

UG0691 User Guide Libero SoC Design Flow Libero SoC v12.3 - SmartFusion2, IGLOO2, RTG4

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.**



a  **MICROCHIP** company

**Microsemi Headquarters**

One Enterprise, Aliso Viejo,
CA 92656 USA

Within the USA: +1 (800) 713-4113

Outside the USA: +1 (949) 380-6100

Fax: +1 (949) 215-4996

Email: sales.support@microsemi.com

www.microsemi.com

©2019 Microsemi, a wholly owned subsidiary of Microchip Technology Inc. All rights reserved. Microsemi and the Microsemi logo are registered trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners.

Microsemi makes no warranty, representation, or guarantee regarding the information contained herein or the suitability of its products and services for any particular purpose, nor does Microsemi assume any liability whatsoever arising out of the application or use of any product or circuit. The products sold hereunder and any other products sold by Microsemi have been subject to limited testing and should not be used in conjunction with mission-critical equipment or applications. Any performance specifications are believed to be reliable but are not verified, and Buyer must conduct and complete all performance and other testing of the products, alone and together with, or installed in, any end-products. Buyer shall not rely on any data and performance specifications or parameters provided by Microsemi. It is the Buyer's responsibility to independently determine suitability of any products and to test and verify the same. The information provided by Microsemi hereunder is provided "as is, where is" and with all faults, and the entire risk associated with such information is entirely with the Buyer. Microsemi does not grant, explicitly or implicitly, to any party any patent rights, licenses, or any other IP rights, whether with regard to such information itself or anything described by such information. Information provided in this document is proprietary to Microsemi, and Microsemi reserves the right to make any changes to the information in this document or to any products and services at any time without notice.

About Microsemi

Microsemi, a wholly owned subsidiary of Microchip Technology Inc. (Nasdaq: MCHP), offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Learn more at www.microsemi.com.

5-02-00691-7/12.19

Table of Contents

Table of Contents	2
Libero SoC Introduction.....	5
Welcome to Microsemi's Libero® SoC v12.3	5
Licensing and Additional Resources.....	5
Libero SoC Design Flow - SmartFusion2, IGLOO2, RTG4	14
Constraint Flow and Design Sources.....	16
Supported Families	18
File Types in Libero SoC.....	19
Software Tools - Libero SoC.....	20
Software IDE Integration	21
Libero Design Flow SmartFusion2, IGLOO2, and RTG4.....	22
Starting the Libero GUI	22
Design Report	23
Using the Libero SoC New Project Wizard	24
Create and Verify Design	32
System Builder	32
MSS - SmartFusion2 only	32
Create with SmartDesign	35
SmartDesign	35
Create Core from HDL	41
Designing with HDL.....	43
Designing with Block Flow	46
Verify Pre-Synthesized Design - RTL Simulation	46
Libero SoC Constraint Management.....	51
Invocation of Constraint Manager from the Design Flow Window	51
Libero SoC Design Flow	51
Introduction to Constraint Manager.....	52
Import a Constraint File.....	56
Constraint Types	61
Constraint Manager – I/O Attributes Tab	62
IO Advisor (SmartFusion2, IGLOO2, and RTG4)	64
Constraint Manager – Timing Tab	70
Derived Constraints.....	73
Constraint Manager – Floor Planner Tab	74
Constraint Manager – Netlist Attributes Tab	75
Implement Design	77

Synthesize.....	77
Verify Post-Synthesized Design.....	82
Configure Flash*Freeze	83
Configure Register Lock Bits	83
Constraint Flow in Implementation.....	85
Place and Route - SmartFusion2, IGLOO2, RTG4.....	90
Multiple Pass Layout Configuration (SmartFusion2, IGLOO2, RTG4)	94
Resource Usage (SmartFusion2, IGLOO2, RTG4)	96
Global Net Report	97
Verify Post Layout Implementation	103
Configure Hardware.....	124
Programming Connectivity and Interface.....	124
Select and Configure Actions and Procedures	125
Programmer Settings	126
Select Programmer	128
Program Design	129
Generate FPGA Array Data	129
Configure Actions and Procedures	134
Configure I/O States During JTAG Programming.....	137
Configure Programming Options (SmartFusion2 and IGLOO2).....	138
Configure Programming Options (RTG4 Only).....	141
Configure Security.....	145
Configure Bitstream	154
Generate Bitstream.....	155
Run Programming Device Actions - SmartFusion2, IGLOO2, RTG4	156
Debug Design.....	169
Generate SmartDebug FPGA Array Data.....	169
SmartDebug	169
Identify Debug Design.....	170
Handoff Design for Production.....	171
Export Bitstream.....	171
Export Bitstream - RTG4.....	176
Export FlashPro Express Job - SmartFusion2, IGLOO2, RTG4	178
Export Job Manager Data - SmartFusion2, IGLOO2.....	181
Export Pin Report.....	182
Export BSDL File.....	183
Export IBIS Model	183
Export uPROM Report - RTG4	184
Handoff Design for Firmware Development	185
Software IDE Integration	185
View/Configure Firmware Cores	185
Export Firmware – SmartFusion2	187

Export SmartDebug Data (Libero SoC)	189
References	192

Libero SoC Introduction



Welcome to Microsemis Libero® SoC v12.3

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.

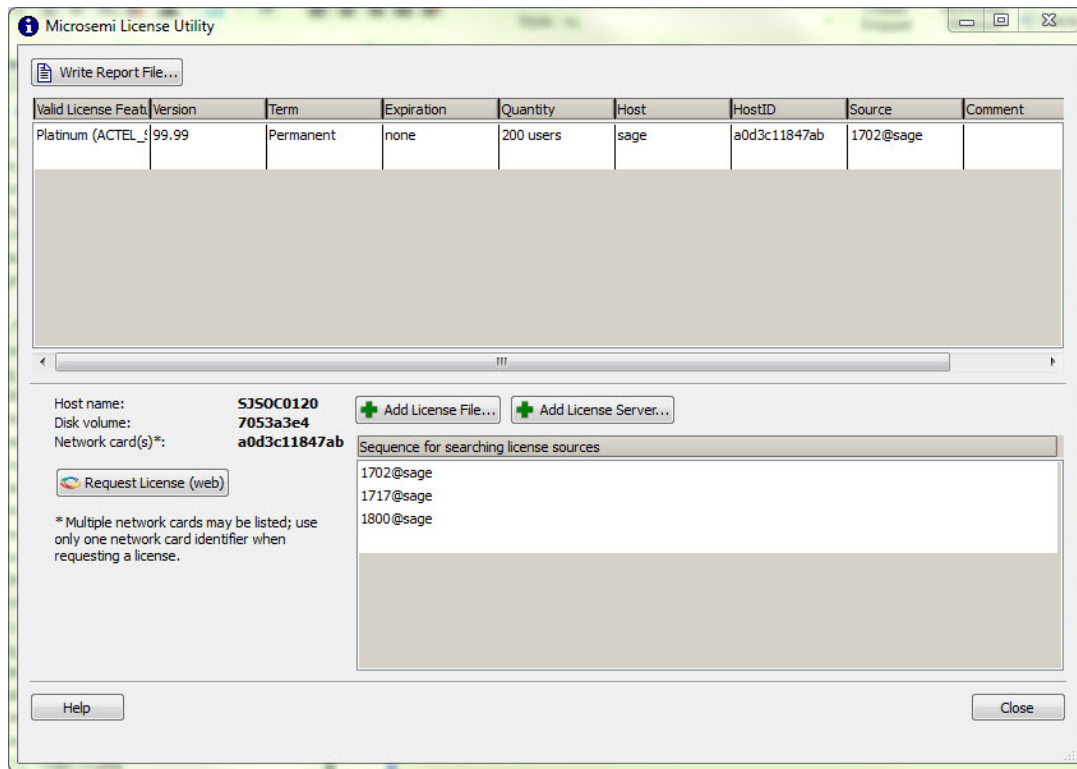


Figure 1 · Microsemi License Utility

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.

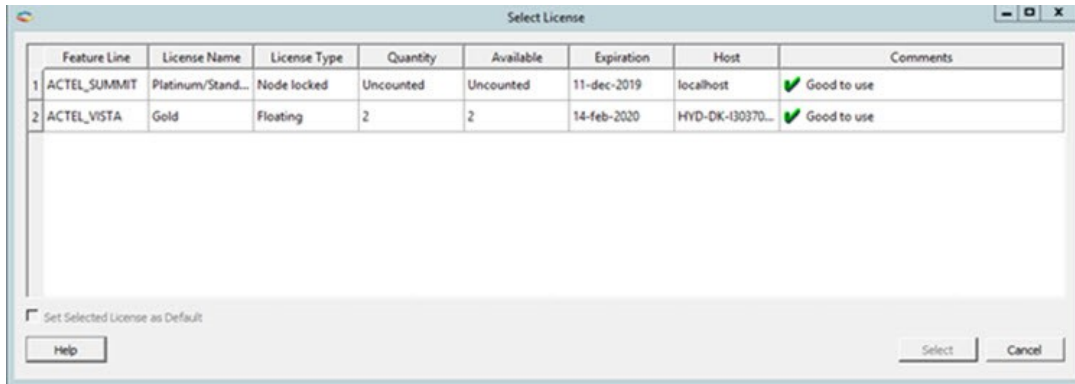


Figure 2 · Select License Dialog Box

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.

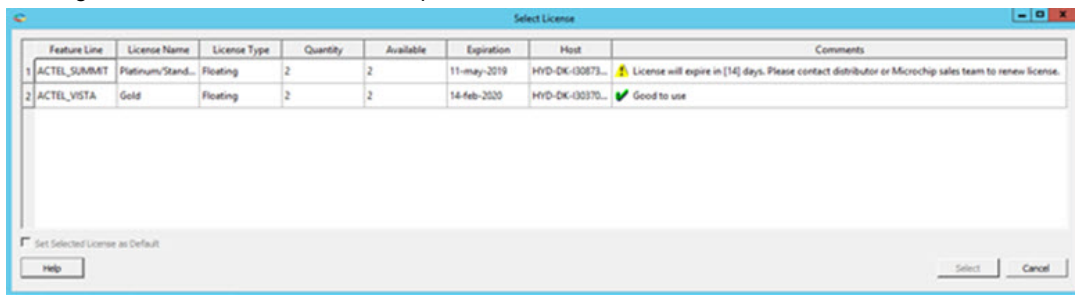


Figure 3 · Select License Dialog Box with License Expiration Warning Message

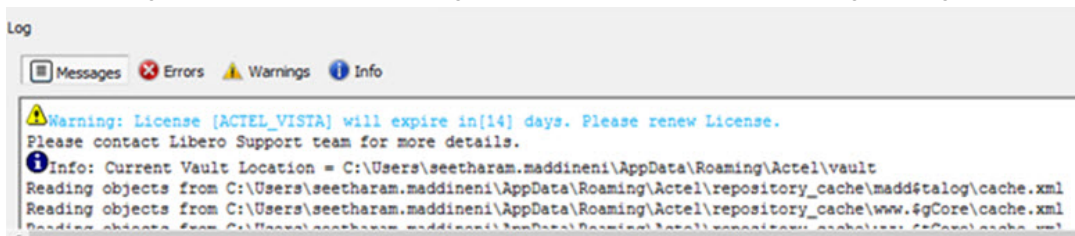


Figure 4 · Log Window with License Expiration Warning Message

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.

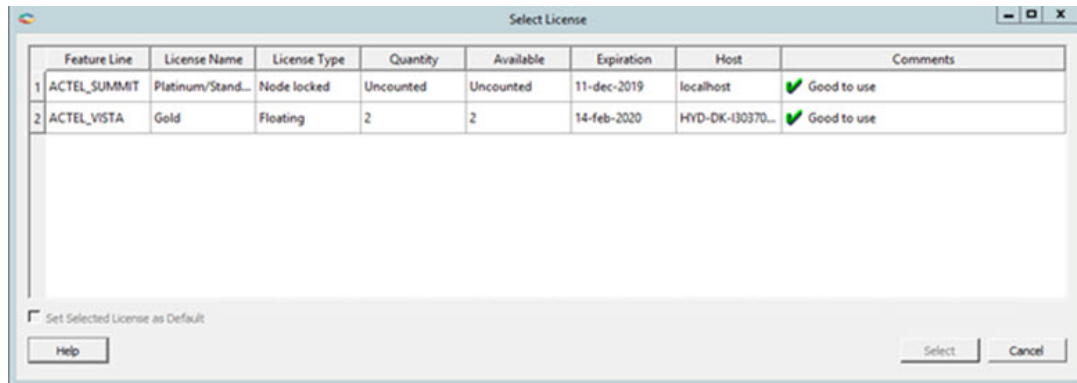
Selecting this option skips the license selection step for future sessions. This option can be used if you want to use the same license features for future sessions.

License Types

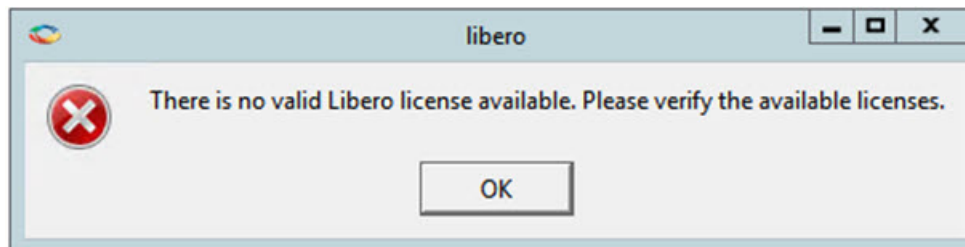
The **License Type** column indicates whether an available license is a Node Locked license or a Floating or Server-based license.

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

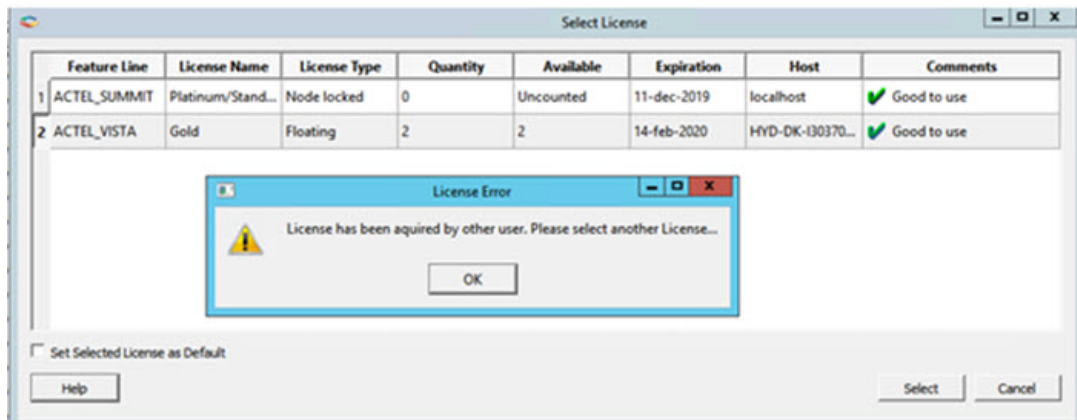
Floating or Server-based Licenses can be used by multiple users, depending on the number of seats available.



If there are no valid licenses available, the following message appears:



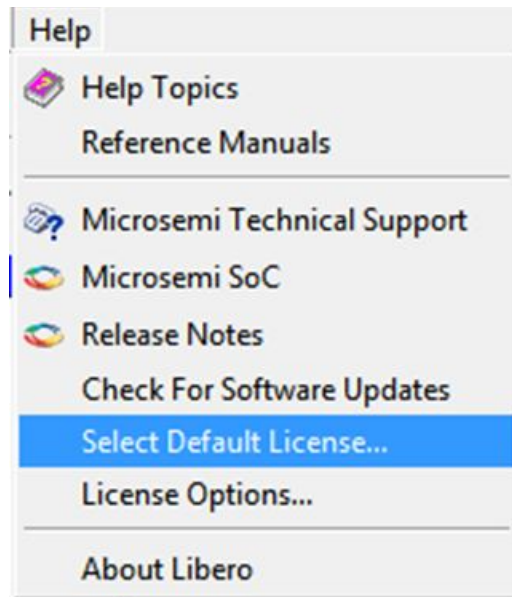
If a selected license has been acquired by another user, you will be notified with the following message:



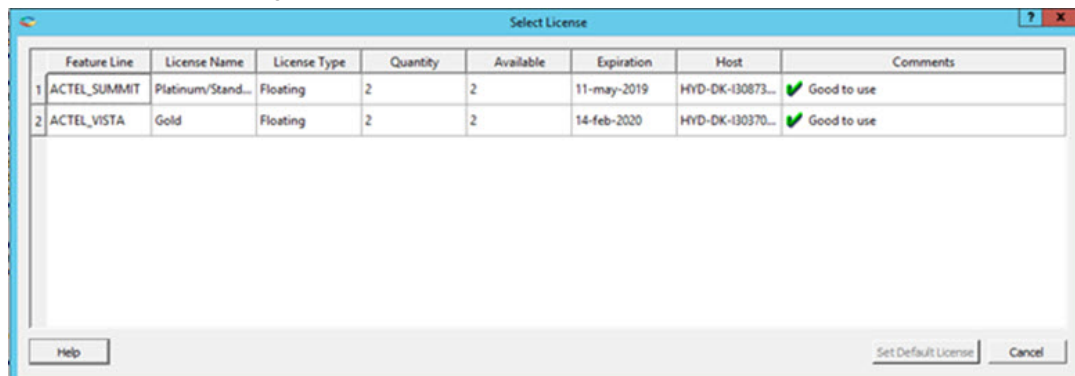
Additional License Information

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**.

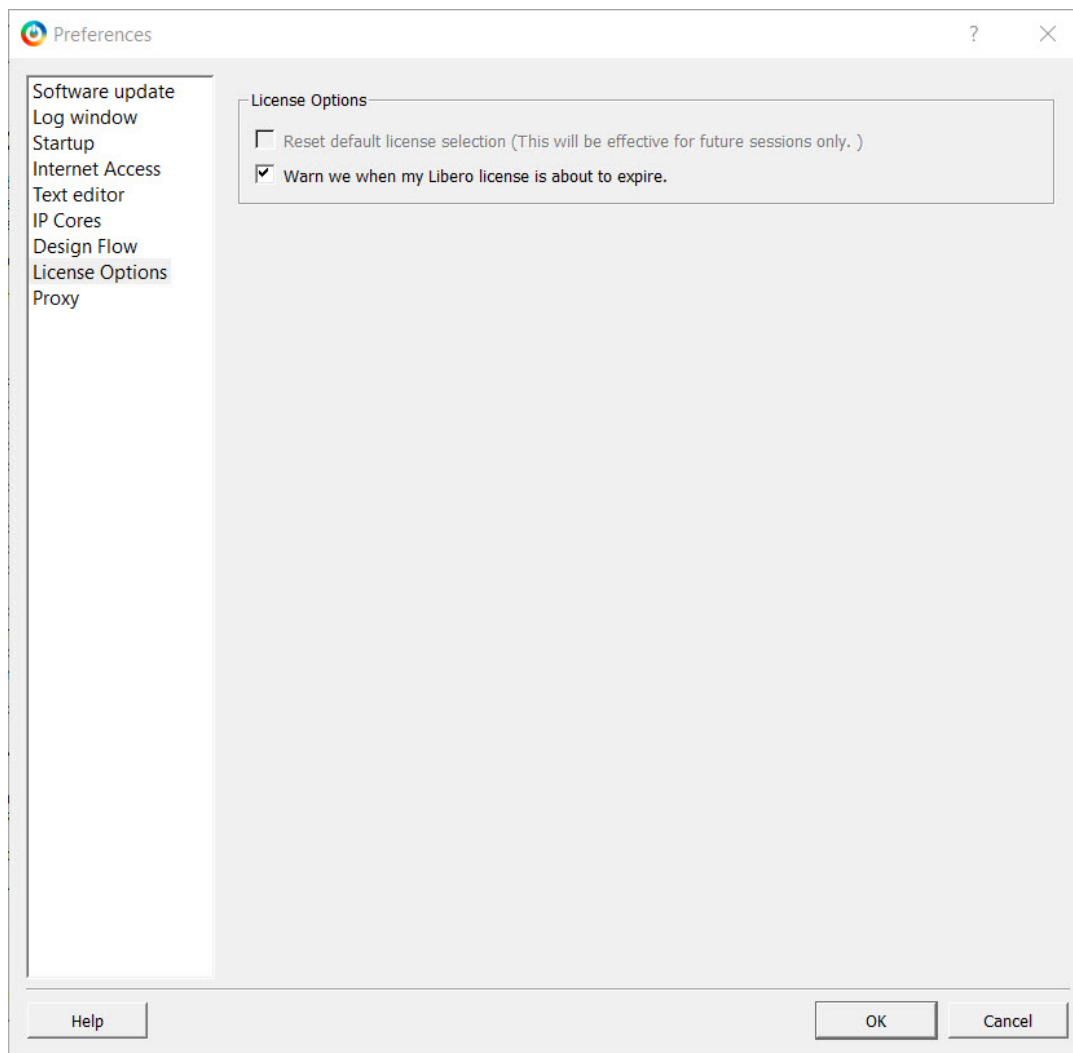


Figure 5 · License Options in Libero Preferences Dialog Box

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

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.

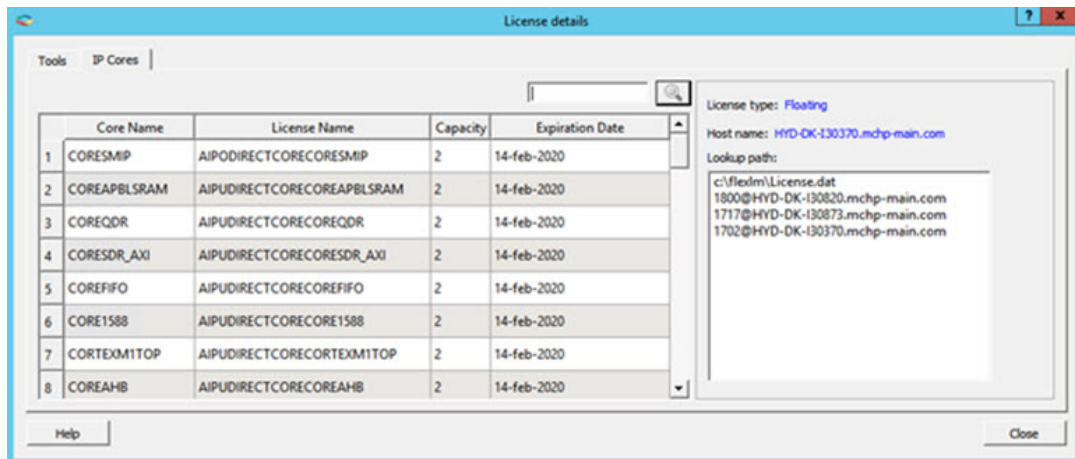


Figure 6 · IP Cores License Details

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](#)

FPGA and SoC Product Documentation Available on the Microsemi Web Site

[General Information about Microsemi's FPGA & SoC products is available here.](#)

Information About Supported Families:

Table 1 · Product Families and Derivatives

Device Family	Family Derivatives	Description
SmartFusion2	N/A	Address fundamental requirements for advanced security, high reliability and low power in critical industrial, military, aviation, communications and medical applications.
IGLOO2	N/A	Low-power mixed-signal programmable solution
RTG4	N/A	Microsemi's new RTG4 family of radiation-tolerant FPGAs

Information About Libero SoC Software

[More information about Libero SoC Software is available here.](#)

Application Notes and Tutorials

Application Notes and Tutorials are generally found on the **Documentation** tab of the supported technology page. For example, [SmartFusion2 Application Notes and Tutorials can be found here.](#)

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 then 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.

Address:	Microsemi SoC Products Group 3870 North First Street San Jose, CA 95134
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 Design Flow - SmartFusion2, IGLOO2, RTG4

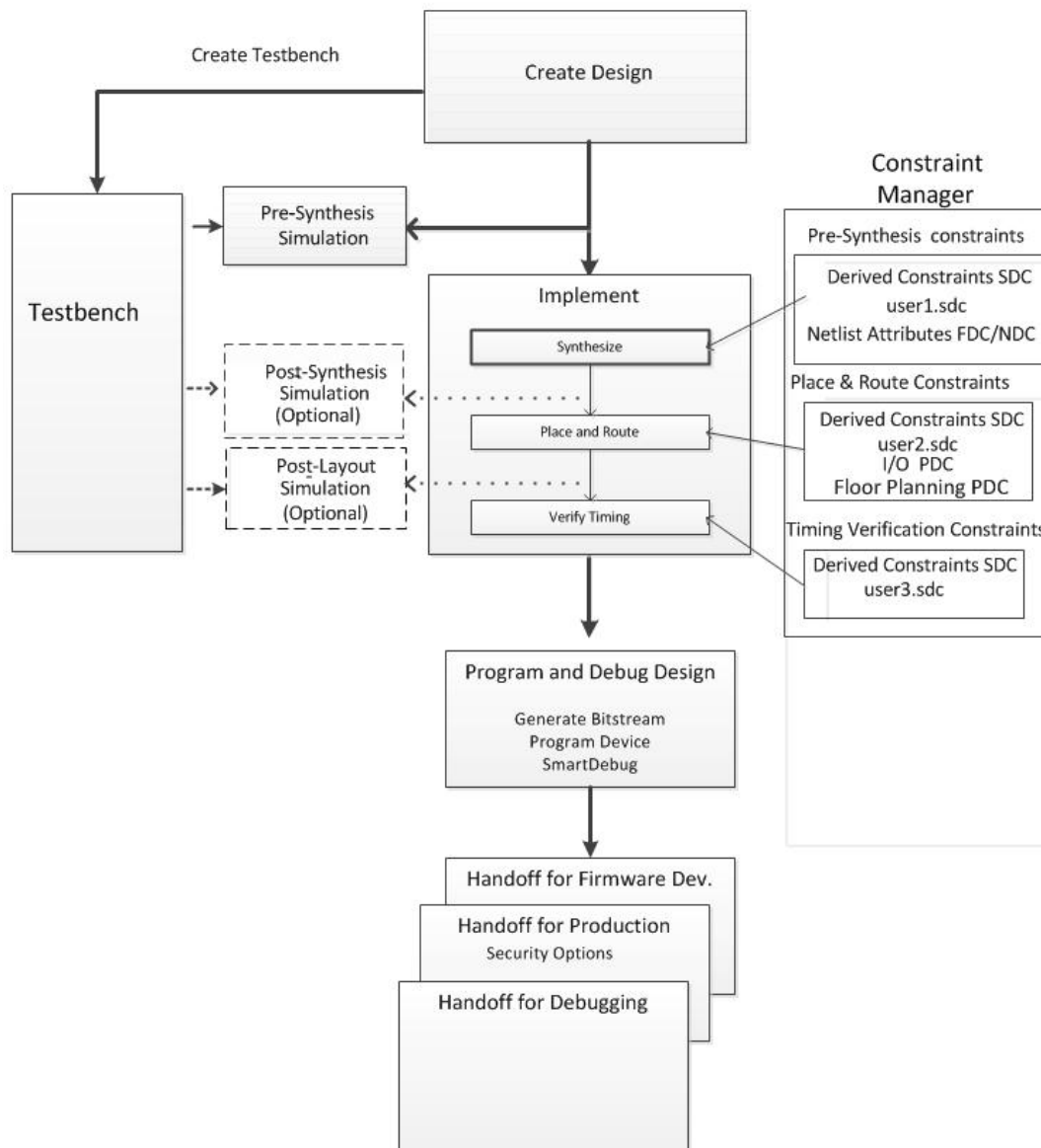



Figure 7 · Libero SoC Design Flow for SmartFusion2, IGLOO2, RTG4 Devices

Create Design

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

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

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.

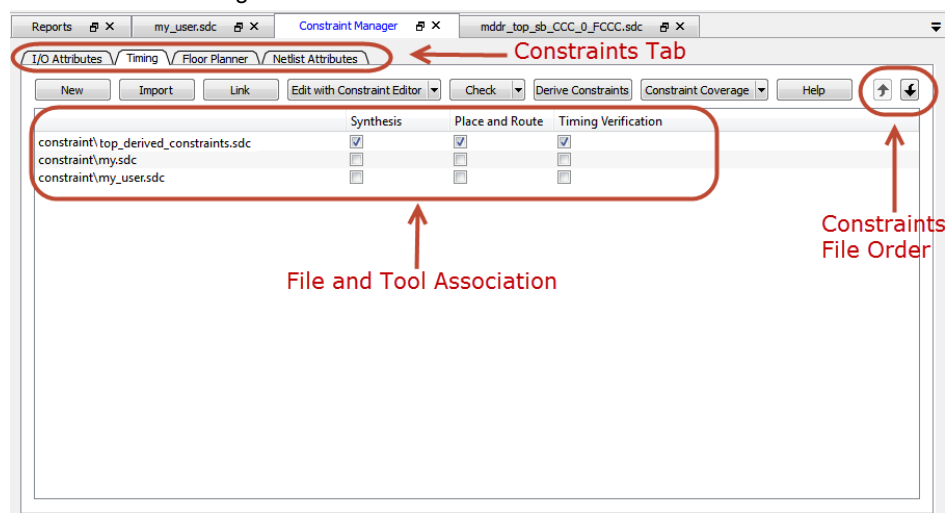



Figure 8 · Constraint Manager

- **See Also**
 - [Constraint Manager](#)
 - [New Project Wizard to import/link design constraints when creating new projects](#)

Implement

- [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 \(Simulate\)](#)
- [Configure Flash*Freeze](#)
- [Configure Register Lock Bits](#)

- [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
 - Simulate
 - Verify Timing - Right click and select [Configure Options](#) to specify a timing report with your desired conditions.
 - [Open SmartTime](#)
 - [Verify Power](#)
 - [SSN Analyzer](#)

Program and Debug Design

- [Generate FPGA Array Data](#)
- [Update eNVM Memory Content](#)
- **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.
 - [Device I/O States During Programming - JTAG Mode Only](#) - Sets your device I/O states during programming; use if your design requires that you change the default I/O states.
- [Configure Programming Options](#)
- [Configure Security Policy Manager](#)
- **Program Design**
 - [Generate Bitstream](#)
 - [Run PROGRAM Action](#)
- **Debug Design**
 - Identify Debug Design
 - [SmartDebug \(User Guide\)](#)
- **Configure Permanent Locks for Production (Configure OTP Security)**

Handoff Design for Production

- [Export Bitstream](#)
- [Export FlashPro Express Job](#)
- [Export Job Manager Data](#)
- [Export Pin Report](#)
- [Export BSDL](#)
- [Export IBIS Model](#)

Handoff Design for Firmware Development

[Handoff Design for Debugging \(Export SmartDebug Data\)](#)

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.

<div><div>HDL Flow</div><div><div>Design Flow</div><div>Top Module(root): top</div><div>Active Synthesis Implementation: synthesis</div><div><div>Tool</div><div><div>Create Design</div><div>Constraints</div><div>Manage Constraints</div><div>Implement Design</div><div>Open Netlist Viewer</div><div>Synthesize</div><div>Verify Post-Synthesized Design</div><div>Configure Flash*Freeze</div><div>Configure Register Lock Bits</div><div>Place and Route</div><div>Verify Post Layout Implementation</div><div>Program and Debug Design</div><div>Handoff Design for Production</div><div>Handoff Design for Firmware Development</div><div>Handoff Design for Debugging</div></div></div></div></div> <div>Design Flow Window</div>	<div><div>Netlist Flow</div><div><div>Design Flow</div><div>Top Module(root): shift32</div><div>Active Synthesis Implementation: synthesis</div><div><div>Tool</div><div><div>Create Design</div><div>Constraints</div><div>Implement Design</div><div>Open Netlist Viewer</div><div>Verify Post-Synthesized Design</div><div>Compile Netlist</div><div>Configure Flash*Freeze</div><div>Configure Register Lock Bits</div><div>Place and Route</div><div>Verify Post Layout Implementation</div><div>Program and Debug Design</div><div>Handoff Design for Production</div><div>Handoff Design for Firmware Development</div><div>Handoff Design for Debugging</div></div></div></div></div> <div>Design Flow Window</div>																		
<div><div>Reports</div><div>top.v</div><div>Constraint Manager*</div><div>StartPage</div><div>I/O Attributes</div><div>Timing*</div><div>Floor Planner</div><div>Netlist Attributes</div><div>New</div><div>Import</div><div>Link</div><div>Edit</div><div>Check</div><div>Derive Constraints</div><table><thead><tr><th></th><th>Synthesis</th><th>Place and Route</th></tr></thead><tbody><tr><td>constraint\user.sdc</td><td><input checked="" type="checkbox"/></td><td><input checked="" type="checkbox"/></td></tr><tr><td>constraint\verif.sdc</td><td><input type="checkbox"/></td><td><input type="checkbox"/></td></tr></tbody></table></div> <div>Constraint Manager</div>		Synthesis	Place and Route	constraint\user.sdc	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	constraint\verif.sdc	<input type="checkbox"/>	<input type="checkbox"/>	<div><div>I/O Attributes</div><div>Timing</div><div>Floor Planner</div><div>Netlist Attributes</div><div>New</div><div>Import</div><div>Link</div><div>Edit</div><div>Check</div><div>Derive Constraints</div><table><thead><tr><th></th><th>Place and Route</th><th>Timing Verification</th></tr></thead><tbody><tr><td>constraint\user.sdc</td><td><input checked="" type="checkbox"/></td><td><input checked="" type="checkbox"/></td></tr><tr><td>constraint\my2.sdc</td><td><input checked="" type="checkbox"/></td><td><input type="checkbox"/></td></tr></tbody></table></div> <div>Constraint Manager</div>		Place and Route	Timing Verification	constraint\user.sdc	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	constraint\my2.sdc	<input checked="" type="checkbox"/>	<input type="checkbox"/>
	Synthesis	Place and Route																	
constraint\user.sdc	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>																	
constraint\verif.sdc	<input type="checkbox"/>	<input type="checkbox"/>																	
	Place and Route	Timing Verification																	
constraint\user.sdc	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>																	
constraint\my2.sdc	<input checked="" type="checkbox"/>	<input type="checkbox"/>																	
<div><div>I/O Attributes</div><div>Timing</div><div>Floor Planner</div><div>Netlist Attributes</div><div>New</div><div>Import</div><div>Link</div><div>Check</div><div>constraint\myndc.ndc</div></div> <div>Constraint Manager - Check *.fdc and *.ndc</div>	<div><div>I/O Attributes</div><div>Timing</div><div>Floor Planner</div><div>Netlist Attributes</div><div>New</div><div>Import</div><div>Link</div><div>Check</div><div>Compile Netlist</div><div>constraint\myndc.ndc</div><div><input checked="" type="checkbox"/></div></div>																		

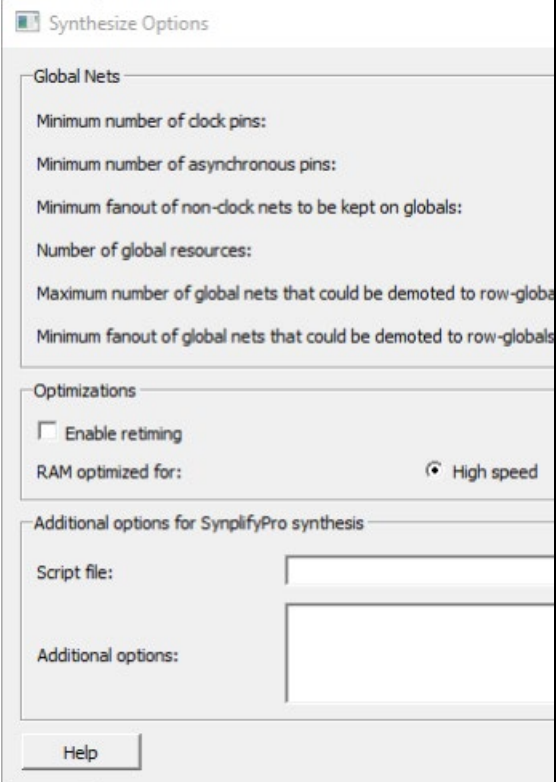
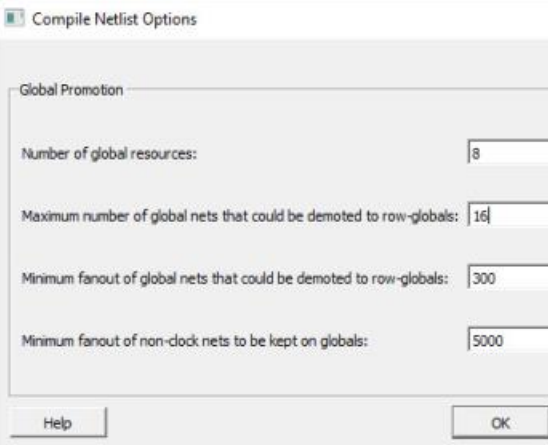
HDL Flow	Netlist Flow
	Constraint Manager - Check *.ndc only
 <p>Global Promotion/Demotion Options set in Synthesis Options Dialog Box</p>	 <p>Global Promotion/Demotion Options set in Compile Netlist Options Dialog Box</p>

Figure 9 · HDL vs. Netlist Flow

Supported Families

Microsemi's Libero SoC software supports the following families of devices:

- SmartFusion2
- IGLOO2
- RTG4

When we specify a family name, we refer to the device family and all its derivatives, unless otherwise specified. See the table below for a list of supported device families and their derivatives:

Table 2 · Product Families and Derivatives

Device Family	Family Derivatives	Description
SmartFusion2	N/A	Address fundamental requirements for advanced security, high reliability and low power in critical industrial, military, aviation, communications and medical applications.
IGLOO2	N/A	Low-power mixed-signal programmable solution
RTG4	N/A	Microsemi's new family of radiation-tolerant FPGAs

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 *.prj file; only one *.prj file is enabled for each Libero SoC project. If you associate Libero SoC as the default program with the *.prj file (Project > Preferences > Startup > Check the default file association (.prj) at startup), you can double-click the *.prj file to open the project with Libero SoC.

component directory - Stores your SmartDesign components (SDB and CFX 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.

Refer to the SmartFusion2/IGLOO2 Custom Flow User Guide for details about how to run synthesis, simulation, firmware development, programming, and power analysis outside the Libero SoC environment.

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 files, *.bfm files 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 - *.edn, *_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.

File	File Extension	Remarks
Routing Segmentation File	*.seg	
Combiner Info	*.cob	
Hierarchical Netlist	*.adl	
Flattened Netlist	*.afl	
Location file	*.loc	
map file	*.map	Fabric Programming File
tieoffs.txt	*.txt	RTG4 devices only

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>

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

Project Manager, HDL Editor targets the creation of HDL code. HDL Editor supports VHDL and Verilog with color, highlighting keywords for both HDL languages.

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:

- **ChipPlanner** displays I/O and logic macros in your design for floorplanning
- **Netlist Viewer** design schematic viewer
- **SmartPower** power analysis tool
- **SmartTime** static timing analysis and constraints editor

ModelSim ME 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 is supported in Libero Gold and Platinum only.)

Software IDE Integration

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

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

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

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

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

See Also

[Exporting Firmware and the Software IDE Workspace](#)

[Running Libero SoC from your Software Tool Chain](#)

[View/Configure Firmware Cores](#)

Libero Design Flow SmartFusion2, IGLOO2, and RTG4

When creating a new project, follow the directions in the New Project Wizard to create a project.

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.

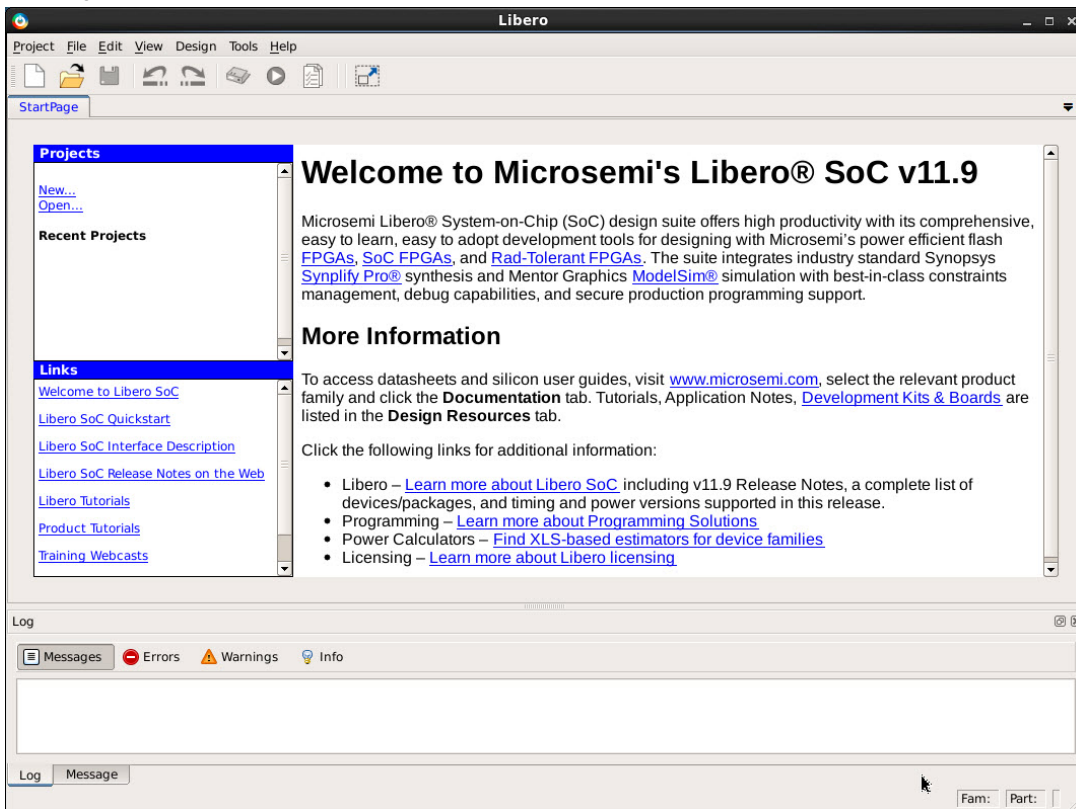


Figure 10 · Libero SoC Start-up GUI

- 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.

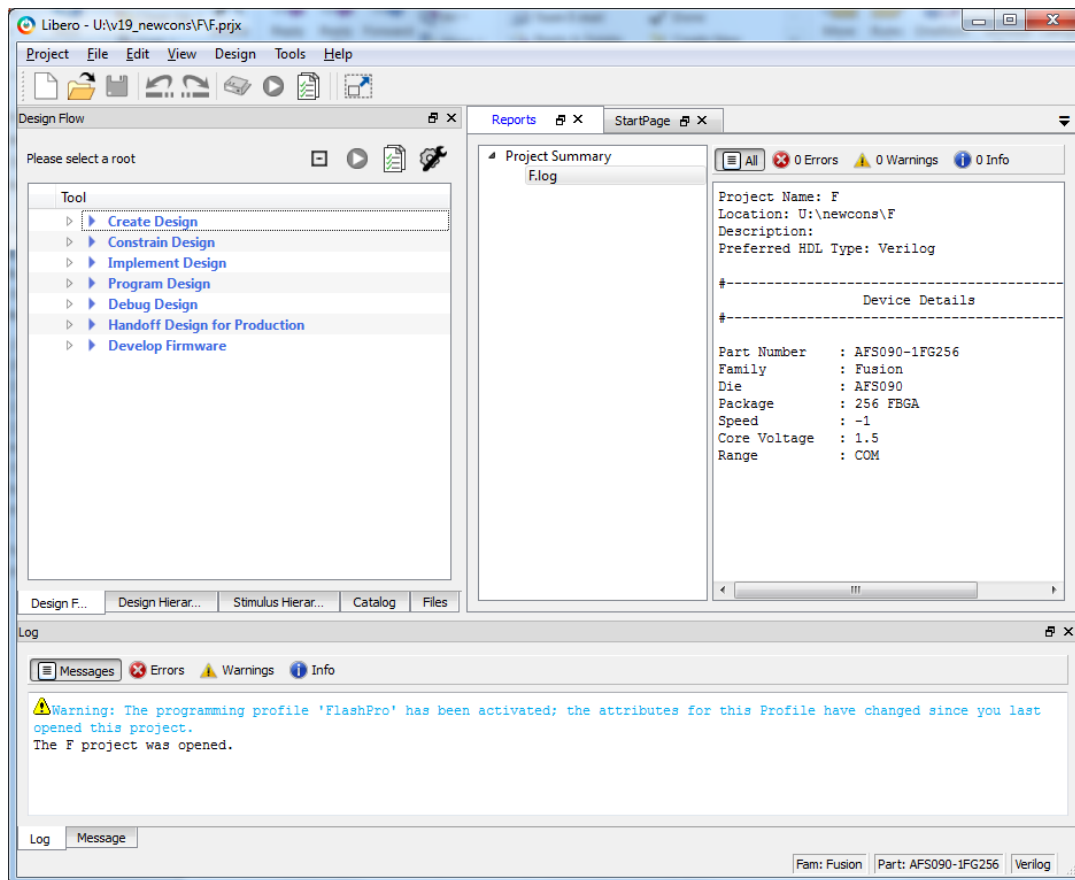


Figure 11 · Design Flow Window

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
- 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 Bitstream](#)
 - [Export Pin Report](#)
 - [Export BSDL File](#)
 - [Export IBIS Model](#)

Using the Libero SoC New Project Wizard

Start a new Libero SoC project by pulling down the Main Libero Menu item **Project** and selecting **New Project**

This will bring up the Libero SoC New Project Wizard which will walk you through the steps to create a new Libero Project:

- [Project Details](#) such as Name and file location
- [Device Selection](#) - Once the device selection has been made, and in any dialog box hereafter, you may click on the "Finish" button.
- [Device Settings](#)
- [Design Template](#) - this dialog box may be not be available if there are no design templates for the chosen technology.
- [Add HDL Sources](#)
- [Add Constraints](#)

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
- Use System Builder or MSS in your design project (SmartFusion2 and IGLOO2 only)
- Import HDL source files and/or design constraint files into your project

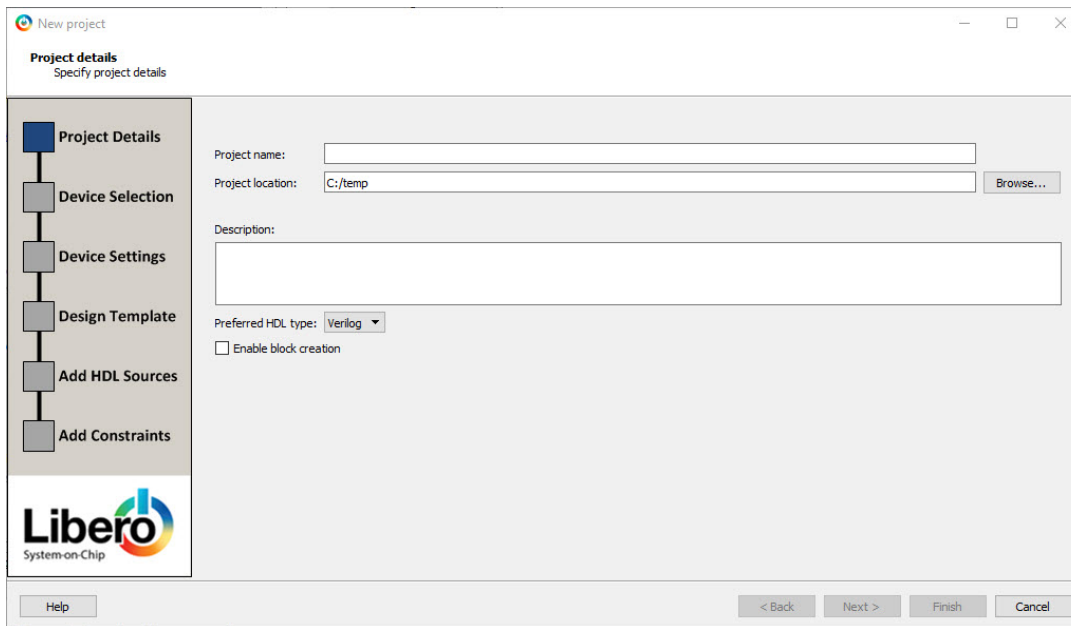


Figure 12 · Libero SoC New Project Creation Wizard

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 - Device Selection](#)

[New Project Creation Wizard – Device Settings](#)

[New Project Creation Wizard – Design Template](#) (SmartFusion2 and IGLOO2 only)

[New Project Creation Wizard – Add HDL Source Files](#)

[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 SmartFusion2 is selected in the Family filter:

- The parts table includes only SmartFusion2 parts.
- The Die filter includes only SmartFusion2 dies in the drop-down list for Die.

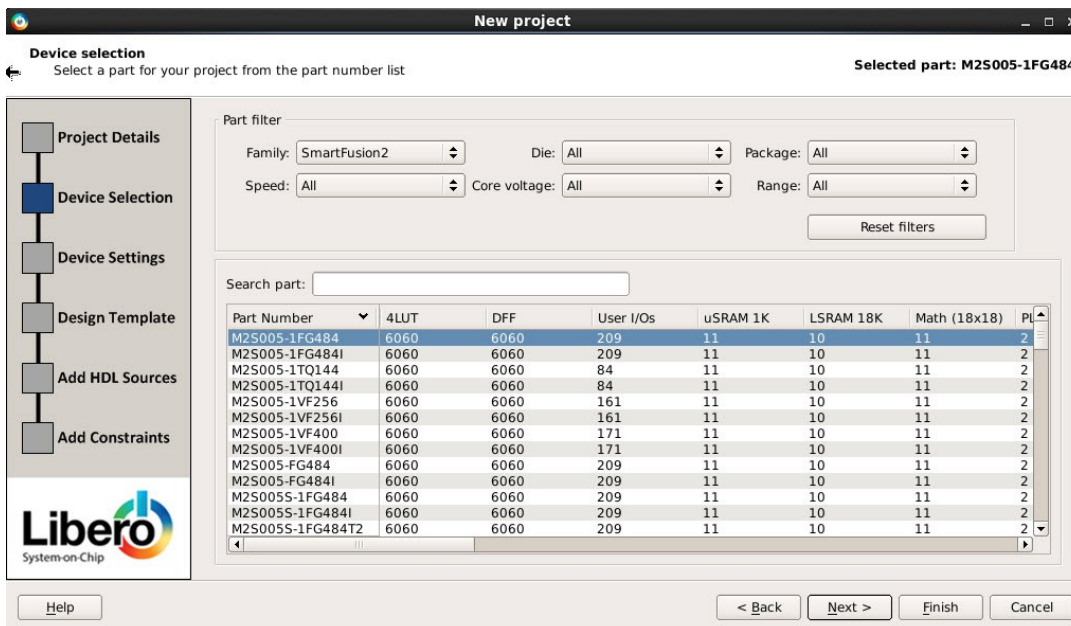


Figure 13 · New Project Creation Wizard - Device Selection Page

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

Die / Package / Speed - Set your device die, package, and speed grade, respectively. Only parts matching the filtering option are listed in the parts table.

Core Voltage - Set the core voltage for your device. Two numbers separated by a "~" are shown if a wide range voltage is supported. For example, 1.2~1.5 means that the device core voltage can vary between 1.2 and 1.5 volts.

Range - From the provided pick list, select the temperature range a device may encounter in your application. Junction temperature is a function of ambient temperature, air flow, and power consumption. Tools such as SmartTime, SmartPower, timing-driven layout, power-driven layout, the timing report, and back-annotated simulation are affected by operating conditions.

Supported ranges include:

- ALL - All ranges
- EXT (Extended)
- COM (Commercial) - Not available for RTG4 devices
- IND (Industrial)
- TGrade1 (Automotive) - Not available for RTG4 devices
- TGrade2 (Automotive) - Not available for RTG4 devices
- MIL (Military)

Note: Supported operating condition ranges vary according to your device and package. Refer to the device datasheet to find your recommended temperature range. The temperature range corresponding to the value selected from the pick list can also be found by checking [Project Settings](#) > Analysis operating conditions.

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

Search Part – 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.

Note: Once the project has been created, many device settings can be modified in the [Project Settings](#) dialog box tabs for "Device selection", "Device Settings", and "Analysis operating conditions".

New Project Creation Wizard – Device Settings

For SmartFusion2 and IGLOO2 devices, the Device Settings page is where you set the Device I/O Technology, PLL Supply Voltage, Reserve pins for Probes and activate the System Controller Suspended Mode.

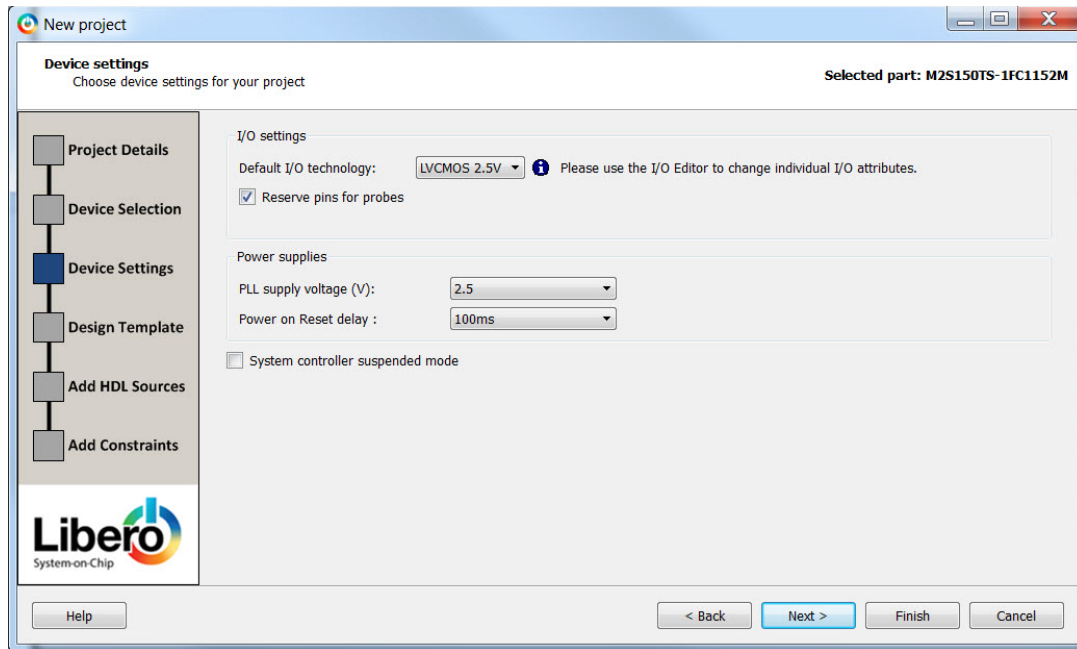


Figure 14 · New Project Creation Wizard – Device Settings Page (SmartFusion2 and IGLOO2)

For RTG4 devices, the Device Settings page is where you set the Device I/O Technology, Reserve pins for Probes, and activate Enable Single Event Transient mitigation.

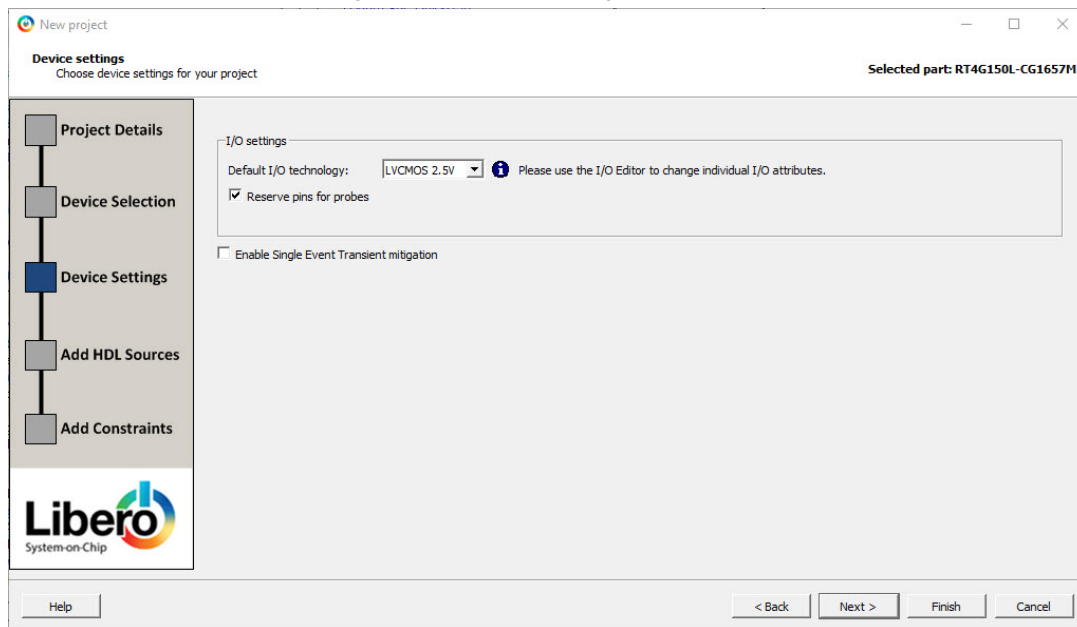


Figure 15 · New Project Creation Wizard – Device Settings Page (RTG4)

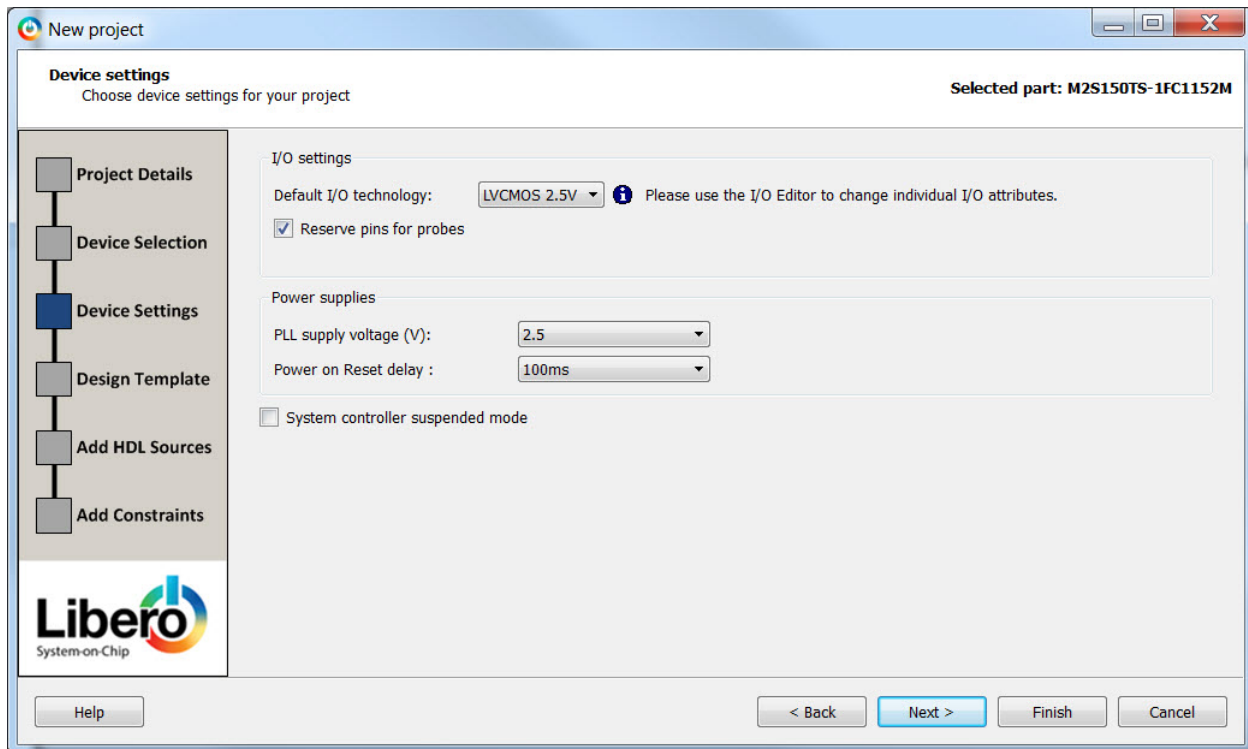


Figure 16 · New Project Creation Wizard – Device Settings Page (SmartFusion2)

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.

Enable Single Event Transient mitigation (RTG4 only) - Controls the mitigation of Single Event Transient (SET) in the FPGA fabric. When this box is checked, SET filters are turned on globally to help mitigate radiation-induced transients. By default, this box is unchecked.

PLL Supply Voltage (V) (SmartFusion2, IGLOO2 only) - Set the voltage for the power supply that you plan to connect to all the PLLs in your design, such as MDDR, FDDR, SERDES, and FCCC.

Maximum Core Voltage Rail Ramp Up Time (SmartFusion2, IGLOO2 only) - Power-up management circuitry is designed into every SmartFusion2 and IGLOO2 SoC FPGA. These circuits ensure easy transition from the powered-off state to the powered-up state of the device. The SmartFusion2, IGLOO2, and RTG4 system controller is responsible for systematic power-on reset whenever the device is powered on or reset. All I/Os are held in a high-impedance state by the system controller until all power supplies are at their required levels and the system controller has completed the reset sequence.

The power-on reset circuitry in SmartFusion2 and IGLOO2 devices requires the VDD and VPP supplies to ramp monotonically from 0 V to the minimum recommended operating voltage within a predefined time. There is no sequencing requirement on VDD and VPP. Four ramp rate options are available during design generation: 50 μ s, 1 ms, 10 ms, and 100 ms. Each selection represents the maximum ramp rate to apply to VDD and VPP.

Device information (such as Die, Package and Speed) can be modified later in the [Project Settings](#) dialog box.

System Controller Suspended Mode (SmartFusion2, IGLOO2 only) - Enables designers to suspend operation of the System Controller. Enabling this bit instructs the System Controller to place itself in a reset state when the device is powered up. This effectively suspends all system services from being performed. For a list of system services for SmartFusion2 and IGLOO2, refer to the System Controller User's Guide for your device on the [Microsemi website](#).

When **Device Settings** is completed, click on:

- **Next**
OR
- **Finish** to complete New Project Creation.

Note: Once the project has been created, many device settings can be modified in the [Project Settings](#) dialog box tabs for "Device selection", "Device Settings", and "Analysis operating conditions".

New Project Creation Wizard – Design Template (SmartFusion2 and IGLOO2 only)

The Design Template page is where you can use Libero SoC's built-in template to automate your design process. The template uses the System Builder tool for system-level design or the Microcontroller Subsystem (MSS) in your design. Both will speed up your design process.

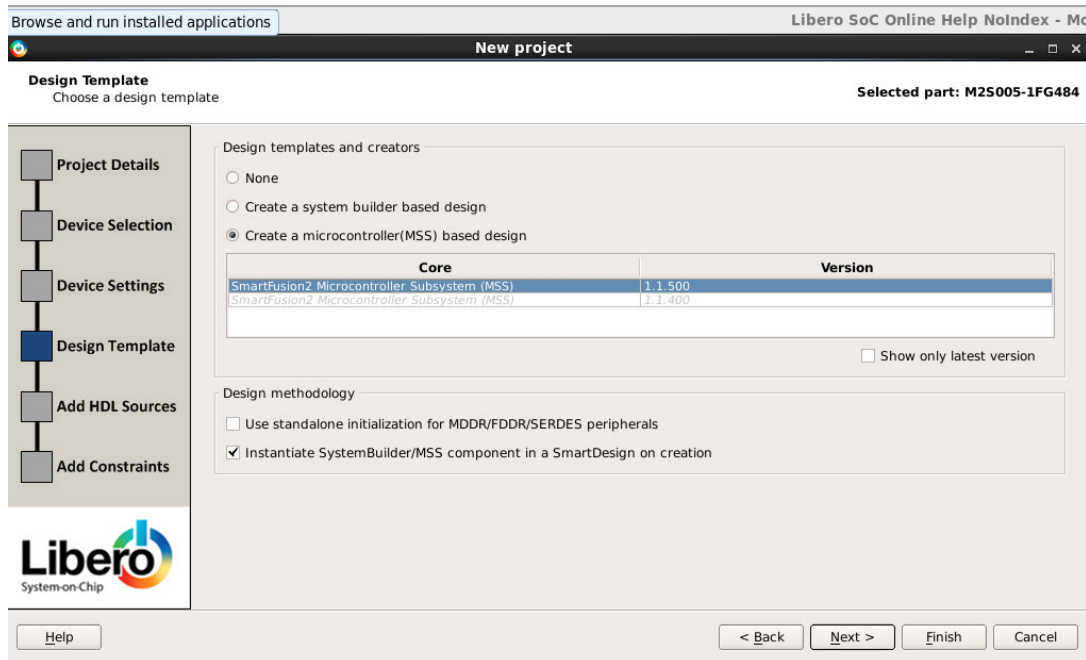


Figure 17 · New Project Creation Wizard – Design Template Page

None - Select if you do not want to use a design template.

Create a System Builder based design – Use System Builder to generate your top-level design.

Create a Microcontroller (MSS) based design – Instantiate a Microcontroller (MSS) in your design. The version of the MSS cores available in your vault is displayed. Select the version you desire.

Use Standalone Initialization for MDDR/FDDR/SERDES Peripherals – Check this box if you want to create your own peripheral initialization logic in SmartDesign for each of your design peripherals (MDDR/FDDR/SERDES). When checked, System Builder does not build the peripherals initialization logic for you. Standalone initialization is useful if you want to make the initialization logic of each peripheral separate from and independent of each other.

Instantiate System Builder/MSS component in a SmartDesign on creation - Uncheck this box if you are using this project to create System Builder or MSS components and do not plan on using them in a SmartDesign based design. This is especially useful for design flows where the System Builder or MSS components are stitched in a design using HDL.

When **Design Template** is completed, click on:

- **Next** to proceed to the [Add HDL Sources](#) page
- OR
- **Finish** to complete New Project Creation.

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.

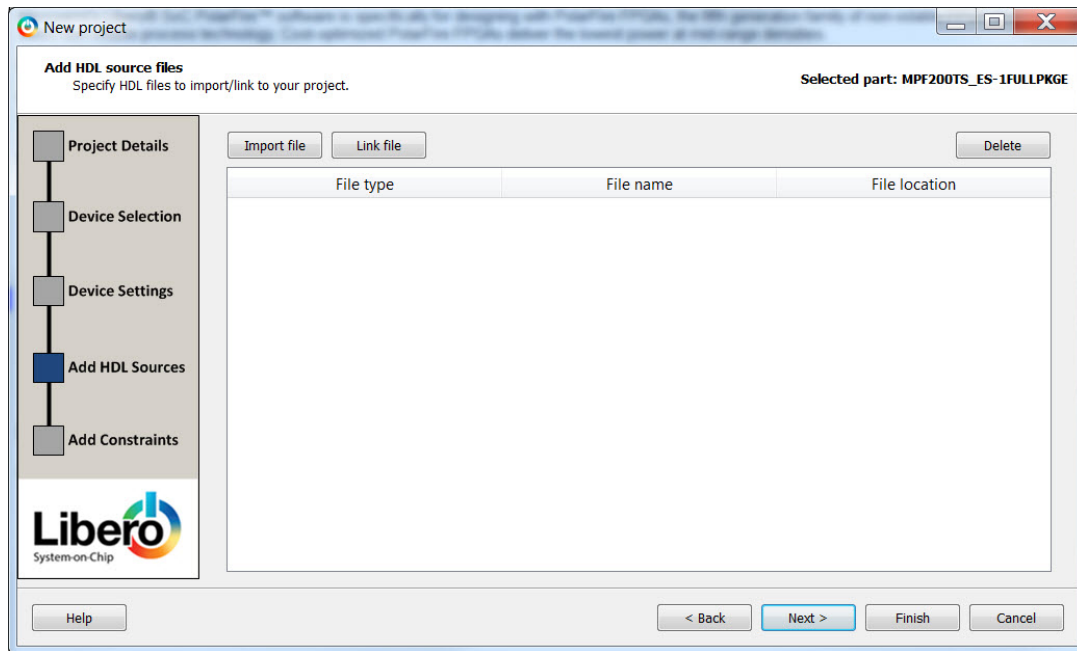


Figure 18 · New Project Creation Wizard - Add HDL Source Files Page

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**. 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
- OR
- **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.

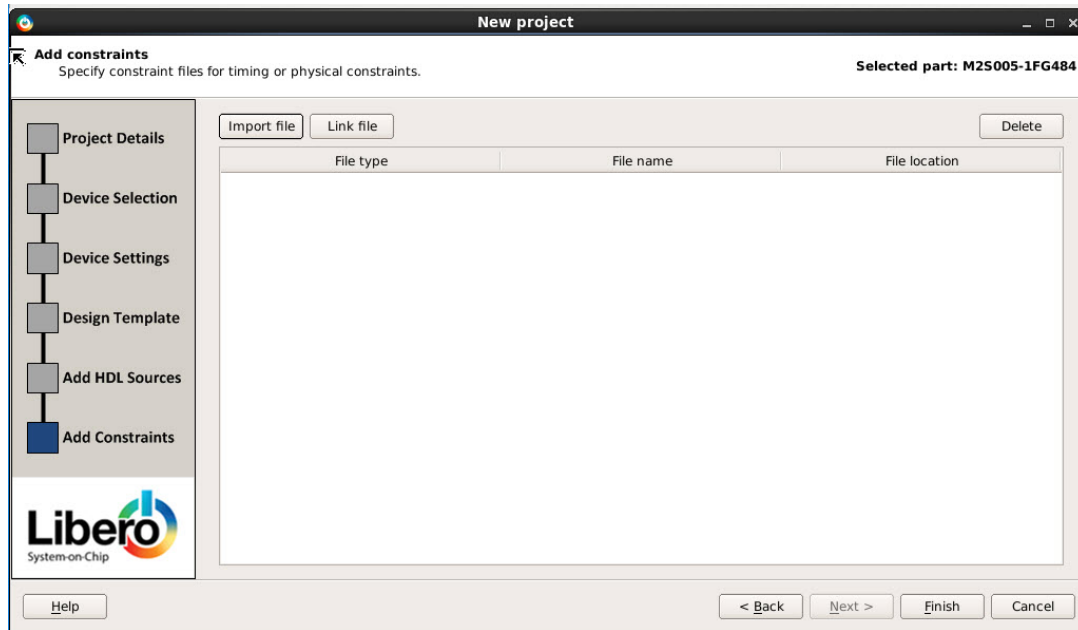


Figure 19 · New Project Creation Wizard – Add Constraints Page

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.

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.

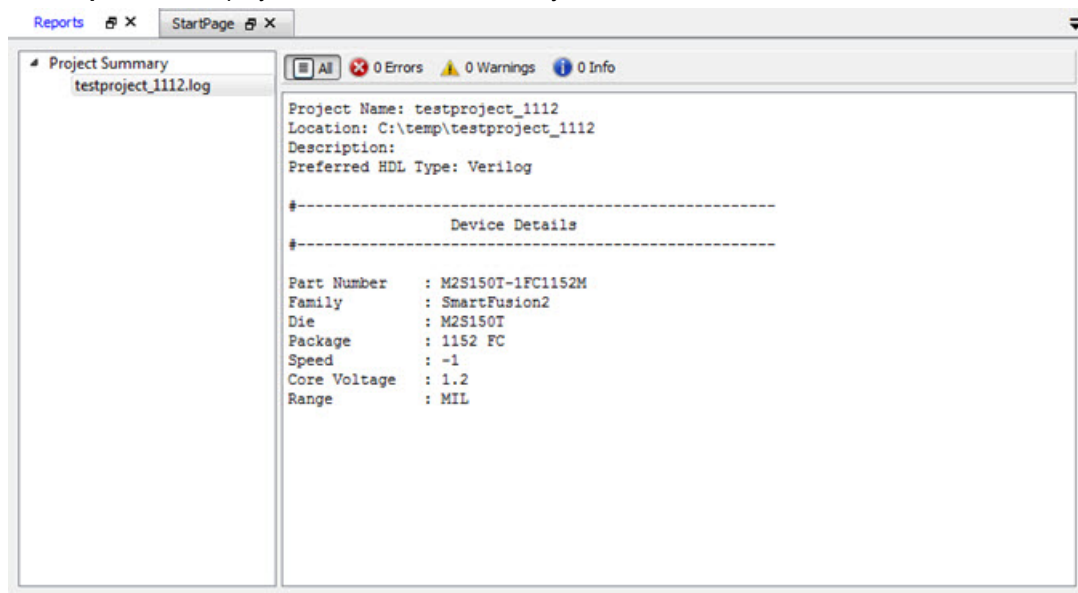


Figure 20 · Reports Tab (SmartFusion2)

Create and Verify Design

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

- [System Builder](#)
- [Create SmartDesign](#)
- [Create HDL](#)
- [Create SmartDesign Testbench](#) (optional, for simulation only)
- [Create HDL Testbench](#) (optional, for simulation only)

System Builder

System Builder is a graphical design wizard that enables you to enter high-level design specifications for SmartFusion2 or IGLOO2.

System Builder takes you through the following steps:

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

To start System Builder, do the following:

1. In the Design Flow window, click **System Builder > Run**.
2. Enter a name in the **Enter a name for your system** dialog box.
3. Click **OK**. The System Builder **Device Features** page opens.

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

See the [SmartFusion2 System Builder documentation](#) or the [IGLOO2 System Builder documentation](#) for a complete family-specific explanation of the tool.

MSS - SmartFusion2 only

Instantiate a SmartFusion2 MSS in your Design

You can configure peripherals within the SmartFusion2 MSS, such as the ARM® Cortex™-M3, embedded nonvolatile memory (eNVM), Ethernet MAC, timer, UART, and SPI to suit your needs. The MSS operates standalone without any dependencies on other logic within the device; however, designs that require functionality beyond a standalone MSS are handled by using SmartDesign to add user logic in the SmartFusion2 FPGA fabric.

You can instantiate a Microcontroller Subsystem into your design from the New Project Creation Wizard when you start a new SmartFusion2 project, or from the Design Flow window after you have created a new project.

To instantiate a SmartFusion2 MSS from the New Project Creation Wizard you must enable **Use Design Tool** (under **Design Templates and Creators**) and click to select **SmartFusion2 Microcontroller Subsystem (MSS)** from the list.

If you opted not to use a Design Tool when you created your project, in the Design Flow window expand **Create Design** and double-click **Configure MSS**. This opens the **Add Microcontroller Subsystem** dialog box. Enter your **Design Name** and click **OK** to continue. A SmartDesign Canvas appears with the MSS added to your project; double-click the MSS to view and [configure MSS components](#).

Configure the SmartFusion2 MSS

Documents for specific SmartFusion2 MSS peripherals are available on the [Peripheral Documents web page](#).

The SmartFusion2 Microcontroller Subsystem (MSS) Configurator (as shown in the figure below) contains the elements listed below. Double-click any element in the MSS to configure it; click the checkbox (if available) to enable or disable it in your design.

MSS ARM® Cortex™-M3**Peripherals**

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

Fabric Interfaces

- MSS Fabric Interface Controllers (FICs)

Additional Information

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

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

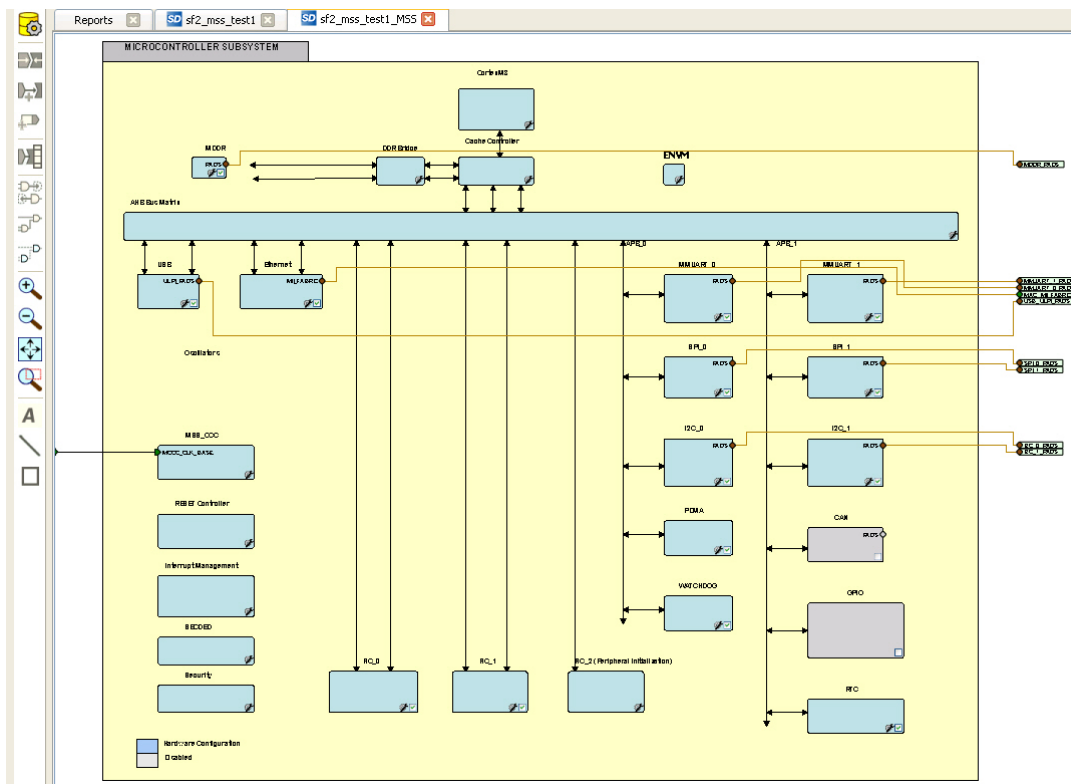


Figure 21 · Microcontroller Subsystem Configurator

Generate SmartFusion2 MSS Files

See the MSS Configurator help for more information on generating SmartFusion2 MSS files.

Click the **Generate Component** button  to create your SmartFusion2 MSS files.

The MSS Configurator generates the following files:

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

Create with SmartDesign

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 (*.xcf 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.

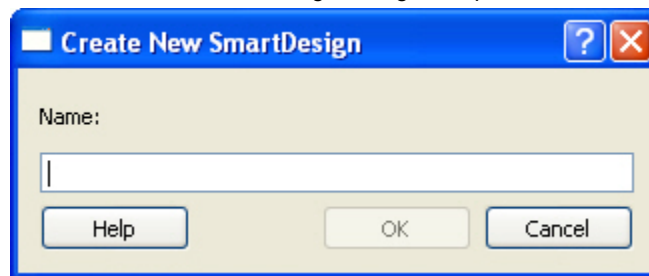


Figure 22 · Create New SmartDesign Dialog Box

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

Note: The Export Component Description(Tcl) option is not supported for the SmartFusion2/IGLOO2 MSS component and the System Builder component.

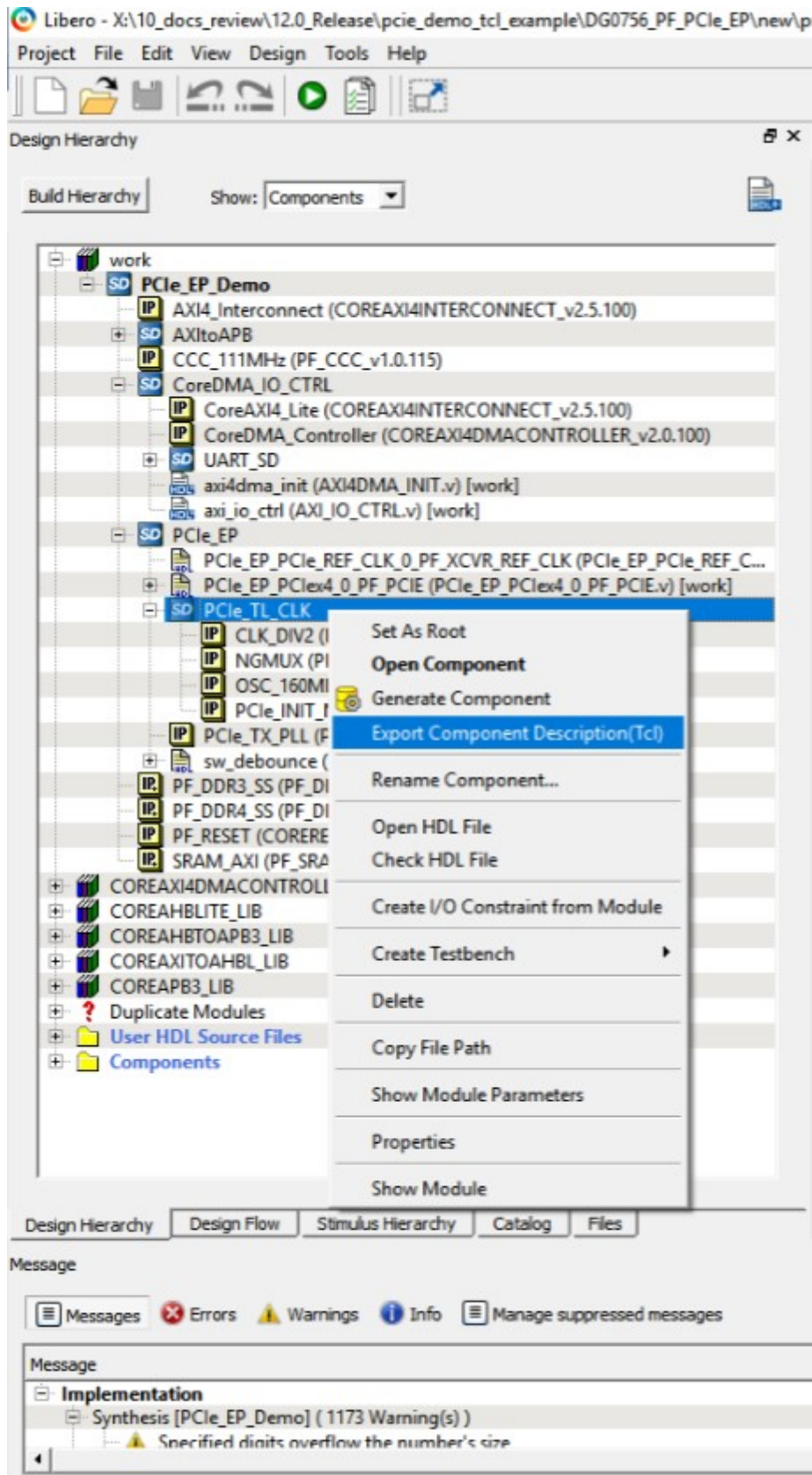


Figure 23 · Export As TCL option for SmartDesign Component

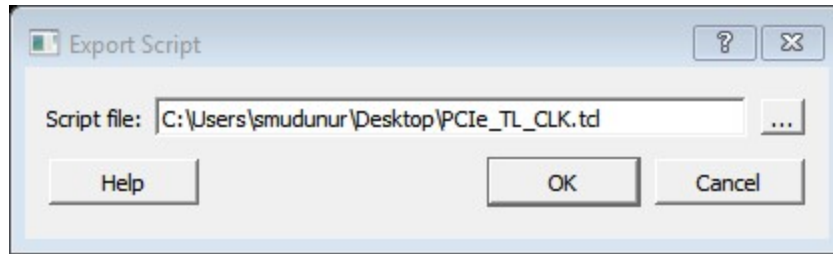


Figure 24 · Export Script Dialog Box

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)

```
# Creating SmartDesign PCie_TL_CLK
set sd_name {PCie_TL_CLK}
create_smartdesign -sd_name ${sd_name}
# Disable auto promotion of pins of type 'pad'
auto_promote_pad_pins -promote_all 0
# Create top level Ports
sd_create_scalar_port -sd_name ${sd_name} -port_name {CLK_125MHz} -port_direction {IN}
sd_create_scalar_port -sd_name ${sd_name} -port_name {TL_CLK} -port_direction {OUT}
sd_create_scalar_port -sd_name ${sd_name} -port_name {DEVICE_INIT_DONE} -port_direction {OUT}
# Add CLK_DIV2_0 instance
sd_instantiate_component -sd_name ${sd_name} -component_name {CLK_DIV2} -instance_name {CLK_DIV2_0}
# Add NGMUX_0 instance
sd_instantiate_component -sd_name ${sd_name} -component_name {NGMUX} -instance_name {NGMUX_0}
# Add OSC_160MHz_0 instance
sd_instantiate_component -sd_name ${sd_name} -component_name {OSC_160MHz} -instance_name {OSC_160MHz_0}
# Add PCie_INIT_MONITOR_0 instance
sd_instantiate_component -sd_name ${sd_name} -component_name {PCie_INIT_MONITOR} -instance_name {PCie_INIT_MONITOR_0}
sd_mark_pins_unused -sd_name ${sd_name} -pin_names {PCie_INIT_MONITOR_0:FABRIC_POR_N}
sd_mark_pins_unused -sd_name ${sd_name} -pin_names {PCie_INIT_MONITOR_0:USRAM_INIT_DONE}
sd_mark_pins_unused -sd_name ${sd_name} -pin_names {PCie_INIT_MONITOR_0:SRAM_INIT_DONE}
sd_mark_pins_unused -sd_name ${sd_name} -pin_names {PCie_INIT_MONITOR_0:XCVR_INIT_DONE}
sd_mark_pins_unused -sd_name ${sd_name} -pin_names {PCie_INIT_MONITOR_0:USRAM_INIT_FROM_SNVM_DONE}
sd_mark_pins_unused -sd_name ${sd_name} -pin_names {PCie_INIT_MONITOR_0:USRAM_INIT_FROM_UPROM_DONE}
sd_mark_pins_unused -sd_name ${sd_name} -pin_names {PCie_INIT_MONITOR_0:USRAM_INIT_FROM_SPI_DONE}
sd_mark_pins_unused -sd_name ${sd_name} -pin_names {PCie_INIT_MONITOR_0:SRAM_INIT_FROM_SNVM_DONE}
sd_mark_pins_unused -sd_name ${sd_name} -pin_names {PCie_INIT_MONITOR_0:SRAM_INIT_FROM_UPROM_DONE}
sd_mark_pins_unused -sd_name ${sd_name} -pin_names {PCie_INIT_MONITOR_0:SRAM_INIT_FROM_SPI_DONE}
sd_mark_pins_unused -sd_name ${sd_name} -pin_names {PCie_INIT_MONITOR_0:AUTOCALIB_DONE}
# Add scalar net connections
```

```

sd_connect_pins -sd_name ${sd_name} -pin_names {"NGMUX_0:CLK1" "CLK_125MHz" }
sd_connect_pins -sd_name ${sd_name} -pin_names {"CLK_DIV2_0:CLK_OUT" "NGMUX_0:CLK0" }
sd_connect_pins -sd_name ${sd_name} -pin_names {"PCIE_INIT_MONITOR_0:DEVICE_INIT_DONE"
"DEVICE_INIT_DONE" }
sd_connect_pins -sd_name ${sd_name} -pin_names {"CLK_DIV2_0:CLK_IN"
"OSC_160MHz_0:RCOSC_160MHZ_CLK_DIV" }
sd_connect_pins -sd_name ${sd_name} -pin_names {"NGMUX_0:SEL"
"PCIE_INIT_MONITOR_0:PCIE_INIT_DONE" }
sd_connect_pins -sd_name ${sd_name} -pin_names {"NGMUX_0:CLK_OUT" "TL_CLK" }
# Re-enable auto promotion of pins of type 'pad'
auto_promote_pad_pins -promote_all 1
# Save the smartDesign
save_smartdesign -sd_name ${sd_name}
# Generate SmartDesign PCIE_TL_CLK
generate_component -component_name ${sd_name}

```

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_DQS_GROUP_SIZE:8" \
"ENABLE_ECC:0" \
"ENABLE_INIT_INTERFACE:false" \
"ENABLE_LOOKAHEAD_PRECHARGE_ACTIVATE:false" \
"ENABLE_PAR_ALERT:false" \
"ENABLE_REINIT:false" \
"ENABLE_TAG_IF:false" \
"ENABLE_USER_ZQCALIB:false" \
"EXPOSE_TRAINING_DEBUG_IF:false" \
"FABRIC_INTERFACE:AXI4" \
"FAMILY:26" \
"MEMCTRLR_INST_NO:1" \
"MEMORY_FORMAT:COMPONENT" \

```

```

"MINIMUM_READ_IDLE:1" \
"ODT_ENABLE_RD_RNK0_ODT0:false" \
"ODT_ENABLE_RD_RNK0_ODT1:false" \
"ODT_ENABLE_RD_RNK1_ODT0:false" \
"ODT_ENABLE_RD_RNK1_ODT1:false" \
"ODT_ENABLE_WR_RNK0_ODT0:true" \
"ODT_ENABLE_WR_RNK0_ODT1:false" \
"ODT_ENABLE_WR_RNK1_ODT0:false" \
"ODT_ENABLE_WR_RNK1_ODT1:true" \
"ODT_RD_OFF_SHIFT:0" \
"ODT_RD_ON_SHIFT:0" \
"ODT_WR_OFF_SHIFT:0" \
"ODT_WR_ON_SHIFT:0" \
"OUTPUT_DRIVE_STRENGTH:RZQ6" \
"PARAM_IS_FALSE:false" \
"PARTIAL_ARRAY_SELF_REFRESH:FULL" \
"PHYONLY:false" \
"PIPELINE:false" \
"QOFF:0" \
"QUEUE_DEPTH:3" \
"RDIMM_LAT:0" \
"READ_BURST_TYPE:SEQUENTIAL" \
"ROW_ADDR_WIDTH:16" \
"RTT_NOM:DISABLED" \
"RTT_WR:OFF" \
"SDRAM_NB_RANKS:1" \
"SDRAM_NUM_CLK_OUTS:1" \
"SDRAM_TYPE:DDR3" \
"SELF_REFRESH_TEMPERATURE:NORMAL" \
"SHIELD_ENABLED:true" \
"SIMULATION_MODE:FAST" \
"TDQS_ENABLE:DISABLE" \
"TGIGEN_ADD_PRESET_WIDGET:true" \
"TIMING_DH:150" \
"TIMING_DQSCK:400" \
"TIMING_DQSQ:200" \
"TIMING_DQSS:0.25" \
"TIMING_DS:75" \
"TIMING_DSH:0.2" \
"TIMING_DSS:0.2" \
"TIMING_FAW:30" \
"TIMING_IH:275" \
"TIMING_INIT:200" \
"TIMING_IS:200" \
"TIMING_MODE:0" \
"TIMING_MRD:4" \
"TIMING_QH:0.38" \
"TIMING_QSH:0.38" \
"TIMING_RAS:36" \
"TIMING_RC:49.5" \
"TIMING_RCD:13.5" \
"TIMING_REFI:7.8" \

```

```

"TIMING_RFC:350" \
"TIMING_RP:13.5" \
"TIMING_RRD:7.5" \
"TIMING_RTP:7.5" \
"TIMING_WR:15" \
"TIMING_WTR:5" \
"TURNAROUND_RTR_DIFFRANK:1" \
"TURNAROUND_RTW_DIFFRANK:1" \
"TURNAROUND_WTR_DIFFRANK:1" \
"TURNAROUND_WTW_DIFFRANK:0" \
"USER_POWER_DOWN:false" \
"USER_SELF_REFRESH:false" \
"WIDTH:16" \
"WRITE_LEVELING:ENABLE" \
"WRITE_RECOVERY:5" \
"ZQ_CALIB_PERIOD:200" \
"ZQ_CALIB_TYPE:0" \
"ZQ_CALIB_TYPE_TEMP:0" \
"ZQ_CAL_INIT_TIME:512" \
"ZQ_CAL_L_TIME:256" \
"ZQ_CAL_S_TIME:64" } -inhibit_configurator 0
# Exporting core PF_DDR3_SS to TCL done

```

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 {\
"CORE:PF_TX_PLL" \
"FAMILY:26" \
"INIT:0x0" \
"PARAM_IS_FALSE:false" \
"SD_EXPORT_HIDDEN_PORTS:false" \
"TxPLL_AUX_LOW_SEL:true" \
"TxPLL_AUX_OUT:125" \
"TxPLL_CLK_125_EN:true" \
"TxPLL_DYNAMIC_RECONFIG_INTERFACE_EN:false" \
"TxPLL_EXT_WAVE_SEL:0" \
"TxPLL_FAB_LOCK_EN:false" \
"TxPLL_FAB_REF:200" \
"TxPLL_JITTER_MODE_SEL:10G SyncE 32Bit" \
"TxPLL_MODE:NORMAL" \
"TxPLL_OUT:2500.000" \
"TxPLL_REF:100" \
"TxPLL_SOURCE:DEDICATED" \
"TxPLL_SSM_DEPTH:0" \

```

```

    "TxPLL_SSM_DIVVAL:1" \
    "TxPLL_SSM_DOWN_SPREAD:false" \
    "TxPLL_SSM_FREQ:64" \
    "TxPLL_SSM_RAND_PATTERN:0" \
    "VCOFREQUENCY:1600" } -inhibit_configurator 1
# Exporting core PCIe_TX_PLL to TCL done

```

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.

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.

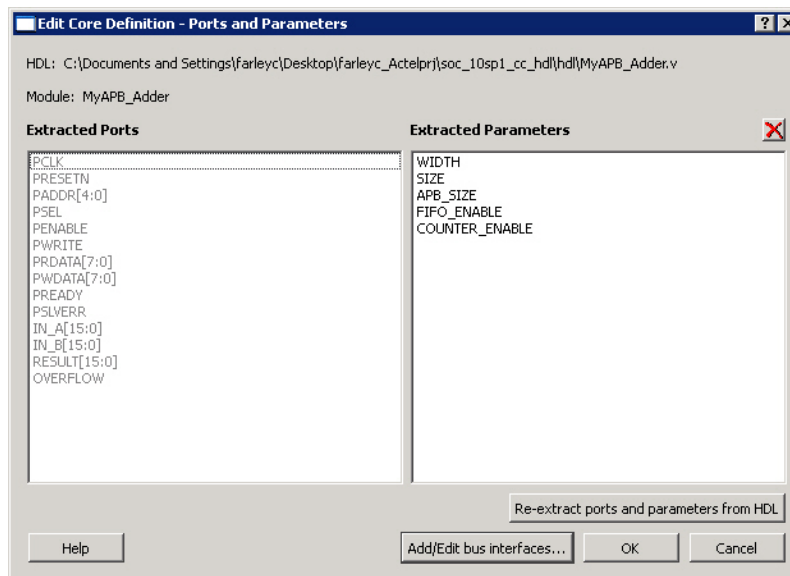


Figure 25 · Edit Core Definition - Ports and Parameters Dialog Box

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.

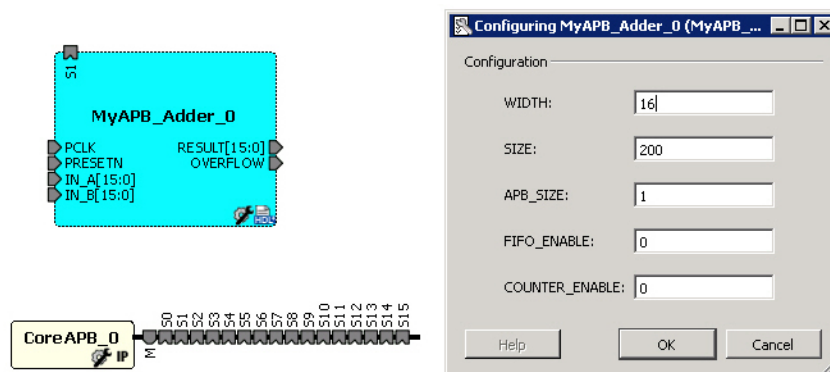


Figure 26 · HDL+ Instance and Configuration Dialog Box

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.

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 (*.vhd/*.vhd) files can be imported.

Mixed-HDL Support in Libero SoC

You must have ModelSim PE or SE 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 (such as the post-layout netlist for simulation) 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.

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.

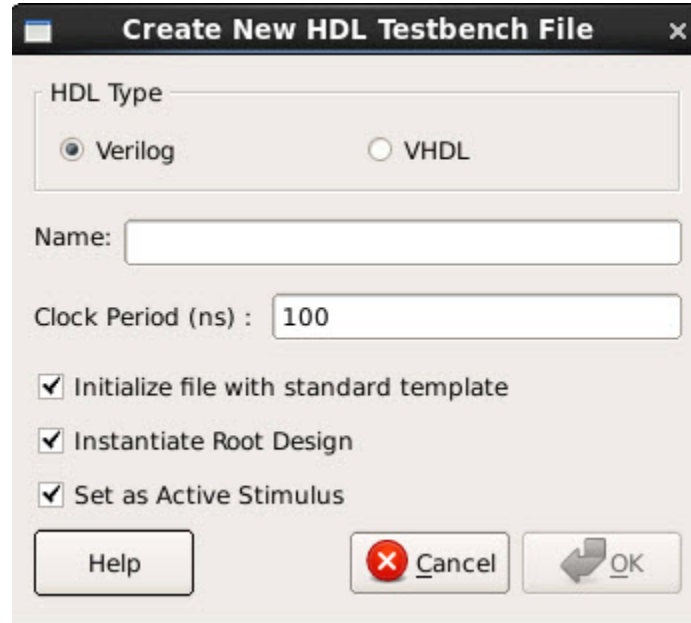


Figure 27 · Create New HDL Testbench File Dialog Box

```

1 -----
2 -- Company: <Name>
3 --
4 -- File: hdl_testbench_1.vhd
5 -- File history:
6 --     <Revision number>: <Date>: <Comments>
7 --     <Revision number>: <Date>: <Comments>
8 --     <Revision number>: <Date>: <Comments>
9 --
10 -- Description:
11 --
12 -- <Description here>
13 --
14 -- Targeted device: <Family::SmartFusion> <Die::A2F200M3F> <Package::484 FBGA>
15 -- Author: <Name>
16 --
17 -----
18
19
20 library ieee;
21 use ieee.std_logic_1164.all;
22
23 entity hdl_testbench_1 is
24 end hdl_testbench_1;
25
26 architecture behavioral of hdl_testbench_1 is
27
28     constant SYSCLK_PERIOD : time := 100 ns;
29
30     signal SYSCLK : std_logic := '0';
31     signal NSYSRESET : std_logic := '0';
32
33     component test_mss
34         -- ports
35         port(
36             -- Inputs
37             UART_1_RXD : in std_logic;
38             UART_0_RXD : in std_logic;
39             SPI_1_DI : in std_logic;
40             SPI_0_DI : in std_logic;
41             MAC_CRSDV : in std_logic;
42             MAC_RXER : in std_logic;
43             MSS_RESET_N : in std_logic;
44             CLKA_PAD : in std_logic;
45             CLKC_PAD : in std_logic;
46             MAC_RXD : in std_logic_vector(1 downto 0);
47

```

Figure 28 · HDL Testbench Example - VHDL, Standard Template and Root Design Enabled

Designing with Block Flow

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

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**.

If you want to perform pre-layout simulation with the post-synthesized netlist, in the Design Flow window, under Verify Post-Synthesized Implementation, double-click **Generate Simulation File** and then double-click **Simulate**. The default tool for RTL simulation in Libero SoC is ModelSim™ ME Pro or ModelSim ME. 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.

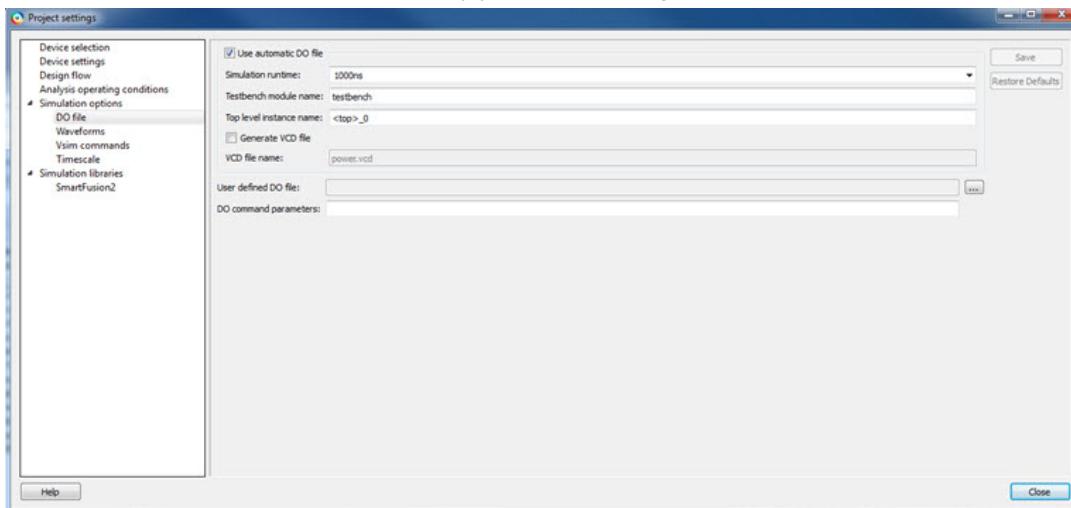


Figure 29 · Project Settings: DO File

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 'add wave /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

- **Post-layout simulation only:**
 - **SDF timing delays** - Select Minimum (Min), Typical (Typ), or Maximum (Max) timing delays in the back-annotated SDF file.
 - **Disable Pulse Filtering during SDF-based Simulations** - When the check box is enabled the **+pulse_int_e/1 +pulse_int_r/1 +transport_int_delays** switch is included with the vsim command for post-layout simulations; the checkbox is disabled by default.
- **Resolution** - The default is family specific (review the dialog box for your default setting), but you can customize it to fit your needs. 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.

Family	Default Resolution
SmartFusion2	1 fs
IGLOO2	1 ps
RTG4	1 ps

- **Additional options** - Text entered in this field is added to the vsim command.
 - **SRAM ECC Simulation (RTG4)** -
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.

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.

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 Simulation 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, 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 μ 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**.

Performing DirectCore Functional Simulation

Libero SoC overwrites all the existing files of the Core when you import a DirectCore project (including testbenches). Save copies of your project stimulus files with new names if you wish to keep them.

You must import a DirectCore BFM file into the Libero SoC in order to complete functional simulation (the BFM is a stimulus file that you can edit to extend the testbench). VEC files are generated automatically from the BFM when you run ModelSim.

The SoC Project Manager overwrites your BFM file if you re-import your project. Edit and save your BFM outside the Libero SoC project to prevent losing your changes. After you re-import your DirectCore project, you can import your modified BFM again.

To perform functional simulation of a DirectCore project:

1. Right-click a stitched module of the DirectCore project and select **Set as root**.
2. 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 μ s and the Wave window opens to display the simulation results.
3. 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.
4. 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.

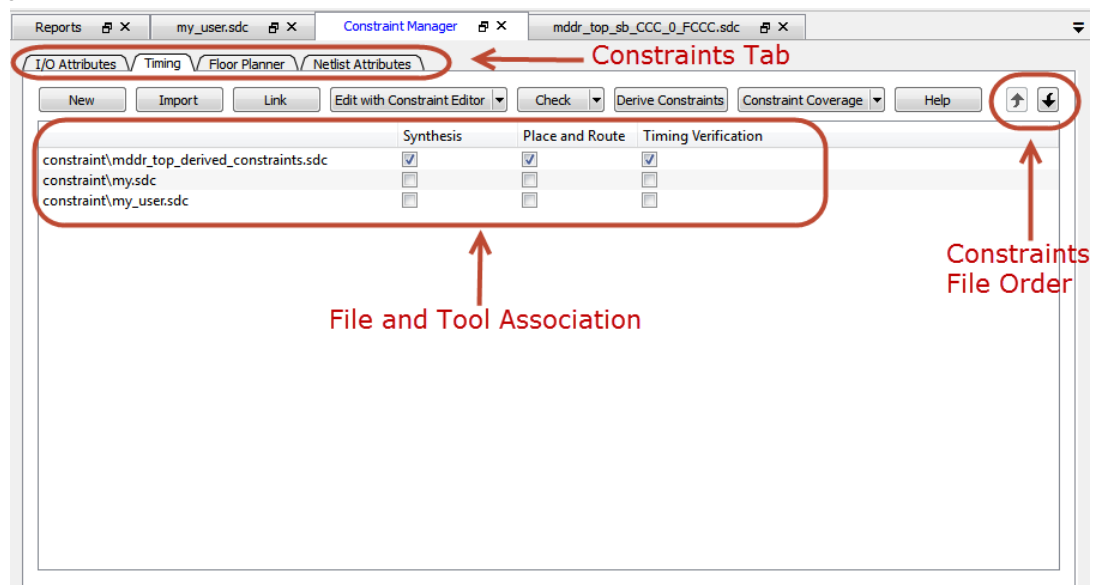


Figure 30 · Constraint Manager

Libero SoC Design Flow

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

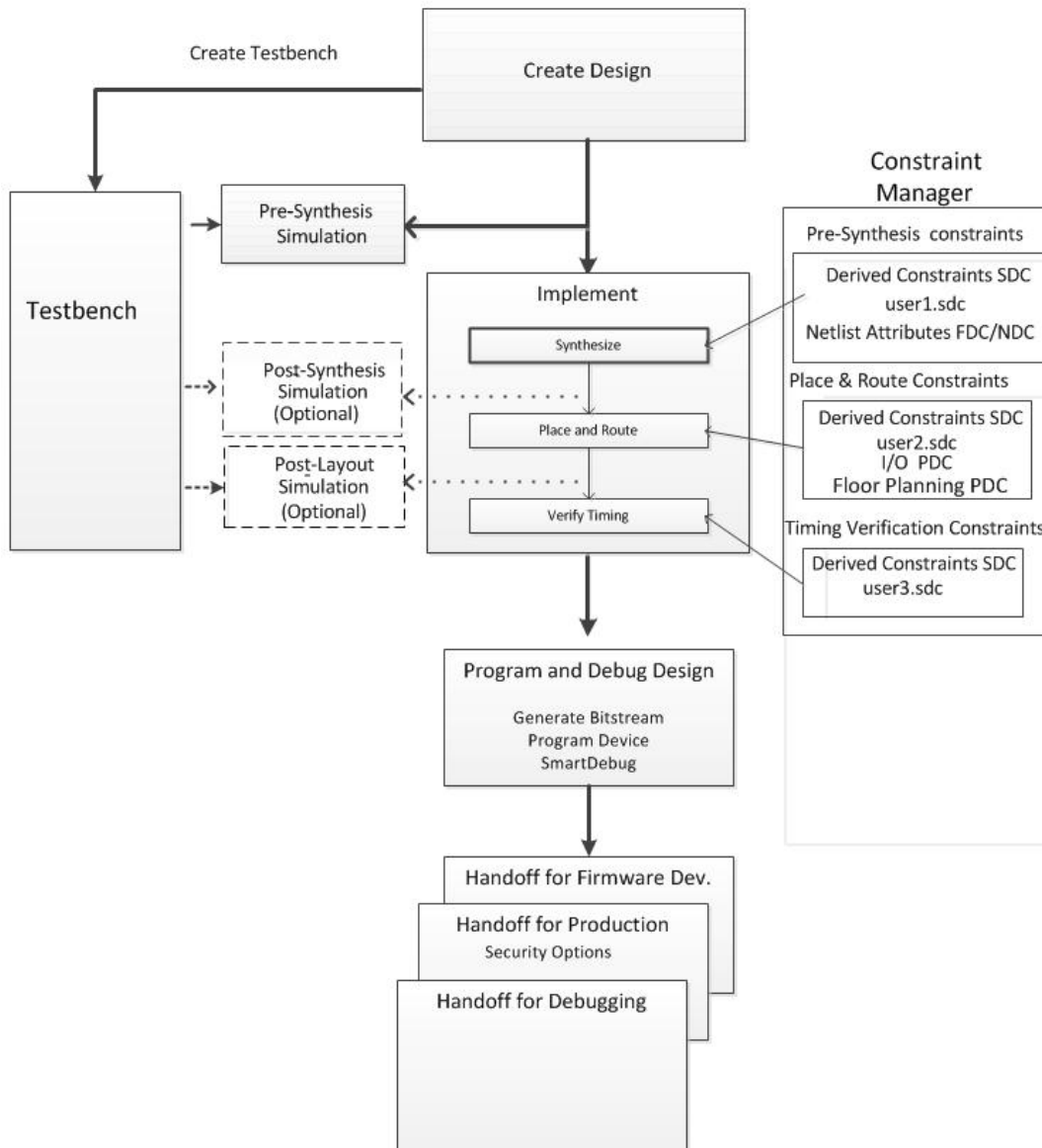


Figure 31 · Constraint Manager in Libero SoC Design Flow

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

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 constraints for Synthesis or Place and Route to guide the tool to focus on a few critical paths. The set of SDC timing constraints associated with Timing Verifications can remain unchanged.

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

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.

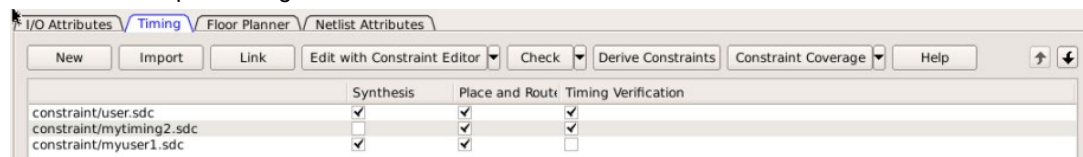




Figure 32 · Constraint File and Tool Association

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

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

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.
 - Chip Planner to edit/create Floorplanning constraints. See [Chip Planner User Guide](#) for details.
 - Constraint Editor to edit/create Timing Constraints. See [Timing Constraints Editor User Guide](#) for details.
3. Create the Constraints in the Interactive Tool. Click **Commit and Save**.
4. Check that Libero SoC creates these files to store the new constraints:
 - Constraints\io\user.pdc file when I/O constraints are added and saved in I/O Editor.
 - Constraints\fp\user.pdc file when floorplanning constraints are added and saved in Chip Planner.
 - Constraints\user.sdc file when Timing Constraints are added and saved in Constraint Editor

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.

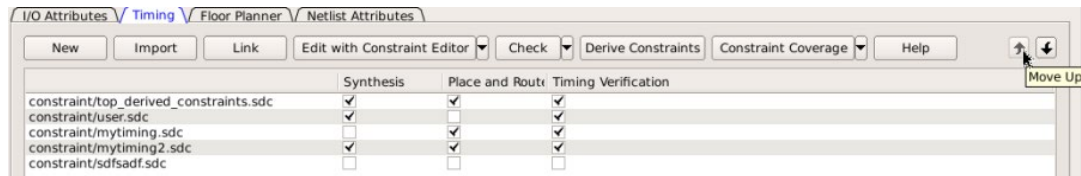


Figure 33 · Move constraint file Up or Down

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.

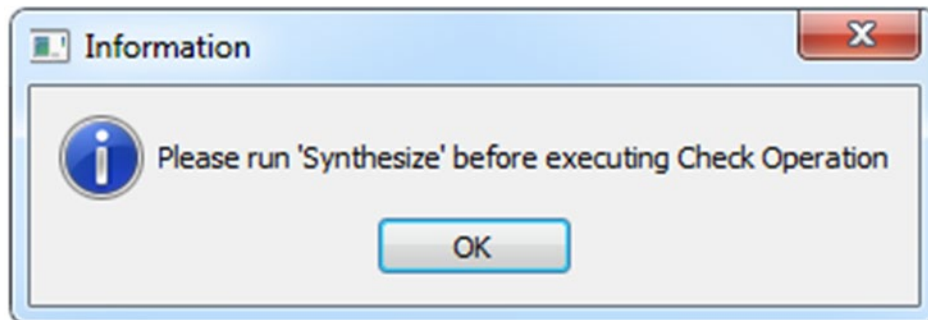
Check a Constraint File

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

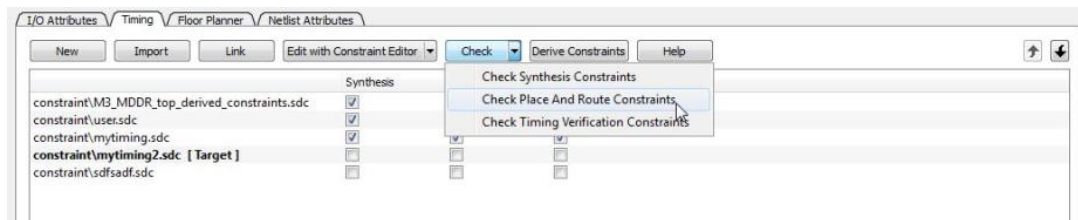


Figure 34 · Check Constraints

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

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

Check Timing Verification Constraints checks only the Constraint Files associated with Timing Verification.

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

- **Check Synthesis Constraints** checks the following files:
 - M3_MDDR_top_derived_constraints.sdc
 - user.sdc
 - mytiming2.sdc
- **Check Place and Route Constraints** checks the following files:
 - M3_MDDR_top_derived_constraints.sdc
 - mytiming.sdc
 - mytiming2.sdc
- **Check Timing Verification Constraints** checks the following files:
 - M3_MDDR_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.

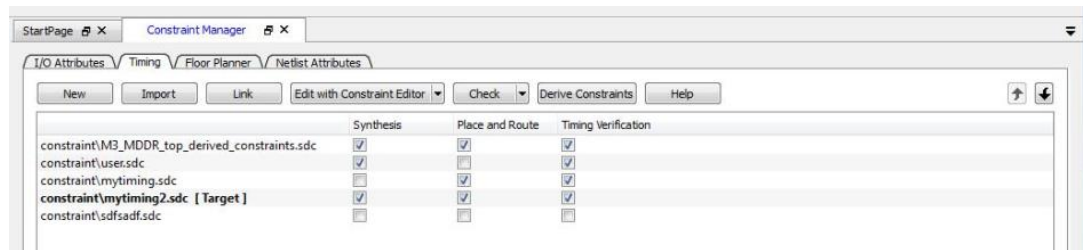


Figure 35 · Timing Constraints SDC file and Tool Association

When a constraint file is checked, the Constraint Manager:

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

Check Result

If the check is successful, this message pops up.

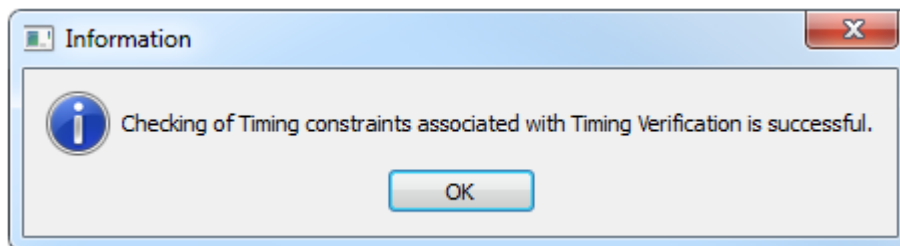


Figure 36 · Check Successful Message

If the check fails, this error message pops up.

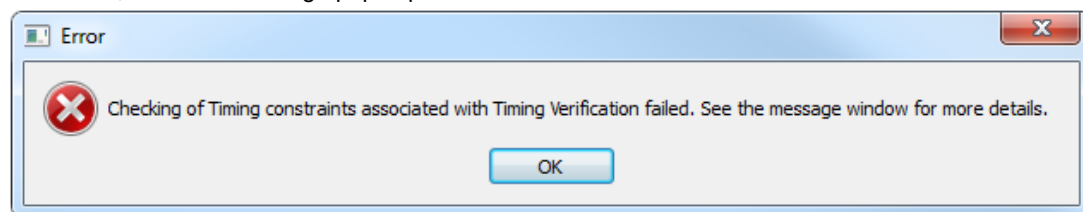


Figure 37 · Check Fails Message

Constraint Type	Check for Tools	Required Design State Before Checks	Netlist Used for Checks	Check Result Details
I/O Constraints	Place and Route	Post-Synthesis	ADL Netlist	Libero Message Window
Floorplanning Constraints	Place and Route	Post-Synthesis	ADL Netlist	Libero Message Window

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

Edit a Constraint File

The Edit button in the Constraint Manager allows you to:

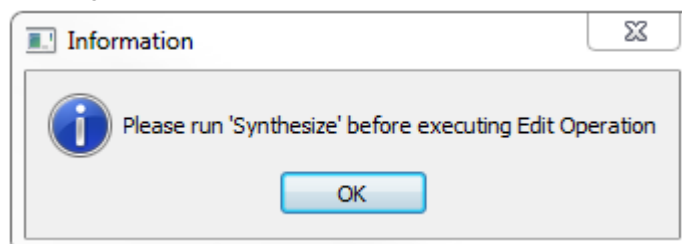
- Create new constraint files. See [To create new constraints from the Constraint Manager using the Text Editor](#) for details.
- Edit existing constraint files.

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)
 - Constraint Editor to edit Timing Constraints (constraints*.sdc). For details, refer to the [Timing Constraints Editor User Guide](#) (Help > Constraints Editor User's Guide)

Note: I/O constraints, Floorplanning constraints, Timing constraints can be edited only when the design is in the proper state. A message pops up if the file is edited when the design state is not proper for edits. If, for example, you open the Constraints Editor (Constraint Manager > Edit) to edit timing constraints when the design state is not post-synthesis, a pop-up message appears.



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

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

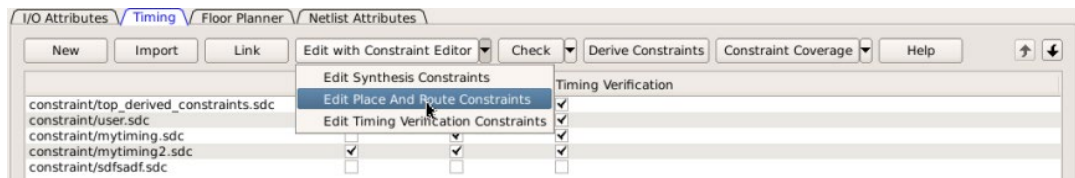


Figure 38 · Edit Drop-down Menu

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

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

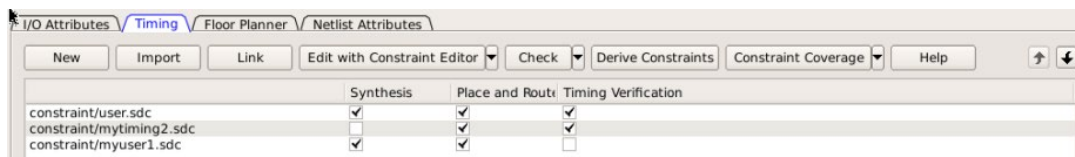


Figure 39 · Timing Constraint File and Tool Association

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

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

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

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

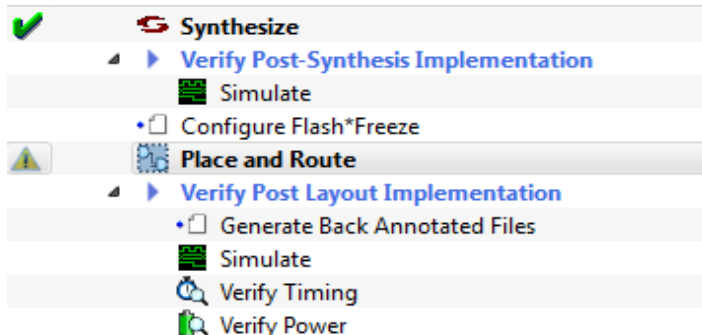


Figure 40 · Place and Route Invalidation

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.

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

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** - The following options are available in the drop-down:
 - **Edit** – Opens the I/O Editor tool to modify the I/O PDC file(s) associated with the Place and Route tool.
 - **Edit with I/O Advisor** – Opens the I/O Advisor tool to modify the I/O attributes in the I/O PDC file(s). This tool helps in reducing power consumption while meeting timing constraints.
- **View** - Opens the I/O Editor tool to view the I/O PDC file(s) associated with the Place and Route tool. You cannot save/commit any changes made to the constraints file. However, you can export the PDC file(s) using the I/O Editor.
- **Check** – Checks the legality of the PDC file(s) associated with the Place and Route tool against the gate level netlist.

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

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

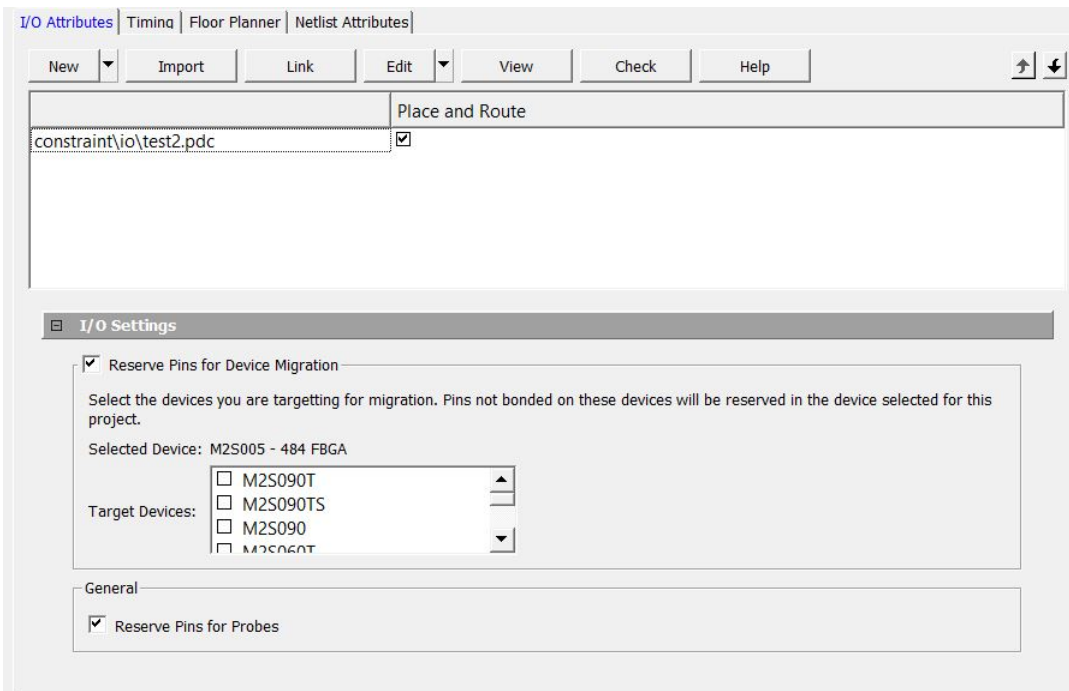


Figure 41 · Constraint Manager – I/O Attributes Tab

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

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

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.

IO Advisor (SmartFusion2, IGLOO2, and RTG4)

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

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

To open the IO Advisor from the Design Flow window, right-click Manage Constraints, select Open Manage Constraints View, select the I/O Attributes tab, select Edit with I/O Advisor (**Design Flow window > Manage Constraints > Open Manage Constraints View > I/O Attributes > Edit > Edit with I/O Advisor**).

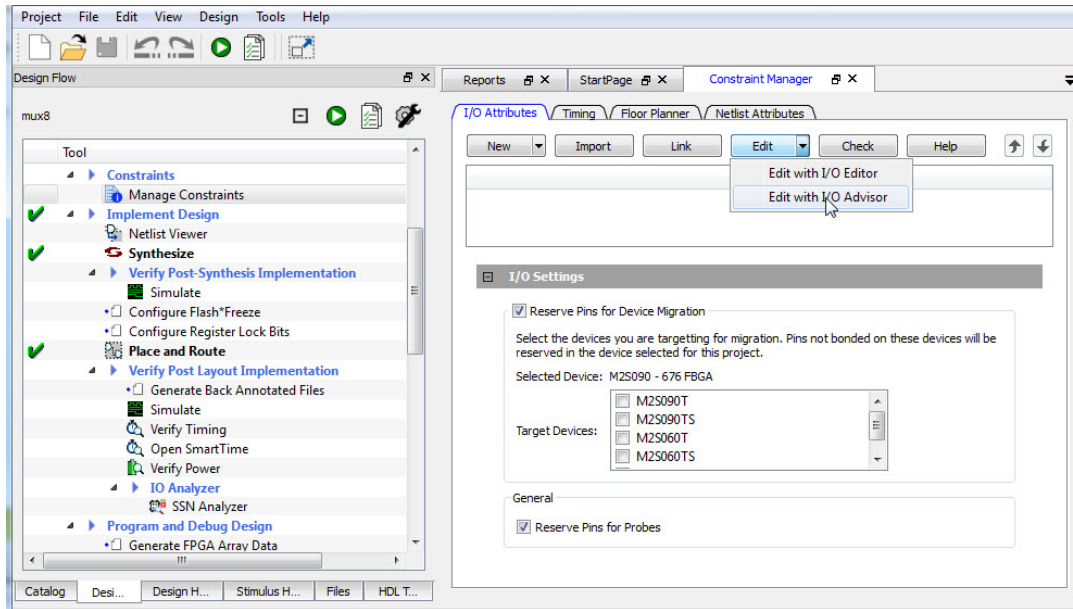


Figure 42 · I/O Advisor

Introduction

The Introduction screen provides general information about the IO Advisor.

The introduction screen provides the navigational panel for you to navigate to the following panels:

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

All steps in the IO Advisor are optional.

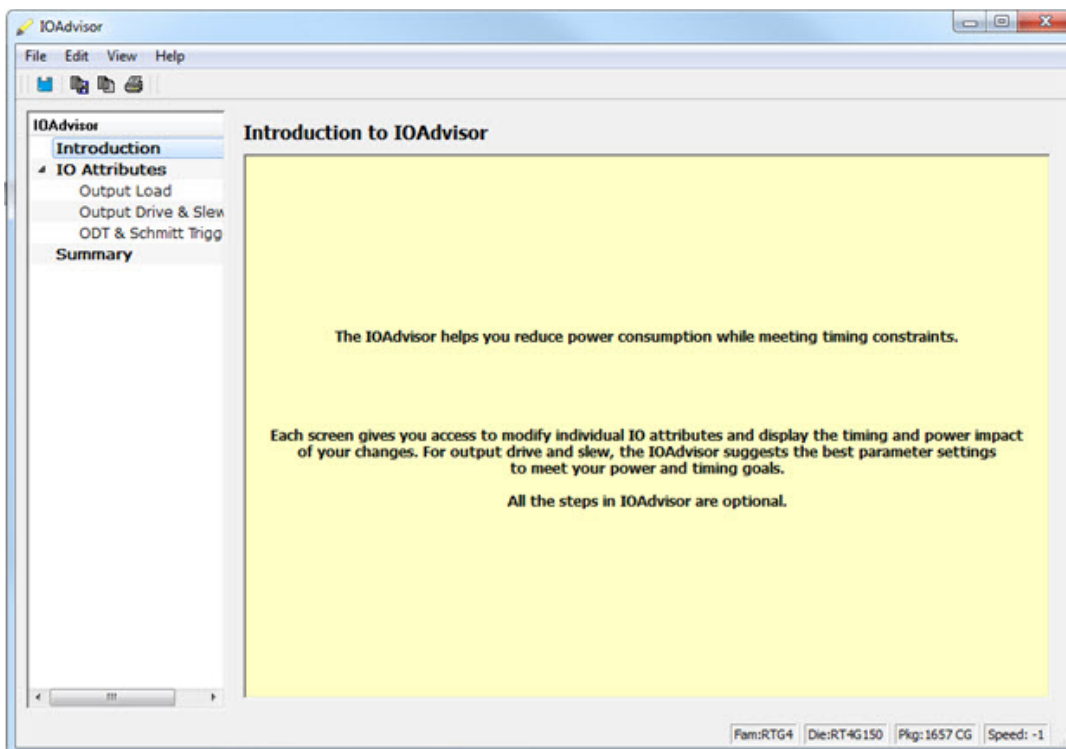


Figure 43 · IO Advisor - Introduction

Output Load

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

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

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

Status	Port	Direction	Bank	IO Standard	State	Output Load(pF)	Power (mW)	Power Change(%)	Delay (ns)	Slack (ns)
✓	FDDR_BA[0]	Output	Bank0 - DQRO	SSTL18I	Initial	5	1636.44	+0.00	3.005	...
✓	FDDR_BA[1]	Output	Bank0 - DQRO	SSTL18I	Current	5	1636.44	+0.00	3.008	...
✓	FDDR_BA[2]	Output	Bank0 - DQRO	SSTL18I	Initial	5	1636.44	+0.00	3.005	...
✓	FDDR_BA[3]	Output	Bank0 - DQRO	SSTL18I	Current	5	1636.44	+0.00	3.008	...
✓	FDDR_CAS_N	Output	Bank0 - DQRO	SSTL18I	Initial	5	1636.44	+0.00	3.005	...
✓	FDDR_CKE	Output	Bank0 - DQRO	SSTL18I	Current	5	1636.44	+0.00	3.008	...
✓	FDDR_CLK	Output	Bank0 - DQRO	SSTL18I	Initial	5	646.55	+0.00	2.951	...
✓	FDDR_CS_N	Output	Bank0 - DQRO	SSTL18I	Current	5	646.55	+0.00	2.951	...
✓	FDDR_CS_N	Output	Bank0 - DQRO	SSTL18I	Initial	5	1636.44	+0.00	3.008	...
✓	FDDR_CS_N	Output	Bank0 - DQRO	SSTL18I	Current	5	1636.44	+0.00	3.008	...
✓	FDDR_DM_RDQ_	Inout	Bank0 - DQRO	SSTL18I	Initial	5	75444.69	+0.00	3.005	...
✓	FDDR_DM_RDQ_	Inout	Bank0 - DQRO	SSTL18I	Current	5	75444.69	+0.00	3.005	...
✓	FDDR_DQ[0]	Inout	Bank0 - DQRO	SSTL18I	Initial	5	138928.39	+0.00	2.951	...
✓	FDDR_DQ[1]	Inout	Bank0 - DQRO	SSTL18I	Current	5	138928.39	+0.00	2.951	...
✓	FDDR_DQ[2]	Inout	Bank0 - DQRO	SSTL18I	Initial	5	138928.39	+0.00	2.951	...
✓	FDDR_DQ[3]	Inout	Bank0 - DQRO	SSTL18I	Current	5	138928.39	+0.00	2.951	...
✓	FDDR_DQ[4]	Inout	Bank0 - DQRO	SSTL18I	Initial	5	1636.44	+0.00	3.008	...
✓	FDDR_DQ[5]	Inout	Bank0 - DQRO	SSTL18I	Current	5	1636.44	+0.00	3.008	...
✓	FDDR_DQ[6]	Inout	Bank0 - DQRO	SSTL18I	Initial	5	75444.69	+0.00	3.005	...
✓	FDDR_DQ[7]	Inout	Bank0 - DQRO	SSTL18I	Current	5	75444.69	+0.00	3.005	...
✓	FDDR_DQ[8]	Inout	Bank0 - DQRO	SSTL18I	Initial	5	75444.69	+0.00	3.005	...
✓	FDDR_DQ[9]	Inout	Bank0 - DQRO	SSTL18I	Current	5	75444.69	+0.00	3.005	...

Figure 44 · IO Advisor - Output Load Panel

Search and Regular Expressions

To search for a specific Port, enter the Port Name in the Port Name Search field and click Search. Regular expressions are accepted for the search. All Port Names matching the regular expression are displayed. The

regular expression “FDDR*”, for example, results in all the output ports beginning with FDDR in the Port Name appearing in the display.

Set Output Load

Port Name: FDDR* Search Sort By: Initial




	Status	Port	Direction	Bank	IO Standard	State	Output Load(pF)	Power (uW)	Power Change(%)	Delay (ns)	Slack (ns)
16	✓	FDDR_ADOR[9]	Output	Bank0 - DORIO	SSTL18I	Initial	5	1656.44	+0.00	3.008	---
17	✓	FDDR_BA[0]	Output	Bank0 - DORIO	SSTL18I	Current	5	1656.44	+0.00	3.005	---
18	✓	FDDR_BA[1]	Output	Bank0 - DORIO	SSTL18I	Initial	5	1656.44	+0.00	3.008	---
19	✓	FDDR_BA[2]	Output	Bank0 - DORIO	SSTL18I	Initial	5	1656.44	+0.00	3.005	---
20	✓	FDDR_CAS_N	Output	Bank0 - DORIO	SSTL18I	Initial	5	1656.44	+0.00	3.008	---
21	✓	FDDR_CKE	Output	Bank0 - DORIO	SSTL18I	Current	5	1656.44	+0.00	3.005	---
22	✓	FDDR_CLK	Output	Bank0 - DORIO	SSTL18I	Initial	5	646.55	+0.00	2.951	---
23	✓	FDDR_CS_N	Output	Bank0 - DORIO	SSTL18I	Current	5	1656.44	+0.00	3.008	---
24	ⓘ	FDDR_DM_RDQ...	Inout	Bank0 - DORIO	SSTL18I	Initial	5	75444.69	+0.00	3.005	---
25	ⓘ	FDDR_DM_RDQ...	Inout	Bank0 - DORIO	SSTL18I	Current	5	75444.69	+0.00	3.005	---
26	ⓘ	FDDR_DQ[0]	Inout	Bank0 - DORIO	SSTL18I	Initial	5	138928.39	+0.00	2.951	---
27	ⓘ	FDDR_DQ[1]	Inout	Bank0 - DORIO	SSTL18I	Initial	5	138928.39	+0.00	2.951	---
28	✓	FDDR_DQ[5_TM...	Output	Bank0 - DORIO	SSTL18I	Current	5	1656.44	+0.00	3.008	---
29	✓	FDDR_DQ[0]	Inout	Bank0 - DORIO	SSTL18I	Initial	5	75444.69	+0.00	3.005	---
30	ⓘ	FDDR_DQ[0]	Inout	Bank0 - DORIO	SSTL18I	Current	5	75444.69	+0.00	3.005	---

Set Output Load Restore Initial Value Select: All None

Figure 45 · Search Field and Regular Expressions

Status Column

The icon in the Status Column displays the status of the Output Port.

Icon	Status and Explanation
	OK - The IO attributes match the suggestion in Output Drive and Slew Table.
	Error – The Timing constraints for this IO are not met in Output Drive and Slew Table.
	Information – you can improve the power and/or timing of the IO by applying the suggestion in Output Drive and Slew Table.

Column Display and Sorting

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

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

Set Output Load

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

Restore Initial Value

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

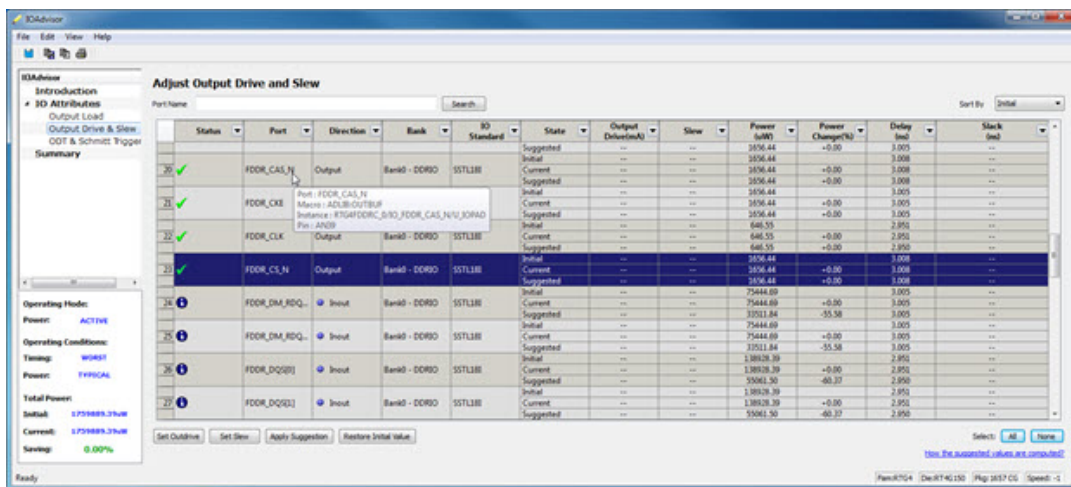
Output Drive and Slew

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

The display can be sorted according to the initial, current or suggested values. To change the sorting, click the Sort By drop-down menu to make your selection.

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

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



Status	Part	Direction	Bank	IO Standard	State	Output Drive (mA)	Slew	Power (uW)	Power Change (%)	Delay (ns)	Slack (ns)
20	FDDR_CAS_N	Output	Bank0 - DQSDI	5STL138	Suggested	3056.44	-0.00	3.008	...
21	FDDR_CKE	Output	Bank0 - DQSDI	5STL138	Suggested	3056.44	-0.00	3.008	...
22	FDDR_CLK	Output	Bank0 - DQSDI	5STL138	Suggested	3056.44	-0.00	3.008	...
23	FDDR_CS_N	Output	Bank0 - DQSDI	5STL138	Suggested	3056.44	-0.00	3.008	...
24	FDDR_DM_RDQ	Inout	Bank0 - DQSDI	5STL138	Suggested	3056.44	-0.00	3.008	...
25	FDDR_DM_RDQ	Inout	Bank0 - DQSDI	5STL138	Suggested	3056.44	-0.00	3.008	...
26	FDDR_DQSDI	Inout	Bank0 - DQSDI	5STL138	Suggested	3056.44	-0.00	3.008	...
27	FDDR_DQSDI	Inout	Bank0 - DQSDI	5STL138	Suggested	3056.44	-0.00	3.008	...

Figure 46 · IO Advisor – Output Drive and Slew

How the Suggested Values Are Computed

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

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

In this screen, you can change the drive strength and slew of the design output I/Os. Select the out drive and/or the slew current value cell. Click the cell to open the combo box. Choose the value you want from the set of valid values. You can restore the initial values by clicking **Restore Initial Value**.

To make changes to multiple I/Os, select multiple I/Os (Control+click), click **Set Slew** or **Set Outdrive**, select the value, and click **OK**.

Apply Suggestion

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

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

Adjust ODT and Schmitt Trigger

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

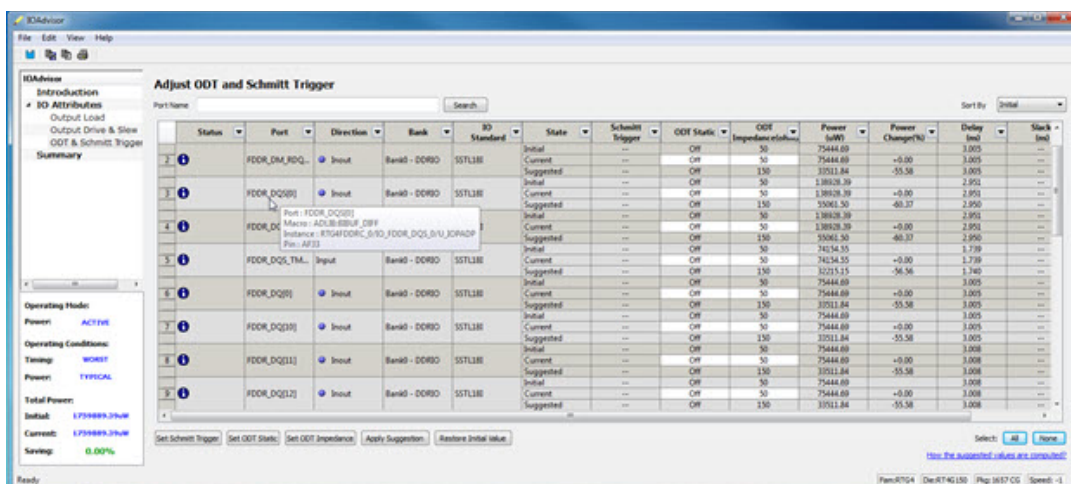


Figure 47 · IO Advisor – Adjust ODT and Schmitt Trigger

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

Search and Regular Expressions

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

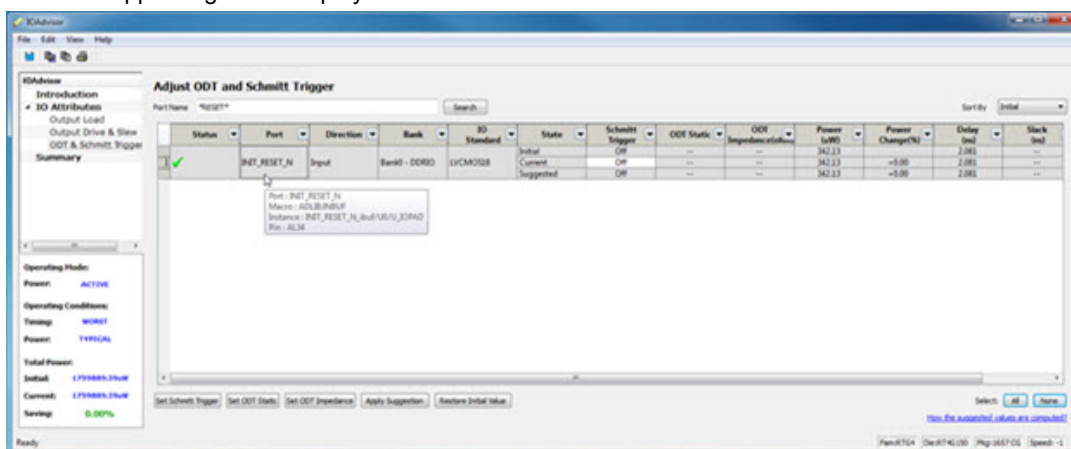





Figure 48 · Search Field and Regular Expressions

Status Column

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

Icon	Status and Explanation
	OK - The IO attributes match the suggestion in the Adjust ODT and Schmitt Trigger Table.
	Error – The Timing constraints for this IO are not met in the Adjust ODT and Schmitt Trigger Table.

Icon	Status and Explanation
	Information – you can improve the power and/or timing of the IO by applying the suggestion in the Adjust ODT and Schmitt Trigger Table.

Column Display and Sorting

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

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

Set Schmitt Trigger

For IO Standards that support the Schmitt Trigger, you can turn the Schmitt Trigger On or Off. Select the IO and click **Set Schmitt Trigger** to toggle on or off. Your setting is displayed in the Schmitt Trigger column for the IO.

Set ODT Static

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

- On – The Termination resistor for impedance matching is located inside the chip.
- Off – The Terminator resistor for impedance matching is located on the printed circuit board.

To turn the ODT Static on or off, click to select the input/inout port and from the pull-down menu, toggle on or off. You can also turn ODT Static on or off by clicking **Set ODT Static** and toggling on or off.

Set ODT Impedance (Ohm)

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

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

Apply Suggestion

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

Restore Initial Value

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

Summary of Changes

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

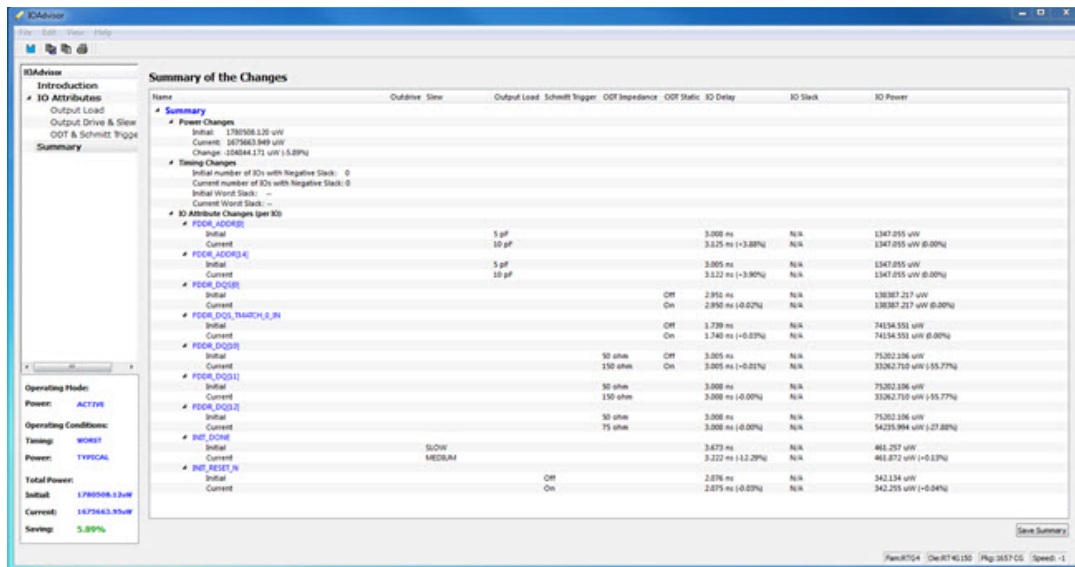


Figure 49 · IO Advisor – Summary

You can save the summary by clicking **Save Summary**, selecting the save format (text or CSV), and clicking **OK**. To commit IO Attribute changes you have made to the database (the `*io_pdc` file), choose **Save** from the File Menu (**File > Save**). Click **OK** in the dialog that appears.

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

Constraint Manager – Timing Tab

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

Available actions are:

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

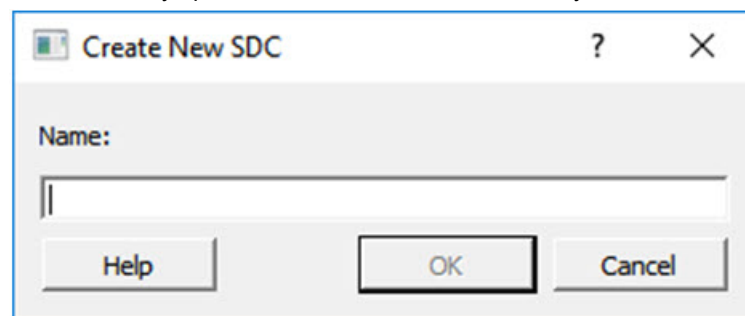


Figure 50 · Create New SDC Dialog Box

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

- o **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.
- o **Timing Verification** - When selected, the timing SDC file(s) associated with the Timing Verification tool is loaded in the constraints editor for editing.

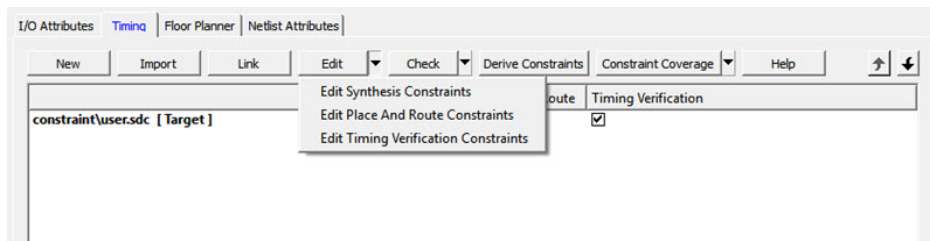


Figure 51 · Timing Constraints Edit Options in Constraint Manager

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

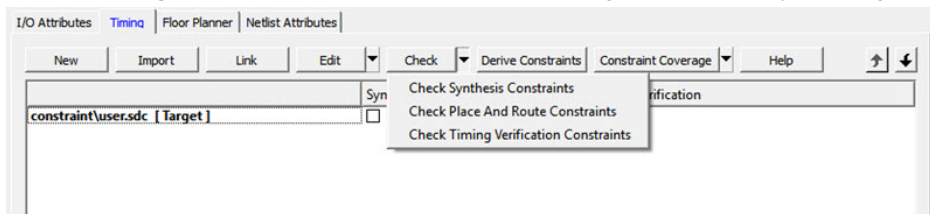


Figure 52 · Timing Constraints Check Options in Constraint Manager

- **Derive Constraints** – When clicked, Libero generates a timing SDC file based on user configuration of IP core, components and component SDC. For Smartfusion2, Design components for which Libero SoC generates timing constraints include MSS, OSC, SERDES and CCC. 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.

Note: Clicking this button also generates a PDC file for certain Microsemi IPs, such as the the CoreConfigP. This PDC file constrains the placement of the IPs in a fixed region automatically created by Libero SoC to ensure that these IPs do not cause timing violations. The PDC file is named <top_level>_derived_constraints.pdc and is displayed in the Floor Planner tab. Associate this PDC file to Place-and-Route.

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

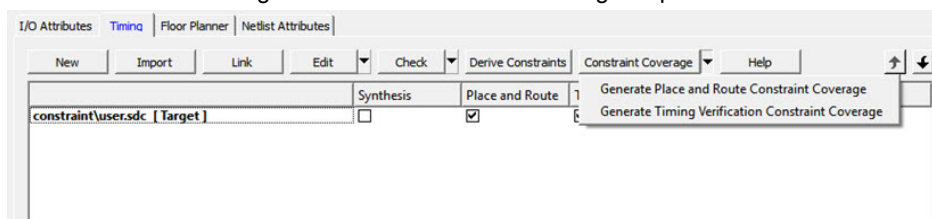


Figure 53 · Constraint Coverage Options for Timing Constraints in Constraint Manager

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

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

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

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

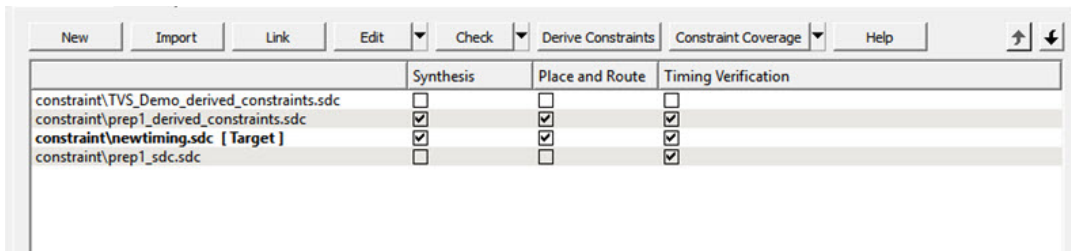


Figure 54 · Constraint Manager – Timing Tab

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

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

File and Tool Association

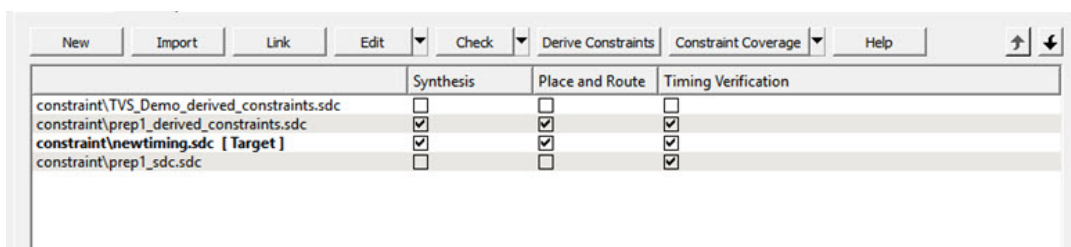
Each timing constraint file can be associated or disassociated with any one, two, or all three of the following tools:

- Synthesis
- Place and route
- Timing Verification

Click the 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 1

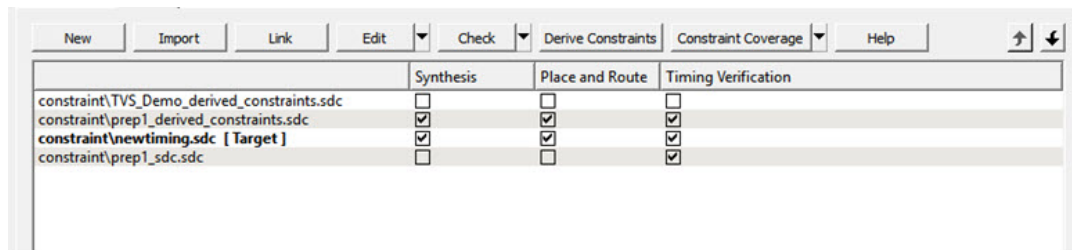


In the context of the graphic above, when Edit Synthesis Constraint is selected, the prep1_derived_constraint.sdc file and the newtiming.sdc file will be read (because these two files are associated to Synthesis). The TVS_Demo_derived_constraints.sdc file and prep1_sdc.sdc file are not read (because they are not associated to

Synthesis). When the SmartTime Constraints Editor opens for edit, the prep1_derived_constraint.sdc file and the newtiming.sdc file are read and loaded into the Constraints Editor. Any changes you made (to the constraints in these two files) and saved in the Constraints Editor will be written back to the two files.

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.

Example 2



In the context of the graphic above, when Check Synthesis Constraint is selected, the prep1_derived_constraints.sdc file and newtiming.sdc will be checked (because these two files are associated to Synthesis) and TVS_Demo_derived_constraints.sdc and prep1_sdc.sdc are not checked (because they are not associated to Synthesis).

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

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, SmartDesign-based, and System Builder-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.

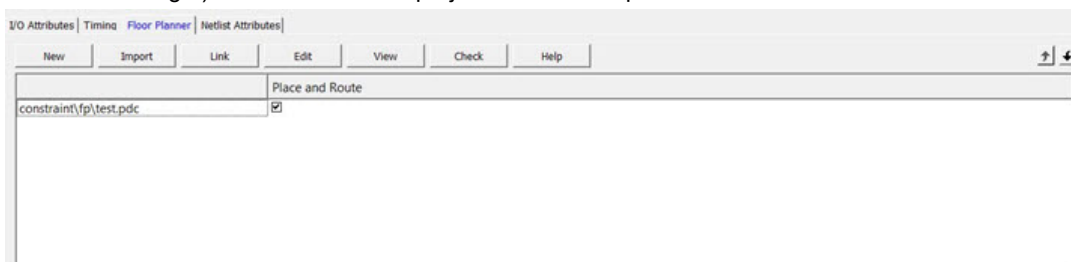


Figure 55 · Constraint Manager – Floor Planner Tab

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

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

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 existing FDC or NDC netlist attribute constraints file (located and maintained outside of the Libero SoC project).
- **Check** – Checks the legality of the FDC and NDC file(s) associated with the Synthesis or Compile tools.

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

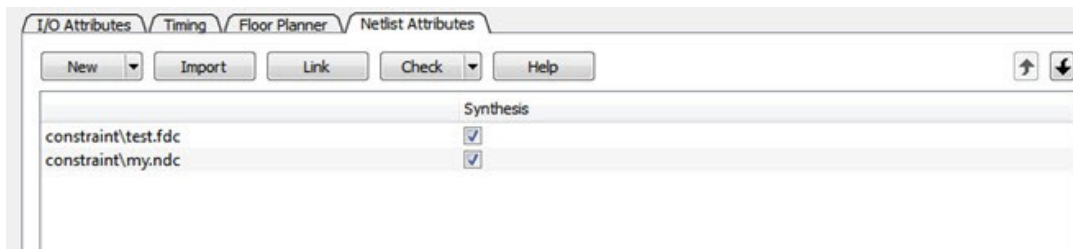


Figure 56 · Constraint Manager – Netlist Attributes Tab

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

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

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.

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.

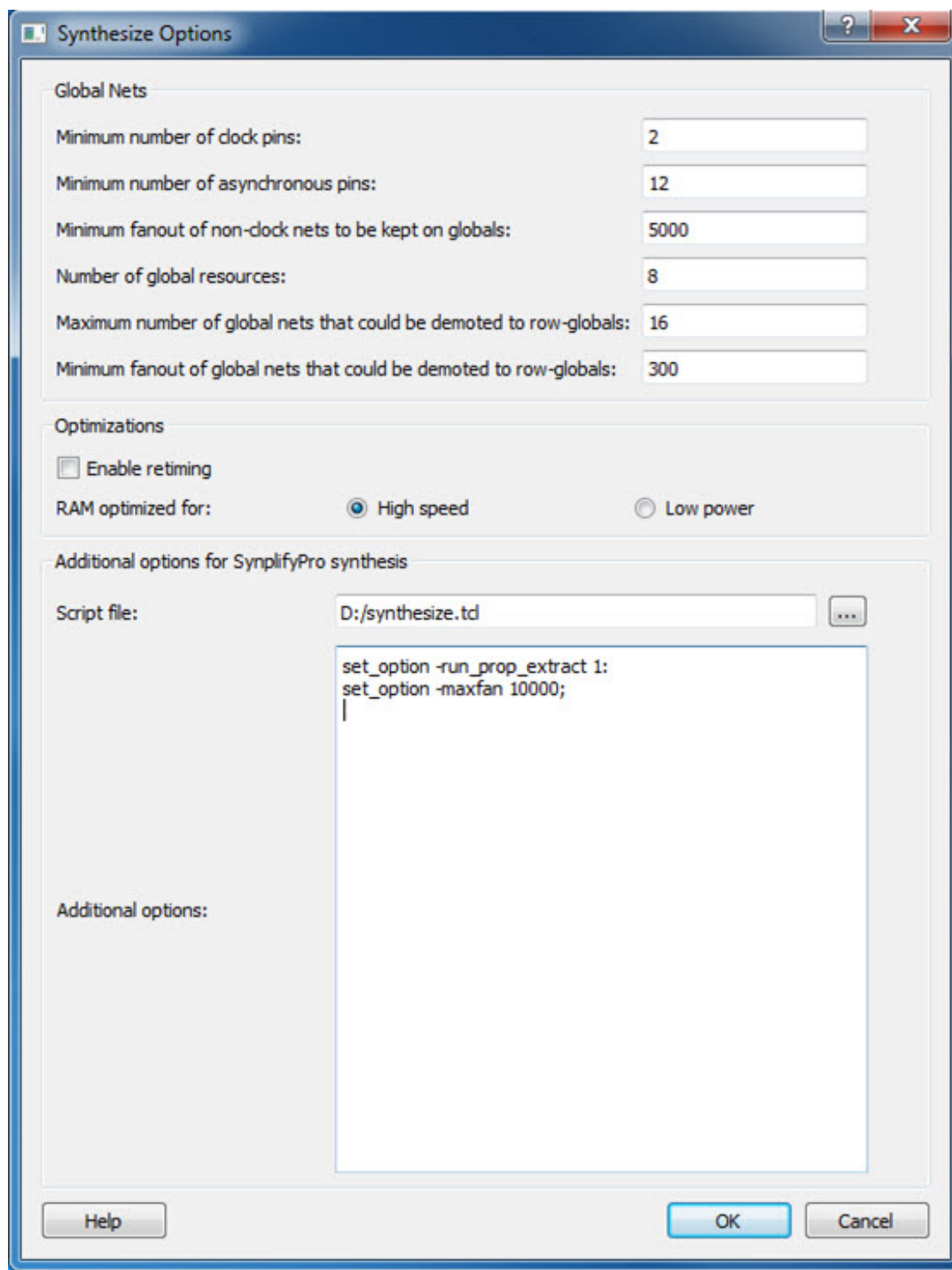


Figure 57 · Synthesize Options Dialog Box

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 12 for all dies of SmartFusion2 and IGLOO2 families and RT4G150 die of the RTG4 family. This option is not available for RT4G150_ES die.
- **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.
- **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.


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.

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.

```
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

```

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:

```

add_file <*>
impl <*>
project_folder <*>
add_folder <*>
constraint_file <*>
project <*>
project_file <*>
open_file <*>
set_option -part
set_option -package
set_option -speed_grade
set_option -top_module
set_option -technology
set_option -opcond
set_option -vlog_std
set_option -vhdl2008
set_option -disable_io_insertion
set_option -async_globalthreshold
set_option -clock_globalthreshold
set_option -globalthreshold

```

```
set_option -low_power_ram_decomp
set_option -retiming
```

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 :

```
/* 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 EDN Netlist file.
7. From the Instrumentor UI specify the sample clock, the breakpoints, and other signals to probe. Synplify creates a new synthesis implementation. Synthesize the design.
8. In Libero SoC, run Synthesis, Place and Route and Generate a Programming File.
Note: Libero SoC works from the edif netlist of the current active implementation, which is the implementation you created in Synplify for Identify debug.
9. Double-click **Identify Debug Design** in the Design Flow window to launch the Identify Debugger.

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.

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 μ s 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**.

Configure Flash*Freeze

Opens the Flash*Freeze Hardware Settings dialog box. For more information on the Flash*Freeze mode see the [SmartFusion2 and IGLOO2 Low Power Design User's Guide](#).

The fabric SRAMs can be put into a Suspend Mode or a Sleep Mode. This applies to both the Large SRAM (LSRAM) instances of RAM1xK18 and the Micro SRAM (uSRAM) instances of RAM64x18. These SRAMs are grouped in rows in Libero® System-on-Chip (SoC) devices

uRAM/LSRAM State

Sleep - Sets to Sleep; LSRAM and uSRAM contents are not retained.

Suspend - Sets to Suspend; LSRAM and uSRAM contents are retained.

MSS Clock Source

The lower the frequency the lower the power will be. But for some peripherals that can remain active (such as SPI or MMUART), you may need a higher MSS clock frequency (such as to meet the baud rate for MMUART).

Options are:

- On-Chip 1 MHz RC Oscillator
- On-Chip 50 MHz RC Oscillator

Configure Register Lock Bits

For SmartFusion2 and IGLOO2 devices, use the Register Lock Bits Configuration tool to lock MSS, SERDES, and FDDR configuration registers and prevent them from being overwritten by Masters that have access to these registers. The register lock bits are set in a text file (*.txt) and imported into a SmartFusion2/IGLOO2 project. From the Design Flow window, click **Configure Register Lock Bits (Design Flow window > Configure Register Lock Bits)** to open the Register Lock Bits Setting dialog box. Click the **Browse** button to navigate to a text file (*.txt) that contains the Register Lock Bit settings.



Figure 58 · Register Lock Bits Settings Dialog Box

Register Lock Bit Text File Template

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

The file is named <proj_location>/designer/<root>/<root>_init_config_lock_bits.txt. This is the initial and the default Lock Bit Configuration File. Use this file as a template to make changes. Modify it to ensure that the lock bits are set to "0" for all register bits you want to lock. Save the file as a *.txt file with a different name and import the file into the project using the Register Lock Bit Settings dialog box (**Design Flow window > Configure Register Lock Bits**).

Register Lock Bit File Syntax

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

- If the lock bit is for a register the parameter name is defined as:
 - <Physical block name>_<register name>_LOCK
- If the lock bit is for a field the parameter name is defined as:
 - <Physical block name>_<register name>_<field name>_LOCK
- The physical block name (which may vary with device family and die) is defined as:
 - MSS
 - FDDR
 - SERDES_IF_x (where x is 0,1,2,3 to indicate the physical SERDES location) for SmartFusion2, and IGLOO2 (010/025/050/150) devices
 - SERDES_IF2 (060/090) devices (only one SERDES block per device for SmartFusion2 and IGLOO2 devices)

Set the lock bit value to '1' to indicate that the register can be written, "0" to indicate that the register cannot be written (locked).

Lines starting with "#" or ";" are comments and empty lines are allowed in the Lock Bit Configuration file.

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

# sb_sb_0/sb_sb_MSS_0/MSS_ADLIB_INST/INST_MSS_050_IP
MSS_ESRAM_CONFIG_LOCK 0
MSS_ESRAM_MAX_LAT_LOCK 1
MSS_DDR_CONFIG_LOCK 1
MSS_ENVM_CONFIG_LOCK 0
MSS_ENVM_REMAP_BASE_LOCK 1
MSS_ENVM_FAB_REMAP_LOCK 1
MSS_CC_CONFIG_LOCK 0
MSS_CC_CACHEREGION_LOCK 1
MSS_CC_LOCKBASEADDR_LOCK 1
MSS_CC_FLUSHINDX_LOCK 0
MSS_DDRB_BUF_TIMER_LOCK 1
MSS_DDRB_NB_ADR_LOCK 1
MSS_DDRB_NB_SIZE_LOCK 0
MSS_DDRB_CONFIG_LOCK 1
MSS_EDAC_ENABLE_LOCK 1
MSS_MASTER_WEIGHT_CONFIG0_LOCK 1
MSS_MASTER_WEIGHT_CONFIG1_LOCK 1
MSS_SOFT_INTERRUPT_LOCK 1
MSS_SOFTRESET_ENVM0_SOFTRESET_LOCK 1
MSS_SOFTRESET_ENVM1_SOFTRESET_LOCK 1
MSS_SOFTRESET_ESRAM0_SOFTRESET_LOCK 1
MSS_SOFTRESET_ESRAM1_SOFTRESET_LOCK 1
MSS_SOFTRESET_MAC_SOFTRESET_LOCK 1
MSS_SOFTRESET_PDMA_SOFTRESET_LOCK 1
MSS_SOFTRESET_TIMER_SOFTRESET_LOCK 1
MSS_SOFTRESET_MMUART0_SOFTRESET_LOCK 1
MSS_SOFTRESET_MMUART1_SOFTRESET_LOCK 1
MSS_SOFTRESET_G4SPI0_SOFTRESET_LOCK 1
MSS_SOFTRESET_G4SPI1_SOFTRESET_LOCK 1
MSS_SOFTRESET_I2C0_SOFTRESET_LOCK 1
MSS_SOFTRESET_I2C1_SOFTRESET_LOCK 1
MSS_SOFTRESET_CAN_SOFTRESET_LOCK 1
MSS_SOFTRESET_USB_SOFTRESET_LOCK 1
MSS_SOFTRESET_COMBLK_SOFTRESET_LOCK 1
MSS_SOFTRESET_FPGA_SOFTRESET_LOCK 1
MSS_SOFTRESET_HPDMA_SOFTRESET_LOCK 1
MSS_SOFTRESET_FIC32_0_SOFTRESET_LOCK 1
MSS_SOFTRESET_FIC32_1_SOFTRESET_LOCK 1
MSS_SOFTRESET_MSS_GPIO_SOFTRESET_LOCK 1
MSS_SOFTRESET_MSS_GPOUT_7_0_SOFT_RESET_LOCK 1
MSS_SOFTRESET_MSS_GPOUT_15_8_SOFT_RESET_LOCK 1
MSS_SOFTRESET_MSS_GPOUT_23_16_SOFT_RESET_LOCK 1
MSS_SOFTRESET_MSS_GPOUT_31_24_SOFT_RESET_LOCK 1
MSS_SOFTRESET_MDDR_CTLR_SOFTRESET_LOCK 1
MSS_SOFTRESET_MDDR_FIC64_SOFTRESET_LOCK 1
MSS_M3_CONFIG_LOCK 1
```

Figure 59 · Lock Bit Configuration File

Validation of Register Lock Bits Configuration File

During Map File generation (**Design Flow window > Generate FPGA Array Data**), Libero SoC validates the Register Lock Bit Configuration file and displays error message when invalid parameter or parameter values are found:






- "Error: Invalid parameter name '<param>' while reading register lock bits configuration file <file name>"
- "Error: Invalid value '<value>' for parameter '<param>' while reading register lock bits configuration file <file name>"

- "Error: Parameter '<param>' cannot be set to '1', while reading register lock bits configuration file <file name>. This error is displayed when the value of SERDES register fields *K_BRIDGE_SPEED is set by the SERDES Configurator to "0" and cannot be changed. It is illegal to change the value to "1".

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.

Status Icon	Tooltip	Description	Possible Causes/Remedy
N/A	Tool has not run yet.	NEW state	Tool has not run or it has been cleaned.
	Tool runs successfully.	Tool runs with no errors. PASS state.	N/A
	Varies with the tool.	Tool runs but with Warnings.	Varies with the tool (e.g., for the Compile Netlist step, not all I/Os have been assigned and locked).
	Tool Fails.	Tool fails to run.	Invalid command options or switches, invalid design objects, invalid design constraints.
	Design State is Out of Date.	Tool state changes from PASS to OUT OF DATE.	Since the last successful run, design source design files, constraint files or constraint file/tool association, constraint files order, tool options, and/or project settings have changed.
	Timing Constraints have not been met.	Timing Verification runs successfully but the design fails to meet timing requirements.	Design fails Timing Analysis. Design has either set-up or hold time violations or both.

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 and System Builder 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.

- o 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.
- o 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
 - o 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:
 - o Device selection
 - o Device settings
 - o Design Flow
 - o Analysis operating conditions

Setting Changed	Applicable Families	Note	Design Flow Tools Affected	New State of the Affected Design Flow Tools
Family	SmartFusion2, IGLOO2, RTG4	Part# is changed	N/A since family cannot be changed once a root is created	N/A
Die	SmartFusion2, IGLOO2, RTG4	Part# is changed	All	CLEANED/NEW
Package	SmartFusion2, IGLOO2, RTG4	Part# is changed	All	CLEANED/NEW
Speed	SmartFusion2, IGLOO2, RTG4	Part# is changed	All	CLEANED/NEW
Core Voltage	SmartFusion2, IGLOO2, RTG4	Part# is changed	All	CLEANED/NEW
Range	SmartFusion2, IGLOO2, RTG4	Part# is changed	All	CLEANED/NEW
Default I/O Technology	SmartFusion2, IGLOO2, RTG4		Synthesize, and all tools below it	OUT OF DATE
Reserve Pins for Probes	SmartFusion2, IGLOO2, RTG4		Place and Route, and all tools below it	OUT OF DATE
Reserve Pins for Device Migration*	SmartFusion2, IGLOO2, RTG4		Place and Route and all tools below it	OUT OF DATE
PLL Supply Voltage (V)	SmartFusion2, IGLOO2		Verify Power, Generate FPGA Array Data and all other "Program	OUT OF DATE

Setting Changed	Applicable Families	Note	Design Flow Tools Affected	New State of the Affected Design Flow Tools
			and Debug Design” tools below it	
Power On Reset Delay	SmartFusion2, IGLOO2		Generate FPGA Array Data and all other “Program and Debug Design” tools below it	OUT OF DATE
System controller suspended mode	SmartFusion2, IGLOO2		Generate FPGA Array Data and all other “Program and Debug Design” tools below it	OUT OF DATE
Preferred Language	SmartFusion2, IGLOO2, RTG4		None	N/A
Enable synthesis	SmartFusion2, IGLOO2, RTG4		All	OUT OF DATE
Synthesis gate level netlist format	SmartFusion2, IGLOO2, RTG4		Synthesize	CLEANED/NEW
Design methodology(standalone initialization)	SmartFusion2 and IGLOO2		None	N/A
Reports(Maximum number of high fanout nets to be displayed)	SmartFusion2, IGLOO2, RTG4		None	N/A
Abort flow if errors are found in PDC	SmartFusion2, IGLOO2, RTG4		None	N/A
Abort flow if errors are found in SDC	SmartFusion2, IGLOO2, RTG4		None	N/A
Temperature range(C)	SmartFusion2, IGLOO2, RTG4		Verify Timing, Post Layout Simulate, and Verify Power	OUT OF DATE
Core voltage range(V)	SmartFusion2, IGLOO2, RTG4		Verify Timing, Post Layout Simulate, and Verify Power	OUT OF DATE
Default I/O voltage range	SmartFusion2, IGLOO2, RTG4		Verify Timing, Post Layout Simulate, and Verify Power	OUT OF DATE

Setting Changed	Applicable Families	Note	Design Flow Tools Affected	New State of the Affected Design Flow Tools
Radiation (krad)	RTG4		Verify Timing, Post Layout Simulate, and Verify Power	OUT OF DATE
Enable Single Event Transient mitigation	RTG4		Synthesize and all tools below it	OUT OF DATE

*These settings are set in the I/O Attributes tab of the Constraint Manager, not in the Project Settings.

- **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.

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

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

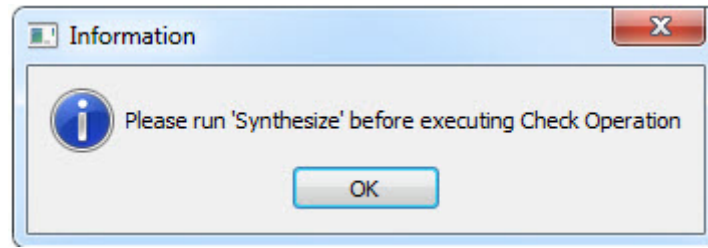


Figure 60 · Pop-Up message: Design State insufficient for Constraints Check operation

Edit Constraints

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

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

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

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

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

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

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

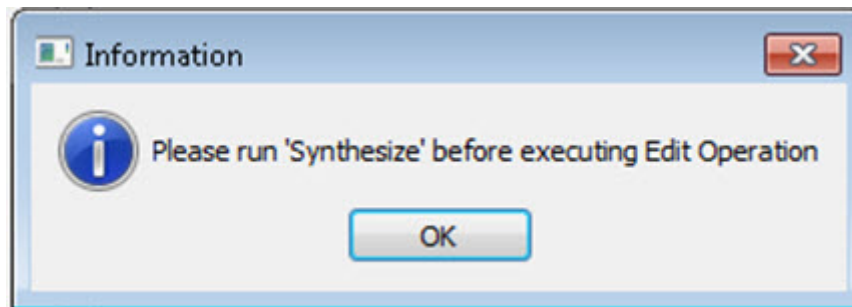
Constraints in the SDC constraint file that are read by the Constraint Editor and subsequently modified by you will be written back to the SDC file when you save the edits and close the Constraint Editor.

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

Constraint Type and Interactive Tool

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

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



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

Place and Route - SmartFusion2, IGLOO2, RTG4

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.

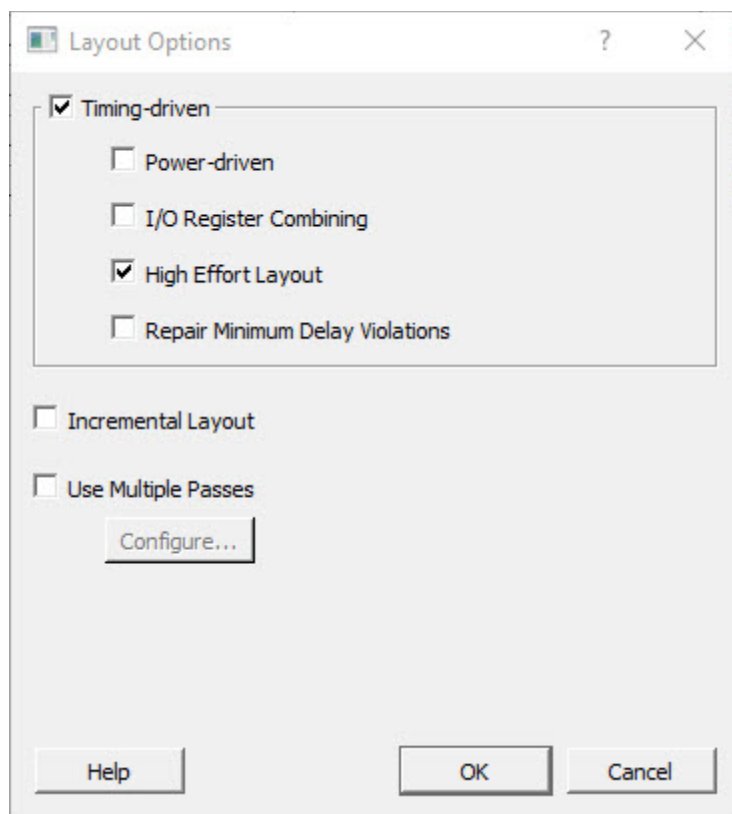


Figure 61 · Layout Options Dialog Box

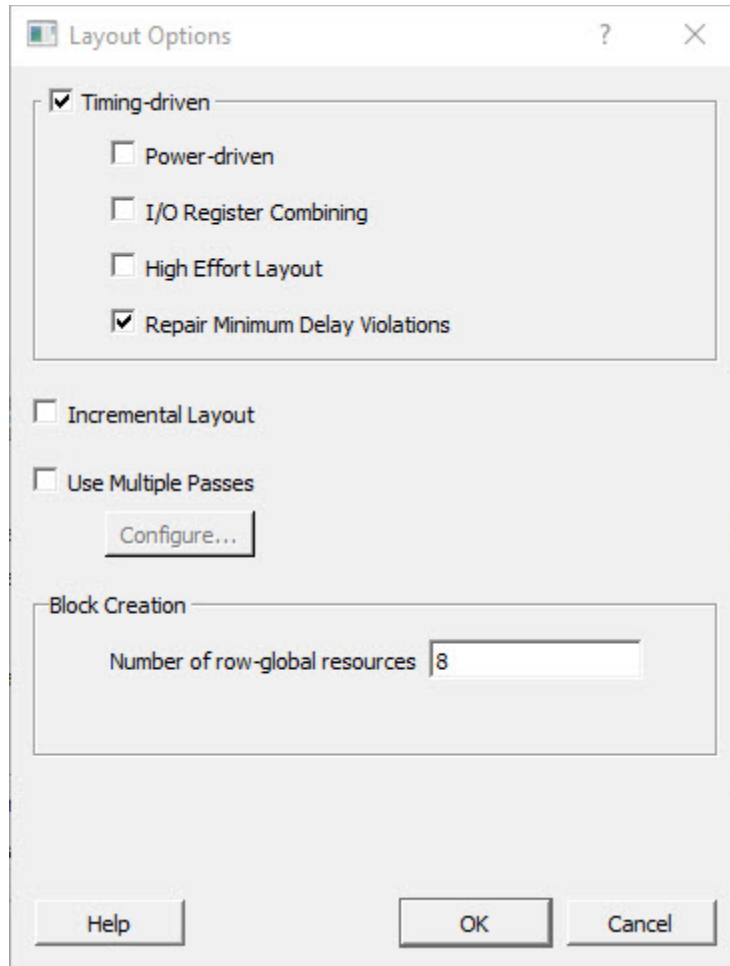


Figure 62 · Layout Options Dialog Box - with Block Flow enabled

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. Refer to the Rules for SmartFusion2 and IGL002 Devices in the topic I/O Register Combining.

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

This option is enabled by default for SmartFusion2, IGLOO2, RTG4 devices.

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)
- Terminates at a register that is clocked by a Global Buffer driven by an MSIO or MSIOD (RTG4 only). RTG4 I/O delay taps cannot be used to fix hold violations for Global Buffers driven by an MSIO or MSIOD.

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. For details see [Multiple Pass Layout Configuration](#) (*SmartFusion2, IGLOO2, RTG4*).

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

[Multiple Pass Layout Configuration](#) (*SmartFusion2, IGLOO2, RTG4*).
[extended_run_lib](#)

Multiple Pass Layout Configuration (SmartFusion2, IGLOO2, RTG4)

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.

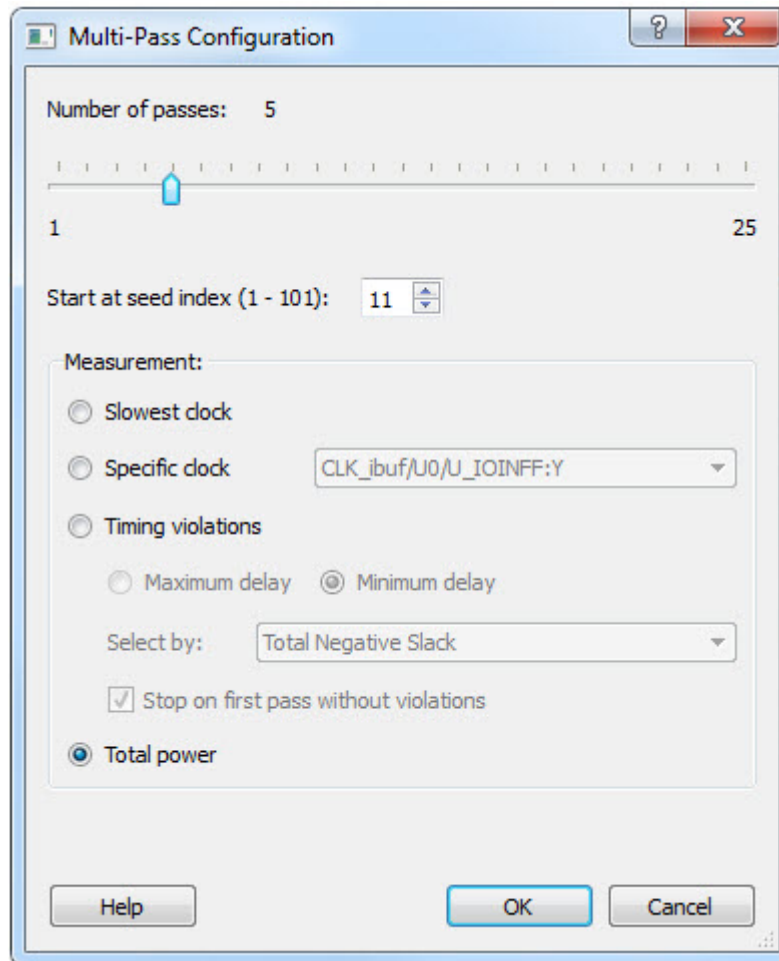


Figure 63 · Multi-Pass Configuration Dialog Box

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.

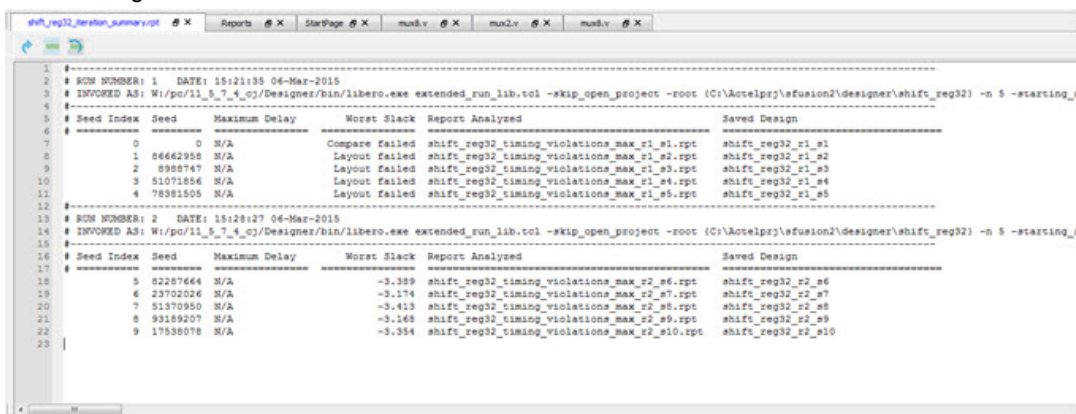


Figure 64 · Iteration Summary Report

See Also

[Place and Route - SmartFusion2, IGLOO2, RTG4 extended run lib](#)

Resource Usage (SmartFusion2, IGLOO2, RTG4)

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.

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

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

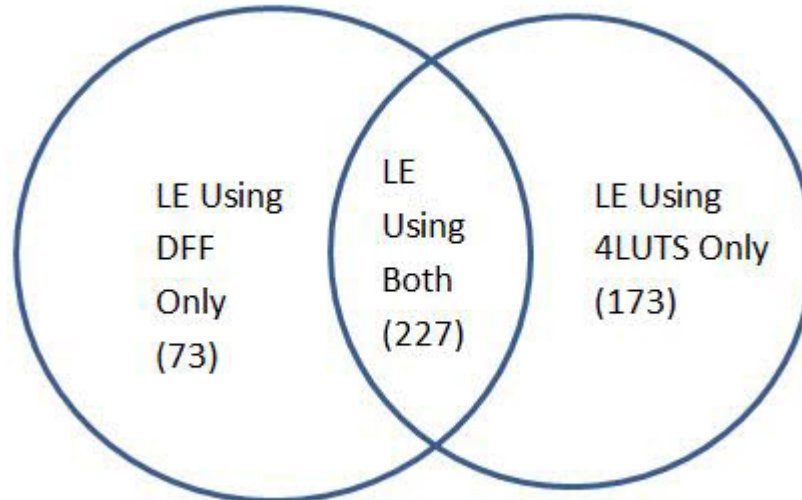
The number of 4LUTs in the report are the total number used for your design, regardless of whether or not they are combined with the DFFs. Similarly, the number of DFFs in the report are the total number used for your design, regardless of whether or not they are combined with 4LUT's.

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.

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



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)
- Global Asynchronous Reset Buffer (GRESET) - RTG4 only
- Row Global Asynchronous Reset Buffers (RGRESET) - RTG4 only

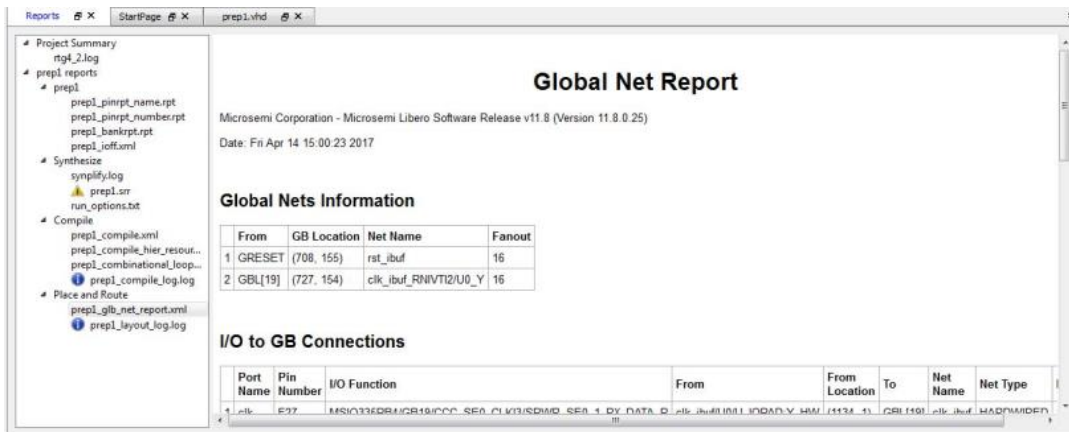


Figure 65 · Global Net Report

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

	From	GB Location	Net Name	Fanout
1	GB[4]	(726, 156)	reset_ctrl_i/R_core_reset_out_RNIFUQ6/U0_YWn	35701
2	GB[8]	(734, 156)	pll_i/GL0_INST/U0_YWn	35488
3	GB[15]	(741, 156)	pll_i/GL3_INST/U0_YWn	2006
4	GB[13]	(739, 156)	reset_ctrl_i/R_global_reset_RNIOT36/U0_YWn	1848
5	GB[14]	(740, 156)	pll_i/GL2_INST/U0_YWn_GEa	1104
6	GB[2]	(724, 156)	serdes_i/SERDES_IF_0/EPCS_1_RX_CLK_keep_RNIEDL3/U0_YWn	514
7	GB[0]	(722, 156)	serdes_i/SERDES_IF_0/EPCS_0_RX_CLK_keep_RNID1J5/U0_YWn	513
8	GB[3]	(725, 156)	serdes_i/SERDES_IF_0/EPCS_2_RX_CLK_keep_RNIFPN1/U0_YWn	511
9	GB[7]	(729, 156)	serdes_i/pcs_gl_0_pcs_0/rx_rst_n_i_0_RN9T3C/U0_YWn	312
10	GB[11]	(737, 156)	serdes_i/pcs_gl_1_pcs_0/rx_rst_n_i_0_RN1AGFA/U0_YWn	310
11	GB[1]	(723, 156)	serdes_i/SERDES_IF_0/EPCS_0_TX_CLK_keep_RNIFLD8/U0_YWn	178
12	GB[5]	(727, 156)	serdes_i/SERDES_IF_0/EPCS_2_TX_CLK_keep_RNIHDI4/U0_YWn	178
13	GB[6]	(728, 156)	serdes_i/SERDES_IF_0/EPCS_1_TX_CLK_keep_RNIG1G6/U0_YWn	178
14	GB[10]	(736, 156)	SPI_SCLK_ibuf_RNIT4T6/U0_YWn_GEa	149
15	GB[9]	(735, 156)	SRAM_CQ_ibuf_RNIB7IC/U0_YWn_GEa	111
16	GB[12]	(738, 156)	SRAM_CQn_ibuf_RNIPMM6/U0_YWn_GEa	18

Figure 66 · 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

Port Name	Pin Number	I/O Function	From	From Location	To	Net Name	Net Type	Fanout
1 SPI_SCLK	D33	MSIO176NB18	SPI_SCLK_ibuf/U0/U_I0IN.Y	(6, 307)	GB[10]	SPI_SCLK_ibuf	ROUTED	1
2 SRAM_CQ	G16	DDRIO120PB2/MDDR_DQ_ECC0/CCC_NE1_CLKI3	SRAM_CQ_ibuf/U0/U_I0IN.Y	North IO #7 (1005, 313)	GB[9]	SRAM_CQ_ibuf	ROUTED	1
3 SRAM_CQn	F16	DDRIO120PB2/MDDR_DQ_ECC1/GB12/CCC_NE1_CLKI2	SRAM_CQn_ibuf/U0/U_I0PAD.Y	North IO #6 (1002, 313)	GB[12]	SRAM_CQn_ibuf	HARDWIRED	1

Figure 67 · 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_CLKI2) 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.

For details about RTG4 devices I/O Function names, see Col B of [CG1657 Package Pin Assignment Table](#), [CQ352 Package Pin Assignment Table](#).

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).

Fabric to GB Connections

	From	From Location	To	Net Name	Net Type	Fanout
1	reset_ctrl_i/R_core_reset_out.Q	(720, 160)	GB[4]	reset_ctrl_i/core_reset_out	ROUTED	1
2	reset_ctrl_i/R_global_reset.Q	(722, 160)	GB[13]	reset_ctrl_i/global_reset_0	ROUTED	3
3	serdes_i/SERDES_IF_0/SERDESIF_INST/INST_SERDESIF_IP:EPCS_RXCLK_1	(72, 2)	GB[2]	serdes_i/SERDES_IF_0/EPCS_1_RX_CLK	ROUTED	1
4	serdes_i/SERDES_IF_0/SERDESIF_INST/INST_SERDESIF_IP:EPCS_RXCLK_0	(72, 2)	GB[0]	serdes_i/SERDES_IF_0/EPCS_0_RX_CLK	ROUTED	1
5	serdes_i/SERDES_IF_0/SERDESIF_INST/INST_SERDESIF_IP:EPCS_RXCLK[0]	(72, 2)	GB[3]	serdes_i/SERDES_IF_0/EPCS_2_RX_CLK	ROUTED	1
6	serdes_i/pcs_gl_0/pcs_0/rx_rst_n_i_0.Y	(269, 54)	GB[7]	serdes_i/pcs_gl_0/pcs_0/rx_rst_n_i_0_1	ROUTED	1
7	serdes_i/pcs_gl_1/pcs_0/rx_rst_n_i_0.Y	(266, 96)	GB[11]	serdes_i/pcs_gl_1/pcs_0/rx_rst_n_i_0	ROUTED	1
8	serdes_i/SERDES_IF_0/SERDESIF_INST/INST_SERDESIF_IP:EPCS_TXCLK_0	(72, 2)	GB[1]	serdes_i/SERDES_IF_0/EPCS_0_TX_CLK	ROUTED	1
9	serdes_i/SERDES_IF_0/SERDESIF_INST/INST_SERDESIF_IP:EPCS_TXCLK[0]	(72, 2)	GB[5]	serdes_i/SERDES_IF_0/EPCS_2_TX_CLK	ROUTED	1
10	serdes_i/SERDES_IF_0/SERDESIF_INST/INST_SERDESIF_IP:EPCS_TXCLK_1	(72, 2)	GB[6]	serdes_i/SERDES_IF_0/EPCS_1_TX_CLK	ROUTED	1

Figure 68 · Fabric to GB connections

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.

CCC to GB Connections

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

Figure 69 · CCC to GB Connections

CCC Input Connections

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

CCC Input Connections

Port Name	Pin Number	I/O Function	From	From Location	To (Pin Swapped for Back Annotation Only)	CCC Location	Net Name	Net Type	Fanout	
1	FPGA_CLK_P	V12	MSIO3PB5/CCC_NE0_CLK0	pl_i/CLK0_PAD_INSTIU_IOPADP_IOUT_P	East IO #0 (1455, 112)	pl_i/CCC_INSTINST_CCC_IP_CLK0_PAD	CCC_NE0 (1428, 302)	fixed_clk	HARDWIRED	1

Figure 70 · 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 backannotated netlist.

For details about RTG4 devices I/O Function names, see Col B of [CG1657 Package Pin Assignment Table](#), [CQ352 Package Pin Assignment Table](#).

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 the clock network for SERDES.

Local Clock Nets to RGB Connections

	From	From Location	Net Name	Fanout		RGB Location	Local Fanout
1	serdes_i/pcs_gl_2_pcs_0/rx_rst_n_i_0:Y	(216, 111)	serdes_i/pcs_gl_2.pcs_0/rx_rst_n_i_0_0	310	1	(364, 72)	15
					2	(364, 75)	44
					3	(364, 78)	37
					4	(364, 81)	19
					5	(364, 84)	28
					6	(364, 87)	20
					7	(364, 90)	25
					8	(364, 93)	25
					9	(364, 96)	36
					10	(364, 99)	19
					11	(364, 102)	14
					12	(364, 111)	17
					13	(364, 114)	11

Figure 71 · 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.

Local Reset Nets to RGRESET Connections (RTG4 only)

This section is available on RTG4 devices only. It lists the nets from local reset signals of components such as FIFO to RGRESET (Row Global Reset).

The location refers to the X-Y co-ordinates of the driver of the net. The **Fanout** column gives the total fanout of the net and the **Local Fanout** column gives the fanout local to RGRESET. The driver in the **From** column is routed to different RGRESETs each with different local fanout.

Local Reset Nets to RGRESET Connections

	From	From Location	Net Name	Fanout	RGRESET Location	Local Fanout
1	I_ARRAY_CORE_ABC/I_DATABUS/I_DB_CONTROL/un1_resetfifol.Y	(1313, 201)	I_ARRAY_CORE_ABC/un1_resetfifol.J	10	1 (1166, 187)	5
					2 (1166, 190)	5
2	I_ARRAY_CORE_DEF/I_DATABUS/I_DB_CONTROL/un1_resetfifol.Y	(1394, 228)	I_ARRAY_CORE_DEF/un1_resetfifol.J	10	1 (1166, 220)	5
					2 (1166, 223)	5
3	I_ARRAY_CORE_ABC/I_TGSLICE_CMP/I_C1_RESIDUE_FIFO/comb_reset.Y	(1170, 300)	I_ARRAY_CORE_ABC/I_TGSLICE_CMP/I_C1_RESIDUE_FIFO/comb_reset.Z	1	(1166, 124)	1
4	I_ARRAY_CORE_ABC/I_TGSLICE_D2/I_D2_RESIDUE_FIFO/comb_reset.Y	(1179, 291)	I_ARRAY_CORE_ABC/I_TGSLICE_D2/I_D2_RESIDUE_FIFO/comb_reset.Z	1	(1166, 181)	1
5	I_ARRAY_CORE_ABC/I_TGSLICE_D2/I_MATCH_FIFO/un1_resetoutl.Y	(1312, 201)	I_ARRAY_CORE_ABC/I_TGSLICE_D2/I_MATCH_FIFO/un1_resetoutl.J	1	(1166, 193)	1
6	I_ARRAY_CORE_ABC/I_TGSLICE/I_MATCH_FIFO/un1_resetoutl.Y	(1314, 201)	I_ARRAY_CORE_ABC/I_TGSLICE/I_MATCH_FIFO/un1_resetoutl.J	1	(1166, 196)	1
7	I_ARRAY_CORE_ABC/I_TGSLICE_CMP/I_MATCH_FIFO_CMP/un1_resetoutl.Y	(1309, 201)	I_ARRAY_CORE_ABC/I_TGSLICE_CMP/I_MATCH_FIFO_CMP/un1_resetoutl.J	1	(1166, 199)	1
8	I_ARRAY_CORE_ABC/I_OUTPUT_FIFO_WR_CTL/un1_resetsyncd.Y	(1191, 240)	I_ARRAY_CORE_ABC/I_OUTPUT_FIFO_WR_CTL/un1_resetsyncd.J	1	(1166, 202)	1
9	I_ARRAY_CORE_DEF/I_TGSLICE_CMP/I_C1_RESIDUE_FIFO/comb_reset.Y	(1191, 288)	I_ARRAY_CORE_DEF/I_TGSLICE_CMP/I_C1_RESIDUE_FIFO/comb_reset.Z	1	(1166, 211)	1
10	I_ARRAY_CORE_DEF/I_TGSLICE/I_MATCH_FIFO/un1_resetoutl.Y	(1178, 252)	I_ARRAY_CORE_DEF/I_TGSLICE/I_MATCH_FIFO/un1_resetoutl.J	1	(1166, 235)	1
11	I_ARRAY_CORE_DEF/I_TGSLICE_CMP/I_MATCH_FIFO_CMP/un1_resetoutl.Y	(1179, 252)	I_ARRAY_CORE_DEF/I_TGSLICE_CMP/I_MATCH_FIFO_CMP/un1_resetoutl.J	1	(1166, 247)	1
12	I_ARRAY_CORE_DEF/I_TGSLICE_D2/I_MATCH_FIFO/un1_resetoutl.Y	(1181, 252)	I_ARRAY_CORE_DEF/I_TGSLICE_D2/I_MATCH_FIFO/un1_resetoutl.J	1	(1166, 250)	1
13	I_ARRAY_CORE_DEF/I_OUTPUT_FIFO_WR_CTL/un1_resetsyncd.Y	(1081, 252)	I_ARRAY_CORE_DEF/I_OUTPUT_FIFO_WR_CTL/un1_resetsyncd.J	1	(1166, 253)	1
14	I_ARRAY_CORE_DEF/I_TGSLICE_D2/I_D2_RESIDUE_FIFO/comb_reset.Y	(1179, 255)	I_ARRAY_CORE_DEF/I_TGSLICE_D2/I_D2_RESIDUE_FIFO/comb_reset.Z	1	(1166, 265)	1

Figure 72 · Local Reset Nets to RGRESET Connections

Global Reset Nets to RGRESET Connections (RTG4 only)

This section is available on RTG4 devices only. It lists the nets from Global Resets to RGRESET (Row Global Resets).

	From	From Location	Net Name	Fanout	RGRESET Location	Local Fanout
1	GRESET	(708, 155)	I_ARRAY_CORE_ABC/I_TGSLICE_D2/I_D2_RESIDUE_FIFO/comb_reset_i_0_0	14	1 (1166, 178)	7
					2 (1166, 184)	7
2	GBR[3]	(735, 153)	I_ARRAY_CORE_DEF/I_TGSLICE_CMP/I_C1_RESIDUE_FIFO/comb_reset_i_0_i	14	(1166, 214)	14
3	GBR[7]	(739, 153)	I_ARRAY_CORE_DEF/I_TGSLICE_D2/I_D2_RESIDUE_FIFO/comb_reset_i_0_i	14	1 (1166, 268)	7
					2 (1166, 271)	7
4	GBR[11]	(743, 153)	I_ARRAY_CORE_ABC/I_TGSLICE_CMP/I_C1_RESIDUE_FIFO/comb_reset_i_0_i	14	1 (1166, 118)	7
					2 (1166, 127)	7

Figure 73 · Global Reset Nets to RGRESET Connections

The **From** Location refers to the X-Y co-ordinates of the driver of the net. The **RGRESET Location** refers to the X-Y co-ordinates of the RGRESET instance. The **Fanout** column gives the total fanout of the net and the **Local Fanout** column gives the fanout local to RGRESET. The driver in the **From** column is hardwired to different RGRESETs each with different local fanout.

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

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

Figure 74 · Global Clock Nets to RGB Connections

Warnings (RTG4 only)

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

- Clocks or resets nets that are routed are not radiation protected.
- Hardwired connections from DDRIO bank are not radiation protected.
- For radiation protection, Microsemi recommends the use of dedicated global clocks that comes with built-in radiation protection.

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

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

Figure 75 · Warning Example 1

- Local resets that are not driven by three separate logic cones are not radiation protected.
- For radiation protection, Microsemi recommends that each of the three inputs of every RGRESET be driven by three separate logic cones replicating the paths from the source registers. Please see the descriptions of RGRESET macro in the [RTG4 Macro Library Guide](#).

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

	From	From Location	To	Net Name	Fanout
1	I_ARRAY_CORE_ABC/I_DATABUS/I_DB_CONTROL/un1_resetfifol.Y	(1313, 201)	RGRESET	I_ARRAY_CORE_ABC/un1_resetfifol_i	10
2	I_ARRAY_CORE_DEF/I_DATABUS/I_DB_CONTROL/un1_resetfifol.Y	(1394, 228)	RGRESET	I_ARRAY_CORE_DEF/un1_resetfifol_i	10
3	I_ARRAY_CORE_ABC/I_TGSLICE_CMP/I_C1_RESIDUE_FIFO/comb_reset.Y	(1176, 300)	RGRESET	I_ARRAY_CORE_ABC/I_TGSLICE_CMP/I_C1_RESIDUE_FIFO/comb_reset_Z	1
4	I_ARRAY_CORE_ABC/I_TGSLICE2/I_D2_RESIDUE_FIFO/comb_reset.Y	(1179, 291)	RGRESET	I_ARRAY_CORE_ABC/I_TGSLICE2/I_D2_RESIDUE_FIFO/comb_reset_Z	1
5	I_ARRAY_CORE_ABC/I_TGSLICE2/I_MATCH_FIFO/un1_resetoutl.Y	(1312, 201)	RGRESET	I_ARRAY_CORE_ABC/I_TGSLICE2/I_MATCH_FIFO/un1_resetoutl_i	1
6	I_ARRAY_CORE_ABC/I_TGSLICE/I_MATCH_FIFO/un1_resetoutl.Y	(1314, 201)	RGRESET	I_ARRAY_CORE_ABC/I_TGSLICE/I_MATCH_FIFO/un1_resetoutl_i	1
7	I_ARRAY_CORE_ABC/I_TGSLICE_CMP/I_MATCH_FIFO_CMP/un1_resetoutl.Y	(1309, 201)	RGRESET	I_ARRAY_CORE_ABC/I_TGSLICE_CMP/I_MATCH_FIFO_CMP/un1_resetoutl_i	1
8	I_ARRAY_CORE_ABC/I_OUTPUT_FIFO_WR_CTL/un1_resetsyncl.Y	(1191, 240)	RGRESET	I_ARRAY_CORE_ABC/I_OUTPUT_FIFO_WR_CTL/un1_resetsyncl_i	1
9	I_ARRAY_CORE_DEF/I_TGSLICE_CMP/I_C1_RESIDUE_FIFO/comb_reset.Y	(1191, 288)	RGRESET	I_ARRAY_CORE_DEF/I_TGSLICE_CMP/I_C1_RESIDUE_FIFO/comb_reset_Z	1
10	I_ARRAY_CORE_DEF/I_TGSLICE/I_MATCH_FIFO/un1_resetoutl.Y	(1178, 252)	RGRESET	I_ARRAY_CORE_DEF/I_TGSLICE/I_MATCH_FIFO/un1_resetoutl_i	1
11	I_ARRAY_CORE_DEF/I_TGSLICE_CMP/I_MATCH_FIFO_CMP/un1_resetoutl.Y	(1179, 252)	RGRESET	I_ARRAY_CORE_DEF/I_TGSLICE_CMP/I_MATCH_FIFO_CMP/un1_resetoutl_i	1
12	I_ARRAY_CORE_DEF/I_TGSLICE2/I_MATCH_FIFO/un1_resetoutl.Y	(1181, 252)	RGRESET	I_ARRAY_CORE_DEF/I_TGSLICE2/I_MATCH_FIFO/un1_resetoutl_i	1
13	I_ARRAY_CORE_DEF/I_OUTPUT_FIFO_WR_CTL/un1_resetsyncl.Y	(1081, 252)	RGRESET	I_ARRAY_CORE_DEF/I_OUTPUT_FIFO_WR_CTL/un1_resetsyncl_i	1
14	I_ARRAY_CORE_DEF/I_TGSLICE2/I_D2_RESIDUE_FIFO/comb_reset.Y	(1179, 255)	RGRESET	I_ARRAY_CORE_DEF/I_TGSLICE2/I_D2_RESIDUE_FIFO/comb_reset_Z	1

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

Figure 76 · Warning Example 2

See Also

[SmartFusion2 and IGLOO2 Clocking Resources User Guide](#)

[RTG4 FPGA Clocking Resources User Guide](#)

[CG1657 Package Pin Assignment Table](#)

[CQ352 Package Pin Assignment Table](#)

Verify Post Layout Implementation

Generate Back Annotated Files - SmartFusion2, IGLOO2, and RTG4

Generates Back Annotated files for your design.

Back Annotated files include:

- *.ba.sdf - Standard Delay Format 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.

To generate these files, in the Design Flow window click **Implement Design** and double-click **Generate Back Annotated Files**.

Right-click **Generate Back Annotated Files** and choose **Configure Options** to open the Generate Back Annotated Files Options dialog box.

Simulator Language Type - Set your simulator language type according to your design.

Timing: Export enhanced min delays for best case - Exports your enhanced min delays to include your best-case timing results in your Back Annotated file.

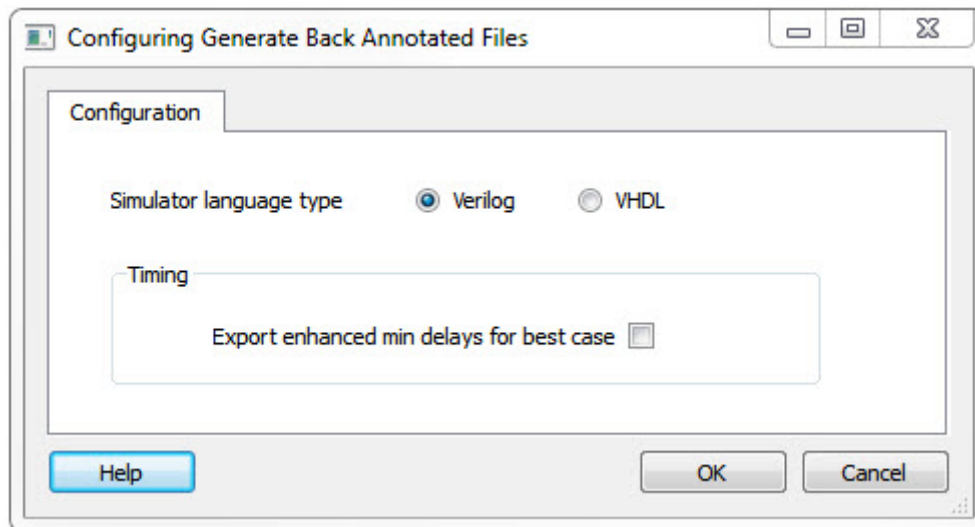


Figure 77 · Configuring Generate Back Annotated Files Dialog Box

Simulate - Opens ModelSim ME

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

If you wish to perform [pre-layout simulation](#): In the Design Flow Window, under Verify Pre-Synthesized design, double-click Simulate.

To perform timing simulation:

1. If you have not done so, back-annotate your design and create your testbench.
2. Right-click **Simulate** (in 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**.

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 μ s 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. If you did not create a testbench with WaveFormer Pro, you may get error messages with the vsim command if the instance names of your testbench do not follow the same conventions as WaveFormer Pro. Ignore the error message and type the correct vsim command.
6. When you are done, from the **File** menu, choose **Quit**.

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:

- 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.

Constraints Coverage Report

- Generate constraints coverage report

The actual values for High/Low Voltage and High/Low Temperature shown in this configuration dialog box are based on the operating conditions: COM, IND, MIL, TGrade1/2, and/or custom settings as set in the Project's settings (Project > Project Settings > Analysis Operating Conditions). Refer to Project Settings > Analysis Operating Conditions for the actual High/Low Voltage and High/Low Temperature values.

The following figure shows an example of the Verify Timing Configuration dialog box.

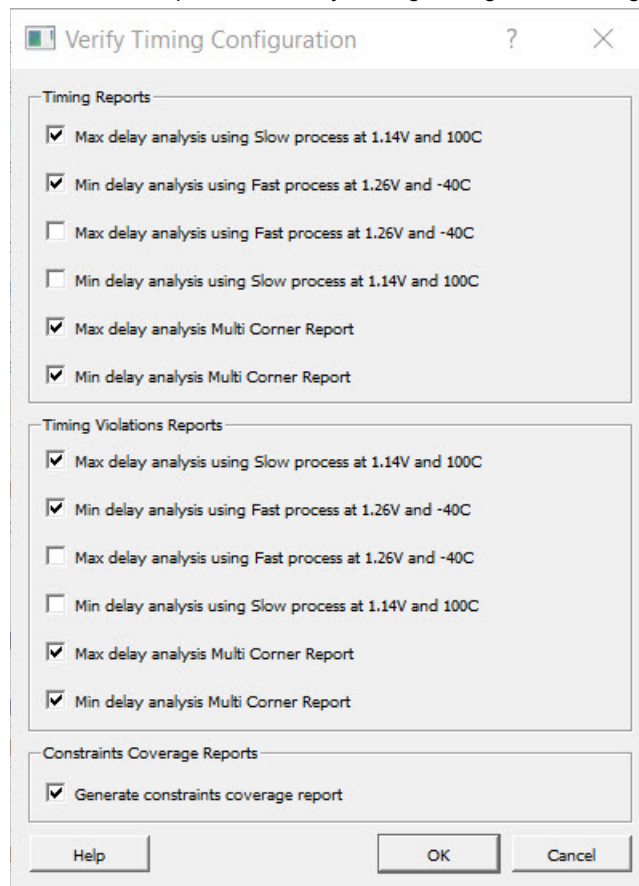


Figure 78 · Verify Timing Configuration Dialog Box

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.

<root>_max_timing_slow_<lv>_<ht>.xml (generated by default)

- Min Delay Static Timing Analysis report based on Fast process, High Voltage and Low Temperature operating conditions.

<root>_max_timing_fast_<hv>_<lt>.xml

- Max Delay Static Timing Analysis report based on Fast process, High Voltage and Low Temperature operating conditions.

<root>_min_timing_fast_<hv>_<lt>.xml (generated by default)

- Min Delay Static Timing Analysis report based on Slow process, Low Voltage and High Temperature operating conditions.

<root>_min_timing_slow_<lv>_<ht>.xml

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 Static Timing Analysis report based on Slow process, Low Voltage and High Temperature operating conditions.

<root>_max_timing_slow_violations_<lv>_<ht>.xml (generated by default)

- Min Delay Static Timing Analysis report based on Fast process, High Voltage and Low Temperature operating conditions.

<root>_max_timing_violations_fast_<hv>_<lt>.xml

- Max Delay Static Timing Analysis report based on Fast process, High Voltage and Low Temperature operating conditions.

<root>_min_timing_fast_violations_<hv>_<lt>.xml (generated by default)

- Min Delay Static Timing Analysis report based on Slow process, Low Voltage and High Temperature operating conditions.

<root>_min_timing_slow_violations_<lv>_<ht>.xml

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)

Combinational loop report – This report displays combinational loops found during initialization.

<root>_timing_combinational_loops.xml (always generated)

Note: The actual values for High/Low Voltage and High/Low Temperature shown in this configuration dialog box are based on the operating conditions: COM, IND, MIL, TGrade1/2, and/or custom settings as set in the Project's settings (**Project > Project Settings > Analysis Operating Conditions**). Refer to **Project Settings > Analysis Operating Conditions** for the actual High/Low Voltage and High/Low Temperature values.



Report Listing Icon Legend	
Icon	Definition
✓	Timing requirement met for this report
✗	Timing requirement not met (violations) for this report
?	Timing report available for generation but has not been selected/configured for generation

Figure 79 · Reports Example

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.

Note: 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 violations
- Identify paths with timing violations
- Analyze delays of paths that have no timing constraints
- Perform inter-clock domain timing verification
- Perform maximum and minimum delay analysis for setup and hold checks

To improve the accuracy of the results, SmartTime evaluates clock skew during timing analysis by 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
- Constraints Coverage Report
- Combinational Loop 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.

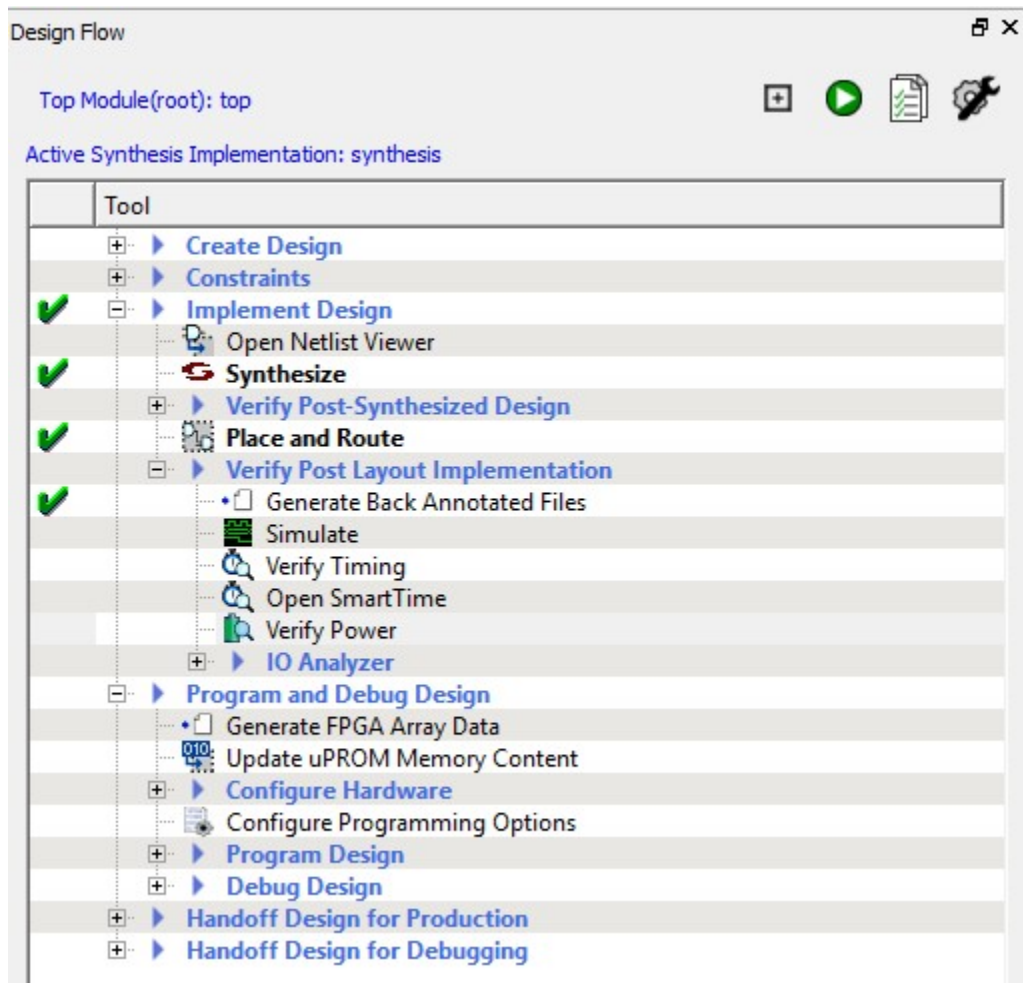



Figure 80 · Verify Power right-click menu

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 - Selects 2.5V or 3.3V as the nominal voltage of VPP on the board, for generation of the power report.

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 [SmartPower User Guide](#)

IO Advisor (SmartFusion2, IGLOO2, and RTG4)

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

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

To open the IO Advisor from the Design Flow window, right-click Manage Constraints, select Open Manage Constraints View, select the I/O Attributes tab, select Edit with I/O Advisor (**Design Flow window > Manage Constraints > Open Manage Constraints View > I/O Attributes > Edit > Edit with I/O Advisor**).

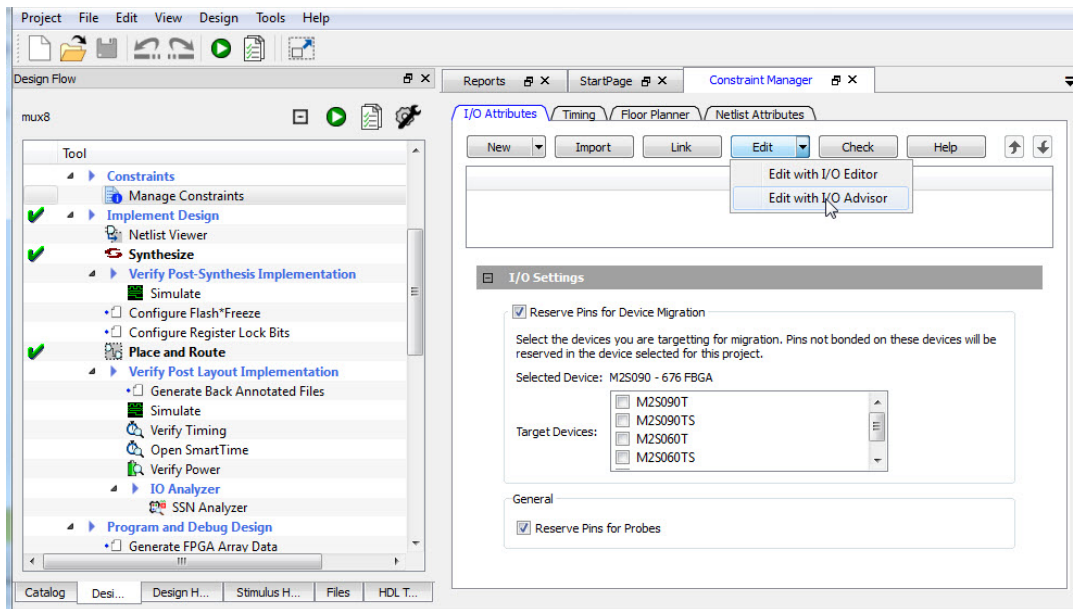


Figure 81 · I/O Advisor

Introduction

The Introduction screen provides general information about the IO Advisor.

The introduction screen provides the navigational panel for you to navigate to the following panels:

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

All steps in the IO Advisor are optional.

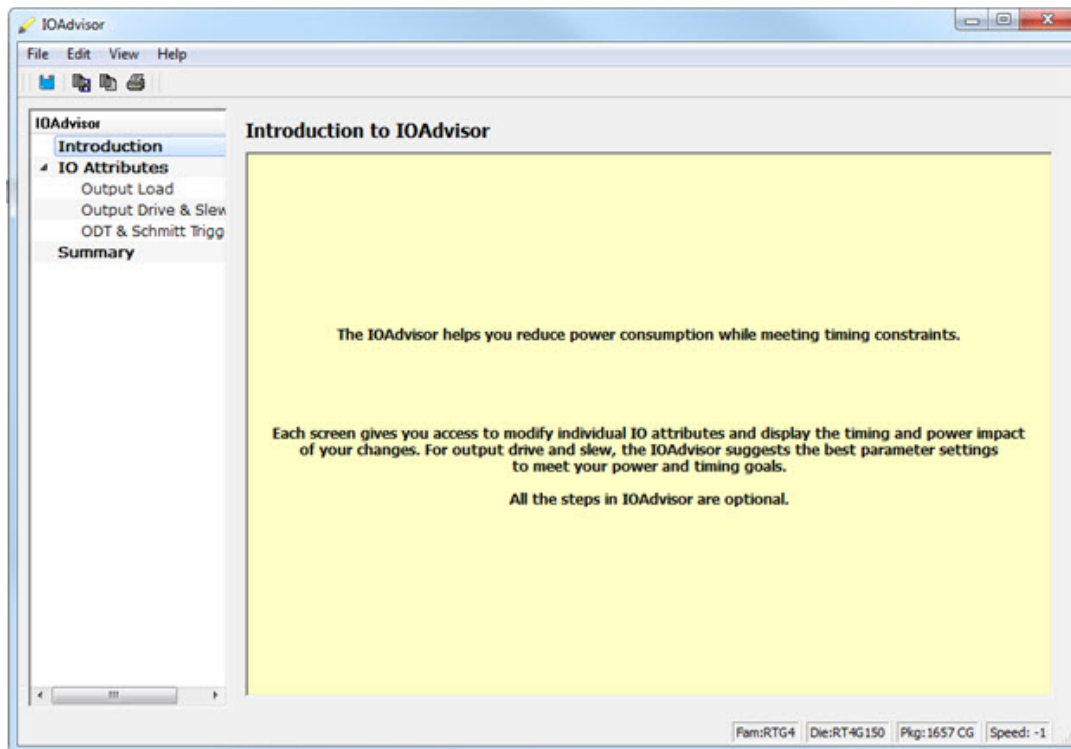


Figure 82 · IO Advisor - Introduction

Output Load

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

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

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

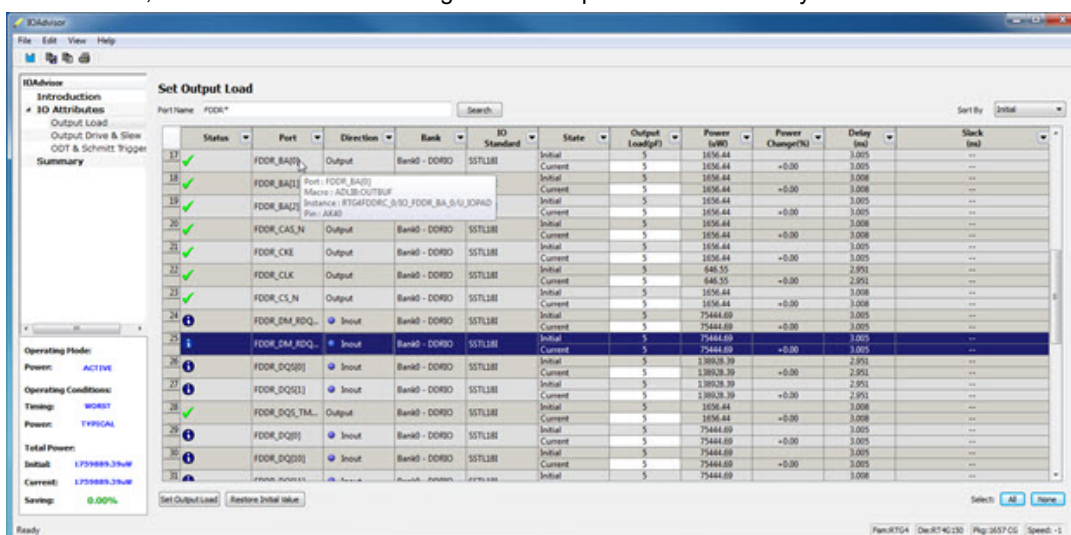


Figure 83 · IO Advisor - Output Load Panel

Search and Regular Expressions

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

Set Output Load

Port Name: FDDR* Sort By: Initial




	Status	Port	Direction	Bank	IO Standard	State	Output Load(pF)	Power (uW)	Power Change(%)	Delay (ns)	Slack (ns)
16	✓	FDDR_ADD[0]	Output	Bank0 - DQSIQ	SSTL18I	Initial	5	1656.44	+0.00	3.008	---
17	✓	FDDR_BA[0]	Output	Bank0 - DQSIQ	SSTL18I	Current	5	1656.44	+0.00	3.008	---
18	✓	FDDR_BA[1]	Output	Bank0 - DQSIQ	SSTL18I	Initial	5	1656.44	+0.00	3.005	---
19	✓	FDDR_BA[2]	Output	Bank0 - DQSIQ	SSTL18I	Current	5	1656.44	+0.00	3.008	---
20	✓	FDDR_CAS_N	Output	Bank0 - DQSIQ	SSTL18I	Initial	5	1656.44	+0.00	3.005	---
21	✓	FDDR_CKE	Output	Bank0 - DQSIQ	SSTL18I	Current	5	1656.44	+0.00	3.008	---
22	✓	FDDR_CLK	Output	Bank0 - DQSIQ	SSTL18I	Initial	5	646.55	+0.00	2.951	---
23	✓	FDDR_CS_N	Output	Bank0 - DQSIQ	SSTL18I	Current	5	1656.44	+0.00	3.008	---
24	ⓘ	FDDR_DM_RDQ...	Inout	Bank0 - DQSIQ	SSTL18I	Initial	5	75444.69	+0.00	3.005	---
25	ⓘ	FDDR_DM_RDQ...	Inout	Bank0 - DQSIQ	SSTL18I	Current	5	75444.69	+0.00	3.005	---
26	ⓘ	FDDR_DQ[0]	Inout	Bank0 - DQSIQ	SSTL18I	Initial	5	138928.39	+0.00	2.951	---
27	ⓘ	FDDR_DQ[1]	Inout	Bank0 - DQSIQ	SSTL18I	Current	5	138928.39	+0.00	2.951	---
28	✓	FDDR_DQ[2]	Inout	Bank0 - DQSIQ	SSTL18I	Initial	5	1656.44	+0.00	3.008	---
29	ⓘ	FDDR_DQ[3]	Inout	Bank0 - DQSIQ	SSTL18I	Current	5	75444.69	+0.00	3.005	---
30	ⓘ	FDDR_DQ[4]	Inout	Bank0 - DQSIQ	SSTL18I	Initial	5	75444.69	+0.00	3.005	---

Set Output Load Select:

Figure 84 · Search Field and Regular Expressions

Status Column

The icon in the Status Column displays the status of the Output Port.

Icon	Status and Explanation
	OK - The IO attributes match the suggestion in Output Drive and Slew Table.
	Error – The Timing constraints for this IO are not met in Output Drive and Slew Table.
	Information – you can improve the power and/or timing of the IO by applying the suggestion in Output Drive and Slew Table.

Column Display and Sorting

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

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

Set Output Load

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

Restore Initial Value

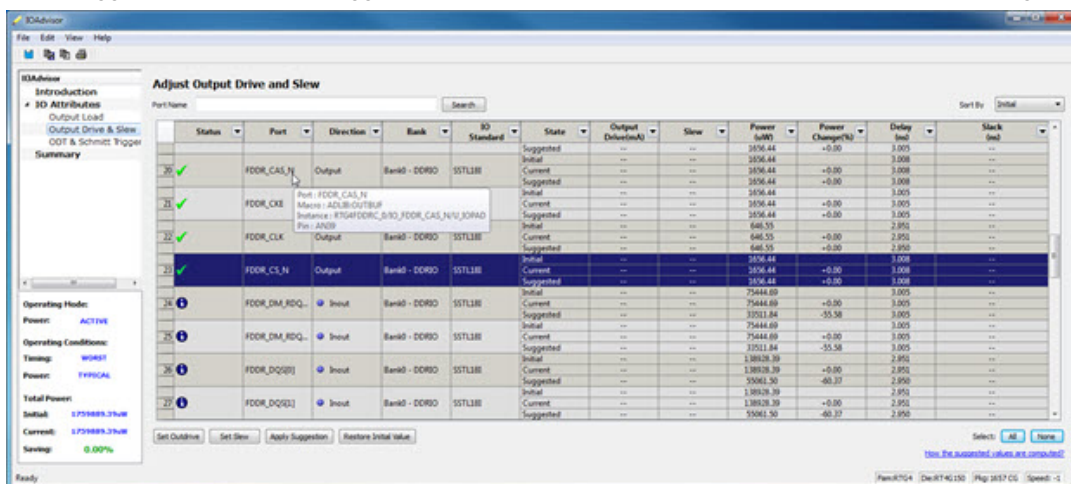
To restore a Port's output load to the initial value, select the output port and click **Restore Initial Value**. The current value changes to become the same value as the initial value.

Output Drive and Slew

The Output Drive and Slew page displays the Output Drive and Slew of all output/inout ports of your design. The display can be sorted according to the initial, current or suggested values. To change the sorting, click the Sort By drop-down menu to make your selection.

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

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



Status	Port	Direction	Bank	IO Standard	State	Output Delay	Slew	Power (W)	Power Change(%)	Delay (ns)	Slack (ns)
✓	FDDR_CAS_N	Output	Bank0 - DQ0D0	SSTL3B	Suggested	1056.44	-0.00	3.000	...
✓	FDDR_CKE	Output	Bank0 - DQ0D0	SSTL3B	Suggested	1056.44	-0.00	3.000	...
✓	FDDR_CLK	Output	Bank0 - DQ0D0	SSTL3B	Suggested	1056.44	-0.00	3.000	...
✓	FDDR_CS_N	Output	Bank0 - DQ0D0	SSTL3B	Suggested	1056.44	-0.00	3.000	...
✓	FDDR_DM_RDQ	Inout	Bank0 - DQ0D0	SSTL3B	Suggested	1056.44	-0.00	3.000	...
✓	FDDR_DM_RDQ	Inout	Bank0 - DQ0D0	SSTL3B	Suggested	1056.44	-0.00	3.000	...
✓	FDDR_DQ[0]	Inout	Bank0 - DQ0D0	SSTL3B	Suggested	1056.44	-0.00	3.000	...
✓	FDDR_DQ[0]	Inout	Bank0 - DQ0D0	SSTL3B	Suggested	1056.44	-0.00	3.000	...

Figure 85 · IO Advisor – Output Drive and Slew

How the Suggested Values Are Computed

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

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

In this screen, you can change the drive strength and slew of the design output I/Os. Select the out drive and/or the slew current value cell. Click the cell to open the combo box. Choose the value you want from the set of valid values. You can restore the initial values by clicking **Restore Initial Value**.

To make changes to multiple I/Os, select multiple I/Os (Control+click), click **Set Slew** or **Set Outdrive**, select the value, and click **OK**.

Apply Suggestion

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

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

Adjust ODT and Schmitt Trigger

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

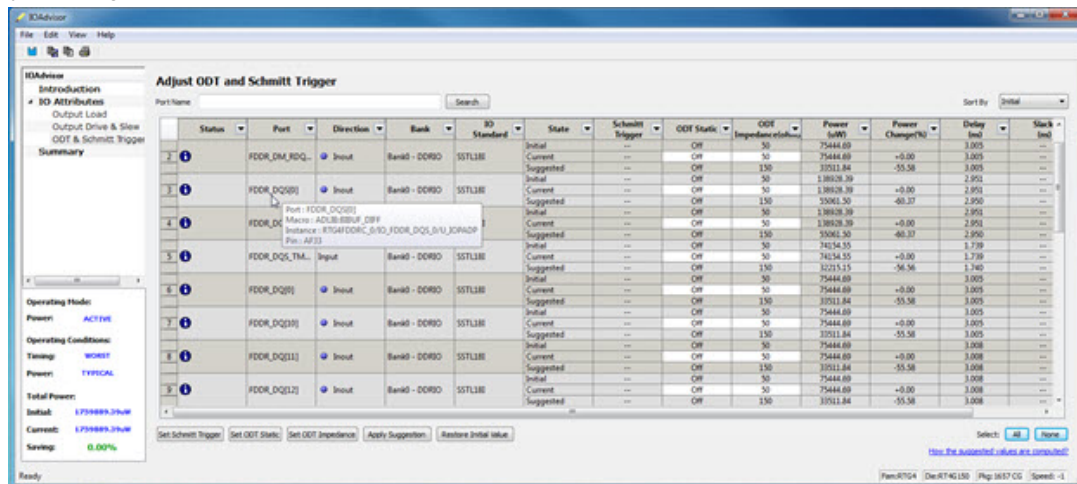


Figure 86 · IO Advisor – Adjust ODT and Schmitt Trigger

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

Search and Regular Expressions

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

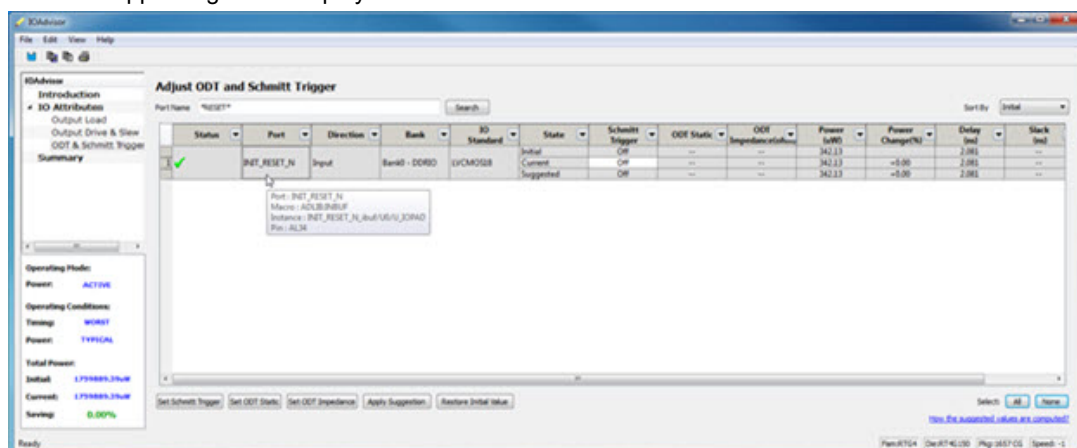





Figure 87 · Search Field and Regular Expressions

Status Column

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

Icon	Status and Explanation
	OK - The IO attributes match the suggestion in the Adjust ODT and Schmitt Trigger Table.
	Error – The Timing constraints for this IO are not met in the Adjust ODT and Schmitt Trigger Table.
	Information – you can improve the power and/or timing of the IO by applying the suggestion in the Adjust ODT and Schmitt Trigger Table.

Column Display and Sorting

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

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

Set Schmitt Trigger

For IO Standards that support the Schmitt Trigger, you can turn the Schmitt Trigger On or Off. Select the IO and click **Set Schmitt Trigger** to toggle on or off. Your setting is displayed in the Schmitt Trigger column for the IO.

Set ODT Static

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

- On – The Termination resistor for impedance matching is located inside the chip.
- Off – The Terminator resistor for impedance matching is located on the printed circuit board.

To turn the ODT Static on or off, click to select the input/inout port and from the pull-down menu, toggle on or off. You can also turn ODT Static on or off by clicking **Set ODT Static** and toggling on or off.

Set ODT Impedance (Ohm)

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

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

Apply Suggestion

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

Restore Initial Value

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

Summary of Changes

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

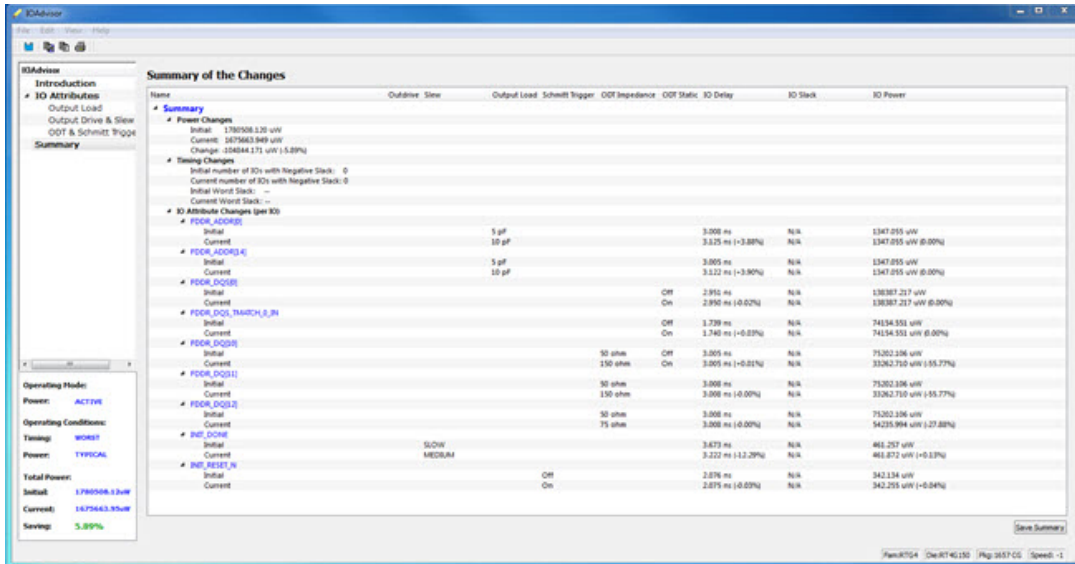


Figure 88 · IO Advisor – Summary

You can save the summary by clicking **Save Summary**, selecting the save format (text or CSV), and clicking **OK**.

To commit IO Attribute changes you have made to the database (the *io_pdc file), choose **Save** from the File Menu (**File > Save**). Click **OK** in the dialog that appears.

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

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 Families

The SSN Analyzer supports the SmartFusion2, IGLOO2, and RTG4 families.

Supported Die/Package

Family	Die	Package
IGLOO2	M2GL150	FC1152
	M2GL090	FG676
	M2GL060	FG676
	M2GL050	FG896
	M2GL025	FG484/VFG400
	M2GL010	FG484
SmartFusion2	M2S150	FC1152
	M2S090	FG676
	M2S060	FG676
	M2S050	FG896
	M2S025	FG484
	M2S010	FG484
RTG4	RT4G150	CG1657

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
- LVTTTL

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 IOs
- Summary

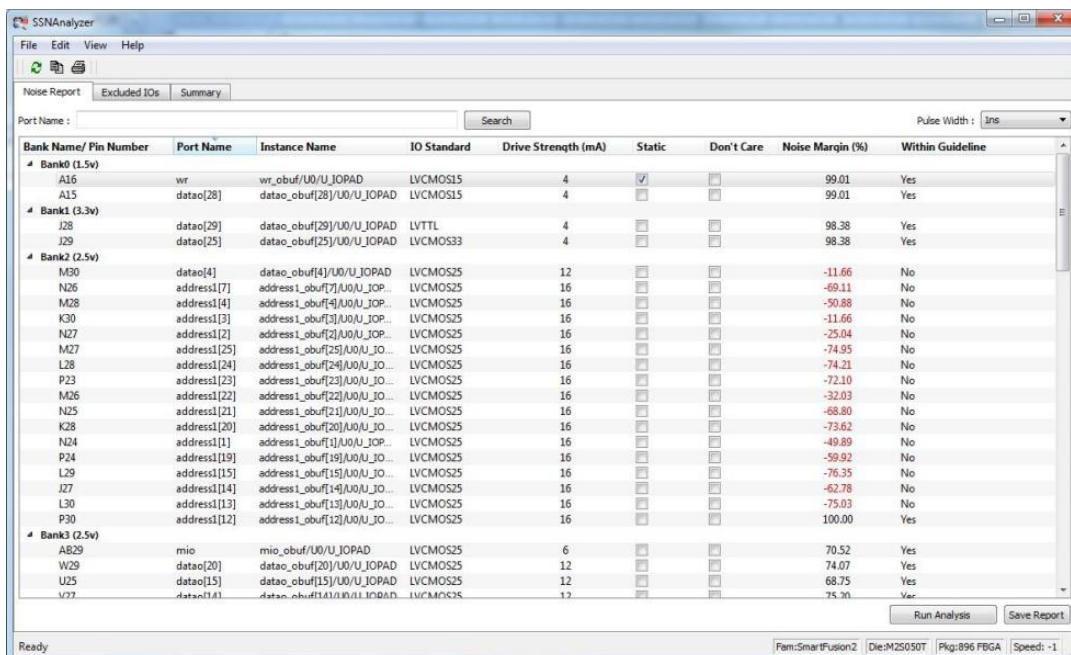
Noise Report

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 LVTTTL.
- **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/LVCMOS (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 LVCMOS 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"



The screenshot shows the SSN Analyzer interface with the Noise Report tab selected. The table below represents the data visible in the report.

Bank Name/ Pin Number	Port Name	Instance Name	IO Standard	Drive Strength (mA)	Static	Don't Care	Noise Margin (%)	Within Guideline
Bank0 (1.5v)								
A16	wr	wr_obuf/U0/U_JOPAD	LVC MOS15	4	<input checked="" type="checkbox"/>	<input type="checkbox"/>	99.01	Yes
A15	datao[28]	datao_obuf[28]/U0/U_JOPAD	LVC MOS15	4	<input type="checkbox"/>	<input type="checkbox"/>	99.01	Yes
Bank1 (3.3v)								
J28	datao[29]	datao_obuf[29]/U0/U_JOPAD	LV TTL	4	<input type="checkbox"/>	<input type="checkbox"/>	98.38	Yes
J29	datao[25]	datao_obuf[25]/U0/U_JOPAD	LVC MOS33	4	<input type="checkbox"/>	<input type="checkbox"/>	98.38	Yes
Bank2 (2.5v)								
M30	datao[4]	datao_obuf[4]/U0/U_JOPAD	LVC MOS25	12	<input type="checkbox"/>	<input type="checkbox"/>	-11.66	No
N26	address[7]	address1_obuf[7]/U0/U_JOP...	LVC MOS25	16	<input type="checkbox"/>	<input type="checkbox"/>	-69.11	No
M28	address[4]	address1_obuf[4]/U0/U_JOP...	LVC MOS25	16	<input type="checkbox"/>	<input type="checkbox"/>	-50.88	No
K30	address[3]	address1_obuf[3]/U0/U_JOP...	LVC MOS25	16	<input type="checkbox"/>	<input type="checkbox"/>	-11.66	No
N27	address[2]	address1_obuf[2]/U0/U_JOP...	LVC MOS25	16	<input type="checkbox"/>	<input type="checkbox"/>	-25.04	No
M27	address[25]	address1_obuf[25]/U0/U_JO...	LVC MOS25	16	<input type="checkbox"/>	<input type="checkbox"/>	-74.95	No
L28	address[24]	address1_obuf[24]/U0/U_JO...	LVC MOS25	16	<input type="checkbox"/>	<input type="checkbox"/>	-74.21	No
P23	address[23]	address1_obuf[23]/U0/U_JO...	LVC MOS25	16	<input type="checkbox"/>	<input type="checkbox"/>	-72.10	No
M26	address[22]	address1_obuf[22]/U0/U_JO...	LVC MOS25	16	<input type="checkbox"/>	<input type="checkbox"/>	-32.03	No
N25	address[21]	address1_obuf[21]/U0/U_JO...	LVC MOS25	16	<input type="checkbox"/>	<input type="checkbox"/>	-68.80	No
K28	address[20]	address1_obuf[20]/U0/U_JO...	LVC MOS25	16	<input type="checkbox"/>	<input type="checkbox"/>	-73.62	No
N24	address[19]	address1_obuf[19]/U0/U_JOP...	LVC MOS25	16	<input type="checkbox"/>	<input type="checkbox"/>	-49.89	No
P24	address[18]	address1_obuf[18]/U0/U_JO...	LVC MOS25	16	<input type="checkbox"/>	<input type="checkbox"/>	-59.92	No
L29	address[15]	address1_obuf[15]/U0/U_JO...	LVC MOS25	16	<input type="checkbox"/>	<input type="checkbox"/>	-76.35	No
J27	address[14]	address1_obuf[14]/U0/U_JO...	LVC MOS25	16	<input type="checkbox"/>	<input type="checkbox"/>	-62.78	No
L30	address[13]	address1_obuf[13]/U0/U_JO...	LVC MOS25	16	<input type="checkbox"/>	<input type="checkbox"/>	-75.03	No
P30	address[12]	address1_obuf[12]/U0/U_JO...	LVC MOS25	16	<input type="checkbox"/>	<input type="checkbox"/>	100.00	Yes
Bank3 (2.5v)								
A829	mio	mio_obuf/U0/U_JOPAD	LVC MOS25	6	<input type="checkbox"/>	<input type="checkbox"/>	70.52	Yes
W29	datao[20]	datao_obuf[20]/U0/U_JOPAD	LVC MOS25	12	<input type="checkbox"/>	<input type="checkbox"/>	74.07	Yes
U25	datao[15]	datao_obuf[15]/U0/U_JOPAD	LVC MOS25	12	<input type="checkbox"/>	<input type="checkbox"/>	68.75	Yes
V27	datao[14]	datao_obuf[14]/U0/U_JOPAD	LVC MOS33	12	<input type="checkbox"/>	<input type="checkbox"/>	75.30	Var

Figure 89 · 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:

- **Configure I/O in I/O Editor** – Allows you to reconfigure I/Os, such as changing the I/O Standard or the Pin Assignment or both to improve the noise margin.
Note: This menu item is only active when the I/O Editor is open.
- **Show in Chip Planner** – Allows you to cross-probe the selected I/Os in Chip Planner.
Note: This menu item is only active when the Chip Planner 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.

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

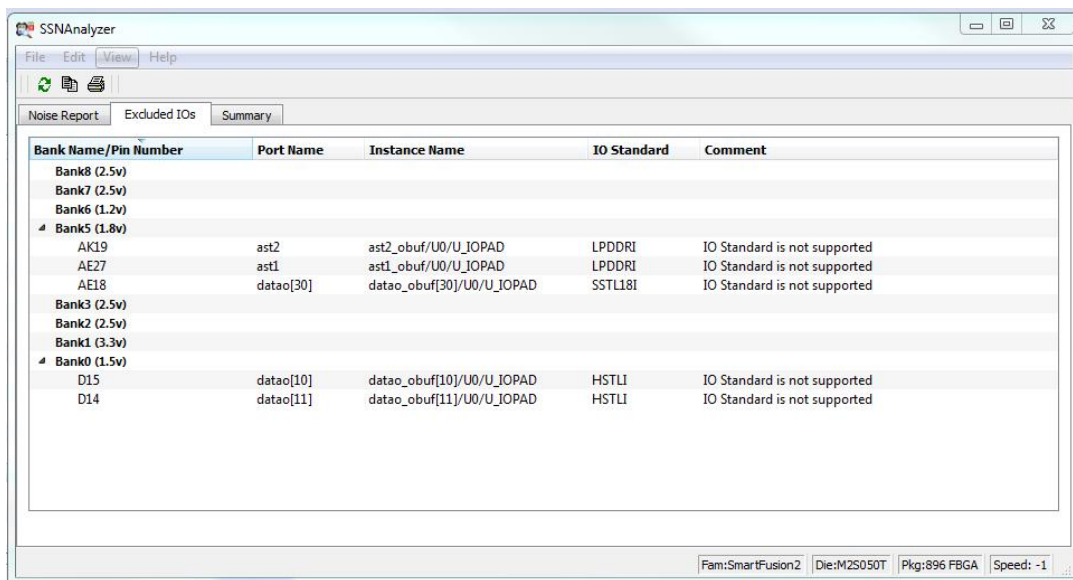


Figure 90 · SSN Analyzer – Excluded I/Os Tab

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.

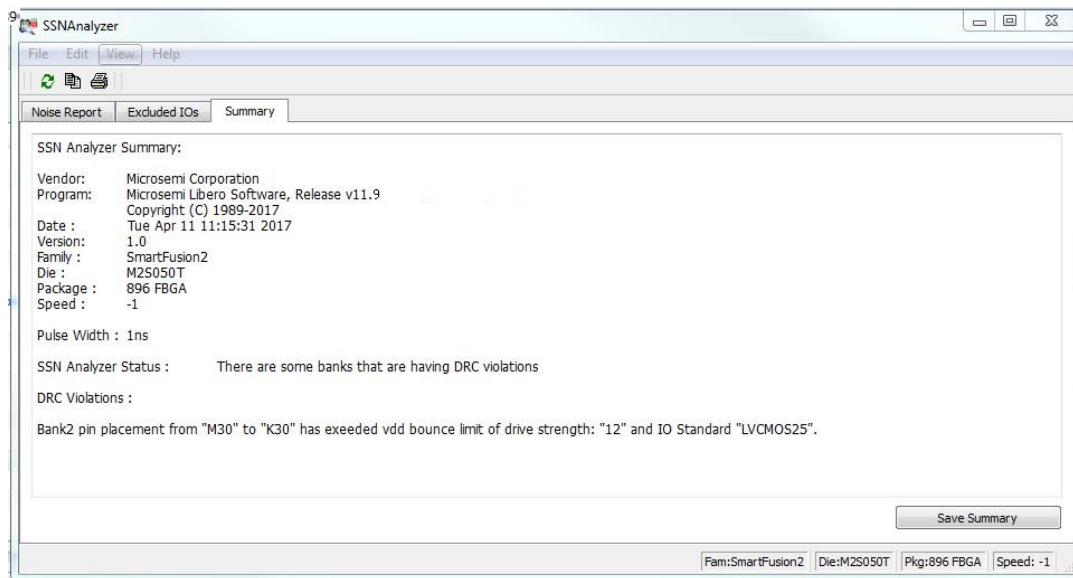


Figure 91 · SSN Analyzer - Summary

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.

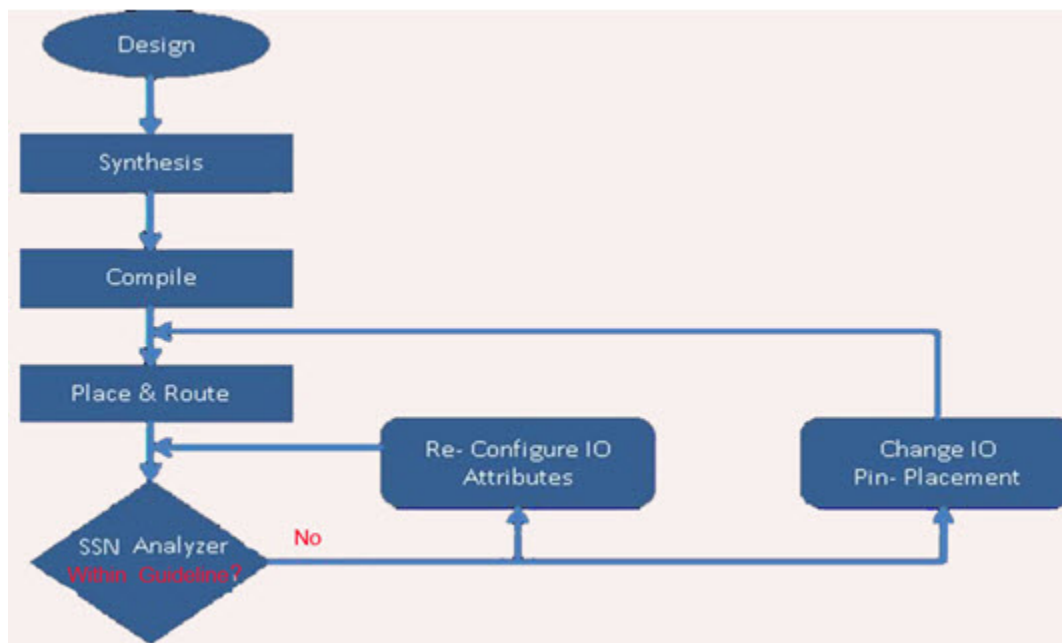


Figure 92 · SSN Analyzer in the Design Flow

See Also

[Simultaneous Switching Noise and Signal Integrity Application Notes](#)

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 FlashPro5 for SmartFusion2 and IGLOO2 devices, and by FlashPro6 for SmartFusion2, IGLOO2 devices. SPI Slave mode is not supported for RTG4 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

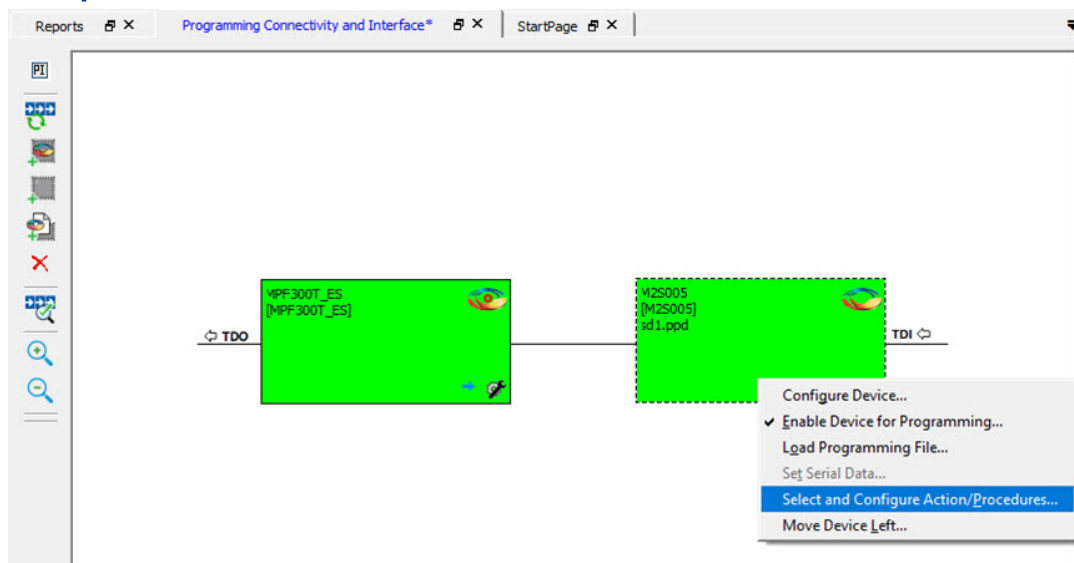


Figure 93 · Right-click Properties

- **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:
 - o **Actions** - List of programming actions for your device.
 - o **Procedures** - Advanced option; enables you to customize the list of recommended and optional procedures for an action.

To configure actions for Libero target devices, use the [Configure Actions and Procedures](#) tool.
- **Move Device Left/Right** – Move device in the chain to left or right.

See Also

[set_programming_interface](#)

[Configure Actions and Procedures](#)

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.

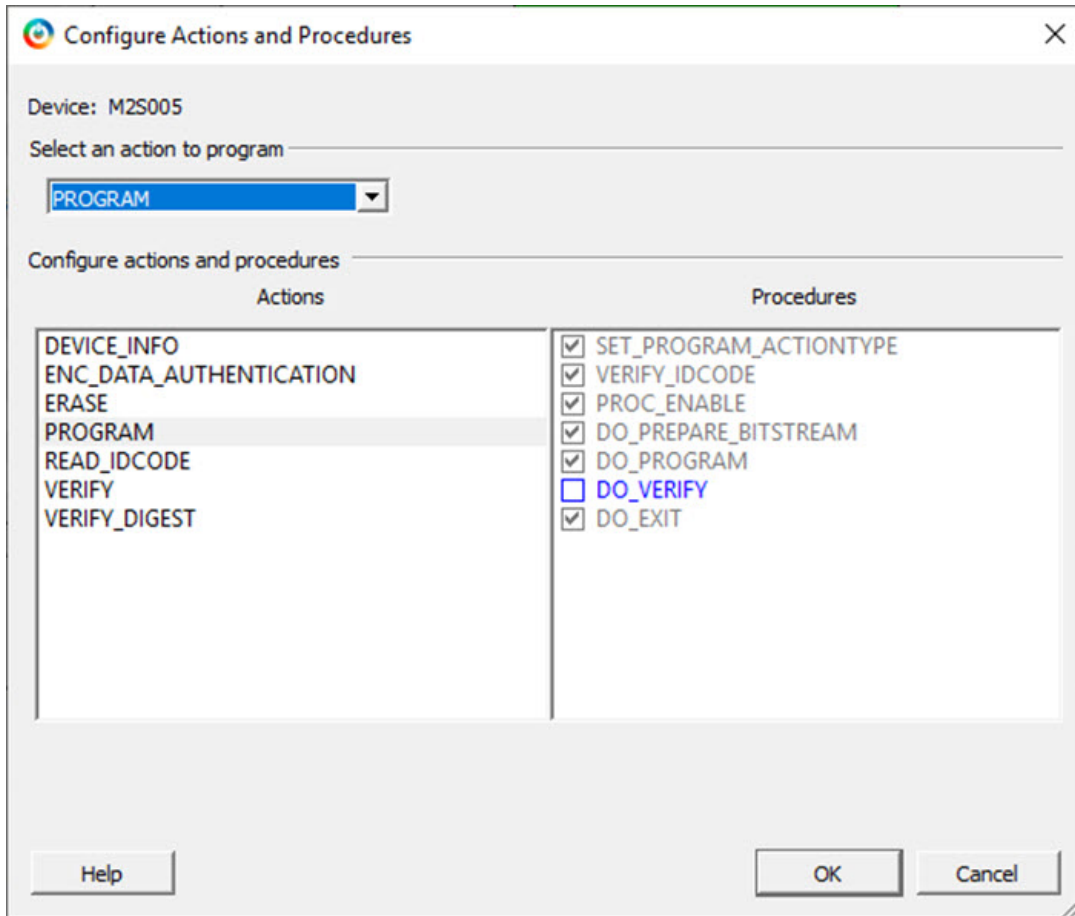


Figure 94 · Select and Configure Actions and Procedures

- Select an action to program - The selected action will be programmed in the Libero environment.
- Configure actions and procedures:
 - o **Actions** - List of programming actions for your device.
 - o **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 FlashPro5 for SmartFusion2 and IGLOO2 devices, and by FlashPro6 for SmartFusion2, IGLOO2 devices. SPI Slave mode is not supported for RTG4 devices.

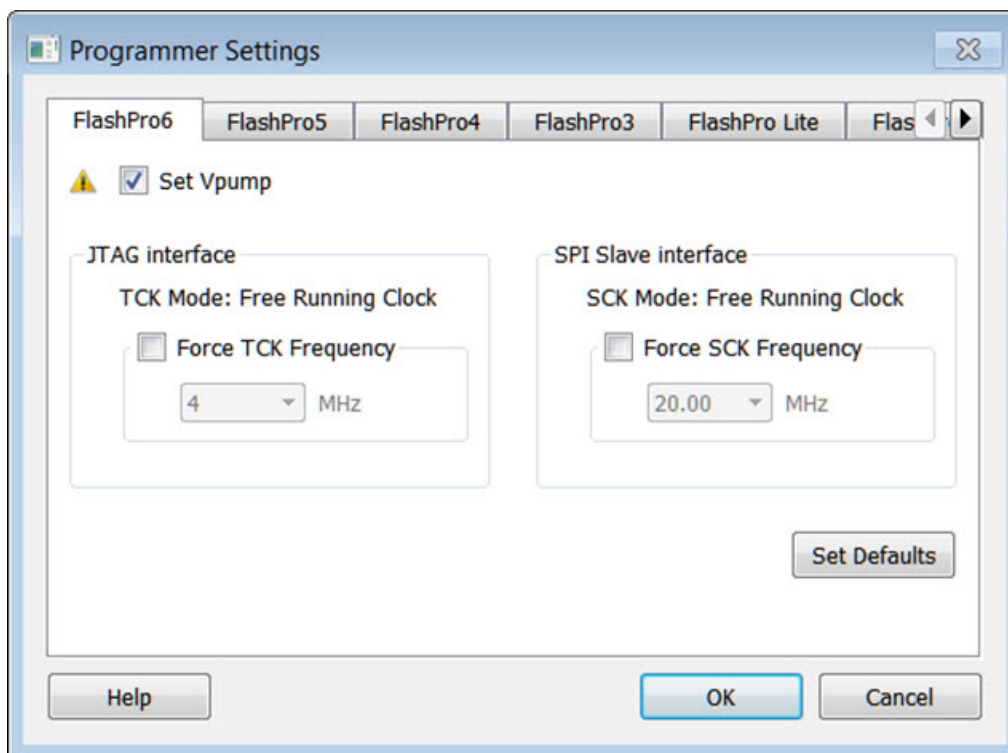


Figure 95 · Programmer Settings (FlashPro6 shown)

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, you can choose the Set Vpump setting or the Force TCK Frequency. 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 Vpump option is checked to instruct the FlashPro5/4/3/3X programmer(s) to supply Vpump to the device.
NOTE: VPUMP voltage will not be checked for the SmartFusion2/IGLOO2 and newer families of devices. VPUMP does not need to be connected to the programmer for these devices.
- 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.

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.

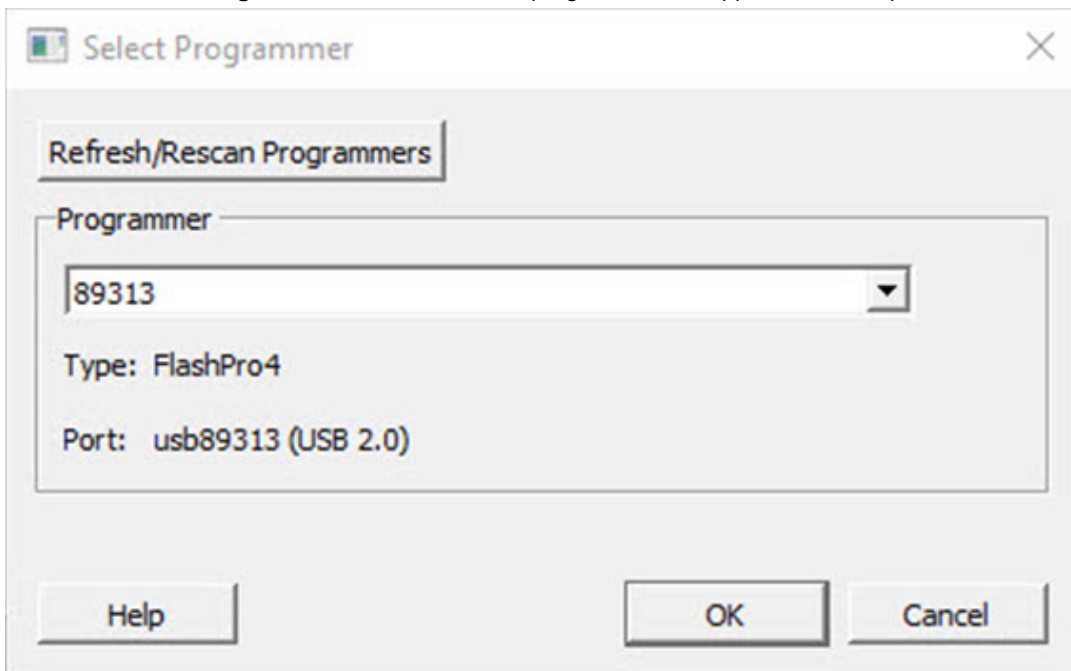


Figure 96 · Select Programmer Dialog

See Also

[Programmer Settings](#)

Tcl command [select_programmer](#)

Program Design

Generate FPGA Array Data

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

- *.map and *.dca files used for Programming

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.

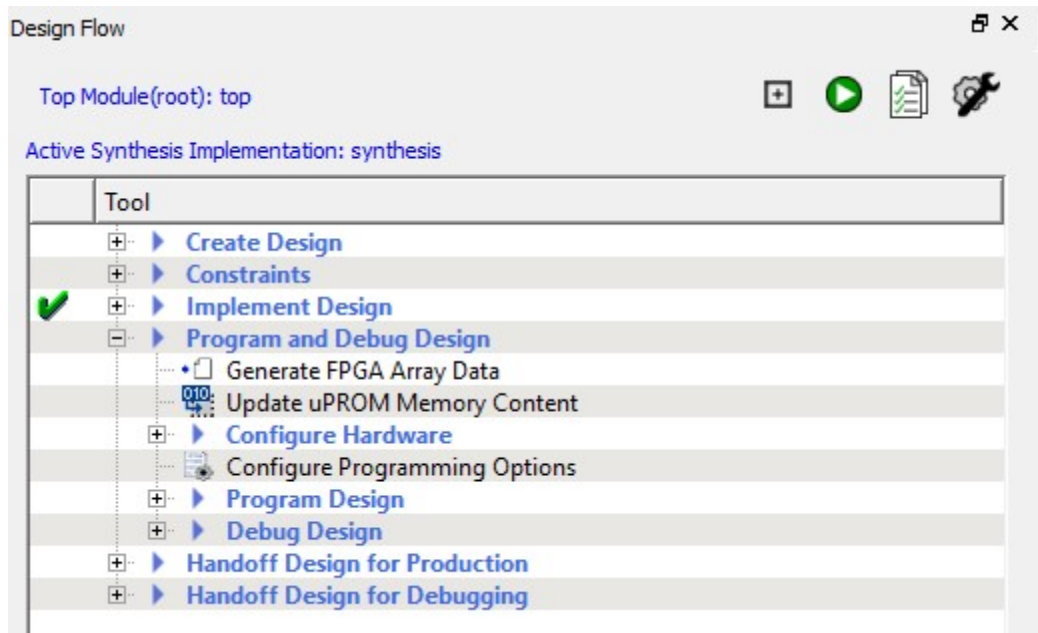


Figure 97 · Generate FPGA Array Data

Update uPROM Memory Content - RTG4 Only

Use the Update uPROM Memory Content tool if you have reserved space in the uPROM Configurator and, after Place and Route, you want to make changes to the uPROM clients. After you have updated the uPROM Memory Content, there is no need to rerun Place and Route.

To update the uPROM Memory Content from the Design Flow Window:

1. Right-click Update uPROM Memory Content in the Design Flow window and choose **Configure Options**.

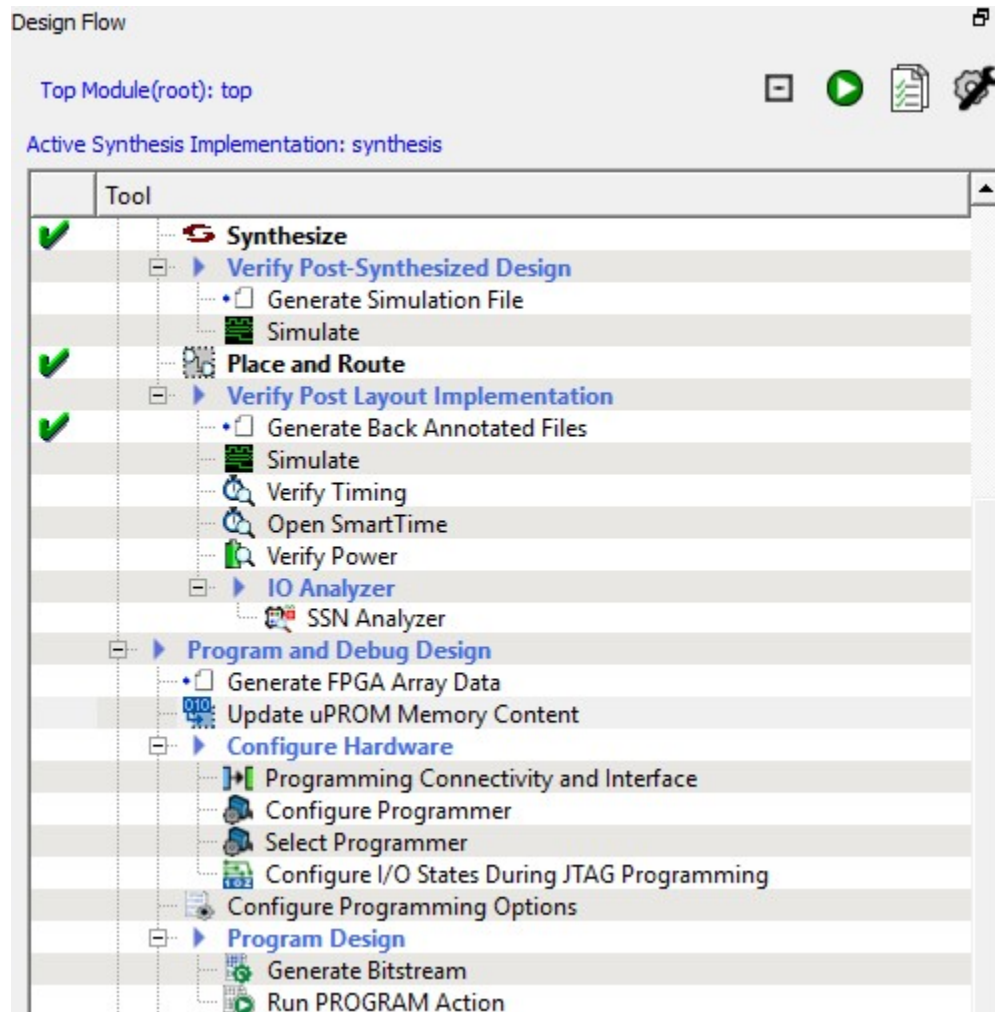


Figure 98 · Update uPROM Memory Content

- When the uPROM Update Tool appears, right-click the Memory Client you want to update and choose **Edit**.

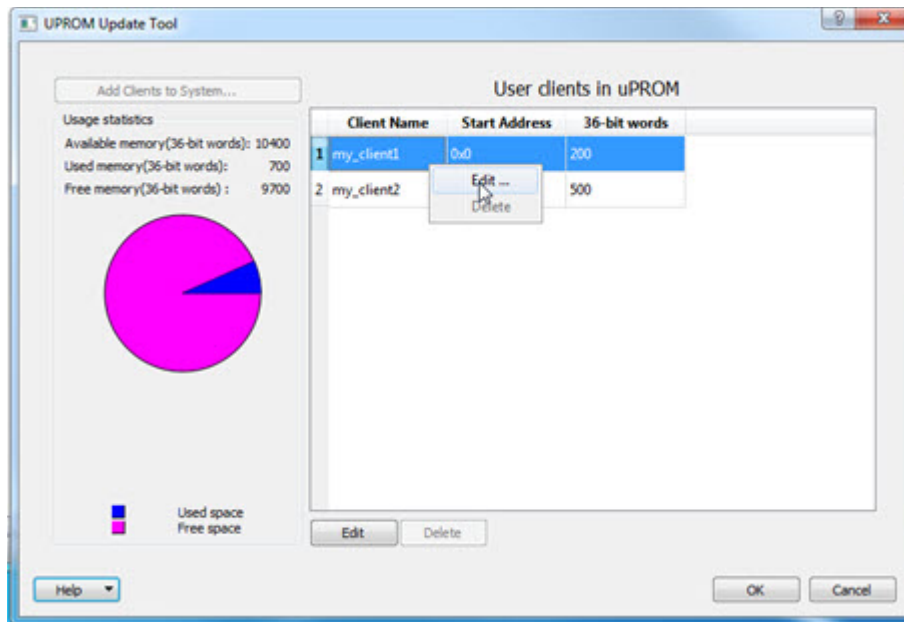


Figure 99 · uPROM Update Tool

The Edit Data Storage Client dialog box appears.

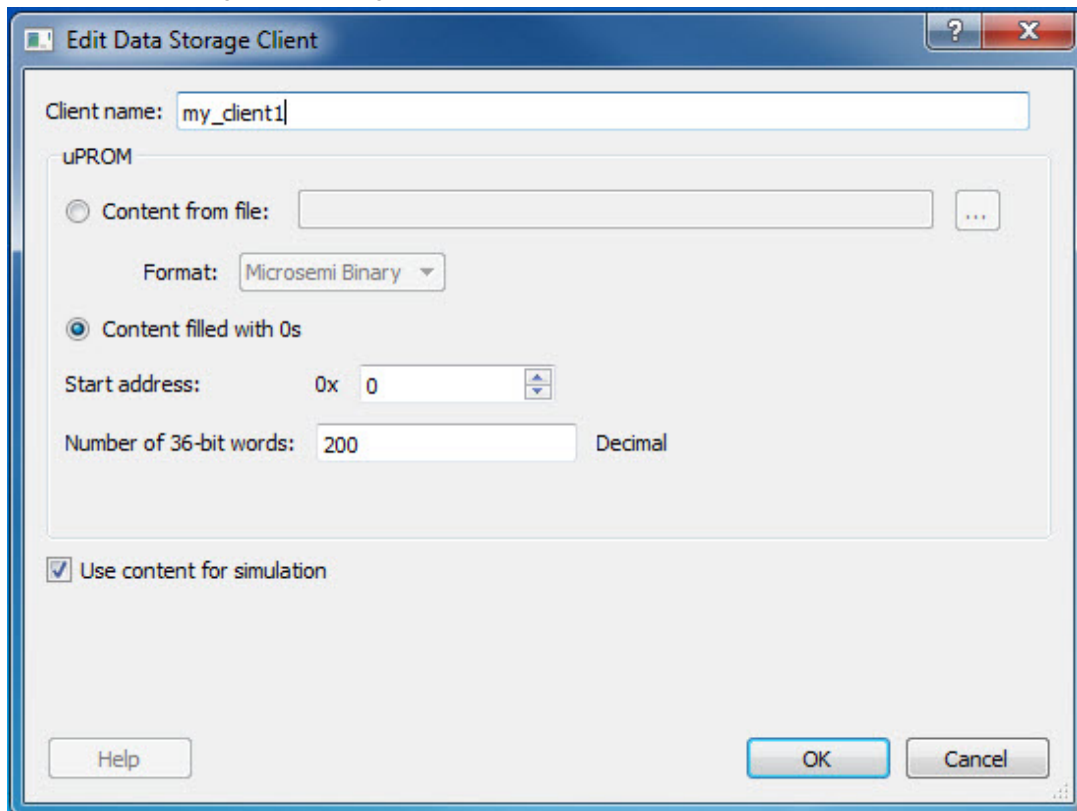


Figure 100 · Edit Data Storage Client Dialog Box

You can make the following changes to the uPROM client:

- Rename a Client
- Change the memory content, memory size and start address of the client
- Reverse your decision on whether or not to use Content for Simulation

Note: You cannot use the Update uPROM tool to add or delete a client. To add or delete a client, you must use the uPROM Configurator to reconfigure your Clients and regenerate your uPROM component and your design.

Update eNVM Memory Content (SmartFusion2 and IGLOO2)

Right-click **Update eNVM Memory Content** and choose **Configure Options** or double-click **Update eNVM Memory Content** to open the dialog box and modify your eNVM client configurations.

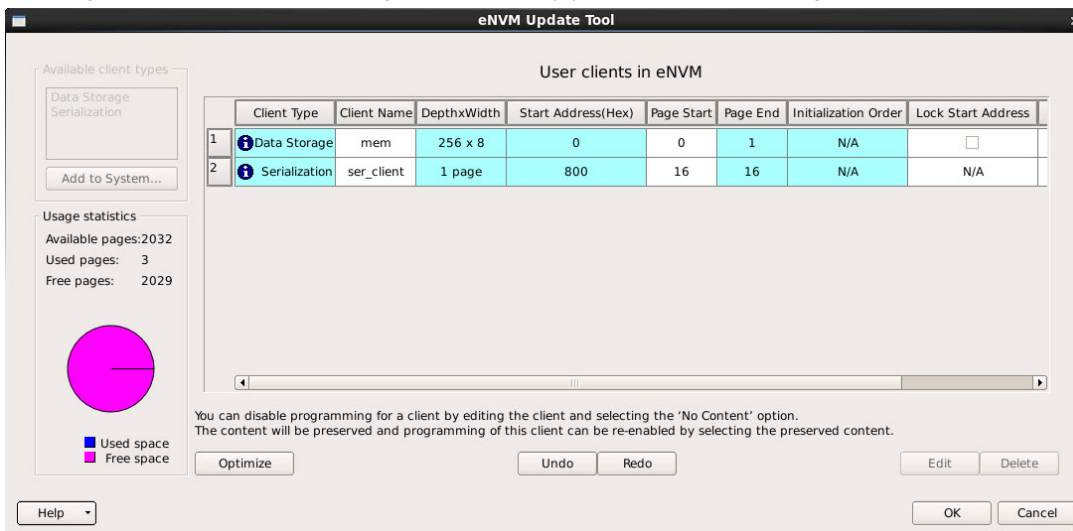


Figure 101 · eNVM Update Dialog Box

The eNVM Update dialog box enables you to update your eNVM content without having to rerun Compile and Place and Route. It is useful if you have reserved space in the eNVM configurator within the MSS for firmware development, for example. Use the eNVM Update dialog box when you have completed your firmware development and wish to incorporate your updated firmware image file into the project.

Note: To disable a client for programming, you must modify the client and select “No Content (Client is a placeholder and will not be programmed)”. The content from the memory file, serialization data file, or auto-incremented serialization content will be preserved if you later decide to enable this client for programming. Clients disabled for programming will not be included in the generated bitstream and will not be programmed.

Note: To delete, create, or rename a eNVM client, you must return to the MSS/System Builder eNVM Configurator. See [MSS Configuration - eNVM \(User Guide\)](#)

Modify Data Storage Client

Double-click the Storage Client to open the Modify Data Storage Client dialog box.

Note: You cannot add, delete or rename a data storage client at this point using the Modify Data Storage Client dialog box. To make such changes, return to the MSS or System Builder eNVM configuration step.

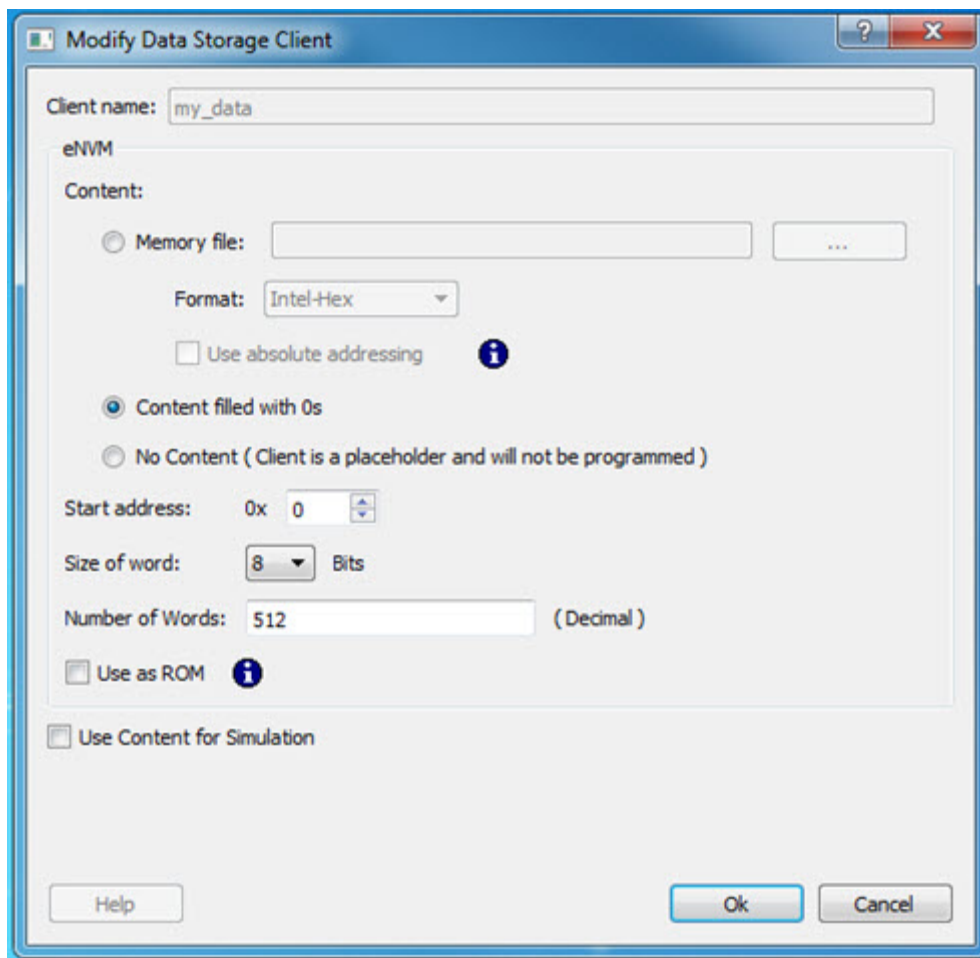


Figure 102 · Modify Data Storage Client Dialog Box

You have three options to specify the eNVM content:

- Import a Memory File
- Fill eNVM content with Zero's
- Assign No Content (eNVM as a Placeholder). The client will not be included in the programming bitstream and will not be programmed

If you have completed Place and Route and you import a memory file for the eNVM content, you do not have to rerun Compile or Place and Route. You can program or export your programming file directly. Programming will generate a new programming file that includes your updated eNVM content.

You can also specify the start address where the data for that client starts, the word size and the number of words to reserve for the data storage client.

Modify Serialization Client

Double-click the Serialization Client to open the Modify Serialization Client dialog box.

Note: You cannot add, delete or rename a Serialization Client in the Modify Serialization Client dialog box. Go to the eNVM configurator inside the MSS/HPMS Configurator or the System Builder Memory page (eNVM tab) to make these changes.

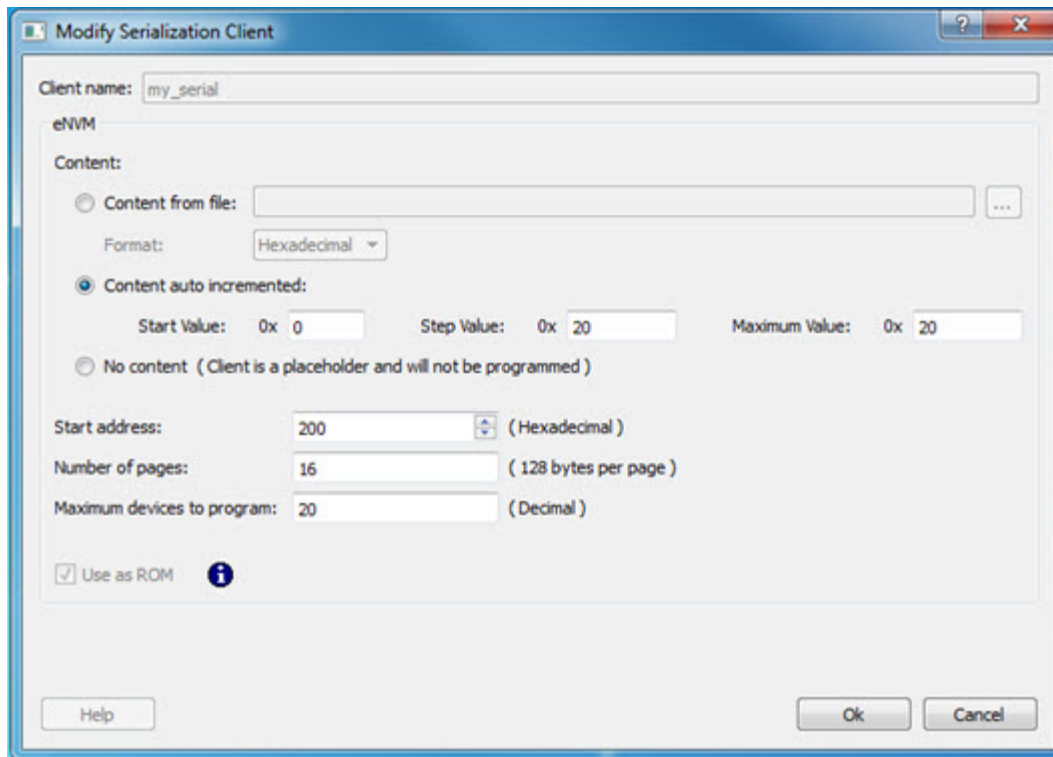


Figure 103 · Modify Serialization Client Dialog Box

You have three options to specify the eNVM content:

- Import a Memory File
- Fill eNVM content with Zero's
- Assign No Content (eNVM as a Placeholder). The client will not be included in the programming bitstream and will not be programmed

If you have completed Place and Route and you import a memory file for the eNVM content, you do not have to rerun Compile or Place and Route. You can program or export your programming file directly. Programming will generate a new programming file that includes your updated eNVM content.

You can also specify the start address where the data for the Serialization Client starts, the number of pages and the maximum number of devices you want to program serialization data into.

Setting a maximum number of devices to program for Serialization clients will generate a programming bitstream file that has serialization content for the number of devices specified. The maximum number of devices to program must match for all serialization clients. If the user would like to program a subset of the devices during production programming, this can be done within the FlashPro Express tool, which allows you to select a range of indices desired for programming for that serialization programming job session. Refer to the [FlashPro Express User Guide](#) for more information.

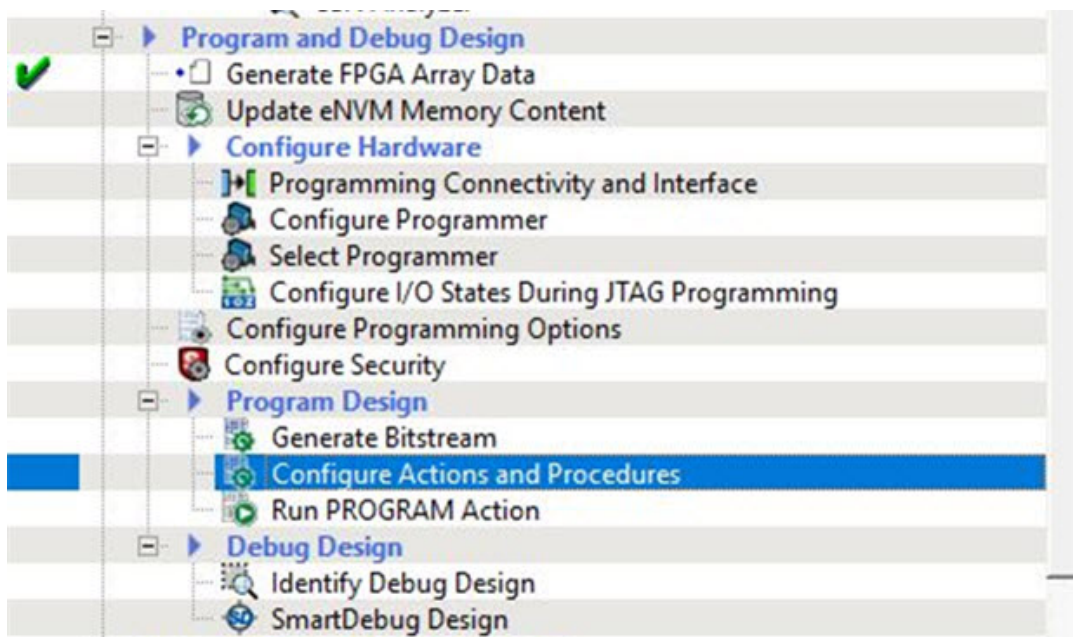
Configure Actions and Procedures

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.

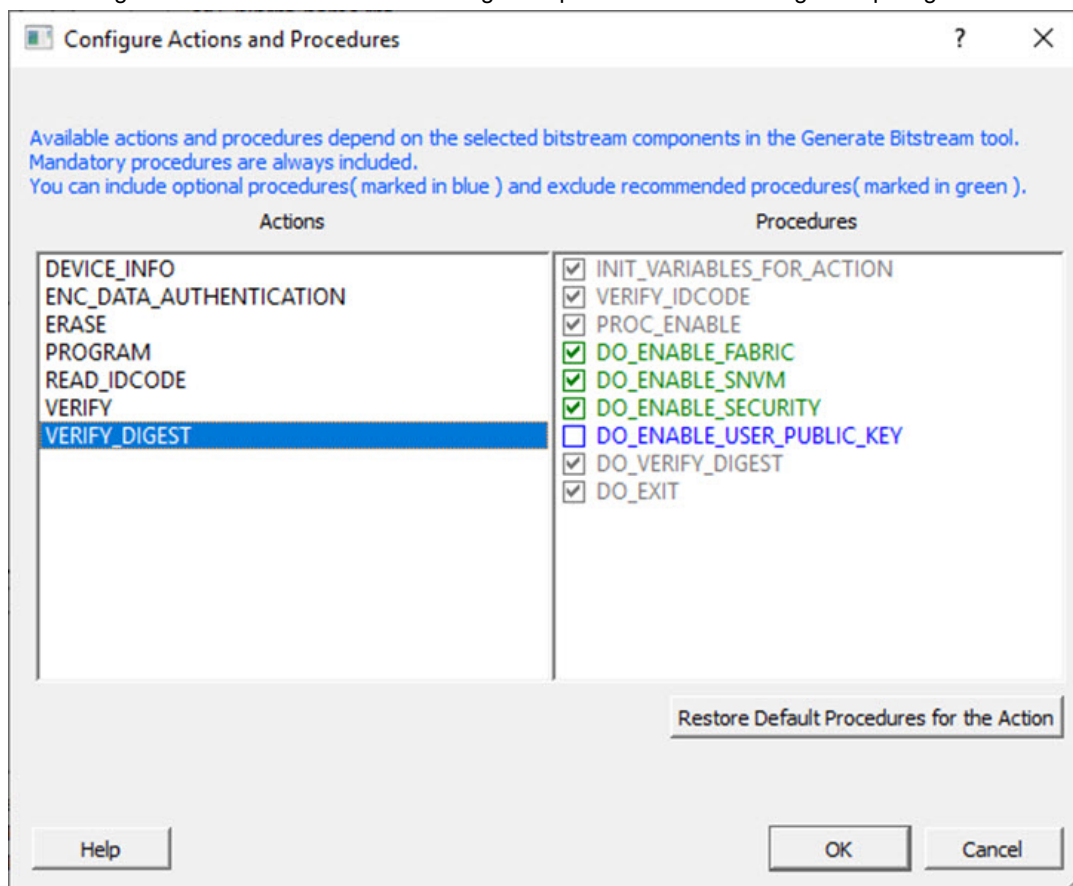


Figure 104 · Configure Actions and Procedures Dialog Box Example

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 3 · Programming File Actions and Supported Procedures

Action	Procedures
DEVICE_INFO	SET_DEVICE_INFO_ACTIONTYPE VERIFY_IDCODE DO_READ_CERTIFICATE DO_DEVICE_INFO DO_EXIT
ENC_DATA_AUTHENTICATION	SET_AUTHENTICATION_ACTIONTYPE VERIFY_IDCODE DO_AUTHENTICATION DO_EXIT
ERASE	SET_ERASE_ACTIONTYPE VERIFY_IDCODE DO_READ_CERTIFICATE PROC_ENABLE DO_SETUP_ENVM DO_PREPARE_BITSTREAM DO_ERASE DO_POST_SETUP_ENVM DO_EXIT
PROGRAM	SET_PROGRAM_ACTIONTYPE VERIFY_IDCODE DO_READ_CERTIFICATE PROC_ENABLE DO_SETUP_ENVM DO_PREPARE_BITSTREAM DO_ERASE_SECURED DO_PROGRAM DO_VERIFY (optional) DO_POST_SETUP_ENVM DO_EXIT
READ_IDCODE	VERIFY_IDCODE DO_READ_CERTIFICATE PRINT_IDCODE DO_EXIT
VERIFY	SET_VERIFY_ACTIONTYPE VERIFY_IDCODE DO_READ_CERTIFICATE PROC_ENABLE DO_SETUP_ENVM DO_PREPARE_BITSTREAM DO_VERIFY

Action	Procedures
	DO_POST_SETUP_ENVM DO_EXIT
VERIFY_DIGEST	VERIFY_IDCODE DO_READ_CERTIFICATE PROC_ENABLE DO_VERIFY_DIGEST DO_EXIT

The table below lists programming file actions and descriptions.

Table 4 · Programming File Actions

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

Options Available in Programming Actions

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

Table 5 · Programming File Actions - Options

Action	Option and Description
PROGRAM	DO_VERIFY - Enables or disables programming verification

To configure actions for other JTAG devices use the [Programming Connectivity and Interface](#) tool.

Configure I/O States During JTAG Programming

In the Libero SoC Design Flow window expand **Edit Design Hardware Configuration** 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.

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)

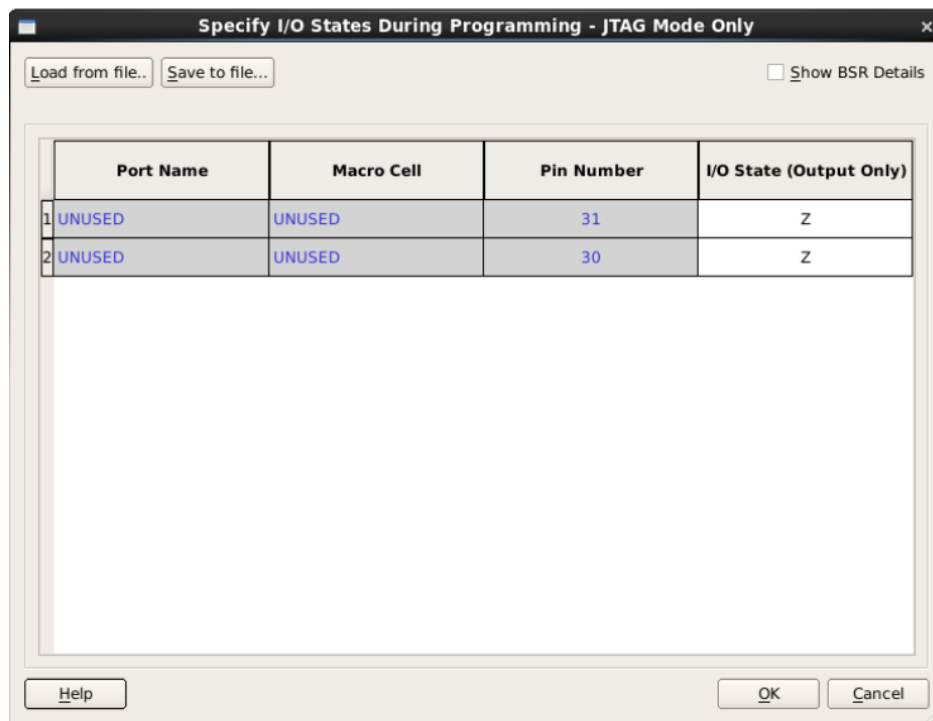


Figure 105 · I/O States During Programming Window

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 (SmartFusion2 and IGLOO2)

From the Design Flow window, double-click **Configure Programming options** or right-click and choose **Configure Options**.

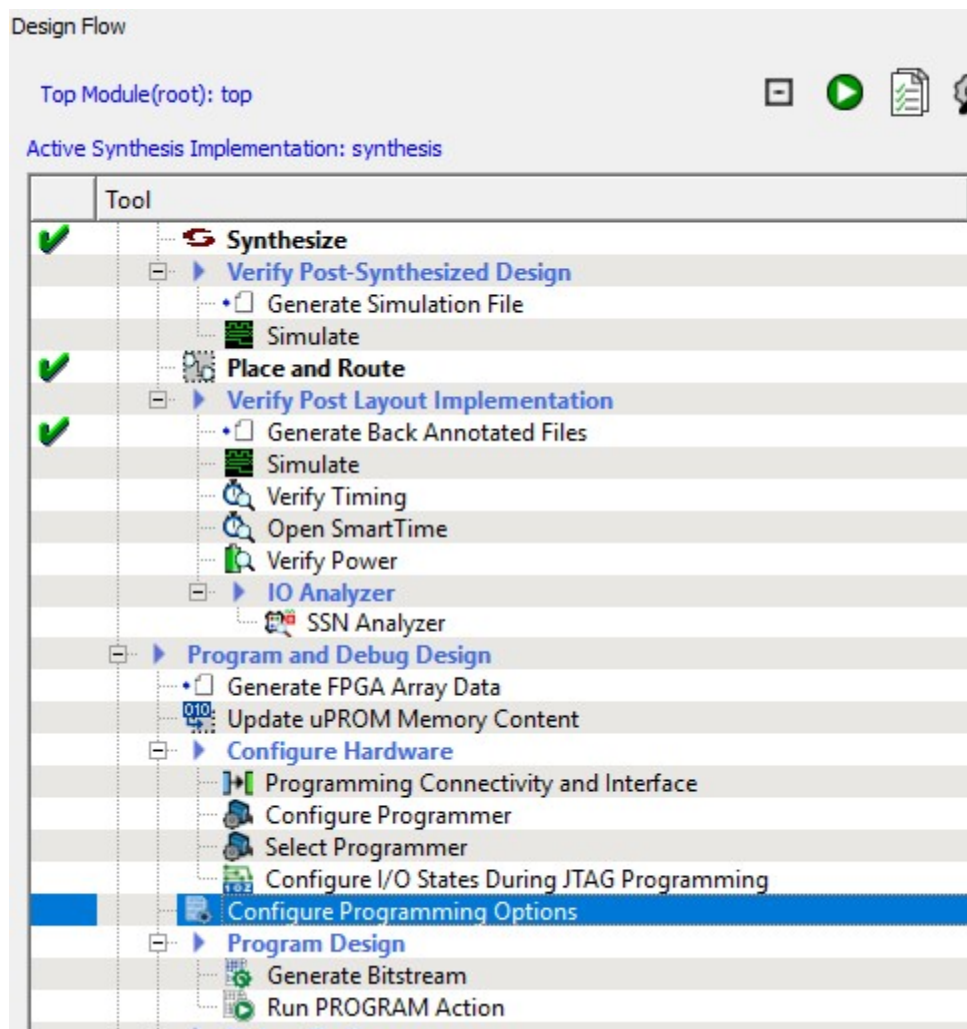


Figure 106 · Configure Programming Bitstream Settings - Configure Options

The Configure Programming Options dialog box appears. This is where you configure programming options.

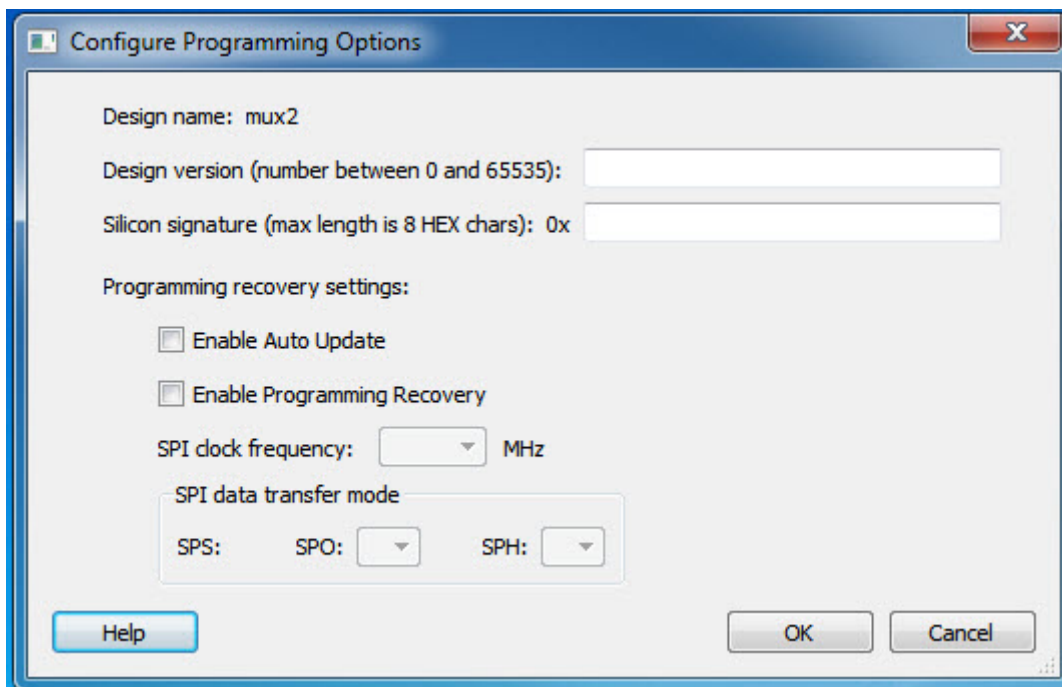


Figure 107 · Configure Programming Options Dialog Box

Options

Design Version - Enter a number between 0 and 65535 for the design version. This is the Design Version used for Auto Update Programming or for Backlevel protection.

Silicon signature (Hex) - Enter up to eight hexadecimal characters

Programming recovery settings:

The Programming Recovery settings enable you to set your Auto Update and Programming Recovery options for programming.

Auto Update takes place during power-up and compares your Update SPI image Design Version against the Design Version programmed in the device. It performs Auto Update programming on your SPI update Image if:

- The device has been programmed AND
- The Update SPI image Design Version is greater than the Design Version on the device

Auto Recovery enables the device to automatically reprogram itself if there is a power failure during programming.

Enable Auto Update - Click the checkbox to auto update the SPI update image at power up. Auto-update occurs only when the SPI update image Design Version is greater than the Design Version already on the device. When enabling Auto Update, Programming Recovery must also be enabled and this checkbox will be disabled.

Enable Programming Recovery - Click the checkbox to enable programming recovery in the event of a power failure during programming.

SPI clock frequency - Sets your SPI clock frequency. SPI is a full duplex, four-wire synchronous transfer protocol that supports programmable clock polarity (SPO) and clock phase (SPH). The state of SPO and SPH control bits decides the data transfer modes. See the [SmartFusion2 Microcontroller Subsystem User's Guide](#) or the [IGLOO2 High Performance Memory Subsystem User's Guide](#) for more information.

Select one of the following for the SPI Clock Frequency Values (MHz):

- 1.00
- 2.08
- 3.13
- 4.16

- 5.00
- 6.25
- 8.30
- 12.50

SPI data transfer mode - Sets your SPI data transfer mode for SPO and SPH. The SPO control bit determines the polarity of the clock and SPS defines the slave select behavior. SPS is hardcoded to b'1 and cannot be changed. The SPH control bit determines the clock edge that captures the data. See the [SmartFusion2 Microcontroller Subsystem User's Guide](#) or the [IGLOO2 High Performance Memory Subsystem User's Guide](#) for more information.

Notes:

Programming Recovery cannot be updated with _UEK1 or _UEK2 programming files. Only the master programming file can be used.

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

Configure Programming Options (RTG4 Only)

From the Design Flow window, double-click **Configure Programming options** or right-click and choose **Configure Options**.

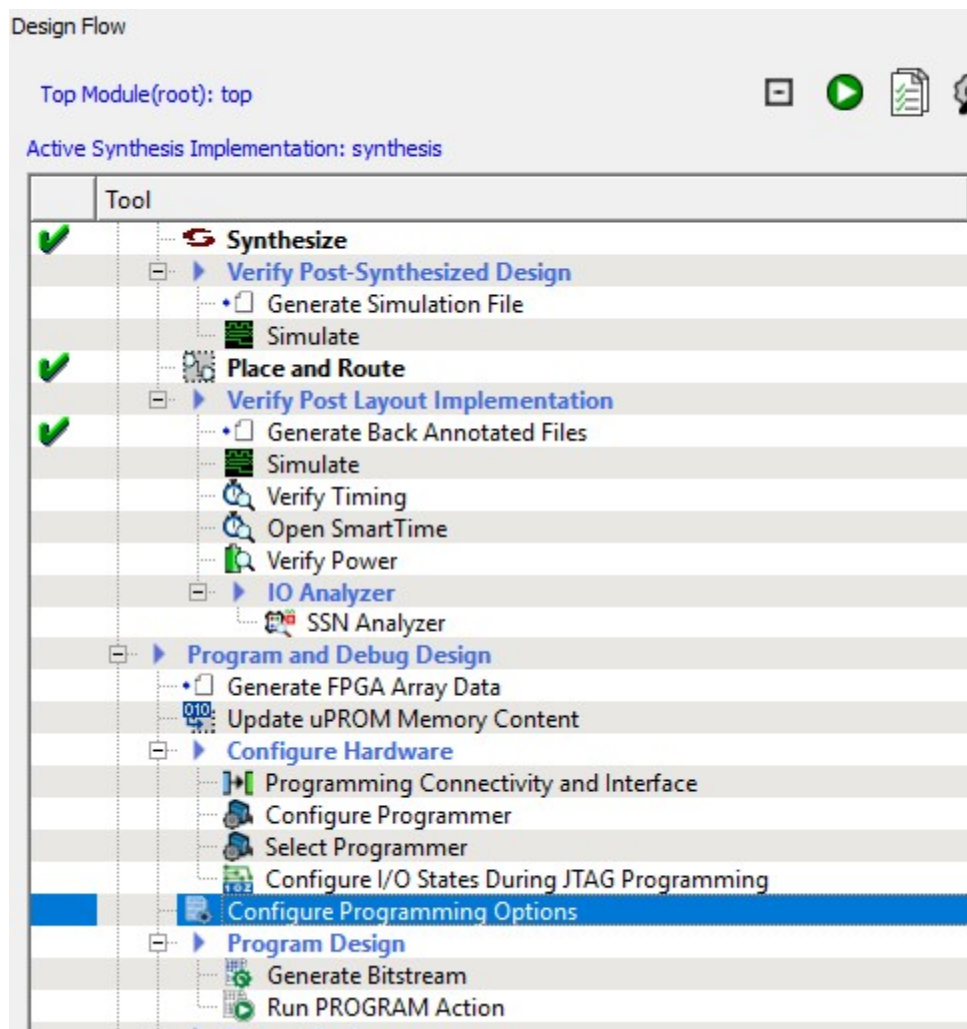


Figure 108 · Configure Programming Bitstream Settings - Configure Options

The Configure Programming Options dialog box appears. This is where you configure programming options.

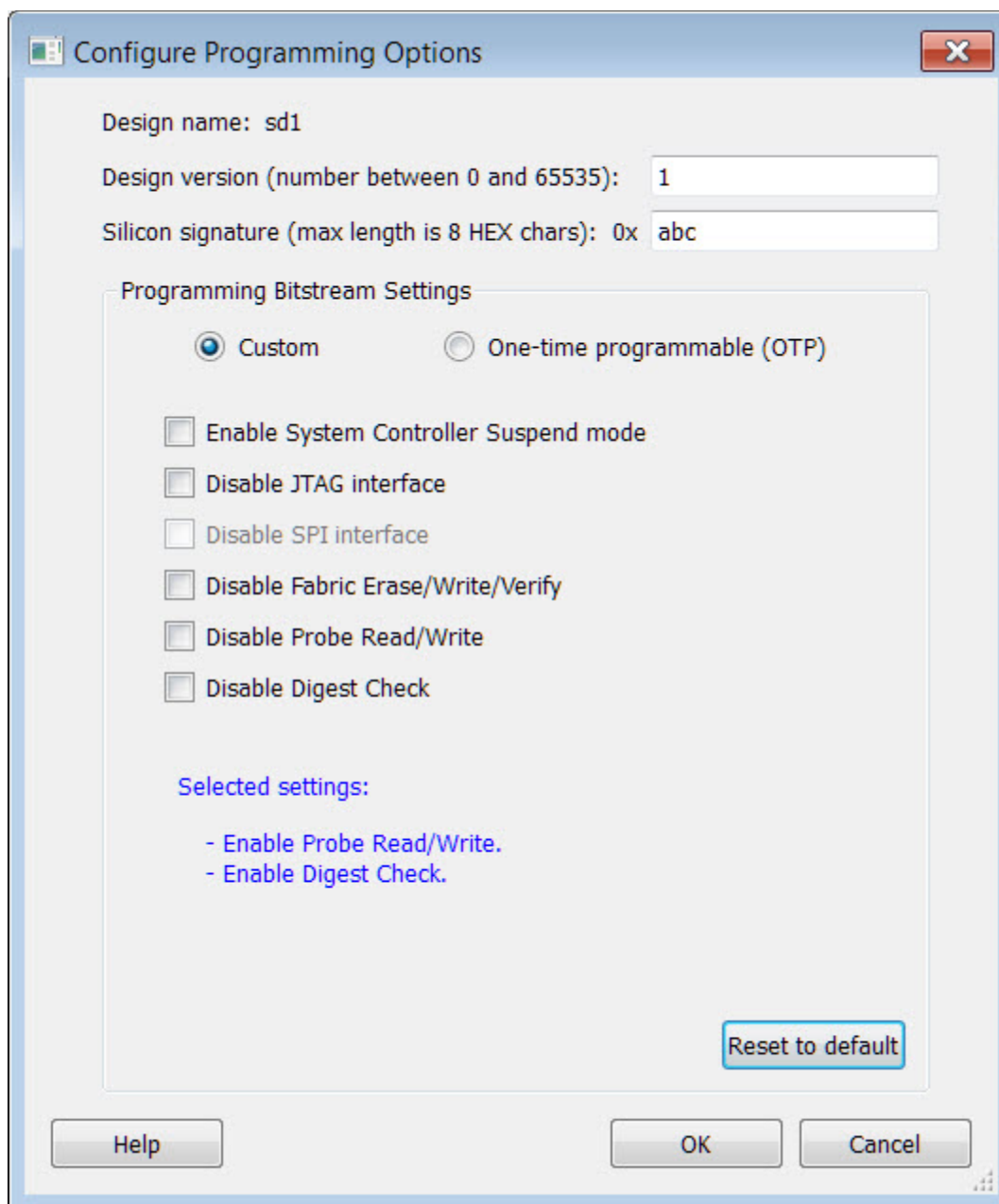


Figure 109 · Programming Bitstream Settings Dialog Box (with Custom selected)

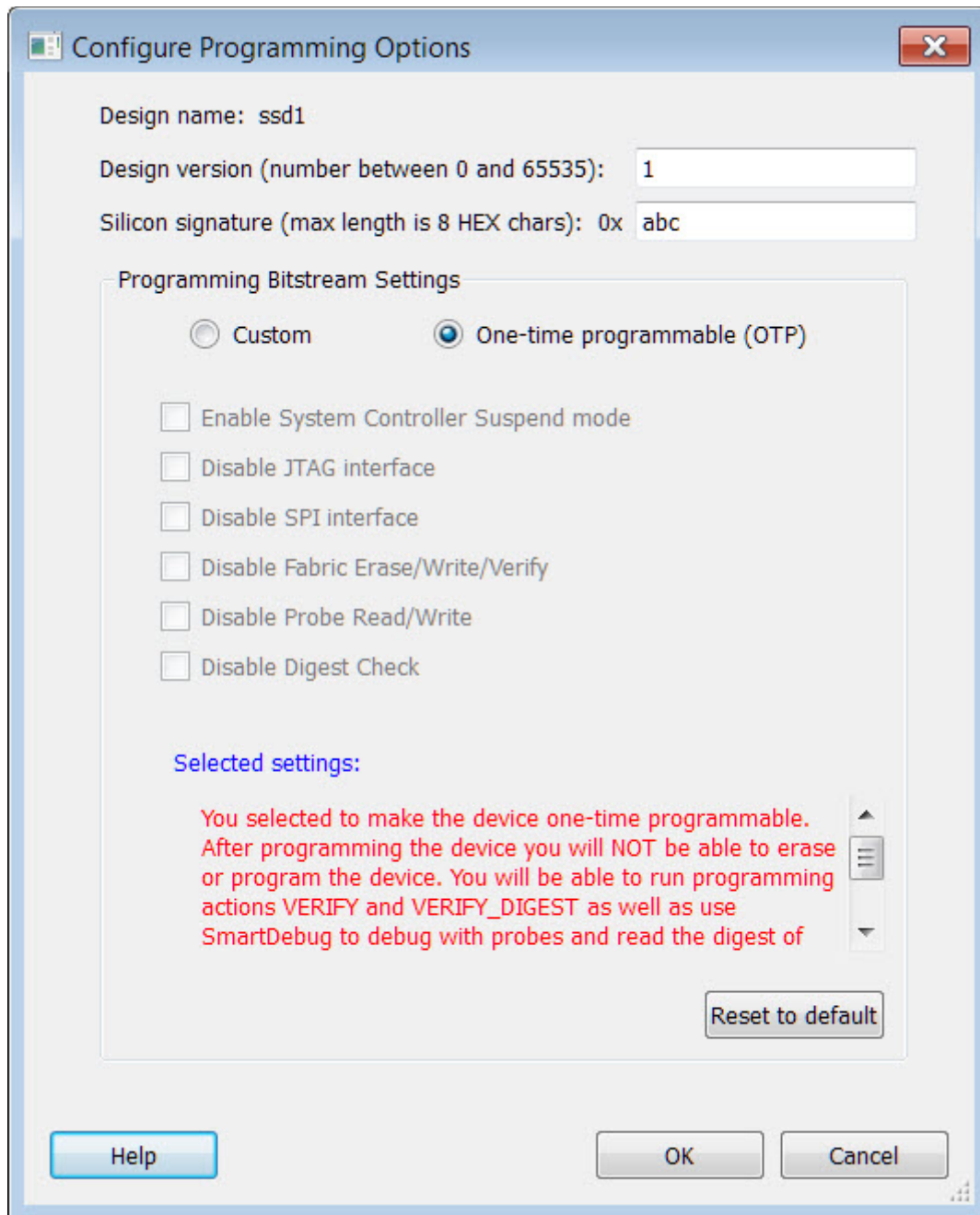


Figure 110 · Programming Bitstream Settings Dialog Box (with One-time programmable (OTP) selected)

Options

Design Version - Enter a number between 0 and 65535 for the design version.

Silicon signature (Hex) - Enter up to eight hexadecimal characters

Bitstream Settings

Custom

One-time programmable (OTP) - Select this option to make the device one-time programmable. After programming the device, you will not be able to erase or reprogram the device. You will be able to run

programming actions VERIFY and VERIFY_DIGEST, as well as use SmartDebug to debug with probes and read the digest of the Fabric.

Note: Refer to Table 4 in the RTG4 FPGA Datasheet for the maximum number of Verify Cycles per Program/Erase cycle after making the device one-time programmable.

Enable System Controller Suspend Mode – Check this box to enable System Controller Suspend Mode when TRSTB is low during device power up. You can exit System Controller Suspend Mode by driving TRSTB high during device power up. By default, this selection is not checked.

Note: By default, when this options is selected, the JTAG interface will be disabled to ensure proper hardening during System Controller Suspend Mode.

Disable JTAG Interface – Check this box to disable the JTAG interface when TRSTB is low during device power up. You can enable the JTAG interface by driving TRSTB high during device power up. By default, this selection is not checked.

Disable SPI Interface – This box is grayed out; the SPI interface is not supported for RGT4.

Note: If JTAG interface is disabled, the following settings will all be disabled for selection.

Disable Fabric Erase/Write/Verify - Check this box to disable Fabric Erase/Write/Verify when TRSTB is low during device power up. You can enable Fabric Erase/Write/Verify by driving TRSTB high during device power up. By default, this selection is not checked.

Disable Probe Read/Write – Check this box to disable Probe Read/Write when TRSTB is low during device power up. You can enable Probe Read/Write by driving TRSTB high during device power up. By default, this selection is not checked.

Note: For this option to be available, you must reserve pins for Probe in the project settings of the Libero project(Project > Project Settings > Device Settings).

Disable Digest Check – Check this box to disable all Fabric reads, such as verify digest, read digest, or reading design or programming information in DEVICE_INFO when TRSTB is low during device power up. You can enable Digest Check by driving TRSTB high during device power up.

Reset to default – Click to reset the Settings to the default values.

Selected device options: This section provides a summary of the settings configured and informs the user about the expected behavior of the device with these options.

Configure Security

Configure Security Policy Manager

In the Design Flow window, double-click **Configure Security** to open the Security Policy Manager dialog box and customize the security settings in your design.

Use this dialog box to set your User Keys, Security Policies, and Microsemi factory test mode access level.

Note: Microsemi enabled default bitstream encryption key modes are disabled after user security is programmed.

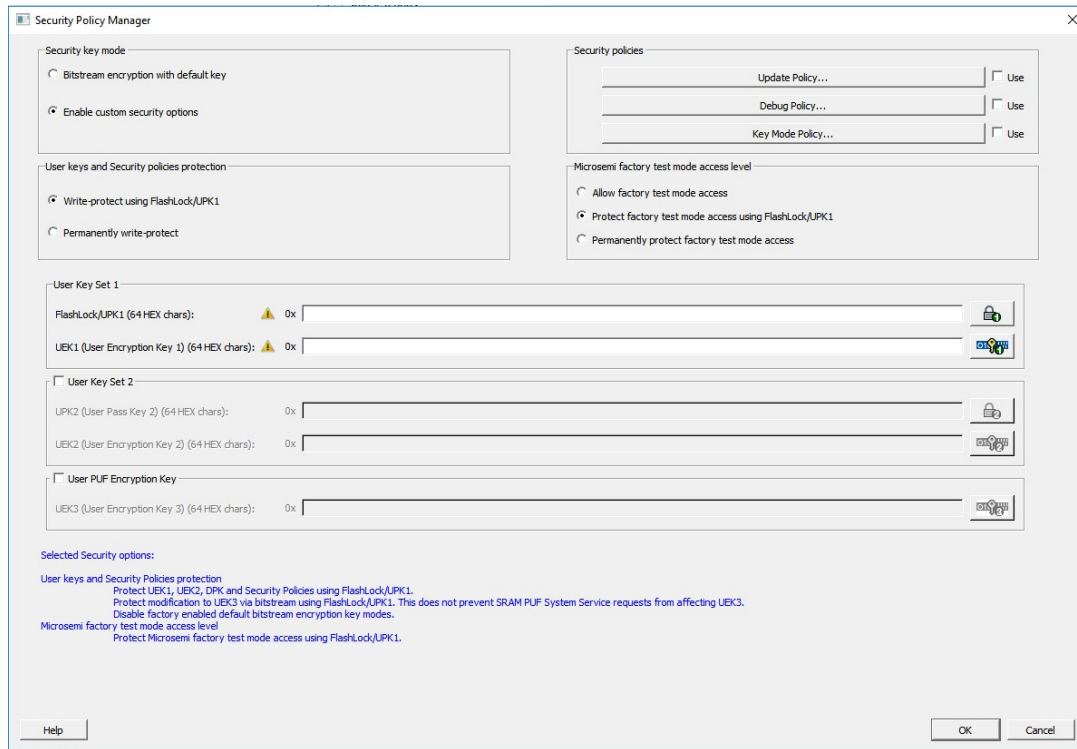


Figure 111 · Security Policy Manager Dialog Box

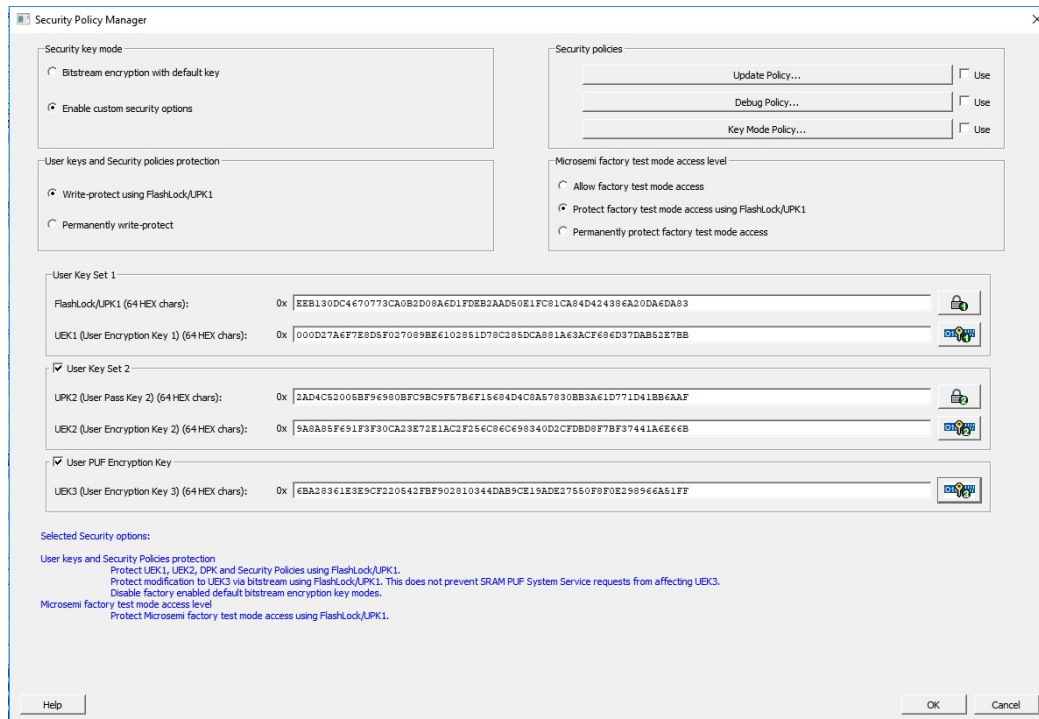


Figure 112 · Security Policy Manager Dialog Box (for devices supporting UEK3)

Security Key Mode

- **Bitstream encryption with default key** - Encrypt bitstream files with Microsemi default key (pre-placed key in silicon). When this option is selected, user keys, security and Microsemi factory test mode access level configurations are disabled.

- **Enable custom security options** - Enables you to set User Keys, Security Policies and Microsemi factory test mode access level (see below for a description).

User keys and Security policies protection

- **Write-protect using FlashLock/UPK1** - Protect UEK1 (User Encryption Key 1), UEK2 (User Encryption Key 2), DPK (Debug Pass Key) and Security Policies using FlashLock/ UPK1. Protect modification to UEK3 via bitstream using FlashLock/UPK1. Note that even after programming Security settings, SRAM-PUF System services can still modify UEK3.

Note: UEK2 (User Encryption Key2) is protected by UPK2 (User Pass Key 2).

Note: UEK3 is only available for M2S060, M2GL060, M2S090, M2GL090, M2S150 and M2GL150 devices. See the [SmartFusion2 SoC FPGA and IGLOO2 FPGA Security Best Practices User Guide](#) for more details.

- **Permanently write-protect** - Permanently protect UEK1 (User Encryption Key 1), UPK2 (User Pass Key 2), UEK2 (User Encryption Key 2), DPK (Debug Pass Key), Security Policies, and Microsemi factory test mode access level. Permanently protect modification to UEK3 via bitstream. Note that even after programming Security settings, SRAM-PUF System services can still modify UEK3. This setting, once programmed will not be modified in the device. Microsemi enabled default bitstream encryption key modes are permanently disabled as well.

Note: When this option is selected, you cannot specify the FlashLock/UPK 1 and UPK2 (User Pass Key 2) value, since the value cannot be used to unlock the corresponding protected features.

Note: UEK3 is only available for M2S060, M2GL060, M2S090, M2GL090, M2S150 and M2GL150 devices. See the [SmartFusion2 SoC FPGA and IGLOO2 FPGA Security Best Practices User Guide](#) for more details.

Microsemi Factory Test Mode Access Level

Allow factory test mode access - Allows access to Microsemi factory test mode.

- **Protect factory test mode access using FlashLock/UPK1** - Protects access to Microsemi factory test mode using Flashlock/ UPK1.
- **Permanently protect factory test mode access** - Permanently locks access to Microsemi factory test mode.

Note: When this option is selected, User Key Set 2 is permanently write-protected. Once programmed, User Key Set 2 cannot be changed in the device. You can specify UEK2 (User Encryption Key 2). However, you cannot specify UPK2 (User Pass Key 2), since the value cannot be used to unlock User Key Set 2.

Security Policies

- **Update Policy** - Sets your Fabric, eNVM and Back Level protections. It also allows you to disable access to certain programming interfaces. See the [Update Policy topic](#) for more information.

Note: If Update Policy is enabled and Fabric update is protected by UPK1:

Fabric update is disabled for Auto Programming, IAP/ISP services, Programming Recovery and Auto update. FlashLock/UPK1 unlocking is only available for JTAG and SPI slave programming.

See the following example.

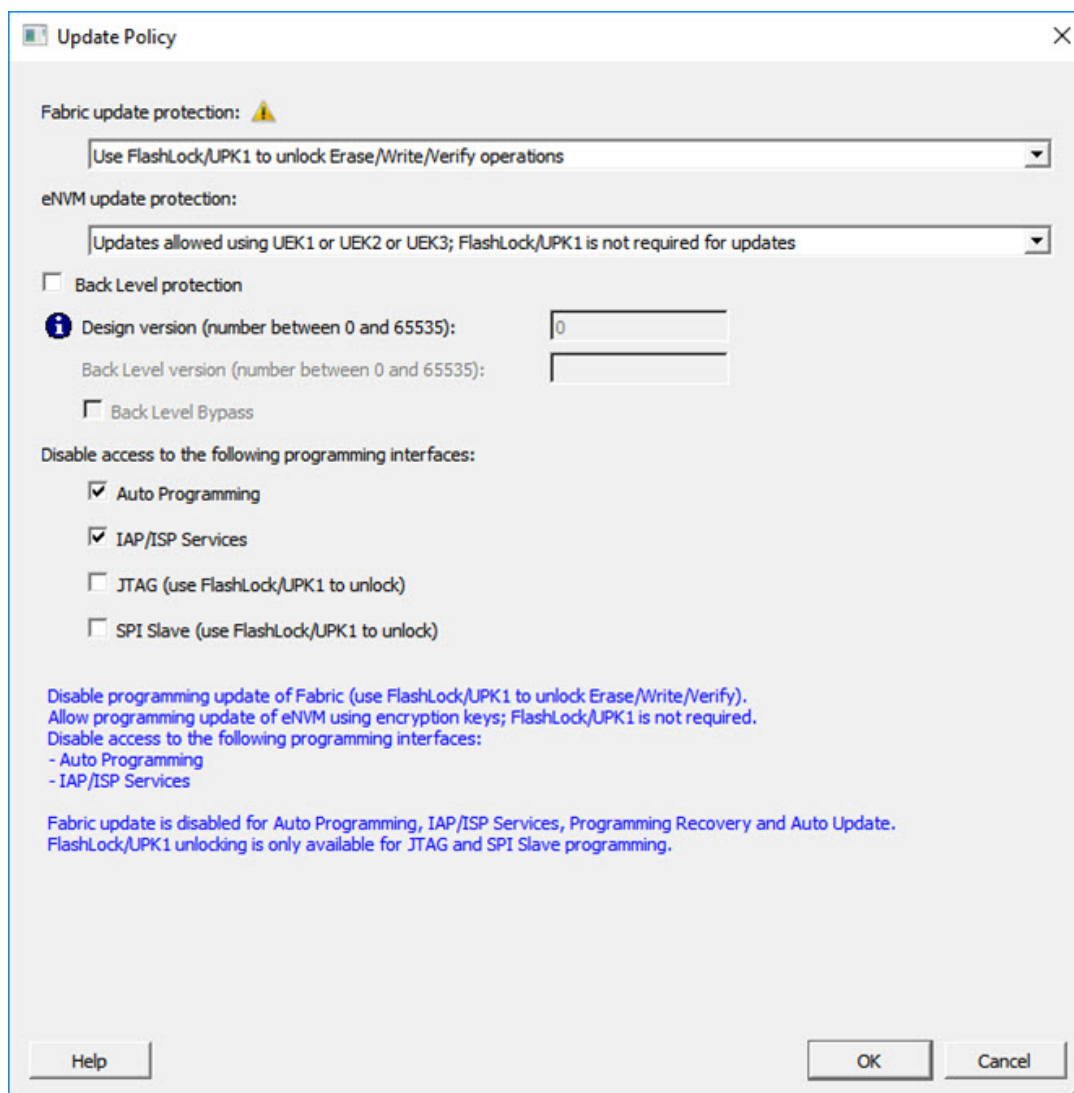


Figure 113 · Update Policy Dialog Box denoting Fabric update protection by Flashlock/UPK1

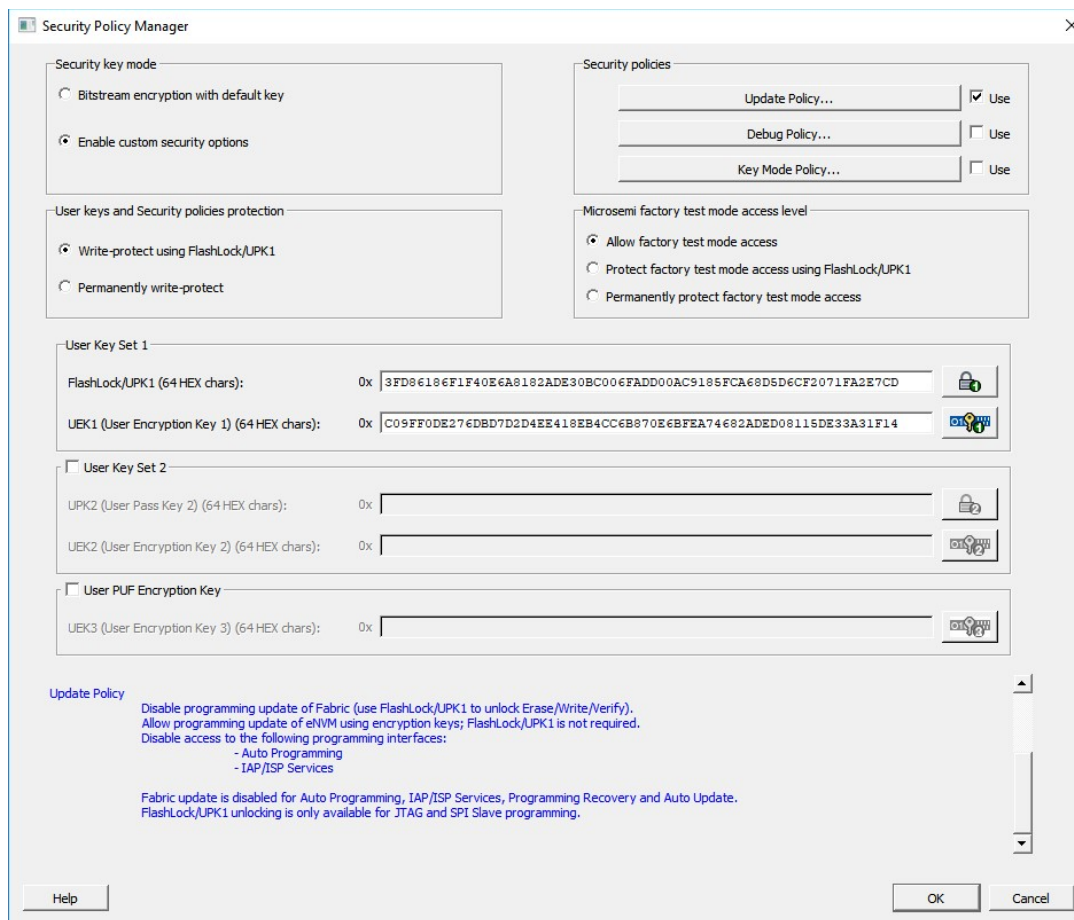


Figure 114 · Security Policy Manager with Update Policy description for Fabric Update Protection by Flashlock/UPK1

- **Debug Policy** - Enables and sets your Debug Pass Key and debug options. See the [Debug Policy topic](#) for more information.
- **Key Mode Policy** - Configures the key mode to enable or disable. See the [Key Mode Policy topic](#) for more information.

Configuring User Keys

- **User Key Set 1** is required. User Key Set 1 includes FlashLock/UPK1 (User Pass Key 1) and UEK1 (User Encryption Key 1).
- **User Key Set 2** is optional. User Key Set 2 includes UPK2 (User Pass Key 2) and UEK2 (User Encryption Key 2). Note that User Pass Key 2 (UPK2) protects only User Encryption Key 2 (UEK2).
- **User PUF Encryption Key** is optional. User PUF Encryption Key includes UEK3 (User Encryption Key 3).

Note: UEK3 is only available for M2S060, M2GL060, M2S090, M2GL090, M2S150 and M2GL150 devices. See the [SmartFusion2 SoC FPGA and IGLOO2 FPGA Security Best Practices User Guide](#) for more details.

Update Policy

This dialog box enables you to specify components that can be updated in the field, and their field-update protection parameters.

Choose your protection options from the drop-down menus; click the appropriate checkbox to set your programming protection preferences.

Fabric update protection

- **Use FlashLock/UPK1 to unlock Erase/Write/Verify operations**- Select this option to require UPK1 to erase, write, or verify the Fabric.

Note: Fabric update is disabled for Auto Programming, IAP/ISP services, Programming Recovery and Auto update. FlashLock/UPK1 unlocking is only available for JTAG and SPI slave.

- **Updates allowed using UEK1 or UEK2 or UEK3; FlashLock/UPK1 is not required for updates -** Encrypted update is allowed with either UEK1 or UEK2 (if enabled).

Note: UEK3 is only available for M2S060, M2GL060, M2S090, M2GL090, M2S150 and M2GL150 devices. See the [SmartFusion2 SoC FPGA and IGLOO2 FPGA Security Best Practices User Guide](#) for more details.

eNVM update protection

- **Use FlashLock/UPK1 to unlock Write/Verify/Read operations-** Select this option to require UPK1 to write, verify or read to the eNVM.

Note: eNVM update is disabled for Auto Programming, IAP/ISP Services, Programming Recovery and Auto Update. FlashLock/UPK1 unlocking is only available for JTAG and SPI Slave programming.

- **Updates allowed using UEK1 or UEK2 or UEK3; Flashlock/UPK1 is not required for updates -** Encrypted update is allowed with either UEK1 or UEK2 (if enabled) or UEK3 (if enabled).

Note: UEK3 is only available for M2S060, M2GL060, M2S090, M2GL090, M2S150 and M2GL150 devices.

Back Level protection - When enabled, a design being loaded must be of a version higher than the Back Level version value in the programmed device.

- **Back Level Protection-** Limits the design versions that the device can update. Only programming bitstreams with Designer Version greater than the Back Level version are allowed for programming.
- **Design version** - Displays the current Design version (set in the [Configure Programming Options](#)). Back level uses the Design version value to determine which bitstreams are allowed for programming.
- **Back Level Bypass** - If selected, design is programmed irrespective of Back Level version.

Note: Back Level Bypass should be set if you allow programming recover with recovery image lower than the Back Level version selected. Alternatively, you should update the design version of the recovery image so that it is always greater than the Back Level version. (Refer to the [Configure Programming Recovery section](#) for details.)

Disable access to the following programming interfaces:

These settings protect the following programming interfaces:

- Auto Programming
- IAP/ISP services
- JTAG (use FlashLock to/UPK1 to unlock)
- SPI Slave (use FlashLock/UPK1 to unlock)

For more technical information on the Protect Programming Interface with Pass Key option see the [SmartFusion2 Programming User's Guide](#).

Note that when the Permanently write-protect option is selected for User keys and Security policies protection in SPM, the dialog box informs you of features that are no longer reprogrammable. In this case, if Use FlashLock/UPK1 to unlock option is selected for Fabric/eNVM update protection then Fabric/eNVM will be One Time Programmable.

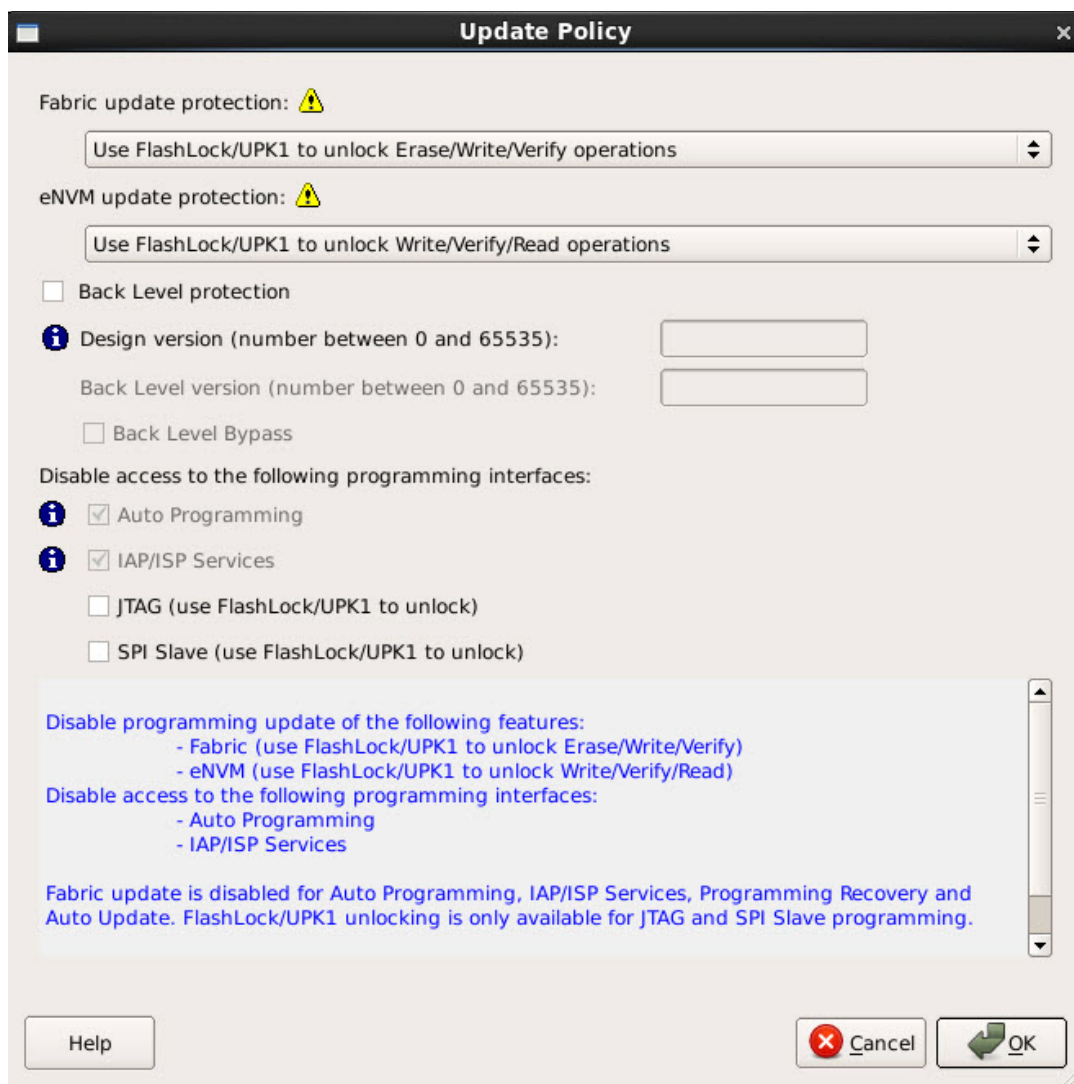


Figure 115 · Update Policy Dialog Box

Debug Security Policy

Debug access to the embedded systems can be controlled via the customer Debug Policy.

Protect Embedded Debug with DPK (Debug Pass Key)

Restrict UJTAG access - Restricts access to UJTAG; DPK is required for access.

Restrict Cortex M3 debug (SmartFusion2 Only) - Restricts Cortex M3 debug/SoftConsole use; DPK is required for debug.

SmartDebug access control

Access control available during debug mode.

Full Access (No restrictions to SmartDebug architecture; DPK is not required)- Enables full debug access to eNVM, uSRAM, LSRAM, eSRAM0/1, DDRAM and Fabric probing.

No debug (Restrict read/write access to SmartDebug architecture; DPK is required for read/write access) - Blocks all debug access to eNVM, uSRAM, LSRAM, eSRAM0/1, DDRAM and Fabric probing.

DPK (Debug Pass Key) (length is 64 HEX characters)

Specify a Debug Pass Key to unlock features protected by DPK.

Restrict external Fabric/eNVM design digest check request via JTAG and SPI. Use FlashLock/UPK1 to allow digest check. - Protects design digest check request with FlashLock/UPK1.

Disable debug access through JTAG (1149.1). Use FlashLock/UPK1 to allow access. - Disables JTAG (1149.1) test instructions. The following JTAG test instructions will be disabled: HIGHZ, EXTEST, INTEST, CLAMP, SAMPLE, and PRELOAD. I/Os will be tri-stated when in JTAG programming mode and BSR control during programming is disabled. BYPASS, IDCODE, and USERCODE instructions will remain functional.

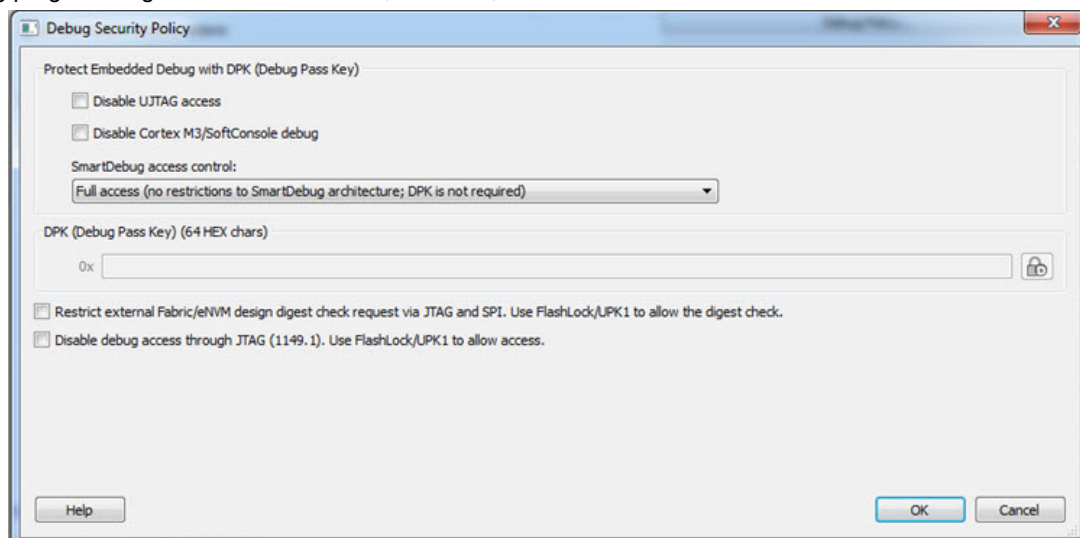


Figure 116 · Debug Security Policy Dialog Box

Key Mode Policy

Protect user encryption key modes with FlashLock/UPK1. If a key mode is disabled, then FlashLock/UPK1 is required to program with that key mode.

The following key modes can be disabled:

- UEK1 (User Encryption Key 1)
- UEK2 (User Encryption Key 2)
- UEK3 (User Encryption Key 3)

Note: UEK3 is only available for M2S060, M2GL060, M2S090, M2GL090, M2S150 and M2GL150 devices. See the [SmartFusion2 SoC FPGA and IGLOO2 FPGA Security Best Practices User Guide](#) for more details.

If all key modes are disabled then device update is impossible. A warning message is displayed in this case.

Note: If a key mode is disabled, the corresponding bitstream file will be disabled.

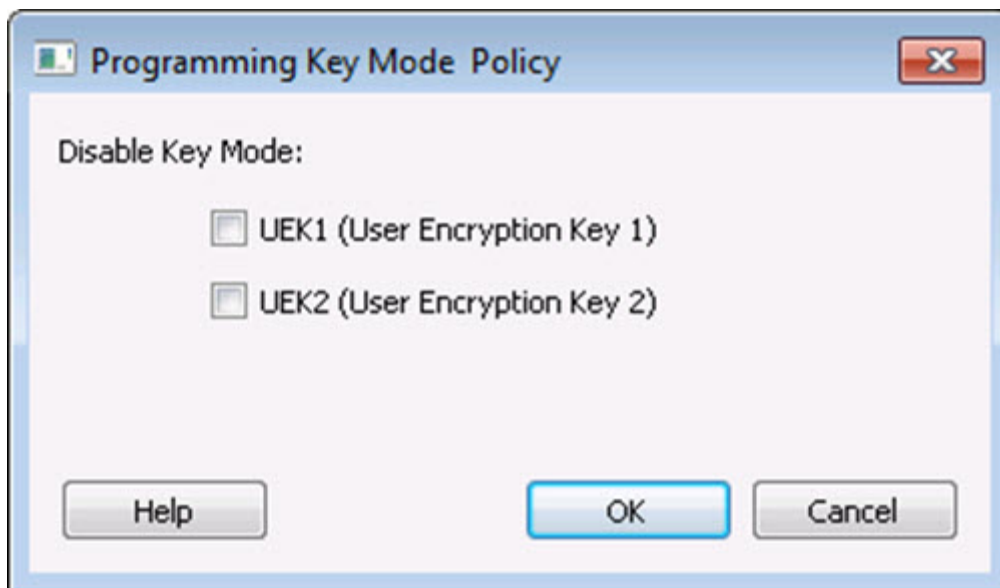


Figure 117 · Programming Key Mode Policy Dialog Box

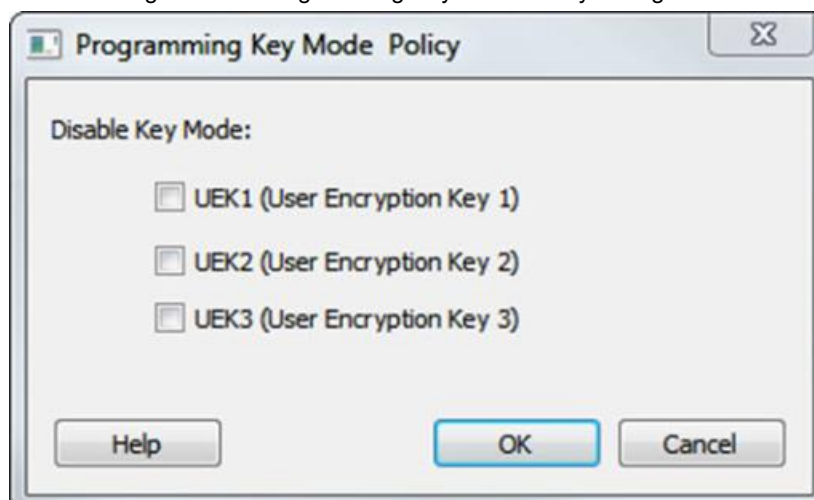


Figure 118 · Programming Key Mode Policy Dialog Box (for devices supporting UEK3)

Security Features Frequently Asked Questions

The following Frequently Asked Questions address the most common queries related to managing and programming SmartFusion2 and IGLOO2 Security Features.

I have configured the "Configure Security Policy Manager" on page 145 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 eNVM 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 1/2/3?

- Configure the "Configure Security Policy Manager" on page 145 and specify User Key Set 1, User Key Set 2, and User PUF Encryption Key. 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/svf/spi/dat), <filename>_uek2.(stp/svf/spi/dat), and <filename>_uek3.(stp/svf/spi/dat) files are encrypted with UEK1, UEK2, and UEK3, respectively. See Security Programming File Descriptions below for more information on programming files.

Note: UEK3 is only available for M2S060S, M2GL060S, M2S090S, M2GL090S, M2S150S, and M2GL150S devices. See the [SmartFusion2 SoC FPGA and IGLOO2 FPGA Security Best Practices User Guide](#) for more details.

What are Security Programming Files?

See the [Security Programming Files topic](#) for more information.

Security Programming Files

[Export Bitstream](#) (expand Handoff Design for Production in the Design Flow window) creates the following files:

<filename>_master.(stp/svf/spi/dat) - Created when Enable custom security options is specified in the "Configure Security Policy Manager" on page 145. This is the master programming file; it includes all programming features enabled, User Key Set 1, User Key Set 2 (optionally if specified), and your security policy settings.

<filename>_security_only_master.(stp /svf/spi/dat) – Created when Enable custom security options is specified in the "Configure Security Policy Manager" on page 145. Master security programming file; includes User Key Set 1, User Key Set 2 (optionally if specified), and your security policy settings.

<filename>_uek1.(stp/svf/spi/dat) – Programming file encrypted with User Encryption Key 1 used for field updates; includes all your features for programming except security.

<filename>_uek2.(stp/svf/spi/dat) – Programming file encrypted with User Encryption Key 2 used for field updates; includes all your features for programming except security.

<filename>_uek3.(stp/svf/spi/dat) – Programming file encrypted with User Encryption Key 3 used for field updates; includes all your features for programming except security.

Note: UEK3 is only available for M2S060S, M2GL060S, M2S090S, M2GL090S, M2S150S, and M2GL150S devices. See the [SmartFusion2 SoC FPGA and IGLOO2 FPGA Security Best Practices User Guide](#) for more details.

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. For example, you cannot select eNVM for programming if you do not have eNVM in your design.

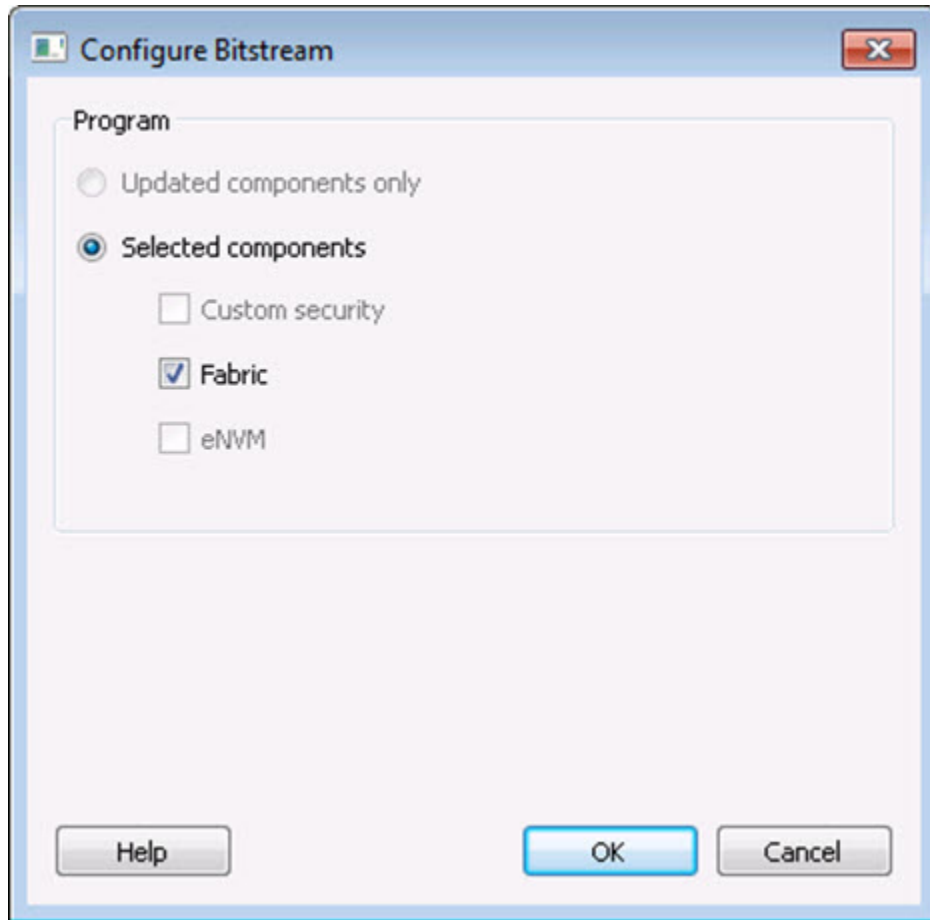


Figure 119 · Configure Bitstream Dialog Box - SmartFusion2 and IGLOO2

Selected components - Updates the components you select, regardless of whether or not they have changed since your last programming.

Note: The Custom security and eNVM components are not available for RTG4 devices.

Generate Bitstream

Generates the bitstream for use with the [Run PROGRAM Action](#) tool.

The tool incorporates the Fabric design, 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. For example, you cannot select eNVM for programming if you do not have an eNVM in your design.

Modifications to the Fabric design, eNVM configuration, or security settings will invalidate this tool and require regeneration of the bitstream file.

The Fabric programming data will only be regenerated if you make changes to the Fabric design, such as in the Create Design, Create Constraints and Implement Design sections of the Design Flow window.

When the process is complete a green check appears next to the operation in the Design Flow window (as shown in the figure below) and information messages appear in the Log window.

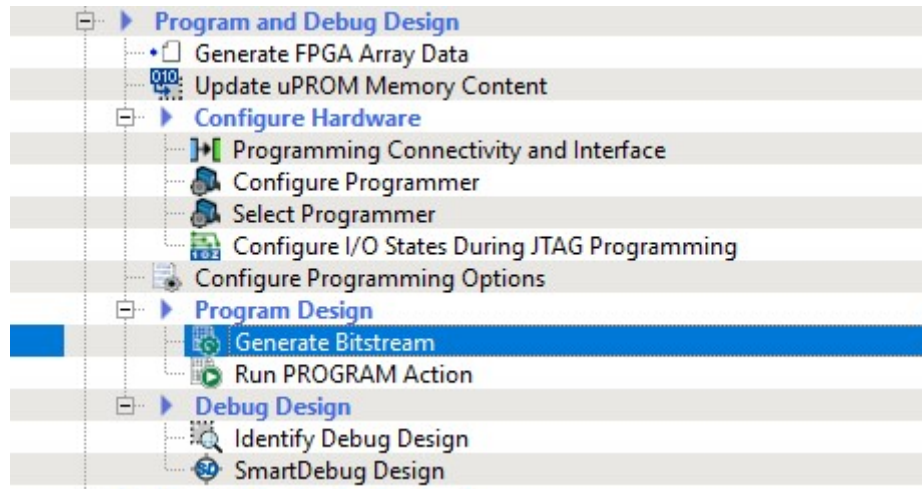


Figure 120 · Generate Bitstream (Complete)

See also[Configure Bitstream Dialog Box](#)

Run Programming Device Actions - SmartFusion2, IGLOO2, RTG4

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.

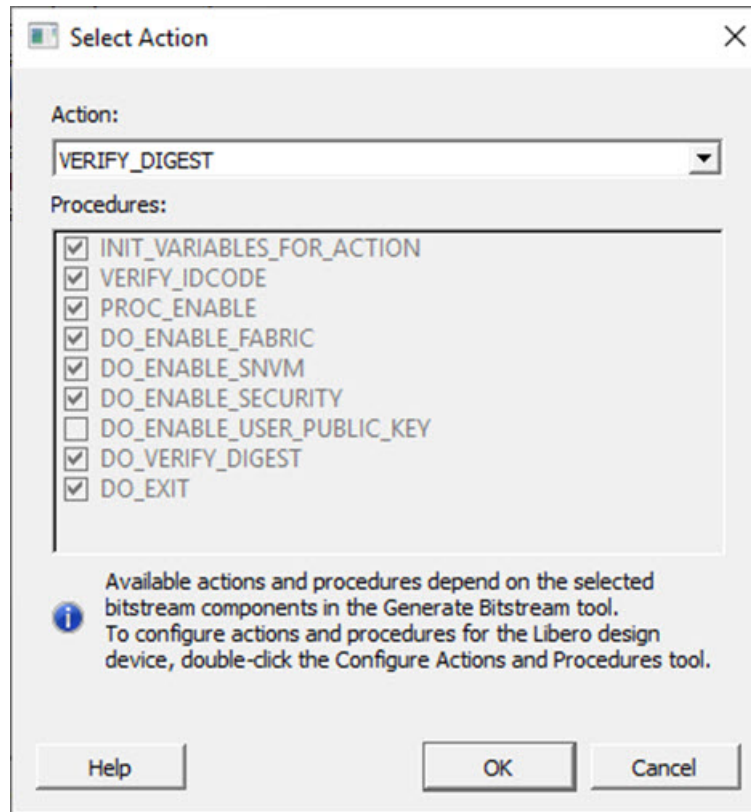
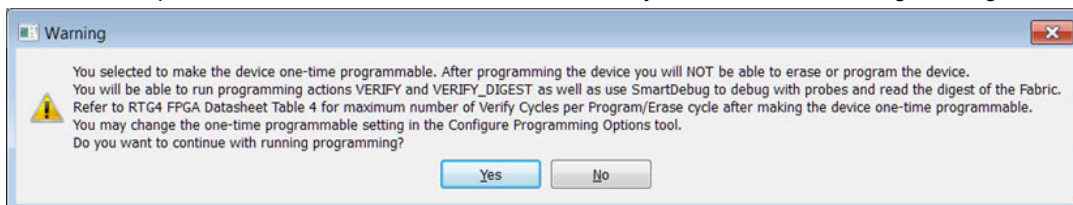


Figure 121 · Select Action Dialog Box

2. Select a programming action from the drop-down list and click **OK**.

To configure programming actions, use the [Configure Actions and Procedures](#) tool.

Note: For RTG4, if you have selected the One-time programmable (OTP) option in Configure Programming Options and the PROGRAM action is selected, you will see the following message:



Click **Yes** to continue or **No** to cancel.

See Also

[Configure Actions and Procedures](#)

Programming File Actions

Libero SoC enables you to program security settings, FPGA Array, and eNVM features.

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.

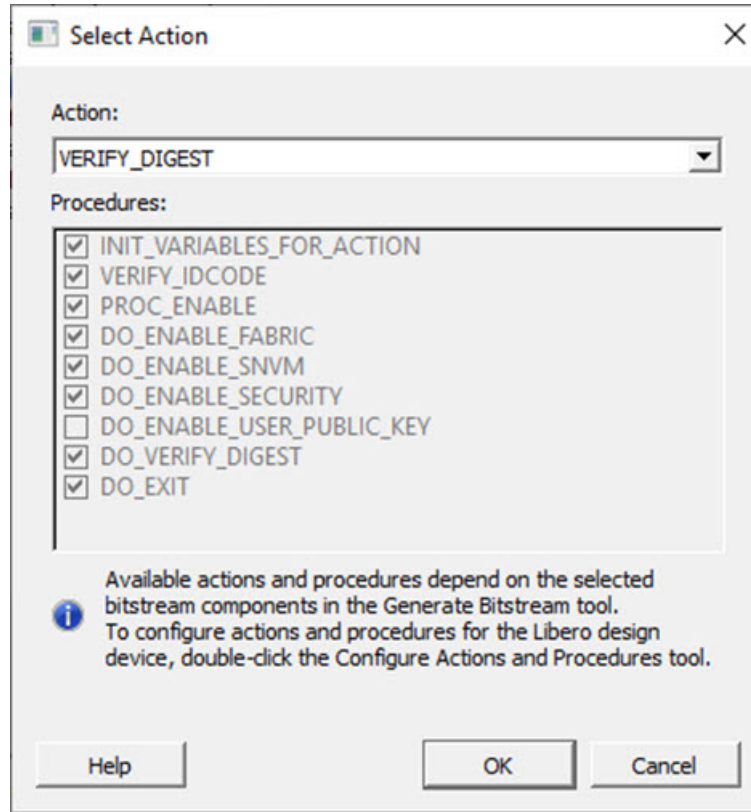


Figure 122 · Select Action Dialog Box

For details about configuring actions and procedures, see [Configure Actions and Procedures](#).

Exit Codes (SmartFusion2 and IGLOO2)

Error Code	Exit Code	Exit Message	Possible Cause	Possible Solution
	0	Passed (no error)	-	-
0x8002	5	Failure to configure device programming at 1.2/1.0 VCC voltage	Unstable voltage level Signal integrity issues on JTAG pins	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.
0x8032	5	Device is busy	Unstable VDDIx voltage level	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.
0x8003	5	Failed to enter programming mode	Unstable voltage level	Monitor related power supplies that cause the issue during programming;

Error Code	Exit Code	Exit Message	Possible Cause	Possible Solution
			Signal integrity issues on JTAG pins DEVRST_N is tied to LOW	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. Tie DEVRST_N to HIGH prior to programming the device.
0x8004	6	Failed to verify IDCODE	Incorrect programming file Incorrect device in chain Signal integrity issues on JTAG pins	Choose the correct programming file and select the correct device in the chain. Measure JTAG pins and noise for reflection. If TRST is left floating then add pull-up to pin. Reduce the length of Ground connection.
0x8005 0x8006 8x804A	10	Failed to program eNVM	Unstable voltage level. Signal integrity issues on JTAG pins.	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.
0x8027 0x8028	10	Authentication Error Bitstream and device mismatch	Libero device selection does not match the target device.	Generate a programming file with the correct device selection for the target device.
0x8007 0x804C	11	Failed to verify FPGA Array Failed to verify Fabric Configuration Failed to verify Security	Device is programmed with a different design or the component is blank. Unstable voltage level. Signal integrity issues on JTAG pins.	Verify the device is programmed with the correct data/design. Monitor related power supplies that cause the issue during programming; check for transients outside of 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.
0x8008 0x8009 0x8049	11	Failed to verify eNVM	Device is programmed with a different design. Unstable voltage level.	Verify the device is programmed with the correct data/design. Monitor related power supplies that

Error Code	Exit Code	Exit Message	Possible Cause	Possible Solution
			Signal integrity issues on JTAG pins.	<p>cause the issue during programming; check for transients outside of Microsemi specifications. See your device datasheet for more information on transient specifications.</p> <p>Monitor JTAG supply pins during programming; measure JTAG signals for noise or reflection.</p>
0x8013	-18	Digest request from SPI/JTAG is protected by User Pass Key 1	Digest request from SPI/JTAG is protected by user pass key 1. Lock bit has been configured in the Debug Policy within SPM (Security Policy Manager)	Provide a programming file with a pass key that matches pass key programmed into the device.
0x8014	-19	Failed to verify digest	<p>>Unstable voltage level</p> <p>Signal integrity issues on JTAG pins</p>	<p>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.</p> <p>Monitor JTAG supply pins during programming; measure JTAG signals for noise or reflection.</p>
0x8015	-20	FPGA Fabric digest verification: FAIL	<p>Programming bitstream components do not match components programmed</p> <p>FPGA Fabric is either erased or the data has been corrupted or tampered with</p>	Use the same programming file that was used to program the device.
0x8016	-20	eNVM_0 digest verification: FAIL	<p>Programming bitstream components do not match components programmed</p> <p>eNVM_0 data has been corrupted or tampered with</p>	Use the same programming file that was used to program the device.
0x8017	-20	eNVM_1 digest verification: FAIL	<p>Programming bitstream components do not match components programmed</p> <p>eNVM_1 data has been corrupted or tampered with</p>	Use the same programming file that was used to program the device.
0x8018	-20	User security policies segment digest verification: FAIL	<p>Programming bitstream components do not match components programmed</p> <p>User security policy segment</p>	Use the same programming file that was used to program the device.

Error Code	Exit Code	Exit Message	Possible Cause	Possible Solution
			data has been corrupted or tampered with	
0x8019	-20	User key set 1 segment digest verification: FAIL	Programming bitstream components do not match components programmed User key set 1 segment data has been corrupted or tampered with	Use the same programming file that was used to program the device.
0x801A	-20	User key set 2 segment digest verification: FAIL	Programming bitstream components do not match components programmed User key set 2 segment data has been corrupted or tampered with	Use the same programming file that was used to program the device.
0x801B	-20	Factory row and factory key segment digest verification: FAIL	Programming bitstream components do not match components programmed Factory row and factory key segment data has been corrupted or tampered with	Use the same programming file that was used to program the device.
0x801C	-20	Fabric configuration segment digest verification: FAIL	Programming bitstream components do not match components programmed. Fabric configuration segment data has been corrupted or tampered with	Use the same programming file that was used to program the device.
0x801D 0x801E 0x804B	-21	Device security prevented operation	The device is protected with user pass key 1 and the bitstream file does not contain user pass key 1. User pass key 1 in the bitstream file does not match the device.	Run DEVICE_INFO to view security features that are protected. Provide a bitstream file with a user pass key 1 that matches the user pass key 1 programmed into the device.
0x801F 0x8020 0x8040	-22	Authentication Error Bitstream or data is corrupted or noisy	eNVM has been locked by a master in your design Running VERIFY action on a blank device. Bitstream file has been corrupted Bitstream was incorrectly generated	Release the lock on the eNVM after your master has completed its access operations. Write 0x00 to "REQACCESS" register in eNVM Control Registers (address 0x600801FC) to release the access. Program the device prior to running VERIFY action Regenerate bitstream file.

Error Code	Exit Code	Exit Message	Possible Cause	Possible Solution
0x8021 0x8022	-23	Authentication Error Invalid/Corrupted encryption key	<p>File contains an encrypted key that does not match the device</p> <p>Attempting to erase a device with no security using master security file</p> <p>File contains user encryption key, but device has not been programmed with the user encryption key</p> <p>Device has user encryption key 1/2 enforced and you are attempting to reprogram security settings</p>	<p>Provide a programming file with an encryption key that matches that on the device.</p> <p>Run DEVICE_INFO action to verify that the device has no security. If the device does not have security, you cannot erase it.</p> <p>First program security with master programming file, then program with user encryption 1/2 field update programming files.</p> <p>You must first ERASE security with the master security file, then you can reprogram new security settings.</p>
0x8041	-23	Authentication Error Invalid/Corrupted encryption key	<p>File contains an encrypted key that does not match the device</p> <p>File contains user encryption key, but device has not been programmed with the user encryption key</p> <p>Attempting to erase a device with no security using master security file</p> <p>Device has user encryption key 1/2 enforced and you are attempting to reprogram security settings</p>	<p>Provide a programming file with an encryption key that matches that on the device.</p> <p>Run DEVICE_INFO action to verify that the device has no security. If the device does not have security, you cannot erase it.</p> <p>First program security with master programming file, then program with user encryption 1/2 field update programming files.</p> <p>You must first ERASE security with the master security file, then you can reprogram new security settings.</p>
0x8023 0x8024 0x8042	-24	Authentication Error Back level not satisfied	Design version is not higher than the back-level programmed device	Generate a programming file with a design version higher than the back level version.
0x8001	-24	Failure to read DSN	<p>Device is in System Controller Suspend Mode</p> <p>Check board connections</p>	TRSTB should be driven High or disable "System Controller Suspend Mode".
0x8025 0x8026 0x8043	-25	Authentication Error DSN binding mismatch	DSN specified in programming file does not match the device being programmed	Use the correct programming file with a DSN that matches the DSN of the target device being programmed.
0x8044	-26	Authentication Error Insufficient device capabilities	Device does not support the capabilities specified in programming file	Generate a programming file with the correct capabilities for the target device.

Error Code	Exit Code	Exit Message	Possible Cause	Possible Solution
0x8027 0x8028	-26	Authentication Error Bitstream and device mismatch	Libero device selection does not match the target device	Generate a programming file with the correct device selection for the target device.
0x8029 0x802A 0x8045	-27	Authentication Error Incorrect DEVICEID	Incorrect programming file Incorrect device in chain Signal integrity issues on JTAG pins	Choose the correct programming file and select the correct device in chain. Measure JTAG pins and noise or reflection. If TRST is left floating, then add pull-up to pin. Reduce the length of ground connection.
0x802B 0x802C	-28	Authentication Error Programming file is out of date, please regenerate	Programming file version is out of date	Generate programming file with latest version of Libero SoC.
0x8046	-28	>Authentication Error Unsupported bitstream protocol version	Old programming file	Generate programming file with latest version of Libero SoC.
0x802F	-30	JTAG interface is protected by UPK1	Invalid or no UPK1 is provided	User needs to provide correct UPK1 to unlock device.
0x8030 0x8031 0x8048	-31	Authentication Error Invalid or inaccessible Device Certificate	M2S090 Rev. A or M2S150 Rev. A: Either certificate is corrupted or the user hasn't provided the application code in the eNVM or provided invalid application code FAB_RESET_N is tied to ground	User can program a valid application code. This can be done with SoftConsole. FAB_RESET_N should be tied to HIGH.
0x8032 0x8033 0x8034 0x8035 0x8036 0x8037 0x8038 0x8039	-32	Instruction timed out	Unstable voltage level Signal integrity issues on JTAG pins	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.
0x8010	-35	Failed to unlock User Pass Key 1	Pass key in file does not match device.	Provide a programming file with a pass key that matches pass key programmed into the device.

Error Code	Exit Code	Exit Message	Possible Cause	Possible Solution
			Plaintext pass key match is disabled. This occurs if HSM was used to program the device.	Match pass key using HSM.
0x8011	-35	Failed to unlock User Pass Key 2	Pass key in file does not match device. Plaintext pass key match is disabled. This occurs if HSM was used to program the device.	Provide a programming file with a pass key that matches pass key programmed into the device. Match pass key using HSM.
0x8012	-35	Failed to unlock debug pass key	Pass key in file does not match device. Plaintext pass key match is disabled. This occurs if HSM was used to program the device.	Provide a programming file with a pass key that matches pass key programmed into the device. Match pass key using HSM.
0x804D	-36	<HSM related error message based on scenario>	HSM communication error. HSM call returns error.	Check if HSM the communication path to HSM is up. Make sure project is loaded properly and that HSM tickets have not been cleaned.
0x804E	-37	Device already has Security programmed. Please erase the device using master file before reprogramming Security Settings.	HSM flow does not support reprogramming device directly if Security has already been programmed.	Erase security and try programming the device.

Exit Codes (RTG4)

Error Code	Exit Message	Possible Cause	Possible Solution
	Passed (no error)	-	-
0x8001	Failure to read DSN	Device is in System Controller Suspend Mode Check board connections	TRSTB should be driven High on device power up. Disable System Controller Suspend Mode in "Programming Bitstream Settings" tool within Libero and reprogram the device.
0x8002	Device is busy	Unstable VDDIx voltage level	Monitor related power supplies that cause the issue during programming; check for transients outside of Microsemi

Error Code	Exit Message	Possible Cause	Possible Solution
			specifications. See your device datasheet for more information on transient specifications.
0x8003	Failed to enter programming mode	Unstable voltage level Signal integrity issues on JTAG pins DEVRST_N is tied to LOW	Monitor related power supplies that cause the issue during programming; check for transients outside of 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. Tie DEVRST_N to HIGH prior to programming the device
0x8004	Failed to verify IDCODE	Incorrect programming file Incorrect device in chain Signal integrity issues on JTAG pins	Choose the correct programming file and select the correct device in the chain. Measure JTAG pins and noise for reflection. If TRST is left floating then add pull-up to pin. Reduce the length of Ground connection.
0x8005	Failed to verify IDCODE RT4G150_ES STAPL file is not compatible with RT4G150 production devices. You must use a STAPL file for RT4G150 device.	Programming file is for RT4G150_ES and device is RT4G150 Incorrect programming file Incorrect device in chain Signal integrity issues on JTAG pins	Generate a programming file for RT4G150 device Choose the correct programming file and select the correct device in the chain. Measure JTAG pins and noise for reflection. If TRST is left floating then add pull-up to pin. Reduce the length of Ground connection.
0x8006	Failed to verify IDCODE RT4G150 STAPL file is not compatible with RT4G150_ES devices. You must use a STAPL file for RT4G150_ES device.	Programming file is for RT4G150 and device is RT4G150_ES Incorrect programming file Incorrect device in chain Signal integrity issues on JTAG pins	Generate a programming file for RT4G150_ES device Choose the correct programming file and select the correct device in the chain. Measure JTAG pins and noise for reflection. If TRST is left floating then add pull-up to pin. Reduce the length of Ground connection.

Error Code	Exit Message	Possible Cause	Possible Solution
0x8007	Failed to verify FPGA Array	Device is programmed with a different design or the component is blank. Unstable voltage level. Signal integrity issues on JTAG pins.	Verify the device is programmed with the correct data/design. Monitor related power supplies that cause the issue during programming; check for transients outside of 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.
0x8008	Device is blank	Attempting to verify digest of a blank device	Program the device prior to running action "VERIFY_DIGEST"
0x8009	FPGA array digest check is disabled	Digest check has been disabled by "Programming Bitstream Settings" tool within Libero	Drive TRSTB high during device power up. Enable digest check in "Programming Bitstream Settings" tool within Libero and reprogram the device.
0x800A	Failed to verify digest: Instruction timed out	Unstable voltage level Signal integrity issues on JTAG pins	Try running VERIFY_DIGEST action again. 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.
0x800B	FPGA Fabric digest verification: FAIL	Programming bitstream components do not match components programmed FPGA Fabric is either erased or the data has been corrupted or tampered with	Use the same programming file that was used to program the device.
0x800C	Factory row segment digest verification: FAIL	Programming bitstream components do not match components programmed Factory row segment data has been corrupted or tampered with	Use the same programming file that was used to program the device.
0x800D	Bitstream Error. Bitstream or	Bitstream file has been corrupted	Regenerate bitstream file

Error Code	Exit Message	Possible Cause	Possible Solution
	data is corrupted or noisy.	Bitstream was incorrectly generated Unstable voltage level Signal integrity issues on JTAG pins	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.
0x800E	Failed to query programming bitstream settings: Instruction timed out	Unstable voltage level Signal integrity issues on JTAG pins	Try running DEVICE_INFO action again. 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.
0x800F	Bitstream Error. Incorrect DEVICEID	Incorrect programming file Incorrect device in chain Signal integrity issues on JTAG pins	Choose the correct programming file and select the correct device in the chain. Measure JTAG pins and noise for reflection. If TRST is left floating then add pull-up to pin. Reduce the length of Ground connection.
0x8010	Operation has been disabled by programming bitstream settings	Operation has been disabled by "Programming Bitstream Settings" tool within Libero User disabled Fabric Erase/Write/Verify and attempted to Erase/Program/Verify the device	Drive TRSTB high during device power up Enable the disabled operation in the "Programming Bitstream Settings" tool with Libero and reprogram the device
0x8011	Failed to check bitstream: Instruction timed out	Unstable voltage level Signal integrity issues on JTAG pins	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.

Error Code	Exit Message	Possible Cause	Possible Solution
0x8012, 0x8013	Failed to erase device: Instruction timed out	Unstable voltage level Signal integrity issues on JTAG pins	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.
0x8014	Failed to program device: Instruction timed out	Unstable voltage level Signal integrity issues on JTAG pins	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.
0x8015	Error, device is not ready.	DEVRST_N may have been driven LOW during programming	Need to ensure that DEVRST_N is driven HIGH during programming. The reliability of the device in space cannot be guaranteed if this has occurred. It is the user's responsibility to ensure that DEVRST_N is driven HIGH during programming.

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.

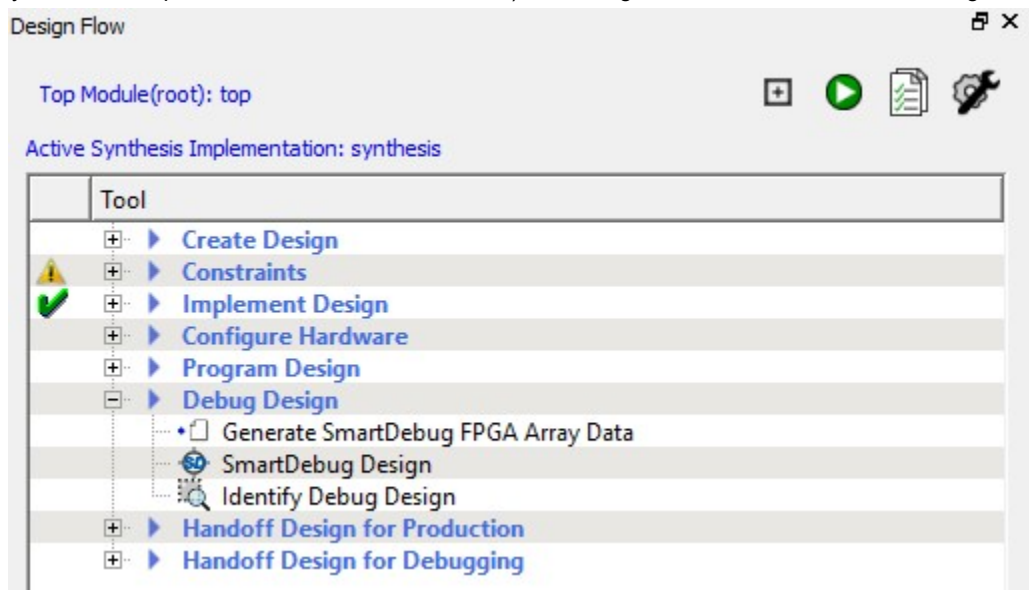


Figure 123 · Generate SmartDebug FPGA Array Data

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 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 EDN Netlist file.
7. From the Instrumentor UI specify the sample clock, the breakpoints, and other signals to probe. Synplify creates a new synthesis implementation. Synthesize the design.
8. In Libero SoC, run Synthesis, Place and Route and Generate a Programming File.
Note: Libero SoC works from the edif netlist of the current active implementation, which is the implementation you created in Synplify for Identify debug.
9. Double-click **Identify Debug Design** in the Design Flow window to launch the Identify Debugger.

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 for SmartFusion2 and IGLOO2:

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:
 - [Bitstream Encryption with the Default Key in the Security Policy Manager](#)
 - [Enable Custom Security Options in the Security Policy Manager](#)
2. 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** or **SVF** if you need support for ISP.
3. You can choose whether you want to export the ASCII HEX file for debugging if DAT bitstream file format is selected.
4. Select whether you want to export files for Microsemi In House Programming (IHP).
5. Set your **eNVM Serialization options**, if any. eNVM Serialization must be enabled via the eNVM Configurator in your MSS.
6. Select to include the bitstream components and Include Pass Keys in Plaintext.
7. Enter your **Bitstream file name** and location to export the selected bitstream files.
8. You can export the SPI directory for programming recovery if needed.
9. Click **OK** to export the selected bitstream files.

To export a bitstream file for RTG4, see the following topic:

- [Export Bitstream - RTG4](#)

See Also

[Digest File](#)

Export Bitstream tool when the device is configured with Bitstream Encryption with Default Key in the Security Policy Manager

See the [Export Bitstream](#) topic for more information about exporting your bitstream.

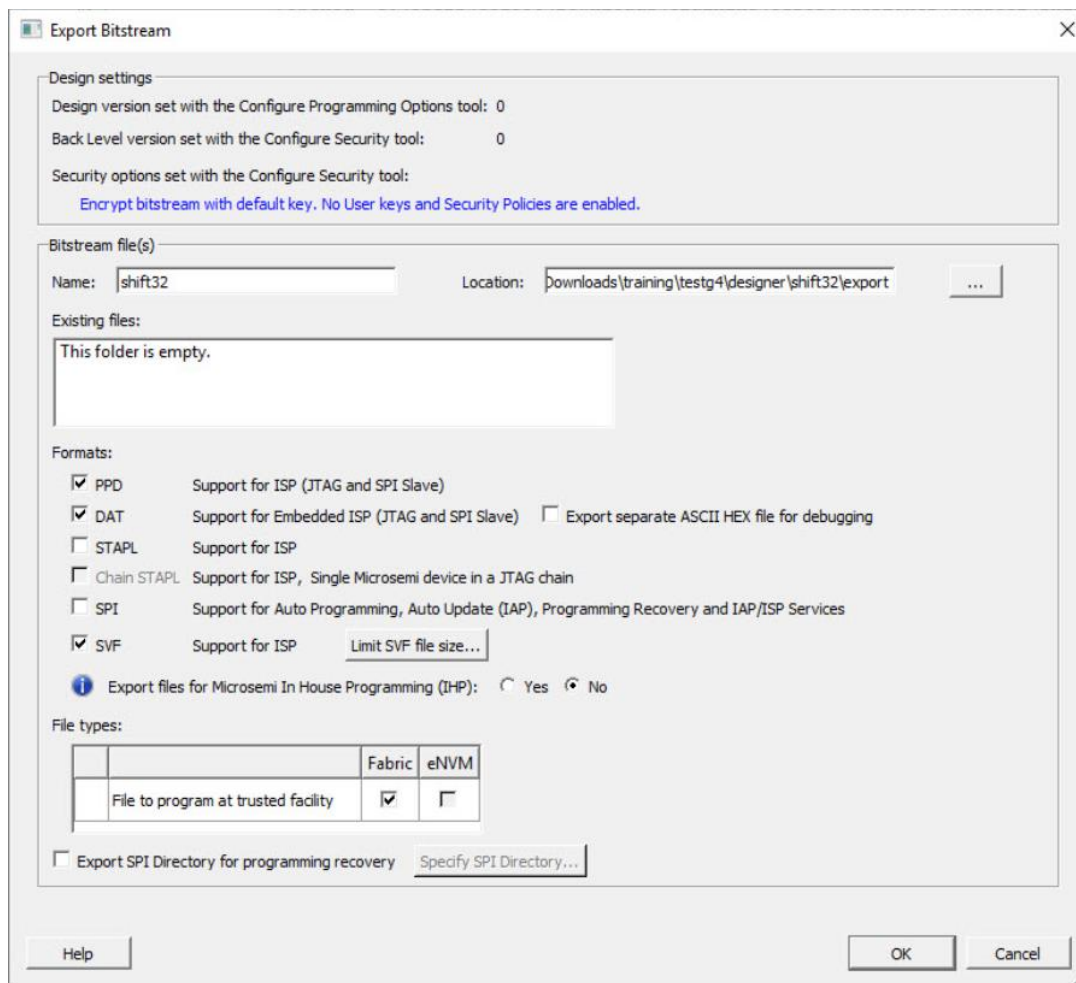


Figure 124 · Export Bitstream with Default Key Dialog Box

Design Settings:

- Displays the Design version set with the Configure Programming Options tool and Back Level version set with the Configure Security 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
 - Chain STAPL (Enabled only when there are two or more devices in the chain)
 - 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. STAPL and DAT are the required file formats for In House Programming

- **File types:** Lists all the bitstream files to be exported
 - **File to program at trusted facility** – Click to include Fabric and/or eNVM into the bitstream files to be programmed at a trusted facility.

Note: Only features that have been added to your design are available for programming. For example, you cannot select eNVM for programming if you do not have eNVM in your design.

- **Export SPI Directory for programming recovery** – Allows you to export SPI directory containing Golden and Update SPI image addresses and design versions, used in Auto-update and Programming Recovery flow. Check this option and click **Specify SPI Directory** to set the required information (see figure below).

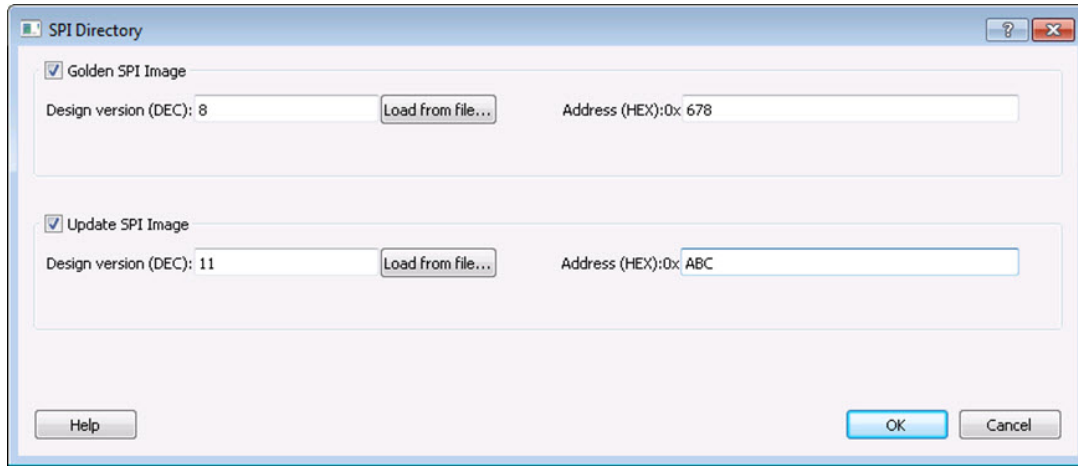


Figure 125 · SPI Directory Dialog Box

Export Bitstream tool when the device is configured with Custom Security option in the Security Policy Manager - SmartFusion2 and IGLOO2

See the [Export Bitstream](#) topic for information on exporting your bitstream.

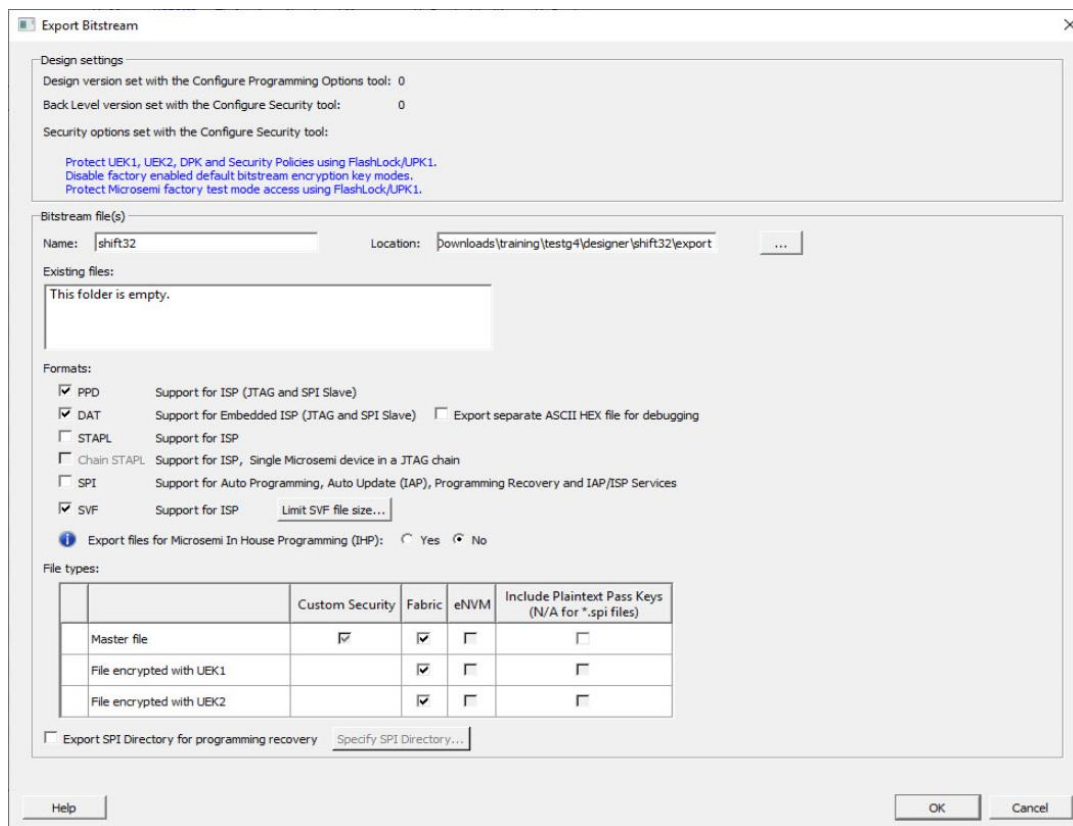


Figure 126 · Export Bitstream Dialog Box with Enable Custom Security Options in the Security Policy Manager

Design Settings:

- Displays the Design version set with the Configure Programming Options tool and Back Level version set with the Configure Security 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
 - Chain STAPL (Enabled only when there are two or more devices in the chain)
 - 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. STAPL and DAT are the required file formats for In House Programming

- 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.

Lists all the bitstream files to be exported

- **Master file** – Click to include Fabric and /or eNVM and/or Include Plaintext Pass Keys into the bitstream files to be programmed at a trusted environment.

Notes

1. Custom Security is always programmed in the Master file.
 - **File encrypted with UEK1 to program at untrusted facility or for Broadcast field update** - Click to include Fabric and /or eNVM and/or Include Plaintext Pass Keys into the bitstream files to be programmed. If the selected features are not protected by UPK1, the bitstream can be programmed at an untrusted location, since it is encrypted with UEK1 that is preprogrammed into the device.
 - **File encrypted with UEK2 to program at untrusted facility or for Broadcast field update** - Click to include Fabric and /or eNVM and/or Include Plaintext Pass Keys into the bitstream files to be programmed. If the selected features are not protected by UPK1, the bitstream can be programmed at an untrusted location, since it is encrypted with UEK2 that is preprogrammed into the device.
 - **File encrypted with UEK3 to program at untrusted facility or for Broadcast field update** - Click to include Fabric and /or eNVM and/or Include Plaintext Pass Keys into the bitstream files to be programmed. If the selected features are not protected by UPK1, the bitstream can be programmed at an untrusted location, since it is encrypted with UEK3 that is preprogrammed into the device.

Note: If eNVM/Fabric is One Time Programmable, it is precluded from bitstream encrypted with UEK1/2/3.

Note: UEK3 is only available for M2S060S, M2GL060S, M2S090S, M2GL090S, M2S150S, and M2GL150S devices. See the [SmartFusion2 SoC FPGA and IGLOO2 FPGA Security Best Practices User Guide](#) for more details.

Note: If a component (for example, eNVM) is not present in the design, it will be disabled in the bitstream component selection.

Export SPI Directory for programming recovery – Allows you to export the SPI directory containing Golden and Update SPI image addresses and design versions, used in Auto-update and Programming Recovery flow. Check this option and click **Specify SPI Directory** to set the required information (see figure below).

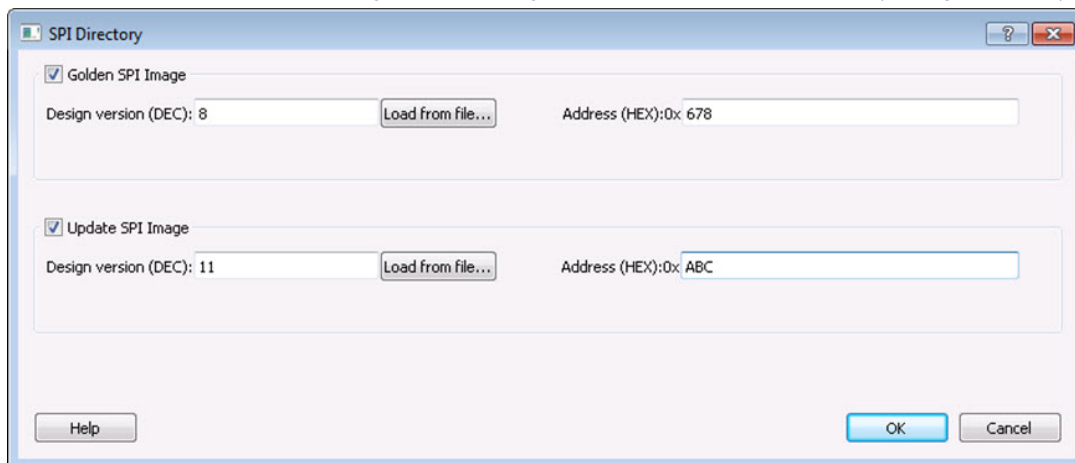


Figure 127 · SPI Directory Dialog Box

Security Programming Files

[Export Bitstream](#) (expand Handoff Design for Production in the Design Flow window) creates the following files:

<filename>_master.(stp/svf/spi/dat) - Created when Enable custom security options is specified in the "Configure Security Policy Manager" on page 145 . This is the master programming file; it includes all programming features enabled, User Key Set 1, User Key Set 2 (optionally if specified), and your security policy settings.

<filename>_security_only_master.(stp /svf/spi/dat) – Created when Enable custom security options is specified in the "Configure Security Policy Manager" on page 145. Master security programming file; includes User Key Set 1, User Key Set 2 (optionally if specified), and your security policy settings.

<filename>_uek1.(stp/svf/spi/dat) – Programming file encrypted with User Encryption Key 1 used for field updates; includes all your features for programming except security .

<filename>_uek2.(stp/svf/spi/dat) – Programming file encrypted with User Encryption Key 2 used for field updates; includes all your features for programming except security.

<filename>_uek3.(stp/svf/spi/dat) – Programming file encrypted with User Encryption Key 3 used for field updates; includes all your features for programming except security.

Note: UEK3 is only available for M2S060S, M2GL060S, M2S090S, M2GL090S, M2S150S, and M2GL150S devices. See the [SmartFusion2 SoC FPGA and IGLOO2 FPGA Security Best Practices User Guide](#) for more details.

Export Bitstream - RTG4

Design Settings:

- Displays the Design version set with the Configure Programming Options tool. It is read-only option and cannot be modified.

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
 - Chain STAPL (Enabled only when there are two or more devices in the chain)
 - 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. STAPL and DAT are the required file formats for In House Programming

Note: Security Programming is not supported for RTG4.

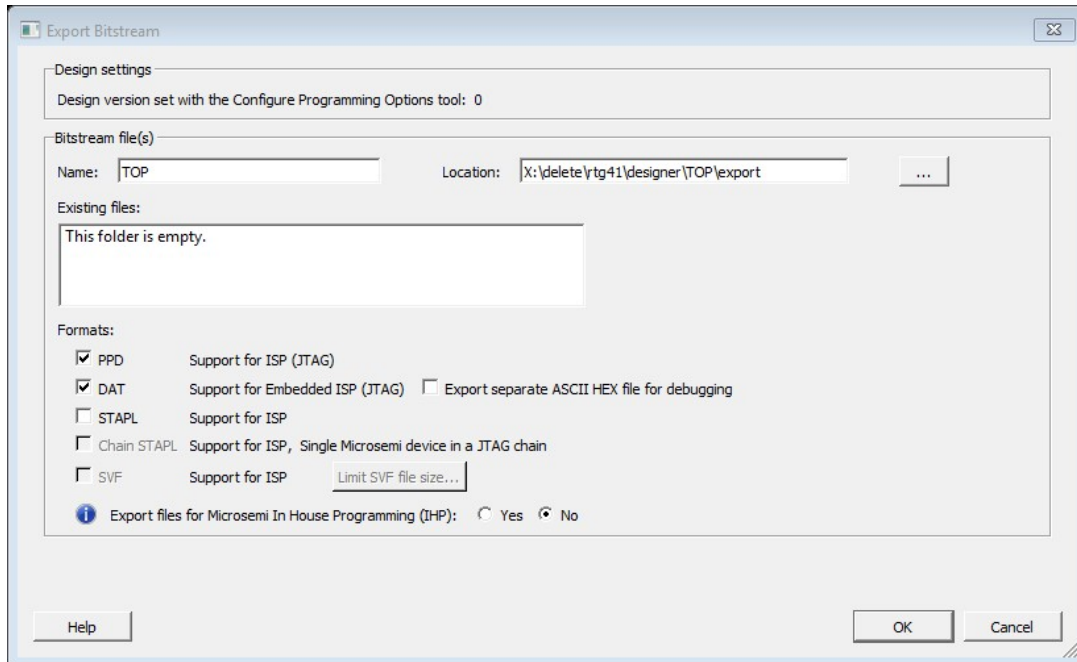
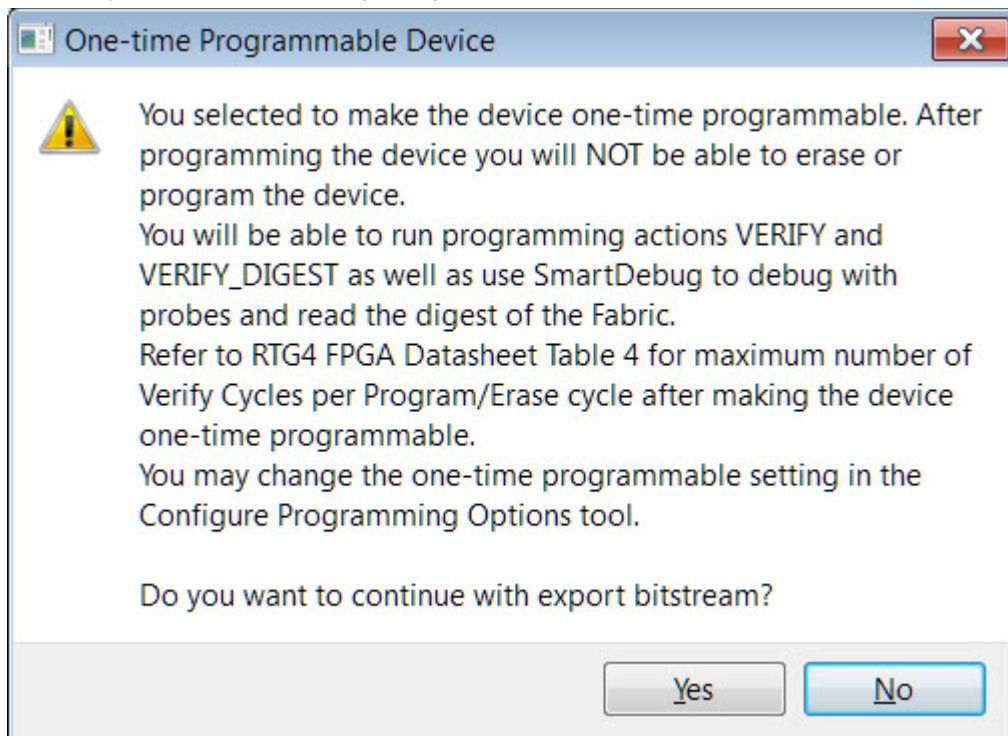


Figure 128 · Export Bitstream Dialog Box

Note: If you have selected the One-time programmable (OTP) option in Configure Programming Options, you will see the following dialog box:



Click **Yes** to continue or **No** to cancel.

Export FlashPro Express Job - SmartFusion2, IGLOO2, RTG4

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, Programming Mode (JTAG/SPI-Slave) and programming files loaded from Programming Connectivity and Interface.

Note: SPI Slave mode is supported by FlashPro5 for SmartFusion2 and IGLOO2 devices, and by FlashPro6 for SmartFusion2, IGLOO2 devices. SPI Slave mode is not supported for RTG4 devices. JTAG is the default interface.

For SmartFusion2 and IGLOO2, Security Programming is supported. Use the Security Policy Manager to configure Security before you export the programming job. The Export FlashPro Express Job dialog box for SmartFusion2 and IGLOO2 displays the Security Options you have configured in the Security Policy Manager.

Note: Security Programming is not supported for RTG4.

For RTG4, Security Programming is not supported. No security programming options are available in the Export FlashPro Express Job dialog box.

The Export FlashPro Express Job dialog box options vary depending on the device you are using and the Security Key Mode you select.

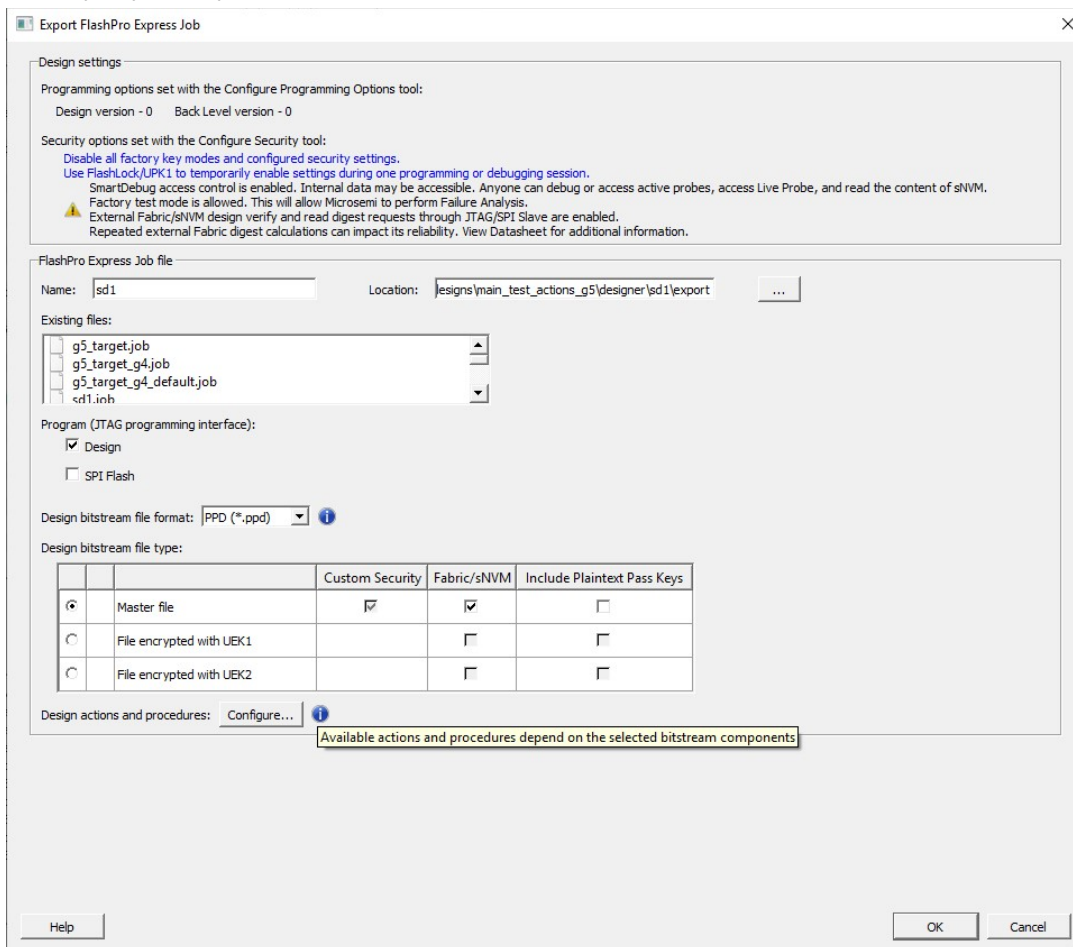


Figure 129 · Export FlashPro Express Job Dialog Box with Bitstream encryption with default key

Design Settings:

- Displays the Design version set with the Configure Programming Options tool and Back Level version set with the Configure Security 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 (Modify via Security Policy Manager)-** 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.
- **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 Security Policy Manager)**- Click to enable programming for Fabric/sNVM 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 Security Policy Manager)** – Click to include:
 - Fabric and /or eNVM.
 - Include Plaintext Pass Keys into the bitstream files to be programmed at a trusted facility.

Note:

- Custom Security is always programmed in the Master file.
- **File encrypted with UEK1 to program at untrusted facility or for Broadcast field update** – Click to include:
 - Fabric and /or 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:
 - Include Plaintext Pass Keys into the bitstream files to be programmed.
 - Fabric and /or eNVM.

Note: Refer to Export Pass Keys for Plaintext to check the availability of Export Pass keys for Plaintext option for the following Bitstream files.

- **File encrypted with UEK3 to program at untrusted facility or for Broadcast field update** - Click to include:
 - Fabric and/or eNVM.
 - Include Plaintext Pass Keys into the bitstream files to be programmed.

Note: If eNVM/Fabric is One Time Programmable, it is precluded from a bitstream encrypted with UEK1/2/3.

Note:UEK3 is only available for M2S060, M2GL060, M2S090, M2GL090, M2S150 and M2GL150 devices. See the [SmartFusion2 SoC FPGA and IGLOO2 FPGA Security Best Practices User Guide](#) for more details.

RTG4

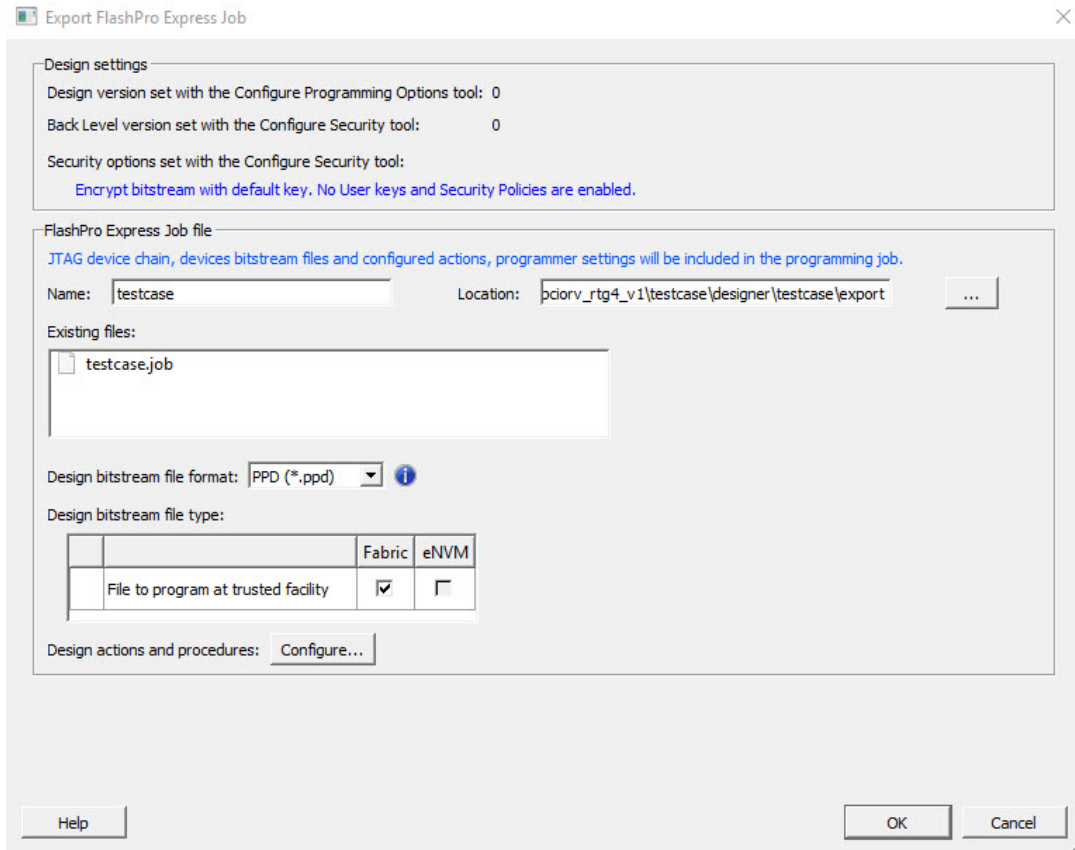


Figure 130 · Export FlashPro Express Job Dialog Box- RTG4

Design Settings:

- Displays the Design version set with the Configure Programming Options tool. It is read-only options and cannot be modified.

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.
- **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.

Note: If you have selected the One-time programmable (OTP) option in Configure Programming Options, you will see the following dialog box:



Click **Yes** to continue or **No** to cancel.

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 - SmartFusion2, IGLOO2

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.

Job Manager Data export is only provided for SmartFusion2, IGLOO2 devices.

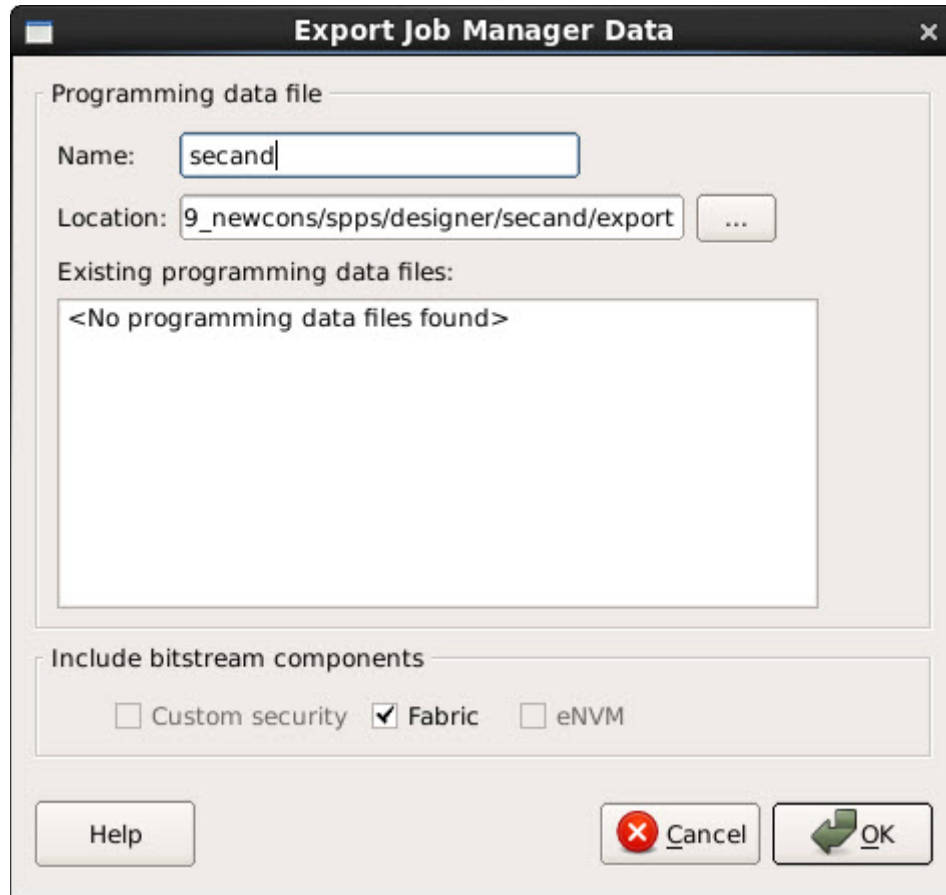


Figure 131 · Export Job Manager Data Dialog Box

Programming data file – export Job Data Container (JDC) file.

Name - All names use a prefix as shown in your software.

Location - Location of the file to be exported.

Existing programming job files - Lists any existing programming job files already in your project.

Include bitstream components - Lists the components of the design that can be saved to the file.

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.

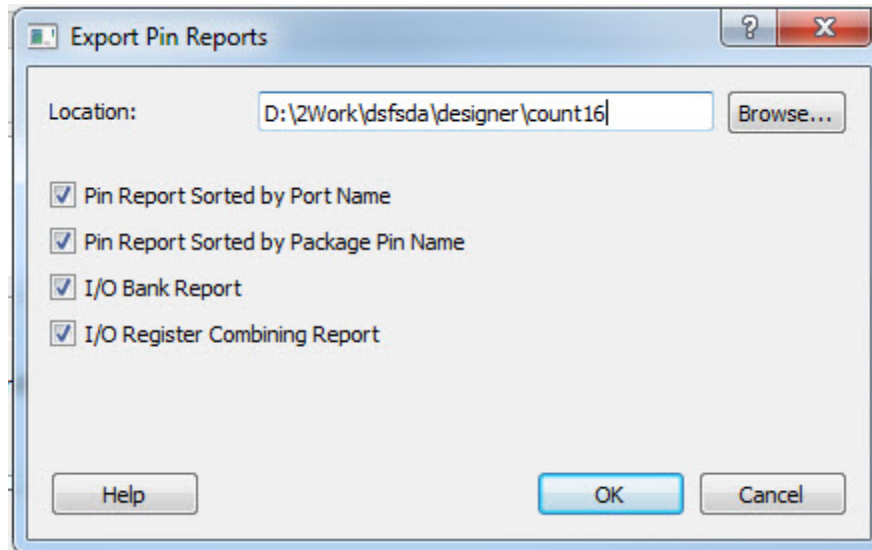


Figure 132 · Export Pin Report Dialog Box

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](#).

Export IBIS Model

Double-click Export IBIS Model (in the Libero SoC Design Flow window, **Handoff Design for Production > Export IBIS Model**) to generate the IBIS Model report.

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.

For SmartFusion2, IGLOO2 and RTG4 devices, the IBIS report *.ibs file exported from Libero SoC supports the Model Selector keyword as specified in the [IBIS 5.0 Specifications](#).

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].


[Pin]	signal_name	model_name	R_pin	I_pin	C_pin
AE33	PAD_O<27>	LVCN0515_Bank15Group1_asi0d	0.75142	5.68346e-009	2.14353e-012
E31	PAD_I<105>	LVCN0525_Bank1Group1_ddrio	0.609967	5.61206e-009	2.47835e-012
A2	PAD_O<33>	LVCN0525_Bank2Group1_ddrio	0.786904	7.52853e-009	2.87418e-012

[Model Selector]
 LVCN0515_6mA_MSIO0
 LVCN0515_2mA_MSIO0
 LVCN0515_4mA_MSIO0




Model Selector for Pin AE33

[Model Selector]
 LVCN0525_2mA_MEDFAST_DDRIO
 LVCN0525_2mA_SLOW_DDRIO
 LVCN0525_2mA_MED_DDRIO
 LVCN0525_2mA_FAST_DDRIO
 LVCN0525_4mA_MEDFAST_DDRIO
 LVCN0525_4mA_SLOW_DDRIO
 LVCN0525_4mA_MED_DDRIO
 LVCN0525_4mA_FAST_DDRIO



Model Selector for Pin E31

[Model Selector]
 LVCN0525_4mA_SLOW_DDRIO
 LVCN0525_4mA_MED_DDRIO
 LVCN0525_4mA_MEDFAST_DDRIO
 LVCN0525_4mA_FAST_DDRIO
 LVCN0525_2mA_SLOW_DDRIO
 LVCN0525_2mA_MED_DDRIO
 LVCN0525_2mA_MEDFAST_DDRIO
 LVCN0525_2mA_FAST_DDRIO



Model Selector for Pin A2

Figure 133 · Model Selector *.ibs File

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 uPROM Report - RTG4

In the Design Flow window:

1. **Expand Handoff Design for Production.**
2. Right-click **Export uPROM Report** to export a uPROM Report.
The **Export uPROM Report Under File** dialog box opens.
3. Navigate to a disk location where you want to save the uPROM Report.

Enter the following command to invoke tool in batch mode:

```
export_uPROM_Report_RTG4 -export_dir {E:\ram_uic_test_capture\designer\sd}
```

The Export uPROM Report generates an XML/TEXT version of the report. The filename is uPROM_Configuration_Report.xml/ uPROM_Configuration_Report.txt.

This report includes the configuration file path, clients details, and uPROM usage statistics.

Handoff Design for Firmware Development

Software IDE Integration

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

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

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

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

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

See Also

[Exporting Firmware and the Software IDE Workspace](#)

[Running Libero SoC from your Software Tool Chain](#)

[View/Configure Firmware Cores](#)

View/Configure Firmware Cores

Use this dialog to select and configure firmware cores (drivers) for your Software IDE project. The Design Firmware tab lists the compatible firmware for the hardware that you have instantiated in your design. In the Design Flow tab, expand **Create Design** and double-click **View/Configure Firmware Cores** to view the DESIGN_FIRMWARE tab.

The Firmware table lists the compatible firmware and drivers based on the hardware peripherals that you have used in your design. Each row represents a firmware core that is compatible with a hardware peripheral in your design. The columns in the Firmware table are:

- **Generate** - Allows you to choose whether you want the files for this firmware core to be generated on disk and added to your Software IDE project. Click the checkbox to generate firmware for each peripheral in your design.
- **Instance Name** - This is the name of the firmware instance. This may be helpful in distinguishing firmware cores when you have multiple firmware cores with the same Vendor:Library:Name:Version (VLNV) in your design.
- **Core Type** - Firmware Core Type is the Name from the VLN id of the core. This generally corresponds to the name of the hardware peripheral with which the firmware core is compatible.
- **Version** - Firmware Core Version; you can upgrade or choose a different version via a dropdown menu in this column.
- **Compatible Hardware Instance** - The hardware instance in your design that is compatible with this firmware core.

Downloading Firmware

Libero attempts to find compatible firmware located in the IP Vault located on your disk, as well as firmware in the IP Repository via the Internet.

If compatible firmware is found in the IP repository but not on your disk, the row will be italicized, indicating that it needs to be downloaded. To download all firmware cores necessary for your project peripherals, click the **Download All Firmware** icon in the vertical toolbar.

Configuring Firmware

Firmware cores that have configurable options will have a wrench icon in the row. Click the wrench icon to configure the firmware core.

It is important that you check the configuration of your firmware cores if they have configurable options. They may have options that target your software IDE (Keil, IAR or Softconsole), or your processor, that are vital configuration options to getting your system to work properly.

Generating Firmware

Click the Generate icon to export the firmware drivers and software IDE project for your project. The firmware drivers are generated into <project>\firmware and the software workspace is exported to <project>\<toolchain>. <toolchain> could be SoftConsole, IAR or Keil, depending on your software IDE.

The firmware drivers are also copied into the <toolchain> folder.

Changing Firmware Core Versions

You can manually change to the latest version by selecting the drop down in the Version column.

There will often be multiple versions of a firmware cores available for a particular peripheral. The MSS Configurator selects the latest compatible version for a new design.

However, once the firmware has been added to your design, Libero will not automatically change to the latest version if one becomes available.

Note: If a core version is shown in italics it is available in the Web Repository but not in your Vault; you must download the firmware core version to use it in your design.

Generating Sample Projects

Firmware cores are packaged with sample projects that demonstrate their usage. They are packaged for specific tool chains, such as Keil, IAR and SoftConsole

To generate a sample project, right-click the firmware core and choose **Generate Sample Project**, then select your IDE tool chain (such as Keil), and choose from the list of available samples.

You will be prompted to select the destination folder for the sample project.

Once this project is generated you can use it as a starting point in your Software IDE tool or use the example project as a reference on how to use the firmware driver.

Fabric Peripherals

Libero SoC also attempts to find compatible firmware for soft (fabric) peripherals that you have added in your top-level SmartDesign if that top-level is Set as Root.

To set your top-level design as a root, right-click your top-level design in the Design Hierarchy and choose **Set as Root**. The root component appears in bold.

The figure below shows CoreGPIO, CorePWM and CoreUARTapb soft cores that have been added into your top-level SmartDesign.

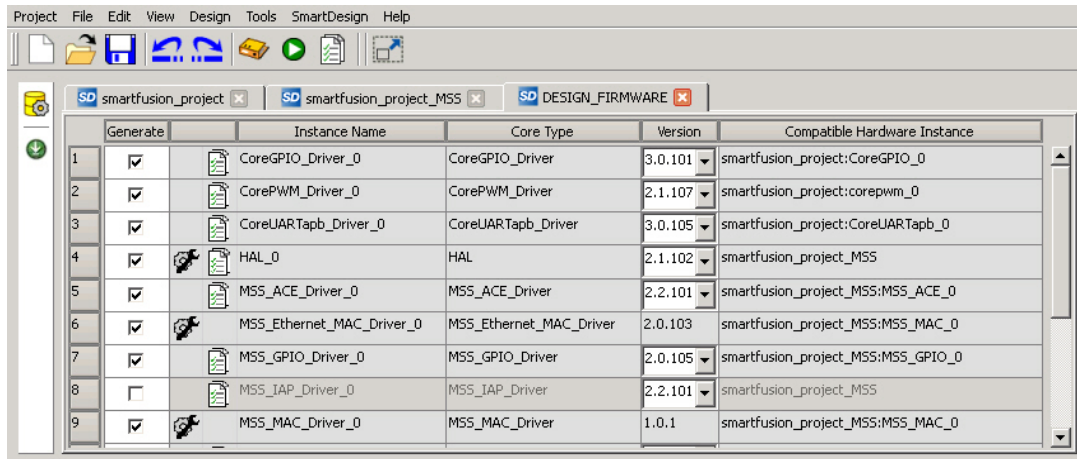


Figure 134 · Firmware Cores Tab (DESIGN_FIRMWARE)

See Also
[Exporting Firmware and the Software IDE Workspace](#)
[Running Libero SoC from your Software Tool Chain](#)
[Software IDE Integration](#)

Export Firmware – SmartFusion2

When your design has been completed, you can export the design firmware configuration using the Export Firmware tool. The firmware configuration contains:

- Register configuration files for MSS, FDDR, and SERDES blocks instantiated in your design. This information must be compiled with your application along with the SmartFusion2 CMSIS firmware core to have proper Peripheral Initialization when the Cortex-M3 boots.
- Firmware drivers compatible with the hard and soft peripherals instantiated in your design.

To export your design firmware configuration, double-click **Export Firmware** in the Libero SoC Design Flow window under Handoff Design for Firmware Development. The Export Firmware dialog box opens.

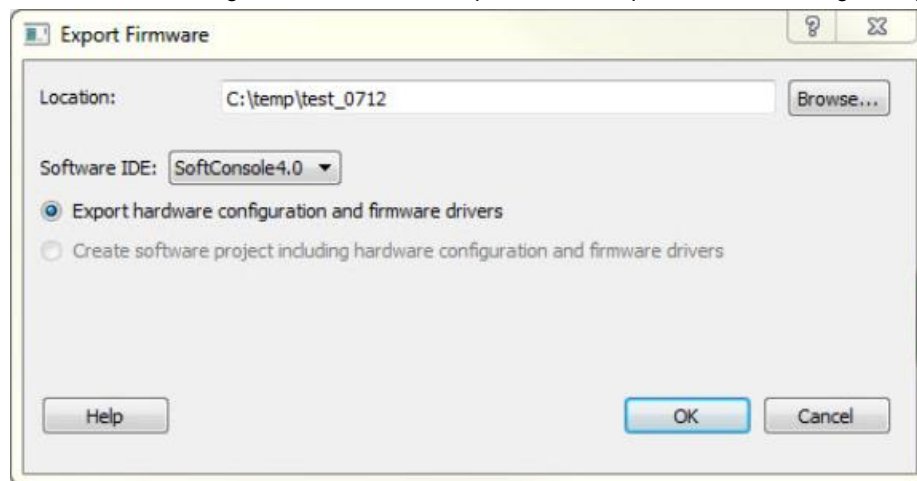


Figure 135 · Export Firmware Dialog Box

Location: Provide the location where you want the firmware configuration files to be exported. When you export the firmware, Libero SoC creates a Firmware folder to store all the drivers and register configuration files.

Software IDE: <selected Software Tool Chain>

Libero SoC creates the firmware project for the IDE tool of your choice and creates the SoftConsole/IAR/Keil (per your choice) folder to store the projects.

Export hardware configuration and firmware drivers: This option is checked by default. Beginning in Libero SoC v11.7, the Export hardware configuration option exports register configuration files for MSS, FDDR and SERDES blocks instantiated in your design. CMSIS and other firmware drivers must be generated using the standalone Firmware Catalog executable. These options are available to support SoftConsole 4.0 flow.

Create software project including hardware configuration and firmware drivers

To enable you to manage your firmware project separately from Libero's automatically generated firmware data, the created software workspace contains two software projects:

hardware_platform - This project contains all the firmware and hardware abstraction layers that correspond to your hardware design. This project is configured as a library and is referenced by your application project. The content of this folder is overwritten every time you export your firmware project.

application - This project produces a program and results in the binary file. It links with the hardware_platform project. This folder does not get overwritten when you re-export your firmware. This is where you can write your own main.c and other application code, as well as add other user drivers and files. You can reference header (*.h) files of any hardware peripherals in the hardware_platform project – include paths are automatically set up for you.

To build your workspace, make sure you have both the hardware_platform and _application projects set to the same compile target (Release or Debug) and build both projects.

To open your exported firmware projects you must invoke your third-party development tool (SoftConsole, Keil or IAR) outside Libero SoC and point it to the exported firmware workspace.

Note: You must re-export firmware if you make any changes to your design

TCL Command

```
export_firmware \  
-export_dir {D:\Designs\software_drivers} \  
-create_project 1 \  
-software_ide {Keil}
```

Version Supported

Libero SoC v11.7 and later supports the following versions of third-party development tools:

- SoftConsole v4.0
- SoftConsole v3.4
- IAR EWARM
- Keil

Export SmartDebug Data (Libero SoC)

Export SmartDebug Data allows the export of SmartDebug Data from Libero to be handed off to the Standalone SmartDebug environment.

In the Libero SoC Design Flow window, expand **Handoff Design for Debugging**, right-click **Export SmartDebug Data** and click **Export** to open the Export SmartDebug Data dialog box. Specify the design debug data file (*.ddc) to be exported. This file is also used as one of the ways to create a Standalone SmartDebug project.

See the following figure for an example.

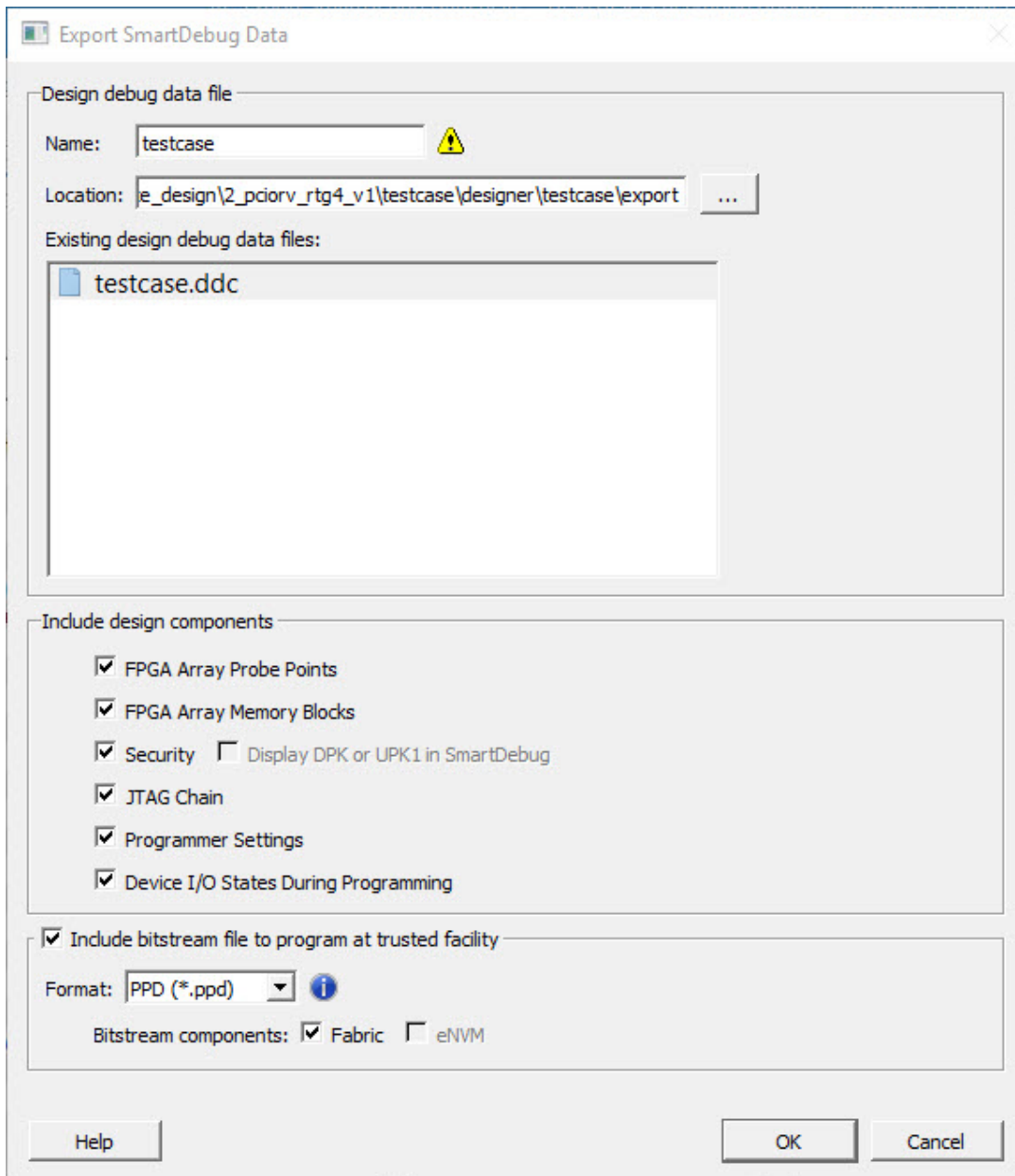


Figure 136 · Export SmartDebug Data Dialog Box

Note: SmartDebug data can be exported without connecting the hardware.

Design debug data file (*.ddc)

Name

The name of the design.

Location

The location of the exported debug file. By default, the *.ddc file is exported to the <project_location>/designer/<design>/export folder and has the *.ddc file extension.

Existing Design Debug Data Files

The existing *.ddc file, if any, in the export folder.

SmartDebug data can be exported after you run Generate FPGA Array Data for the design in the Libero Design Flow. You can also directly export SmartDebug data after running Synthesize on the design. Other tools, such as Place and Route, Generate FPGA Array Data, and so forth) are implicitly run before the Export SmartDebug Data dialog box is displayed.

Include design components

A DDC file can contain the following components:

- **FPGA Array Probe Points** – When checked, Libero SoC exports Live and Active probes information (<design>_probe.db file) into the *.ddb container file.
- **FPGA Array Memory Blocks** – When checked, Libero SoC exports information about FPGA memories (<design>_sii_block.db) into the *.ddb container file:
 - o names and addresses of the memory blocks instantiated by the design
 - o 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; for example, RTG4 devices.
 - **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 eNVM. 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.

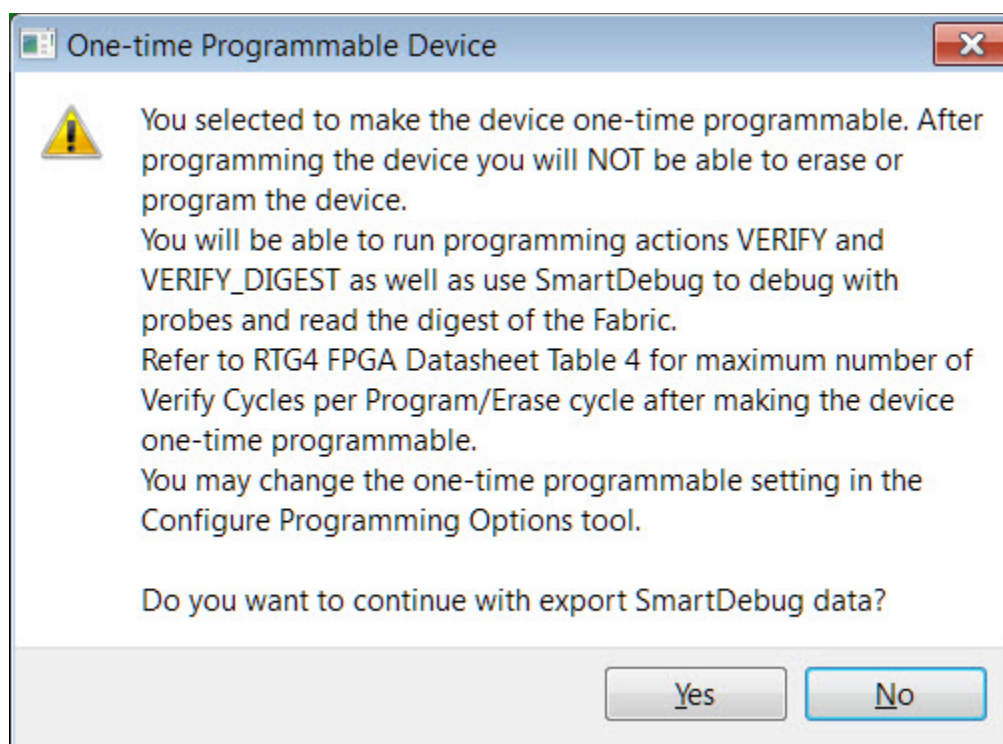
Include Bitstream file to program at trusted facility

- Bitstream components: Fabric (SmartFusion2, IGLOO2, and RTG4 devices)
- Bitstream components: eNVM (SmartFusion2 and IGLOO2 devices only)

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.

Note: If you have selected the One-time programmable (OTP) option in Configure Programming Options, you will see the following dialog:



Click **Yes** to continue or **No** to cancel.

References

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).

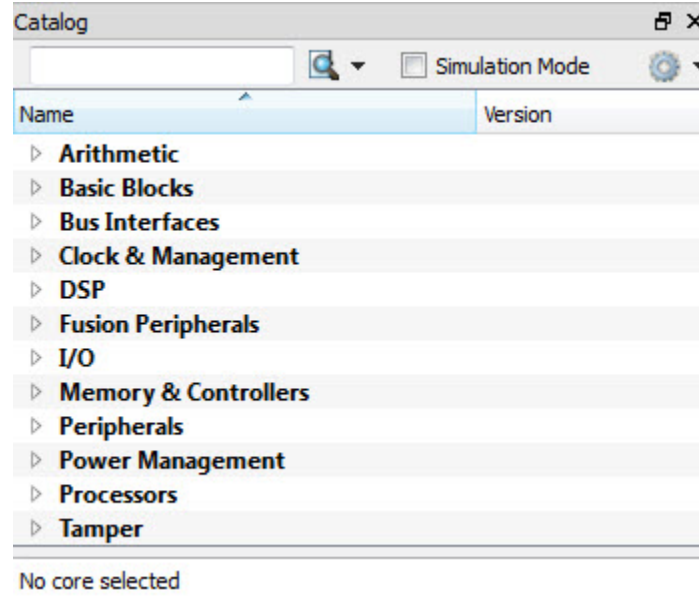


Figure 137 · Libero SoC Catalog

From the Catalog, you can create a component from the list of available cores, add a processor or peripheral, [add a bus interface to your SmartDesign component](#), instantiate simulation cores or add a macro (Arithmetic, Basic Block, etc.) to your SmartDesign component.

Double-click a core to configure it and add it to your design. Configured cores are added to your list of Components/Modules in the Design Explorer.

Click the Simulation Mode checkbox to instantiate simulation cores in your [SmartDesign Testbench](#). Simulation cores are basic cores that are useful for stimulus, such as driving clocks, resets, and pulses.

Viewing Cores in the Catalog

The font indicates the status of the core:

- Plain text - In vault and available for use
- Asterisk after name (*) - Newer version of the core (VLN) available for download
- *Italics* - Core is available for download but not in your vault
- ~~Strikethrough~~ - core is not valid for this version of Libero SoC

The colored icons indicate the license status. Blank means that the core is not license protected in any way. Colored icons mean that the core is license protected, with the following meanings:

Green Key - Fully licensed; supports the entire design flow.

Yellow Key - Has a limited or evaluation license only. Precompiled simulation libraries are provided, enabling the core to be instantiated and simulated within Libero SoC. Using the Evaluation version of the core it is possible to create and simulate the complete design in which the core is being included. The

design is not synthesizable (RTL code is not provided). No license feature in the license.dat file is needed to run the core in evaluation mode. You can purchase a license to generate an obfuscated or RTL netlist.

Yellow Key with Red Circle - License is protected; you are not licensed to use this core.


Right-click any item in the Catalog and choose Show Details for a short summary of the core specifications. Choose Open Documentation for more information on the Core. Right-click and choose Configure Core to open the core generator.

Click the **Name** column heading to sort the cores alphabetically.

You can filter the cores according to the data in the Name and Description fields. Type the data into the filter field to view the cores that match the filter. You may find it helpful to set the Display setting in the [Catalog Options](#) to **List cores alphabetically** when using the filters to search for cores. By default the filter contains a beginning and ending '*', so if you type 'controller' you get all cores with controller in the core name (case insensitive search) or in the core description. For example, to list all the Accumulator cores, in the filter field type:

accu

Catalog Options

Click the Options button  (or the drop-down arrow next to it) to import a core, reload the Catalog, or modify the [Catalog Options](#).


You may want to import a core from a file when:

- You do not have access to the internet and cannot download the core, or
- A core is not complete and has not been posted to the web (you have an evaluation core)

Manually Downloading MegaVaults and Individual CPZ files

When Libero is used in an environment without automatic access to 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 .

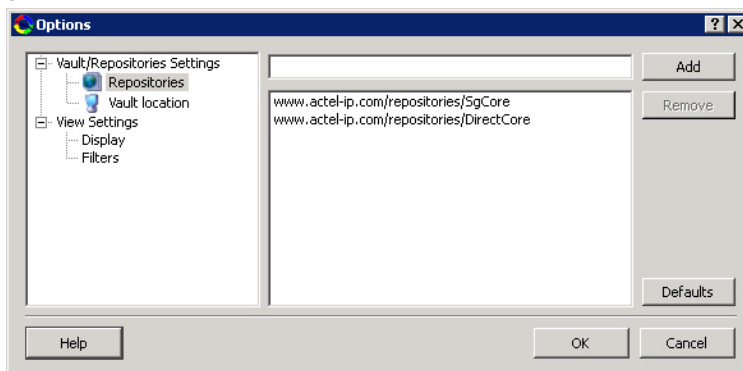


Figure 138 · Catalog Display Options Dialog Box

Vault/Repositories Settings

Repositories

A repository is a location on the web that contains cores that can be included in your design.

The Catalog Options dialog box enables you to specify which repositories you want to display in your Vault. The Vault displays a list of cores from all your repositories, and the [Catalog](#) displays all the cores in your Vault.

The default repository cannot be permanently deleted; it is restored each time you open the Manage Repositories dialog box.

Any cores stored in the repository are listed by name in your Vault and Catalog; repository cores displayed in your Catalog can be filtered like any other core.

Type in the address and click the **Add** button to add new repositories. Click the **Remove** button to remove a repository (and its contents) from your Vault and Catalog. Removing a repository from the list removes the repository contents from your Vault.

Vault location

Use this option to choose a new vault location on your local network. Enter the full domain pathname in the Select new vault location field. Use the format:

```
\\server\share
```

and the cores in your Vault will be listed in the Catalog.

Set ENV variable to set vault location - In addition to setting the vault location using the Catalog dialog box, you can set the vault location using the environment variable `MSCC_IDE_VAULT_LOCATION`. Setting the vault through the environment variable takes precedence over all other options to set vault location.

To set the vault location on Linux, type the following command:

```
setenv MSCC_IDE_VAULT_LOCATION /home/temp_dir
```

To set the vault location on Windows:

Add a new environment variable `MSCC_IDE_VAULT_LOCATION` in System Properties and specify your vault location.

Read only vault

In read only Mega Vault mode, you cannot download, add, or remove cores. However, you can configure and generate cores by creating a temporary extract location to extract the core. This temporary extract location can be set by setting the environment variable `MSCC_IDE_VAULT_EXTRACT_LOCATION`. By setting this environment variable, your configured cores are retained across sessions.

To set the extract location on Linux, type the following command:

```
setenv MSCC_IDE_VAULT_EXTRACT_LOCATION /home/vault_extract
```

To set the extract vault location on Windows:

Add a new environment variable `MSCC_IDE_VAULT_EXTRACT_LOCATION` in System Properties and specify your extract location.

If you do not specify the extract location, a temporary location will be created by Libero and it will be accessed only while the current session is active. If the session is no longer active, the temporary extract location will be cleaned up by Libero. If you specify the extract location, it will be available for any instance of libero on that machine, and it is your responsibility to clean up the extract location.

View Settings

Display

Group cores by function - Displays a list of cores, sorted by function. Click any function to expand the list and view specific cores.

List cores alphabetically - Displays an expanded list of all cores, sorted alphabetically. Double click a core to configure it. This view is often the best option if you are using the filters to customize your display.

Show core version - Shows/hides the core version.

Filters

Filter field - Type text in the Filter Field to display only cores that match the text in your filter. For example, to view cores that include 'sub' in the name, set the Filter Field to **Name** and type **sub**.

Display only latest version of a core - Shows/hides older versions of cores; this feature is useful if you are designing with an older family and wish to use an older core.

Show all local and remote cores - Displays all cores in your Catalog.

Show local cores only - Displays only the cores in your local vault in your Catalog; omits any remote cores.

Show remote cores that are not in my vault - Displays remote cores that have not been added to your vault in your Catalog.

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.

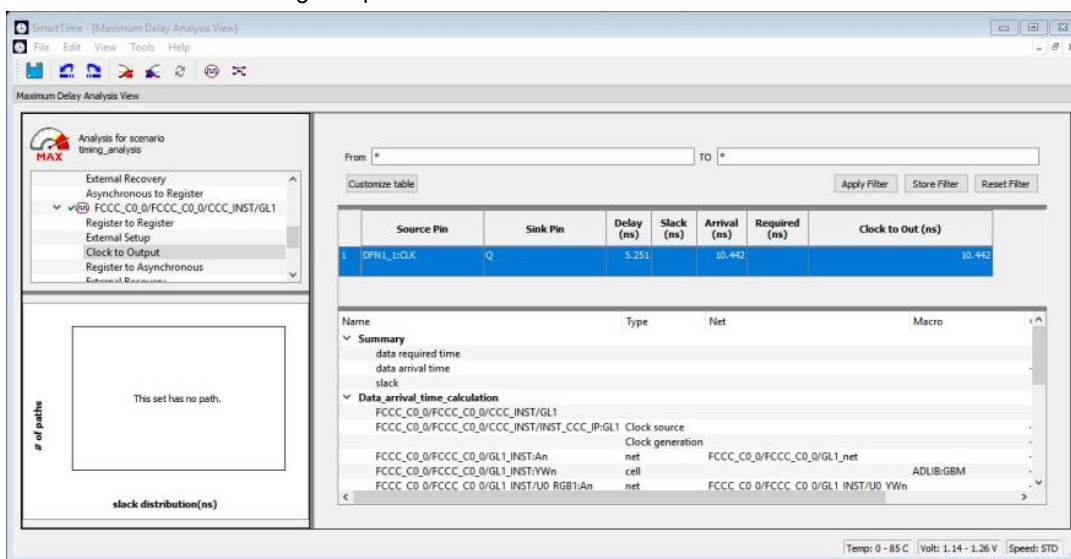


Figure 139 · Maximum Delay Analysis View

If your board has output capacitance of 15 pf on Q, you must perform the following steps to update the timing number:

1. Open the I/O Attribute Editor and change the output load to 15 pf.

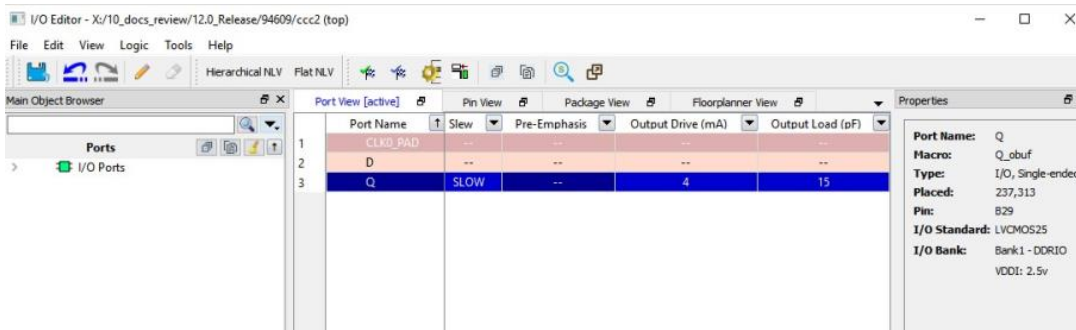


Figure 140 · I/O Attribute Editor View

2. Select **File > Save**.
3. Select **File > Close**.
4. Open the SmartTime Timing Analyzer.

You can see that the Clock to Output delay changed to 5.952 ns.

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. For example, you cannot select eNVM for programming if you do not have eNVM in your design.

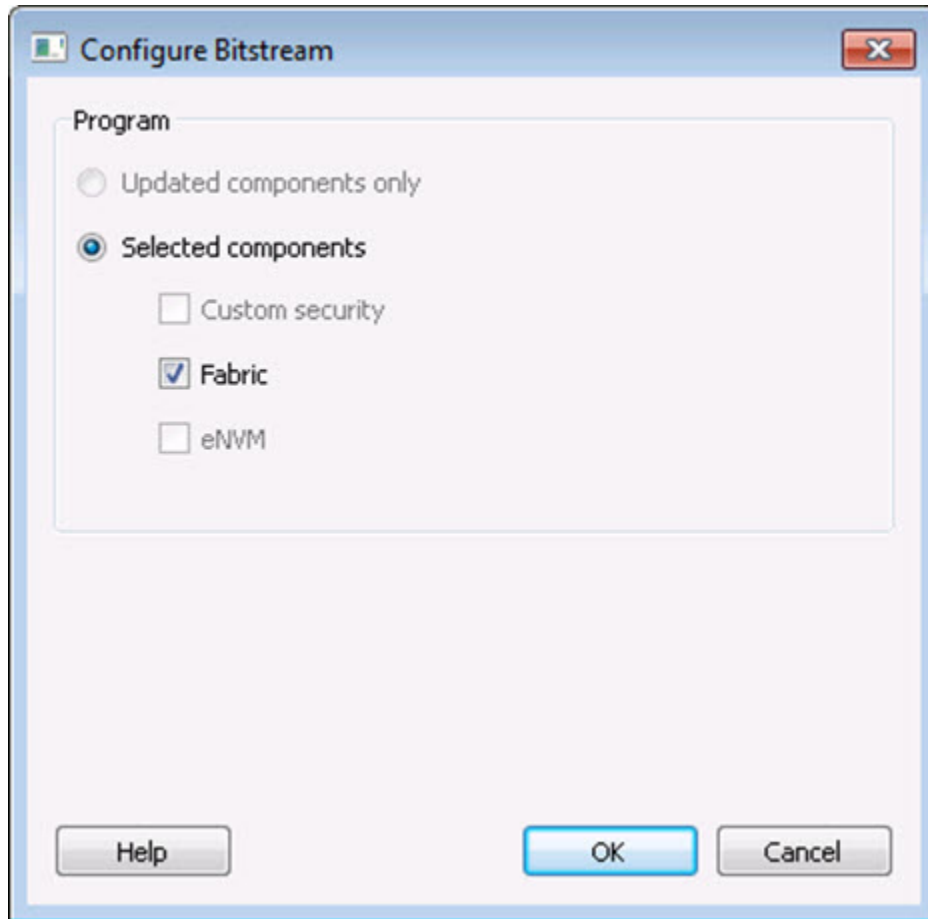


Figure 141 · Configure Bitstream Dialog Box - SmartFusion2 and IGLOO2

Selected components - Updates the components you select, regardless of whether or not they have changed since your last programming.

Note: The Custom security and eNVM components are not available for RTG4 devices.

Importing Source Files – Copying Files Locally

Designer in Libero SoC cannot import files from outside your project without copying them to your local project folder. You may import source files from other locations, but they are always copied to your local folder. Designer in Libero SoC always audits the local file after you import; it does not audit the original file.

When the Project Manager asks you if you want to copy files "locally", it means 'copy the files to your local project folder'. If you do not wish to copy the files to your local project folder, you cannot import them. Your local project folder contains [files](#) related to your Libero SoC project.

Files copied to your local folders are copied directly into their relevant directory: netlists are copied to the *synthesis* folder; source files are copied to *hdl* folder and constraint files to *constraint* folder, etc. The files are also added to the Libero SoC project; they appear in the Files tab.

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**.

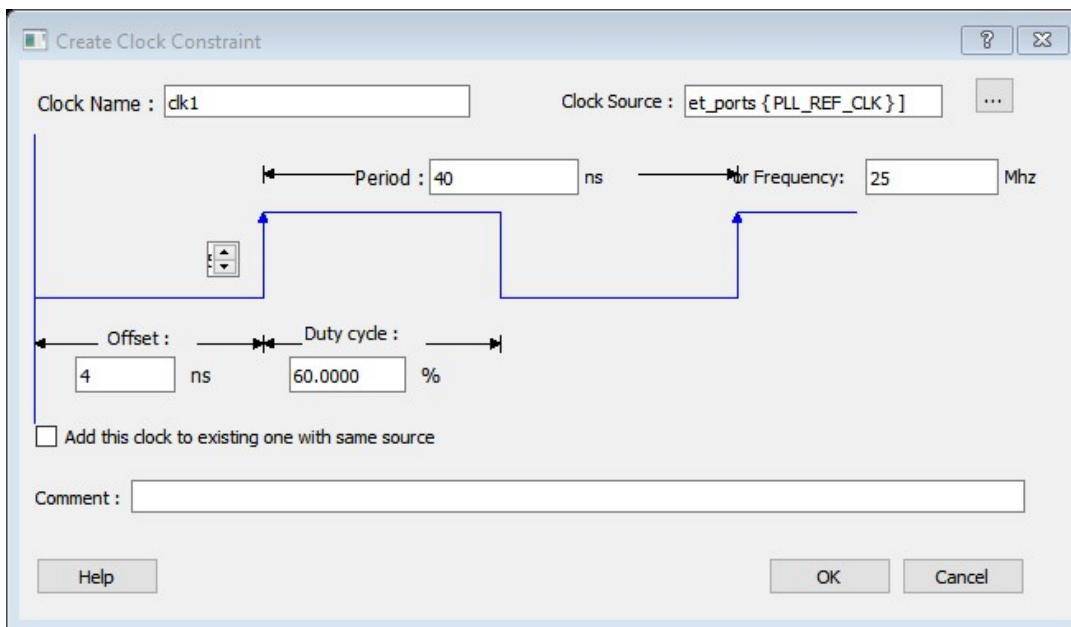


Figure 142 · Create Clock Constraint Dialog Box

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](#)

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**.

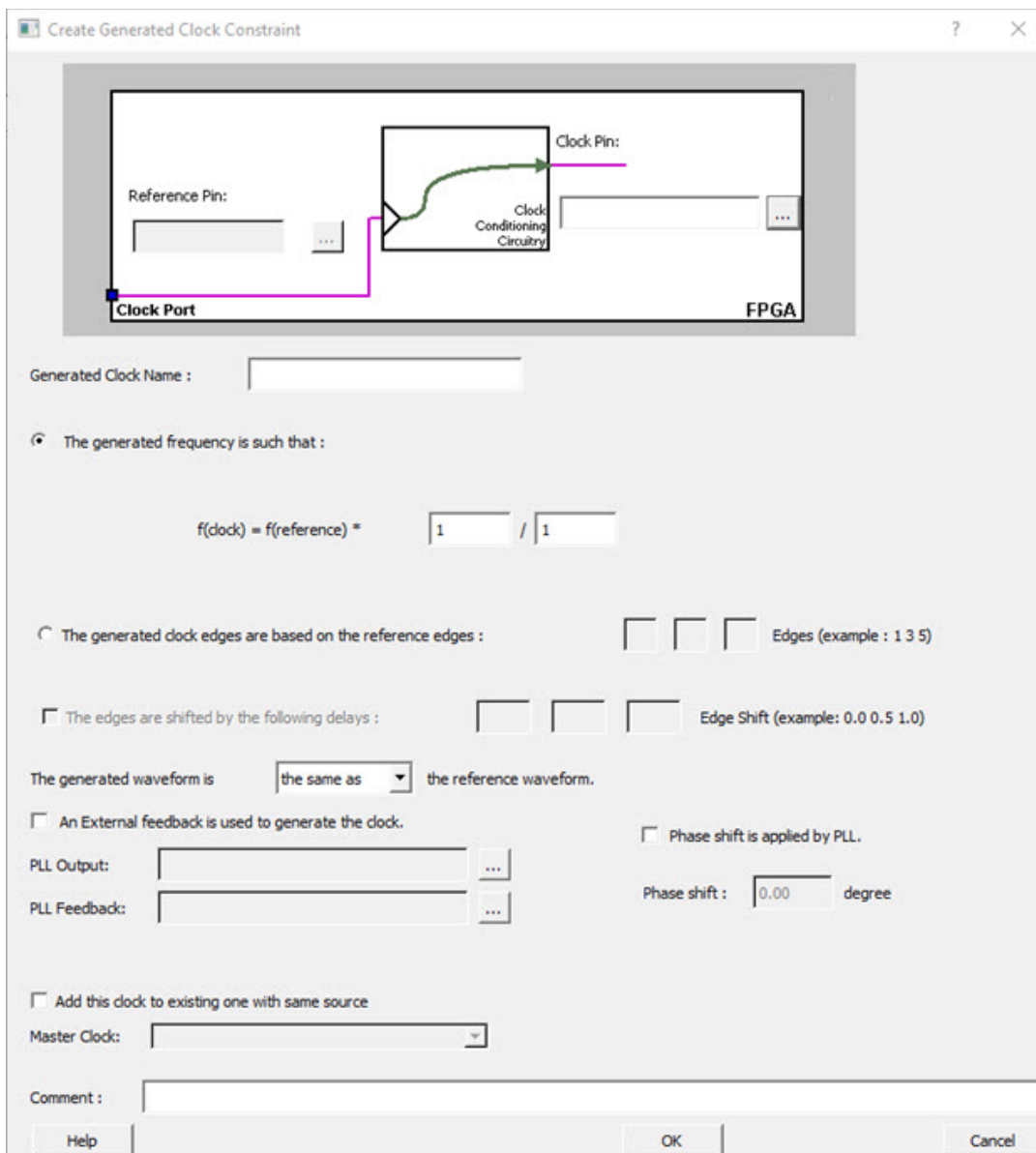


Figure 143 · Create Generated Clock Constraint

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

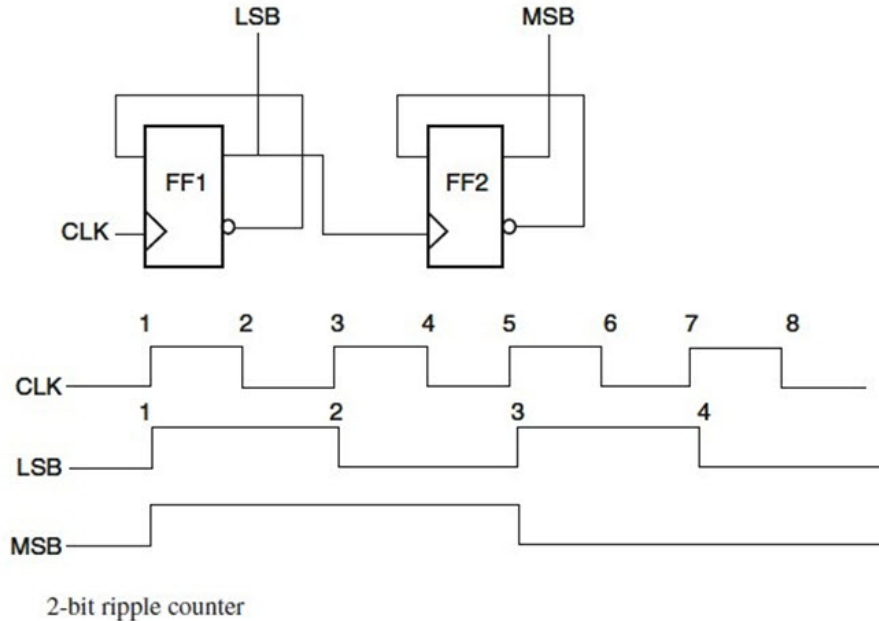
Specifies the name of the Generated clock constraint. This field is required for virtual clocks when no clock source is provided.

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](#)

Design Hierarchy in the Design Explorer

The Design Hierarchy tab displays a hierarchical representation of the design based on the source files in the project. It also displays elaborated hierarchy constructed by propagating correct values for parameters and generics. The software continuously analyzes the source files and updates the content. The Design Hierarchy tab (see figure below) displays the structure of the modules and components as they relate to each other along with parameter/generic names and its values on the tool tip for which the module is instantiated. It also displays architecture name for a given entity and Configuration for VHDL modules.

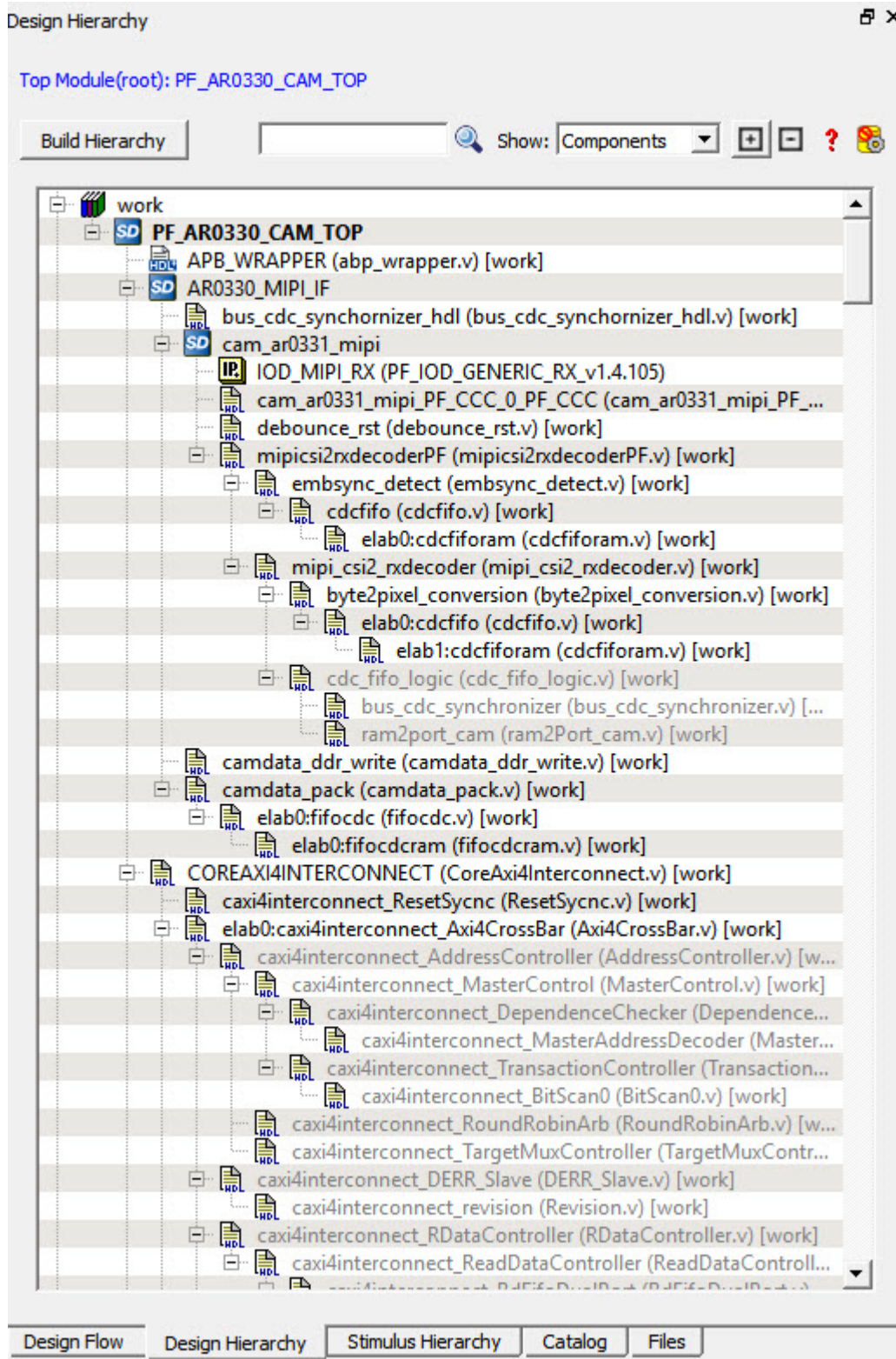


Figure 144 · Design Hierarchy

A module can have multiple elaborations depending on the different parameters/generics used in the instantiation of the module and all of these elaborated modules will be shown in the Design Hierarchy. The parameterized instantiated module will be shown as

elab<num>:<modulename>

Modules are instantiated with their actual names in the SmartDesign. If a module with elaborated name in the Design Hierarchy has to be instantiated in the SmartDesign, an instance of the original module is created in the SmartDesign. The following figure shows the design hierarchy with elaborated modules.



Figure 145 · Design Hierarchy with Elaborated modules (Verilog)



Figure 146 · Design Hierarchy with Elaborated modules (VHDL)

Modules which are not part of the elaboration will be shown in the complete hierarchy but they remain grayed out. When you create a core from a module, all the elaborated modules of that module will be shown as HDL+ core modules. You can get the parameter value of an elaborated module by selecting **Show Module parameters** on the right click menu of the elaborated module.

Note: A tool tip on each module shows all the parameters with their values for the instantiated module.

Note: Synthesis output will be the same for different elaborations of the same module i.e. **elab0:module1** and **elab1:module1** will have the same synthesis output. When one of the elaborated module is set as root, all the elaborations will be highlighted in the Design Hierarchy as shown in the below figure.

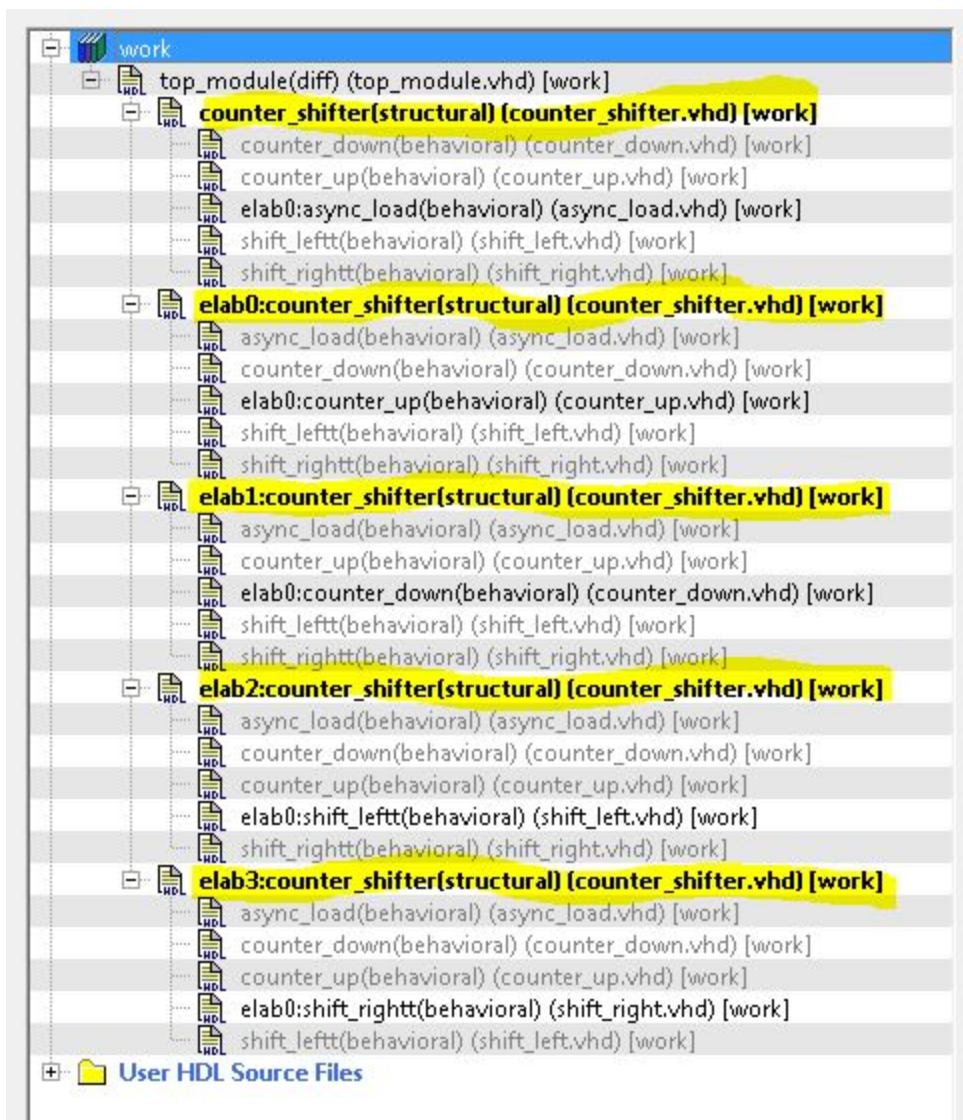


Figure 147 · Design Hierarchy when one of the elaborated module is set as root

You can change the display mode of the Design Hierarchy by selecting **Components** or **Modules** from the **Show** drop-down list. The components view displays the entire design hierarchy; the modules view displays only schematic and HDL modules.

You can build the Design Hierarchy and Simulation Hierarchy by clicking the **Build Hierarchy** button.

A yellow icon  **Build Hierarchy** 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 6 · Design Hierarchy Icons

Icon	Description
	SmartDesign component
	SmartDesign component with HDL netlist not generated
	IP core was instantiated into SmartDesign but the HDL netlist has not been generated
	Core
	Error during core validation
	Updated core available for download
	HDL netlist
	Shows ungenerated components
	Shows unknown modules
	Expands all the files and folders in the Design Hierarchy
	Collapses all the files and folders in the Design Hierarchy
	Finds the files in the Design Hierarchy

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.

Example Using STAPL File

If a STAPL file is exported, the digests will be printed in the log window, as shown in the example below.

Libero log:

```
Opened 'D:\flashpro_files\m2s005_digest1\designer\al_MSS\al_MSS_fp\al_MSS.pro'
The 'open_project' command succeeded.
PDB file
'D:\flashpro_files\m2s005_digest1\designer\al_MSS\4a8552f8-57ee-4baa-97ee-2baa57ee2baa.pdb' has
been loaded successfully.
DESIGN : al_MSS; CHECKSUM : DE15; PDB_VERSION : 1.9
The 'load_programming_data' command succeeded.
Sucessfully exported STAPL file:
'D:\flashpro_files\m2s005_digest1\designer\al_MSS\export\al_MSS.stp'; file programs
Fabric
and eNVM.
Fabric component digest:
276fbefb0a18cc0de1d45efc84589745ee02fc2adbcc1259fbeb674094754014
eNVM component digest:
6b2c2353e25c5982643c32640ac16c581874c8950300135622c126ee22d8b1de
Finished: Thu Jan 22 12:37:32 2015 (Elapsed time 00:00:06)
The 'export_single_stapl' command succeeded.
The 'set_programming_file' command succeeded.
Project saved.
The 'save_project' command succeeded.
Project closed.
```

The export folder will contain the exported STAPL file along with digest file. In this example, there will be two files, "a1_MSS.stp" and "a1_MSS_stp.digest". The content of the a1_MSS_stp.digest file is shown below:

```
Fabric component digest: 276fbefb0a18cc0de1d45efc84589745ee02fc2adbcc1259fbeb674094754014
eNVM component digest: 6b2c2353e25c5982643c32640ac16c581874c8950300135622c126ee22d8b1de
```

When the device is programmed in the production programming house by loading the STAPL file in FlashPro, the log will be as follows:

```
programmer '73207' : Scan Chain...
Warning: programmer '73207' : Vpump has been selected on programmer AND an externally
provided Vpump has also been detected. Using externally provided Vpump voltage source.
programmer '73207' : Check Chain...
programmer '73207' : Scan and Check Chain PASSED.
programmer '73207' : device 'M2S/M2GL005(S)' : Executing action PROGRAM
programmer '73207' : device 'M2S/M2GL005(S)' : Family: SmartFusion2
programmer '73207' : device 'M2S/M2GL005(S)' : Product: M2S005
programmer '73207' : device 'M2S/M2GL005(S)' : EXPORT ISC_ENABLE_RESULT[32] = 007c6b44
programmer '73207' : device 'M2S/M2GL005(S)' : EXPORT CRCERR: [1] = 0
programmer '73207' : device 'M2S/M2GL005(S)' : EXPORT EDCERR: [1] = 0
programmer '73207' : device 'M2S/M2GL005(S)' : TEMPGRADE: ROOM
programmer '73207' : device 'M2S/M2GL005(S)' : EXPORT VPPRANGE: [3] = 2
programmer '73207' : device 'M2S/M2GL005(S)' : VPPRANGE: HIGH
```

```

programmer '73207' : device 'M2S/M2GL005(S)' : EXPORT TEMP: [8] = 6b
programmer '73207' : device 'M2S/M2GL005(S)' : EXPORT VPP: [8] = 7c
programmer '73207' : device 'M2S/M2GL005(S)' : Programming FPGA Array and eNVM...
programmer '73207' : device 'M2S/M2GL005(S)' : EXPORT Fabric component digest[256] =
276fbefb0a18cc0de1d45efc84589745ee02fc2adbcc1259fbeb674094754014
programmer '73207' : device 'M2S/M2GL005(S)' : EXPORT eNVM component digest[256] =
6b2c2353e25c5982643c32640ac16c581874c8950300135622c126ee22d8b1de
programmer '73207' : device 'M2S/M2GL005(S)' :
=====
programmer '73207' : device 'M2S/M2GL005(S)' : EXPORT DSN[128] =
c6e99c2d1a992f13cf8231c4be847acb
programmer '73207' : device 'M2S/M2GL005(S)' :
=====
programmer '73207' : device 'M2S/M2GL005(S)' : Finished: Thu Jan 22 17:57:37 2015
(Elapsed time 00:00:19)
programmer '73207' : device 'M2S/M2GL005(S)' : Executing action PROGRAM PASSED.
programmer '73207' : Chain programming PASSED.
Chain Programming Finished: Thu Jan 22 17:57:37 2015 (Elapsed time 00:00:19)
o - o - o - o - o - o

```

The log file is saved and sent back to the customer, who can verify that the device was programmed with the correct design by comparing the digests in the log file to the contents of the a1_MSS_stp.digest file.

Example Using Programming Job

If a programming job is exported, the digests will be printed in the log window, as shown in the example below.

Libero log:

```

Software Version: 11.5.1.5
Opened 'D:/flashpro_files/m2s005_digest1/designer/a1_MSS/a1_MSS_fp/a1_MSS.pro'
The 'open_project' command succeeded.
PDB file
'D:\flashpro_files\m2s005_digest1\designer\al_MSS\83ce6816-1e56-496b-9e56-
d96b1e56d96b.pdb' has
been loaded successfully.
DESIGN : a1_MSS; CHECKSUM : DE15; PDB_VERSION : 1.9
The 'load_programming_data' command succeeded.
Sucessfully exported STAPL file:
'D:\flashpro_files\m2s005_digest1\designer\al_MSS\export\al_MSS_M2S005.stp'; file
programs
Fabric and eNVM.
Fabric component digest:
276fbefb0a18cc0de1d45efc84589745ee02fc2adbcc1259fbeb674094754014
eNVM component digest:
6b2c2353e25c5982643c32640ac16c581874c8950300135622c126ee22d8b1de
Finished: Wed Jan 28 16:48:56 2015 (Elapsed time 00:00:06)
The 'export_single_stapl' command succeeded.
The 'set_programming_file' command succeeded.
Project saved.
The 'save_project' command succeeded.
Project closed.

```

The export folder will contain the exported programming job along with digest file. In this example, there will be two files, "a1_MSS.job" and "a1_MSS_job.digest". The content of the a1_MSS_job.digest file is shown below:

```

Fabric component digest: 276fbefb0a18cc0de1d45efc84589745ee02fc2adbcc1259fbeb674094754014
eNVM component digest: 6b2c2353e25c5982643c32640ac16c581874c8950300135622c126ee22d8b1de

```


When the device is programmed in the production programming house by loading the programming job in FlashPro Express, the log will be as follows:

```

programmer '73207' : Scan Chain...
Warning: programmer '73207' : Vpump has been selected on programmer AND an externally
provided Vpump has also been detected. Using externally provided Vpump voltage source.
programmer '73207' : Check Chain...
programmer '73207' : Scan and Check Chain PASSED.
programmer '73207' : device 'M2S/M2GL005(S)' : Executing action PROGRAM
programmer '73207' : device 'M2S/M2GL005(S)' : Family: SmartFusion2
programmer '73207' : device 'M2S/M2GL005(S)' : Product: M2S005
programmer '73207' : device 'M2S/M2GL005(S)' : EXPORT ISC_ENABLE_RESULT[32] = 007c6b44
programmer '73207' : device 'M2S/M2GL005(S)' : EXPORT CRCERR: [1] = 0
programmer '73207' : device 'M2S/M2GL005(S)' : EXPORT EDCERR: [1] = 0
programmer '73207' : device 'M2S/M2GL005(S)' : TEMPGRADE: ROOM
programmer '73207' : device 'M2S/M2GL005(S)' : EXPORT VPPRANGE: [3] = 2
programmer '73207' : device 'M2S/M2GL005(S)' : VPPRANGE: HIGH
programmer '73207' : device 'M2S/M2GL005(S)' : EXPORT TEMP: [8] = 6b
programmer '73207' : device 'M2S/M2GL005(S)' : EXPORT VPP: [8] = 7c
programmer '73207' : device 'M2S/M2GL005(S)' : Programming FPGA Array and eNVM...
programmer '73207' : device 'M2S/M2GL005(S)' : EXPORT Fabric component digest[256] =
276fbefb0a18cc0de1d45efc84589745ee02fc2adbbcc1259fbeb674094754014
programmer '73207' : device 'M2S/M2GL005(S)' : EXPORT eNVM component digest[256] =
6b2c2353e25c5982643c32640ac16c581874c8950300135622c126ee22d8b1de
programmer '73207' : device 'M2S/M2GL005(S)' :
=====
programmer '73207' : device 'M2S/M2GL005(S)' : EXPORT DSN[128] =
c6e99c2d1a992f13cf8231c4be847acb
programmer '73207' : device 'M2S/M2GL005(S)' :
=====
programmer '73207' : device 'M2S/M2GL005(S)' : Finished: Thu Jan 22 17:57:37 2015
(Elapsed time 00:00:19)
programmer '73207' : device 'M2S/M2GL005(S)' : Executing action PROGRAM PASSED.
programmer '73207' : Chain programming PASSED.
Chain Programming Finished: Thu Jan 22 17:57:37 2015 (Elapsed time 00:00:19)
o - o - o - o - o - o - o

```

The log file is saved and sent back to the customer, who can verify that the device was programmed with the correct design by comparing the digests in the log file above to the contents of the a1_MSS_job.digest file.

See Also

[Export Bitstream - SmartFusion2, IGLOO2, and RTG4 Only](#)

Editable Constraints Grid

The Constraints Editor enables you to add, edit and delete.

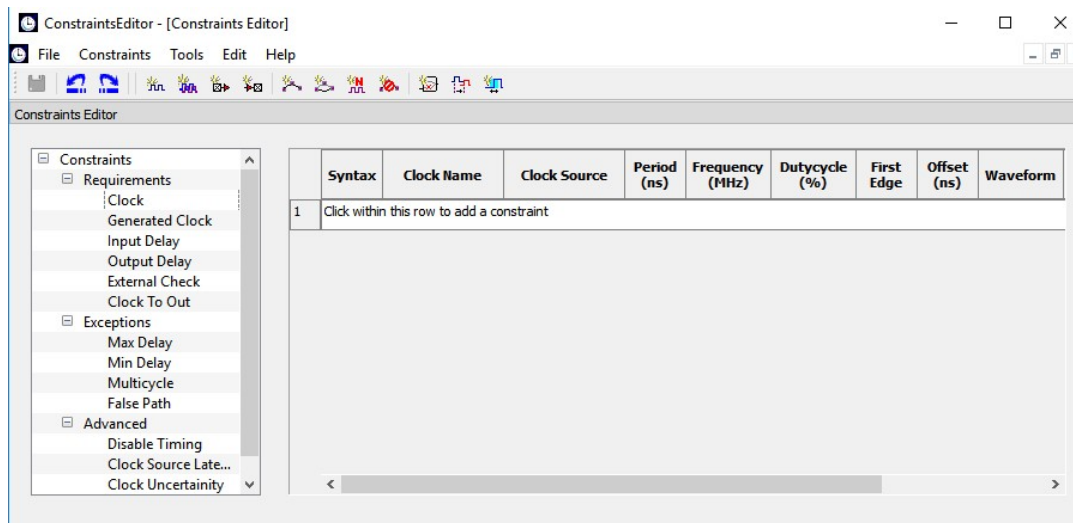


Figure 148 · Constraints Editor

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**.

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
logfile:extended_run.log "script_args:-root path/designer/module_name [-n numPasses] [-
starting_seed_index numIndex] [-compare_criteria value] [-c clockName] [-analysis value] [-
slack_criteria value] [-stop_on_success] [-timing_driven|standard] [-power_driven value]
[-placer_high_effort value]"
```

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.

Value	Description
frequency	Use clock frequency as criteria for comparing the results between passes. This option can be used in conjunction with the <code>-c</code> option (described below).
violations	Use timing violations as criteria for comparing the results between passes. This option can be used in conjunction with the <code>-analysis</code> , <code>-slack_criteria</code> and <code>-stop_on_success</code> options (described below).
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:

Value	Description
max	Examines timing violations (slack) obtained from maximum delay analysis. This is the default.
min	Examines timing violations (slack) obtained from minimum delay analysis.

`[-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:

Value	Description
worst	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.

Value	Description
tns	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.

`[-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:

Value	Description
off	Does not run power-driven layout.
on	Enables power-driven layout.

`[-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:

Value	Description
off	Runs layout in regular effort.
on	Activates high effort layout mode.

Return

A non-zero value will be returned on error.

Supported Families

SmartFusion2, IGLOO2, RTG4

Exceptions

None

See Also

[Place and Route - SmartFusion2, IGLOO2, RTG4](#)

[Multiple Pass Layout - SmartFusion2, IGLOO2, RTG4](#)

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.

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, PRB (for Silicon Explorer), TCL (used to run designer), designer.log (logfile)

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

simulation directory - meminit.dat, modelsim.ini files

smartgen directory - GEN files and LOG files from generated cores

stimulus directory - BTIM and VHD stimulus files

synthesis directory - *.edn, *_syn.prj (Synplify log file), *.srr (Synplify logfile), *.tcl (used to run synthesis) and many other files generated by the tools (not managed by Libero SoC)

tooldata directory - includes the log file for your project with device details.

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

File Type	File Extension
Behavioral and Structural VHDL; VHDL Package	*.vhd, *.vhdl
Design Block Core	*.gen
Verilog Include	*.h

File Type	File Extension
Behavioral and Structural Verilog	*.v, *.sv
Netlist Verilog	*.vm
Stimulus	*.vhd, *.vhdl, *.v, *.sv
EDIF Netlist	*.edn
Memory file	*.mem
Components (Designer Blocks, Synplify DSP)	*.cxf

Layout Error Message: layoutg4DesignHard

This design is very difficult to place, and high-effort techniques were required to make it fit. This may lead to increased layout runtime and diminished timing performance.

This message typically appears in designs with high utilization -- a very full design, or a design with region constraints which are, themselves, very full. It can also occur in designs with moderate utilization but with numerous, long carry chains.

No immediate action is required on the user's part. However, if this notice is observed during Layout, the resultant performance of the design and the runtime of the Layout tools may not be optimal, and there is a strong possibility that reducing the size of the design, or relaxing region and floorplanning constraints, will help to improve timing closure and runtime.

Layout Error Message: layoutg4NoValidPlacement

This is a generic error produced by the placer when it is unable to place a design. The most common cause for this failure is that the placer was unable to find a solution which could fit the design into the chip, either because the design is close to maximum utilization, or logic cannot be fit into user-defined region constraints.

If Libero is unable to find a legal placement, a list of unplaced cells will be provided in the log. The cells in this list may not be the cause of the placement problem; it is quite possible that some other constrained block of logic 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.

list_clock_groups

This Tcl command lists all existing clock groups in the design.

```
list_clock_groups
```

Arguments

None

Supported Families

SmartFusion2

IGLOO2

RTG4

Example

```
list_clock_groups
```

See Also

[set_clock_groups](#)

[remove_clock_groups](#)

Project Manager – Catalog — Cores

This Catalog in Project Manager enables you to download cores from a [web repository](#) into a Vault.

A Vault is a local directory (either local to your machine or on the local network) that contains cores downloaded from one or more repositories. A repository is a location on the web that contains cores that can be included in your design.

The Catalog displays all the cores in your Vault.

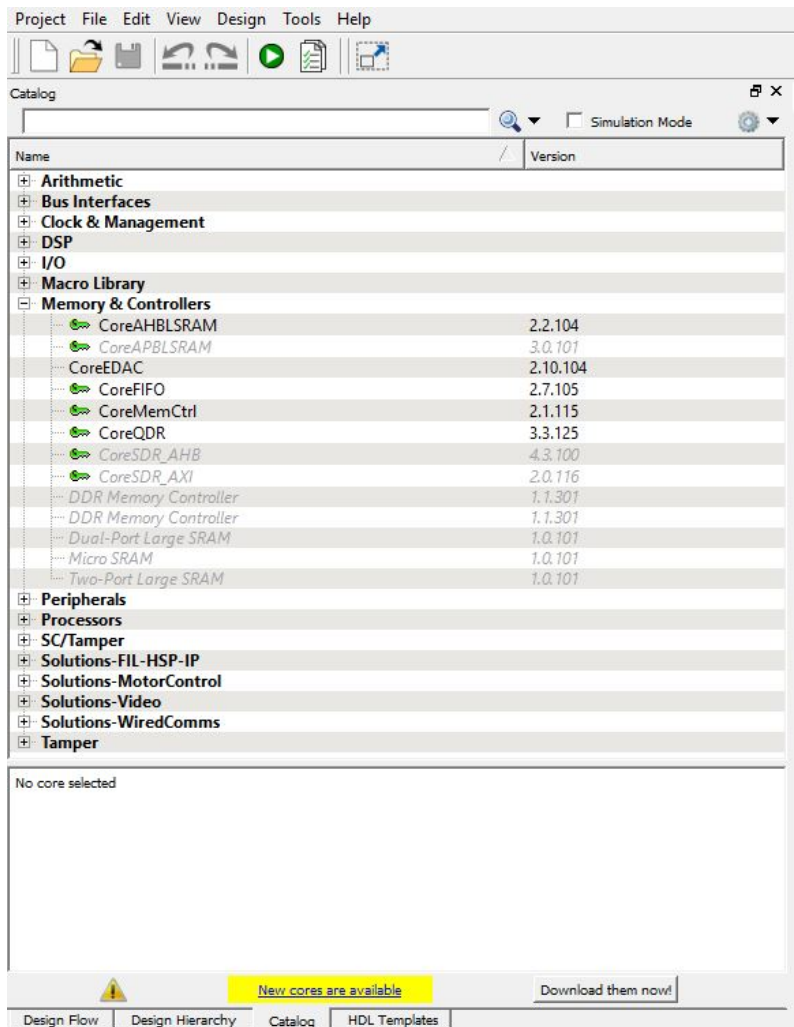


Figure 149 · Project Manager - Catalog

Use the Filter to find any string that exists either in the core name or the Core Description. 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.

The colored icons indicate the license status. Blank means that the core is not license protected in any way. Colored icons means 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 the Microsemi 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.

Stop Downloads - Interrupts the download for any cores being added to your Vault.

Project Settings Dialog Box

The Project Settings dialog box enables you to modify your Device, HDL, and Design Flow settings and your Simulation Options. In Libero SoC, from the Project menu, click **Project Settings**.

The following figure shows an example of the Project Settings dialog box.

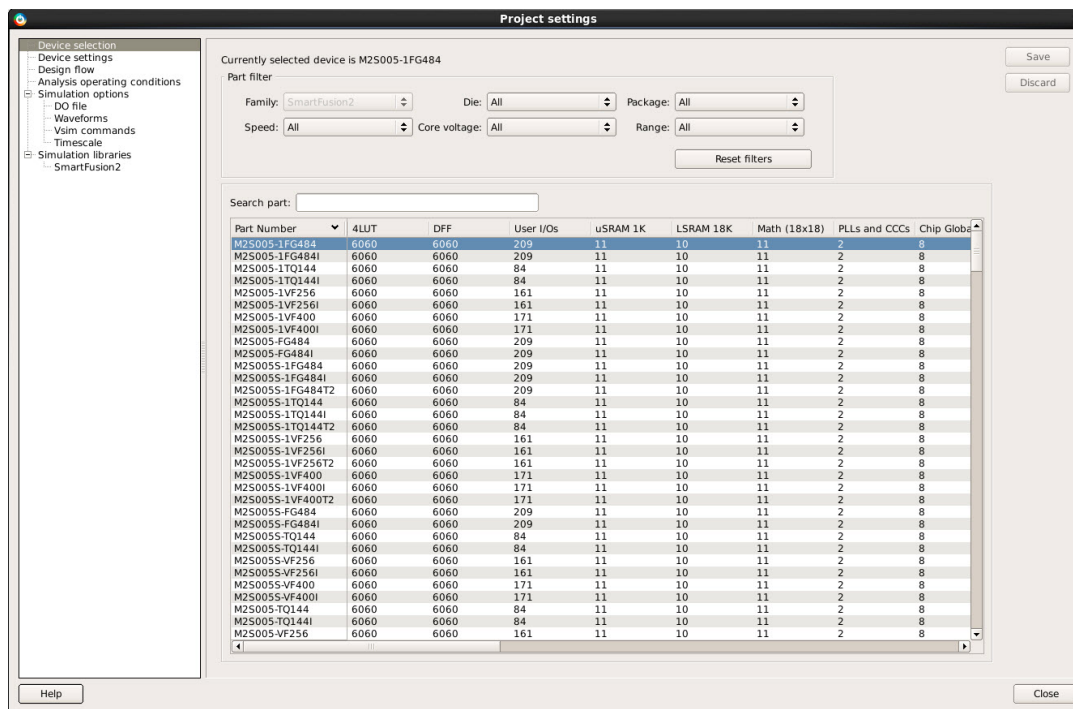


Figure 150 · Project Settings Dialog Box

Device Selection

Sets the device Family, 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.

Enable Single Event Transient mitigation (RTG4 only) - Controls the mitigation of Single Event Transient (SET) in the FPGA fabric. When this box is checked, SET filters are turned on globally (including URAM, LSRAM, MACC, I/O FF, Regular FF, DDR_IN, DDR_OUT) to help mitigate radiation-induced transients. By default, this box is unchecked.

PLL Supply Voltage (V) (SmartFusion2, IGLOO2 only) - Sets the voltage for the power supply that you plan to connect to all the PLLs in your design, such as MDDR, FDDR, SERDES, and FCCC. Select either 2.5V or 3.3V.

Note: This voltage setting must match the PLL analog supply voltage on the board to ensure that the PLL works properly.

VDD Supply Ramp Time (SmartFusion2 and IGLOO2 only) - The power-on reset circuitry in the SmartFusion2 and IGLOO2 devices requires the VDD and VPP power supplies to ramp monotonically from 0 V to the minimum recommended operating voltage within a predefined time. Select one of four values for the predefined time: 50 us, 1 ms, 10 ms, and 100 ms.

System controller suspended mode - When enabled (usually for safety-critical applications), the System Controller is held in a reset state after the completion of device initialization. This state protects the device from unintended device programming or zeroization of the device due to SEUs (Single Event Upsets). In this mode, the System Controller cannot provide any system services such as Flash*Freeze service, cryptographic services or programming services.

Design Flow

See the [Project Settings: Design flow](#) topic for more information.

Analysis Operating Conditions (For SmartFusion2, IGLOO2, RTG4)

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.

Once the "Range" value is set, the Minimum/Typical/Maximum values for the selected range are displayed.

Radiation (krad) - For RTG4 only, enter the radiation value (in krads) for your device. Valid range is from 0 to 300.

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: Design flow

To access the Design flow page, from the Project menu choose **Project Settings** and click the **Design flow** tab.

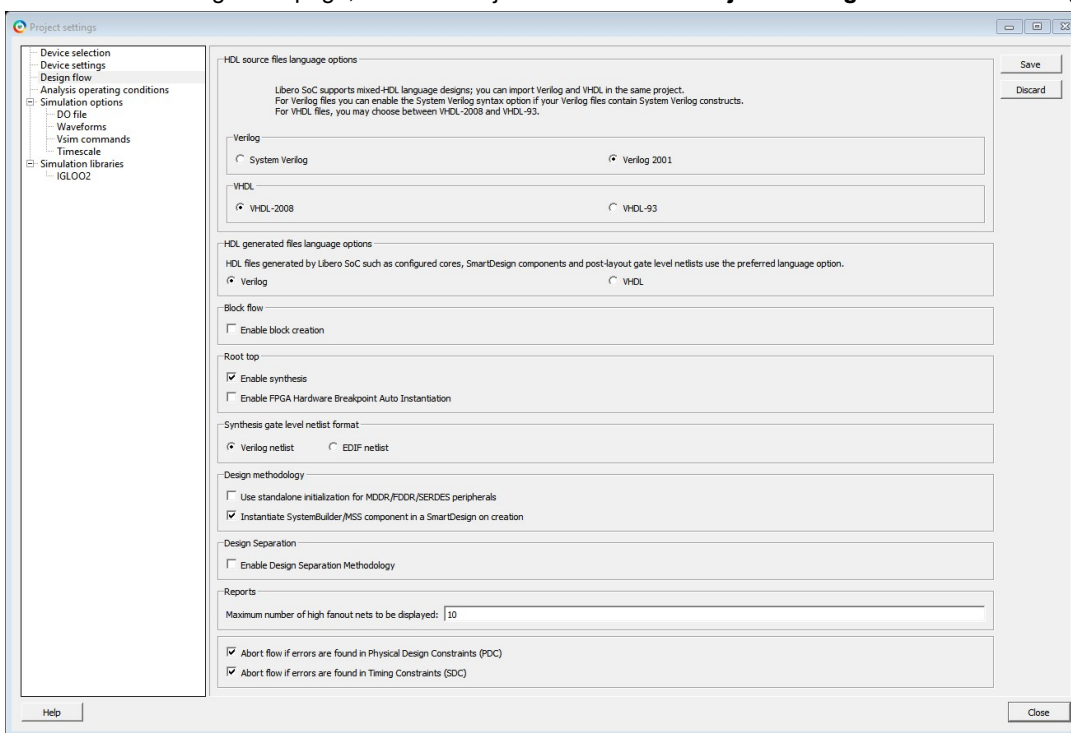


Figure 151 · Project Settings Dialog Box – Design Flow Tab

HDL source files language options

Libero SoC supports mixed-HDL language designs. You can import Verilog and VHDL in the same project.

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 2005 (IEEE Standard 1364-2005)
- Verilog 2001 (IEEE Standard 1364-2001)
- Verilog 1995 (IEEE Standard 1364-1995)
- SystemVerilog 2012 (IEEE Standard 1800-2012)
- VHDL-2008 (IEEE Standard 1076-2008)

- VHDL-93 (IEEE Standard 1076-1993)

HDL generated files language options

HDL files generated by Libero SoC can be set to use VHDL or Verilog. If there are no other considerations, it is generally recommended to use the same HDL language as you are using for HDL source files, as this may reduce the cost of simulation licenses.

Block flow

Enable block creation - Enables you to create and publish design blocks (*.cxz files) in Libero SoC. Design blocks are low-level components that may have completed the place-and-route step and met the timing and power requirements. These low-level design blocks can then be imported into a Libero SoC project and re-used as components in a higher level design. See [Designing with Designer Block Components](#) in Online Help for more information.

Root <module_name>

Enable synthesis - Option to enable or disable synthesis for your root file; useful if you wish to skip synthesis on your root file by default.

Enable FPGA Hardware Breakpoint Auto Instantiation - The 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: This option is enabled when you select Verilog netlist.

Synthesis gate level netlist format

Sets your gate level netlist format to Verilog or EDIF. For Secure IP design flow, you must set the format to Verilog. See the [Microsemi website](#) for more information about the Secure IP flow.

Design methodology (Available only in SmartFusion2 and IGLOO2)

Use standalone initialization for MDDR/FDDR/SERDES peripherals – Enables you to create your own peripheral initialization logic in SmartDesign for each of your design peripherals (MDDR/FDDR/SERDES). When checked, System Builder does not build the peripherals initialization logic for you. Standalone initialization is useful if you want to make the initialization logic of each peripheral separate from and independent of each other. For more information, refer to the [SmartFusion2 Standalone Peripheral Initialization User Guide](#) or the [IGLOO2 Standalone Peripheral Initialization User Guide](#).

Instantiate SystemBuilder/MSS component in a SmartDesign on creation - Un-check this box if you are using this project to create System Builder or MSS components and do not plan on using them in a SmartDesign based design. This is especially useful for design flows where the System Builder or MSS component is stitched in a design using HDL.

Design Separation(Available only in SmartFusion2 and IGLOO2)

Enable Design Separation Methodology - Enables you to create independent critical subsystems required to implement security and safety critical applications on a single FPGA. When checked, the design is divided into sub-systems published in terms of Block Elements.

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.

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.

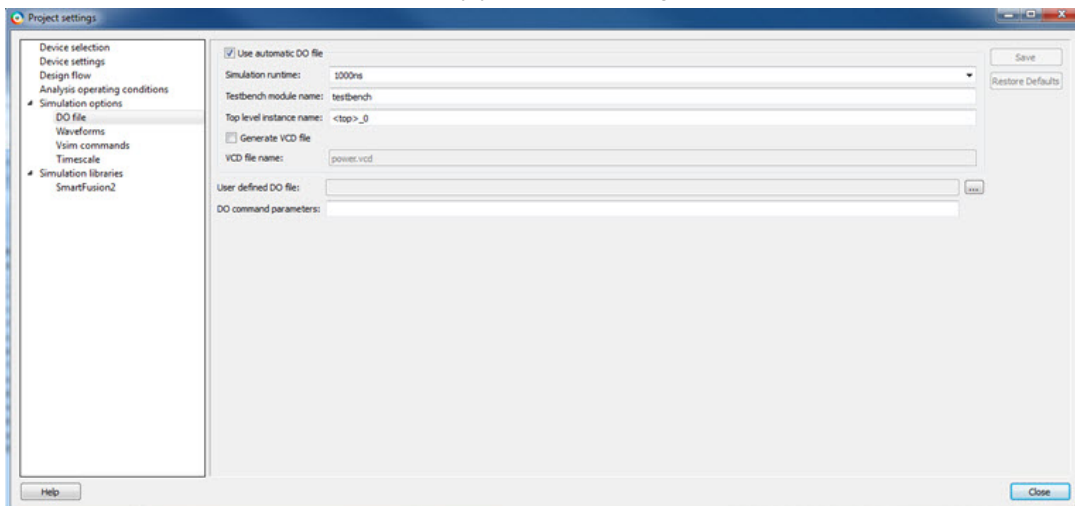


Figure 152 · Project Settings: DO File

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 'add wave /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

- **Post-layout simulation only:**
 - **SDF timing delays** - Select Minimum (Min), Typical (Typ), or Maximum (Max) timing delays in the back-annotated SDF file.
 - **Disable Pulse Filtering during SDF-based Simulations** - When the check box is enabled the **+pulse_int_e/1 +pulse_int_r/1 +transport_int_delays** switch is included with the vsim command for post-layout simulations; the checkbox is disabled by default.
- **Resolution** - The default is family specific (review the dialog box for your default setting), but you can customize it to fit your needs. 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.

Family	Default Resolution
SmartFusion2	1 fs
IGLOO2	1 ps
RTG4	1 ps

- **Additional options** - Text entered in this field is added to the vsim command.
 - SRAM ECC Simulation (RTG4) -
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.

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.

remove_clock_groups

This Tcl command removes a clock group by name or by ID.

```
remove_clock_groups [-id id# | -name groupname] \
[-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]

Supported Families

SmartFusion2

IGLOO2

RTG4

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](#)

Running Libero SoC from your Software Tool Chain

When launched from your software toolchain, Libero SoC becomes solely an MSS configurator. This can be useful if you are responsible for the embedded code development for the SmartFusion device and are more comfortable in your existing software tool chain.

Any FPGA fabric development needs to be done using the regular Libero® SoC tool flow. Using the Libero SoC in the software toolchain mode only enables you to configure the SmartFusion Microcontroller Subsystem (MSS) and not the FPGA fabric.

The MSS Configurator can be integrated in any software development IDE that supports external tools. Configure your IDE to start the Libero SoC executable and use the parameters below to customize your interface. For SoftConsole, Keil and IAR the parameters are:

```
"PROJECT_LOCATION:<path>" //Project directory location, and the location of your
generated MSS files.
"DESIGN_NAME:<name>" //Name of your design.
"STARTED_BY:<tool>" //Identifies which tool invoked the MSS Configurator; can be
SoftConsole, Keil, or IAR EWARM
```

See Also

[Exporting Firmware and the Software IDE Workspace](#)

[Software IDE Integration](#)

[View/Configure Firmware Cores](#)

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.

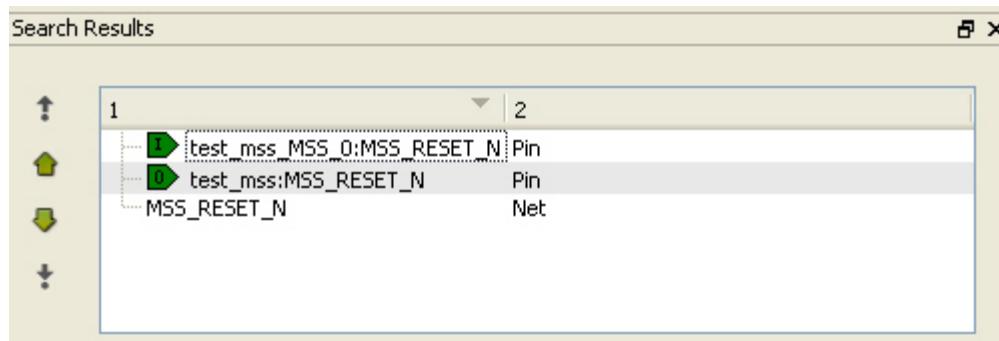


Figure 153 · Search Results

Current Open SmartDesign

Searches your open SmartDesign, returns results in the Search window.

Type: Choose Instance, Net or Pin to narrow your search.

Query: Query options vary according to Type.

Type	Query Option	Function
Instance	Get Pins	Search restricted to all pins
	Get Nets	Search restricted to all nets
	Get Unconnected Pins	Search restricted to all unconnected pins
Net	Get Instances	Searches all instances
	Get Pins	Search restricted to all pins
Pin	Get Connected Pins	Search restricted to all connected pins
	Get Associated Net	Search restricted to associated nets
	Get All Unconnected Pins	Search restricted to all unconnected pins

Current Open Text Editor

Searches the open text file. If you have more than one text file open you must place the cursor in it and click CTRL + F to search it.

Find All: Highlights all finds in the text file.

Next: Proceed to next instance of found text.

Previous: Proceed to previous instance of found text.

Replace with: Replaces the text you searched with the contents of the field.

Replace: Replaces a single instance.

Replace All: Replaces all instances of the found text with the contents of the field.

Design Hierarchy

Searches your Design Hierarchy; results appear in the Search window.

Find All: Displays all finds in the Search window.

Stimulus Hierarchy

Searches your Stimulus Hierarchy; results appear in the Search window.

Find All: Displays all finds in the Search window.

Log Window

Searches your Log window; results are highlighted in the Log window - they do not appear in the Search Results window.

Find All: Highlights all finds in the Log window.

Next: Proceed to next instance of found text.

Previous: Proceed to previous instance of found text.

Reports

Searches your Reports; returns results in the Reports window.

Find All: Highlights all finds in the Reports window.

Next: Proceed to next instance of found text.

Previous: Proceed to previous instance of found text.

Files

Searches your local project file names for the text in the Search field; returns results in the Search window.

Find All: Lists all search results in the Search window.

Files on disk

Searches the files' content in the specified directory and subdirectories for the text in the Search field; returns results in the Search window.

Find All: Lists all finds in the Search window.

File type: Select a file type to limit your search to specific file extensions, or choose *.* to search all file types.

Select Generated Clock Reference Dialog Box

Use this dialog box to find and choose the generated clock reference pin from the list of available pins.

To open the Select Generated Clock Reference dialog box (shown below) from the SmartTime Constraints Editor, open the [Create Generated Clock Constraint Dialog Box](#) dialog box and click the **Browse** button for the **Clock Reference**.

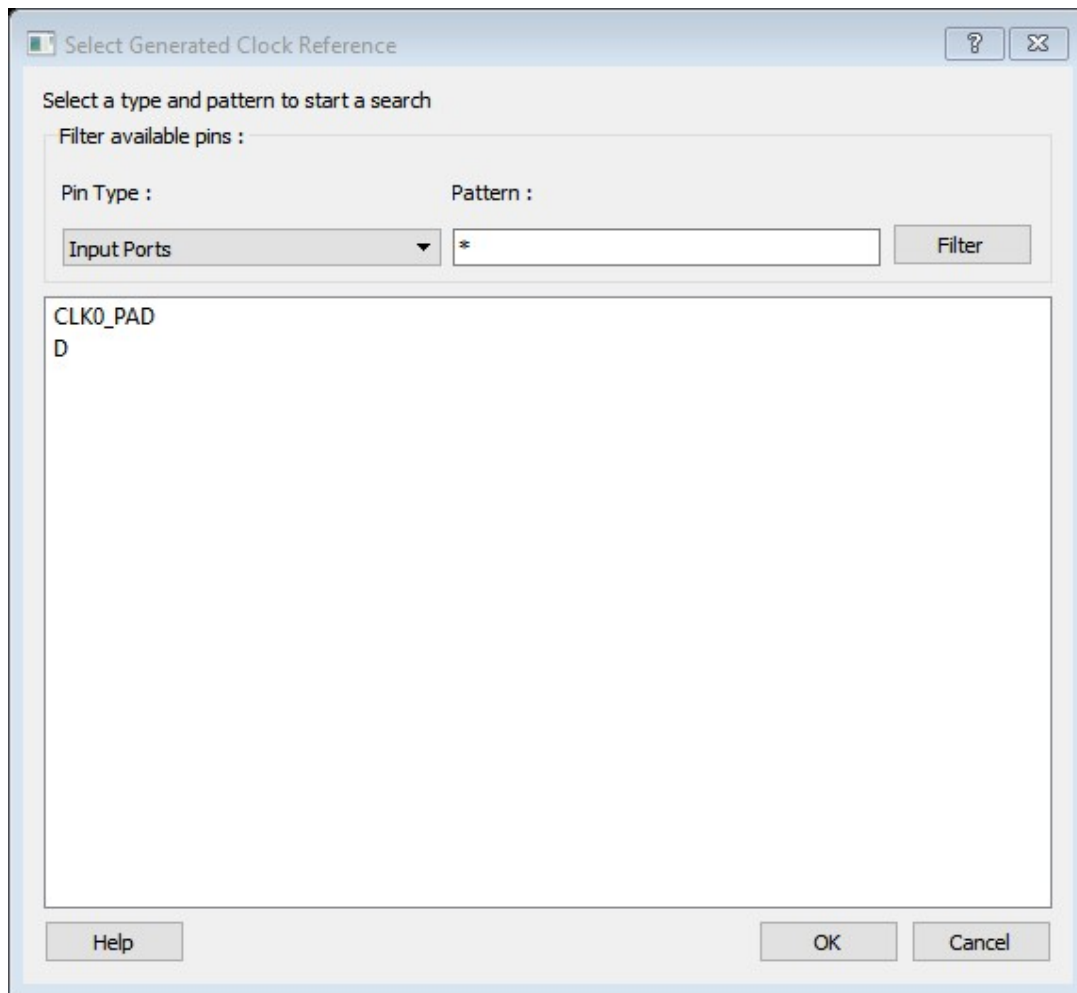


Figure 154 · Select Generated Clock Reference Dialog Box

Filter Available Pins

Pin type – Displays the Available Pin types. The Pin Type options for Generated Clock Reference are:

- Input Ports
- All Pins

Pattern – The default pattern is *, which is a wild-card match for all. You can specify any string value.

Select **Filter** to filter the available pins based on the specified Pin Type and Pattern.

The list box displays the list of available pins based on the filter. Select the pins from the list and click **OK** to select the Generated Clock Reference Pin.

See Also

[Specifying generated clock constraints](#)

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 , open the [Create Generated Clock Constraint](#) dialog box and click the **Browse** button for the **Clock Pin**.

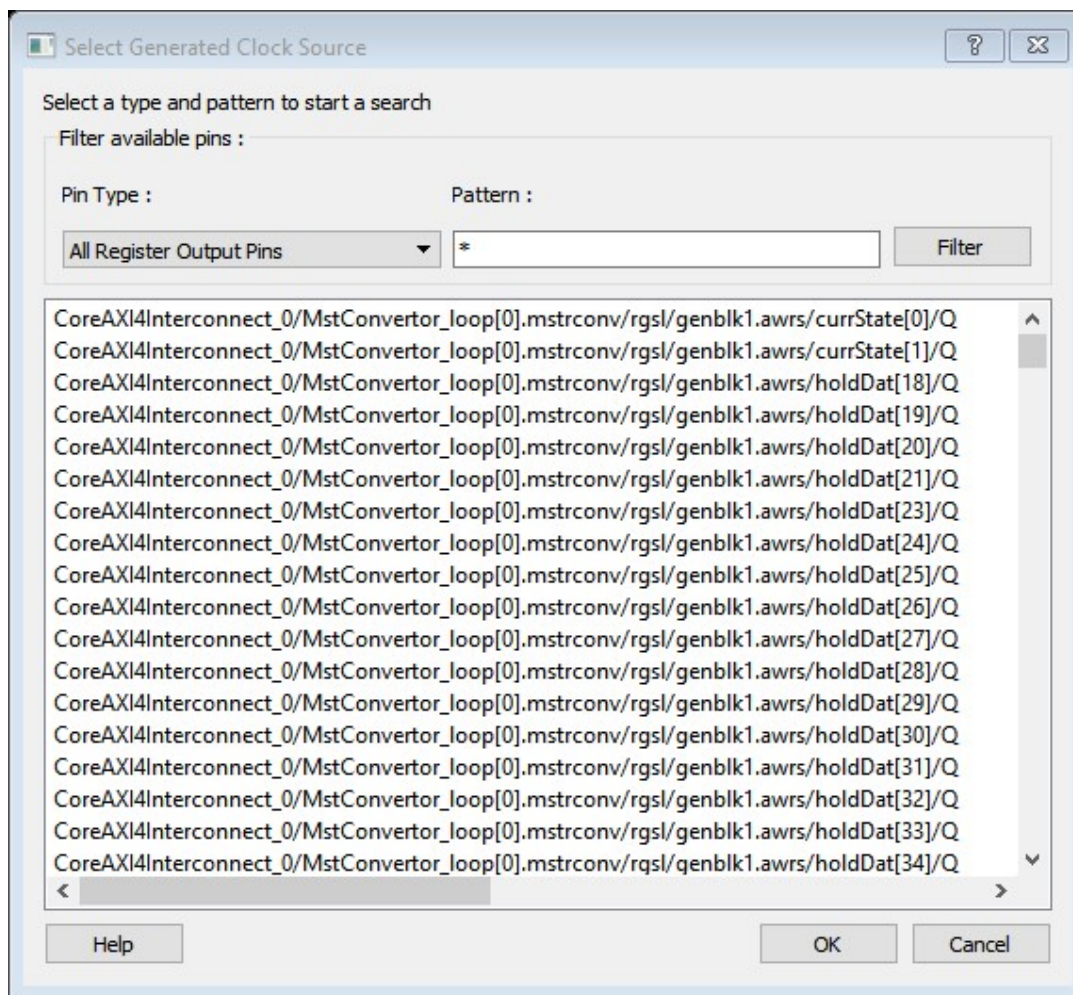


Figure 155 · Select Generated Clock Source Dialog Box

Filter Available Pins

Pin type – Displays the Available Pin types. The Pin Type options for Generated Clock Source are:

- Output Ports
- All Register Output Pins
- All Pins
- All Nets
- Input Ports

Pattern – The default pattern is *, which is a wild-card match for all. You can specify any string value.

Select **Filter** to filter the available pins based on the specified Pin Type and Pattern.

The list box displays the list of available pins based on the filter. Select the pins from the list and click **OK** to select the Generated Clock Source Pin.

Select Source or Destination Pins for Constraint Dialog Box

This dialog box opens when you select the browse button for Source/From, Intermediate/Through and Destination/To pins for Timing Exception Constraints: False Path Constraints, Multicycle Path Constraints, and Maximum/Minimum Delay Constraints.

To open the Select Source or Destination Pins for Constraint dialog box from the Constraints Editor, choose **Constraint > Timing Exception Constraint Name**. Click the browse button to select the source. The following figure shows an example dialog box for **Select Source Pins for Multicycle Constraint**.

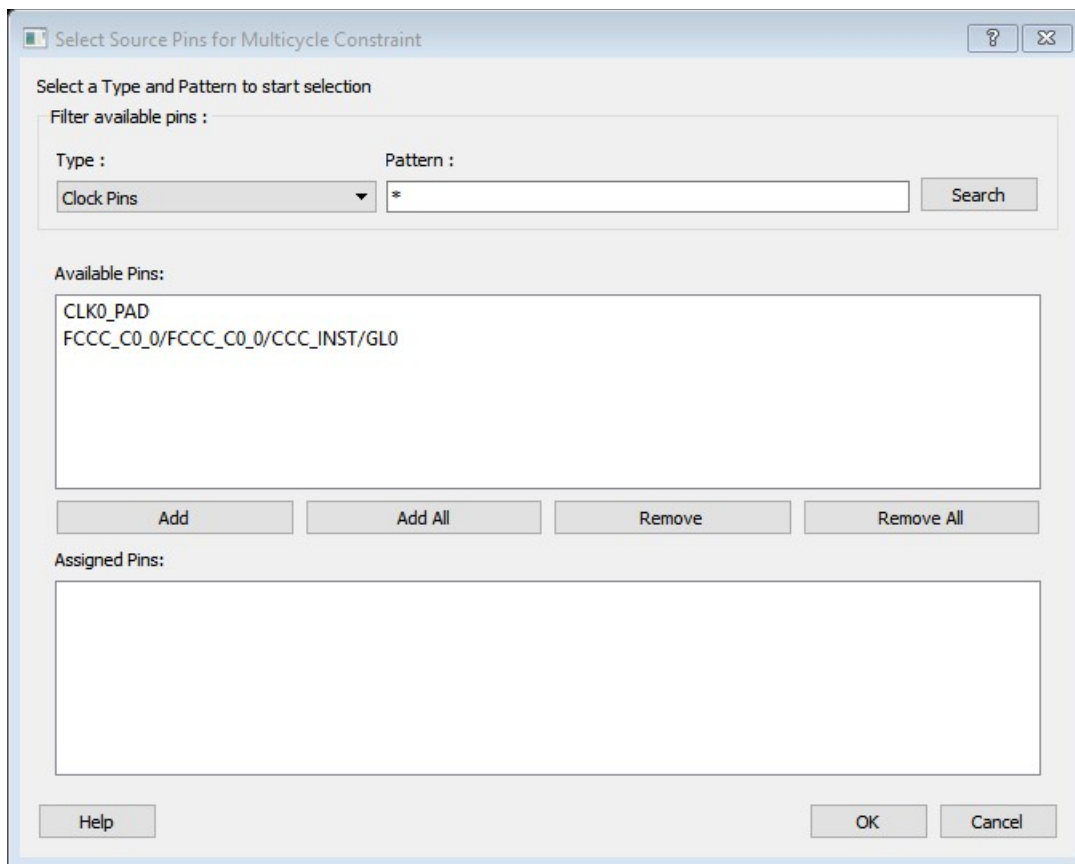


Figure 156 · Select Source Pins for Multicycle Constraint

Filter Available Pins

Type – Displays the Type of the Available Pins in the design. The pin Type options available for the Source are:

- Clock Pins
- Input Ports
- All Register Output Pins

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 Multicycle constraint.

Select Source Pins for Clock Constraint Dialog Box

Use this dialog box to find and choose the clock source from the list of available pins.

To open the Select Source Pins for the Clock Constraint dialog box (shown below) from the SmartTime Constraints Editor, click the **Browse** button to the right of the Clock source field in the [Create Clock Constraint](#) dialog box.

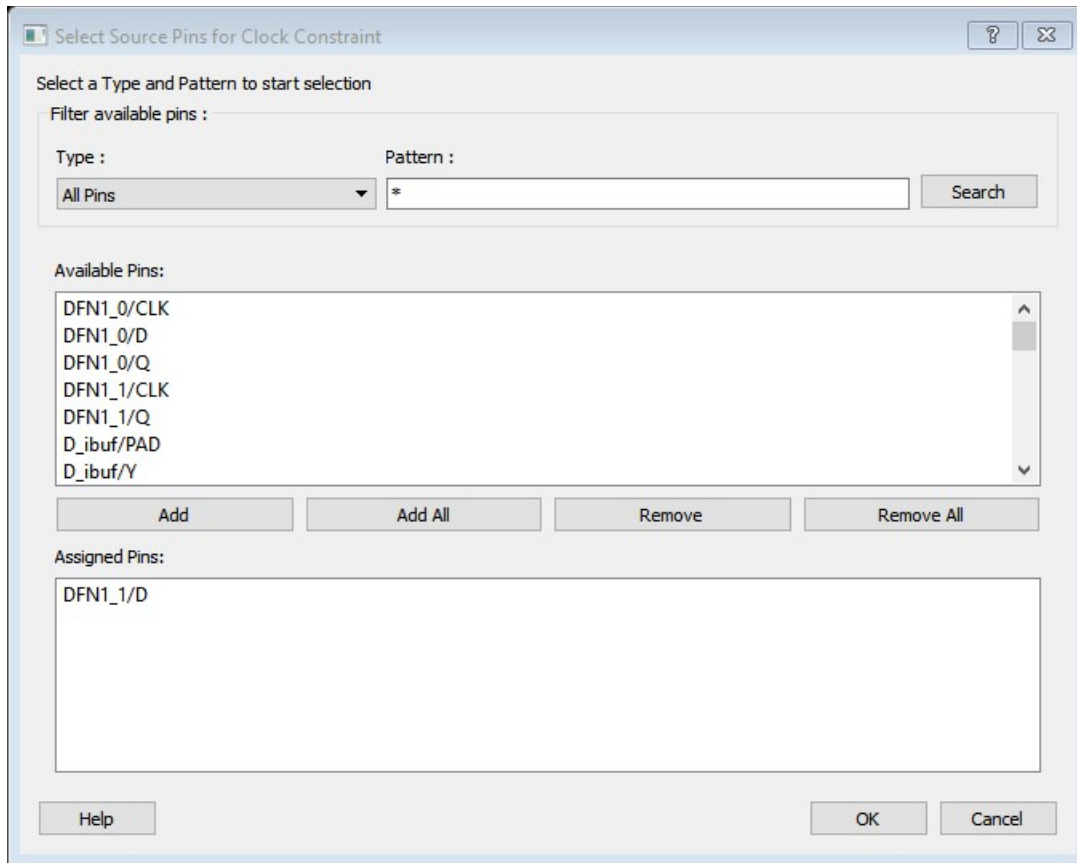


Figure 157 · Select Source Pins for Clock Constraint Dialog Box

Filter Available Pins

Type – Displays the Type of the Available Pins in the design. The Pin Type options available for the Source are:

- All Pins
- Input Ports
- All Nets

Pattern – The default is *, which is a wild-card match for all. You can specify any string value.

Click **Search** to filter the available pins based on the specified pin Type and Pattern.

Available Pins

The list box displays the available pins. If you change the pattern value, the list box shows the available pins based on the filter.

Use **Add**, **Add All** to add the pins from the Available Pins list to Assigned Pins or **Remove**, **Remove All** to delete the pins from the Assigned Pins list.

Assigned Pins

Displays pins selected from the Available Pins list. Select Pins from this list and click **OK** to add the Source Pins for Clock Constraint.


See Also

[Specifying clock constraints](#)

Set a Disable Timing Constraint

Use disable timing constraint to specify the timing arcs to be disabled for timing consideration.

Note: This constraint is for the Place and Route tool and the Verify Timing tool. It is ignored by the Synthesis tool. To specify a Disable Timing constraint, open the Set Constraint to Disable Timing Arcs dialog box in one of the following four ways:

- From the Constraints Browser, choose **Advanced > Disable Timing**.
- Double-click the Add Disable Timing Constraint icon .
- Choose **Disable Timing** from the Constraints drop-down menu (**Constraints > Disable Timing**).
- Right-click any row in the Disable Timing Constraints Table and choose **Add Constraint to Disable Timing**.

The Set Constraint to Disable Timing Arcs dialog box appears.

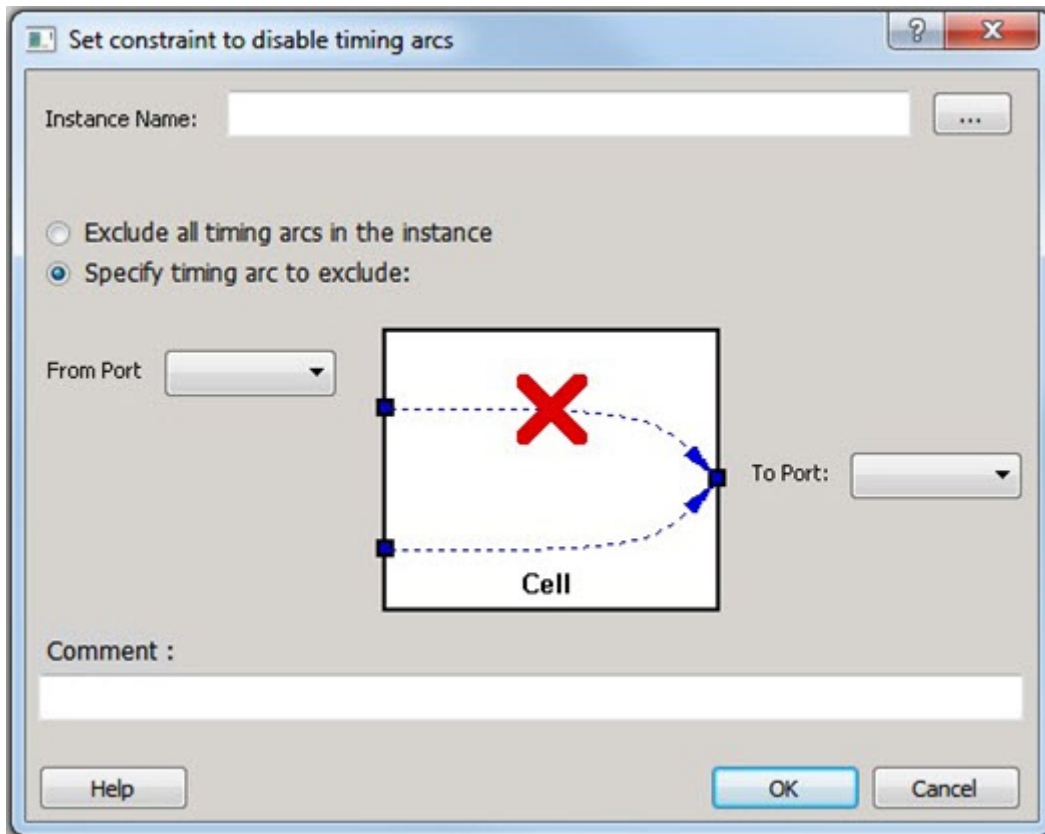


Figure 158 · Set constraint to disable timing arcs Dialog Box

Instance Name

Specifies the instance name for which the disable timing arc constraint will be created.

Click the browse button next to the Instance Name field to open the Select instance to constrain dialog box.

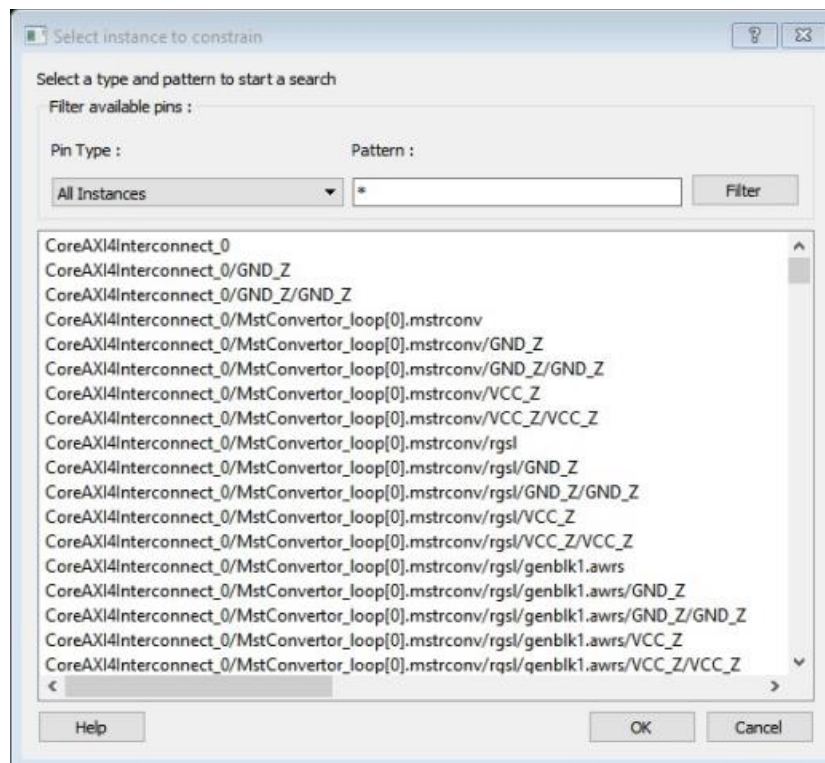


Figure 159 · Select instance to constrain Dialog Box

The Pin Type selection is limited to All Instances only.

Exclude All Timing Arcs in the Instance

This option enables you to exclude all timing arcs in the specified instance.

Specify Timing Arc to Exclude

This option enables you to specify the timing arc to exclude. In this case, you need to specify the from and to ports:

From Port

Specifies the starting point for the timing arc.

To Port

Specifies the ending point for the timing arc.

Comment

Enter a one-line comment for the constraint.

Set Clock Source Latency Dialog Box

Use this dialog box to define the delay between an external clock source and the definition pin of a clock within SmartTime.

To open the Set Clock Source Latency dialog box (shown below) from the Timing Analysis View, you must first [create a clock constraint](#). From the **Constraints** menu, choose **Clock Source Latency**.

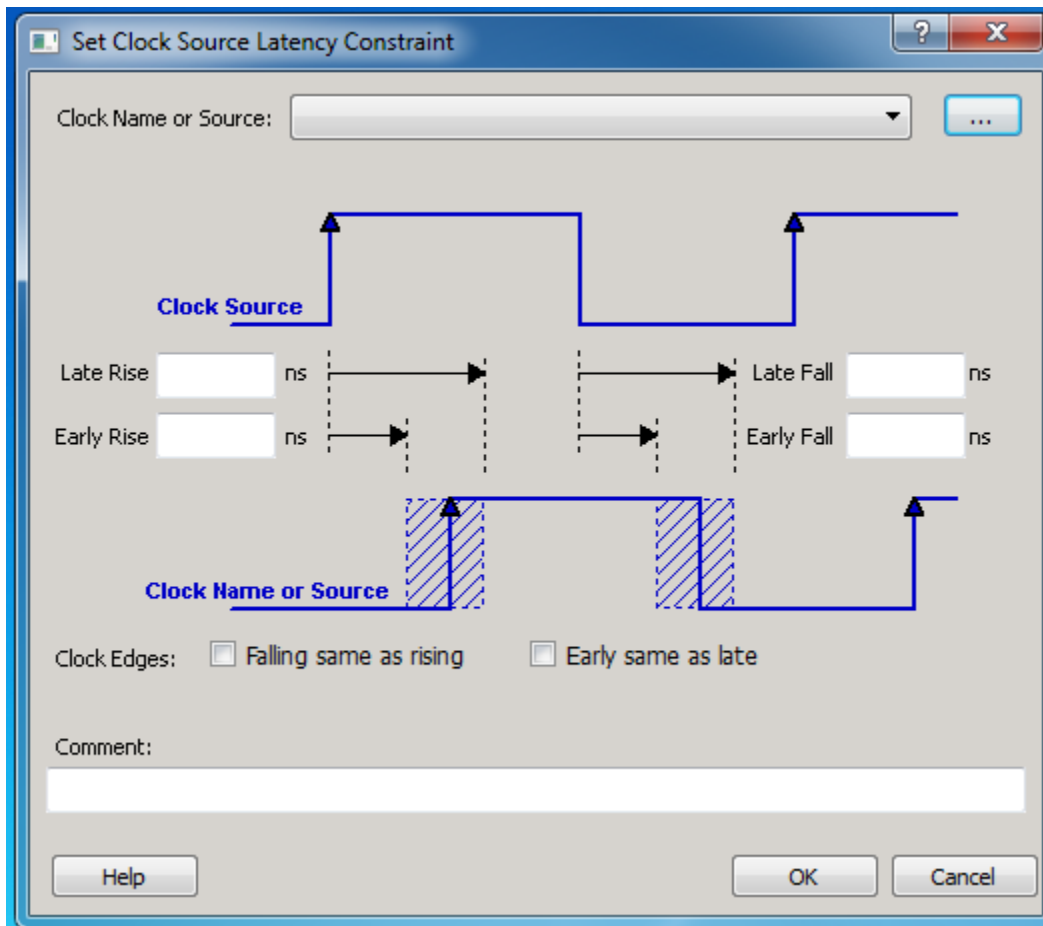


Figure 160 · Set Clock Source Latency Dialog Box

Clock Name or Source

Displays a list of clock ports or pins that do not already have a clock source latency specified. Select the clock name or source for which you are specifying the clock source latency.

Late Rise

Specifies the largest possible latency, in nanoseconds, of the rising clock edge at the clock port or pin selected, with respect to its source. Negative values are acceptable, but may lead to overly optimistic analysis.

Early Rise

Specifies the smallest possible latency, in nanoseconds, of the rising clock edge at the clock port or pin selected, with respect to its source. Negative values are acceptable, but may lead to overly optimistic analysis.

Late Fall

Specifies the largest possible latency, in nanoseconds, of the falling clock edge at the clock port or pin selected, with respect to its source. Negative values are acceptable, but may lead to overly optimistic analysis.

Early Fall

Specifies the smallest possible latency, in nanoseconds, of the falling clock edge at the clock port or pin selected, with respect to its source. Negative values are acceptable, but may lead to overly optimistic analysis.

Clock Edges

Select the latency for the rising and falling edges:

Falling same as rising: Specifies that Rising and Falling clock edges have the same latency.

Early same as late : Specifies that the clock source latency should be considered as a single value, not a range from "early" to "late".

Comment

Enables you to save a single line of text that describes the clock source latency.

See Also

[Specifying Clock Constraints](#)

Set Constraint to Disable Timing Arcs Dialog Box

Use this dialog box to specify the timing arcs being disabled to fix the combinational loops in the design.

To open the Set Constraint to Disable Timing Arcs dialog box (shown below) from the Timing Analysis View, from the **Constraints** menu, choose **Disable Timing**.

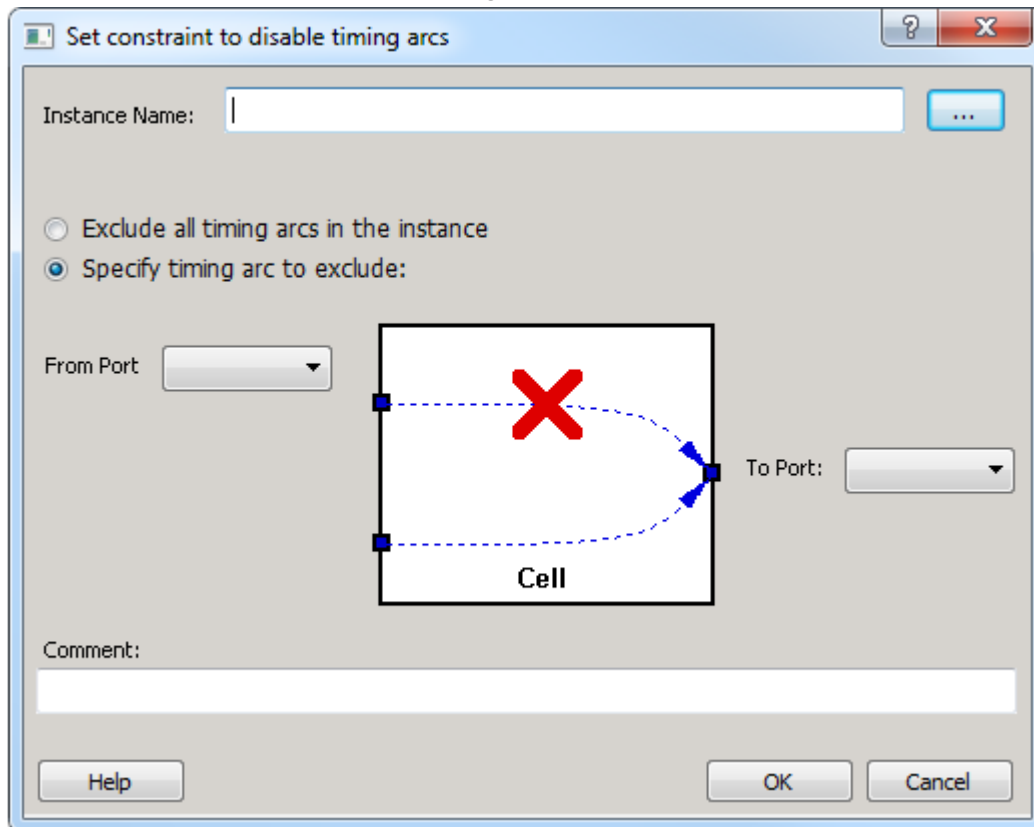


Figure 161 · Set Constraint to Disable Timing Arcs Dialog Box

Instance Name

Specifies the instance name for which the disable timing arc constraint will be created.

Exclude All Timing Arcs in the Instance

This option enables you to exclude all timing arcs in the specified instance.

Specify Timing Arc to Exclude

This option enables you to specify the timing arc to exclude. In this case, you need to specify the from and to ports:

From Port

Specifies the starting point for the timing arc.

To Port

Specifies the ending point for the timing arc.

Comment

Enables you to save a single line of text that describes the disable timing arc.

See Also

[Specifying Disable Timing Constraint](#)

Set False Path Constraint Dialog Box

Use this dialog box to define specific timing paths as being false.

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 path ending points are the register data pins or output ports. This constraint disables setup and hold checking for the specified paths.

Note: The false path information always takes precedence over multiple cycle path information and overrides maximum delay constraints.

To open the Set False Path Constraint dialog box (shown below) from the SmartTime Constraints Editor, choose **Constraints > Exceptions False Path > Add False Path Constraint**.

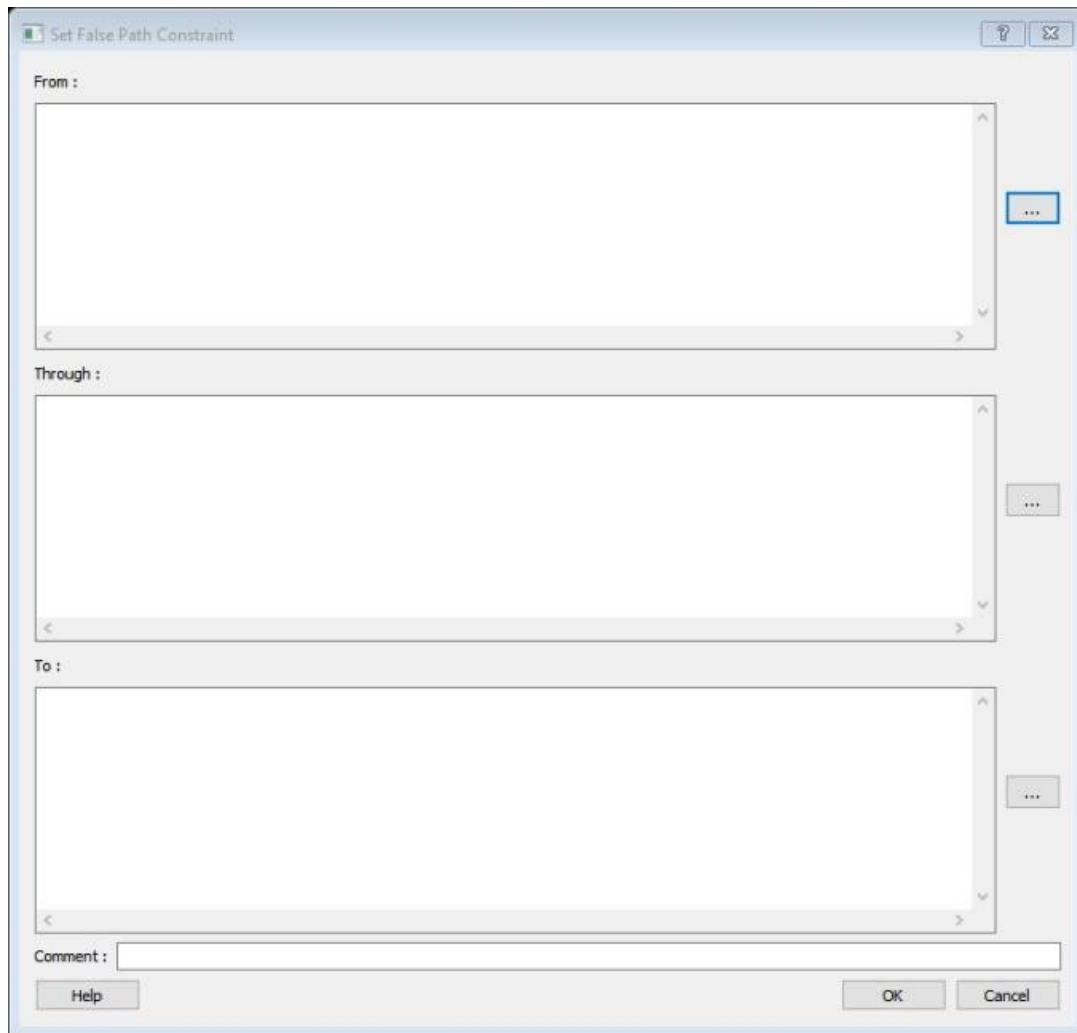


Figure 162 · Set False Path Constraint Dialog Box

From

Specifies the starting points for false path. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

Through

Specifies a list of pins, ports, cells, or nets through which the disabled paths must pass.

To

Specifies the ending points for false path. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

Comment

Enables you to provide comments for this constraint.

Set Maximum Delay Constraint Dialog Box

Use this dialog box to specify the required maximum delay for timing paths in the current design.

SmartTime automatically derives the individual maximum delay targets from clock waveforms and port input or output delays. So 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 multiple cycle path constraint.

To open the Set Maximum Delay Constraint dialog box (shown below) from the Constraints Editor, click the **Constraints** menu and choose **Max Delay** (**Constraints > Max Delay**).

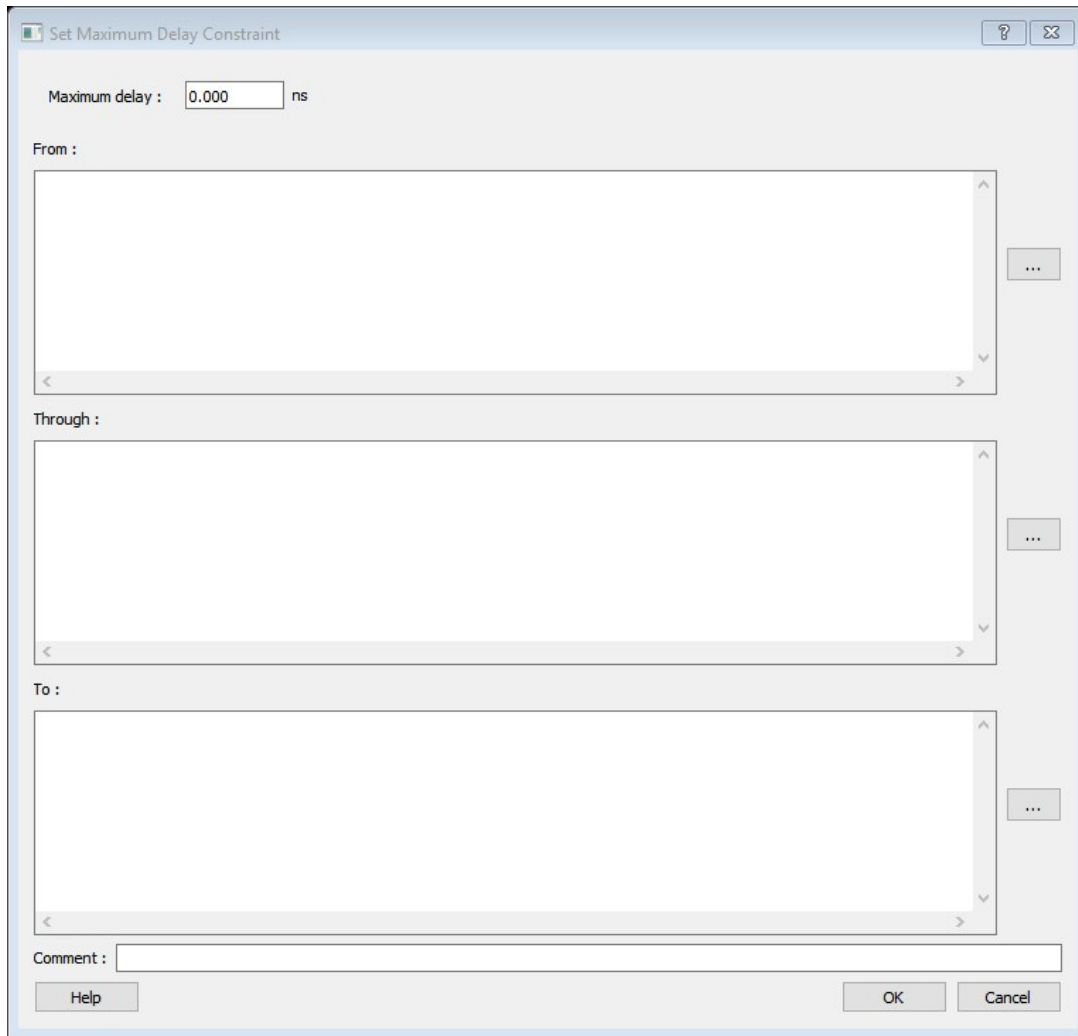


Figure 163 · Set Maximum Delay Constraint Dialog Box

Maximum Delay

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, SmartTime includes clock skew in the computed delay.

If the path starting point has an input delay specified, SmartTime adds that delay value to the path delay.

If the path ending point is on a sequential device, SmartTime includes clock skew and library setup time in the computed delay.

If the ending point has an output delay specified, SmartTime adds that delay to the path delay.

From

Specifies the starting points for max delay constraint. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

Through

Specifies the through points for the multiple cycle constraint.

To

Specifies the ending points for maximum delay constraint. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

Comment

Enables you to provide comments for this constraint.

See Also

[Specifying a Maximum Delay Constraint](#)

Set Minimum Delay Constraint Dialog Box

Use this dialog box to specify the required minimum delay for timing paths in the current design.

SmartTime automatically derives the individual minimum delay targets from clock waveforms and port input or output delays. So 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 multiple cycle path constraint.

To open the Set Minimum Delay Constraint dialog box (shown below) from the Constraints Editor, click the **Constraints** menu and choose **Min Delay (Constraints > Min Delay)**.

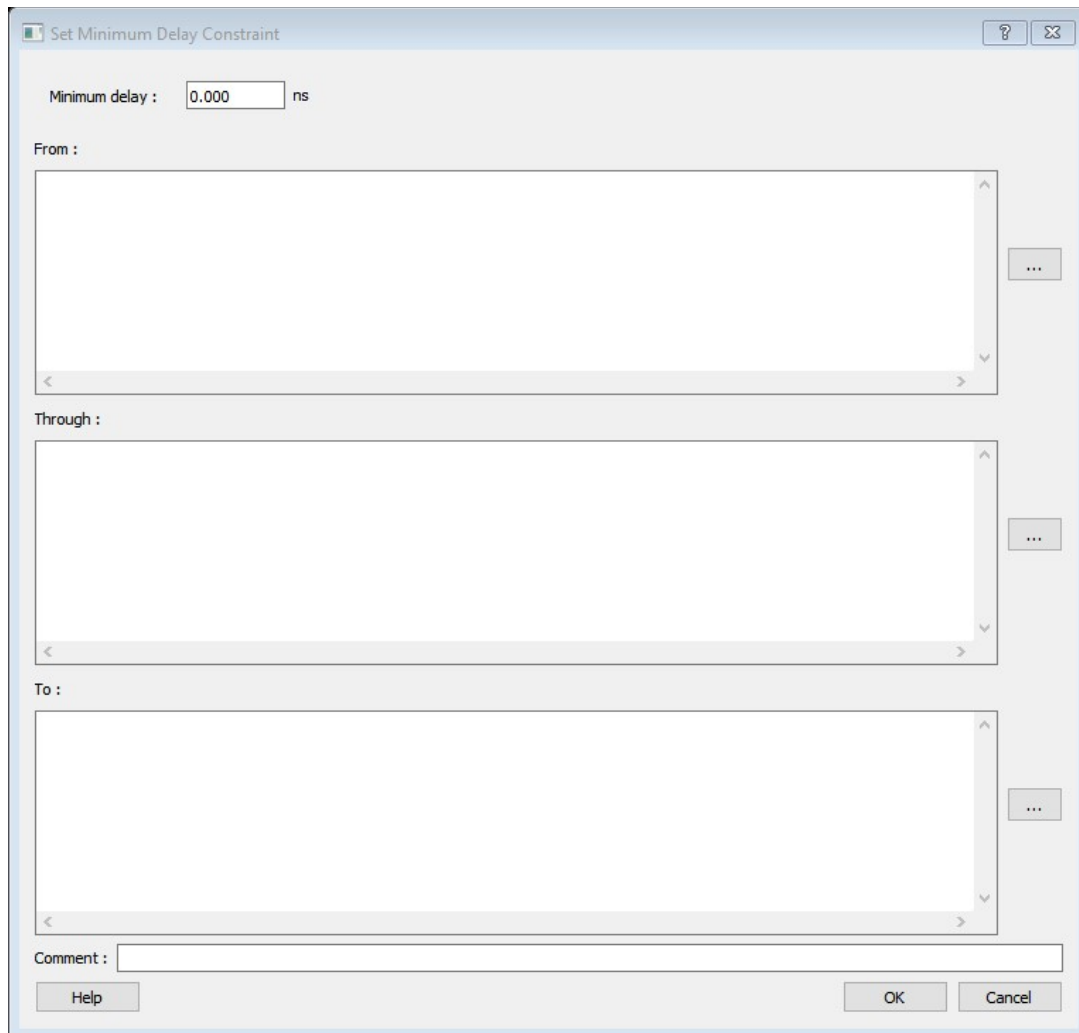


Figure 164 · Set Minimum Delay Constraint Dialog Box

Minimum Delay

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, SmartTime includes clock skew in the computed delay.

If the path starting point has an input delay specified, SmartTime adds that delay value to the path delay.

If the path ending point is on a sequential device, SmartTime includes clock skew and library setup time in the computed delay.

If the ending point has an output delay specified, SmartTime adds that delay to the path delay.

From

Specifies the starting points for minimum delay constraint. A valid timing starting point is a clock, a primary input, an input port, or a clock pin of a sequential cell.

Through

Specifies the through points for the multiple cycle constraint.

To

Specifies the ending points for minimum delay constraint. A valid timing ending point is a clock, a primary output, an input port, or a data pin of a sequential cell.

Comment

Enables you to provide comments for this constraint.

See Also

[Specifying a Minimum Delay Constraint](#)

Set Multicycle Constraint Dialog Box

Use this dialog box to specify the paths that take multiple clock cycles in the current design.

Setting the 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.

Note: When multiple timing constraints are set on the same timing path, the false path constraint has the highest priority and always takes precedence over multiple cycle path constraint. A specific maximum delay constraint overrides a general multicycle path constraint.

To open the Set Multicycle Constraint dialog box (shown below) from the Constraints Editor, choose **Constraints > Multicycle**.

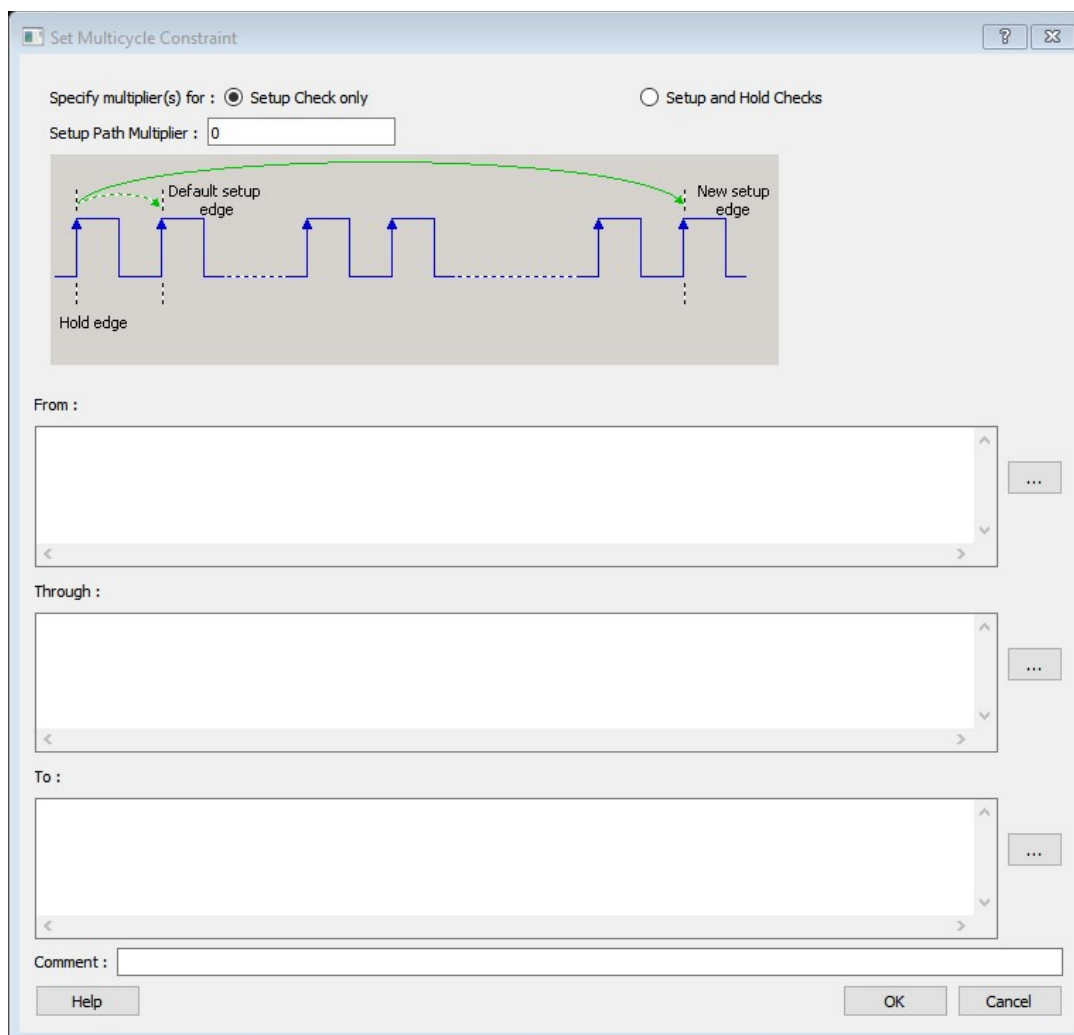


Figure 165 · Set Multicycle Constraint Dialog Box

Setup Path Multiplier

Specifies an integer value that represents a number of cycles the data path must have for a setup check. No hold check will be performed.

From

Specifies the starting points for the multiple cycle constraint. A valid timing starting point is a clock, a primary input, an inout port, or the clock pin of a sequential cell.

Through

Specifies the through points for the multiple cycle constraint.

To

Specifies the ending points for the multiple cycle constraint. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

Comment

Enables you to provide comments for this constraint.

When you select the Setup and Hold Checks option, an additional field appears in this dialog box: **Hold Path Multiplier**.

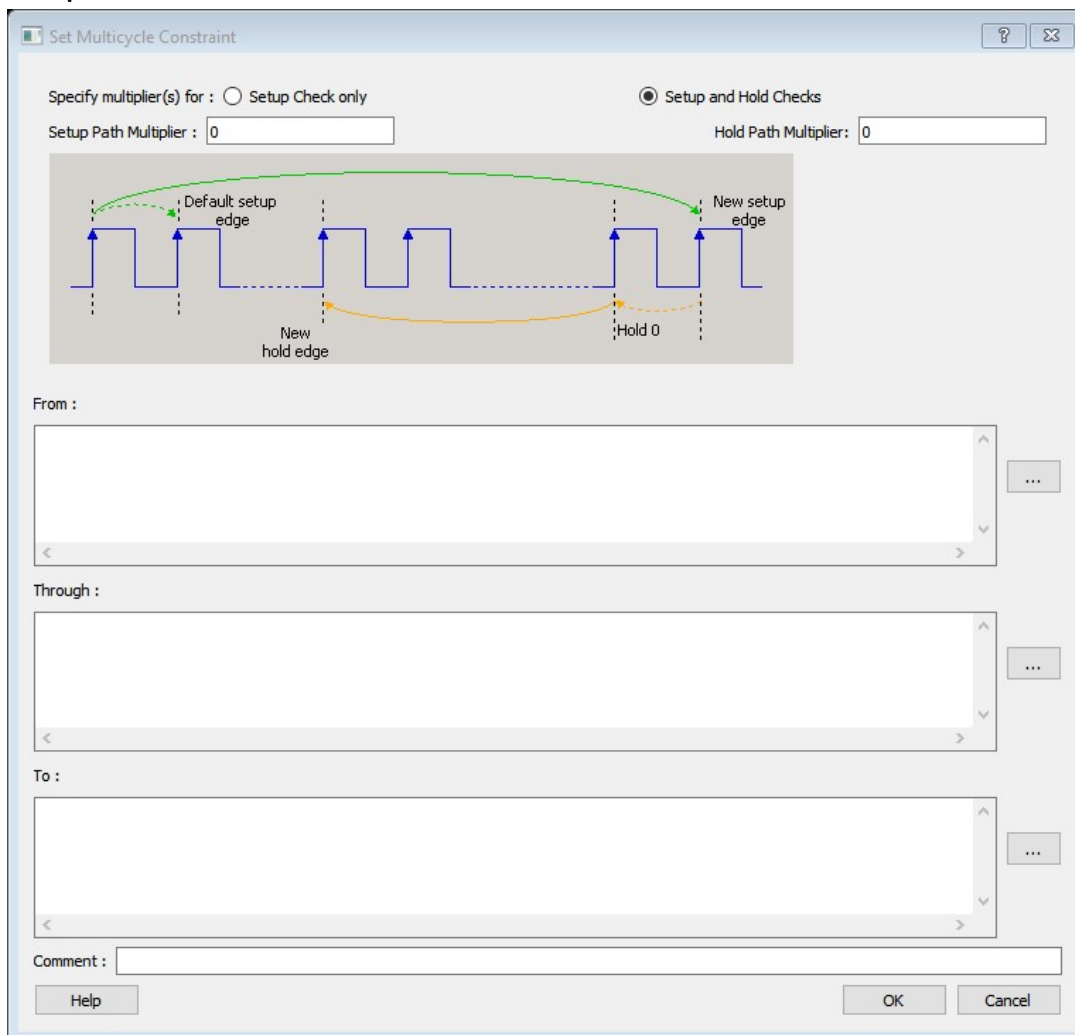


Figure 166 · Set Multicycle Constraint Dialog Box with Setup and Hold Checks Selected

Hold Path Multiplier

Specifies an integer value that represents a number of cycles the data path must have for a hold check, starting from one cycle before the setup check edge.

See Also

[Specifying a Multicycle Constraint](#)

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.

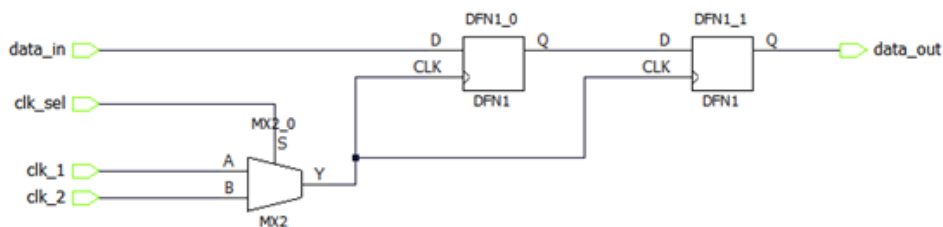
Note: The exclusive flags for the arguments above are all mutually exclusive. Only one can be specified.

-group *clock_list*

Specifies a list of clocks. There can any number of groups specified in the set_clock_groups command.

Examples

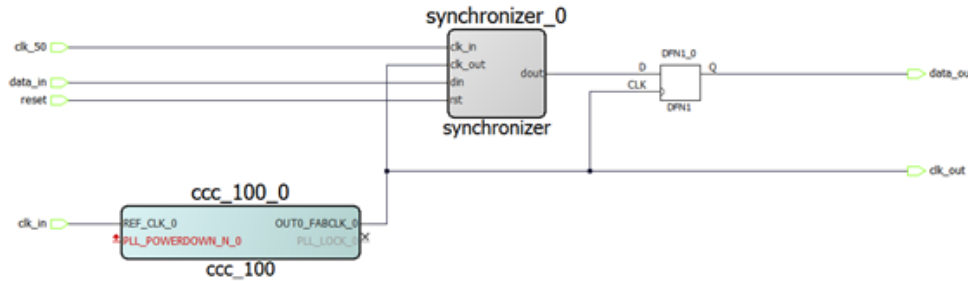
1) Here, there are two muxed clocks in a clock group.



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.



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 } ] \
[ get_ports clk_out ]
set_clock_groups -asynchronous -group { clk_in ccc_100 clk_out } -group clk_50
```

See Also

[list_clock_groups](#)

[remove_clock_groups](#)

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 ({}).

Supported Families

SmartFusion2

IGLOO2

RTG4

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]
```

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.

Supported Families

SmartFusion2

IGLOO2

RTG4

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:

```
set_clock_uncertainty 2 -setup [get_clocks clk]
set_clock_uncertainty 2 [get_clocks clk]
```

Clock to Clock Uncertainty constraint examples:

```
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 {*} ]
```

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 (`{}`).

Supported Families

SmartFusion2, IGLOO2, RTG4

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_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.

Supported Families

SmartFusion2, IGLOO2, RTG4

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 multicyle 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](#)

Organize Source Files Dialog Box – Synthesis

The Organize Source Files dialog box enables you to set the source file order in the Libero SoC.

Click the **Use list of files organized by User** radio button to Add/Remove source files for the selected tool.

To specify the file order:

1. In the Design Flow window under Implement Design, right-click **Synthesize** and choose **Organize Input Files > Organize Source Files**. The Organize Source Files dialog box appears.
2. Click the **Use list of files organized by User** radio button to Add/Remove source files for the selected tool.
3. Select a file and click the Add or Remove buttons as necessary. Use the Up and Down arrows to change the order of the Associated Source files.
4. Click **OK**.

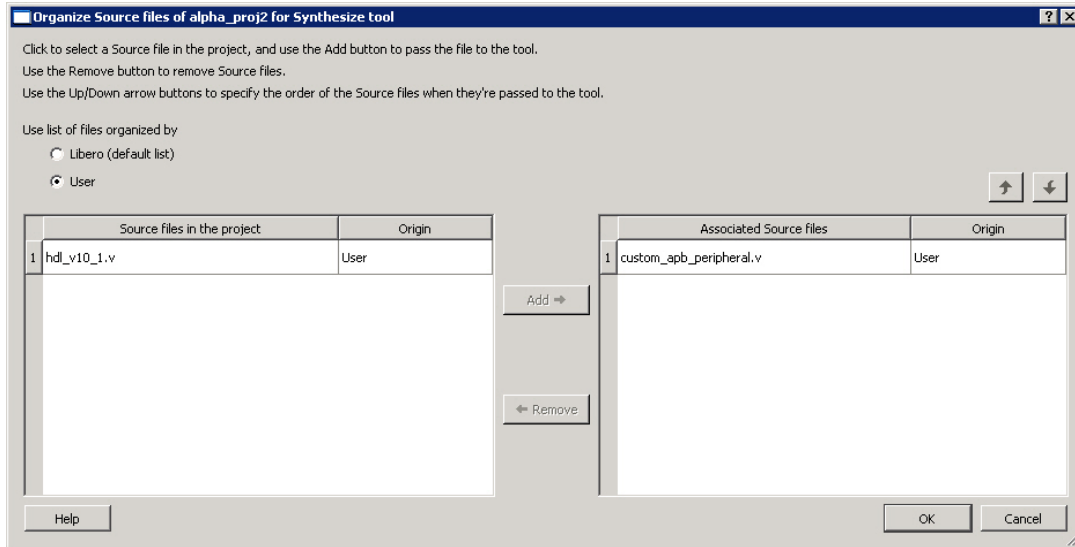


Figure 167 · Organize Source Files Dialog Box

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 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

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.

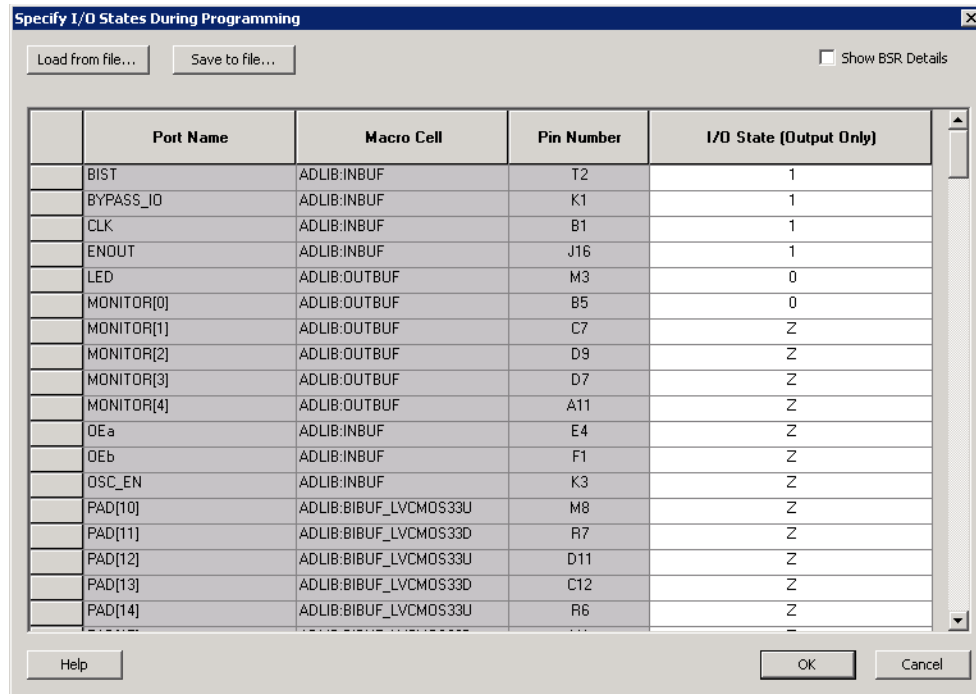



Figure 168 · I/O States During Programming Dialog Box

Specifying a False Path Constraint

You set options in the [Set False Path Constraint](#) dialog box to define specific timing paths as false.

To specify False Path constraints:

1. Add the constraint in the [Editable Constraints Grid](#) or open the [Set False Path Constraint](#) dialog box. You can do this by using one of the following methods:
 - From the **Constraints** drop-down menu, choose **False Path**.
 - Click the  icon.
 - From the Constraints Browser, choose **False Path**.
 - Right-click **False Path** in the Constraint Browser and choose **Add False Path Constraint**.

The Set False Path Constraint dialog box appears (as shown below).

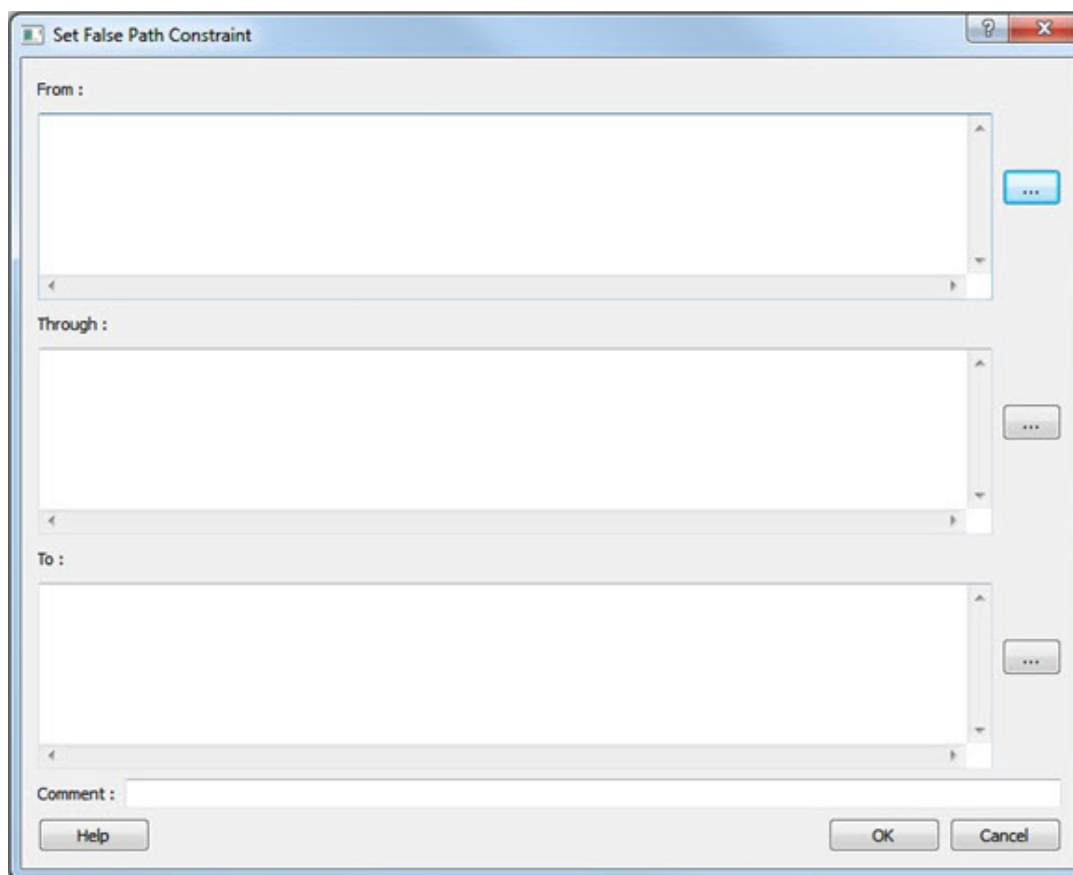


Figure 169 · Set False Path Constraint Dialog Box

2. Specify the **From** pin(s). Click the **Browse** button next to **From** to open the Select Source Pins for False Path Constraint dialog box (as shown below).

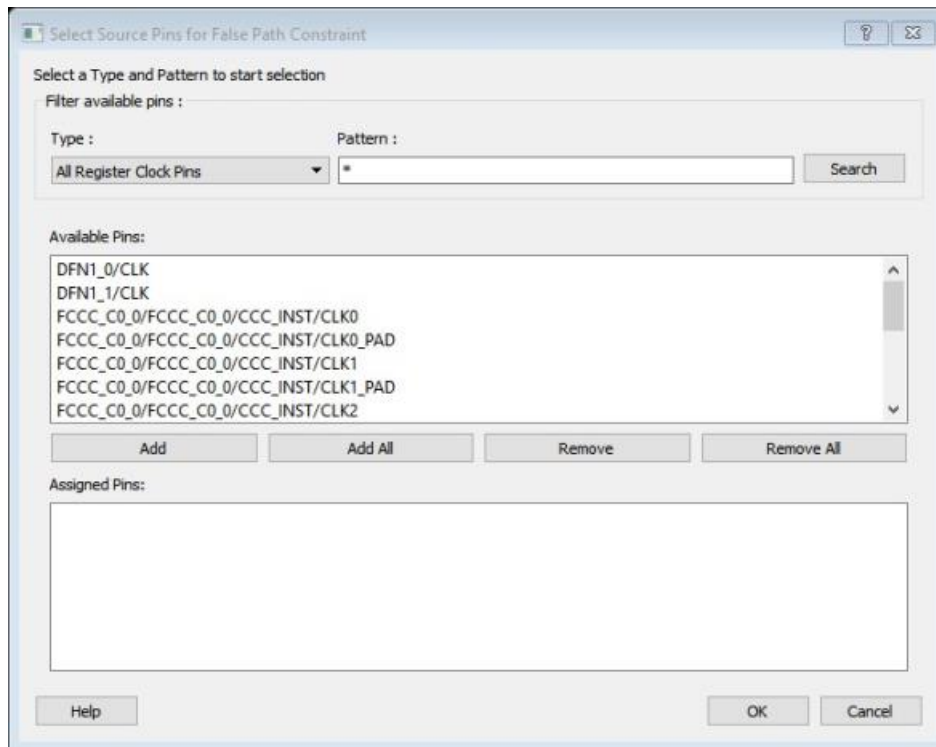


Figure 170 · Select Source Pins for False Path Constraint Dialog Box


3. Use **Filter available pins** to narrow the pin list based on the selected **Type** and **Pattern**. Select the pin(s) from the **Available Pins** list. You can select multiple pins in this window.
4. Click **Add** or **Add All** to add the pins from the **Available Pins** list to the **Assigned Pins** list. Click **Remove** or **Remove All** to remove the pins from the **Assigned Pins** list.
5. Select the pins from the **Assigned Pins** list and click **OK**. The **Set False Path Constraint** dialog box displays the updated **From** pin(s) list.
6. Click the **Browse** button for **Through** and **To** and add the appropriate pin(s). The displayed list shows the pins reachable from the previously selected pin(s) list.
7. Enter comments in the **Comment** section.
8. Click **OK**.

The False Path constraints to the Constraints List in the Timing Constraints Editor.

Specifying a Maximum Delay Constraint

You set options in the [Set Maximum Delay Constraint](#) dialog box to relax or to tighten the original clock constraint requirement on specific paths.

To specify Max delay constraints:

1. Add the constraint in the [Editable Constraints Grid](#) or open the [Set Maximum Delay Constraint](#) dialog box using one of the following methods:
 - Click the  icon in the Constraints Editor.
 - From the Constraints Browser, choose **Max Delay**.
 - Choose **Max Delay** from the Constraints drop-down menu (**Constraints > Max Delay**).
 - From the Max Delay Constraints Table, right-click any row and choose **Add Maximum Delay Constraint**.

The Set Maximum Delay Constraint dialog box appears (as shown below).

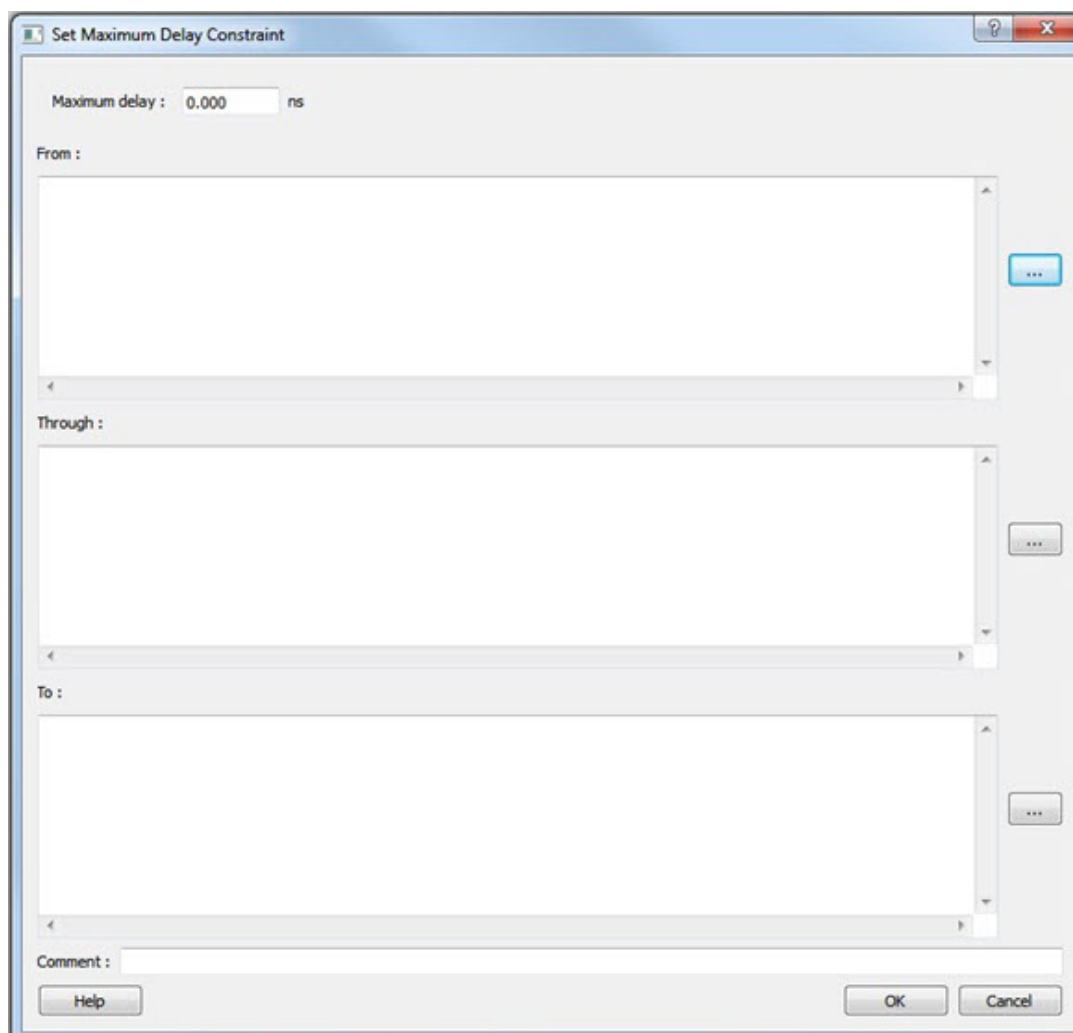


Figure 171 · Set Maximum Delay Constraint Dialog Box

2. Specify the delay in the **Maximum delay** field.
3. Specify the **From** pin(s). Click the **Browse** button next to **From** to open the Select Source Pins for Max Delay Constraint dialog box (as shown below).

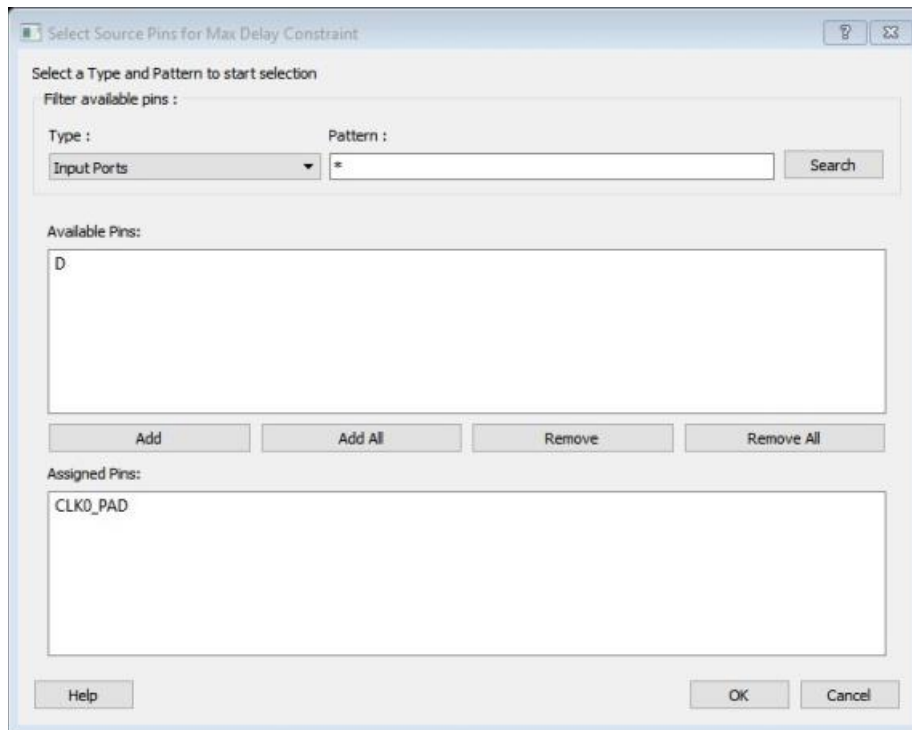


Figure 172 · Select Source Pins for Max Delay Constraint Dialog Box

4. Use **Filter available pins** to narrow the pin list based on the selected **Type** and **Pattern**. Select the pin(s) from the **Available Pins** list. You can select multiple pins in this window.
5. Click **Add** or **Add All** to add the pins from the **Available Pins** list to the **Assigned Pins** list. Click **Remove** or **Remove All** to remove the pins from the **Assigned Pins** list.
6. Select the pins from the **Assigned Pins** list and click **OK**. The **Set Maximum Delay Constraint** dialog box displays the updated **From** pin(s) list.
7. Click the **Browse** button for **Through** and **To** and add the appropriate pin(s). The displayed list shows the pins reachable from the previously selected pin(s) list.
8. Enter comments in the **Comment** section.
9. Click **OK**.

SmartTime adds the maximum delay constraints to the Constraints List in the SmartTime Constraints Editor.


See Also

[Timing Exceptions Overview](#)

Specifying a Minimum Delay Constraint

You set options in the [Set Minimum Delay Constraint](#) dialog box to relax or to tighten the original clock constraint requirement on specific paths.

To specify Min delay constraints:

1. Open the [Set Minimum Delay Constraint](#) dialog box using one of the following methods:
 - Click the  icon in the Constraints Editor.
 - From the Constraints Browser, choose **Min Delay**.
 - Choose **Min Delay** from the Constraints drop-down menu (**Constraints > Min Delay**).
 - Right click on any row in the Min Delay Constraints Table and select **Add Minimum Delay Constraint**.

The Set Minimum Delay Constraint dialog box appears (as shown below).

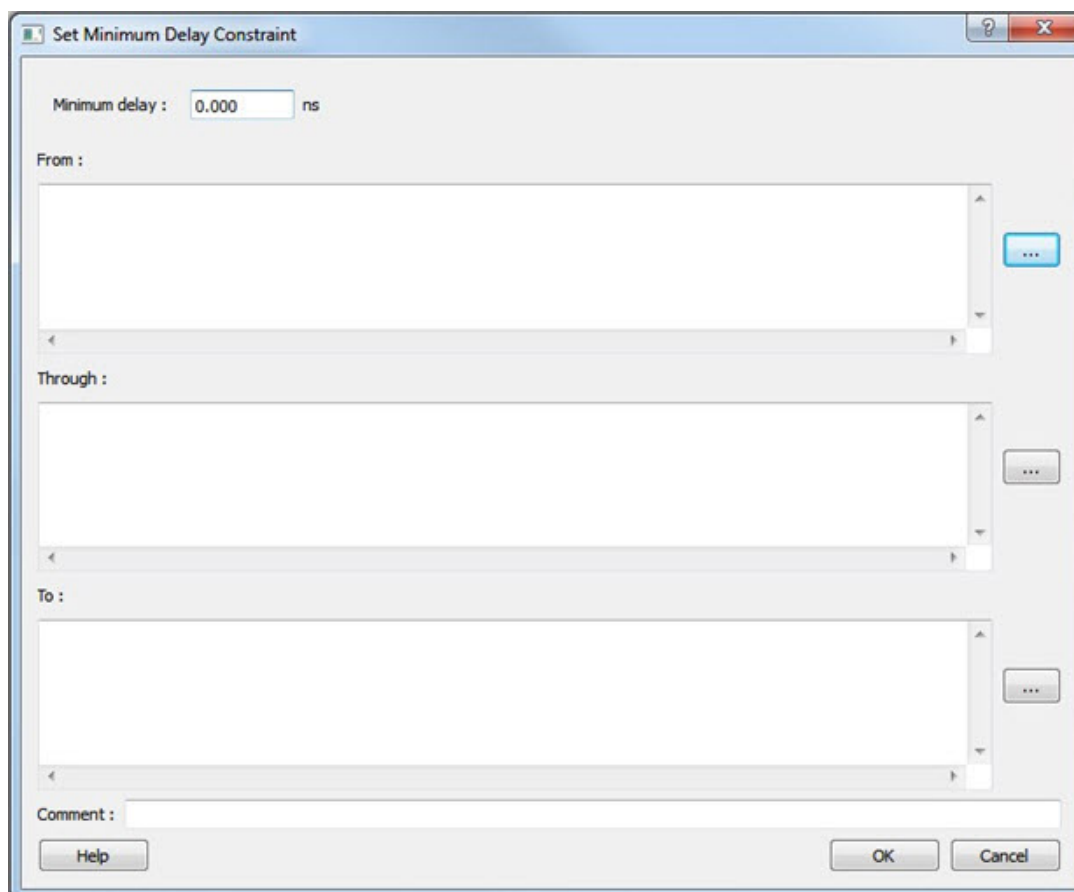


Figure 173 · Set Minimum Delay Constraint Dialog Box

2. Specify the delay in the **Minimum delay** field.
3. Specify the **From** pin(s). Click the **Browse** button next to **From** to open the Select Source Pins for Min Delay Constraint dialog box (as shown below).

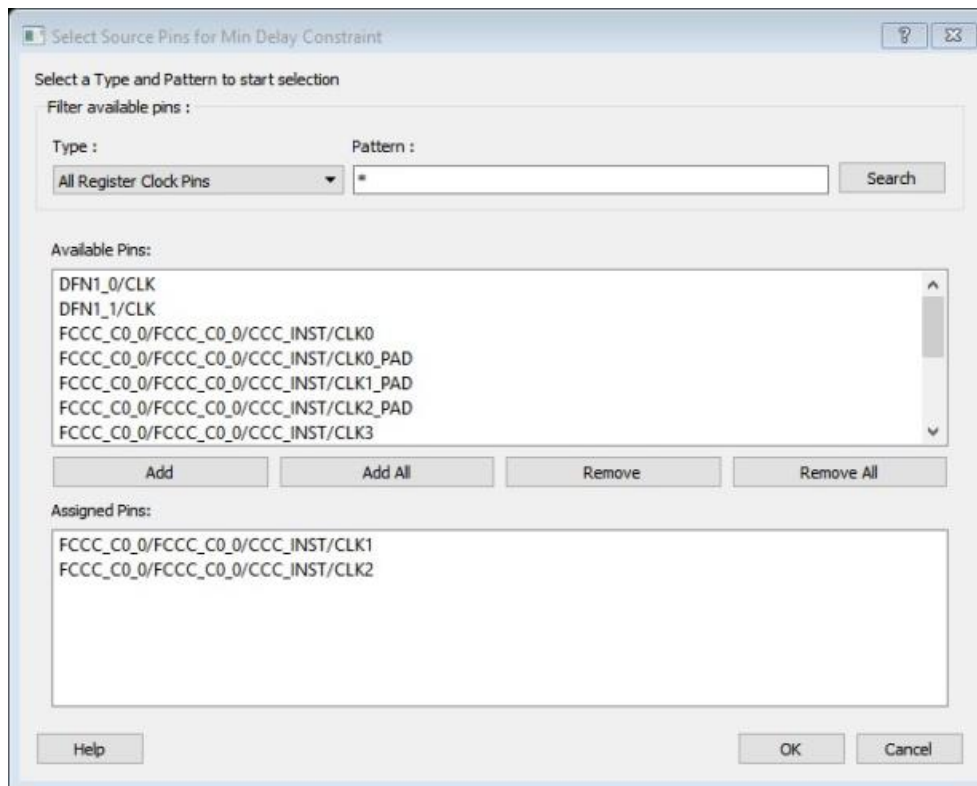


Figure 174 · Select Source Pins for Min Delay Constraint Dialog Box

4. Use **Filter available pins** to narrow the pins list based on the selected **Type** and **Pattern**. Select the pin(s) from the **Available Pins** list. You can select multiple pins in this window.
5. Click **Add** or **Add All** to add the pins from the **Available Pins** list to the **Assigned Pins** list. Click **Remove** or **Remove All** to remove the pins from the **Assigned Pins** list.
6. Select the pins from the **Assigned Pins** list and click **OK**. The **Set Minimum Delay Constraint** dialog box displays the updated **From** pin(s) list.
7. Click the **Browse** button for **Through** and **To** and add the appropriate pin(s). The displayed list shows the pins reachable from the previously selected pin(s) list.
8. Enter comments in the **Comment** section.
9. Click **OK**.

The minimum delay constraints are added to the Constraints List.


See Also

[Timing Exceptions Overview](#)

Specifying a Multicycle Constraint

You set options in the [Set Multicycle Constraint](#) dialog box to specify paths that take multiple clock cycles in the current design.

To specify multicycle constraints:

1. Add the constraint in the [Editable Constraints Grid](#) or open the [Set Multicycle Constraint](#) dialog box using one of the following methods:
 - From the Timing Constraints Editor, choose **Constraint > MultiCycle**.
 - Click the  icon.
 - From the Constraints Browser, choose **Multicycle**.

- Right-click the **Multicycle** option in the Constraint Browser and select **Add Multicycle Path Constraint**.

The Set Multicycle Constraint dialog box appears (as shown below).

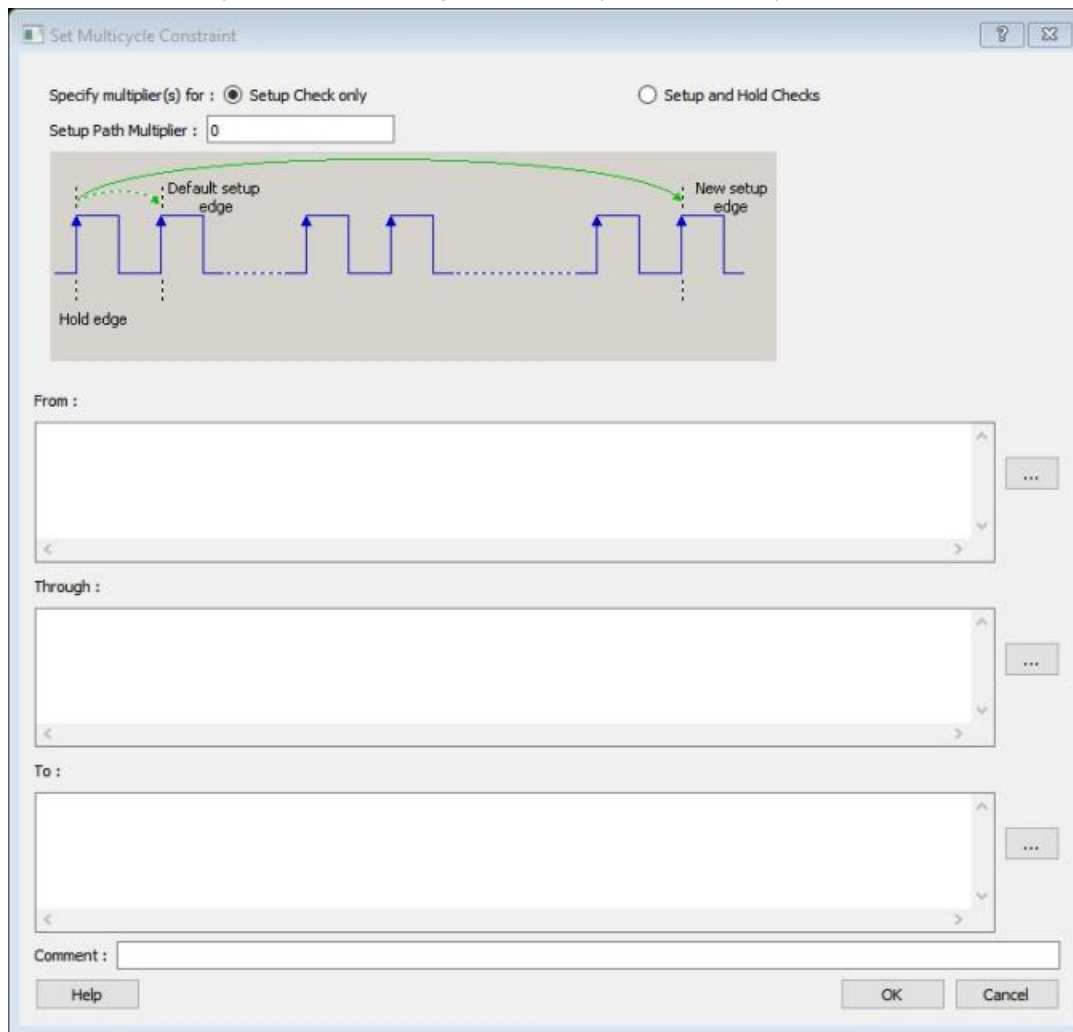
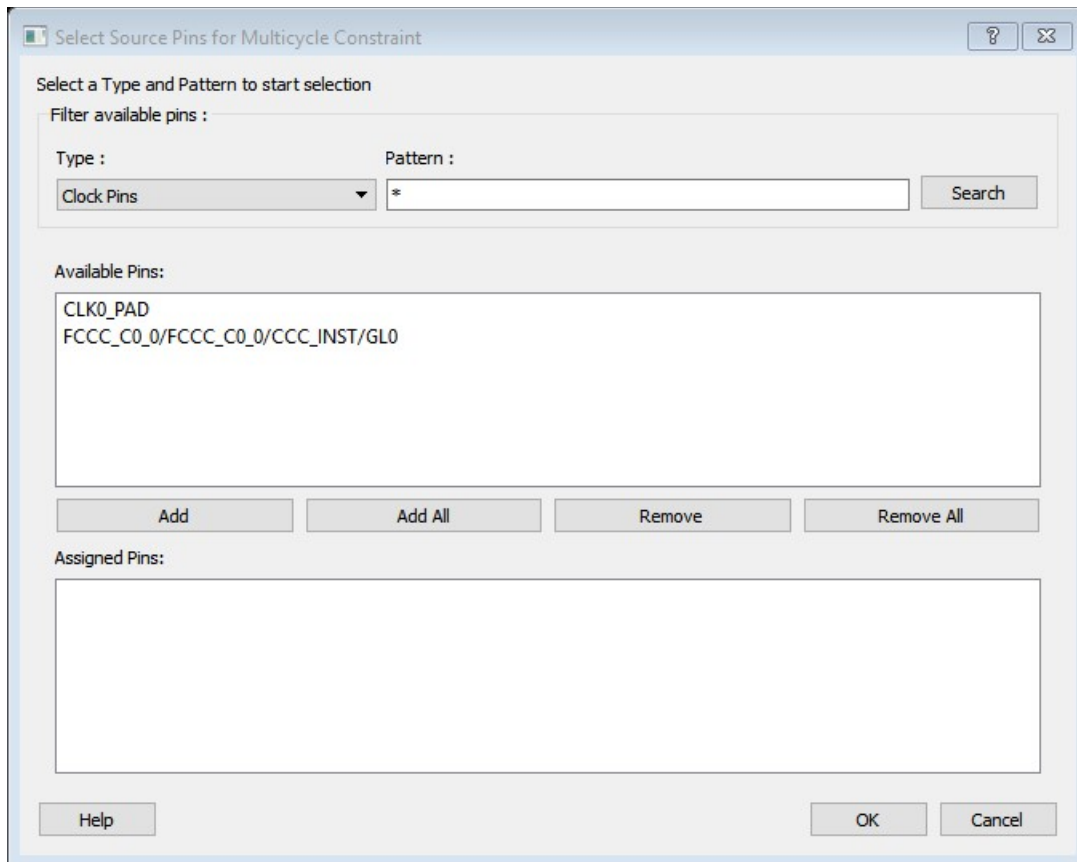


Figure 175 · Set Multicycle Constraint Dialog Box

2. Specify the number of cycles in the **Setup Path Multiplier**.
3. Specify the **From** pin(s). Click the **Browse** button next to **From** to open the Select Source Pins for Multicycle Constraint dialog box (as shown below).



Select Source Pins for Multicycle Constraint Dialog Box

4. Use **Filter available pins** to narrow the pin list based on the selected **Type** and **Pattern** . Select the pin(s) from the **Available Pins** list. You can select multiple pins in this window.
5. Click **Add** or **Add All** to add the pins from the **Available Pins** list to the **Assigned Pins** list. Click **Remove** or **Remove All** to remove the pins from the **Assigned Pins** list.
6. Select the pins from the **Assigned Pins** list and click **OK**. The **Set Multicycle Constraint** dialog box displays the updated **From** pin(s) list.
7. Click the browse button for **Through** and **To** and add the appropriate pins. The displayed list shows the pins reachable from the previously selected pin(s) list
8. Enter comments in the **Comment** section.
9. Click **OK**. The Timing Constraints Editor adds the multicycle constraints to the Constraints List.


See Also

[Set Multicycle Constraint Dialog Box](#)

Specifying Disable Timing Constraint

Use disable timing constraint to specify the timing arcs being disabled.

To specify the disable timing constraint:

1. Add the constraint in the [Editable Constraints Grid](#) or open the [Set Constraint to Disable Timing Arcs Dialog Box](#) using one of the following methods:
 - From the Timing Constraints Editor, choose **Constraints Menu > Disable Timing**.
 - Click the  icon in the Constraints Editor.
 - In the Constraints Editor, right-click **Disable Timing** and choose **Add Disable Timing Constraints**.

2. Select an instance from your design.
3. Select whether you want to exclude all timing arcs in the instance or if you want to specify the timing arc to exclude. If you selected specify timing arc to exclude, select a from and to port for the timing arc.
4. Enter any comments to be attached to the constraint.
5. Click **OK**. The new constraint appears in the constraints list.

Note: Note: When you choose Save from the File menu, the newly created constraint is saved in the database.

See Also


[Set Constraint to Disable Timing Arcs Dialog Box](#)

Specifying Clock Constraints

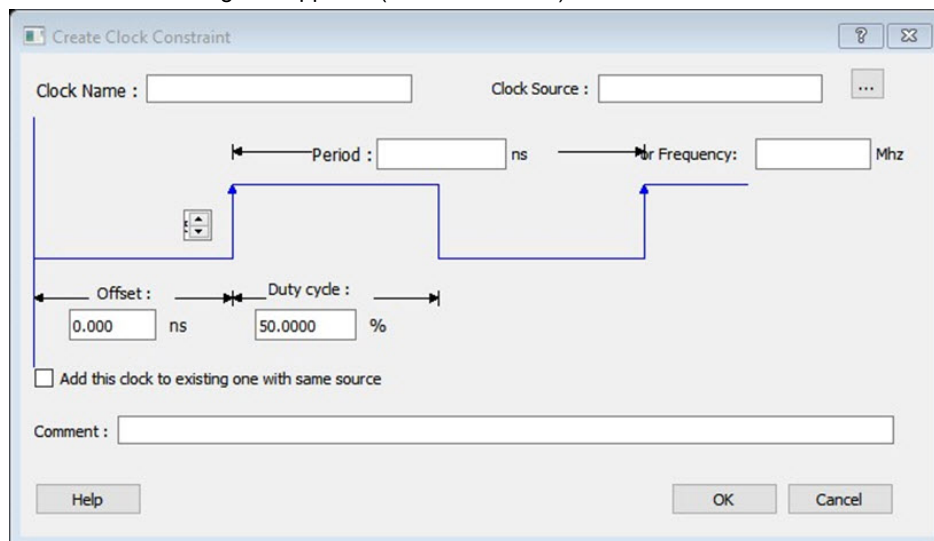
Specifying clock constraints is the most effective way to constrain and verify the timing behavior of a sequential design. Use clock constraints to meet your performance goals.

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).



The dialog box titled "Create Clock Constraint" contains the following fields and controls:

- Clock Name :** A text input field.
- Clock Source :** A text input field with a browse button (three dots).
- Period :** A text input field with a unit of "ns".
- Frequency :** A text input field with a unit of "Mhz".
- Offset :** A text input field with a value of "0.000" and a unit of "ns".
- Duty cycle :** A text input field with a value of "50.0000" and a unit of "%".
- ☐ Add this clock to existing one with same source
- Comment :** A text input field.
- Buttons:** Help, OK, and Cancel.

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, saves the newly created constraint in the database.

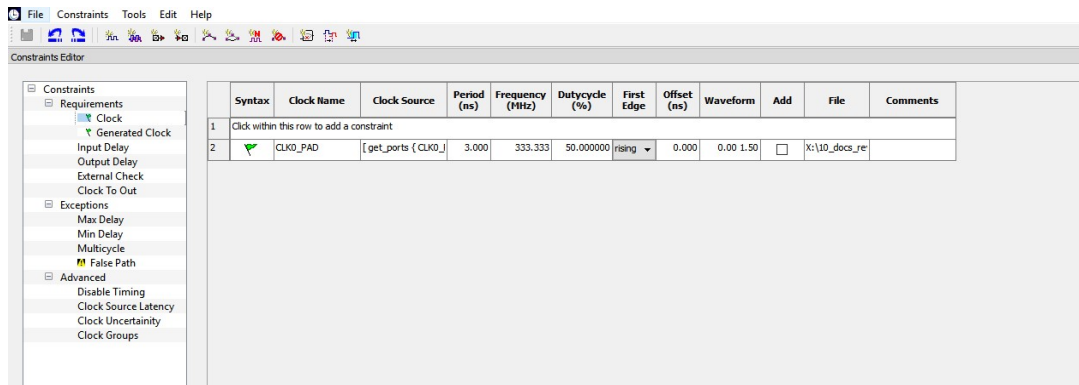



Figure 176 · Timing Constraint View

Specifying Generated Clock Constraints

Specifying a generated clock constraint enables you to define an internally generated clock for your design and verify its timing behavior. Use generated clock constraints and [clock constraints](#) to meet your performance goals.

To specify a generated clock constraint:

1. Open the [Create Generated Clock Constraint](#) dialog box using one of the following methods:
 - Click the  icon.
 - Right-click the **Generated Clock** in the Constraint Browser and choose **Add Generated Clock**.
 - Double-click the Generated Clock Constraints grid. The Create Generated Clock Constraint dialog box appears (as shown below).

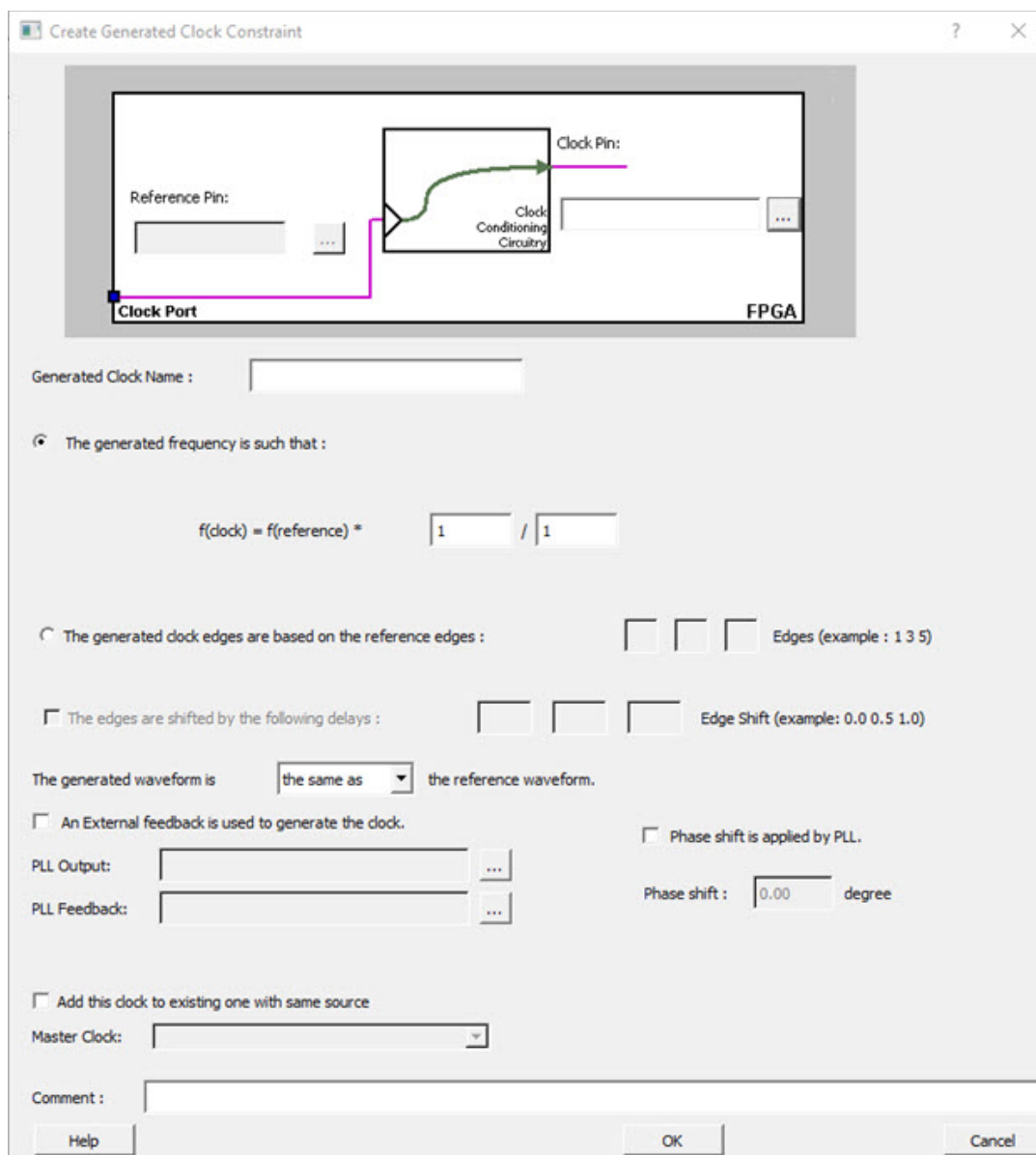


Figure 177 · Create Generated Clock Constraint

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).

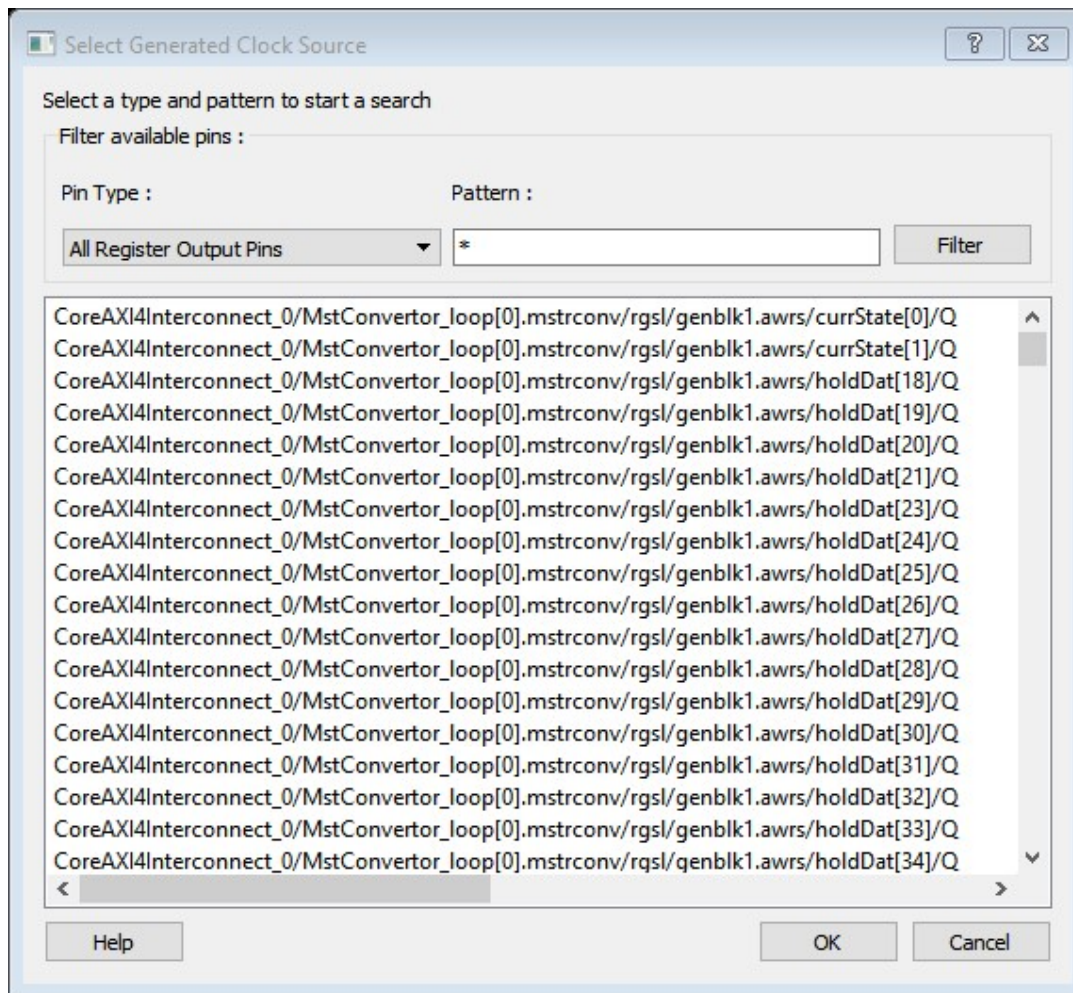


Figure 178 · Select Generated Clock Source Dialog Box

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.

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 7 · Default I/O Output Settings

Output State	Settings		
	Input	Control (Output Enable)	Output
Z (Tri-State)	1	0	0
0 (Low)	1	1	0
1 (High)	0	1	1
Last_Known_State	Last_Known_State	Last_Known_State	Last_Known_State

Table Key:

- 1 – 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.

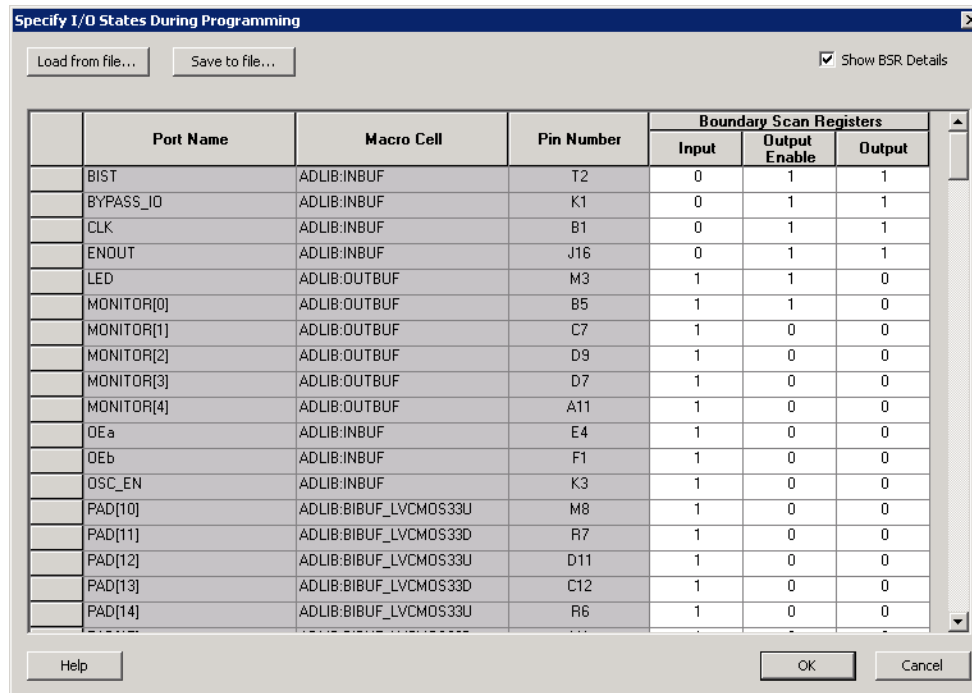
Table 8 · BSR Details I/O Output Settings

Output State	Settings		
	Input	Output Enable	Output
Z (Tri-State)	Don't Care	0	Don't Care
0 (Low)	Don't Care	1	0
1 (High)	Don't Care	1	1
Last Known State	Last State	Last State	Last State

Table Key:

- 1 – High: I/Os are set to drive out logic High
- 0 – Low: I/Os are set to drive out logic Low
- Don't Care – Don't Care values have no impact on the other settings.
- Last_Known_State – Sampled value: I/Os are set to the last value that was driven out prior to entering the programming mode, and then held at that value during programming

The figure below shows an example of Boundary Scan Register settings.



	Port Name	Macro Cell	Pin Number	Boundary Scan Registers		
				Input	Output Enable	Output
	BIST	ADLIB:INBUF	T2	0	1	1
	BYPASS_IO	ADLIB:INBUF	K1	0	1	1
	CLK	ADLIB:INBUF	B1	0	1	1
	ENOUT	ADLIB:INBUF	J16	0	1	1
	LED	ADLIB:OUTBUF	M3	1	1	0
	MONITOR[0]	ADLIB:OUTBUF	B5	1	1	0
	MONITOR[1]	ADLIB:OUTBUF	C7	1	0	0
	MONITOR[2]	ADLIB:OUTBUF	D9	1	0	0
	MONITOR[3]	ADLIB:OUTBUF	D7	1	0	0
	MONITOR[4]	ADLIB:OUTBUF	A11	1	0	0
	OEa	ADLIB:INBUF	E4	1	0	0
	OEb	ADLIB:INBUF	F1	1	0	0
	OSC_EN	ADLIB:INBUF	K3	1	0	0
	PAD[10]	ADLIB:BIBUF_LVCMOS33U	M8	1	0	0
	PAD[11]	ADLIB:BIBUF_LVCMOS33D	R7	1	0	0
	PAD[12]	ADLIB:BIBUF_LVCMOS33U	D11	1	0	0
	PAD[13]	ADLIB:BIBUF_LVCMOS33D	C12	1	0	0
	PAD[14]	ADLIB:BIBUF_LVCMOS33U	R6	1	0	0

Figure 179 · Boundary Scan Registers

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.

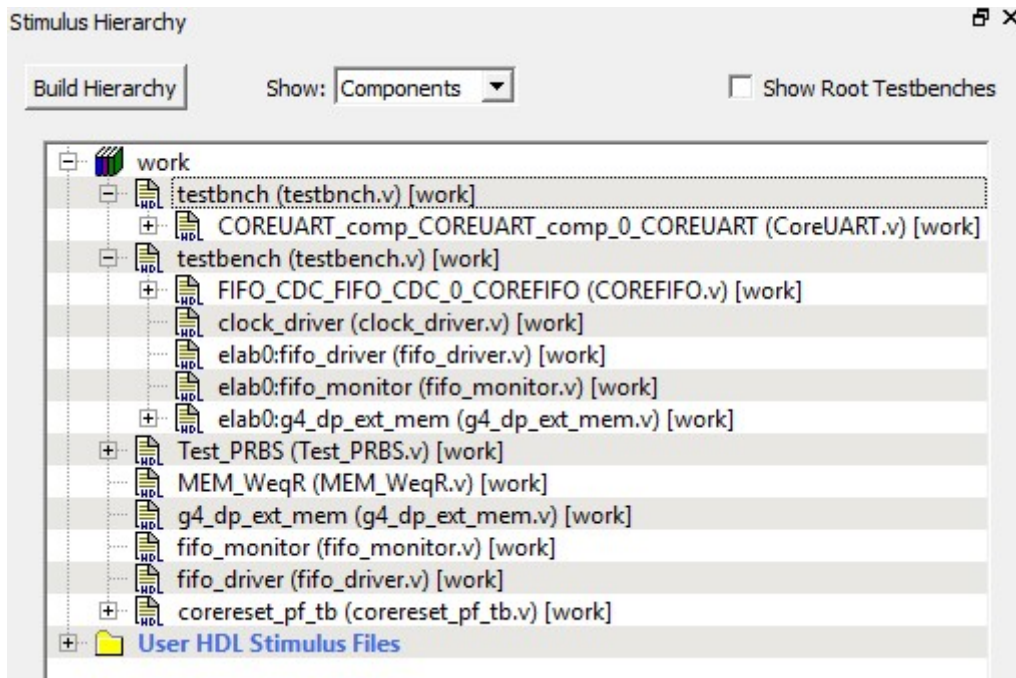


Figure 180 · Stimulus Hierarchy Dialog Box

Expand the hierarchy to view stimulus and simulation files. Right-click an individual component and choose **Show Module** to view the module for only that component.

Select **Components**, instance or **Modules** from the **Show** drop-down list to change the display mode. The Components view displays the stimulus hierarchy; the modules view displays HDL modules and stimulus files.

The file name (the file that defines the module or component) appears in parentheses.

Click **Show Root Testbenches** to view only the root-level testbenches in your design.

Right-click and choose **Properties**; the Properties dialog box displays the pathname, created date, and last modified date.

All integrated source editors are linked with the SoC software; if you modify a stimulus file the Stimulus Hierarchy automatically updates to reflect the change.






To open a stimulus file:


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 9 · Design Hierarchy Icons

Icon	Description
	SmartDesign component
	SmartDesign component with HDL netlist not generated
	SmartDesign testbench
	SmartDesign testbench with HDL netlist not generated
	IP core was instantiated into SmartDesign but the HDL netlist has not been generated

Icon	Description
	HDL netlist

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.

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.

Tool	Supported Version
Modelsim ME	10.5c
Modelsim ME Pro	10.5c
Synplify Pro ME	N-2017.09M SP1-1
Identify ME	N-2017.09M SP1

Table 10 · Table for supported tool versions

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**.

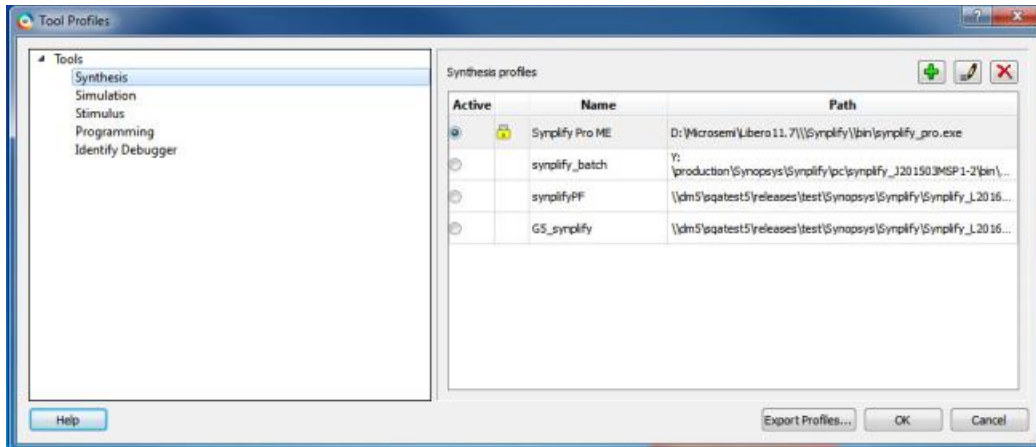


Figure 181 · Libero SoC Tool Profiles Dialog Box

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.

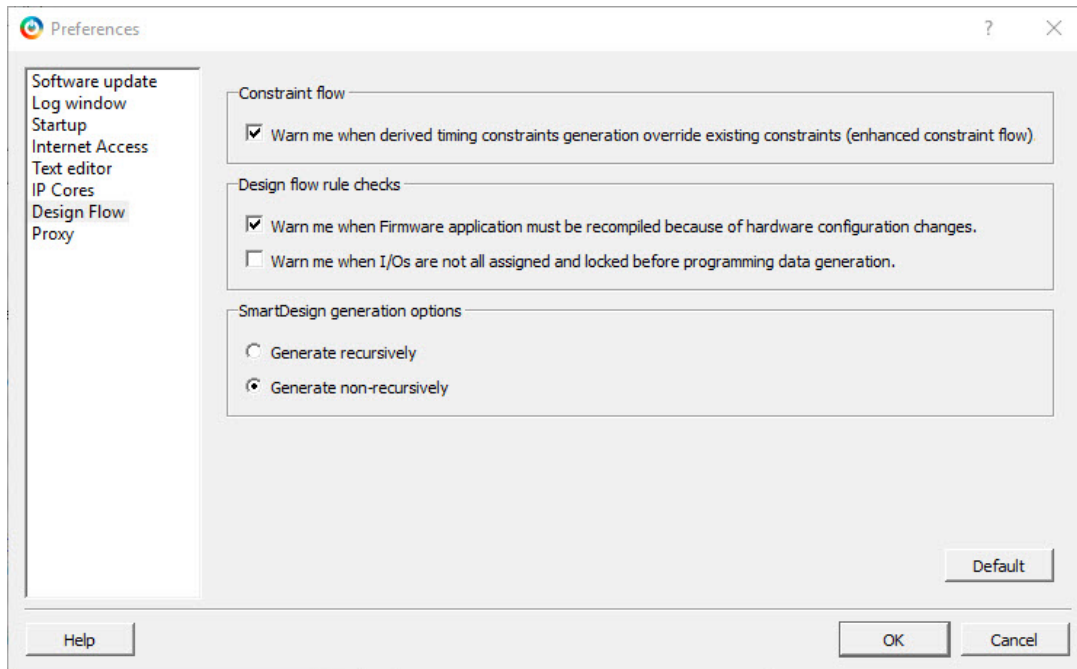


Figure 182 · Preferences Dialog Box – Design Flow Preferences

Constraint Flow

- **Warn me when derived timing constraints generation override existing constraints.**

Libero SoC can generate/derive timing constraints for known hardware blocks and IPs such as SERDES, CCC, MSS/HPMS. 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.

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.

Command	Action
create_clock	Creates a clock and defines its characteristics
create_generated_clock	Creates an internally generated clock and defines its characteristics
remove_clock_uncertainty	Removes a clock-to-clock uncertainty from the current timing scenario.
set_clock_latency	Defines the delay between an external clock source and the definition pin of a clock within SmartTime
set_clock_uncertainty	Defines the timing uncertainty between two clock waveforms or maximum skew
set_false_path	Identifies paths that are to be considered false and excluded from the timing analysis
set_input_delay	Defines the arrival time of an input relative to a clock
set_max_delay	Specifies the maximum delay for the timing paths
set_min_delay	Specifies the minimum delay for the timing paths
set_multicycle_path	Defines a path that takes multiple clock cycles

Command	Action
set_output_delay	Defines the output delay of an output relative to a clock

See Also

[SDC Syntax Conventions](#)

Libero DesignFlow SDC Commands

SDC Syntax Conventions

The following table shows the typographical conventions that are used for the SDC command syntax.

Syntax Notation	Description
command -argument	Commands and arguments appear in <i>Courier New</i> typeface.
<i>variable</i>	Variables appear in blue, italic <i>Courier New</i> typeface. You must substitute an appropriate value for the variable.
[-argument <i>value</i>]	Optional arguments begin and end with a square bracket.

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:

Wildcard	What it does
\	Interprets the next character literally
*	Matches any string

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}
```

See Also

[About SDC Files](#)

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.

Supported Families

SmartFusion2, IGLOO2, RTG4

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

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:

```
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:

```
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.

```
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.

```
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.

```
create_generated_clock -name clock_name [-add] [-master_clock clock_name] -source
reference_pin [-divide_by divide_factor] [-multiply_by multiply_factor] [-invert] source -
pll_output pll_feedback_clock -pll_feedback pll_feedback_input [-edges values] [-edge_shift
values]
```

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 in conjunction with -add option of the generated clock.

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}.

Supported Families

SmartFusion2, IGLOO2, RTG4

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.

```
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 $\frac{3}{4}$ of the period at the reference pin clk.

```
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.

```
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.

```
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.

```
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 $\frac{3}{4}$ of the period at the reference pin clk.

```
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.

```
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.

Supported Families

SmartFusion2, IGLOO2, RTG4

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.

Supported Families

SmartFusion2, IGLOO2, RTG4

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 ({}).

Supported Families

SmartFusion2

IGLOO2

RTG4

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.

Supported Families

SmartFusion2, IGLOO2, RTG4

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:

```
set_clock_uncertainty 2 -setup [get_clocks clk]
set_clock_uncertainty 2 [get_clocks clk]
```

Clock to Clock Uncertainty constraint examples:

```
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](#)
[remove_clock_uncertainty](#)

set_disable_timing

SDC command; disables timing arcs within the specified cell and returns the ID of the created constraint if the command succeeded.

```
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.

Supported Families

SmartFusion2, IGLOO2, RTG4

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 ({}).

Supported Families

SmartFusion2, IGLOO2, RTG4

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.

Supported Families

SmartFusion2, IGLOO2, RTG4

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.

```
set_input_delay delay_value -clock clock_ref [-max] [-min] [-clock_fall] [-rise] [-fall] [-add_delay] input_list
```

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

Supported Families

SmartFusion2, IGLOO2, RTG4

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:

```
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:

```
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:

```
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:

```
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.

```
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_load

SDC command; sets the load to a specified value on a specified port.

```
set_load capacitance port_list
```

Arguments

capitance

Specifies the capacitance value that must be set on the specified ports.

port_list

Specifies a list of ports in the current design on which the capacitance is to be set.

Description

The load constraint enables the Designer software to account for external capacitance at a specified port. You cannot set load constraint on the nets. When you specify this constraint on the output ports, it impacts the delay calculation on the specified ports.

Examples

The following examples show how to set output capacitance on different output ports:

```
set_load 35 out_p
set_load 40 {O1 O2}
set_load 25 [get_ports out]
```

Supported Families

SmartFusion2, IGLOO2, RTG4

Microsemi Implementation Specifics

- In SDC, you can use the `set_load` command to specify capacitance value on nets. Microsemi Implementation only supports output ports.

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.

Supported Families

SmartFusion2, IGLOO2, RTG4

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.

Supported Families

SmartFusion2, IGLOO2, RTG4

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.

Supported Families

SmartFusion2, IGLOO2, RTG4

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.

```
set_output_delay delay_value -clock clock_ref [-max] [-min] [-clock_fall] [-rise] [-fall] [-add_delay] output_list
```

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

Supported Families

SmartFusion2, IGLOO2, RTG4

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:

Design Object	Access Command
Cell	get_cells
Clock	get_clocks
Net	get_nets
Port	get_ports
Pin	get_pins
Input ports	all_inputs
Output ports	all_outputs
Registers	all_registers

See Also[About SDC Files](#)**all_inputs**[Design object access command](#); returns all the input or inout ports of the design.

```
all_inputs
```

Arguments

- None

Supported Families

SmartFusion2, IGLOO2, RTG4

Exceptions

- None

Example

```
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.

```
all_outputs
```

Arguments

- None

Supported Families

SmartFusion2, IGLOO2, RTG4

Exceptions

- None

Example

```
set_max_delay -from [all_inputs] -to [all_outputs]
```

Microsemi Implementation Specifics

None

See Also

[SDC Syntax Conventions](#)

all_registers

[Design object access command](#); returns either a collection of register cells or register pins, whichever you specify.

```
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.

Supported Families

SmartFusion2, IGLOO2, RTG4

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

```
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.

Supported Families

SmartFusion2, IGLOO2, RTG4

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.

Supported Families

SmartFusion2, IGLOO2, RTG4

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

Example

```
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.

Supported Families

SmartFusion2, IGLOO2, RTG4

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.

Supported Families

SmartFusion2, IGLOO2, RTG4

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.

Supported Families

SmartFusion2, IGLOO2, RTG4

Exceptions

None

Example

```
create_clock -period 10[get_ports CK1]
```

Microsemi Implementation Specifics

None

See Also

[SDC Syntax Conventions](#)