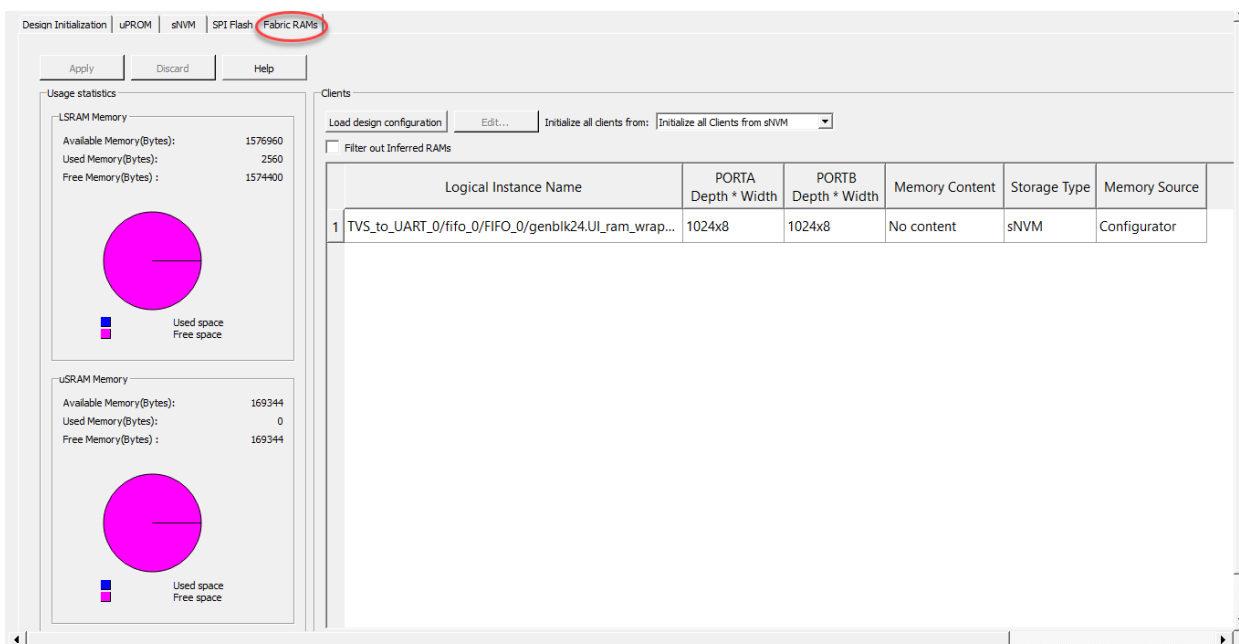
To delete a SPI Flash client:

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 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 [\(Ask a Question\)](#)

The **Fabric RAMs** tab allows you to select initialization options for Dual-Port SRAM, Two-Port SRAM, and μ SRAM memory blocks in your design.

Figure 8-29. Fabric RAMs Tab



The following table describes the elements in the **Flash RAMs** tab.

Table 8-18. Elements in the Flash RAMs Tab

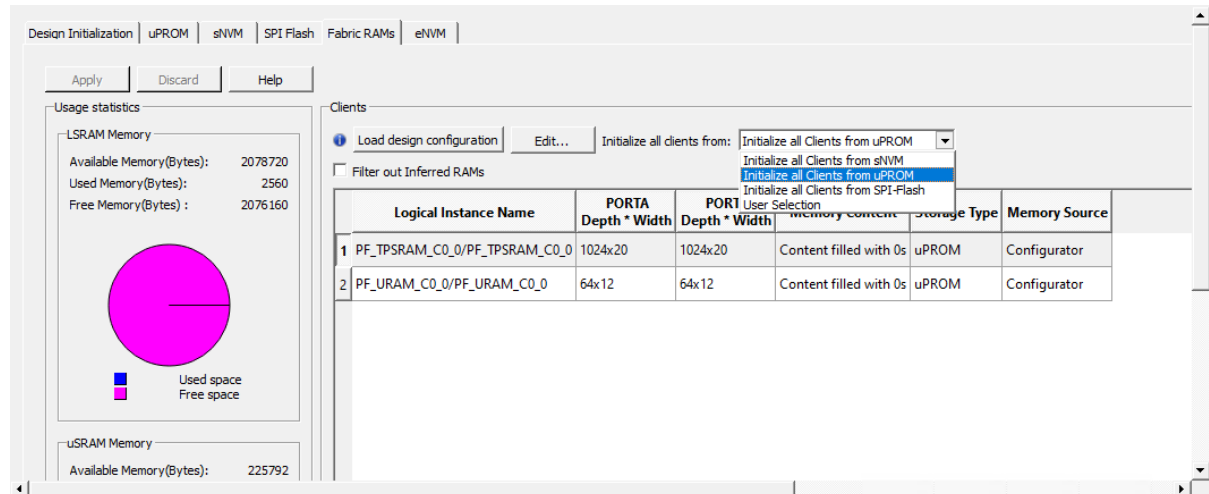
Element	Description
Load design configuration button	Resets all Fabric RAM clients to the initial configuration that was in effect the first time you clicked Apply. Clicking this button overrides any subsequent commands you applied and resets the Fabric RAM client's table.
Edit button	Edits Fabric RAMs clients.
Initialize all clients from drop-down list	Initializes clients from SmartDesign.
Filtered out Inferred RAMs check box	Filters 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 [\(Ask a Question\)](#)

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 8-30. Fabric RAMs Client Selections



8.2.5.1.1. Fabric RAMs Clients Table [\(Ask a Question\)](#)

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 8-31. Fabric RAMs Clients Table

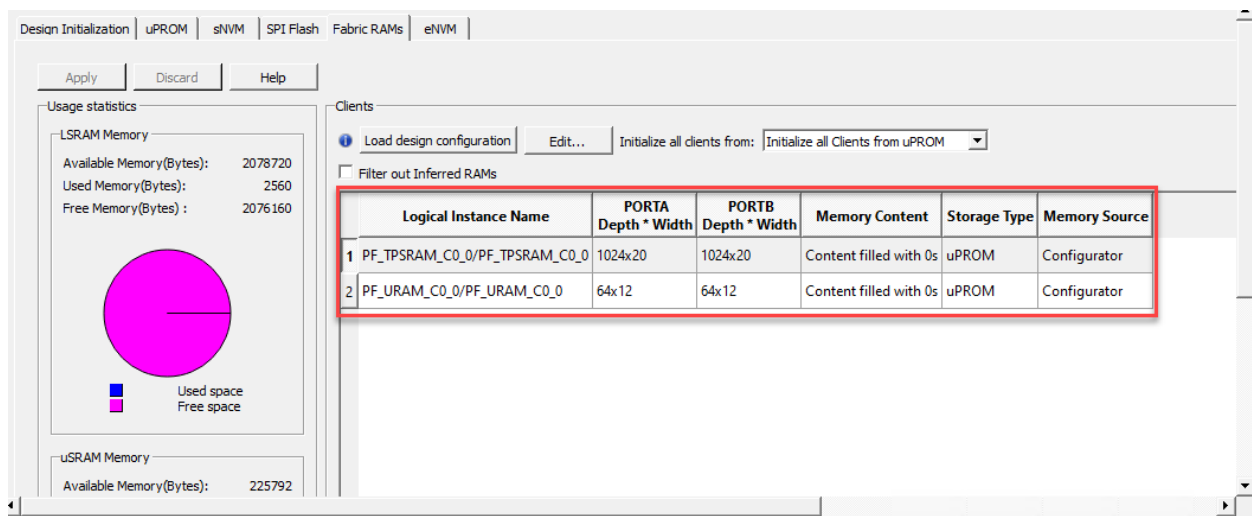


Table 8-19. Columns in the Fabric RAMs Clients Table

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 [\(Ask a Question\)](#)

If you need to change the settings for a Fabric RAM client, you can edit the client.

To edit a Fabric RAM client:

1. 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 click the **Edit** button.
 - Right-click the client you want to edit and select **Edit**.
2. When the Edit Fabric RAM Initialization dialog box appears, complete the fields (see the following figure and table).
3. Click **OK**.
4. Click the **Apply** button.

Figure 8-32. Edit Fabric RAM Initialization Client Dialog Box

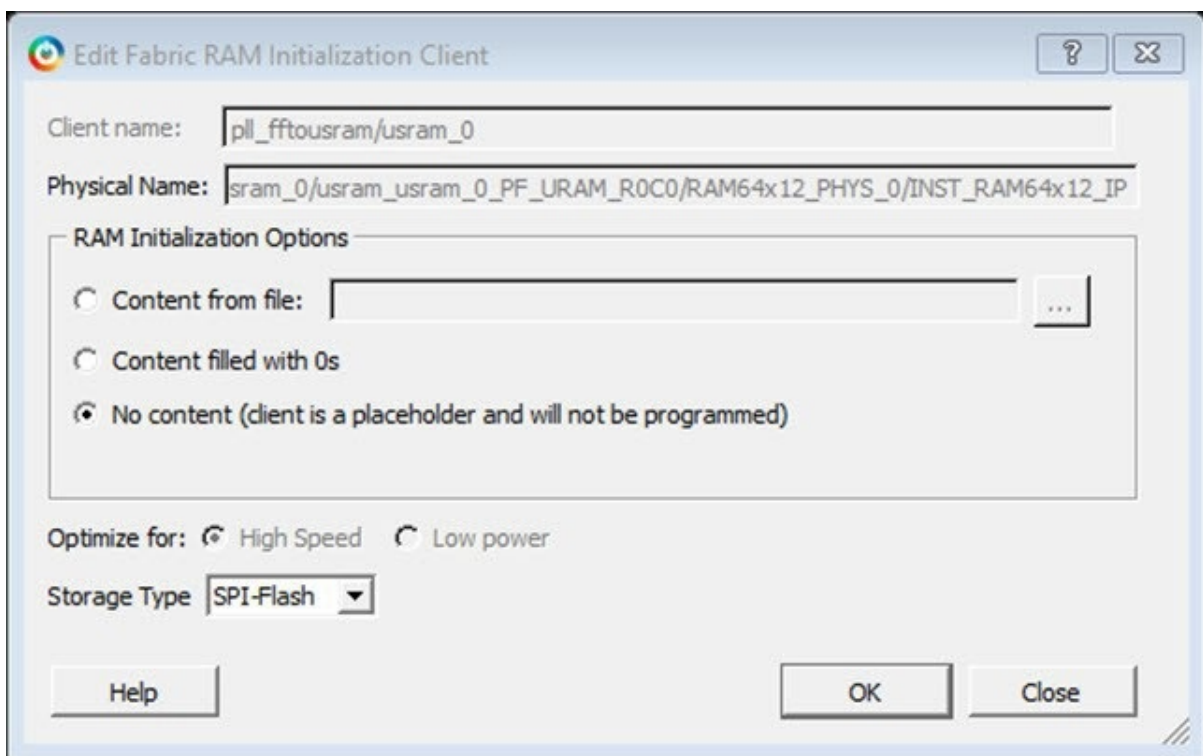


Table 8-20. Fields in the Edit Fabric RAM Initialization Client Dialog Box

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.

Table 8-20. Fields in the Edit Fabric RAM Initialization Client Dialog Box (continued)

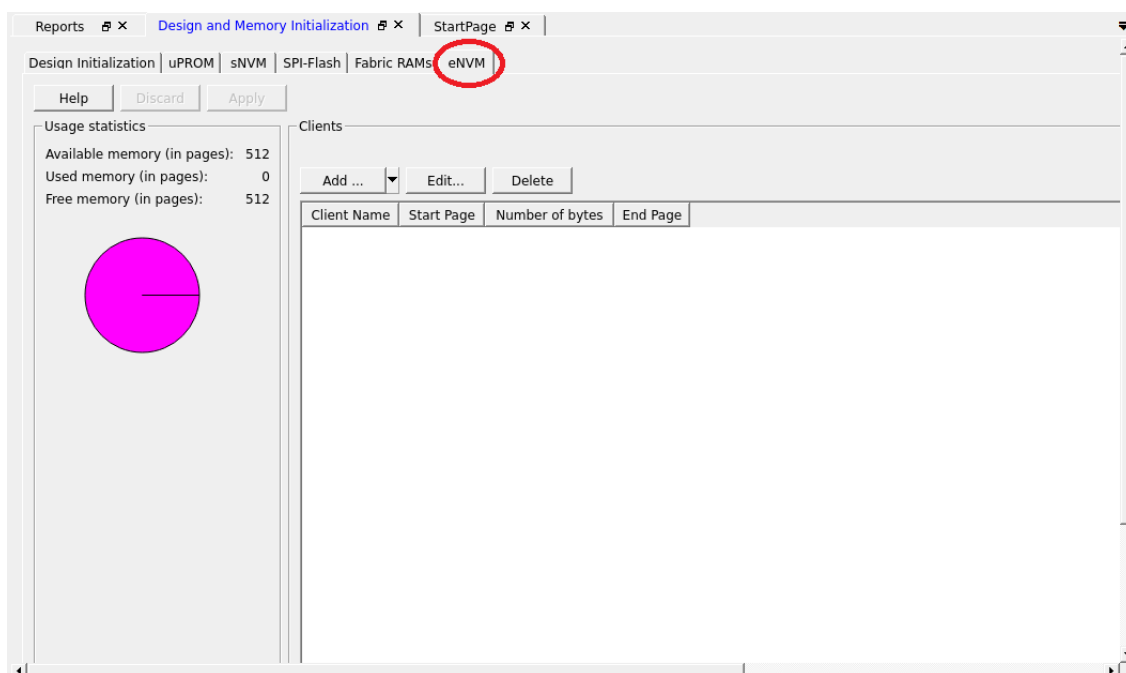
Option	Description
RAM Initialization Options	Choices are: - Content from file: Click the Browse button to go to the location of a memory file, and then import the file to the memory block. By default, the same memory file specified in the memory configurator is used. Supported memory file formats are: - Intel-HEX (*.hex) - Motorola (*.s) - Simple-Hex (*.shx) - Microsemi-Binary (*.mem) - Content Filled with 0s: Memory block is filled with zeros for initialization. - No Content: Memory block is not initialized.
Optimize for	Read-only field.
Storage Type	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: - sNVM (default) - uPROM - SPI-Flash

8.2.6. eNVM Tab (PolarFire SoC Only) [\(Ask a Question\)](#)

PolarFire SoC users can use the **eNVM** tab to configure and manage eNVM clients. Libero SoC supports 512 pages.

The procedures for adding, editing, and deleting eNVM clients are similar to those for sNVM clients. The supported eNVM client types are Plaintext Data, Boot Mode 1, and Boot Mode 3.

Figure 8-33. eNVM Tab

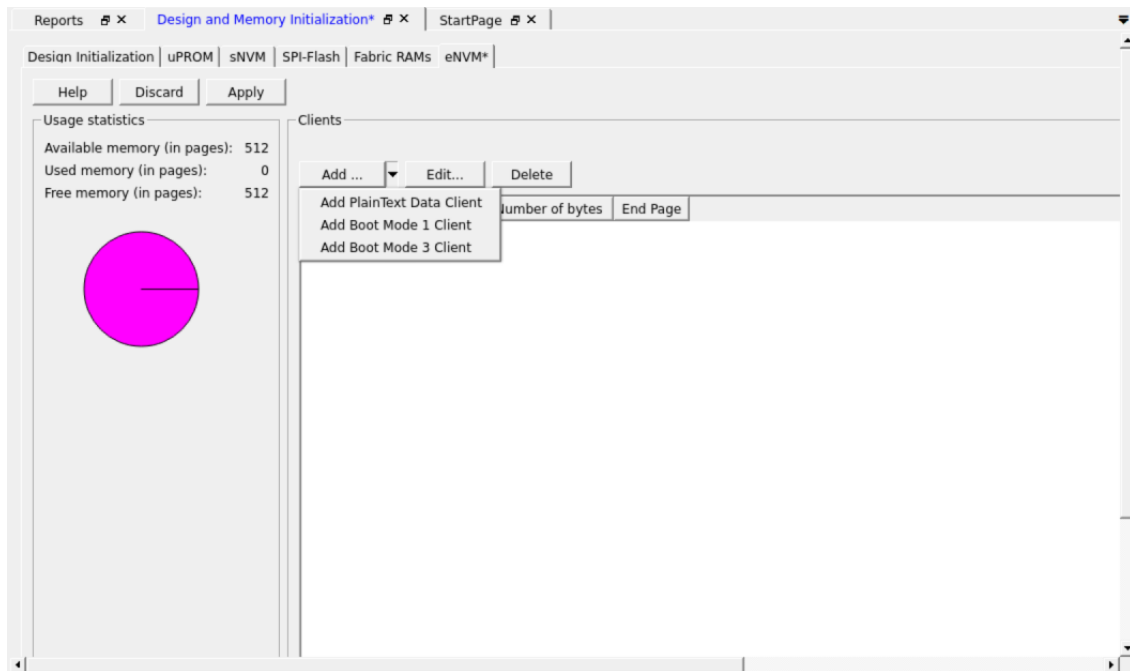


8.2.6.1. Adding eNVM Clients [\(Ask a Question\)](#)

To add eNVM clients, perform the following steps:

1. In the **eNVM** tab, click the **Add** drop-down list, and then select the client you want to add.

Figure 8-34. eNVM Client Selections



2. Complete the fields in the dialog box, and then click **OK** (see the following sections).
3. Click **Apply**. The client is added to the [eNVM clients table](#).

8.2.6.1.1. Adding a Plaintext Data Client in eNVM Memory [\(Ask a Question\)](#)

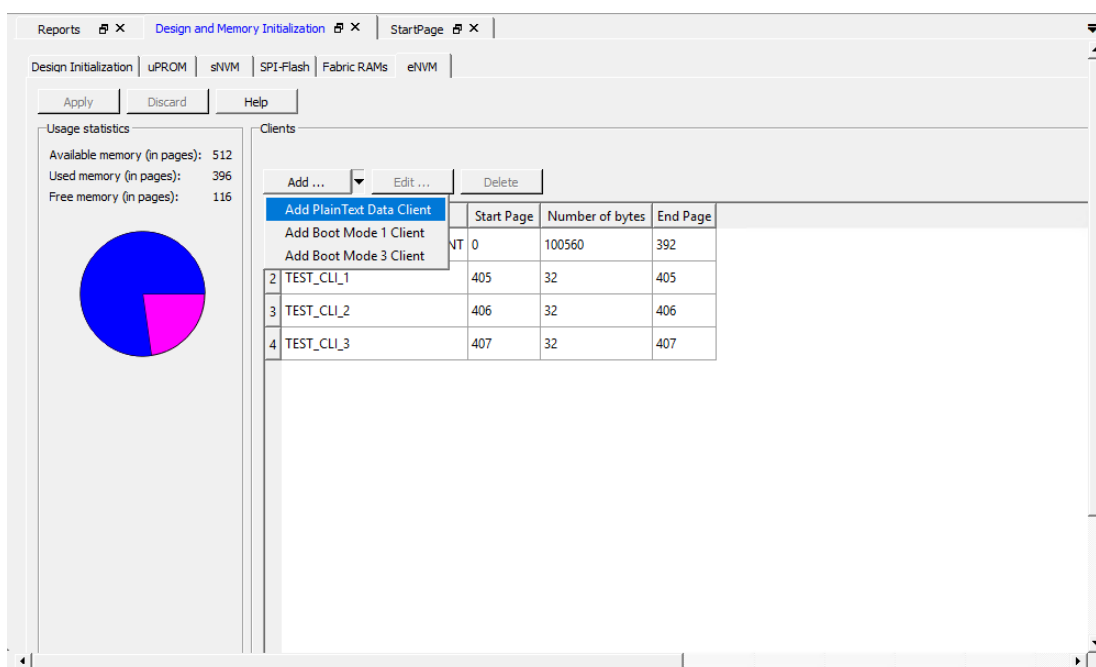
The **eNVM** tab provides an option that allows you to add a plaintext data client in eNVM memory within the **Design and Memory Initialization** tab. Using this option, you can define a simple, user-managed block of non-volatile memory that stores raw data without any protocol, structure enforcement, or controller-managed formatting.

A plaintext data client has the following features:

- Allocates a region in eNVM
- Stores user-defined raw data (binary or ASCII)
- Does not include metadata, framing, or secure encapsulation
- Is accessed directly by the system (Fabric, MSS, or both)

To add a plaintext data client in eNVM memory, perform the following steps:

1. In the **eNVM** tab, open the drop down list and select the **Add PlainText Data Client** option.



2. When the **Add PlainText Data Client** dialog box appears, complete the fields (see the following table).
3. Click **OK**.

Table 8-21. Fields in the Add PlainText Data Client Dialog Box

Field	Description
Client name	Name of the eNVM client to be added.
Content from file	Navigate to this field and select the file containing the eNVM initialization data.  Attention: 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
Base address	Base address specified in the memory file
Content filled with 0s	Populates the eNVM with zero
No content	Client is a placeholder and will not be programmed.
Start page	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, an error message appears.
Number of bytes	Total number of bytes to populate the eNVM, in decimal notation. If the number of bytes exceeds the memory size of the eNVM, an out-of-bounds error message appears.
Use content for simulation	Check if this client must be loaded for the simulation run.
Include this client in eNVM digest calculation	Check to include this client in the eNVM digest calculation if the client is updated through bitstream (JTAG, SPI slave, or SPI master programming). Client must not be included if it is updated using any other methods. If included, this client page(s) will be checked while running the VERIFY_DIGEST action.

8.2.6.1.2. Settings for Boot Mode 1 Clients [\(Ask a Question\)](#)

In this Boot mode, the Core Complex boots directly from a specified address in eNVM with no authentication. The start page of Boot mode 1 client cannot be modified.

Note: The eNVM memory can have one client only: Boot Mode 1 or [Boot Mode 3](#).

Figure 8-35. Boot Mode 1 Client Dialog Box

Table 8-22. 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. Note: 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 SoftConsole.
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, then their start page cannot be the same. Otherwise, a warning message appears. Range: 0–220 (decimal)

Table 8-22. Fields in the Boot Mode 1 Client Dialog Box (continued)

Field	Description
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
Include this client in eNVM digest calculation	Check to include this client in the eNVM digest calculation if the client is updated via bitstream (JTAG, SPI slave, or SPI master programming). Client must not be included if it is updated using any other methods. If included, this client page(s) will be checked while running the VERIFY_DIGEST action.

8.2.6.1.3. Settings for Boot Mode 3 Clients (PolarFire SoC) [\(Ask a Question\)](#)

PolarFire SoC supports Boot mode 3. In this Boot mode, you specify the start page in eNVM. You must specify the public key X and Y. See the following table.

Table 8-23. Public Key Coordinates

Field	Description
Public Key X coordinate	Secure Boot mode 3 public key elliptical curve point X coordinate
Public Key Y coordinate	Secure Boot mode 3 public key elliptical curve point Y coordinate

Note: An eNVM client can have one client only: [Boot Mode 1](#) or Boot Mode 3.

Figure 8-36. Boot Mode 3 Client Dialog Box

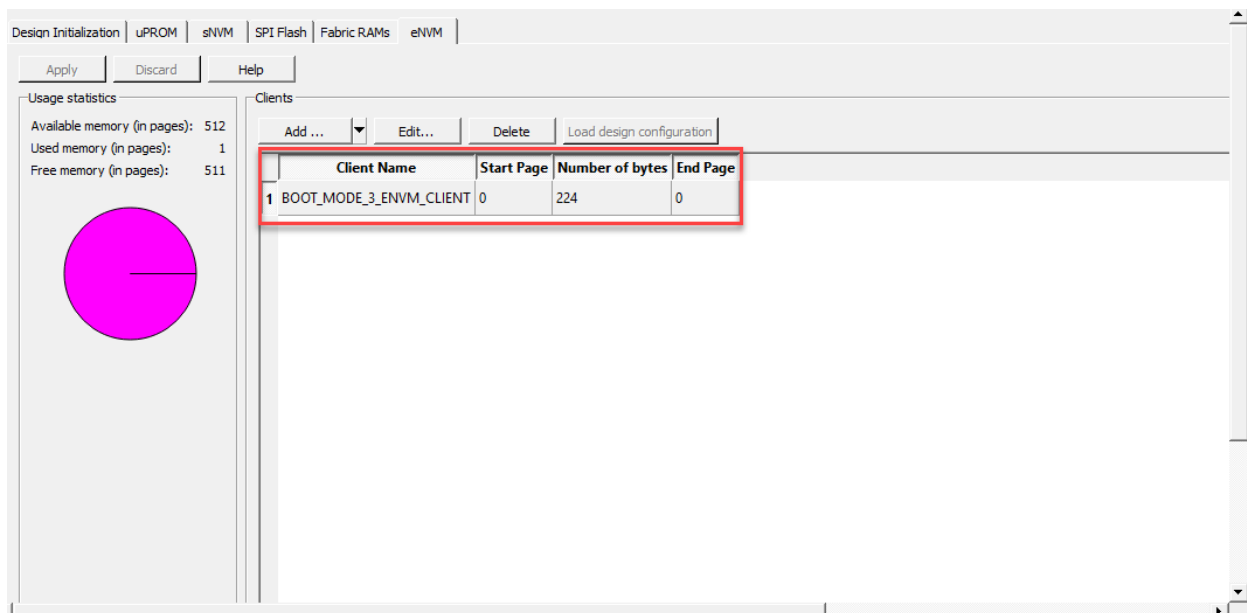
Table 8-24. Fields in the Boot Mode 3 Client Dialog Box

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. Note: 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 SoftConsole.
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 x9.org/ .
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 x9.org/ .
Include this client in eNVM digest calculation	Check to include this client in the eNVM digest calculation if the client is updated via bitstream (JTAG, SPI slave, or SPI master programming). Client should not be included if it is updated using any other methods. If included, this client page(s) will be checked when running the VERIFY_DIGEST action.

8.2.6.1.4. eNVM Clients Table [\(Ask a Question\)](#)

The following figure shows the eNVM Clients table.

Figure 8-37. eNVM Clients Table



The following table lists 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.

Table 8-25. Columns in 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 determined based on the start page you provided.

8.2.6.2. Editing eNVM Clients [\(Ask a Question\)](#)

If you need to change the settings for an eNVM client, you can edit the client.

To edit an eNVM client:

1. 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 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 [Settings for Boot Mode 1 Clients](#) or [Settings for Boot Mode 3 Clients \(PolarFire SoC\)](#)).
3. Click **OK**.
4. Click the **Apply** button.

8.2.7. Deleting eNVM Clients [\(Ask a Question\)](#)

If you no longer need an eNVM client, you can delete the client.



A warning message does not appear before you delete a client. Therefore, ensure you no longer need a client before you delete it.

To delete an eNVM client:

1. In the table at the bottom of the **eNVM** tab, perform one of the following steps:
 - Click the client you want to delete, and 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 (PolarFire) [\(Ask a Question\)](#)

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 [Initializing Design Blocks \(PolarFire and PolarFire SoC\)](#).

Table 8-26. 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 Broadcast instructions to initialize RAM's to zero's is disabled in the Design Initialization 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 Design Initialization 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 can specify the target memory for each Fabric RAM separately or for all Fabric RAMs at once in the Fabric RAMs tab, and then specify the start address of the target memory in the Design Initialization tab of the Configure Design Initialization Data and Memories tool. If there are PCIe or XCVR blocks used in the design along with Fabric RAMs with target memory type set to sNVM, a combined initialization client is created for sNVM that has the initialization sequence/instructions for the PCIe or XCVR blocks followed by Fabric RAM.

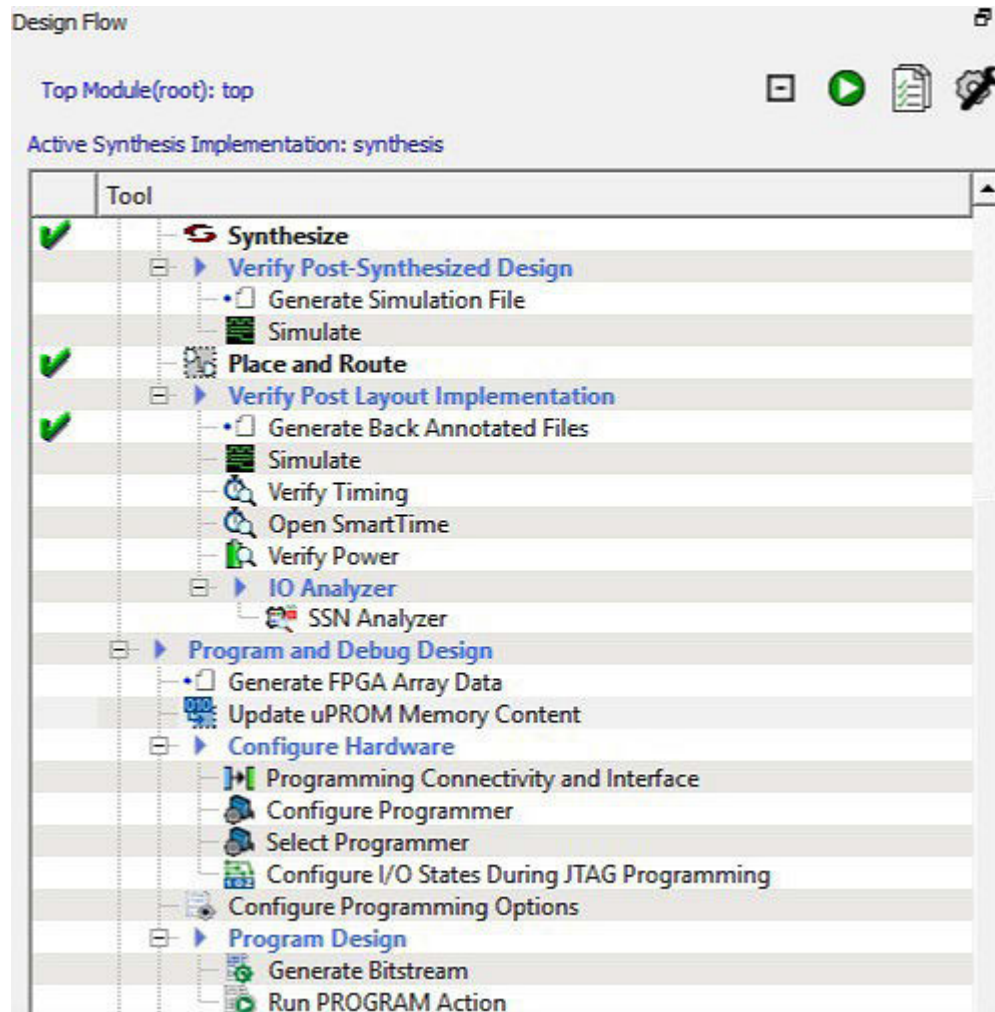
8.4. Update uPROM Memory Content (RTG4) [\(Ask a Question\)](#)

If you reserved space in the uPROM Configurator and want to make changes to the uPROM clients after Place and Route, use the Update uPROM Memory Content tool. After you update the uPROM memory content, you do not need to rerun Place and Route.

To update the uPROM memory content:

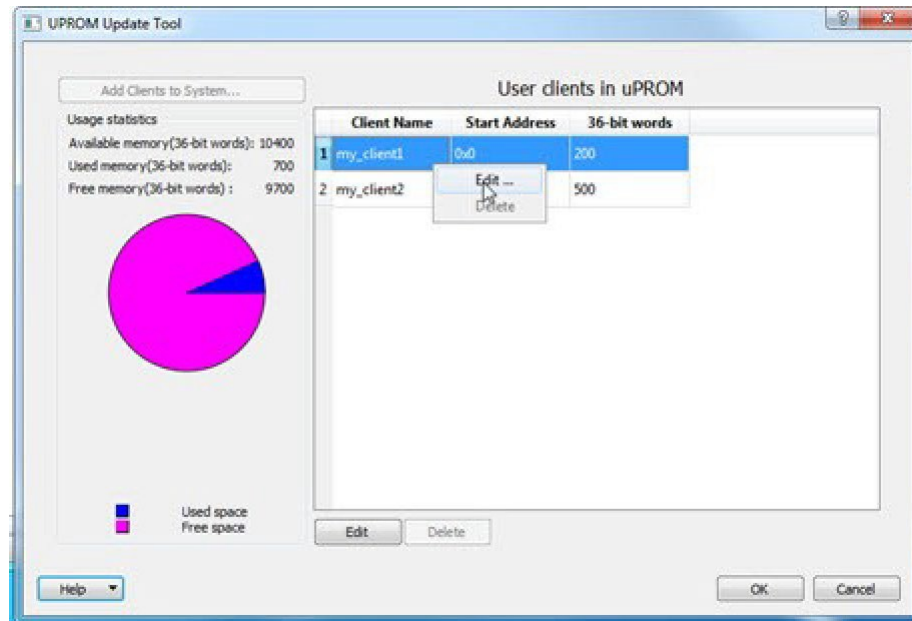
1. In the Design Flow window, right-click **Update uPROM Memory Content** and choose **Configure Options**.

Figure 8-38. Update uPROM Memory Content



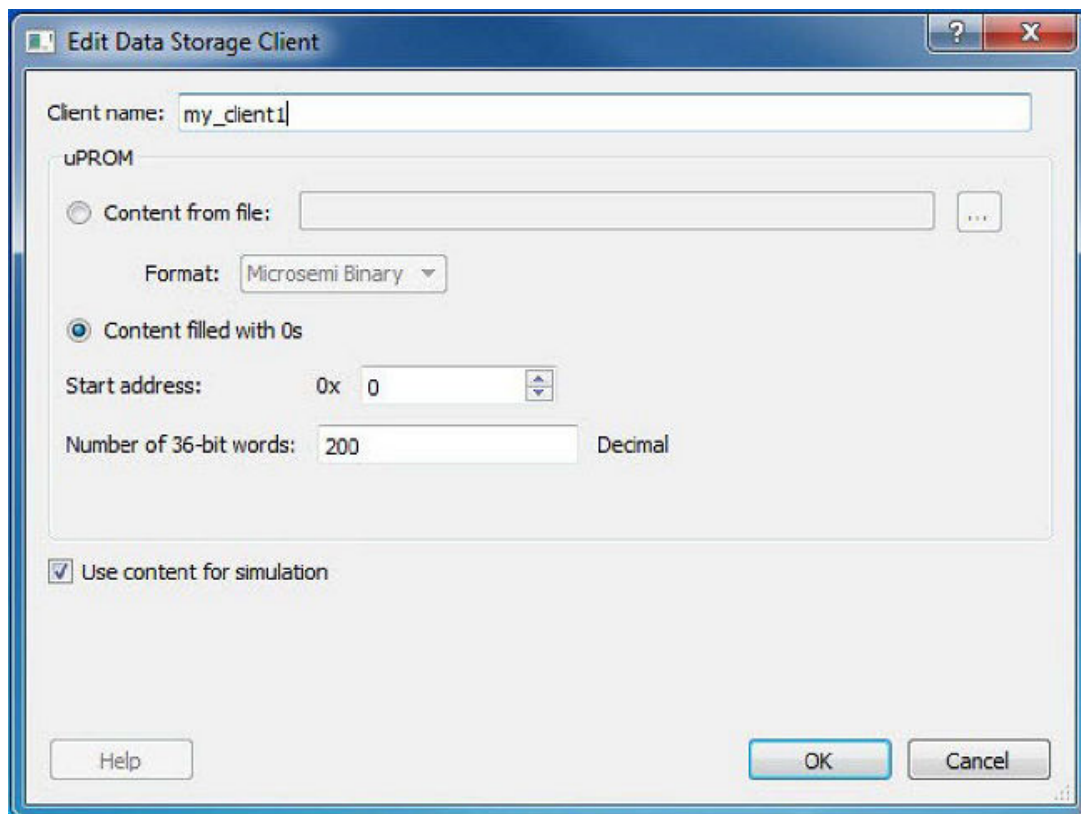
- When the uPROM Update Tool appears, right-click the Memory Client you want to update and choose **Edit**.

Figure 8-39. uPROM Update Tool



- When the Edit Data Storage Client dialog box appears, you can make the following changes:
 - Rename a client
 - Change the memory content, memory size and start address of the client
 - Decide whether use content for simulation

Figure 8-40. Edit Data Storage Client Dialog Box



4. When you finish, click **OK**.

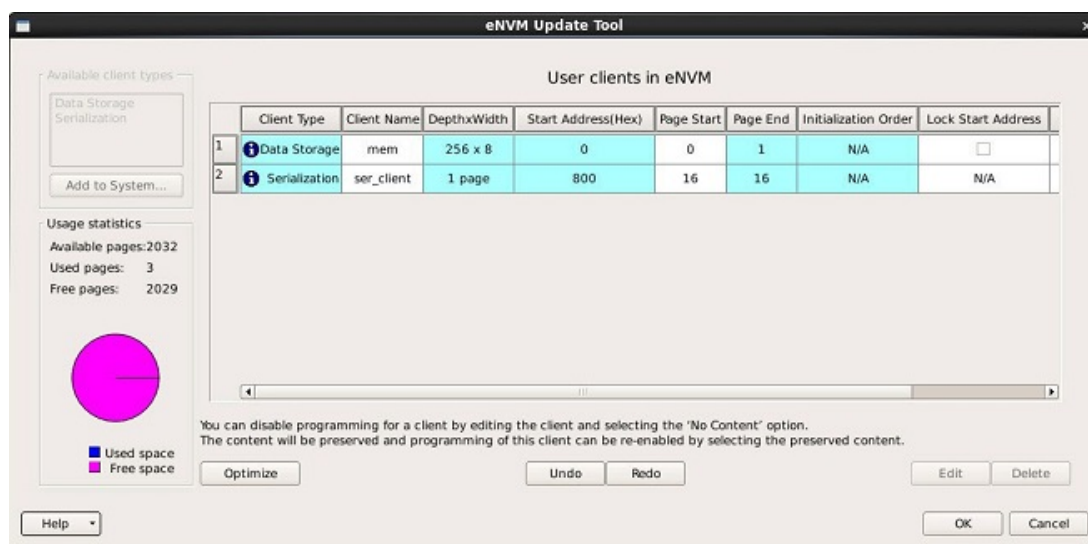
Note: You cannot use the Update uPROM tool to add or delete a client. To add or delete a client, use the uPROM Configurator to reconfigure your clients and regenerate your uPROM component and your design.

8.5. Update eNVM Memory Content (SmartFusion 2 and IGLOO 2) [\(Ask a Question\)](#)

The eNVM Update dialog box allows you to update your eNVM content without having to rerun Compile and Place and Route. It is useful, for example, if you reserved space in the eNVM configurator within the MSS for firmware development. Use the eNVM Update dialog box after you complete your firmware development and want to incorporate your updated firmware image file into the project.

To display the eNVM Update dialog box, right-click **Update eNVM Memory Content** and choose **Configure Options** or double-click **Update eNVM Memory Content** to display the dialog box.

Figure 8-41. eNVM Update Dialog Box



Note: To disable a client for programming, modify the client and select **No Content (Client is a placeholder and will not be programmed)**. The content from the memory file, serialization data file, or auto-incremented serialization content will be preserved if you later decide to enable this client for programming. Clients disabled for programming will not be included in the generated bitstream and will not be programmed.

To delete, create, or rename an eNVM client, return to the MSS/System Builder eNVM Configurator. See the [MSS Configuration - eNVM \(User Guide\)](#).

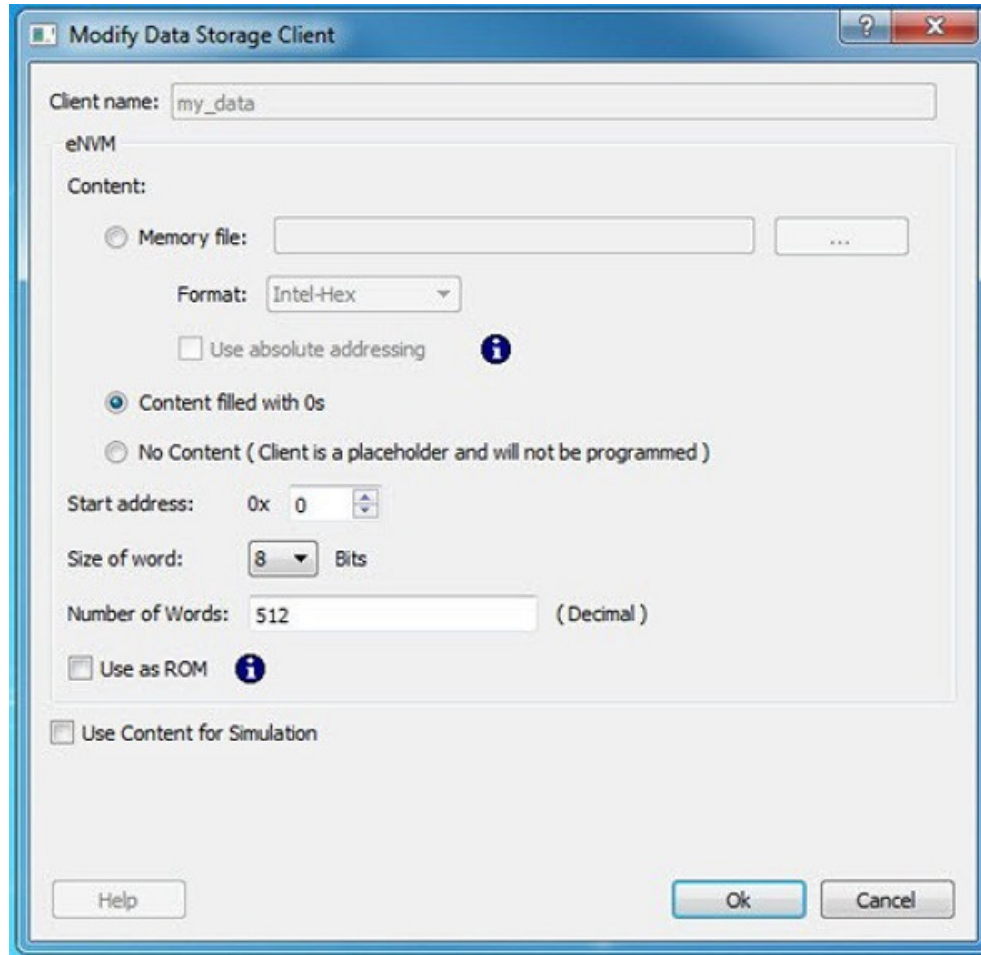
8.6. Modify Data Storage Client (SmartFusion 2 and IGLOO 2) [\(Ask a Question\)](#)

The Modify Data Storage Client dialog box allows you to import a memory file, fill eNVM content with zero's, and assign no content (eNVM as a placeholder). The last option excludes the client from the programming bitstream and does not program the client. You can also specify the start address where the data for that client starts, the word size, and the number of words to reserve for the data storage client.

To display the Modify Data Storage Client dialog box, double-click the storage client.

Note: You cannot add, delete, or rename a data storage client at this point using the Modify Data Storage Client dialog box. To make such changes, return to the MSS or System Builder eNVM configuration step.

Figure 8-42. Modify Data Storage Client Dialog Box



If you completed Place and Route and imported a memory file for the eNVM content, you do not have to rerun Compile or Place and Route. You can program or export your programming file directly. Programming generates a new programming file that includes your updated eNVM content.

8.7. Modify Serialization Client (SmartFusion 2 and IGLOO 2) [\(Ask a Question\)](#)

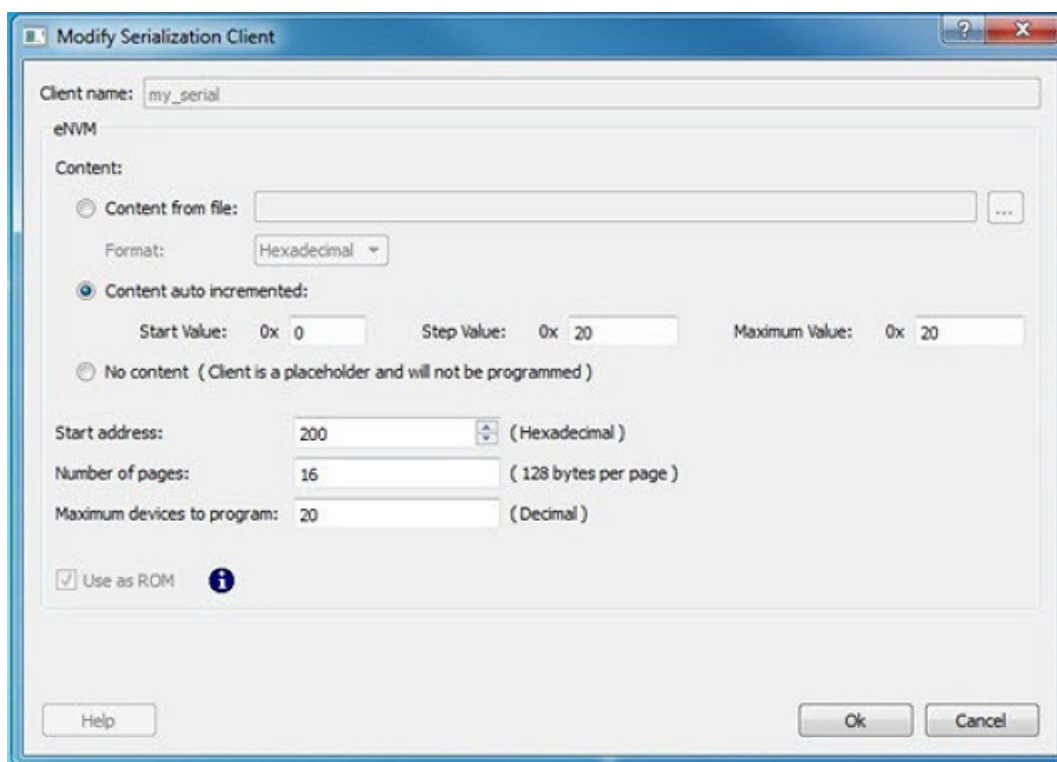
The Modify Serialization Client dialog box allows you to import a memory file, fill eNVM content with zero's, and assign no content (eNVM as a placeholder). The last option excludes the client from the programming bitstream and does not program the client. You can also specify the start address where the data for that client starts, the word size, and the number of words to reserve for the data storage client. You can also specify the start address where the data for the serialization client starts, the number of pages, and the maximum number of devices into which you want to program serialization data.

Setting a maximum number of devices to program for Serialization clients generates a programming bitstream file that has serialization content for the number of devices specified. The maximum number of devices to program must match for all serialization clients. To program a subset of the devices during production programming, use the FlashPro Express tool, which allows you to select a range of indices desired for programming for that serialization programming session. For more information, see the [Macro Library User Guide for SmartFusion2 and IGLOO2](#).

To open the Modify Serialization Client dialog box, double-click the serialization client.

Note: You cannot add, delete, or rename a serialization client in the Modify serialization client dialog box. Go to the eNVM configurator inside the MSS/HPMS Configurator or the System Builder Memory page (eNVM tab) to make these changes.

Figure 8-43. Modify Serialization Client Dialog Box



If you completed Place and Route and imported a memory file for the eNVM content, you do not have to rerun Compile or Place and Route. You can program or export your programming file directly. Programming will generate a new programming file that includes your updated eNVM content.

8.8. Configuring I/O States During JTAG Programming [\(Ask a Question\)](#)

In the Libero SoC Design Flow window, expand the Program Design and double-click **Configure I/O States During JTAG Programming**. The default state for all I/Os is Tri-state.

8.8.1. Specifying I/O States During Programming [\(Ask a Question\)](#)

Use the following procedure to specify I/O states during programming or while exporting a bitstream.

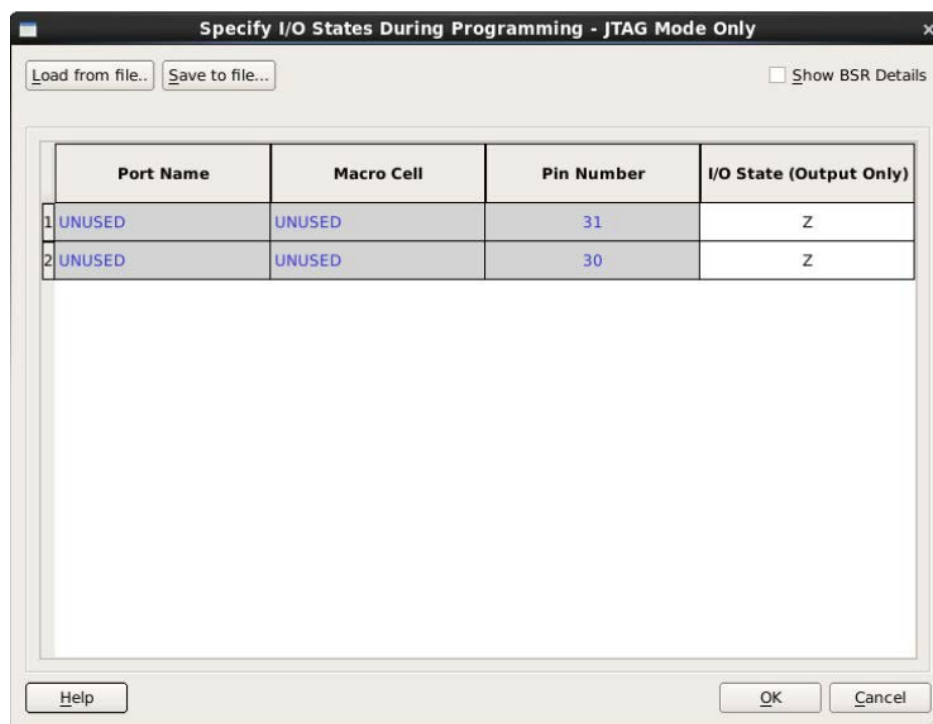
1. Click a column header to sort the entries by that header, and then select the I/Os you want to modify.
2. Set the I/O output state using either basic I/O settings to accept the default I/O settings for your pins (see the following table) or [custom I/O settings](#) to customize the settings for each 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.

Table 8-27. Basic I/O State Settings (continued)

Setting	Description
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	(PolarFire) I/O is tristated with weak pull-up (10k Ω).

Figure 8-44. I/O States During Programming Window

- Click **OK** to save your settings.

8.8.1.1. Configuring Custom I/O Settings [\(Ask a Question\)](#)

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 8-45. I/O States During Programming Dialog Box

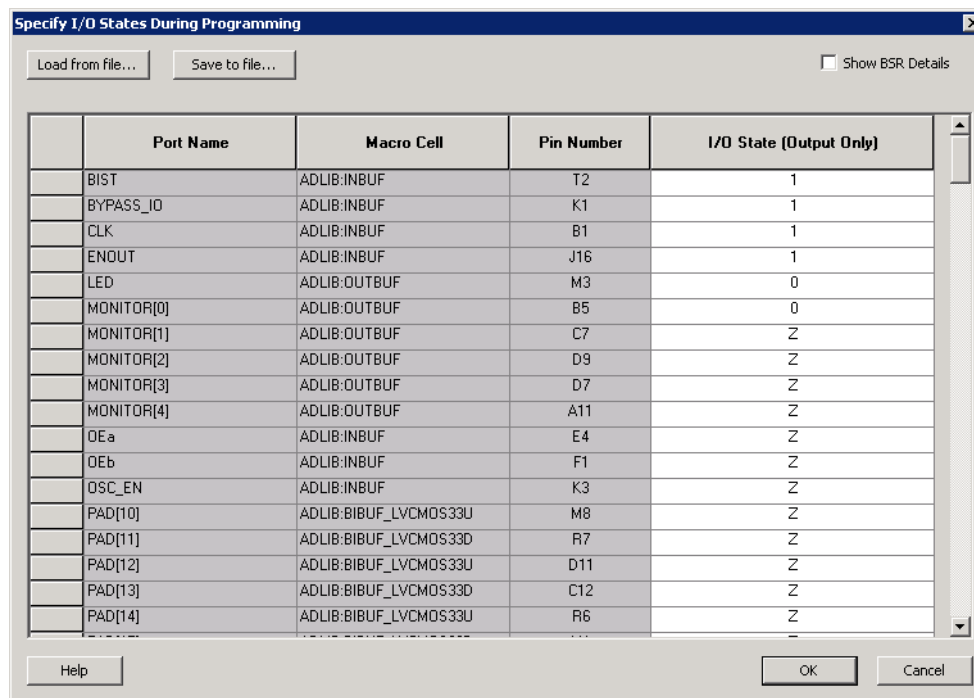


Table 8-28. Elements in the I/O States During Programming Dialog Box

Element	Description
Load from file button	Loads an I/O settings (*.ios) file you can use to import saved custom settings for your I/Os. The exported IOS file has the following format: - Used I/Os have a file entry with the following format: <pre>set_prog_io_state -portName {<design_port_name>} -input <value> -outputEnable <value> - output <value></pre> - Unused I/Os have a file entry with the following format: <pre>set_prog_io_state -pinNumber {<device_pinNumber>} -input <value> -outputEnable <value> - output <value></pre> In the preceding formats, <value> is one of the values shown in the key in the Default I/O Output Settings following table.
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 following Default I/O Output Settings table). This column header changes to Boundary Scan Register if you select the Show BSR Details check box.

Table 8-28. Elements in the I/O States During Programming Dialog Box (continued)

Element	Description
Boundary Scan Registers - Enabled with Show BSR Details	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 following BSR Details I/O Output Settings table). Examples: - To Tri-State a pin during programming, set Output Enable to 0; the Don't Care indicates that the other two values are immaterial. - 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.

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
1 (High)	0	1	1
Last_Known_State	Last_Known_State	Last_Known_State	Last_Known_State

Table Key:

- 1: I/Os are set to drive out logic High.
- 0: 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 tri-stated.

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: I/Os are set to drive out logic High.
- 0: 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: 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: I/O is tri-stated.

The following figure shows an example of Boundary Scan Register settings.

Figure 8-46. I/O States During Programming Dialog Box

Specify I/O States During Programming

Load from file... Save to file... ☒ Show BSR Details

	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
	ENDOUT	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

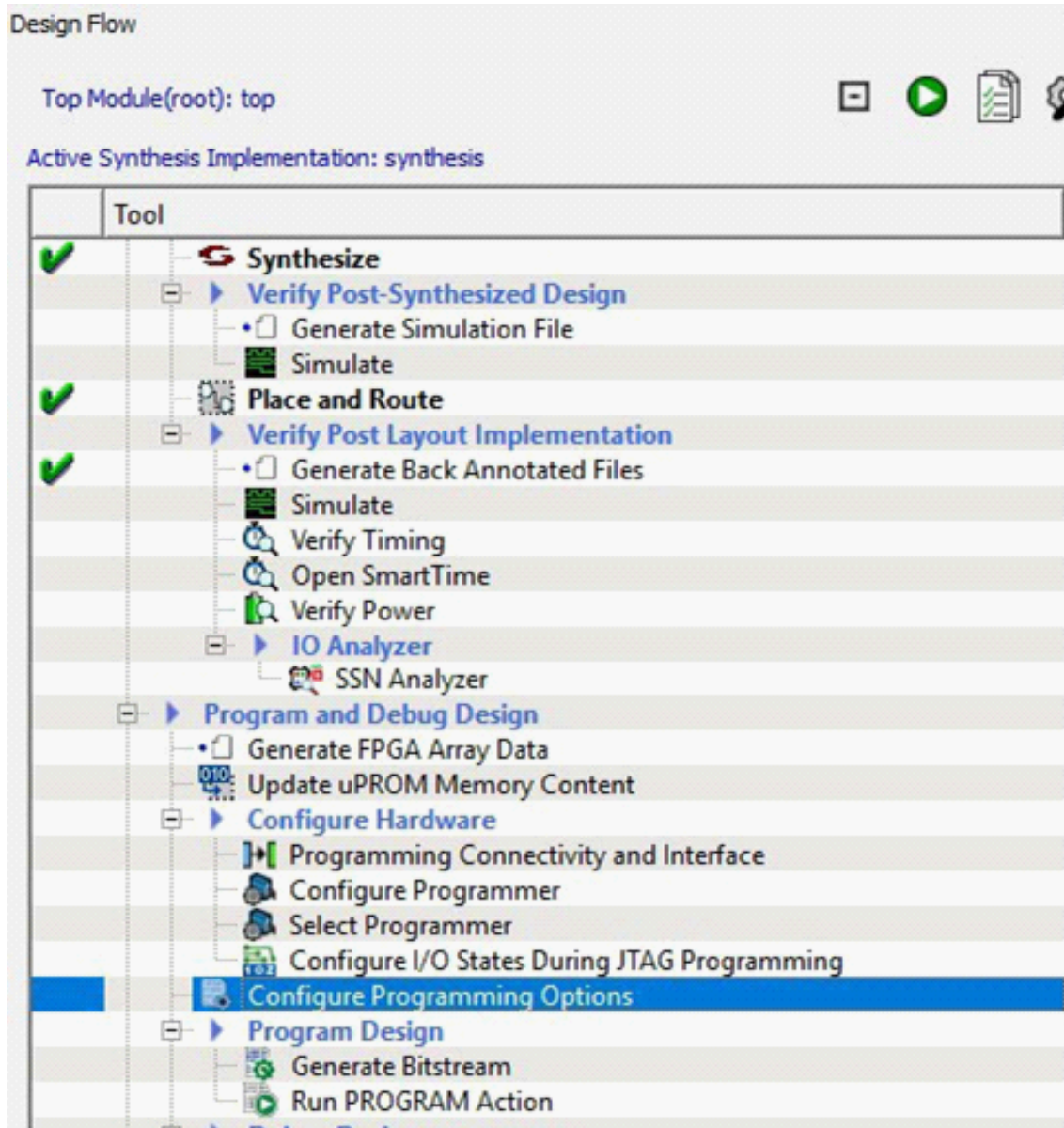
Help OK Cancel

8.9. Configuring Programming Options [\(Ask a Question\)](#)

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 8-47. Selecting Configure Programming Options



8.9.1. PolarFire Programming Options [\(Ask a Question\)](#)

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 8-48. PolarFire Programming Options

Configure Programming Options

Design name: fftousram_new

Design version (number between 0 and 65535): 3

⚠ Back Level version (number between 0 and 65535): 3

Silicon signature (max length is 8 HEX chars): 0x2

Help OK 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 Update Policy 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 Update Policy 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.

8.9.2. SmartFusion 2 and IGLOO 2 Programming Options [\(Ask a Question\)](#)

The following figure shows the programming options for SmartFusion 2 and IGLOO 2. 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 8-49. SmartFusion 2 and IGLOO 2 Programming Options

Table 8-32. SmartFusion 2 and IGLOO 2 Programming Options

Option	Description
Design name	Read-only field that identifies your design.
Design version	Design Version used for Auto Update Programming or for Back Level protection. Enter a number between 0 and 65535 for the design version.
Silicon signature	Enter up to eight hexadecimal characters.
Enable Auto Update	Click to auto update the SPI update image at power-up. Auto update compares your update SPI image Design Version against the Design Version programmed in the device, and then auto updates the programming on your SPI update Image if the: • Device is programmed and • Update SPI image Design Version is greater than the Design Version on the device Auto Recovery also allows the device to automatically reprogram itself if there is a power failure during programming. Note: If this option is enabled, the programming recovery option must also be enabled, which disables this check box.
Enable Programming Recovery	Click to enable programming recovery in case a power failure occurs during programming. Note: Programming Recovery cannot be updated with _UEK1 or _UEK2 programming files. Only the master programming file can be used.

Table 8-32. SmartFusion 2 and IGLOO 2 Programming Options (continued)

Option	Description
SPI clock frequency	Sets your SPI clock frequency. SPI is a full duplex, four-wire synchronous transfer protocol that supports programmable clock polarity (SPO) and clock phase (SPH). The state of the SPO and SPH control bits decides the data transfer modes. For more information, see the SmartFusion 2 Microcontroller Subsystem User's Guide or the IGLOO 2 High Performance Memory Subsystem User's Guide. Choices are in MHz: • 1.00 • 2.08 • 3.13 • 4.16 • 5.00 • 6.25 • 8.30 • 12.50
SPI data transfer mode	Sets your SPI data transfer mode for SPO and SPH. The SPO control bit determines the polarity of the clock and SPS defines the slave select behavior. SPS is hardcoded to b'1 and cannot be changed. The SPH control bit determines the clock edge that captures the data. For more information, see the SmartFusion 2 Microcontroller Subsystem User's Guide or the IGLOO 2 High Performance Memory Subsystem User's Guide.

Note: SPI file programming for Auto Programming, Auto Update (IAP), Programming Recovery, and IAP/ISP Services can program security only once with the master file. Update files cannot update the security settings. In addition, Programming Recovery, Silicon Signature, Firewall, and Tamper Macro can be programmed with the master file only and cannot be updated.

8.9.3. RTG4 Programming Options [\(Ask a Question\)](#)

RTG4 programming options are organized into two sections in the **Configure Programming Options** dialog box:

- The top section always displays the three options described in the following table.
- The **Programming Bitstream Settings** section displays options based on whether **System controller suspend mode** is enabled in the project's device settings and whether **One-Time Programmable (OTP)** is selected.

Table 8-33. Common RTG4 Programming Elements

Element	Description
Design name field	Read-only field that identifies your design
Design version field	Enter a number between 0 and 65535 for the design version
Silicon signature field	Enter up to eight hexadecimal characters
Reset to default button	Click to reset settings to their default values

8.9.3.1. Programming Options with OTP Mode Disabled [\(Ask a Question\)](#)

The following figures show the programming options when the **One-Time Programmable (OTP)** option is disabled. The following table describes all the options.



Attention: When **System controller suspend mode** is enabled in **Device Settings**, the check boxes used to disable individual programming interfaces and actions are automatically selected and disabled (grayed out), as required for this mode. For RTG4 devices, the typical configuration is to enable **System controller suspend mode** and leave **One-Time Programmable (OTP)** disabled.

Figure 8-50. RTG4 Programming Options: SCSM Enabled, OTP Not Selected, All Other Check Boxes Selected

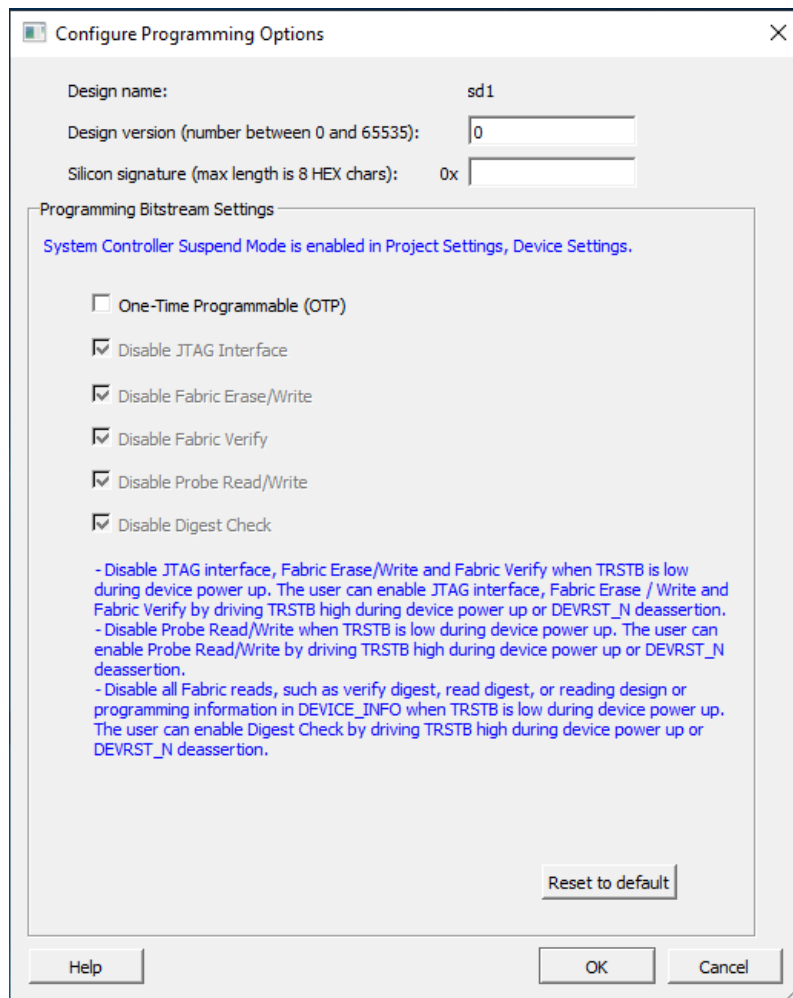


Figure 8-51. RTG4 Programming Options: SCSM Disabled, All Check Boxes Not Selected

Configure Programming Options

Design name: sd1

Design version (number between 0 and 65535): 0

Silicon signature (max length is 8 HEX chars): 0x

Programming Bitstream Settings

System Controller Suspend Mode is disabled in Project Settings, Device Settings. For high-reliability applications, ensure that System Controller Suspend Mode is enabled before generating production/flight design bitstreams.

☐ One-Time Programmable (OTP)

☐ Disable JTAG Interface

☐ Disable Fabric Erase/Write

☐ Disable Fabric Verify

☐ Disable Probe Read/Write

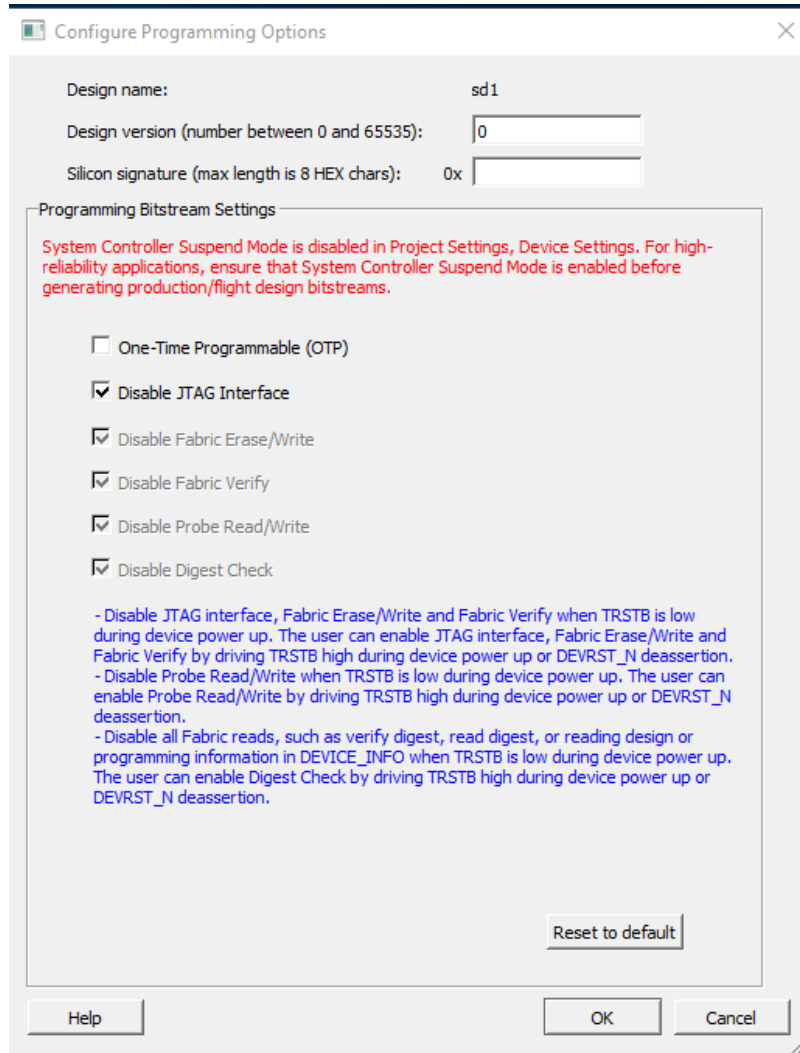
☐ Disable Digest Check

- Enable JTAG interface, Fabric Erase/Write and Fabric Verify.
- Enable Probe Read/Write.
- Enable Digest Check.

Reset to default

Help OK Cancel

Figure 8-52. RTG4 Programming Options: SCSM Disabled, OTP Not Selected, and Disable JTAG Interface Selected



The following table describes the fields shown in the preceding dialog boxes. By default, SCSM is enabled and the default settings correspond to those shown in [Figure 8-50](#). If SCSM is disabled, the default settings correspond to those shown in [Figure 8-51](#).

Table 8-34. RTG4 Programming Options—Programming Bitstream Settings with OTP Disabled

Field	Description
One-Time Programmable (OTP)	Makes the device one-time programmable. The following options are available when this option is not selected/enabled.
Disable JTAG Interface	Disables the JTAG interface when TRSTB is LOW during device power-up. You can enable the JTAG interface by driving TRSTB HIGH during device power-up or DEVRST_N de-assertion. Note: Selecting this option automatically disables all other settings in the dialog box.
Disable Fabric Erase/Write	Disables Fabric Erase/Write when TRSTB is LOW during device power-up. You can enable Fabric Erase/Write by driving TRSTB HIGH during device power-up or DEVRST_N de-assertion.
Disable Fabric Verify	Disables Fabric Verify when TRSTB is LOW during device power-up. You can enable Fabric Verify by driving TRSTB HIGH during device power-up or DEVRST_N de-assertion.

Table 8-34. RTG4 Programming Options—Programming Bitstream Settings with OTP Disabled (continued)

Field	Description
Disable Probe Read/Write	Disables Probe Read/Write when TRSTB is LOW during device power-up. You can enable Probe Read/Write by driving TRSTB HIGH during device power-up or DEVRST_N de-assertion.
Disable Digest Check	Disables all Fabric reads, such as verify digest, read digest, or reading design or programming information in DEVICE_INFO when TRSTB is LOW during device power-up. You can enable Digest Check by driving TRSTB HIGH during device power-up or DEVRST_N de-assertion.

8.9.3.2. Programming Options with OTP Mode Enabled [\(Ask a Question\)](#)

The following figures show the programming options when **One-Time Programmable (OTP)** is enabled. The following table describes all the options.

Figure 8-53. RTG4 Programming Options: SCSM Enabled, OTP Enabled

Configure Programming Options

Design name: sd1

Design version (number between 0 and 65535): 0

Silicon signature (max length is 8 HEX chars): 0x

Programming Bitstream Settings

System Controller Suspend Mode is enabled in Project Settings, Device Settings.

- ☒ One-Time Programmable (OTP)
- ☒ Permanently disable JTAG Interface
- ☒ Permanently disable Fabric Erase/Write
- ☒ Permanently disable Fabric Verify
- ☒ Permanently disable Probe Read/Write
- ☒ Permanently disable Digest Check

You have selected to make the device One-Time programmable (OTP) combined with enabling System Controller Suspend Mode. Once programmed with the generated bitstream, no further JTAG access will be possible. The device cannot be reprogrammed, erased, verified, debugged, or have any digest or device info read-back. Boundary scan JTAG instructions will no longer function since the JTAG interface is permanently disabled. Powering-up or releasing DEVRST_N with JTAG TRSTB driven high will NOT exit System Controller Suspend Mode.

Reset to default

Help OK Cancel

Figure 8-54. RTG4 Programming Options: SCSM Disabled, OTP and Permanently disable Fabric Erase/Write Selected

Configure Programming Options

Design name: sd1

Design version (number between 0 and 65535): 0

Silicon signature (max length is 8 HEX chars): 0x

Programming Bitstream Settings

System Controller Suspend Mode is disabled in Project Settings, Device Settings. For high-reliability applications, ensure that System Controller Suspend Mode is enabled before generating production/flight design bitstreams.

☒ One-Time Programmable (OTP)

☐ Permanently disable JTAG Interface

☒ Permanently disable Fabric Erase/Write

☐ Permanently disable Fabric Verify

☐ Permanently disable Probe Read/Write

☐ Permanently disable Digest Check

You have selected to make the device One-Time programmable (OTP) without System Controller Suspend Mode enabled.

Once programmed with the generated bitstream, you will NOT be able to reprogram or erase the device. You will be able to verify the fabric and digest, use SmartDebug to perform probe read/write, and read the digest and DEVICE_INFO.

In One-Time programmable (OTP) mode, without System Controller Suspend Mode enabled, instead of being suspended, the System Controller will enter its idle state after completing the device power-up to functional process, to enable support for the remaining JTAG actions available in One-Time programmable (OTP) mode.

During flight, the JTAG_TRSTB device pin must be asserted LOW, such as using an external 1K-Ohm pulldown resistor to GND, to hold the JTAG state machine in reset and prevent accidental JTAG commands.

Reset to default

Help OK Cancel

Table 8-35. RTG4 Programming Options—Programming Bitstream Settings with OTP Enabled

Field	Description
One-Time Programmable (OTP)	Permanently configures the device as one-time programmable (OTP). If you enable One-Time Programmable (OTP) without System controller suspend mode (see Figure 8-54) programming a bitstream that enables OTP permanently prevents the device from being erased or reprogrammed. After OTP is enabled, you can still perform the following actions: - Run the VERIFY and VERIFY_DIGEST programming operations. - Use SmartDebug with probes for device debugging. - Read the Fabric digest. To permanently disable these actions, enable the additional options described in this table. These options are available only when One-Time Programmable (OTP) is enabled and System controller suspend mode is disabled. If you enable One-Time Programmable (OTP) with System controller suspend mode (see Figure 8-53), all subsequent bitstream options are automatically enabled. After the generated bitstream is programmed, JTAG access to the device is permanently disabled. This table shows the default settings that apply when One-Time Programmable (OTP) is enabled and System controller suspend mode is disabled.
Permanently disable JTAG Interface	Permanently disables the JTAG interface so it can no longer be used for programming, debugging, or device access. Selecting this option prevents unauthorized readback, reprogramming, or tampering through JTAG. This setting is irreversible once applied, and the device becomes non-JTAG-accessible. Default: Disabled (not checked)
Permanently disable Fabric Erase/Write	Permanently disables fabric erase and write operations, preventing the device from being reprogrammed or erased. Once selected, the existing configuration in the fabric cannot be erased or updated. This setting is irreversible and is typically used to lock the design and protect it from unauthorized modification. Default: Enabled (checked)
Permanently disable Fabric Verify	Permanently disables fabric verify operations, preventing any verification or readback of the programmed fabric. Once enabled, the device configuration cannot be checked for integrity or compared against the original design. This setting is irreversible and is typically used to enhance security by blocking design verification or inspection. Default: Disabled (not checked)
Permanently disable Probe Read/Write	Permanently disables probe read and write access, preventing any interaction with on-chip debug probes. Once enabled, SmartDebug probe-based debugging, data capture, and signal observation are no longer possible. This setting is irreversible and is typically used to protect the design from unauthorized inspection. Default: Disabled (not checked)
Permanently disable Digest Check	Permanently disables the digest clock, preventing any digest-related operations from being performed on the device. Default: Disabled (not checked)

8.9.4. Interaction Between RTG4 SCSM and OTP [\(Ask a Question\)](#)

RTG4 design bitstreams typically enable **System controller suspend mode** without enabling the **One-Time Programmable (OTP)** option. When such a bitstream is programmed into the device, the internal system controller completes device startup at power-up or when DEVRST_N is released, and samples the JTAG TRSTB input.

The following are the possible device behaviors depending on the state of the JTAG TRSTB signal during startup:

- If TRSTB is low during startup, the system controller enters a suspended static state after initialization and remains suspended until the next power cycle or until DEVRST_N is released. In this state, the JTAG interface is disabled, and all JTAG operations result in a scan chain failure.
- If TRSTB is high during startup, the system controller remains active, allowing JTAG operations such as programming and debugging. For more information, see the RTG4 System Controller User Guide.

If an RTG4 design bitstream with **One-Time Programmable (OTP)** enabled is programmed into the device, the device cannot be erased or reprogrammed, regardless of the state of the JTAG TRSTB signal during power-up.

The following list describes the effects of enabling **One-Time Programmable (OTP)** with different **System controller suspend mode** settings:

- If **One-Time Programmable (OTP)** is used without **System controller suspend mode**, you can selectively permanently disable additional JTAG actions, such as Fabric Verify, Probe Read/Write, Digest Read/Verify, and reading DEVICE_INFO. These additionally disabled actions become permanent. For more information, see [Figure 8-54](#).
- If **One-Time Programmable (OTP)** is enabled together with **System controller suspend mode**, the device becomes permanently one-time programmable, and the JTAG interface is permanently disabled regardless of the JTAG TRSTB state during power-up. All JTAG access, including boundary-scan instructions, is disabled. At power-up or when DEVRST_N is released, the system controller completes device startup and then enters a suspended static state, remaining suspended until the next power cycle or when DEVRST_N is released. See [Figure 8-53](#).

The following table summarizes the effects of enabling RTG4 **System controller suspend mode** in combination with **One-Time Programmable (OTP)**.

Table 8-36. RTG4 SCSM Interactions with OTP

Feature	SCSM Disabled, OTP Disabled	SCSM Enabled, OTP Disabled	SCSM Disabled, OTP Enabled	SCSM Enabled, OTP Enabled
JTAG Interface	User configurable in Programming Bitstream Settings	Disabled if TRSTB is low during power-up or when DEVRST_N is released	Permanently enabled	Permanently disabled
Fabric Erase/Write	User configurable in Programming Bitstream Settings	Disabled if TRSTB is low during power-up or when DEVRST_N is released	Permanently disabled	Permanently disabled
Fabric Verify	User configurable in Programming Bitstream Settings	Disabled if TRSTB is low during power-up or when DEVRST_N is released	Permanently user configurable in Programming Bitstream Settings	Permanently disabled
Probe Read/Write	User configurable in Programming Bitstream Settings	Disabled if TRSTB is low during power-up or when DEVRST_N is released	Permanently user configurable in Programming Bitstream Settings	Permanently disabled
Digest and Device Info	User configurable in Programming Bitstream Settings	Disabled if TRSTB is low during power-up or when DEVRST_N is released	Permanently user configurable in Programming Bitstream Settings	Permanently disabled

8.10. Configuring Security [\(Ask a Question\)](#)

The following sections describe how to configure security for your device.

- PolarFire and Polar Fire SoC users: see [Configure Security Wizard \(PolarFire and PolarFire SoC\)](#).
- SmartFusion 2 and IGLOO 2: see [Configure Security Policy Manager \(SmartFusion 2 and IGLOO 2\)](#).

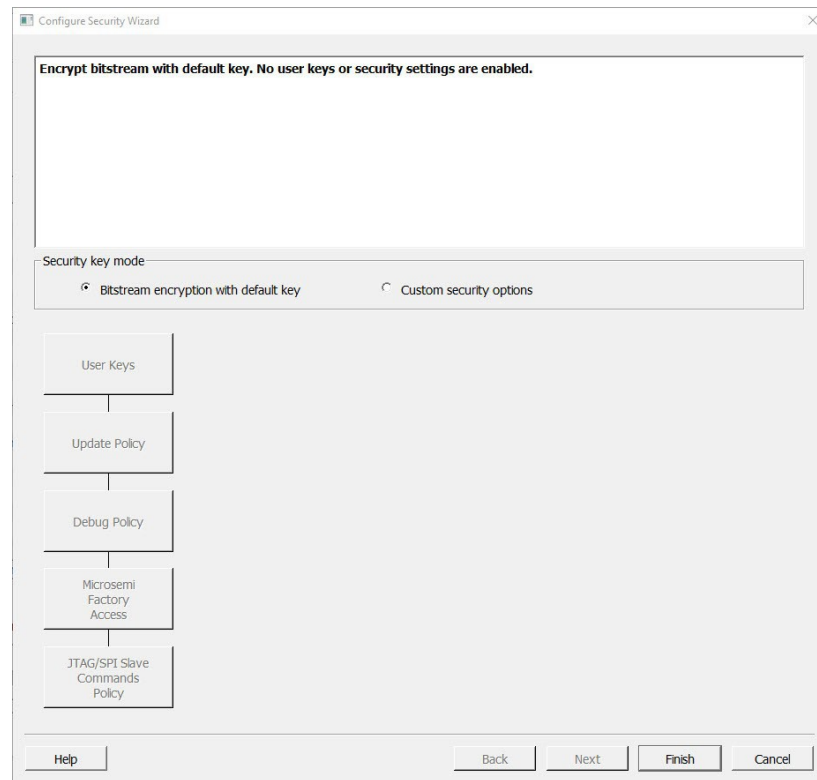
➔ Important: If you use an SPI file generated with Libero v2022.2 or later version with custom security to create or edit the SPI bitstream client, then an error message tells you that the SPI file cannot contain security.

8.10.1. Configure Security Wizard (PolarFire and PolarFire SoC) [\(Ask a Question\)](#)

The Configure Security Wizard guides PolarFire and PolarFire SoC users 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](#)

Figure 8-55. Configure Security Wizard



The following table describes the elements in the Configure Security Wizard.

Table 8-37. 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.

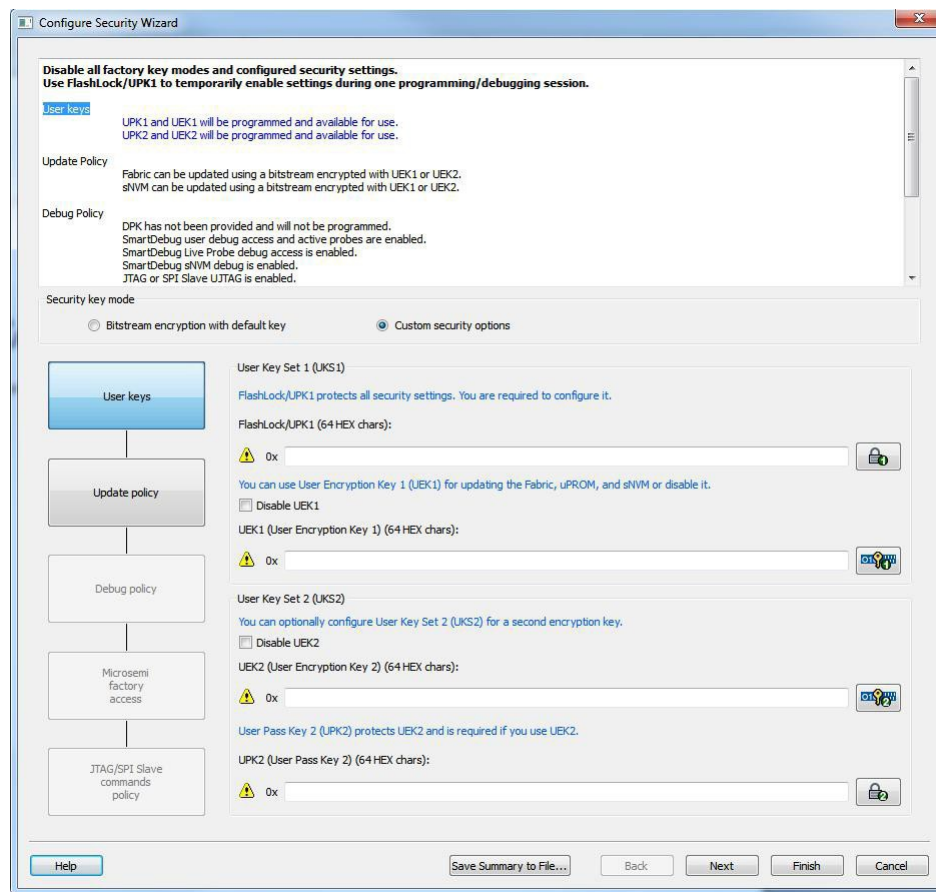
Table 8-37. Elements in the Configure Security Wizard (continued)

Element	Description
Security key mode	Two security key modes are available: - 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. - Custom security Mode: Configures custom security keys and settings. All steps are enabled, as are the Next and Back buttons.
Back	Click to return to previous step.
Next	Click to proceed to next step.
Finish	Click to skip steps and complete the configuration.

8.10.1.1. Step 1: User Keys [\(Ask a Question\)](#)

In step 1 of the Configure Security Wizard, you configure user keys. All keys are 256 bits (64 HEX characters).

Figure 8-56. Configure Security Wizard - User Keys



The following table describes the options in this step.

Table 8-38. Options in the User Keys Step

Option	Description
FlashLock/UPK1	Protects all security settings. This key is required and must be a string of 64 HEX characters. Enter the key or click the padlock icon at the far right to generate a random key. Default: enabled
User Encryption Key 1 (UEK1)	Used for updating the Fabric, uPROM, and sNVM. This key is required and must be a string of 64 HEX characters. Enter the key or click the padlock icon at the far right to generate a random key. To disable it, click Disable. Default: enabled
User Encryption Key 2 (UEK2)	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. Enter the key or click the padlock icon at the far right to generate a random key. To disable it, click Disable. Default: enabled
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.

8.10.1.2. Step 2: Update Policy ([Ask a Question](#))

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 8-57. Configure Security Wizard - Update Policy

Configure Security Wizard

Update Policy

Fabric can be updated using a bitstream encrypted with UEK1.
 sNVM can be updated using a bitstream encrypted with UEK1.
 eNVM can be updated using a bitstream encrypted with UEK1.
 Back Level protection is disabled.
 Auto Programming, Auto Update, IAP Services, and Programming Recovery are enabled for update.
 JTAG interface is enabled for update.
 SPI Slave interface is enabled for update.
 Program action is enabled for JTAG and SPI Slave interfaces.
 Authenticate action is enabled for JTAG and SPI Slave interfaces.
 Verify action is enabled for JTAG and SPI Slave interfaces.

Debug Policy

DPK has not been provided and will not be programmed.
 SmartDebug user debug screen and action modes are enabled.

Security key mode

☐ Bitstream encryption with default key ☒ Custom security options

User Keys

Update Policy

Field updates are enabled by default. You can disable updates by setting options below.
 Use FlashLock/UPK1 to temporarily enable disabled settings.

Fabric/sNVM update protection: Updates allowed using user defined encryption keys; FlashLock/UPK1 is not required for updates

eNVM update protection: Updates allowed using user defined encryption keys; FlashLock/UPK1 is not required for updates

☐ Enable Back Level protection

Design version: 0

Back Level version: 0

☐ Disable Auto Programming and IAP Services

Disable programming interfaces:

☐ JTAG

☐ SPI Slave

Disable bitstream programming actions (JTAG/SPI Slave):

☐ Program

☐ Authenticate

☐ Verify

Reset to default

Help Back Next Finish Cancel

The following table describes the options in this step.

Table 8-39. Options in the Update Policy Step

Option	Description
Fabric/sNVM update protection	Choices are: - Disable Erase/Write operations. Note: The field-update files encrypted with UEK1 and UEK2 include plaintext FlashLock/UPK1. - Updates allowed using user-defined encryption keys: FlashLock/UPK1 is not required for updates.
eNVM update protection	(PolarFire® SoC only) Updates allowed using user-defined encryption keys; FlashLock/UPK1 is not required for updates.

Table 8-39. Options in the Update Policy Step (continued)

Option	Description
Enable Back Level protection	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. - Design version (number between 0–65535): Display the current Design version set in the Configure Programming Options tool. - Back Level version (number between 0–65535): 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. Note: If Back Level Protection is disabled and Back Level version is greater than zero, Generate Bitstream and Export Bitstream tools error out. The examples in the following tables show Back Level protection enabled in the Configure Security tool.
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. Note: If a PolarFire SoC user disables this option manually, the Enable One-way passcode options in the Configure Security Wizard is not available and SPI files cannot be exported.
Disable programming interfaces	You can disable the following programming interfaces: - JTAG - SPI Slave You cannot disable Auto Programming and IAP Services and both the programming interfaces. If you try, an error message appears. If you check JTAG, Libero checks the following options automatically: Disable UTAG command through JTAG interface Disable JTAG boundary scan If you enable back the JTAG programming interface by unchecking JTAG, the options Disable UTAG commands through JTAG interface and Disable JTAG boundary scan remain checked.
Disable bitstream programming actions (JTAG/SPI Slave)	Choices are: - Program - Authenticate - Verify For PolarFire SoC devices, disabling the Program action, disables the programming action for all the interfaces. Note: The field-update files encrypted with UEK1 and UEK2 include plaintext FlashLock/UPK1.
Reset to default	Reset the options to default values. All options are unchecked.

Table 8-40. 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 8-41. 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
7	_uek1_1	PROGRAM	4	1	1	Fail due to back level protection
8	uek1_2	VERIFY	5	4	4	Pass

8.10.1.3. Step 3: Debug Policy ([Ask a Question](#))

Debugging is enabled by default. Use this page to configure Debug Protections.

Figure 8-58. Configure Security Wizard

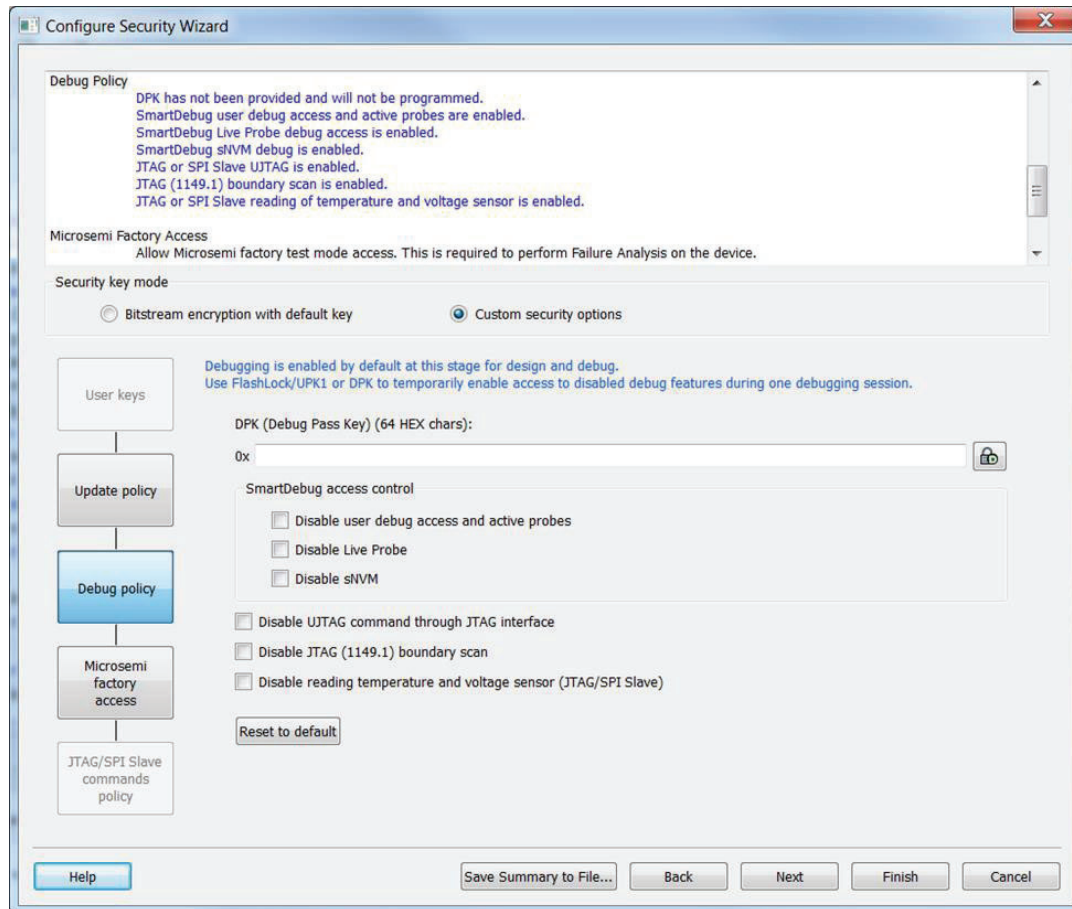


Table 8-42. Elements in Debug Policy Step

Element	Description
Debug with DPK (Debug Pass Key)	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, check at least one option.

Table 8-42. Elements in Debug Policy Step (continued)

Element	Description
SmartDebug Access Control	All the following are enabled by default for SmartDebug access. Check to disable access. • Disable User Debug Access and Active Probe • Disable Live Probe • Disable sNVM  WARNING 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.  WARNING 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. Three additional options are: • Disable UJTAG command through JTAG Interface. • Disable JTAG (1149.1) boundary scan: Imposes the following limitations: – JTAG (1149.1) boundary scan is disabled. – During JTAG device programming, all I/Os are tristated and weakly pulled-up. – BSR control during JTAG programming is disabled. – User modifications to the default I/O States during JTAG programming are not applied. – The following JTAG instructions are disabled: EXTEST, INTEST, CLAMP, SAMPLE, PRELOAD, and HIGHZ. – BYPASS, IDCODE, and USERCODE instructions remain functional. – Boundary scan cannot be unlocked with the FlashLock/UPK1 key or DPK key. To remove these limitations, reprogram the device with updated security settings that re-enable the JTAG (1149.1) boundary scan. • Disable reading temperature and voltage sensor (JTAG/SPI Slave): The summary at the top of the page displays the results of the selection.

8.10.1.4. Step 4: Microsemi Factory Access Policy [\(Ask a Question\)](#)

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 8-59. Configure Security Wizard - Microsemi Access Policy

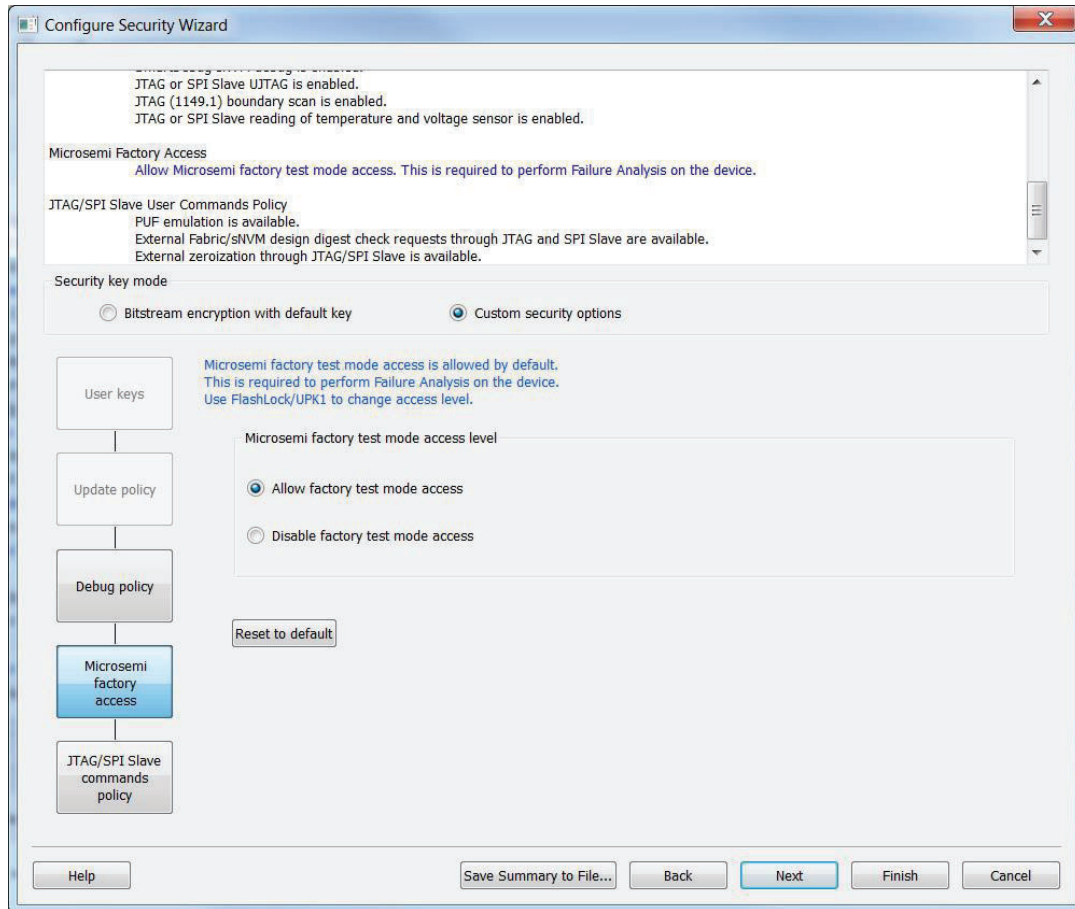


Table 8-43. Option in the Microsemi Access Policy Step

Option	Description
Microsemi factory access level	Choices are: - Allow factory test mode access: Do not use this setting for production devices. (<i>default</i>) - Disable factory test mode access. Note: Use FlashLock/UPK1 to change access level.

8.10.1.5. Step 5: JTAG/SPI Slave Command Policy [\(Ask a Question\)](#)

In step 5 of the Configure Security Wizard, you configure the policy for JTAG/SPI slave user commands.

Figure 8-60. Configure Security Wizard - JTAG/SPI Slave Commands Policy

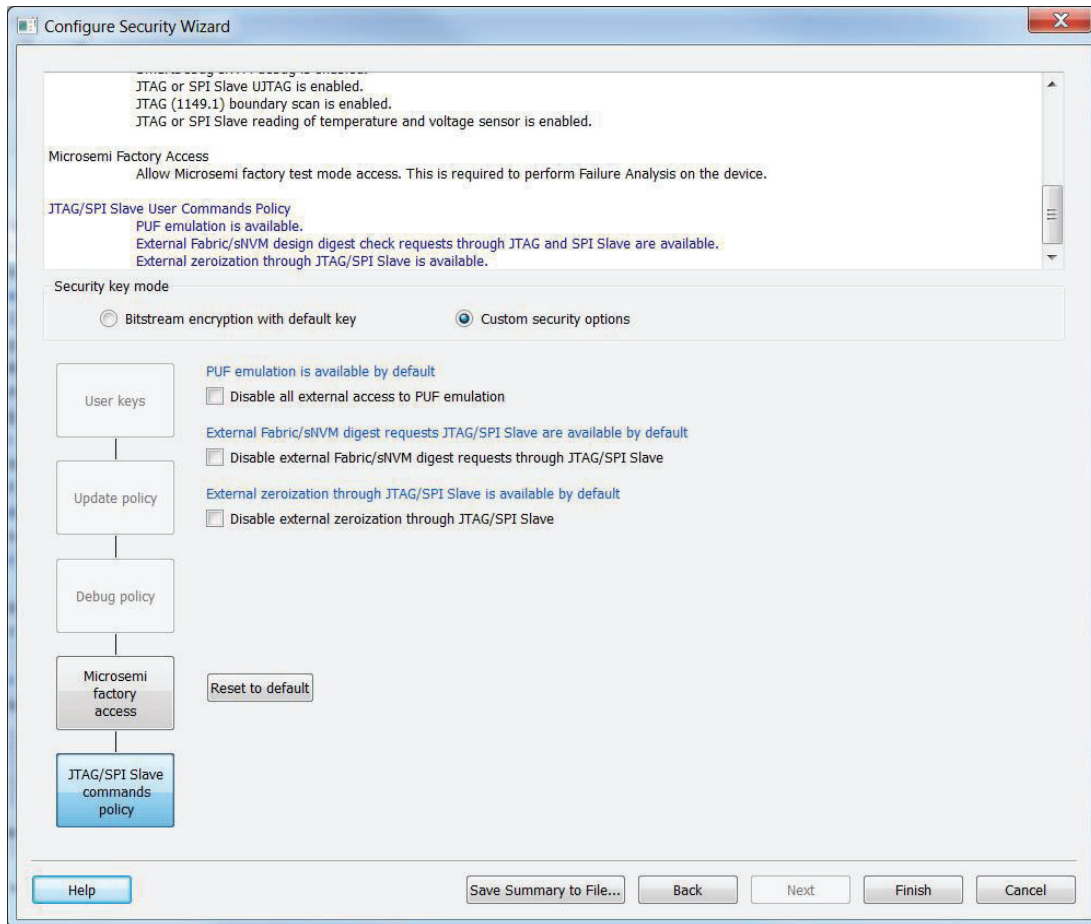


Table 8-44. Options in the JTAG/SPI Slave Command Policy Step

Option	Description
Disable all external access to PUF emulation	Determines whether PUF emulation is available by default. Choices are: - Checked: Disable all external access to PUF emulation through the JTAG/SPI slave. - Not checked: Enable all external access to PUF emulation through the JTAG/SPI slave. (<i>default</i>)
Disable external Fabric/sNVM digest requests through JTAG/SPI Slave	Determines whether external Fabric/sNVM digest requests through the JTAG/SPI slave are available by default. Choices are: - Checked: Disable external Fabric/sNVM digest requests through JTAG/SPI slave. - Not checked: Enable external Fabric/sNVM digest requests through JTAG/SPI slave. (<i>default</i>)  WARNING Repeated external Fabric digest calculations can impact device reliability. For more information, see the product data sheet.

Table 8-44. Options in the JTAG/SPI Slave Command Policy Step (continued)

Option	Description
Disable external zeroization through JTAG/SPI Slave	Determines whether external zeroization through the JTAG/SPI slave is available by default. Choices are: - Checked: Disable external zeroization through JTAG/SPI slave. - Not checked: Enable external zeroization through JTAG/SPI slave. (<i>default</i>)  WARNING Do not enable zeroization for production devices.

8.10.2. Configure Security Policy Manager (SmartFusion 2 and IGLOO 2) [\(Ask a Question\)](#)

In the Design Flow window, double-click **Configure Security** to open the Security Policy Manager dialog box and customize the security settings in your design.

Use this dialog box to set your User Keys, Security Policies, and Microchip factory test mode access level.

Note: Microchip-enabled default bitstream encryption key modes are disabled after user security is programmed.

Figure 8-61. Security Policy Manager Dialog Box

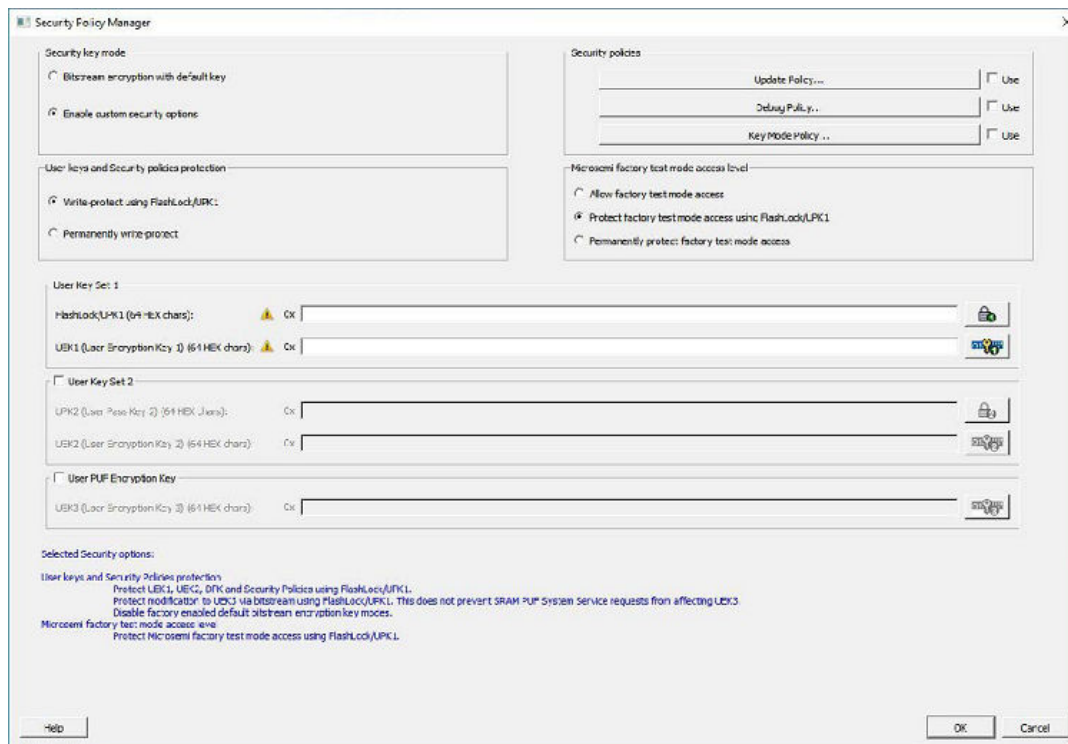
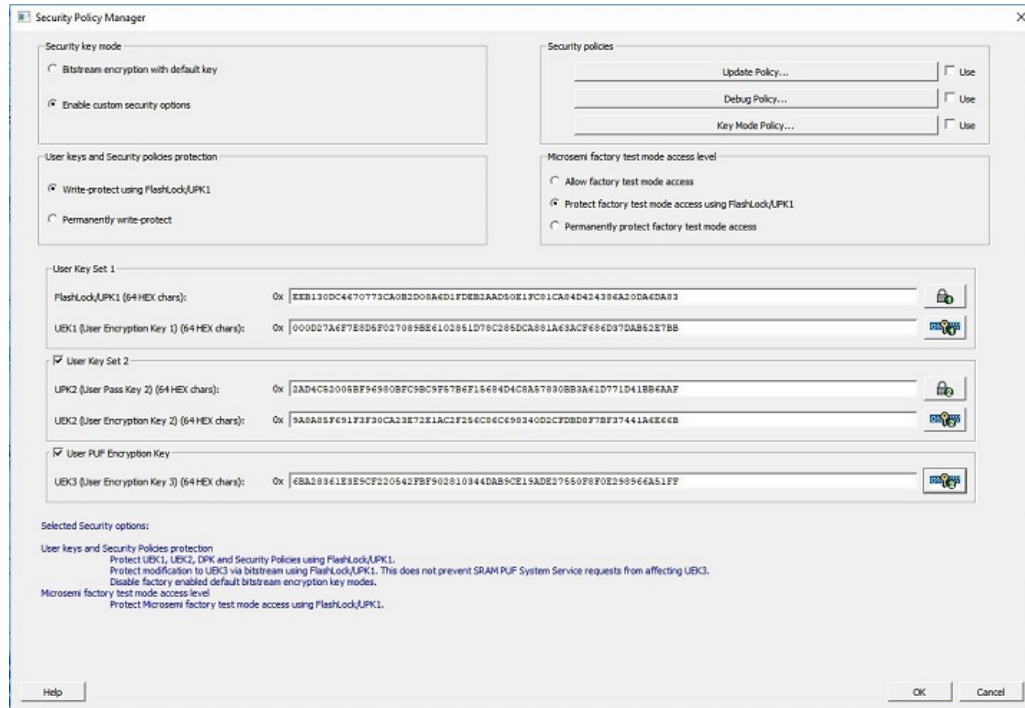


Figure 8-62. Security Policy Manager Dialog Box (for devices supporting UEK3)



Security Key Mode

- **Bitstream encryption with default key-** Encrypt bitstream files with Microchip default key (pre-placed key in silicon). When this option is selected, user keys, security, and Microchip factory test mode access level configurations are disabled.
- **Enable custom security options-** Enables you to set User Keys, Security Policies and Microchip factory test mode access level (see the following description).

User keys and Security policies protection

- **Write-protect using FlashLock/UPK1** - Protect UEK1 (User Encryption Key 1), UEK2 (User Encryption Key 2), DPK (Debug Pass Key), and Security Policies using FlashLock/ UPK1. Protect modification to UEK3 via bitstream using FlashLock/UPK1. SRAM-PUF System services can still modify UEK3 after programming Security settings.
Note: UEK2 (User Encryption Key2) is protected by UPK2 (User Pass Key 2). UEK3 is only available for M2S060, M2GL060, M2S090, M2GL090, M2S150, and M2GL150 devices.
- **Permanently write-protect** - Permanently protect UEK1 (User Encryption Key 1), UPK2 (User Pass Key 2), UEK2 (User Encryption Key 2), DPK (Debug Pass Key), Security Policies, and Microchip factory test mode access level. Permanently protect modification to UEK3 via bitstream. Note that even after programming Security settings, SRAM-PUF System services can still modify UEK3. This setting, once programmed will not be modified in the device. Microchip enabled default bitstream encryption key modes are permanently disabled as well.
Note: When this option is selected, you cannot specify the FlashLock/UPK 1 and UPK2 (User Pass Key 2) value because the value cannot be used to unlock the corresponding protected features. UEK3 is only available for M2S060, M2GL060, M2S090, M2GL090, M2S150, and M2GL150 devices.

Microchip Factory Test Mode Access Level

- **Allow factory test mode access** - Allows access to Microchip factory test mode.
- **Protect factory test mode access using FlashLock/UPK1** - Protects access to Microchip factory test mode using Flashlock/ UPK1.

- **Permanently protect factory test mode access** - Permanently locks access to Microchip factory test mode.

Note: When this option is selected, User Key Set 2 is permanently write-protected. Once programmed, User Key Set 2 cannot be changed in the device. You can specify UEK2 (User Encryption Key 2). However, you cannot specify UPK2 (User Pass Key 2) because the value cannot be used to unlock User Key Set 2.

Security Policies

- **Update Policy** - Sets your Fabric, eNVM and Back Level protections. It also allows you to disable access to certain programming interfaces. See the [Update Policy](#) topic for more information.

Note: If Update Policy is enabled and Fabric update is protected by UPK1:

Fabric update is disabled for Auto Programming, IAP/ISP services, Programming Recovery and Auto update. FlashLock/UPK1 unlocking is only available for JTAG and SPI slave programming. See the following example.

Figure 8-63. Update Policy Dialog Box Denoting Fabric Update Protection by Flashlock/UPK1

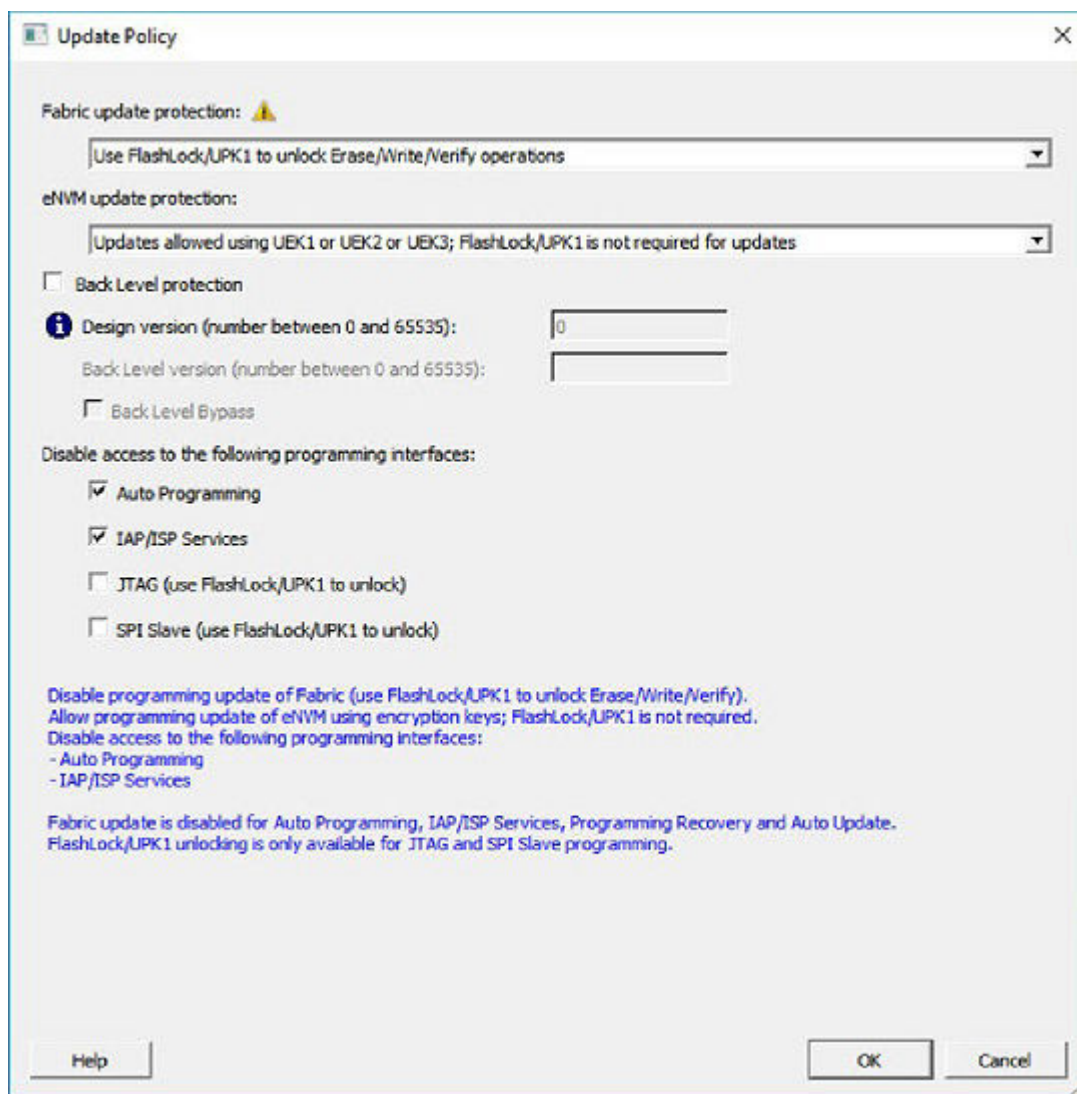
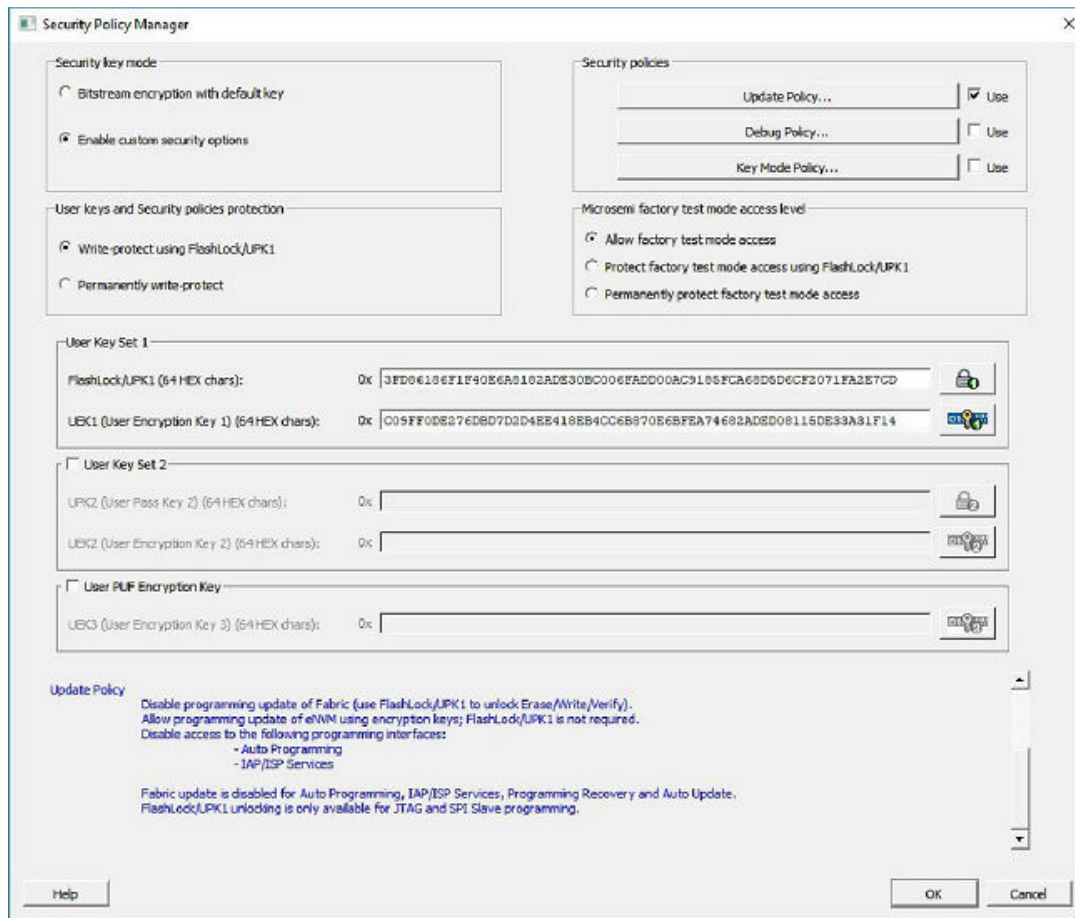


Figure 8-64. Security Policy Manager with Update Policy Description for Fabric Update Protection by Flashlock/UPK1



- **Debug Policy** - Enables and sets your Debug Pass Key and debug options. See the [Debug Policy topic](#) for more information.
- **Key Mode Policy** - Configures the key mode to enable or disable. See the [Key Mode Policy topic](#) for more information.

Configuring User Keys

- **User Key Set 1** is required. User Key Set 1 includes FlashLock/UPK1 (User Pass Key 1) and UEK1 (User Encryption Key 1).
- **User Key Set 2** is optional. User Key Set 2 includes UPK2 (User Pass Key 2) and UEK2 (User Encryption Key 2). Note that User Pass Key 2 (UPK2) protects only User Encryption Key 2 (UEK2).
- **User PUF Encryption Key** is optional. User PUF Encryption Key includes UEK3 (User Encryption Key 3).

Note: UEK3 is only available for M2S060, M2GL060, M2S090, M2GL090, M2S150, and M2GL150 devices.

8.10.2.1. Update Policy [\(Ask a Question\)](#)

This dialog box enables you to specify components that can be updated in the field, and their field-update protection parameters.

Choose your protection options from the drop-down menus; click the appropriate check box to set your programming protection preferences.

Fabric Update Protection

- **Use FlashLock/UPK1 to unlock Erase/Write/Verify operations-** Select this option to require UPK1 to erase, write, or verify the Fabric.

Note: Fabric update is disabled for Auto Programming, IAP/ISP services, Programming Recovery, and Auto update. FlashLock/UPK1 unlocking is only available for JTAG and SPI slave.

- **Updates allowed using UEK1 or UEK2 or UEK3; FlashLock/UPK1 is not required for updates -** Encrypted update is allowed with either UEK1 or UEK2 (if enabled).

Note: UEK3 is only available for M2S060, M2GL060, M2S090, M2GL090, M2S150, and M2GL150 devices.

eNVM Update Protection

- **Use FlashLock/UPK1 to unlock Write/Verify/Read operations-** Select this option to require UPK1 to write, verify or read to the eNVM.

Note: eNVM update is disabled for Auto Programming, IAP/ISP Services, Programming Recovery, and Auto Update. FlashLock/UPK1 unlocking is only available for JTAG and SPI Slave programming.

- **Updates allowed using UEK1 or UEK2 or UEK3; Flashlock/UPK1 is not required for updates -** Encrypted update is allowed with either UEK1 or UEK2 (if enabled) or UEK3 (if enabled).

Note: UEK3 is only available for M2S060, M2GL060, M2S090, M2GL090, M2S150, and M2GL150 devices.

Back Level protection - When enabled, a design being loaded must be of a version higher than the Back Level version value in the programmed device.

- **Back Level Protection-** Limits the design versions that the device can update. Only programming bitstreams with Designer Version greater than the Back Level version are allowed for programming.
- **Design version** - Displays the current Design version (set in the [Configuring Programming Options](#)). Back level uses the Design version value to determine which bitstreams are allowed for programming.
- **Back Level Bypass** - If selected, design is programmed irrespective of Back Level version.

Note: Back Level Bypass should be set if you allow programming recover with recovery image lower than the Back Level version selected. Alternatively, you should update the design version of the recovery image so that it is always greater than the Back Level version.

Disable Access to the Following Programming Interfaces

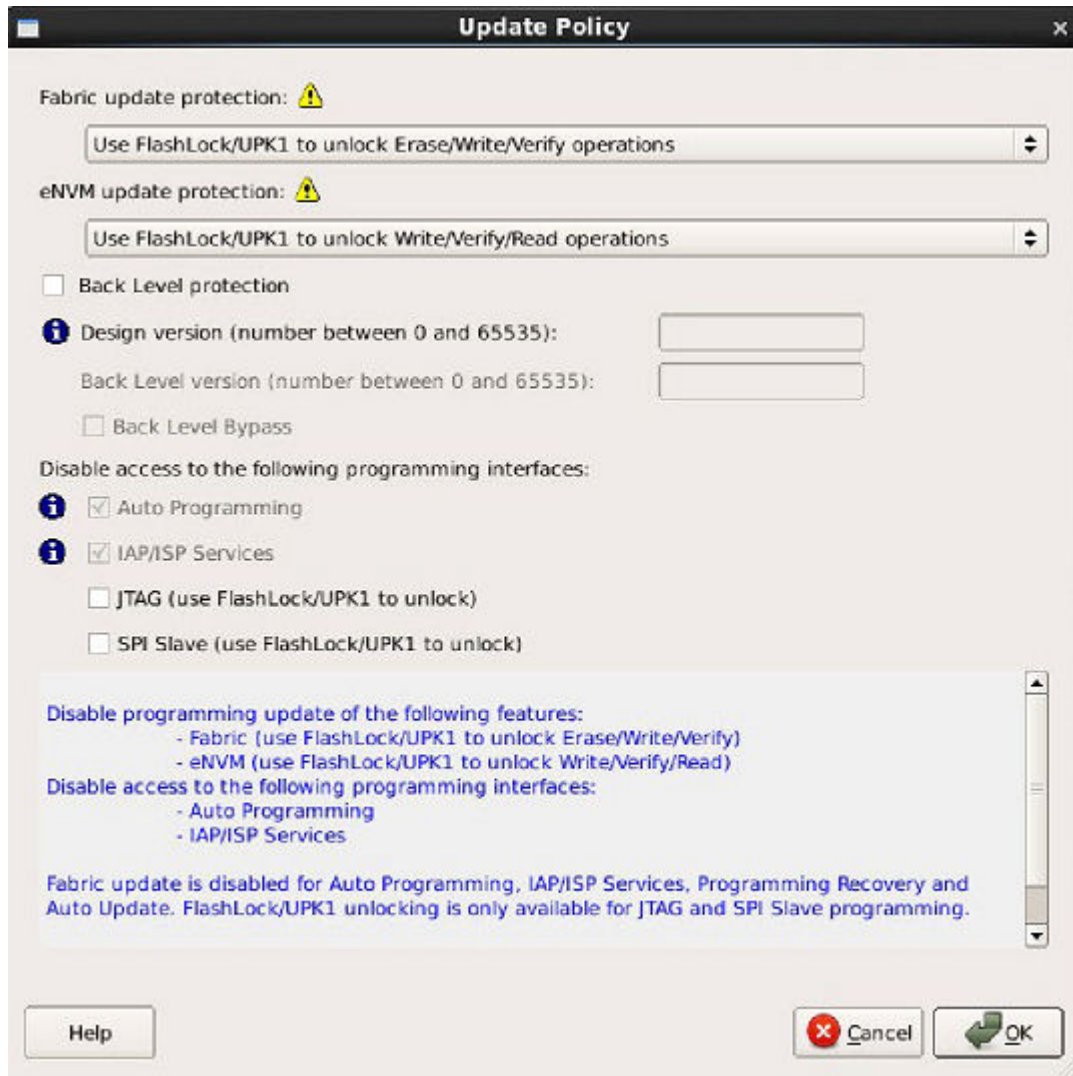
These settings protect the following programming interfaces:

- Auto Programming
- IAP/ISP services
- JTAG (use FlashLock to/UPK1 to unlock)
- SPI Slave (use FlashLock/UPK1 to unlock)

For more technical information on the Protect Programming Interface with Pass Key option see the [SmartFusion 2 Programming User's Guide](#)

Note: When the **Permanently write-protect** option is selected for User keys and Security policies protection in SPM, the dialog box informs you of features that are no longer reprogrammable. In this case, if Use FlashLock/UPK1 to unlock option is selected for Fabric/eNVM update protection, Fabric/eNVM will be set to **One-Time Programmable (OTP)**.

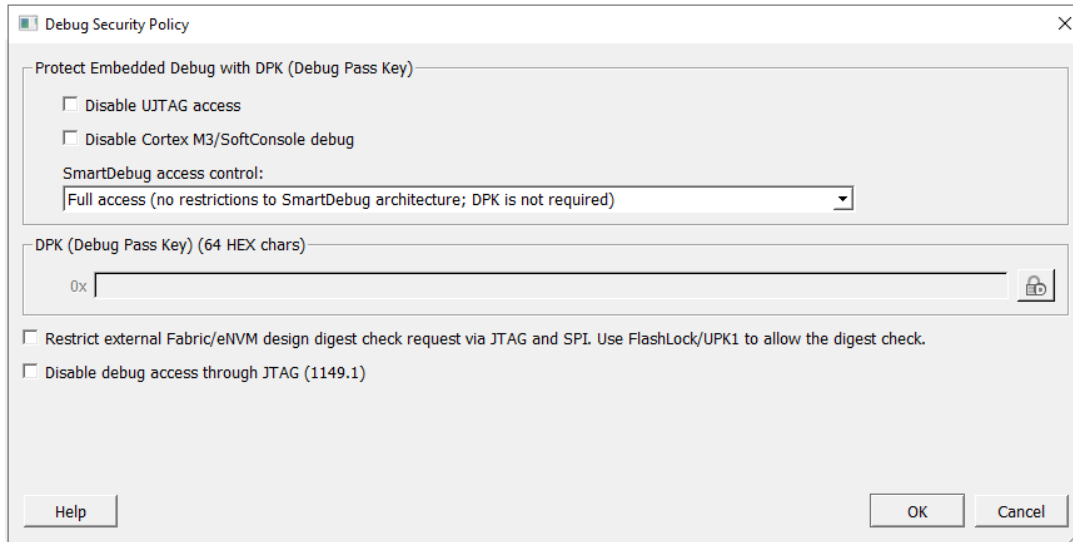
Figure 8-65. Update Policy Dialog Box



8.10.2.2. Debug Security Policy [\(Ask a Question\)](#)

Debug access to the embedded systems can be controlled via the customer Debug Policy.

Figure 8-66. Debug Security Policy Dialog Box



Protect Embedded Debug with DPK (Debug Pass Key)

Restrict UJTAG access - Restricts access to UJTAG; DPK is required for access.

Restrict Cortex M3 debug (SmartFusion 2 Only) - Restricts Cortex M3 debug/SoftConsole use; DPK is required for debug.

SmartDebug Access Control

Access control available during debug mode.

Full Access (No restrictions to SmartDebug architecture; DPK is not required)- Enables full debug access to eNVM, uSRAM, LSRAM, eSRAM0/1, DDRAM and Fabric probing.

No debug (Restrict read/write access to SmartDebug architecture; DPK is required for read/write access) - Blocks all debug access to eNVM, uSRAM, LSRAM, eSRAM0/1, DDRAM and Fabric probing.

DPK (Debug Pass Key) (length is 64 HEX characters)

Specify a Debug Pass Key to unlock features protected by DPK.

Restrict external Fabric/eNVM design digest check request via JTAG and SPI. Use FlashLock/UPK1 to allow digest check. - Protects design digest check request with FlashLock/UPK1.

Disable debug access through JTAG (1149.1). - Disables JTAG (1149.1) test instructions. The following JTAG test instructions 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.

8.10.2.3. Key Mode Policy [\(Ask a Question\)](#)

Protect user encryption key modes with FlashLock/UPK1. If a key mode is disabled, then FlashLock/UPK1 is required to program with that key mode.

The following key modes can be disabled:

- UEK1 (User Encryption Key 1)
- UEK2 (User Encryption Key 2)
- UEK3 (User Encryption Key 3)

Note: UEK3 is only available for M2S060, M2GL060, M2S090, M2GL090, M2S150, and M2GL150 devices.

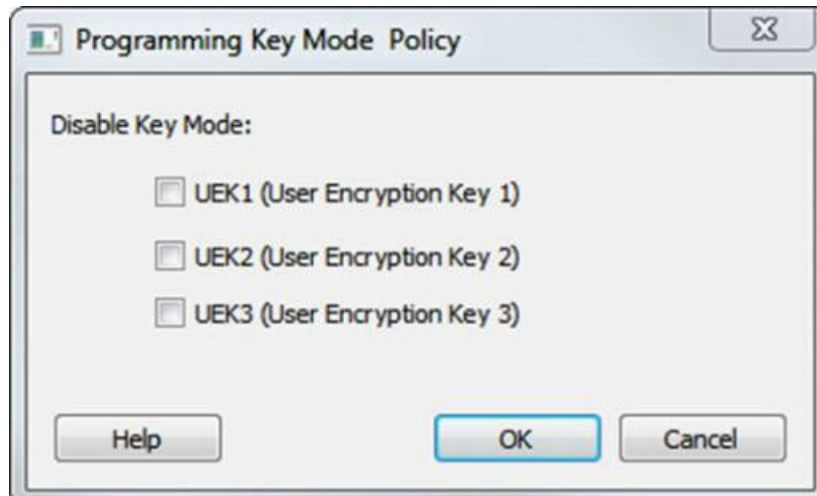
If all key modes are disabled, device update is impossible and a warning message is displayed.

Note: If a key mode is disabled, the corresponding bitstream file will be disabled.

Figure 8-67. Programming Key Mode Policy Dialog Box



Figure 8-68. Programming Key Mode Policy Dialog Box (for devices supporting UEK3)



8.10.2.4. Security Programming Files [\(Ask a Question\)](#)

[Exporting Bitstreams](#)(expand Handoff Design for Production in the Design Flow window) creates the following files:

<filename>_master.(stp/svf/spi/dat) - Created when Enable custom security options is specified in the [Configuring Security](#). 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 /svf/spi/dat) - Created when Enable custom security options is specified in the [Configuring Security](#). Master security programming file; includes User Key Set 1, User Key Set 2 (optionally if specified), and your security policy settings.

<filename>_uek1.(stp/svf/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/svf/spi/dat) – Programming file encrypted with User Encryption Key 2 used for field updates; includes all your features for programming except security.

<filename>_uek3.(stp/svf/spi/dat) – Programming file encrypted with User Encryption Key 3 used for field updates; includes all your features for programming except security.

Note: UEK3 is available only for M2S060S, M2GL060S, M2S090S, M2GL090S, M2S150S, and M2GL150S devices.

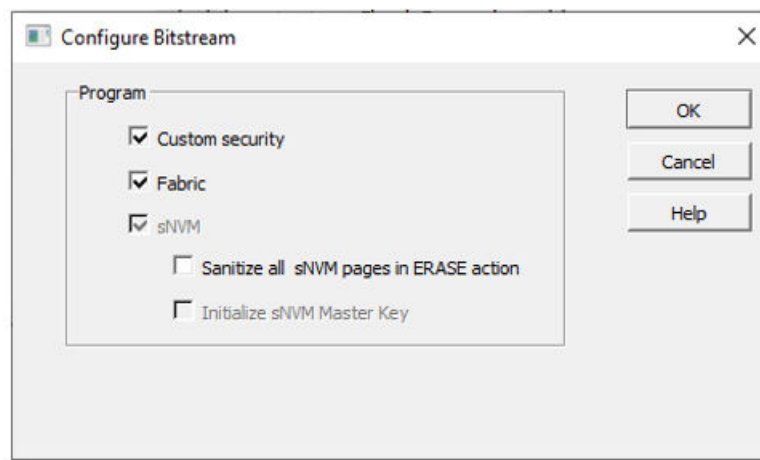
8.11. Configuring Bitstreams [\(Ask a Question\)](#)

8.11.1. Configure Bitstream (PolarFire and PolarFire SoC) [\(Ask a Question\)](#)

The Configure Bitstream dialog box allows you to select which components you wish to program. Only features that are added to your design are available for programming. To display the dialog box, right click **Generate Bitstream** in the Design Flow window and choose **Configure Options**.

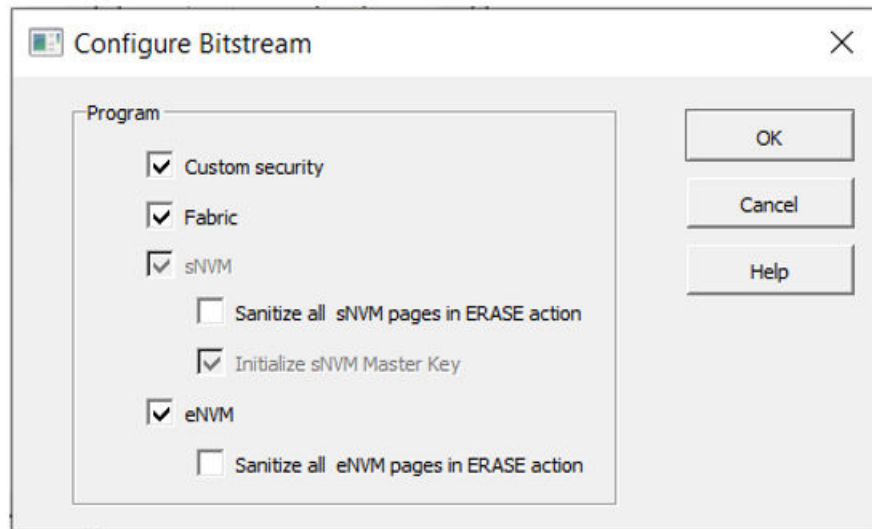
In the dialog box, the option **Sanitize all sNVM pages in ERASE action** is supported for PolarFire and PolarFire SoC. The option is enabled in the dialog box if the **Fabric/sNVM** option is selected. If the design includes uPROM, it will be included in the Fabric.

Figure 8-69. Configure Bitstream (PolarFire)



PolarFire SoC devices also provide an **eNVM** option if eNVM clients are present and being programmed.

Figure 8-70. Configure Bitstream with eNVM Option (PolarFire SoC)



Observe the following guidelines:

Notes:

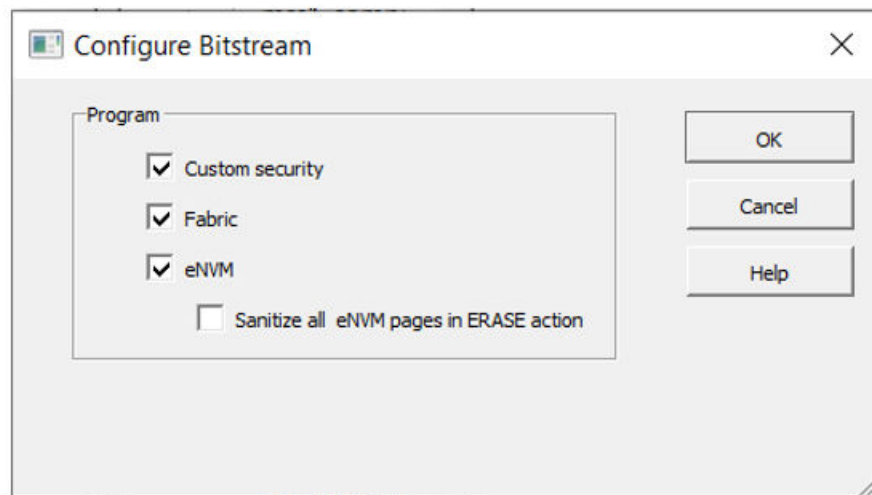
- Custom security is enabled if security was configured.
- All available features are selected by default.
- sNVM is always programmed with Fabric for PolarFire devices. PolarFire SoC allows you to program either Fabric with sNVM or sNVM only.

8.11.2. Configure Bitstream (RTG4, SmartFusion 2, and IGLOO 2) [\(Ask a Question\)](#)

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 are added to your design are available for programming. For example, you cannot select eNVM for programming if you do not have eNVM in your design.

Figure 8-71. Configure Bitstream Dialog Box (SmartFusion 2 and IGLOO 2)



Note: The Custom security and eNVM components are not available for RTG4 devices.

Sanitize all eNVM pages in ERASE action - eNVM option will be available only if eNVM clients are present and are being programmed.

8.12. Generating the Bitstream [\(Ask a Question\)](#)

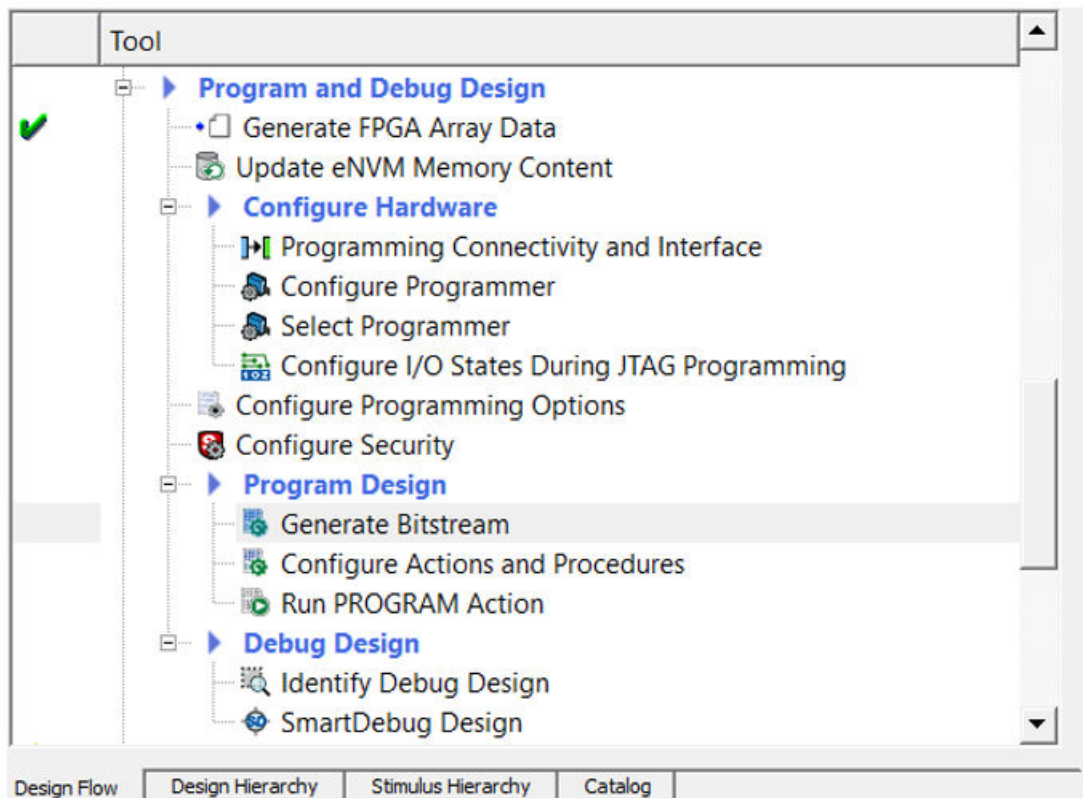
The Generate Bitstream option generates the bitstream for use with the Run PROGRAM Action tool.

The tool incorporates the Fabric design, sNVM configuration, eNVM configuration (if configured) and custom security settings (if configured) to generate the bitstream file. Before you generate the bitstream, configure the bitstream. Otherwise, default settings with all available features included are used. To display the Configure Bitstream dialog box, expand **Program Design**, right-click **Generate Bitstream** and choose **Configure Options** to select the components you want to program. Only features that are added to your design are available for programming. When the process is complete, a green check mark appears next to the operation in the Design Flow window (as shown in the following figure) and information messages appear in the Log window.

Observe the following guidelines:

- If the design includes uPROM, it will be included in the Fabric.
- The **eNVM** option will be available for PolarFire SoC, SmartFusion 2, and IGLOO 2 devices only if eNVM clients are present and being programmed.
- 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 be regenerated only 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.

Figure 8-72. Generate Bitstream (Complete)



8.13. Configuring Actions and Procedures [\(Ask a Question\)](#)

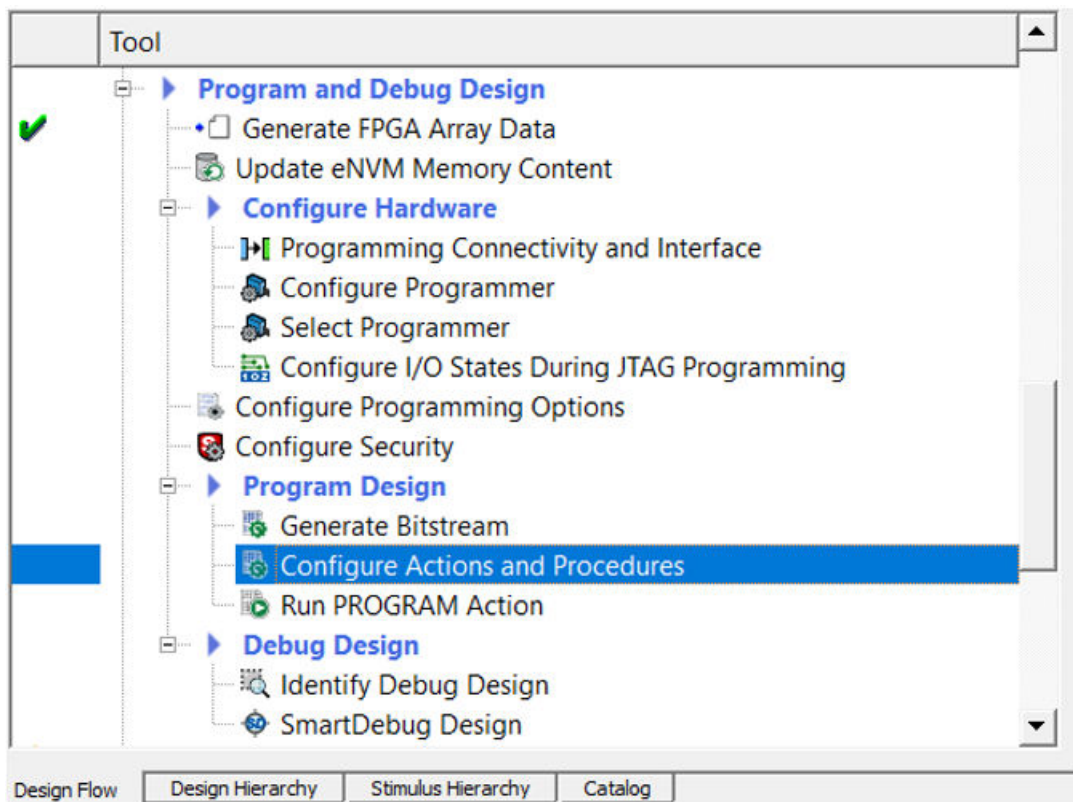
The Configure Actions and Procedures tool allows you to configure actions with optional or recommended procedures for a Libero target device. The information is saved and can be used by the Run Action tool.

Observe the following guidelines when using this tool:

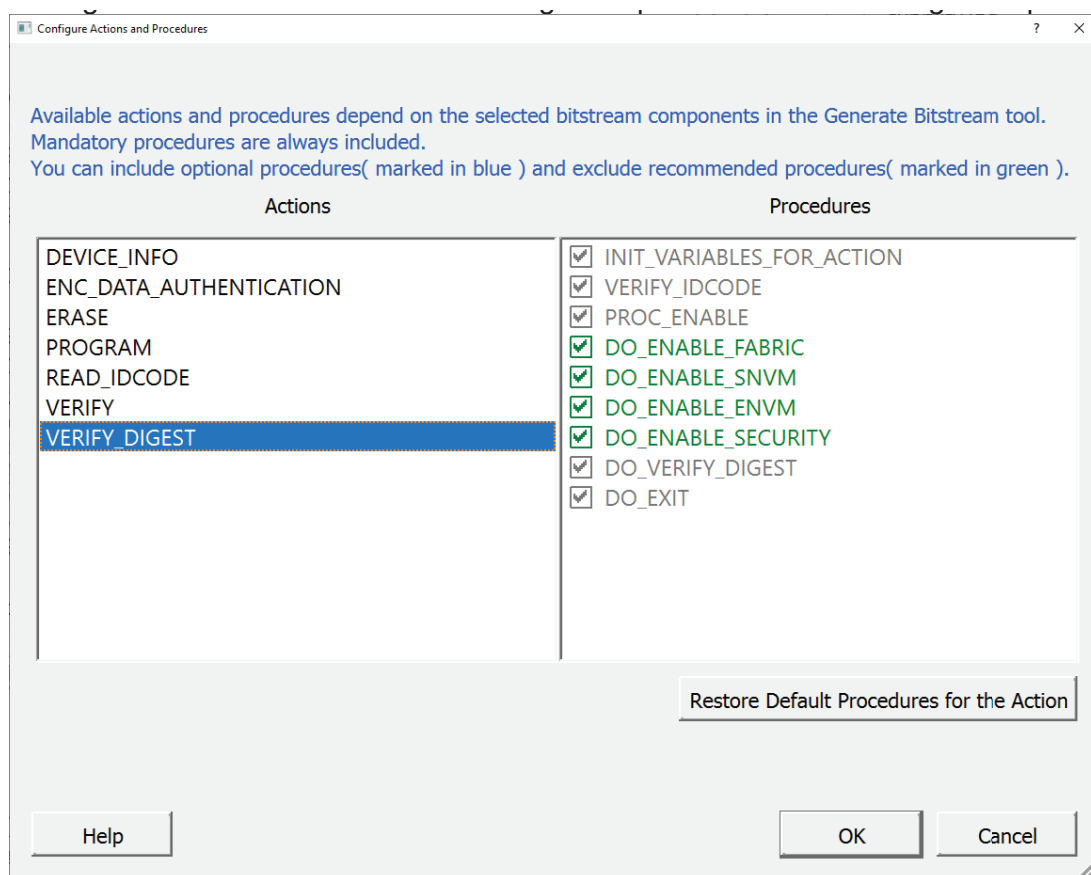
- Available actions and their procedures depend on current bitstream components selected in the Generate Bitstream and Configure Options tools.
- Changing procedures for the action selected to run invalidates the Run Action tool state. Changing any other action does not affect the Run Action tool state.

To run the Configure Actions and Procedures tool, from the Libero Design Flow window, expand **Program Design** and double-click **Configure Actions and Procedures**.

Figure 8-73. 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 8-74. Configure Actions and Procedures Dialog Box

8.13.1. Programming File Actions and Supported Procedures [\(Ask a Question\)](#)

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 8-45. Programming File Actions and Supported Procedures

Action	Procedures
DEVICE_INFO	• INIT_VARIABLES_FOR_ACTION • SET_DEVICE_INFO_ACTIONTYPE • VERIFY_IDCODE • DO_DEVICE_INFO • DO_EXIT
ENC_DATA_AUTHENTICATION	• INIT_VARIABLES_FOR_ACTION • SET_AUTHORIZATION_ACTIONTYPE • VERIFY_IDCODE • DO_AUTHENTICATION • DO_EXIT

Table 8-45. Programming File Actions and Supported Procedures (continued)

Action	Procedures
ERASE	- INIT_VARIABLES_FOR_ACTION - SET_ERASE_ACTIONTYPE - VERIFY_IDCODE - PROC_ENABLE - DO_ERASE DO_EXIT
PROGRAM	- INIT_VARIABLES_FOR_ACTION - SET_PROGRAM_ACTIONTYPE - VERIFY_IDCODE - PROC_ENABLE - DO_PROGRAM - DO_VERIFY (optional) - DO_EXIT
READ_IDCODE	- INIT_VARIABLES_FOR_ACTION - SET_READ_IDCODE - VERIFY_IDCODE - PRINT_IDCODE - DO_EXIT
VERIFY	- INIT_VARIABLES_FOR_ACTION - SET_VERIFY_ACTIONTYPE - VERIFY_IDCODE - PROC_ENABLE - DO_VERIFY - DO_EXIT
VERIFY_DIGEST	- INIT_VARIABLES_FOR_ACTION - VERIFY_IDCODE PROC_ENABLE - DO_ENABLE_FABRIC (recommended) - DO_ENABLE_SNVM (recommended) - DO_ENABLE_ENVM(recommended) - DO_ENABLE_SECURITY (recommended) - DO_VERIFY_DIGEST (recommended) - DO_EXIT VERIFY_DIGEST also exports (that is, prints in the programming log) the newly calculated digests and compares them with the stored digests.
ZEROIZE_LIKE_NEW	- INIT_VARIABLES_FOR_ACTION - VERIFY_IDCODE - DO_ZEROIZE_LIKE_NEW DO_EXIT
ZEROIZE_UNRECOVERABLE	- INIT_VARIABLES_FOR_ACTION - VERIFY_IDCODE - DO_ZEROIZE_UNRECOVERABLE - DO_EXIT

8.13.2. Programming File Actions and Descriptions [\(Ask a Question\)](#)

The following table lists programming file actions and descriptions.

Table 8-46. Programming File Actions

Action	Description
PROGRAM	Programs all selected bitstream components: • FPGA Array • Targeted sNVM clients • Targeted eNVM clients • Security settings For more information, see FPGA Programmed Content Integrity Checking and Programming File Tracking using Cryptographic Digests .
ERASE	Erases the selected family features for: • FPGA Array • Security settings
VERIFY_DIGEST	Calculates the digests for the components (Custom Security, Fabric, or sNVM) included in the bitstream from the device and compares them against the programmed/stored digest values in the fabric security segment of the device. VERIFY_DIGEST also exports (that is, prints in the programming log) the newly calculated digests and compares them with the stored digests.
VERIFY	Verifies all selected family features for: • FPGA Array • Targeted sNVM clients • Targeted eNVM clients • Security settings
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.

8.13.3. Options for Specific Programming Actions [\(Ask a Question\)](#)

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 8-47. 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.

8.14. Running Programming Device Actions [\(Ask a Question\)](#)

If you have a device programmer connected, double click **Run PROGRAM Action** to execute your program in batch mode with default settings.

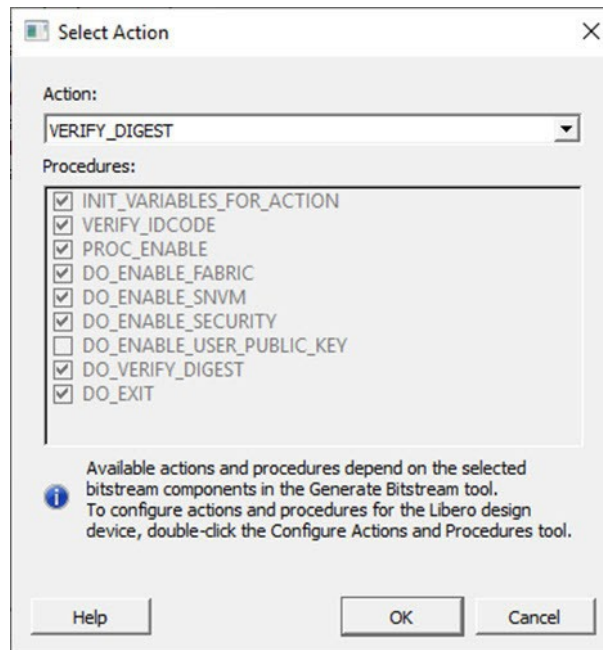


Do not interrupt programming action when it runs as shown in the pop-up. Otherwise, it might damage the part.

If your programmer is not connected or if your default settings are invalid, then the Reports view lists the errors. To select a programming action to run:

1. Right click **Run PROGRAM Action** and choose **Select Action**. The **Select Action** dialog box appears.

Figure 8-75. 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.

8.14.1. Programming File Actions [\(Ask a Question\)](#)

Libero SoC enables you to program security settings, FPGA Array, sNVM, and eNVM features.

Notes:

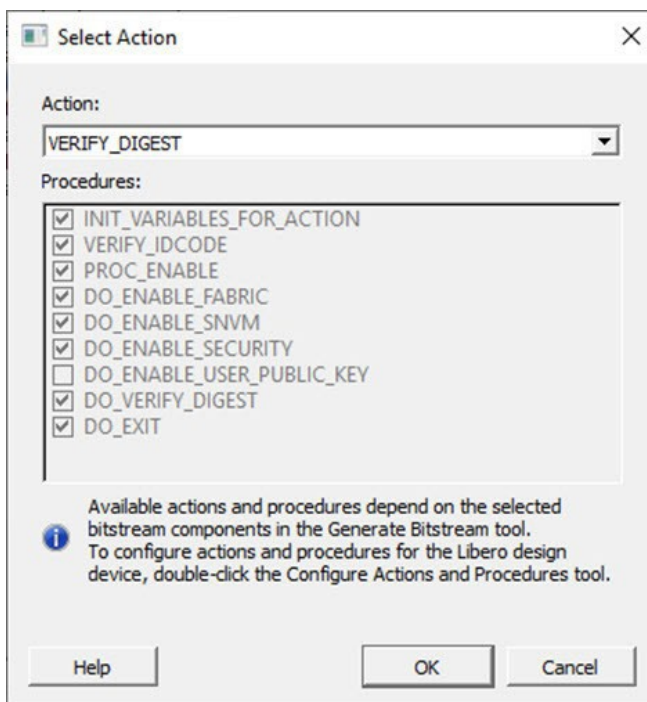
- eNVM features are available for PolarFire SoC, SmartFusion 2, and IGLOO 2 devices only.
- If the design includes μ PROM, it will be included in the Fabric.

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 8-76. Select Action Dialog Box



For details about configuring actions and procedures, see [Configuring Actions and Procedures](#).

Table 8-48. 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 data sheet for more information on transient specifications. Monitor JTAG supply pins during programming; measure JTAG signals for noise or reflection.
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 data sheet for more information on transient specifications.
0x8003	Failed to enter programming mode	5	Unstable voltage level Signal integrity issues on JTAG pins DEVIRST_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 data sheet for more information on transient specifications. Monitor JTAG supply pins during programming; measure JTAG signals for noise or reflection. Tie DEVIRST_N to HIGH prior to programming the device.

Table 8-48. Exit Codes (PolarFire) (continued)

Error Code	Exit Message	Exit Code	Possible Cause	Possible Solution
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 that 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 data sheet 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 are corrupted or tampered with	If the Fabric is erased, deselect procedure DO_ENABLE_FABRIC from action VERIFY_DIGEST
0x8016	sNVM digest verification: FAIL Deselect procedure DO_ENABLE_SNVM to remove this digest check.	-20	sNVM is either erased or the data are corrupted or tampered with	If the sNVM is erased, deselect procedure DO_ENABLE_SNVM 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 are 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 are 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 are 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 is erased through zeroization or the data has been corrupted or tampered with	—

Table 8-48. Exit Codes (PolarFire) (continued)

Error Code	Exit Message	Exit Code	Possible Cause	Possible Solution
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
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 are 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 data sheet for more information on transient specifications. Monitor JTAG supply pins during programming; measure JTAG signals for noise or reflection.
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 is programmed with the user encryption key	Provide a programming file with an encryption key with the one matches that on the device. First program security with master programming file, then program with user encryption 1/2 field update programming files.

Table 8-48. Exit Codes (PolarFire) (continued)

Error Code	Exit Message	Exit Code	Possible Cause	Possible Solution
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 must be driven HIGH or disable System controller suspend mode .
0x8027	Programming Error. Insufficient device capabilities	-26	Device does not support the capabilities specified in the 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, regenerate.	-28	Programming file version is out of date	Generate programming file with the latest version of Libero SoC.
0x8030	Programming Error Invalid or inaccessible Device Certificate	-31	FAB_RESET_N is tied to ground	FAB_RESET_N must 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 data sheet 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.
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 is 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 data sheet 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 are corrupted or tampered with.	If the eNVM is erased, deselect procedure DO_ENABLE_ENVM from action VERIFY_DIGEST .

8.15. Programming SPI Flash Image (PolarFire and PolarFire SoC) [\(Ask a Question\)](#)

The following topics describe how to program a SPI Flash image on PolarFire and PolarFire SoC devices.

8.15.1. Generating a SPI Flash Image [\(Ask a Question\)](#)

The Generate SPI Flash Image tool generates a <design>_spi_flash.bin file in the implementation folder. The tool depends on the Configure Design Initialization Data and Memories tool and the Generate Design Initialization Data tool. While 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).

To run this tool, expand **Program SPI Flash Image**, right-click **Generate SPI Flash Image**, and select **Run**.

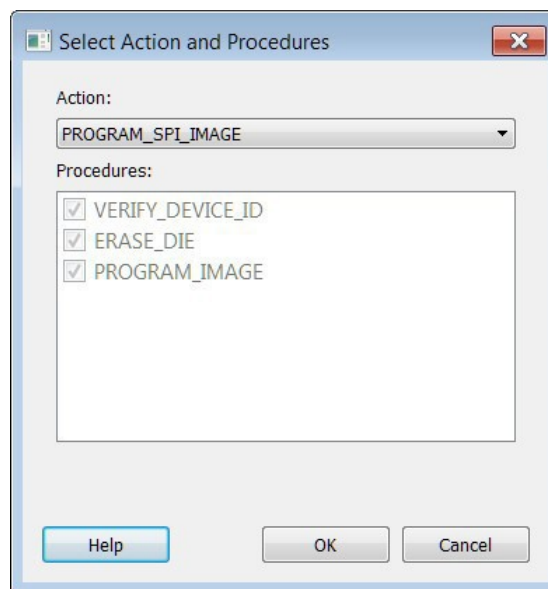
Figure 8-77. Selecting the Run Command



8.15.2. Configure SPI Flash Image Actions and Procedures [\(Ask a Question\)](#)

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 8-78. Select Action and Procedures Dialog Box



The following table lists the actions and procedures for the Run PROGRAM_SPI_Flash tool.

Table 8-49. 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 erases 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: Partial programming of the SPI flash is supported when using PolarFire®/PolarFire SoC FPGAs and the FlashPro6 programmer. In this case, the ERASE_SPI_FLASH action allows you to erase only selected clients.

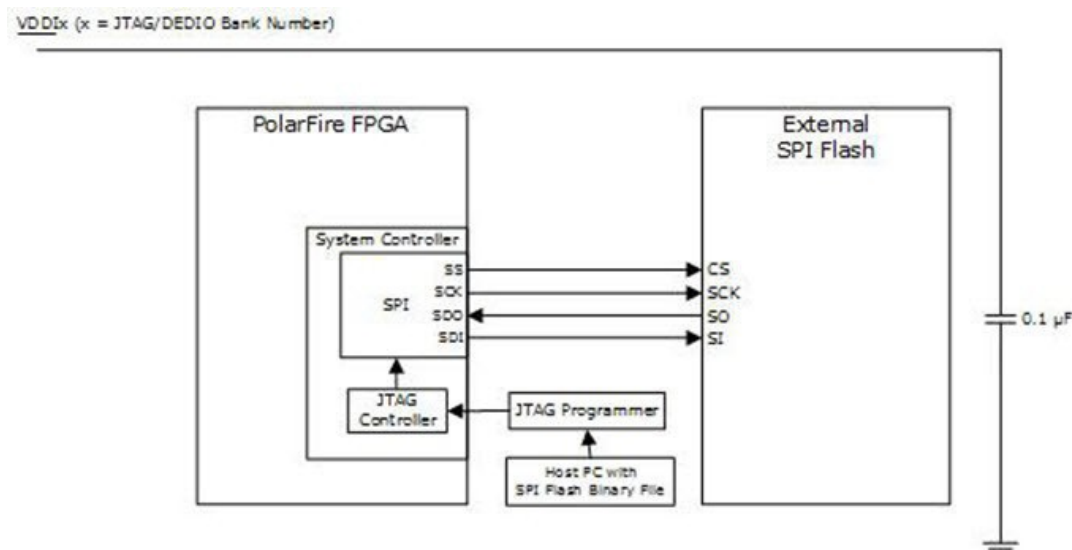
Note: If the device ID does not match while running any action, the action will fail.

8.15.3. Running Programming SPI Flash Actions [\(Ask a Question\)](#)

The Run Programming SPI Flash Actions tool allows you to program the SPI Flash device connected to the PolarFire device through the JTAG programming interface. Only the Micron 1Gb SPI flash furnished with the Evaluation Kit is supported. 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.

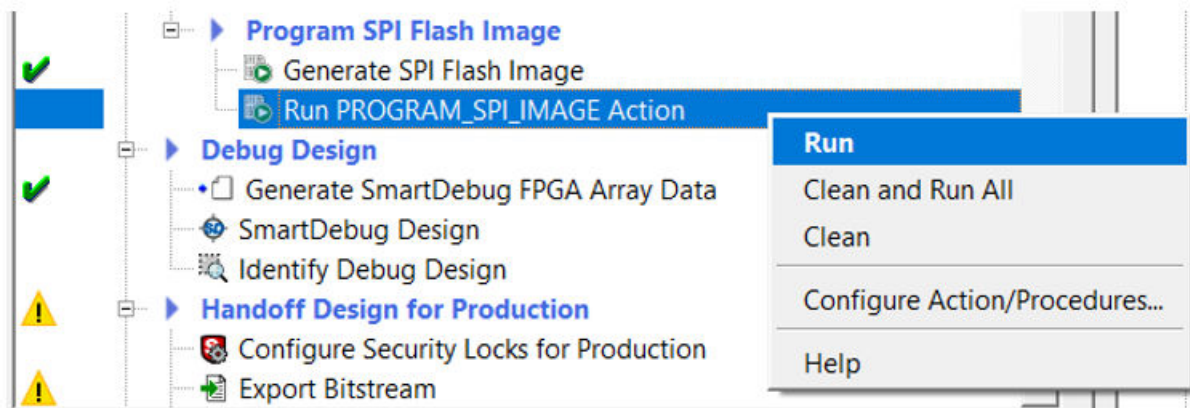
Note: This version of the programmer does not support SPI Flash security. Disable device security options such as **Hardware Write Protect** for the External SPI Flash device.

If SPI Flash is configured, you can execute the Run PROGRAM_SPI_IMAGE Action and select SPI Flash Image actions and procedures. In the Design Flow window, expand **Program SPI Flash Image**, right-click **Run PROGRAM_SPI_Image Action**, and select **Configure Action/Procedures**.



Note: The SPI pins are controlled by the Boundary Scan Register one bit at a time.

Figure 8-79. SPI Flash Programming with PolarFire Device



The following table provides the expectations of programming the SPI flash with an FlashPro5 programmer. Future programmers are planned and must greatly improve programming times. Timing is indicated in hh:mm:ss.

Table 8-50. Expectations of Programming the SPI Flash with an FlashPro5 Programmer

SPI Size	ERASE	PROGRAM	VERIFY/READ	TCK	Programmer
1 MB	3:55	00:00:45	00:10:46	4 MHz	FP5
1 MB	3:55	00:00:28	00:10:05	15 MHz	FP5
9 MB	3:55	00:06:38	01:19:15	4 MHz	FP5
9 MB	3:55	00:04:26	01:08:49	10 MHz	FP5
18 MB	3:55	00:09:04	02:32:43	10 MHz	FP5
128 MB	3:55	00:58:38	22:07:55	15 MHz	FP5

Observe the following recommendations:

- Because the verify time is currently not optimized, it is recommended to authenticate the SPI bitstreams with system services for quicker verification.
- Because 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 before programming other data on the SPI Flash.
- Because 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.

8.15.4. Partial Programming Support [\(Ask a Question\)](#)

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 Look-up Table (LUT) 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:

Figure 8-80. Sample List of Clients

Program	Name	Type	Index	Content File	Start Address	End Address
☒	i1	SPI Bitstream for IAP	2	C:\Workspace\Projects\Libero\hex_files\hex_files\ai_no_bypas...	0x700	0x148f
☒	g1	SPI Bitstream for Recovery/Golden	0	C:\Workspace\Projects\Libero\hex_files\hex_files\ai_golden.spi	0x1500	0x228f
☒	u1	SPI Bitstream for Auto Update	1	C:\Workspace\Projects\Libero\hex_files\hex_files\ai_no_bypas...	0x2400	0x318f
☒	INIT_STAGE_3_SPI_CLIENT	Design Initialization		designer\test\test_uic.bin	0x666	0x685
☒	Client1	Data Storage		C:\Workspace\Projects\Libero\hex_files\hex_files\random_256k...	0xfffff	0x140000
☒	Empty1	Data Storage			0x1affcd	0x1b00069
☒	Middle	Data Storage		C:\Workspace\Projects\Libero\hex_files\hex_files\random_13b...	0x3ffffa	0x4000006

The memory regions between the clients are left intact. However, SPI-Flash erases data in the selected sector, subsector, or block if the client's target Start Address is outside of sector, subsector, or block boundaries.

The software reports an error code along with messages that indicate if original data is lost.

The verify operation verifies only the data of the selected clients.

The following example is a sample log from FlashProExpress 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 = 0x1AFFCD. 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 = 0x01afff00 - 0x01afffcc
Restoring data at address = 0x01b0006a - 0x01b00fff
Operation Status: Passed

```

FlashPro3, FlashPro4, and FlashPro5 Partial Support

The partial programming support described for eFP6 and FP6 does not apply for FP3, FP4, or FP5 programmers. FP3, FP4, and FP5 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 FlashPro5 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)

```

Observe the following guidelines:

- **Backward Compatibility:** FlashPro3/4/5/6 that use the software version 2021.1 release support all jobs created using older versions of the Libero software. Do not use older FlashProExpress versions of software with FlashProExpress 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 FlashPro6 but verified with FlashPro3/4/5 programmers might fail using software version 2021.1 release. These jobs fail because the memory gap between clients is skipped during verification but are verified against 0xff if FlashPro3/4/5 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

eFlashPro6/FlashPro6 programmers also erases the SPI lookup address table. For FlashPro3/4/5 programmers, erase operations erase the entire SPI-Flash memory device.

8.15.5. SPI-Flash Bitstream Format [\(Ask a Question\)](#)

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 defined in the following table.

Table 8-51. 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.
X	X	Payload based on previous flag
Repeat for other clients		
4	FF FF FF FF	End Tag

9. Debug Design [\(Ask a Question\)](#)

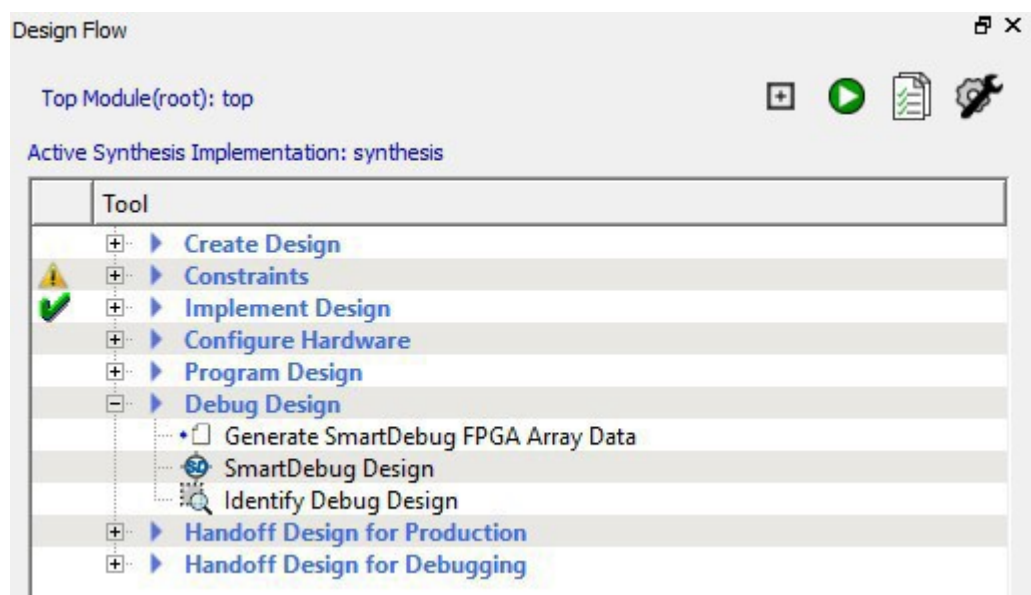
9.1. Generating SmartDebug FPGA Array Data (PolarFire) [\(Ask a Question\)](#)

The Generate SmartDebug FPGA Array Data tool generates database files used in downstream tools. These files end with a: *.db extension and are used to debug FPGA Fabric in SmartDebug.

To generate SmartDebug FPGA array data:

1. Make sure the design completed the Place and Route step. Otherwise, Libero SoC runs implicitly the upstream tools (Synthesis, Compile Netlist, and Place and Route) before it generates the FPGA SmartDebug Array Data.
2. Either double-click **Generate SmartDebug FPGA Array Data** or right-click **Generate SmartDebug FPGA Array Data** in the Design Flow window and click **Run**.

Figure 9-1. Generate SmartDebug FPGA Array Data



9.2. Using the SmartDebug Tool [\(Ask a Question\)](#)

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 allows you to check the state of inputs and outputs in real-time without having to re-layout the design.

You can run SmartDebug in two modes:

- [Integrated mode](#) from the Libero Design Flow
- [Stand-alone mode](#)

9.2.1. Integrated Mode [\(Ask a Question\)](#)

When you run SmartDebug in integrated mode from Libero, SmartDebug can access all design and programming hardware information without requiring any extra setup. Running SmartDebug in Integrated mode also makes the Probe Insertion feature available in the Debug FPGA Array.

To open SmartDebug in the Libero Design Flow window, expand **Debug Design** and double-click **SmartDebug Design**.

9.2.2. Stand-alone Mode [\(Ask a Question\)](#)

You can install SmartDebug 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 debugging purposes. In this mode, SmartDebug is launched outside the Libero Design Flow. Before launching SmartDebug in stand-alone mode, perform the SmartDebug project creation and import a Design Debug Data Container (.ddc) file that is exported from Libero to access all debug features in the supported devices.

Note: In Stand-alone mode, the Probe Insertion feature is not available in FPGA Array Debug because it requires incremental routing to connect the user net to the specified I/O.

9.3. Identifying the Debug Design [\(Ask a Question\)](#)

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

The following list summarizes key 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 operations. This feature reduces the amount of time required to 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 following debugging flow.

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

To use the Identify Instrumentor and Debugger:

1. Create your source file as you normally would 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 set up Identify.
5. In Synplify, click **Project > New Identify Implementation** to create an Identify implementation.
6. In the Implementations Options dialog box, make sure **Implementation Results > Results Directory** points to a location under `<libero project>\synthesis\`; otherwise, Libero SoC cannot detect your resulting Verilog Netlist file.
7. From the Instrumentor UI, specify the sample clock, 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. In the Design Flow window, double-click **Identify Debug Design** to launch the Identify Debugger.

10. Handoff Design for Production [\(Ask a Question\)](#)

10.1. Configuring Security Locks (and One-Way Passcode for PolarFire SoC) for Production [\(Ask a Question\)](#)

The Configure Security Locks for Production is a GUI-based tool that guides PolarFire and PolarFire SoC users on how to configure security locks for production. The tool also allows PolarFire SoC users to configure one-way passcode for production.

The wizard has six steps that are executed in sequential order. **One-Time Programmable (OTP)** settings are applied to configured Security settings from the Configure Security tool. The subsequent pages in the wizard have read-only fields that will be affected by settings for Permanent Locks (and One-way Passcode for PolarFire SoC). These settings can be configured only by the [Configure Security](#) tool.

If you configure any Permanent Lock settings in the wizard, you will be forced to go through each page to review the settings to make sure they are as desired. If PolarFire SoC users enable **Enable One-way Passcode (OWP)**, the **Finish** button becomes active and users are not required to go through each page.

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](#)

10.1.1. Summary Window [\(Ask a Question\)](#)

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.

10.1.2. Back [\(Ask a Question\)](#)

Click **Back** to return to the previous step.

10.1.3. Next [\(Ask a Question\)](#)

Click **Next** to proceed to the next step.

10.1.4. Finish [\(Ask a Question\)](#)

Click **Finish** to complete the configuration after executing the all the steps in sequential order.

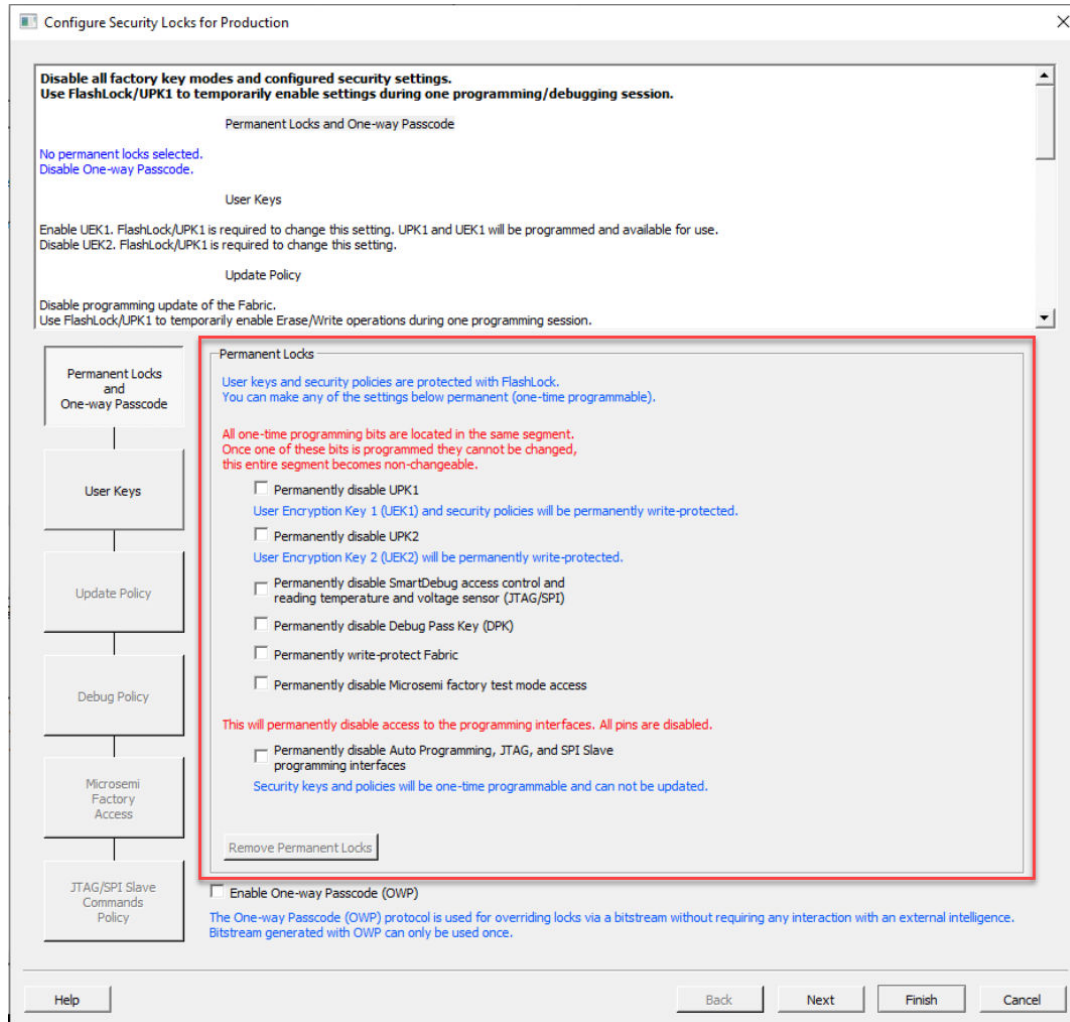
10.1.5. Save Summary to File [\(Ask a Question\)](#)

Click **Save Summary to File** to save the display in the Summary field to a file.

10.1.6. onePermanent Security Locks for Production [\(Ask a Question\)](#)

This page allows PolarFire and PolarFire SoC users to configure Permanent Locks. Security Locks must be configured after the Design/Debug phase is completed. The settings will not be applied to programming within Libero. They are applied only to the Export tools used for Production programming. Once the settings are programmed, they cannot be changed. Configuring the Security Locks for Production affects the settings on the subsequent pages and must be reviewed carefully. The settings cannot be changed after they are programmed.

Figure 10-1. Configure Security Locks for Production



All the user keys and security policies are protected with FlashLock and can be made OTP by configuring the Permanent Lock settings.

You can select one or more of the following options to lock permanently.

Table 10-1. Lockable Options (PolarFire and PolarFire SoC)

Option	Description
Permanently disable UPK1	Permanently disables FlashLock/UPK1 from being matched by the device. Any feature that is disabled will be disabled permanently. Any feature that is available will be available permanently.
Permanently disable UPK2	Permanently disables FlashLock/UPK2 from being able to be matched by the device. If UEK2 is enabled and selected for programming, it cannot be changed.
Permanently disable SmartDebug access control and reading temperature and voltage sensor	Permanently disables 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)	Permanently disables the FlashLock/DPK from being matched by the device. If DPK was programmed, it can no longer be used for SmartDebug access.

Table 10-1. Lockable Options (PolarFire and PolarFire SoC) (continued)

Option	Description
Permanently write-protect Fabric	Makes the Fabric OTP. Verification of the Fabric is possible. Erase/Program of the Fabric is disabled permanently.
Permanently disable Microsemi factory test mode access	Permanently disables Microsemi factory test mode access. Microchip will not be able to perform Failure Analysis on this device.
Permanently disable Auto Programming, JTAG and SPI Slave programming interfaces	Permanently disables all programming interfaces. 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 becomes OTP and you cannot Erase/Program/Verify the device.

10.1.7. Remove Permanent Locks [\(Ask a Question\)](#)

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, it 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.

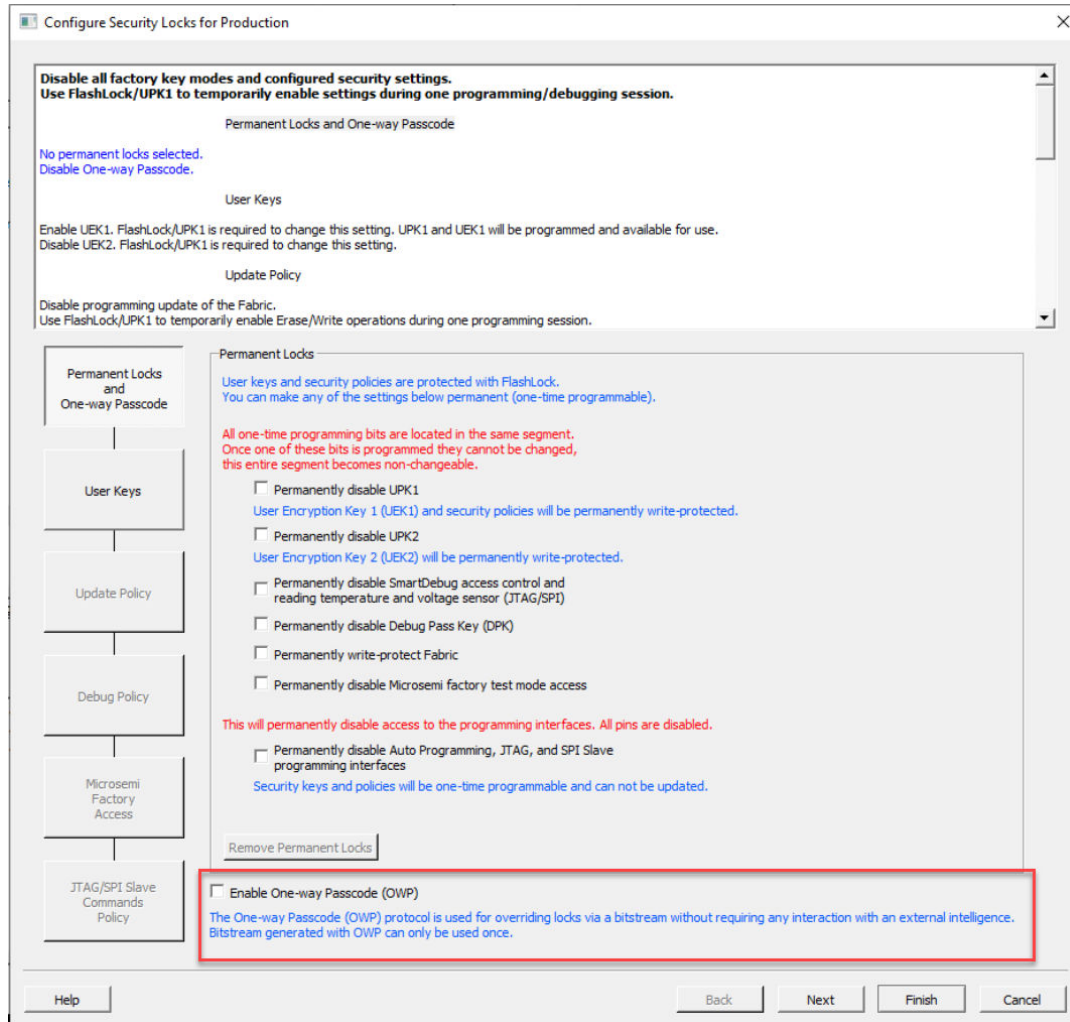
10.1.8. Enable One-way Passcode [\(Ask a Question\)](#)

The **Enable One-way Passcode (OWP)** option is provided for PolarFire SoC only. This option is available when Auto Programming and IAP Services are enabled, and either the Fabric, sNVM, or eNVM component is protected for update but not permanently locked.

If a PolarFire SoC user enables this option, the user must provide a high watermark in the Advanced options (spi file only) of the Export Bitstream file.

Note: After enabling one-way passcode, unlocking update protection for Fabric/sNVM and eNVM using the Configure Security tool disables one-way passcode automatically. For more information about one-way passcode, see the [PolarFire Family FPGA Security User Guide](#).

Figure 10-2. Enable One-way Passcode for PolarFire SoC



10.1.9. Disable One-way Pass code [\(Ask a Question\)](#)

To disable the **One-way Passcode** option, either right-click on the tool in the Design Tree or uncheck the check box in the UI.

10.2. Exporting Bitstreams [\(Ask a Question\)](#)

The Export Bitstream tool allows you to export PPD, STAPL, DAT, and SPI programming files. It also allows SmartFusion 2 users to export SVF files.

PolarFire users can use Export Bitstream to export the Fabric with sNVM. PolarFire SoC users can use Export Bitstream to export either the Fabric with sNVM or the sNVM component only.

To export a bitstream file:

- Under **Handoff Design for Production**, double click **Export Bitstream**. The Export Bitstream dialog box opens. The dialog box options depend on the device family, Custom Security settings, and **Permanent Locks** (and **One-way Passcode** for PolarFire SoC) for production settings:
 - Device Configured with Bitstream Encryption with Default Key in the Configure Security Tool (PolarFire and PolarFire SoC)
 - Device Configured with Custom Security Option in the Configure Security Tool (PolarFire and PolarFire SoC)

- Device Configured with Security Locks for Production Tool (PolarFire)
 - Device Configured with Bitstream Encryption with Default Key in the Security Policy Manager (SmartFusion 2 and IGLOO 2)
 - Device Configured with Custom Security Option in the Security Policy Manager (SmartFusion 2 and IGLOO 2)
 - Export Bitstream (RTG4)
2. Enter your bitstream file name and select the location to export the bitstream file.
 3. Choose the options that apply to your device (see [Export Bitstream Dialog Box Options](#)).
 4. Click **OK** to export the selected bitstream files.

10.2.1. Export Bitstream Dialog Boxes [\(Ask a Question\)](#)

The following sections show the Export Bitstream dialog boxes for the various Microchip product families.

10.2.1.1. Device Configured with Bitstream Encryption with Default Key in the Configure Security Tool (PolarFire and PolarFire SoC) [\(Ask a Question\)](#)

The following figures show the Export Bitstream options for PolarFire and PolarFire SoC devices configured with bitstream encryption with a default key in the Configure Security tool.

Figure 10-3. Export Bitstream Encrypted with Default Key Dialog Box (PolarFire)

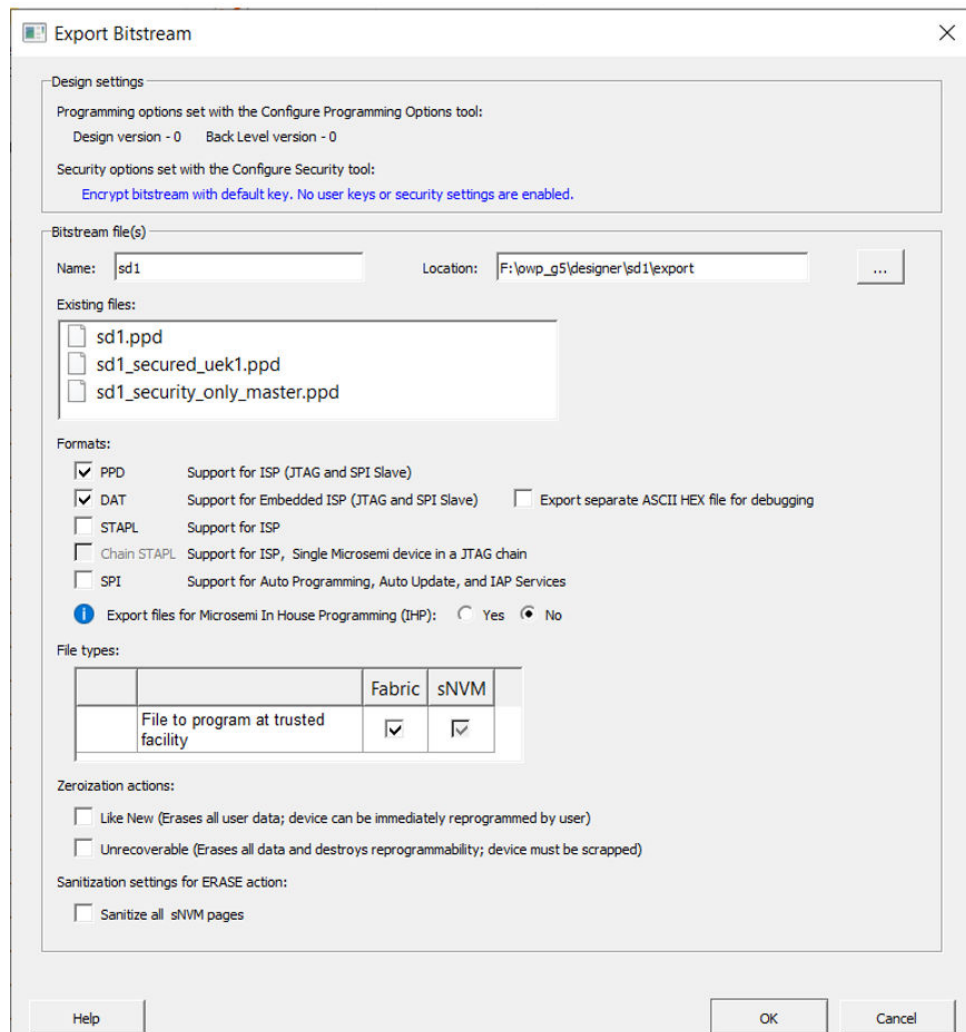
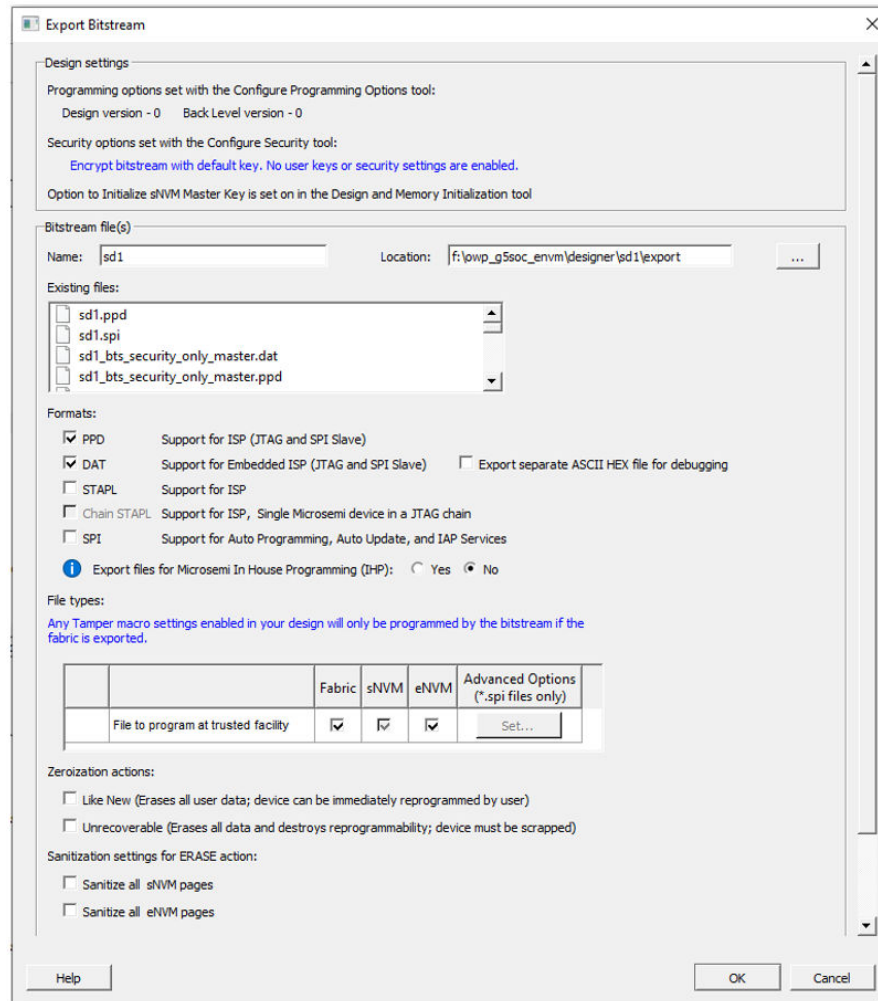


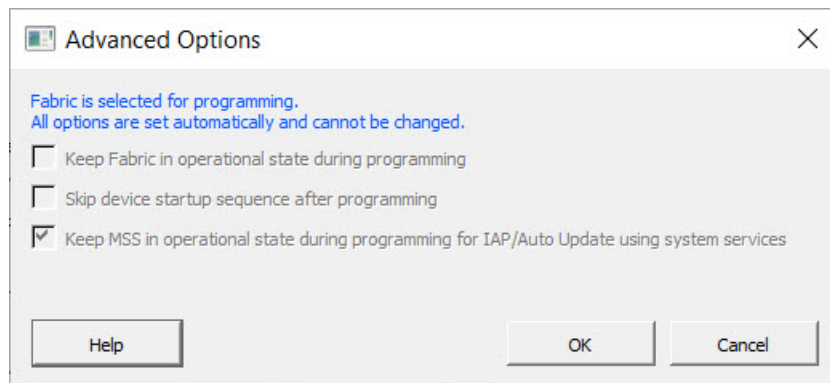
Figure 10-4. Export Bitstream Encrypted with Default Key Dialog Box (PolarFire SoC)



Advanced Options for a Fabric Programmed without Custom Security (SPI Files Only)

If Fabric is being programmed without custom security, all options are set automatically and cannot be changed (see the following figure).

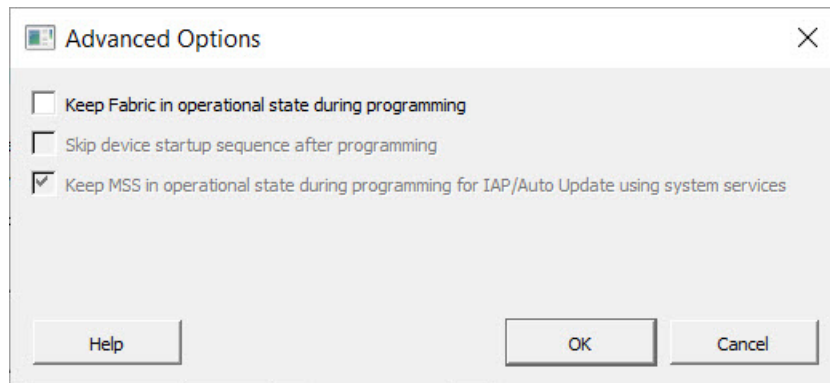
Figure 10-5. Advanced Options



Advanced Options for a Fabric Not Being Programmed (SPI Files Only)

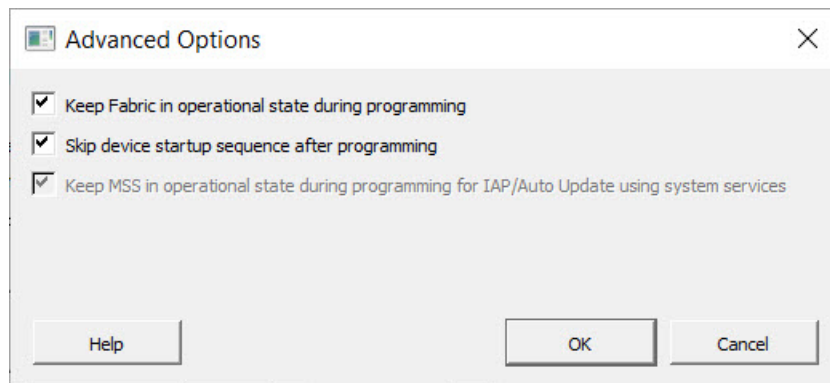
If Fabric is not being programmed, you can choose to **Keep Fabric in operational state during programming** (see the following figure).

Figure 10-6. Advanced Options



If **Keep Fabric in operational state during programming** is selected, you can also select **Skip device start-up sequence after programming** (see the following figure).

Figure 10-7. Advanced Options



10.2.1.2. Device Configured with Custom Security Option in the Configure Security Tool (PolarFire and PolarFire SoC) [\(Ask a Question\)](#)

The following figures show the Export Bitstream options for PolarFire and PolarFire SoC devices.

Figure 10-8. Export Bitstream with Custom Security Dialog Box (PolarFire)

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:
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: sd1 Location: F:\pwp_g5\designer\sd1\export ...

Existing files:
sd1.ppd
sd1_secured_uek1.ppd
sd1_security_only_master.ppd

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:
Any Tamper macro settings enabled in your design will only be programmed by the master bitstream, not by UEK1/UEK2 bitstreams, if the fabric is exported.

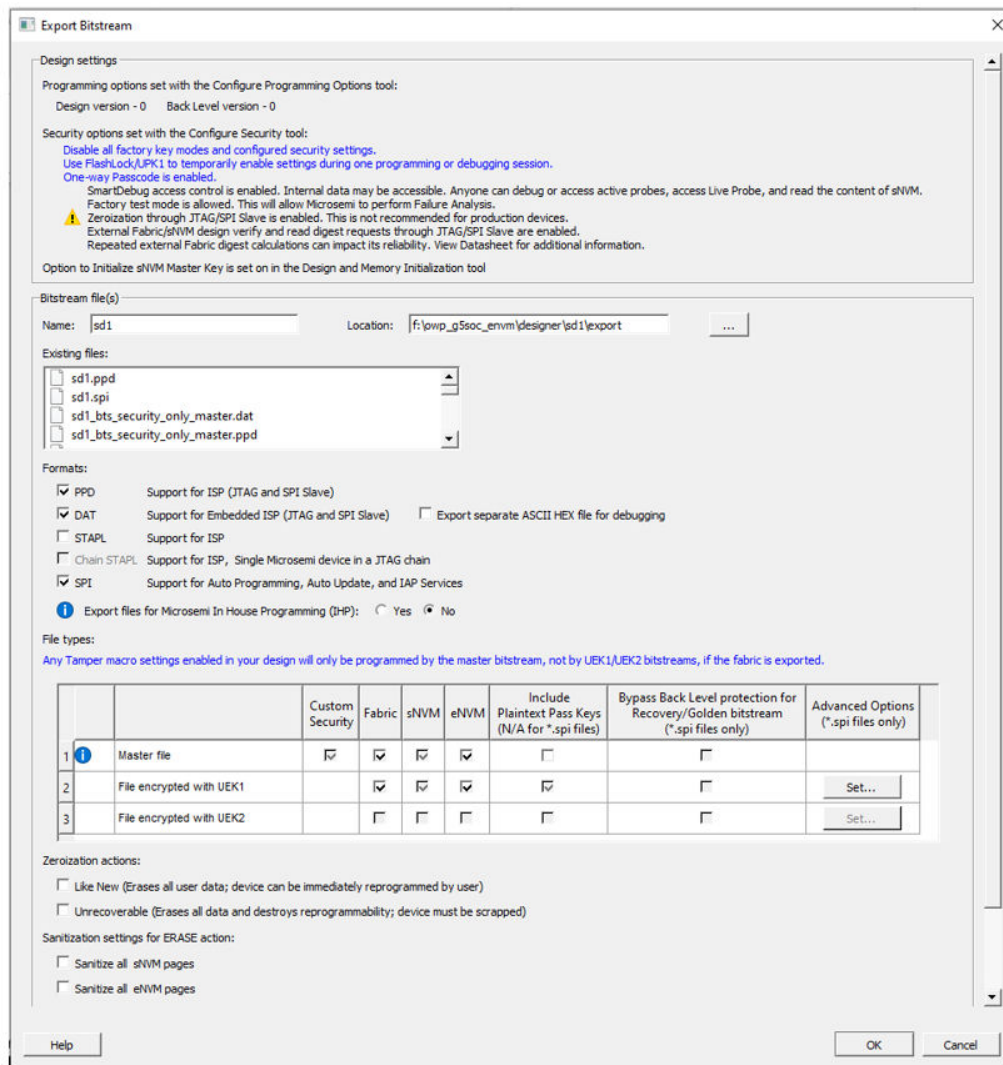
		Custom Security	Fabric	sNVM	Include Plaintext Pass Keys (N/A for *.spi files)	Bypass Back Level protection for Recovery/Golden bitstream (*.spi files only)
1	Master file	☒	☒	☒	☐	☐
2	File encrypted with UEK1	☐	☒	☒	☐	☐
3	File encrypted with UEK2	☐	☐	☐	☐	☐

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

Figure 10-9. Export Bitstream with Custom Security Dialog Box (PolarFire SoC)



Include Plaintext Pass Keys

The Export Bitstream with Custom Security dialog box has an **Include Plaintext Pass Keys** option for including plaintext UPK1 and plaintext UPK2 (if set) in a master bitstream file and adding plaintext UPK1 in update bitstream files. You can use this option for programming updates with protected bitstream components or disabled programming actions (program, authenticate, verify) through JTAG and SPI Slave interfaces. Programming through IAP services does not require a passkey, so the **Include Plaintext Pass Keys** option is not supported for SPI bitstreams.

The following product families support plaintext pass keys:

- PolarFire
- PolarFire SoC

Master and update bitstream files are handled in the following way:

- For a master bitstream file, the plaintext pass keys option is allowed if UPK1 is not permanently locked or if UPK2 is set and not permanently locked.
- For update bitstream files, the **Include Plaintext Pass Keys** option is allowed if UPK1 is not permanently locked and either:
 - The Fabric, sNVM, or eNVM is protected.
 OR

- Any programming action in the security update policy is disabled.
- . The **Include Plaintext Pass Keys** option is automatically enabled when passkeys are required for updates.

Advanced Options for a File Encrypted with UEK1 or UEK2 when Fabric is Being Programmed

If Fabric is being programmed for a file encrypted with UEK1 or UEK2, all advanced options are set automatically and cannot be changed (see the following figure).

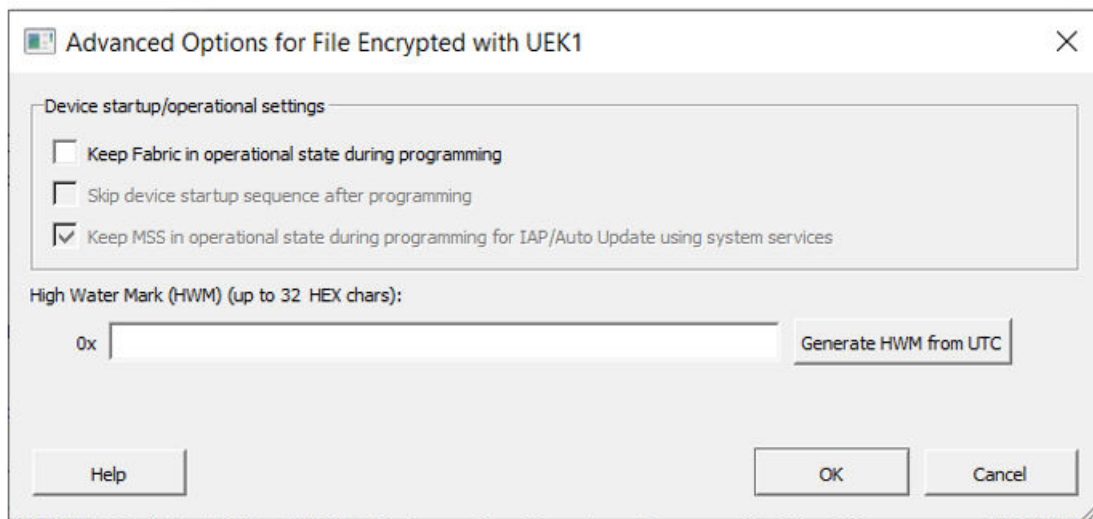
Figure 10-10. Advanced Options for File Encrypted with UEK1



Advanced Options for a File Encrypted with UEK1 or UEK2 when Fabric is Not Being Programmed

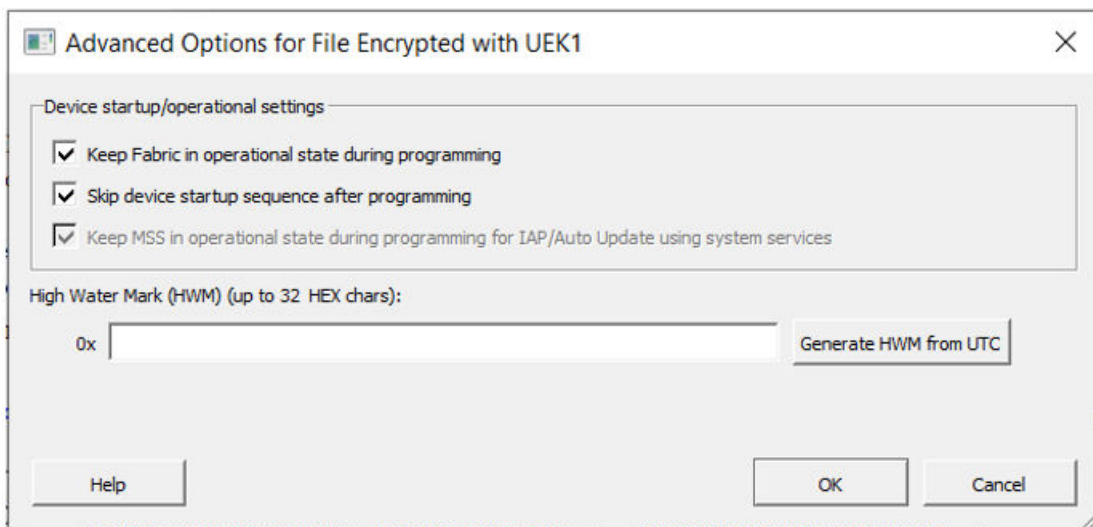
PolarFire SoC no longer has advanced options for the master file. The Advanced Options dialog has been updated with **High Water Mark** options related to **One-way Passcode**. If Fabric is not being programmed for a file encrypted with UEK1 or UEK2, you can select **Keep Fabric in operational state during programming** (see the following figure).

Figure 10-11. Advanced Options for File Encrypted with UEK1



If **Keep Fabric in operational state during programming** is selected, you can also select **Skip device start-up sequence after programming** (see the following figure).

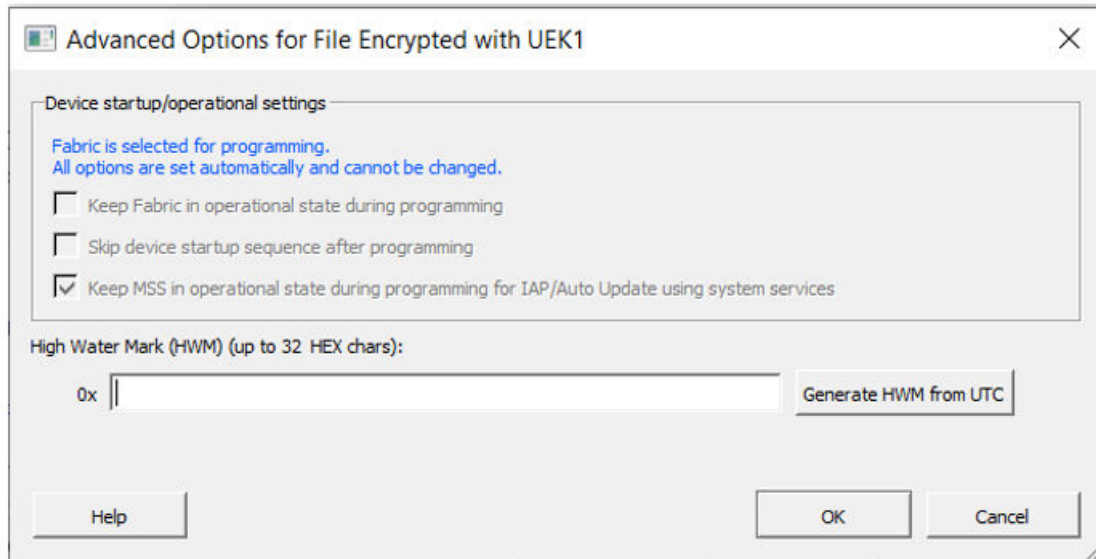
Figure 10-12. Advanced Options for File Encrypted with UEK1



High Water Mark

If the **One-way Passcode (OWP)** option is enabled, a **High Water Mark** is required (see the following figure).

Figure 10-13. Advanced Options for File Encrypted with UEK1



Examples of Using Bypass Back Level Protection

The following examples show how to use Bypass Back Level Protection.

Table 10-2. 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 preceding example, the device will have Back Level version 2 after step 2. Trying to program with bitstream in step 3 fails. If the device tries to reprogram, it initiates Programming Recovery with the Golden/Recovery bitstream. Because the Golden/Recovery Bitstream has Design version 2, which is less than or equal to the Back Level version in the device, it fails. If the Bypass Back Level version option is selected, this back level protection check is bypassed for the Golden/Recovery Bitstream only and it succeeds.

Table 10-3. 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 try to program with bitstream in step 3, it succeeds because the Golden/Recovery Design version is greater than the Back Level version on the device.

Table 10-4. 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. Trying to program with bitstream in step 3 fails. If the device tries to reprogram, it initiates Programming Recovery with the Golden/Recovery bitstream. Because the Golden/Recovery Bitstream has Design version 1, which is less than or equal to the Back Level version in the device, it fails. If the Bypass Back Level version option is selected, this back level protection check is bypassed for the Golden/Recovery Bitstream only and it succeeds.

10.2.1.3. Device Configured with Security Locks for Production Tool (PolarFire) [\(Ask a Question\)](#)

If Security Locks for Production are configured along with Custom Security settings, Export Bitstream Security options are set with the Configure Security Locks for Production tool.

For examples of using Bypass Black Level protection, see [Device Configured with Custom Security Option in the Configure Security Tool \(PolarFire and PolarFire SoC\)](#).

Figure 10-14. Export Bitstream Dialog Box with Security Options Set with the Configure Security 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 Security Locks for Production tool:
 ⚠️ Permanently disable all factory key modes and configured security settings.
 Fabric and sNVM update is one-time programmable.
 SmartDebug access control is permanently 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 permanently allowed. This will allow Microsemi to perform Failure Analysis.
 Zeroization through JTAG/SPI Slave is permanently enabled. This is not recommended for production devices.
 External Fabric/sNVM design verify and read digest requests through JTAG/SPI Slave are permanently enabled.
 Repeated external Fabric digest calculations can impact its reliability. View Datasheet for additional information.
 Option to Initialize sNVM Master Key is set on in the Design and Memory Initialization tool

Bitstream file(s)

Name: Location: ...

Existing files:

- sd1.ppd
- sd1.spi
- sd1_bts_security_only_master.dat
- sd1_bts_security_only_master.ppd

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:
Any Tamper macro settings enabled in your design will only be programmed by the master bitstream, not by UEK1/UEK2 bitstreams, if the fabric is exported.

		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	☒	☒	☒	☒	☐	☐	
2	File encrypted with UEK1	☐	☐	☐	☒	☐	☐	Set...
3	File encrypted with UEK2	☐	☐	☐	☐	☐	☐	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:

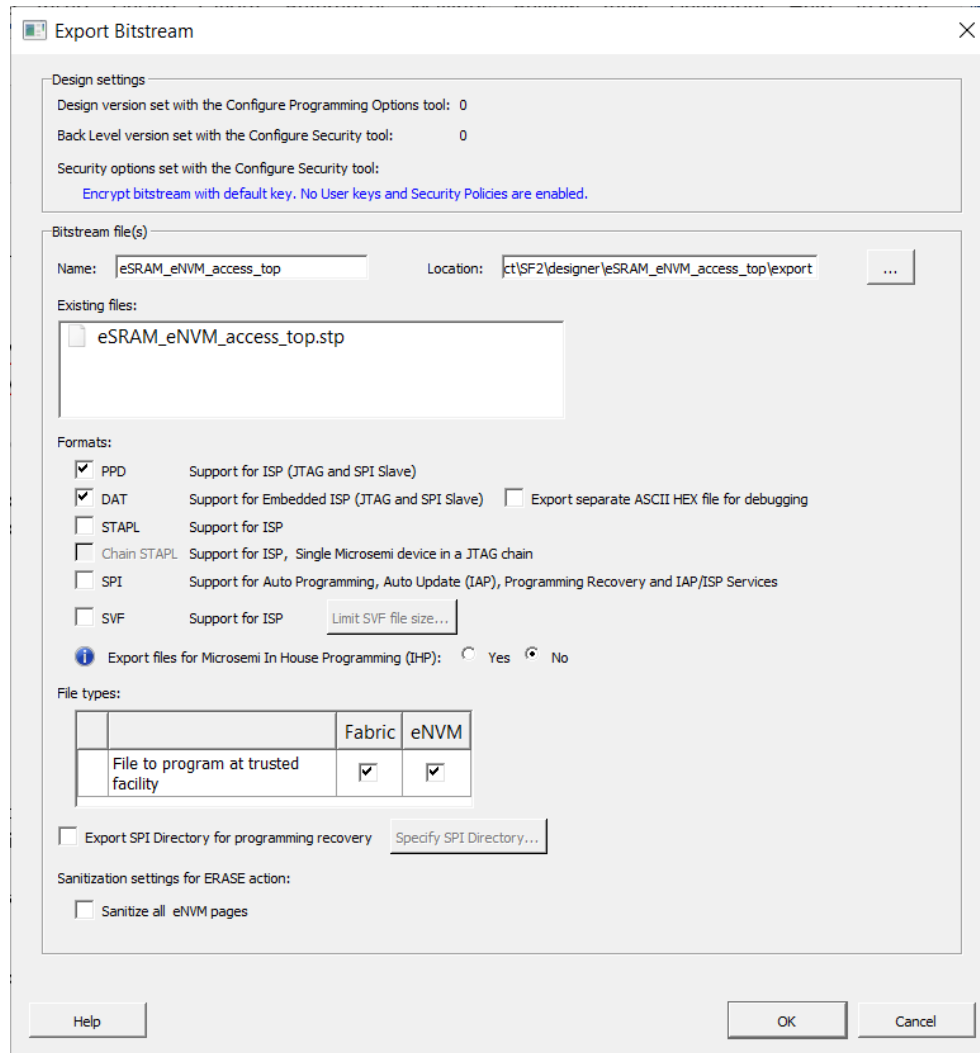
- ☐ Sanitize all sNVM pages
- ☐ Sanitize all eNVM pages

Help OK Cancel

10.2.1.4. Device Configured with Bitstream Encryption with Default Key in the Security Policy Manager (SmartFusion 2 and IGLOO 2) [\(Ask a Question\)](#)

The following figure shows the Export Bitstream options for a SmartFusion 2 or IGLOO 2 device configured with the default key option in the Security Policy Manager.

Figure 10-15. Export Bitstream Dialog Box with Default Key



10.2.1.5. Device Configured with Custom Security Option in the Security Policy Manager (SmartFusion 2 and IGLOO 2) [\(Ask a Question\)](#)

The following figure shows the Export Bitstream options for a SmartFusion 2 or IGLOO 2 device configured with the Custom Security option in the Security Policy Manager.

Figure 10-16. Export Bitstream Dialog Box with Configured Custom Security Options in the Security Policy Manager

Export Bitstream

Design settings

Design version set with the Configure Programming Options tool: 0

Back Level version set with the Configure Security tool: 0

Security options set with the Configure Security tool:

- Protect UEK1, UEK2, DPK and Security Policies using FlashLock/UPK1.
- Protect modification to UEK3 via bitstream using FlashLock/UPK1. This does not prevent SRAM PUF System Service requests from affecting UEK3.
- Disable factory enabled default bitstream encryption key modes.
- Protect Microsemi factory test mode access using FlashLock/UPK1.

Bitstream file(s)

Name: eSRAM_eNVM_access_top Location: ct\SF2\designer\eSRAM_eNVM_access_top\export

Existing files:

- eSRAM_eNVM_access_top.stp

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 (IAP), Programming Recovery and IAP/ISP Services
- ☐ SVF Support for ISP [Limit SVF file size...](#)

☒ Export files for Microsemi In House Programming (IHP): ☐ Yes ☒ No

File types:

	Custom Security	Fabric	eNVM	Include Plaintext Pass Keys (N/A for *.spi files)
Master file	☒	☒	☒	☐
File encrypted with UEK1		☒	☒	☐
File encrypted with UEK2		☒	☒	☐
File encrypted with UEK3		☐	☐	☐

☐ Export SPI Directory for programming recovery [Specify SPI Directory...](#)

Sanitization settings for ERASE action:

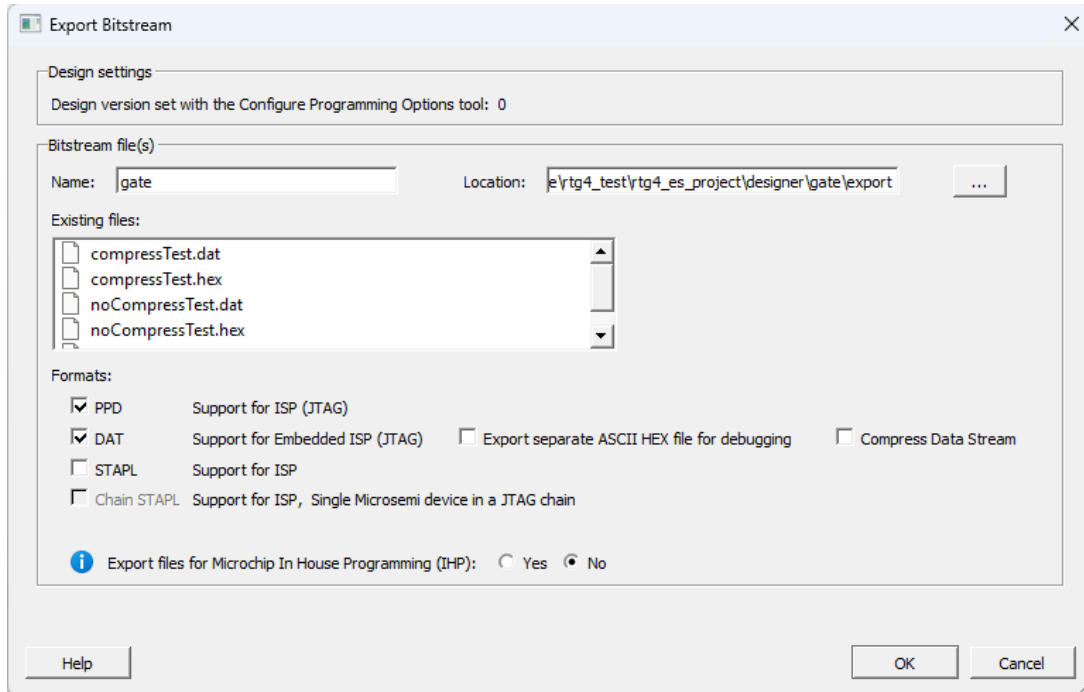
- ☐ Sanitize all eNVM pages

Help OK Cancel

10.2.1.6. Export Bitstream (RTG4) [\(Ask a Question\)](#)

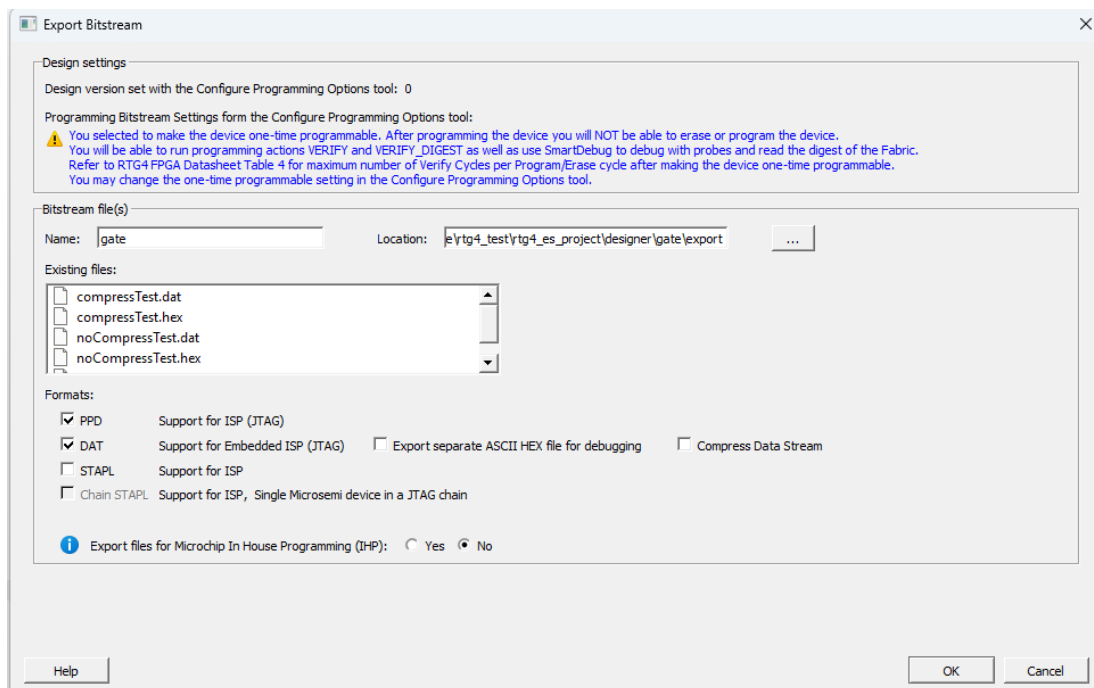
The following figures show the Export Bitstream options for RTG4 devices.

Figure 10-17. Export Bitstream Dialog Box with Compress Data Stream Option



If you select the **One-time programmable (OTP)** option in Configure Programming Options, a message appears. Click **Yes** to continue or **No** to cancel.

Figure 10-18. Export Bitstream Dialog Box with OTP



10.2.2. Export Bitstream Dialog Box Options [\(Ask a Question\)](#)

The following sections describe the options in the Export Bitstream dialog boxes for the various product families.

10.2.2.1. Design Settings [\(Ask a Question\)](#)

Read-only design settings that vary with the product family, such as design version, back level version, and security options.

10.2.2.2. Bitstream Files [\(Ask a Question\)](#)

Name. Set the name of your bitstream file. The default name is the design name.

Location. Specify the location where the exported file will be saved.

10.2.2.3. Existing files [\(Ask a Question\)](#)

Programming job files at the selected location.

10.2.2.4. Formats [\(Ask a Question\)](#)

Select the bitstream file format you want to export. Choices are:

- PPD
- DAT
- Export separate ASCII HEX file for debugging- Exports DAT file in HEX format. This option is available only when DAT file format is selected.
- STAPL
- Chain STAPL- Enabled when there are two or more devices in the chain.
- SPI
- SVF (SmartFusion 2)
- Export separate ASCII HEX file for debugging- Exports DAT file in HEX format. This option is available only when DAT file format is selected.
- Export files for Microchip In House Programming (IHP)- Check **Yes** to export DAT and STP file formats.

PPD and DAT file formats are the default file formats. If Auto programming is disabled in Configure Security, neither master nor update files can be exported in SPI format. STAPL and DAT are required for In House Programming.

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 PolarFire SoC, 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 the SPI bitstream file for update cannot include a Fabric component. If Fabric is open and eNVM is locked then, SPI bitstream file for update cannot include eNVM.

10.2.2.5. File Types [\(Ask a Question\)](#)

Security-only programming must be performed on erased or new devices only. If performed on a device with Fabric programmed, the Fabric will be disabled after performing security-only programming, and you will have to reprogram the Fabric to re-enable it.

PolarFire, PolarFire SoC, SmartFusion 2, and IGLOO 2 with Default Security

File to Program at trusted facility. Available when the design is configured with Bitstream Encryption using the Default Key in the Configure Security tool. Choices are:

- **Fabric sNVM.** Check to enable programming for Fabric/sNVM bitstream components at a trusted facility.
- **eNVM.** Check to enable programming for eNVM bitstream components at a trusted facility. This feature is not supported for PolarFire.

PolarFire, PolarFire SoC, SmartFusion 2, and IGLOO 2 with Custom Security

Master file. The master file is used to program a trusted facility. Available when the design is configured with Custom Security options in the Configure Security tool. For PolarFire, the master file is also available when the design is configured with Permanent Locks in the Configure Permanent Locks for Production tool. Choices are:

- **Custom Security-** Always programmed in the master file.
- **Fabric sNVM-** Check to enable programming for Fabric/sNVM bitstream components.
- **eNVM-** Check to enable programming for eNVM bitstream components. This feature is not supported for PolarFire.
- **Include Plaintext Pass Keys-** Check to add plaintext pass keys to the bitstream files to be programmed. This option is available if UPK1 is not permanently locked and the Update Policy Fabric /sNVM or eNVM in PolarFire SoC is passcode protected or any of program, authenticate, or verify action is locked.

File encrypted with UEK1- Used to program at untrusted facility or for Broadcast field update. Choices are:

- **Fabric sNVM**
- **Include Plaintext Pass Keys**

File encrypted with UEK2- Used to program at untrusted facility or for Broadcast field update. Choices are:

- **Fabric sNVM**
- **Include Plaintext Pass Keys**

File encrypted with UEK3- Used to program at untrusted facility or for Broadcast field update. Available for M2S060, M2GL060, M2S090, M2GL090, M2S150, and M2GL150 devices. Choices are:

- **Fabric**
- **eNVM**
- **Include Plaintext Pass Keys**

Notes: Observe the following guidelines:

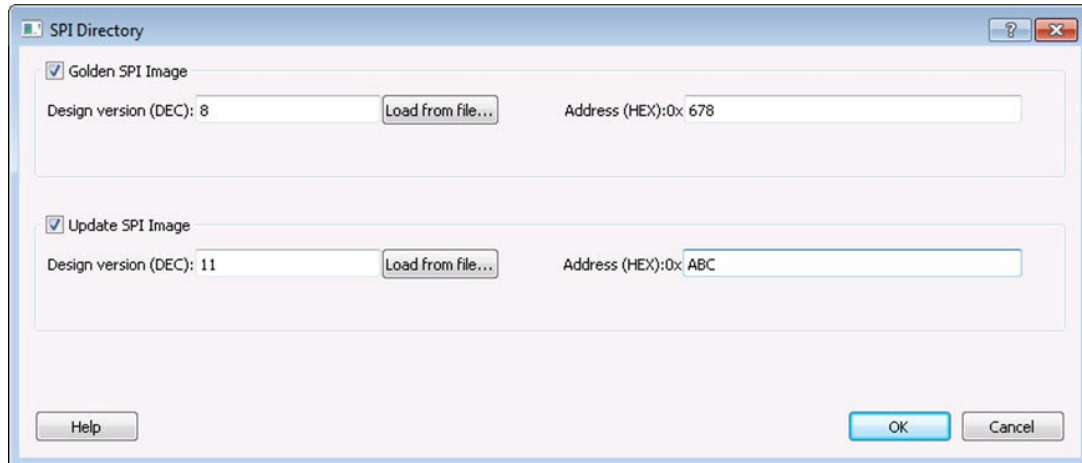
- If eNVM/Fabric is OTP, it is precluded from bitstream encrypted with UEK1/2/3. Because eNVM is always open for updates, the eNVM OTP warning is not applicable for SmartFusion 2 and IGLOO 2.
- UEK3 is available only for M2S060S, M2GL060S, M2S090S, M2GL090S, M2S150S, and M2GL150S devices.
- If a component such as eNVM is not present in the design, it will be disabled in the bitstream component selection.

10.2.2.6. Export SPI Directory for Programming Recovery (SmartFusion 2 and IGLOO 2) [\(Ask a Question\)](#)

The **Export SPI Directory for programming recovery** option appears for SmartFusion 2 and IGLOO 2 devices configured with Custom Security Option in the Security Policy Manager.

This option allows you to export an SPI directory containing Golden and Update SPI image addresses and design versions used in Autoupdate and Programming Recovery flow. Check this option and click **Specify SPI Directory** to set the required information in the SPI Directory dialog box.

Figure 10-19. SPI Directory Dialog Box



10.2.2.7. Compressing DAT Files (RTG4) [\(Ask a Question\)](#)

DAT files generated for RTG4 devices can exceed the available memory capacity. To reduce file size and system memory usage, enable the **Compress Data Stream** option when exporting a DAT bitstream for an RTG4 device. Compressed DAT files are comparable in size to STP files and are easier to store in memory.



Attention: DAT files are supported in DirectC applications only.

When using compressed DAT files, the following differences are visible in the hexadecimal representation:

- **Header Section:** The first header byte changes from 0x52 (RTG4) to 0x43 (CTG4) for compressed files. The header size also changes from 0x38 to 0x3C.
- **Constant Data Block:** A new field is added that specifies the compressed image size.
- **CRC Values:** The final two CRC bytes at the end of the file differ from those in an uncompressed DAT file.

Note: When reading the Constant Data Block, be aware that hexadecimal values are presented in reverse order to match the required endianness.

10.2.2.8. Zeroization Actions [\(Ask a Question\)](#)

Zeroization actions appears in the Export Bitstream dialog box when a PolarFire or PolarFire SoC device is configured with:

- The Custom Security option in the Configure Security Tool
- Bitstream encryption with a default key in the Configure Security Tool

Note: **Zeroization actions** is not supported for XT and ES devices.

Zeroization actions can have two options, depending on the device and configured features:

- **Like New-** Erase all user data and reprogram the device immediately.
- **Unrecoverable-** Erase all data and destroy reprogrammability. If selected, the device must be scrapped.

10.2.2.9. Sanitization Settings for ERASE Action [\(Ask a Question\)](#)

Sanitization settings for ERASE action appears in the Export Bitstream dialog box when a PolarFire, PolarFire SoC, SmartFusion 2, or IGLOO 2 device is configured with:

- The Custom Security option in the Configure Security Tool
- Bitstream encryption with a default key in the Configure Security Tool

Sanitization settings for ERASE action can have two options, depending on the device and configured features.

PolarFire and PolarFire SoC

- **Sanitize all sNVM pages.** Available if Fabric/sNVM component is selected for at least one Master file or update file.
- **Sanitize all eNVM pages.** Available for PolarFire SoC only if eNVM is configured and selected for at least one master or update file.

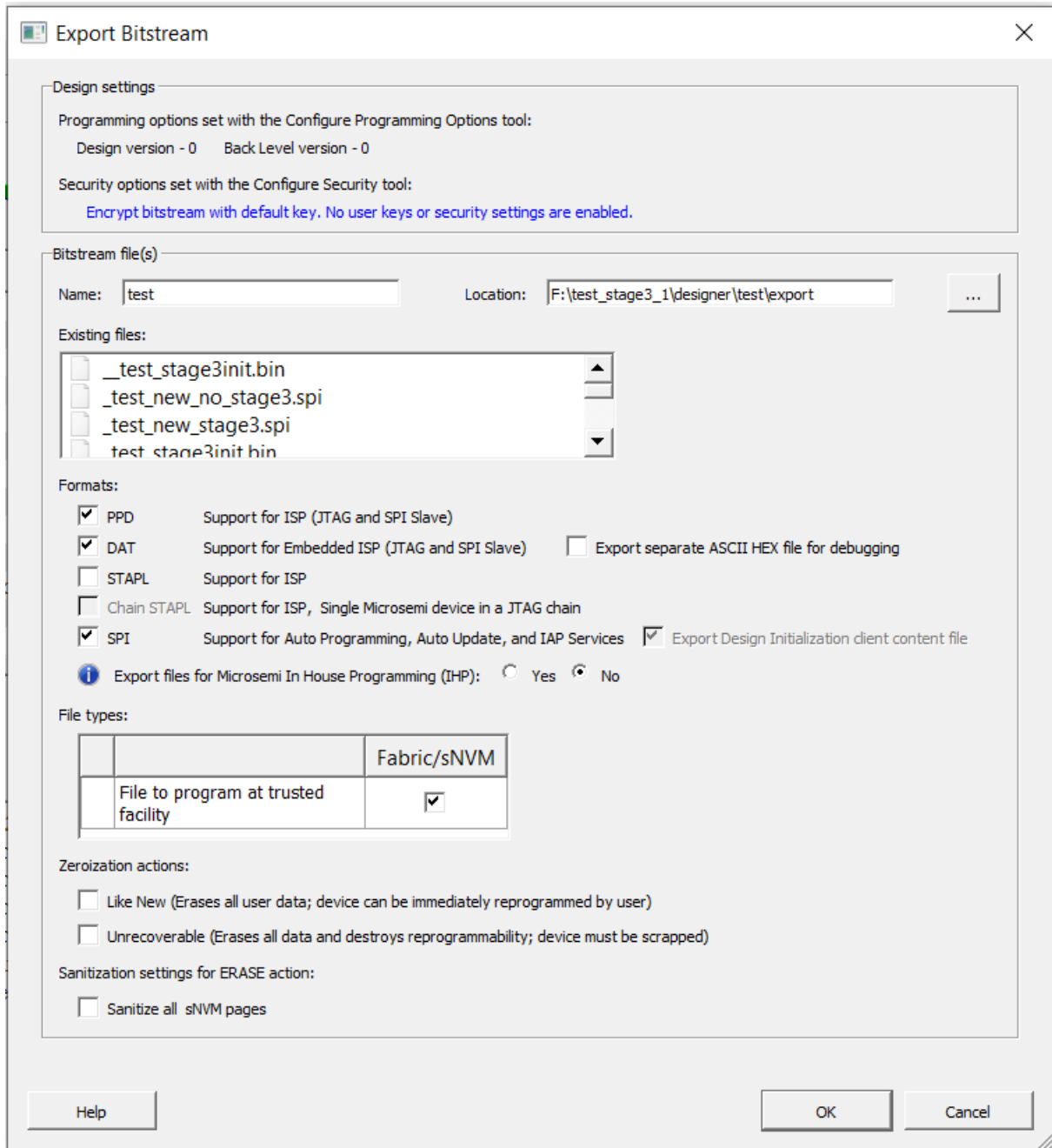
SmartFusion 2 and IGLOO 2

Sanitize all eNVM pages. Available only if eNVM is configured and selected for at least one master or update file.

10.2.2.10. STAGE3 Initialization Clients [\(Ask a Question\)](#)

Starting with Libero v2021.2, when you export a SPI file when SPI Flash has an `INIT_STAGE_3_SPI_CLIENT`, PolarFire and PolarFire SoC always export the STAGE3 Initialization client content file (`.bin`) along with a SPI file. The name of the STAGE3 Initialization file is based on the name of the SPI file: `<spi_file_name>_stage3init.bin`. The following figure shows an example of exported STAGE3 Initialization files in the **Existing files** list.

Figure 10-20. Example of an Export Bitstream Tool with a STAGE3 Initialization Client



When you configure SPI Flash and use the SPI file with STAGE3 data, the start address for the STAGE3 Initialization client is taken from the SPI file. The SPI file also contains a digest to verify the STAGE3 Initialization client file.

10.2.3. Security Programming Files [\(Ask a Question\)](#)

Expanding **Handoff Design for Production** in the Design Flow window, and then selecting **Export Bitstream** creates the following files.

Table 10-5. Files Created by Export Bitstream

File	Description
PolarFire: <filename>_master.(stp/spi/dat) SmartFusion2, IGLOO2, and RTG4: PolarFire: <filename>_master.(stp/svf/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.
PolarFire: <filename>_security_only_master.(stp/spi/dat) SmartFusion2, IGLOO2, and RTG4: PolarFire: <filename>_security_only_master.(stp/svf/spi/dat)	Created when Enable custom security options is specified in the Security Wizard. Master security programming file. This file includes User Key Set 1, User Key Set 2 (optionally if specified), and your security policy settings.
PolarFire: <filename>_uek1.(stp/spi/dat) SmartFusion2, IGLOO2, and RTG4: PolarFire: <filename>_uek1.(stp/svf/spi/dat)	Programming file encrypted with User Encryption Key 1 used for field updates. This file includes all your features for programming, except security.
PolarFire: <filename>_uek2.(stp/spi/dat) SmartFusion2, IGLOO2, and RTG4: PolarFire: <filename>_uek2.(stp/svf/spi/dat)	Programming file encrypted with User Encryption Key 2 used for field updates. This file includes all your features for programming, except security.
SmartFusion2, IGLOO2, and RTG4: <filename>_uek3. (stp/svf/spi/dat)	Programming file encrypted with User Encryption Key 3 used for field updates. This file includes all your features for programming except security. Note: UEK3 is available only for M2S060S, M2GL060S, M2S090S, M2GL090S, M2S150S, and M2GL150S devices.

10.3. Exporting FlashPro Express Jobs [\(Ask a Question\)](#)

The following topics describe the exporting of FlashPro Express jobs.

10.3.1. Prepare Design for Production Programming in FlashPro Express [\(Ask a Question\)](#)

After you export a programming job, you can handoff this programming job to the FlashPro Express tool for production programming.

Note: Preparing a design for production programming in FlashPro Express is the same for all Libero device families.

1. In FlashPro Express, from the **File** menu choose **Create Job Project From a Programming Job**. A prompt asks you to specify the Programming Job location you just exported from Libero and the location where to store the Job Project. The Job Project name uses the programming job name automatically and cannot be changed.
2. Click **OK**. A new Job Project is created and opened for production programming.

10.3.2. Export FlashPro Express Job Dialog Boxes [\(Ask a Question\)](#)

The following sections show the Export FlashPro Express Job dialog box for PolarFire, SmartFusion 2, IGLOO 2, and RTG4. Dialog box options vary, depending on the device you are using and the security settings.

10.3.2.1. Export FlashPro Express Job (PolarFire and PolarFire SoC) [\(Ask a Question\)](#)

To program the design using the stand-alone FlashPro Express tool on Linux or Windows®, 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.

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.

Note: PolarFire SoC users can use Export FlashPro Express to export either Fabric with sNVM or sNVM component only. The eNVM feature is displayed for PolarFire SoC devices when eNVM is used in the design. eNVM features are available for PolarFire SoC devices only.

Figure 10-21. 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.

Option to Initialize sNVM Master Key is set on in the Design and Memory Initialization tool

FlashPro Express Job file

Name: Location: ...

Existing files:

- sd1.job
- sd1_job.job

Program (JTAG programming interface):

☒ Design
☐ SPI Flash

Design bitstream file format: ⓘ

Design bitstream file type:
Any Tamper macro settings enabled in your design will only be programmed by the bitstream if the fabric is exported.

	Fabric	sNVM	eNVM
File to program at trusted facility	☒	☒	☒

Design actions and procedures: ⓘ

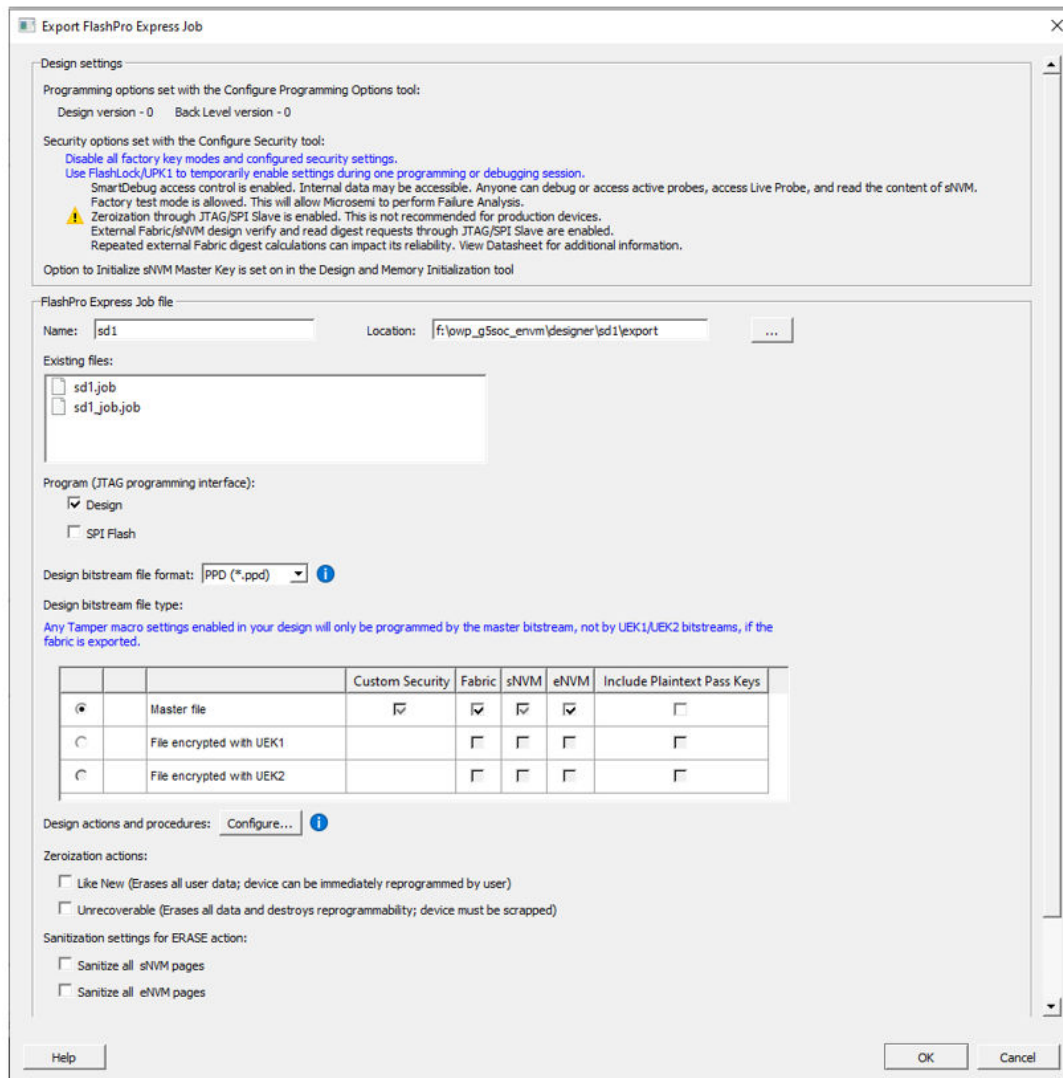
Zeroization actions:

☐ 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

Figure 10-22. Export FlashPro Express Job Dialog Box with Custom Security Options Set with the Configure Security Tool



Attention: If exporting FlashPro Express jobs that have protected bitstream components or disabled programming actions (program, authenticate, verify) through JTAG and SPI slave interfaces, see section [Device Configured with Custom Security Option in the Configure Security Tool \(PolarFire and PolarFire SoC\)](#) for information about using the **Include Plaintext Pass Keys** option.

10.3.2.2. Export FlashPro Express Job (SmartFusion 2 and IGLOO 2) [\(Ask a Question\)](#)

To program the design using the stand-alone FlashPro Express tool on Linux or Windows, export a FlashPro Express Job. The job file will include chain configuration, Programmer Settings, Programming Mode (JTAG/SPI-Slave), and programming files loaded from Programming Connectivity and Interface.

Note: SPI Slave mode is supported by FlashPro5 for SmartFusion 2 and IGLOO 2 devices, and by FlashPro6 for SmartFusion 2 and IGLOO 2 devices. JTAG is the default interface.

For SmartFusion 2 and IGLOO 2, Security Programming is supported. Use the Security Policy Manager to configure Security before you export the programming job. The Export FlashPro Express Job dialog box for SmartFusion 2 and IGLOO 2 displays the Security Options you configured in the Security Policy Manager.

The Export FlashPro Express Job dialog box options vary, depending on the device you are using and the Security Key Mode you selected.

Figure 10-23. Export FlashPro Express Job Dialog Box with Default Bitstream Encryption

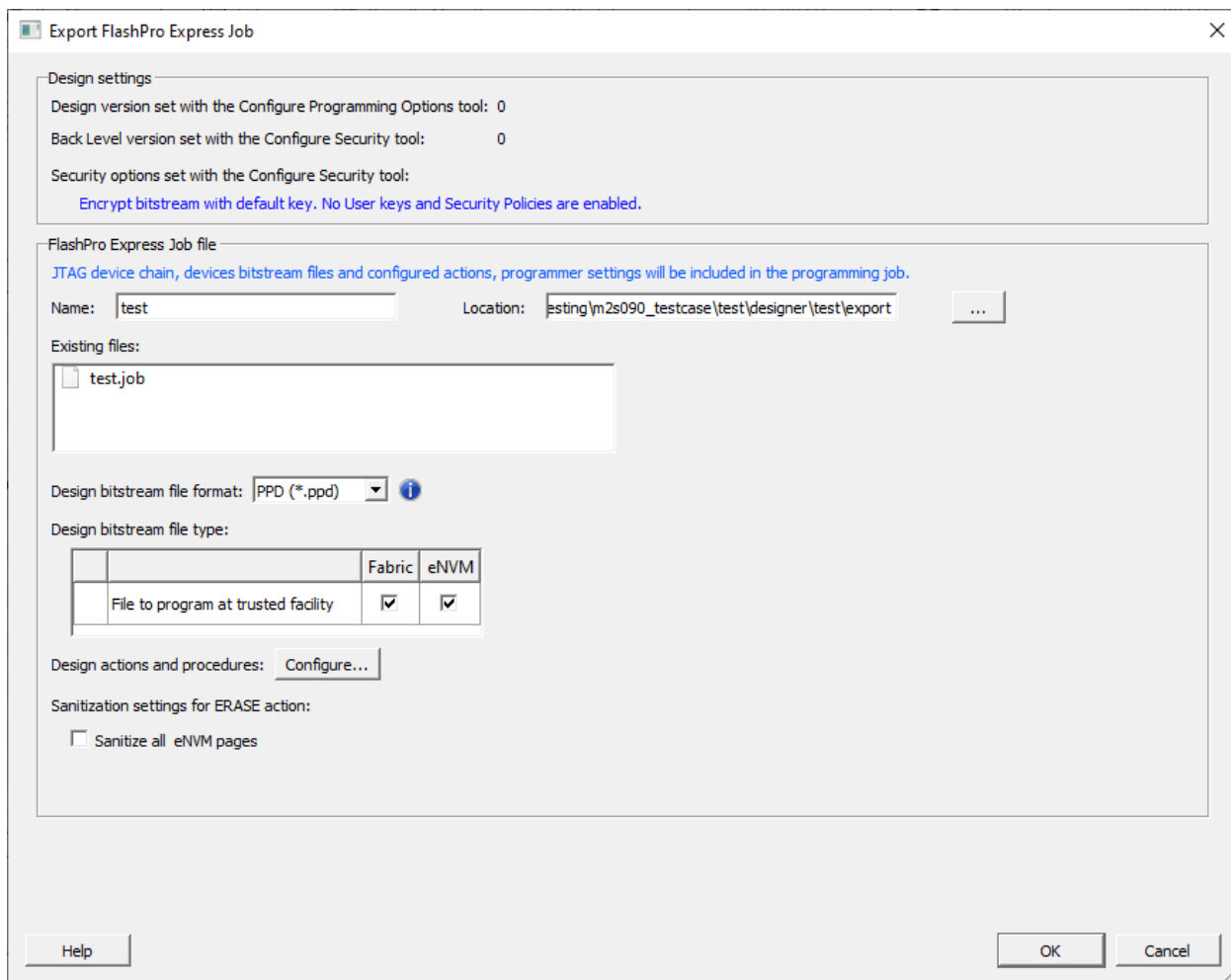
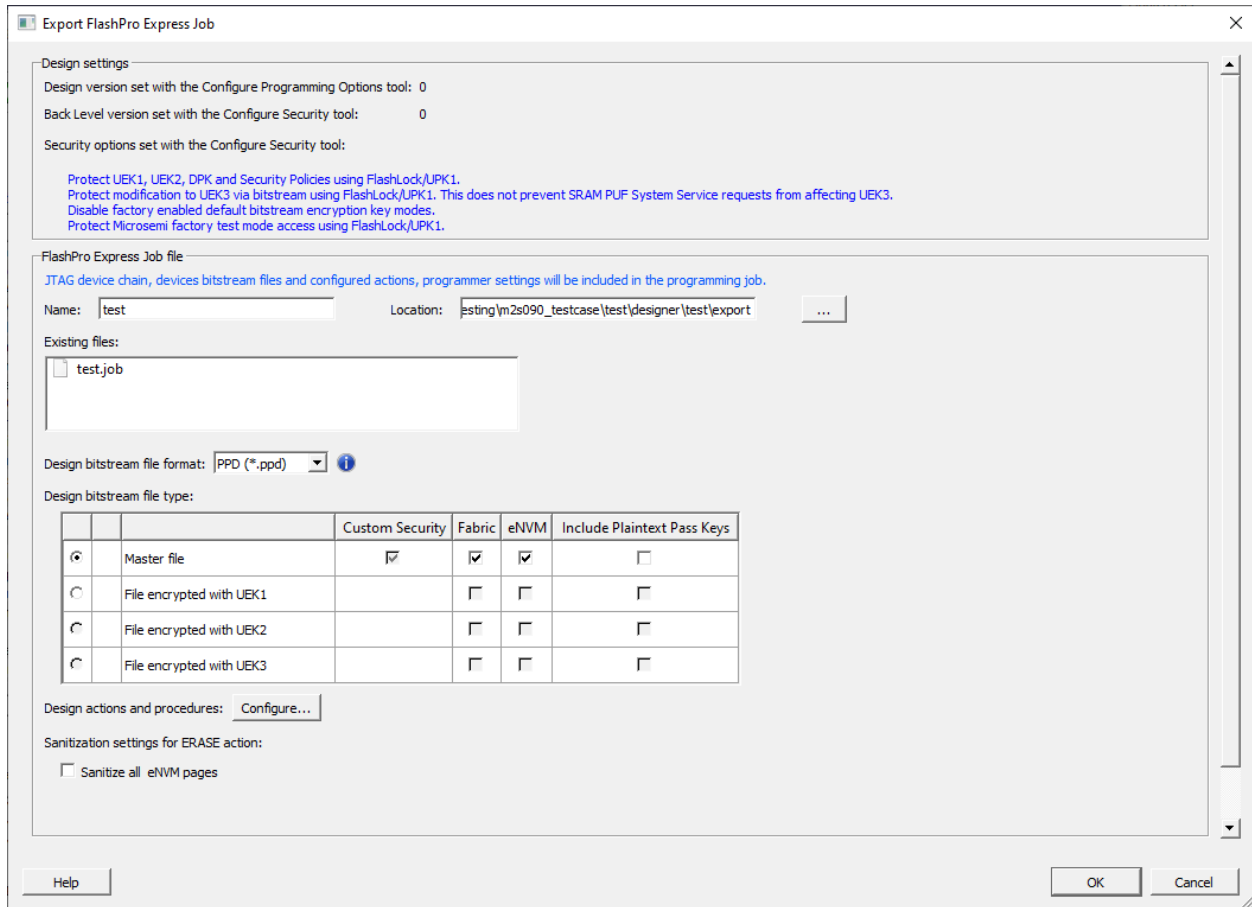


Figure 10-24. Export FlashPro Express Job Dialog Box with Custom Security Options

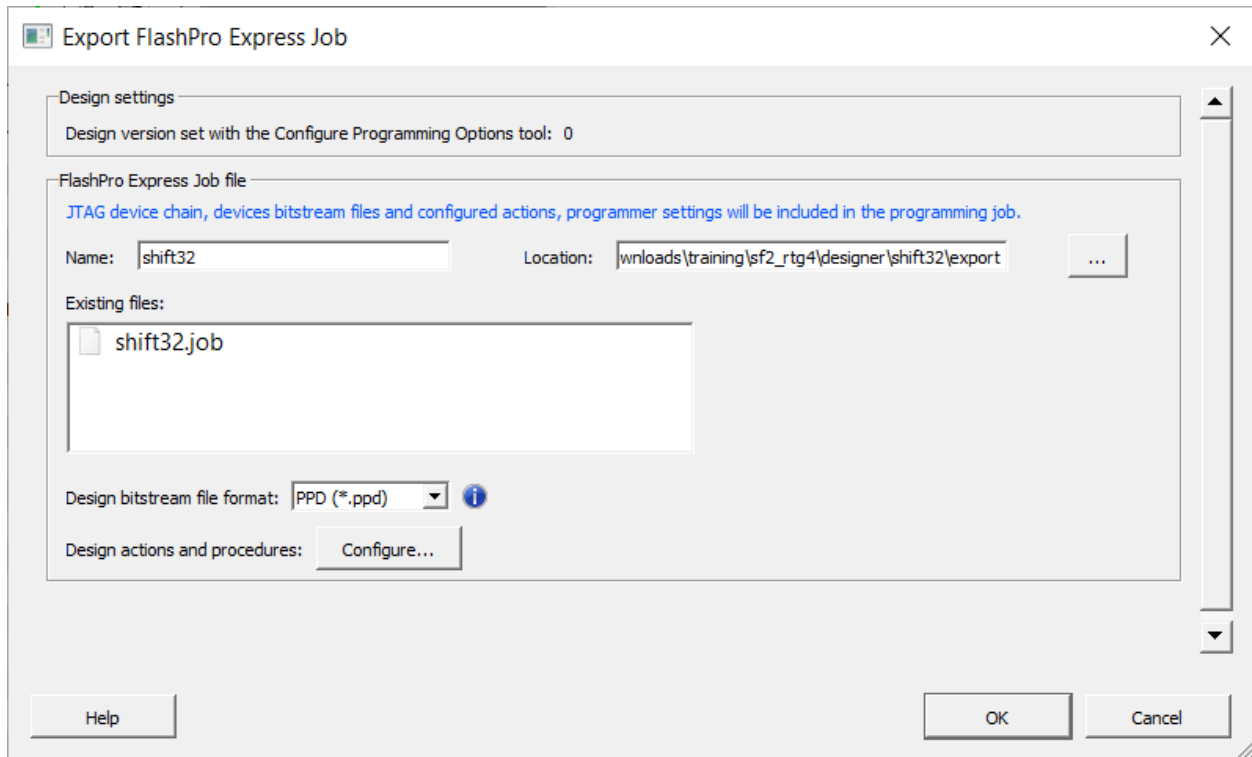


10.3.2.3. Export FlashPro Express Job (RTG4) [\(Ask a Question\)](#)

To program the design using stand-alone FlashPro Express tool on Linux or Windows, export a FlashPro Express Job. The job file will include chain configuration, Programmer Settings, Programming Mode (JTAG/SPI-Slave) and programming files loaded from Programming Connectivity and Interface.

Note: SPI Slave mode is not supported for RTG4 devices. JTAG is the default interface. Security Programming is not supported for RTG4.

Figure 10-25. Export FlashPro Express Job Dialog Box (RTG4)



If you selected the **One-time programmable (OTP)** option in Configure Programming Options, the following message appears. Click **Yes** to continue or **No** to cancel.



10.3.3. Export FlashPro Express Job Dialog Box Options [\(Ask a Question\)](#)

10.3.3.1. Design Settings [\(Ask a Question\)](#)

Read-only design settings that vary with the product family, such as design version, back level version, and security options.

10.3.3.2. FlashPro Express Job File [\(Ask a Question\)](#)

Name. Set the name of your job file. The default name is the design name.

Location. Location to save the exported file.

10.3.3.3. Existing files [\(Ask a Question\)](#)

Programming job files at the selected location.

10.3.3.4. Program (JTAG programming interface) [\(Ask a Question\)](#)

This option is available for PolarFire and PolarFire SoC only. Choices are:

- **Design.** Check to include the configured device chain with bitstream files.
- **SPI Flash.** Check to include the configured device chain with a SPI flash binary file.

JTAG is the default programming interface for PolarFire Devices. SPI Flash option is available only if SPI Flash is configured in the Configure Design Initialization Data and Memories tool; otherwise, it remains grayed out. Info message appears based on the configuration selected

10.3.3.5. Design Bitstream File Format [\(Ask a Question\)](#)

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 list. PPD is the default bitstream file format.

10.3.3.6. Design Bitstream File Type (PolarFire, PolarFire SoC, SmartFusion 2, and IGLOO 2) [\(Ask a Question\)](#)

Security-only programming must be performed on erased or new devices only. If performed on a device with Fabric programmed, the Fabric will be disabled after performing security-only programming, and you will have to reprogram the Fabric to re-enable it.

PolarFire, PolarFire SoC, SmartFusion 2, and IGLOO 2 with Default Security

File to Program at trusted facility. Available when the design is configured with Bitstream Encryption using the Default Key in the Configure Security tool. Options are:

- **Fabric sNVM-** Check to enable programming for Fabric/sNVM bitstream components at a trusted facility.
- **eNVM-** Check to enable programming for eNVM bitstream components at a trusted facility. This feature is not supported for PolarFire.

PolarFire, PolarFire SoC, SmartFusion 2, and IGLOO 2 with Custom Security

Master file- is used to program a trusted facility. Option is available when the design is configured with Custom Security options in the Configure Security tool. For PolarFire, the master file is also available when the design is configured with Permanent Locks in the Configure Permanent Locks for Production tool. Options are:

- **Custom Security-** Always programmed in the master file.
- **Fabric sNVM-** Check to enable programming for Fabric/sNVM bitstream components.
- **eNVM-** Check to enable programming for eNVM bitstream components. This feature is not supported for PolarFire.
- **Include Plaintext Pass Keys-** Check to add plaintext pass keys to the bitstream files to be programmed. This option is available if UPK1 is not permanently locked and the Update Policy Fabric /sNVM or eNVM in PolarFire SoC is passcode protected or any of program, authenticate, or verify action is locked.

File encrypted with UEK1- Used to program at untrusted facility or for Broadcast field update. Choices are:

- Fabric sNVM
- Include Plaintext Pass Keys

File encrypted with UEK2- Used to program at untrusted facility or for Broadcast field update. Choices are:

- Fabric sNVM
- Include Plaintext Pass Keys

File encrypted with UEK3- Used to program at untrusted facility or for Broadcast field update. Available for M2S060, M2GL060, M2S090, M2GL090, M2S150, and M2GL150 devices. Choices are:

- Fabric
- eNVM
- Include Plaintext Pass Keys

10.3.3.7. Design Actions and Procedures [\(Ask a Question\)](#)

Clicking the **Configure** button displays the options described in [Configuring Actions and Procedures](#).

10.3.3.8. Zeroization Actions (PolarFire and PolarFire SoC) [\(Ask a Question\)](#)

This option is available for PolarFire only. Choices are:

- **Like New**. Erase all user data and reprogram the device immediately.
- **Unrecoverable**. Erase all data and destroy reprogrammability. If selected, the device must be scrapped.

10.3.3.9. Sanitization Settings for ERASE Action (PolarFire and PolarFire SoC) [\(Ask a Question\)](#)

PolarFire and PolarFire SoC

- **Sanitize all sNVM pages**- Available if Fabric/sNVM component is selected for at least one master file or update file.
- **Sanitize all eNVM pages**- Available for PolarFire SoC only if eNVM is configured and selected for at least one master or update file.

SmartFusion 2 and IGLOO 2

- **Sanitize all eNVM pages**- Available only if eNVM is configured and selected for at least one master or update file.

10.4. Exporting Job Manager Data [\(Ask a Question\)](#)

Job Manager is Microchip's HSM-based security software for job management. It is supported by PolarFire SoC, PolarFire, SmartFusion 2, and IGLOO 2.

As a part of the SPPS flow, the Export Job Manager Data dialog box allows you 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.

Note: PolarFire SoC users can use Export Job Manager to export either Fabric with sNVM or sNVM component only. Job Manager data export is not supported by RTG4.

To export Job Manager data:

1. In the Design Flow window, expand **Handoff Design for Production**.
2. Double-click **Export Job Manager Data**.
3. Complete the fields in the Export Job Manager Data dialog box.
4. Click **OK**.

Figure 10-26. Export Job Manager Data Dialog Box (PolarFire)

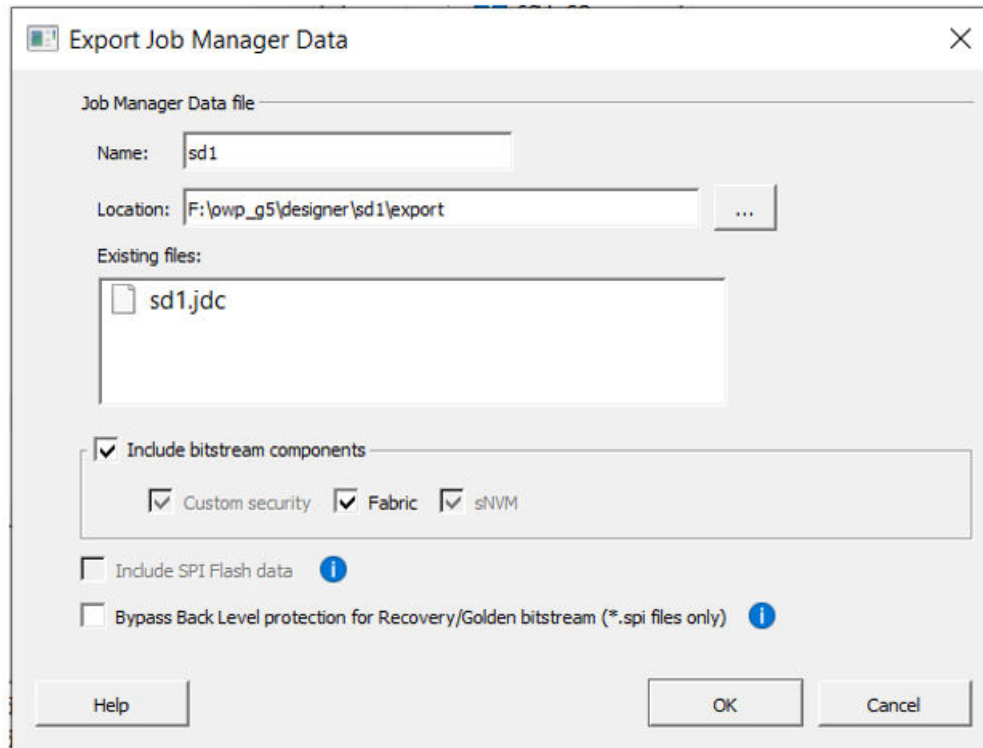


Table 10-6. Fields in the Export Job Manager Data Dialog Box

Field	Description
Name	All names use a prefix as shown in your software.
Location	Location of the file to be exported.
Existing programming data files	Existing programming job files already in your project.
Include bitstream components	Components of the design that can be saved to the file.
Include SPI Flash data (requires Fabric/sNVM components)	PolarFire only: If SPI Flash is configured in Configure Design Initialization Data and Memories, use this option to include Fabric/sNVM components.
Bypass Black Level protection for Recovery/Golden bitstream	Bypasses black-level protection for the SPI bitstream.

10.5. Export a SPI Flash Image (PolarFire and PolarFire SoC) [\(Ask a Question\)](#)

The Export SPI Flash Image tool allows PolarFire and PolarFire SoC users to export a SPI Flash Image file. The SPI Flash is defined in the [SPI Flash](#) tab in the Configure Design Initialization Data and Memories tool.

To export the SPI Flash Image:

1. In the **Design Flow** window, expand **Handoff Design for Production**.
2. Double-click **Export SPI Flash Image**.
3. Complete the fields in the **Export SPI Flash Image** dialog box.
4. Click **OK**.

Figure 10-27. Export SPI Flash Image

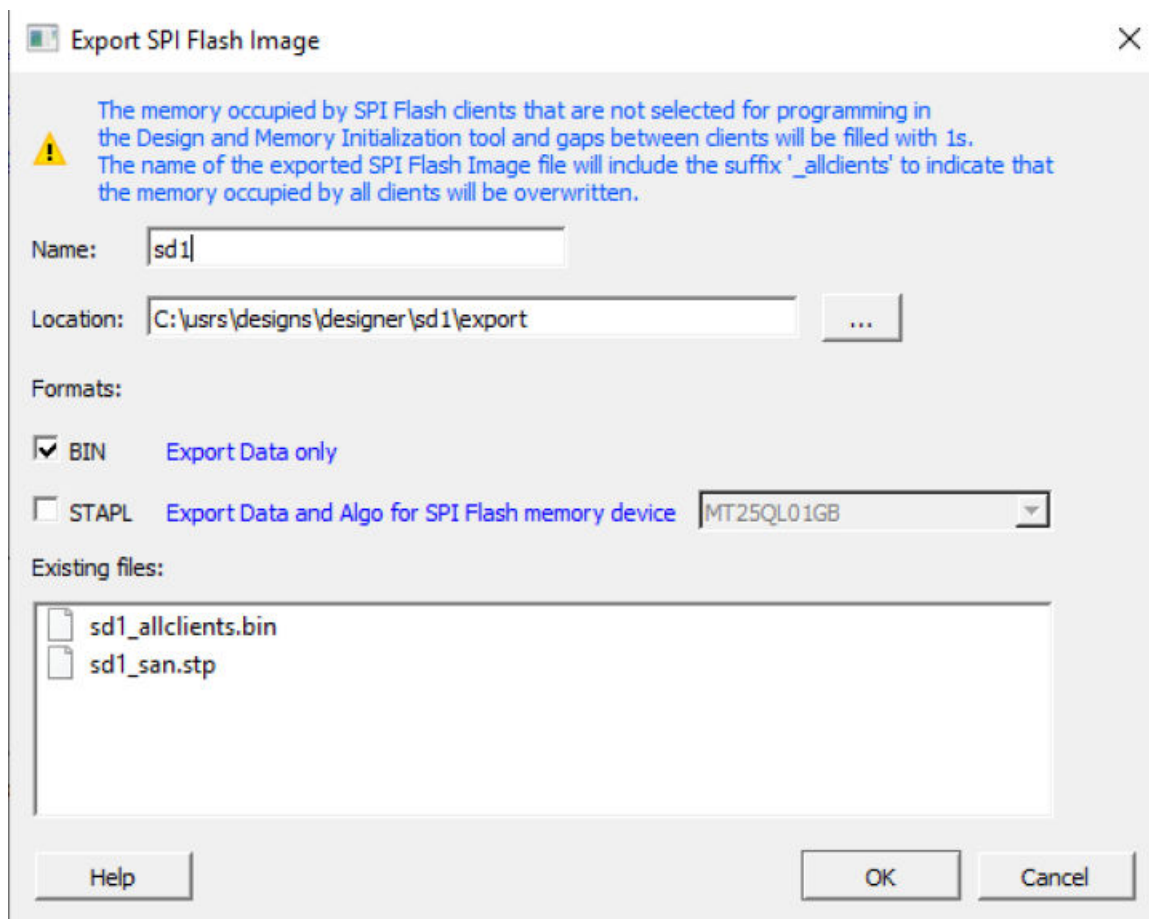


Table 10-7. Fields in the Export SPI Flash Image Dialog Box

Field	Description
Name	All names use a prefix as shown in your software.
Location	Location of the file to be exported.
Formats	Allows you to export the SPI Flash Image file in BIN format, STAPL format, or both formats.
Existing files	Existing files already in your project.

10.5.1. Name [\(Ask a Question\)](#)

This is the top level design name by default. Use this field to change the default name. By default, SPI Flash Image files are exported in binary format, acquire the *.bin file extension, and are named <design_name>_allclients.bin.

10.5.2. Location [\(Ask a Question\)](#)

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.

10.5.3. Formats [\(Ask a Question\)](#)

Select the bitstream file format you want to export. The following file format options are available:

- BIN

- STAPL

BIN files acquire a .bin file extension is added to the end of a file name and denotes that the content of the file is in binary format.

STAPL files are used for programming and testing Programmable Logic Devices (PLDs) and FPGAs through JTAG, and are in Standard Test and Programming Language (STAPL) format. If you select STAPL, select a SPI Flash memory device from the dropdown menu. When the STAPL file is generated, it acquires a STP file extension.

10.5.4. Existing files [\(Ask a Question\)](#)

Lists existing SPI Flash Image files.

10.6. Exporting Pin Reports [\(Ask a Question\)](#)

The pin report lists the pins in your device. When you export the report, you can specify whether you want the pins to be sorted by port name, package pin name, or both. Based on your selection, the pin report generates one or two files:

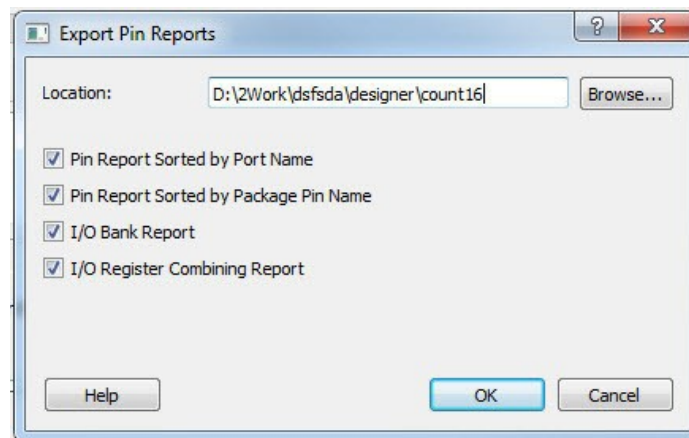
- <design>_pinrpt_name.rpt. Pin report sorted by name.
- <design>_pinrpt_number.rpt. Pin report sorted by pin number.

By default, exporting a pin report generates a Bank report with the filename <design>-bankrpt.rpt. Export Pin Report also generates an I/O Register Combining report that lists the I/Os that are combined into a Register to improve timing performance.

To export the pin report:

1. In the Design Flow window, expand **Handoff Design for Production**.
2. Double-click **Export Pin Report**.

Figure 10-28. Export Pin Reports Dialog Box



3. Click **Browse** and go to the location where you want the pin report to be saved.
4. Check or uncheck the check boxes to suit your requirements.
5. Click **OK**.

10.7. Exporting BSDL Files [\(Ask a Question\)](#)

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.

To generate the BSDL file to your [Design Report](#):

1. In the Design Flow window, expand **Handoff Design for Production**.

2. Double-click **Export BSDL File**.
3. When prompted, go to the location where you want the BSDL file to be exported.
4. Click **Save**.

For more information about BSDL models, see the [Microchip website](#).

10.8. Exporting IBIS Models [\(Ask a Question\)](#)

Export Input Output Buffer Information Specification (IBIS) is supported for all PolarFire “T” devices, except the “XT” device. The Export IBIS feature is supported in two stages:

- Pre-Layout Flow
- Post-Layout Flow

To export the IBIS Model in the Pre-Layout Flow, the design must pass until the Synthesis/Compile stage. You must 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 must pass until the Place and Route stage. If a design runs through the Place and Route tool, the IBIS model for the Post-Layout Flow will be exported by default.

To export the IBIS Model:

1. In the Design Flow window, expand **Handoff Design for Production**.
2. Double-click **Export IBS Model**.
3. Complete the fields in the Export IBS Model dialog box.
4. Click **OK**.

Figure 10-29. Export IBIS Model Dialog Box

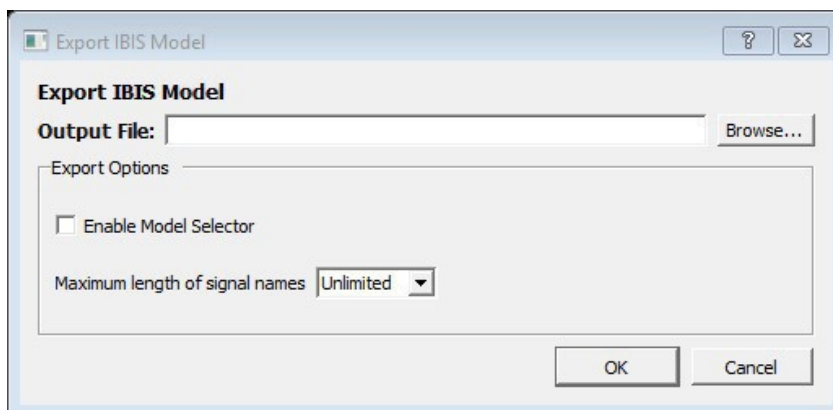


Table 10-8. Fields in the Export IBS Model Dialog Box

Field	Description
Output File	Click Browse and go to the location where you want to export the IBS Model.
Enable Model Selector	To generate Model Selector support in the exported IBIS file, check this box.
Maximum length of signal names	Length of Model Selector name or Model name referred to as the Signal name in the IBIS file.

10.8.1. Output File [\(Ask a Question\)](#)

Click **Browse** to specify the location to export the IBIS output file along with the output file name.

10.8.2. Export Options [\(Ask a Question\)](#)

The IBIS report *.ibs file exported from Libero SoC can optionally support the [Model Selector] keyword specified in the IBIS 5.0 Specifications. To generate Model Selector support in the exported IBIS file, check **Enable Model Selector** in the Export Options and click **OK** to export the IBIS Model. In this way, you can also optionally limit the maximum signal name to 40 characters.

The IBIS model report provides an industry-standard file format for recording parameters such as driver output impedance, rise/fall time, and input loading, which can 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 appears in the **Files** tab.

In the **[Pin]** section of the IBIS *.ibs file, the Model Selector tag appears below the **model_name**. 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 10-30. 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

[Model Selector]	LVCN0515_Bank15Group1_asiod	
LVCN0515_6mA_MSIO	LVCN0515_6mA_MSIO	
LVCN0515_2mA_MSIO	LVCN0515_2mA_MSIO	
LVCN0515_4mA_MSIO	LVCN0515_4mA_MSIO	

Model Selector for Pin AE33

[Model Selector]	LVCN0525_Bank1Group1_ddrio	
LVCN0525_2mA_MEDFAST_DDRIO	LVCN0525_2mA_MEDFAST_DDRIO	
LVCN0525_2mA_SLOW_DDRIO	LVCN0525_2mA_SLOW_DDRIO	
LVCN0525_2mA_MED_DDRIO	LVCN0525_2mA_MED_DDRIO	
LVCN0525_2mA_FAST_DDRIO	LVCN0525_2mA_FAST_DDRIO	
LVCN0525_4mA_MEDFAST_DDRIO	LVCN0525_4mA_MEDFAST_DDRIO	
LVCN0525_4mA_SLOW_DDRIO	LVCN0525_4mA_SLOW_DDRIO	
LVCN0525_4mA_MED_DDRIO	LVCN0525_4mA_MED_DDRIO	
LVCN0525_4mA_FAST_DDRIO	LVCN0525_4mA_FAST_DDRIO	

Model Selector for Pin E31

[Model Selector]	LVCN0525_Bank2Group1_ddrio	
LVCN0525_4mA_SLOW_DDRIO	LVCN0525_4mA_SLOW_DDRIO	
LVCN0525_4mA_MED_DDRIO	LVCN0525_4mA_MED_DDRIO	
LVCN0525_4mA_MEDFAST_DDRIO	LVCN0525_4mA_MEDFAST_DDRIO	
LVCN0525_4mA_FAST_DDRIO	LVCN0525_4mA_FAST_DDRIO	
LVCN0525_2mA_SLOW_DDRIO	LVCN0525_2mA_SLOW_DDRIO	
LVCN0525_2mA_MED_DDRIO	LVCN0525_2mA_MED_DDRIO	
LVCN0525_2mA_MEDFAST_DDRIO	LVCN0525_2mA_MEDFAST_DDRIO	
LVCN0525_2mA_FAST_DDRIO	LVCN0525_2mA_FAST_DDRIO	

Model Selector for Pin A2

The Model Selector feature allows you to 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.

For more information, navigate to the [Microchip website](#), and then use the search function to search for IBIS models.

10.9. Export Initialization Data and Memory Report (PolarFire and PolarFire SoC) [\(Ask a Question\)](#)

The Export Initialization Data and Memory report provides standard initialization data and memory report based on the configuration performed in the Configure Design Initialization and Memories tool.

To generate the Export Initialization Data and Memory report:

1. In the Design Flow window, expand **Handoff Design for Production**.

2. Double-click **Export Design Initialization Data and Memory Report**.
3. When prompted, go to the location where you want the initialization data and memory report to be exported.
4. Click **Select Folder**.

10.10. Exporting μ PROM Reports (RTG4) [\(Ask a Question\)](#)

RTG4 users can generate an Export μ PROM report that includes the configuration file path, client details, and μ PROM usage statistics.

To export a μ PROM report:

1. In the Design Flow window, expand **Handoff Design for Production**.
2. Double-click **Export μ PROM Report** to export the report. The Export μ PROM Report Under dialog box appears.
3. When prompted, go to the location where you want to save the Export μ PROM report.
4. Click **Select Folder**.

The Export μ PROM report generates an XML/TEXT version of the report with the filename. For example: `uPROM_Configuration_Report.xml/ uPROM_Configuration_Report.txt`

11. Handoff Design for Firmware Development (SmartFusion 2 and IGLOO 2) [\(Ask a Question\)](#)

The following sections apply to SmartFusion 2 and IGLOO 2 devices only.

11.1. Software IDE Integration (SmartFusion 2 and IGLOO 2) [\(Ask a Question\)](#)

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 any 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 workspaces for your selected development environment are generated automatically with the proper settings and flags so you can write your application immediately.

11.2. Viewing/Configuring Firmware Cores (SmartFusion 2 and IGLOO 2) [\(Ask a Question\)](#)

The **Design Firmware** tab allows you to select and configure firmware cores (drivers) for your Software IDE project. The tab lists the compatible firmware for the hardware you instantiated in your design.

To display the **Design Firmware** tab in the Design Flow tab, expand **Create Design** and double-click **View/Configure Firmware Cores**.

The **Firmware** table lists the compatible firmware and drivers based on the hardware peripherals used in your design. Each row represents a firmware core that is compatible with a hardware peripheral in your design. The following table describes the columns in the **Firmware** table.

Table 11-1. Columns in the Firmware Table

Column	Description
Generate	Choose whether you want the files for this firmware core to be generated on disk and added to your Software IDE project. Click the checkbox to generate firmware for each peripheral in your design.
Instance Name	Name of the firmware instance. This can help to identify firmware cores when you have multiple firmware cores with the same Vendor:Library:Name:Version (VLNV) in your design.
Core Type	Name from the VLNVD ID of the core. This name generally corresponds to the name of the hardware peripheral with which the firmware core is compatible.
Version	Firmware core version. Use the drop-down menu to upgrade or choose a different version.
Compatible Hardware Instance	Hardware instance in your design that is compatible with this firmware core.

11.2.1. Downloading Firmware [\(Ask a Question\)](#)

Libero attempts to find compatible firmware located in the IP Vault located on your disk, as well as firmware in the IP Repository via the Internet.

If compatible firmware is found in the IP repository but not on your disk, the row will be italicized, indicating that it needs to be downloaded. To download all firmware cores necessary for your project peripherals, click the **Download All Firmware** icon in the vertical toolbar.

Libero IDE Fails to Connect to Default Repositories over the Internet

If Libero IDE fails to connect to the web repositories on a computer connected to the Internet, it typically means that the connection is being blocked by a network firewall or that the Internet connection is made via a proxy server. Verify that the URLs shown in your catalog options window are correct: **Libero > Catalog > Options > Vault Repositories Settings/Repositories**.

If your Internet gateway uses a proxy server:

1. From the Libero IDE toolbar, go to **Project > Preferences** and click the **Proxy** tab.
2. Check the **Use a Proxy Server** check box.
3. Enter the Proxy Server name on the network. In most cases, using the proxy server name solves the problem. If the server name does not work, enter the IP address of the proxy server.

11.2.2. Configuring Firmware [\(Ask a Question\)](#)

Firmware cores that have configurable options will have a wrench icon in the row. Click the wrench icon to configure the firmware core.

It is important that you check the configuration of your firmware cores if they have configurable options. They might have options that target your software IDE (Keil, IAR, or SoftConsole), or your processor, that are vital configuration options for the system to work properly.

11.2.3. Generating Firmware [\(Ask a Question\)](#)

Click the UI control icon to export the firmware drivers and software IDE project for your project. The firmware drivers are generated into <project>\firmware and the software workspace is exported to <project>\<toolchain>.

<toolchain> could be SoftConsole, IAR, or Keil, depending on your software IDE. The firmware drivers are also copied into the <toolchain> folder.

11.2.4. Changing Firmware Core Versions [\(Ask a Question\)](#)

You can manually change to the latest version by selecting the drop-down in the Version column.

There will often be multiple versions of a firmware cores available for a particular peripheral. The MSS Configurator selects the latest compatible version for a new design.

However, once the firmware is added to your design, Libero will not automatically change to the latest version if one becomes available.

Note: If a core version is shown in italics, it is available in the Web Repository but not in your Vault. You must download the firmware core version to use it in your design.

11.2.5. Generating Sample Projects [\(Ask a Question\)](#)

Firmware cores are packaged with sample projects that demonstrate their usage. They are packaged for specific tool chains, such as Keil, IAR, and SoftConsole

To generate a sample project,

1. Right-click the firmware core
2. Choose **Generate Sample Project**
3. Select your IDE tool chain (such as Keil) and choose from the list of available samples.

You will be prompted to select the destination folder for the sample project.

Once this project is generated you can use it as a starting point in your Software IDE tool or use the example project as a reference on how to use the firmware driver.

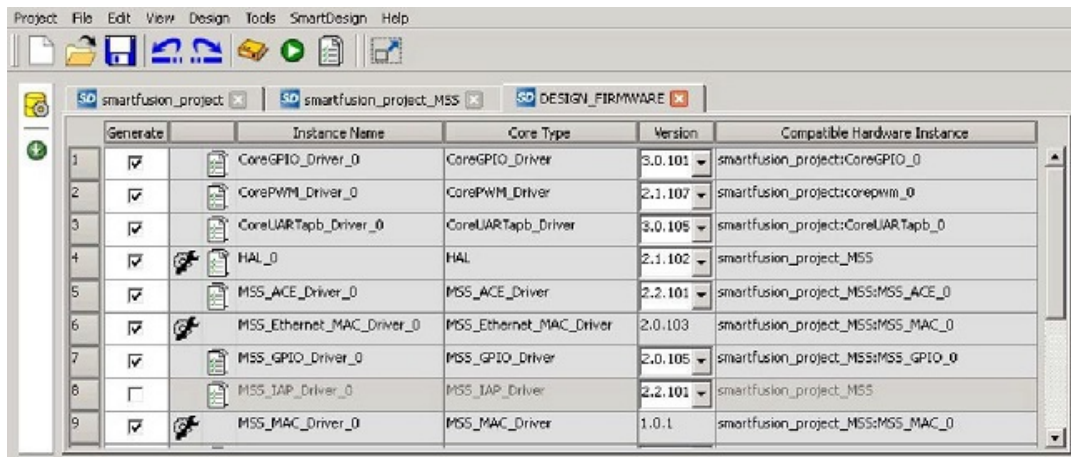
11.2.6. Fabric Peripherals [\(Ask a Question\)](#)

Libero SoC also attempts to find compatible firmware for soft (fabric) peripherals that you have added in your top-level SmartDesign if that top-level is Set as Root.

To set your top-level design as a root, right-click your top-level design in the Design Hierarchy and choose **Set as Root**. The root component appears in bold.

The following figure shows CoreGPIO, CorePWM, and CoreUARTapb soft cores that are added into your top-level SmartDesign.

Figure 11-1. Firmware Cores Tab (DESIGN_FIRMWARE)



	Generate	Instance Name	Core Type	Version	Compatible Hardware Instance
1	☒	CoreGPIO_Driver_0	CoreGPIO_Driver	3.0.101	smartfusion_projects:CoreGPIO_0
2	☒	CorePWM_Driver_0	CorePWM_Driver	2.1.107	smartfusion_projects:corepwm_0
3	☒	CoreUARTapb_Driver_0	CoreUARTapb_Driver	3.0.105	smartfusion_projects:CoreUARTapb_0
4	☒	HAL_0	HAL	2.1.102	smartfusion_project_MSS
5	☒	MSS_ACE_Driver_0	MSS_ACE_Driver	2.2.101	smartfusion_project_MSS:MSS_ACE_0
6	☒	MSS_Ethernet_MAC_Driver_0	MSS_Ethernet_MAC_Driver	2.0.103	smartfusion_project_MSS:MSS_MAC_0
7	☒	MSS_GPIO_Driver_0	MSS_GPIO_Driver	2.0.105	smartfusion_project_MSS:MSS_GPIO_0
8	☐	MSS_IAP_Driver_0	MSS_IAP_Driver	2.2.101	smartfusion_project_MSS
9	☒	MSS_MAC_Driver_0	MSS_MAC_Driver	1.0.1	smartfusion_project_MSS:MSS_MAC_0

11.3. Exporting Firmware (SmartFusion 2 and IGLOO 2) [\(Ask a Question\)](#)

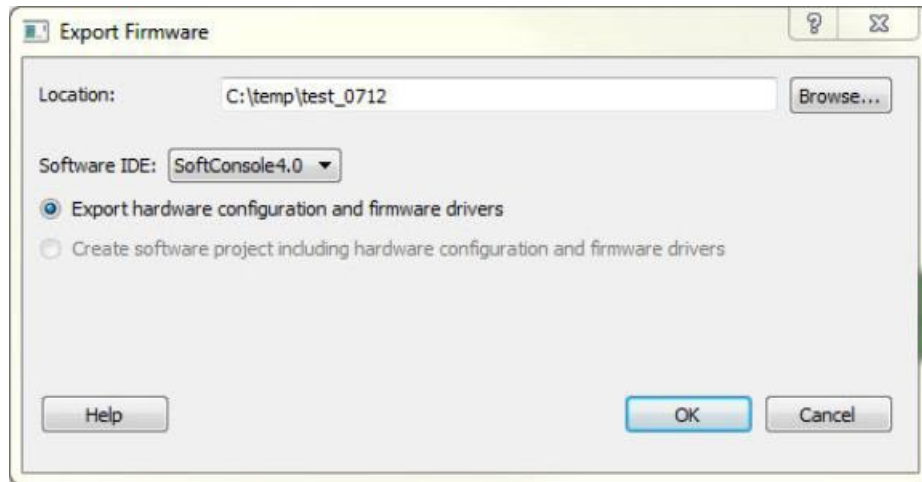
When your design is completed, you can export the design firmware configuration using the Export Firmware tool. The firmware configuration contains:

- Register configuration files for MSS, FDDR, and SERDES blocks instantiated in your design. This information, along with the SmartFusion 2 CMSIS firmware core, must be compiled with your application to have proper Peripheral Initialization when the Cortex-M3 boots.
- Firmware drivers compatible with the hard and soft peripherals instantiated in your design.

Note: If you make any changes to your design, you must re-export firmware.

To export your design firmware configuration:

1. In the Libero SoC Design Flow window, under **Handoff Design for Firmware Development**, double-click **Export Firmware**. The Export Firmware dialog box appears.
2. Complete the fields in the dialog box.
3. Click **OK**.

Figure 11-2. Export Firmware Dialog Box**Table 11-2.** Fields in the Export Firmware Dialog Box

Field	Description
Location	Location where you want the firmware configuration files to be exported. When you export the firmware, Libero SoC creates a Firmware folder to store all the drivers and register configuration files.
Software IDE: <selected Software Tool Chain>	Libero SoC creates the firmware project for the IDE tool of your choice and stores the projects in the folder <code>SoftConsole/IAR/Keil</code> (per your choice).
Export hardware configuration and firmware drivers	This option is selected by default. This setting exports register configuration files for MSS, FDDR, and SERDES blocks instantiated in your design. CMSIS and other firmware drivers must be generated using the stand-alone Firmware Catalog executable. These options are available to support SoftConsole 4.0 flow.
Create software project including hardware configuration and firmware drivers	To enable you to manage your firmware project separately from Libero's automatically generated firmware data, the created software workspace contains two software projects: <code>hardware_platform</code> contains all the firmware and hardware abstraction layers that correspond to your hardware design. This project is configured as a library and is referenced by your application project. The content of this folder is overwritten each time you export your firmware project. <code>application</code> produces a program and results in the binary file. It links with the <code>hardware_platform</code> project. This folder does not get overwritten when you re-export your firmware. This is where you can write your own <code>main.c</code> and other application code, as well as add other user drivers and files. You can reference header (<code>*.h</code>) files of any hardware peripherals in the <code>hardware_platform</code> project – include paths are automatically set up for you. To build your workspace, have the <code>hardware_platform</code> and <code>application</code> projects set to the same compile target (Release or Debug) and build both projects. To open your exported firmware projects, start your third-party development tool (SoftConsole, Keil, or IAR) outside Libero SoC and point it to the exported firmware workspace.

11.3.1. TCL Command [\(Ask a Question\)](#)

```
export_firmware \  
-export_dir {D:\Designs\software_drivers} \  
-create_project 1 \  
-software_ide {Keil}
```

11.3.2. Version Supported [\(Ask a Question\)](#)

Libero SoC v11.7 and later supports the following versions of third-party development tools:

- SoftConsole v4.0
- SoftConsole v3.4
- IAR EWARM
- Keil

12. Handoff Design for Debugging [\(Ask a Question\)](#)

12.1. Exporting SmartDebug Data [\(Ask a Question\)](#)

The following procedure describes how to export SmartDebug Data from Libero to the stand-alone SmartDebug environment. You can export SmartDebug data to any location that has read/write permission without connecting the hardware.

1. In the Design Flow window, expand **Handoff Design for Production**.
2. Expand **Handoff Design for Debugging**, right-click **Export SmartDebug Data**, and click **Export**. The Export SmartDebug Data dialog box appears.
3. Select the design debug data (* .ddc) file you want to export.

You can use this file to create a stand-alone SmartDebug project (see the following figure for an example).

Figure 12-1. Export SmartDebug Data Dialog Box

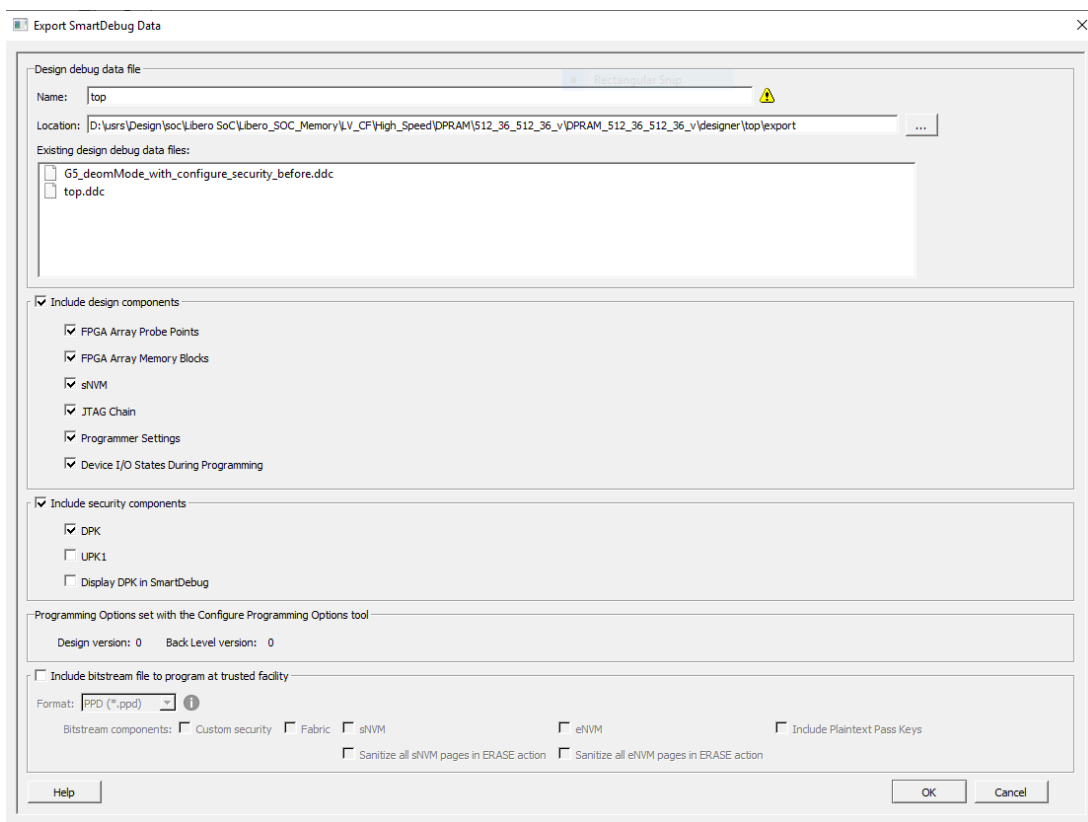


Table 12-1. Fields in the Export SmartDebug Data Dialog Box

Field	Description
Name	Name of the design.
Location	Location of the exported debug file. By default, the *.ddc file has the *.ddc file extension and exports to the <project_location>/designer/<design>/export folder.

Table 12-1. Fields in the Export SmartDebug Data Dialog Box (continued)

Field	Description
Existing design debug data files	You can export *.ddc files in the export folder after running Generate FPGA Array Data for the design in the Libero® Design Flow. You can also export SmartDebug data directly after running Synthesize on the design. Other tools, such as Place and Route and Generate FPGA Array Data) are run implicitly before the Export SmartDebug Data dialog box is displayed.
Include design components	A *.ddc file can contain the following components: - FPGA Array Probe Points: Check to export Live and Active probes information (<design>_probe.db file) into the *.ddb container file. - FPGA Array Memory Blocks: Check to export the following 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 you selected in the design. - sNVM: Appears if sNVM is in your design. Check to export sNVM components. - Security: Contains the security locks, keys, and security policy information for debugging. This might be default or custom security (<design>.spm file). This is hidden if security is not supported for the device. - Display DPK or UPK1 in SmartDebug: Enabled when custom security is provided. Default: not checked - JTAG Chain: Device chain information configured using Programming Connectivity and Interface in Libero. Check to export chain data, including devices, their programming files if loaded, and device properties, to the <design>.pro file. If not checked, the default JTAG chain with Libero design device is added only to the *.ddc file. - Programmer Settings: If not checked, the default programmer settings are added to the *.ddc file. - Device I/O States During Programming (<design>.ios file): Used by some SmartDebug features for programming sNVM. It is not used during device programming in SmartDebug; programming files used to program devices already have I/O states data.

Table 12-1. Fields in the Export SmartDebug Data Dialog Box (continued)

Field	Description
Include security components	Choices are: - DPK (Debug Pass Key): If security is configured and DPK is set, this option is selected by default. - UPK1 (User Pass Key 1): If security is configured and DPK is set, this option is selected by default. If DPK is not set, this option is selected by default. - Display DPK In SmartDebug/Display UPK1 In SmartDebug: This option is not selected by default. Select this option to display DPK or UPK1 in the Standalone SmartDebug tool. If DPK is selected, DPK is passed to (*.ddc) for unlocking the debug features. Similarly, if UPK1 is selected, UPK1 is passed to (*.ddc) for unlocking the debug features. If both DPK and UPK1 are selected, both are passed to (*.ddc), but only DPK is used for unlocking the debug features.  Attention: If DPK is permanently disabled for PolarFire®, UPK1 is used for unlocking the device.
Programming Options set with the Configure Programming Options tool	Read-only values that show the design version and back level version set in the Configure Programming Options Wizard. From these values, you can use the bitstream file information for programming the device in Standalone SmartDebug. Note: Info and warning messages appear based on the value set for the back level version.

Table 12-1. Fields in the Export SmartDebug Data Dialog Box (continued)

Field	Description
Include bitstream file to program at trusted facility	Choices are: • Format: Select the bitstream file format you want to export. Choices are PPD and STAPL. • Bitstream components: Choices are: – Custom security: Appears if Custom Security is selected in the Configure Security Wizard. – Fabric/sNVM: 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 you to unselect. Otherwise, the sNVM component can be enabled or disabled for programming. If selected, Custom Security is enabled by default and grayed-out. – eNVM: Supported for PolarFire only. – Include Plaintext Pass Keys: By default, all exported bitstream files exclude plaintext pass keys. You must explicitly select to include them if desired. Use bitstream files that include plaintext pass keys only in trusted environments. FlashLock/Pass keys are needed to unlock any locked features. If selected, Custom Security is enabled by default and grayed-out. If a user design has security configured, this option is required to include the plaintext passkeys along with programming bitstream file for REPROGRAM/ERASE actions in SASD. – Sanitize all sNVM pages in ERASE action. Available if Fabric/sNVM component is selected for at least one master file or update file. Supported for PolarFire only. Sanitize all eNVM pages in ERASE action: Available if eNVM is configured and selected for at least one master or update file. Supported for PolarFire SoC only.

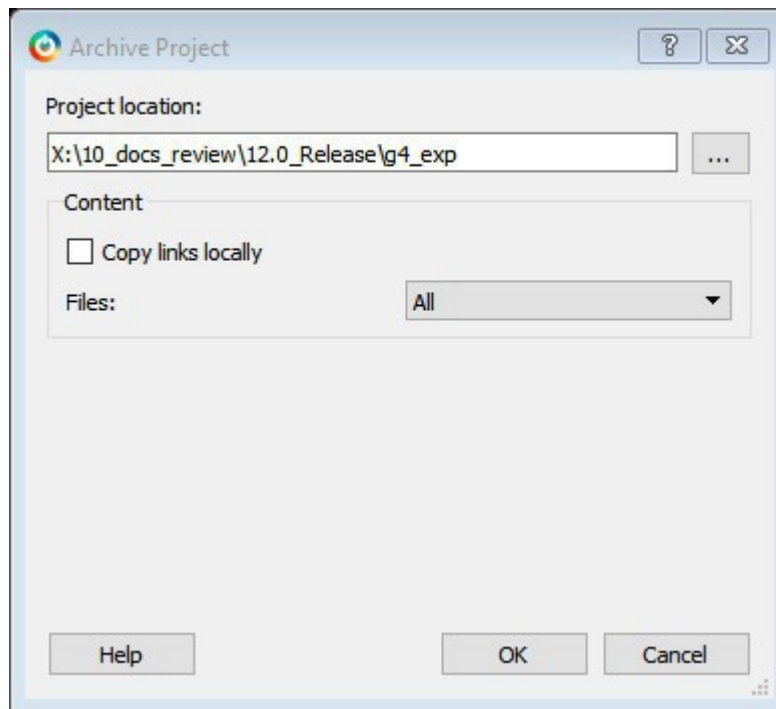
13. References [\(Ask a Question\)](#)

13.1. Archive Project Dialog Box [\(Ask a Question\)](#)

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. **Archive Project** and **Save As Project** are functionally equivalent; however, 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, timestamps might 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 13-1. Archive Project Dialog Box



Project Location. Accept the default location or browse to a new location where you want to save the archive (*.zip) file. The *.zip file is named <project_name>.zip.

Copy links locally. Check this check box to copy the links from your current project into your archive. Otherwise, the links will not be copied and you must add them manually.

Files. Specifies the kind of files that 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 toolchain 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 for different selections as shown in the following table.

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.

13.2. Adding or Modifying Bus Interfaces in SmartDesign [\(Ask a Question\)](#)

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):

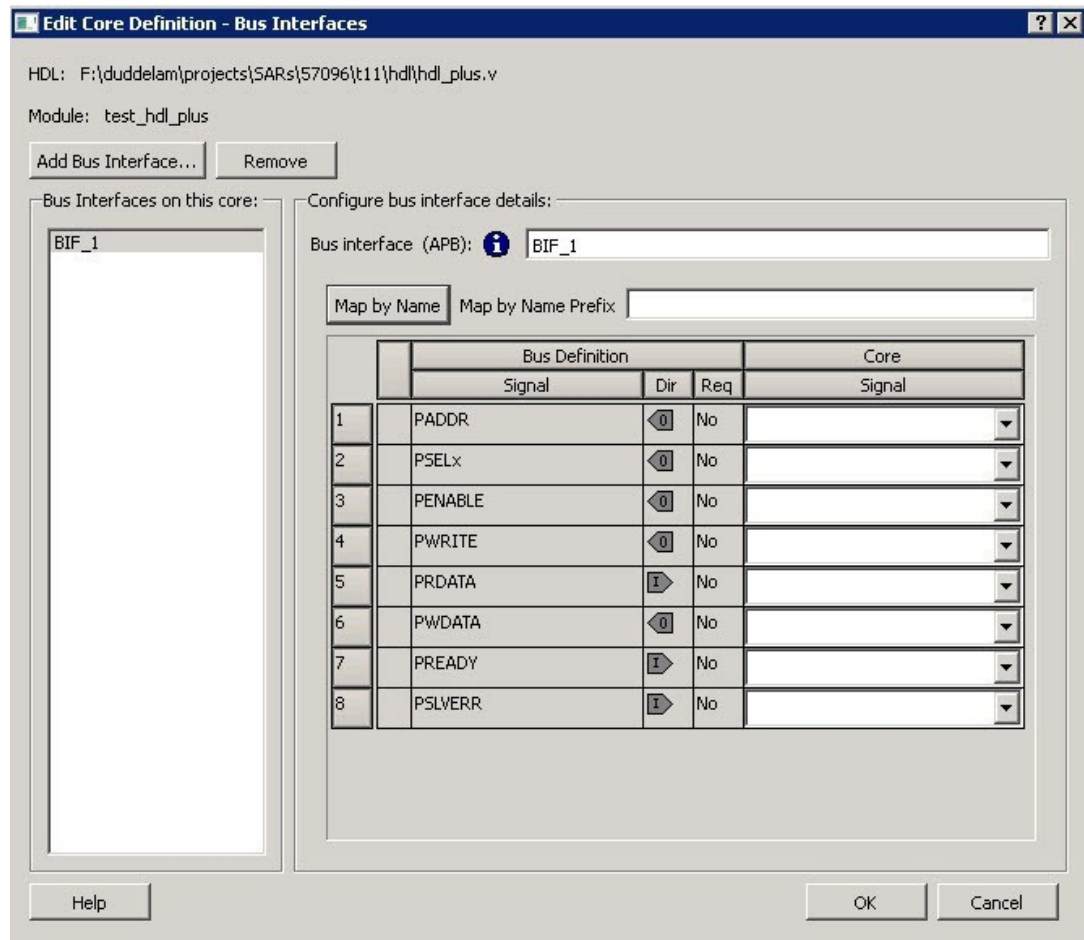
1. Right-click your custom HDL block and choose **Create Core from HDL**. The Libero SoC creates your core and asks if you want to add bus interfaces.
2. Click **Yes** to open the Edit Core Definition dialog box and add bus interfaces. Add the bus interfaces, as necessary.
3. Click **OK** to continue.

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 following figure.

Figure 13-2. Edit Core Definition Dialog Box



2. Click **Add Bus Interface**. Select the bus interface you want 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. It can be edited as required. **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.

13.2.1. Bus Interfaces [\(Ask a Question\)](#)

When you add a bus interface, the Edit Core Definition dialog box provides the following Microchip Libero SoC-specific bus interfaces:

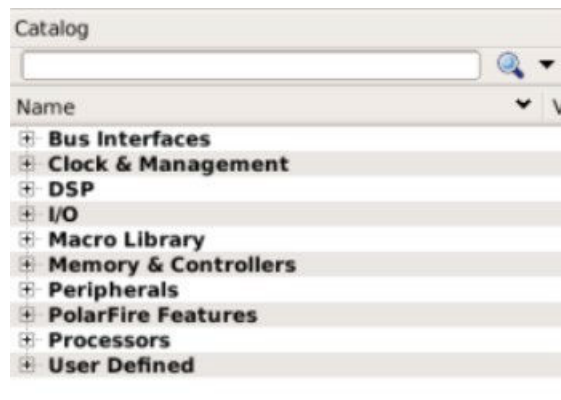
- AHB – Master, Slave, MirroredMaster, MirroredSlave
- APB – Master, Slave, MirroredMaster, MirroredSlave
- AXI – Master, Slave, MirroredMaster, MirrorSlave, System
- AXI 4 – Master, Slave, MirroredMaster, MirrorSlave

13.3. Catalog [\(Ask a Question\)](#)

In the Libero SoC, from the **View** menu choose **Windows > Catalog**.

The Catalog displays a list of available cores, buses, and macros.

Figure 13-3. 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, and so on.) to your SmartDesign component.

1. 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.
2. Check the Simulation Mode check box 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:

- Plaintext - 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 descriptions.

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.

1. Right-click any item in the Catalog and choose Show Details for a short summary of the core specifications.
2. Choose Open Documentation for more information on the Core.
3. Right-click and choose Configure Core to open the core generator.
4. Click the **Name** column heading to sort the cores alphabetically.
5. Filter the cores according to the data in the Name and Description fields.
6. Type the data into the filter field to view the cores that match the filter.
7. Set the Display setting in the Catalog Options to **List cores alphabetically** while 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 UI control button  (or the drop-down arrow next to it) to import a core, reload the Catalog, or modify the Catalog Options.

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 completed and not posted to the web (you have an evaluation core)

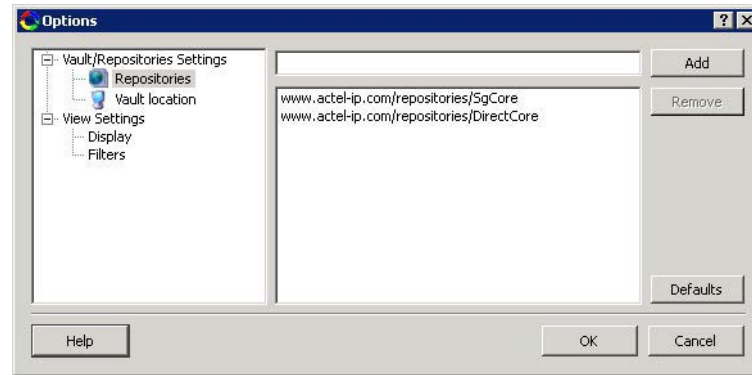
Manually Downloading MegaVaults and Individual CPZ files

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](#).

13.3.1. Catalog Options Dialog Box [\(Ask a Question\)](#)

The following Catalog Options dialog box allows 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 13-4. 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:

1. Add a new environment variable `MSCC_IDE_VAULT_EXTRACT_LOCATION` in System Properties and specify your extract location.
 2. 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.

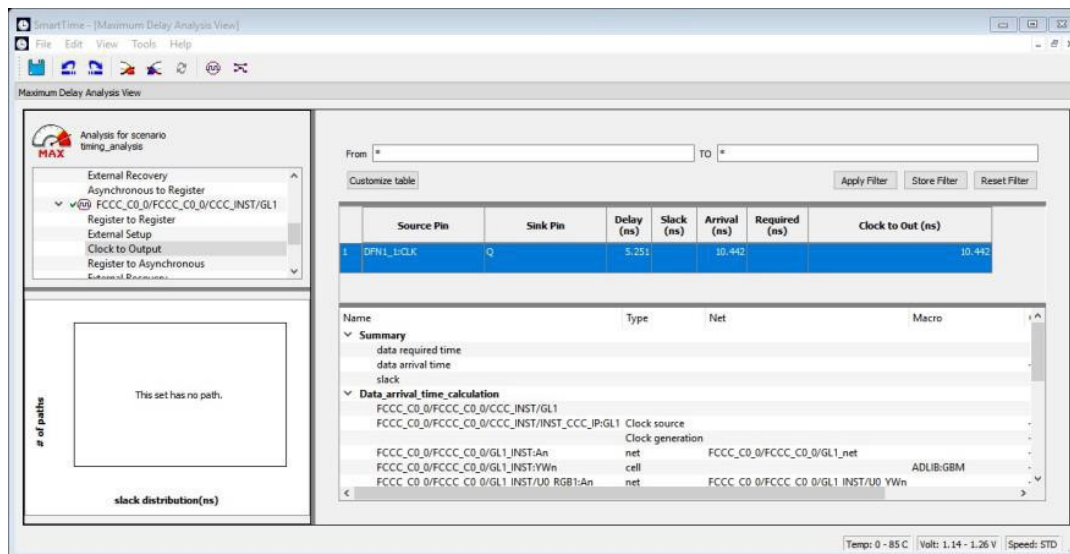
Show remote cores that are not in my vault - Displays remote cores that have not been added to your vault in your Catalog.

13.4. Changing Output Port Capacitance [\(Ask a Question\)](#)

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 following figure 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.

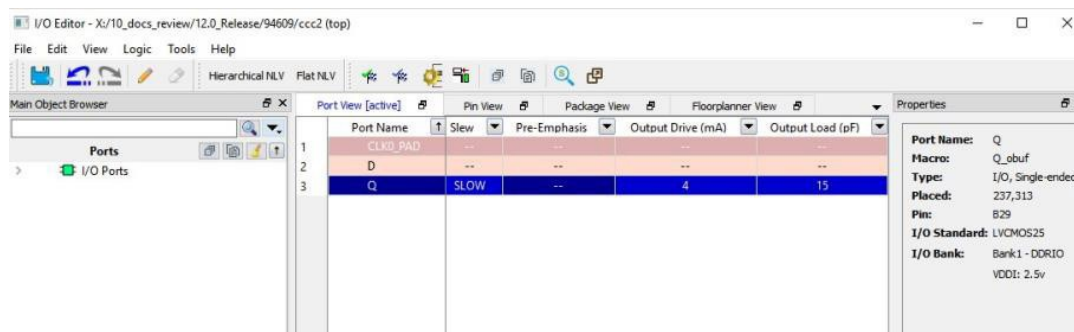
Figure 13-5. Maximum Delay Analysis View



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 13-6. 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.

13.5. Core Manager [\(Ask a Question\)](#)

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 might 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 13-1. 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.

13.6. Configure Security Locks for Production [\(Ask a Question\)](#)

Configure Security Locks for Production is a GUI-based tool that guides the user 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 are programmed.

1. [Permanent Locks and One-way Passcode](#)
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.

13.7. Importing Source Files by Copying Files Locally [\(Ask a Question\)](#)

Designer in Libero SoC cannot import files from outside your project without copying them to your local project folder. You might 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 directories: netlists are copied to the *synthesis* folder; source files are copied to *hdl* folder, constraint files to *constraint* folder, and so on. The files are also added to the Libero SoC project and appear in the **Files** tab.

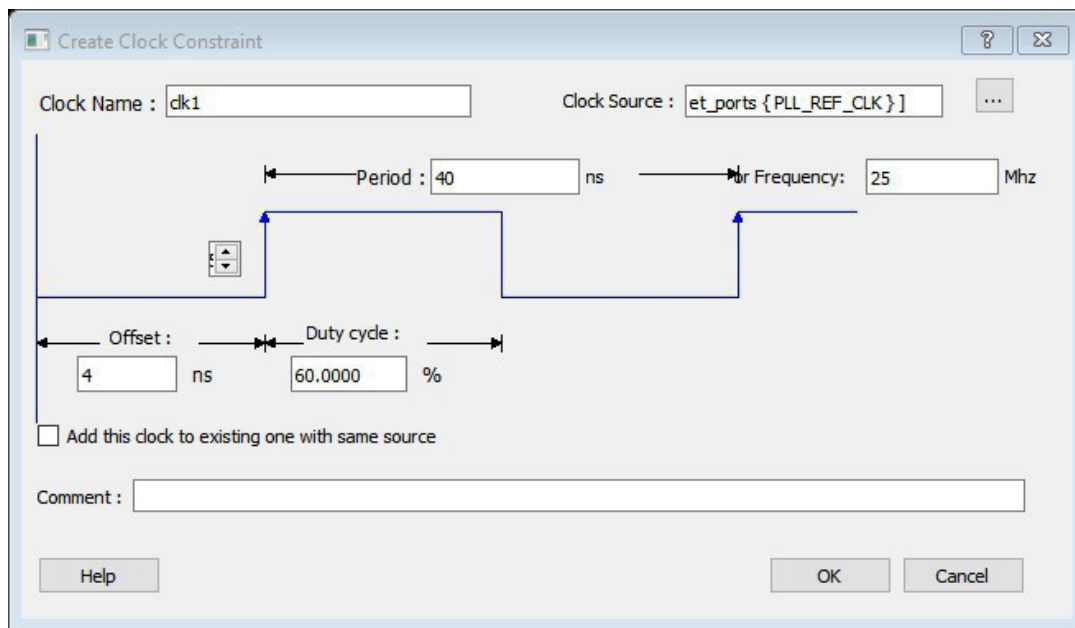
13.8. Create Clock Constraint Dialog Box [\(Ask a Question\)](#)

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 following Create Clock Constraint dialog box from the SmartTime Constraints Editor, choose **Constraints > Clock**.

Figure 13-7. 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

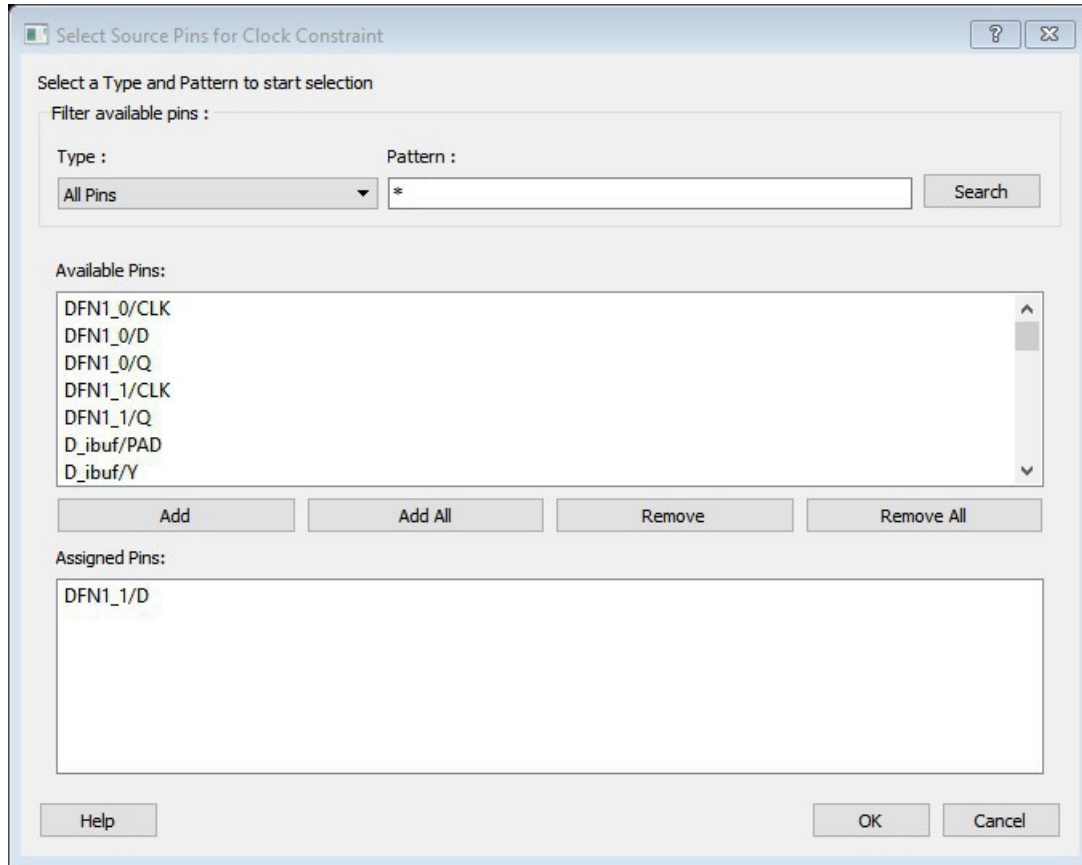
[Specifying Clock Constraints](#)

13.9. Select Source Pins for Clock Constraint Dialog Box [\(Ask a Question\)](#)

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 following Clock Constraint dialog box 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 13-8. 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](#)

13.10. Specifying Clock Constraints [\(Ask a Question\)](#)

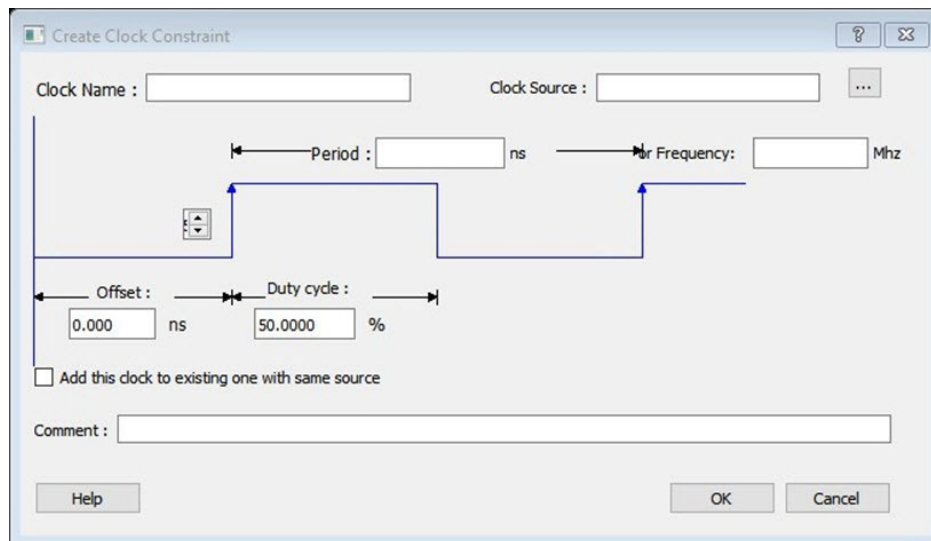
Specifying clock constraints is the most effective way to constrain and verify the timing behavior of a sequential design. Use clock constraints to meet your performance goals.

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 following Create Clock Constraint dialog box appears.

Figure 13-9. Create Clock Constraint Dialog Box



2. Select the pin to use as the clock source. You can click the **Browse** button to display the following [Select Source Pins for Clock Constraint Dialog Box](#).

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.

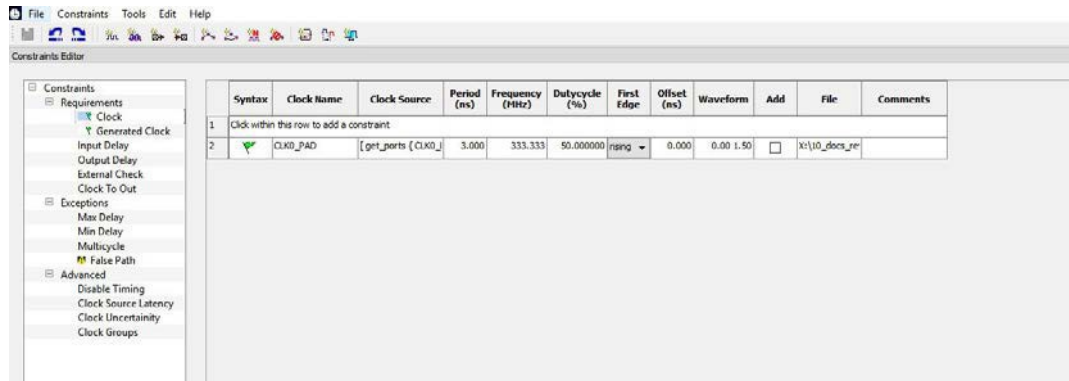
Use the Choose the Clock Source Pin dialog box to display a list of source pins from which you can choose. By default, it displays the explicit clock sources of the design. To choose other pins in the design as clock source pins, select **Filter available objects - Pin Type** as **Explicit clocks, Potential 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.

3. Click **OK** to save these dialog box settings.
4. Specify the **Period** in nanoseconds (ns) or **Frequency** in megahertz (MHz).
5. Modify the **Clock Name**. The name of the first clock source is provided as default.
6. Modify the **Duty cycle**, if needed.
7. Modify the **Offset** of the clock, if needed.
8. Modify the first edge direction of the clock, if needed.

9. Select the check box for Add this clock to an existing one with the same source, if needed.
10. 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 13-10. Timing Constraint View



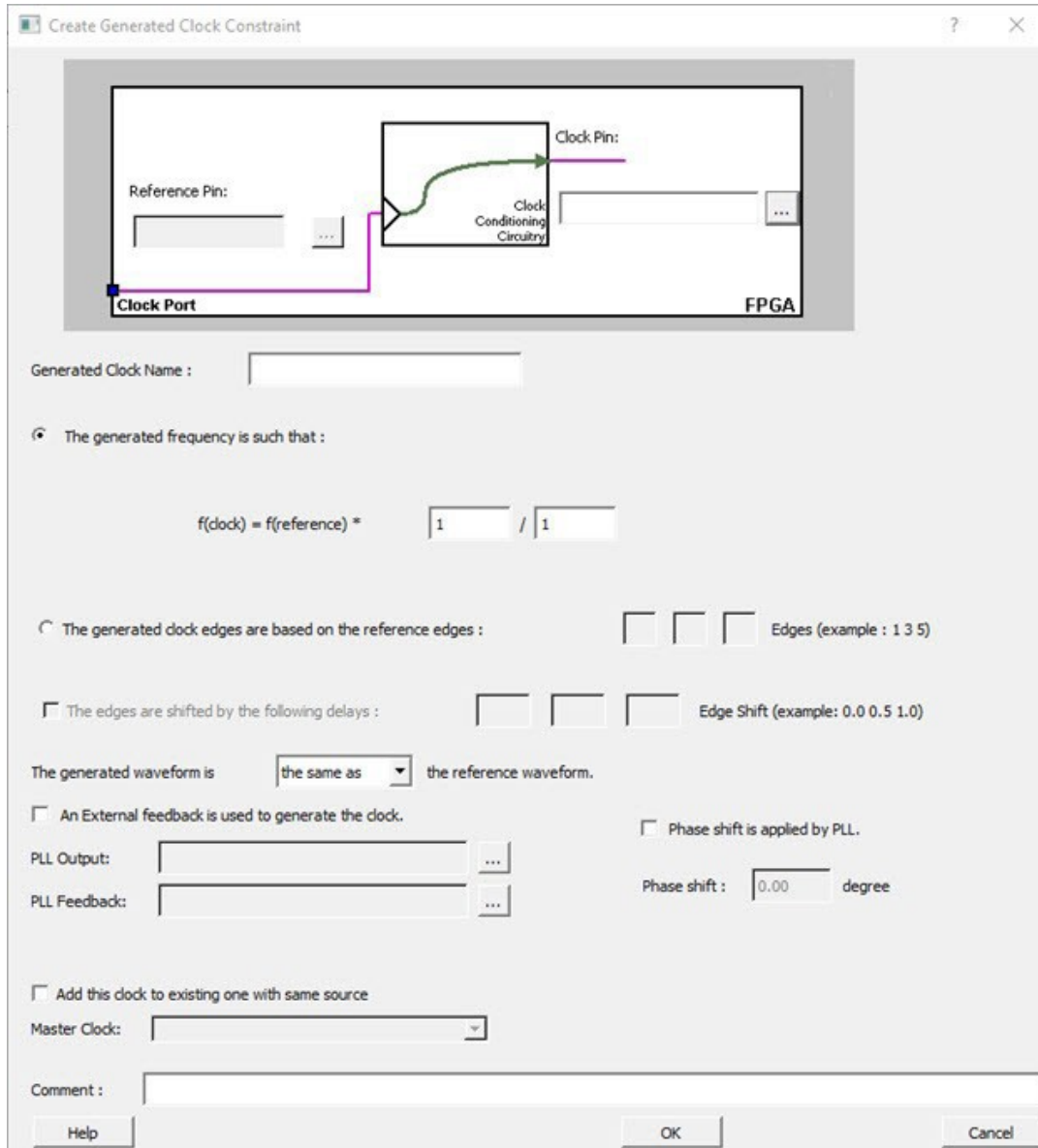
13.10.1. Create Generated Clock Constraint Dialog Box [\(Ask a Question\)](#)

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 following Create Generated Clock Constraint dialog box from the SmartTime Constraints Editor, choose **Constraints > Generated Clock**.

Figure 13-11. Create Generated Clock Constraint



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.

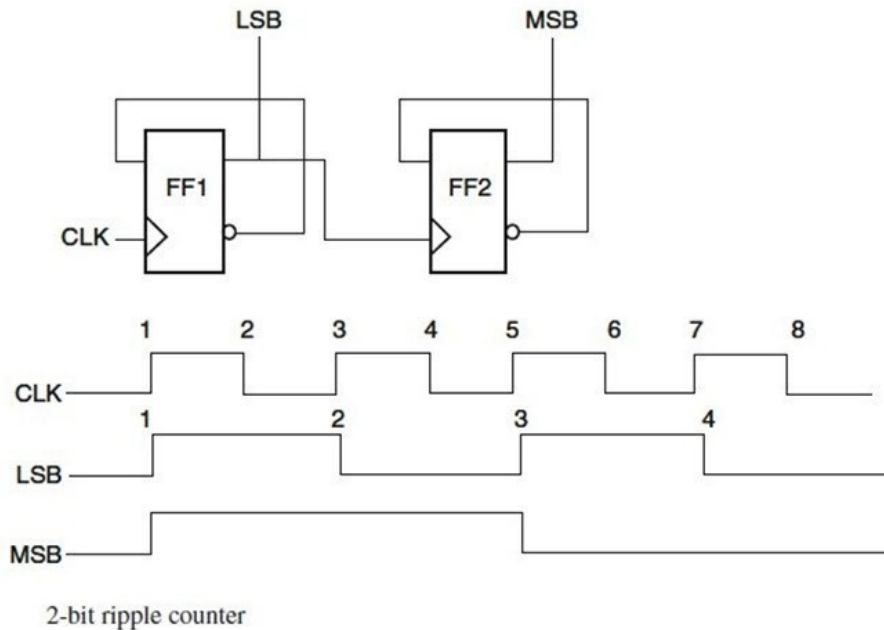
Generated Frequency

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 tooltip 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, will shift each derived edge by 1 time unit. To create a 200 MHz clock from a 100 MHz 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 way in which the GL0/1/2/3 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 must 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

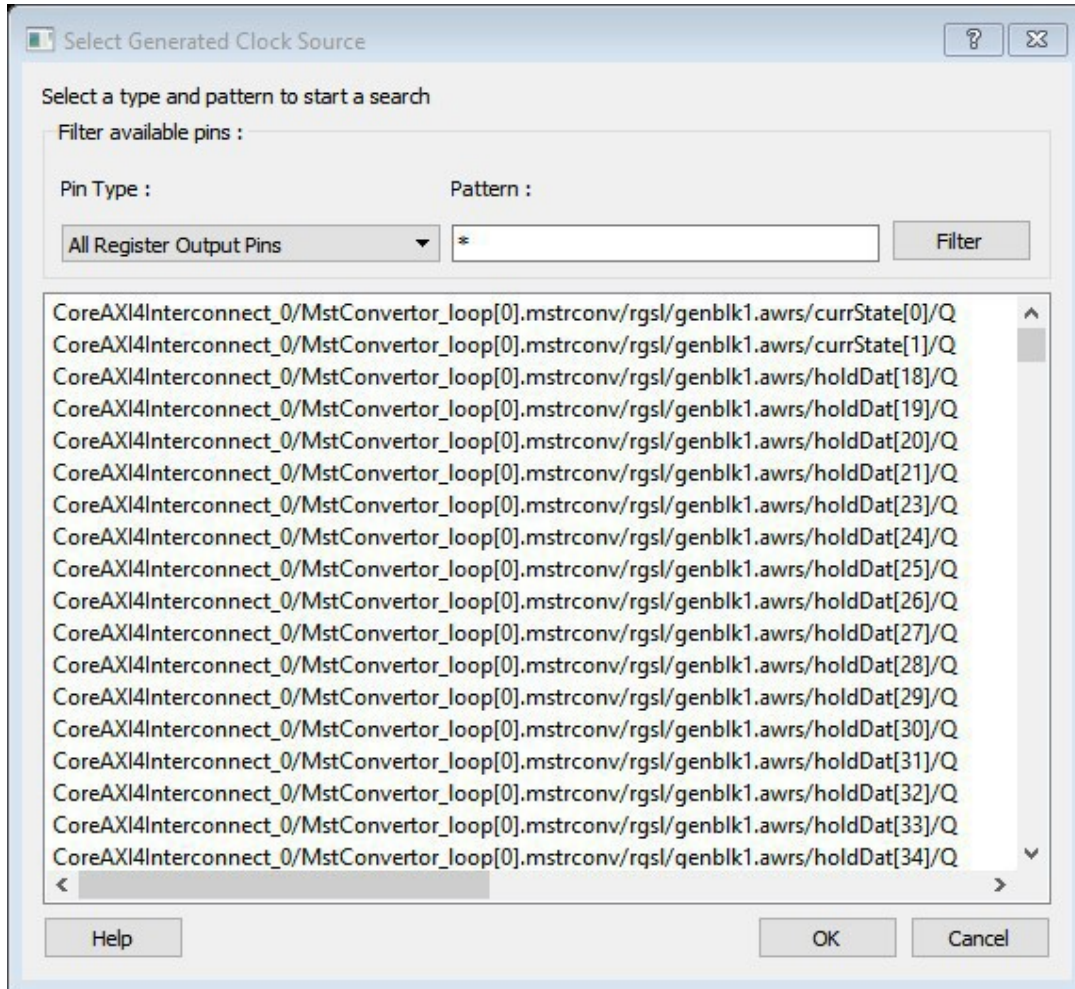
Enter a single line of text that describes the generated clock constraints purpose.

13.10.2. Select Generated Clock Source Dialog Box ([Ask a Question](#))

Use this dialog box to find and choose the generated clock source from the list of available pins.

To open the following Select Generated Clock Source dialog box from the **Timing Constraints Editor**, open the [Create Generated Clock Constraint](#) dialog box and click the **Browse** button for the **Clock Pin**.

Figure 13-12. 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.

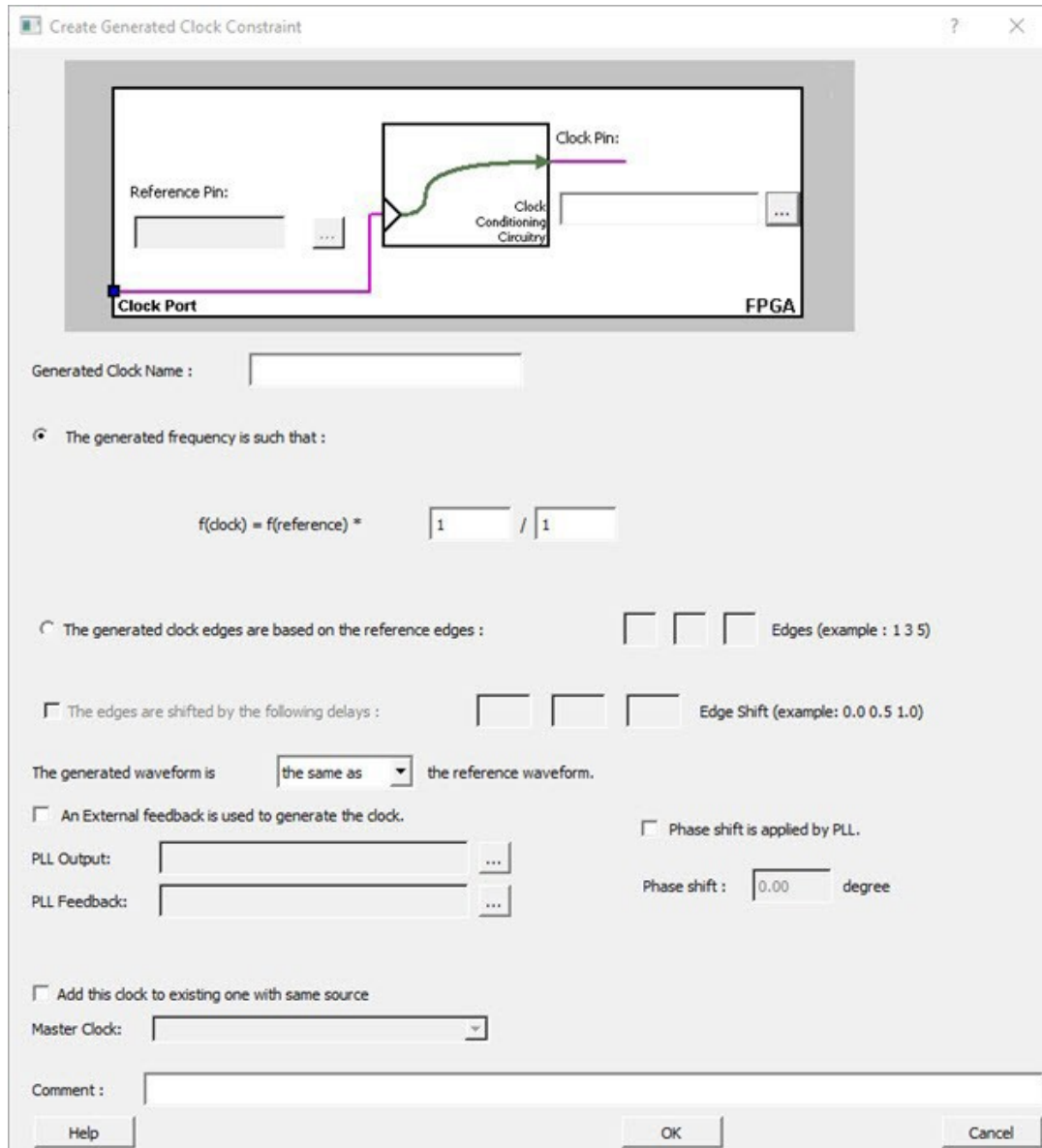
13.11. Specifying Generated Clock Constraints [\(Ask a Question\)](#)

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:
 - a. Click the  icon.
 - b. Right-click the **Generated Clock** in the Constraint Browser and choose **Add Generated Clock**.
 - c. Double-click the Generated Clock Constraints grid. The following Create Generated Clock Constraint dialog box appears.

Figure 13-13. 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

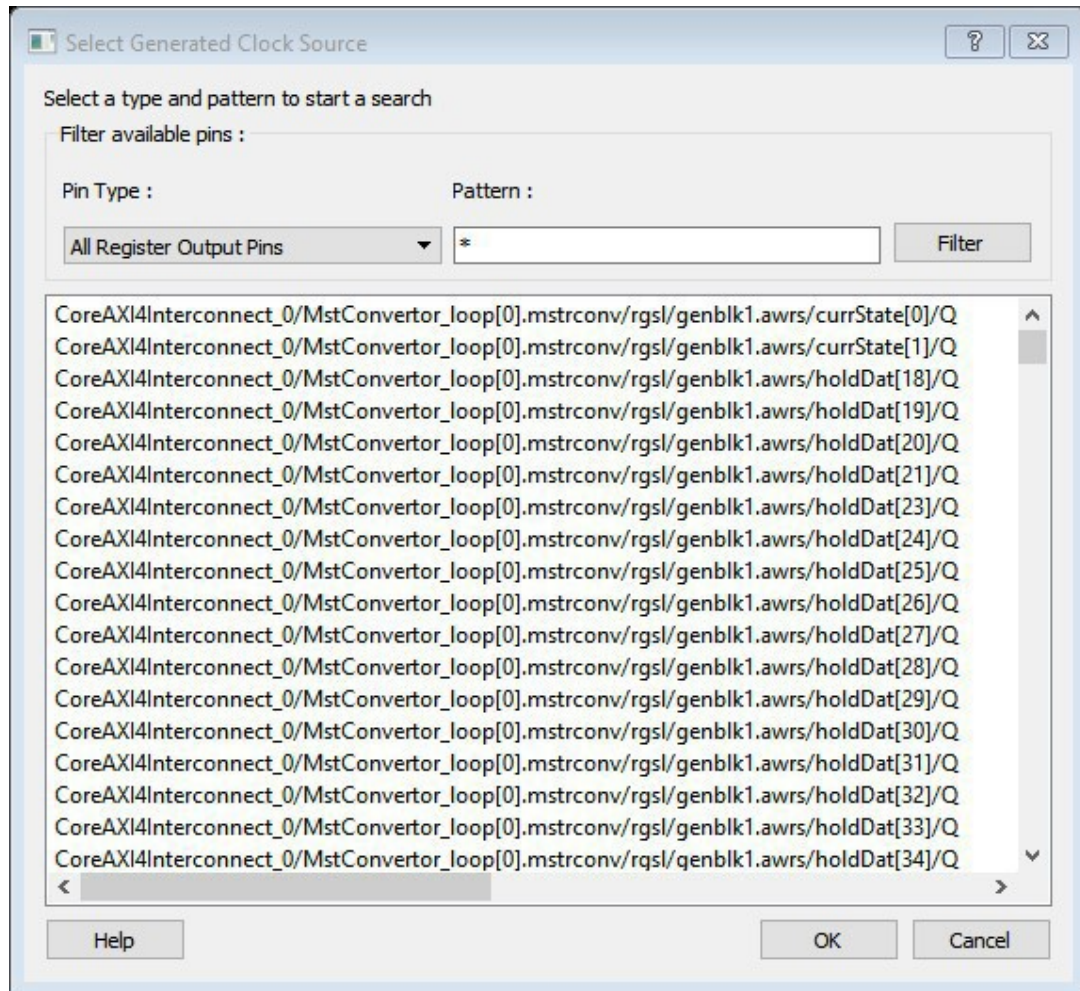
Master Clock:

Comment :

Help OK Cancel

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 following [Select Generated Clock Source](#) dialog box appears.

Figure 13-14. 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.

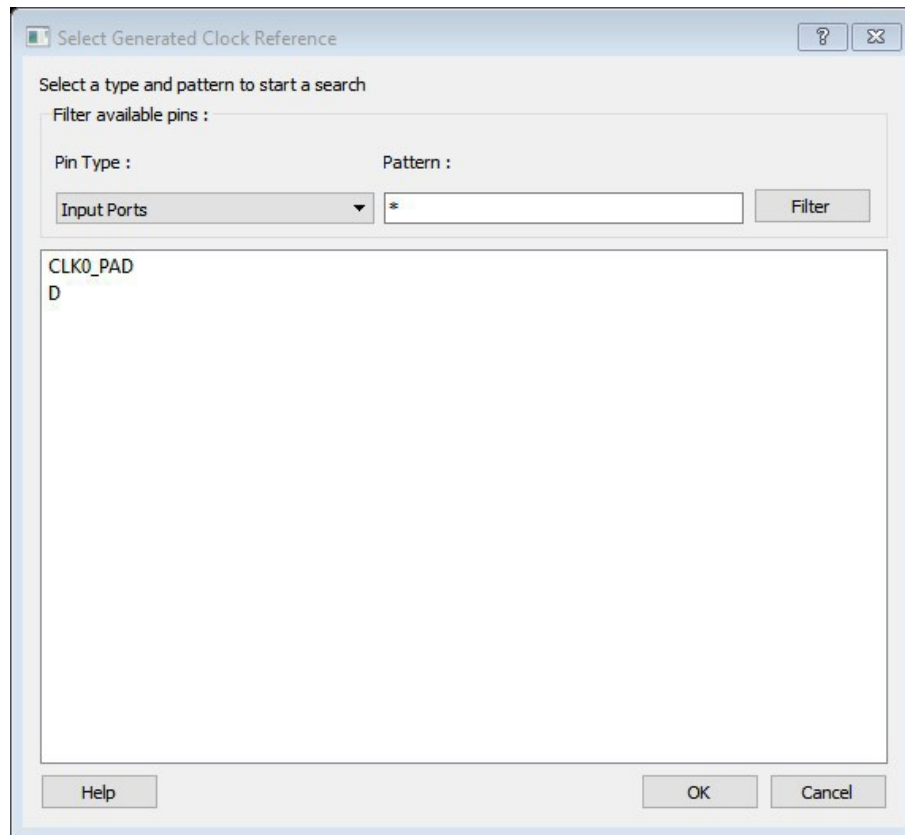
Tip: From the File menu, choose Save to save the newly created constraint in the database.

13.11.1. Select Generated Clock Reference Dialog Box [\(Ask a Question\)](#)

Use this dialog box to find and choose the generated clock reference pin from the list of available pins.

To open the following Select Generated Clock Reference dialog box 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 13-15. Select Generated Clock Reference Dialog Box



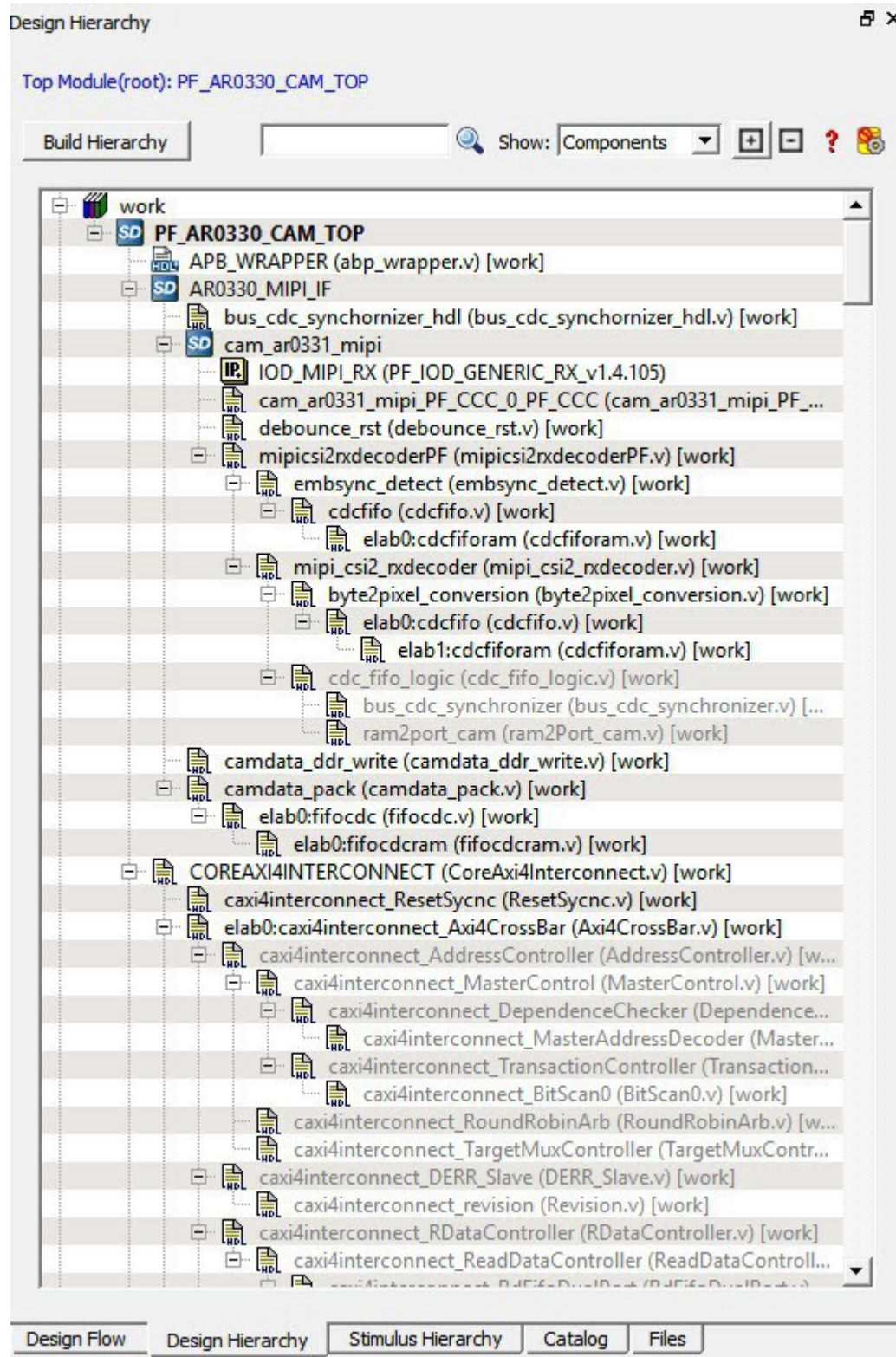
Filter Available Pins

- **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.

13.12. Design Hierarchy in the Design Explorer [\(Ask a Question\)](#)

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 following Design Hierarchy tab displays the structure of the modules and components as they relate to each other along with parameter/generic names and its values on the tooltip for which the module is instantiated. It also displays architecture name for a given entity and Configuration for VHDL modules.

Figure 13-16. Design Hierarchy



A module can have multiple elaborations depending on the different parameters/generics used in the instantiation of the module and all 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 must 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 13-17. Design Hierarchy with Elaborated Modules (Verilog)

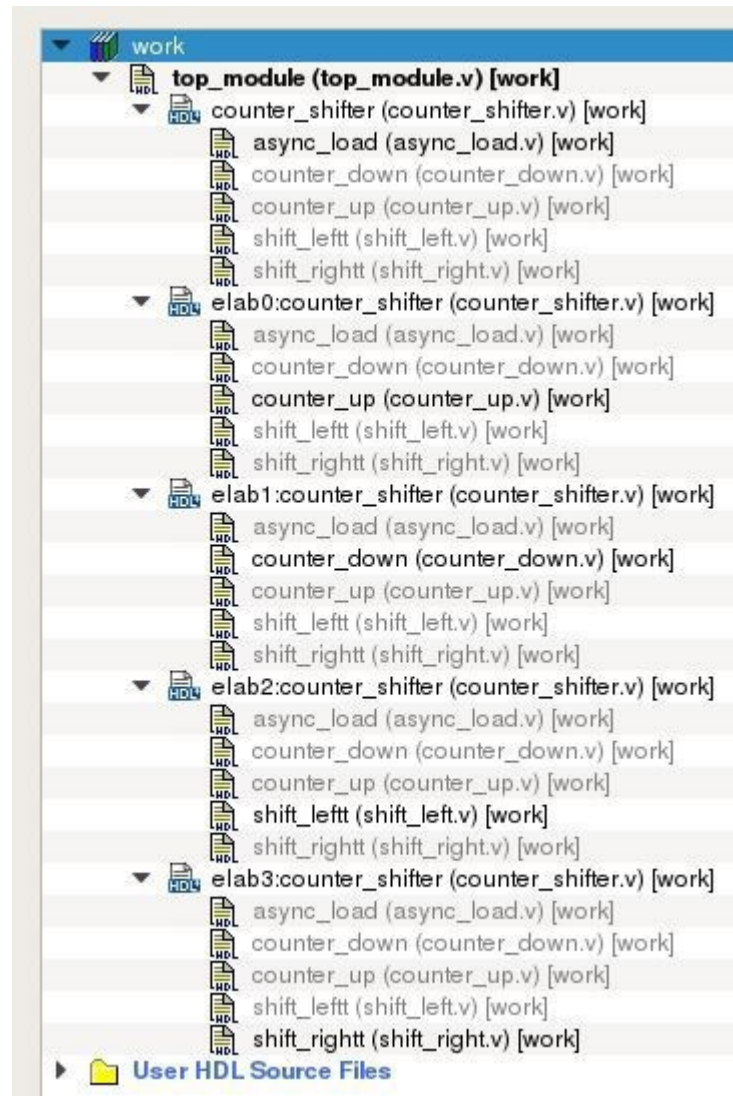
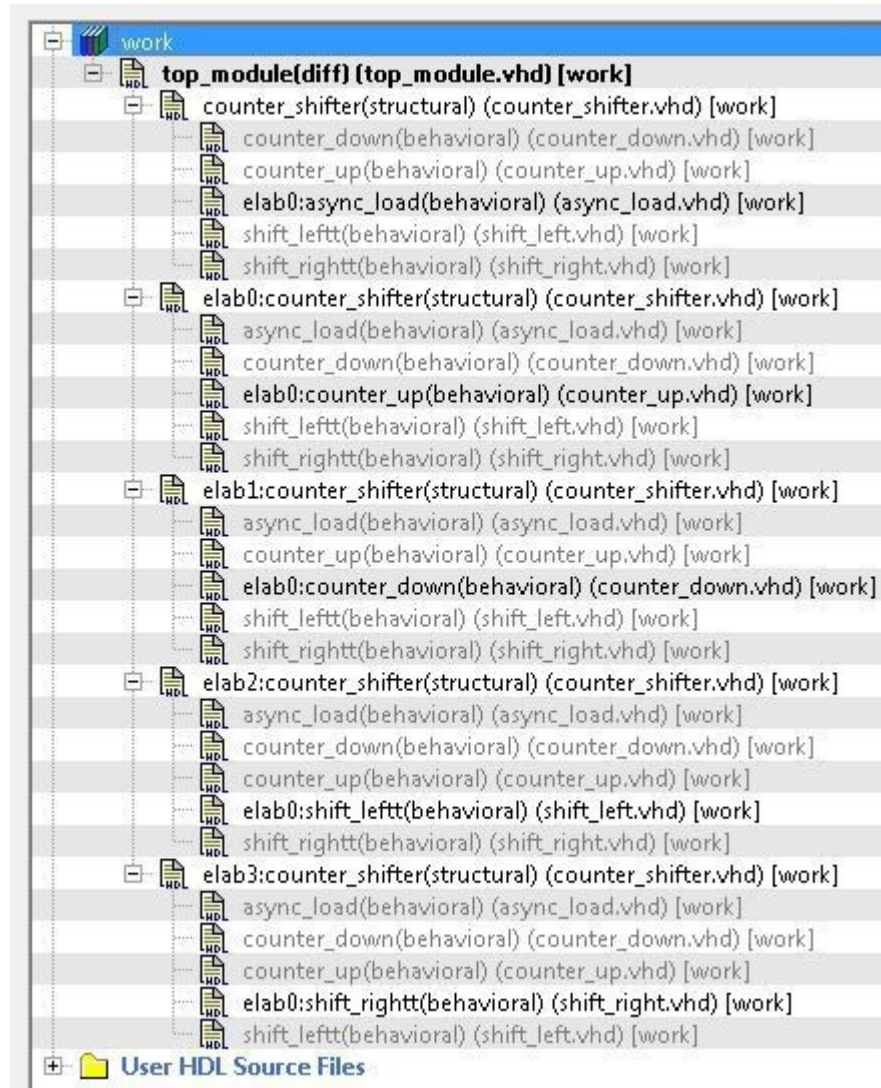


Figure 13-18. Design Hierarchy with Elaborated Modules (VHDL)

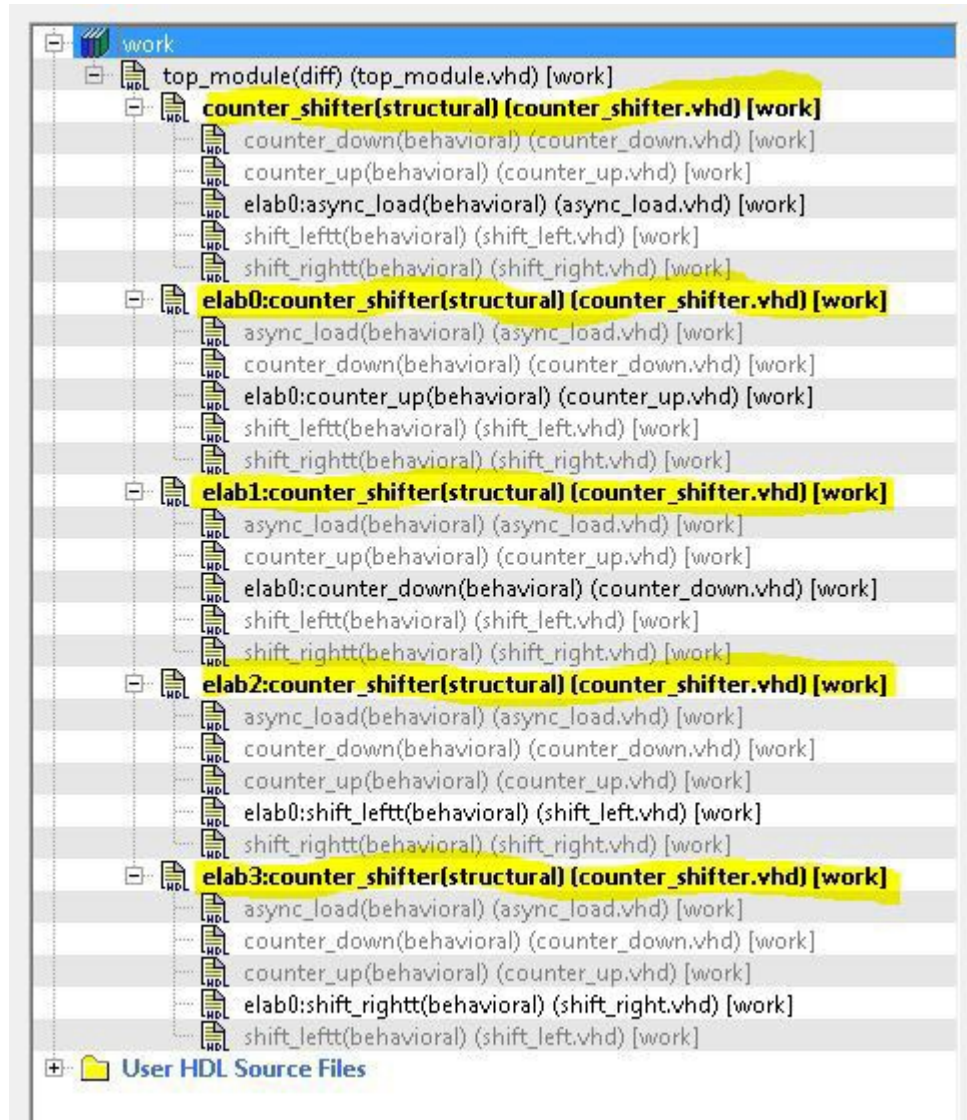


Modules which are not part of the elaboration will be shown in the complete hierarchy but they remain grayed out. When you create a core from a module, all the elaborated modules of that module will be shown as HDL+ core modules. You can get the parameter value of an elaborated module by selecting **Show Module parameters** on the right-click menu of the elaborated module.

Note: A tooltip on each module shows all the parameters with their values for the instantiated module.

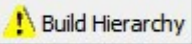
Note: Synthesis output will be the same for different elaborations of the same module, that is **elab0:module1** and **elab1:module1** will have the same synthesis output. When one of the elaborated modules is set as root, all the elaborations will be highlighted in the Design Hierarchy, as shown in the following figure.

Figure 13-19. Design Hierarchy When One of the Elaborated Modules 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.



Attention: If a module from one HDL file instantiates another module defined in a different HDL file located in a separate library, the Stimulus Hierarchy might not appear correctly. Additionally, if the same modules are defined in multiple libraries, the hierarchies might not build correctly.

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 following table.

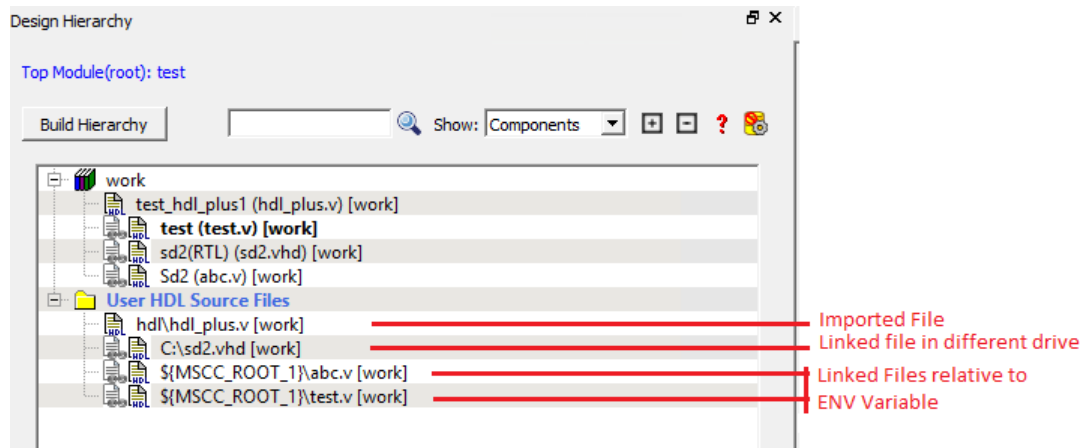
Table 13-2. Design Hierarchy Icons

Icon	Description
	SmartDesign component
	SmartDesign component with HDL netlist not generated
	IP core was instantiated into SmartDesign but the HDL netlist is not 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

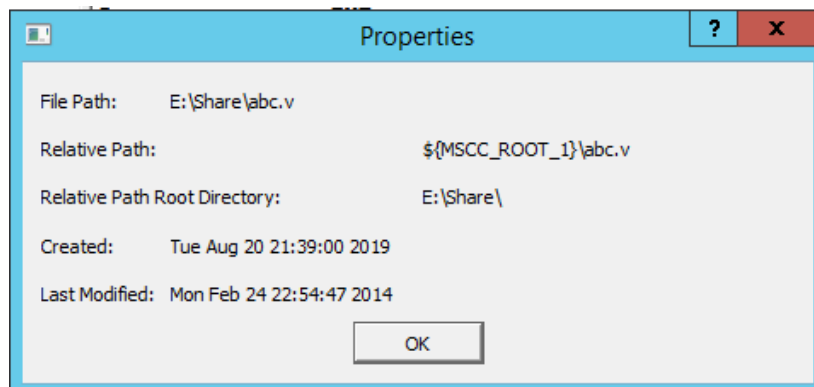
The following figure shows linked files that have a relative path in the design hierarchy with the Environment variable. Linked files that are in a drive different to the drive specified in Environment variable path are shown as absolute files in Windows.

Figure 13-20. 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 13-21. File Path Properties in Design Hierarchy Representing Relative Path and Relative Path Root Directory



13.13. Digest File [\(Ask a Question\)](#)

To verify which bitstream file was used on each programmed device, run the VERIFY or VERIFY_DIGEST action on the respective device. 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 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 contains a combination of any of the three primary bitstream components.

For more information, see [FPGA Programmed Content Integrity Checking and Programming File Tracking using Cryptographic Digests](#).

See Also

[Export Bitstream](#)

13.14. Design Rules Check [\(Ask a Question\)](#)

The Design Rules Check runs automatically when you generate your SmartDesign; the results appear in the **Reports** tab. To initiate a Design Rules Check manually, in **SmartDesign Canvas**

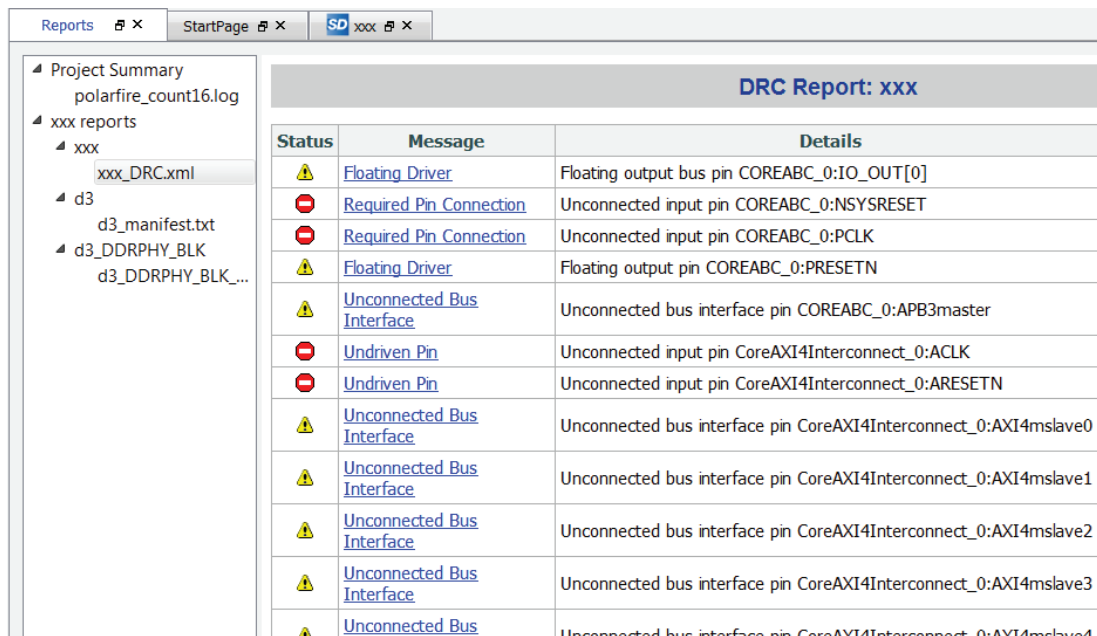
tab, click  icon.

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 following figure). Error messages are shown with a small red sign and warning messages with a yellow exclamation point. 
- **Message** identifies the specific error/warning (see the following list); click any message to see where it appears on the Canvas.
- **Details** provides information related to the Message.

Note: If the following message appears, see the [Simultaneous Switching Noise and Signal Integrity Application Note](#) on the Microchip website for information about SSO limits: When debugging pin placement issues in Simultaneous Switching Output (SSO) systems, Given the current pin placements in Bank X, the SSO limit for the selected drive strength <strength> and IO Standard <standard> has been exceeded.

Figure 13-22. Design Rules Check Results



Status	Message	Details
	Floating Driver	Floating output bus pin COREABC_0:IO_OUT[0]
	Required Pin Connection	Unconnected input pin COREABC_0:NSYSRESET
	Required Pin Connection	Unconnected input pin COREABC_0:PCLK
	Floating Driver	Floating output pin COREABC_0:PRESETN
	Unconnected Bus Interface	Unconnected bus interface pin COREABC_0:APB3master
	Undriven Pin	Unconnected input pin CoreAXI4Interconnect_0:ACLK
	Undriven Pin	Unconnected input pin CoreAXI4Interconnect_0:ARESETN
	Unconnected Bus Interface	Unconnected bus interface pin CoreAXI4Interconnect_0:AXI4mslave0
	Unconnected Bus Interface	Unconnected bus interface pin CoreAXI4Interconnect_0:AXI4mslave1
	Unconnected Bus Interface	Unconnected bus interface pin CoreAXI4Interconnect_0:AXI4mslave2
	Unconnected Bus Interface	Unconnected bus interface pin CoreAXI4Interconnect_0:AXI4mslave3
	Unconnected Bus Interface	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, that is, tie LOW (GND) or tie HIGH (VCC).

Floating Driver - You can mark the pin unused if it is not being 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. Visit the Microchip [Licensing page](#) to obtain the necessary license.

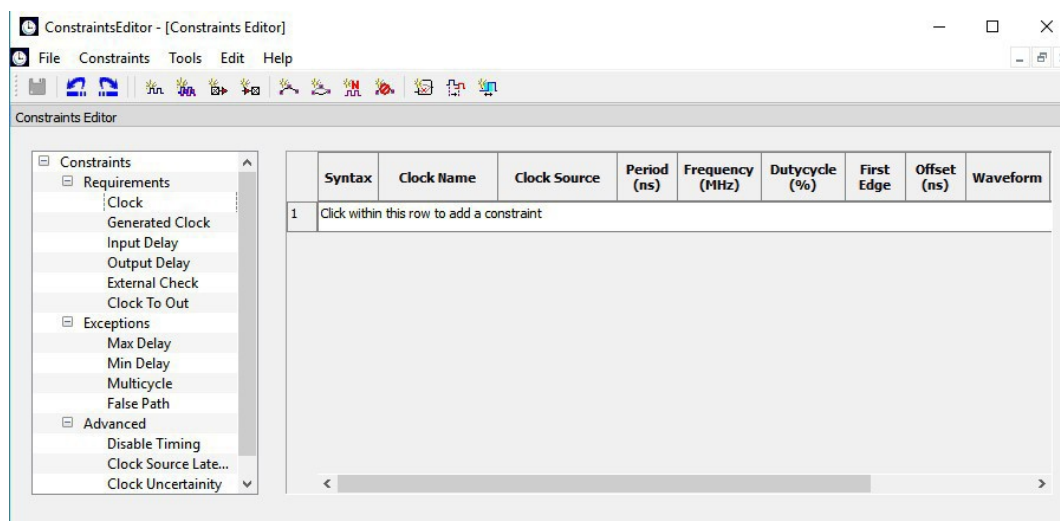
No Top-level Ports - There are no ports on the top-level. To auto-connect top-level ports, right-click the Canvas and choose Auto-connect.

Self-Instantiation - A component cannot instantiate itself. This is reported only in the Log/Message Window.

13.15. Editable Constraints Grid [\(Ask a Question\)](#)

The Constraints Editor enables you to add, edit, and delete.

Figure 13-23. 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**.

13.16. Files Tab and File Types [\(Ask a Question\)](#)

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 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 Libero SoC version 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 (log file) **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 log file), *.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.

13.17. Importing Files [\(Ask a Question\)](#)

Anything that describes your design, or is needed to program the device, is a project source. These might include schematics, HDL files, simulation files, testbenches, and so on. 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 13-3. 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)	*.cdf
MSS Components	*.cxz

13.18. Layout Error Message: layoutg4NoValidPlacement [\(Ask a Question\)](#)

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 might 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 might 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 fails.

- Remove all region constraints and re-run Place and Route. Some designers make it a practice to put all their region constraints in a single, separate PDC file; in which case they need to disable that file.
 - If this Place and Route re-run still fails, there might be wider issues with the design's size and complexity that cannot be addressed by changes to Place and Route options.
 - If the unconstrained Place and Route re-run succeeds, then the you must 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 and Route with each constraint restoration and repeat the process until the failing region(s) is identified.
Depending on requirements, the failing region might be handled by removing or changing its constraints, or revising its design to use less resources.

You can 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 can produce a placement solution that might not meet tight timing constraints because 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 are exhausted.

13.19. Layout Error Message: layoutg4DesignHard [\(Ask a Question\)](#)

This design is very difficult to place, and high-effort techniques were required to fit it. This might lead to increased layout runtime and diminished timing performance.

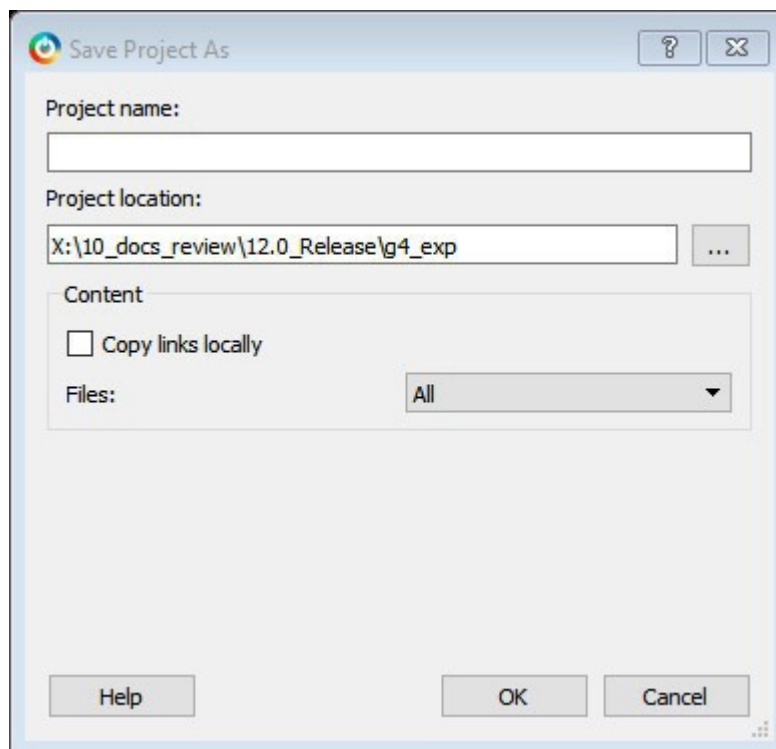
This message typically appears in designs with high utilization — a very full design, or a design with region constraints which are, themselves, very full. It can also occur in designs with moderate utilization but with numerous, long carry chains.

No immediate action is required. However, if this notice is observed during Layout, the resultant performance of the design and the runtime of the Layout tools might 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.

13.20. Save Project As Dialog Box [\(Ask a Question\)](#)

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 13-24. 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 - Check this check box to copy the links from your current project into your new project. If you do not check this check box, 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 following table.

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

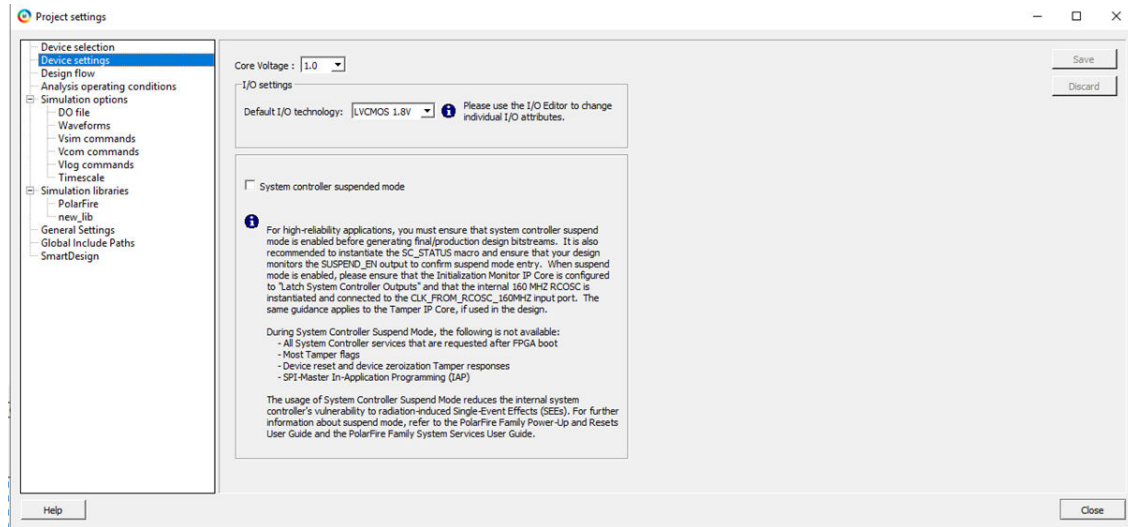
Note: *.edn files are not supported in PolarFire.

13.21. Project Settings Dialog Box [\(Ask a Question\)](#)

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 13-25. Project Settings Dialog Box



Device Selection	Sets the device Die and Package for your project. For a detailed description of the options, see the New Project Creation Wizard: Device Selection section.
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 suspend mode	If System controller suspend mode is enabled in any PolarFire design, the following operations and features are not available during normal operation. For more information, see system services in the PolarFire Family System Services user guide. • 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, System controller suspend mode is enabled by default for all new RT PolarFire projects. If System controller suspend mode is disabled, it increases vulnerability to radiation single event effects in the system controller.
Design Flow	For more information, see the Project Settings: Design Flow .
Analysis Operating Conditions	Allows you to change the operation conditions based on your device by setting ranges for operating temperature, core voltage, and default I/O voltage available in the pick-list. Based on the selected device family, the operating conditions differ. The following list describes the device families with family-specific operating conditions: PolarFire Currently, for all the PolarFire devices the Operating Temperature Range can be set to EXT or IND. For some of the devices within this family the Operating Temperature Range can be set to TGrade2 or MIL as well. The RT PolarFire devices, allow you to set the Operating Temperature Range to either MIL or IND. PolarFire SoC Currently, for all the PolarFire SoC devices, the Operating Temperature Range can be set to EXT, IND or TGrade2.

The RT PolarFire devices allow you to set the **Operating Temperature Range** to either **MIL** or **IND**.



Important: For both **PolarFire** and **PolarFire SoC**, the **Core Voltage Range** and **Default I/O Voltage Range** options are disabled.

- SmartFusion 2** Currently, for all the SmartFusion 2 devices, the **Operating Temperature Range**, **Core Voltage Range**, and **Default I/O Voltage Range** can be set to **COM**, **IND**. If desired, the ranges can be defined manually by selecting **Custom**.
- IGLOO 2** Currently, for all the IGLOO 2 devices, the **Operating Temperature Range**, **Core Voltage Range**, and **Default I/O Voltage Range** can be set to **COM**, **IND**, **TGrade1**. If desired, the ranges can be defined manually by selecting **Custom**.
- RTG4 FPGA** Currently, for all the RTG4 devices, the **Operating Temperature Range**, **Core Voltage Range**, and **Default I/O Voltage Range** can be set to **MIL**. Alternatively, select **Custom** to define the ranges manually.

These settings are propagated to **Verify Timing**, **Verify Power**, and **Backannotated Netlist** for you to perform **Timing/Power Analysis**.

After you select the range for operating temperature, the **Minimum**, **Typical**, and **Maximum** temperature values are automatically displayed for the selected device.

Simulation Options and Simulation Libraries

Sets your simulation options. For more information, see the [Project Settings: Simulation - Options and Libraries](#).

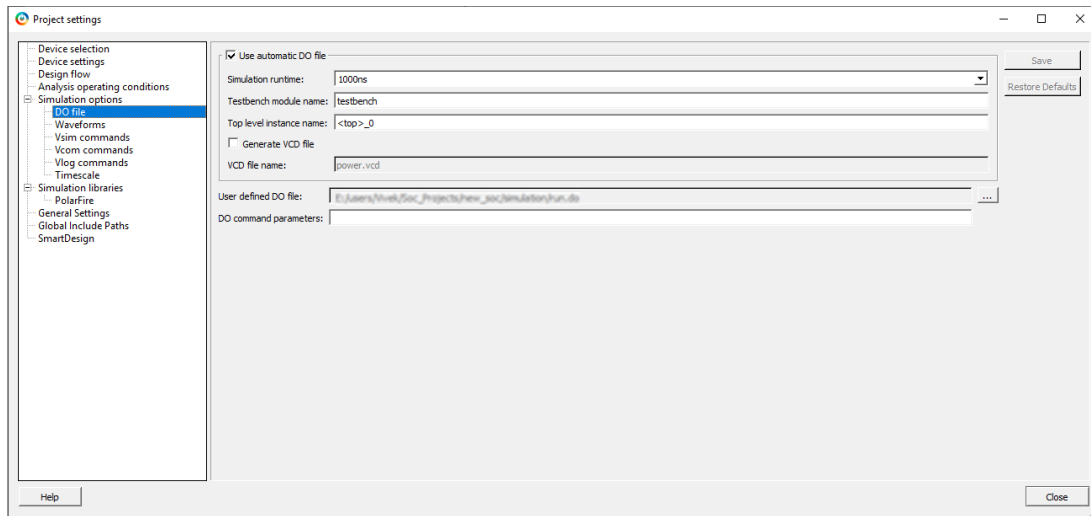
13.21.1. Project Settings: Simulation - Options and Libraries [\(Ask a Question\)](#)

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 want to edit: **DO file**, **Waveforms**, **Vsim commands**, **Timescale**. For **Simulation libraries** click the library whose path you want to change.

Figure 13-26. Project Settings: DO File

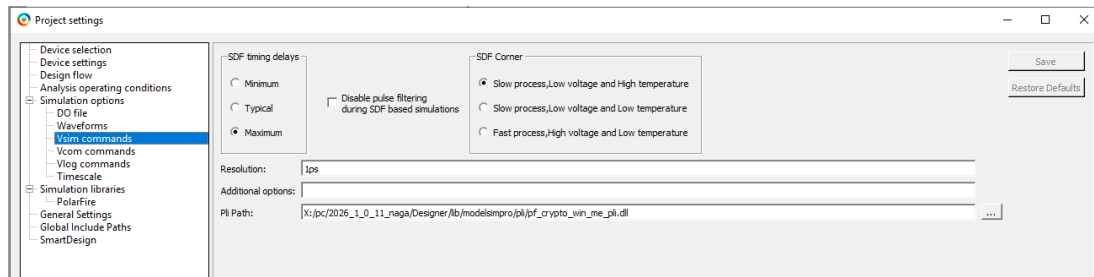


DO file:

- **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 must 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 run simulation(presynth/postsynth/postlayout).
 - **Generate VCD file** - Check the check box 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 QuestaSim ME Pro.
 - **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.

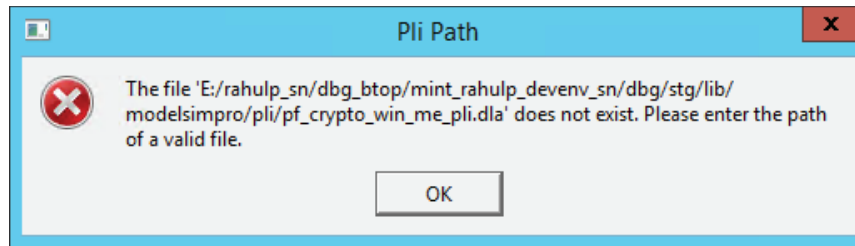
Vsim Commands

Figure 13-27. Vsim Commands

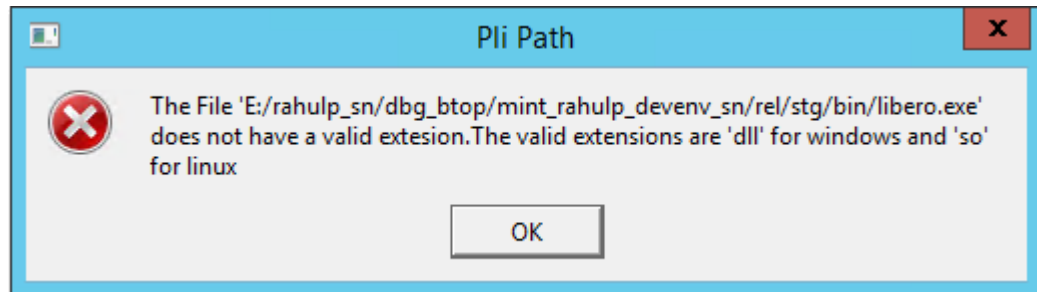


- **Resolution - The default is 1ps.** Some custom simulation resolutions might not work with your simulation library. Consult your simulation help for more information on how to work with your simulation library and detect infinite zero-delay loops caused by high resolution values.
- **Additional options** - 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 $0 \leq \text{value} \leq 1$
 - -gCORRECTION_PROBABILITY=<value>, where $0 \leq \text{value} \leq 1$
 - 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 you run 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, you must 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 in the following example.


```
vlog -sv -work postlayout "$(PROJECT_DIR)/designer/sd1/sd1_fast_hv_lt_ba.v" vsim
-L PolarFire -L postlayout -t 1ps -sdfmax
/sd1=$(PROJECT_DIR)/designer/sd1/sd1_fast_hv_lt_ba.sdf +pulse_int_e/1
+pulse_int_r/1 +transport _int_delays postlayout.sdl
```
 - To run a simulation on a design having Crypto core, you need to enter the PLI path in the PLI Path field. You 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. You 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 you enter a wrong PLI path and click on Save.



2. If the user enters a wrong file extension, the following error message is displayed.

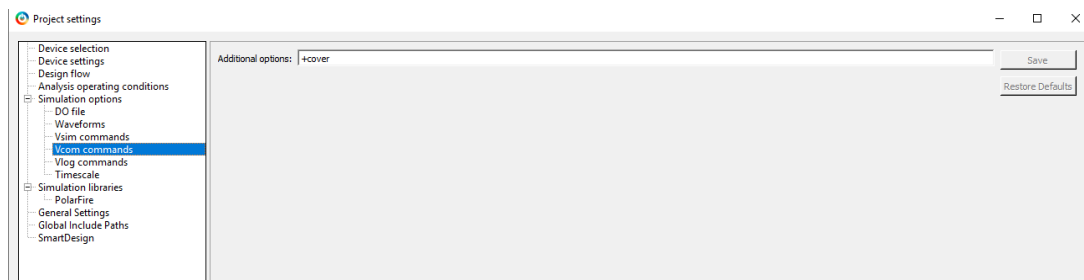


- To compile all the files under System Verilog Multi-File Compilation Unit (MFCU) mode, group them based on the library. This applies to all the .hdl files, simulation files, and core files. The following code example provides a general syntax to compile the files in System Verilog MFCU mode.

```
vlog "+incdir+<incdir1>" "+incdir+<incdir2>" -sv -mfcu -work library1 "file1"
"file2" "file3"
vlog "+incdir+<incdir1>" "+incdir+<incdir2>" -sv -mfcu -work library2 "file4"
"file5" "file6"
```

Vcom Commands

Figure 13-28. Vcom Commands



- **Additional options** Text in this field is added to the VCOM command.

Vlog Commands

Figure 13-29. Vlog Commands



- **Additional options** Text in this field is added to the VLOG command.

Timescale: Complete the following options:

- **TimeUnit** - Enter a value and select s, ms, us, ns, ps, or fs from the drop-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.

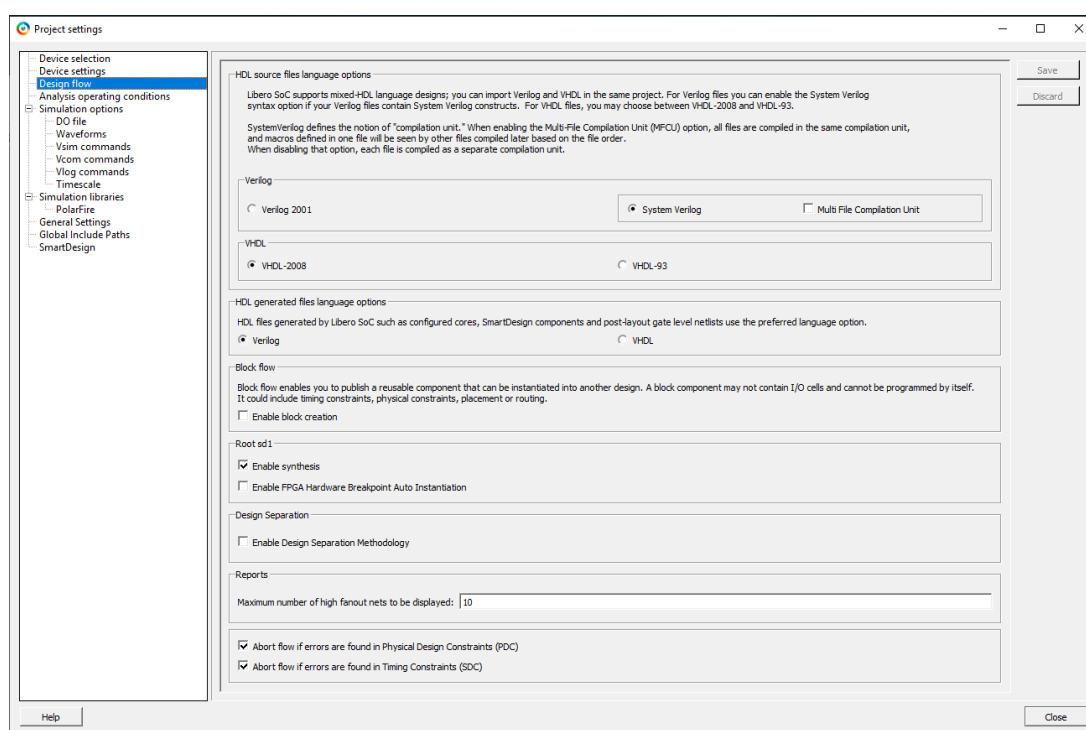
Simulation Libraries: Complete the following options:

- **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 **Browse** to navigate to your library directory.

13.21.2. Project Settings: Design Flow [\(Ask a Question\)](#)

To access the Design flow page, from the Project menu choose **Project settings**. From the **Project settings** window, select the **Design flow** from the left pane.

Figure 13-30. 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 QuestaSim ME Pro 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 because it can 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 might have completed the Place and Route step and met the timing and power requirements. These low-level design blocks can be imported into a Libero SoC project and re-used as components in a higher level design. See *Designing with Designer Block Components* in the 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 check box 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 check box to abort Place and Route if the Timing Constraint SDC file contains errors.

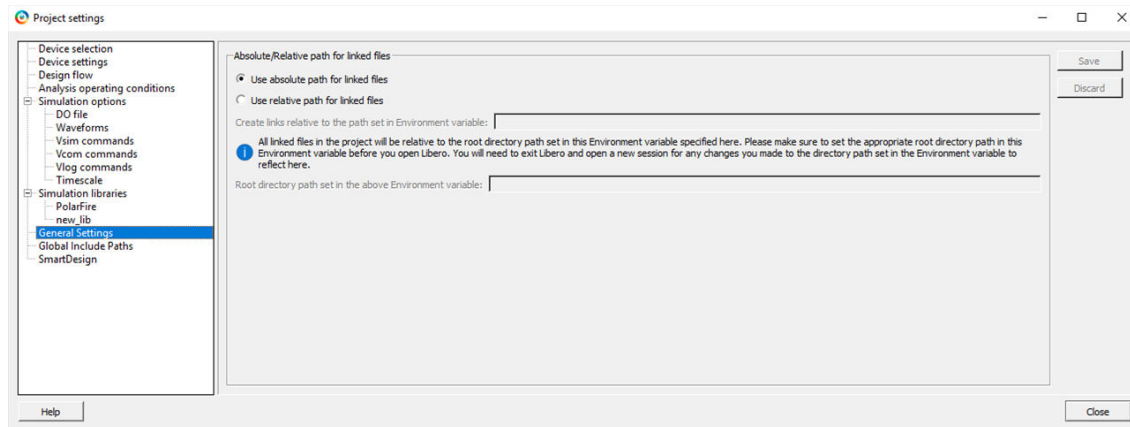
Abort flow if 3.3 V I/Os are found in the design – Check this check box to abort Place and Route if 3.3 V I/Os are found in the design. This allows the tool to generate an error similar to the following. However, if the check box is unchecked, the Place and Route tool will generate the same error message as a warning without interrupting the design flow process.

Error: I/Os at 3.3 V are vulnerable to Single Event Latch-up at low LET levels. Review RT PolarFire radiation data at [Microchip.com](https://www.microchip.com).

13.21.3. Project Settings: General Settings [\(Ask a Question\)](#)

To access the General Settings page, from the Project menu choose **Project Settings** and click the General Settings tab.

Figure 13-31. Project Settings Dialog Box – General Settings Tab



Absolute/Relative path for linked files

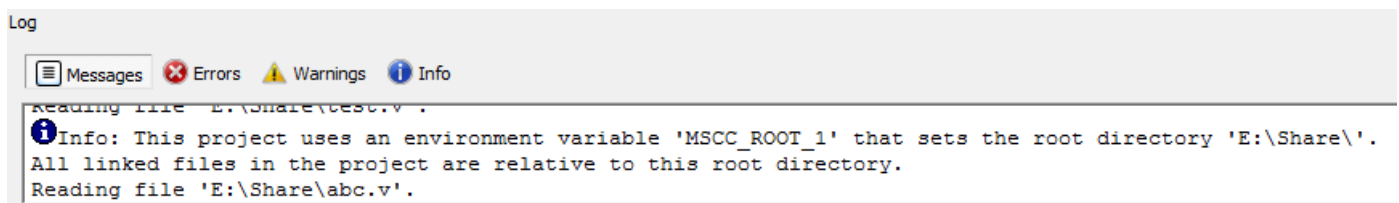
Use either "absolute path for linked files" or "relative path for the linked files" for the files linked in the project. The path set in the environment variable is read only.

If "Use absolute path for linked files" option is selected, all the linked files are stored with the absolute path.

If "Use relative path for linked files" option is selected, 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 you set the relative path and provides an environment variable that has a valid path, the following message appears in the log window.

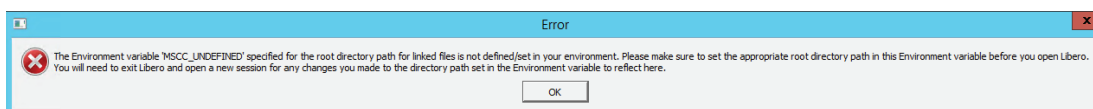
Figure 13-32. Message Displayed in Log Window After Environment Variable is Set



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.

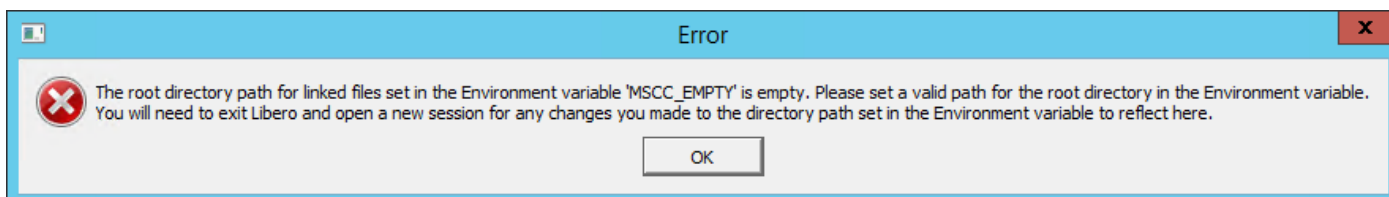
The environment variable set by the user is validated before proceeding. If the environment variable set by the user is not present or if it is set after starting Libero, the tool issues the following error.



Error message: The Environment variable `<env_variable_name>` specified for the root directory path for linked files is not defined/set in your environment. Ensure to set the appropriate root directory path in the Environment variable before you open Libero. Exit Libero and open a new session for any changes you made to the directory path set in the Environment variable to reflect here.

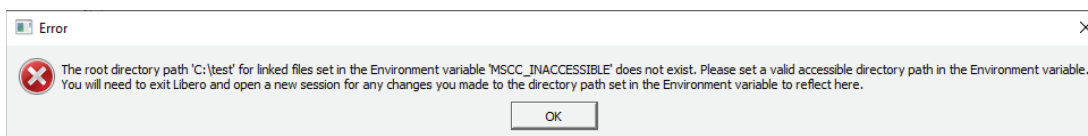
Other error messages that can occur are:

- Empty path in Environment variable



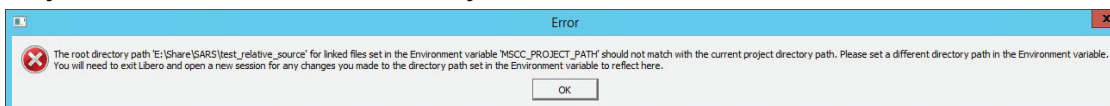
Error message: The root directory path for linked files set in the Environment variable `<env_variable_name>` is empty. Set a valid path for the root directory in the Environment variable. Exit Libero and open a new session for any changes you made to the directory path set in the Environment variable 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. 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.

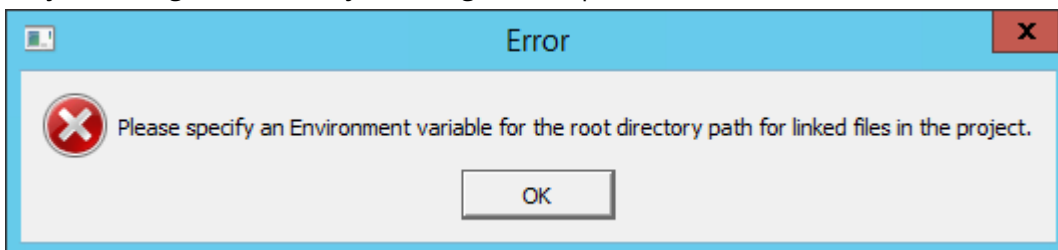
- Project location is set as root directory in environment variable



Error message: The root directory path `<env_variable_path>` for the linked files set in the Environment variable

`<env_variable_name>` must not match with the current project directory path. Set a different directory path in the Environment variable. You 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



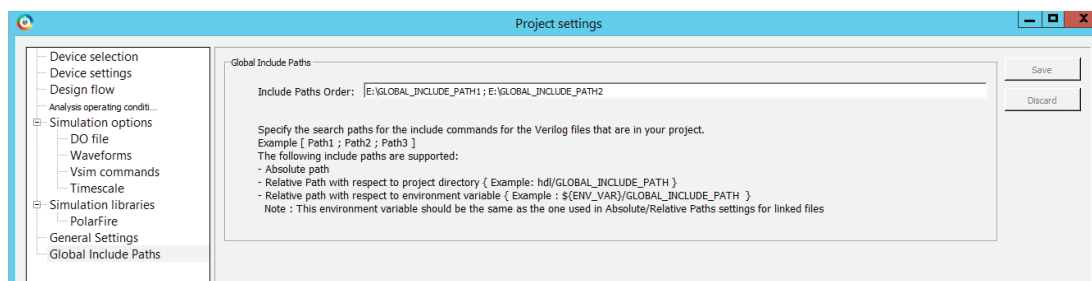
Error message: Specify an Environment variable for the root directory path for linked files in the project.

Note: Set the Environment Variable before starting Libero. Otherwise, Libero will not be able to get the path set in Environment Variable.

13.21.4. Project Settings: Global Include Paths [\(Ask a Question\)](#)

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 13-33. 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

You can 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 with respect to Environment Variable: { **\$(ENV_VAR)**/GLOBAL_INCLUDE_PATH_1 }

Here, the environment variable must 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 the following cases:
 - 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.

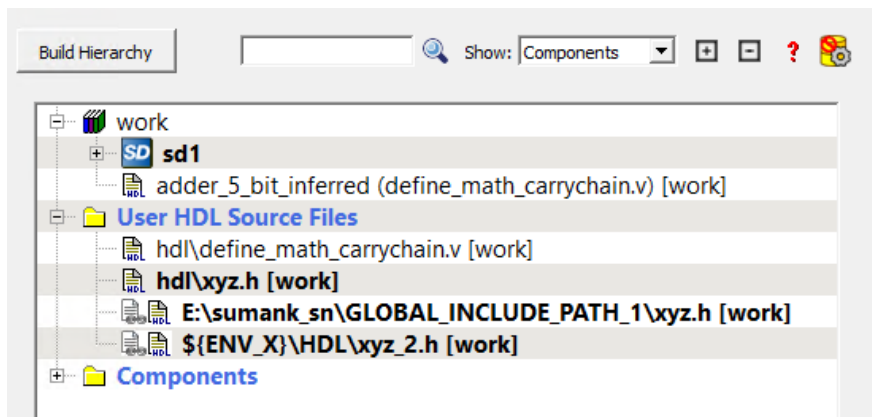
The files are audited only if they are part of the project.

13.22. Global Include Files [\(Ask a Question\)](#)

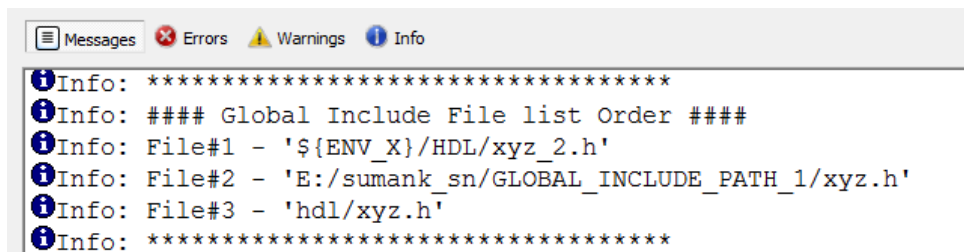
This option facilitates the users to provide global include file/s in Libero and manage how these files are sent to downstream tools occurring in Design Flow process.

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 in the following figure.



The selected order of global include files is shown in Info tab in log window:



Note:

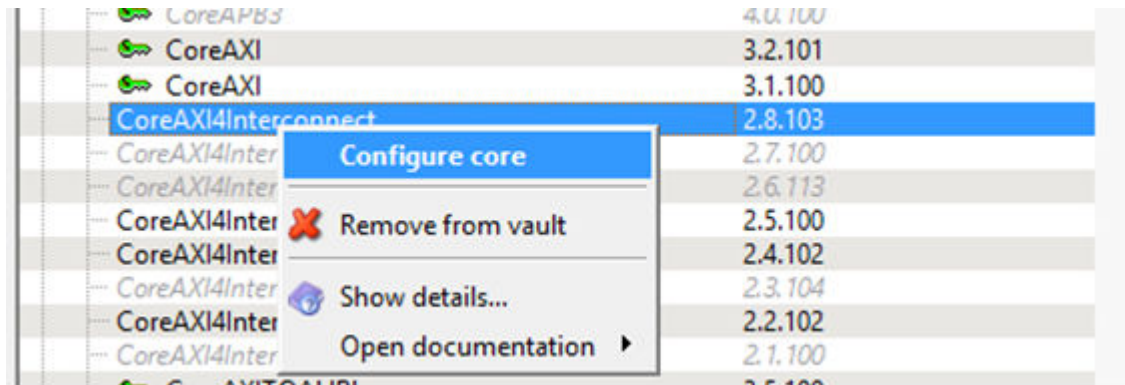
Setting Global Include File will use System Verilog MFCU mode internally. An info message will be generated in the log window as below:

Info: Global Include file requires System Verilog setting to be in MFCU mode. Enabling Multi File Compilation Unit internally.

13.23. Create and Configure Core Component [\(Ask a Question\)](#)

1. The user can create, configure, and generate a component for any core in the Libero SoC Catalog window.
2. Select a core in the Catalog window by double clicking or right clicking on it and choose the option 'Configure core'.
3. Specify a name to the core component to be created in the pop-up dialog box.
4. 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 13-34. 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 13-35. 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" \
```

13.24. Search in Libero SoC [\(Ask a Question\)](#)

Search options vary depending on your search type.

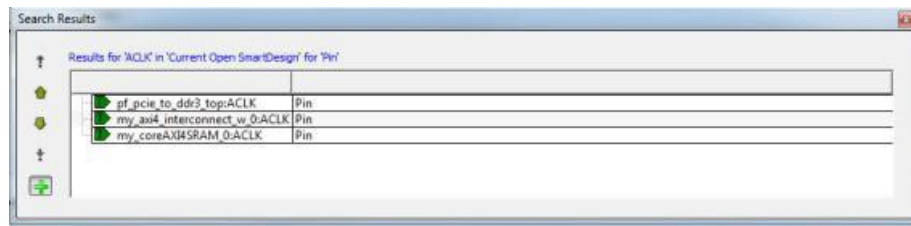
To find a file:

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; 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 following figure).

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 13-36. Search Results



Currently 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

Currently Open Text Editor

Searches the open text file. If you have more than one text file open, 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.

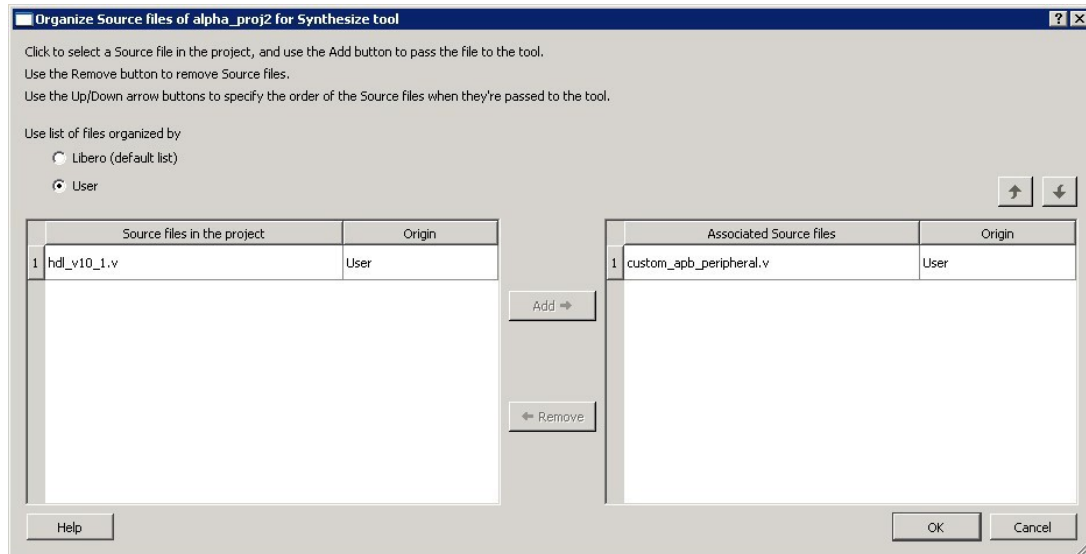
13.25. Organize Source Files Dialog Box – Synthesis [\(Ask a Question\)](#)

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 13-37. Organize Source Files Dialog Box

13.26. SmartDesign Testbench [\(Ask a Question\)](#)

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.

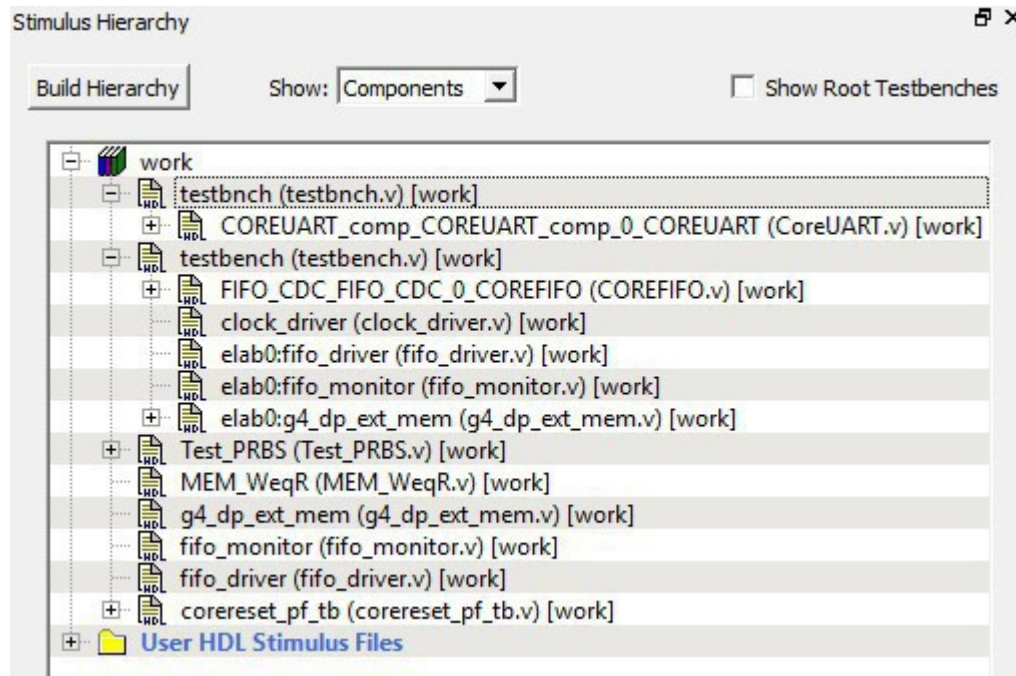
You can instantiate your own stimulus HDL or simulation models into the SmartDesign Testbench Canvas and connect them to your DUT (design under test). You can also instantiate Simulation Cores from the [Catalog](#). Simulation cores are simulation models (such as DDR memory simulation models) or basic cores that are useful for stimulus generation (such as Clock Generator, Pulse Generator, or Reset Generator). Click the Simulation Mode check box in the Catalog to view available simulation cores. Refer to the [SmartDesign User Guide](#) for more information.

13.27. Stimulus Hierarchy [\(Ask a Question\)](#)

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 13-38. 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 13-4. 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

Table 13-4. Design Hierarchy Icons (continued)

Icon	Description
	HDL netlist

13.28. Timing Exceptions Overview [\(Ask a Question\)](#)

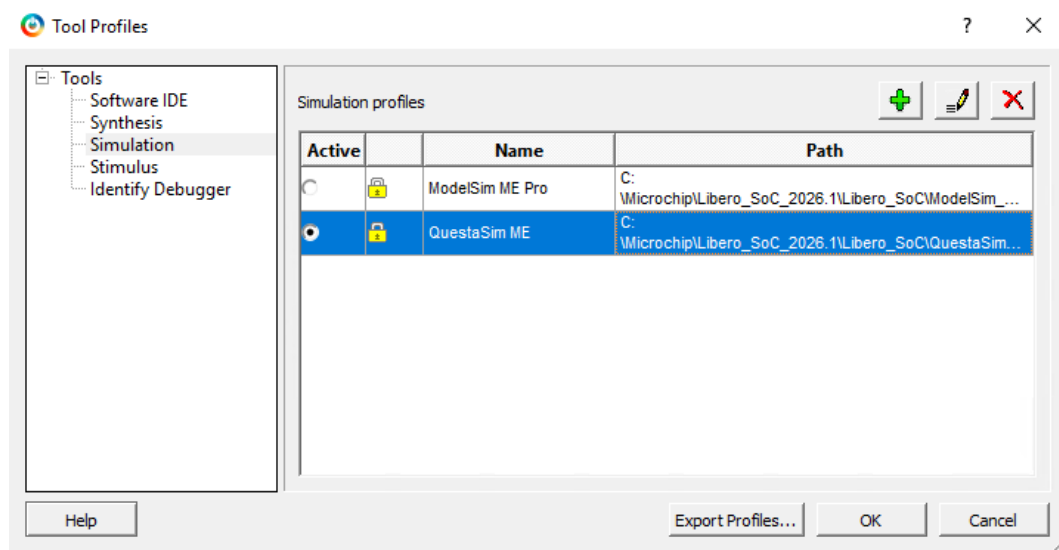
Use timing exceptions to overwrite the default behavior of the design path. Timing exceptions include:

- 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.

13.29. Admin Profile Tool Dialog Box [\(Ask a Question\)](#)

This dialog box provides a way to use the same set of tools and same vault location if Libero SoC is started from the same installed location. An admin (the person who generally takes care of Libero SoC installation) creates an admin profile. This profile is common to all Libero SoC users 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, start `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 13-39. 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
- QuestaSim ME Pro 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.

To create an 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 of Libero SoC users.

Using admin profile, you can 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.

13.30. Tool Profiles Dialog Box [\(Ask a Question\)](#)

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.

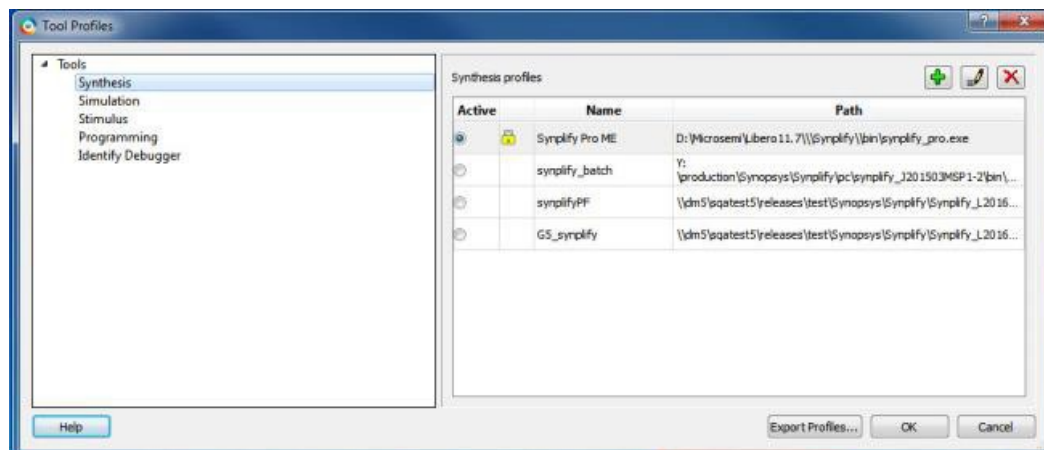
The following table shows the supported tool versions in this release.

Tool	Supported Version
QuestaSim® ME Pro	2025.3
SynplifyPro® ME	synplify_Y202603M
Identify ME	V202309M-2

To set or change your tool profile:

- 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 13-40. Libero SoC Tool Profiles Dialog Box



- When you are done, click **OK**.

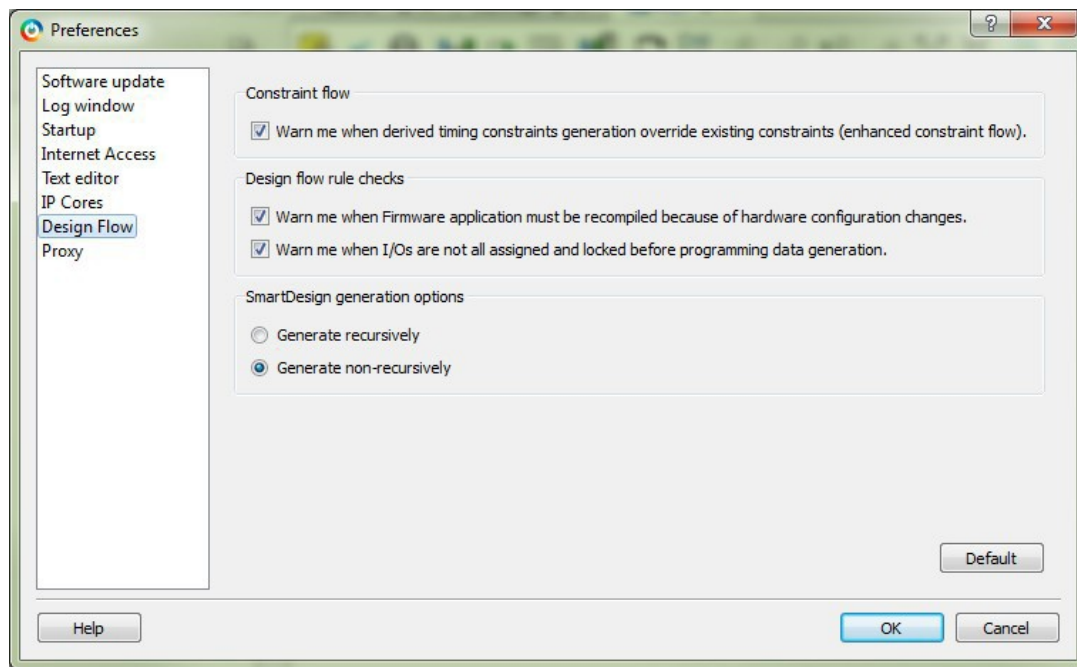
The tool profile with the padlock icon indicates that it is a pre-defined tool profile (the default tool that comes with the Libero SoC Installation).

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.

13.31. User Preferences Dialog Box – Design Flow Preferences [\(Ask a Question\)](#)

This dialog box allows you to set your personal preferences for how Libero SoC manages the design flow across the projects you create.

Figure 13-41. 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 must 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 sub-designs must be successfully generated before a parent can be generated. An attempt to generate a SmartDesign results in an automatic attempt to generate all sub-designs.
- 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.

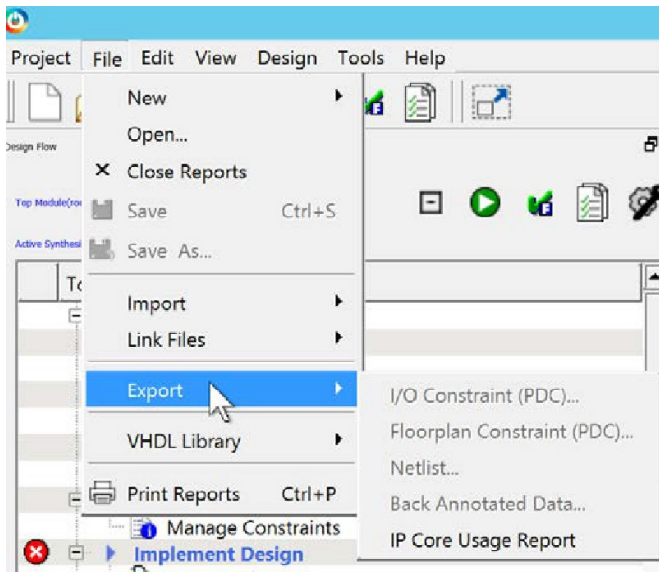
13.32. IP Report [\(Ask a Question\)](#)

Libero tool generates a report listing all the IPs used in the design project.

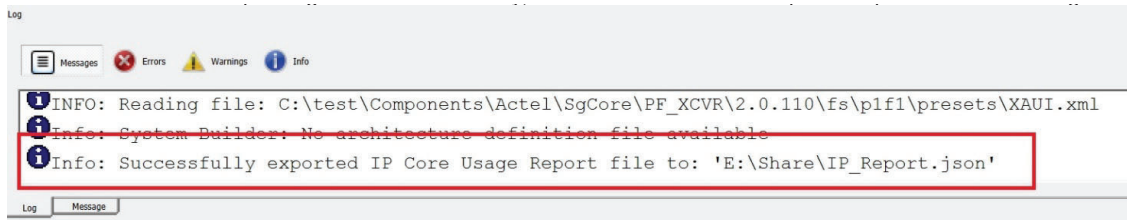
The report consists of the following:

- Project Information
- List of components in the project with the following 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 SmartDesign.
 - Component Name – In the form of "smartdesign_name>_<instance_name>"
 - Component State – State of the instance in the SmartDesign
 - 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 SystemBuilder components
- Multi Page SystemBuilder components without parameter list of latest core versions available for the cores used in the project. This list is present at the end of an open project.

To export the **IP CORE Usage Report**, go to **File > Export > IP Core Usage Report**. When clicked, a file dialog box appears to navigate to the location and file format to save the report. The file can be generated in either "JSON" or "txt" format.



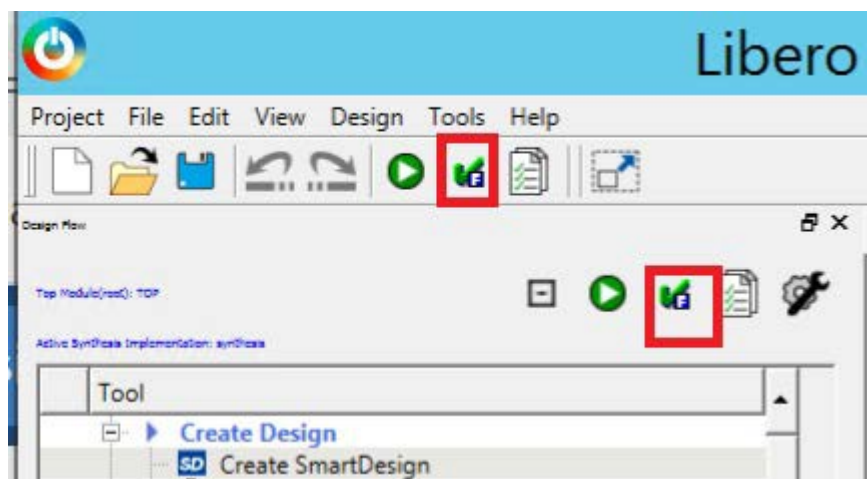
Once the report is generated successfully, the location to where the report is exported is shown in log window.



13.33. Force Update Design Flow [\(Ask a Question\)](#)

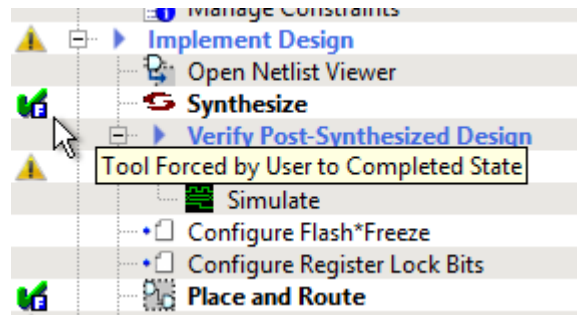
A global option to force update the design flow to Pass state is added to Libero SoC v12.5. The tool's state is force updated to PASS state only for Synthesize/Compile and Place and Route tools. The remaining tools remain in the same state they were in before the force update action was performed.

Figure 13-42. Force Update Action Button at Top of Design Flow Window



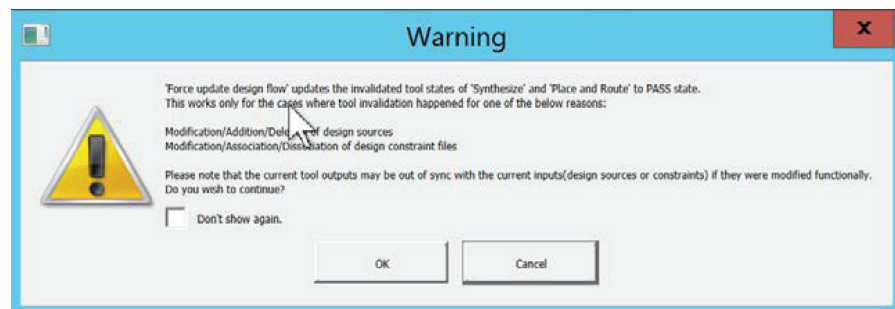
To differentiate between PASS state and FORCE UPDATE PASS state, the following icon and tooltip for the FORCE UPDATE PASS state (force update action) is added.

Figure 13-43. Force Update Action Icon with Tooltip



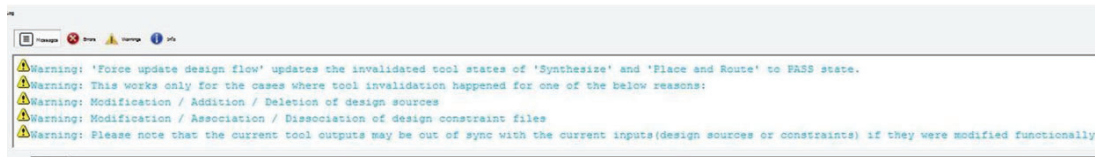
If you click the **Force update action** button, the following warning message appears. If you click **OK** in the message, the action proceeds. Otherwise, the action is canceled.

Figure 13-44. Warning Message Pop-up

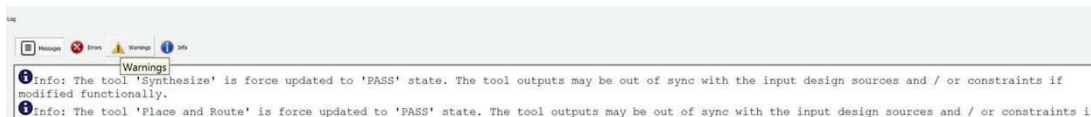


The message log window shows when the Force Update Action is performed.

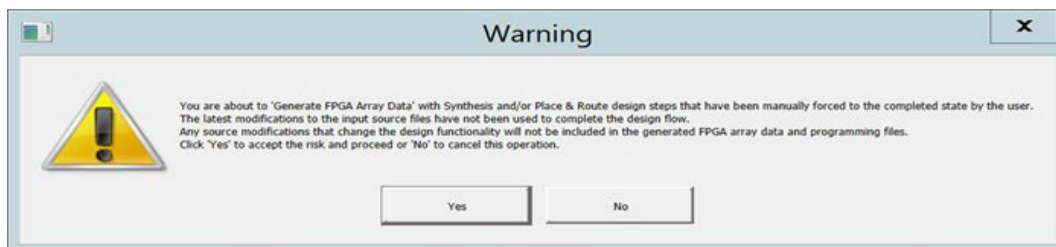
Figure 13-45. Warning Message in Log Window



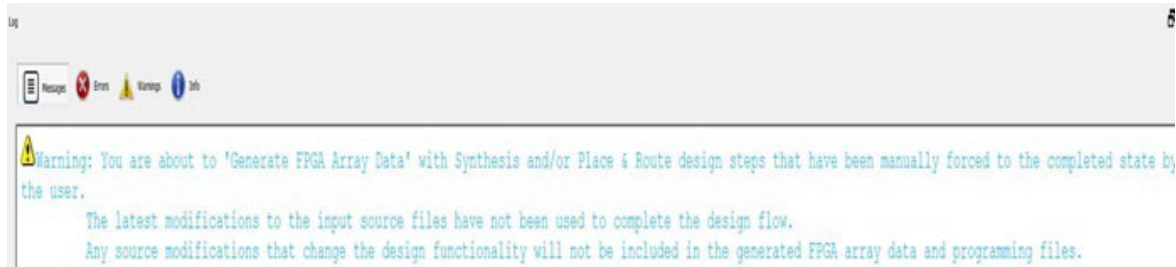
After the tool states are force updated to PASS state from Out of Date Design state, the following message appears in the log window.



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.



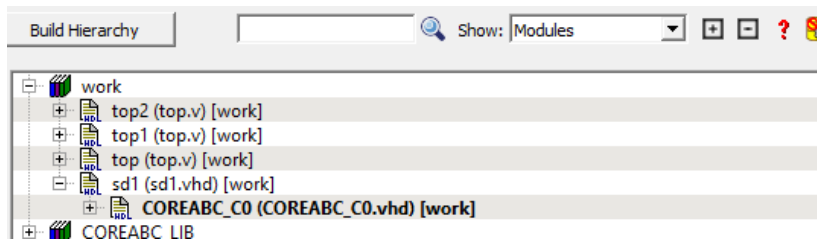
The log window shows the following message.



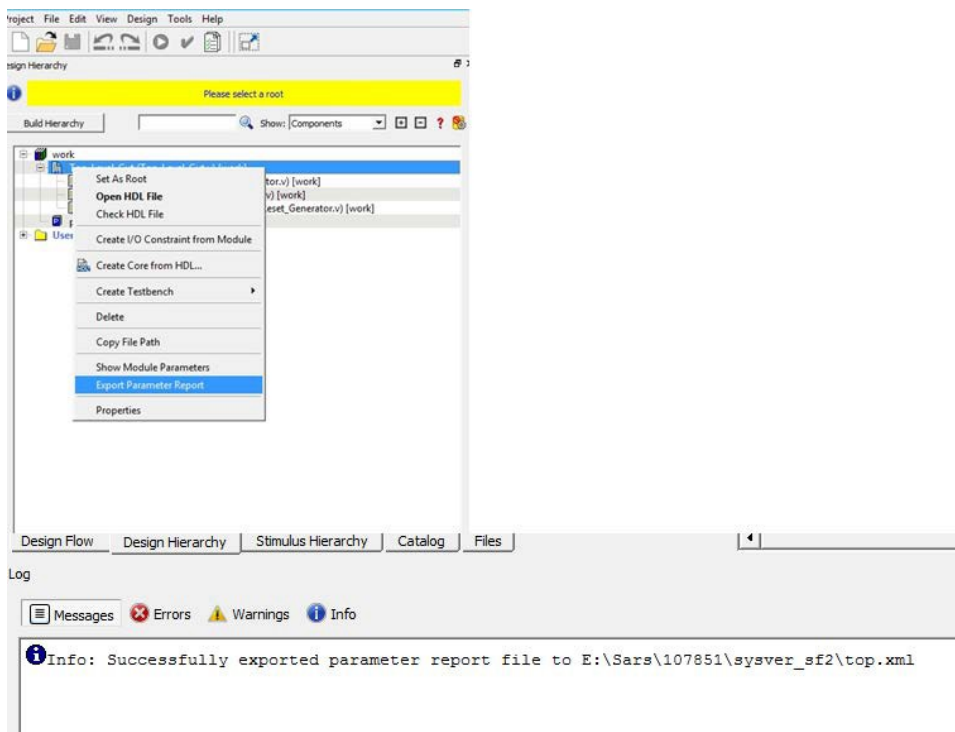
13.34. Generic/Parameter Report [\(Ask a Question\)](#)

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 the following 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.

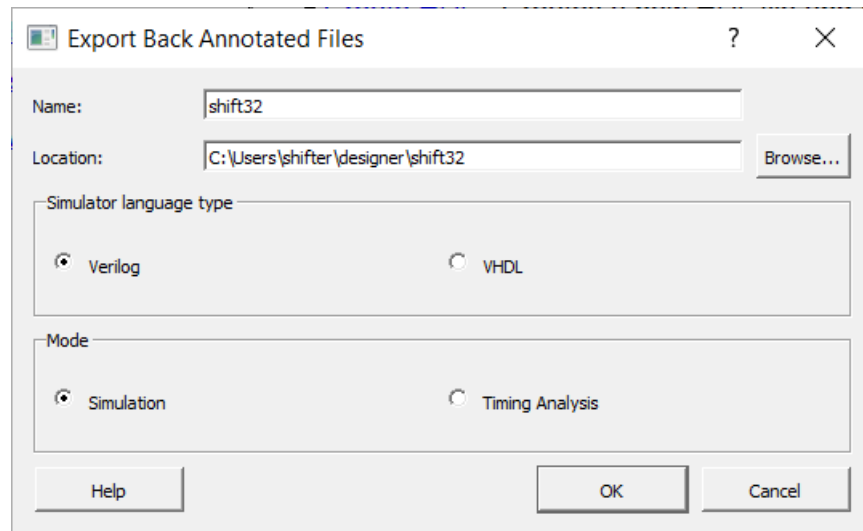


Error message is flagged in cases where the user tries to generate report for a module that is instantiated in another module.

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.

13.35. Export Back Annotated Files [\(Ask a Question\)](#)

Figure 13-46. Export Back Annotated Files



This feature can be accessed by going to **File > Export > Back Annotated Data**.

The following table describes the elements in the Export Back Annotated Files dialog box.

Table 13-5. Elements in the Export Back Annotated Files Dialog Box

Element	Description
Name	Name of the folder where the back annotated files will be exported.
Location	Location of the folder where the back annotated files will be exported.
Mode	Set the mode to Simulation or Timing Analysis.

Simulation Mode

This is the default mode. The tool generates three .sdf files and a netlist. The three .sdf files generated are {file_name}_fast_hv_lt_ba.sdf {file_name}_slow_lv_ht_ba.sdf and {file_name}_slow_lv_lt_ba.sdf. The netlist file generated is {filename}_ba.v.

Timing Analysis Mode

The tool generates three .sdf files, a netlist and a .tcl file. The .tcl file is a template on how to run a generated netlist and .sdf files in a PrimeTime shell. The three .sdf files generated are {file_name}_fast_hv_lt_sta_ba.sdf {file_name}_slow_lv_ht_sta_ba.sdf and {file_name}_slow_lv_lt_sta_ba.sdf. The netlist file generated is {filename}_sta_ba.v.

Limitations

The limitations of Export Back Annotated Files feature are:

1. Encrypted Verilog

This is a limitation in PrimeTime and because of that, Back Annotated tool in Timing Analysis mode will not support encrypted netlist. User should not use or include encrypted blocks in Timing Analysis mode as PrimeTime does not support encrypted netlists.

2. SDC File Option
User needs to populate user.sdc (needed by PrimeTime) using sdc file found under **constraint** directory.
3. Differences in clock reconvergence pessimism removal (CRPR)
When arrival and required common clock paths use different edges, SmartTime and PrimeTime have different values.

13.36. Absolute and Relative Path Support for MSS [\(Ask a Question\)](#)

This feature supports absolute and relative path in batch mode for PolarFireSOC MSS Configurator.

The format of the batch command is as follows:

```
pfsoc_mss.exe \
-CONFIGURATION_FILE:<path_to_file>/<file_name>.cfg \
-OUTPUT_DIR:<path_to_file>/MSS \
-EXPORT_HDL:false \
-LOGFILE:<path_to_file>/logfile.txt
```

Table 13-6. Arguments

Parameter	Type	Description
CONFIGURATION_FILE	string	Specifies the configuration file. The path to configuration (.cfg) file can be an absolute path or a relative path.
OUTPUT_DIR	string	Specifies the output directory location. The output directory can be an absolute path or a relative path.
EXPORT_HDL	string	Set to True or 1 to export the HDL files. Default is false or 0.
LOGFILE	string	Specifies the logfile. The path to log file can be an absolute path or relative path.

Example 13-1. Absolute Path

```
D:/dir_1/dir_2/dir_3/pfsoc_mss.exe \
-CONFIGURATION_FILE:./soft/sqadir/user/mss_configuration.cfg \
-OUTPUT_DIR:d:/soft/sqadir/user/MSS \
-EXPORT_HDL:false \
-LOGFILE:/soft/sqadir/user//logfile.txt
```

Example 13-2. Relative Path

```
D:/dir_1/dir_2/dir_3/pfsoc_mss.exe \
-CONFIGURATION_FILE:../../mss_configuration.cfg \
-OUTPUT_DIR:d:/MSS \
-EXPORT_HDL:false \
-LOGFILE:d:/logfile.txt
```

In this example, the .cfg file is expected to be present in the following absolute location:

```
<current_working_directory>/../../mss_configuration.cfg
```

And the design files are generated in the following output directory location:

```
<current_working_directory>/MSS
```

14. Revision History [\(Ask a Question\)](#)

Revision	Date	Description
S	06/2026	The following changes are made in this revision: - Updated the instances of ModelSim ME Pro to QuestaSim® ME Pro throughout the document - In PolarFire Device Settings section, updated Figure 3-3 - In SmartFusion 2 and IGLOO 2 Device Settings section, updated Figure 3-4 - In RTG4 Device Settings section, updated Figure 3-5 and the options in Table 3-7 - In Project Settings: Simulation - Options and Libraries section, updated step 3 and Figure 4-20 - Added VCOM Command section - Added VLOG Command section - In following sections, added the remote programming information: Working with Local and Remote Programmers Configuring and Managing Remote Servers Remote Programming Server (RPS) Installation Instructions Remote Programmer Troubleshooting - In Adding eNVM Clients section, updated Figure 8-34 - Added Adding a Plaintext Data Client in eNVM Memory section - In the RTG4 Programming Options section, updated the GUI behavior based on SCSM device settings and the OTP programming option - Added Interaction Between RTG4 SCSM and OTP section - In Export Bitstream (RTG4) section, updated Figure 10-17 and Figure 10-18 - Added Compressing DAT Files (RTG4) section - In Project Settings: Simulation - Options and Libraries section, added <code>vcom</code> and <code>vlog</code> commands and updated Figure 13-26 and Figure 13-27 - In Tool Profiles Dialog Box section, updated the QuestaSim and SynplifyPro® ME versions
R	12/2025	The following change is made in this revision: In SPI Flash Clients Table section, updated Figure 8-24. Added a note about the SPI bitstream client.

Revision History (continued)

Revision	Date	Description
Q	05/2025	The following change is made in this revision: • In Creating a Core from Your HDL section, updated step1. • In Resource Usage section: – Updated the Table 6-15. – Added description for all metrics. • In Overlapping of Resource Reporting section: – Updated the content. – Created General Formulas for Logic Element Utilization section. – Created Example Application of Logic Element Utilization section. • In Supported Dies and Packages section, added new die (RTPF500ZT) and package details in the table. • In Programming File Actions and Supported Procedures section, updated the note for VERIFY_DIGEST. • In Third Stage (uPROM/sNVM/SPI Flash) section, added a table that lists combinations of bitstream types and stage 3 client binding. • In Programming File Actions and Descriptions section, updated the descriptions for PROGRAM and VERIFY_DIGEST actions. • In Export a SPI Flash Image (PolarFire and PolarFire SoC) section: – Updated Figure 10-27. – Added the Formats option to the fields in the Table 10-7. • Added Formats section. • In Digest File section, added a cross reference to the FPGA Programmed Content Integrity Checking and Programming File Tracking using Cryptographic Digests white paper. • In Design Hierarchy in the Design Explorer section, added a note that the Stimulus Hierarchy might not appear correctly when a module from one HDL file instantiates another module defined in a different HDL file located in a separate library. • In Design Rules Check section, added a note about SSO limits. • In Project Settings Dialog Box section: – Updated Figure 13-25. – In Analysis Operating Conditions, added operating ranges for all available device families.

Revision History (continued)

Revision	Date	Description
P	09/2024	The following changes are made in this revision: - In section Supported Dies and Packages, added dies for the SmartFusion[®]2/IGLOO[®]2 and RTG4[™] families. - In section Programming File Actions and Descriptions: - For the PROGRAM action, added that the resulting PROGRAM action log contains information about component digests that are generated while programming the bitstream file. - For the VERIFY_DIGEST action, updated the description and added that the VERIFY_DIGEST also exports the freshly calculated digests besides checking against stored digests. - In section Configure SPI Flash Image Actions and Procedures, added a note that partial programming of the SPI flash is supported when using PolarFire[®]/PolarFire SoC FPGAs and the FlashPro6 programmer. - In section Exporting SmartDebug Data: - Added a note for the Include bitstream file to program at trusted facility field that if DPK is permanently disabled for PolarFire, UPK1 is used for unlocking the device. - Removed the statement that the field Include security components applies only to PolarFire families. - Removed the statement that the field Include bitstream file to program at trusted facility applies only to SmartFusion 2, IGLOO 2, and RTG4. - For the Include bitstream file to program at trusted facility field, revised the descriptions for options Sanitize all sNVM pages in ERASE action and Sanitize all eNVM pages in ERASE action. - In section Digest File, described the additional information contained in the digest file. - In section Tool Profiles Dialog Box, updated the supported versions of the tools are listed in the table.
N	02/2024	The following changes are made in this revision: - In section Global Nets (Promotions and Demotions), revised the descriptions for the options Minimum number of asynchronous pins and Minimum fanout of global nets that could be demoted to row-globals. Also, revised the default value for the option Number of global resources. - In section Exporting SmartDebug Data, replaced the figure of the Export SmartDebug Data dialog box and described the new field Include security components.
M	08/2023	The following change is made in this revision: In section Step 3: Debug Policy, listed the limitations that are applied when the Disable JTAG (1149.1) boundary scan option is selected.
L	05/2023	The following change is made in this revision: In section Exporting SmartDebug Data, for the Include bitstream file to program at trusted facility field, mentioned that if a user design has security configured, the Include Plaintext Pass Keys option is required.
K	04/2023	The following changes are made in this revision: - In section Verify Timing, updated the following figures: - Positive slack distribution and negative slack distribution - Cell vs. nets stacked bar chart (value and percentage views) and expanded path popup - Pie chart - Logic level distribution - In section Step 2: Update Policy, for Disable programming interfaces, described behavior when JTAG is checked and unchecked. - In section Debug Security Policy, updated the screen shot of the Debug Security Policy dialog box. - In section Device Configured with Custom Security Option in the Configure Security Tool (PolarFire and PolarFire SoC), added information about Include Plaintext Pass Key option.

Revision History (continued)

Revision	Date	Description
J	12/2022	The following changes are made in this revision: - Updated section Microchip License Utility. - In section Create SmartDesign, added a note about the direct cores that cannot be generated in Libero using Batch mode on Linux machines. - In section Verify Timing, described the corners analyzed for multi-corner analysis and noted that reports show only Multi Corner Timing Analysis reports show only two conditions. Also, described cell vs net delays option and logic level distribution. - In Step 2: Update Policy, added a note to the Disable Auto Programming and IAP Services option that if a PolarFire SoC user disables the Disable programming interfaces option, the One-way Passcode options are not available and SPI files cannot be exported. - In sections Configuring Security Locks (and One-Way Passcode for PolarFire SoC) for Production, Enable One-way Passcode, and Disable One-way Pass code, described the new Enable One-Way Passcode (OWP) option. - In section Exporting Bitstreams, added a note that PolarFire SoC users can use Export Bitstream to export either Fabric with sNVM or sNVM component only. Also, noted that PolarFire SoC no longer has advanced options for the master file; the Advanced Options dialog has been updated with High Water Mark options related to One-way Passcode. - In section Device Configured with Custom Security Option in the Configure Security Tool (PolarFire and PolarFire SoC), updated the PolarFire and PolarFire SoC Export Bitstream with Custom Security Dialog Box screen shots and the Advanced Options for File Encrypted with UEK1 screen shots. - In section Export FlashPro Express Job (PolarFire and PolarFire SoC), updated screen shots. Also, added a note that PolarFire SoC users can use Export FlashPro Express to export either Fabric with sNVM or sNVM component only. - In section Exporting Job Manager Data, updated the PolarFire screen shot. Also, added a note that PolarFire SoC users can use Export Job Manager to export either Fabric with sNVM or sNVM component only. - In section Downloading Firmware, described what to do if you have an Internet connection, but Libero IDE does not connect to the default repositories. - Where appropriate, instances of Microsemi have been replaced with Microchip.
I	08/2022	The following changes are made in this revision: - Added new section Automatic Clock Jitter Constraint. - Added PolarFire SoC to the devices mentioned in section Exporting Job Manager Data.
H	08/2022	The following changes are made in this revision: - Added guidelines for using SPI files with Libero v2022.2 and later versions in section Settings for Add SPI Bitstream Client. - Added a description for the option Include this client in eNVM digest calculation in sections Settings for Boot Mode 1 Clients and Settings for Boot Mode 3 Clients (PolarFire SoC). - Added table about the public key X and Y coordinates in section Settings for Boot Mode 3 Clients (PolarFire SoC). - Added a note that using a SPI file generated with Libero v2022.2 or later version with custom security to create or edit the client displays an error message that the SPI file cannot contain security in section Configuring Security. - Listed the direct cores that cannot be used in Libero SoC using batch mode on Linux machines via Tcl when the DISPLAY variable is not set in section Running Programming Device Actions. - In section Exporting Job Manager Data, described the new option Bypass Black Level protection for Recovery/Golden bitstream.
G	05/2022	Enabled 'Ask A Question' hyperlink for each section in the document.

Revision History (continued)

Revision	Date	Description
F	04/2022	The following changes are made in this revision: - In section Supported Dies and Packages, added part FCG784 to the packages shown. - In section Programming Connectivity and Interface, added note about updating out-of-date FlashPro6 programmers. - In section Programming Connectivity and Interface, added note about a pop-up window appearing during program action in the Program Connectivity and Interface when an out-of-date FlashPro6 programmer is detected. - Updated Step 2: Update Policy. - Updated Configuring Security Locks (and One-Way Passcode for PolarFire SoC) for Production. - Updated Verify Timing.
E	12/2021	The following changes are made in this revision: - Updated sections HDL Testbench and Create a New SmartDesign Testbench. - Updated section Import MSS. - In section Derived Constraints, added a tip to use IP configurators to generate design components when applicable for constraints to be derived accurately. - Updated section Settings for Add Data Storage Client. - Updated section Configuring I/O States During JTAG Programming. - Added the new section Export Initialization Data and Memory Report (PolarFire and PolarFire SoC). - Added the new section Absolute and Relative Path Support for MSS.
D	9/2021	The following changes are made in this revision: - Updated the screen shots in section Synthesize Options. - Added new section Active Implementation.
C	8/2021	The following changes are made in this revision: - Consolidated the Design Flow Guide by adding content for SmartFusion 2, IGLOO 2, and RTG4, and then identified which content applies to which device families. Implementing Designs: added note about how Libero uses multiple cores automatically. Programming Connectivity and Interface: added icons to the table of programming connectivity and interface options. Settings for Add SPI Bitstream Client and STAGE3 Initialization Clients: documented the new STAGE3 initialization client feature.
B	4/2021	The following changes are made in this revision: Viewing Configured Components and SmartDesigns in a Project: added this new topic. Global Nets (Promotions and Demotions): added Clock Domain Crossing and effects of synchronizer. Post Layout Editing of I/O Signal Integrity and Delay Parameters: added this new topic. Initializing Design Blocks (PolarFire and PolarFire SoC): reorganized and described Boot Mode 1 and Boot Mode 3. Configuring Security: added GUI screens and greater detail about security options. Update Policy: updated this section. Partial Programming Support: added this new section. Export Bitstream: added Advanced Options. References: removed commands. Project Settings: Design Flow: added Multi-File Compilation Unit. Force Update Design Flow: added this new section.
A	11/2020	Initial Revision

Microchip FPGA Support

Microchip FPGA products group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, and worldwide sales offices. Customers are suggested to visit Microchip online resources prior to contacting support as it is very likely that their queries have been already answered.

Contact Technical Support Center through the website at www.microchip.com/support. Mention the FPGA Device Part number, select appropriate case category, and upload design files while creating a technical support case.

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

- From North America, call **800.262.1060**
- From the rest of the world, call **650.318.4460**
- Fax, from anywhere in the world, **650.318.8044**

Microchip Information

Trademarks

The “Microchip” name and logo, the “M” logo, and other names, logos, and brands are registered and unregistered trademarks of Microchip Technology Incorporated or its affiliates and/or subsidiaries in the United States and/or other countries (“Microchip Trademarks”). Information regarding Microchip Trademarks can be found at <https://www.microchip.com/en-us/about/legal-information/microchip-trademarks>.

Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications 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. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

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.

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip products:

- 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, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip products are strictly prohibited and may violate the Digital Millennium Copyright Act.
- 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. Microchip is committed to continuously improving the code protection features of our products.