NOTE: PDF files are intended to be viewed on the printed page; links and cross-references in this PDF file may point to external files and generate an error when clicked. View the online help included with software to enable all linked content.
# Table of Contents

**Libero SoC Introduction** .................................................................................................................. 4
- Welcome to Microsemi Libero® SoC v12.5......................................................................................... 4
- Licensing and Additional Resources ............................................................................................. 4
- Libero SoC PolarFire Design Flow ................................................................................................. 13
- Constraint Flow and Design Sources ............................................................................................ 16
- File Types in Libero SoC ................................................................................................................ 18
- Software Tools - Libero SoC ......................................................................................................... 19

**Libero Design Flow** ...................................................................................................................... 20
- Starting the Libero GUI .................................................................................................................. 20
- Design Report ................................................................................................................................ 21
- Using the New Project Wizard to Start a Project ............................................................................ 22

**Create and Verify Design** ............................................................................................................ 29
- Create SmartDesign.......................................................................................................................... 29
- Create Core from HDL .................................................................................................................... 37
- Designing with HDL ....................................................................................................................... 38
- Designing with Block Flow ............................................................................................................. 40
- Create New SmartDesign Testbench ............................................................................................. 40
- HDL Testbench .............................................................................................................................. 41
- Import MSS ................................................................................................................................. 43
- Verify Pre-Synthesized Design - RTL Simulation .......................................................................... 43

**Libero SoC Constraint Management** .......................................................................................... 47
- Invocation of Constraint Manager from the Design Flow Window .............................................. 47
- Libero SoC Design Flow ................................................................................................................ 47
- Introduction to Constraint Manager ............................................................................................ 49
- Import a Constraint File ................................................................................................................. 52
- Constraint Types ............................................................................................................................ 57
- Constraint Manager – I/O Attributes Tab ....................................................................................... 58
- Constraint Manager – Timing Tab .................................................................................................. 60
- Derived Constraints ....................................................................................................................... 63
- Constraint Manager – Floor Planner Tab ....................................................................................... 63
- Constraint Manager – Netlist Attributes Tab ................................................................................ 65

**Implement Design** ........................................................................................................................ 67
- Synthesize ...................................................................................................................................... 67
- Verify Post-Synthesized Design ..................................................................................................... 71
- Compile Netlist ............................................................................................................................... 72
- Constraint Flow in Implementation ............................................................................................... 73
- Place and Route ............................................................................................................................. 78
- Multiple Pass Layout Configuration .............................................................................................. 82
Resource Usage .................................................................................................................................. 84
Global Net Report ............................................................................................................................. 86
Verify Post Layout Implementation .................................................................................................. 90

Configure Hardware.......................................................................................................................... 102
Programming Connectivity and Interface ....................................................................................... 102
Select and Configure Actions and Procedures ................................................................................ 104
Programmer Settings ......................................................................................................................... 104
Select Programmer ............................................................................................................................ 106

Program Design ............................................................................................................................... 107
Generate FPGA Array Data ................................................................................................................ 107
Configure Design Initialization Data and Memories .......................................................................... 107
Generate Design Initialization Data ..................................................................................................... 128
Configure Actions and Procedures ..................................................................................................... 129
Configure I/O States During JTAG Programming ............................................................................ 133
Configure Programming Options ....................................................................................................... 134
Configure Security .............................................................................................................................. 134
Configure Permanent Locks for Production ..................................................................................... 142
Configure Bitstream .......................................................................................................................... 144
Run Programming Device Actions ................................................................................................... 145
Program SPI Flash Image .................................................................................................................. 152

Debug Design ..................................................................................................................................... 156
Generate SmartDebug FPGA Array Data .......................................................................................... 156
SmartDebug ....................................................................................................................................... 156
Identify Debug Design ........................................................................................................................ 157

Handoff Design for Production ........................................................................................................ 158
Export Bitstream ............................................................................................................................... 158
Export FlashPro Express Job ............................................................................................................ 167
Configure Actions and Procedures .................................................................................................... 170
Export SPI Flash Image ...................................................................................................................... 171
Export Pin Report ............................................................................................................................... 172
Export BSDL File ............................................................................................................................... 171
Export IBIS Model ............................................................................................................................ 171

Export SmartDebug Data (Libero SoC) ............................................................................................. 174

References .......................................................................................................................................... 176
Libero SoC Introduction

Welcome to Microsemi Libero® SoC v12.5

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

More Information

To access datasheets and silicon user guides, visit www.microsemi.com, select the relevant product family and click the Documentation tab. Tutorials, Application Notes, Development Kits & Boards are listed in the Design Resources tab.

Click the following links for additional information:

- Libero SoC – Learn more about Libero SoC including Release Notes, a complete list of devices/packages, and timing and power versions supported in this release.
- Programming – Learn more about Programming Solutions
- Power Calculators – Find XLS-based estimators for device families
- Licensing – Learn more about Libero licensing

Licensing and Additional Resources

Microsemi License Utility

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

To start the Microsemi License Utility, run it from Start > All Programs > Microsemi Libero SoC vx.xx > Microsemi License Utility.

To request a license, click Request License to go to the Microsemi license website. You can select and copy (right-click, Copy) the disk volume value displayed in the window and paste the value into the Microsemi license web form.

The following licenses are available:

- 1-year Platinum - Purchased license that supports all devices
- 1-year Gold - Purchased licenses 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 programming is disabled
When you have received your license file, follow the instructions and save the license to your local disk. In the Microsemi License Utility window, click **Add License File** and browse/select the license file from your disk. If you are using a floating license, click **Add License Server** and enter the Port Number and Name of the license server host.

The list of features for which you are licensed will show all versions, but your license must have a version equal to or greater than your design tools release version in order 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, with the first file read at the top of the list.

Click **Write Report File** to view and/or print the Microsemi Tools Licenses Report, or to save it as a TXT file.

Figure 1 · Microsemi License Utility

The [Microsemi Libero SoC License Information Web Page](#) provides additional licensing-related information including links to troubleshooting and FAQ documents.

License Selection

This topic describes the Select License and License Details dialog boxes.

Select License Dialog Box

The Select License dialog box appears when there is more than one license available and a default license has not been selected. Available licenses are displayed in list format, and you can select the feature license you want to use.
If a license will expire within 15 days, you will see a warning in the Comments field of the dialog box and also in the Log window, as shown in the examples below.

Actions

Click Select to use the selected license to invoke Libero. This button is disabled by default, and is enabled once a License is selected.

Click Cancel to close the license selection window and exit Libero.

Click Help to open the online help topic for License Selection.

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. This option can be used if you want to use the same license features for future sessions.

License Types

The License Type column indicates whether an available license is a Node Locked license or a Floating or Server-based license.

The total number of licenses and number of available licenses are shown in the Quantity and Available columns, respectively.

Floating or Server-based Licenses can be used by multiple users, depending on the number of seats available.
If there are no valid licenses available, the following message appears:

![Message](image)

If a selected license has been acquired by another user, you will be notified with the following message:

![Message](image)

**Additional License Information**

**Set Default License After Libero Invocation**

This Libero Help menu option sets/updates the default Libero license and allows it to be used for future sessions. To set the default license after Libero invocation, click **Select Default License**, as shown below.
The Select License dialog box appears, as shown in the example below.

Actions

Click **Set Default License** to set the default Libero license. You can also double-click the license row to set the default license. This button is enabled if you select any row in the list of licenses.

Click **Cancel** to close the default license selection window without performing any action.

Click **Help** to open the online help document for License Selection.

**Set License Options in Libero Preferences Dialog Box**

To set license options in the Libero Preferences dialog box, click **Project > Preferences**.
Figure 5 · License Options in Libero Preferences Dialog Box

**Reset Default License Selection**: This check box is enabled only when there is a default license available. When this check box is checked, the default license will be cleared and the check box will be disabled.

**“Warn we when my Libero license is about to expire”**: This check box enables/disables Notification of License expiry. If the selected license expiry date is within 15 days, a popup message appears. This option can be used only when the expiry date is more than 5 days and less than 15 days.

**See Also**
[License Details](#)

**License Details Dialog Box**

The License Details dialog box has two tabs:

- **Tools**
- **IP Cores**

**Tools**
The Tools tab displays tool license details.

**IP Cores**
The IP Cores tab displays IP Cores license details, as shown in the following example.
Figure 6 · IP Cores License Details

Actions

Click **Close** to close the current dialog box.
Click **Help** to open the online help topic for License Selection.
Click **Filter** to search for the pattern specified in the text edit box. Filtered rows will be shown in the Cores table.

**Lookup Path**: Shows the list of License hosts included in the LM_LICENSE_FILE.

**See Also**

- License Selection
- General Information about Microsemi's FPGA & SoC products is available here.

**Information About PolarFire Technology**

More information about PolarFire Technology is available here.

**Information About Supported Families:**

Table 1 · Product Families and Derivatives

<table>
<thead>
<tr>
<th>Device Family</th>
<th>Family Derivatives</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PolarFire</td>
<td>N/A</td>
<td>FPGA family that delivers the lowest power at mid-range densities, and is optimized for cost while providing high end features.</td>
</tr>
</tbody>
</table>

**Information About Libero SoC PolarFire Software**

More information about Libero SoC PolarFire Software is available here.

**Application Notes and Tutorials**

Application Notes and Tutorials for PolarFire can be found here on the Documentation tab of the PolarFire Technology page.

**Online Help - Libero SoC**

This 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 it with Microsoft's Internet Explorer browser (use version 4.x or later for complete functionality).
Viewing HTML Files on Linux

You may need to set your LINUX_HTMLREADER variable such that it enables a HTML viewer. For example:

```
setenv LINUX_HTMLREADER /usr/bin/firefox
```

If you do not set this variable then some HTML files, such as the help, will not be available from within software.

See Also
- Navigation tabs
- User's Guides

Using Navigation tabs

Libero SoC online Help, which is generated using Microsoft HTML Help, includes the following navigation tabs:

**Contents**
The Contents tab displays 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 (sub-books and pages). When you click an open book, it closes. When you click pages, you select topics to view in the right-hand pane of the HTML Help viewer.

**Search**
The Search tab enables you to search for words in the Help system and locate topics containing those words. Full-text searching looks through every word in the online Help to find matches. When the search is completed, a list of topics is displayed so you can select a specific topic to view.

Reading User Guides

Libero SoC includes online manuals. The online manuals are in PDF format and available from Libero SoC Start Menu. Note that PDF files are for printing and viewing offline; use the online help to view user support on your workstation.

From the Start menu, choose **All Programs > Microsemi > Libero SoC > Libero SoC Reference Manuals**. You must have Adobe Acrobat Reader or similar PDF viewer to open and view the PDF user guides.

**Viewing PDF Files on Linux**

You may need to set your LINUX_PDFREADER variable such that it enables a PDF viewer. For example:

```
setenv LINUX_PDFREADER /usr/bin/kpdf
```

If you do not set this variable, some PDF files, such as the SmartTime User's Guide, will not be available from within software.

Microsemi SoC Products Group Headquarters

Microsemi Corporation is a supplier of innovative programmable logic solutions, including field-programmable gate arrays (FPGAs) based on antifuse and flash technologies, high-performance intellectual property (IP) cores, software development tools, and design services targeted for the high-speed communications, application-specific integrated circuit (ASIC) replacement, and radiation-tolerant markets.

<table>
<thead>
<tr>
<th>Address:</th>
<th>Microsemi SoC Products Group</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3870 North First Street</td>
</tr>
<tr>
<td></td>
<td>San Jose, CA 95134</td>
</tr>
</tbody>
</table>

| Phone:         | 408-643-6000               |
Contact Information

For the most up-to-date contact information, check the Microsemi Home Page.
Contact information for FPGAs & SoCs can be found at the FPGAs and SoCs Support Page.
If you do not have internet access, the following information was accurate at the time of publication:

- Technical Support
  - Web: https://soc.microsemi.com/mycases
  - Phone (NA): 800.262.1060
  - Phone (Int'l): +1 650.313.4460
  - Email: soc_tech@microsemi.com

- Customer (non-technical) Support
  - Phone: +1 650.318.2470
  - Email: customer.service@microsemi.com

- Sales Support
  - For pricing, order status and lead time information for all Microsemi SoC products, contact your Microsemi Sales Representative

- Technical Support for RH and RT FPGAs that are regulated by International Traffic in Arms Regulations (ITAR)
  - Phone (NA): 888.988.ITAR
  - Phone (Int'l): +1 650.318.4900
  - Email: soc_tech_itar@microsemi.com
Libero SoC PolarFire Design Flow

Create Design

Create your design with any or all of the following design capture tools:

- Create SmartDesign
- Create HDL
- Create SmartDesign Testbench (optional, for simulation only)
- Create HDL Testbench (optional, for simulation only)
Once the design is created, you can invoke simulation for pre-synthesis verification.

It is also possible to click the button, to execute the Libero SoC software through Place and Route with default settings. However this bypasses constraint management.

**Constraints**

- **Manage Constraints**

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

Libero SoC provides the Constraint Manager as the cockpit to manage your design constraint needs. This is a single centralized graphical interface for you to create, import, link, check, delete, 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.

**Invocation of Constraint Manager From the Design Flow Window**

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

[Figure 8 · Constraint Manager]

- **See Also**
  - Constraint Manager
  - New Project Wizard to import/link design constraints when creating new projects

**Implement**

- **Open Netlist Viewer (User Guide)**
- **Synthesize** - Double-click Synthesize to run synthesis on your design with the default settings. The constraints associated with Synthesis in the Constraint Manager are passed to Synplify.
- **Verify Post-Synthesis Implementation (Simulate)**
PolarFire FPGA Design Flow User Guide

- **Place and Route** - Place and Route takes the design constraints from the Constraint Manager and runs with default settings. This is the last step in the push-button design flow execution.
- **Verify Post Layout Implementation**
  - Generate Back Annotated Files
  - Verify Timing - Right click and select Configure Options to specify a timing report with your desired conditions.
  - Open SmartTime
  - Verify Power
  - Open SN Analyzer

**Configure Hardware**
- **Programming Connectivity and Interface** - Organizes your programmer(s) and devices.
- **Configure Programmer** - Opens your programmer settings; use if you wish to program using settings other than default.
- **Select Programmer**
- **Device I/O States During Programming** - Sets your device I/O states during programming; use if your design requires that you change the default I/O states.

**Program Design**
- **Generate FPGA Array Data**
- **Configure Design and Memory Initialization**
- **Configure I/O States During JTAG Programming**
- **Configure Programming Options**
- **Configure Security Wizard**
- **Configure Permanent Locks (OTP)**
- **Generate Bitstream**
- **Run PROGRAM Action**
- **Program SPI Flash Image**

**Debug Design**
- **SmartDebug (User Guide)**
- **Identify Debug Design**

**Handoff Design for Production**
- **Export Bitstream**
- **Export SPI Flash Image**
- **Export FlashPro Express Job**
- **Export Job Manager Data**
- **Export Pin Report**
- **Export BSDL**
- **Export IBIS Model**

**Handoff Design for Debugging (Export SmartDebug Data)**
Constraint Flow and Design Sources

The Constraint Flow supports HDL and Netlist design sources. The Libero SoC Design Flow window and the Constraint Manager are context-sensitive to the type of design sources: HDL or Netlist.

Constraint Flow for HDL designs

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.

Constraint Flow for EDIF designs

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 HDL flow versus the Netlist Flow is compared and contrasted below.

<table>
<thead>
<tr>
<th>HDL Flow</th>
<th>Netlist Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Design Flow Window</strong></td>
<td><strong>Design Flow Window</strong></td>
</tr>
<tr>
<td><strong>Constraint Manager</strong></td>
<td><strong>Constraint Manager</strong></td>
</tr>
</tbody>
</table>

HDL Flow

- Top Module (root): shift32
- Active Synthesis Implementation: synthesis
- Create Design
- Constraints
- Manage Constraints
- Implement Design
  - Open Netlist Viewer
  - Synthesize
- Verify Post Synthesized Design
- Place and Route
- Verify Post Layout Implementation
- Configure Design
- Debug Design
- Handoff Design for Production
- Handoff Design for Debugging

Netlist Flow

- Top Module (root): shift32
- Active Synthesis Implementation: synthesis
- Create Design
- Constraints
- Implement Design
  - Open Netlist Viewer
  - Verify Post Synthesized Design
- Compile Netlist
  - Place and Route
  - Verify Post Layout Implementation
- Configure Hardware
- Program Design
- Debug Design
- Handoff Design for Production
- Handoff Design for Debugging
### HDL Flow

- Constraint Manager - Check *.fdc and *.ndc

### Netlist Flow

- Constraint Manager - Check *.ndc only

![HDL vs. Netlist Flow](image)

**Figure 9 · HDL vs. Netlist Flow**

**Synthesize Options**

- Global Nets
  - Minimum number of clock pins:
  - Minimum number of asynchronous pins:
  - Minimum fanout of non-clock nets to be kept on globals:
  - Number of global resources:
  - Maximum number of global nets that could be demoted to row-global:
  - Minimum fanout of global nets that could be demoted to row-global:

- Optimizations
  - Enable retiming
  - Enable automatic compile point

- RAM optimized for:
  - High

- Map seq-shift register components to:
  - Register

- Map ROM components to:
  - Logic

**Compile Netlist Options**

- Global Promotion
  - Number of global resources:
  - Maximum number of global nets that could be demoted to row-global:
  - Minimum fanout of global nets that could be demoted to row-global:
  - Minimum fanout of non-clock nets to be kept on globals:

**Global Promotion/Demotion Options set in**

- Compile Netlist Options Dialog Box

**Global Promotion/Demotion Options set in**

- Synthesis Options Dialog Box
File Types in Libero SoC

When you create a new project in Libero SoC it automatically creates new directories and project files. Your project directory contains all of 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 enables you to manage your files as you import them. If you want to store and maintain your design source files and design constraint files in a central location outside the Project location, Libero gives you the option to link them to your Libero project folders when you first create your project. These linked files are not copied but rather linked to your project folder.

Depending on your project preferences and the version of Libero SoC you installed, the software creates directories for your project.

The top level directory (<project_name>) contains your *.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 double-click the *.prjx file to open the project with Libero SoC.

**component** directory - Stores your SmartDesign components (SDB and CXF files) and the *._manifest.txt file for each design components in your Libero SoC project. Refer to the *._manifest.txt file if you want to run synthesis, simulation, and firmware development with your own point tools outside the Libero SoC environment. For each design component, Libero SoC generates a <component_name>_manifest.txt file which 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.

**Note:** While importing components, it is expected to have both .sdb and .cxf files in same location directory.

**constraint** directory - All your constraint files (SDC timing constraint files, floorplanning PDC files, I/O PDC files, Netlist Attributes NDC files)

**designer** directory - *_.ba.sdf, *_.ba.v(hd), STP, PRB (for Silicon Explorer), TCL (used to run designer), impl.prj_des (local project file relative to revision), designer.log (logfile)

**hdl** directory - all hdl sources. *_.vhd if VHDL, *_.v and *_.h if Verilog

**simulation** directory - meminit.dat, modelsim.ini filesfiles and *_.vec file, run.do file for simulation.

**smartgen** directory - GEN files and LOG files from generated cores

**stimulus** directory - BTIM, Verilog, and VHDL stimulus files

**synthesis** directory - *.vm (Verilog Netlist output), *_.syn.prj (Synplify log file), *.psp (Precision project file), *.srr (Synplify logfile), precision.log (Precision logfile), *.tcl (used to run synthesis) and many other files generated by the tools (not managed by Libero SoC)

**viewdraw** directory - viewdraw.ini files

Internal Files

Libero SoC generates the following internal files. They may or may not be encrypted. They are for Libero SoC housekeeping and are not for users.

<table>
<thead>
<tr>
<th>File</th>
<th>File Extension</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Routing Segmentation File</td>
<td>*.seg</td>
<td></td>
</tr>
<tr>
<td>Combiner Info</td>
<td>*.cob</td>
<td></td>
</tr>
<tr>
<td>Hierarchical Netlist</td>
<td>*.adl</td>
<td></td>
</tr>
<tr>
<td>Flattened Netlist</td>
<td>*.afl</td>
<td></td>
</tr>
<tr>
<td>map file</td>
<td>*.map</td>
<td>Fabric Programming File</td>
</tr>
</tbody>
</table>
Software Tools - Libero SoC

The Libero SoC integrates design tools, streamlines your design flow, manages design and log files, and passes design data between tools. For more information on Libero SoC tools, visit: https://www.microsemi.com/products/fpga-soc/design-resources/design-software/libero-soc#overview

<table>
<thead>
<tr>
<th>Function</th>
<th>Tool</th>
<th>Company</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Manager, HDL Editor, Core Generation</td>
<td>Libero SoC</td>
<td>Microsemi SoC</td>
</tr>
<tr>
<td>Synthesis</td>
<td>Synplify® Pro ME</td>
<td>Synopsys</td>
</tr>
<tr>
<td>Simulation</td>
<td>ModelSim® ME Pro</td>
<td>Mentor Graphics</td>
</tr>
<tr>
<td>Timing/Constraints, Power Analysis, Netlist Viewer, Floorplanning, Package Editing, Place-and-Route, Debugging</td>
<td>Libero SoC</td>
<td>Microsemi SoC</td>
</tr>
</tbody>
</table>

**Project Manager, HDL Editor** targets the creation of HDL code. HDL Editor supports VHDL and Verilog with color, highlighting keywords for both HDL languages.  
**Synplify Pro ME** from Synopsys is integrated as part of the design package, enabling designers to target HDL code to specific devices.  
**Microsemi SoC** software package includes:
- **Chip Planner** 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

**ModelSim ME Pro** from Mentor Graphics enables source level verification so designers can verify HDL code line by line. Designers can perform simulation at all levels: behavioral (or pre-synthesis), structural (or post-synthesis), dynamic simulation. (ModelSim ME Pro is supported in Libero Gold and Platinum only.)
Libero Design Flow

Starting the Libero GUI

When starting Libero SoC GUI, the user will be presented with the option of either creating a new project, or opening an old one.

Clicking on **Open** ... opens a pre-existing Libero SoC project.

Clicking on **New**... starts the **New Project Wizard**. Upon completion of the wizard, a new Libero SoC project is created and opened.

Having opened a project, the Libero SoC GUI presents a Design Flow window on the left hand side, a log and message window at the bottom, and project information windows on the right. Below we see the GUI of a newly created project with only the top level Design Flow Window steps visible.
The Design Flow Window

The Design Flow Window for each technology family may be slightly different. The Constraint Flow choice made during new project creation may also affect the exact elements of design flow. However, all flows include some version of the following design steps:

- Create
- Constrain
- Implement
- Configure Hardware
- Program Design
- Debug Design
- Handoff

Design Report

The Design Report Tab lists all the reports available for your design, and displays the selected report. Reports are added automatically as you move through design development. For example, Timing reports are added when you run timing analysis on your design. The reports are updated each time you run timing analysis. If the Report Tab is not visible, you can expose it at any time by clicking on the main menu item Design > Reports.

If a report is not yet listed, you may have to create it manually. For example, you must invoke Verify Power manually before its report will be available.

Reports for the following steps are available for viewing here:

- Project Summary
- Synthesize
- Place and Route
- Verify Timing
- Verify Power
- Programming
  - Generate FPGA Array Data
  - Generate Bitstream
- Export
  - Export Pin Report
  - Export BSDL File
Using the New Project Wizard to Start a Project

New Project Creation Wizard – Project Details

You can create a Libero SoC project using the New Project Creation Wizard. You can use the pages in the wizard to:

- Specify the project name and location
- Select the device family and parts
- Set the I/O standards
- Import HDL source files and/or design constraint files into your project

![Libero SoC New Project Creation Wizard](image)

Project

- **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. The information entered appears in your Datasheet Report View.
- **Preferred HDL type** - Sets your HDL type: Verilog or VHDL. Libero-generated files (SmartDesigns, SmartGen cores, etc.) are created in your specified HDL type. Libero SoC supports mixed-HDL designs.
- **Enable Block Creation** - Enables you to build blocks for your design. These blocks can be assembled in other designs, and may have already completed Layout and been optimized for timing and power performance for a specific Microsemi device. Once optimized, the same block or blocks can be used in multiple designs.

When you are finished, click **Next** to proceed to the **Device Selection** page.

See Also

- New Project Creation Wizard - DeviceSelection
- New Project Creation Wizard – DeviceSettings
- New Project Creation Wizard – Add HDL SourceFiles
- New Project Creation Wizard - Add Constraints
New Project Creation Wizard – Device Selection

The Device Selection page is where you specify the Microsemi 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 page contains a table of all parts with associated FPGA resource details generated as a result of a value entered in a filter.

When a value is selected for a filter:

- 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, when PolarFire is selected in the Family filter:

- The parts table includes only PolarFire parts.
- The Die filter includes only PolarFire dies in the drop-down list for Die.

**Family** – Specify the Microsemi device family. Only devices belonging to the family are listed in the parts table.

**Die / Package / Speed** - Select your device die, package, and speed grade. Use the Die/Package/Speed filters to help in selection. The Die/Package/Speed grades available for selection depend on the level of Libero SoC license (Evaluation/Silver/Gold/Platinum) - refer to the Libero SoC Licensing Web Page for details.

**Range** - Define the voltage and temperature ranges a device may 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.

Supported ranges include:

- All – All ranges
- EXT – All parts that support operating temperature range from 0 to 100 degrees Celsius
- IND – All parts that support operating temperature range from -40 to 100 degrees Celsius

**Note:** Supported operating condition ranges vary according to your device and package. Refer to your device datasheet to find your recommended temperature range.

**Reset Filters** – Reset all filters to the default ALL option except **Family**.

**Search Parts** – Enter a character-by-character search for parts. Search results appear in the parts table.

When **Device Selection** is completed, click on:
• **Next** to proceed to the Device Settings page
  OR
• **Finish** to complete New Project Creation with all remaining defaults.

**New Project Creation Wizard – Device Settings**

The Device Settings page is where you set the Device I/O Technology and Reserve pins for Probes.

![New Project Creation Wizard – Device Settings Page](image)

**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.

**System controller suspended mode.** If the System controller suspended mode is enabled in any PolarFire design, the following operations will not be available during normal operation. For more information, refer to the System Services section in the PolarFire FPGA Security User Guide (UG0753).

- All System controller services requested after power-up is complete and the System controller is suspended
- System controller generated Tamper flags
- Device reset and zeroization Tamper responses
- SPI-Master In-Application Programming (IAP)
- SPI-Slave programming mode

For RT PolarFire devices, the System controller suspended mode is enabled by default for all new RT PolarFire projects. If the System controller suspended mode is disabled, it increases vulnerability to radiation single event effects in the System controller.

**New Project Creation Wizard – Add HDL Source Files**

The Add HDL Source Files page is where you add HDL design source files to your Libero SoC project. The HDL source files can be imported or linked to the Libero SoC Project.
Figure 15 · New Project Creation Wizard - Add HDL Source Files Page

Import File – Navigate to the disk location of the HDL source. Select the HDL file and click Open. The HDL file is copied to the Libero Project in the `<prj_folder>/hdl` folder.

Link File – Navigate to the disk location of the HDL source. Select the HDL file and click Open.

Figure 16 · New Project Creation Wizard – Link Files dialog box

This dialog box allows the user to continue with an absolute path or choose a relative path for linked files.

Create links relative to the path set in Environment variable – This option requires users to enter 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 the user selects relative path and provides an Environment variable for the relative path, the user cannot switch to absolute path. Once the environment variable is set, this option becomes read-only in all other link files dialog boxes.

The HDL file is linked to the Libero Project. Use this option if the HDL source file is located and maintained outside of the Libero project.

Delete - Delete the selected HDL source file from your project. If the HDL source file is linked to the Libero project, the link will be removed.

When Add HDL Sources is completed, click on:

- **Next** to proceed to the Add Constraints page
- **Finish** to complete New Project Creation.

**New Project Creation Wizard - Add Constraints**

The Add Constraints page 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.

**Import File** – Navigate to the disk location of the constraints file. Select the constraints file and click Open. The constraints file is copied to the Libero Project in the <prj_folder>/constraint folder.

**Link File** – Navigate to the disk location of the constraints file. Select the constraints file and click Open. The constraints file is linked to the Libero Project. Use this option if the constraint file is located and maintained outside of the Libero project.
This dialog box allows the user to continue with an absolute path or choose a relative path for linked files.

**Create links relative to the path set in Environment variable** – This option requires users to enter 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 the user selects relative path and provides an Environment variable for the relative path, the user cannot switch to absolute path. Once the environment variable is set, this option becomes read-only in all other link files dialog boxes.

**Delete** - Remove the selected constraints file from your project. If the constraints file is linked to the Libero project, the link will be removed.

When **Add Constraints** is completed, click on:

- **Finish** to complete New Project Creation.
The **Reports** tab displays the result of the New Project creation.

![Figure 19 · Reports Tab](image-url)
Create and Verify Design

Create your design with any or all of the following design capture tools:

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

Create SmartDesign

About SmartDesign

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

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

- Microsemi IP Cores
- User-generated or third-party IP Cores
- HDL design files
- HDL + design files
- Basic macros
- Other SmartDesign components (*.cxf files) generated from SmartDesign in the current Libero SoC project or may be imported from other Libero SoC projects.
- Re-usable design blocks (*.cxz files) published from Libero SoC

For more information see the SmartDesign User Guide.

Create New SmartDesign

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

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

   ![Create New SmartDesign Dialog Box](image)

   Figure 20 - 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.
Export Component Description(Tcl)

Components such as SmartDesign components, configured cores and HDL+ cores can be separately exported as Tcl with the Export Component Description option. To export a SmartDesign component, configured core or HDL+ core as Tcl, right click the component and choose “Export Component Description(Tcl)” option.

![Figure 21 · Export As TCL option for SmartDesign Component](image)
Figure 22 · Export Script Dialog Box

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

**Examples**

Example exported Tcl script for a SmartDesign Component(PCIe_TL_CLK)

```tcl
# 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:SBM_INIT_FROM_SNVM_DONE}
sd_mark_pins_unused -sd_name ${sd_name} -pin_names {PCIe_INIT_MONITOR_0:SBM_INIT_FROM_UPROM_DONE}
sd_mark_pins_unused -sd_name ${sd_name} -pin_names {PCIe_INIT_MONITOR_0:SBM_INIT_FROM_SPI_DONE}
sd_mark_pins_unused -sd_name ${sd_name} -pin_names {PCIe_INIT_MONITOR_0:AUTOCALIB_DONE}
# Add scalar net connections
```

---
Example 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_QDS_GROUP_SIZE:8" \\
    "ENABLE_ECC:0" \\
    "ENABLE_INIT_INTERFACE:false" \\
    "ENABLE_LOOKAHEAD_PRECHARGE_ACTIVATE:false" \\
    "ENABLE_PAR_ALERT:0" \\
    "ENABLE_REINIT:0" \\
    "ENABLE_TAG_IF:0" \\
    "ENABLE_USER_ZQCALIB:0" \\
    "EXPOSE_TRAINING_DEBUG_IF:0" \\
    "FABRIC_INTERFACE:AXI4" \\
    "FAMILY:26" "MEMCTRLR_INST_NO:1" \\
    "MEMORY_FORMAT:COMPONENT" \\
```
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MINIMUM_READ_IDLE</td>
<td>1</td>
</tr>
<tr>
<td>ODT_ENABLE_RD_RNK0_ODT</td>
<td>false</td>
</tr>
<tr>
<td>ODT_ENABLE_RD_RNK0_ODT1</td>
<td>false</td>
</tr>
<tr>
<td>ODT_ENABLE_RD_RNK1_ODT0</td>
<td>false</td>
</tr>
<tr>
<td>ODT_ENABLE_RD_RNK1_ODT1</td>
<td>false</td>
</tr>
<tr>
<td>ODT_ENABLE_WR_RNK0_ODT0</td>
<td>true</td>
</tr>
<tr>
<td>ODT_ENABLE_WR_RNK0_ODT1</td>
<td>false</td>
</tr>
<tr>
<td>ODT_ENABLE_WR_RNK1_ODT0</td>
<td>false</td>
</tr>
<tr>
<td>ODT_ENABLE_WR_RNK1_ODT1</td>
<td>true</td>
</tr>
<tr>
<td>ODT_RD_OFF_SHIFT</td>
<td>0</td>
</tr>
<tr>
<td>ODT_RD_ON_SHIFT</td>
<td>0</td>
</tr>
<tr>
<td>ODT_WR_OFF_SHIFT</td>
<td>0</td>
</tr>
<tr>
<td>ODT_WR_ON_SHIFT</td>
<td>0</td>
</tr>
<tr>
<td>OUTPUT_DRIVE_STRENGTH</td>
<td>RZQ6</td>
</tr>
<tr>
<td>PARAM_IS_FALSE</td>
<td>false</td>
</tr>
<tr>
<td>PARTIAL_ARRAY_SELF_REFRESH</td>
<td>FULL</td>
</tr>
<tr>
<td>PHYONLY</td>
<td>false</td>
</tr>
<tr>
<td>PIPELINE</td>
<td>false</td>
</tr>
<tr>
<td>QOFF</td>
<td>0</td>
</tr>
<tr>
<td>QUEUE_DEPTH</td>
<td>3</td>
</tr>
<tr>
<td>RDIMM_LAT</td>
<td>0</td>
</tr>
<tr>
<td>READ_BURST_TYPE</td>
<td>SEQUENTIAL</td>
</tr>
<tr>
<td>ROW_ADDR_WIDTH</td>
<td>16</td>
</tr>
<tr>
<td>RTT_NOM</td>
<td>DISABLED</td>
</tr>
<tr>
<td>RTT_WR</td>
<td>OFF</td>
</tr>
<tr>
<td>SDRAM_NB_RANKS</td>
<td>1</td>
</tr>
<tr>
<td>SDRAM_NUM_CLK_OUTS</td>
<td>1</td>
</tr>
<tr>
<td>SDRAM_TYPE</td>
<td>DDR3</td>
</tr>
<tr>
<td>SELF_REFRESH_TEMPERATURE</td>
<td>NORMAL</td>
</tr>
<tr>
<td>SHIELD_ENABLED</td>
<td>true</td>
</tr>
<tr>
<td>SIMULATION_MODE</td>
<td>FAST</td>
</tr>
<tr>
<td>TDOQ_ENABLE</td>
<td>DISABLED</td>
</tr>
<tr>
<td>TGIGEN_ADD_PRESET_WIDGET</td>
<td>true</td>
</tr>
<tr>
<td>TIMING_DH</td>
<td>150</td>
</tr>
<tr>
<td>TIMING_DQSCCK</td>
<td>400</td>
</tr>
<tr>
<td>TIMING_DQSQ</td>
<td>200</td>
</tr>
<tr>
<td>TIMING_DQSS</td>
<td>0.25</td>
</tr>
<tr>
<td>TIMING_DS</td>
<td>75</td>
</tr>
<tr>
<td>TIMING_DSH</td>
<td>0.2</td>
</tr>
<tr>
<td>TIMING_DSS</td>
<td>0.2</td>
</tr>
<tr>
<td>TIMING_FETCH</td>
<td>30</td>
</tr>
<tr>
<td>TIMING_IH</td>
<td>275</td>
</tr>
<tr>
<td>TIMING_INIT</td>
<td>200</td>
</tr>
<tr>
<td>TIMING_IS</td>
<td>200</td>
</tr>
<tr>
<td>TIMING_MODE</td>
<td>0</td>
</tr>
<tr>
<td>TIMING_MRD</td>
<td>4</td>
</tr>
<tr>
<td>TIMING_QH</td>
<td>0.38</td>
</tr>
<tr>
<td>TIMING_QSH</td>
<td>0.38</td>
</tr>
<tr>
<td>TIMING_QSH</td>
<td>0.38</td>
</tr>
<tr>
<td>TIMING_RAS</td>
<td>36</td>
</tr>
<tr>
<td>TIMING_RC</td>
<td>49.5</td>
</tr>
<tr>
<td>TIMING_RCD</td>
<td>13.5</td>
</tr>
<tr>
<td>TIMING_REFI</td>
<td>7.8</td>
</tr>
</tbody>
</table>
"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

Example exported Tcl script for a HDL+ core

# Exporting core pattern_gen_checker to TCL
# Exporting Create HDL core command for module pattern_gen_checker
create_hdl_core -file
{X:/10_docs_review/12.0_Release/pcie_demo_tcl_example/DG0756_PF_PCIe_EP/new/project/hdl/ PATTERN_GEN_CHECKER.v} -module {pattern_gen_checker} -library {work} -package {}

# Exporting BIF information of HDL core command for module pattern_gen_checker

Example 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 {\n"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" \n"TxPLL_SSM_DEPTH:0"}
Hierarchical Export Component Description (Tcl)

This option exports the complete design and its subcomponents to Tc1. When this option is executed on a SmartDesign, it iterates through all the instances and gathers information about the pins, groups and nets present in the SmartDesign. All the TCL scripts generated are exported to a folder which is selected by the user when the Hierarchical Export Component Description (Tcl) option is executed.

This exported folder consists of the following files and subfolders—

- **Subfolders**
  - HDL folder – consists of all the imported HDL source files
  - Stimulus folder – consists of all the imported HDL stimulus files
  - Components folder – contains all the Tcl files of the components used in the SmartDesign

- **Files**
  - hdl_source.tcl – contains the tcl for imported and linked files
  - <component>_recursive.tcl – this is the top level tcl used to recreate the design
  - Un_Supported_Cores_List.txt – this file consists of all the cores where the export function cannot be performed.

User can run this option by right clicking on the desired component for which the information needs to be exported in Tcl in the Design/ Stimulus Hierarchy.
The files on getting exported to the desired folder are shown below.

![Files and Subfolders Generated in the Exported Folder](image)

**Figure 25 · Files and Subfolders Generated in the Exported Folder**

Limitations:
Hierarchical Export Component Description (Tcl) support is not available for the following –
- Fabric DDR with Initialization (FDDR_INIT)
- Blocks

Messages:
The tool generates the following messages in the log window as below:
- "Error: Please check the permission of the specified folder." - If the folder specified by the user is not writable.
- "Error: Unable to Export Component 'top' to path" - if the export is unsuccessful.
- "Info: Component 'top' exported successfully to path" – if the export is successful

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

Click the **Generate** button to generate a SmartDesign component.
This will generate a HDL file in the directory `<libero_project>/components/<library>/<yourdesign>`.

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

Generating a SmartDesign component may fail if there are any DRC errors. DRC errors must be corrected before you generate your SmartDesign design.

If the ports of a sub-design have changed, then the parent SmartDesign component will be annotated with the icon ![ ] in the Design Hierarchy tab of the Design Explorer.

**Generate Recursively vs. Non-Recursively**
These options are set in the **Project Preference Dialog Box - Design Flow Preferences section**.

- In the "Recursive generation" mode, the **Generate** button will attempt to generate all sub-design SmartDesigns, depth first. The parent SmartDesign will only be generated if all the sub-designs are generated successfully.
- In the "Non-Recursive generation" mode, the **Generate** button will only attempt to generate the specified SmartDesign. The generation can be marked as successful even if a sub-design is un-generated (either never attempted or unsuccessful). An un-generated component will be annotated with the icon ![ ] in the Design Hierarchy tab of the Design Explorer.
Create Core from HDL

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

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

To create a core from your HDL:

1. Import or create a new HDL source file; the HDL file appears in the Design Hierarchy.
2. Select the HDL file in the Design Hierarchy and click the HDL+ icon or right-click the HDL file and choose Create Core from HDL.
   The Edit Core Definition – Ports and Parameters dialog appears. It shows you which ports and parameters were extracted from your HDL module.
3. Remove parameters that are not intended to be configurable by selecting them from the list and clicking the X icon. Remove parameters that are used for internal variables, such as state machine enumerations. If you removed a parameter by accident, click Re-extract ports and parameters from HDL file to reset the list so it matches your HDL module.
4. (Optional) Click Add/Edit Bus Interfaces to add bus interfaces to your core.

After you have specified 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.

If you added bus interfaces to your HDL+ core, then it will show up in your SmartDesign with a bus interface pin that can be used to easily connect to the appropriate bus IP core.
If your HDL+ has configurable parameters then double-clicking the object on the Canvas (or right-click and select Configure) invokes a configuration dialog that enables you to set these values. On generation, the specific configuration values per instance are written out to the SmartDesign netlist.

![Image of HDL+ Instance and Configuration Dialog Box]

Figure 27 · HDL+ Instance and Configuration Dialog Box
You can right-click the instance and choose Modify HDL to open the HDL file inside the text editor.

**Edit Core Definition**
You can edit your core definition after you created it by selecting your HDL+ module in the design hierarchy and clicking the HDL+ icon.

**Remove Core Definition**
You may decide that you do not want or need the extended information on your HDL module. You can convert it back to a regular HDL module. To do so, right-click the HDL+ in the Design Hierarchy and choose Remove Core Definition. After removing your definition, your instances in your SmartDesign that were referencing this core must be updated. Right-click the instance and choose Replace Component for Instance.

**Designing with HDL**

**Create HDL**
Create HDL opens the HDL editor with a new VHDL or Verilog file. Your new HDL file is saved to your /hdl directory; 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 or not 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; Libero SoC adds one for you. The filename must follow Verilog or VHDL file naming conventions.
4. Click OK.

After creating your HDL file, click the Save button to save your file to the project.
Using the HDL Editor

The HDL Editor is a text editor designed 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.

Editing

Right-click inside the HDL Editor to open the Edit menu items. Available editing functions include cut, copy, paste, Go to line, Comment/Uncomment Selection and Check HDL File. These features are also available in the toolbar.

Word Wrap mode and Font size change option has been added in Libero SoC v12.5.

By default, the word wrap mode will be disabled. The user must right click in the editor and select the Word Wrap:OFF option to enable Word Wrap mode for the file which changes the Word Wrap option to Word Wrap:ON. To disable the Word Wrap mode, user can follow a similar approach by selecting Word Wrap:ON option which later changes the option to Word Wrap:OFF.

The user can increase or decrease the font size of the text in the editor by right clicking in editor and selecting the Increase Font or Decrease Font option.

Saving

You must save your file to add it to your Libero SoC project. Select Save in the File menu, or click the Save icon in the toolbar.

Printing

Print is available from the File menu and the toolbar.

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

HDL Syntax Checker

To run the syntax checker:

In the Files list, double-click the HDL file to open it. Right-click in the body of the HDL editor and choose Check HDL File.

The syntax checker parses the selected HDL file and looks for typographical mistakes and syntactical errors. Warning and error messages for the HDL file appear in the Libero SoC Log Window.

Commenting Text

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

To comment or uncomment out text:

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

Find

In the File menu, choose Find and the Find dialog box appears below the Log/Message window. You can search for a whole word or part of a word, with or without matching the case.

You can search for:

- Match Case
- Match whole word
- Regular Expression

The Find to Replace function is also supported.
Column Editing

Column Editing is supported. Press ALT+click to select a column of text to edit.

Importing HDL Source Files

To import an HDL source file:

1. In the Design Flow window, right-click Create HDL and choose Import Files. The Import Files window appears.
2. Navigate to the drive/folder that contains the HDL file.
3. Select the file to import and click Open.

Note: SystemVerilog (*.sv), Verilog (*.v) and VHDL (*.vhdl/*.vhdl) files can be imported.

Mixed-HDL Support in Libero SoC

You must have ModelSim ME Pro to use mixed HDL in the Libero SoC. You must also have Synplify Pro to synthesize a mixed-HDL design.

When you create a project, you must select a preferred language. The HDL files generated in the flow are created in the preferred language.

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.)

If your preferred language is Verilog, the post-synthesis and post-layout netlists are in Verilog 2001.

Designing with Block Flow

For information about designing with Block Flow, see Designing with Blocks for Libero SoC Enhanced Constraint Flow.

Create New SmartDesign Testbench

The SmartDesign Testbench component may be the top level of the design or it may be used as a lower level SmartDesign Testbench component (after successful generation) in another design.

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

   Figure 28 · Create New SmartDesign Testbench

2. Enter a name, select the Set as Active Stimulus radio button if you want to have this SmartDesign Testbench as your active stimulus, and click OK. The component appears in the Stimulus 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.
HDL Testbench

You can create a HDL Testbench by right-clicking a SmartDesign in the Design Hierarchy and choosing Create Testbench > HDL.

HDL Testbench automatically instantiates the selected SmartDesign into the Component.

You can also double-click Create HDL Testbench to open the Create New HDL Testbench dialog box. The dialog box enables you to create a new testbench file and gives you the option to include standard testbench content and your design data.

HDL Type

Set your HDL Type: Verilog or VHDL for the testbench.

Name

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

Clock Period (ns)

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

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 run.do file that Libero generates to drive the simulation. Setting one testbench as the Active Stimulus is necessary when there are multiple testbenches in the stimulus hierarchy.

Initialize with Standard Template adds boilerplate for a minimal standard test module. This test module does not include an instantiation of the root module under test.

Instantiate Root Design Creates a test module that includes an instance of the root module under test, and clocking logic in the test module which drives the base clock of the root module under test.
Figure 29 · Create New HDL Testbench File Dialog Box

Figure 30 · HDL Testbench Example - VHDL, Standard Template and Root Design Enabled
Import MSS

When a device from PolarFireSoC family is used, a new tool Import MSS is added under Create Design in the design flow process. When the user runs this tool, an Import MSS Component dialog box opens, which allows the user to select the MSS Component (*.cxz) files. Once the selected module file (*.cxz) is imported, it shows up in the Design Hierarchy window preceded by a new icon as shown below.

![MSS Component file in Design Hierarchy Window](image)

Verify Pre-Synthesized Design - RTL Simulation

To perform pre-synthesis simulation, double-click **Simulate** under Verify Pre-Synthesized Design in the Design Flow window. Alternatively, in the Stimulus Hierarchy right-click the testbench and choose **Simulate Pre-Synth Design > Run**.

The default tool for RTL simulation in Libero SoC **PolarFire** is ModelSim™ ME Pro. ModelSim ME works with all levels of Libero SoC license (Eval, Silver, Gold and Platinum) whereas ModelSim Pro ME works with all levels of Libero SoC license except Silver.

ModelSim ME and ModelSim ME Pro are custom editions of ModelSim PE that are integrated into Libero SoC’s design environment. ModelSim for Microsemi is an OEM edition of Mentor Graphics ModelSim tools. ModelSim ME Pro supports mixed VHDL, Verilog, and SystemVerilog simulation but ModelSim ME does not. Both ModelSim editions only work with Microsemi simulation libraries and they are supported by Microsemi.

Other editions of ModelSim are supported by Libero SoC. To use other editions of ModelSim, do not install ModelSim ME from the Libero SoC media.

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

See the following topics for more information on simulation in Libero SoC:

- Simulation Options
- Selecting a Stimulus File for Simulation
- Selecting additional modules for simulation
- Performing Functional Simulation
Project Settings: Simulation - Options and Libraries

Using this dialog box, you can set change how Libero SoC handles Do files in simulation, import your own Do files, set simulation run time, and change the DUT name used in your simulation. You can also change your library mapping.

To access this dialog box, from the Project menu choose Project Settings and click to expand Simulation options or Simulation libraries.

For Simulation options click the option you wish to edit: DO file, Waveforms, Vsim commands, Timescale. For Simulation libraries click on the library you wish to change the path for.

Figure 32 · Project Settings: DO File

DO file

- **Use automatic DO file** - Select if you want the Project Manager to automatically create a DO file that will enable you to simulate your design.
- **Simulation Run Time** - Specify how long the simulation should run. If the value is 0, or if the field is empty, there will not be a run command included in the run.do file.
- **Testbench module name** - Specify the name of your testbench entity name. Default is “testbench,” the value used by WaveFormer Pro.
- **Top Level instance name** - Default is <top_0>, the value used by WaveFormer Pro. The Project Manager replaces <top> with the actual top level macro when you run simulation (presynth/postsynth/postlayout).
- **Generate VCD file** - Click the checkbox to generate a VCD file.
- **VCD file name** - Specifies the name of your generated VCD file. The default is power.vcd; click power.vcd and type to change the name.
- **User defined DO file** - Enter the DO file name or click the browse button to navigate to it.
- **DO command parameters** - Text in this field is added to the DO command.

Waveforms

- **Include DO file** - Including a DO file enables you to customize the set of signal waveforms that will be displayed in ModelSim.
- **Display waveforms for** - You can display signal waveforms for either the top-level testbench or for the design under test. If you select top-level testbench then Project Manager outputs the line ‘addwave /testbench/*’ in the DO file run.do. If you select DUT then Project Manager outputs the line ‘add wave /testbench/DUT/*’ in the run.do file.
- **Log all signals in the design** - Saves and logs all signals during simulation.
Vsim Commands

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

- **Additional options -** Text entered in this field is added to the vsim command.
  - **SRAM ECC Simulation** - Two options can be added to specify the simulated error and correction probabilities of all ECC SRAMs in the design.
    - `-gERROR_PROBABILITY=<value>`, where 0 <= value <= 1
    - `-gCORRECTION_PROBABILITY=<value>`, where 0 <= value <= 1
  - During Simulation, the SB_CORRECT and DB_DETECT flags on each SRAM block will be raised based on generated random numbers being below the specified <value>s.

- When a user runs the Post-Layout Simulation Tool, a run.do file is created, which consists of information which needs to be sent to a simulator. To run a simulation on a corner, user should select one of the SDF corners along with the type of delay needed from one of the options in SDF timing delays section.
  - **SDF Corners:**
    - Slow Process, Low Voltage and High Temperature
    - Slow Process, Low Voltage and Low Temperature
    - Fast Process, High Voltage and Low Temperature
  - **SDF Timing Delays:**
    - Minimum
    - Typical
    - Maximum
  - Disable pulse filtering during SDF based simulations: This option disables the pulse filtering during SDF simulations.
  - After the user selects the corner, appropriate files needed for simulation are written in the run.do file as shown below.

```
vlog -sv-work postlayout "$(PROJECT_DIR)/designer/sd1/sd1_fast_hv_lt.ba.v"
vsim -L PolarFire -L postlayout -t 1ps -sdfmax/sd1=$(PROJECT_DIR)/designer/sd1/sd1_fast_hv_lt.ba.sdf +pulse_int_e/1 +pulse_int_r/1 +transport _int_delays postlayout.sd1
```

**Timescale**

- **TimeUnit** - Enter a value and select s, ms, us, ns, ps, or fs from the pull-down list, which is the time base for each unit. The default setting is ns.

- **Precision** - Enter a value and select s, ms, us, ns, ps, or fs from the pull-down list. The default setting is ps.

**Simulation Libraries**

- **Restore Defaults** - Sets the library path to default from your Libero SoC installation.

- **Library path** - Enables you to change the mapping for your simulation library (both Verilog and VHDL).
  Type the path name or click the Browse button to navigate to your library directory.
Selecting a Stimulus File for Simulation

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

To associate a stimulus:

1. Run simulation or in the Design Flow window under Verify Pre-Synthesized Design right-click **Simulate** and choose **Organize Input Files > Organize Stimulus Files**. The Organize Stimulus Files dialog box appears.

2. Associate your testbench(es):
   - In the Organize Stimulus Files 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 only 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 a testbench**: To remove or change the file(s) in the Associated Source Files list box, select the file(s) and click **Remove**.
     - **To order testbenches**: Use the up and down arrows to define the order you want the testbenches compiled. The top level-entity should be at the bottom of the list.

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

Selecting Additional Modules for Simulation

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

If you need additional modules in simulation, in the Design Flow window right-click **Simulate** and choose **Organize Input Files > Organize Source Files**. The Organize Files for Simulation dialog box appears.

Select the HDL modules you wish to add from the Simulation Files in the Project list and click **Add** to add them to the Associated Stimulus Files list.

Performing Functional Simulation

To perform functional simulation:

1. 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** from the right-click menu.
   - 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 only 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 a testbench**: To remove or change the file(s) in the Associated Source Files list box, select the file(s) and click **Remove**.

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

4. To start ModelSim ME Pro, right-click **Simulate** in the Design Hierarchy window and choose **Open Interactively**.
   - ModelSim starts and compiles the appropriate source files. When the compilation completes, the simulator runs for 1`s and the Wave window opens to display the simulation results.

5. 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.

6. From the **File** menu, select **Quit**.
Libero SoC Constraint Management

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

Libero SoC provides the Constraint Manager as the cockpit to manage your design constraint needs. This is a single centralized graphical interface for you to create, import, link, check, delete, 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.

Invocation of Constraint Manager from the Design Flow Window

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

Libero SoC Design Flow

The Constraint Manager is Libero SoC’s single centralized Graphical User Interface for managing constraints files in the design flow.
Figure 34 - Constraint Manager in Libero SoC Design Flow
**Introduction to Constraint Manager**

**Synthesis Constraints**

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)

**Synplify Netlist Constraints (*.fdc)**

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

**Compile Netlist Constraints (*.ndc)**

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

**SDC Timing Constraints (*.sdc)**

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

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

**Derived Timing Constraints (*.sdc)**

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

**Place and Route Constraints**

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

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

**I/O PDC Constraints**

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

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

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

Timing SDC Constraint file (*.sdc)

These are timing constraint SDC files for Timing-driven Place and Route. From the Constraint Manager Timing tab, you may import (Timing > Import) or create in the Text Editor (Timing > New) a timing SDC file. Click the checkbox 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).

Timing Verifications Constraints

The Constraint Manager manages the SDC timing constraints for Libero SoC’s SmartTime, which is a Timing Verifications/Static Timing analysis tool. SDC timing constraints provide the timing requirements (e.g. create_clock and create_generated_clock) and design-specific timing exceptions (e.g. set_false_path and set_multicycle_path) for Timing Analysis.

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

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

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

Constraint Manager Components

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

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

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

Constraint File and Tool Association

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

Figure 35 · Constraint File and Tool Association
Note: Libero SoC’s Design Flow window displays the state the tool is in. A green check mark indicates successful completion. A warning icon indicates invalidation of the state because the input files for the tool have changed since the last successful run. Association of a new constraint file with a tool or dis-association of an existing constraint file with a tool invalidates the state of the tool with which the constraint file is associated.

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

- I/O Editor
- Chip Planner
- Constraint Editor

<table>
<thead>
<tr>
<th>Constraint Type</th>
<th>Constraint File Extension</th>
<th>Location inside Project</th>
<th>Associated with Design Tool</th>
<th>Interactive Tool (For Editing)</th>
</tr>
</thead>
<tbody>
<tr>
<td>I/O Attributes</td>
<td>PDC (*.pdc)</td>
<td>&lt;proj&gt;\constraints\io.pdc</td>
<td>Place and Route</td>
<td>I/O Editor</td>
</tr>
<tr>
<td>Floorplanning</td>
<td>PDC (*.pdc)</td>
<td>&lt;proj&gt;\constraints\fp.pdc</td>
<td>Place and Route</td>
<td>Chip Planner</td>
</tr>
<tr>
<td>Timing</td>
<td>SDC (*.sdc)</td>
<td>&lt;proj&gt;\constraints*.sdc</td>
<td>Synthesis, Place and Route, Timing Verification</td>
<td>Constraint Editor</td>
</tr>
<tr>
<td>Netlist Attributes</td>
<td>FDC (*.fdc)</td>
<td>&lt;proj&gt;\constraints*.fdc</td>
<td>Synthesis</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td>NDC (*.ndc)</td>
<td>&lt;proj&gt;\constraints*.ndc</td>
<td>Synthesis</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Derive Constraints in Timing Tab

The Constraint Manager can generate timing constraints for IP cores used in your design. These IP cores, available in the Catalog, are family/device-dependent. Once 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).

Create New Constraints

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

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

To create new constraints from the Constraint Manager using the Text Editor:
1. Select the Tab that corresponds to the type of constraint you want to create.
2. Click New.
3. When prompted, enter a file name to store the new constraint.
4. Enter the constraint in the Text Editor.
5. Click OK.
   The Constraint file is saved and visible in the Constraint Manager in the tab you select:
   - I/O Attributes constraint file (<proj>\io\.pdc) in the I/O Attributes tab
   - Floorplanning constraints (<proj>\fp\.pdc) in the Floor Planner tab
   - Timing constraints (<proj>\constraints\*.sdc) in the Timing tab
6. (Optional) Double-click the constraint file in the Constraint Manager to open and add more constraints to the file.

To create new constraints from the Constraint Manager using Interactive Tools:
Note: Netlist Attribute constraints cannot be created by an Interactive Tool. Netlist Attribute files can only be created with a Text Editor.

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

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

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

**Constraint File Order**

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

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

**Figure 36 · 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.*

**Import a Constraint File**

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

*To import a constraint file:*

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

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

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

**Check a Constraint File**

Use the Constraint Manager to check a constraint file.

**To check a constraint file:**
1. Select the tab for the constraint type to check.
2. Click **Check**.

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

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

For Timing Constraints, select from the Check drop-down menu one of the following:
- Check Synthesis Constraints
- Check Place and Route Constraints
- Check Timing Verification Constraints

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

**Check Place and Route Constraints** checks only the constraint files associated with Place and Route
Check Timing Verification Constraints checks only the Constraint Files associated with Timing Verification.

For the constraint files and tool association shown in the SDC file and Tool Association Figure below:

- **Check Synthesis Constraints** checks the following files:
  - top_derived_constraints.sdc
  - user.sdc
  - mytiming2.sdc

- **Check Place and Route Constraints** checks the following files:
  - top_derived_constraints.sdc
  - mytiming.sdc
  - mytiming2.sdc

- **Check Timing Verification Constraints** checks the following files:
  - top_derived_constraints.sdc
  - user.sdc
  - mytiming.sdc
  - mytiming2.sdc

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

![Figure 38 · Timing Constraints SDC file and Tool Association](image)

When a constraint file is checked, the Constraint Manager:

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

**Check Result**

If the check is successful, this message pops up.

![Figure 39 · Check Successful Message](image)

If the check fails, this error message pops up.

![Figure 40 · Check Fails Message](image)
<table>
<thead>
<tr>
<th>Constraint Type</th>
<th>Check for Tools</th>
<th>Required Design State Before Checks</th>
<th>Netlist Used for Checks</th>
<th>Check Result Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>I/O Constraints</td>
<td>Place and Route</td>
<td>Post-Synthesis</td>
<td>ADL Netlist</td>
<td>Libero Message Window</td>
</tr>
<tr>
<td>Floorplanning Constraints</td>
<td>Place and Route</td>
<td>Post-Synthesis</td>
<td>ADL Netlist</td>
<td>Libero Message Window</td>
</tr>
<tr>
<td>Timing Constraints</td>
<td>Synthesis</td>
<td>Pre-Synthesis</td>
<td>RTL Netlist</td>
<td>synthesis_sdc_check.log</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>placer_sdc_check.log</td>
</tr>
<tr>
<td>Timing Verifications</td>
<td></td>
<td></td>
<td></td>
<td>timing_sdc_check.log</td>
</tr>
<tr>
<td>Netlist Attributes (*.fdc)</td>
<td>Synthesis</td>
<td>Pre-Synthesis</td>
<td>RTL Netlist</td>
<td>*cck.srr file</td>
</tr>
<tr>
<td>Netlist Attributes (*.ndc)</td>
<td>Synthesis</td>
<td>Pre-Synthesis</td>
<td>RTL Netlist</td>
<td>Libero Log Window</td>
</tr>
</tbody>
</table>
Edit a Constraint File

The Edit button in the Constraint Manager allows you to:

- Create new constraint files. See To create new constraints from the Constraint Manager using the Text Editor for details.
- Edit existing constraint files.

To edit a constraint file

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

1. Select the tab for the constraint type to edit. An Interactive Tool is opened to make the edits.
2. Click Edit.
   - All constraint files associated with the tool are edited. Files not associated with the tool are not edited.
   - When a constraint file is edited, the constraints in the file are read into the Interactive Tool.
   - Different Interactive Tools are used to edit different constraints/different files:
     - I/O Editor to edit I/O Attributes (<proj>/io/*.pdc). For details, refer to the I/O Editor User Guide
     - Chip Planner to edit Floorplanning Constraints (<proj>/fp/*.pdc). For details, refer to the Chip Planner User Guide (Chip Planner > Help > Reference Manuals)

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.

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

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

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

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

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

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

Refer to the *Timing Constraints Editor User Guide*(Help > Constraints Editor User’s Guide) for details on how to enter/modify timing constraints.

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

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

**Constraint Types**

Libero SoC manages four types of constraints:

- **I/O Attributes Constraints** – Used to constrain placed I/Os in the design. Examples include setting I/O standards, I/O banks, and assignment to Package Pins, output drive, and so on. These constraints are used by Place and Route.

- **Timing Constraints** – Specific to the design set to meet the timing requirements of the design, such as clock constraints, timing exception constraints, and disabling certain timing arcs. These constraints are passed to Synthesis, Place and Route, and Timing Verification.

- **Floor Planner Constraints** – Non-timing floorplanning constraints created by the user or Chip Planner and passed to Place and Route to improve Quality of Routing.
- **Netlist Attributes** - Microsemi-specific attributes that direct the Synthesis tool to synthesize/optimize the, leveraging the architectural features of the Microsemi devices. Examples include setting the fanout limits, specifying the implementation of a RAM, and so on. These constraints are passed to the Synthesis tool only.

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

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

**Constraint Manager – I/O Attributes Tab**

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

Available actions are:

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

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

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

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.

**File and Tool Association**

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

**I/O Settings**

- **Reserve Pins for Device Migration** – This option allows you to reserve pins in the currently selected device that are not bonded in a device or list of devices you may later decide to migrate your design to. Select the target device(s) you may migrate to later to ensure that there will be no device/package incompatibility if you migrate your design to that device.
- **Reserve Pins for Probes** – Check this box if you plan to use live probes when debugging your design with SmartDebug.

**Constraint Manager – Timing Tab**

The Timing tab allows you to manage timing constraints throughout the design process. Timing constraints files (SDC) have the `.sdc` file extension and are placed in the `<Project_location>`\constraint folder. Available actions are:

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

  ![Create New SDC Dialog Box](Image)

- **Import** – Imports an existing timing SDC file into the Libero SoC project. The timing SDC file is copied into the `<Project_location>`\constraint folder.
- **Link** – Creates a link in the project’s constraint folder to an existing timing SDC file (located and maintained outside of the Libero SoC project).
- **Edit** – Opens the Timing Constraints Editor (see Timing Constraints Editor User Guide for details) to modify the SDC file(s) associated with one of the three tools:
  - **Synthesis** – When selected, the timing SDC file(s) associated with the Synthesis tool is loaded in the constraints editor for editing.
  - **Place and Route** - When selected, the timing SDC file(s) associated with the Place and Route tool is loaded in the constraints editor for editing.
  - **Timing Verification** - When selected, the timing SDC file(s) associated with the Timing Verification tool is loaded in the constraints editor for editing.
• **Check** – Check the legality of the SDC file(s) associated with one of the three tools described below:
  - **Synthesis** – The check is performed against the pre-synthesis HDL design.
  - **Place and Route** – The check is performed against the post-synthesis gate level netlist.
  - **Timing Verification** – The check is performed against the post-synthesis gate level netlist.

• **Derive Constraints** – When clicked, Libero generates a timing SDC file based on user configuration of IP core, components and component SDC. It generates the create_clock and create_generated_clock SDC timing constraints. This file is named <top_level_> derived_constraints.sdc. The component SDC and the generated <root>_derived_constraint.sdc files are dependent on the IP cores and vary with the device family.

**Examples**:

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

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

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

The generated report will be visible in the respective nodes of the report view (Design > Reports).
When the SmartTime Constraint Editor tool is invoked or the constraint check is performed all the files associated with the targeted tool – Synthesis, Place and Route, Timing Verification – are being passed for processing.

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

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

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

**File and Tool Association**

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

- Synthesis
- Place and route
- Timing Verification

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

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

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

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

**Derived Constraints**

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

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

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

**To generate SDC timing constraints for IP cores:**

1. Configure and generate the IP Core.
2. From the Constraint Manager’s Timing tab, click Derive Constraints (**Constraint Manager > Timing > Derive Constraints**).
   The Constraint Manager generates the `<root>_derived_constraints.sdc` file and places it in the Timing Tab along with other user SDC constraint file.
3. When prompted for a **Yes** or **No** on whether or not you want the Constraint Manager to automatically associate the derived SDC file to Synthesis, Place and Route, and Timing Verification, click **Yes** to accept automatic association or **No** and then check or uncheck the appropriate checkbox for tool association.

**Note:** Microsemi recommends the `<root>_derived_constraints.sdc` be always associated with all three tools: Synthesis, Place and Route, and Verify Timing. Before running SynplifyPro Synthesis, associate the `<root>_derived_constraints.sdc` file with Synthesis and Place and Route. This will ensure that the design objects (such as nets and cells) in the `<root>_derived_constraints.sdc` file are preserved during the synthesis step and the subsequent Place and Route step will not error out because of design object mismatches between the post-synthesis netlist and the `<root>_derived_constraints.sdc` file.

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

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

**Constraint Manager – Floor Planner Tab**

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

Available actions are:

- **New** – Creates a new floorplanning PDC file and saves it into the `<Project_location>\constraint\fp` folder.
- **Import** – Imports an existing floorplanning PDC file into the Libero SoC project. The floorplanning PDC file is copied into the `<Project_location>\constraint\fp` folder.
- **Link** – Creates a link in the project’s constraint folder to an existing floorplanning PDC file (located and maintained outside of the Libero SoC project).
- **Edit** – Opens the Chip Planner tool to modify the floorplanning PDC file(s) associated with the Place and Route tool.
- **View** - Opens the Chip Planner tool to view the floorplanning PDC file(s) associated with the Place and Route tool. You cannot save/commit any changes made to the constraints file. However, you can export the PDC file(s) using Chip Planner.
- **Check** – Checks the legality of the PDC file(s) associated with the Place and Route tool against the gate level netlist.
When the Chip Planner tool is invoked or the constraint check is performed, all files associated with the Place and Route tool are passed for processing.

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

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

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

**File and Tool Association**

Each floorplanning constraint file can be associated or disassociated to the Place and Route tool.

Click the checkbox under **Place and Route** to associate/disassociate the file from the tool.

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

**See Also**

Chip Planner User Guide
Constraint Manager – Netlist Attributes Tab

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

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

Available operations are:

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

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

![Constraint Manager – Netlist Attributes Tab](image)

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

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

**File and Tool Association**

Each netlist attributes constraint file can be associated with or disassociated from the Synthesis tool. Click the checkbox under **Synthesis** (Compile) to associate/disassociate the file from Synthesis (Compile).

When a file is associated, Libero passes the file to Synthesis (Compile) for processing when Synthesis is run.

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

When Synthesis is OFF (Project > Project Settings > Design Flow > Enable synthesis [not checked]) for a project, the Design Flow Synthesis action is replaced by the Compile action and runs the compile engine on the gate-level netlist (EDIF or Verilog) available in the project.
**Note:** Linked files in constraint manager are shown with relative path if relative option is set for linked files. When a constraint file is missing during environment variable change or a path in an environment variable changes, the constraint manager does show any broken links. Therefore, if a constraint file is missing, an error message appears in the log window if the user tries to access the file.
Implement Design

Synthesize

Double-click **Synthesize** to run synthesis on your design with the default settings specified in the synthesis tool.

If you want to run the synthesis tool interactively, right-click **Synthesize** and choose **Open Interactively**. If you open your tool interactively, you must complete synthesis from within the synthesis tool.

The default synthesis tool included with Libero SoC is Synplify Pro ME. If you want to use a different synthesis tool, you can change the settings in your Tool Profiles.

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

Synthesize Options

Some families enable you to set or change synthesis configuration options for your synthesis tool. To do so, in the Design Flow window, expand **Implement Design**, right-click **Synthesize** and choose **Configure Options**. This opens the Synthesize Options dialog box.

![Synthesize Options Dialog Box](image)

Figure 53 · Synthesize Options Dialog Box
HDL Synthesis Language Settings

HDL Synthesis language options are no longer specified in this dialog box. Please refer to Project Settings: Design Flow Options.

Global Nets (Promotions and Demotions)

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

- **Minimum number of clock pins** – Specifies the threshold value for Clock pin promotion. The default value is 2.
- **Minimum number of asynchronous pins** – Specifies the threshold value for Asynchronous pin promotion. The default is 800 for PolarFire.
- **Minimum fanout of non-clock nets to be kept on globals** – Specifies the threshold value for data pin promotion to global resources. It is the minimum fanout of non-clock (data) nets to be kept on globals (no demotion). The default is 5000 (must be between 1000 and 200000).
- **Number of global resources** – This can be used to control number of Global resources you want to use in your design. By default this displays the number of available global resources for the die you have selected for the project and varies with different die sizes. **For PolarFire, the default is 24 for all dies.**
- **Maximum number of global nets that could be demoted to row-globals** – Specifies the maximum number of global nets that could be demoted to row-globals. The default is 16 (must be between 0 to 50).
- **Minimum fanout of global nets that could be demoted to row-globals** – Specifies the minimum fanout of global nets that could be demoted to row-global. It is undesirable to have high fanout nets demoted using row globals because it may result in high skew. The default is 300. (Must be between 25 to 5000). 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.

**Note:** Hardwired connections to global resources, such as CCC hardwired connections to GB, IO Hardwired connections to GB, and so on, cannot be controlled by these options.

Optimizations

**Enable retiming** – Check this box to enable Retiming during synthesis. Retiming is the process of automatically moving registers (register balancing) across combinational gates to improve timing, while ensuring identical logic behavior. The default is no retiming during synthesis.

**Enable automatic compile point** – Check this box to enable Automatic Compile Point during synthesis. The Automatic Compile Point is disabled by default.

**RAM optimized for:**

Use this option to guide the Synthesis tool to optimize RAMs to achieve your design goal.

- **High speed** – RAM Optimization is geared towards Speed. The resulting synthesized design achieves better performance (higher speed) at the expense of more FPGA resources.
- **Low power** – RAM Optimization is geared towards Low Power. RAMs are inferred and configured to ensure the lowest power consumption.

**Map seq-shift register components to:**

Use this option to select the mapping of sequential logic:

- **Registers** – When selected, sequential shift logic in the RTL is mapped to registers.
- **RAM64x12** – When selected, sequential shift logic in the RTL is mapped to a 64x12 RAM block. This is the default setting.

**Map ROM components to:**

Use this option to select the mapping of ROM components. Libero default is to map ROM components to Logic during synthesis.

- **Logic** - When selected, ROM components will be mapped to Logic.
- **RAM** - When selected, ROM components will be mapped to RAM.
Additional options for Synplify Pro synthesis

Script File

Click the Browse button to navigate to a Synplify Tcl file that contains the Synplify Pro-specific options. Libero passes the options in the Tcl file to Synplify Pro for processing.

Additional Options

Use this field to enter additional Synplify options. Put each additional option on a separate line.

Note: Libero passes these additional options "as-is" to Synplify Pro for processing; no syntax checks are performed. All of these options are set on the Active Implementation only.

The list of recommended Synthesis Tcl options below can be added or modified in the Tcl Script File or Additional Options Editor.

Note: The options from the Additional Options Editor will always have priority over the Tcl Script file options if they are same.

```text
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
set_option -automatic_compile_point 1
```

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

Refer to the Synplify Pro 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:

```text
add_file <**>
impl <**>
project_folder <**>
add_folder <**>
```
Synplify Pro ME

Synplify Pro ME is the default synthesis tool for Libero SoC.

To run synthesis using Synplify Pro ME and default settings, right-click Synthesize and choose Run.

If you wish to use custom settings you must run synthesis interactively.

To run synthesis using Synplify Pro ME with custom settings:

1. If you have set Synplify as 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 a few pre-set 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 in between translate_on and translate_off synthesis directives as follows:

```vhdl
/* 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 are visible in the Files list, under Synthesis Files.

   Should any errors appear after you click the Run button, you can edit the file using the Synplify editor. Double-click the file name in the Synplify window showing the loaded design files. Any changes you make are saved to your original design file in your project.

6. From the File menu, choose Exit to close Synplify. A dialog box asks you if you would like to save any settings that you have made while in Synplify. Click Yes.

   Note: See the Microsemi Attribute and Directive Summary in the Synplify online help for a list of attributes related to Microsemi devices.

   Note: To add a clock constraint in Synplify you must add "n:<net_name>" in your SDC file. If you put the net_name only, it does not work.
Identify Debug Design

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

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

Identify features:

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

You must have both the Identify RTL Debugger and the Identify Instrumentor to run the debugging flow outlined below.

To use the Identify Instrumentor and Debugger:

1. Create your source file (as usual) and run pre-synthesis simulation.
2. (Optional) Run through an entire flow (Synthesis - Compile - Place and Route - Generate a Programming File) without starting Identify.
3. Right-click Synthesize and choose Open Interactively in Libero SoC to launch Synplify.
4. In Synplify, click Options > Configure Identify Launch to setup Identify.
5. In Synplify, create an Identify implementation; to do so, click Project > New Identify Implementation.
6. In the Implementations Options dialog, make sure the Implementation Results > Results Directory points to a location under <libero project>/synthesis/, otherwise Libero SoC is unable to detect your resulting Verilog Netlist file.
7. From the Instrumentor UI specify the sample clock, the breakpoints, and other signals to probe. Synplify creates a new synthesis implementation. Synthesize the design.
8. In Libero SoC, run Synthesis, Place and Route and Generate a Programming File.
   Note: Libero SoC works from the edif netlist of the current active implementation, which is the implementation you created in Synplify for Identify debug.

The Identify RTL Debugger, Synplify, and FlashPro must be synchronized in order to work properly. See the Release Notes for more information on which versions of the tools work together.

Verify Post-Synthesized Design

Generate Simulation File

This step generates the post-synthesis *.v Verilog or *.vhd VHDL netlist for post-synthesis simulation. Post-synthesis simulation verifies the post-synthesis implementation of the design.

The netlist file is located in the synthesis folder of the project. Libero SoC passes this file to the simulator for the post-synthesis simulation run. This step must be preceded by a successful synthesis. If synthesis is not yet run, generating Simulation Files automatically initiates a synthesis run as a requirement to this step.

Verify Post-Synthesis Implementation - Simulate

The steps for performing functional (post-synthesis) and timing (post-layout) simulation are nearly identical. Functional simulation is performed before place-and-route and simulates only the functionality of the logic in the design. Timing simulation is performed after the design has gone through place-and-route and uses timing information based on the delays in the placed and routed designs.

Note: Post-Layout Simulation is not supported for PolarFire Family.
To perform functional simulation:

1. If you have not done so, 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** from the right-click menu.

   In the Organize Files for Source dialog box, all 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 only 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 a testbench**: To remove or change the file(s) in the Associated Source Files list box, select the file(s) and click **Remove**.

   **To order testbenches**: Use the up and down arrows to define the order you want the testbenches compiled. The top level-entity should be at the bottom of the list.

3. When you are satisfied with the Associated Simulation Files list, click **OK**.
4. To start ModelSim ME, right-click **Simulate** in the Design Hierarchy window and choose **Open Interactively**. ModelSim starts and compiles the appropriate source files. When the compilation completes, the simulator runs for 1 μs and the Wave window opens to display the simulation results.
5. Scroll in the Wave window to verify the logic works as intended. Use the cursor and zoom buttons to zoom in and out and measure timing delays.
6. When you are done, from the **File** menu, choose **Quit**.

### Compile Netlist

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

The Compile Netlist step appears in the Design Flow window only after unchecking the Enable Synthesis option in the Project > Project Settings > Design Flow page. This option is only visible 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 simply double-click **Compile Netlist**).

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

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

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

### Options

The Compile Netlist Options sets the threshold value for global resource promotion and demotion when Place and Route is executed.
Number of global resources - The number of available global resources for the die is reported in this field. The number varies with the die size you select for the Libero SoC project.

The following options allow you to set the maximum/minimum values for promotion and demotion of global routing resources.

Maximum Number of global nets that could be demoted to row-globals – Specifies the maximum number of global nets that can be demoted to row-globals. The default is 16.

Minimum fanout of global nets that could be demoted to row-globals – Specifies the minimum fanout of global nets that can be demoted to row-global. The default is 1000. 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.

Minimum fanout of non-clock nets to be kept on globals – Specifies the minimum fanout of non-clock (data) nets to be kept on globals (no demotion). The default is 5000 (valid range is 1000 to 200000). If you run out of global routing resources for your design, increase this number or select a larger die for your design.

Constraint Flow in Implementation

Design State Invalidation

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

<table>
<thead>
<tr>
<th>Status Icon</th>
<th>Tooltip</th>
<th>Description</th>
<th>Possible Causes/Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td>Tool has not run yet.</td>
<td>NEW state</td>
<td>Tool has not run or it has been cleaned.</td>
</tr>
<tr>
<td></td>
<td>Tool runs successfully.</td>
<td>Tool runs with no errors. PASS state.</td>
<td>N/A</td>
</tr>
<tr>
<td>Status Icon</td>
<td>Tooltip</td>
<td>Description</td>
<td>Possible Causes/Remedy</td>
</tr>
<tr>
<td>-------------</td>
<td>---------</td>
<td>-------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Green check mark</td>
<td>Varies with the tool.</td>
<td>Tool runs but with Warnings.</td>
<td>Varies with the tool (e.g., for the Compile Netlist step, not all I/Os have been assigned and locked).</td>
</tr>
<tr>
<td>Red X</td>
<td>Tool Fails.</td>
<td>Tool fails to run.</td>
<td>Invalid command options or switches, invalid design objects, invalid design constraints.</td>
</tr>
<tr>
<td>Yellow exclamation mark</td>
<td>Design State is Out of Date.</td>
<td>Tool state changes from PASS to OUT OF DATE.</td>
<td>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.</td>
</tr>
<tr>
<td>Red X</td>
<td>Timing Constraints have not been met.</td>
<td>Timing Verification runs successfully but the design fails to meet timing requirements.</td>
<td>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.</td>
</tr>
</tbody>
</table>

**Constraints and Design Invalidation**

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

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

- HDL sources (for Synthesis), gate level netlist (for Compile), and Smart Design components
- Design Blocks (*.cxz files) – low-level design units which may have completed Place and Route and re-used as components in a higher-level design
- Constraint files associated with a tool
- Upstream tools in the Design Flow:
  - If the tool state of a Design Flow tool changes from PASS to OUT OF DATE, the tool states of all the tools below it in the Design Flow, if already run and are in PASS state, also change to OUT OF DATE with appropriate tooltips. For example, if the Synthesis tool state changes from PASS to OUT OF DATE, the tool states of Place and Route tool as well as all the tools below it in the Design Flow change to OUT OF DATE.
  - If a Design Flow tool is CLEANED, the tool states of all the tools below it in the Design Flow, if already run, change from PASS to OUT OF DATE.
  - If a Design Flow tool is rerun, the tool states of all the tools below it in the Design Flow, if already run, are CLEANED.
- Tool Options
  - If the configuration options of a Design Flow tool (right-click the tool and choose Configure Options) are modified, the tool states of that tool and all the other tools below it in the Design Flow, if already run, are changed to OUT OF DATE with appropriate tooltips.
- Project Settings:
  - Device selection
  - Device settings
  - Design Flow
  - Analysis operating conditions
<table>
<thead>
<tr>
<th>Setting Changed</th>
<th>Note</th>
<th>Design Flow Tools Affected</th>
<th>New State of the Affected Design Flow Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Die</td>
<td>Part# is changed</td>
<td>All</td>
<td>CLEANED/NEW</td>
</tr>
<tr>
<td>Package</td>
<td>Part# is changed</td>
<td>All</td>
<td>CLEANED/NEW</td>
</tr>
<tr>
<td>Speed</td>
<td>Part# is changed</td>
<td>All</td>
<td>CLEANED/NEW</td>
</tr>
<tr>
<td>Core Voltage</td>
<td>Part# is changed</td>
<td>All</td>
<td>CLEANED/NEW</td>
</tr>
<tr>
<td>Range</td>
<td>Part# is changed</td>
<td>All</td>
<td>CLEANED/NEW</td>
</tr>
<tr>
<td>Default I/O Technology</td>
<td></td>
<td>Synthesize, and all tools below it</td>
<td>OUT OF DATE</td>
</tr>
<tr>
<td>Reserve Pins for Probes</td>
<td></td>
<td>Place and Route, and all tools below it</td>
<td>OUT OF DATE</td>
</tr>
<tr>
<td>PLL Supply Voltage (V)</td>
<td></td>
<td>Verify Power, Generate FPGA Array Data and all other &quot;Program and Debug Design&quot; tools below it</td>
<td>OUT OF DATE</td>
</tr>
<tr>
<td>Power On Reset Delay</td>
<td></td>
<td>Generate FPGA Array Data and all other &quot;Program and Debug Design&quot; tools below it</td>
<td>OUT OF DATE</td>
</tr>
<tr>
<td>System controller suspended mode</td>
<td></td>
<td>Generate FPGA Array Data and all other &quot;Program and Debug Design&quot; tools below it</td>
<td>OUT OF DATE</td>
</tr>
<tr>
<td>Preferred Language</td>
<td>None</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>Enable synthesis</td>
<td>All</td>
<td>OUT OF DATE</td>
<td></td>
</tr>
<tr>
<td>Synthesis gate level netlist format</td>
<td>Synthesize</td>
<td>CLEANED/NEW</td>
<td></td>
</tr>
<tr>
<td>Reports(Maximum number of high fanout nets to be displayed)</td>
<td>None</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>Abort flow if errors are found in PDC</td>
<td>None</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>Abort flow if errors are found in SDC</td>
<td>None</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>Temperature range(C)</td>
<td>Verify Timing, Verify Power</td>
<td>OUT OF DATE</td>
<td></td>
</tr>
<tr>
<td>Core voltage range(V)</td>
<td>Verify Timing, Verify Power</td>
<td>OUT OF DATE</td>
<td></td>
</tr>
</tbody>
</table>
### Setting Changed

<table>
<thead>
<tr>
<th>Note</th>
<th>Design Flow Tools Affected</th>
<th>New State of the Affected Design Flow Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default I/O voltage range</td>
<td>Verify Timing, Verify Power</td>
<td>OUT OF DATE</td>
</tr>
</tbody>
</table>

- **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.

### Check Constraints

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

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

### Design State and Constraints Check

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

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

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

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

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

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

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

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

<table>
<thead>
<tr>
<th>Constraint Type</th>
<th>Synthesis</th>
<th>Place and Route</th>
<th>Timing Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.sdc</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>b.sdc</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>c.sdc [target]</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

- **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.

Constraint Type and Interactive Tool

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

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

![Information Window](image)

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

**Place and Route**

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

**Place and Route Options**

To change your Place and Route settings from the Design Flow window, expand **Implement Design**, right-click **Place and Route** and choose **Configure Options**. This opens the Layout Options dialog box.
Figure 56 · Layout Options Dialog Box
Figure 57 · Layout Options Dialog Box - with Block Flow Enabled

Timing-Driven

Timing-Driven Place and Route is selected by default. The primary goal of timing-driven Place and Route is to meet timing constraints, specified by you or generated automatically. Timing-driven Place and Route typically delivers better performance than Standard Place and Route. If you do not select Timing-driven Place and Route, timing constraints are not considered by the software, although a delay report based on delay constraints entered in SmartTime can still be generated for the design.

Power-Driven

Enable this option to run Power-Driven layout. The primary goal of power-driven layout is to reduce dynamic power while still maintaining timing constraints.
I/O Register Combining

Enable this option to combine any register directly connected to an I/O when it has a timing Constraint. If there are multiple registers directly connected to a (bi-directional) I/O, select one register to combine in the following order: input-data, output-data, output-enable.

Global Pins Demotion

This option is selected by default when Timing-Driven Place and Route is enabled. When this option is enabled, the layout tool selects 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).
In addition, if the driver for the global was a fabric register, the driver is replicated. When a driver register is replicated, the duplicate names are printed. Each set of names should be used in place of the original register in any specified timing constraint.

Driver Replication

Enable this option to enable an algorithm to replicate critical net drivers to reduce timing violations. The algorithm prints the list of registers along with the duplicate names. Each set of names should be used in place of the original register in any specified timing constraint.

High Effort Layout

Enable this option to improve the likelihood of achieving layout success. The layout runtime will increase if you select this option. Timing performance may suffer as well. Users are urged to consider other methods for achieving layout success before utilizing this option.

Repair Minimum Delay Violations

Enable this option to instruct the Router engine to repair Minimum Delay violations for Timing-Driven Place and Route mode (Timing-Driven Place and Route option enabled). The Repair Minimum Delay Violations option, when enabled, performs an additional route that attempts to repair paths that have minimum delay and hold time violations. This is done by increasing the length of routing paths and inserting routing buffers to add delay to the top 100 violating paths.

When this option is enabled, Libero adjusts the programmable delays through I/Os to meet hold time requirements from input to registers. For register-to-register paths, Libero adds buffers.
Libero iteratively analyzes paths 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, Libero will not repair a connection or path which:
- Is a hardwired, preserved, or global net
- Has a sink pin which is a clock pin
- Is violating a maximum delay constraint (that is, the maximum delay slack for the pin is negative)
- May cause the maximum delay requirement for the sink pin to be violated (setup violations)

Typically, this option is enabled in conjunction with the Incremental Layout option when a design’s maximum delay requirements have been satisfied.
Every effort is made to avoid creating max-delay timing violations on worst case paths.
Min Delay Repair produces a report in the implementation directory which lists all of the paths that were considered.
If your design continues to have internal hold time violations, you may wish to rerun repair Minimum Delay Violations (in conjunction with Incremental Layout). This will analyze additional paths if you originally had more than 100 violating paths.

Incremental Layout

Choose Incremental Layout to use previous placement data as the initial placement for the next run. If a high number of nets fail, relax constraints, remove tight placement constraints, deactivate timing-driven mode, or select a bigger device and rerun Place and Route.

You can preserve portions of your design by employing Compile Points, which are RTL partitions of the design that you define before synthesis. The synthesis tool treats each Compile Point as a block which enables 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 make it easy for you to mark portions of a design as black boxes, and let you divide the design effort between designers or teams. See the Synopsys FPGA Synthesis Pro ME User Guide for more information.

Use Multiple Pass

Check Multiple Pass to run multiple pass of Place and Route to get the best Layout result. Click Configure to specify the criteria you want to use to determine the best layout result.

Block Creation – Number of row-global resources

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 restricts the number of row-global resources available in every half-row of the device. During Place and Route of the block, the tool will not exceed this capacity on any half-row. 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 will be able to integrate with the rest of the design if they consume the remaining row-global capacity.

See Also
extended_run_lib

Multiple Pass Layout Configuration

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

- Before running Multiple Pass Layout, save your design.
- Multiple Pass Layout is supported by all families.
- Multiple Pass Layout saves your design file with the pass that has the best layout results. If you want to preserve your existing design state, you should save your design file with a different name before proceeding. To do this, from the File menu, choose Save As.
- Four types of reports (timing, maximum delay timing violations, minimum delay timing violations, and power) for each pass are written to the working directory to assist you in later analysis:
  - `<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 your multiple pass options:

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

![Multi-Pass Configuration Dialog Box](image)

3. Set the options and click OK.

**Number of passes**: Set the number of passes (iterations) using the slider. 1 is the minimum and 25 is the maximum. The default is 5.

**Start at seed index**: Set the specific index into the array of random seeds which is to be the starting point for the passes. If not specified, the default behavior is to continue from the last seed index that was used.

**Measurement**: Select the measurement criteria you want to compare layout results against.

- **Slowest clock**: Select to use the slowest clock frequency in the design in a given pass as the performance reference for the layout pass.
- **Specific clock**: Select to use a specific clock frequency as the performance reference for all layout passes.
- **Timing violations**: This is the default. Select Timing Violations to use the pass that best meets the slack or timing-violations constraints.

**Note**: You must enter your own timing constraints through SmartTime or SDC.

- **Maximum delay**: Select to examine timing violations (slacks) obtained from maximum delay analysis. This is the default.
- **Minimum delay**: Select to examine timing violations (slacks) obtained from minimum delay analysis.
- **Select by**: Worst Slack or Total Negative Slack to specify the slack criteria.
• When Worst Slack (default) is selected, the 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.

• When Total Negative Slack is selected, the 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: Select to stop performing remaining passes if all timing constraints have been met (when there are no negative slacks reported in the timing violations report).

• **Total power**: Select to determine the best pass to be the one that has the lowest total power (static + dynamic) of all layout passes.

### Iteration Summary Report

The file `<root_module>_iteration_summary.rpt` records a summary of how the multiple pass run was invoked either through the GUI or extended_run_lib Tcl script, with arguments for repeating each run. Each new run appears with its own header in the Iteration Summary Report with fields RUN_NUMBER and INVOKED AS, followed by a table containing Seed Index, corresponding Seed value, Comparison data, Report Analyzed, and Saved Design information.

![Iteration Summary Report](image)

**Figure 59 · Iteration Summary Report**

### See Also

- Place and Route
- extended_run_lib

### Resource Usage

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

From the Design menu, choose **Reports (Design > Reports)**. Click `<design_name>_layout_log.log` to open the log file.

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

<table>
<thead>
<tr>
<th>Type</th>
<th>Used</th>
<th>Total</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>4LUT</td>
<td>400</td>
<td>86184</td>
<td>0.46</td>
</tr>
<tr>
<td>DFF</td>
<td>300</td>
<td>86184</td>
<td>0.34</td>
</tr>
<tr>
<td>I/O Register</td>
<td>0</td>
<td>795</td>
<td>0.00</td>
</tr>
</tbody>
</table>
4LUTs are 4-input Look-up Tables that can implement any combinational logic functions with up to four inputs. The Logic Element is a logic unit in the fabric. It may contain a 4LUT, a DFF, or both. The number of Logic Elements in the report includes all Logic Elements, regardless of whether they contain 4LUT only, DFF only, or both.

**Overlapping of Resource Reporting**

The number of 4LUTs in the report are the total number used for your design, regardless of whether or not they are combined with the DFFs. Similarly, the number of DFFs in the report are the total number used for your design, regardless of whether or not they are combined with 4LUT's.

In the report above, there is a total of 473 Logic Elements (LEs) used for the design.

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

400 of the 473 LEs have 4LUTs inside, which means 73 (473-400) of them have no 4LUTS in them. These 73 LEs are using only the DFF portion of the LE.

<table>
<thead>
<tr>
<th>Type</th>
<th>Used</th>
<th>Total</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic Element</td>
<td>473</td>
<td>86184</td>
<td>0.55</td>
</tr>
</tbody>
</table>

LEs using DFF Only = 473-400 = 73

LEs using 4LUTS only = 473-300= 173

= 246 (Total of LEs using 4LUTS ONLY or DFF ONLY)

Report's Overlapped resource = 227 (LEs using both 4LUTS and DFF)

Total number of LEs used = 473

The area where the two circles overlap represents the overlapped resources in the Resource Usage report.
Global Net Report

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

The global routing resources in Microsemi 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)

The Global Net Report has following sections:

Global Nets Information

The GB Location refers to the location of the Global routing resource/instance name of the macro on the chip. The location is indicated by X-Y co-ordinates of the global resource macro.
Global Nets Information

<table>
<thead>
<tr>
<th>From</th>
<th>GB Location</th>
<th>Net Name</th>
<th>Fanout</th>
</tr>
</thead>
<tbody>
<tr>
<td>GB[4]</td>
<td>(725, 156)</td>
<td>reset_ctrl_i/R_core_reset_out_RNIF/UQ6/U0_YWh</td>
<td>35701</td>
</tr>
<tr>
<td>GB[8]</td>
<td>(734, 156)</td>
<td>pli_i/GL0_INST/U0_YWn</td>
<td>35488</td>
</tr>
<tr>
<td>GB[15]</td>
<td>(741, 156)</td>
<td>pli_i/GL3_INST/U0_YWn</td>
<td>2006</td>
</tr>
<tr>
<td>GB[13]</td>
<td>(739, 156)</td>
<td>reset_ctrl_i/R_global_reset_RNIO738/U0_YWn</td>
<td>1848</td>
</tr>
<tr>
<td>GB[14]</td>
<td>(740, 156)</td>
<td>pli_i/GL2_INST/U0_YWn_GEast</td>
<td>1104</td>
</tr>
<tr>
<td>GB[2]</td>
<td>(724, 156)</td>
<td>serdes_i/SERDES_IF_0/EP0SC_1_RX_CLK_keep_RN TừDL3/U0_YWn</td>
<td>514</td>
</tr>
<tr>
<td>GB[0]</td>
<td>(722, 156)</td>
<td>serdes_i/SERDES_IF_0/EP0SC_0_RX_CLK_keep_RNI015/U0_YWn</td>
<td>513</td>
</tr>
<tr>
<td>GB[3]</td>
<td>(725, 156)</td>
<td>serdes_i/SERDES_IF_0/EP0SC_2_RX_CLK_keep_RNI010/U0_YWn</td>
<td>511</td>
</tr>
<tr>
<td>GB[7]</td>
<td>(729, 156)</td>
<td>serdes_i/pcs_pl_0/pcs_0/rx_p_c_r_0_RN1032/U0_YWn</td>
<td>312</td>
</tr>
<tr>
<td>GB[11]</td>
<td>(737, 156)</td>
<td>serdes_i/pcs_pl_1/pcs_0/rx_p_c_r_0_RN1030/U0_YWn</td>
<td>310</td>
</tr>
<tr>
<td>GB[1]</td>
<td>(723, 156)</td>
<td>serdes_i/SERDES_IF_0/EP0SC_0_TX_CLK_keep_RNI032/U0_YWn</td>
<td>178</td>
</tr>
<tr>
<td>GB[5]</td>
<td>(727, 156)</td>
<td>serdes_i/SERDES_IF_0/EP0SC_2_TX_CLK_keep_RNI032/U0_YWn</td>
<td>176</td>
</tr>
<tr>
<td>GB[6]</td>
<td>(728, 156)</td>
<td>serdes_i/SERDES_IF_0/EP0SC_1_TX_CLK_keep_RN1032/U0_YWn</td>
<td>178</td>
</tr>
<tr>
<td>GB[10]</td>
<td>(736, 156)</td>
<td>SPI_SCLK_ibuf_RNI045/U0_YWn_GEast</td>
<td>149</td>
</tr>
<tr>
<td>GB[9]</td>
<td>(735, 156)</td>
<td>SRAM_CQ_ibuf_RNI037/U0_YWn_GEast</td>
<td>111</td>
</tr>
<tr>
<td>GB[12]</td>
<td>(738, 156)</td>
<td>SRAM_CQ_ibuf_RNI036/U0_YWn_GEast</td>
<td>18</td>
</tr>
</tbody>
</table>

Figure 61 · Global Net Information

I/O to GB Connections

This section lists all the I/Os connected to the Global Resource/instance name of the macro.

I/O to GB Connections

<table>
<thead>
<tr>
<th>Port Name</th>
<th>Pin Number</th>
<th>I/O Function</th>
<th>From</th>
<th>From Location</th>
<th>To</th>
<th>Net Name</th>
<th>Net Type</th>
<th>Fanout</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPI_SCLK</td>
<td>CSO</td>
<td>M6_O7_RNIF868</td>
<td>SPI_SCLK_ibuf/RNI045/U0_YWn_GEast</td>
<td>(64, 97)</td>
<td>GB[19]</td>
<td>SPI_SCLK_ibuf ROUTED</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>SRAM_CQ</td>
<td>G16</td>
<td>M6_O7_RNIF822</td>
<td>SRAM_CQ_ibuf/RNI037/U0_YWn_GEast</td>
<td>(5, 10)</td>
<td>GB[1]</td>
<td>SPI_SCLK_ibuf ROUTED</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>SRAM_CQ</td>
<td>G13</td>
<td>M6_O7_RNIF822</td>
<td>SRAM_CQ_ibuf/RNI036/U0_YWn_GEast</td>
<td>(5, 10)</td>
<td>GB[1]</td>
<td>SPI_SCLK_ibuf ROUTED</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Figure 62 · I/O to GB Connections

Net type is either 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), varies from iteration to iteration.

The I/O function name column describes all the connection details about the I/O such as the bank name, hardwired GB or hardwired CCC connections, if any, and/or dedicated SERDES/DDR connections, if any. For hardwired connections, the function name (DDRIO120PB2/MDDR_DQ_ECC1/GB12/CCC_NE1_CLK12) contains the GB index (GB12 in this case) that matches the GB index in the To column (GBL[12] in this case) whereas for routed connections the Function name does not contain the proper GB index.

Fabric to GB Connections

This section lists all the nets originating from the fabric to the Global Resources/Instance name of the macro. The From Location refers to the X-Y co-ordinates of the driver pin of the net. Generally speaking, the nets are routed nets (not hardwired).
Figure 63 · Fabric to GB connections

**Fabric to GB Connections**

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

**CCC to GB Connections**

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

<table>
<thead>
<tr>
<th>From</th>
<th>From Location</th>
<th>To</th>
<th>I/O Function</th>
<th>Net Name</th>
<th>Net Type</th>
<th>Fanout</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>pll_IP/INST/INST_CCC_IP:GL0</td>
<td>CCC_NE0 (1428, 302)</td>
<td>GB[8]</td>
<td>pll_ip/GL0_net</td>
<td>HARDWIRED</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>pll_IP/INST/INST_CCC_IP:GL3</td>
<td>CCC_NE0 (1428, 302)</td>
<td>GB[15]</td>
<td>pll_ip/GL3_net</td>
<td>HARDWIRED</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>pll_IP/INST/INST_CCC_IP:GL2</td>
<td>CCC_NE0 (1428, 302)</td>
<td>GB[14]</td>
<td>pll_ip/GL2_net</td>
<td>HARDWIRED</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 64 · CCC to GB Connections

**CCC Input Connections**

This section lists the nets from the I/O Pins to the CCC inputs.

**CCC Input Connections**

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.

The I/O function column describes all the connection details about the I/O such as the bank name, hardwired GB or hardwired CCC connections, if any, and/or dedicated SERDES/DDR connections, if any. For hardwired connections, the I/O function name contains the CCC location (CCC_NE0 in this case) and the To (Pin Swapped for Back Annotation Only) column contains the actual input pin of the CCC in the back annotated netlist.

**Local Clock Nets to RGB Connections**

This 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 the use of RGBs to drive regional clocks spanning a small fabric area, such as the clock network for SERDES.
Figure 66 · Local Clock Nets to RGB Connections

The location refers to the X-Y co-ordinates on the chip. The fanout column gives the total fanout of the net and the local fanout column gives the fanout at the local RGB only. The driver in the From column is routed to different RGBs each with different local fanout.

The From column refers to the X-Y co-ordinates of the driver of the net. The driver in the From column is routed to different RGBs each with different local fanout. The Fanout column gives the total fanout of the net and the Local Fanout column gives the fanout at the local RGB only.

**Global Clock Nets to RGB Connections**

This section lists all nets from Globals (GBs) to Row Globals (RGBs).

The From location refers to the X-Y co-ordinates on the chip. The Fanout column gives the total fanout of the net and the Local Fanout column gives the fanout local to RGB. The driver in the From column is hardwired to different RGBs each with different local fanout.
Global Clock Nets to RGB Connections

<table>
<thead>
<tr>
<th>From</th>
<th>From Location</th>
<th>Net Name</th>
<th>Fanout</th>
<th>RGB Location</th>
<th>Local Fanout</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GBR[18]</td>
<td>clk16_iobuf_RNI69L/U40_Y</td>
<td>5003</td>
<td>(1166, 114)</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>(1166, 120)</td>
<td></td>
<td></td>
<td>(1166, 120)</td>
<td>53</td>
</tr>
<tr>
<td>3</td>
<td>(1166, 123)</td>
<td></td>
<td></td>
<td>(1166, 123)</td>
<td>40</td>
</tr>
<tr>
<td>4</td>
<td>(1166, 129)</td>
<td></td>
<td></td>
<td>(1166, 129)</td>
<td>39</td>
</tr>
<tr>
<td>5</td>
<td>(1166, 132)</td>
<td></td>
<td></td>
<td>(1166, 132)</td>
<td>48</td>
</tr>
<tr>
<td>6</td>
<td>(1166, 135)</td>
<td></td>
<td></td>
<td>(1166, 135)</td>
<td>50</td>
</tr>
<tr>
<td>7</td>
<td>(1166, 138)</td>
<td></td>
<td></td>
<td>(1166, 138)</td>
<td>77</td>
</tr>
<tr>
<td>8</td>
<td>(1166, 141)</td>
<td></td>
<td></td>
<td>(1166, 141)</td>
<td>100</td>
</tr>
<tr>
<td>9</td>
<td>(1166, 144)</td>
<td></td>
<td></td>
<td>(1166, 144)</td>
<td>64</td>
</tr>
<tr>
<td>10</td>
<td>(1166, 147)</td>
<td></td>
<td></td>
<td>(1166, 147)</td>
<td>53</td>
</tr>
<tr>
<td>11</td>
<td>(1166, 150)</td>
<td></td>
<td></td>
<td>(1166, 150)</td>
<td>63</td>
</tr>
<tr>
<td>12</td>
<td>(1166, 156)</td>
<td></td>
<td></td>
<td>(1166, 156)</td>
<td>39</td>
</tr>
<tr>
<td>13</td>
<td>(1166, 159)</td>
<td></td>
<td></td>
<td>(1166, 159)</td>
<td>128</td>
</tr>
<tr>
<td>14</td>
<td>(1166, 162)</td>
<td></td>
<td></td>
<td>(1166, 162)</td>
<td>146</td>
</tr>
<tr>
<td>15</td>
<td>(1166, 165)</td>
<td></td>
<td></td>
<td>(1166, 165)</td>
<td>146</td>
</tr>
<tr>
<td>16</td>
<td>(1166, 168)</td>
<td></td>
<td></td>
<td>(1166, 168)</td>
<td>130</td>
</tr>
<tr>
<td>17</td>
<td>(1166, 171)</td>
<td></td>
<td></td>
<td>(1166, 171)</td>
<td>139</td>
</tr>
<tr>
<td>18</td>
<td>(1166, 174)</td>
<td></td>
<td></td>
<td>(1166, 174)</td>
<td>146</td>
</tr>
<tr>
<td>19</td>
<td>(1166, 177)</td>
<td></td>
<td></td>
<td>(1166, 177)</td>
<td>184</td>
</tr>
<tr>
<td>20</td>
<td>(1166, 180)</td>
<td></td>
<td></td>
<td>(1166, 180)</td>
<td>20</td>
</tr>
<tr>
<td>21</td>
<td>(1166, 183)</td>
<td></td>
<td></td>
<td>(1166, 183)</td>
<td>145</td>
</tr>
<tr>
<td>22</td>
<td>(1166, 186)</td>
<td></td>
<td></td>
<td>(1166, 186)</td>
<td>127</td>
</tr>
<tr>
<td>23</td>
<td>(1166, 189)</td>
<td></td>
<td></td>
<td>(1166, 189)</td>
<td>121</td>
</tr>
<tr>
<td>24</td>
<td>(1166, 192)</td>
<td></td>
<td></td>
<td>(1166, 192)</td>
<td>127</td>
</tr>
<tr>
<td>25</td>
<td>(1166, 195)</td>
<td></td>
<td></td>
<td>(1166, 195)</td>
<td>110</td>
</tr>
<tr>
<td>26</td>
<td>(1166, 198)</td>
<td></td>
<td></td>
<td>(1166, 198)</td>
<td>94</td>
</tr>
<tr>
<td>27</td>
<td>(1166, 201)</td>
<td></td>
<td></td>
<td>(1166, 201)</td>
<td>30</td>
</tr>
</tbody>
</table>

Figure 67 · Global Clock Nets to RGB Connections

Verify Post Layout Implementation

Generate Back Annotated Files

This tool when executed generates two files - *ba.sdf and *ba.v/.vhd files.

- *ba.sdf – Delay file in Standard Delay Format (SDF) for back annotation to the simulator.
- *ba.v/.vhd - Post-layout flattened netlist used exclusively for back-annotated timing simulation. May contain low level macros not immediately recognizable to you; these were added by the software to improve your design performance.

Export Enhanced min delays for best case - Exports your enhanced min delays to include the best-case timing results in your Back Annotated File.
Verify Timing

Verify Timing Configuration

Use this dialog box to configure the 'Verify Timing' tool to generate a timing constraint coverage report and detailed static timing analysis and violation reports based on different combinations of process speed, operating voltage, and temperature.

For the timing and timing violation reports you can select:

**Slow process, Low voltage and Low temperature operating conditions**
- Max Delay Analysis Timing Report
- Min Delay Analysis Timing Report
- Max Delay Analysis Timing Violations Report
- Min Delay Analysis Timing Violations Report

**Fast process, High voltage and Low temperature operating conditions**
- Max Delay Analysis Timing Report
- Min Delay Analysis Timing Report
- Max Delay Analysis Timing Violations Report
- Min Delay Analysis Timing Violations Report

**Slow process, Low voltage and High temperature operating conditions**
- Max Delay Analysis Timing Report
- Min Delay Analysis Timing Report
- Max Delay Analysis Timing Violations Report
- Min Delay Analysis Timing Violations Report

**Multi Corner**
- Max Analysis Timing Multi Corner Report
- Min Analysis Timing Multi Corner Report
- Max Analysis Timing Violation Multi Corner Report
- Min Analysis Timing Violation Multi Corner Report

**Constraints Coverage Report**
- Generate constraints coverage report

The following figure shows an example of the Verify Timing Configuration dialog box.
Figure 68 · Verify Timing Configuration Dialog Box – Report Tab
The reports can be generated in either xml or text format.

**Timing Report**

The Timing Report has following options:

- **Limit the number of reported paths per section**: Sets the number of reported paths under each section. The number of paths set can be anywhere from 1-20000.

- **Limit the number of expanded paths per section**: Sets the number of expanded paths under each section. The number of paths set can be anywhere from 1-20000.
Limit the number of parallel paths per expanded path to: Sets the number of parallel paths for each expanded path. The number of paths set can be anywhere from 1-20000.

**Timing Violations Report**

The Timing Report has following options:

Limit the number of reported paths per section: Sets the number of reported paths under each section. The number of paths set can be anywhere from 1-20000.

Limit the number of expanded paths per section: Sets the number of expanded paths under each section. The number of paths set can be anywhere from 0-20000.

Limit the number of parallel paths per expanded path to: Sets the number of parallel paths for each expanded path. The number of paths set can be anywhere from 1-20000.

Maximum Slack: Sets a maximum slack threshold value in nanoseconds. Paths will be filtered based on the slack threshold value only in Timing Violation reports

**Note:** If any of the options are left blank in Report Settings Tab, the tool will show a tool tip error icon as in image below:

Types of Timing Reports

From the **Design Flow window > Verify Timing**, you can generate the following types of reports:

- **Timing reports** – These reports display timing information organized by clock domain. Four types of timing reports are available. You can configure which reports to generate using the 'Verify Timing' configuration dialog box (**Design Flow > Verify Timing > Configure Options**). 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** – These reports display timing information organized by clock domain. Four types of timing violations reports are available. You can configure which reports to generate using the ‘Verify Timing’ configuration dialog (Design Flow > Verify Timing > Configure Options). 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** – This report displays the overall coverage of the timing constraints set on the current design.

<root>_timing_constraints_coverage.xml (generated by default)
SmartTime

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

See the Timing Constraints Editor User Guide for help with creating and editing timing constraints.

Static Timing Analysis (STA)

Static timing analysis (STA) offers an efficient technique for identifying timing violations in your design and ensuring that it meets all your timing requirements. You can communicate timing requirements and timing exceptions to the system by setting timing constraints. A static timing analysis tool will then check and report setup and hold violations as well as violations on specific path requirements.

STA is particularly 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.

The major disadvantage of STA is that the STA tools do not automatically detect false paths in their algorithms as it reports all possible paths, including false paths, in the design. False paths are timing paths in the design that do not propagate a signal. To get a true and useful timing analysis, you need to identify those false paths, if any, as false path constraints to the STA tool and exclude them from timing considerations.

Timing Constraints

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

Timing Analysis

SmartTime provides a selection of analysis types that enable you to:

- Find the minimum clock period/highest frequency that does not result in a timing violation
- 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 individually computing clock insertion delays for each register.

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

SmartTime and Place and Route

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

SmartTime and Timing Reports

From SmartTime > Tools > Reports, the following report files can be generated:

- Timing Report (for both Max and Min Delay Analysis)
- Timing Violations Report (for both Max and Min Delay Analysis)
- Bottleneck Report
SmartTime and Cross-Probing into Chip Planner

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

- Ports
- Macros
- Timing Paths


SmartTime and Cross-Probing into Constraint Editor

From SmartTime, you can cross-probe into the Constraint Editor. Select a Timing Path in SmartTime’s Analysis View and add a Timing Exception Constraint (False Path, Multicycle Path, Max Delay, Min Delay). The Constraint Editor reflects the newly added timing exception constraint.

Refer to the SmartTime Static Timing Analyzer User Guide for details.

Verify Power

Right-click on the Verify Power command in the Design Flow window to see the following menu of options:

<table>
<thead>
<tr>
<th>Tool</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Create Design</td>
<td></td>
</tr>
<tr>
<td>2. Constraints</td>
<td></td>
</tr>
<tr>
<td>3. Implement Design</td>
<td></td>
</tr>
<tr>
<td>4. Synthesize</td>
<td></td>
</tr>
<tr>
<td>5. Open Netlist Viewer</td>
<td></td>
</tr>
<tr>
<td>6. Verify Post-Synthesized Design</td>
<td></td>
</tr>
<tr>
<td>7. Generate Simulation File</td>
<td></td>
</tr>
<tr>
<td>8. Simulate</td>
<td></td>
</tr>
<tr>
<td>9. Place and Route</td>
<td></td>
</tr>
<tr>
<td>10. Verify Post Layout Implementation</td>
<td></td>
</tr>
<tr>
<td>11. Verify Timing</td>
<td></td>
</tr>
<tr>
<td>12. Open SmartTime</td>
<td></td>
</tr>
<tr>
<td>13. Verify Power</td>
<td></td>
</tr>
<tr>
<td>14. Open SSN Analyzer</td>
<td></td>
</tr>
<tr>
<td>15. Configure Hardware</td>
<td></td>
</tr>
<tr>
<td>16. Program Design</td>
<td></td>
</tr>
<tr>
<td>17. Debug Design</td>
<td></td>
</tr>
<tr>
<td>18. Handoff Design for Production</td>
<td></td>
</tr>
<tr>
<td>19. Handoff Design for Debugging</td>
<td></td>
</tr>
</tbody>
</table>

Figure 71 · Verify Power right-click menu
Verify Power sub-commands

- **Run** - Runs the default power analysis and produces a power report. This is also the behavior of a double-click to Verify Power.
- **Clean and Run All** - Identical to the sequence of commands "Clean" (see below) and "Run"
- **Open interactively** - Brings up the SmartPower for Libero SoC tool (see below)
- **Clean** - Clears the history of any previous default power analysis, including deletion of any reports. The flow task completion icon will also be cleared.
- **Configure Options** - This sub-command is only available if there are options to configure, in which case a dialog box will pop-up presenting the user with technology-specific choices.
- **View Report** - This sub-command is only available and visible if a report is available. When **View Report** is invoked, the Report tab will be added to the Libero SoC GUI window, and the Power Report will be selected and made visible.

SmartPower

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

SmartPower provides a detailed and accurate way to analyze designs for Microsemi SoC FPGAs: from top-level summaries to deep down specific functions within the design, such as gates, nets, IOs, memories, clock domains, blocks, and power supply rails.

You can analyze the hierarchy of block instances and specific instances within a hierarchy, and each can be broken down in different ways to show the respective power consumption of the component pieces.

SmartPower also analyses 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 that may have been created in the design. Custom modes can also be used for testing "what if" potential operating modes.

SmartPower has a very unique feature that enables you to create test scenario profiles. A profile enables you to create sets of operational modes, so you can understand the average power consumed by this combination of functional modes. An example may be a combination of Active, Sleep, and Flash*Freeze modes – as would be used over time in an actual application.

SmartPower generates detailed hierarchical reports of the power consumption of a design for easy evaluation. This enables you to locate the power consumption source and take appropriate action to reduce the power if possible.

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

See the SmartPower User Guide.

Simultaneous Switching Noise

Introduction

Simultaneous Switching Noise (SSN) is the Libero SoC voltage noise analysis tool. It provides a detailed analysis of the noise margin on each I/O pin in the design based on the pin information as well as all the 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 placement of the pin. The SSN Analyzer helps you achieve the desired voltage noise margin, resulting in improved signal integrity.

Right-click **SSN Analyzer** in the Design Flow window and select **Open Interactively** to open the SSN Analyzer.
Supported Die/Package

<table>
<thead>
<tr>
<th>Family</th>
<th>Die</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>PolarFire</td>
<td>MPF300XT</td>
<td>FCG1152</td>
</tr>
<tr>
<td></td>
<td>MPF100T</td>
<td>FCG484</td>
</tr>
<tr>
<td></td>
<td>MPF200T</td>
<td>FCG484</td>
</tr>
<tr>
<td></td>
<td>MPF300T</td>
<td>FCG484/FCG1152</td>
</tr>
<tr>
<td></td>
<td>MPF500T</td>
<td>FCG1152</td>
</tr>
</tbody>
</table>

Note: 1 ns pulse width is only supported for MPF300XT/FCG1152.

Dies and packages for which characterization data is unavailable are not supported.

Supported I/O Standard

The SSN Analyzer supports the following I/O Standards:
- LVCMOS 3.3V
- LVCMOS 2.5V
- LVCMOS 1.8V
- LVCMOS 1.5V
- LVCMOS 1.2V
- LVTTL

Supported I/O Types

Only single-end I/Os are supported. Differential I/Os are not supported.

SSN Analyzer

Three tabs are available in the SSN Analyzer:
- Noise Report
- Excluded I/Os
- Summary

Noise Report

The Noise Report tab displays by default when the SSN Analyzer opens, and lists all of the design’s Output and Inout ports. Input I/Os are not supported. The displayed columns are:
- Bank Name/Pin Number – Shows the Bank Number and the Package Pin Number of the Port
- Port Name – Shows the Port Name
- 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 and LVCMOS 1.2V and LVTTL.
- Drive Strength (mA) – Drive Strength selections are available from 2 to 12.
- **Static** – When this checkbox is checked, the I/O is considered neither as an Aggressor nor as a Victim. It is excluded from SSN Analysis.

- **Don’t Care** – When this checkbox is checked, the I/O is excluded from consideration as a Victim for Noise Margin computation. However, it is considered as an Aggressor for Noise Margin computation of other I/Os.
  
  **Note:** Static and Don’t Care are mutually exclusive.

- **Noise Margin (%)** – This is the Noise Margin number computed by the SSN Analyzer. A negative number (shown in red) indicates that it is outside the guideline of SSN analysis.

- **Within Guideline** – Either 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/LVC莫斯 (3.3V)** – A Yes (within guideline) is defined as follows:
    - A ground bounce voltage less than or equal to 1.25V and a pulse width of less than or equal to 1 ns
    - A VDD dip voltage greater than or equal to \( V_{IH_{min}} \) and a pulse width of less than or equal to 1 ns
  - All other LVC莫斯 Standards (2.5V, 1.8V, 1.5V, 1.2V) - A Yes (within guideline) is defined as follows:
    - A ground bounce voltage less than or equal to \( V_{IL_{max}} \) for ground bounce and a pulse width of less than or equal to 1 ns
    - A VDD dip voltage greater than or equal to \( V_{IH_{min}} \) 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”.

**Figure 72 · SSN Analyzer – Noise Report Tab**

**Right-click Menu Items**

The following menu items are available when you right-click an I/O. You can select multiple I/Os and then right-click to apply the menu items to all selected I/Os. Available menu items are:

- **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 only 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 proximity (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 not beginning with DATA are excluded from the list. Filtering allows you to focus on I/Os you are interested in for SSN Analysis.

**Pulse Width**
The Pulse Width is the settling time of the signal bounce. It is a threshold value which the signal bounce must exceed before the signal bounce is recognized for SSN calculation. Select 1ns or 0ns. Selecting 0ns means that any signal bounce with a pulse width above 0ns is recognized for SSN calculation. A selection of 1ns means only signal bounces with a pulse width at or above 1ns are recognized for SSN calculation. Changing the Pulse Width selection discards all the 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 only supported for the MPF300XT/FCG1152 die/package.

**Run Analysis**
This button is not active when SSN first opens. It is activated only when you have made changes in the Noise Report. These changes may include one or more of the following:
• Checking/unchecking the Don’t Care checkbox for one or more I/Os.
• Checking/unchecking the Static checkbox for one or more I/Os.
When you have made your changes, click **Run Analysis** and SSN will recompute the Noise Margin number.

**Save Report**
Click **Save Report** to save the Noise Report in one of three formats:
• Text – Text file with *.txt file extension
• CSV – Spreadsheet file with *.csv file extension
• XML – XML file with *.xml file extension

**Excluded I/Os**
This tab displays 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
The Noise Report includes these columns:

- Bank Name/Pin Number
- Port Name
- Instance Name
- I/O Standard
- Comment – Specifies the reason for exclusion, e.g., unsupported I/O Standards or Marked as Static I/Os

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.

**Summary**

The Summary tab displays a summary of the SSN Analyzer. Click **Save Summary** to save the summary in Text, CSV, or XML format.
User Action When SSN Noise Analyzer Reports Failure

When the SSN Noise Analyzer reports poor Noise Margin or Failure, take the following steps 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. After making these changes, 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 will reduce 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 75 · SSN Analyzer in the Design Flow
Configure Hardware

Programming Connectivity and Interface

In the Libero SoC Design Flow window, expand Configure Hardware and double-click Programming Connectivity and Interface to open the Programming Connectivity and Interface window. The Programming Connectivity and Interface window displays the physical chain from TDI to TDO or SPI Slave configuration.

The Programming Connectivity and Interface view enables the following actions for non-target devices:

- **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** - Automatically construct the physical chain
- **Add Microsemi Device** – Add a Microsemi device to the chain
- **Add Non-Microsemi Device** – Add a non-Microsemi device to the chain
- **Add Microsemi Devices From Files** – Add a Microsemi device from a programming file
- **Delete Selected Devices** – Delete selected devices in the grid
- **Scan and Check Chain** – Scan the physical chain connected to the programmer and check if it matches the chain constructed in the grid
- **Zoom In** – Zoom into the grid
- **Zoom Out** – Zoom out of the grid

Hover Information

The device tooltip displays the following device information if you hover your pointer over a device in the grid:

- **Name** - User-specified device name. If you have two or more identical devices in your chain you can use this field to give them unique names.
- **Device** - Device name.
- **File** - Path to programming file.
- **Programming action** – When a programming file is loaded, the user can select a programming action for any device which is not the Libero design device.
- **IR** - Device instruction length.
- **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.

Device Chain Details

The device within the chain has the following details:

- **Libero design device** – Has a red circle within Microsemi logo. Libero design device cannot be disabled.
- **Left/right arrow** – Move device to left or right according to the physical chain.
- **Enable Device** - Select to enable the device for programming; enabled devices are green, disabled devices are gray.
- **Name** - Displays your specified device name.
- **File** - Path to programming file.
Right-Click Options

- **Set as Libero Design Device** - The user needs to set Libero design device when there are multiple identical Libero design devices in the chain.
- **Configure Device** – Ability to reconfigure the device (for a Libero SoC target device the dialog appears but only the device name is editable).
- **Enable Device for Programming** - Select to enable the device for programming; enabled devices are green, disabled devices are gray.
- **Load Programming File** – Load programming file for selected device. (Not supported for Libero SoC target design device.)
- **Set Serial Data** - Opens the Serial Settings dialog box; enables you to set your serialization data.
- **Select and Configure Action/Procedures** (for devices other than the Libero SoC target design device):
  - Select an action to program - The selected action will be 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; enables you to customize the list of recommended and optional procedures for an action.
- **Move Device Left/Right** – Move device in the chain to left or right.

Figure 76 · Right-click Properties
Select and Configure Actions and Procedures

In the Programming and connectivity Interface, right-click a non-target device and choose **Select and Configure Actions/Procedures**. The Configure Actions and Procedures dialog box opens, allowing you to configure actions and procedures for devices other than the Libero SoC target design device and select an action to program in the Libero environment.

![Configure Actions and Procedures](image)

**Figure 77 · Select and Configure Actions and Procedures**

- **Select an action to program** - The selected action will be programmed in the Libero environment.
- **Configure actions and procedures:**
  - **Actions** - List of programming actions for your device.
  - **Procedures** - Advanced option; enables you to customize the list of recommended and optional procedures for an action.

**Note:** You cannot select an action or configure actions in the chain view for the target device. To configure actions for Libero target devices, use the Configure Actions and Procedures tool.

Programmer Settings

In the Libero SoC Design Flow window, expand **Configure Hardware**, double-click **Configure Programmer**, or right-click **Configure Programmer** and choose **Programmer Settings** to view the Programmer Settings dialog. For the JTAG interface, you can set specific voltage and force TCK frequency values for your programmer in this dialog. For the SPI Slave interface, you can set specific voltage and force SCK frequency values for your programmer in this dialog.

**Note:** SPI Slave mode is supported by FlashPro6 for PolarFire devices.
The Programmer Settings dialog includes setting options for FlashPro6/5/4/3/3X.

Limitation of the TCK frequency for the selected programmer:
- 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

TCK frequency limits by target device:
- Refer to target device data sheet

During execution, the frequency set by the FREQUENCY statement in the PDB/STAPL file overrides the TCK frequency setting selected by you in the Programmer Settings dialog box unless you also select the Force TCK Frequency checkbox.

Limitation of the SCK frequency for the selected programmer: 1.00, 2.00, 2.50, 3.33, 4.00, 5.00, 6.67, 8.00, 10.00, 13.33, 20.00 MHz
FlashPro5/4/3/3X Programmer Settings

For FlashPro5/4/3/3X, if you choose the 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. Discrete clocking should be used when there is a JTAG non-compliant device in a chain with Microsemi devices. After you have made your selections(s), click OK.

Default Settings

- The Force TCK Frequency option is unchecked to instruct the FlashPro5/4/3/3X to use the TCK frequency specified by the Frequency statement in the PDB/STAPL file(s).
- FlashPro5/4/3/3X default TCK mode setting is Free running clock.

Note: The “Set Vpump” checkbox has been removed. For older projects prior to Libero SoC v12.5, if the Set Vpump checkbox is selected, the warning "Set Vpump parameter is obsolete. VPUMP will not be sensed or driven for all devices." is displayed in the log window, when the design is opened for the first time in Libero SoC v12.5.

TCK Setting (ForceTCK Frequency)

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

Default TCK frequency

When the IPD/STAPL file or Chain does not exist, the default TCK frequency is set to 4MHz. When more than one Microsemi flash device is targeted in the chain, the FlashPro Express software passes through all of the files and searches for the "freq" keyword and the "MAX_FREQ" Note field. The FlashPro Express software uses the lesser value of all the TCK frequency settings and the "MAX_FREQ" Note field values.

Select Programmer

In the Libero SoC Design Flow window, expand Configure Hardware and double-click Select Programmer to open the Select Programmer dialog. You can also right-click Select Programmer to open it. The dialog displays the name, type, and port of your programmer if it is connected.

A drop-down list shows all connected programmers, allowing you to select the programmer you want. If no programmers are connected, you can connect a programmer without closing the dialog and then click Refresh/Rescan Programmers. The connected programmer will appear in the drop-down list.

Figure 79 · Select Programmer Dialog
Program Design

Generate FPGA Array Data

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

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

Double-click **Generate FPGA Array Data** or right-click **Generate FPGA Array Data** in the Design Flow window and click Run to generate FPGA Array Data. Before running this tool, the design should have completed the Place and Route step. If not, Libero SoC runs implicitly the upstream tools (Synthesis, Compile Netlist, and Place and Route) before it generates the FPGA Array Data.

![Design Flow Window](image)

Configure Design Initialization Data and Memories

User design blocks such as LSRAM, uSRAM, XCVR (transceivers), and PCIe can be initialized using data stored in non-volatile uPROM, sNVM, or external SPI Flash storage memory. The Configure Design Initialization Data and Memories tool allows you to define the specification of this Design Initialization sequence, and the specification of the initialization clients in separate tabs:

- **Design Initialization Specification**
- **Configure uPROM**
- **Configure sNVM**
- **Configure SPI Flash**
- **Configure Fabric RAMs Initialization**
**Note:** The Configure Design Initialization Data and Memories tool can be invoked only after successful completion of the **Generate FPGA Array Data** step.

Use the tabs in this GUI to configure the Design Initialization data and memories.

Once the Configuration is complete, follow these remaining steps to program the initialization data:

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

**Note:** While importing memory files, the option ‘Use relative path from project directory’ has been updated to ‘Use relative path’. The relative option was updated to either choose 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 to in various configurators as well as sNVM/Uprom/SPI-Flash update tools.

**See Also**

- Generate Design Initialization Data
- Configure Fabric RAMs Initialization
- Configure Programming Options
- Configure Security Wizard
Design Initialization Specification Tab

Figure 81 · Design Initialization Tab

Common Commands for All Configure Design Initialization Data and Memories Tabs

If a Configure Design Initialization Data and Memories Tab title has an asterisk (*) next to it, an item on that tab has been changed, but not yet "Applied". The common options on every tab page are:

- **Apply** - Click the **Apply** button to save the changes made in this tab.  
  **Note**: The Apply button only saves the configuration changes. For the initialization of the memory block to take effect, go back to the Device Initialization tab and click Generate Initialization Clients.

- **Discard** - Click the **Discard** button to discard any changes made in this tab.

First Stage (sNVM)

In the first stage, the initialization sequence de-asserts the FABRIC_POR_N signal and starts the I/O calibration routine. The initialization client for this stage is always placed in sNVM, and it uses the last two pages of the sNVM memory space. There are no options for this stage.

Second Stage (sNVM)

In the second stage, the initialization sequence initializes the PCIe and XCVR blocks present in the design. This stage is grayed out if the design does not have PCIe or XCVR Blocks. The initialization client for this stage is named INIT_STAGE_2_SNVM_CLIENT. It is always placed in sNVM at the start address of the user’s choice. The start address can only be at the start of an sNVM page (page boundary). Each sNVM page is 256 bytes in size, so the valid start addresses (Hex) are 0x0, 0x100, 0x200, and so on. Only the plain text non-authenticated client is supported for initialization.
Third Stage (uPROM/sNVM/SPI Flash)

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

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

- **uPROM** - For uPROM, the name of the initialization client is INIT_STAGE_3_UPROM_CLIENT. Its start address is at the user's choice, subject to the limitation that the start address can only be at the start of a UPROM block. Each UPROM block is 256 words, so the allowed start addresses (Hex) are 0x0, 0x100, 0x200, and so on.

- **sNVM** - For sNVM, the name of the initialization client is INIT_STAGE_3_SNVM_CLIENT. Its start address is at the user's choice, subject to the limitation that the start address can only be at the start of an sNVM page (page boundary). Each sNVM page is 256 bytes long, so the allowed start addresses (HEX) are 0x0, 0x100, 0x200, and so on.

- **SPI-Flash** - For SPI-Flash, the name of the initialization client is INIT_STAGE_3_SPIFLASH_CLIENT.

- **SPI-Flash Binding**: there are four Binding Options from which the user can select:
  1. No Binding Plaintext: When this option is selected, the <root>_uic.bin file is a script file that can be opened to see readable text.
  2. Binding Encrypted with Default Key: When this option is selected, the <root>_uic.bin file is encrypted with the default encryption key. Also, the design version is displayed and this design version can be modified from Configure Programming Options. When Default key is selected, the user does not need to specify any other details.
  3. Binding Encrypted with User Encryption Key 1 (UEK1): When this option is selected, the <root>_uic.bin file is encrypted with UEK1. Also, the design version is displayed and this design version can be modified from Configure Programming Options. The user needs to configure SPM along with UEK1. If UEK1 is not specified, the Generate SPI Flash Image and Export SPI Flash Image steps will result in error. UEK1 can be configured using the Configure Security Tool.
  4. Binding Encrypted with User Encryption Key 2 (UEK2): When this option is selected, the <root>_uic.bin file is encrypted with UEK2. Also, the design version is displayed and this design version can be modified from Configure Programming Options. The user must configure SPM along with UEK2. If UEK2 is not specified, the Generate SPI Flash Image and Export SPI Flash Image steps will result in error. UEK2 can be configured using the Configure Security Tool.

- **SPI Clock divider value**: The user can also select the SPI clock divider value using the adjacent drop-down menu to set the clock divider value for the clock that is generated by the System Controller. Choose the value that meets the minimum clock width requirement of the external SPI Flash. The allowed values are 1, 2, 4, or 6. The default value is 1.

- **Broadcast instructions to initialize RAM's to zero's** - This global option affects all RAM blocks (LSRAM and uSRAM) in the device. When this option is selected, all RAM blocks will be initialized to zeros before FABRIC PoR is asserted. To accommodate the additional instructions, the sNVM start page will be 202.

  **Note**: 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. In this case, you must wait for the corresponding RAM INIT complete signal before accessing those RAMs (refer to the PolarFire FPGA Device Powerup and Reset User Guide, which describes the INIT DONE/COMPLETE signals). The sNVM start page will be 219 when this global option is not selected.

**Time Out**

A time-out of up to 128 seconds can be selected from the drop-down menu for the completion of all three stages of initialization process. The default setting is 128.
Auto Calibration Time Out

The Auto Calibration Time Out value specifies the time out before which the IO Calibration instructions must be completed. The default value is 3000 milliseconds. This time out value is applicable only for MPF100T, MPF200T, MPF300T and MPF500T devices.

Custom Configuration File

The Custom Configuration File contains signal integrity parameters for Transceivers. Click the Browse button at the far right to navigate to and select a custom configuration file for Transceiver solutions. Contact Microsemi Tech Support for details.

See Also

Configure Fabric RAMs Initialization
Configure Security Wizard

Configure uPROM

Use the uPROM tab to manage and configure user-specific data clients targeted for uPROM memory.

Figure 82 - Configure uPROM Tab
Common Commands for All Configure Design Initialization Data and Memories Tabs

If a Configure Design Initialization Data and Memories Tab title has an asterisk (*) next to it, an item on that tab has been changed, but not yet "Applied". The common options on every tab page are:

- **Apply** - Click the Apply button to save the changes made in this tab.  
  **Note:** The Apply button only saves the configuration changes. For the initialization of the memory block to take effect, go back to the Device Initialization tab and click Generate Initialization Clients.
- **Discard** - Click the Discard button to discard any changes made in this tab.

Add

Use the Add button to add a uPROM client. When a uPROM client is added, it appears in the spreadsheet-like list.  
See [Add uPROM Client](configure_uprom)

Edit

Use the Edit button to edit the uPROM client. If there are multiple uPROM clients, first select the client from the spreadsheet-like list and then click Edit.

When changes are made to the configuration of any of the uPROM client, then their ‘Edited’ state is indicated by an asterisk (*) next to the uPROM tab’s title, and an asterisk (*) next to the title of the main window of the Design and Memory Initialization step. When the edits are saved, then the uPROM tab’s asterisk (*) disappears. All the edits present in all the tab can be saved in one go by clicking on the ‘Save’ icon on Libero’s toolbar. When there are no edits present in any of the tabs, then the asterisk (*) next to the title of the main window disappears.  
See [Add uPROM Client](configure_uprom)

Delete

Use the Delete button to delete an uPROM client. If there are multiple uPROM clients, first select the client and then click Delete.

Load Design Configuration

Click this button to load in the design’s original uPROM configuration file in <project>/component/work/UPROM.cfg. This button is grayed out if the design does not have an original uPROM configuration file. This configuration is changed whenever the design is updated in the design window. If there are changes made to this design configuration after the latest Apply, Libero SoC gives a clear visual indication that a newer design configuration is available by two means:

- an info icon appears next to the button ‘Load design configuration’,  
- an info icon is shown next to the uPROM tab title.

The tool-tip on both icons contains the time-stamp information of the design configuration file. The icons disappear after the user clicks Apply the next time.

Usage Statistics

Memory usage for the uPROM is reported in the pie chart.  
See Also  
configure_uprom
Add/Edit uPROM Client

Click Add or Edit to open a dialog to add/edit a uPROM client.

![Add Data Storage Client Dialog](image)

**Client name**

Enter the name of the uPROM client to be added

**Content from File**

Navigate to and specify a file, the content of which is to be used to fill the uPROM.

**Content filled with 0s**

Populates the uPROM with zero's.

**Start Address**

Specifies the start address (in HEX) of the uPROM client. If there are multiple uPROM clients, the start address must not overlap. A warning message appears if there is address overlapping of uPROM clients. Valid start addresses range from 0 to CBFF (Hex).
Number of 9-bit words

Specifies the number (in decimal) 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

This option is disabled and unavailable from the Design and Memory Initialization tool.

Use Content for simulation

This option is disabled and unavailable from the Design and Memory Initialization tool.

Configure sNVM

Use the sNVM tab to manage and configure sNVM clients.

Figure 84 · sNVM Tab
Common Commands for All Configure Design Initialization Data and Memories Tabs

If a **Configure Design Initialization Data and Memories** Tab title has an asterisk (*) next to it, an item on that tab has been changed, but not yet "Applied". The common options on every tab page are:

- **Apply** - Click the **Apply** button to save the changes made in this tab.
  
  **Note:** The Apply button only saves the configuration changes. For the initialization of the memory block to take effect, go back to the Device Initialization tab and click Generate Initialization Clients.

- **Discard** - Click the **Discard** button to discard any changes made in this tab.

**Add**

Click **Add** to open the Add Client dialog. Four different types of sNVM clients can be added:

- PlainText NonAuthenticated: 252 user bytes per page.
- PlainText Authenticated: 236 user bytes per page.
- CipherText Authenticated: 236 user bytes per page.
- USK: 96 user bytes. The USK client occupies exactly 1 page.

**Note:** Only one USK client in the sNVM is allowed.

When an authenticated client is present in the sNVM, a USK client must be necessarily present too.

**Adding Text Clients**

See **Add Text Client**

**Adding a USK Client**

See **Add USK Client**

**Edit**

Use the Edit button to edit the sNVM client's configuration. If there are multiple sNVM clients, first select the client you want and then click Edit.

When changes are made to the configuration of any of the sNVM client, then their 'Edited' state is indicated by an asterisk (*) next to the sNVM tab's title, and also an asterisk (*) next to the title of the main window of the Design and Memory Initialization step. When the edits are saved, then the sNVM tab's asterisk (*) disappears. All the edits present in all the tab can be saved in one go by clicking on the 'Save' icon on Libero's toolbar. When there are no edits present in any of the tabs, then the asterisk (*) next to the title of the main window disappears.

**Delete**

Use the Delete button to delete an sNVM client. If there are multiple sNVM clients, first select the client you want and then click Delete.

**Load Design Configuration**

Click this button to load in the design's original sNVM configuration file in `<project>/component/work/sNVM.cfg`. This button is grayed out if the design does not have an original sNVM configuration file. This configuration is changed whenever the design is updated in the design window. If there are changes made to this design configuration after the latest **Apply**, Libero SoC gives a clear visual indication that a newer design configuration is available by two means:

- an info icon appears next to the button 'Load design configuration',
- an info icon is shown next to the sNVM tab title.
The tool-tip on both icons contains the time-stamp information of the design configuration file. The icons disappear after the user clicks **Apply** the next time.

**Usage Statistics**

Memory usage for the sNVM is reported in the pie chart.

**See Also**

- configure_snvm
- PolarFire FPGA Programming User Guide

**Add sNVM Clients**

Two different kinds of sNVM clients can be added:

- Text Client
- USK Client

**Add Text client**

Use the dialog box to add text clients:

- PlainText NonAuthenticated: 252 user bytes per page.
- PlainTextAuthenticated: 236 user bytes per page.
- CipherTextAuthenticated: 236 user bytes per page.

![Add PlainText NonAuthenticated client](image)

*Figure 85 · Add Plain Text client*
Client name

Enter the name of the sNVM client to be added.

Content from File

Navigate to and specify a file, the content of which is to be used to fill the sNVM.

Content filled with 0s

Populates the sNVM with zero’s.

No Content

client is a placeholder and will not be programmed.

Start Page

Specifies the start page (in decimal) of the sNVM client. sNVM client address starts at page boundaries. If there are multiple sNVM clients, their start page cannot be the same. A warning message appears if there is address overlapping of sNVM clients. Valid start page range from 0 to 220 (Decimal).

Number of bytes

Specifies the total number (in decimal) 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. Valid ranges is from 1 to 47376

Use Content for simulation

Check if this client should be loaded for the simulation run.

Use as ROM

Check if this client should be used as read-only-memory (ROM).

Add USK client

This client holds the USK. It is always 1 page (96 bytes) in size. There is no total byte entry for the USK client.
Start Page

Start page can vary between 0 and 220.

USK Key

Enter a USK key (24 Hex characters). A random key can be generated by clicking the padlock icon to the right of this field.

Reprogram

Check if this client should be programmed.

Use Content for Simulation

Check if this client should be loaded for the simulation run.

Use as ROM

Check if this client should be used as read-only-memory (ROM).

Configure SPI Flash

The SPI Flash tab allows you to enable Auto Update, select the SPI Flash Manufacturer, and configure SPI Flash Clients. The configuration is saved in the spiflash.cfg file in the Libero design implementation folder.
Common Commands for All Configure Design Initialization Data and Memories Tabs

If a Configure Design Initialization Data and Memories Tab title has an asterisk (*) next to it, an item on that tab has been changed, but not yet "Applied". The common options on every tab page are:

- **Apply** - Click the **Apply** button to save the changes made in this tab.
  
  **Note:** The Apply button only saves the configuration changes. For the initialization of the memory block to take effect, go back to the Device Initialization tab and click Generate Initialization Clients.

- **Discard** - Click the **Discard** button to discard any changes made in this tab.

Enable Auto Update

Check Enable Auto Update to enable Auto Update on the target device. The bitstream generated within Libero will enable this feature. If this is checked, a total of two SPI Bitstreams can be added. One SPI Bitstream will be for Auto Update and the other will be for Recovery/Golden. The tool enforces the Recovery/Golden bitstream to be at index 0 and the Auto Update bitstream to be at Index 1. The Auto Update Bitstream Design version must be greater than the Design Version of the Recovery/Golden bitstream.

Manufacturer

Click the pull-down menu to see the supported SPI Flash manufacturer/vendors and the part number. The table below lists the supported vendors and part number.

<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Part Number</th>
<th>Capacity</th>
<th>Sector Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>MICRON</td>
<td>MT25QL01GBBB8ESF-0SIT</td>
<td>1GB</td>
<td>4KB</td>
</tr>
</tbody>
</table>

The SPI Flash Part Number is displayed to the right of the manufacturer/vendor name. The Memory size (in MB) for the SPI Flash is displayed in the Usage Statistics above the pie chart.

**Note:** This version of the programmer does not support SPI Flash security. Device security options such as "Hardware Write Protect" should be disabled for the External SPI Flash device.
Usage Statistics

Available Memory (MB) reflects the SPI Flash vendor and part selected.
- MICRON

Memory usage for the SPI Flash is reported in the pie chart.

SPI Flash Clients

SPI flash clients appear in a spreadsheet-like format when they are added and configured.

Figure 88 · SPI Flash Clients

Note: Bypass Back Level protection feature is only supported for SPI Bitstream clients for Recovery/Golden.

SPI Bitstream Client for Recovery/Golden

A SPI client for Recovery/Golden is required if a SPI Bitstream is added. There can only be one SPI Bitstream configured as Recovery/Golden. It is highlighted in yellow in the spreadsheet-like display. An error message appears if none is configured or more than one is configured.

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.

The master file should not be used for Recovery/Golden client with IAP.

If Back Level Protection is enabled in Configure Security tool, then Programming Recovery will fail 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 in this situation, you can import a Bitstream that has been exported with the Bypass Back Level Protection option.

Index 0 is reserved for this client.

SPI Bitstream Client for Auto Update

This client is highlighted in green in the spreadsheet-like display. To add a SPI Client for Auto Update, the Enable Auto Update checkbox must first be checked. 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 will reject a Bitstream file with Bypass Back Level Protection enabled for this type of client.

SPI Bitstream Client for IAP

To add a SPI Client for IAP, select the SPI Bitstream file for IAP in Add/Edit SPI Bitstream Client window. The total number of SPI Bitstream Clients allowed including Recovery/Golden and Auto Update Clients is 255.
Index for this client can be range of 2-255.

Data Storage Client
See add_data_storage_client to add a Data Storage Client. Bypass Back Level Protection feature is not applicable to Data Storage Client. Also, they don't have a Design Version number and an Index value.

Add

Click Add to add a SPI Bitstream client or Data Storage client. A total of up to 255 SPI Bitstream clients (including one client for Recovery/Golden and one client for Auto Update and the rest for IAP) can be added. One of the clients must be the Recovery/Golden client.

Edit

Click Edit to modify the configuration of the SPI Bitstream client or Data Storage client. If there are multiple clients in the list, select the client you want to modify and click Edit.

Delete

Use Delete to delete a SPI Flash client. If there are multiple SPI Flash clients, first select the client you want to delete and then click Delete.

See Also
Add/Edit SPI Bitstream Client
Add Data Storage Client
Configure SPI Flash
PolarFire FPGA Programming User Guide

Add/Edit SPI Bitstream Client

Click the Add button in the SPI Flash tab of the Configure Design Initialization Data and Memories tool to add a SPI Bitstream client and the Edit button to modify an existing SPI Bitstream client.
Name
Enter the name of the SPI Bitstream client to be added. Up to 32 alphanumeric characters are allowed. When editing an existing SPI Flash client, the client name cannot be changed.

Content
Check one of the following to select the type of SPI Bitstream to be added. A total of 255 SPI clients can be added for IAP.

SPI Bitstream for IAP
Check this checkbox to add a SPI bitstream client for IAP.

SPI Bitstream for Recovery/Golden
Check this checkbox to add a client for Recovery/Golden. It is mandatory and only one is allowed. This option is disabled if one is already added. The existing one can be edited or deleted.

SPI Bitstream for Auto Update
This is available only when Auto Update is checked in the Configure SPI Bitstream tab. The SPI bitstream client is optional and only one is allowed. This option is disabled if one is already added. The existing one can be edited or deleted.
**Browse Button**
Click the Browse button to navigate to a SPI file (*.spi) location. The content of this file is used to export the SPI Bitstream Image file.

**Filled with 1s**
Check this checkbox to fill the content of the SPI Bitstream client with 1s. If the content is filled with 1s, specify a client size. It must be greater than 0. Golden or Auto Update SPI clients cannot be filled with zeros. They require a content file.

**Start Address (HEX)**
The first available start address is 0x400 (the first 1024 bytes are reserved for the SPI directory and are not available).

**Size in bytes (decimal)**
This field displays the number of bytes (in decimal) of the client based on the specified bitstream (*.spi) file used to load in the content. The size can be increased but not decreased.
A new client is validated against existing clients. Address overlapping of clients is not allowed and is flagged as an error.

**See Also**
- set_client
- PolarFire FPGA Programming User Guide

---

**Add/Edit Data Storage Client for SPI Flash**
Click the **Add** button in the SPI Flash tab of the Configure Design Initialization Data and Memories tool to add a Data Storage client. Click the **Edit** button to modify an existing Data Storage client.

![Add Data Storage Client (SPI Flash Tab)](image)

**Figure 90** · Add Data Storage Client (SPI Flash Tab)
Note: Intel Hex file type is supported for this release.

Name
Enter the name of the Data Storage client to be added. Up to 32 alphanumeric characters are allowed. When editing an existing Data Storage client, the client name cannot be changed.

Content
Select one of the following options.

Memory file
Enter the name of the Memory file or click the Browse button to navigate to a Memory file location. Currently, only Intel-Hex format memory files are supported. The memory file will be loaded into the SPI-Flash at the desired start address.

Filled with 1s
Check this checkbox to fill the content of the Data Storage client with 1s. If the content is filled with 1s, specify a client size. It must be greater than 0.

Start Address (HEX)
The first available start address is 0x400 (the first 1024 bytes are reserved for the SPI directory and are not available).

Size in bytes (decimal)
This field displays the number of bytes (in decimal) of the client based on the specified Data Storage file used to load in the content. The size can be increased but not decreased.

A new client is validated against existing clients. Address overlapping of clients is not allowed and is flagged as an error.
Configure Fabric RAMs Initialization

The Fabric RAMs Initialization tab allows you to select the Initialization options for the memory blocks in the design. The memory blocks include:

- Dual-Port SRAM
- Two-Port SRAM
- uSRAM

![Fabric RAMs Initialization](image)

Figure 92 · Fabric RAMs Initialization

Common Commands for All Configure Design Initialization Data and Memories Tabs

If a Configure Design Initialization Data and Memories Tab title has an asterisk (*) next to it, an item on that tab has been changed, but not yet "Applied". The common options on every tab page are:

- **Apply** - Click the Apply button to save the changes made in this tab.
  
  **Note**: The Apply button only saves the configuration changes. For the initialization of the memory block to take effect, go back to the Device Initialization tab and click Generate Initialization Clients.

- **Discard** - Click the Discard button to discard any changes made in this tab.

Fabric RAM Clients Configuration:

- Click on Load design configuration to reset all Fabric RAM clients to their first configuration (the values specified by the first Apply for each client). This overrides any subsequent Apply commands that have been made.

- Click the Initialize all clients from pull-down menu to select from:
  
  - sNVM
  - uPROM
  - SPI-Flash
  - User Selection

- Instruction commands generated by Libero to initialize the memory blocks (Fabric RAM clients) can be saved in any of the storage types (sNVM, uPROM, or SPI-Flash). As a convenience, however, the user can select to initialize all Fabric RAM clients from the same storage type. If you select any one of the storage types from this pick-list, the instruction commands to initialize all clients are generated in sNVM,
uPROM, or SPI, as you selected. Choosing **User Selection** indicates that each Fabric RAM Client will be configured separately.

**Fabric RAM Client Table**

Each row in the Fabric RAM Client Table provides information about a Fabric RAM Client. The first three items are information-only, and cannot be modified from this dialog box:

1. Logical Instance Name
2. PortA Depth x Width
3. PortB Depth x Width

Items 4-6 are configurable; either by the **Initialize all clients** mechanism previously described for all clients, or by individual **User Selection** configuration:

4. Memory Content
5. Storage Type
6. Memory Source

To configure a Fabric RAM client with client-specific initialization options, double-click the Fabric RAM client row (or right-click on the row and select **Edit...** or single click on the row and click on the **Edit** button above the table). This will bring up an **Edit Fabric RAM Initialization Client** dialog box (see below) for this particular client.

**Note:** If you change the **Storage Type** for a specific client to something other than that previously chosen for all clients, this will also change the **Initialize all clients from** value to **User selection**.

**Edit Fabric RAM Initialization Client**

![Edit Fabric RAM Initialization Client](image)

**Figure 93 · Edit Fabric RAM Initialization Client Dialog Box**

**Dialog Box Sections:**

- **Client Name** - Information only. Cannot be changed in this dialog box
- **Physical Name** - Information only. Cannot be changed in this dialog box
- **RAM Initialization Options**:
  - **No Content** - The memory block is not initialized.
  - **Content Filled with Zeros** - The memory block is filled with zeros for initialization.
  - **Memory File** - Click the Browse button to navigate to the location of a memory file and import the file to the memory block. By default, the same memory file as specified in the memory configurator
is used. The supported memory file formats are Intel-HEX (*.hex), Motorola (*.s), Simple-Hex (*.shx), or Microsemi-Binary (*.mem).

- **Optimize for** - Information only. Cannot be changed in this dialog box.
- **Storage Type** - User can select from the available Storage Types:
  - sNVM
  - uPROM
  - SPI-Flash

**See Also**

Configure Design Initialization Data and Memories

**Configure eNVM**

This feature is only available for PolarFire SoC.

Use the envm tab to manage and configure envm clients. The maximum number of pages supported is 512.

![Figure 94 · ENVM tab in Design and Memory Initialization](image)

Addition of eNVM clients and supported file formats are the same as sNVM.
Generate Design Initialization Data

To generate the initialization clients, do one of the following in the Design Flow window:

- Double-click **Generate Design Initialization Data**

  or

- Right-click **Generate Design Initialization Data** and choose **Run**

Libero SoC carries out the following three actions:

- Generates the memory files corresponding to the three stages of the initialization sequence.
- Removes all pre-existing initialization clients.
• Creates the initialization clients for each stage and places them in their target memories.
  o The first stage initialization client is always created, and is always placed in sNVM. It is always placed at start address 0xDC00 (page 220).
  o The second stage initialization client is created only when there are PCIe blocks present in the design. It is always placed in sNVM. It is placed at the start address specified by the user in the ‘Design Initialization’ tab of the ‘Configure Design Initialization Data and Memories’ tool.
  o The third stage initialization client is created only when there are Fabric RAMs in the design or non-PCIe transceiver blocks in the design. It can be placed in any of uPROM, sNVM, or SPI memories at the user-specified start address. The user can specify both the target memory and the target start address in the ‘Design Initialization’ tab of the ‘Configure Design Initialization Data and Memories’ tool.

See Also
Configure Design Initialization Data and Memories
generate_design_initialization_data

Configure Actions and Procedures
The Configure Actions and Procedures tool allows you to configure any action with optional or recommended procedures for a Libero target device. The information is saved and is used by the Run Action tool.

Notes:
• 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 this tool, from the Libero Design Flow window, expand Program Design and double-click Configure Actions and Procedures.
The Configure Actions and Procedures dialog box opens. See the following example figure.

**Figure 96 · Configure Actions and Procedures Dialog Box Example**

*Note:* The actions and procedures shown will vary depending on the device family you are using and the bitstream components selected in the Generate Bitstream and Configure Options tools.

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 by the user.
- Optional procedures are shown in blue, and can be included or excluded by the user.
<table>
<thead>
<tr>
<th>Action</th>
<th>Procedures</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEVICE_INFO</td>
<td>INIT_VARIABLES_FOR_ACTION, SET_DEVICE_INFO_ACTIONTYPE, VERIFY_IDCODE, DO_DEVICE_INFO, DO_EXIT</td>
</tr>
<tr>
<td>ENC_DATA_AUTHENTICATION</td>
<td>INIT_VARIABLES_FOR_ACTION, SET_AUTHORIZATION_ACTIONTYPE, VERIFY_IDCODE, DO_AUTHENTICATION, DO_EXIT</td>
</tr>
<tr>
<td>ERASE</td>
<td>INIT_VARIABLES_FOR_ACTION, SET_ERASE_ACTIONTYPE, VERIFY_IDCODE, PROC_ENABLE, DO_ERASE, DO_EXIT</td>
</tr>
<tr>
<td>PROGRAM</td>
<td>INIT_VARIABLES_FOR_ACTION, SET_PROGRAM_ACTIONTYPE, VERIFY_IDCODE, PROC_ENABLE, DO_PROGRAM, DO_VERIFY (optional), DO_EXIT</td>
</tr>
<tr>
<td>READ_IDCODE</td>
<td>INIT_VARIABLES_FOR_ACTION, SET_READ_IDCODE, VERIFY_IDCODE, PRINT_IDCODE, DO_EXIT</td>
</tr>
<tr>
<td>VERIFY</td>
<td>INIT_VARIABLES_FOR_ACTION, SET_VERIFY_ACTIONTYPE, VERIFY_IDCODE, PROC_ENABLE, DO_VERIFY, DO_EXIT</td>
</tr>
<tr>
<td>VERIFY_DIGEST</td>
<td>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</td>
</tr>
<tr>
<td>ZEROIZE_LIKE_NEW</td>
<td>INIT_VARIABLES_FOR_ACTION, VERIFY_IDCODE, DO_ZEROIZE_LIKE_NEW, DO_EXIT</td>
</tr>
</tbody>
</table>
### Action Procedures

<table>
<thead>
<tr>
<th>Action Procedures</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT_VARIABLES_FOR_ACTION</td>
</tr>
<tr>
<td>VERIFY_IDCODE</td>
</tr>
<tr>
<td>DO_ZEROIZE_UNRECOVERABLE</td>
</tr>
<tr>
<td>DO_EXIT</td>
</tr>
</tbody>
</table>

The table below lists programming file actions and descriptions.

#### Table 3 · Programming File Actions

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROGRAM</td>
<td>Programs all selected family features: FPGA Array, targeted sNVM clients, targeted eNVM clients, and security settings.</td>
</tr>
<tr>
<td>ERASE</td>
<td>Erases the selected family features: FPGA Array and Security settings.</td>
</tr>
<tr>
<td>VERIFY_DIGEST</td>
<td>Calculates the digests for the components (Custom Security, Fabric, or sNVM or eNVM) included in the bitstream and compares them against the programmed values.</td>
</tr>
<tr>
<td>VERIFY</td>
<td>Verifies all selected family features: FPGA Array, targeted sNVM clients, targeted eNVM clients, and security settings.</td>
</tr>
<tr>
<td>ENC_DATA_AUTHENTICATION</td>
<td>Encrypted bitstream authentication data.</td>
</tr>
<tr>
<td>READ_IDCODE</td>
<td>Reads the device ID code from the device.</td>
</tr>
<tr>
<td>DEVICE_INFO</td>
<td>Displays the IDCODE, the design name, the checksum, and device security settings and programming environment information programmed into the device.</td>
</tr>
</tbody>
</table>

#### Options Available in Programming Actions

The table below shows the options available for specific programming actions.

#### Table 4 · Programming File Actions - Options

<table>
<thead>
<tr>
<th>Action</th>
<th>Option and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROGRAM</td>
<td>DO_VERIFY - Enables or disables programming verification</td>
</tr>
<tr>
<td>VERIFY_DIGEST</td>
<td>DO_ENABLE_FABRIC - Includes Fabric and Fabric configuration in the digest check</td>
</tr>
<tr>
<td></td>
<td>DO_ENABLE_SNVM - Includes the sNVM in the digest check</td>
</tr>
<tr>
<td></td>
<td>DO_ENABLE_ENVM – Includes the eNVM in the digest check</td>
</tr>
<tr>
<td></td>
<td>DO_ENABLE_SECURITY - Includes security policy settings, and UPK1, UEK1, User Key Set 2 (UPK2 and UEK2), DPK and SMK security segments in the digest check</td>
</tr>
</tbody>
</table>
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.

Configure I/O States During JTAG Programming

In the Libero SoC Design Flow window expand Program Design and double-click Configure I/O States During JTAG Programming to specify the I/O states prior to programming. This feature is only available once Layout is completed.

The default state for all I/Os is Tri-state.

Note: This feature is not supported for PolarFireSoC and the Libero SoC v12.5 will hardcode all the I/Os to tri-state.

To specify I/O states during programming:

1. Sort the pins as desired by clicking any of the column headers to sort the entries by that header. Select the I/Os you wish to modify (as shown in the figure below).

2. Set the I/O Output state. You can set Basic I/O settings if you want to use the default I/O settings for your pins, or use Custom I/O settings to customize the settings for each pin. See the Specifying I/O States During Programming - I/O States and BSR Details help topic for more information on setting your I/O state and the corresponding pin values. Basic I/O state settings are:
   - 1 – I/O is set to drive out logic High
   - 0 – I/O is set to drive out logic Low
   - Last Known State: I/O is set to the last value that was driven out prior to entering the programming mode, and then held at that value during programming
   - Z - Tri-State: I/O is tristated with weak pull up (10k ohm)

3. Click OK to save your settings.

Note: I/O States During programming will be used during programming or when exporting the bitstream.
Configure Programming Options

Sets your Design version, Back Level version and Silicon signature. 

**Design name** is a read-only field that identifies your design.

**Design version (number between 0 and 65535)** - Specifies the Design version to be programmed to the device. This value is also used for Back Level protection in "Update Policy" on page 130 of the Configure Security tool.

**Back Level version (number between 0 and 65535)** - Specifies the Back Level version to be programmed to the device. This has to be always less than or equal to Design version number. A warning message appears if the Back Level version is equal to the Design version and an info message appears if it is less than the Design version. Move your mouse over warning or info icon to view the message respectively. This value is used for Back Level protection (if enabled) in Update Policy of the Configure Security tool.

**Silicon signature (max length is 8 HEX chars)** - 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.

![Configure Programming Options Dialog Box](image)

**Figure 98 · Configure Programming Options Dialog Box**

**Notes**

SPI file programming for Auto Programming, Auto Update (IAP) and IAP/ISP Services currently can only program security once with the master file. Update files cannot update the Security settings. In addition, Silicon signature, and Tamper Macro can only be programmed with the master file and cannot be updated.

Configure Security

**Configure Security Wizard**

The Configure Security Wizard is a GUI-based wizard that guides the user step by step on how to configure custom security settings. The wizard has five steps executed in this sequential order:

1. **User keys**
2. **Update Policy**
3. **Debug Policy**
4. **Microsemi Policy**
5. **JTAG/SPI Slave Commands**
Summary Window

The summary window displays the summary of the current configuration settings. The window will scroll to the current page as you move from page to page.

Security Key Mode

Two security key modes are available.

- **Bitstream encryption with default key**
  This mode uses the default encryption key for security. The Next and Back button are disabled. All steps are disabled. Custom User Keys and security settings are disabled.

- **Custom security Mode**
  This mode allows you to configure custom security keys and settings. All steps are enabled. The Next and Back button are enabled.

Back

Click **Back** to return to the previous step.
Next

Click **Next** to proceed to the next step.

Finish

Click **Finish** to skip steps and complete the configuration.

**User Keys**

The User Keys are configured in this step. All keys are 256 bits (64 HEX characters).

![Configure Security Wizard](image)

**FlashLock/UPK1**

FlashLock/UPK1 is enabled by default. This key 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.
User Encryption Key 1 (UEK1)

UEK1 is enabled by default. Click Disable if you want to disable it. If enabled, the 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. Use UEK1 for updating the Fabric, uPROM, and sNVM or disable it.

User Encryption Key 2 (UEK2)

UEK2 is enabled by default. Click Disable if you want to disable it. If enabled, the 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. Use UEK2 as a second encryption key for updating the Fabric, uPROM, and sNVM or disable it.

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.

Update Policy

Field updates are enabled by default. Use this page to disable field updates and to specify field-update protection parameters.
Fabric/sNVM update protection

Two options are available:

- Disable Erase/Write operations
  
  WARNING: The field update STAPL files (_uek1/_uek2) will include plain-text FlashLock/UPK1

- Updates allowed using user defined encryption keys; FlashLock/UPK1 is not required for updates
Enable Back Level protection

When enabled, a field update design being programmed must be of a version higher than the Back Level version value in the programmed device. This protection will prevent field update designs with back level versions less than or equal to the design version programmed in the device.

**Design version (number between 0 to 65535)**
Displays the current Design version (as set in the Configure Programming Options tool).

**Back Level version (number between 0 to 65535)**
Displays the current Back level version (as 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, then Generate Bitstream and Export Bitstream tools will error out.

The following are the examples with Back Level protection enabled in the Configure Security tool:

**Example 1: Programming the Back Level version to the same version as the Design version**

<table>
<thead>
<tr>
<th>Step</th>
<th>Bitstream</th>
<th>Action</th>
<th>Bitstream Design version</th>
<th>Bitstream Back Level version</th>
<th>Device Back Level version</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>_master</td>
<td>PROGRAM</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Pass</td>
</tr>
<tr>
<td>2</td>
<td>_master</td>
<td>VERIFY</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Pass, if device has UPK1</td>
</tr>
<tr>
<td>3</td>
<td>_master</td>
<td>ERASE</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Pass, if device has UPK1</td>
</tr>
<tr>
<td>4</td>
<td>_master</td>
<td>AUTHENTICATE</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Pass, if device has UPK1</td>
</tr>
<tr>
<td>5</td>
<td>_master</td>
<td>DEVICE_INFO</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Always passes</td>
</tr>
<tr>
<td>6</td>
<td>_uek1</td>
<td>PROGRAM</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>Pass</td>
</tr>
<tr>
<td>7</td>
<td>_master</td>
<td>PROGRAM</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>Fail, due to back level protection</td>
</tr>
</tbody>
</table>

**Example 2: Programming the Back Level version less than the Design version**

<table>
<thead>
<tr>
<th>Step</th>
<th>Bitstream</th>
<th>Action</th>
<th>Bitstream Design version</th>
<th>Bitstream Back Level version</th>
<th>Device Back Level version</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>_master</td>
<td>PROGRAM</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>Pass</td>
</tr>
<tr>
<td>2</td>
<td>_uek1</td>
<td>PROGRAM</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>Pass</td>
</tr>
<tr>
<td>3</td>
<td>_uek1_1</td>
<td>PROGRAM</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>Pass</td>
</tr>
<tr>
<td>4</td>
<td>_uek1_2</td>
<td>PROGRAM</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>Pass</td>
</tr>
</tbody>
</table>
### Step Bitstream Action Bitstream Design version Bitstream Back Level version Device Back Level version Result

<table>
<thead>
<tr>
<th>Step</th>
<th>Bitstream</th>
<th>Action</th>
<th>Bitstream Design version</th>
<th>Bitstream Back Level version</th>
<th>Device Back Level version</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>_master</td>
<td>PROGRAM</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>Fail, due to back level protection</td>
</tr>
<tr>
<td>6</td>
<td>_uek1</td>
<td>PROGRAM</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>Fail, due to back level protection</td>
</tr>
<tr>
<td>7</td>
<td>_uek1_1</td>
<td>PROGRAM</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>Fail, due to back level protection</td>
</tr>
<tr>
<td>8</td>
<td>uek1_2</td>
<td>VERIFY</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>Pass</td>
</tr>
</tbody>
</table>

### Disable programming interfaces

The following programming interfaces can be disabled:

- Auto Programming and IAP services
- JTAG
- SPI Slave

Disabling of all three interfaces is not allowed and an error message appears if all three interfaces are disabled.

### Disable Bitstream Programming Actions (JTAG/SPI Slave)

- Program
- Authenticate
- Verify

**WARNING:** The field update STAPL files (_uek1/_uek2) will include plain-text FlashLock/UPK1

The summary at the top of the wizard summarizes the result of the selection.

### Reset to Default

Reset the options to default values. All options are unchecked.

### Debug Policy

The Debug Policy page allows you to configure Debug Protections. By default, debugging is enabled.

### Debug with DPK (Debug Pass Key) - Optional

Protect Debug with a 256-bit (64-character HEX) Debug Pass Key. Enter the key in the field or click the padlock icon at the far right to generate a random key. This key is optional if you want a separate passkey to enable access to disabled debug features during one debugging session.

If the DPK key is entered, then at least one option must be checked.
SmartDebug Access Control

All of 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.

Three additional options are:

- Disable UJTAG command through JTAG Interface
- Disable JTAG (1149.1) boundary scan
  Disables JTAG (1149.1) commands. The following JTAG commands will be disabled: HIGHZ, EXTEST, INTEST, CLAMP, SAMPLE, and PRELOAD. I/Os will be tri-stated when in JTAG programming mode and BSR control during programming is disabled. BYPASS, IDCODE, and USERCODE instructions will remain functional.
- Disable reading temperature and voltage sensor (JTAG/SPISlave)

The summary at the top of the page displays the results of the selection.

**Microsemi Factory Access Policy**

The page allows you to configure the policy for Microsemi Test Mode Access. Test mode access is required for Failure Analysis on the device.
Two options are available:

• Allow factory test mode access (default setting).

  **WARNING:** This is not recommended for production devices.

• Disable factory test mode access

  **NOTE:** Use FlashLock/UPK1 to change access level

### JTAG/SPI Slave Command Policy

The page allows you to configure the policy for JTAG/SPI Slave User Commands. Three options are available. Enabled is the default setting for all three options. Click the checkbox to disable any of the settings.

• Disable all external access to PUF emulation through JTAG/SPI Slave

• Disable external Fabric/sNVMe digest requests through JTAG/SPI Slave

  **WARNING:** Repeated external Fabric digest calculations can impact device reliability. View Datasheet for additional information.

• Disable external zeroization through JTAG/SPI Slave

  **WARNING:** It is not recommended to leave zeroization enabled for production devices
Security Features Frequently Asked Questions

I have configured the Security Wizard and enabled security in my design but I do not want to program my design with the Security Policy Manager features enabled. What do I do?

Go to Configure Bitstream and uncheck Security.

What is programmed when I click Program Device?

All features configured in your design and enabled in the Configure Bitstream tool. Any features you have configured (such as sNVM or Security) are enabled for programming by default.

When I click Program Device is the programming file encrypted?

All programming files are encrypted. To generate programming files encrypted with UEK1 or UEK2 you must generate them from Export Bitstream for field updates.

Note: Once security is programmed, you must erase the security before attempting to reprogram the security.

How do I generate encrypted programming files with User Encryption Key 1/2?

- Configure the Security Wizard and specify User Key Set 1, User Key Set 2. Ensure the Security programming feature is enabled in Configure Bitstream; it is enabled by default once you configure the Security Policy Manager.

- Export Bitstream from Handoff Design for Production - <filename>_uek1.(stp/spi/dat), <filename>_uek2.(stp/spi/dat) files are encrypted with UEK1, UEK2, respectively. See Security Programming File Descriptions below for more information on programming files.

What are Security Programming Files?

See the Security Programming Files topic for more information.
Configure Permanent Locks for Production

Configure Permanent Locks for Production is a GUI-based tool that guides the user step by step on how to configure the Permanent Locks for Production. The wizard has six steps/pages executed in sequential order. One Time Programmable (OTP) settings in the Permanent Locks page are applied to configured Security settings from the Configure Security tool. The subsequent pages have read only fields, which will be affected by Permanent Lock settings. These settings can only be configured by the Configure Security tool.

If you configure any Permanent Lock settings, you will be forced to go through each page to review the Security settings to make sure they are as desired. The settings cannot be changed once they have been programmed.

**Note:** This feature is not supported for PolarFire SoC devices.

1. Permanent Locks
2. User keys in Configure Security
3. Update Policy in Configure Security
4. Debug Policy in Configure Security
5. Microsemi Factory Access in Configure Security
6. JTAG/SPI Slave Commands Policy in Configure Security

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 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.

**Permanent Locks**

This page allows you to configure Permanent Locks for Production programming. Permanent Locks should be configured after the Design/Debug phase has been completed. The Permanent Lock settings will not be applied to programming within Libero. They are only applied to the Export tools used for Production programming. Once the Permanent Locks are programmed, they cannot be changed. Configuring the Permanent Locks will affect the settings on the subsequent pages and should be reviewed carefully. The settings cannot be changed once they are programmed.
All the user keys and security policies are protected with FlashLock and can be made One-Time Programmable, by configuring the Permanent Lock settings.

You can select one or more of the below options to be locked permanently:

- **Permanently disable UPK1** - This will permanently disable FlashLock/UPK1 from being able to be matched by the device. Any feature that is disabled will be permanently disabled. Any feature that is available will be permanently available.

- **Permanently disable UPK2** - This will permanently disable FlashLock/UPK2 from being able to be matched by the device. If UEK2 is enabled and selected for programming, then it cannot be exchanged.

- **Permanently disable SmartDebug access control and reading temperature and voltage sensor** - This will permanently disable SmartDebug access control for user debug and active probes, live probes and, sNVM along with the ability to read the temperature and voltage sensor.

- **Permanently disable Debug Pass Key(DPK)** - This will permanently disable the FlashLock/DPK from being able to be matched by the device. If DPK was programmed, then it can no longer be used for SmartDebug access.

- **Permanently write-protect Fabric** - This will make the Fabric One-Time Programmable. Verify of the Fabric will still be possible. Erase/Program of the Fabric is permanently disabled.

- **Permanently disable Microsemi factory test mode access** - This will permanently disable Microsemi factory test mode access. Microsemi will not be able to perform a Failure Analysis on this device.

- **Permanently disable Auto Programming, JTAG and SPI Slave programming interfaces** - This will permanently disable all programming interfaces. The actual JTAG and SPI Slave ports are disabled and you
cannot access the device for any operations including reading the IDCODE of the device. The device will become One-Time Programmable and there will be no way to Erase/Program/Verify the device.

The summary window displays the summary of the current configuration set in Permanent Locks page. The summary can be exported to a file and is recommended to keep a record of the settings configured. It is important to review the summary carefully and ensure the settings and behaviors are as expected since they cannot be changed once the device is programmed.

Remove Permanent Locks

You can remove the Permanent Lock settings by either right-clicking on the tool in the Design Tree or by clicking the button in the UI.

When selected removes all the Permanent Locks selected and restores to initial security settings configured in Configure Security tool. This option is highlighted only when at least one of the Permanent Locks is enabled.

See Also
Configure Security
Tcl command for Permanent Locks
remove_permanent_locks

Configure Bitstream

Right-click Generate Bitstream in the Design Flow window and choose Configure Options to open the Configure Bitstream dialog box.

The Configure Bitstream dialog box enables you to select which components you wish to program. Only features that have been added to your design are available for programming.

If the design includes uPROM, it will be included in the Fabric.

Note: eNVM option will available only if eNVM clients are present and are being programmed and is available for PolarFire SoC devices only.

![Configure Bitstream Dialog Box - PolarFire](image)

**Notes:**
- Custom security is enabled if security was configured.
- All available features are selected by default.
- sNVM is always programmed with Fabric.

See Also
Generate Bitstream
Generate Bitstream

Generates the bitstream for use with the Run PROGRAM Action tool. The tool incorporates the Fabric design, sNVM configuration, eNVM configuration (if configured) and custom security settings (if configured) to generate the bitstream file. You need to configure the bitstream before you generate the bitstream. Otherwise, default settings with all available features included will be used. Right-click Generate Bitstream and choose Configure Options to open the Configure Bitstream dialog box to select which components you wish to program. Only features that have been added to your design are available for programming. If the design includes uPROM, it will be included in the Fabric. eNVM option will available only if eNVM clients are present and are being programmed and is available for PolarFire SoC devices only.

Modifications to the Fabric design, sNVM configuration, eNVM configuration or security settings will invalidate this tool and require regeneration of the bitstream file. The Fabric programming data will only be regenerated if you make changes to the Fabric design, such as in the Create Design, Create Constraints and Implement Design sections of the Design Flow window.

When the process is complete a green check appears next to the operation in the Design Flow window (as shown in the figure below) and information messages appear in the Log window.

See also

Configure Bitstream Dialog Box

Run Programming Device Actions

If you have a device programmer connected, you can double-click Run PROGRAM Action to execute your programming in batch mode with default settings. If your programmer is not connected, or if your default settings are invalid, the Reports view lists the error(s). To select a programming action to run:

1. Right-click Run PROGRAM Action and choose Select Action. The Select Action dialog box opens.
Figure 108 · Select Action Dialog Box

2. Select a programming action from the drop-down list and click **OK**.

To configure programming actions, use the Configure Actions and Procedures tool.

**See Also**
Configure Actions and Procedures

**Programming File Actions**

Libero SoC enables you to program security settings, FPGA Array, sNVM and eNVM features.

**Note:** eNVM features are available for PolarFire SoC devices only.

**Note:** If the design includes UPROM, it will be included in the Fabric.

You can program these features separately using different programming files or you can combine them into one programming file.

In the Design Flow window, expand Program Design, click Run PROGRAM Action, and right-click Select Action.

The Select Action dialog box opens. See the following example figure.
For details about configuring actions and procedures, see Configure Actions and Procedures.

### Exit Codes (PolarFire)

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Exit Message</th>
<th>Exit Code</th>
<th>Possible Cause</th>
<th>Possible Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passed (no error)</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>Monitor related power supplies that cause the issue during programming; check for transients outside of Microsemi specifications. See your device datasheet for more information on transient specifications.</td>
</tr>
<tr>
<td>0x8002</td>
<td>Failed to disable programming mode</td>
<td>5</td>
<td>Unstable voltage level</td>
<td>Monitor JTAG supply pins during programming; measure JTAG signals for noise or reflection.</td>
</tr>
<tr>
<td></td>
<td>Failed to set programming mode</td>
<td></td>
<td>Signal integrity issues on JTAG pins</td>
<td></td>
</tr>
<tr>
<td>0x8032</td>
<td>Device is busy</td>
<td>5</td>
<td>Unstable VDDIx voltage level</td>
<td>Monitor related power supplies that cause the issue during programming; check for transients outside of Microsemi specifications. See your device datasheet for more information on transient specifications.</td>
</tr>
<tr>
<td>0x8003</td>
<td>Failed to enter programming mode</td>
<td>5</td>
<td>Unstable voltage level</td>
<td>Monitor related power supplies that cause the issue during programming; check for transients outside of Microsemi specifications. See your device datasheet for more information on transient specifications.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Signal integrity issues on JTAG pins</td>
<td></td>
</tr>
<tr>
<td>Error Code</td>
<td>Exit Message</td>
<td>Exit Code</td>
<td>Possible Cause</td>
<td>Possible Solution</td>
</tr>
<tr>
<td>------------</td>
<td>--------------</td>
<td>-----------</td>
<td>----------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>0x8004</td>
<td>Failed to verify IDCODE</td>
<td>6</td>
<td>Incorrect programming file</td>
<td>Choose the correct programming file and select the correct device in the chain.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Incorrect device in chain</td>
<td>Measure JTAG pins and noise for reflection. If TRST is left floating then add pull-up to pin.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Signal integrity issues on JTAG pins</td>
<td>If TRST is left floating then add pull-up to pin.</td>
</tr>
<tr>
<td>0x8005</td>
<td>Failed to verify FPGA Array</td>
<td>11</td>
<td>Device is programmed with a different design or the component is blank.</td>
<td>Verify the device is programmed with the correct data/design.</td>
</tr>
<tr>
<td>0x8006</td>
<td>Failed to verify Fabric Configuration</td>
<td></td>
<td>Unstable voltage level.</td>
<td>Monitor related power supplies that cause the issue during programming; check for transients outside of Microsemi specifications. See your device datasheet for more information on transient specifications.</td>
</tr>
<tr>
<td>0x8007</td>
<td>Failed to verify Security</td>
<td></td>
<td>Signal integrity issues on JTAG pins.</td>
<td>Monitor JTAG supply pins during programming; measure JTAG signals for noise or reflection.</td>
</tr>
<tr>
<td>0x8008</td>
<td>Failed to verify sNVM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x8009</td>
<td>Failed to verify eNVM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x8013</td>
<td>External digest check via JTAG/SPI Slave is disabled.</td>
<td>-18</td>
<td>External Digest check via JTAG/SPI Slave is disabled.</td>
<td>Need to use a bitstream file which has a valid FlashLock/UPK1 to enable external digest check via JTAG/SPI Slave.</td>
</tr>
<tr>
<td>0x8015</td>
<td>FPGA Fabric digest verification: FAIL</td>
<td>-20</td>
<td>FPGA Fabric is either erased or the data has been corrupted or tampered with</td>
<td>If the Fabric is erased, deselect procedure &quot;DO_ENABLE_FABRIC&quot; from action &quot;VERIFY_DIGEST&quot;</td>
</tr>
<tr>
<td>0x8016</td>
<td>sNVM digest verification: FAIL</td>
<td>-20</td>
<td>sNVM is either erased or the data has been corrupted or tampered with</td>
<td>If the sNVM is erased, deselect procedure &quot;DO_ENABLE_SNVM&quot; from action &quot;VERIFY_DIGEST&quot;</td>
</tr>
<tr>
<td>0x8018</td>
<td>User security policies segment digest verification: FAIL</td>
<td>-20</td>
<td>Security segment is either erased or the data has been corrupted or tampered with</td>
<td>If the security is erased, deselect procedure &quot;DO_ENABLE_SECURITY&quot; from action &quot;VERIFY_DIGEST&quot;</td>
</tr>
<tr>
<td>Error Code</td>
<td>Exit Message</td>
<td>Exit Code</td>
<td>Possible Cause</td>
<td>Possible Solution</td>
</tr>
<tr>
<td>------------</td>
<td>--------------</td>
<td>-----------</td>
<td>----------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>0x8019</td>
<td>UPK1 segment digest verification: FAIL</td>
<td>-20</td>
<td>UPK1 segment is either erased or the data has been corrupted or tampered with</td>
<td>If the UPK1 is erased, deselect procedure &quot;DO_ENABLE_SECURITY&quot; from action &quot;VERIFY_DIGEST&quot;</td>
</tr>
<tr>
<td></td>
<td>Deselect procedure ‘DO_ENABLE_SECURITY’ to remove this digest check.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x801A</td>
<td>UPK2 segment digest verification: FAIL</td>
<td>-20</td>
<td>UPK2 segment is either erased or the data has been corrupted or tampered with</td>
<td>If the UPK2 is erased, deselect procedure &quot;DO_ENABLE_UKS2&quot; from action &quot;VERIFY_DIGEST&quot;</td>
</tr>
<tr>
<td></td>
<td>Deselect procedure ‘DO_ENABLE_UKS2’ to remove this digest check.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x801B</td>
<td>Factory row and factory key segment digest verification: FAIL</td>
<td>-20</td>
<td>Factory row and factory key segment have been erased through zeroization or the data has been corrupted or tampered with</td>
<td>If the Fabric configuration is erased, deselect procedure &quot;DO_ENABLE_FABRIC&quot; from action &quot;VERIFY_DIGEST&quot;</td>
</tr>
<tr>
<td></td>
<td>Deselect procedure ‘DO_ENABLE_FABRIC’ to remove this digest check.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x801C</td>
<td>Fabric configuration segment digest verification: FAIL</td>
<td>-20</td>
<td>Fabric configuration segment is either erased or has been corrupted or tampered with</td>
<td>If the Fabric configuration is erased, deselect procedure &quot;DO_ENABLE_FABRIC&quot; from action &quot;VERIFY_DIGEST&quot;</td>
</tr>
<tr>
<td></td>
<td>Deselect procedure ‘DO_ENABLE_FABRIC’ to remove this digest check.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x8052</td>
<td>UEK1 segment digest verification: FAIL</td>
<td>-20</td>
<td>UEK1 segment is either erased or the data has been corrupted or tampered with</td>
<td>If the UEK1 is erased, deselect procedure &quot;DO_ENABLE_UEK1&quot; from action &quot;VERIFY_DIGEST&quot;</td>
</tr>
<tr>
<td></td>
<td>Deselect procedure ‘DO_ENABLE_UEK1’ to remove this digest check.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x8053</td>
<td>UEK2 segment digest verification: FAIL</td>
<td>-20</td>
<td>UEK2 segment is either erased or the data has been corrupted or tampered with</td>
<td>If the UEK2 is erased, deselect procedure &quot;DO_ENABLE_UEK2&quot; from action &quot;VERIFY_DIGEST&quot;</td>
</tr>
<tr>
<td></td>
<td>Deselect procedure ‘DO_ENABLE_UEK2’ to remove this digest check.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error Code</td>
<td>Exit Message</td>
<td>Exit Code</td>
<td>Possible Cause</td>
<td>Possible Solution</td>
</tr>
<tr>
<td>------------</td>
<td>--------------------------------------------------</td>
<td>-----------</td>
<td>------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0x8054</td>
<td>DPK segment digest verification: FAIL</td>
<td>-20</td>
<td>DPK segment is either erased or the data has been corrupted or tampered with</td>
<td>If the DPK is erased, deselect procedure &quot;DO_ENABLE_DPK&quot; from action &quot;VERIFY_DIGEST&quot;</td>
</tr>
<tr>
<td></td>
<td>Deselect procedure 'DO_ENABLE_DPK' to remove this digest check.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x8057</td>
<td>SMK segment digest verification: FAIL</td>
<td>-20</td>
<td>SMK segment is either erased or the data has been corrupted or tampered with</td>
<td>If the SMK is erased, deselect procedure &quot;DO_ENABLE_SMK&quot; from action &quot;VERIFY_DIGEST&quot;</td>
</tr>
<tr>
<td>0x8058</td>
<td>User Public Key segment digest verification: FAIL</td>
<td>-20</td>
<td>User Public Key segment is either erased or the data has been corrupted or tampered with</td>
<td>If the User Public Key is erased, deselect procedure &quot;DO_ENABLE_USER_PUBLIC_KEY&quot; from action &quot;VERIFY_DIGEST&quot;</td>
</tr>
<tr>
<td>0x801D</td>
<td>Device security prevented operation</td>
<td>-21</td>
<td>The device is protected with user pass key 1 and the bitstream file does not contain user pass key 1.</td>
<td>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.</td>
</tr>
<tr>
<td>0x801F</td>
<td>Programming Error. Bitstream or data is corrupted or noisy</td>
<td>-22</td>
<td>Bitstream file has been corrupted or was incorrectly generated. Unstable voltage level. Signal integrity issues on JTAG pins.</td>
<td>Regenerate bitstream file Monitor related power supplies that cause the issue during programming; check for transients outside of Microsemi specifications. See your device datasheet for more information on transient specifications. Monitor JTAG supply pins during programming; measure JTAG signals for noise or reflection.</td>
</tr>
<tr>
<td>0x8021</td>
<td>Programming Error. Invalid/Corrupted encryption key</td>
<td>-23</td>
<td>File contains an encrypted key that does not match the device File contains user encryption key, but device has not been programmed with the user encryption key</td>
<td>Provide a programming file with an encryption key that matches that on the device First program security with master programming file, then program with user encryption 1/2 field update programming files</td>
</tr>
<tr>
<td>0x8023</td>
<td>Programming Error. Back level not satisfied</td>
<td>-24</td>
<td>Design version is not higher than the back-level programmed device</td>
<td>Generate a programming file with a design version higher than the back level version</td>
</tr>
<tr>
<td>0x8001</td>
<td>Failure to read DSN</td>
<td>-24</td>
<td>Device is in System Controller Suspend Mode</td>
<td>TRSTB should be driven High or disable &quot;System Controller Suspend Mode&quot;.</td>
</tr>
<tr>
<td>Error Code</td>
<td>Exit Message</td>
<td>Exit Code</td>
<td>Possible Cause</td>
<td>Possible Solution</td>
</tr>
<tr>
<td>------------</td>
<td>--------------</td>
<td>-----------</td>
<td>----------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>0x8027</td>
<td>Programming Error. Insufficient device capabilities</td>
<td>-26</td>
<td>Device does not support the capabilities specified in programming file</td>
<td>Generate a programming file with the correct capabilities for the target device</td>
</tr>
<tr>
<td>0x8029</td>
<td>Programming Error. Incorrect DEVICEID</td>
<td>-27</td>
<td>Incorrect programming file, Incorrect device in chain, Signal integrity issues on JTAG pins</td>
<td>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</td>
</tr>
<tr>
<td>0x802B</td>
<td>Programming Error. Programming file is out of date, please regenerate.</td>
<td>-28</td>
<td>Programming file version is out of date</td>
<td>Generate programming file with latest version of Libero SoC</td>
</tr>
<tr>
<td>0x8030</td>
<td>Programming Error Invalid or inaccessible Device Certificate</td>
<td>-31</td>
<td>FAB_RESET_N is tied to ground</td>
<td>FAB_RESET_N should be tied to HIGH</td>
</tr>
<tr>
<td>0x8032 0x8034 0x8036 0x8038</td>
<td>Instruction timed out</td>
<td>-32</td>
<td>Unstable voltage level, Signal integrity issues on JTAG pins</td>
<td>Monitor related power supplies that cause the issue during programming; check for transients outside of Microsemi specifications. See your device datasheet for more information on transient specifications. Monitor JTAG supply pins during programming; measure JTAG signals for noise or reflection.</td>
</tr>
<tr>
<td>0x8010</td>
<td>Failed to unlock user pass key 1</td>
<td>-35</td>
<td>Pass key in file does not match device</td>
<td>Provide a programming file with a pass key that matches pass key programmed into the device.</td>
</tr>
<tr>
<td>0x8011</td>
<td>Failed to unlock user pass key 2</td>
<td>-35</td>
<td>Pass key in file does not match device</td>
<td>Provide a programming file with a pass key that matches pass key programmed into the device.</td>
</tr>
<tr>
<td>0x804F</td>
<td>Bitstream programming action is disabled</td>
<td>-38</td>
<td>Unstable voltage level, Bitstream programming action has been disabled in Security Policy Manager</td>
<td>Monitor related power supplies that cause the issue during programming; check for transients outside of Microsemi specifications. See your device datasheet for more information on transient specifications. Need to use a bitstream file which has a valid FlashLock/UPK1 to enable the bitstream programming action.</td>
</tr>
</tbody>
</table>
### Program SPI Flash Image

**Generate SPI Flash Image**

This tool generates the `<design>_spi_flash.bin` file in the implementation folder.

To run this tool; under the Program SPI Flash Image, right-click Generate SPI Flash Image and choose Run.

![Selecting the Run Command](image)

This tool depends on the Configure Design Initialization Data and Memories tool and the Generate Design Initialization Data tool. When running, the tool verifies that the SPI Flash configuration data is saved and valid; and that the SPI Flash initialization client was generated successfully (if required).

### Configure SPI Flash Image Actions and Procedures

If SPI Flash is configured, you can select supported SPI Flash Image actions and procedures in the Select Action and Procedures dialog box. See the following example.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Exit Message</th>
<th>Exit Code</th>
<th>Possible Cause</th>
<th>Possible Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x805B</td>
<td>Error, security must be either programmed on a blank device or with the FPGA Fabric design</td>
<td>-42</td>
<td>Security only bitstream programming on a programmed device</td>
<td>Use this bitstream on a blank device or generate a new bitstream that contains the FPGA Fabric design along with the security</td>
</tr>
<tr>
<td>0x805C</td>
<td>eNVM digest verification: FAIL Deselect procedure ‘DO_ENABLE_ENVM’ to remove this digest check</td>
<td>-20</td>
<td>eNVM is either erased or the data has been corrupted or tampered with.</td>
<td>If the eNVM is erased, deselect procedure ‘DO_ENABLE_ENVM’ from action ‘VERIFY_DIGEST’</td>
</tr>
</tbody>
</table>
The following table lists the actions and procedures for the Run PROGRAM_SPI_Flash tool.

<table>
<thead>
<tr>
<th>Action</th>
<th>Mandatory Procedures</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROGRAM_SPI_IMAGE</td>
<td>VERIFY_DEVICE_ID</td>
<td>This action will erase the entire SPI flash then program the SPI image.</td>
</tr>
<tr>
<td></td>
<td>ERASE_DIE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PROGRAM_IMAGE</td>
<td></td>
</tr>
<tr>
<td>VERIFY_SPI_IMAGE</td>
<td>VERIFY_DEVICE_ID</td>
<td>This action verifies the SPI Image on the SPI Flash.</td>
</tr>
<tr>
<td></td>
<td>VERIFY_IMAGE</td>
<td></td>
</tr>
<tr>
<td>READ_SPI_IMAGE</td>
<td>VERIFY_DEVICE_ID</td>
<td>This action reads the SPI Image from the SPI Flash.</td>
</tr>
<tr>
<td></td>
<td>READ_IMAGE</td>
<td></td>
</tr>
<tr>
<td>ERASE_SPI_FLASH</td>
<td>VERIFY_DEVICE_ID</td>
<td>This action erases the entire SPI Flash.</td>
</tr>
<tr>
<td></td>
<td>ERASE_DIE</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** If the device ID does not match when running any action, the action will fail.
Run Programming SPI Flash Actions

This tool allows the user to program the SPI Flash device connected to the PolarFire device through the JTAG programming interface. Currently, only the Micron 1Gb SPI flash is supported, and is included with the Evaluation Kit. This feature minimizes cost by not requiring a mux and external SPI pins on the board for SPI flash programming by another tool. This tool always erases the entire SPI flash prior to programming. Programming starts at address 0 of the SPI flash until the last client. Any gaps in the SPI flash are programmed with all 1’s.

Note: This version of the programmer does not support SPI Flash security. Device security options such as "Hardware Write Protect" should be disabled for the External SPI Flash device.

Figure 112 · SPI Flash Programming with PolarFire Device
The following table provides the expectations of programming the SPI flash with a FP5 programmer. Future programmers are planned, and should greatly improve programming times. All times are in hh:mm:ss.

<table>
<thead>
<tr>
<th>SPI Size</th>
<th>ERASE</th>
<th>PROGRAM</th>
<th>VERIFY/READ</th>
<th>TCK</th>
<th>Programmer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 MB</td>
<td>3:55</td>
<td>00:00:45</td>
<td>00:10:46</td>
<td>4MHz</td>
<td>FP5</td>
</tr>
<tr>
<td>1 MB</td>
<td>3:55</td>
<td>00:00:28</td>
<td>00:10:05</td>
<td>15MHz</td>
<td>FP5</td>
</tr>
<tr>
<td>9 MB</td>
<td>3:55</td>
<td>00:06:38</td>
<td>01:19:15</td>
<td>4MHz</td>
<td>FP5</td>
</tr>
<tr>
<td>9 MB</td>
<td>3:55</td>
<td>00:04:26</td>
<td>01:08:49</td>
<td>10MHz</td>
<td>FP5</td>
</tr>
<tr>
<td>18 MB</td>
<td>3:55</td>
<td>00:09:04</td>
<td>02:32:43</td>
<td>10MHz</td>
<td>FP5</td>
</tr>
<tr>
<td>128 MB</td>
<td>3:55</td>
<td>00:58:38</td>
<td>22:07:55</td>
<td>15MHz</td>
<td>FP5</td>
</tr>
</tbody>
</table>

**Recommendations:**

1. Since the verify time is currently not optimized, it is recommended to authenticate the SPI bitstreams with system services for quicker verification.
2. Since this tool erases the SPI flash prior to programming and currently does not support Data Storage clients for user data, it is recommended to program the SPI flash with Libero prior to programming other data on the SPI flash.
3. Since programming time is currently not optimized, it is recommended to not have huge gaps between clients in the SPI flash, since gaps are currently programmed with 1’s.

If SPI Flash is configured, you can execute Run PROGRAM_SPI_IMAGE Action and select SPI Flash Image actions and procedures.

In the Design Flow window, under Program SPI Flash Image, right-click Run PROGRAM_SPI_Image Action and choose Configure Action/Procedures.

**Note:** In this release, SPI Flash programming is supported for MICRON devices only.

See Configure SPI Flash Image Actions and Procedures for information about supported actions and procedures.
Debug Design

Generate SmartDebug FPGA Array Data

The Generate SmartDebug FPGA Array Data tool generates database files used in downstream tools:
- *.db used for debugging FPGA Fabric in SmartDebug

Double-click Generate SmartDebug FPGA Array Data or right-click Generate SmartDebug FPGA Array Data in the Design Flow window and click Run to generate SmartDebug FPGA Array Data. Before running this tool, the design should have completed the Place and Route step. If not, Libero SoC runs implicitly the upstream tools (Synthesis, Compile Netlist, and Place and Route) before it generates the FPGA SmartDebug Array Data.

SmartDebug

Design debug is a critical phase of FPGA design flow. Microsemi’s SmartDebug tool complements design simulation by allowing verification and troubleshooting at the hardware level. SmartDebug can provide access to Microsemi FPGA device’s built-in probe logic, which enables designers to check the state of inputs and outputs in real-time without re-layout of the design.

SmartDebug can be run in two modes:
- Integrated mode from the Libero Design Flow
- Standalone mode

Integrated Mode

When run in integrated mode from Libero, SmartDebug can access all design and programming hardware information. No extra setup step is required. In addition, the Probe Insertion feature is available in Debug FPGA Array.

To open SmartDebug in the Libero Design Flow window, expand Debug Design and double-click SmartDebug Design.
Standalone Mode

SmartDebug can be installed separately in the setup containing FlashPro Express and Job Manager. This provides a lean installation that includes all the programming and debug tools to be installed in a lab environment for debug. In this mode, SmartDebug is launched outside of the Libero Design Flow. Prior to launch of SmartDebug standalone mode, you must to go through SmartDebug project creation and import a Design Debug Data Container (DDC) file, exported from Libero, to access all debug features in the supported devices.

**Note:** In standalone mode, the Probe Insertion feature is not available in FPGA Array Debug, as it requires incremental routing to connect the user net to the specified I/O.

See Also

[SmartDebug User Guide](#)

Identify Debug Design

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

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

**Identify features:**

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

You must have both the Identify RTL Debugger and the Identify Instrumentor to run the debugging flow outlined below.

**To use the Identify Instrumentor and Debugger:**

1. Create your source file (as usual) and run pre-synthesis simulation.
2. (Optional) Run through an entire flow (Synthesis - Compile - Place and Route - Generate a Programming File) without starting Identify.
3. Right-click **Synthesize** and choose **Open Interactively** in Libero SoC to launch Synplify.
4. In Synplify, click **Options > Configure Identify Launch** to setup Identify.
5. In Synplify, create an Identify implementation; to do so, click **Project > New Identify Implementation**.
6. In the Implementations Options dialog, make sure the Implementation Results > Results Directory points to a location under `<libero project>\synthesis\`, otherwise Libero SoC is unable to detect your resulting Verilog Netlist file.
7. From the Instrumentor UI specify the sample clock, the breakpoints, and other signals to probe. Synplify creates a new synthesis implementation. Synthesize the design.
8. In Libero SoC, run Synthesis, Place and Route and Generate a Programming File.

**Note:** Libero SoC works from the edif netlist of the current active implementation, which is the implementation you created in Synplify for Identify debug.

The Identify RTL Debugger, Synplify, and FlashPro must be synchronized in order to work properly. See the [Release Notes](#) for more information on which versions of the tools work together.
Handoff Design for Production

Export Bitstream

Export Bitstream enables you to export PPD, STAPL, DAT, and SPI programming files.

To export a bitstream file

1. Under Handoff Design for Production, double-click Export Bitstream. The Export Bitstream dialog box opens. The dialog box options depend on your Custom Security settings and Permanent Locks for Production settings:
   - Export Bitstream tool when the device is configured with Bitstream Encryption with Default key option in the Configure Security tool
   - Export Bitstream tool when the device is configured with Custom Security options in the Configure Security tool
   - Export Bitstream tool when the device is configured with Permanent Locks for Production tool
2. Enter your Bitstream file name and select the location to export the bitstream file.
3. Choose your options, such as DAT file if you wish to include support for Embedded ISP, SPI file if you need support for Auto programming, Auto Update and IAP services, PPD or STAPL if you need support for ISP.
4. You can choose whether you want to export the ASCII HEX file for debugging when DAT bitstream file format is selected.
5. Select whether you want to export files for Microsemi In House Programming (IHP).
6. Select to include the bitstream components, Bypass Back Level protection option for Recovery/Golden Bitstream and Include Pass Keys in Plaintext.
7. Choose the Zeroization Action that you want to use to reprogram the device. Note: Zeroization Action is not supported for XT and ES devices.
8. Click OK to export the selected bitstream files.

See Also
Digest File

Export Bitstream tool when the device is configured with Bitstream Encryption with Default Key in the Configure Security tool
Design Settings:

- **Programming Options set with Configure Programming Options tool**: Displays the Design version and Back Level version set in the Configure Programming Options tool. They are read-only options and cannot be modified.

  **Note**: Info and warning messages appear based on the value set for Back Level version.

- **Security options set with the Configure Security tool**: Provides a brief description of current security options.

 **Bitstream file(s):**

- **Name**: Sets the name of your bitstream file. The default name is the design name.
- **Location**: Location to save the exported file.
- **Existing files**: Lists bitstream files previously created.
- **Formats**: Select the Bitstream File format you want to export:
  - PPD
  - DAT
  - STAPL
  - SPI
  - Export separate ASCII HEX file for debugging – Exports DAT file in HEX format. This option is active only when DAT file format is selected.
  - Export files for Microsemi In House Programming (IHP) – Exports DAT and STP file formats if ‘Yes’ is selected.

  PPD and DAT file formats are the default file formats.

- **File types**: Lists all the bitstream files to be exported
  - **File to program at trusted facility**: Click to include Fabric /sNVM into the bitstream files to be programmed at a trusted facility.
- **Zeroization Actions** – Select the **Like New** checkbox to erase all user data to make the device like new and reprogram the device immediately, or select the **Unrecoverable** checkbox to erase all data and destroy reprogrammability (in this case, the device must be scrapped).

In case of eNVM file type, the Export Bitstream has an eNVM option to select for bitstream files to be programmed at a trusted facility.

![Figure 115 · Export Bitstream with eNVM Option](image-url)
Export Bitstream tool when the device is configured with Custom Security option in the Configure Security tool

**Note:** If Permanent Locks for Production are configured along with Custom Security settings, Export Bitstream Security options are set with Configure Permanent Lock for Production tool. Otherwise, they are set with the Configure Security tool. The Permanent Lock Security settings can be removed by either right-clicking on the tool in the Design Tree or by clicking the Remove Permanent Locks button in the UI.

Figure 116 · Export Bitstream Dialog Box with Custom Security Options Set with the Configure Security Tool

**Design Settings:**
- **Programming Options set with Configure Programming Options tool:** Displays the Design version and Back Level version set in the Configure Programming Options tool. They are read-only options and cannot be modified.

  **Note:** Info and warning messages appear based on the value set for Back Level version.

- **Security options set with the Configure Security tool:** Provides a brief description of current security settings.

**Bitstream file(s):**
- **Name** - Sets the name of your bitstream file. The default name is the design name.
• **Location** - Location to save the exported file.
• **Existing files** - Lists bitstream files previously created.
• **Formats** - Select the Bitstream File format you want to export:
  - PPD
  - DAT
  - STAPL
  - SPI
  - Export separate ASCII HEX file for debugging – Exports DAT file in HEX format for debugging. This option is active only when DAT file format is selected.
  - Export files for Microsemi In House Programming (IHP) – Exports DAT and STP file formats if ‘Yes’ is selected.

PPD and DAT file formats are the default file formats.

• **File types:**

  **Note:** Refer to Include Plaintext Pass Keys to check the availability of Export Pass keys for Plaintext option for the following Bitstream files.

  **Note:** eNVM option is only visible when eNVM is used in the design. eNVM features are available for PolarFireSoC devices only.

Lists all the bitstream files to be exported

• **Master file** – Click to include:
  - Fabric/sNVM.
  - eNVM
  - Bypass Back Level protection for Recovery/Golden bitstream (this option is available only if Back Level protection is enabled in the Configure Security tool and SPI file format is selected).
  - Include Plaintext Pass Keys.

**Notes**

1. Custom Security is always programmed in the Master file.
2. Security-only programming must be performed only on erased or new devices. If performed on a device with fabric programmed, the fabric will be disabled after performing security-only programming. You must reprogram the fabric to re-enable it.
3. Depending on the Permanent Locks configured in the Configure Permanent Locks for Production tool, some of the bitstream components for Master file are selected by default and are unavailable for users to change.

• **File encrypted with UEK1 to program at untrusted facility or for Broadcast field update** - Click to include:
  - Fabric/sNVM.
  - eNVM
  - Bypass Back Level protection for Recovery/Golden bitstream (this option is available only if Back Level protection is enabled in the Configure Security tool and SPI file format is selected).
  - Include Plaintext Pass Keys.

• **File encrypted with UEK2 to program at untrusted facility or for Broadcast field update** - Click to include:
  - Fabric/sNVM.
  - eNVM
  - Bypass Back Level protection for Recovery/Golden bitstream (this option is available only if Back Level protection is enabled in the Configure Security tool and SPI file format is selected).
  - Include Plaintext Pass Keys.
**Note:** If sNVM/Fabric is One Time Programmable, it is precluded from bitstream encrypted with UEK1/2.

- **Zeroization Actions** – Select the Like New checkbox to erase all user data to make the device like new and reprogram the device immediately, or select the Unrecoverable checkbox to erase all data and destroy reprogrammability (in this case, the device must be scrapped).

**Bypass Back Level protection for Recovery/Golden bitstream** - If Back Level protection is enabled in the Configure Security tool, you can bypass the back level protection for SPI bitstreams to prevent Programming Recovery failures.

The following examples show the usage of Bypass Back Level Protection.

**Example 1 – Fails without Bypass Back Level Protection**

<table>
<thead>
<tr>
<th>Step</th>
<th>Bitstream</th>
<th>Action</th>
<th>Result</th>
<th>Design version</th>
<th>Back Level version</th>
<th>Device Back Level version</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Golden/Recovery</td>
<td>Auto Programming</td>
<td>Pass</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>IAP/Update Bitstream</td>
<td>Auto Update/IAP</td>
<td>Pass</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>IAP/Update Bitstream</td>
<td>Auto Update/IAP</td>
<td>Fail, Attempt Programming Recovery</td>
<td>4</td>
<td>N/A</td>
<td>2</td>
</tr>
</tbody>
</table>

In the above example, the device will have Back Level version 2 after step 2. If you attempt to program with bitstream in step 3, it will fail. If the device attempts to reprogram, it will initiate Programming Recovery with the Golden/Recovery bitstream. Since the Golden/Recovery Bitstream has Design version 2, which is less than or equal to the Back Level version in the device, it will fail. If the Bypass Back Level version option is selected, this back level protection check will be bypassed for the Golden/Recovery Bitstream only and it will succeed.

**Example 2 – Does not require Bypass Back Level Protection**

<table>
<thead>
<tr>
<th>Step</th>
<th>Bitstream</th>
<th>Action</th>
<th>Result</th>
<th>Design version</th>
<th>Back Level version</th>
<th>Device Back Level version</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Golden/Recovery</td>
<td>Auto Programming</td>
<td>Pass</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>IAP/Update Bitstream</td>
<td>Auto Update/IAP</td>
<td>Pass</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>IAP/Update Bitstream</td>
<td>Auto Update/IAP</td>
<td>Fail, Attempt Programming Recovery</td>
<td>4</td>
<td>N/A</td>
<td>1</td>
</tr>
</tbody>
</table>

In this example, the device will have Back Level version 1 after step 2. If you attempt to program with bitstream in step 3, it will succeed since the Golden/Recovery Design version is greater than the Back Level version on the device.

**Example 3 – Requires Bypass Back Level Protection**

<table>
<thead>
<tr>
<th>Step</th>
<th>Bitstream</th>
<th>Action</th>
<th>Result</th>
<th>Design version</th>
<th>Back Level version</th>
<th>Device Back Level version</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Golden/Recovery</td>
<td>Auto Programming</td>
<td>Pass</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>IAP/Update Bitstream</td>
<td>Auto Update/IAP</td>
<td>Pass</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>IAP/Update Bitstream</td>
<td>Auto Update/IAP</td>
<td>Fail, Attempt Programming Recovery</td>
<td>3</td>
<td>N/A</td>
<td>1</td>
</tr>
</tbody>
</table>
In this example, the device will have Back Level version 1 after step 2. If you attempt to program with bitstream in step 3, it will fail. If the device attempts to reprogram, it will initiate Programming Recovery with the Golden/Recovery bitstream. Since the Golden/Recovery Bitstream has Design version 1, which is less than or equal to the Back Level version in the device, it will fail. If the Bypass Back Level version option is selected, this back level protection check will be bypassed for the Golden/Recovery Bitstream only and it will succeed.

Export Bitstream tool when the device is configured with Permanent Locks for Production tool

**Note:** If Permanent Locks for Production are configured along with Custom Security settings, Export Bitstream Security options are set with the Configure Permanent Lock for Production tool. Otherwise, they are set with the Configure Security tool. The Permanent Lock Security settings can be removed by either right-clicking on the tool in the Design Tree or by clicking the **Remove Permanent Locks** button in the UI.

**Design Settings:**
- **Programming Options set with the Configure Programming Options tool**: Displays the Design version and Back Level version set in the Configure Programming Options tool. They are read-only options and cannot be modified.
  
  **Note:** Info and warning messages appear based on the value set for Back Level version.

- **Security options set with the Configure Permanent Locks for Production tool**: Provides a brief description of current security settings.

**Bitstream file(s):**
- **Name**: Sets the name of your bitstream file. The default name is the design name.
• **Location** - Location to save the exported file.
• **Existing files** - Lists bitstream files previously created.
• **Formats** - Select the Bitstream File format you want to export:
  • PPD
  • DAT
  • STAPL
  • SPI
  • Export separate ASCII HEX file for debugging – Exports DAT file in HEX format for debugging. This option is active only when DAT file format is selected.
  • Export files for Microsemi In House Programming (IHP) – Exports DAT and STP file formats if ‘Yes’ is selected.

PPD and DAT file formats are the default file formats.

• **File types:**

  **Note:** Refer to Include Plaintext Pass Keys to check the availability of Include Plaintext Pass keys option for the following Bitstream files.

  Lists all the bitstream files to be exported
  
  • **Master file** – Click to include:
    • Fabric/sNVM.
    • Bypass Back Level protection for Recovery/Golden bitstream (this option is available only if Back Level protection is enabled in the Configure Security tool and SPI file format is selected).
    • Include Plaintext Pass Keys.

  **Notes**
  1. Custom Security is always programmed in the Master file.
  2. Security-only programming must be performed only on erased or new devices. If performed on a device with fabric programmed, the fabric will be disabled after performing security-only programming. You must reprogram the fabric to re-enable it.
  3. Depending on the Permanent Locks configured in the Configure Permanent Locks for Production tool, some of the bitstream components for Master file are selected by default and are unavailable for users to change.

  • **File encrypted with UEK1 to program at untrusted facility or for Broadcast field update** - Click to include:
    • Fabric/sNVM.
    • Bypass Back Level protection for Recovery/Golden bitstream (this option is available only if Back Level protection is enabled in the Configure Security tool and SPI file format is selected).
    • Include Plaintext Pass Keys.

  • **File encrypted with UEK2 to program at untrusted facility or for Broadcast field update** - Click to include:
    • Fabric/sNVM.
    • Bypass Back Level protection for Recovery/Golden bitstream (this option is available only if Back Level protection is enabled in the Configure Security tool and SPI file format is selected).
    • Include Plaintext Pass Keys.

  **Note:** If sNVM/Fabric is One Time Programmable, it is precluded from bitstream encrypted with UEK1/2.

• **Zeroization Actions** – Select the Like New checkbox to erase all user data to make the device like new and reprogram the device immediately, or select the Unrecoverable checkbox to erase all data and destroy reprogrammability (in this case, the device must be scrapped).

  **Bypass Back Level protection for Recovery/Golden bitstream** - If Back Level protection is enabled in the Configure Security tool, you can bypass the back level protection for SPI bitstreams to prevent Programming Recovery failures.
The following examples show the usage of Bypass Back Level Protection.

**Example 1 – Fails without Bypass Back Level Protection**

<table>
<thead>
<tr>
<th>Step</th>
<th>Bitstream</th>
<th>Action</th>
<th>Result</th>
<th>Design version</th>
<th>Back Level version</th>
<th>Device Back Level version</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Golden/Recovery</td>
<td>Auto Programming</td>
<td>Pass</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>IAP/Update Bitstream</td>
<td>Auto Update/IAP</td>
<td>Pass</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>IAP/Update Bitstream</td>
<td>Auto Update/IAP</td>
<td>Fail, Attempt Programming</td>
<td>4</td>
<td>N/A</td>
<td>2</td>
</tr>
</tbody>
</table>

In the above example, the device will have Back Level version 2 after step 2. If you attempt to program with bitstream in step 3, it will fail. If the device attempts to reprogram, it will initiate Programming Recovery with the Golden/Recovery bitstream. Since the Golden/Recovery Bitstream has Design version 2, which is less than or equal to the Back Level version in the device, it will fail. If the Bypass Back Level version option is selected, this back level protection check will be bypassed for the Golden/Recovery Bitstream only and it will succeed.

**Example 2 – Does not require Bypass Back Level Protection**

<table>
<thead>
<tr>
<th>Step</th>
<th>Bitstream</th>
<th>Action</th>
<th>Result</th>
<th>Design version</th>
<th>Back Level version</th>
<th>Device Back Level version</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Golden/Recovery</td>
<td>Auto Programming</td>
<td>Pass</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>IAP/Update Bitstream</td>
<td>Auto Update/IAP</td>
<td>Pass</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>IAP/Update Bitstream</td>
<td>Auto Update/IAP</td>
<td>Fail, Attempt Programming</td>
<td>4</td>
<td>N/A</td>
<td>1</td>
</tr>
</tbody>
</table>

In this example, the device will have Back Level version 1 after step 2. If you attempt to program with bitstream in step 3, it will succeed, since the Golden/Recovery Design version is greater than the Back Level version on the device.

**Example 3 – Requires Bypass Back Level Protection**

<table>
<thead>
<tr>
<th>Step</th>
<th>Bitstream</th>
<th>Action</th>
<th>Result</th>
<th>Design version</th>
<th>Back Level version</th>
<th>Device Back Level version</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Golden/Recovery</td>
<td>Auto Programming</td>
<td>Pass</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>IAP/Update Bitstream</td>
<td>Auto Update/IAP</td>
<td>Pass</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>IAP/Update Bitstream</td>
<td>Auto Update/IAP</td>
<td>Fail, Attempt Programming</td>
<td>3</td>
<td>N/A</td>
<td>1</td>
</tr>
</tbody>
</table>

In this example, the device will have Back Level version 1 after step 2. If you attempt to program with bitstream in step 3, it will fail. If the device attempts to reprogram, it will initiate Programming Recovery with the Golden/Recovery bitstream. Since the Golden/Recovery Bitstream has Design version 1, which is less than or
equal to the Back Level version in the device, it will fail. If the Bypass Back Level version option is selected, this
back level protection check will be bypassed for the Golden/Recovery Bitstream only and it will succeed.

Security Programming Files

Export Bitstream (expand Handoff Design for Production in the Design Flow window) creates the following files:

<filename>_master.(stp/spi/dat) - Created when Enable custom security options is specified in the Security
Wizard. This is the master programming file; it includes all programming features enabled, User Key Set 1, User
Key Set 2 (optionally if specified), and your security policy settings.

<filename>_security_only_master.(stp/spi/dat) – Created when Enable custom security options is specified in
the Security Wizard. Master security programming file; includes User Key Set 1, User Key Set 2 (optionally if
specified), and your security policy settings.

<filename>_uek1.(stp/spi/dat) – Programming file encrypted with User Encryption Key 1 used for field updates;
includes all your features for programming except security.

<filename>_uek2.(stp/spi/dat) – Programming file encrypted with User Encryption Key 2 used for field updates;
includes all your features for programming except security.

Export FlashPro Express Job

To program the design using standalone FlashPro Express tool on Linux or Windows, the user must export a
FlashPro Express Job. The job file will include chain configuration, Programmer Settings and programming files
loaded from Programming Connectivity and Interface.

**Note:** SPI Slave mode is supported by FlashPro6 for PolarFire devices. JTAG is the default interface.

**Note:** The Export FlashPro Express Job uses Permanent Locks for Production configuration, if
Permanent Locks are configured. If Permanent Locks are not configured, it uses the Configure Security
configuration.

Use the Configure Security tool before you export the programming job to add security. The Export FlashPro
Express Job dialog box displays the Security Options you have configured through the Configure Security tool or
Configure Permanent Locks for Production tool.

The Export FlashPro Express Job dialog box options vary depending on the device you are using and the
Security Settings.

**Note:** The eNVM option is visible when eNVM is used in the design. eNVM features are available for PolarFire SoC
devices only
Design Settings:

- **Programming Options set with Configure Programming Options tool** - Displays the Design version and Back Level version set in the Configure Programming Options tool. They are read-only options and cannot be modified.

  **Note:** Info and warning messages appear based on the value set for Back Level version.

- **Security options set with the Configure Security tool** - Provides a brief description of current security settings.

**FlashPro Express Job file:**

- **Name** - Sets the name of your bitstream file. The default name is the design name.
- **Location** - Location of the file to be exported.
- **Existing files** - Lists any existing programming job files at the selected location.
- **Program (JTAG Programming Interface) - Existing files** - Click Design or SPI Flash or both to include the configured device chain with bitstream files and/or SPI flash binary file. JTAG is the default programming interface for PolarFire Devices. SPI Flash option is only available if SPI Flash is configured in Configure Design Initialization Data and Memories tool. Otherwise it remains grayed out. Info message appears based on the configuration selected.

- **Design bitstream file format** - Lists all the available bitstream files, one of which will be included in the programming job for the current target device. The format of the Bitstream file can be selected from the Format drop down menu. PPD is the default bitstream file format.
- **Design bitstream file type**
- **File to Program at trusted facility** (only available when the design is configured with Bitstream Encryption with Default Key in the **Configure Security tool** – Click to enable programming for Fabric/sNVM and eNVM bitstream components at a trusted facility.

- **Master file to program at trusted facility** (available when the design is configured with Custom Security options in the **Configure Security tool** or when the design is configured with Permanent Locks in the **Configure Permanent Locks for Production tool**) – Click to include:
  - Fabric/sNVM.
  - eNVM
  - Include Plaintext Pass Keys into the bitstream files to be programmed at a trusted facility.

![Figure 119: Export FlashPro Express Job Dialog Box with Custom Security Options Set with the Configure Security Tool](image)

**Notes:**
- Custom Security is always programmed in the Master file.
- Security-only programming must be performed only on erased or new devices. If performed on a device with fabric programmed, the fabric will be disabled after performing security-only programming. You must reprogram the fabric to re-enable it.
- **File encrypted with UEK1 to program at untrusted facility or for Broadcast field update** – Click to include:
  - Fabric/sNVM.
  - eNVM
  - Include Plaintext Pass Keys into the bitstream files to be programmed.
• File encrypted with UEK2 to program at untrusted facility or for Broadcast field update - 
  Click to include:
  • Fabric/sNVM.
  • eNVM
  • Include Plaintext Pass Keys into the bitstream files to be programmed.

  **Note:** Refer to Export Pass Keys for Plaintext to check the availability of Export Pass keys for Plaintext
  option for the following Bitstream files.

• **Zeroization Actions** – Click the **Like New** option to erase all user data to make the device like new and
  reprogram the device immediately, or click the **Unrecoverable** option to erase all data and destroy
  reprogrammability (in this case, the device must be scrapped).

---

**Prepare Design for Production Programming in FlashPro Express**

After you have exported a programming job you can handoff this programming job to the FlashPro Express tool
for production programming. To do so:

In FlashPro Express, from the **File** menu choose **Create Job Project From a Programming Job**. You will be
prompted to specify the Programming Job location that you just exported from Libero and the location of where to
store the Job Project. The Job Project name automatically uses the programming job name and cannot be
changed. Click **OK** and a new Job Project will be created and opened for production programming.

**See Also**

**Configure Actions and Procedures**

**Export Job Manager Data**

Job Manager is Microsemi’s HSM-based security software for job management.

As a part of the SPPS flow, the Export Job Manager Data dialog box allows a design engineer (user) to export
Libero design data to Job Manager. Exported data is used by an operation engineer (OE) using Job Manager to
prepare the manufacturing process for HSM or non-HSM flow.
**Programming data file** – export Job Data Container (JDC) file.

**Name** - All names use a prefix as shown in your software.

**Location** - Location of the file to be exported.

**Existing programming job files** - Lists any existing programming job files already in your project.

**Include bitstream components** - Lists the components of the design that can be saved to the file.

**Export SPI Flash Image**

This tool depends on the “Configure Design Initialization Data and Memories” tool. The SPI Flash configuration can be exported to a binary file here. Use this dialog to export a SPI Flash Image file.
Name

This is the top level design name by default. Use this field to change the default name. SPI Flash Image files are exported in binary format and have the *.bin file extension and are named <design_name>.bin.

Location

The default location for the exported image file is <project_folder>\designer\<top_level_design>\export. Use the browse button to navigate to and specify a different location for the exported SPI Flash Image file.

Existing files

Existing SPI Flash Image files are listed.

See Also

export_spi_flash_image

PolarFire FPGA Programming User Guide

Export Pin Report

In the Design Flow window, expand Handoff Design for Production. Right-click Export Pin Report to export a pin report.

The Export Pin Reports dialog box opens. Click Browse to navigate to a disk location where you want the pin report to be saved to.

Check the checkbox to make your selections:

- Pin Report sorted by Port Name
- Pin Report sorted by Package Pin Name
- I/O Bank Report
- I/O Register Combining Report

The pin report lists the pins in your device sorted according to your preference: sort by Port Name or Sorted by Package Pin Name. The pin report generates two files:

- <design>_pinrpt_name.rpt - Pin report sorted by name.
- <design>_pinrpt_number.rpt - Pin report sorted by pin number.

You must select at least one report.
Export Pin Report generates a Bank Report by default; the filename is `<design>-bankrpt.rpt`. Export Pin Report also generates an I/O Register Combining Report listing the I/Os which have been combined into a Register for better timing performance.

![Export Pin Report Dialog Box](image)

**Figure 122 · Export Pin Report Dialog Box**

## Export BSDL File

Double-click Export BSDL File (in the Libero SoC Design Flow window, Handoff Design for Production > Export BSDL File) to generate the BSDL File report to your Design Report.

The BSDL file provides a standard file format for electronics testing using JTAG. It describes the boundary scan device package, pin description and boundary scan cell of the input and output pins. BSDL models are available as downloads for many Microsemi SoC devices.

See the [Microsemi website for more information on BSDL Models](http://www.microsemi.com).

## Export IBIS Model

**Note:** Export Input Output Buffer Information Specification (IBIS) is supported for all the “T” devices of PolarFire Family except the “XT” device.

Export IBIS feature is supported in the following two stages:

- Pre-Layout Flow
- Post-Layout Flow

To export the IBIS Model in the Pre-Layout Flow, the design has to pass until the Synthesis/Compile stage. You need to provide the I/O placement and configuration details in the I/O PDC files to export the resultant IBIS files in Pre-Layout Flow.

To export the IBIS Model in the Post-Layout Flow, the design has to pass until the Place and Route stage. If a design has run through the Place and Route tool, the IBIS model for the Post-Layout Flow will be exported by default.

Double-click **Export IBIS Model** (in the Libero SoC Design Flow window, Handoff Design for Production > Export IBIS Model) to open the Export IBIS Model dialog box.
Output File:

Click **Browse** to specify the location to export the IBIS output file along with the output file name.

Export Options:

The IBIS report *.ibs file exported from Libero SoC can optionally support the [Model Selector] keyword as specified in the IBIS 5.0 Specifications. To generate Model Selector support in the exported IBIS file, check the **Enable Model Selector** box in the Export Options. Using this, the user can also optionally limit the maximum signal name to 40 characters.

Click **OK** to export the IBIS Model.

The IBIS model report provides an industry-standard file format for recording parameters like driver output impedance, rise/fall time, and input loading, which may then be used by software applications such as Signal Integrity tools or IBIS simulators.

The exported IBIS file has the file extension *.ibs (named <root>.ibs) and is displayed in the Files tab.

In the [Pin] section of the IBIS *.ibs file, listed under the model_name are the Model Selector tag. The IBIS *.ibs file has a [Model Selector] section that describes the model selector and its list of models. The Model Selector tag in the [Pin] section establishes the relationship between the pin and the [Model Selector].
The advantage of Model Selector feature is that you can load the *.ibs file from Libero SoC into Signal Integrity applications or IBIS simulators and switch the I/O to different models for individual I/Os on-the-fly in the tools. There is no need to go back to the Libero SoC I/O Attribute Editor to change the I/O settings and run Compile to switch to different I/O settings.

See the Microsemi Website for more information on IBIS Models.
Export SmartDebug Data (Libero SoC)

Export SmartDebug Data allows the export of SmartDebug Data from Libero to be handed off to the Standalone SmartDebug environment.

In the Libero SoC Design Flow window, expand Handoff Design for Debugging, right-click Export SmartDebug Data and click Export to open the Export SmartDebug Data dialog box. Specify the design debug data file (*.ddc)) to be exported. This file is also used as one of the ways to create a Standalone SmartDebug project. See the following figure for an example.

![Export SmartDebug Data Dialog Box](image)

**Note:** SmartDebug data can be exported without connecting the hardware.

**Design debug data file (*.ddc)**

**Name**
The name of the design.

**Location**
The location of the exported debug file. By default, the *.ddc file is exported to the `<project_location>/designer/<design>/export` folder and has the *.ddc file extension.

**Existing Design Debug Data Files**
The existing *.ddc file, if any, in the export folder.
SmartDebug data can be exported after you run Generate FPGA Array Data for the design in the Libero Design Flow. You can also directly export SmartDebug data after running Synthesize on the design. Other tools, such as Place and Route, Generate FPGA Array Data, and so forth) are implicitly run before the Export SmartDebug Data dialog box is displayed.

**Include design components**

A DDC file can contain the following components:

- **FPGA Array Probe Points** – When checked, Libero SoC exports Live and Active probes information (<design>_probe.db file) into the *.ddb container file.

- **FPGA Array Memory Blocks** – When checked, Libero SoC exports information about FPGA memories (<design>_sii_block.db) into the *.ddb container file:
  - names and addresses of the memory blocks instantiated by the design
  - data formats selected by the user in the design

- **sNVM** – When checked, Libero SoC exports sNVM components.

- **Security** – This contains the security locks, keys, and security policy information needed for debug. This may be default or custom security (<design>.spm file). It is hidden if security is not supported for the device.
  - **Display DPK or UPK1 in SmartDebug** - This option is enabled only when custom security is provided, and is unchecked by default.

- **JTAG Chain** (device chain information configured using Programming Connectivity and Interface in Libero) – When checked, Libero SoC exports chain data including devices, their programming files if loaded, device properties, and so on (<design>.pro file). If JTAG chain is unchecked, the default JTAG chain with Libero design device only is added to the *.ddc file.

- **Programmer Settings** (<design>.pro file) – If Programmer Settings is unchecked, the default programmer settings are added to the *.ddc file.

- **Device I/O States During Programming** (<design>.ios file) – This setting is used by some SmartDebug features, for example, for programming sNVM. It is NOT used during device programming in SmartDebug; programming files used to program devices already have I/O states data.

In addition, you can include bitstream file information, which can be used for programming the device in Standalone SmartDebug.

**Programming Options set with Configure Programming Options tool**- Displays the Design version and Back Level version set in the Configure Programming Options Wizard. They are read only options and cannot be modified.

**Note:** Info and warning messages appear based on the value set for the Back Level version.

**Include Bitstream file to program at trusted facility**

- Bitstream components: Fabric
- Bitstream components: sNVM
- Bitstream components: Custom security (this option is only visible if Custom Security Option is selected in Configure Security wizard).

**Note:** If the Fabric Bitstream component is programmed, the sNVM Bitstream component must be programmed and it is selected by default. In this case, sNVM is disabled for the user to unselect. Otherwise, the sNVM component can either be enabled or disabled for programming.

The default location of the DDC file is: <Libero_Project_directory>/designer/<design_name>/export.

The DDC file can be exported to any user-specified location if the location has read and write permission.
Opening a Project
To open a project:

1. From the File menu, choose Open Project.
2. Select the project file ending in .prjx or .prj to open.
3. Click Open.

Note:
While opening a project that was created using relative path for linked files, when user deletes the environment variable before opening the project, an error message is generated if the environment variable does not exist or the path set in environment variable is empty and aborts opening the project.

So, when Environment variable is not defined/set following error message appears.

Error message: 
The Environment variable <env_variable_name> specified for the root directory path for linked files in the project you are trying to open is not defined/set in your environment. Please make sure to set the appropriate root directory path in this Environment variable before you open Libero. You will need to exit Libero and open a new session for any changes you made to the directory path set in the Environment variable to reflect here and allow you to open the project.

set_client

This Tcl command specifies the client that will be added to SPI Flash Memory. This command is added to the SPI Flash Memory configuration file that is given as the parameter to the configure_spiflash command.

```
set_client \
  -client_name {} \
  -client_type {FILE_SPI | FILE_SPI_GOLDEN | FILE_SPI_UPDATE |FILE_DATA_STORAGE_INTELHEX \
  -content_type {MEMORY_FILE | STATIC_FILL} \n  -content_file {} \n  -start_address {} \n  -client_size {} \n  -program {0|1}
```

Arguments
- `-client_name`
The name of the client. Maximum of 32 characters, letters or numbers or "." or "_".
- `-client_type`
The `-client_type` can be `FILE_SPI`, `FILE_SPI_GOLDEN`, `FILE_SPI_UPDATE` or `FILE_DATA_STORAGE_INTELHEX`.
  - `FILE_SPI` - SPI Bitstream
  - `FILE_SPI_GOLDEN` - Recovery/Golden SPI Bitstream
  - `FILE_SPI_UPDATE` - Auto Update SPI Bitstream; available only if Auto Update is enabled. See `set_auto_update_mode`.
  - `FILE_DATA_STORAGE_INTELHEX` - Data Storage client
- `-content_type`
The `-content_type` can be `MEMORY_FILE` or `STATIC_FILL`.
  - `MEMORY_FILE` - content_file parameter must be specified. See below.
configure_uprom

Tcl command; configures the uPROM clients in the uPROM tab of the Design and Memory Initialization tool. Can specify user uPROM clients using this command.

**Note:** You must run Generate Design Initialization Data (generate_design_initialization_data) after configuring uPROM (configure_uprom) and/or Design Initialization (configure_design_initialization_data).

```tcl
configure_uprom 
-cfg_file <path_to_configuration_file.cfg>
```
configure_spiflash

Tcl command; configures the SPI Flash clients in the SPI Flash tab of the Design and Memory Initialization tool. Can specify user SPI FLASH clients using this command.

**Note:** You must run Generate Design Initialization Data (generate_design_initialization_data) after configuring SPI Flash(configure_spiflash) and/or Design Initialization (configure_design_initialization_data).

```bash
configure_spiflash -cfg_file <path_to_configuration_file.cfg>
```

Arguments

- **-cfg_file path_to_configuration_file.cfg**

  Specifies the path to the configuration file of the SPI FLASH client. It is mandatory.

Examples

```bash
configure_spiflash -cfg_file {../../Downloads/mpf_dg0852_liberosocv12p0_df/Libero_Project/TVS_Demo/designer/TVS_Demo/spiflash.cfg}
```

See Also

generate_design_initialization_data

**Archive Project Dialog Box**

The Archive Project dialog box enables you to create an archive (*.zip file) of your existing project and save it at the specified location. This is useful if you want to create a quick zip file of your project. Internally, **Archive Project** and **Save As Project** do the same task, but if you are saving your Libero application to the SVN repository, it is important to archive your project. When you do an SVN checkout after checkin, there is a chance that the timestamps will be outdated and the tool states will remain invalidated. Archive Project helps in retaining the tool states by preserving the database in compressed format.
To access this dialog box, choose **Archive Project** from the **Project menu**.

**Figure 126 · Archive Project Dialog Box**

**Project Location**
Accept the default location or Browse to a new location where the archive (*.zip) file is to be created and stored. The *.zip file is named <project_name>.zip.

**Content**

**Copy links locally** - Select this checkbox to copy the links from your current project into your archive. If you do not select this checkbox, the links will not be copied and you must add them manually.

**Files**: Specifies what kind of files are archived.

- **All** - Includes in the archive all your project and source files; the state of the project is retained.
- **Project files only** – Includes in the archive only the project-related information required to retain the state of the project.
- **Source files only** - Includes in the archive only the source files. This means the configuration of all the tools in the tool chain is retained but the states are not. Source files means constraint information and component information available in the component, hdl and smartgen directories.

Files are archived as shown in the table below for the different selections.

<table>
<thead>
<tr>
<th>Folder Name</th>
<th>Files</th>
<th>Project Files Only</th>
<th>Source Files Only</th>
</tr>
</thead>
<tbody>
<tr>
<td>component</td>
<td>All Files</td>
<td>All Files</td>
<td>All Files</td>
</tr>
<tr>
<td>constraint</td>
<td>All Files</td>
<td>All Files</td>
<td>All Files</td>
</tr>
<tr>
<td>hdl</td>
<td>All Files</td>
<td>All Files</td>
<td>All Files</td>
</tr>
<tr>
<td>stimulus</td>
<td>All Files</td>
<td>All Files</td>
<td>All Files</td>
</tr>
<tr>
<td>viewdraw</td>
<td>All Files</td>
<td>All Files</td>
<td>All Files</td>
</tr>
<tr>
<td>smartgen</td>
<td>All Files</td>
<td>All Files</td>
<td>All Files</td>
</tr>
</tbody>
</table>
Adding or Modifying Bus Interfaces in SmartDesign

SmartDesign supports automatic creation of data driven configurators based on HDL generics/parameters. You can add a bus interface from your HDL module or you can add it from the Catalog.

To add a bus interface using your custom HDL block:

If your block has all the necessary signals to interface with the AMBA bus protocol (such as address, data, and control signals):

1. Right-click your custom HDL block and choose Create Core from HDL. The Libero SoC creates your core and asks if you want to add bus interfaces.
2. Click Yes to open the Edit Core Definition dialog box and add bus interfaces. Add the bus interfaces as necessary.
3. Click OK to continue.

Now your instance has a proper AMBA bus interface on it. You can manually connect it to the bus or let Auto Connect find a compatible connection.

To add (or modify) a bus interface to your Component:

Note: *edn files are not supported in PolarFire.
1. Right-click your Component and choose **Edit Core Definition**. The Edit Core Definition dialog box opens, as shown in the figure below.

![Edit Core Definition Dialog Box](image)

**Figure 127 · Edit Core Definition Dialog Box**

2. Click **Add Bus Interface**. Select the bus interface you wish to add and click **OK**.

3. If necessary, edit the bus interface details.

4. Click **Map by Name** to map the signals automatically. Map By Name attempts to map any similar signal names between the bus definition and pin names on the instance. During mapping, bus definition signal names are prefixed with text entered in the **Map by Name Prefix** field.

5. Click **OK** to continue.
Bus Interface Details

**Bus Interface:** Name of bus interface. Edit as necessary.

**Bus Definition:** Specifies the name of the bus interface.

**Role:** Identifies the bus role (master or slave).

**Vendor:** Identifies the vendor for the bus interface.

**Version:** Identifies the version for the bus interface.

Configuration Parameters

Certain bus definitions contain user configurable parameters.

**Parameter:** Specifies the parameter name.

**Value:** Specifies the value you define for the parameter.

**Consistent:** Specifies whether a compatible bus interface must have the same value for this bus parameter. If the bus interface has a different value for any parameters that are marked with consistent set to yes, this bus interface will not be connectable.

Signal Map Definition

The signal map of the bus interface specifies the pins on the instance that correspond to the bus definition signals. The bus definition signals are shown on the left, under the Bus Interface Definition. This information includes the name, direction and required properties of the signal.

The pins for your instance are shown in the columns under the Component Instance. The signal element is a drop-down list of the pins that can be mapped for that definition signal.

If the Req field of the signal definition is Yes, you must map it to a pin on your instance for this bus interface to be considered legal. If it is No, you can leave it unmapped.

Catalog

In the Libero SoC, from the View menu choose Windows > Catalog.

The Catalog displays a list of available cores, busses and macros (see image below).

![Figure 128 · Libero SoC Catalog](image)

From the Catalog, you can create a component from the list of available cores, add a processor or peripheral, add a bus interface to your SmartDesign component, instantiate simulation cores or add a macro (Arithmetic, Basic Block, etc.) to your SmartDesign component.

Double-click a core to configure it and add it to your design. Configured cores are added to your list of Components/Modules in the Design Explorer.

Click the Simulation Mode checkbox to instantiate simulation cores in your SmartDesign Testbench. Simulation cores are basic cores that are useful for stimulus, such as driving clocks, resets, and pulses.
Viewing Cores in the Catalog

The font indicates the status of the core:

- Plain text - In vault and available for use
- Asterisk after name (*) - Newer version of the core (VLN) available for download
- Italics - Core is available for download but not in your vault
- Strikethrough - core is not valid for this version of Libero SoC

The colored icons indicate the license status. Blank means that the core is not license protected in any way. Colored icons mean that the core is license protected, with the following meanings:

- **Green Key** - Fully licensed; supports the entire design flow.
- **Yellow Key** - Has a limited or evaluation license only. Precompiled simulation libraries are provided, enabling the core to be instantiated and simulated within Libero SoC. Using the Evaluation version of the core it is possible to create and simulate the complete design in which the core is being included. The design is not synthesizable (RTL code is not provided). No license feature in the license.dat file is needed to run the core in evaluation mode. You can purchase a license to generate an obfuscated or RTL netlist.
- **Yellow Key with Red Circle** - License is protected; you are not licensed to use this core.

Right-click any item in the Catalog and choose Show Details for a short summary of the core specifications. Choose Open Documentation for more information on the Core. Right-click and choose Configure Core to open the core generator.

Click the **Name** column heading to sort the cores alphabetically.

You can filter the cores according to the data in the Name and Description fields. Type the data into the filter field to view the cores that match the filter. You may find it helpful to set the Display setting in the Catalog Options to **List cores alphabetically** when using the filters to search for cores. By default the filter contains a beginning and ending '*', so if you type 'controller' you get all cores with controller in the core name (case insensitive search) or in the core description. For example, to list all the Accumulator cores, in the filter field type:

```
accu
```

**Catalog Options**

Click the Options button (or the drop-down arrow next to it) to import a core, reload the Catalog, or modify the Catalog Options.

You may want to import a core from a file when:

- You do not have access to the internet and cannot download the core, or
- A core is not complete and has not been posted to the web (you have an evaluation core)

**Manually Downloading MegaVaults and Individual CPZ files**

When Libero is used in an environment without automatic access to Microsemi's online IP repositories via the Internet; see this article explaining how to download MegaVaults and individual CPZ files.
Catalog Options Dialog Box

The Catalog Options dialog box (as shown below) enables you to customize your Catalog. You can add a repository, set the location of your vault, and change the View Settings for the Catalog. To display this dialog box, click the Catalog Options button.

![Catalog Display Options Dialog Box](Figure 129 · 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:

```bash
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...
variable, your configured cores are retained across sessions.

To set the extract location on Linux, type the following command:

```
setenv MSCC_IDE_VAULT_EXTRACT_LOCATION /home/vault_extract
```

To set the extract vault location on Windows:

Add a new environment variable MSCC_IDE_VAULT_EXTRACT_LOCATION in System Properties and specify your extract location.

If you do not specify the extract location, a temporary location will be created by Libero and it will be accessed only while the current session is active. If the session is no longer active, the temporary extract location will be cleaned up by Libero. If you specify the extract location, it will be available for any instance of libero on that machine, and it is your responsibility to clean up the extract location.

**View Settings**

**Display**

**Group cores by function** - Displays a list of cores, sorted by function. Click any function to expand the list and view specific cores.

**List cores alphabetically** - Displays an expanded list of all cores, sorted alphabetically. Double click a core to configure it. This view is often the best option if you are using the filters to customize your display.

**Show core version** - Shows/hides the core version.

**Filters**

**Filter field** - Type text in the Filter Field to display only cores that match the text in your filter. For example, to view cores that include 'sub' in the name, set the Filter Field to **Name** and type **sub**.

**Display only latest version of a core** - Shows/hides older versions of cores; this feature is useful if you are designing with an older family and wish to use an older core.

**Show all local and remote cores** - Displays all cores in your Catalog.

**Show local cores only** - Displays only the cores in your local vault in your Catalog; omits any remote cores.

**Show remote cores that are not in my vault** - Displays remote cores that have not been added to your vault in your Catalog.
Changing Output Port Capacitance

Output propagation delay is affected by both the capacitive loading on the board and the I/O standard. The I/O Attribute Editor in ChipPlanner provides a mechanism for setting the expected capacitance to improve the propagation delay model. SmartTime automatically uses the modified delay model for delay calculations.

To change the output port capacitance and view the effect of this change in SmartTime Timing Analyzer, refer to the following example. The figure below shows the delay from DFN1 to output port Q. It shows a delay of 6.603 ns based on the default loading of 5 pF.

1. Open the I/O Attribute Editor and change the output load to 15 pF.
2. Select File > Save.
4. Open the SmartTime Timing Analyzer.

You can see that the Clock to Output delay changed to 5.952 ns.

Core Manager

The Core Manager only lists cores that are in your current project. If any of the cores in your current project are not in your vault, you can use the Core Manager to download them all at once.

For example, if you download a sample project and open it, you may not have all the cores in your local vault. In this instance you can use the Core Manager to view and download them with one click. Click Download All to add any missing cores to your vault. To add any individual core, click the green download button.
configure_design_initialization_data

This Tcl command sets the parameter values needed for generating initialization data.

```
configure_design_initialization_data
-second_stage_start_address {<sNVM_address_for_the_second_initialization_client>}
-third_stage_uprom_start_address {<uPROM_address_for_the_third_initialization_stage_client>}
-third_stage_snvm_start_address {<sNVM_address_for_the_third_initialization_stage_client>}
-third_stage_spi_type {<SPIFLASH_NO_BINDING_PLAINTEXT | SPIFLASH_BINDING_DEFAULT | SPIFLASH_BINDING_UEK1 | SPIFLASH_BINDING_UEK2>}
-third_stage_spi_clock_divider {<1 | 2 | 4 | 6>}
-auto_calib_timeout {<Auto_Calibration_timeout_value_in_milliseconds>}
-broadcast_RAMS {<0 | 1>}
-custom_cfg_file {<Initialization_file_for_custom_configuration>}
```

Arguments

- `second_stage_start_address`
  String parameter for the start address of the second stage initialization client. Specified as a 32-bit hexadecimal string.
  
  The first stage client is always placed in sNVM, so it must be a valid sNVM address aligned on a page boundary.
  There are 221 sNVM pages and each page is 256 bytes long, so the address will be between 0 and DC00.

Notes:

Although the actual size of each page is 256 bytes, only 252 bytes are available to the user. The first stage initialization client is always added to SNVM at 0xDC00 (page 220). So the valid addresses for the second stage initialization client are 0x0 (page 0) to 0xDB00 (page 219).

- `third_stage_uprom_start_address`
  String parameter for the uPROM start address of the third stage initialization client. It is optional. Specified as a 32-bit hexadecimal string and must be valid uPROM address aligned on a block boundary.

- `third_stage_snvm_start_address`
  String parameter for the sNVM start address of the third stage initialization client. It is optional. Specified as a 32-bit hexadecimal string and must be valid sNVM address.

- `third_stage_spi_start_address`
  String parameter for the SPIFLASH start address of the third stage initialization client. It is optional. Specified as a 32-bit hexadecimal string and must be valid SPIFLASH address.

- `third_stage_spi_type`
  The value must be one of SPIFLASH_NO_BINDING_PLAINTEXT or SPIFLASH_BINDING_DEFAULT or SPIFLASH_BINDING_UEK1 or SPIFLASH_BINDING_UEK2.
  This parameter determines the valid value for parameter ‘third_stage_start_address’.
-third_stage_spi_clock_divider

Specifies the clock frequency appropriate for the SPIFLASH memory on board. The value can be 1, 2, 4, or 6. The default value is 1 which is 80 MHz. The other values are 2-40 MHz, 4-20 MHz and 6-13.33 Mhz.

-init_timeout

Timeout value in seconds. Initialization is aborted if it does not complete before timeout expires.

The value can be between 1 and 128. The default value is 128.

-broadcast_RAMs

Specifies broadcast instructions to initialize RAM's to zero's. Value can be either 0 or 1. It is optional.

-custom_cfg_file

Specifies the initialization file for custom configuration. It is optional.

Example

Example to initialize data with sNVM client

```
configure_design_initialization_data
-secound_stage_start_address {0x0000aaa0}
-third_stage_uprom_start_address {0x00000000}
-third_stage_snvm_start_address {0x0000aaa0}
-third_stage_spi_start_address {0x000000400}
-third_stage_spi_type {SPIFLASH_NO_BINDING_PLAINTEXT}
-third_stage_spi_clock_divider {4}
-init_timeout 85
-auto_calib_timeout {1400}
-broadcast_RAMs {0}
```

Example to initialize data with uPROM client

```
configure_design_initialization_data
-secound_stage_start_address {0x00000000}
-third_stage_uprom_start_address {0xfffffee2}
-third_stage_snvm_start_address {0x00000000}
-third_stage_spi_start_address {0x00000000400}
-third_stage_spi_type {SPIFLASH_NO_BINDING_PLAINTEXT}
-third_stage_spi_clock_divider {4}
-init_timeout 45
-auto_calib_timeout {2000}
-broadcast_RAMs {0}
```

Example to initialize data with SPI-FLASH client

```
configure_design_initialization_data
-secound_stage_start_address {0x00000000}
-third_stage_uprom_start_address {0x00000000}
-third_stage_snvm_start_address {0x00000000}
-third_stage_spi_start_address {0x000000AC120}
-third_stage_spi_type {SPIFLASH_BINDING_UK2}
-third_stage_spi_clock_divider {2}
-init_timeout 20
-auto_calib_timeout {500}
-broadcast_RAMs {1}
```

See Also

generate_design_initialization_data
Configure Permanent Locks for Production

Configure Permanent Locks for Production is a GUI-based tool that guides the user step by step on how to configure the Permanent Locks for Production. The wizard has six steps/pages executed in sequential order. One Time Programmable (OTP) settings in the Permanent Locks page are applied to configured Security settings from the Configure Security tool. The subsequent pages have read only fields, which will be affected by Permanent Lock settings. These settings can only be configured by the Configure Security tool.

If you configure any Permanent Lock settings, you will be forced to go through each page to review the Security settings to make sure they are as desired. The settings cannot be changed once they have been programmed.

1. Permanent Locks
2. User keys in Configure Security
3. Update Policy in Configure Security
4. Debug Policy in Configure Security
5. Microsemi Factory Access in Configure Security
6. JTAG/SPI Slave Commands Policy in Configure Security

Summary Window

The summary window displays the summary of the current page configuration settings. Based on the selection made in the first page, the summary for the subsequent pages change. The window will scroll to the current page as you move from page to page.

Back
Click Back to return to the previous step.

Next
Click Next to proceed to the next step.

Finish
Click Finish to complete the configuration after executing the all the steps in sequential order.

Save Summary to File
Click Save Summary to File to save the display in the Summary field to a file.
configure_snvm

Tcl command; configures the sNVM clients in the sNVM tab of the Design and Memory Initialization tool. Can specify user sNVM clients using this command.

**Note:** You must run Generate Design Initialization Data (generate_design_initialization_data) after configuring sNVM (configure_snvm) and/or Design Initialization (configure_design_initialization_data).

configure_snvm
-cfg_file <path_to_configuration_file.cfg>

**Arguments**

- **-cfg_file** path_to_configuration_file.cfg
  
  Specifies the path to the configuration file of the sNVM client. It is mandatory.

**Examples**

configure_snvm
-cfg_file
{../../Downloads/mpf_dg0852_liberosocv12p0_df/Libero_Project/TVS_Demo/designer/TVS_Demo/ SNVM.cfg

**See Also**

generate_design_initialization_data

**SPM_OTP**

Configures the parameters for SPM_OTP.

configure_tool –name {SPM_OTP} parameter:value pair

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>permanently_disable_debugging</td>
<td>bool</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>permanently_disable_dpk</td>
<td>bool</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------</td>
<td>------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>permanently_disable_factory_access</td>
<td>bool</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>permanently_disable_prog_interfaces</td>
<td>bool</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>permanently_disable_upk1</td>
<td>bool</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>permanently_disable_upk2</td>
<td>bool</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>permanently_write_protect_fabric</td>
<td>bool</td>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>
Examples
The following example specifies that SPM_OTP tool is configured to permanently disable user keys UPK1 and UPK2.

```bash
configure_tool
- name {SPM_OTP} \
- params {permanently_disable_debugging:false} \ 
- params {permanently_disable_dpk:false} \ 
- params {permanently_disable_factory_access:false} \ 
- params {permanently_disable_programming_interfaces:false} \ 
- params {permanently_disable_upk1:true} \ 
- params {permanently_disable_upk2:true} \ 
- params {permanently_write_protect_fabric:false}
```

The following example specifies that SPM_OTP tool is configured to permanently disable programming interfaces.

```bash
configure_tool
- name {SPM_OTP} \
- params {permanently_disable_debugging:false} \ 
- params {permanently_disable_dpk:false} \ 
- params {permanently_disable_factory_access:false} \ 
- params {permanently_disable_programming_interfaces:true} \ 
- params {permanently_disable_upk1:false} \ 
- params {permanently_disable_upk2:false} \ 
- params {permanently_write_protect_fabric:false}
```

See Also
remove_permanent_locks

Importing Source Files – Copying Files Locally

Designer in Libero SoC cannot import files from outside your project without copying them to your local project folder. You may import source files from other locations, but they are always copied to your local folder. Designer in Libero SoC always audits the local file after you import; it does not audit the original file.

When the Project Manager asks you if you want to copy files "locally", it means 'copy the files to your local project folder'. If you do not wish to copy the files to your local project folder, you cannot import them. Your local project folder contains files related to your Libero SoC project.

Files copied to your local folders are copied directly into their relevant directory: netlists are copied to the synthesis folder; source files are copied to the hdl folder and constraint files to constraint folder, etc. The files are also added to the Libero SoC project; they appear in the Files tab.

Create Clock Constraint Dialog Box

Use this dialog box to enter a clock constraint setting.

It displays a typical clock waveform with its associated clock information. You can enter or modify this information, and save the final settings as long as the constraint information is consistent and defines the clock waveform completely. The tool displays errors and warnings if information is missing or incorrect.

To open the Create Clock Constraint dialog box (shown below) from the SmartTime Constraints Editor, choose
Constraints > Clock.

Figure 132 · 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

Select Source Pins for Clock Constraint Dialog Box

Use this dialog box to find and choose the clock source from the list of available pins.
To open the Select Source Pins for the Clock Constraint dialog box (shown below) from the SmartTime Constraints Editor, click the Browse button to the right of the Clock source field in the Create Clock Constraint dialog box.

![Select Source Pins for Clock Constraint Dialog Box](image)

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

Specifying Clock Constraints

Specifying clock constraints is the most effective way to constrain and verify the timing behavior of a sequential design. Use clock constraints to meet your performance goals.

To specify a clock constraint:
1. Add the constraint in the editable constraints grid or open the Create Clock Constraint dialog box using one of the following methods:
   - Click the icon in the Constraints Editor.
   - Right-click the Clock in the Constraint Browser and choose Add Clock Constraint.
   - Double-click Clock in the Constraint Browser.
   - Choose Clock from the Constraints drop-down menu (Constraints > Clock)

   The Create Clock Constraint dialog box appears (as shown below).

   Figure 134 · Create Clock Constraint Dialog Box

2. Select the pin to use as the clock source. You can click the Browse button to display the Select Source Pins for Clock Constraint Dialog Box (as shown below).

   Note: Do not select a source pin when you specify a virtual clock. Virtual clocks can be used to define a clock outside the FPGA that is used to synchronize I/Os.

   Use the Choose the Clock Source Pin dialog box to display a list of source pins from which you can choose. By default, it displays the explicit clock sources of the design. To choose other pins in the design as clock source pins, select Filter available objects - Pin Type as Explicit clocks, Potential clocks, All Ports, All Pins, All Nets, Pins on clock network, or Nets in clock network. To display a subset of the displayed clock source pins, you can create and apply a filter.

   Multiple source pins can be specified for the same clock when a single clock is entering the FPGA using multiple inputs with different delays.

   Click OK to save these dialog box settings.

3. Specify the Period in nanoseconds (ns) or Frequency in megahertz (MHz).
4. Modify the Clock Name. The name of the first clock source is provided as default.
5. Modify the Duty cycle, if needed.
6. Modify the Offset of the clock, if needed.
7. Modify the first edge direction of the clock, if needed.
8. Select the check box for Add this clock to an existing one with the same source, if needed.
9. Click OK. The new constraint appears in the Constraints List.

Note: When you choose File > Save, the Timing Constraints Editor saves the newly created constraint in the database.

Figure 135 · Timing Constraint View

Create Generated Clock Constraint Dialog Box

Use this dialog box to specify generated clock constraint settings.

It displays a relationship between the clock source and its reference clock. You can enter or modify this information, and save the final settings as long as the constraint information is consistent. The tool displays errors and warnings if the information is missing or incorrect.

To open the Create Generated Clock Constraint dialog box (shown below) from the SmartTime Constraints Editor, choose Constraints > Generated Clock.
**Figure 136 · 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 tool tip indicates an error. The following example shows how to specify the clock edges.

If LSB is the generated clock from CLK clock source, the edge values must be [1 3 5].
If MSB is the generated clock from CLK clock source, the edge values must be [1 5 9].

**Edge Shift**
Specify a list of three floating point numbers that represents the amount of shift, in library time units, that the specified edges are to undergo to yield the final generated clock waveform. These floating point values can be positive or negative. Positive value indicates a shift later in time, while negative indicates a shift earlier in time. For example: An edge shift of {1 1 1} on the LSB generated clock, would shift each derived edge by 1 time unit. To create a 200MHz clock from a 100MHz clock, use edge {1 2 3} and edge shift {0 -2.5 -5.0}

**Generated Waveform**
Specify whether the generated waveform is the same or inverted with respect to the reference waveform. Click OK.

**Phase**
This field is primarily used to report the information captured from the CCC configuration process, and when constraint is auto-generated. Meaningful phase values are: 0, 45, 90, 135, 180, 225, 270, and 315. This field is used to report the information captured from the CCC configuration process, and when the constraint is auto-generated.

**PLL Output**
This field refers to the CCC GL0/1/2/3 output that is fed back to the PLL (in the CCC). This field is primarily used to report the information captured from the CCC configuration process, and when constraint is auto-generated.
PLL Feedback
This field refers to the manner in which the GL/0/1/2/3 output signal of the CCC is connected to the PLL’s FBCLK input. This field is primarily used to report the information captured from the CCC configuration process, and when constraint is auto-generated.

Add Clock to Existing Clock
Specifies that the generated clock constraint is a new clock constraint in addition to the existing one at the same source. The name of the clock constraint should be different from the existing clock constraint. When this option is selected, master clock must be specified.

Master Clock
Specifies the master clock used for the generated clock when multiple clocks fan into the master pin. It can be selected from the drop-down menu. This option is used in conjunction with the add option of the generated clock.

Comment
Enter a single line of text that describes the generated clock constraints purpose.

See Also
create_generated_clock (SDC)
Specifying Generated Clock Constraints
Select Generated Clock Source

Select Generated Clock Source Dialog Box
Use this dialog box to find and choose the generated clock source from the list of available pins.
To open the Select Generated Clock Source dialog box (shown below) from the Timing Constraints Editor, open the Create Generated Clock Constraint dialog box and click the Browse button for the Clock Pin.
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.
Specifying Generated Clock Constraints

Specifying a generated clock constraint enables you to define an internally generated clock for your design and verify its timing behavior. Use generated clock constraints and clock constraints to meet your performance goals.

To specify a generated clock constraint:
1. Open the Create Generated Clock Constraint dialog box using one of the following methods:
   - Click the icon.
   - Right-click the Generated Clock in the Constraint Browser and choose Add Generated Clock.
   - Double-click the Generated Clock Constraints grid. The Create Generated Clock Constraint dialog box appears (as shown below).

2. Select a Clock Pin to use as the generated clock source. To display a list of available generated clock source pins, click the Browse button. The Select Generated Clock Source dialog box appears (as shown below).
Figure 139 · 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.
Select Generated Clock Reference Dialog Box

Use this dialog box to find and choose the generated clock reference pin from the list of available pins.

To open the Select Generated Clock Reference dialog box (shown below) from the SmartTime Constraints Editor, open the Create Generated Clock Constraint Dialog Box dialog box and click the Browse button for the Clock Reference.

![Select Generated Clock Reference Dialog Box](image)

Filter Available Pins

- **Pin type** – Displays the Available Pin types. The Pin Type options for Generated Clock Reference are:
  - Input Ports
  - All Pins

- **Pattern** – The default pattern is *, which is a wild-card match for all. You can specify any string value.
  - Select Filter to filter the available pins based on the specified Pin Type and Pattern.
  - The list box displays the list of available pins based on the filter. Select the pins from the list and click OK to select the Generated Clock Reference Pin.

See Also

- Specifying generated clock constraints

Design Hierarchy in the Design Explorer

The Design Hierarchy tab displays a hierarchical representation of the design based on the source files in the project. It also displays elaborated hierarchy constructed by propagating correct values for parameters and generics. The software continuously analyzes the source files and updates the content. The Design Hierarchy tab
(see figure below) displays the structure of the modules and components as they relate to each other along with parameter/generic names and its values on the tool tip for which the module is instantiated. It also displays architecture name for a given entity and Configuration for VHDL modules.
A module can have multiple elaborations depending on the different parameters/generics used in the instantiation of the module and all of these elaborated modules will be shown in the Design Hierarchy. The parameterized instantiated module will be shown as `elab<num>;<modulename>`.

Modules are instantiated with their actual names in the SmartDesign. If a module with elaborated name in the Design Hierarchy has to be instantiated in the SmartDesign, an instance of the original module is created in the SmartDesign. The following figure shows the design hierarchy with elaborated modules.

![Design Hierarchy with Elaborated modules (Verilog)](image_url)
Figure 143 · Design Hierarchy with Elaborated modules (VHDL)

Modules which are not part of the elaboration will be shown in the complete hierarchy but they remain grayed out. When you create a core from a module, all the elaborated modules of that module will be shown as HDL+ core modules. You can get the parameter value of an elaborated module by selecting **Show Module parameters** on the right click menu of the elaborated module.

**Note:** A tool tip on each module shows all the parameters with their values for the instantiated module.

**Note:** Synthesis output will be the same for different elaborations of the same module i.e. `elab0:module1` and `elab1:module1` will have the same synthesis output. When one of the elaborated module is set as root, all the elaborations will be highlighted in the Design Hierarchy as shown in the below figure.
You can change the display mode of the Design Hierarchy by selecting Components or Modules from the Show drop-down list. The components view displays the entire design hierarchy; the modules view displays only schematic and HDL modules.

You can build the Design Hierarchy and Simulation Hierarchy by clicking the Build Hierarchy button.

A yellow icon indicates that the Design Hierarchy is out of date (invalidated). Any change to the design sources/stimuli invalidates the Design Hierarchy/ Stimulus Hierarchy. Click the Build Hierarchy button to rebuild the Design Hierarchy.

The file name (the file that defines the block) appears next to the block name in parentheses.

To view the location of a component, right-click and choose Properties. The Properties dialog box displays the path name, created date, and last modified date.

All integrated source editors are linked with the SoC software. If a source is modified and the modification changes the hierarchy of the design, the Build Hierarchy automatically updates to reflect the change.

If you want to update the Design Hierarchy, from the View menu, choose Refresh Design Hierarchy.
To open a component:

Double-click a component in the Design Hierarchy to open it. Depending on the block type and design state, several possible options are available from the right-click menu. You can instantiate a component from the Design Hierarchy to the SmartDesign Canvas. See the SmartDesign User Guide for more information.

Icons in the Hierarchy indicate the type of component and the state, as shown in the table below.

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Icon]</td>
<td>SmartDesign component</td>
</tr>
<tr>
<td>![Icon]</td>
<td>SmartDesign component with HDL netlist not generated</td>
</tr>
<tr>
<td>![Icon]</td>
<td>IP core was instantiated into SmartDesign but the HDL netlist has not been generated</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Core</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Error during core validation</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Updated core available for download</td>
</tr>
<tr>
<td>![Icon]</td>
<td>HDL netlist</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Shows ungenerated components</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Shows unknown modules</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Expands all the files and folders in the Design Hierarchy</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Collapses all the files and folders in the Design Hierarchy</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Finds the files in the Design Hierarchy</td>
</tr>
</tbody>
</table>
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.

![Linked Files with Relative Path in Design Hierarchy with Environment Variable](image)

Properties of File in Design Hierarchy Shows Relative Path and Relative Path root directory

![File Path Properties in Design Hierarchy Representing Relative Path and Relative Path Root Directory](image)

**Digest File**

Users can verify which bitstream file was programmed onto their devices by running the VERIFY or VERIFY_DIGEST actions on each device that was programmed. This is a costly and time-consuming process. To speed up the verification process, digests are printed during bitstream generation and bitstream programming. These digests can be compared to verify that all of the devices were programmed with the correct bitstream file.

The bitstream file is divided into three major component sections: FPGA fabric, eNVM, and Security. A valid bitstream will contain a combination of any of the three primary bitstream components.

**Use Case**

When a customer creates a design in Libero and then exports the STAPL file (for FlashPro) or programming job (for FlashPro Express), the digest of each of the primary components is printed in the Libero log window and saved in a digest file under the export folder. The digest file is a text file containing the bitstream component name with its corresponding digest. The name of the digest file will match the name of the STAPL/programming job exported, and will be appended with a ".digest" extension.
The customer then sends the STAPL/programming job to a production programming house. Now, when the devices are programmed, the digest of each of the primary components is printed in the log window. The production programming house saves the log files and sends the devices along with log files back to the customer. The customer can then verify that the correct design was programmed on the device by matching the digests in the log file with that in the *.digest file under the Libero export folder.

See Also

Export Bitstream

Design Rules Check

The Design Rules Check runs automatically when you generate your SmartDesign; the results appear in the Reports tab. You can also initiate a Design Rules Check by clicking on the button of the SmartDesign Canvas tab menu.

To view the results, from the Design menu, choose Reports.

- **Status** displays an icon to indicate if the message is an error or a warning (as shown in the figure below). Error messages are shown with a small red sign and warning messages with a yellow exclamation point.
- **Message** identifies the specific error/warning (see list below); click any message to see where it appears on the Canvas.
- **Details** provides information related to the Message.

![Figure 147 · Design Rules Check Results](image)

**Message Types:**

**Unused Instance** - You must remove this instance or connect at least one output pin to the rest of the design.

**Out-of-date Instance** - You must update the instance to reflect a change in the component referenced by this instance.

**Undriven Pin** - To correct the error you must connect the pin to a driver or change the state, i.e. tie low (GND) or tie high (VCC).

**Floating Driver** - You can mark the pin unused if it is not going to be used in the current design. Pins marked unused are ignored by the Design Rules Check.
**Unconnected Bus Interface** - You must connect this bus interface to a compatible port because it is required connection.

**Required Bus Interface Connection** – You must connect this bus interface before you can generate the design. These are typically silicon connection rules.

**Exceeded Allowable Instances for Core** – Some IP cores can only be instantiated a certain number of times for legal design because of silicon limitations. You must remove the extra instances.

**Incompatible Family Configuration** – The instance is not configured to work with this project’s Family setting. Either it is not supported by this family or you need to re-instantiate the core.

**Incompatible Die Configuration** – The instance is not configured to work with this project’s Die setting. Either it is not supported or you need to reconfigure the Die configuration.

**No RTL License, No Obfuscated License, No Evaluation License** – You do not have the proper license to generate this core. Contact Microsemi SoC to obtain the necessary license.

**No Top level Ports** - There are no ports on the top level. To auto-connect top-level ports, right-click the Canvas and choose Auto-connect

**Self-Instantiation** - A component cannot instantiate itself-This is reported only in the Log/Message Window.

### Editable Constraints Grid

The Constraints Editor enables you to add, edit and delete.

![Constraints Editor](image)

**Figure 148 · 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**.
export_spiflash_image

This Tcl command exports a SPI Flash image file to a specified directory.

```
export_spiflash_image -file_name {name of file} -export_dir {absolute path to folder location}
```

Arguments

- **-file_name name of file**
  The name of the image file.
- **-export_dir absolute path to folder location**
  Folder/directory location.

extended_run_lib

**Note:** This is not a Tcl command; it is a shell script that can be run from the command line.

The extended_run_lib Tcl script enables you to run the multiple pass layout in batch mode from a command line.

```
$ACTEL_SW_DIR/bin/libero script:$ACTEL_SW_DIR/scripts/extended_run_lib.tcl
```

**Note:**
- There is no option to save the design files from all the passes. Only the (Timing or Power) result reports from all the passes are saved.

Arguments

- **-root path/designer/module_name**
  The path to the root module located under the designer directory of the Libero project.
- **-n numPasses**
  Sets the number of passes to run. The default number of passes is 5.
- **-starting_seed_index numIndex**
  Indicates the specific index into the array of random seeds which is to be the starting point for the passes. Value may range from 1 to 100. If not specified, the default behavior is to continue from the last seed index that was used.
- **-compare_criteria value**
  Sets the criteria for comparing results between passes. The default value is set to frequency when the –c option is given or timing constraints are absent. Otherwise, the default value is set to violations.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>frequency</td>
<td>Use clock frequency as criteria for comparing the results between passes. This option can be used in conjunction with the -c option (described below).</td>
</tr>
<tr>
<td>violations</td>
<td>Use timing violations as criteria for comparing the results between passes. This option can be used in conjunction with the -analysis, -slack_criteria and -stop_on_success options (described below).</td>
</tr>
</tbody>
</table>
Value Description
---
Power Use total power as criteria for comparing the results between passes, where lowest total power is the goal.

`[-c clockName]`
Applies only when the clock frequency comparison criteria is used. Specifies the particular clock that is to be examined. If no clock is specified, then the slowest clock frequency in the design in a given pass is used. The clock name should match with one of the Clock Domains in the Summary section of the Timing report.

`[-analysis value]`
Applies only when the timing violations comparison criteria is used. Specifies the type of timing violations (the slack) to examine. The following table shows the acceptable values for this argument:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max</td>
<td>Examines timing violations (slack) obtained from maximum delay analysis. This is the default.</td>
</tr>
<tr>
<td>Min</td>
<td>Examines timing violations (slack) obtained from minimum delay analysis.</td>
</tr>
</tbody>
</table>

`[-slack_criteria value]`
Applies only when the timing violations comparison criteria is used. Specifies how to evaluate the timing violations (slack). The type of timing violations (slack) is determined by the -analysis option. The following table shows the acceptable values for this argument:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worst</td>
<td>Sets the timing violations criteria to Worst slack. For each pass obtains the most amount of negative slack (or least amount of positive slack if all constraints are met) from the timing violations report. The largest value out of all passes will determine the best pass. This is the default.</td>
</tr>
<tr>
<td>Tns</td>
<td>Sets the timing violations criteria to Total Negative Slack (tns). For each pass it obtains the sum of negative slack values from the first 100 paths from the timing violations report. The largest value out of all passes determines the best pass. If no negative slacks exist for a pass, then the worst slack is used to evaluate that pass.</td>
</tr>
</tbody>
</table>

`[-stop_on_success]`
Applies only when the timing violations comparison criteria is used. The type of timing violations (slack) is determined by the -analysis option. Stops running the remaining passes if all timing constraints have been met (when there are no negative slacks reported in the timing violations report).

`[-timing_driven|-standard]`
Sets layout mode to timing driven or standard (non-timing driven). The default is -timing_driven or the mode used in the previous layout command.

`[-power_driven value]`
Enables or disables power-driven layout. The default is off or the mode used in the previous layout command. The following table shows the acceptable values for this argument:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off</td>
<td>Does not run power-driven layout.</td>
</tr>
<tr>
<td>On</td>
<td>Enables power-driven layout.</td>
</tr>
</tbody>
</table>
[<-placer_high_effort value] Sets placer effort level. The default is off or the mode used in the previous layout command. The following table shows the acceptable values for this argument:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off</td>
<td>Runs layout in regular effort.</td>
</tr>
<tr>
<td>On</td>
<td>Activates high effort layout mode.</td>
</tr>
</tbody>
</table>

Return
A non-zero value will be returned on error.

Exceptions
None

Files Tab and File Types
The Files tab displays all the files associated with your project, listed in the directories in which they appear. Right-clicking a file in the Files tab provides a menu of available options specific to the file type. You can delete files from the project and the disk by selecting Delete from the right-click menu.

You can instantiate a component by dragging the component to a SmartDesign Canvas or by selecting Instantiate in SmartDesign from the right-click menu. See the SmartDesign User Guide for more details.

You can configure a component by double-clicking the component or by selecting Open Component from the right-click menu.

File Types
When you create a new project in the Libero SoC it automatically creates new directories and project files. Your project directory contains all of your 'local' project files. If you import files from outside your current project, the files must be copied into your local project folder. (The Project Manager enables you to manage your files as you import them.)

Depending on your project preferences and the version of Libero SoC you installed, the software creates directories for your project.

The top level directory (<project_name>) contains your PRJ file; only one PRJ file is enabled for each Libero SoC project.

- **component** directory - Stores your SmartDesign components (SDB and CXF files) for your Libero SoC project.
- **constraint** directory - All your constraint files (SDC, PDC)
- **designer** directory - *.ba.sdf, *.ba.v(hd), STP, TCL (used to run designer), designer.log (logfile)
- **hdl** directory - all hdl sources. *.vhd if VHDL, *.v and *.h if Verilog, *.sv if SystemVerilog
- **simulation** directory - meminit.dat, modelsim.ini files
- **smartgen** directory - GEN files and LOG files from generated cores
- **stimulus** directory - BTIM and VHD stimulus files
- **synthesis** directory - *.edm, *_syn.prj (Synplify log file), *.srr (Synplify logfile), *.tcl (used to run synthesis) and many other files generated by the tools (not managed by Libero SoC)
- **tooldata** directory - includes the log file for your project with device details.
**generate_design_initialization_data**

This Tcl command creates the memory files on disk, adds the initialization clients to the target memories, and writes the configuration files to disk.

This command also runs validation on the saved configuration files and writes out errors (if any) in the log. This command causes the UI of the Configure Design Initialization Data and Memories tool to refresh and show the latest configuration and validation errors (if any) in the tables.

This command takes no parameters.

---

**See Also**

configure_design_initialization_data

---

**Importing Files**

Anything that describes your design, or is needed to program the device, is a project source. These may include schematics, HDL files, simulation files, testbenches, etc. Import these source files.

**To import a file:**

1. From the **File** menu, choose **Import Files**.
2. In **Files of type**, choose the file type.
3. In **Look in**, navigate to the drive/folder where the file is located.
4. Select the file to import and click **Open**.

**Note:** You cannot import a Verilog File into a VHDL project and vice versa.

---

**File Types for Import**

<table>
<thead>
<tr>
<th>File Type</th>
<th>File Extension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Behavioral and Structural VHDL; VHDL Package</td>
<td>*.vhd, *.vhdl</td>
</tr>
<tr>
<td>Design Block Core</td>
<td>*.gen</td>
</tr>
<tr>
<td>Verilog Include</td>
<td>*.h</td>
</tr>
<tr>
<td>Behavioral and Structural Verilog</td>
<td>*.v, *.sv</td>
</tr>
<tr>
<td>Netlist Verilog</td>
<td>*.vm</td>
</tr>
<tr>
<td>Stimulus</td>
<td>*.vhd, *.vhdl, *.v, *.sv</td>
</tr>
<tr>
<td>Memory file</td>
<td>*.mem</td>
</tr>
<tr>
<td>Components (Designer Blocks, Synplify DSP)</td>
<td>*.cxf</td>
</tr>
<tr>
<td>MSS Components</td>
<td>*.cxz</td>
</tr>
</tbody>
</table>

---

**Bus Interfaces**

When you add a bus interface the Edit Core Definition dialog box provides the following Microsemi SoC-specific bus interfaces:

- AHB – Master, Slave, Mirrored Master, MirroredSlave
- APB – Master, Slave, MirroredMaster, MirroredSlave
- AXI – Master, Slave, MirroredMaster, MirrorSlave, System
- AXI 4 - Master, Slave, MirroredMaster, MirrorSlave

**Layout Error Message: layout4NoValidPlacement**

This is a generic error produced by the placer when it is unable to place a design. The most common cause for this failure is that the placer was unable to find a solution which could fit the design into the chip, either because the design is close to maximum utilization, or logic cannot be fit into user-defined region constraints.

If Libero is unable to find a legal placement, a list of unplaced cells will be provided in the log. The cells in this list may not be the cause of the placement problem; it is quite possible that some other constrained block of logic which 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 & Route.

However, the cells in this list may not be the cause of the placement problem; it is quite possible that some other constrained block of logic which was placed first and now prohibits further placement. If removing the placement constraints on the unplaced cells does not succeed:

- Remove all region constraints and re-run Place & Route. Some designers make it a practice to put all their region constraints in a single, separate PDC file; in which case they need only disable that file.
  - If this Place & Route re-run still fails, there may be wider issues with the design's size and complexity that cannot be addressed by changes to P&R options.
  - If the unconstrained Place & Route re-run succeeds, then the user should add back constraints a few regions at a time in order of "simplicity". Usually, big regions with lots of free space are "simpler" for the placer, whereas tall/narrow regions with high utilization are "harder". Re-run Place & Route with each constraint restoration and repeat the process until the failing region(s) is identified.

Depending on requirements, the failing region may be handled by removing or changing its constraints, or revising its design to use less resources.

The user may also re-run the Placer in high-effort mode. Applying high-effort mode to a design which is very full can incur additional runtime and may produce a placement solution which may not meet tight timing constraints, owing to the fact that the placer will aggressively attempt to fit the design. In practice, customers are encouraged to apply the previous suggestions first; and utilize high-effort mode only when other approaches have been exhausted.

**Layout Error Message: layout4DesignHard**

This design is very difficult to place, and high-effort techniques were required to make it fit. This may lead to increased layout runtime and diminished timing performance.

This message typically appears in designs with high utilization -- a very full design, or a design with region constraints which are, themselves, very full. It can also occur in designs with moderate utilization but with numerous, long carry chains.

No immediate action is required on the user's part. However, if this notice is observed during Layout, the resultant performance of the design and the runtime of the Layout tools may not be optimal, and there is a strong possibility that reducing the size of the design, or relaxing region and floorplanning constraints, will help to improve timing closure and runtime.

**list_clock_groups**

This Tcl command lists all existing clock groups in the design.

```
list_clock_groups
```
Arguments
None

Example
list_clock_groups

See Also
set_clock_groups
remove_clock_groups

Specifying I/O States During Programming - I/O States and BSR Details

The I/O States During Programming dialog box enables you to set custom I/O states prior to programming.

I/O State (Output Only)
Sets your I/O states during programming to one of the values shown in the list below.

- 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/Os are tristated

When you set your I/O state, the Boundary Scan Register cells are set according to the table below. Use the Show BSR Details option to set custom states for each cell.

<table>
<thead>
<tr>
<th>Output State</th>
<th>Input</th>
<th>Control (Output Enable)</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last_Known_State</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z (Tri-State)</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0 (Low)</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1 (High)</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table Key:

- 1 – High: I/Os are set to drive out logic High
- 0 – Low: I/Os are set to drive out logic Low
- Last_Known_State - I/Os are set to the last value that was driven out prior to entering the programming mode, and then held at that value during programming

Boundary Scan Registers - Enabled with Show BSR Details
Sets your I/O state to a specific output value during programming AND enables 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 table below).

For example, if you want to Tri-State a pin during programming, set Output Enable to 0; the Don't Care indicates that the other two values are immaterial.

If you want a pin to drive a logic High and have a logic 1 stored in the Input Boundary scan cell during programming, you may set all the values to 1.
Table 7 · BSR Details I/O Output Settings

<table>
<thead>
<tr>
<th>Output State</th>
<th>Settings</th>
<th>Input</th>
<th>Output Enable</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z (Tri-State)</td>
<td>Don't Care</td>
<td>0</td>
<td>Don't Care</td>
<td></td>
</tr>
<tr>
<td>0 (Low)</td>
<td>Don't Care</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1 (High)</td>
<td>Don't Care</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Last Known State</td>
<td>Last State</td>
<td>Last State</td>
<td>Last State</td>
<td></td>
</tr>
</tbody>
</table>

Table Key:
- 1 – High: I/Os are set to drive out logic High
- 0 – Low: I/Os are set to drive out logic Low
- Don’t Care – Don’t Care values have no impact on the other settings.
- Last_Known_State – Sampled value: I/Os are set to the last value that was driven out prior to entering the programming mode, and then held at that value during programming

The figure below shows an example of Boundary Scan Register settings.

Figure 149 · Boundary Scan Registers

Save Project As Dialog Box

The Save Project As dialog box enables you to save your entire project with a new name and location. To access this dialog box, choose **Save Project As** from the **Project menu**.
Figure 150 · Save Project As Dialog Box

**Project Name**
Type the project name for your modified project.

**Project Location**
Accept the default location or **Browse** to the new location where you can save and store your project. All files for your project are saved in this directory.

**Content**
- **Copy links locally** - Select this checkbox to copy the links from your current project into your new project. If you do not select this checkbox, the links will not be copied and you must add them manually.

**Files**
- **All** - Includes all your project and source files; the state of the project is retained.
- **Project files only** - Copies only the project-related information required to retain the state of the project.
- **Source files only** - Copies all the source files into the specified location. This means the configuration of all the tools in the tool chain is retained but the states are not. Source files means constraint information and component information available in the component, hdl and smartgen directories.

Files are saved as shown in the table below.

<table>
<thead>
<tr>
<th>Folder Name</th>
<th>Files</th>
<th>Files</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Project</td>
<td>Source</td>
<td></td>
</tr>
<tr>
<td>All Files</td>
<td>All Files</td>
<td>All Files</td>
<td></td>
</tr>
<tr>
<td>All Files</td>
<td>All Files</td>
<td>All Files</td>
<td></td>
</tr>
<tr>
<td>All Files</td>
<td>All Files</td>
<td>All Files</td>
<td></td>
</tr>
<tr>
<td>All Files</td>
<td>All Files</td>
<td>All Files</td>
<td></td>
</tr>
<tr>
<td>Folder Name</td>
<td>Files</td>
<td>Source</td>
<td>Project</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------</td>
<td>--------------</td>
<td>---------</td>
</tr>
<tr>
<td>stimulus</td>
<td>All Files</td>
<td>All Files</td>
<td></td>
</tr>
<tr>
<td>smartgen</td>
<td>All Files</td>
<td>All Files</td>
<td></td>
</tr>
<tr>
<td>firmware</td>
<td>All Files</td>
<td>All Files</td>
<td></td>
</tr>
<tr>
<td>CoreConsole</td>
<td>All Files</td>
<td>All Files</td>
<td></td>
</tr>
<tr>
<td>SoftConsole/Keil/IAR</td>
<td>All Files</td>
<td>All Files</td>
<td></td>
</tr>
<tr>
<td>Phy_Synthesis</td>
<td>All Files</td>
<td>All Files</td>
<td></td>
</tr>
<tr>
<td>Designer/impl1</td>
<td>All Files</td>
<td>All Files</td>
<td></td>
</tr>
<tr>
<td>Designer/&lt;root&gt;</td>
<td>All Files</td>
<td>All Files</td>
<td></td>
</tr>
<tr>
<td>tooldata</td>
<td>All Files</td>
<td>All Files</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** *.edn files are not supported in PolarFire.

### Project Settings Dialog Box

The Project Settings dialog box enables you to modify your Device, HDL, and Design Flow settings and your Simulation Options. In Libero SoC, from the Project menu, click **Project Settings**.

The following figure shows an example of the Project Settings dialog box.
Device Selection
Sets the device Die and Package for your project. See the New Project Creation Wizard - Device Selection page for a detailed description of the options.

Device Settings
Default I/O Technology - Sets all your I/Os to a default value. You can change the values for individual I/Os in the I/O Attributes Editor.

System controller suspended mode. If the System controller suspended mode is enabled in any PolarFire design, the following operations will not be available during normal operation. For more information, refer to the System Services section in the PolarFire FPGA Security User Guide (UG0753).
- All System controller services requested after power-up is complete and the System controller is suspended
- System controller generated Tamper flags
- Device reset and zeroization Tamper responses
- SPI-Master In-Application Programming (IAP)
- SPI-Slave programming mode
For RT PolarFire devices, the System controller suspended mode is enabled by default for all new RT PolarFire projects. If the System controller suspended mode is disabled, it increases vulnerability to radiation single event effects in the System controller.

Design Flow
See the Project Settings: Design flow topic for more information.

Analysis Operating Conditions
Sets the Operating Temperature Range, the Core Voltage Range, and Default I/O Voltage Range from the picklist's provided. Typical values are COM/IND/MIL; but others are sometimes defined.
Only EXT and IND ranges are available for PolarFire at present.
Once the "Range" value is set, the Minimum/Typical/Maximum values for the selected range are displayed.
These settings are propagated to Verify Timing, Verify Power, and Backannotated Netlist for you to perform Timing/Power Analysis.

Simulation Options and Simulation Libraries
Sets your simulation options. See the Project Settings: Simulation Options topic for more information.
Project Settings: Simulation - Options and Libraries

Using this dialog box, you can set change how Libero SoC handles Do files in simulation, import your own Do files, set simulation run time, and change the DUT name used in your simulation. You can also change your library mapping.

To access this dialog box, from the Project menu choose Project Settings and click to expand Simulation options or Simulation libraries.

For Simulation options click the option you wish to edit: DO file, Waveforms, Vsim commands, Timescale.
For Simulation libraries click on the library you wish to change the path for.

![Figure 152: Project Settings: DO File](image)

**DO file**
- **Use automatic DO file** - Select if you want the Project Manager to automatically create a DO file that will enable you to simulate your design.
- **Simulation Run Time** - Specify how long the simulation should run. If the value is 0, or if the field is empty, there will not be a run command included in the run.do file.
- **Testbench module name** - Specify the name of your testbench entity name. Default is “testbench,” the value used by WaveFormer Pro.
- **Top Level instance name** - Default is <top_0>, the value used by WaveFormer Pro. The Project Manager replaces <top> with the actual top level macro when you run simulation(presynth/postsynth/postlayout).
- **Generate VCD file** - Click the checkbox to generate a VCD file.
- **VCD file name** - Specifies the name of your generated VCD file. The default is power.vcd; click power.vcd and type to change the name.
- **User defined DO file** - Enter the DO file name or click the browse button to navigate to it.
- **DO command parameters** - Text in this field is added to the DO command.

**Waveforms**
- **Include DO file** - Including a DO file enables you to customize the set of signal waveforms that will be displayed in ModelSim.
- **Display waveforms for** - You can display signal waveforms for either the top-level testbench or for the design under test. If you select **top-level testbench** then Project Manager outputs the line 'addwave /testbench/*' in the DO file run.do. If you select **DUT** then Project Manager outputs the line ‘add wave /testbench/DUT/*’ in the run.do file.
- **Log all signals in the design** - Saves and logs all signals during simulation.
Vsim Commands

- **Resolution - The default is 1ps.** Some custom simulation resolutions may not work with your simulation library. Consult your simulation help for more information on how to work with your simulation library and detect infinite zero-delay loops caused by high resolution values.
- **Additional options -** Text entered in this field is added to the vsim command.
  - SRAM ECC Simulation - Two options can be added to specify the simulated error and correction probabilities of all ECC SRAMs in the design.
    - `-gERROR_PROBABILITY=<value>`, where $0 \leq \text{value} \leq 1$
    - `-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 a user runs the Post-Layout Simulation Tool, a run.do file is created, which consists of information which needs to be sent to a simulator. To run a simulation on a corner, a user should select one of the SDF corners along with the type of delay needed from one of the options in SDF timing delays section.
    - **SDF Corners:**
      - Slow Process, Low Voltage and High Temperature
      - Slow Process, Low Voltage and Low Temperature
      - Fast Process, High Voltage and Low Temperature
    - **SDF Timing Delays:**
      - Minimum
      - Typical
      - Maximum
    - Disable pulse filtering during SDF based simulations: This option disables the pulse filtering during SDF simulations.
    - After the user selects the corner, appropriate files needed for simulation are written in the run.do file as shown below.
      ```
      vlog -sv-work postlayout "$(PROJECT_DIR)/designer/sd1/sd1_fast_hv_lt_ba.v"
      vsim -L PolarFire -L postlayout -t 1ps -sdfmax
      /sd1=$(PROJECT_DIR)/designer/sd1/sd1_fast_hv_lt_ba.sdf +pulse_int_e/1
      +pulse_int_r/1 +transport _int_delays postlayout.sd1
      ```
  - To run a simulation on a design having Crypto core, user needs to enter the PLI path in the PLI Path field. User can use the browse button to select the appropriate file. After the file is selected the PLI Path field is populated with the path and file. The user can also manually enter the path in the PLI path field.
    - Errors will be generated in case of wrong path or wrong file extension as follows:
      i. If the user enters a wrong PLI path and clicks on Save
ii. If the user enters a wrong file extension.

Timescale
- **TimeUnit** - Enter a value and select s, ms, us, ns, ps, or fs from the pull-down list, which is the time base for each unit. The default setting is ns.
- **Precision** - Enter a value and select s, ms, us, ns, ps, or fs from the pull-down list. The default setting is ps.

Simulation Libraries
- **Restore Defaults** - Sets the library path to default from your Libero SoC installation.
- **Library path** - Enables you to change the mapping for your simulation library (both Verilog and VHDL). Type the pathname or click the Browse button to navigate to your library directory.
Project Settings: Design flow

To access the Design flow page, from the Project menu choose Project Settings and click the 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. Sets your HDL to VHDL or Verilog. For VHDL, you can choose VHDL-2008 or VHDL-93. For Verilog, you can choose System Verilog (if your Verilog files contain System Verilog constructs) or Verilog 2001.

Note: Libero SoC supports the following Verilog and VHDL IEEE standards for Modelsim and SynplifyPro:

- Verilog 2001 (IEEE Standard 1364-2001)
- SystemVerilog 2012 (IEEE Standard 1800-2012)

Project Settings: General Settings

To access the General Settings page, from the Project menu choose Project Settings and click the General Settings tab.
Figure 154 · Project Settings Dialog Box – General Settings Tab
Absolute/Relative path for linked files

Users can choose to use either "absolute path for linked files" or "relative path for the linked files" for the files linked in the project. The path set in the environment variable is read only.

If the user selects "Use absolute path for linked files" option, all the linked files are stored with the absolute path.

If the user selects "Use relative path for linked files" option, the user is given an option to provide an environment variable that contains the root directory path that will be used as base location for the relative paths. All the files linked will be relative to the base location specified in environment variable by the user.

After the user sets the relative path and provides an environment variable that has a valid path, the following message appears in the log window.

This message can appear in the log window when the user:

- Opens a project that was created using relative path for linked files.
- Creates a project with relative path for linked files.
- Changes the environment variable from the project settings.

The environment variable set by the user is validated before proceeding. If the environment variable set by the user is not present or if it is set after invoking Libero, the tool issues the following error.

Error message: The Environment variable <env_variable_name> specified for the root directory path for linked files is not defined/set in your environment. Please make sure to set the appropriate root directory path in this Environment variable before you open Libero. You will need to exit Libero and open a new session for any changes you made to the directory path set in the Environment variable to reflect here.

Other error messages that can occur are:

- Empty path in Environment variable

Error message: The root directory path for linked files set in the Environment variable <env_variable_name> is empty. Please set a valid path for the root directory in the Environment variable. You will need to exit Libero and open a new session for any changes you made to the directory path set in the Environment variable to reflect here.
• Inaccessible path in environment variable

Error message: The root directory path `<env_variable_directory>` for linked files set in the Environment variable `<env_variable_name>` does not exist. Please set a valid accessible directory path in the Environment variable. You will need to exit Libero and open a new session for any changes you made to the directory path set in the Environment variable to reflect here.

• Project location is set as root directory in environment variable

Error message: The root directory path `<env_variable_path>` for the linked files set in the Environment variable `<env_variable_name>` should not match with the current project directory path. Please set a different directory path in the Environment variable. You will need to exit Libero and open a new session for any changes you made to the directory path set in the Environment variable to reflect here.

• Project settings are saved by selecting relative path and the environment variable is not specified

Error message: Please specify an Environment variable for the root directory path for linked files in the project.

NOTE: Users should set the Environment Variable before invoking Libero. Otherwise, Libero will not be able to get the path set in Environment Variable.

HDL generated files language options

HDL files generated by Libero SoC can be set to use VHDL or Verilog. If there are no other considerations, it is generally recommended to use the same HDL language as you are using for HDL source files, as this may reduce the cost of simulation licenses.

Block flow

Enable block creation - Enables you to create and publish design blocks (*.cxz files) in Libero SoC. Design blocks are low-level components that may have completed the place-and-route step and met the timing and power requirements. These low-level design blocks can then be imported into a Libero SoC project and re-used as components in a higher level design. See Designing with Designer Block Components in Online Help for more information.

Root `<module_name>`

Enable synthesis - Option to enable or disable synthesis for your root file; useful if you wish to skip synthesis on your root file by default.

Enable FPGA Hardware Breakpoint Auto Instantiation - The FHB (FPGA Hardware Breakpoint) 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 Smart Debug UI provide control of the debugging cycle.

**Note:** The default format for Synthesis gate level netlist is Verilog for PolarFire devices.

### Reports

**Maximum number of high fanout nets to be displayed** - Enter the number of high fanout nets to be displayed. The default value is 10. This means the top 10 nets with the highest fanout will appear in the `<root>_compile_netlist_resource.xml>` Report.

### Abort Flow Conditions

**Abort Flow if Errors are found in Physical Design Constraints (PDC)** – Check this checkbox to abort Place and Route if the I/O or Floorplanning PDC constraint file contains errors.

**Abort Flow if Errors are found in Timing Constraints (SDC)** – Check this checkbox to abort Place and Route if the Timing Constraint SDC file contains errors.

**Abort flow if 3.3V I/Os are found in the design** – Check this checkbox to abort Place and Route if 3.3 V I/Os are found in the design. This allows the tool to generate an error as shown below. However, if the checkbox is unchecked, the Place and Route tool will generate the same error message as a warning without interrupting the design flow process.

**Error:** I/Os at 3.3V are vulnerable to Single Event Latch-up at low LET levels. Please review RT PolarFire radiation data at Microchip.com

### Project Settings: Global Include Paths

This setting facilitates the user with an option to provide Global Include paths in Libero and manage how the files are sent to downstream tools. It is the user’s responsibility to provide the paths in order (Include Paths Order).

**Example:**

If a user has a defines file which exists in all the three paths – source file path(default), global_inlcude_path_1 and global_inlcude_path_2; the order of priority of files will be as follows:

- Source file path(default)
- Global_inlcude_path_1
- Global_inlcude_path_2

**Absolute and Relative paths in Global Include Paths**

Users are given an option to specify both relative and absolute files in global include paths.

**Example** -

- Absolute Path: `{ E:/User/GLOBAL_INCLUDE_PATH_1 }`
- Relative Path: `{ ../hdl/GLOBAL_INCLUDE_PATH_1 }`
- Relative Path w.r.t Environment Variable: `{ ${ENV_VAR}/GLOBAL_INCLUDE_PATH_1 }`

Here, the environment variable should be the one used in absolute ad relative paths options only.

**Note:**

1. Changing the Global Include Paths will invalidate the Design Hierarchy but not the Design flow.
2. Change in content of Include Files present in Global Include Paths has impact as in cases below:
a. Case 1: If the Global Include Path is part of the project – Design flow will be invalidated.
b. 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.

Global Include Files
This option facilitates the users to provide global include file/s in Libero and manage how these files are sent to downstream tools occurring in Design Flow process.

Libero GUI facilitates user to provide Global Include Files. On right clicking a file in Design Hierarchy view, user can either set a file as global include by selecting “Set Global Include” option or unset a file as global include by selecting “Unset Global Include” option. Linked files can also be set as Global Include files. The files that are set are highlighted in Design hierarchy view as shown below.

The selected order of global include files is shown in Info tab in log window as below:

```
1 Info: ********************************************
2 Info: ### Global Include File list Order ###
3 Info: File#1 - '${ENV_X}/HDL/xyz_2.h'
4 Info: File#2 - 'E:/sumank_sn/GLOBALINCLUDE_PATH_1/xyz.h'
5 Info: File#3 - 'hdl/xyz.h'
6 Info: ********************************************
```
remove_clock_groups

This Tcl command removes a clock group by name or by ID.

```
remove_clock_groups [-id id# | -name groupname] \n                   [-physically_exclusive | -logically_exclusive | -asynchronous]
```

**Note:** The exclusive flag is not needed when removing a clock group by ID.

**Arguments**

- **-id id#**
  Specifies the clock group by the ID.

- **-name groupname**
  Specifies the clock group by name (to be always followed by the exclusive flag).

  [+physically_exclusive | -logically_exclusive | -asynchronous]

**Example**

Removal by group name

```
remove_clock_groups -name mygroup3 -physically_exclusive
```

Removal by group ID

```
remove_clock_groups -id 12
```

**See Also**

- `set_clock_groups`
- `list_clock_groups`

remove_permanent_locks

Removes all the locks configured in SPM_OTP. This command can only be used when at least one lock is disabled using SPM_OTP.

```
remove_permanent_locks
```

**Example**

```
remove_permanent_locks
```

**See Also**

- `SPM_OTP`
Search in Libero SoC

Search options vary depending on your search type.

**To find a file:**
1. Use CTRL + F to open the Search window.
2. Enter the name or part of name of the object you wish to find in the Find field. ‘*’ indicates a wildcard, and [*-*] indicates a range, such as if you search for a1, a2, ... a5 with the string a[1-5].
3. Set the Options for your search (see below for list); options vary depending on your search type.
4. Click **Find All** (or **Next** if searching Text).

Searching an open text file, Log window or Reports highlights search results in the file itself. All other results appear in the Search Results window (as shown in the figure below).

**Match case:** Select to search for case-sensitive occurrences of a word or phrase. This limits the search so it only locates text that matches the upper- and lowercase characters you enter.

**Match whole word:** Select to match the whole word only.

![Figure 157 · Search Results](image)

**Current Open SmartDesign**
Searches your open SmartDesign, returns results in the Search window.

**Type:** Choose Instance, Net or Pin to narrow your search.

**Query:** Query options vary according to Type.

<table>
<thead>
<tr>
<th>Type</th>
<th>Query Option</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance</td>
<td>Get Pins</td>
<td>Search restricted to all pins</td>
</tr>
<tr>
<td></td>
<td>Get Nets</td>
<td>Search restricted to all nets</td>
</tr>
<tr>
<td></td>
<td>Get Unconnected Pins</td>
<td>Search restricted to all unconnected pins</td>
</tr>
<tr>
<td>Net</td>
<td>Get Instances</td>
<td>Searches all instances</td>
</tr>
<tr>
<td></td>
<td>Get Pins</td>
<td>Search restricted to all pins</td>
</tr>
<tr>
<td>Pin</td>
<td>Get Connected Pins</td>
<td>Search restricted to all connected pins</td>
</tr>
<tr>
<td></td>
<td>Get Associated Net</td>
<td>Search restricted to associated nets</td>
</tr>
<tr>
<td></td>
<td>Get All Unconnected Pins</td>
<td>Search restricted to all unconnected pins</td>
</tr>
</tbody>
</table>

**Current Open Text Editor**
Searches the open text file. If you have more than one text file open you must place the cursor in it and click CTRL + F to search it.

**Find All:** Highlights all finds in the text file.

**Next:** Proceed to next instance of found text.

**Previous:** Proceed to previous instance of found text.

**Replace with:** Replaces the text you searched with the contents of the field.
set_clock_groups

set_clock_groups is an SDC command which disables timing analysis between the specified clock groups. No paths are reported between the clock groups in both directions. Paths between clocks in the same group continue to be reported.

```
set_clock_groups [-name name]
[-physically_exclusive | -logically_exclusive | -asynchronous]
[-comment comment_string]
-group clock_list
```

**Note:** If you use the same name and the same exclusive flag of a previously defined clock group to create a new clock group, the previous clock group is removed and a new one is created in its place.
Arguments

- **name** name

   Name given to the clock group. Optional.

- **physically_exclusive**

   Specifies that the clock groups are physically exclusive with respect to each other. Examples are multiple clocks feeding a register clock pin. The exclusive flags are all mutually exclusive. Only one can be specified.

- **logically_exclusive**

   Specifies that the clocks groups are logically exclusive with respect to each other. Examples are clocks passing through a mux.

- **asynchronous**

   Specifies that the clock groups are asynchronous with respect to each other, as there is no phase relationship between them. The exclusive flags are all mutually exclusive. Only one can be specified.

Note: The exclusive flags for the arguments above are all mutually exclusive. Only one can be specified.

- **group clock_list**

   Specifies a list of clocks. There can any number of groups specified in the set_clock_groups command.

Examples

1) Here, there are two muxed clocks in a clock group.

   ![Diagram of two muxed clocks]

   SDC:
   ```
   create_clock -name clk_1 -period 5 [ get_ports clk_1 ]
   create_clock -name clk_2 -period 10 [ get_ports clk_2 ]
   set_clock_groups -logically_exclusive -group clk_1 -group clk_2
   ```

2) Here, there are three synchronous clocks receiving data from an asynchronous clock.

   ![Diagram of three synchronous clocks with an asynchronous clock]

   SDC:
   ```
   create_clock -name clk_in -period 10 [ get_ports clk_in ]
   create_clock -name clk_50 -period 20 [ get_ports clk_50 ]
   create_generated_clock -name ccc_100 -divide_by 2
   -source [ get_pins ccc_100_0/ccc_100_0/pll_inst_0/REF_CLK_0 ]
   [ get_pins ccc_100_0/ccc_100_0/pll_inst_0/OUT0 ]
   create_generated_clock -name clk_out -divide_by 1
   -source [ get_pins { ccc_100_0/ccc_100_0/pll_inst_0/OUT0 } ]
   ```
set_clock_groups -asynchronous -group { clk_in ccc_100 clk_out } -group clk_50

See Also
list_clock_groups
remove_clock_groups

set_auto_update_mode

This command enables or disables auto update.

```
set_auto_update_mode {0|1}
```

If `set_auto_update_mode` is 0, auto update is disabled. If `set_auto_update_mode` is 1, auto update is enabled.

set_plain_text_client

This Tcl command is added to the sNVM .cfg file that is given as the parameter to the configure_snvm command. Plain-text Non-Authenticated clients have 252 bytes available for user data in each page of sNVM.

```
set_plain_text_client
-client_name {<name>}
-number_of_bytes <number>
-content_type {MEMORY_FILE | STATIC_FILL}
-content_file_format {Microsemi-Binary 8/16/32 bit}
-content_file {<path>}
-start_page <number>
-use_for_simulation 0
-reprogram 0 | 1
-use_as_rom 0 | 1
```

Arguments

-clients_name
The name of the client. Needs to start with an alphabetic letter. Underscores and numerals are allowed at all positions other than the first.
-number_of_bytes
The size of the client specified in bytes.
-content_type
Source of data for the client. This can either be a memory file, or all zeros. Allowed values are MEMORY_FILE or STATIC_FILL
-content_file_format
Only ‘Microsemi-Binary 8/16/32 bit’ is supported at this time.
-content_file
Path of the memory file. This can be absolute, or relative to the project.
-start_page
The page number in sNVM where data for this client will be placed.
-use_for_simulation
Only value 0 is allowed.
-reprogram
Boolean field; specifies whether the client will be programmed into the final design or not. Possible values are 0 or 1.
-use_as_rom
0
Boolean field; specifies whether the client will allow only reads, or both read and writes. Possible values are 0 or 1.

**Example**
```
set_plain_text_client \
    -client_name {a} \
    -number_of_bytes 12 \
    -content_type {MEMORY_FILE} \
    -content_file_format {Microsemi-Binary 8/16/32 bit} \
    -content_file {D:/local_z_folder/work/memory_files/binary8x12.mem} \
    -start_page 1 \
    -use_for_simulation 0 \
    -reprogram 1 \
    -use_as_rom 0
```

**See Also**
- `set_plain_text_auth_client`
- `set_cipher_text_auth_client`
- `set_usk_client`

### `set_plain_text_auth_client`

This Tcl command is added to the sNVM .cfg file that is given as the parameter to the configure_snvm command. Plain-text Authenticated clients have 236 bytes available for user data in each page of sNVM.

```
set_plain_text_auth_client \
    -client_name {<name>}
```

**Arguments**
- `-number_of_bytes <number>`
- `-content_type {MEMORY_FILE | STATIC_FILL}`
- `-content_file_format {Microsemi-Binary 8/16/32 bit}`
- `-content_file {<path>}`
- `-start_page <number>`
- `-use_for_simulation 0`
- `-reprogram 0 | 1`
- `-use_as_rom 0 | 1`

- `-client_name`
  The name of the client. Needs to start with an alphabetic letter. Underscores and numerals are allowed at all positions other than the first.
- `-number_of_bytes`
  The size of the client specified in bytes.
- `-content_type`
  Source of data for the client. This can either be a memory file, or all zeros. Allowed values are MEMORY_FILE or STATIC_FILL
- `-content_file_format`
  Only `Microsemi-Binary 8/16/32 bit` is supported at this time.
- `-content_file`
  Path of the memory file. This can be absolute, or relative to the project.
- `-start_page`
  The page number in sNVM where data for this client will be placed.
- `-use_for_simulation`
  Only value 0 is allowed.
- `-reprogram`
  Boolean field; specifies whether the client will be programmed into the final design or not. Possible values are 0 or 1.
- `-use_as_rom`
Boolean field; specifies whether the client will allow only reads, or both read and writes. Possible values are 0 or 1.

PolarFire

Example

```tcl
set_cipher_text_auth_client \
-client_name {b} \
-number_of_bytes 12 \
-content_type {MEMORY_FILE} \
-content_file_format {Microsemi-Binary 8/16/32 bit} \
-content_file {D:/local_z_folder/work/memory_files/binary8x12.mem} \
-start_page 2 \
-use_for_simulation 0 \
-reprogram 1 \
-use_as_rom 0
```

See Also

- `set_cipher_text_auth_client`
- `set_plain_text_auth_client`
- `set_usk_client`

**set_cipher_text_auth_client**

This Tcl command is added to the sNVM .cfg file that is given as the parameter to the configure_snvm command.

Cipher-text Authenticated clients have 236 bytes available for user data in each page of sNVM.

```tcl
set_cipher_text_auth_client \
-client_name {<name>} \
-number_of_bytes {<number>} \
-content_type {MEMORY_FILE | STATIC_FILL} \
-content_file_format {Microsemi-Binary 8/16/32 bit} \
-content_file {<path>} \
-start_page {<number>} \
-use_for_simulation 0 \
-reprogram 0 | 1 \
-use_as_rom 0 | 1
```

**Arguments**

- `-client_name`

  The name of the client. Needs to start with an alphabetic letter. Underscores and numerals are allowed at all positions other than the first.

- `-number_of_bytes`

  The size of the client specified in bytes.

- `-content_type`

  Source of data for the client. This can either be a memory file, or all zeros. Allowed values are MEMORY_FILE or STATIC_FILL.

- `-content_file_format`

  Only 'Microsemi-Binary 8/16/32 bit' is supported at this time.

- `-content_file`

  Path of the memory file. This can be absolute, or relative to the project.

- `-start_page`

  The page number in sNVM where data for this client will be placed.

- `-use_for_simulation`

  Only value 0 is allowed.

- `-reprogram`

  Boolean field; specifies whether the client will be programmed into the final design or not. Possible values are 0 or 1.
set_clock_uncertainty

Tcl command; specifies simple clock uncertainty for single clock and clock-to-clock uncertainty between two clocks (from and to).

```
set_clock_uncertainty [ -setup ] [ -hold ] uncertainty [ object_list -from from_clock | -rise_from rise_from_clock | -fall_from fall_from_clock -to to_clock | -rise_to rise_to_clock | -fall_to fall_to_clock ]
```

- **-setup**: Specifies the setup uncertainty.
- **-hold**: Specifies the hold uncertainty.
- **uncertainty**: The magnitude of the uncertainty.
- **object_list**: The list of objects to which the uncertainty applies.
- **from_clock**: The clock from which the uncertainty is specified.
- **rise_from_clock**: The rise transition from the specified clock.
- **fall_from_clock**: The fall transition from the specified clock.
- **to_clock**: The clock to which the uncertainty is specified.
- **rise_to_clock**: The rise transition to the specified clock.
- **fall_to_clock**: The fall transition to the specified clock.

Example

```
set_clock_uncertainty -setup 5 -hold 4
```

**See Also**

- `set_config_file`
- `set_config_file_auth_client`
- `set_cipher_text_auth_client`
- `set_cipher_text_auth_client`
- `set_cipher_text_auth_client`
Arguments

- **uncertainty**
  Specifies the time in nanoseconds that represents the amount of variation between two clock edges.
- **object_list**
  Specifies a list of clocks, ports, or pins for simple uncertainty; the uncertainty is applied either to
destination flops clocked by one of the clocks in the object list option, or destination flops whose clock
pins are in the fanout of a port or a pin specified in the object_list option.
- **-from**
  Specifies that the clock-to-clock uncertainty applies to both rising and falling edges of the source clock
list. Only one of the -from, -rise_from, or -fall_from arguments can be specified for the constraint to be
valid.
- **-rise_from**
  Specifies that the clock-to-clock uncertainty applies only to rising edges of the source clock list. Only one
of the -from, -rise_from, or -fall_from arguments can be specified for the constraint to be valid.
- **-fall_from**
  Specifies that the clock-to-clock uncertainty applies only to falling edges of the source clock list. Only one
of the -from, -rise_from, or -fall_from arguments can be specified for the constraint to be valid.
- **from_clock/rise_from_clock/fall_from_clock**
  Specifies the list of clock names as the uncertainty source.
- **-to**
  Specifies that the clock-to-clock uncertainty applies to both rising and falling edges of the destination clock
list. Only one of the -to, -rise_to, or -fall_to arguments can be specified for the constraint to be valid.
- **-rise_to**
  Specifies that the clock-to-clock uncertainty applies only to rising edges of the destination clock list. Only one
of the -to, -rise_to, or -fall_to arguments can be specified for the constraint to be valid.
- **-fall_to**
  Specifies that the clock-to-clock uncertainty applies only to falling edges of the destination clock list. Only one
of the -to, -rise_to, or -fall_to arguments can be specified for the constraint to be valid.
- **to_clock/rise_to_clock/fall_to_clock**
  Specifies the list of clock names as the uncertainty destination.
- **-setup**
  Specifies that the uncertainty applies only to setup checks. If none or both -setup and -hold are present,
the uncertainty applies to both setup and hold checks.
- **-hold**
  Specifies that the uncertainty applies only to hold checks. If none or both -setup and -hold are present,
the uncertainty applies to both setup and hold checks.

Description

**set_clock_uncertainty** command sets the timing uncertainty of clock networks. It can be used to model clock jitter
or add guard band in timing analysis. Either simple clock uncertainty or clock-to-clock uncertainty can be
specified.

Simple clock uncertainty can be set on a clock or on any pin in the clock network. It will then apply to any path
with the capturing register in the forward cone of the uncertainty. If multiple simple uncertainty applies to a
register, the last one (in the propagation order from the clock source to the register) is used.

Clock-to-clock uncertainty applies to inter-clock paths. Both “from” clock and “to” clock must be specified. Clock-
to-clock uncertainty has higher priority than simple uncertainty. If both are set (a clock-to-clock uncertainty and a
simple clock uncertainty on the “to” clock), the simple clock uncertainty will be ignored for inter-clock paths, only
the clock-to-clock uncertainty will be used.
Examples

Simple Clock Uncertainty constraint examples:
```plaintext
set_clock_uncertainty 2 -setup [get_clocks clk]
set_clock_uncertainty 2 [get_clocks clk]
```

Clock to Clock Uncertainty constraint examples:
```plaintext
set_clock_uncertainty 10 -from Clk1 -to Clk2
set_clock_uncertainty 0 -from Clk1 -fall_to { Clk2 Clk3 } -setup
set_clock_uncertainty 4.3 -fall_from { Clk1 Clk2 } -rise_to *
set_clock_uncertainty 0.1 -rise_from [ get_clocks { Clk1 Clk2 } ] -fall_to { Clk3 Clk4 } -setup
set_clock_uncertainty 5 -rise_from Clk1 -to [ get_clocks {*} ]
```

Organize Source Files Dialog Box – Synthesis

The Organize Source Files dialog box enables you to set the source file order in the Libero SoC. Click the Use list of files organized by User radio button to Add/Remove source files for the selected tool.

To specify the file order:

1. In the Design Flow window under Implement Design, right-click Synthesize and choose Organize Input Files > Organize Source Files. The Organize Source Files dialog box appears.
2. Click the Use list of files organized by User radio button to Add/Remove source files for the selected tool.
3. Select a file and click the Add or Remove buttons as necessary. Use the Up and Down arrows to change the order of the Associated Source files.
4. Click OK.

![Organize Source Files Dialog Box](image)

SmartDesign Testbench

SmartDesign Testbench is a GUI-based tool that enables you to design your testbench hierarchy. Use SmartDesign Testbench to instantiate and connect stimulus cores or modules to drive your design. You can create a SmartDesign Testbench by right-clicking a SmartDesign component in the Design Hierarchy and choosing Create Testbench > SmartDesign.

SmartDesign Testbench automatically instantiates the selected SmartDesign component into the Canvas.
You can also double-click Create SmartDesign Testbench in the Design Flow window to add a new SmartDesign testbench to your project.

New testbench files appear in the Stimulus Hierarchy.

SmartDesign Testbench automatically instantiates your SmartDesign component into the Canvas.
You can instantiate your own stimulus HDL or simulation models into the SmartDesign Testbench Canvas and connect them to your DUT (design under test). You can also instantiate Simulation Cores from the Catalog. Simulation cores are simulation models (such as DDR memory simulation models) or basic cores that are useful for stimulus generation (such as Clock Generator, Pulse Generator, or Reset Generator).

Click the Simulation Mode checkbox in the Catalog to view available simulation cores.

Refer to the SmartDesign User Guide for more information.
Specify I/O States During Programming Dialog Box

The I/O States During Programming dialog box enables 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.

Load from file
Load from file enables you to load an I/O Settings (*.ios) file. You can use the IOS file to import saved custom settings for all your I/Os. The exported IOS file have the following format:

- Used I/Os have an entry in the IOS file with the following format:
  ```
  set_prog_io_state -portName {<design_port_name>} -input <value> -outputEnable <value> -output <value>
  ```
- Unused I/Os have an entry in the IOS file with the following format:
  ```
  set_prog_io_state -pinNumber {<device_pinNumber>} -input <value> -outputEnable <value> -output <value>
  ```

Where <value> is:
- 1 – I/O is set to drive out logicHigh
- 0 – I/O is set to drive out logicLow
- Last_Known_State: I/O is set to the last value that was driven out prior to entering the programming mode, and then held at that value during programming
- Z - Tri-State: I/O is tristated

Save to file
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 in conjunction with a PDC file.

Port Name
Lists the names of all the ports in your design.

Macro Cell
Lists the I/O type, such as INBUF, OUTBUF, PLLs, etc.

Pin Number
The package pin associate with the I/O.

I/O State (Output Only)
Your custom I/O State set during programming. This heading changes to Boundary Scan Register if you select the BSR Details checkbox; see the Specifying I/O States During Programming - I/O States and BSR Details help topic for more information on the BSR Details option.
The I/O States During Programming dialog box enables you to set custom I/O states prior to programming.

**I/O State (Output Only)**

Sets your I/O states during programming to one of the values shown in the list below.

- **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/Os are tristated

When you set your I/O state, the Boundary Scan Register cells are set according to the table below. Use the Show BSR Details option to set custom states for each cell.

**Table 8 · Default I/O Output Settings**

<table>
<thead>
<tr>
<th>Output State</th>
<th>Settings</th>
<th>Input</th>
<th>Control (Output Enable)</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z (Tri-State)</td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0 (Low)</td>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1 (High)</td>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Last_Known_State</td>
<td></td>
<td></td>
<td>Last_Known_State</td>
<td>Last_Known_State</td>
</tr>
</tbody>
</table>
Table Key:
- 1 – High: I/Os are set to drive out logic High
- 0 – Low: I/Os are set to drive out logic Low
- Last_Known_State - I/Os are set to the last value that was driven out prior to entering the programming mode, and then held at that value during programming

**Boundary Scan Registers - Enabled with Show BSR Details**
Sets your I/O state to a specific output value during programming AND enables 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 table below).
For example, if you want to Tri-State a pin during programming, set Output Enable to 0; the Don't Care indicates that the other two values are immaterial.
If you want a pin to drive a logic High and have a logic 1 stored in the Input Boundary scan cell during programming, you may set all the values to 1.

<table>
<thead>
<tr>
<th>Output State</th>
<th>Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Input</td>
</tr>
<tr>
<td>Z (Tri-State)</td>
<td>Don't Care</td>
</tr>
<tr>
<td>0 (Low)</td>
<td>Don't Care</td>
</tr>
<tr>
<td>1 (High)</td>
<td>Don't Care</td>
</tr>
<tr>
<td>Last Known State</td>
<td>Last State</td>
</tr>
</tbody>
</table>

Table Key:
- 1 – High: I/Os are set to drive out logic High
- 0 – Low: I/Os are set to drive out logic Low
- Don't Care – Don't Care values have no impact on the other settings.
- Last_Known_State – Sampled value: I/Os are set to the last value that was driven out prior to entering the programming mode, and then held at that value during programming

The figure below shows an example of Boundary Scan Register settings.
Stimulus Hierarchy

To view the Stimulus Hierarchy, from the View menu choose Windows > Stimulus Hierarchy. The Stimulus Hierarchy tab displays a hierarchical representation of the stimulus and simulation files in the project. The software continuously analyzes and updates files and content. The tab (see figure below) displays the structure of the modules and component stimulus files as they relate to each other.
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:**

Double-click a stimulus file to open it in the HDL text editor.

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.

Icons in the Hierarchy indicate the type of component and the state, as shown in the table below.

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="SmartDesign component" /></td>
<td>SmartDesign component</td>
</tr>
<tr>
<td><img src="image" alt="SmartDesign component with HDL netlist not generated" /></td>
<td>SmartDesign component with HDL netlist not generated</td>
</tr>
<tr>
<td><img src="image" alt="SmartDesign testbench" /></td>
<td>SmartDesign testbench</td>
</tr>
<tr>
<td><img src="image" alt="SmartDesign testbench with HDL netlist not generated" /></td>
<td>SmartDesign testbench with HDL netlist not generated</td>
</tr>
<tr>
<td><img src="image" alt="IP core was instantiated into SmartDesign but the HDL netlist has not been generated" /></td>
<td>IP core was instantiated into SmartDesign but the HDL netlist has not been generated</td>
</tr>
<tr>
<td><img src="image" alt="HDL netlist" /></td>
<td>HDL netlist</td>
</tr>
</tbody>
</table>

**Timing Exceptions Overview**

Use timing exceptions to overwrite the default behavior of the design path. Timing exceptions include:

- Setting multicycle constraint to specify paths that (by design) will take more than one cycle.
- Setting a false path constraint to identify paths that must not be included in the timing analysis or the optimization flow.
- Setting a maximum/minimum delay constraint on specific paths to relax or to tighten the original clock constraint requirement.

**Admin Profile Tool Dialog Box**

This allows the users a way to use same set of tools and same vault location if Libero SoC is invoked from the same installed location. An admin (the person who generally takes care of Libero SoC installation) creates an admin profile. This profile will be common to all the users who are using Libero SoC from the same installed location. Individual users do not need to have to create the tool profile and vault location.

- To create an admin profile, user needs to invoke adminProfile.exe from Libero/Designer/bin location.
This admin profile will be common to all users who are using Libero SoC from the installed path, by providing the following values:

- Synplify location
- Modelsim location
- Vault location
- Any other tool mentioned in admin profile utility

**Note:** It is not mandatory for the user to create admin profile. If the user skips the creation of admin profile, the tool will continue to work normally.

If the user wishes to create admin profile, all the fields in the dialog box must entered correctly. The vault location provided during admin profile creation will have precedence and will be used by all the other users.

**Vaults and tools location**

Vault location: Location of megavault consisting of all the cores for the current Libero SoC version must be mentioned.

Tools location: Correct tool executable location must be entered which is accessible by all users of Libero SoC.

Using admin profile, users will also have an option to use Libero SoC provided tools (which are a part of Libero SoC Design Suite) by selecting the **Use the tool from Libero SoC Design Suite installation** option. In this case, location of specific tools will be used from Libero SoC installation.

**Tool Profiles Dialog Box**

The Tool Profiles dialog box enables you to add, edit, or delete your project tool profiles.

Each Libero SoC project can have a different profile, enabling you to integrate different tools with different projects.

The following table shows the supported tool versions in this release.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Supported Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modelsim ME</td>
<td>10.5c</td>
</tr>
<tr>
<td>Modelsim ME Pro</td>
<td>10.5c</td>
</tr>
<tr>
<td>Synplify Pro ME</td>
<td>N-2017.09M SP1-1</td>
</tr>
<tr>
<td>Identify ME</td>
<td>N-2017.09M SP1</td>
</tr>
</tbody>
</table>
To set or change your tool profile:

1. From the Project menu, choose Tool Profiles. Select the type of tool you wish to add.
   - To add a tool: Select the tool type and click the Add button. Fill out the tool profile and click OK.
   - To change a tool profile: After selecting the tool, click the Edit button to select another tool, change the tool name, or change the tool location.
   - To remove a tool from the project: After selecting a tool, click the Remove button.

2. When you are done, click OK.

Figure 162 · Libero SoC Tool Profiles Dialog Box

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.

User Preferences Dialog Box – Design Flow Preferences

This dialog box allows you to set your personal preferences for how Libero SoC manages the design flow across the projects you create.
Figure 163 · Preferences Dialog Box – Design Flow Preferences

Constraint Flow

- **Warn me when derived timing constraints generation override existing constraints (enhanced constraint flow).**
  Libero SoC can generate/derive timing constraints for known hardware blocks and IPs such as SERDES, CCC. Check this box to have Libero SoC pop up a warning message when the generated timing constraints for these blocks override the timing constraints you set for these blocks. This box is checked by default.

Design Flow Rule Checks

- **Warn me when Firmware applications must be recompiled because of hardware configuration changes.**
  Check this box if you want Libero SoC to display a warning message. This box is checked by default.

- **Warn me when I/Os are not all assigned and locked before programming data generation.**
  I/Os should always be assigned and locked before programming data generation. Check this box if you want Libero SoC to display a warning message. This box is checked by default.

SmartDesign Generation Options

- **Generate recursively**
  In this mode, all subdesigns must be successfully generated before a parent can be generated. An attempt to generate a SmartDesign results in an automatic attempt to generate all subdesigns.

- **Generate non-recursively**
  In this mode, the generation of only explicitly selected SmartDesigns is attempted. The generation of a design can be marked as successful even if a subdesign is ungenerated (either never attempted or unsuccessful).

**Note:** These preferences are stored on a per-user basis across multiple projects; they are not project-specific.
Generic/Parameter Report:

Libero tool generates hierarchical reports that show parameter/generic values used in instantiation of the modules. Reports are generated for the active top module. Report will be generated for only that active top module which is not instantiated in any other modules. The report is generated in xml format.

In below example, report will not be generated for the active root(top) module COREABC_C0 because it is instantiated in sd1. The file sd1 needs to be set as a root to generate the report.

![Diagram showing design flow hierarchy]

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.

![Log window showing successful export]

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. Please 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"
IP Report:

Libero tool generates a report listing all the IPs used in the design project.

The report consists of the following:

- Project Information
- List of components in the project with the below information:
  a. Component name
  b. Component State
  c. IP Core (Vendor, Library, Name, Version) used to create the component
  d. TCL Parameters of the core used to create the component
- Information of core instances used in the Smart Design.
  a. Component Name – In the form of “smartdesign_name>_<instance_name>
  b. Component State – State of the instance in the Smart Design
  c. Vendor, Library, Name, Version of the core used to create the instance
  d. TCL Parameters of the core used to create the instance
- Single Page System Builder components
- Multi Page System Builder components without parameter list of latest core versions available for the cores used in the project. This is the list that is present at the end of open project.

In order to export the **IP CORE Usage Report**, go to File -> Export -> IP Core Usage Report. When clicked, a file dialog box launched asking for the location and file format to dump the report. The file can be generated in either “JSON” or “txt” format.

Once the report is generated successfully, the location to where the report is exported is shown in log window.
Force Update Design Flow

A global option to force up-to-date the design flow to pass state has been added to Libero SoC v12.5. The tools state is force updated to PASS state only for Synthesize/Compile and Place and Route tools. The remaining tools will be in the same state as they were before performing force update action.

![Force Update Action Button at Top of Design Flow Window](image1)

Figure 164 · Force Update Action Button at Top of Design Flow Window

To differentiate between PASS state and FORCE UPDATE PASS state, a new icon for the FORCE UPDATE PASS state (force update action) has been added along with tool tip as shown in image below.

![Force Update Action Icon with Tool Tip](image2)

Figure 165 · Force Update Action Icon with Tool Tip

When user clicks Force update action, a warning dialog box pops up before proceeding with actual action. If the user selects OK option, the action proceeds else the action is aborted. Below is the warning message which will appear always when the Force Update Action is performed. Figure: Warning message pop-up

![Warning](image3)

The message log window will show the below warning in all cases when the Force Update Action is performed.

![Warning message in log window](image4)

Figure 166: Warning message in log window
On running programming tool – Generate FPGA Array Data, a warning message pops up if either of Synthesize or Place and Route or both are in Force Update Pass state.

The log window shows the following message.

**Synopsys Design Constraints (SDC)**

Synopsys Design Constraints (SDC) is a Tcl-based format used by Synopsys tools to specify the design intent, including the timing and area constraints for a design. Microsemi tools use a subset of the SDC format to capture supported timing constraints. Any timing constraint that you can enter using Designer tools can also be specified in an SDC file.

Use the SDC-based flow to share timing constraint information between Microsemi tools and third-party EDA tools.

<table>
<thead>
<tr>
<th>Command</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>create_clock</td>
<td>Creates a clock and defines its characteristics</td>
</tr>
<tr>
<td>create_generated_clock</td>
<td>Creates an internally generated clock and defines its characteristics</td>
</tr>
<tr>
<td>set_clock_latency</td>
<td>Defines the delay between an external clock source and the definition pin of a clock within SmartTime</td>
</tr>
<tr>
<td>set_clock_uncertainty</td>
<td>Defines the timing uncertainty between two clock waveforms or maximum skew</td>
</tr>
<tr>
<td>set_false_path</td>
<td>Identifies paths that are to be considered false and excluded from the timing analysis</td>
</tr>
<tr>
<td>set_input_delay</td>
<td>Defines the arrival time of an input relative to a clock</td>
</tr>
<tr>
<td>set_max_delay</td>
<td>Specifies the maximum delay for the timing paths</td>
</tr>
<tr>
<td>set_min_delay</td>
<td>Specifies the minimum delay for the timing paths</td>
</tr>
<tr>
<td>Command</td>
<td>Action</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>set_multicycle_path</td>
<td>Defines a path that takes multiple clock cycles</td>
</tr>
<tr>
<td>set_output_delay</td>
<td>Defines the output delay of an output relative to a clock</td>
</tr>
</tbody>
</table>

**See Also**

*SDC Syntax Conventions*

**libero_design_flow_SDC_commands**

**SDC Syntax Conventions**

The following table shows the typographical conventions that are used for the SDC command syntax.

<table>
<thead>
<tr>
<th>Syntax Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>command -argument</td>
<td>Commands and arguments appear in <strong>Courier New</strong> typeface.</td>
</tr>
<tr>
<td>variable</td>
<td>Variables appear in blue, italic <strong>Courier New</strong> typeface. You must substitute an appropriate value for the variable.</td>
</tr>
<tr>
<td>[-argument value]</td>
<td>Optional arguments begin and end with a square bracket.</td>
</tr>
</tbody>
</table>

**Note:** SDC commands and arguments are case sensitive.

**Example**

The following example shows syntax for the `create_clock` command and a sample command:

```
create_clock -period period_value [-waveform edge_list] source
```

```
create_clock -period 7 -waveform {2 4}{CLK1}
```

**Wildcard Characters**

You can use the following wildcard characters in names used in the SDC commands:

<table>
<thead>
<tr>
<th>Wildcard</th>
<th>What it does</th>
</tr>
</thead>
<tbody>
<tr>
<td>\</td>
<td>Interprets the next character literally</td>
</tr>
<tr>
<td>*</td>
<td>Matches any string</td>
</tr>
</tbody>
</table>

**Note:** The matching function requires that you add a backslash (\) before each slash in the pin names in case the slash does not denote the hierarchy in your design.

**Special Characters ([ ], { }, and \)**

Square brackets ([ ]) are part of the command syntax to access ports, pins and clocks. In cases where these netlist objects names themselves contain square brackets (for example, buses), you must either enclose the names with curly brackets ({}) or precede the open and closed square brackets ([ ]) characters with a backslash (\). If you do not do this, the tool displays an error message.

For example:

```
create_clock -period 3 clk\[0\]
set_max_delay 1.5 -from [get_pins ff1\[5\]:CLK] -to [get_clocks {clk[0]}]
```
Although not necessary, Microsemi recommends the use of curly brackets around the names, as shown in the following example:

```
set_false_path –from {data1} –to \[get_pins {reg1:D}\]
```

In any case, the use of the curly bracket is mandatory when you have to provide more than one name.

For example:

```
set_false_path –from {data3 data4} –to \[get_pins {reg2:D reg5:D}\]
```

### Entering Arguments on Separate Lines

If a command needs to be split on multiple lines, each line except the last must end with a backslash (\) character as shown in the following example:

```
set_multicycle_path 2 -from \[get_pins {reg1*}\] \to {reg2:D}
```

### create_clock

**SDC command; creates a clock and defines its characteristics.**

```
create_clock -name clock_name -add -period period_value [-waveform edge_list] source
```

**Arguments**

- **-name clock_name**
  Specifies the name of the clock constraint. This parameter is required for virtual clocks when no clock source is provided.

- **-add**
  Specifies that a new clock constraint is created at the same source as the existing clock without overriding the existing constraint. The name of the new clock constraint with the -add option must be different than the existing clock constraint. Otherwise, it will override the existing constraint, even with the -add option. The -name option must be specified with the -add option.

- **-period period_value**
  Specifies the clock period in nanoseconds. The value you specify is the minimum time over which the clock waveform repeats. The period_value must be greater than zero.

- **-waveform edge_list**
  Specifies the rise and fall times of the clock waveform in ns over a complete clock period. There must be exactly two transitions in the list, a rising transition followed by a falling transition. You can define a clock starting with a falling edge by providing an edge list where fall time is less than rise time. If you do not specify -waveform option, the tool creates a default waveform, with a rising edge at instant 0.0 ns and a falling edge at instant (period_value/2)ns.

- **source**
  Specifies the source of the clock constraint. The source can be ports or pins in the design. If you specify a clock constraint on a pin that already has a clock, the new clock replaces the existing one. Only one source is accepted. Wildcards are accepted as long as the resolution shows one port or pin.

**Description**

Creates a clock in the current design at the declared source and defines its period and waveform. The static timing analysis tool uses this information to propagate the waveform across the clock network to the clock pins of all sequential elements driven by this clock source.

The clock information is also used to compute the slacks in the specified clock domain that drive optimization tools such as place-and-route.

**Exceptions**

None

Arguments

- **-name clock_name**
  Specifies the name of the clock constraint. This parameter is required for virtual clocks when no clock source is provided.

- **-add**
  Specifies that the generated clock constraint is a new clock constraint in addition to the existing one at the same source. The name of the clock constraint should be different from the existing clock constraint. With this option, -master_clock option and -name options must be specified.

- **-master_clock clock_name**
  Specifies the master clock used for the generated clock when multiple clocks fan into the master pin. This option must be used with -add option of the generated clock.

Examples

The following example creates two clocks, one on port CK1 with a period of 6, and the other on port CK2 with a period of 6, a rising edge at 0, and a falling edge at 3:

```plaintext
create_clock -name {my_user_clock} -period 6 CK1
create_clock -name {my_other_user_clock} -period 6 -waveform {0 3} {CK2}
```

The following example creates a clock on port CK3 with a period of 7, a rising edge at 2, and a falling edge at 4:

```plaintext
create_clock -period 7 -waveform {2 4} [get_ports {CK3}]
```

The following example creates a new clock constraint clk2, in addition to clk1, on the same source port clk1 without overriding it.

```plaintext
create_clock -name clk1 -period 10 -waveform {0 5} [get_ports clk1]
create_clock -name clk2 -add -period 20 -waveform {0 10} [get_ports clk1]
```

The following example does not add a new clock constraint, even with the -add option, but overrides the existing clock constraint because of the same clock names. **Note:** To add a new clock constraint in addition to the existing clock constraint on the same source port, the clock names must be different.

```plaintext
create_clock -name clk1 -period 10 -waveform {0 5} [get_ports clk1]
create_clock -name clk1 -add -period 50 -waveform {0 25} [get_ports clk1]
```

Microsemi Implementation Specifics

- The -waveform in SDC accepts waveforms with multiple edges within a period. In Microsemi design implementation, only two waveforms are accepted.

- SDC accepts defining a clock on many sources using a single command. In Microsemi design implementation, only one source is accepted.

- The source argument in SDC create_clock command is optional. This is in conjunction with the -name argument in SDC to support the concept of virtual clocks. In Microsemi implementation, source is a mandatory argument as -name and virtual clocks concept is not supported.

- The -domain argument in the SDC create_clock command is not supported.

See Also

SDC Syntax Conventions

create_generated_clock

SDC command; creates an internally generated clock and defines its characteristics.
Notes:

1. The master_clock option is used only with the -add option for the generated clocks.
2. If there are multiple master clocks fanning into the same reference pin, the first generated clock specified will always use the first master clock as its source clock.
3. The subsequent generated clocks specified with the -add option can choose any of the master clocks as their source clock (including the first master clock specified).

- **source reference_pin**
  Specifies the reference pin in the design from which the clock waveform is to be derived.

- **divide_by divide_factor**
  Specifies the frequency division factor. For instance if the divide_factor is equal to 2, the generated clock period is twice the reference clock period.

- **multiply_by multiply_factor**
  Specifies the frequency multiplication factor. For instance if the multiply_factor is equal to 2, the generated clock period is half the reference clock period.

- **invert**
  Specifies that the generated clock waveform is inverted with respect to the reference clock.

- **source**
  Specifies the source of the clock constraint on internal pins of the design. If you specify a clock constraint on a pin that already has a clock, the new clock replaces the existing clock. Only one source is accepted. Wildcards are accepted as long as the resolution shows one pin.

- **pll_output pll_feedback_clock**
  Specifies the output pin of the PLL which is used as the external feedback clock. This pin must drive the feedback input pin of the PLL specified using the –pll_feedback option. The PLL will align the rising edge of the reference input clock to the feedback clock. This is a mandatory argument if the PLL is operating in external feedback mode.

- **pll_feedback pll_feedback_input**
  Specifies the feedback input pin of the PLL. This pin must be driven by the output pin of the PLL specified using the –pll_output option. The PLL will align the rising edge of the reference input clock to the external feedback clock. This is a mandatory argument if the PLL is operating in external feedback mode.

- **edges values**
  Specify the integer values that represent the edges from the source clock that form the edges of the generated clock. Three values must be specified to generate the clock. If you specify less than three, a tool tip indicates an error.

- **edge_shift values**
  Specify a list of three floating point numbers that represents the amount of shift, in nanoseconds, that the specified edges are to undergo to yield the final generated clock waveform. These floating point values can be positive or negative. Positive value indicates a shift later in time, while negative indicates a shift earlier in time.

  For example: An edge shift of \{1 1 1\} on the LSB generated clock, would shift each derived edge by 1 nanosecond.

  To create a 200MHz clock from a 100MHz clock, use edge \{1 2 3\} and edge shift \{0 -2.5 -5.0\}.

Description

Creates a generated clock in the current design at a declared source by defining its frequency with respect to the frequency at the reference pin. The static timing analysis tool uses this information to compute and propagate its waveform across the clock network to the clock pins of all sequential elements driven by this source.

The generated clock information is also used to compute the slacks in the specified clock domain that drive optimization tools such as place-and-route.
Examples

The following example creates a generated clock on pin U1/reg1:Q with a period twice as long as the period at the reference port CLK.

```shell
create_generated_clock -name {my_user_clock} -divide_by 2 -source [get_ports {CLK}] U1/reg1:Q
```

The following example creates a generated clock at the primary output of myPLL with a period ¾ of the period at the reference pin clk.

```shell
create_generated_clock -divide_by 3 -multiply_by 4 -source clk [get_pins {myPLL:CLK1}]
```

The following example creates a new generated clock gen2 in addition to gen1 derived from same master clock as the existing generated clock, and the new constraints is added to pin r1/CLK.

```shell
create_generated_clock -name gen1 -multiply_by 1 -source [get_ports clk1] [get_pins r1/CLK]
create_generated_clock -name gen2 -add -master_clock clk1 -source [get_ports clk1] -multiply_by 2 [get_pins r1/CLK]
```

The following example does not create a new generated clock constraint in addition to the existing clock, but will override even with the -add option enabled, because the same names are used.

```shell
create_generated_clock -name gen2 -source [get_ports clk1] -multiply_by 3 [get_pins r1/CLK]
create_generated_clock -name gen2 -source [get_ports clk1] -multiply_by 4 -master_clock clk1 -add [get_pins r1/CLK]
```

The following example creates a generated clock on pin U1/reg1:Q with a period twice as long as the period at the reference port CLK.

```shell
create_generated_clock -name {my_user_clock} -divide_by 2 -source [get_ports {CLK}] U1/reg1:Q
```

The following example creates a generated clock at the primary output of myPLL with a period ¾ of the period at the reference pin clk.

```shell
create_generated_clock -divide_by 3 -multiply_by 4 -source clk [get_pins {myPLL/CLK1}]
```

The following example creates a generated clock named system_clk on the GL2 output pin of FCCC_0 with a period equal to half the period of the source clock. The constraint also identifies GL2 output pin as the external feedback clock source and CLK2 as the feedback input pin for FCCC_0.

```shell
create_generated_clock -name { system_clk } -multiply_by 2 -source { FCCC_0/CCC_INST/CLK3_PAD } -pll_output { FCCC_0/CCC_INST/GL2 } -pll_feedback { FCCC_0/CCC_INST/CLK2 } { FCCC_0/CCC_INST/GL2 }
```

Microsemi Implementation Specifics

- SDC accepts either --multiply_by or -divide_by option. In Microsemi design implementation, both are accepted to accurately model the PLL behavior.
- SDC accepts defining a generated clock on many sources using a single command. In Microsemi design implementation, only one source is accepted.
- The -duty_cycle, -edges and --edge_shift options in the SDC create_generated_clock command are not supported in Microsemi design implementation.

See Also

SDC Syntax Conventions
remove_clock_uncertainty

SDC command; Removes a clock-to-clock uncertainty from the current timing scenario.

```
remove_clock_uncertainty -from | -rise_from | -fall_from from_clock_list -to | -rise_to| -fall_to to_clock_list -setup {value} -hold {value}
remove_clock_uncertainty -id constraint_ID
```

**Arguments**

- **from**
  Specifies that the clock-to-clock uncertainty applies to both rising and falling edges of the source clock list. You can specify only one of the `-from`, `-rise_from`, or `-fall_from` arguments for the constraint to be valid.

- **rise_from**
  Specifies that the clock-to-clock uncertainty applies only to rising edges of the source clock list. You can specify only one of the `-from`, `-rise_from`, or `-fall_from` arguments for the constraint to be valid.

- **fall_from**
  Specifies that the clock-to-clock uncertainty applies only to falling edges of the source clock list. You can specify only one of the `-from`, `-rise_from`, or `-fall_from` arguments for the constraint to be valid.

  `from_clock_list`
  Specifies the list of clock names as the uncertainty source.

- **to**
  Specifies that the clock-to-clock uncertainty applies to both rising and falling edges of the destination clock list. You can specify only one of the `-to`, `-rise_to`, or `-fall_to` arguments for the constraint to be valid.

- **rise_to**
  Specifies that the clock-to-clock uncertainty applies only to rising edges of the destination clock list. You can specify only one of the `-to`, `-rise_to`, or `-fall_to` arguments for the constraint to be valid.

- **fall_to**
  Specifies that the clock-to-clock uncertainty applies only to falling edges of the destination clock list. You can specify only one of the `-to`, `-rise_to`, or `-fall_to` arguments for the constraint to be valid.

  `to_clock_list`
  Specifies the list of clock names as the uncertainty destination.

- **setup**
  Specifies that the uncertainty applies only to setup checks. If none or both `-setup` and `-hold` are present, the uncertainty applies to both setup and hold checks.

- **hold**
  Specifies that the uncertainty applies only to hold checks. If none or both `-setup` and `-hold` are present, the uncertainty applies to both setup and hold checks.

- **id constraint_ID**
  Specifies the ID of the clock constraint to remove from the current scenario. You must specify either the exact parameters to set the constraint or its constraint ID.

**Description**

Removes a clock-to-clock uncertainty from the specified clock in the current scenario. If the specified arguments do not match clocks with an uncertainty constraint in the current scenario, or if the specified ID does not refer to a clock-to-clock uncertainty constraint, this command fails.

Do not specify both the exact arguments and the ID.

**Exceptions**

None
Examples

remove_clock_uncertainty -from Clk1 -to Clk2
remove_clock_uncertainty -from Clk1 -fall_to { Clk2 Clk3 } -setup
remove_clock_uncertainty 4.3 -fall_from { Clk1 Clk2 } -rise_to *
remove_clock_uncertainty 0.1 -rise_from { get_clocks { Clk1 Clk2 } } -fall_to { Clk3 Clk4 } -setup
remove_clock_uncertainty 5 -rise_from Clk1 -to [ get_clocks {*} ]
remove_clock_uncertainty -id $clockId

See Also
SDC Syntax Conventions
set_clock_uncertainty

set_clock_latency

SDC command; defines the delay between an external clock source and the definition pin of a clock within SmartTime.

set_clock_latency -source [-rise][-fall][-early][-late] delay clock

Arguments

-source
Specifies a clock source latency on a clock pin.

-rise
Specifies the edge for which this constraint will apply. If neither or both rise are passed, the same latency is applied to both edges.

-fall
Specifies the edge for which this constraint will apply. If neither or both rise are passed, the same latency is applied to both edges.

-invert
Specifies that the generated clock waveform is inverted with respect to the reference clock.

-late
Optional. Specifies that the latency is late bound on the latency. The appropriate bound is used to provide the most pessimistic timing scenario. However, if the value of "-late" is less than the value of "-early", optimistic timing takes place which could result in incorrect analysis. If neither or both "-early" and "-late" are provided, the same latency is used for both bounds, which results in the latency having no effect for single clock domain setup and hold checks.

-early
Optional. Specifies that the latency is early bound on the latency. The appropriate bound is used to provide the most pessimistic timing scenario. However, if the value of "-late" is less than the value of "-early", optimistic timing takes place which could result in incorrect analysis. If neither or both "-early" and "-late" are provided, the same latency is used for both bounds, which results in the latency having no effect for single clock domain setup and hold checks.

-delay
Specifies the latency value for the constraint.

-clock
Specifies the clock to which the constraint is applied. This clock must be constrained.

Description
Clock source latency defines the delay between an external clock source and the definition pin of a clock within SmartTime. It behaves much like an input delay constraint. You can specify both an "early" delay and a "late" delay for this latency, providing an uncertainty which SmartTime propagates through its calculations. Rising and falling edges of the same clock can have different latencies. If only one value is provided for the clock source latency, it is taken as the exact latency value, for both rising and falling edges.
 Exceptions
None

 Examples
The following example sets an early clock source latency of 0.4 on the rising edge of main_clock. It also sets a clock source latency of 1.2, for both the early and late values of the falling edge of main_clock. The late value for the clock source latency for the falling edge of main_clock remains undefined.

```
set_clock_latency -source -rise -early 0.4 { main_clock }
set_clock_latency -source -fall 1.2 { main_clock }
```

 Microsemi Implementation Specifics
SDC accepts a list of clocks to -set_clock_latency. In Microsemi design implementation, only one clock pin can have its source latency specified per command.

 See Also
SDC Syntax Conventions

 set_clock_to_output

SDC command; defines the timing budget available inside the FPGA for an output relative to a clock.

```
set_clock_to_output delay_value -clock clock_ref [-max] [-min] output_list
```

 Arguments
 delay_value
Specifies the clock to output delay in nanoseconds. This time represents the amount of time available inside the FPGA between the active clock edge and the data change at the output port.

- clock clock_ref
Specifies the reference clock to which the specified clock to output is related. This is a mandatory argument.

- max
Specifies that delay_value refers to the maximum clock to output at the specified output. If you do not specify -max or -min options, the tool assumes maximum and minimum clock to output delays to be equal.

- min
Specifies that delay_value refers to the minimum clock to output at the specified output. If you do not specify -max or -min options, the tool assumes maximum and minimum clock to output delays to be equal.

 output_list
Provides a list of output ports in the current design to which delay_value is assigned. If you need to specify more than one object, enclose the objects in braces ({}).

 set_clock_uncertainty

SDC command; specifies simple clock uncertainty for single clock and clock-to-clock uncertainty between two clocks (from and to).

```
set_clock_uncertainty [-setup] [-hold] uncertainty [object_list -from from_clock | -rise_from rise_from_clock | -fall_from fall_from_clock -to to_clock | -rise_to rise_to_clock | -fall_to fall_to_clock]
```

 Arguments
 uncertainty
Specifies the time in nanoseconds that represents the amount of variation between two clock edges.

 object_list
Specifies a list of clocks, ports, or pins for simple uncertainty; the uncertainty is applied either to
destination flops clocked by one of the clocks in the object list option, or destination flops whose clock pins are in the fanout of a port or a pin specified in the object_list option.

-from

Specifies that the clock-to-clock uncertainty applies to both rising and falling edges of the source clock list. Only one of the -from, -rise_from, or -fall_from arguments can be specified for the constraint to be valid.

-rise_from

Specifies that the clock-to-clock uncertainty applies only to rising edges of the source clock list. Only one of the -from, -rise_from, or -fall_from arguments can be specified for the constraint to be valid.

-fall_from

Specifies that the clock-to-clock uncertainty applies only to falling edges of the source clock list. Only one of the -from, -rise_from, or -fall_from arguments can be specified for the constraint to be valid.

from_clock/rise_from_clock/fall_from_clock

Specifies the list of clock names as the uncertainty source.

to

Specifies that the clock-to-clock uncertainty applies to both rising and falling edges of the destination clock list. Only one of the -to, -rise_to, or -fall_to arguments can be specified for the constraint to be valid.

-rise_to

Specifies that the clock-to-clock uncertainty applies only to rising edges of the destination clock list. Only one of the -to, -rise_to, or -fall_to arguments can be specified for the constraint to be valid.

-fall_to

Specifies that the clock-to-clock uncertainty applies only to falling edges of the destination clock list. Only one of the -to, -rise_to, or -fall_to arguments can be specified for the constraint to be valid.

to_clock/rise_to_clock/fall_to_clock

Specifies the list of clock names as the uncertainty destination.

-setup

Specifies that the uncertainty applies only to setup checks. If none or both -setup and -hold are present, the uncertainty applies to both setup and hold checks.

-hold

Specifies that the uncertainty applies only to hold checks. If none or both -setup and -hold are present, the uncertainty applies to both setup and hold checks.

Description

set_clock_uncertainty command sets the timing uncertainty of clock networks. It can be used to model clock jitter or add guard band in timing analysis. Either simple clock uncertainty or clock-to-clock uncertainty can be specified.

Simple clock uncertainty can be set on a clock or on any pin in the clock network. It will then apply to any path with the capturing register in the forward cone of the uncertainty. If multiple simple uncertainty applies to a register, the last one (in the propagation order from the clock source to the register) is used.

Clock-to-clock uncertainty applies to inter-clock paths. Both “from” clock and “to” clock must be specified. Clock-to-clock uncertainty has higher priority than simple uncertainty. If both are set (a clock-to-clock uncertainty and a simple clock uncertainty on the “to” clock), the simple clock uncertainty will be ignored for inter-clock paths, only the clock-to-clock uncertainty will be used.

 Exceptions

None
Examples

Simple Clock Uncertainty constraint examples:
```plaintext
set_clock_uncertainty 2 -setup [get_clocks clk]
set_clock_uncertainty 2 [get_clocks clk]
```
Clock to Clock Uncertainty constraint examples:
```plaintext
set_clock_uncertainty 10 -from Clk1 -to Clk2
set_clock_uncertainty 0 -from Clk1 -fall_to { Clk2 Clk3 } -setup
set_clock_uncertainty 4.3 -fall_from { Clk1 Clk2 } -rise_to *
set_clock_uncertainty 0.1 -rise_from [ get_clocks { Clk1 Clk2 } ] -fall_to { Clk3 Clk4 } -setup
set_clock_uncertainty 5 -rise_from Clk1 -to [ get_clocks {*} ]
```

Microsemi Implementation Specifics

- SDC accepts a list of clocks to `-set_clock_uncertainty`

See Also

- SDC Syntax Conventions

set_disable_timing

SDC command; disables timing arcs within the specified cell and returns the ID of the created constraint if the command succeeded.

```plaintext
set_disable_timing [-from from_port] [-to to_port] cell_name
```

Arguments

- `-from from_port`
  Specifies the starting port.
- `-to to_port`
  Specifies the ending port.
- `cell_name`
  Specifies the name of the cell in which timing arcs will be disabled.

Description

This command disables the timing arcs in the specified cell, and returns the ID of the created constraint if the command succeeded. The –from and –to arguments must either both be present or both omitted for the constraint to be valid.

Examples

The following example disables the arc between a2:A and a2:Y.
```
set_disable_timing -from port1 -to port2 cellname
```
This command ensures that the arc is disabled within a cell instead of between cells.

Microsemi Implementation Specifics

- None

See Also

- SDC Syntax Conventions
set_external_check

SDC command; defines the external setup and hold delays for an input relative to a clock.

```
set_external_check delay_value -clock clock_ref [-setup] [-hold] input_list
```

Arguments

- `delay_value`
  Specifies the external setup or external hold delay in nanoseconds. This time represents the amount of time available inside the FPGA for the specified input after a clock edge.

- `clock clock_ref`
  Specifies the reference clock to which the specified external check is related. This is a mandatory argument.

- `-setup` or `-hold`
  Specifies that `delay_value` refers to the setup/hold check at the specified input. This is a mandatory argument if `-hold` is not used. You must specify either `-setup` or `-hold` option.

- `input_list`
  Provides a list of input ports in the current design to which `delay_value` is assigned. If you need to specify more than one object, enclose the objects in braces `{}`.

Description

The `set_external_check` command specifies the external setup and hold times on input ports relative to a clock edge. This usually represents a combinational path delay from the input port to the clock pin of a register internal to the current design. For in/out (bidirectional) ports, you can specify the path delays for both input and output modes. The tool uses external setup and external hold times for paths starting at primary inputs.

A clock is a singleton that represents the name of a defined clock constraint. This can be an object accessor that will refer to one clock. For example:

```
[get_clocks {system_clk}]
[get_clocks {sys*_clk}]
```

Examples

The following example sets an external setup check of 12 ns and an external hold check of 6 ns for port `data_in` relative to the rising edge of `CLK1`:

```
set_external_check 12 -clock [get_clocks CLK1] -setup [get_ports data_in]
set_external_check 6 -clock [get_clocks CLK1] -hold [get_ports data_in]
```

See Also

- SDC Syntax Conventions

set_false_path

SDC command; identifies paths that are considered false and excluded from the timing analysis.

```
set_false_path [-from from_list] [-through through_list] [-to to_list]
```

Arguments

- `-from from_list`
  Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

- `-through through_list`
  Specifies a list of pins, ports, cells, or nets through which the disabled paths must pass.

- `-to to_list`
  Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.
Description
The `set_false_path` command identifies specific timing paths as being false. The false timing paths are paths that do not propagate logic level changes. This constraint removes timing requirements on these false paths so that they are not considered during the timing analysis. The path starting points are the input ports or register clock pins, and the path ending points are the register data pins or output ports. This constraint disables setup and hold checking for the specified paths.

The false path information always takes precedence over multiple cycle path information and overrides maximum delay constraints. If more than one object is specified within one `-through` option, the path can pass through any objects.

Examples
The following example specifies all paths from clock pins of the registers in clock domain clk1 to data pins of a specific register in clock domain clk2 as false paths:

```
set_false_path -from [get_clocks {clk1}] -to reg_2:D
```

The following example specifies all paths through the pin U0/U1:Y to be false:

```
set_false_path -through U0/U1:Y
```

Microsemi Implementation Specifics
SDC accepts multiple `-through` options in a single constraint to specify paths that traverse multiple points in the design. In Microsemi design implementation, only one `-through` option is accepted.

See Also
SDC Syntax Conventions

---

**set_input_delay**

SDC command; defines the arrival time of an input relative to a clock.

```
```

Arguments

- `delay_value`
  Specifies the arrival time in nanoseconds that represents the amount of time for which the signal is available at the specified input after a clock edge.

- `-clock clock_ref`
  Specifies the clock reference to which the specified input delay is related. This is a mandatory argument. If you do not specify `-max` or `-min` options, the tool assumes the maximum and minimum input delays to be equal.

- `-max`
  Specifies that `delay_value` refers to the longest path arriving at the specified input. If you do not specify `-max` or `-min` options, the tool assumes maximum and minimum input delays to be equal.

- `-min`
  Specifies that `delay_value` refers to the shortest path arriving at the specified input. If you do not specify `-max` or `-min` options, the tool assumes maximum and minimum input delays to be equal.

- `-clock_fall`
  Specifies that the delay is relative to the falling edge of the clock reference. The default is the rising edge.

- `-rise`
  Specifies that the delay is relative to a rising transition on the specified port(s). If `-rise` or `-fall` is not specified, then rising and falling delays are assumed to be equal.

- `-fall`
  Specifies that the delay is relative to a falling transition on the specified port(s). If `-rise` or `-fall` is not specified, then rising and falling delays are assumed to be equal.
-add_delay

Specifies that this input delay constraint should be added to an existing constraint on the same port(s). The -add_delay option is used to capture information on multiple paths with different clocks or clock edges leading to the same input port(s).

input_list

Provides a list of input ports in the current design to which delay_value is assigned. If you need to specify more than one object, enclose the objects in braces ({}).

Notes:

- The behavior of the -add_delay option is identical to that of PrimeTime(TM)
- If, using the -add_delay mechanism, multiple constraints are otherwise identical, except they specify different -max or -min values
  - the surviving -max constraint will be the maximum of the -max values
  - the surviving -min constraint will be the minimum of the -min values

Description

The set_input_delay command sets input path delays on input ports relative to a clock edge. This usually represents a combinational path delay from the clock pin of a register external to the current design. For in/out (bidirectional) ports, you can specify the path delays for both input and output modes. The tool adds input delay to path delay for paths starting at primary inputs.

A clock is a singleton that represents the name of a defined clock constraint. This can be:

- a single port name used as source for a clock constraint
- a single pin name used as source for a clock constraint; for instance reg1:CLK. This name can be hierarchical (for instance toplevel/block1/reg2:CLK)
- an object accessor that will refer to one clock: [get_clocks {clk}]

Examples

The following example sets an input delay of 1.2ns for port data1 relative to the rising edge of CLK1:
```plaintext
set_input_delay 1.2 -clock [get_clocks CLK1] [get_ports data1]
```

The following example sets a different maximum and minimum input delay for port IN1 relative to the falling edge of CLK2:
```plaintext
set_input_delay 1.0 -clock_fall -clock CLK2 -min {IN1}
set_input_delay 1.4 -clock_fall -clock CLK2 -max {IN1}
```

The following example demonstrates an override condition of two constraints. The first constraint is overridden because the second constraint specifies a different clock for the same output:
```plaintext
set_input_delay 1.0 -clock CLK1 -max {IN1}
set_input_delay 1.4 -clock CLK2 -max {IN1}
```

The next example is almost the same as the previous one, however, in this case, the user has specified -add_delay, so both constraints will be honored:
```plaintext
set_input_delay 1.0 -clock CLK1 -max {IN1}
set_input_delay 1.4 -add_delay -clock CLK2 -max {IN1}
```

The following example is more complex:

- All constraints are for an input to port PAD1 relative to a rising edge clock CLK2. Each combination of {-rise, -fall} x {-max, -min} generates an independent constraint. But the max rise delay of 5 and the max rise delay of 7 interfere with each other.
- For a -max option, the maximum value overrides all lower values. Thus the first constraint will be overridden and the max rise delay of 7 will survive.
```plaintext
set_input_delay 5 -max -rise -add_delay [get_clocks CLK2] [get_ports PAD1] # will be overridden
set_input_delay 3 -min -fall -add_delay [get_clocks CLK2] [get_ports PAD1]
set_input_delay 3 -max -fall -add_delay [get_clocks CLK2] [get_ports PAD1]
set_input_delay 7 -max -rise -add_delay [get_clocks CLK2] [get_ports PAD1]
```
Microsemi Implementation Specifics

In SDC, the -clock is an optional argument that allows you to set input delay for combinational designs. Microsemi’s implementation currently requires this argument.

See Also

SDC Syntax Conventions

set_max_delay (SDC)

SDC command; specifies the maximum delay for the timing paths.

```
set_max_delay delay_value [-from from_list] [-to to_list]
```

Arguments

delay_value

Specifies a floating point number in nanoseconds that represents the required maximum delay value for specified paths.

- If the path starting point is on a sequential device, the tool includes clock skew in the computed delay.
- If the path starting point has an input delay specified, the tool adds that delay value to the path delay.
- If the path ending point is on a sequential device, the tool includes clock skew and library setup time in the computed delay.
- If the ending point has an output delay specified, the tool adds that delay to the path delay.

-from from_list

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

-to to_list

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

Description

This command specifies the required maximum delay for timing paths in the current design. The path length for any startpoint in from_list to any endpoint in to_list must be less than delay_value.

The tool automatically derives the individual maximum delay targets from clock waveforms and port input or output delays. For more information, refer to the create_clock, set_input_delay, and set_output_delay commands.

The maximum delay constraint is a timing exception. This constraint overrides the default single cycle timing relationship for one or more timing paths. This constraint also overrides a multicycle path constraint.

Examples

The following example sets a maximum delay by constraining all paths from ff1a:CLK or ff1b:CLK to ff2e:D with a delay less than 5 ns:

```
set_max_delay 5 -from {ff1a:CLK ff1b:CLK} -to {ff2e:D}
```

The following example sets a maximum delay by constraining all paths to output ports whose names start by “out” with a delay less than 3.8 ns:

```
set_max_delay 3.8 -to [get_ports out*]
```

Microsemi Implementation Specifics

The –through option in the set_max_delay SDC command is not supported.

See Also

SDC Syntax Conventions
set_min_delay

SDC command; specifies the minimum delay for the timing paths.

```
set_min_delay delay_value [-from from_list] [-to to_list]
```

Arguments

delay_value
Specifies a floating point number in nanoseconds that represents the required minimum delay value for specified paths.

- If the path starting point is on a sequential device, the tool includes clock skew in the computed delay.
- If the path starting point has an input delay specified, the tool adds that delay value to the path delay.
- If the path ending point is on a sequential device, the tool includes clock skew and library setup time in the computed delay.
- If the ending point has an output delay specified, the tool adds that delay to the path delay.

-from from_list
Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

-to to_list
Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

Description

This command specifies the required minimum delay for timing paths in the current design. The path length for any startpoint in from_list to any endpoint in to_list must be less than delay_value.

The tool automatically derives the individual minimum delay targets from clock waveforms and port input or output delays. For more information, refer to the create_clock, set_input_delay, and set_output_delay commands.

The minimum delay constraint is a timing exception. This constraint overrides the default single cycle timing relationship for one or more timing paths. This constraint also overrides a multicycle path constraint.

Examples

The following example sets a minimum delay by constraining all paths from ff1a:CLK or ff1b:CLK to ff2e:D with a delay less than 5 ns:

```
set_min_delay 5 -from {ff1a:CLK ff1b:CLK} -to {ff2e:D}
```

The following example sets a minimum delay by constraining all paths to output ports whose names start by "out" with a delay less than 3.8 ns:

```
set_min_delay 3.8 -to [get_ports out*]
```

Microsemi Implementation Specifics

The –through option in the set_min_delay SDC command is not supported.

See Also

SDC Syntax Conventions
set_multicycle_path

SDC command; defines a path that takes multiple clock cycles.

```
set_multicycle_path ncycles [-setup] [-hold] [setup_only] [-from from_list] [-through through_list] [-to to_list]
```

**Arguments**

- `ncycles` Specifies an integer value that represents a number of cycles the data path must have for setup or hold check. The value is relative to the starting point or ending point clock, before data is required at the ending point.
- `-setup` Optional. Applies the cycle value for the setup check only. This option does not affect the hold check. The default hold check will be applied unless you have specified another set_multicycle_path command for the hold value.
- `-hold` Optional. Applies the cycle value for the hold check only. This option does not affect the setup check.

**Note:** If you do not specify `-setup` or `-hold`, the cycle value is applied to the setup check and the default hold check is performed (`ncycles -1`).
- `-setup_only` Optional. Specifies that the path multiplier is applied to setup paths only. The default value for hold check (which is 0) is applied.
- `-from from_list` Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.
- `-through through_list` Specifies a list of pins or ports through which the multiple cycle paths must pass.
- `-to to_list` Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

**Description**

Setting multiple cycle paths constraint overrides the single cycle timing relationships between sequential elements by specifying the number of cycles that the data path must have for setup or hold checks. If you change the multiplier, it affects both the setup and hold checks.

False path information always takes precedence over multiple cycle path information. A specific maximum delay constraint overrides a general multiple cycle path constraint.

If you specify more than one object within one `-through` option, the path passes through any of the objects.

**Examples**

The following example sets all paths between reg1 and reg2 to 3 cycles for setup check. Hold check is measured at the previous edge of the clock at reg2.

```
set_multicycle_path 3 -from [get_pins {reg1}] -to [get_pins {reg2}]
```

The following example specifies that four cycles are needed for setup check on all paths starting at the registers in the clock domain `ck1`. Hold check is further specified with two cycles instead of the three cycles that would have been applied otherwise.

```
set_multicycle_path 4 -setup -from [get_clocks {ck1}]
set_multicycle_path 2 -hold -from [get_clocks {ck1}]
```

The following example specifies that four cycles are needed for setup only check on all paths starting at the registers in the clock domain `REF_CLK_0`.

```
set_multicycle_path -setup_only 4 -from [get_clocks {REF_CLK_0}]
```
Microsemi Implementation Specifics

- SDC allows multiple priority management on the multiple cycle path constraint depending on the scope of the object accessors. In Microsemi design implementation, such priority management is not supported. All multiple cycle path constraints are handled with the same priority.

See Also
SDC Syntax Conventions

set_output_delay

SDC command; defines the output delay of an output relative to a clock.

```
```

Arguments

- `delay_value`
  Specifies the amount of time before a clock edge for which the signal is required. This represents a combinational path delay to a register outside the current design plus the library setup time (for maximum output delay) or hold time (for minimum output delay).

- `clock clock_ref`
  Specifies the clock reference to which the specified output delay is related. This is a mandatory argument. If you do not specify -max or -min options, the tool assumes the maximum and minimum input delays to be equal.

- `-max`
  Specifies that delay_value refers to the longest path from the specified output. If you do not specify -max or -min options, the tool assumes the maximum and minimum output delays to be equal.

- `-min`
  Specifies that delay_value refers to the shortest path from the specified output. If you do not specify -max or -min options, the tool assumes the maximum and minimum output delays to be equal.

- `-clock_fall`
  Specifies that the delay is relative to the falling edge of the clock reference. The default is the rising edge.

- `-rise`
  Specifies that the delay is relative to a rising transition on the specified port(s). If -rise or -fall is not specified, then rising and falling delays are assumed to be equal.

- `-fall`
  Specifies that the delay is relative to a falling transition on the specified port(s). If -rise or -fall is not specified, then rising and falling delays are assumed to be equal.

- `-add_delay`
  Specifies that this output delay constraint should be added to an existing constraint on the same port(s). The -add_delay option is used to capture information on multiple paths with different clocks or clock edges leading from the same output port(s).

- `output_list`
  Provides a list of output ports in the current design to which delay_value is assigned. If you need to specify more than one object, enclose the objects in braces ({}).

Notes:

- The behavior of the -add_delay option is identical to that of PrimeTime(TM)
- If, using the -add_delay mechanism, multiple constraints are otherwise identical, except they specify different -max or -min values
  - the surviving -max constraint will be the maximum of the -max values
  - the surviving -min constraint will be the minimum of the -min values
Description
The set_output_delay command sets output path delays on output ports relative to a clock edge. Output ports have no output delay unless you specify it. For in/out (bidirectional) ports, you can specify the path delays for both input and output modes. The tool adds output delay to path delay for paths ending at primary outputs.

Examples
The following example sets an output delay of 1.2ns for port OUT1 relative to the rising edge of CLK1:

```
set_output_delay 1.2 -clock [get_clocks CLK1] [get_ports OUT1]
```

The following example sets a different maximum and minimum output delay for port OUT1 relative to the falling edge of CLK2:

```
set_output_delay 1.0 -clock_fall -clock CLK2 -min {OUT1}
set_output_delay 1.4 -clock_fall -clock CLK2 -max {OUT1}
```

The following example demonstrates an override condition of two constraints. The first constraint is overridden because the second constraint specifies a different clock for the same output:

```
set_output_delay 1.0 (OUT1) -clock CLK1 -max
set_output_delay 1.4 (OUT1) -clock CLK2 -max
```

The next example is almost the same as the previous one, however, in this case, the user has specified -add_delay, so both constraints will be honored:

```
set_output_delay 1.0 (OUT1) -clock CLK1 -max
set_output_delay 1.4 (OUT1) -add_delay -clock CLK2 -max
```

The following example is more complex:

- All constraints are for an output to port PAD1 relative to a rising edge clock CLK2. Each combination of {-rise, -fall} x {-max, -min} generates an independent constraint. But the max rise delay of 5 and the max rise delay of 7 interfere with each other.
- For a -max option, the maximum value overrides all lower values. Thus the first constraint will be overridden and the max rise delay of 7 will survive.

```
set_output_delay 5 [get_clocks CLK2] [get_ports PAD1] -max -rise -add_delay # will be overridden
set_output_delay 3 [get_clocks CLK2] [get_ports PAD1] -min -fall -add_delay
set_output_delay 3 [get_clocks CLK2] [get_ports PAD1] -max -fall -add_delay
set_output_delay 7 [get_clocks CLK2] [get_ports PAD1] -max -rise -add_delay
```

Microsemi Implementation Specifics
- In SDC, the -clock is an optional argument that allows you to set the output delay for combinational designs. Microsemi Implementation currently requires this option.

See Also
SDC Syntax Conventions

Design Object Access Commands
Design object access commands are SDC commands. Most SDC constraint commands require one of these commands as command arguments.

Microsemi software supports the following SDC access commands:

<table>
<thead>
<tr>
<th>Design Object</th>
<th>Access Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell</td>
<td>get_cells</td>
</tr>
<tr>
<td>Clock</td>
<td>get_clocks</td>
</tr>
<tr>
<td>Net</td>
<td>get_nets</td>
</tr>
<tr>
<td>Port</td>
<td>get_ports</td>
</tr>
<tr>
<td>Pin</td>
<td>get_pins</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Input ports</td>
<td>all_inputs</td>
</tr>
<tr>
<td>Output ports</td>
<td>all_outputs</td>
</tr>
<tr>
<td>Registers</td>
<td>all_registers</td>
</tr>
</tbody>
</table>

**See Also**

*About SDC Files*

**all_inputs**

*Design object access command:* returns all the input or inout ports of the design.

Arguments

- None

Exceptions

- None

Example

```bsh
set_max_delay -from [all_inputs] -to [get_clocks ck1]
```

**Microsemi Implementation Specifics**

- None

See Also

*SDC Syntax Conventions*

**all_outputs**

*Design object access command:* returns all the output or inout ports of the design.

Arguments

- None

Exceptions

- None

Example

```bsh
set_max_delay -from [all_inputs] -to [all_outputs]
```
all_registers

**Design object access command**: returns either a collection of register cells or register pins, whichever you specify.

```plaintext
all_registers [-clock clock_name] [-cells] [-data_pins ]
[-clock_pins] [-async_pins] [-output_pins]
```

### Arguments

- **-clock clock_name**
  
  Creates a collection of register cells or register pins in the specified clock domain.

- **-cells**
  
  Creates a collection of register cells. This is the default. This option cannot be used in combination with any other option.

- **-data_pins**
  
  Creates a collection of register data pins.

- **-clock_pins**
  
  Creates a collection of register clock pins.

- **-async_pins**
  
  Creates a collection of register asynchronous pins.

- **-output_pins**
  
  Creates a collection of register output pins.

### Description

This command creates either a collection of register cells (default) or register pins, whichever is specified. If you do not specify an option, this command creates a collection of register cells.

### Exceptions

- None

### Examples

```plaintext
set_max_delay 2 -from [all_registers] -to [get_ports {out}]
set_max_delay 3 -to [all_registers -async_pins]
set_false_path -from [all_registers -clock clk150]
set_multicycle_path -to [all_registers -clock c* -data_pins
- clock_pins]
```

### Microsemi Implementation Specifics

- None

**See Also**

- SDC Syntax Conventions

**get_cells**

**Design object access command**: returns the cells (instances) specified by the pattern argument.
get_cells pattern

Arguments

**pattern**

Specifies the pattern to match the instances to return. For example, "get_cells U18" returns all instances starting with the characters "U18", where "*" is a wildcard that represents any character string.

Description

This command returns a collection of instances matching the pattern you specify. You can only use this command as part of a –from, -to, or –through argument for the following constraint exceptions: set_max delay, set_multicycle_path, and set_false_path design constraints.

Exceptions

None

Examples

set_max_delay 2 -from [get_cells {reg*}] -to [get_ports {out}]
set_false_path -through [get_cells {Rblock/muxA}]

Microsemi Implementation Specifics

- None

See Also

SDC Syntax Conventions

get_clocks

Design object access command; returns the specified clock.

get_clocks pattern

Arguments

**pattern**

Specifies the pattern to match to the SmartTime on which a clock constraint has been set.

Description

- If this command is used as a –from argument in maximum delay (set_max_path_delay), false path (set_false_path), and multicycle constraints (set_multicycle_path), the clock pins of all the registers related to this clock are used as path start points.
- If this command is used as a –to argument in maximum delay (set_max_path_delay), false path (set_false_path), and multicycle constraints (set_multicycle_path), the synchronous pins of all the registers related to this clock are used as path endpoints.

Exceptions

- None
set_max_delay -from [get_ports data1] -to \[get_clocks ck1]

Microsemi Implementation Specifics

None

See Also
SDC Syntax Conventions

get_pins

Design object access command; returns the specified pins.

get_pins pattern

Arguments

pattern

Specifies the pattern to match the pins.

Exceptions

None

Example

create_clock -period 10 [get_pins clock_gen/reg2:Q]

Microsemi Implementation Specifics

• None

See Also
SDC Syntax Conventions

get_nets

Design object access command; returns the named nets specified by the pattern argument.

get_nets pattern

Arguments

pattern

Specifies the pattern to match the names of the nets to return. For example, "get_nets N_255*" returns all nets starting with the characters "N_255", where '*' is a wildcard that represents any character string.

Description

This command returns a collection of nets matching the pattern you specify. You can only use this command as source objects in create clock (create_clock) or create generated clock (create_generated_clock) constraints and as -through arguments in set false path (set_false_path), set minimum delay (set_min_delay), set maximum delay (set_max_delay), and set multicycle path (set_multicycle_path) constraints.
Exceptions

None

Examples

set_max_delay 2 -from [get_ports RDATA1] -through [get_nets {net_chkp1 net_chkqi}]
set_false_path -through [get_nets {Tblk/rm/n*}]
create_clock -name mainCLK -per 2.5 [get_nets {cknet}]

Microsemi Implementation Specifics

None

See Also

SDC Syntax Conventions

get_ports

Design object access command; returns the specified ports.

get_ports pattern

Argument

pattern

Specifies the pattern to match the ports. This is equivalent to the macros $in()[<pattern>] when used as –from argument and $out()[<pattern>] when used as –to argument or $ports()[<pattern>] when used as a –through argument.

Exceptions

None

Example

create_clock -period 10[get_ports CK1]

Microsemi Implementation Specifics

None

See Also

SDC Syntax Conventions