This document is released with Libero SoC Design Suite v11.8 SP4 without changes from v11.8 SP3.

Classic Constraint Flow User Guide

Libero SoC v11.8 SP1, SP2, and SP3

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.**

Microsemi

Power Matters.™

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.

Microsemi Corporate 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

©2017 Microsemi Corporation. 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.

About Microsemi

Microsemi Corporation (Nasdaq: MSCC) 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. Microsemi is headquartered in Aliso Viejo, California, and has approximately 4,800 employees globally. Learn more at **www.microsemi.com.**

5-02-9120-38/07.17

Table of Contents

Table of Contents 4

Libero SoC Introduction 12
Welcome to Microsemi's Libero® SoC v11.8 SP1 12
Libero SoC Design Flow (Classic Constraint Flow) 12
Microsemi License Utility 15
File Types in Libero SoC 16
Software Tools - Libero SoC 17
Frequently Asked Questions - Libero SoC 18
Firmware Cores Frequently Asked Questions 20
Software IDE Integration 21
Running Libero SoC from your Software Tool Chain 21

Create Design 22
System Builder 22
Instantiate a SmartFusion MSS in your Design 22
Configure the SmartFusion MSS 22
Generate SmartFusion MSS Files 24
Instantiate a SmartFusion2 MSS in your Design 25
Configure the SmartFusion2 MSS 25
Generate SmartFusion2 MSS Files 26
About SmartDesign 27
SmartDesign Design Flow 27
Using Existing Projects with SmartDesign 28
SmartDesign Frequently Asked Questions 28
General Questions 28
Instantiating Into Your SmartDesign 28
Working in SmartDesign 29
Working with Processor-Based Designs in SmartDesign 30
VHDL Construct Support in SmartDesign 31
Making your Design Look Nice 32
Generating your Design 32
Creating a New SmartDesign Component 33
Opening an Existing SmartDesign Component 33
Saving/Closing a SmartDesign Component 33
Generating a SmartDesign Component 33

Importing a SmartDesign Component 34
Deleting a SmartDesign Component from the Libero SoC Project 34
Generating a Memory Map 34
Modify Memory Map Dialog Box 37
Canvas Overview 38
Displaying Connections on the Canvas 40
Making Connections Using the Canvas 42
Simplifying the Display of Pins on an Instance using Pin Groups 43
Bus Instances 44
Adding Graphic Objects 44
Auto-Arranging Instances 45
Replace Component for Instance 46
Replace Instance Version 46
Slicing 46
Rename Net 47
Organizing Your Design on the Canvas 48
Renaming a Component 48
Adding Components and Modules (Instantiating) 48
Adding or Modifying Top Level Ports 49
Automatic Connections 50
QuickConnect 52
Manual Connections 53
Deleting Connections 53
Top-Level Connections 54
About Bus Interfaces 54
Using Bus Interfaces in SmartDesign 55
Adding or Modifying Bus Interfaces in SmartDesign 56
Bus Interfaces 58
DirectCore Bus Interfaces 59
Show/Hide Bus Interface Pins 59
Default Tie-offs with Bus Interfaces 60
Tying Off (Disabling) Unused Bus Interfaces 60
Required vs. Optional Bus Interfaces 61
Promoting Bus Interfaces to Top-level 61
Reconfiguring a Component 61
Fixing an Out-of-Date Instance 61
Replacing Component Version 62
Design State Management 63
Design Rules Check 63
Generating a SmartDesign Component 65
SmartDesign Menu 65
SmartDesign Glossary 66
Canvas Icons 68
VHDL Special Types - Examples and meta.out File Format 74
Create Core from HDL 78
Create HDL 80

Using the HDL Editor 80
Importing HDL Source Files 81
Mixed-HDL Support in Libero SoC 81
SmartDesign Testbench 82
HDL Testbench 82
View/Configure Firmware Cores 84
Designing with Designer Block Components 86
Designer Blocks and Synthesis 87
Managing I/Os in a Designer Block Component 88
Globals and Designer Block Components 88
Designer Block Compile Report 88
Designer Block Component Limitations 89
Creating a Designer Block Component in Libero SoC 89
Instantiating a Designer Block in Libero SoC 90
RTL Simulation 92
Simulation Options 92
Selecting a Stimulus File for Simulation 94
Selecting Additional Modules for Simulation 94
Performing Functional Simulation 94
Performing DirectCore Functional Simulation 95

Create Constraints 96
Constraints Overview 96
I/O Constraints - SmartFusion2, IGLOO2, RTG4 96
Timing Constraints - SmartFusion2, IGLOO2, and RTG4 97
Floorplan Constraints - SmartFusion2, IGLOO2, RTG4 98

Implement Design 99
Synthesize (Classic Constraint Flow) 99
Synplify Pro ME 102
Precision RTL 103
Compile - SmartFusion, IGLOO, ProASIC3, Fusion 103
Compile - SmartFusion2, IGLOO2, and RTG4 104
Compile Options (SmartFusion, IGLOO, ProASIC3, and Fusion) 107
Configure Flash*Freeze 112
Place and Route - SmartFusion2, IGLOO2, RTG4 113
Global Net Report 116
Place and Route 123
Multiple Pass Layout Configuration (SmartFusion2, IGLOO2, RTG4) 125
SmartFusion, IGLOO, ProASIC3 and Fusion Place and Route Advanced Options 127
Simulate - Opens ModelSim ME 129
Generate Back Annotated Files - SmartFusion2, IGLOO2, and RTG4 130
Generate Bitstream 131
Verify Timing Configuration 132

Types of Timing Reports....134
Verify Timing with SmartTime....136
Verify Power with SmartPower....136
IO Advisor (SmartFusion2, IGLOO2, and RTG4)....137
Simultaneous Switching Noise....144
Generate Back Annotated Files - SmartFusion2, IGLOO2, and RTG4....150

Device Programming....152

Programming Connectivity and Interface....153
Programmer Settings....155
Device I/O States During Programming -- JTAG Mode Only....158
Configure Programming Options (SmartFusion2 and IGLOO2)....159
Configure Programming Options (RTG4 Only)....161
Update eNVM Memory Content (SmartFusion2 and IGLOO2)....164
Update uPROM Memory Content - RTG4 Only....167
Configure Security Policy Manager....169
Update Policy....172
Debug Security Policy....174
Key Mode Policy....174
Security Features Frequently Asked Questions....176
Security Programming Files....176
Configure Bitstream....177
Generate Bitstream....177
Run PROGRAM Action - SmartFusion2, IGLOO2, and RTG4 Only....178
Program Device - SmartFusion, IGLOO, ProASIC3, and Fusion....178
SmartFusion2 and IGLOO2 Programming - Default Settings....179
SmartFusion2 and IGLOO2 Programming - Custom Settings....179
Exit Codes (SmartFusion2 and IGLOO2)....179
Bitstream Encryption with Default Key in the Security Policy Manager....189
Enable Custom Security Options in the Security Policy Manager - SmartFusion2 and IGLOO2190
Programming SmartFusion in the Libero SoC....193
Generate a Programming File in FlashPoint....194
Programming File Types....195
Generate a Programming File for SmartFusion....197
Generate a Programming File for CoreMP7/Cortex-M1 Device Support....197
Generate a Programming File for AFS Device Support - Designer Only....198
Generate a Programming File for Serialization Support in In House Programming (IHP)....199
Creating a Programming Database (PDB) File in Designer....202
Programming Embedded Flash Memory Block....203
Programming the FlashROM....203
Silicon Signature....205
Programming Security Settings....205
Custom Security Levels....207
Reprogramming a Secured Device....214
Custom Serialization Data for FlashROM Region....216
Custom Serialization Data File Format....217

Specifying I/O States During Programming 219
Custom I/O Settings and Boundary Scan Registers 220
Specifying I/O States During Programming - I/O States and BSR Details 220
Specify I/O States During Programming Dialog Box 222
Generate a DAT file 224
FlashLock® 224
Generating Bitstream and STAPL files 225

SmartFusion2 and IGLOO2 Programming Tutorials 226
SPI Programming Tutorial Overview 226
Auto Programming 226
In Application Programming (IAP) Tutorial 227
Programming Recovery Tutorial 228
SmartFusion2 Programming Tutorial Overview 230
MSS Configuration - eNVM 231
Generate Bitstream 234
Edit Design Hardware Configuration - Device I/O States During Programming 235
Configure Security Policy Manager 236
Update eNVM Memory Content (SmartFusion2 and IGLOO2) 239
Program Design - Run PROGRAM Action 241
Handoff Design for Production 242

Debug Design 243
Identify Debug Design 243
SmartDebug 244

Handoff Design for Production 245
Export Bitstream – SmartFusion2 and IGLOO2 245
Export Bitstream - RTG4 245
Export Pin Report 246
Export BSDL File 247
Export IBIS Model 247
Export Firmware – SmartFusion2 248

Reference Topics, Menus and Dialog Boxes 251

Catalog 251
Catalog Options Dialog Box 252
Changing Device Information 253
Core Manager 254
Deleting Files 254
Design Hierarchy in the Design Explorer 255
Design Menu - Libero SoC 256
Designer in Libero SoC 257
Edit Core Definition - Ports and Parameters Dialog Box 257
Edit Menu - Libero SoC 258

Execute Script Dialog Box 258
Export Script Dialog Box....258
File Menu - Libero SoC....259
Files Tab and File Types....260
HDL Templates in Libero SoC 261
Help Menu - Libero SoC 262
Import Files Dialog Box (Project Manager) 263
License Details 264
Link Files....264
Log Window 264
New Project Creation Wizard – Project Details....265
New Project Creation Wizard – Device Selection 266
New Project Creation Wizard – Device Settings 267
New Project Creation Wizard – Design Template....269
New Project Creation Wizard – Add HDL Source Files....270
New Project Creation Wizard - Add Constraints 270
New File Dialog Box 272
Open Project Dialog Box....273
Opening your Libero SoC project 274
Organize Constraint Files 275
Organize Simulation Files Dialog Box 275
Organize Source Files Dialog Box – Synthesis....276
Organize Stimulus Files Dialog Box 277
Physical Synthesis and the Libero SoC....277
Preferences 278
User Preferences Dialog Box....278
Project Menu - Libero SoC....279
Project Settings Dialog Box 280
Project Settings: Simulation 282
Project Sources 284
Reserved Microsemi Keywords....285
Right-Click (Shortcut) Menu Options in Libero SoC Design Hierarchy 285
Save Project As Dialog Box 285
Saving Files....287
Script Export Options Dialog Box....288
Search in Libero SoC....288
Select a Workspace Dialog Box....290
Organize Source Files Dialog Box – Synthesis....291
Stimulus Hierarchy 291
Text Editor....293
Tool Profiles Dialog Box....293
Tools Menu - Libero SoC....294
Vault/Repositories Settings Dialog Box 294
Videos - Libero SoC 295
View Design Datasheet/Report 296
View Menu - Libero SoC....296

VHDL Library - Add, Remove, or Rename.....297
Application Notes.....297
Tutorials and Training Modules.....298

Product Support.....299

Libero SoC Introduction

Welcome to Microsemi's Libero® SoC

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 – 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

Libero SoC Design Flow (Classic Constraint Flow)

The Libero SoC Build button enables you to proceed from synthesis to Place and Route in one click (using default settings).

The basic design flow is shown in the figure below.

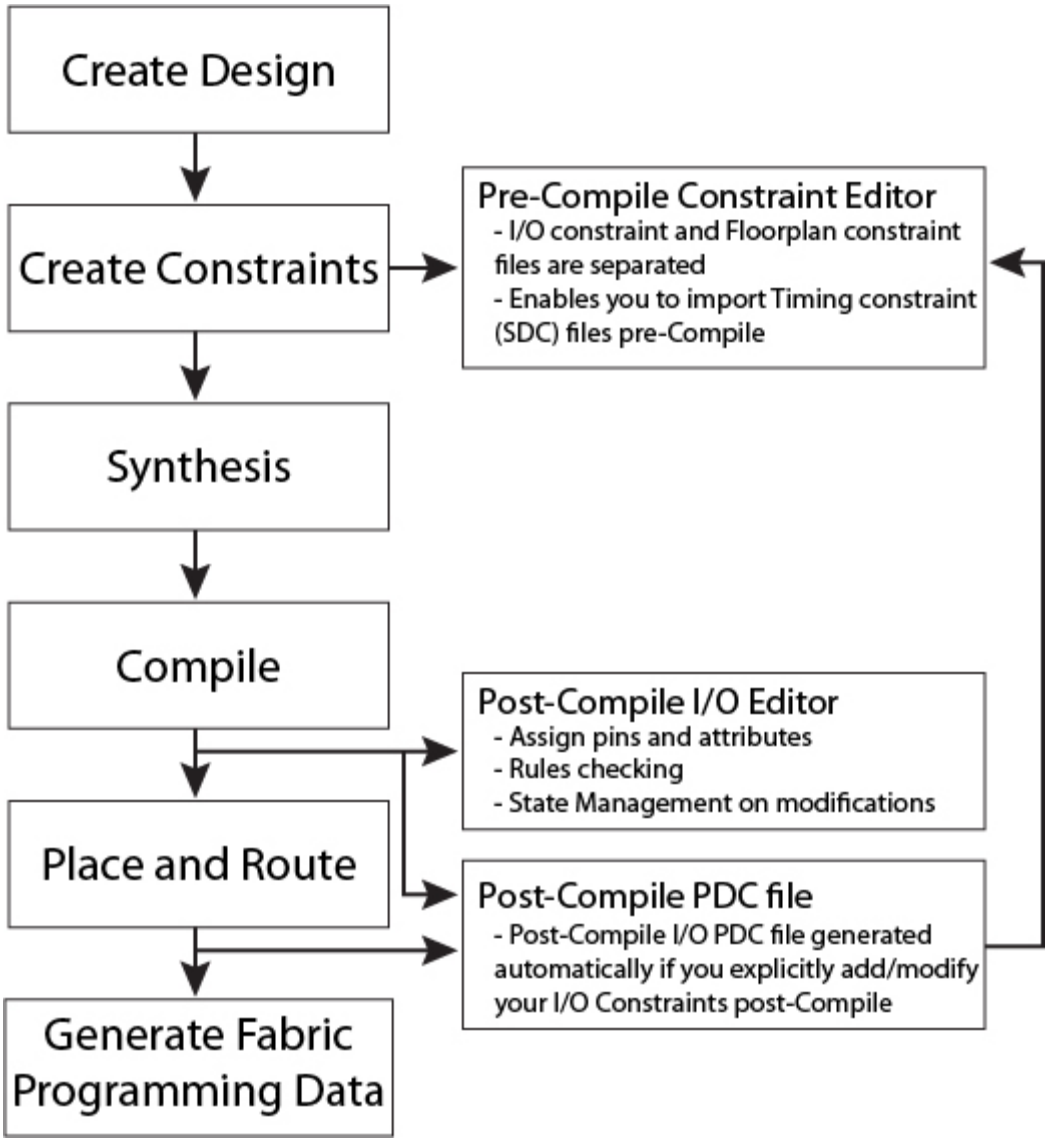


Figure 1 · Libero SoC Classic Constraint Design Flow

Create Design

Once you create your design (using System Builder; using the MSS builder - the flow for which is similar to the MSS flow for SmartFusion; create SmartDesign; Create HDL; SmartDesign Testbench) and click the **Build** button the software automatically executes the operations below with default settings (if it encounters no errors).

Create Constraints - Pre-Compile

I/O constraint PDC files are separate from Floorplan constraint PDC files; if you have a PDC file that contains both I/O and Floorplan constraints then Libero SoC errors out with an invalid constraint error.

- I/O Constraints - To add an I/O constraint, in the Design Flow window expand Create Constraints, right-click **I/O Constraints** and choose **Import Files**.

- Timing Constraints - Enables you to import SDC files pre-Compile.
- Floorplan Constraints - Created with the Floorplanner or a text editor; to add a Floorplan constraint, in the **Design Flow** window expand **Create Constraints**, right-click **Floorplan Constraints** and choose **Import Files**.

Synthesis

Double-click **Synthesize** to run synthesis on your design automatically; automatic synthesis uses the default settings in your synthesis tool.

Compile

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

Place and Route

Place and Route runs automatically with default settings as part of the push-button design flow in Libero SoC.

Edit Constraints - Post-Compile

- I/O Constraints - The Post-Compile I/O Editor displays all assigned and unassigned I/O macros and their attributes in a spreadsheet format; use this editor to view, sort, select, edit, lock and unlock assigned attributes.

  The post compile editor ensures that the Compile/Place and Route state is maintained (you do not have to rerun Compile or Place and Route), if you make changes to the attributes that do not require it.

  However, if you modify the I/O PDC file directly, it is equivalent to modifying the source file of the design, which means the tools starting from Compile will become out of date because one of the source files was modified.
- Timing Constraints - Run SmartTime to perform Min/Max analysis and manage timing constraints.
- Floorplan Constraints - Use to create and edit regions on your chip and assign logic to these regions.

Generate Fabric Programming Data

Generates programming data for your design.

IGLOO, ProASIC3, and Fusion Programming

SmartFusion, IGLOO, ProASIC3, and Fusion devices use the FlashPoint program file generator to create a programming file. For SmartFusion, IGLOO, and ProASIC3, the FlashPoint interface also enables the advanced security features.

To open FlashPoint, expand Implement Design, right-click Generate Programming Data, and choose **Open Interactively**. FlashPoint allows you to generate the following programming files:

- IEEE 1532 files (*.bsd, *.isc)
- DirectC files (*.dat)
- Programming Data File (*.pdb)
- STAPL file (*.stp)
- Serial Vector Files (*.svf)

You must have completed your design to generate your programming (*.stp or STAPL) file.

SmartFusion2, IGLOO2, and RTG4 Programming

You do not have to open FlashPro or FlashPoint to program your SmartFusion2 device. All programming functionality is available from within the Design Flow window, including:

Programming Connectivity and Interface - Organizes your programmer(s) and devices.

Programmer Settings - Opens your programmer settings; use if you wish to program using settings other than default.

Device I/O States During Programming - Sets your device I/O states during programming; use if your design requires that you change the default I/O states.

Security Policy Manager - Enables you to set your Secured Programming Use Model, User Key Entry and Security Policies for your design.

Microsemi License Utility

The Microsemi SoC Products Group LicenseTroubleshooting Guide answers the most common licensing-related frequently asked questions.

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 v***x.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.

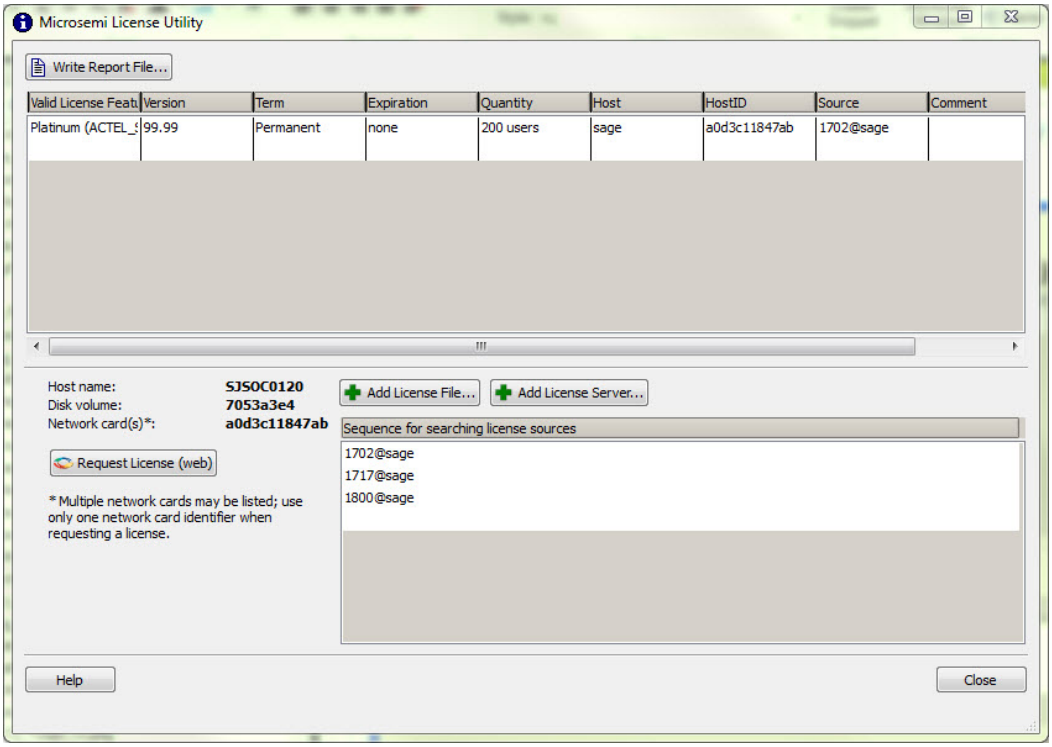


Figure 2 · Microsemi License Utility

File Types in Libero SoC

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

The Project Manager enables you to manage your files as you import them. If you want to store and maintain your design source files and design constraint files in a central location outside the Project location, Libero gives you the option to link them to your Libero project folders when you first create your project. These linked files are not copied but rather linked to your project folder.

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

The top level directory (<project_name>) contains your *.prjx file; only one *.prjx file is enabled for each Libero SoC project. If you associate Libero SoC as the default program with the *.prjx file (Project > Preferences > Startup > Check the default file association (.prjx) at startup), you can double-click the *.prjx file to open the project with Libero SoC.

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

- HDL source files to be used for synthesis and simulations
- Stimulus files and configuration files for simulation
- Firmware files for software IDE tools
- Configuration files for programming

- Configuration files for power analysis.

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 **-** ADB files (Microsemi Designer project files), *_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)

Note: The Microsemi ADB file memory requirement is equivalent to 2x the size of the ADB file. If your computer does not have 2x the size of your ADB file's memory available, please make memory available on your hard drive.

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:

http://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	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), and back-annotated (post-layout), dynamic simulation. (ModelSim is supported in Libero Gold and Platinum only.)

Frequently Asked Questions - Libero SoC

The collection of Frequently Asked Questions is useful for anyone that is new to Libero SoC. All the information listed below is explained in detail in other sections of the help, but the information is summarized here for easy reference. Click any question to go to the corresponding explanation.

Libero SoC Frequently Asked Questions

1. How do I set my Multi-Pass place and route options?
2. How do I set FlashPro security options?
3. How do I instantiate my HDL in SmartDesign?
4. How do I add a bus interface to my HDL code and then add it to SmartDesign?
5. I don't see any DirectCore IP's in the Catalog but I have both Libero IDE 9.1 and Libero SoC 10.0 installed. Where are the DirectCore IP's?
6. How do I assign I/O/s in Libero SoC?
7. How do I make sure that my design is using the latest driver(s)?
8. How do I improve the timing of my design?
9. How do I manage clocks?
10. How do I write a testbench?

Firmware Cores Frequently Asked Questions

1. Where are the firmware files generated?
2. Why are some firmware in italics?
3. Why am I getting the following error on generation? "Error: 'Missing Core Definition': Core 'Actel:Firmware:MSS_SPI_Driver:2.0.101 ' is missing from the vault."?
4. Why is my firmware view empty?
5. Why are there multiple firmware instances of the same type?

Libero SoC Frequently Asked Questions

How do I set my Multi-Pass place and route options?

The steps to run Multi-Pass place and route vary depending on the device family.

For SmartFusion, IGLOO, ProAsic and Fusion: In the Design Flow window, expand **Implement Design**, right-click **Place and Route** and choose **Open Interactively**. Designer opens. Click **Layout** to open the Layout Options dialog box and choose **Use Multiple Pass**. Click the Configure button to open the Multi-Pass Configurator. Set your Multi-Pass Options. Once Layout is complete, save your ADB to retain your custom place and route options.

For SmartFusion2, IGLOO2, RTG4: In the Design Flow window, expand **Implement Design**, right-click **Place and Route** and choose **Configure Options**. When the Layout Options dialog box appears, check the **Use Multiple Passes** option and click **Configure**. When the Multi-Pass configuration dialog box appears, set your Multi-Pass options.

How do I set FlashPro security options?

In the Libero SoC Design Flow window, expand **Program Design**, right-click **Program Device** and choose **Open Interactively**. FlashPro opens and enables you to set/change your security options. See the FlashPro help for more information.

How do I instantiate my HDL in SmartDesign?

Import your HDL file into the Libero SoC (File > Import Files). After you do this, your HDL module appears in the Project Manager Hierarchy. Then, drag-and-drop it from the Hierarchy onto your SmartDesign Canvas.

How do I add a bus interface to my HDL code and then add it to SmartDesign?

If you want to add a bus interface to your HDL code and then add it to SmartDesign, see the Adding or Modifying Bus Interfaces in SmartDesign topic.

I don't see any DirectCore IP's in the Catalog but I have both Libero IDE 9.1 and Libero SoC 10.0 installed. Where are the DirectCore IP's?

Make sure the vault location is correct. Click the Catalog Options button to open the Catalog Options dialog box. Then check and, if necessary, update your vault location.

How do I assign I/O's in Libero SoC?

In the Design Flow window, expand **Implement Design**, then expand **Constrain Place and Route**. Right-click **Edit I/O Attributes** and choose **Open Interactively** to open the I/O Attribute Editor.

How do I make sure that my design is using the latest driver(s)?

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. Use the Version drop down menus to check for the latest firmware and firmware drivers.

How do I improve the timing of my design?

The SmartTime tool enables you to set clock constraints, analyze timing, identify critical paths, and find the minimum cycle time that does not result in a timing violation.

To improve the timing of your design:

1. Run timing analysis to identify timing violations.
2. View the paths with timing violations.
3. Modify timing constraints on the critical path(s) in order to meet your timing requirements.
4. Run Timing-Driven Place and Route.

For more information on improving timing, see the Analysis and Optimization application notes. The Designing for Performance on Flash-Based FPGAs application note is a good starting point.

How do I manage clocks?

Specify clock constraints in your design. See the sections on explicit clocks, potential clocks and clock networks for more information on clocks in Libero SoC.

How do I write a testbench?

You can write or edit a testbench manually using the HDL editor, or you can create a new HDL testbench and automatically populate it with all your design information with Create New HDL Testbench in Libero SoC. Create New HDL Testbench is in the Design Flow window under **Create Design**.

Testbench file are generated automatically when you generate a SmartDesign. You can find them in your Files window in Libero SoC (**View > Window > Files**).

Firmware Cores Frequently Asked Questions

Where are the firmware files generated?

The firmware files are generated to the firmware working directory <project>\firmware. Your software IDE workspace is generated to <project>\<software IDE tool chain>.

Why are some firmware in italics?

This indicates the firmware is in the IP repository but not in your local IP vault. You must download it to your local IP vault so that the Libero SoC will generate the firmware files.

Why am I getting the following error on generation? "Error: 'Missing Core Definition': Core 'Actel:Firmware:MSS_SPI_Driver:2.0.101 ' is missing from the vault."?

This happens when a firmware that is in your design but the VLNV definition could not be found in your IP vault. This can happen if you:

- Changed your vault settings to point to another vault
- Opened a project that was created on another machine

Why is my firmware view empty?

Check that you are pointing to the proper firmware repository:

www.actel-ip.com/repositories/Firmware

Check with your network administrator to make sure you can communicate with Microsemi's IP repository URL.

Why are there multiple firmware instances of the same type?

Some firmware cores have configurable options, and in certain cases you will have two peripherals of the same firmware VLNV. In this situation, you may want to configure each peripheral driver separately.

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

Libero SoC Frequently Asked Questions

Running Libero SoC from your Software Tool Chain

View/Configure Firmware Cores

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

Libero SoC Frequently Asked Questions

Software IDE Integration

View/Configure Firmware Cores

Create Design

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

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.

Instantiate a SmartFusion MSS in your Design

You can configure peripherals within the SmartFusion 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 SmartFusion FPGA fabric.

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

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

Documents for specific SmartFusion MSS peripherals are available on the Peripheral Documents web page.

The SmartFusion 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

- ACE Configuration
- ACE Simulation
- AHB Bus Matrix Configuration
- Clock Configuration

- Configurator Overview
- Embedded FlashROM (eFROM) Configuration
- Embedded Nonvolatile Memory (eNVM) Configuration
- Ethernet MAC Configuration
- External Memory Controller (EMC) Configuration
- Firmware
- GPIO Configuration
- I2C Configuration
- Interrupt Management
- I/O Configuration
- I/O Editor
- Peripheral DMA Configuration
- Real Time Counter (RTC) Configuration
- Reset Management Configuration
- SPI Configuration
- Timer Configuration
- UART Configuration
- Watchdog Configuration

Fabric

- Dedicated Fabric Clock Conditioning Circuit with PLL Integration

Interfaces

- How to Create a MSS and Fabric AMBA AHBLite Design (MSS Master Mode)
- How to Create a MSS and Fabric AMBA APB3 Design (MSS Master Mode)
- How to Create a MSS and Fabric AMBA AHBLite/APB3 Design (MSS Master Mode)

SmartFusion SmartDesign Documents

- SmartDesign MSS Canvas
- SmartDesign MSS Simulation
- SmartDesign MSS Running the MSS Configurator in your Software Tool Chain

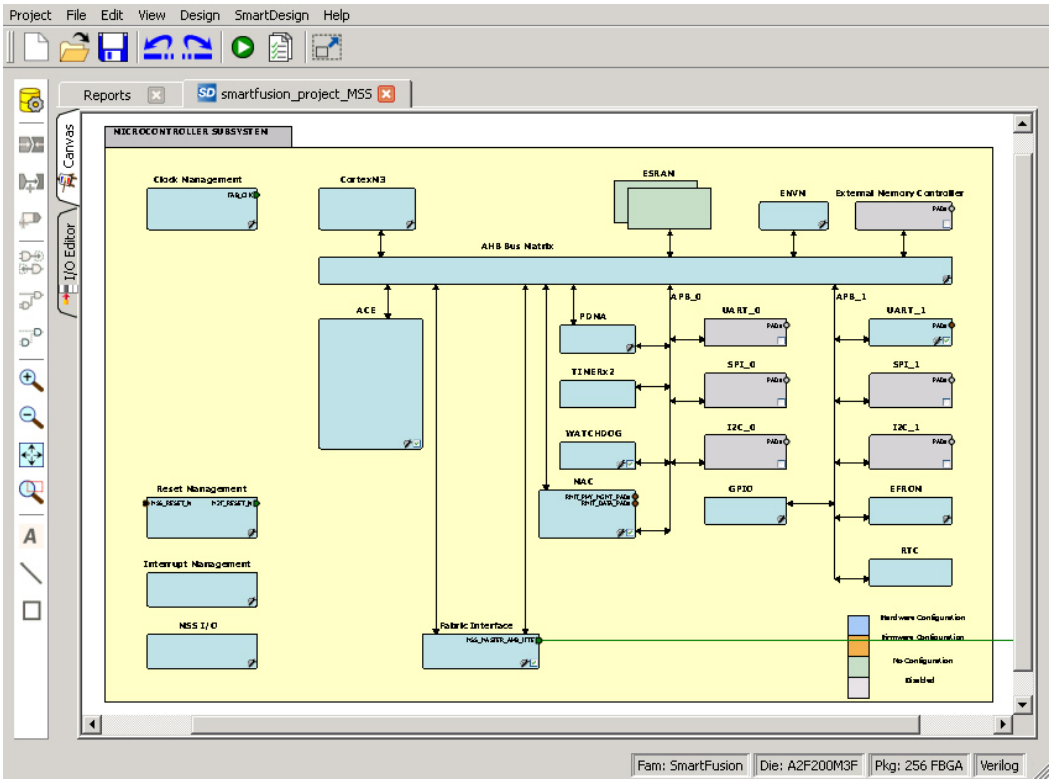


Figure 3 · SmartFusion MSS

Generate SmartFusion MSS Files

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

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

The MSS Configurator generates the following files:

- HDL files for the MSS design and its sub-components: MSS CCC, etc. HDL files are automatically managed by the Libero SoC and passed to the Synthesis and Simulation point tools.
- EFC File. MSS hardware configuration that is loaded into eNVM. FlashPro automatically detects this file and includes it in your final programming file.
- UFC file. Contains your Embedded FlashROM configuration and data. FlashPro automatically detects this file and includes it 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 point 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.

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 SECDED Configurator
- MSS Security Configurator

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

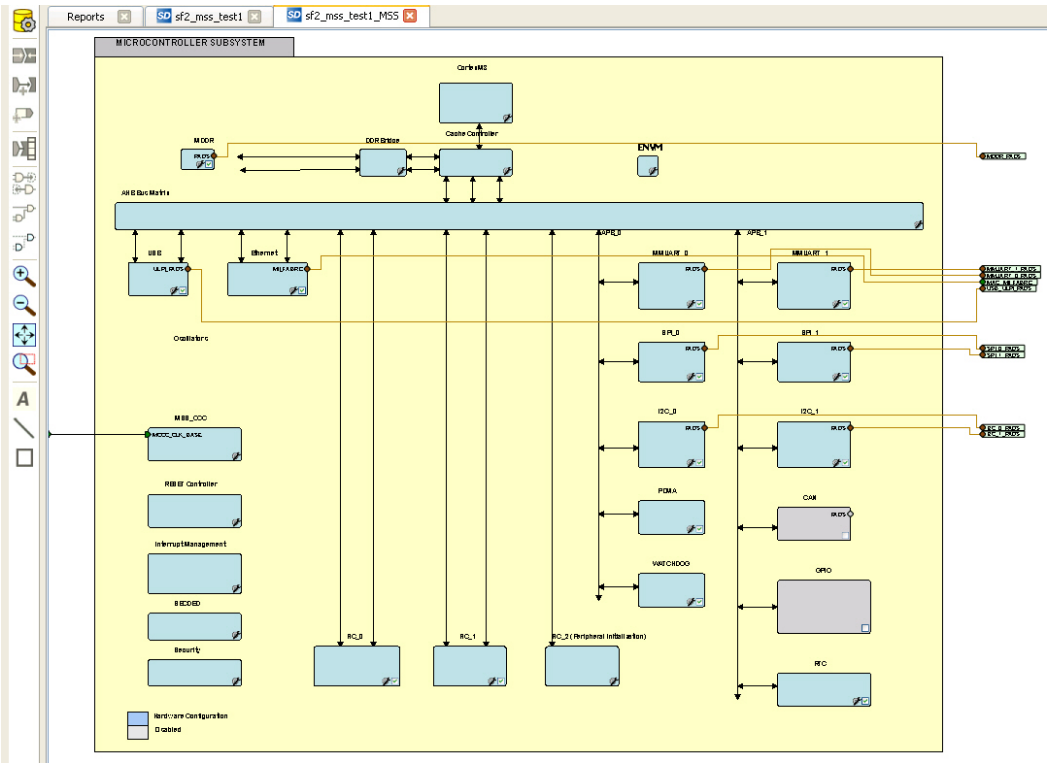


Figure 4 · Microcontroller Subsystem Configurator

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.

About SmartDesign

SmartDesign is a visual block-based design creation tool for instantiation, configuration and connection of Microsemi IP, user-generated IP, custom/glue-logic HDL modules. The final result 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 of a larger design.

Instantiate IP cores, macros and HDL modules by dragging them from the Catalog onto the Canvas, where they are viewed as blocks in a functional block diagram. From the Canvas you can:

- Configure your blocks
- Make connections between your blocks
- Generate your SmartDesign
    - This step generates the HDL and testbench files required to proceed with Synthesis and Simulation.

SmartDesign supports all Microsemi SoC product families.

SmartDesign Design Flow

SmartDesign enables you to stitch together design blocks of different types (HDL, IP, etc.) and generate a top-level design. The Files tab lists your SmartDesign files in alphabetical order.

You can build your design using SmartDesign with the following steps:

Step One – Instantiating components: In this step you add one or more building blocks, HDL modules, components, and schematic modules from the project manager to your design. The components can be blocks, cores generated from the core Catalog, and IP cores.

Step Two – Connecting bus interfaces: In this step, you can add connectivity via standard bus interfaces to your design. This step is optional and can be skipped if you prefer manual connections. Components generated from the Catalog may include pre-defined interfaces that allow for automatic connectivity and design rule checking when used in a design.

Step Three – Connecting instances: The Canvas enables you to create manual connections between ports of the instances in your design. Unused ports can be tied off to GND or VCC (disabled); input buses can be tied to a constant, and you can leave an output open by marking it as unused.

Step Four – Generating the SmartDesign component: In this step, you generate a top-level (Top) component and its corresponding HDL file. This component can be used by downstream processes, such as synthesis and simulation, or you can add your SmartDesign HDL into another SmartDesign.

When you generate your SmartDesign the Design Rules Check verifies the connectivity of your design; this feature adds information to your report; design errors and warnings are organized by type and message and displayed in your Datasheet / Report.

You can save your SmartDesign at any time.

Using Existing Projects with SmartDesign

You can use existing Libero SoC projects with available building blocks in the project to assemble a new SmartDesign design component. You do not have to migrate existing top-level designs to SmartDesign and there is no automatic conversion of the existing design blocks to the SmartDesign format.

SmartDesign Frequently Asked Questions

The collection of SmartDesign Frequently Asked Questions are useful for anyone that is new to SmartDesign. All the information listed below is explained in detail in other sections of the help, but the information is summarized here for easy reference. Click any question to go to the corresponding explanation.

General Questions

- **What is SmartDesign?**

  SmartDesign is a design entry tool. It's the first tool in the industry that can be used for designing System on a Chip designs, custom FPGA designs or a mixture of both types in the same design. A SmartDesign can be the entire FPGA design, part of a larger SmartDesign, or a user created IP that can be stored and reused multiple times. It's a simple, intuitive tool with powerful features that enables you to work at the abstraction level at which you are most comfortable.

  It can connect blocks together from a variety of sources, verify your design for errors, manage your memory map, and generate all the necessary files to allow you to simulate, synthesize, and compile your design.

- **How do I create my first SmartDesign?**

  In the Libero SoC Project Manager Design Flow window, under Create Design, double-click **Create SmartDesign**.

Instantiating Into Your SmartDesign

- **Where is the list of Cores that I can instantiate into my SmartDesign?**

  The list of available cores is displayed in the Catalog. This catalog contains all DirectCore IP, Design Block cores, and macros.

- **How do I instantiate cores into my SmartDesign?**

  Drag and drop the core from the Catalog onto your SmartDesign Canvas. An instance of your Core appears on the Canvas; double-click to configure it.

- **I have a block that I wrote in VHDL (or Verilog), can I use that in my SmartDesign?**

  Yes! Import your HDL file into the Project Manager (File > Import Files). After you do this, your HDL module will appear in the Project Manager Hierarchy. Then, drag-and-drop it from the Hierarchy onto your SmartDesign Canvas.

- **My HDL module has Verilog parameters or VHDL generics declared, how can I configure those in SmartDesign?**

  If your HDL module contains configurable parameters, you must create a 'core' from your HDL before using it in SmartDesign. Once your HDL module is in the Project Manager Design Hierarchy, right-click it and choose **Create Core from HDL**. You will then be allowed to add bus interfaces to your module if necessary. Once this is complete, you can drag your new HDL+ into the SmartDesign Canvas and configure your parameters by double-clicking it.

Working in SmartDesign

- **How do I make connections?**

  Let SmartDesign do it for you. Right-click the Canvas and choose **Auto Connect**.

- **Auto Connect didn't connect everything for me, how do I make manual connections?**

  Enter **Connection Mode** and click and drag from one pin to another. Click the Connection Mode button in the Canvas to enter Connection Mode.

  Alternatively:

  1. Select the pins you want connected by using the mouse and the CTRL key.
  2. Right-click one of the selected pins and choose **Connect**.

- **How do I connect a pin to the top level?**

  Right-click the pin and choose **Promote to Top Level**. You can even do this for multiple pins at a time, just select all the pins you want to promote, right-click one of the pins and choose **Promote to Top Level**. All your selected pins will be promoted to the top level.

- **Oops, I just made a connection mistake. How do I disconnect two pins?**

  Use CTRL+Z to undo your last action. If you want to undo your 'undo', hit redo (CTRL+Y).

  To disconnect pins you can:

  - Right-click the pin you want to disconnect and choose **Disconnect**
  - Select the net and hit the delete key

- **I need to apply some simple 'glue' logic between my cores. How do I do that?**

  For basic inversion of pins, you can right-click a pin and choose **Invert**. An inverter will be placed at this pin when the design is generated. You can also right-click a pin and choose Tie Low or Tie High if you want to connect the pin to either GND or VCC.

  To tie an input bus to a constant, right-click the bus and choose **Tie to Constant**. To mark an output pin as unused, right-click the pin and choose **Mark as Unused**.

  To clear these, just right-click on the pin again and choose **Clear Attribute**.

- **My logic is a bit more complex than inversion and tie offs - what else can I do?**

  You have full access to the library macros, including AND, OR, and XOR logic functions. These are located in the Catalog, listed under Macro Library. Drag the logic function you want onto your SmartDesign Canvas.

- **How do I create a new top level port for my design?**

  Click the **Add Port** button in the Canvas toolbar

- **How do I rename one of my instances?**

  Double-click the instance name on the Canvas and it will become editable. The instance name is located directly above the instance on the Canvas.

- **How do I rename my top level port?**

  Right-click the port you want to rename and choose **Modify Port**.

- **How do I rename my group pins?**

  Right-click the group pin you want to rename and choose **Rename Group**.

- **I need to reconfigure one of my Cores, can I just double-click the instance?**

  Yes.

- **I want more Canvas space to work with!**

  Maximize your workspace (CTRL-W), and your Canvas will maximize within the Project Manager. Press CTRL-W again if you need to see your Hierarchy or Catalog.

Working with Processor-Based Designs in SmartDesign

- **How do I connect my peripherals to the bus?**

  Click **Auto Connect** and it will help you build your bus structure based on the processor and peripherals that you have instantiated.

- **But I need my peripheral at a specific address or slot.**

  Right-click the Canvas and choose **Modify Memory Map** to invoke the Modify Memory Map dialog that enables you to set a peripheral to a specific address on the bus.

  The bus core will show the slot numbers on the bus interface pins. These slot numbers correspond to a memory address on the bus.

  Verify that your peripheral is mapped to the right bus address by viewing your design's Memory Map.

- **How do I view the Memory Map of my design?**

  Generate your project and open datasheet in the **Report View**.

  The memory map section will also show the memory details of each peripheral, including any memory mapped registers.

- **How do I simulate my processor design?**

  SmartDesign automatically generates the necessary Bus Functional Model (BFM) scripts required to simulate your processor based design. A top level testbench for your SmartDesign is generated automatically as well.

  Create your processor design, generate it, and you will be able to simulate it in ModelSim.

- **I have my own HDL block that I want to connect as a peripheral on the AMBA bus. How can I do that?**

  SmartDesign supports automatic creation of data driven configurators based on HDL generics/parameters.

  If your block has all the necessary signals to interface with the AMBA bus protocol (ex: address, data, control signals):

  1. Right-click your custom HDL block and choose **Create Core from HDL**. The Libero SoC creates your core and asks if you want to add bus interfaces.
  2. Click **Yes** to open the Edit Core Definition dialog box and add bus interfaces. Add the bus interfaces as necessary.
  3. Click **OK** to continue.

  Now your instance has a proper AMBA bus interface on it. You can manually connect it to the bus or let Auto Connect find a compatible connection. See the DirectCore Advanced Microcontroller Bus Architecture - Bus Functional Model User's Guide for more information on CoreAMBA BFM commands.

- **How do I generate the firmware drivers for my design?**

  SmartDesign automatically finds all the compatible firmware drivers based on your peripherals and processor. You can view the list of firmware drivers that the design found by going to the design flow and choosing **View/Configure Firmware Cores**.

- **How do I start writing my application code for my design?**

  Libero SoC simplifies the embedded development process by automatically creating the workspace and project files for the Software IDE that you specify in the Tools profile.

  Once you have generated your design, the firmware and workspace files will automatically be created. Click **Write Application Code** in the Design Flow tab and the Software IDE tool will open your design's workspace files.

VHDL Construct Support in SmartDesign

- **What VHDL constructs do you support?**

  VHDL types Record, Array, Array of Arrays, Integer and Unsigned are supported on entity ports of imported VHDL files - these are treated as special types in Libero.

- **How can I import files with VHDL Special Types into SmartDesign?**

  To work with a VHDL file with Special Types you must:

  1. Drag and drop the entity into SmartDesign and connect it just as you would with any other SmartDesign instance.
  2. Generate the Mapping File (meta.out):
     Navigate to the Design Hierarchy view, under the current SmartDesign. Right-click every VHDL file or every top hierarchical file and choose **Create Mapping File (VHDL)**.
  3. Generate the SmartDesign
  4. Continue with the Libero SoC Design Flow steps (Synthesis, Simulation, etc.)

  If you do not generate the Mapping File, and try to Generate your SmartDesign, you will see the following error in the log window:

```
Error: Select the HDL file in the Design Hierarchy and right-click the HDL file and
choose Create Mapping File(VHDL) because at least one entity port is of type Array or
Record.
```

  The above is reported only if the entity port is of type Record, Array, Array of Array, or Unsigned.

- **What is the purpose of the mapping file?**

  The mapping file contains the mapping information between the SmartDesign ports and original user-specified data types of ports in design files, and is used for type casting of signals during design generation.

- **Where will the mapping file meta.out be generated?**

  The file is generated in your $project_dir/hdl folder. This file will be used to during SmartDesign generation.

- **What are the VHDL special types that are not generated automatically?**

  The following types are not automatically generated from the right-click menu option **Create Mapping File(VHDL)**:

  - Array of array is not supported
  - Array of record is not supported
  - Enum in range of array is not supported.
  - Constants are not supported.
  - Buffer output ports are not supported

- **What do I do if I am using VHDL types that are not generated automatically?**

  You must manually write the mapping information in the meta.out file for unsupported types (types which are not generated automatically) in the prescribed format. Click the link to see an example.

  - Integer
  - Unsigned
  - Array and Array of Arrays
  - Record

- **What is the meta.out file format?**

  See the meta.out file format topic for more information.

Making your Design Look Nice

- **Can the tool automatically place my instances on the Canvas to make it look nice?**

  Yes. Right-click the Canvas white space and choose **Auto Arrange Instances**.

- **My design has a lot of connections, and the nets are making my design hard to read. What do I do?**

  You can disable the display of the nets in the menu bar (RMC > Hide Nets). This automatically hides all the nets in your design.

  You can still see how pins are connected by selecting a connected pin, the net will automatically be visible again.

  You can also selectively show certain nets, so that they are always displayed, just right click on a connected pin and choose **Show Net**.

- **My instance has too many pins on it; how can I minimize that?**

  Try grouping functional or unused pins together. For example, on the CoreInterrupt there are 8 FIQSource* and 32 IRQSource* pins, group these together since they are similar in functionality.

  To group pins: Select all the pins you want to group, then right-click one of the pins and choose **Add pins to group**.

  If a pin is in a group, you are still able to use it and form connections with it. Expand the group to gain access to the pin.

- **Oops, I missed one pin that needs to be part of that group? How do I add a pin after I already have the group?**

  Select the pin you want to add and the group pin, right-click and choose **Add pins to <name> group**.

- **I have a pin that I don't want inside the group, how do I remove it?**

  Right-click the pin and choose **Ungroup selected pins**.

- How can I better see my design on the Canvas?

  There are zoom icons in the Canvas toolbar. Use them to Zoom in, Zoom out, Zoom to fit, and Zoom selection. You can also maximize your workspace with CTRL-W.

Generating your Design

- **Ok, I'm done connecting my design, how do I 'finish' it so that I can proceed to synthesis?**

  In the Canvas toolbar, click the Generate Project icon .

- **I get a message saying it's unable to generate my SmartDesign due to errors, what do I do? What is the Design Rules Check?**

  The Design Rules Check is included in your Report View. It lists all the errors and warnings in your design, including unconnected input pins, required pin connections, configuration incompatibilities between cores, etc.

  Errors are shown with a small red stop sign and must be corrected before you can generate; warnings may be ignored.

- **What does this error mean? How do I fix it?**

  Review the Design Rules Check topic for an explanation of errors in the Design Rules Check and steps to resolve them.

- **How do I generate my firmware?**

  In the Design Flow window, expand **Handoff Design for Firmware Development** and double-click **Configure Firmware Cores**and **Export Firmware**.

Creating a New SmartDesign Component

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

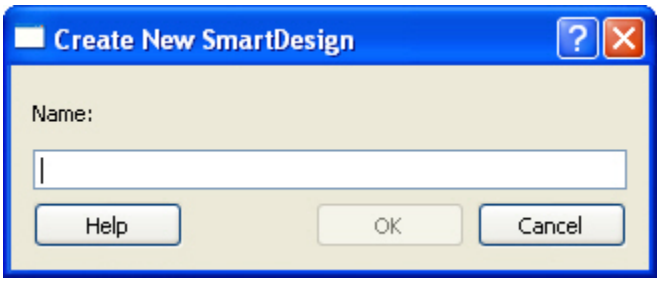


Figure 5 · Create New SmartDesign Dialog Box

2. Enter a component name and click **OK**. The component appears in the Hierarchy tab of the Design Explorer. Also, the main window displays the design Canvas.

Note: The component name must be unique in your project.

Opening an Existing SmartDesign Component

To open an existing component do one of the following:

Click the **Design Hierarchy** tab and double-click the component you want to open.

The main window displays the SmartDesign Canvas for the SmartDesign component.

Saving/Closing a SmartDesign Component

To save the current SmartDesign design component, from the **File** menu, choose **Save** <component_name>. Saving a SmartDesign component only saves the current state of the design; to generate the HDL for the design refer to Generating a SmartDesign component.

To close the current SmartDesign component without saving, from the **File** menu, choose **Close**. Select **NO** when prompted to save.

To save the active SmartDesign component with a different name use Save As. From the **File** menu choose **Save SD_<filename> As**. Enter a new name for your component and click **OK**.

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.

Generating a Datasheet (SmartFusion, IGLOOe, ProASIC3L, ProASIC3E, Fusion)

If your SmartDesign is the root design in your project, then a Memory Map / Datasheet that contains your design information is produced.

Generating Firmware and Software IDE Workspace (SmartFusion, IGLOOe, ProASIC3L, ProASIC3E, Fusion)

If your SmartDesign is the root design in your project, then any compatible firmware drivers for your peripherals are generated to <project>/firmware.

The datasheet provides all the specifics of the generated firmware drivers.

Importing a SmartDesign Component

From the **File** menu, choose **Impor**t and select the CXF file type.

Importing an existing SmartDesign component into a SmartDesign project will not automatically import the sub-components of that imported SmartDesign component.

You must import each sub-component separately.

After importing the sub-components, you must open the SmartDesign component and replace each sub-component so that it references the correct component in your project. .

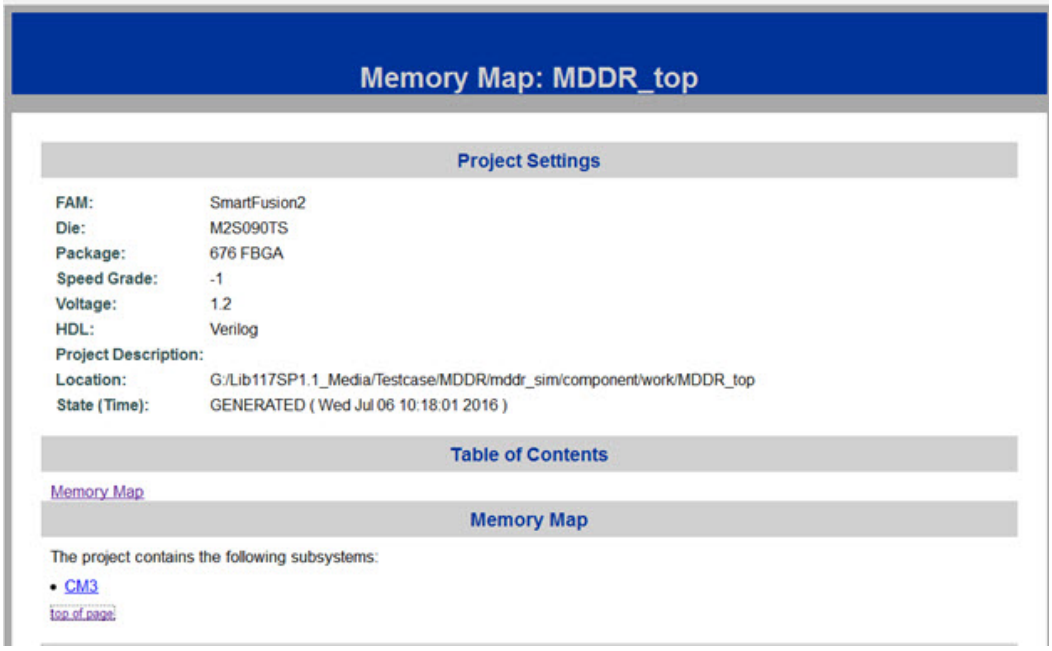
Deleting a SmartDesign Component from the Libero SoC Project

To delete a SmartDesign component from the project:

1. In the **Design Hierarchy** tab, select the SmartDesign component that you want to delete.
2. Right-click the component name and select **Delete from Project** or **Delete from Disk and Project**, or click the **Delete** key to delete from project.

Generating a Memory Map

Generate Memory Map generates a Memory Address Map for subsystems which can be accessed by the Cortex-M3 Processor or Fabric Masters in your design. It also contains Register Adddress Mapping of the SYSTEM REGISTER (SYSREG). The Memory Map Information is in XML format and put in the <design_name>_DataSheet.xml file in the Reports panel under the Generate Memory Map node.



Memory Map: MDDR_top

Project Settings

FAM:	SmartFusion2
Die:	M2S090TS
Package:	676 FBGA
Speed Grade:	-1
Voltage:	1.2
HDL:	Verilog
Project Description:	
Location:	G:/Lib117SP1.1_Media/Testcase/MDDR/mddr_sim/component/work/MDDR_top
State (Time):	GENERATED (Wed Jul 06 10:18:01 2016)

Table of Contents

Memory Map

Memory Map

The project contains the following subsystems:

- CM3

top of page

The design's memory map is determined by the connections made to the bus component. A bus component is divided into multiple slots for slave peripherals or instances to plug into. Each slot represents a different address location and range to the Master of the bus component.

Refer to this memory map for address locations if you want to access the subsystems/components from the MSS or from Fabric Masters.

CM3 Subsystem

Master(s) on this bus:

- CM3
- FABRIC2MSSFIC2

	Address Range
DDR_0_SPACE_3	0xD0000000 - 0xDFFFFFFF
DDR_0_SPACE_2	0xC0000000 - 0xCFFFFFFF
DDR_0_SPACE_1	0xB0000000 - 0xBFFFFFFF
DDR_0_SPACE_0	0xA0000000 - 0xAFFFFFFF
AHB2ENVM1_REGISTERS	0x600C0000 - 0x600FFFFF
AHB2ENVM0_REGISTERS	0x60080000 - 0x600BFFFF
ENTIRE_ENVM	0x60000000 - 0x6007FFFF
CACHE_BACKDOOR	0x40400000 - 0x4040FFFF
SYSREG : RegisterMap	0x40038000 - 0x40038FFF
CFGM	0x40020800 - 0x40020FFF
COMBLK	0x40016000 - 0x40016FFF
HDMA	0x40014000 - 0x40014FFF
H2FINTERRUPT	0x40006000 - 0x40006FFF

If your design contains standard Bus Instances such as the DirectCore AMBA bus cores, CoreAPB or CoreAHB, you can view the Memory Map Configuration of your design in the Report View. To do so, generate your top level design and click the Reports button in the toolbar.

The design's memory map is determined by the connections made to the bus component. A bus component is divided into multiple slots for slave peripherals or instances to plug into. Each slot represents a different address location and range to the Master of the bus component.

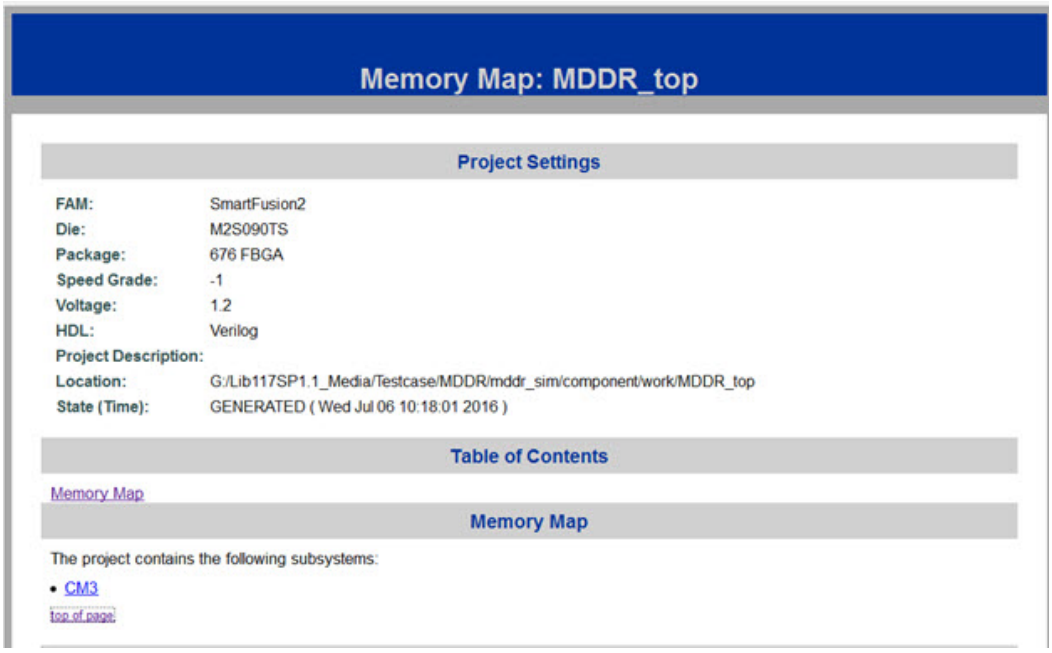
The datasheet reports the memory map of the different subsystems of your design, where a subsystem is any independent bus structure with a Master and Slave peripheral attached.

Connecting peripherals to busses can be accomplished using the normal SmartDesign connectivity options:

- **Auto-Connect -** the system creates a bus structure based on the peripherals that you have instantiated and finds compatible bus interfaces and connects them together
- The Modify Memory Map dialog box
- Canvas - Make connections between your blocks.

Your application and design requirements dictate which address location (or slots) is most suitable for your bus peripherals. For example, the memory controller should be connected to Slot0 of the CoreAHB bus because on Reset, the processor will begin code execution from the bottom of the memory map.

An example of the datasheet is shown in the figure below.



Memory Map: MDDR_top

Project Settings

FAM:	SmartFusion2
Die:	M2S090TS
Package:	676 FBGA
Speed Grade:	-1
Voltage:	1.2
HDL:	Verilog
Project Description:	
Location:	G:/Lib117SP1.1_Media/Testcase/MDDR/mddr_sim/component/work/MDDR_top
State (Time):	GENERATED (Wed Jul 06 10:18:01 2016)

Table of Contents

Memory Map

Memory Map

The project contains the following subsystems:

- CM3

top of page

Figure 6 · Example Memory Map

Modify Memory Map Dialog Box

The Modify Memory Map dialog box (shown in the figure below) enables you to connect peripherals to buses via a drop-down menu. To open the dialog box, right-click the bus instance and choose **Modify Memory Map**.

This dialog box simplifies connecting peripherals to specific base addresses on the bus. The dialog box shows all the busses in the design; select a bus in the left pane to assign or view the peripherals on a bus. Busses that are bridged to other busses are shown beneath the bus in the hierarchy.

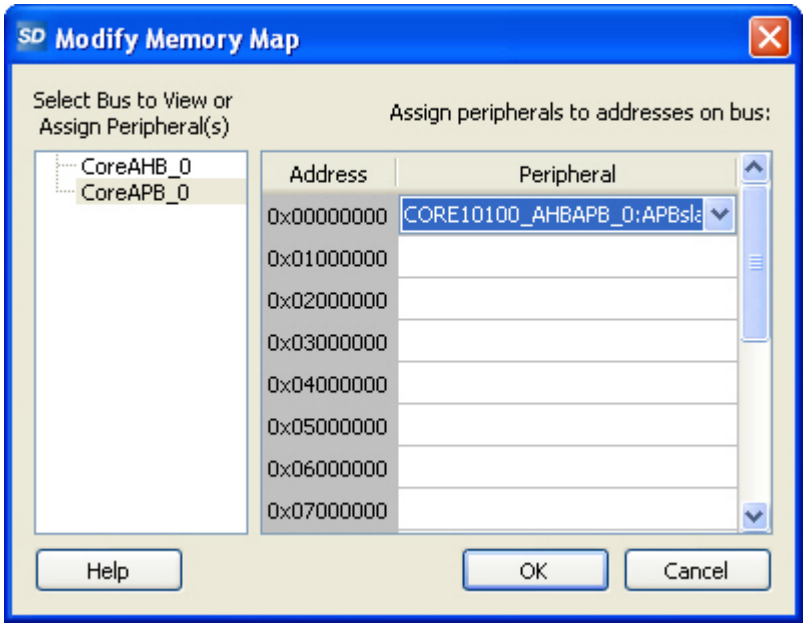


Figure 7 · Modify Memory Map Dialog Box

Click the Peripheral drop-down menu to select the peripheral you wish to assign to each address. To remove (unassign) a peripheral from an address, click the drop-down and select the empty element.

Click **OK** to create the connections between the busses and peripherals in the design.

Canvas Overview

The SmartDesign Canvas is like a whiteboard where functional blocks from various sources can be assembled and connected; interconnections between the blocks represent nets and busses in your design.

You can use the Canvas to manage connections, set attributes, add or remove components, etc. The Canvas displays all the pins for each instance (as shown in the figure below).

The Canvas enables you to drag a component from the Design Hierarchy or a core from the Catalog and add an instance of that component or core in the design. Some blocks (such as Basic Blocks) must be configured and generated before they are added to your Canvas. When you add/generate a new component it is automatically added to your Design Hierarchy.

To connect two pins on the Canvas, click the **Connection Mode button** to enable it and click and drag between the two pins you want to connect. The Connection Mode button is disabled if you attempt to illegally connect two pins.

Click the **Maximize Work Area button** to hide the other windows and show more of the Canvas. Click the button again to return the work area to the original size.

The Canvas displays bus pins with a + sign (click to expand the list) or - (click to hide list). If you add a slice on a bus the Canvas adds a + to the bus pin.

Components can be reconfigured any time by double-clicking the instance on the Canvas. You can also add bus interfaces to instances using this view. In the Canvas view, you can add graphic objects and text to your design.

Inputs and bi-directional pins are shown on the left of components, and output pins are shown on the right.

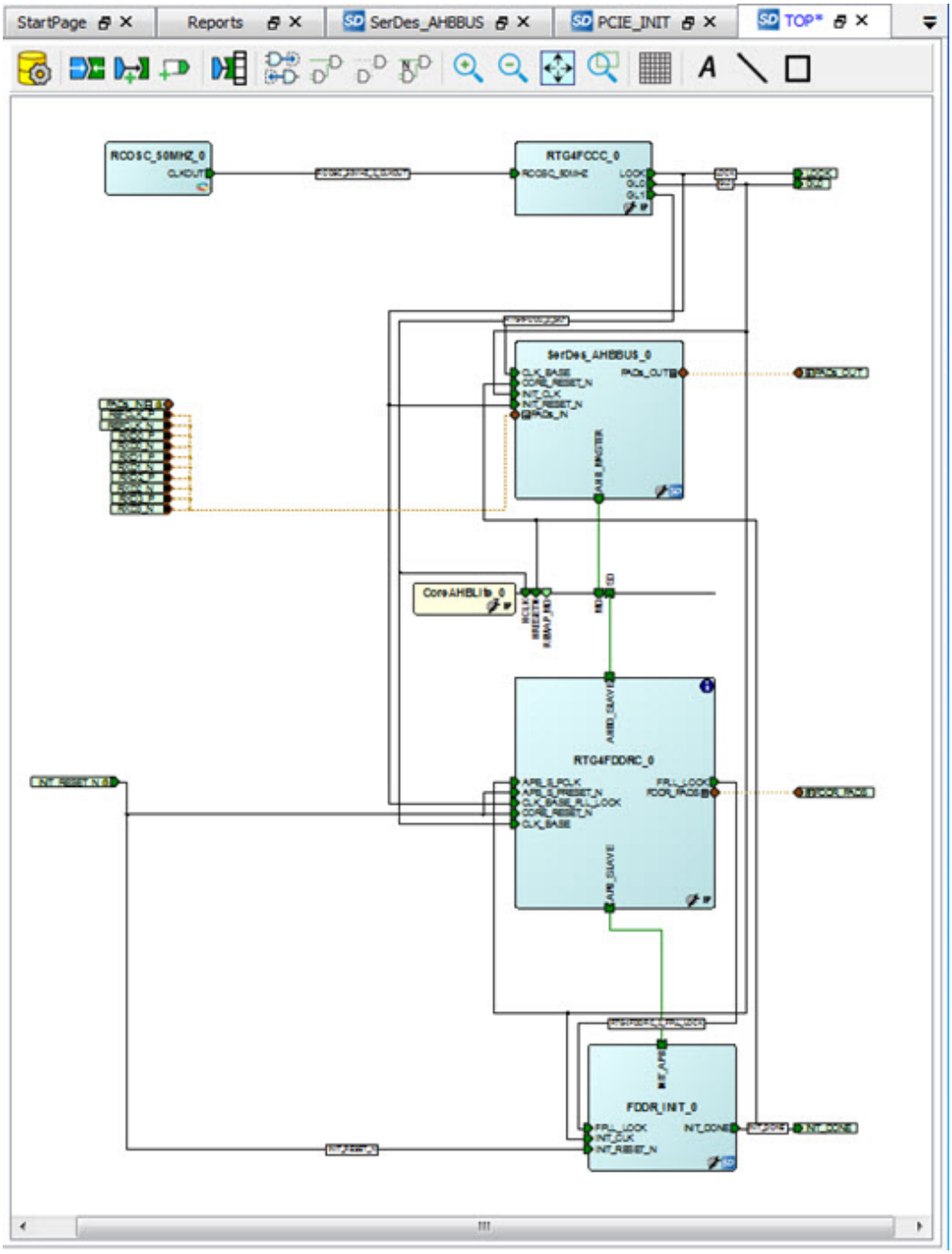


Figure 8 · SmartDesign Canvas

See Also

Canvas Icons

Displaying Connections on the Canvas

The Canvas shows the instances and pins in your design (as shown in the figure below). Right-click the Canvas and choose Show Net Names to display nets.

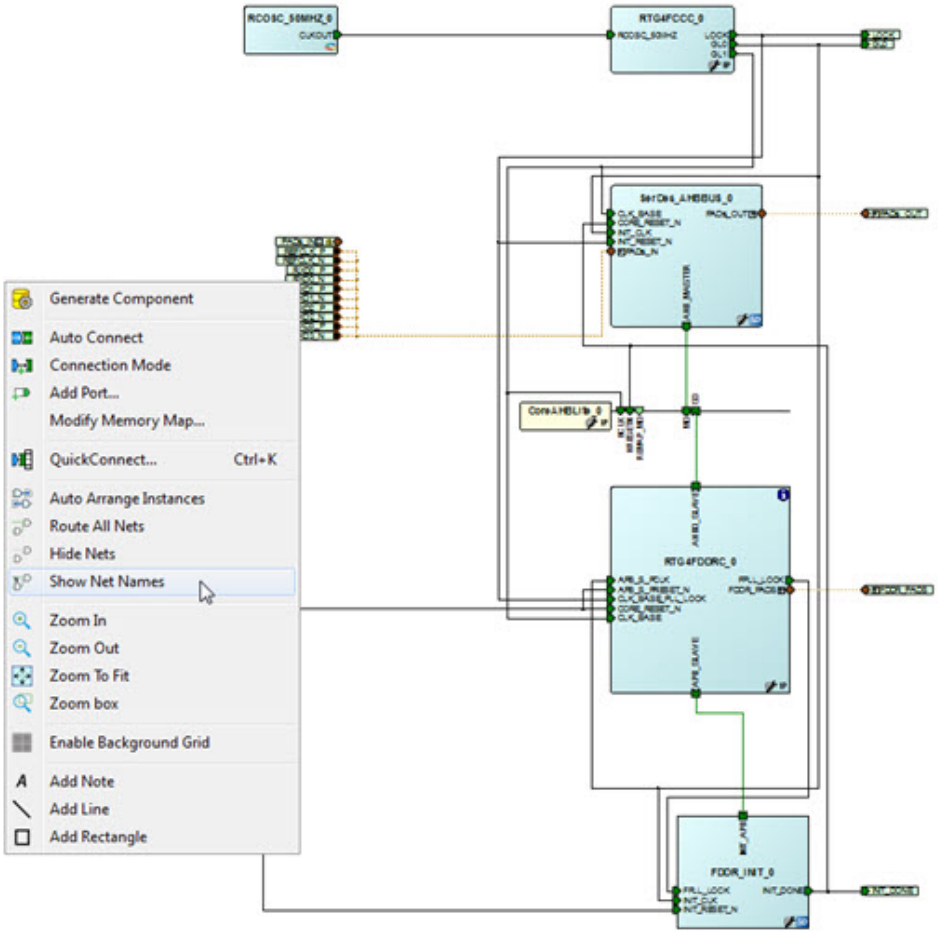


Figure 9 · Components in SmartDesign

The net names are shown in the Canvas.

Pin and Attribute Icons

Unconnected pins that do not require a connection are gray.

Unconnected pins that require a connection are red.

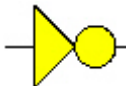
Unconnected pins that have a default tie-off are pale green.

Connected pins are green.

Right-click a pin to assign an attribute.

Pins assigned attributes are shown with an icon, as shown in the table below.

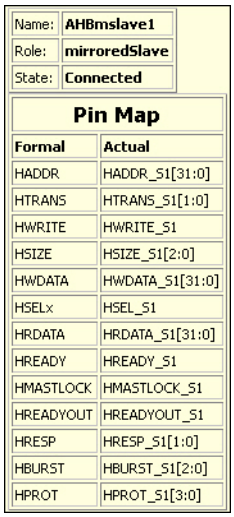
Table 1 · Pin Attribute Icons

Attribute	Icon
Tie Low	
Tie High	
Invert	
Mark as Unused	
Tie to Constant	

See the Canvas Icons reference page for definitions for each element on the Canvas.

Each connection made using a bus interface is shown in a separate connection known as a bus-interface net.

Move the mouse over a bus interface to display its details (as shown below).

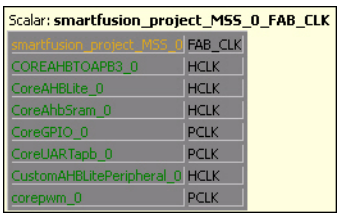


Name:	AHBmslave1
Role:	mirroredSlave
State:	Connected

Pin Map

Formal	Actual
HADDR	HADDR_S1[31:0]
HTRANS	HTRANS_S1[1:0]
HWRITE	HWRITE_S1
HSIZE	HSIZE_S1[2:0]
HWDATA	HWDATA_S1[31:0]
HSELx	HSEL_S1
HRDATA	HRDATA_S1[31:0]
HREADY	HREADY_S1
HMASTLOCK	HMASTLOCK_S1
HREADYOUT	HREADYOUT_S1
HRESP	HRESP_S1[1:0]
HBURST	HBURST_S1[2:0]
HPROT	HPROT_S1[3:0]

Hover over a bus interface net to see details (as shown below).



Making Connections Using the Canvas

Use the Canvas or Connectivity dialog box to make connections between instances.

You can use Connection Mode on the Canvas to quickly connect pins. Click the **Connection Mode** button to start, then click and drag between any two pins to connect them. Illegal connections are disabled. Click the Connection Mode button again to exit Connection Mode.

To connect two pins on the Canvas, select any two (Ctrl + click to select a pin), right-click one of the pins you selected and choose **Connect**. Illegal connections are disabled; the Connect menu option is unavailable.

Promoting Ports to Top Level

To automatically promote a port to top level, select the port, right-click, and choose **Promote To Top Level**. This automatically creates top-level ports of that name and connects the selected ports to them. If a port name already exists, a choice is given to either connect to the existing ports or to create a new port with a name <port name>_<i> where i = 1…n.

Double-click a top-level port to rename it.

Bus slices cannot be automatically promoted to top level. You must create a top level port of the bus slice width and then manually connect the bus slice to the newly created top level port.

Tying Off Input Pins

To tie off ports, select the port, right-click and choose **Tie High** or **Tie Low**.

Tying to Constant

To tie off bus ports to a constant value, select the port, right-click and choose **Tie to a Constant**. A dialog appears (as shown in the figure below) and enables you to specify a hex value for the bus.

To remove the constant, right-click the pin and choose **Clear Attribute** or **Disconnect**.

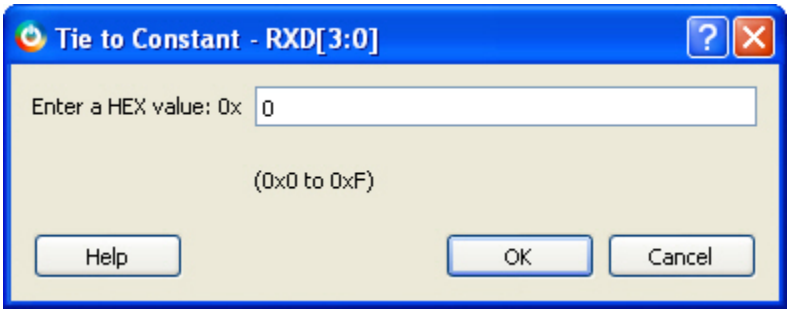


Figure 10 · Tie to Constant Dialog Box

Making Driver and Bus Interface Pins Unused

Driver or bus interface pins can be marked unused (floating/dangling) if you do not intend to use them as a driver in the design. If you mark a pin as unused the Design Rules Check does not return Floating Driver or Unconnected Bus Interface messages on the pin.

Once a pin is explicitly marked as unused it cannot be used to drive any inputs. The unused attribute must be explicitly removed from the pin in order to connect it later. To mark a driver or bus interface pin as unused, right-click the driver or bus interface pin and choose **Mark as Unused**.

See Also

Show/Hide Bus Interface Pins

Simplifying the Display of Pins on an Instance using Pin Groups

The Canvas enables you to group and ungroup pins on a single instance to simplify the display. This feature is useful when you have many pins in an instance, or if you want to group pins at the top level. Pin groups are cosmetic and affect only the Canvas view; other SmartDesign views and the underlying design are not affected by the pin groups.

Grouping pins enables you to:

- Hide pins that you have already connected
- Hide pins that you intend to work on later
- Group pins with similar functionality
- Group unused pins
- Promote several pins to Top Level at once

To group pins:

1. Ctrl + click to select the pins you wish to group. If you try to click-and-drag inside the instance you will move the instance on the Canvas instead of selecting pins.
2. Right-click and choose **Add pins to group** to create a group. Click + to expand a group. The icon associated with the group indicates if the pins are connected, partially connected, or unconnected (as shown in the figure below).

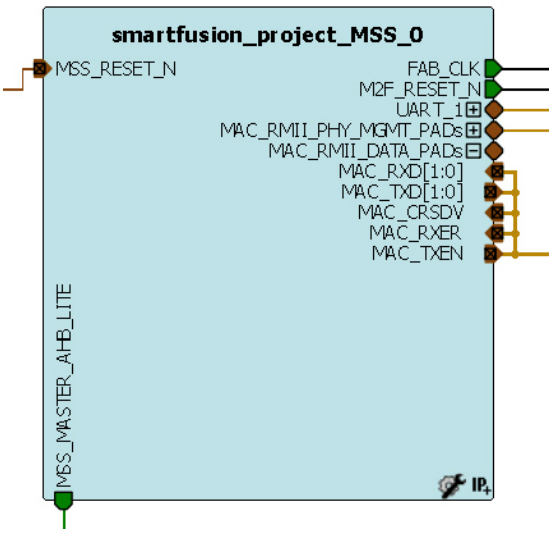


Figure 11 · Groups in an Instance on the Canvas

To add a pin to a group, Ctrl + click to select both the pin and the group, right-click and choose **Add pin to group**.

To name a group:

To name a group, right-click the port name and choose **Rename Group**.

To ungroup pins:

1. Click + to expand the group.
2. Right-click the pin you wish to remove from the group and choose **Ungroup selected pins**. Ctrl + click to select and remove more than one pin in a group.

A group remains in your instance after you remove all the pins. It has no effect on the instance; you can leave it if you wish to add pins to the group later, or you can right-click the group and choose **Delete Group** to remove it from your instance.

If you delete a group from your instance any pins still in the group are unaffected.

To promote a group to Top level:

1. Create a group of pins.
2. Right-click the group and choose **Promote to Top Level**.

Bus Instances

Bus Components in the Core Catalog, such as CoreAHB or CoreAPB, implement an on-chip bus fabric. When these components are instantiated into your canvas they are displayed as horizontal or vertical lines. Double-click the bus interfaces of your component to edit the connections.

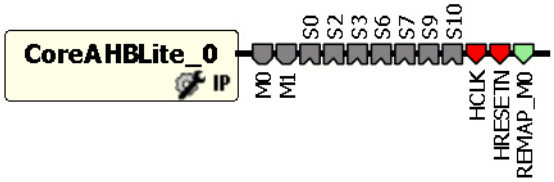


Figure 12 · Bus Instance in SmartDesign

Adding Graphic Objects

You can document your design by adding comments and notations directly on the Canvas.

The Canvas toolbar enables you to add and modify decorative graphic objects, such as shapes, labels and lines on the Canvas.

Adding and Deleting Lines and Shapes

To add a line or a shape:

1. Select the line or shape button.
2. Click, drag and release on the Canvas. The table below provides a description of each button.

Button	Description
	Line

Button	Description
☐	Rectangle

Note: Hold the Shift key to constrain line and arrow to 45 degree increments or constrain the proportions of the rectangle (square).

To change the line and fill properties:

1. Select the element(s), right-click it, and choose **Properties**.
   - Select **Line** to modify the color, style and width of the line.
   - Select **Fill** to modify the crosshatch and the foreground and background colors.
2. Click **OK**.

To delete a line or shape, select the object and press Delete.

Adding Text

To add text, select the text tool and click the Canvas to create a text box. To modify the text, double-click the text box and then type.

To modify the text box properties:

1. Select the text box, right-click it, and choose **Properties**.
   - Select **Text** to modify the text alignment.
   - Select **Line** to modify the color, style and width of the line.
   - Select **Fill** to modify the crosshatch and the foreground and background colors.
   - Select **Font** to modify the font properties.
2. Click **OK**.

Editing Properties for Graphic Objects on the Canvas

Right-click any graphic object to update properties, such as Fill, and Line properties for shapes and lines, or Font options for text properties.

Auto-Arranging Instances

Right-click the Canvas and choose **Auto Arrange Instances** from the right-click menu to auto-arrange the instances on the Canvas.

Locking Instance and Top Level Port Positions

You can lock the placement of instances on the Canvas. Right-click the instance or Top-level port and choose **Lock** to lock the placement. When you lock placement you can click and drag to move the instance manually but the Auto Arrange Instances menu option has no effect on the instance.

To unlock an instance, right-click the instance and choose **Unlock**.

Right-click a top level port and choose **Unlock Position** to return it to its default position.

See Also

Bus Instances

Simplifying the Display of Pins on an Instance using Pin Groups

Replace Component for Instance

You can use the Replace Component for Instance dialog box (shown in the figure below) to restore or update version instances on your Canvas without creating a new instance and losing your connections.

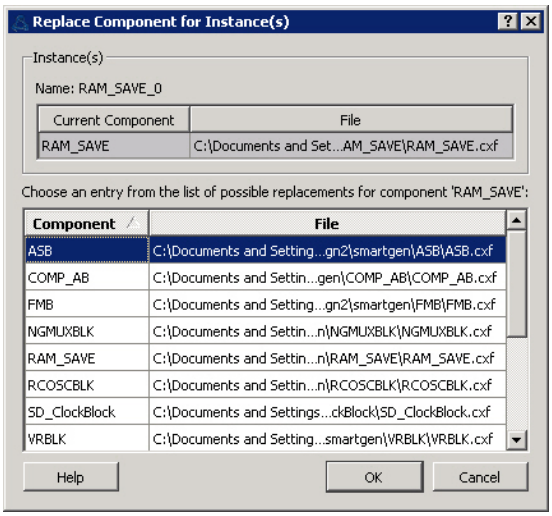


Figure 13 · Replace Component for Instance Dialog Box

To change the version of an instance:

1. From the right-click menu choose **Replace Component for Instance**. The Replace Component for Instance dialog box appears.
2. Select a component and choose a new version from the list. Click **OK**.

Replace Instance Version

The Replace Instance Version dialog box enables you to replace an IP instance with another version. You can restore or replace your IP instance without creating a new instance or losing your connections.

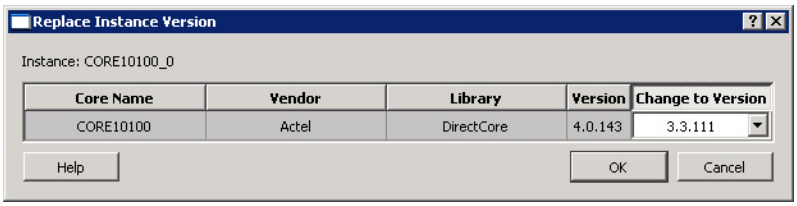


Figure 14 · Replace Instance Version Dialog Box

To replace an instance version:

1. Right click any IP instance and choose **Replace Instance Version**. The dialog box appears.
2. Choose the version you wish to use from the **Change to Version** dropdown menu (as shown in the figure above) and click OK to continue.

Slicing

Bus ports can be sliced or split using Slicing. Once a slice is created, other bus ports or slices of compatible size can be connected to it.

The Edit Slices dialog box enables you to automatically create bus slices of a specified width.

To create a slice:

1. Select a bus port, right-click, and choose **Edit Slice.** This brings up the **Edit Slices** dialog box (see figure below).

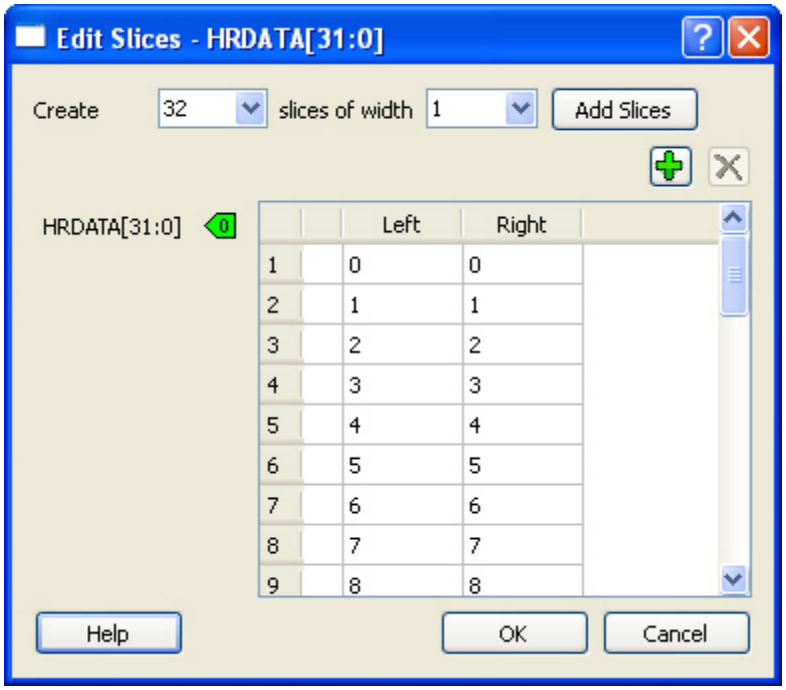


Figure 15 · Edit Slices Dialog Box

2. Enter the parameters for the slice and click **Add Slices**. You can also create individual slices and specify their bus dimensions manually.
3. Click **OK** to continue.

Note: Overlapping slices cannot be created for IN and INOUT ports on instances or top-level OUT ports.

To remove a slice, select the slice, right-click, and choose **Delete Slice.**

Rename Net

To rename a net:

1. Right-click the net on the Canvas and choose **Rename Net**. This opens the Rename Net dialog box.
2. Type in a new name for the net.

Note: The system automatically assigns net names to nets if they are not explicitly specified. Once you have specified a name for a net, that name will not be over-written by the system.

Automatic Names of Nets

Nets are automatically assigned names by the tool according to the following rules:

In order of priority

1. If user named then name = user name
2. If net is connected to top-level port then name = port name; if connected to multiple ports then pick first port

3. If the net has no driver, then name = net_[i]
4. If the net has a driver, name = instanceName_driverpinName

Slices

For slices, name = instanceName_driverpinName_sliceRange; for example u0_out1_4to6.

GND and VCC Nets

The default name for GND/VCC nets is net_GND and net_VCC.

Expanded Nets for Bus Interface Connections

Expanded nets for bus interface connections are named busInterfaceNetName_<i>_driverPinName.

Organizing Your Design on the Canvas

You may find it easier to create and navigate your SmartDesign if you organize and label the instances and busses on the Canvas.

You can show and hide nets, lock instances, rotate busses, group and ungroup pins, rename instances / groups / pins, and auto-arrange instances.

To organize your design:

1. Right-click the Canvas and choose **Auto Arrange Instances** from to automatically arrange instances. SmartDesign's auto-arrange feature optimizes instance location according to connections and instance size.
2. Right-click any instance and choose Lock Location to fix the placement. Auto-Arrange will not move any instances that are locked.
3. Click Auto-Arrange again to further organize any unlocked instances. Continue arranging and locking your instances until you are satisfied with the layout on the Canvas.

If your design becomes cluttered, group your pins. It may help to group pins that are functionally similar, or to group pins that are already connected or will be unused in your design.

To further customize your design's appearance:

Double-click the names of instances to add custom names. For example, it may be useful to rename an instance based on a value you have set in the instance: the purpose of an instance named 'array_adder_bus_width_5' is easier to remember than 'array_adder_0'.

Renaming a Component

To rename a component, right-click the component inside the Design Hierarchy and select Rename Component. Enter a new name for the Component Name.

Adding Components and Modules (Instantiating)

SmartDesign components, Design Block cores, IP cores, and HDL modules are displayed in the Design Hierarchy and Files tabs.

To add a component, do either of the following:

- Select the component in the Design Hierarchy tab or Catalog and drag it to the Canvas.
- Right-click a component in the Design Hierarchy tab or Catalog and choose **Instantiate in <SmartDesign name>.**

The component is instantiated in the design.

SmartDesign creates a default instance name. To rename the instance, double-click the instance name in the Canvas.

Adding a SmartDesign Component

SmartDesign components can be instantiated into another SmartDesign component.

Once a SmartDesign is generated, the exported netlist can be instantiated into HDL like any other HDL module.

Note: HDL modules with syntax errors cannot be instantiated in SmartDesign. However, since SmartDesign requires only the port definitions, the logic causing syntax errors can be temporarily commented out to allow instantiation of the component.

Adding or Modifying Top Level Ports

You can add ports to, and/or rename ports in your SmartDesign.

Add Prefixes to Bus Interface / Group Names on Top-level Ports:

Bus Interfaces and Groups are composed of other ports. On the top level, you can add prefixes to the group or bus interface port name to the sub-port names. To do so, right-click the group or bus interface port and choose **Prefix <name> to Port Names**.

Adding/Renaming Ports

To add ports:

1. From the **SmartDesign** menu, choose **Add port**. The Add Port dialog box appears (as shown below).

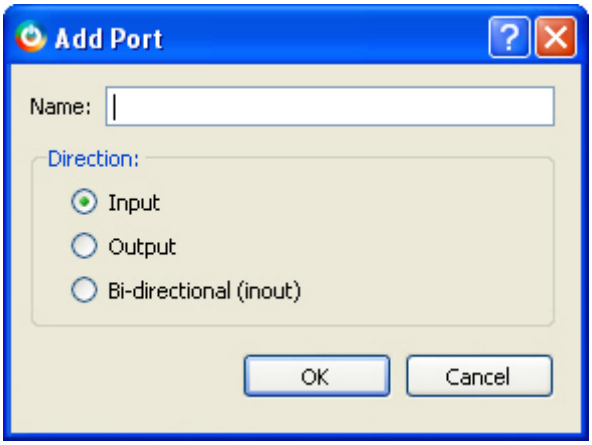


Figure 16 · Add New Port Dialog Box

2. Specify the name of the port you wish to add. You can specify a bus port by indicating the bus width directly into the name using brackets [ ], such as mybus[3:0].
3. Select the direction of the port.

To remove a port from the top level, right-click the port and choose **Delete Top Level Port**.

Modify Port

To rename a top-level port, right-click the top-level port and choose **Modify Top Level Port**. You can rename the port, change the bus width (if the port is a bus), and change the port direction.

Right-click a top-level port and choose **Modify Port** to change the name and/or direction (if available).

See Also

Top Level Connections

Automatic Connections

Using automatic connections (as shown in the figure below) enables the software to connect your design efficiently, reducing time required for manual connections and the possibility of introducing errors.

Auto Connect also constructs your bus structure if you have a processor with peripherals instantiated. Based on the type of processor and peripherals, the proper busses and bridges are added to your design.

To auto connect the bus interfaces in your design, right-click the design Canvas and select **Auto Connect**, or from the **SmartDesign** menu, choose **Auto Connect**.

SmartDesign searches your design and connects all compatible bus interfaces.

SmartDesign will also form known connections for any SoC systems such as the processor CLK and RESET signals.

If there are multiple potential interfaces for a particular bus interface, Auto Connect will not attempt to make a connection; you must connect manually. You can use the Canvas to make the manual connections.

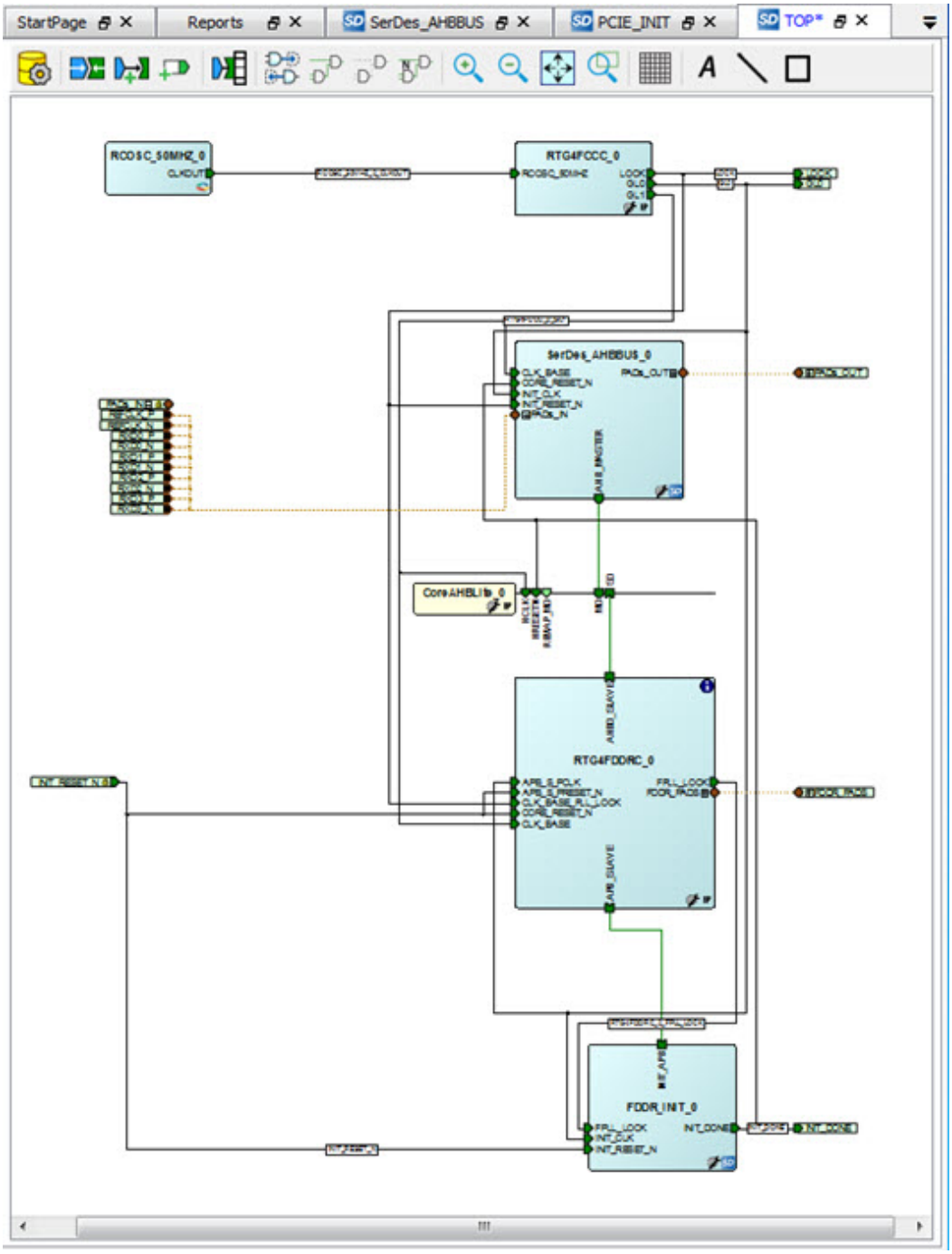


Figure 17 · Auto-Connected Cores

QuickConnect

The QuickConnect dialog box enables you to make connections in your design without using the Canvas. It is useful if you have a large design and know the names of the pins you wish to connect. Connections are reflected in the Canvas as you make them in the dialog box; error messages appear in the Log window immediately. It may be useful to resize the QuickConnect dialog box so that you can view the Log window or Canvas while you make connections.

To connect pins using QuickConnect:

1. Find the **Instance Pin** you want to connect and click to select it.
2. In **Pins to Connect**, find the pin you wish to connect, right-click and choose **Connect**. If necessary, use the **Search** field to narrow down the list of pins displayed in Pins to Connect.

   Note that if the connection is invalid then Connect is grayed out.

If you wish to invert or tie a pin high, low or Mark Unused:

1. Find the **Instance Pin** you want to invert or tie high/low
2. Right-click **Connection** and choose **Invert**, **Tie High**, **Tie Low** or **Mark Unused**.

If you wish to promote a pin to the top level of your design:

1. Find the Instance Pin you want to promote.
2. Right-click the pin and choose **Promote to Top**.

You can perform all connectivity actions that are available in the Canvas, including: slicing bus pins, tying bus pins to a constant value, exposing pins from a bus interface pins and disconnecting pins. All actions are accessible from the right-click context menu on the pin.

Instance Pins lists all the available instance pins in your design and their connection (if any). Use the drop-down list to view only unconnected pins, or to view the pins and connections for specific elements in your design.

Pins to Connect lists the instances and pins in your design. Use the Search field to find a specific instance or pin. The default wildcard search is '*.*'. Wildcard searches for CLK pins (*.*C*L*K) and RESET pins (*.*R*S*T) are also included.

Here are some of the sample searches that you may find useful:

- *UART*.*: show all pins of any instances that contain UART in the name
- MyUART_0.*: only show the pins of the “MyUART_0” instance
- *.p: show all pins in the design that contain the letter ‘p’

Double-click an instance in Pins to Connect to expand or collapse it.

The pin letters and icons in the QuickConnect dialog box are the same as the Canvas icons and communicate information about the pin. Inputs, Outputs and I/Os are indicted by I, O, and I/O, respectively.

Additional information is communicated by the color:

- Red - Mandatory connection, unconnected
- Green - Connected
- Grey - Optional, unconnected pin
- Brown - Pad
- Light Green - Connected to a default connection on generation
- Blue - Driver pin

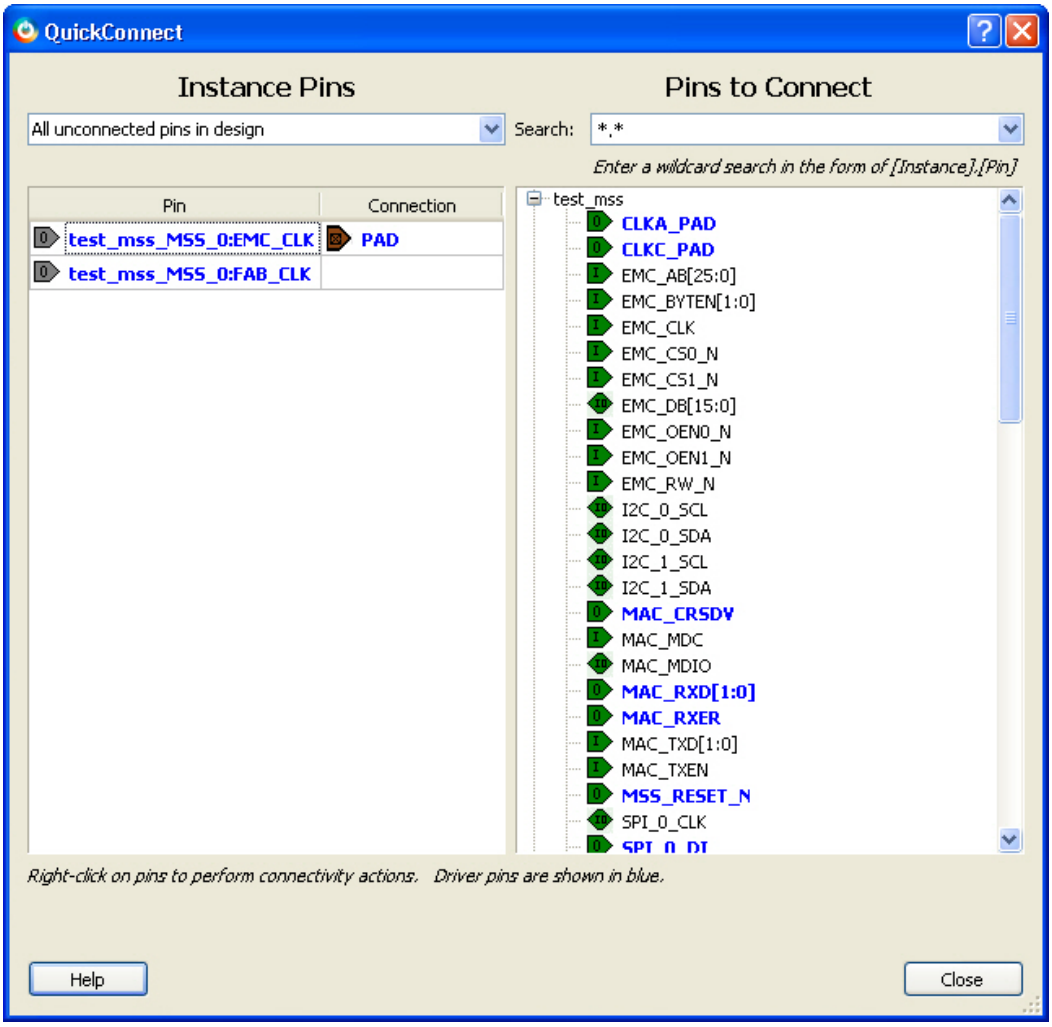


Figure 18 · QuickConnect Dialog Box

Manual Connections

You can use Connection Mode to click and drag and connect pins. Click the Connection Mode button to toggle it, and click and drag between any two pins to connect them. Illegal connections will not be allowed.

To make manual connections between to pins on the Canvas, select both pins (use CTRL + click), right-click either pin and choose **Connect**. If the pins cannot be legally connected the connection will fail.

Deleting Connections

To delete a net connection on the Canvas, click to select the net and press the Delete key, or right-click and choose **Delete**.

To remove all connections from one or more instances on the Canvas, select the instances on the Canvas, right-click and choose **Clear all Connections**. This disconnects all connections that can be disconnected legally.

Certain connections to ports with PAD properties cannot be disconnected. PAD ports must be connected to a design's top level port. PAD ports will eventually be assigned to a package pin. In SmartDesign, these ports are automatically promoted to the top level and cannot be modified or disconnected.

Top-Level Connections

Connections between instances of your design normally require an OUTPUT (Driver Pin) on one instance to one or more INPUT(s) on other instances. This is the basic connection rule that is applied when connecting.

However, directions of ports at the top level are specified from an external viewpoint of that module. For example, an INPUT on the top level is actually sending ('driving') signals to instances of components in your design. An OUTPUT on the top level is receiving ('sinking') data from a Driver Pin on an internal component instance in your SmartDesign design.

The implied direction is essentially reversed at the top-level. Making connections from an OUTPUT of a component instance to an OUTPUT of top-level is legal.

This same concept applies for bus interfaces; with normal instance to instance connections, a MASTER drives a SLAVE interface. However, they go through a similar reversal on the top-level.

About Bus Interfaces

A bus interface is a standard mechanism for specifying the interconnect rules between components or instances in a design. A bus definition consists of the roles, signals, and rules that define that bus type. A bus interface is the instantiation of that bus definition onto a component or instance.

The available roles of a bus definition are master, slave, and system.

A master is the bus interface that initiates a transaction (such as read or write) on a bus.

A slave is the bus interface that terminates/consumes a transaction initiated by a master interface.

A system is the bus interface that does not have a simple input/output relationship on both master/slave. This could include signals that only drive the master interface, or only drive the slave interface, or drive both the master and slave interfaces. A bus definition can have zero or more system roles. Each system role is further defined by a group name. For example, you may have a system role for your arbitration logic, and another for your clock and reset signals.

Mirror roles are for bus interfaces that are on a bus core, such as CoreAHB or CoreAPB. They are equivalent in signal definition to their respective non-mirror version except that the signal directions are reversed.

The diagram below is a conceptual view of a bus definition.

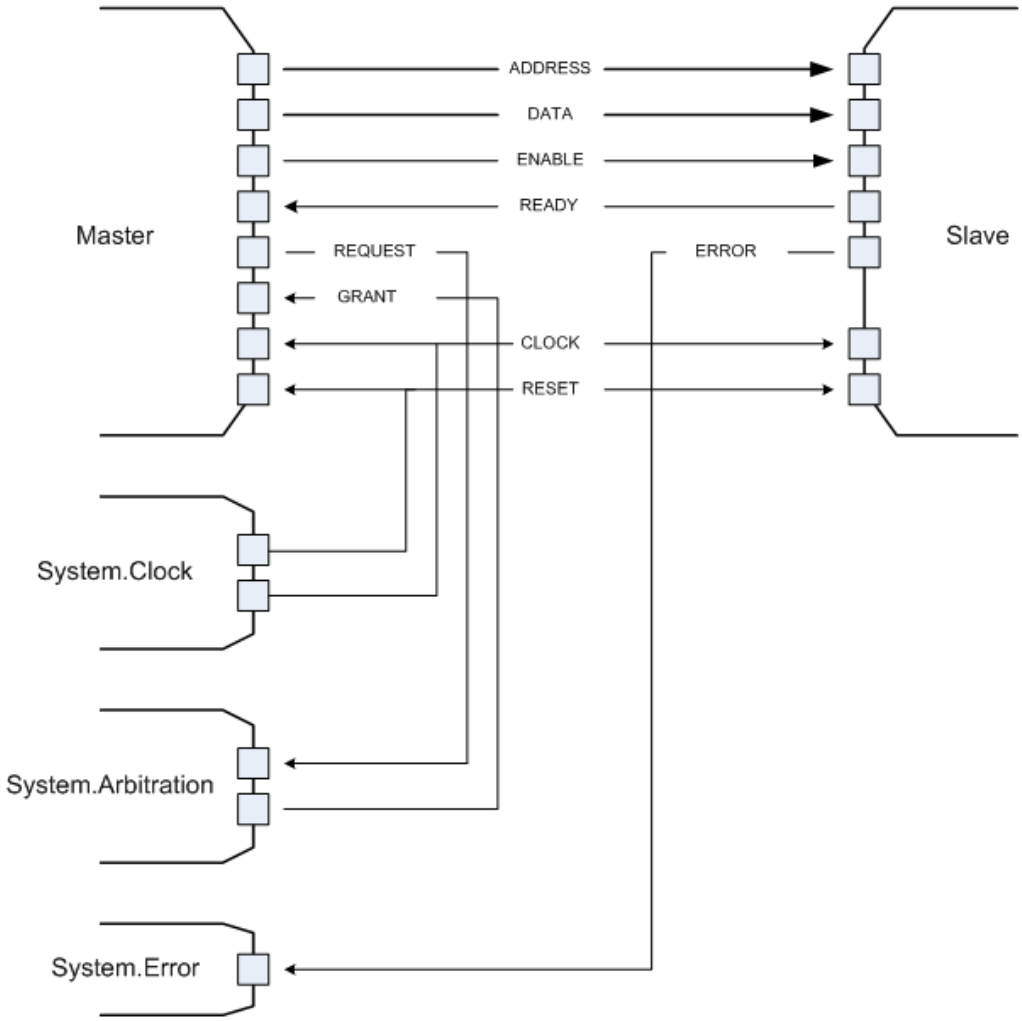


Figure 19 · Bus Definition

See Also:

Using bus interfaces in SmartDesign

Using Bus Interfaces in SmartDesign

Adding bus interfaces to your design enables SmartDesign to do the following:

- Auto connect compatible interfaces
- Enforce DRC rules between instances in your design
- Search for compatible components in the project

The Catalog in the Project Manager contains a list of Microsemi SoC-specific and industry standard bus definitions, such as AMBA.

You can add bus interfaces to your design by dragging the bus definitions from the Bus Interfaces tab in the Catalog onto your instances inside SmartDesign.

SmartDesign supports automatic creation of data driven configurators based on HDL generics/parameters.

If your block has all the necessary signals to interface with the AMBA bus protocol (ex: address, data, control signals):

1. Right-click your custom HDL block and choose **Create Core from HDL**. The Libero SoC creates your core and asks if you want to add bus interfaces.
2. Click **Yes** to open the Edit Core Definition dialog box and add bus interfaces. Add the bus interfaces as necessary.
3. Click **OK** to continue.

Now your instance has a proper AMBA bus interface on it. You can manually connect it to the bus or let Auto Connect find a compatible connection.

Some cores have bus interfaces that are instantiated during generation.

Certain bus definitions cannot be instantiated by a user. Typically these are the bus definitions that define a hardwired connection and are specifically tied to a core/macro. They are still available in the catalog for you to view their properties, but you will not be able to add them onto your own instances or components. These bus definitions are grayed out in the Catalog.

A hardwired connection is a required silicon interconnect that must be present and specifically tied to a core/macro. For example, when using the Real Time Counter in a Fusion design you must also connect it to a Crystal Oscillator core.

Maximum masters allowed - Indicates how many masters are allowed on the bus.

Maximum slaves allowed - Indicates how many slaves are allowed on the bus.

Default value - indicates the value that the input signal will be tied to if unused. See Default tie-offs with bus interfaces.

Required connection - Indicates if this bus interface must be connected for a legal design.

Adding or Modifying Bus Interfaces in SmartDesign

SmartDesign supports automatic creation of data driven configurators based on HDL generics/parameters. You can add a bus interface from your HDL module or you can add it from the Catalog.

To add a bus interface using your custom HDL block:

If your block has all the necessary signals to interface with the AMBA bus protocol (such as address, data, and control signals):

1. Right-click your custom HDL block and choose **Create Core from HDL**. The Libero SoC creates your core and asks if you want to add bus interfaces.
2. Click **Yes** to open the Edit Core Definition dialog box and add bus interfaces. Add the bus interfaces as necessary.
3. Click **OK** to continue.

Now your instance has a proper AMBA bus interface on it. You can manually connect it to the bus or let Auto Connect find a compatible connection.

To add (or modify) a bus interface to your Component:

1. Right-click your Component and choose **Edit Core Definition**. The Edit Core Definition dialog box opens, as shown in the figure below.

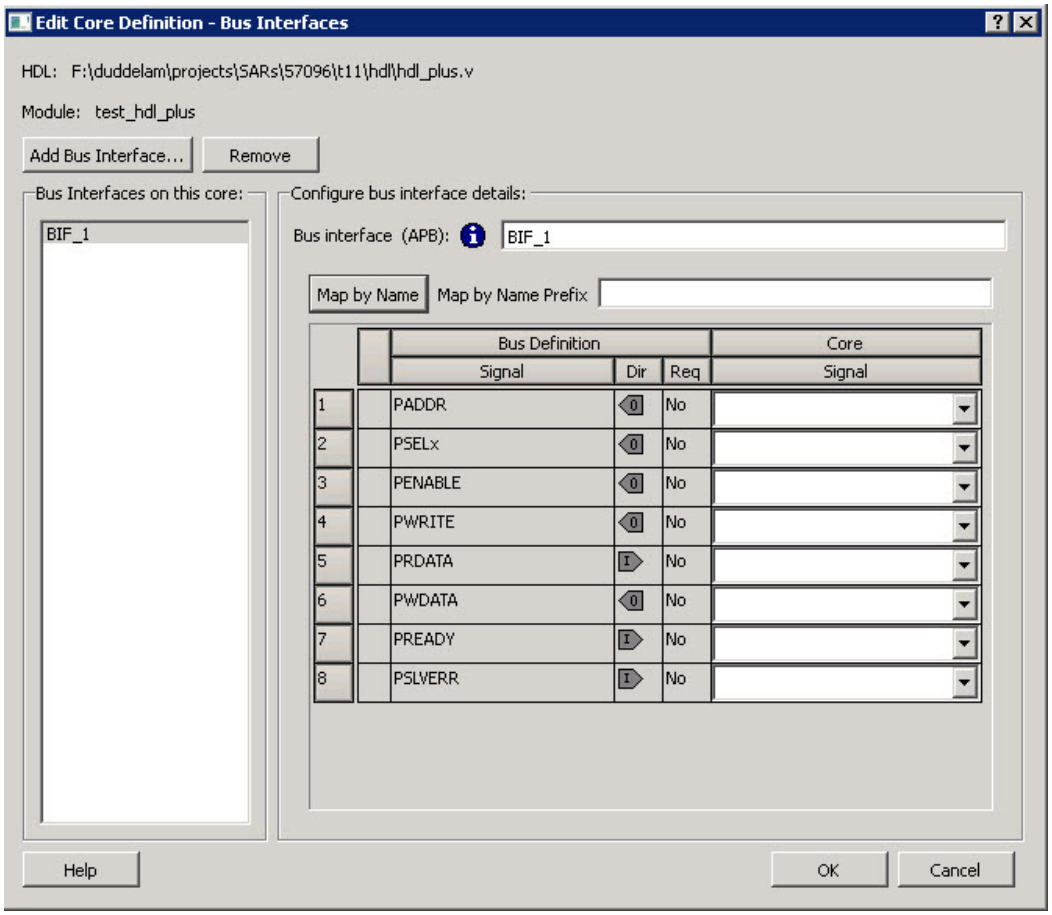


Figure 20 · Edit Core Definition Dialog Box

2. Click **Add Bus Interface**. Select the bus interface you wish to add and click **OK**.
3. If necessary, edit the bus interface details.
4. Click **Map by Name** to map the signals automatically. Map By Name attempts to map any similar signal names between the bus definition and pin names on the instance. During mapping, bus definition signal names are prefixed with text entered in the **Map by Name Prefix** field.
5. Click **OK** to continue.

Bus Interface Details

Bus Interface: Name of bus interface. Edit as necessary.

Bus Definition: Specifies the name of the bus interface.

Role: Identifies the bus role (master or slave).

Vendor: Identifies the vendor for the bus interface.

Version: Identifies the version for the bus interface.

Configuration Parameters

Certain bus definitions contain user configurable parameters.

Parameter: Specifies the parameter name.

Value: Specifies the value you define for the parameter.

Consistent: Specifies whether a compatible bus interface must have the same value for this bus parameter. If the bus interface has a different value for any parameters that are marked with consistent set to **yes**, this bus interface will not be connectable.

Signal Map Definition

The signal map of the bus interface specifies the pins on the instance that correspond to the bus definition signals. The bus definition signals are shown on the left, under the **Bus Interface Definition**. This information includes the name, direction and required properties of the signal.

The pins for your instance are shown in the columns under the Component Instance. The signal element is a drop-down list of the pins that can be mapped for that definition signal. .

If the Req field of the signal definition is Yes, you must map it to a pin on your instance for this bus interface to be considered legal. If it is No, you can leave it unmapped.

Bus Interfaces

When you add a bus interface the Edit Core Definition dialog box provides the following Microsemi SoC-specific bus interfaces:

ExtSeqCtrl

This bus interface defines the set of signals required to interface to the Analog System External Sequence Control. If the Analog System is configured with more than a single procedure, it will export this bus interface. Your own logic would need to connect to this bus interface to properly communicate and control the sequencer.

RTCXTL

This bus interface represents the hardwired connection needed between the Real Time Counter and the Crystal Oscillator.

RTCVR

This bus interface represents the hardwired connection needed between the Real Time Counter and the Voltage Regulator Power Supply Monitor.

InitCfg

This is the initialization and configuration interface that is generated as part of the Flash Memory Builder. Any clients can be initialized from the Flash Memory as long as it can connect to this bus interface. This is for pure initialization clients that do not require save-back to the Flash Memory.

InitCfgSave

This is the initialization and configuration interface that is generated as part of the Flash Memory Builder. Any client can be initialized or saved-back to the Flash Memory as long as it can connect to this bus interface. This is for clients that require initialization and save-back capabilities to the Flash Memory.

InitCfgCtrl

This interface is used to initiate the save-back procedure of the Flash Memory.

InitCfgAnalog

This interface is required between the Flash Memory System and the Analog System core.

FlashDirect

This bus interface defines the set of signals that are required to interface directly to the Flash Memory. From the Flash Memory, if you add a data storage client, this interface will be exported. Interfacing to this interface enables direct access to the Flash Memory.

XTLOscClk

This interface represents the Crystal Oscillator clock.

RCOscClk

This interface represents the RC Oscillator clock.

DirectCore Bus Interfaces

When you add a bus interface the Edit Core Definition dialog box provides the following DirectCore bus interfaces.

AHB

The AMBA AHB defines the set of signals for a component to connect to an AMBA AHB or AHBLite bus. The bus interface that is defined in the system is a superset of the signals in the AHB and AHBLite protocol. You can use the AHB bus interface in the bus definition catalog to connect your module to an AHB or AHBLite bus.

APB

The AMBA APB defines the set of signals for a component to connect to an AMBA APB or APB3 bus. The bus interface that is defined in the system is a superset of the signals in the APB and APB3 protocol. You can use the APB bus interface in the bus definition catalog to connect your module to an APB or APB3 bus.

SysInterface

The SysInterface is the interface used between the CoreMP7 and CoreMP7Bridge cores.

DBGInterface

This is the set of debug ports on the CoreMP7 core.

CPInterface

This is the co-processor interface on the CoreMP7 core.

Show/Hide Bus Interface Pins

Pins that are contained as part of a bus interface will automatically be filtered out of the display. These ports are considered to be connected and used as part of a bus interface.

However, there are situations where you may wish to use the ports that are part of the bus interface as an individual port, in this situation you can choose to expose the pin from the bus interface.

To Show/Hide pins in a Bus Interface:

1. Select a bus interface port, right-click, and choose **Show/Hide BIF Pins**. The Show/Hide Pins to Expose dialog box appears (as shown below).

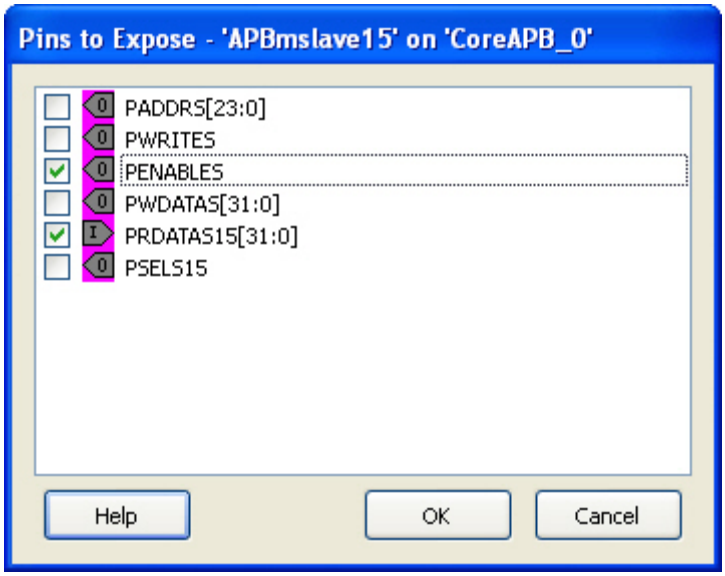


Figure 21 · Expose Driver Pin Dialog Box

2. Click the checkbox associated with the driver pin you want to show. Once the port is shown it appears on the Canvas and is available for individual connection.

   Note: If you have already connected the bus interface pin, then you will not be able to expose the non-driver pins. They will be shown grayed out in the dialog. This is to prevent the pin from being driven by two different sources.

To un-expose a driver pin, right-click the exposed port and choose **Show/Hide BIF Pins** and de-select the pin.

Default Tie-offs with Bus Interfaces

Bus definitions can contain default values for each of the defined signals. These default values specify what the signal should be tied to if it is mapped to an unconnected input pin on the instance.

Bus definitions are specified as required connection vs optional connection that defines the behavior of tie-offs during SmartDesign generation.

Required bus interfaces - The signals that are not required to be mapped will be tied off if they are mapped to an unconnected input pin.

Optional bus interfaces - All signals will be tied off if they are mapped to an unconnected input pin.

Tying Off (Disabling) Unused Bus Interfaces

Tying off (disabling) a bus interface sets all the input signals of the bus interface to the default value.

To tie off a bus interface, right-click the bus interface and select Tie Off.

This is useful if your core includes a bus interface you plan to use at a later time. You can tie off the bus interface and it will be disabled in your design until you manually set one of the inputs.

Some bus interfaces are required; you cannot tie off a bus interface that is required. For example, the Crystal Oscillator to RTC (RTCXTL) bus interface is a silicon interface and must be connected.

To enable your pin, right-click the pin and choose **Clear Attribute**.

Required vs. Optional Bus Interfaces

A required bus interface means that it must be connected for the design to be considered legal. These are typically used to designate the silicon interconnects that must be present between certain cores.

An optional bus interface means that your design is still considered legal if it is left unconnected. However, it may not functionally behave correctly.

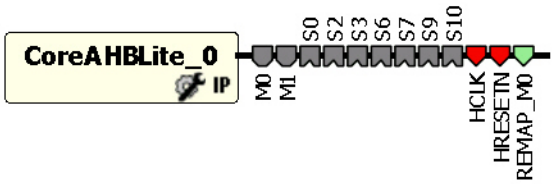


Figure 22 · Required Unconnected, Optional Unconnected, and Connected Bus Interfaces

See Also

Canvas icons

Promoting Bus Interfaces to Top-level

To automatically connect a bus interface to a top-level port, select the bus interface, right-click, and choose **Promote To Top Level**.

This automatically creates a top-level bus interface port of that name and connects the selected port to it. If a bus interface port name already exists, a choice is given to either connect to the existing bus interface port or to create a new bus interface port with a name <port name>_<i> where i = 1...n.

The signals that comprise the bus interface are also promoted.

Promoting a bus interface is a shortcut for creating a top-level port and connecting it to an instance pin.

Reconfiguring a Component

To reconfigure a component used in a SmartDesign:

- In the Canvas, select the instance and double-click the instance to bring up the appropriate configurator, or the HDL editor; or select the instance, right-click it, and choose **Configure Component**.
- Select the component in the Design Hierarchy tab and from the right-click menu select **Open Component**.

When the configurator is launched from the canvas, you cannot change the name of the component.

See Also

Design state management

Replacing components

Fixing an Out-of-Date Instance

Any changes made to the component will be reflected in the instance with an exclamation mark when you update the definition for the instance. An instance may be out-of-date with respect to its component for the following reasons:

- If the component interface (ports) is different – after reconfiguration - from that of the instance
- If the component has been removed from the project
- If the component has been moved to a different VHDL library
- If the SmartDesign has just been imported

You can fix an out-of-date instance by:

- Replacing the component with a new component (as shown in the figure below)
- Updating with the latest component

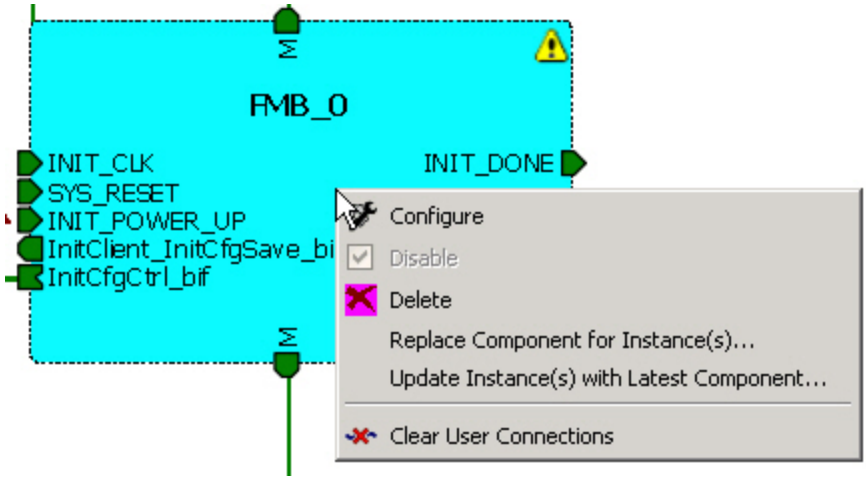


Figure 23 · Right-Click Menu - Replace Component for this Instance

See Also

Design state management

Reconfiguring components

Replacing components

Replacing Component Version

Components of an instance on the Canvas can be replaced with another component and maintain connections to all ports with the same name.

To replace a component in your design:

1. Select the component in the Design Hierarchy, right-click, and choose **Replace Component Version**. The Replace Component for Instance dialog box appears (see figure below).

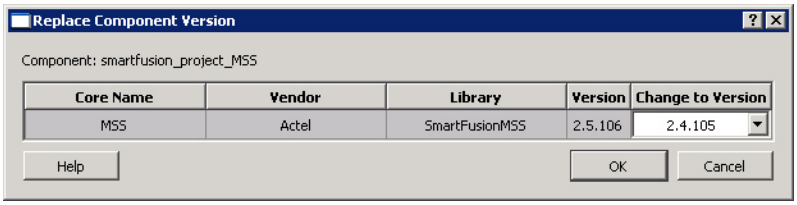


Figure 24 · Replace Component Version Dialog Box

2. Select the version you want to replace it with and click **OK**.

Design State Management

When any component with instances in a SmartDesign design is changed, all instances of that component detect the change.

If the change only affects the memory content, then your changes do not affect the component's behavior or port interface and your SmartDesign design does not need to be updated.

If the change affects the behavior of the instantiated component, but the change does not affect the component's port interface, then your design must be resynthesized, but the SmartDesign design does not need to be updated.

If the port interface of the instantiated component is changed, then you must reconcile the new definition for all instances of the component and resolve any mismatches. If a port is deleted, SmartDesign will remove that port and clear all the connections to that port when you reconcile all instances. If a new port is added to the component, instances of that component will contain the new port when you reconcile all instances.

The affected instances are identified in your SmartDesign design in the Grid and the Canvas with an exclamation point. Right-click the instance and choose **Update With Latest Component.**

Note: For HDL modules that are instantiated into a SmartDesign design, if the modification causes syntax errors, SmartDesign does not detect the port changes. The changes will be recognized when the syntax errors are resolved.

Changing memory content

For certain cores such as Analog System Builder, Flash Memory, or FlexRAM it is possible to change the configuration such that only the memory content used for programming is altered. In this case Project Manager (SoC) will only invalidate your programming file, but your synthesis, compile, and place-and-route results will remain valid.

When you modify the memory content of a core such as Analog System Builder or RAM with Initialization that is used by a Flash Memory core, the Flash Memory core indicates that one of its dependent components has changed and that it needs to be regenerated. This indication will be shown in the Hierarchy or Files Tab .

RAM with Initialization core - You can modify the memory content without invalidating synthesis.

Analog System Builder core - You can modify the following without invalidating synthesis:

- Existing flag settings: threshold levels, assertion/de-assertion counts, OVER/UNDER type
- Modifying sequence order or adding sequence operations
- Changing acquisition times
- Resistor Value for the Current Monitor
- RTC time settings
- Gate Driver source current

Flash Memory System Builder core - You can modify the following without invalidating synthesis:

- Modifying memory file or memory content for clients
- JTAG protection for Init Clients

Design Rules Check

The Design Rules Check runs automatically when you generate your SmartDesign; the results appear in the Reports tab. To view the results, from the **Design** menu, choose **Reports**.

- **Status** displays an icon to indicate if the message is an error or a warning (as shown in the figure below). Error messages are shown with a small red sign and warning messages with a yellow exclamation point.

- **Message** identifies the specific error/warning (see list below); click any message to see where it appears on the Canvas
- **Details** provides information related to the Message

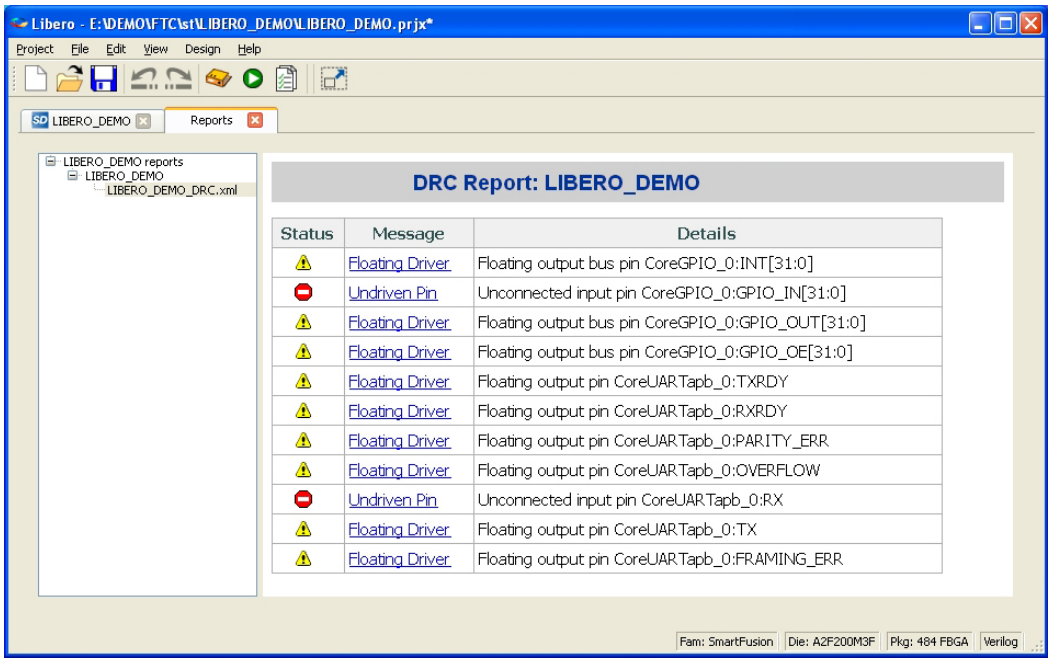


Figure 25 · Design Rules Check Results

Message Types:

Unused Instance - You must remove this instance or connect at least one output pin to the rest of the design.

Out-of-date Instance - You must update the instance to reflect a change in the component referenced by this instance; see Fixing an out-of-date instance.

Undriven Pin - To correct the error you must connect the pin to a driver or change the state, i.e. tie low (GND) or tie high (VCC).

Floating Driver - You can mark the pin unused if it is not going to be used in the current design. Pins marked unused are ignored by the Design Rules Check.

Unconnected Bus Interface - You must connect this bus interface to a compatible port because it is required connection.

Required Bus Interface Connection – You must connect this bus interface before you can generate the design. These are typically silicon connection rules.

Exceeded Allowable Instances for Core – Some IP cores can only be instantiated a certain number of times for legal design. For example, there can only be one CortexM1 or CoreMP7 in a design because of silicon limitations. You must remove the extra instances.

Incompatible Family Configuration – The instance is not configured to work with this project's Family setting. Either it is not supported by this family or you need to re-instantiate the core.

Incompatible Die Configuration – The instance is not configured to work with this project's Die setting. Either it is not supported or you need to reconfigure the Die configuration.

Incompatible 'Debug' Configuration – You must ensure your CoreMP7 and CoreMP7Bridge have the same 'Debug' configuration. Reconfigure your instances so they are the same.

No RTL License, No Obfuscated License, No Evaluation License – You do not have the proper license to generate this core. Contact Microsemi SoC to obtain the necessary license.

No Top level Ports - There are no ports on the top level. To auto-connect top-level ports, right-click the Canvas and choose Auto-connect

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.

Generating a Datasheet (SmartFusion, IGLOOe, ProASIC3L, ProASIC3E, Fusion)

If your SmartDesign is the root design in your project, then a Memory Map / Datasheet that contains your design information is produced.

Generating Firmware and Software IDE Workspace (SmartFusion, IGLOOe, ProASIC3L, ProASIC3E, Fusion)

If your SmartDesign is the root design in your project, then any compatible firmware drivers for your peripherals are generated to <project>/firmware.

The datasheet provides all the specifics of the generated firmware drivers.

SmartDesign Menu

Command	Icon	Function
Generate Component		Generates the SmartDesign component
Auto Connect		Auto-connects instances
Connection Mode		Toggles connection mode on or off
Add Port		Opens the Add Port dialog box, adds a port to the top SmartDesign component
QuickConnect		Opens the QuickConnect dialog box, enables you to view, find and connect pins

Command	Icon	Function
Auto-Arrange Instances		Adds a port to the top of the SmartDesign component
Route All Nets		Re-routes your nets; useful if you are unsatisfied with the default display
Show/Hide Nets		Enables you to show or hide nets on the Canvas
Show/Hide Net Names		Enables you to show or hide net names on the Canvas
Zoom In		Zooms in on the Canvas
Zoom Out		Zooms out on the Canvas
Zoom to Fit		Zooms in or out to include all the elements on the Canvas in the view
Zoom Box		Zooms in on the selected area
Enable/Disable Back Background Grid		Toggle switch to enable or disable the background grid display in the Canvas
Add Note		Adds text to your Canvas
Add Line		Enables you to add a line to the Canvas
Add Rectangle		Enables you to add a rectangle to the Canvas

SmartDesign Glossary

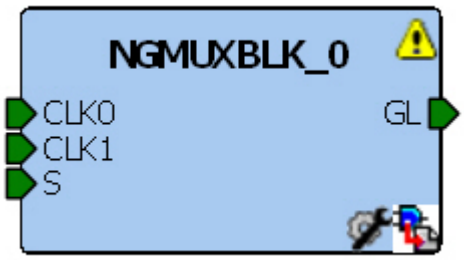
Term	Description
BIF	Abbreviation for bus interface.
bus	An array of scalar ports or pins, where all scalars have a common base name and have unique indexes in the bus.
Bus Definition	Defines the signals that comprise a bus interface. Includes which signals are present on a master, slave, or system interface, signal direction, width, default value, etc. A bus definition is not specific to a

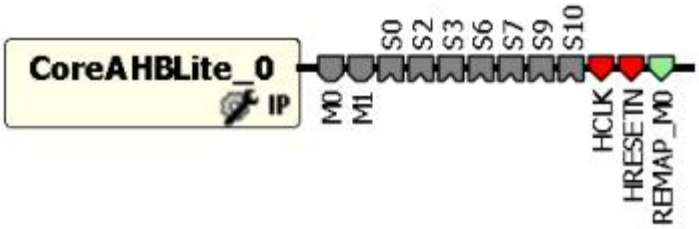
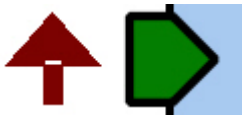
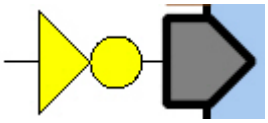
Term	Description
	logic or design component but is a type or protocol.
Bus Interface	Logical grouping of ports or pins that represent a single functional purpose. May contain both input and output, scalars or busses. A bus interface is a specific mapping of a bus definition onto a component instance.
Bus Interface Net	A connection between 2 or more compatible bus interfaces.
Canvas	Block diagram, connections represent data flow; enables you to connect instances of components in your design.
Component	Design element with a specific functionality that is used as a building block to create a SmartDesign core. A component can be an HDL module, non-IP core generated from the Catalog, SmartDesign core, Designer Block, or IP core. When you add a component to your design, SmartDesign creates a specific instance of that component.
Component Declaration	VHDL construct that refers to a specific component.
Component Port	An individual port on a component definition.
Driver	A driver is the origin of a signal on a net. The input and slave BIF ports of the top-level or the output and Master BIF ports from instances are drivers.
Instance	A specific reference to a component/module that you have added to your design. You may have multiple instances of a single component in your design. For each specific instance, you usually will have custom connections that differ from other instances of the same component.
Master Bus Interface	The bus interface that initiates a transaction (such as a read or write) on a bus.
Net	Connection between individual pins. Each net contains a single output pin and one or more input pins, or one or more bi-directional pins. Pins on the net must have the same width.
PAD	The property of a port that must be connected to a design's top level port. PAD ports will eventually be assigned to a package pin. In SmartDesign, these ports are automatically promoted to the top-level and cannot be modified.
Pin	An individual port on a specific instance of a component.

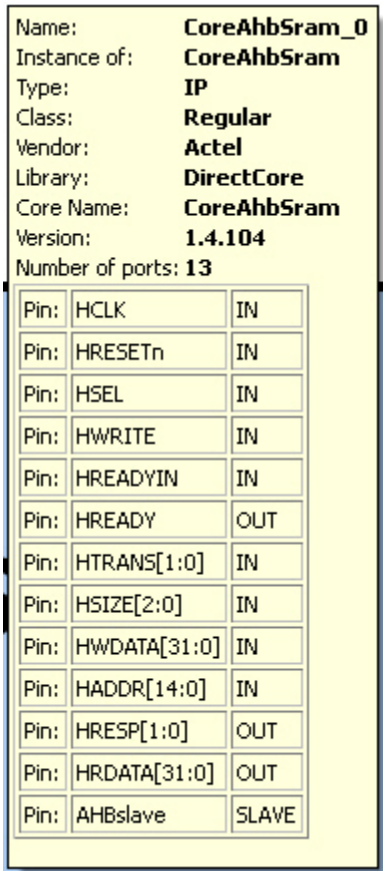
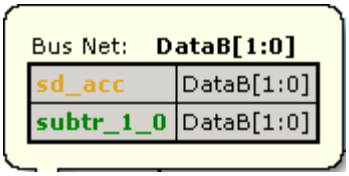
Term	Description
Port	An individual connection point on a component or instance that allows for an electrical signal to be received or sent. A port has a direction (input, output, bi-directional) and may be referred to as a 'scalar port' to indicate that only a single unit-level signal is involved. In contrast, a bus interface on an instance may be considered as a non-scalar, composite port. A component port is defined on a component and an instance port (also known as a 'pin') is part of a component instance.
Signal	A net or the electrical message carried on a net.
Slave Bus Interface	Bus interface that terminates a transaction initiated by a master interface.
System Bus Interface	Interface that is neither master nor slave; enables specialized connections to a bus.
Top Level Port	An external interface connection to a component/module. Scalar if a 1-bit port, bus if a multiple-bit port.

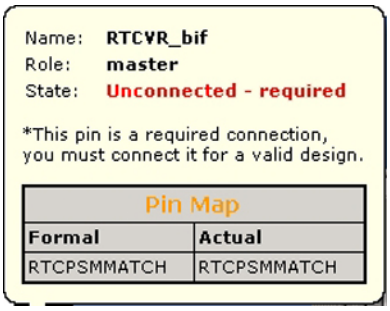
Canvas Icons

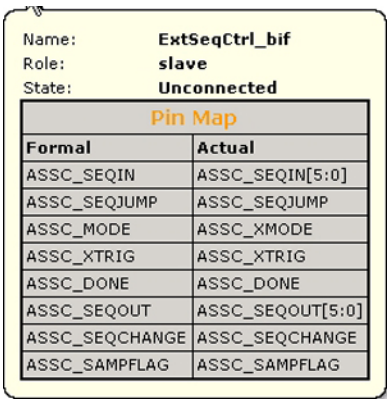
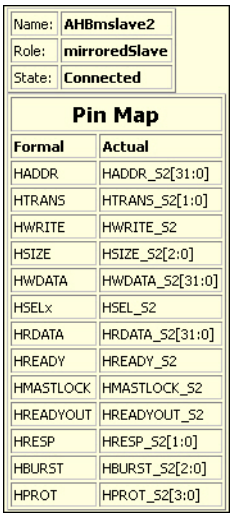
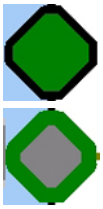
Hover your pointer over any icon in the SmartDesign Canvas view to display details.

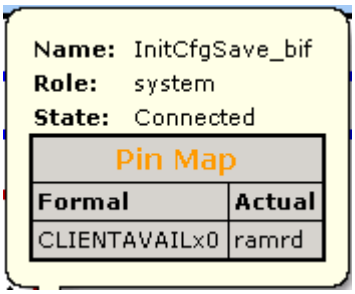
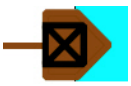
Icon	Description
	Representation of an instance in your design. An instance is a component that has been added to your SmartDesign component. The name of the instance appears at the top and the name of the generic component at the bottom. The instance type is indicated by an icon inside the instance. There are specific icons

Icon	Description
	for instances from SmartDesign, and HDL. The instance icon at left indicates a Microsemi SoC core.
	Bus instance; you can click and drag the end of a bus instance to resize it; also, the bus instance will resize based on the number of instances that you connect to it.
	Optional unconnected pin. Required pins are red.
	Connected pin
	Pin with default Tie Off
	Pin tied low
	Pin tied high
	Pin inverted
	Pin marked as unused
	Pin tied to constant

Icon	Description
	Instance details. If there are less than twenty ports, they are listed in the details.
	Bus Net details.
	Master bus interface icon. A master is a bus interface that initiates a transaction on a bus interface net. An unconnected master BIF with REQUIRED connection is red (shown at left).

Icon	Description
	A master BIF with unconnected OPTIONAL connection is gray.
	Master BIF details, showing name, role, and state. The Pin Map shows the Formal name of the pin assigned by the component (in this example, RCCLKOUT) and the Actual, or representative name assigned by the user (CLKOUT).
	Slave BIF (shown at left). Unconnected slave icons with REQUIRED connections are red. Unconnected slave icons with OPTIONAL connections are gray.

Icon	Description
	Slave BIF details, showing name, role, and state. The Pin Map shows the Formal name of the pin assigned by the component (in this example, ASSC_MODE) and the Actual, or representative name assigned by the user (ASSC_XMODE).
	Master-slave bus interface connection
	Master-slave bus interface connection details.
	Groups of pins in an instance. Fully connected groups are solid green. Partially

Icon	Description
	connected groups are gray with a green outline. Unconnected groups (no connections) are gray with a black outline.
	A system BIF is the bus interface that does not have a simple input/output relationship on both master/slave. This could include signals that only drive the master interface, or only drive the slave interface, or drive both the master and slave interfaces.
	System BIF details, showing name, role, and state. The Pin Map shows the Formal name of the pin assigned by the component (in this example, CLIENTAVAILx0), and the Actual name assigned by the user (in this example: ramrd).
	Pad port icon; indicates a hardwired chip-level pin

VHDL Special Types - Examples and meta.out File Format

The VHDL Special Types are:

- Integer
- Unsigned
- Array and Array of Arrays
- Record

The meta.out file format follows the examples.

Integer

-- Package Declaration

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
package universal_pkg is
  subtype integer1 is integer range 0 to 127;
  subtype integer2 is integer range 0 to 127;
end package universal_pkg;
```

--Entity Declaration

```
entity adder is
port (
  D1 , D2 : in integer1;
  D3 , D4 : in integer2;
  int_out1 : out integer range 0 to 255;
  int_out2 : out integer range 0 to 255
);
end entity adder;
```

Meta.out file:

```
package universal_pkg
integer integer1 [ 0 : 127 ]
integer integer2 [ 0 : 127 ]
end
entity adder
D1 integer1
D2 integer1
D3 integer2
D4 integer2
int_out1 integer [ 0 : 255 ]
int_out2 integer [ 0 : 255 ]
end
```

Unsigned

Entity declaration:

```
entity unsigned_2multiply_acc is
port( A : in unsigned(16 downto 0);
```

```
    B : in unsigned(34 downto 0);
    C : in unsigned(13 downto 0);
    D : in unsigned(37 downto 0);
    E : in unsigned(51 downto 0);
    P : out unsigned(51 downto 0);
    clk : in std_logic
  );
  end unsigned_2multiply_acc;
```

Meta.out file:

```
entity unsigned_2multiply_acc
A unsigned [ 16 : 0 ]
B unsigned [ 34 : 0 ]
C unsigned [ 13 : 0 ]
D unsigned [ 37 : 0 ]
E unsigned [ 51 : 0 ]
P unsigned [ 51 : 0 ]
End
```

Array and Array of Arrays

--Package Declaration

```
library IEEE;
use IEEE.std_logic_1164.all;
package array_package is
  subtype ram_input is std_logic_vector(31 downto 0);
   type ram_in is array(1 downto 0) of ram_input;
   type ram_out is array(1 downto 0) of ram_input;
end package array_package;
```

-- Entity Declaration

```
entity ram_inference is
   port (
         ram_init : in ram_in;
         write_enable : in std_logic;
         read_enable : in std_logic;
         CLK : in std_logic;
         write_address : in integer range 63 downto 0;
         read_address : in integer range 63 downto 0;
         read_data : out ram_out
         );
  end entity ram_inference;
```

Meta.out file:

```
package array_package
array_of_array ram_in [ 1 : 0 ]
end
array_of_array ram_out [ 1 : 0 ]
end
```

```
end
entity ram_inference
ram_init[1] ram_in
ram_init[0] ram_in
write_address integer [ 63 : 0 ]
read_address integer [ 63 : 0 ]
read_data[1] ram_out
read_data[0] ram_out
end
```

Record

- Package Declaration

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
package record_pkg is
  type array1 is array(3 downto 0) of std_logic;
  type array2 is array(3 downto 0) of std_logic_vector(3 downto 0);
  type test is record
        test_std_logic : std_logic;
        test_std_logic_vector : std_logic_vector(1 downto 0);
        test_integer : integer range 0 to 127;
        test_array : array1;
        test_array_of_array : array2;
        test_unsigned : unsigned(2 downto 0);
  end record;
end package record_pkg;
```

-- Entity Declaration

```
entity MUX is
generic ( N : integer := 1 );
port (
  mux_in1, mux_in2 : in test;
  sel_lines : in std_logic_vector(N-1 downto 0);
  mux_out : out test;
  mux_array : out array1
);
end entity MUX;
```

Meta.out file:

```
package record_pkg
array array1
end
array_of_array array2 [ 3 : 0 ]
end
record test
test_integer integer [ 0 : 127 ]
```

```
test_array array1
test_array_of_array array2
test_unsigned unsigned [ 2 : 0 ]
end
end
entity MUX
mux_in1.test_std_logic test
mux_in1.test_std_logic_vector test
mux_in1.test_integer test
mux_in1.test_array test
mux_in1.test_array_of_array[0] test
mux_in1.test_array_of_array[1] test
mux_in1.test_array_of_array[2] test
mux_in1.test_array_of_array[3] test
mux_in1.test_unsigned test
mux_in2.test_std_logic test
mux_in2.test_std_logic_vector test
mux_in2.test_integer test
mux_in2.test_array test
mux_in2.test_array_of_array[0] test
mux_in2.test_array_of_array[1] test
mux_in2.test_array_of_array[2] test
mux_in2.test_array_of_array[3] test
mux_in2.test_unsigned test
mux_out.test_std_logic test
mux_out.test_std_logic_vector test
mux_out.test_integer test
mux_out.test_array test
mux_out.test_array_of_array[0] test
mux_out.test_array_of_array[1] test
mux_out.test_array_of_array[2] test
mux_out.test_array_of_array[3] test
mux_out.test_unsigned test
mux_array array1
end
```

meta.out File Format

MetaFile : MetaLibraryItem | MetaPackageList | MetaEntityList

MetaLibraryItem : **library** <lib_name>

MetaPackageList : MetaPackageItem MetaPackageList

MetaPackageItem : **package** <package_name> MetaItemDeclarationList **end**

MetaItemDeclarationList: MetaItem MetaItemDeclarationList

MetaItem : (MetaRecordItem | MetaArrayOfArrayItem | MetaIntegerType | MetaArrayItem)

MetaIntegerItem : (MetaIntegerType | MetaIntegerWithoutType)

MetaIntegerType : **integer** <integer_name> NumericRange

MetaIntegerWithoutType : **integer** NumericRange

MetaUnsignedItem : **unsigned** <name> NumericRange

MetaArrayOfArrayItem : **array_of_array** < MetaArrayOfArrayName> Range [MetaArrayItem] end

MetaRecordItem : **record** <record_name> RecordItemList **end**

RecordItemList : RecordItem RecordItemList

RecordItem : <Inst_name> (MetaArrayOfArrayName | MetaIntegerItem | MetaUnsignedItem | MetaSimpleArray)

MetaEnumuratedItem : **enum** <enum_name> (Item_name{,Item_name})

Range : [NumericRange | MetaEnumuratedItem]

NumericRange : lsd : msd

MetaArrayItem :**array** <array_name> [<record_name>] **end**

MetaEntityList : **entity** <entity_name> MetaEntityItemList **end**

MetaEntityItemList : MetaEntityItem MetaEntityItemList

MetaEntityItem : (RecordEntityItemList | IntegerEntityItemList | ArrayEntityItemList | ArrayOfArrayEntityItemList | UnsignedEntityItemList | BufferPortItemList)

RecordEntityItemList : RecordEntityItem RecordEntityItemList

RecordEntityItem : (RecordNormalItem | RecordArrayOfArrayItemList)

RecordNormalItem : <user_port_name>. RecordItem <record_name>

RecordArrayOfArrayItemList : <record_port_name>[index]. RecordItem <record_name>

BufferPortItemList : BufferPortItem BufferPortItemList

BufferPortItem : **buffer** <buffer_name>

IntegerEntityItemList : IntegerEntityItem IntegerEntityItemList

IntegerEntityItem : <user_port_name> (MetaIntegerType | MetaIntegerWithoutType)

ArrayEntityItemList : ArrayEntityItem ArrayEntityItemList

ArrayEntityItem : <user_port_name> MetaArrayItem

ArrayOfArrayEntityItemList : ArrayOfArrayEntityItem ArrayOfArrayEntityItemList

ArrayOfArrayEntityItem : <port_name> < MetaArrayOfArrayName>

UnsignedEntityItemList : UnsignedEntityItem UnsignedEntityItemList

UnsignedEntityItem : <user_port_name> MetaUnsignedItem

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.

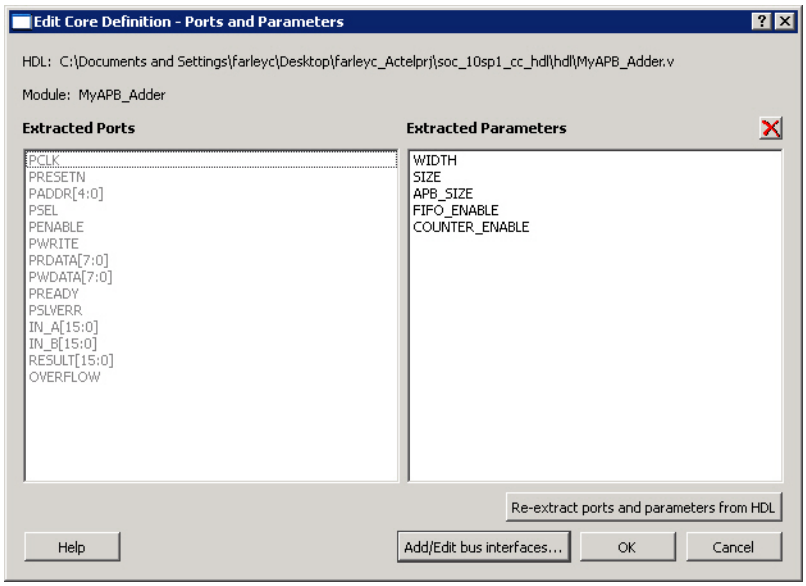


Figure 26 · Edit Core Definition - Ports and Parameters Dialog Box

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.

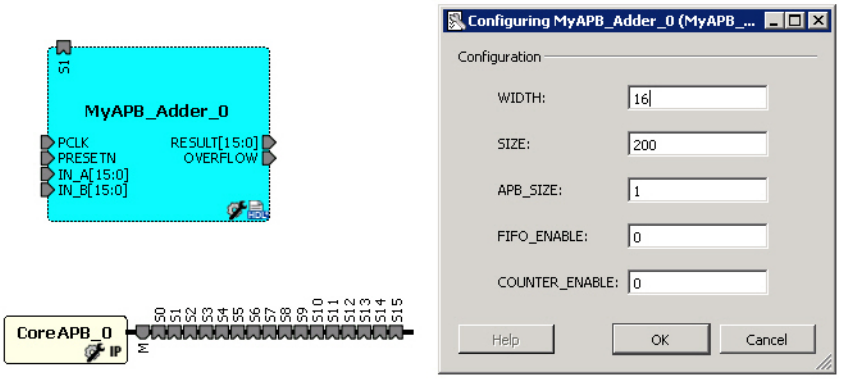


Figure 27 · HDL+ Instance and Configuration Dialog Box

You can right-click the instance and choose **Modify HDL** to open the HDL file inside the text editor.

Edit Core Definition

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

Remove Core Definition

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

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/*.vhdl) 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.

SmartDesign Testbench

SmartDesign Testbench is a GUI-based tool that enables you to design your testbench hierarchy. Use SmartDesign Testbench to instantiate and connect stimulus cores or modules to drive your Root design.

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

SmartDesign Testbench automatically instantiates the selected SmartDesign into the Canvas.

You can also double-click **Create SmartDesign Testbench** in the Design Flow window to add a new SmartDesign testbench to your project.

New testbench files appear in the Stimulus Hierarchy.

SmartDesign Testbench automatically instantiates your root design into the Canvas.

You can instantiate your own stimulus HDL or simulation models into the SmartDesign Testbench Canvas and connect them to your DUT (design under test). You can also instantiate Simulation Cores from the Catalog. Simulation cores are basic cores that are useful for stimulus generation, such as driving clocks, resets, and pulses.

Click the Simulation Mode checkbox in the Catalog to view available simulation cores.

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.

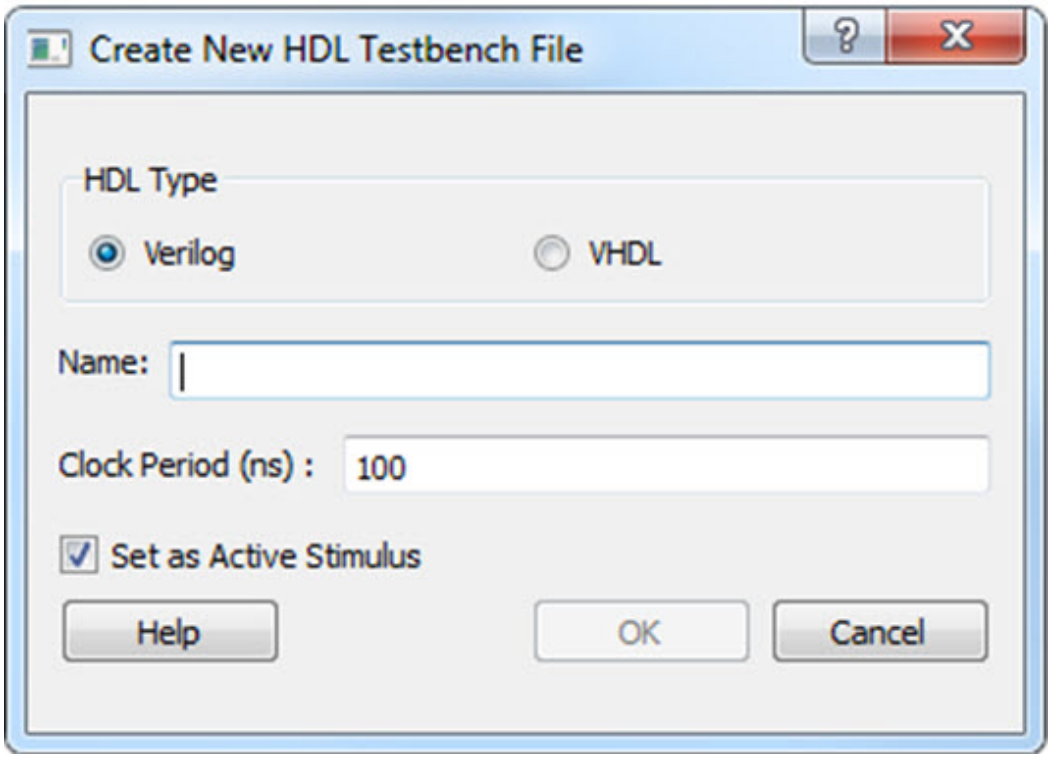


Figure 28 · Create New HDL Testbench File Dialog Box

```
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 29 · HDL Testbench Example - VHDL, Standard Template and Root Design Enabled

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 VLNV 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.

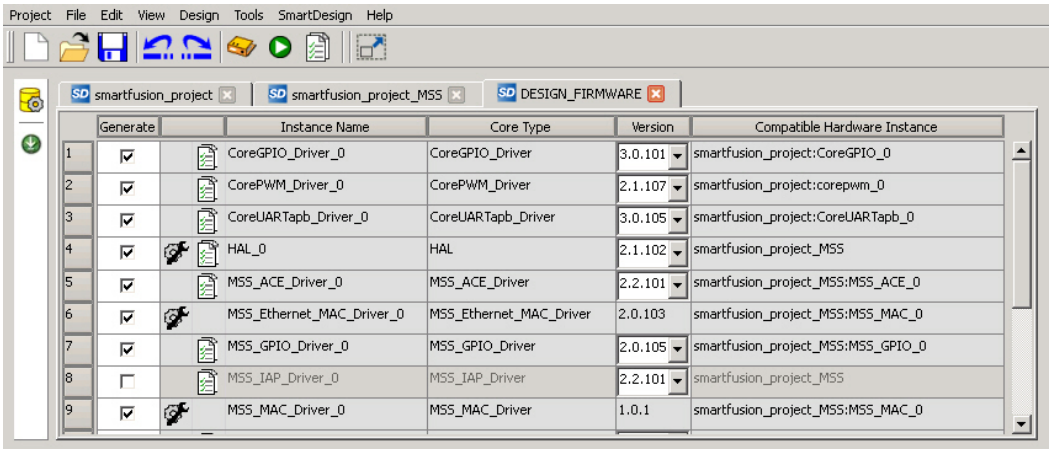


Figure 30 · Firmware Cores Tab (DESIGN_FIRMWARE)

See Also

Exporting Firmware and the Software IDE Workspace

Libero SoC Frequently Asked Questions

Running Libero SoC from your Software Tool Chain

Software IDE Integration

Designing with Designer Block Components

Designer Blocks (also generically called "components") enable you to partition a design and optimize critical sections. You can reuse them later in new applications, ensuring consistent performance. Designing with blocks enables multiple designers to work independently on parts of a single design.

Designer Block Advantages

- You can focus on the timing of critical blocks and ensure the timing across the blocks meets requirements before proceeding to the top-level flow.

- Changes in other blocks have no impact on your own block, you can re-use your block without re-calculating the timing.
- The block can be re-used in multiple designs
- Shorter verification time. You need to re-verify only the portion of the design that has changed.

Designer Block Features

- You can create a Designer Block with or without I/Os.
- A Designer Block can be synthesized, simulated, and placed-and-routed the same way as a regular design.
- You can lock the place-and-route of the Designer Block to ensure performance does not change.
- Performance and place-and-route can be fixed absolutely; however these rules can be relaxed gradually, if necessary, to ensure that you can integrate the Designer Block into your <top> project.
- You can use all the features in Designer Blocks in SmartDesign.

Use Designer Blocks When

- The design is congested (uses 90% of the resources on a given die).
- You have difficulty meeting timing by doing the design in its entirety. Blocks enable you to compartmentalize the design and optimize sections before you optimize the entire design.
- You want to re-use some elements of your design.
- You want to use the identical elements multiple times in a single design.

You cannot use Designer Blocks with all families, they are family and die specific; if your Designer Block has I/Os it is also package specific.

Supported Families

SmartFusion, IGLOO, ProASIC3, Fusion

Designer Blocks and Synthesis

You must run the synthesis tool in No I/O mode when you create your component. The Designer Block is not a full design; Libero SoC sets this option for Synplify if you Enable Designer Block creation.

When you Publish a Designer Block, the Project Manager creates a timing shell that enables the synthesis tool to better synthesize the <top> project. The timing shell is named <blockname>_syn.v(.vhd) if you are using Synplify or <blockname>_pre.v(.vhd) if you are using Precision.

When you are working in your <top> project, the synthesis tool does not know how many globals you have in your Designer Block, or if there will be clock sharing. The synthesis tool promotes as many globals as it can and if you have globals in the Designer Block you will exceed the total number of globals allowed in your device.

In this case, you must limit the number of globals added by the synthesis tool so that the total number (Designer Block plus <top> project) does not exceed the number available on your device.

To add an internal global, you can use either the Synplify constraints editor (SCOPE) or an SDC file.

For example, to add a CLKINT after a CLK port, the command is:

```
define_attribute {n:CLK} syn_insert_buffer {CLKINT}
```

See Also

Creating a component in Designer

Creating a component in Libero SoC

Managing I/Os in a Designer Block Component

If you use I/Os in your Designer Block, use the following rules:

- If the I/O is placed in the block, placement and VCCI of the I/O cannot be changed in the <top> design.
- The register combining option cannot be changed in the <top> design.
- Attributes and Vref pins can be changed if the values are legal (the I/O will not be unplaced).

Globals and Designer Block Components

You must manage your globals when creating a Designer Block to ensure that you have some available after you import the Block into your <top> project.

There is no limit to the number of globals you can use in a Designer Block.

Global Sharing

You can share a global between the Designer Block and the <top> project. You must:

- Use an internal global in the Designer Block.
- Drive the global port in the <top> project with a global net.

Libero SoC removes the internal global and re-routes the entire net.

You can use other global macros in the Designer Block, but you cannot share them with the <top> project.

Global Sharing with SmartFusion, IGLOO, ProASIC3 and Fusion - Use CLKINT in the Designer Block to share the global in the component with the <top> project.

See the list of Physical Design Constraint (PDC) files for more information on how to assign constraints.

Local Clock

You can use local clocks in your component to save on globals, but you may need to do some floorplanning in your <top> project.

Limitations

When you create your block, you cannot assign a port-connected net to a local clock.

The routing for local clocks from the blocks cannot always be preserved.

For all other families, local clocks are rerouted only if they are used in more than one block. The local clock constraint is preserved and the only difference in the routing is from the driver to the entry point of the clock network (when it gets to the clock network you end up with the same routing since the macros are locked in the same location).

Designer Block Compile Report

If you instantiate Designer Blocks in your design, the Compile report includes a description of the blocks you used. The report appears in the Log window after Compile is complete.

The report lists the name of the module, the name of the instance, the number of macros and nets used in the blocks, and information on how conflicts between blocks were resolved by the Compile options or PDC commands (if any). For example:

```
Block Information Report :
==========================
Conflict resolution from Compile options :
==========================================
```

```
Placement : Resolve conflict/Keep and Lock non conflicting
placement
 Routing : Resolve conflict/Keep and Lock non conflicting routing
-----------------------------------------------------
 Block Name : core1
Instance Name : core1_inst
| Locked | Total
 ---------------------------
 Instances | 4 | 4 (100.00%)
 Nets | 3 | 3 (100.00%)
-----------------------------------------------------
Block Name : core1
Instance Name : core11_inst
PDC Constraints :
 =================
 Move : move_block -inst_name {core11_inst} -left 10 -up 0 -
non_logic UNPLACE
| Locked | Total
 ---------------------------
 Instances | 4 | 4 (100.00%)
Nets | 0 | 3 (0.00%)
```

Designer Block Component Limitations

If you instantiate the same Designer Block many times in the <top> design, only the first instance retains the place-and-route information (if it has any); the others do not. Only the netlist is preserved.

To preserve the relative placement and routing of other blocks you must move the blocks using a PDC command. This PDC file must be imported as a source file along with the netlist(s) and CDB files. If possible, routing is preserved when you move the blocks with a PDC command.

See the move_block PDC command for more information.

Creating a Designer Block Component in Libero SoC

You must create two Libero SoC projects in order to instantiate your Designer Block in Libero SoC: one to create and publish your Designer Block, and another in which to instantiate your Designer Block. This section describes how to create your Designer Block.

See Instantiating a Designer Block in Libero SoC for more information.

The general design flow for creating a Designer Block in Libero SoC is shown in the figure below.

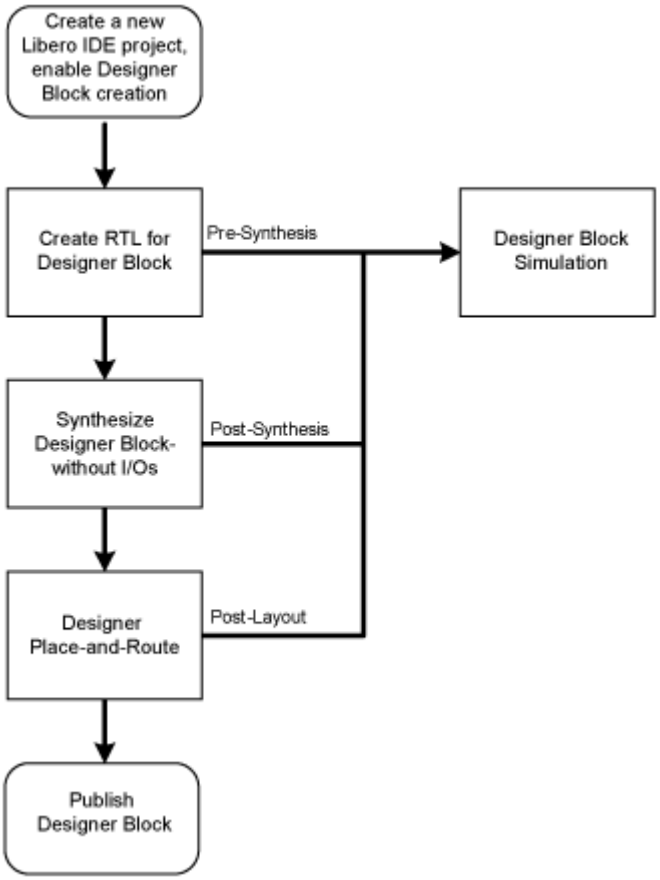


Figure 31 · Create a Designer Block Flow in Libero SoC

To create a Designer Block in Libero SoC with a new design:

1. Start a new project. You must select a family that supports Block designs (SmartFusion2, SmartFusion, IGLOO, ProASIC3, Fusion). After your project opens, from the **Project** menu, choose **Settings > Flow**, and click the **Enable Block Creation** checkbox.
2. Create a design in Libero SoC (standard design flow - create RTL, synthesize, run place-and-route and generate the block using Designer).

To create a Designer Block in the Libero SoC with an existing design, open your design and from the **Project** menu, choose **Setting > Flow**, and click the **Enable Block Creation** checkbox. Note that your design must use a device family that supports Designer Blocks (SmartFusion2, SmartFusion, IGLOO, ProASIC3, Fusion).

Instantiating a Designer Block in Libero SoC

You must have two projects in order to instantiate your Designer Block in Libero SoC: one to create and publish your Designer Blocks, and another in which to instantiate your Designer Block. This topic and the flow shown in the figure below describe how to instantiate your Designer Block in the Libero SoC.

See Creating a Designer Block in Libero SoC for information on how to create a Designer Block. You can also import your Designer Blocks into SmartDesign.

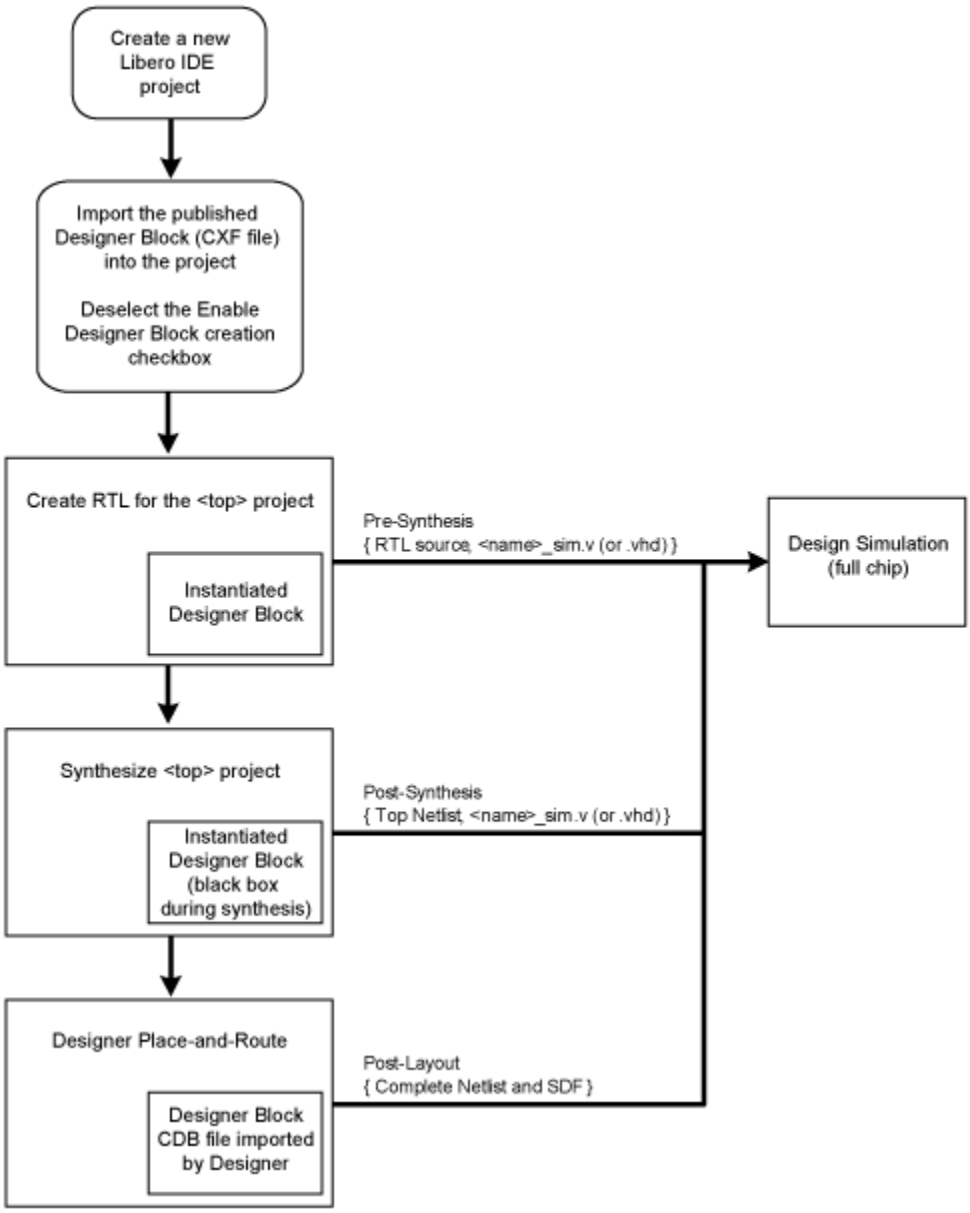


Figure 32 · Libero SoC Designer Block Instantiation Flow

To instantiate (import) a Designer Block in Libero SoC, import your design netlist and CXF file(s). The CXF file imports all the files you need for your Designer Block. After you import your files, the design flow is the same as regular Libero SoC designs. There is no limit to the number of CXF files you can import, but you cannot import the same Designer Block more than once, and the family and device for your imported block must match your project.

After you import the CXF file, the Project Manager displays the imported files in the Design Hierarchy tab.

The Designer Block(s) you instantiate must have the same family and die (and package, if it contains I/Os) as your current <top> project. If the family, die, and package do not match, Libero SoC asks if you want to change the current setting to match the one from the Designer Block.

The Project Manager passes all the Designer Block files to Designer automatically.

Note: Note

- Disable Designer Block creation when you import a component into your <top> project. If you are using a Designer Block component to create another Designer Block, leave it enabled.
- If you already have an HDL component with the same name as the one you imported, the new Designer Block component is not be used by default. You must and right-click the Designer Block component in the Project Manager and choose **Use this file** to make it use your Designer Block.

RTL Simulation

To perform pre-synthesis simulation, in the Stimulus Hierarchy right-click the testbench and choose **Simulate Pre-Synth Design > Run**.

If you want to perform pre-layout simulation, in the Design Flow window, under Verify Pre-Synthesized design, double-click **Simulate**.

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

ModelSim™ ME is a custom edition of ModelSim PE that is integrated into Libero SoC's design environment. ModelSim for Microsemi is an OEM edition of Model Technology Incorporated's (MTI) tools. ModelSim for Microsemi supports VHDL or Verilog. It only works with Microsemi libraries and is supported by Microsemi.

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

Note: ModelSim for Microsemi comes with its own 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

Simulation Options

You can set a variety of simulation options for your project.

To set your simulation options:

1. From the **Project** menu, choose **Project Settings**.
2. Click the simulation option you wish to edit: **DO file**, **Waveforms**, or **Vsim commands**.
3. Click **Close** to save your settings.

DO File

- **Use automatic Do file -** Select to execute the wave.do or other specified Do file. Use the wave.do file to customize the ModelSim Waveform window display settings.
- **Simulation Run Time -** Specify how long the simulation should run in nanoseconds. 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 Libero SoC replaces <top> with the actual top level macro when you run ModelSim.

- **Generate VCD file** - Select this checkbox to have ModelSim automatically generate a VCD file based on the current simulation. VCD files can be used in SmartPower. For best results, Microsemi recommends that a postlayout simulation be used to generate the VCD.
- **VCD filename** - Specify the name of the VCD file that will be automatically generated by ModelSim
- **User defined DO file** - Available if you opt not to use the automatic DO file. Input the path or browse to your user-defined DO file.
- **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 Libero SoC outputs the line 'add wave /testbench/*' in the DO file run.do. If you select DUT then Libero SoC outputs the line 'add wave /testbench/*' in the run.do file.
- **Log all signals in the design** - Saves and logs all signals during simulation.

Vsim Commands

- **SDF timing delays** - Select Minimum, Typical, or Maximum timing delays in the back-annotated SDF file.
- **Resolution**: The default is family-specific, but you can customize it to fit your needs.
  Some custom simulation resolutions may not work with your simulation library. For example, simulation resolutions above 1 ps will cause errors if you are using ProASIC3 devices (the simulation errors out because of an infinite zero-delay loop). 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
ProASIC3	1 ps
IGLOO	1 ps
SmartFusion and Fusion	1 ps
SmartFusion2	1 fs
IGLOO2	1 ps
RTG4	1 ps

- **Additional options:** Text entered in this field is added to the vsim command.

Simulation Libraries

- **Verilog (or VHDL) library path** - Enables you to choose the default library for your device, or to specify your own library. Enter the full pathname of your own library to use it for simulation.
- **Restore Defaults**: Restores factory settings.

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 Model*Sim* without a testbench.

To associate a stimulus:

1. Run simulation or in the Design Flow window under Verify Pre-Synthesized Design right-click **Simulate** and choose **Organize Input Files > Organize Stimulus Files**. The Organize Stimulus Files dialog box appears.
2. Associate your testbench(es):

   In the Organize Stimulus Files dialog box, all the stimulus files in the current project appear in the Source Files in the Project list box. Files already associated with the block appear in the Associated Source Files list box.

   In most cases you will only have one testbench associated with your block. However, if you want simultaneous association of multiple testbench files for one simulation session, as in the case of PCI cores, add multiple files to the Associated Source Files list.

   To add a testbench: Select the testbench you want to associate with the block in the Source Files in the Project list box and click **Add** to add it to the Associated Source Files list.

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

   To order testbenches: Use the up and down arrows to define the order you want the testbenches compiled. The top level-entity should be at the bottom of the list.
3. When you are satisfied with the Associated Source Files list, click **OK**.

Selecting Additional Modules for Simulation

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

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

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

Performing Functional Simulation

To perform functional simulation:

1. Create your testbench.
2. Right-click **Simulate** (in the Design Flow window, Implement Design > Verify Post-Synthesis Implementation > Simulate) and choose **Organize Input Files > Organize Simulation Files** from the right-click menu.

   In the Organize Files for Source dialog box, all the stimulus files in the current project appear in the Source Files in the Project list box. Files already associated with the block appear in the Associated Source Files list box.

   In most cases you will only have one testbench associated with your block. However, if you want simultaneous association of multiple testbench files for one simulation session, as in the case of PCI cores, add multiple files to the Associated Source Files list.

   To add a testbench: Select the testbench you want to associate with the block in the Source Files in the Project list box and click **Add** to add it to the Associated Source Files list.

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

3. When you are satisfied with the Associated Simulation Files list, click **OK**.
4. To start ModelSim ME, right-click **Simulate** in the Design Hierarchy window and choose **Open Interactively**.

   ModelSim starts and compiles the appropriate source files. When the compilation completes, the simulator runs for 1 µs and the Wave window opens to display the simulation results.
5. Scroll in the Wave window to verify 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 µs 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**.

Create Constraints

Constraints Overview

I/O Constraints (PDC Files)

The software enables you to specify the physical constraints to define the size, shape, utilization, and pin/pad placement of a design. You can specify these constraints based on the utilization, aspect ratio, and dimensions of the die. The pin/pad placement depends on the external physical environment of the design, such as the placement of the device on the board.

Timing Constraints (SDC Files)

Timing constraints represent the performance goals for your designs. Software uses timing constraints to guide the timing-driven optimization tools in order to meet these goals.

You can set timing constraints either globally or to a specific set of paths in your design.

You can apply timing constraints to:

- Specify the required minimum speed of a clock domain
- Set the input and output port timing information
- Define the maximum delay for a specific path
- Identify paths that are considered false and excluded from the analysis
- Identify paths that require more than one clock cycle to propagate the data
- Provide the external load at a specific port

To get the most effective results you need to set the timing constraints close to your design goals. Sometimes slightly tightening the timing constraint helps the optimization process to meet the original specifications.

Floorplanning Constraints (PDC files)

You can use a PDC file to control floorplanning. Import a Floorplanning PDC file to:

- Create Regions
- Assign logic to regions
- Assign nets to regions
- Floorplan a Designer Block

Note: For SmartFusion2, IGLOO2 and RTG4, do not mix I/O Constraints and Floorplanning Constraints in the same PDC file. If you do, Libero SoC errors out with an invalid constraint error.

I/O Constraints - SmartFusion2, IGLOO2, RTG4

SmartFusion2, IGLOO2, and RTG4 I/O constraints are PDC files. Note that for these devices I/O constraint PDC files are separate from Floorplan constraint PDC files; if you try to import a PDC file that contains both I/O and Floorplan constraints, Libero SoC errors out with an invalid constraint error.

Libero SoC generates an I/O PDC file automatically if you explicitly add/modify your I/O Constraints in the post-Compile I/O Editor. Your new I/O PDC file is added to the project and marked as Used.

I/O Constraints enables you to:

Import Files - If you do not have a compiled project, right-click I/O Constraints and choose Import Files to open the Import Files dialog box and import I/O constraint files (*.pdc files).

Create a New Constraint from Your Root Module - Right-click and choose **Create a New Constraint from your Root Module** to create a new constraint if you already have a compiled project.

Link Files - Right-click **I/O Constraints** and choose **Link Files** to link PDC constraint files from other projects. Linked files are not copied into your local project directory; instead the path is stored in your project, enabling you (or others) to update the file separately from Libero SoC. If your linked file is updated then the Project Manager indicates that the file has been changed and asks you if you wish to recompile, as appropriate.

Linked files appear in your **Files window** (**View > Windows > Files**), where they can be opened, deleted from the project, updated, or unlinked and copied to your local project.

Once you import an I/O Constraint file you can double-click the file in the **Design Flow** window (**Create Constraints > I/O Constraints > <filename>**) to open it in the Libero text editor, or right-click the file to:

Use for Compile - Includes the constraint file when you run Compile.

Open in Text Editor - Opens the file in the Text Editor so that you can update the code manually.

Save as - Opens the Save As dialog box, enables you to save the constraint with a different filename. This is useful if you want to preserve the settings of a particular constraint.

Delete from Project - Removes the file from the project.

Delete from Disk and Project - Removes the file from the project and deletes it from the disk.

Timing Constraints - SmartFusion2, IGLOO2, and RTG4

Timing Constraints enables you to:

Import Files - Double-click Timing Constraints to open the Import Files dialog box and import timing constraint files (*.sdc files).

Link Files - Right-click **Timing Constraints** and choose **Link Files** to link SDC constraint files from other projects. Linked files are not copied into your local project directory; instead the path is stored in your project, enabling you (or others) to update the file separately from Libero SoC. If your linked file is updated then the Project Manager indicates that the file has been changed and asks you if you wish to recompile, as appropriate.

Linked files appear in your Files window (**View > Windows > Files**), where they can be opened, deleted from the project, updated, or unlinked and copied to your local project.

Once you import or generate a Timing Constraint file, you can double-click the file in the **Design Flow** window (**Create Constraints > Timing Constraints > <filename>**) to open it in the Text Editor, right-click the file to:

Use for Synthesis - Uses the file for synthesis.

Use for Compile - Includes the file during Compile.

Open in Text Editor - Opens the file in the Project Manager Text Editor.

Save as - Opens the Save As dialog box, enables you to save the constraint in a different location and/or filename. This is useful if you want to preserve the settings of a particular constraint, or to save it outside your project.

Delete from Project - Removes the file from the project.

Delete from Disk and Project - Removes the file from the project and deletes it from the disk.

Floorplan Constraints - SmartFusion2, IGLOO2, RTG4

SmartFusion2, IGLOO2, RTG4 Floorplan constraints are PDC files. Floorplan constraint PDC files are separate from I/O constraint PDC files; if you try to import a PDC file that contains both I/O and Floorplan constraints then Libero SoC errors out with an invalid constraint error.

Floorplan Constraints enables you to:

Import Files - Double-click Floorplan Constraints to open the Import Files dialog box and import Floorplan constraint files (*.pdc files).

Link Files - Right-click **FloorPlan Constraints** and choose **Link Files** to link PDC constraint files from other projects. Linked files are not copied into your local project directory; instead the path is stored in your project, enabling you (or others) to update the file separately from Libero SoC. If your linked file is updated then the Project Manager indicates that the file has been changed and asks you if you wish to recompile, as appropriate.

Linked files appear in your **Files window** (**View > Windows > Files**), where they can be opened, deleted from the project, updated, or unlinked and copied to your local project.

Once you import Floorplan Constraint file you can double-click the file in the **Design Flow** window (**Create Constraints > Floorplan Constraints > <filename>**) to open it in the Text Editor, or right-click the file to:

Use for Compile - Includes the constraint file when you run Compile.

Open in Text Editor - Opens the file in the Project Manager Text Editor.

Save as - Opens the Save As dialog box, enables you to save the constraint with a different filename. This is useful if you want to preserve the settings of a particular constraint.

Delete from Project - Removes the file from the project.

Delete from Disk and Project - Removes the file from the project and deletes it from the disk.

Implement Design

Synthesize (Classic Constraint Flow)

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.

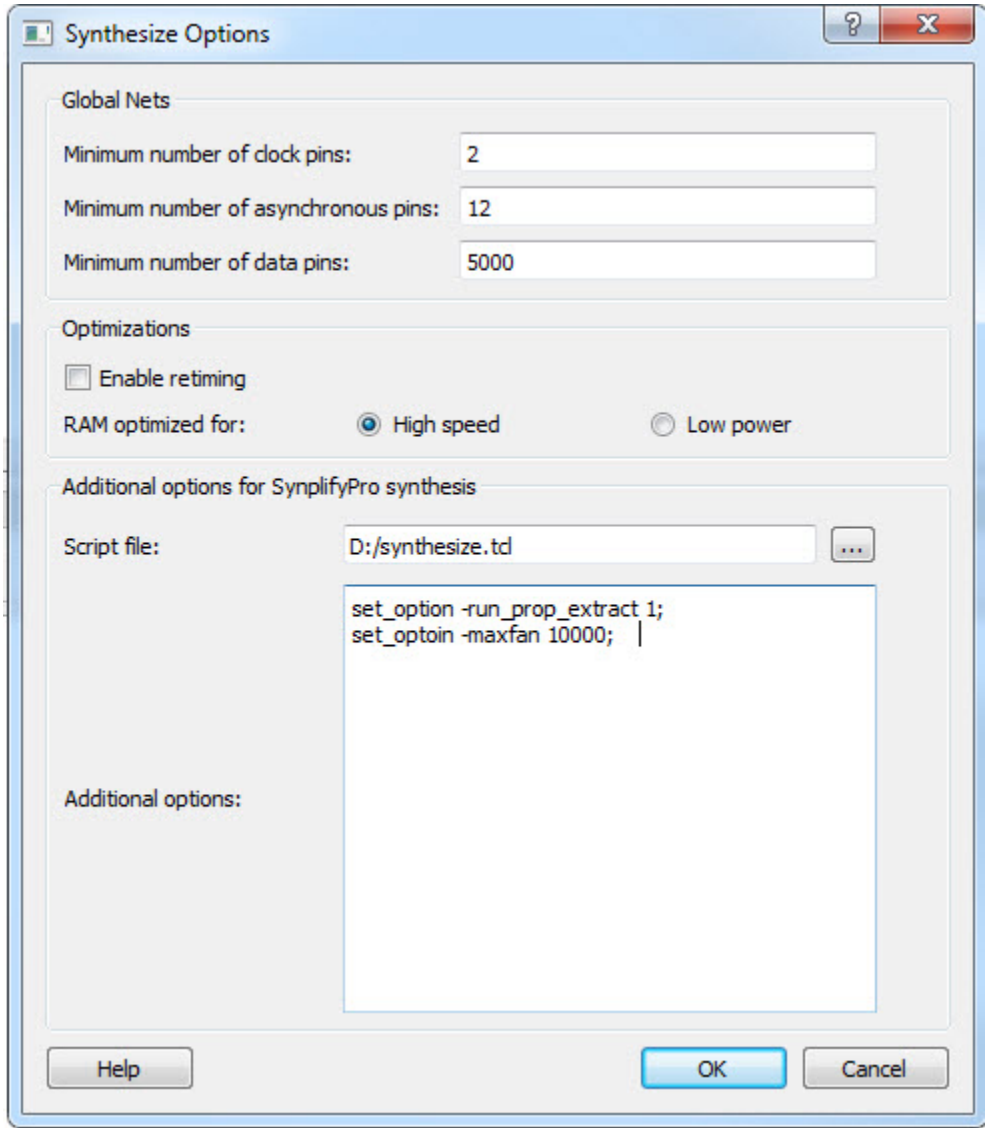


Figure 33 · Synthesize Options Dialog Box

Global Nets (Promotions and Demotion)

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 the RT4G150 die of the RTG4 family. This option is not available for the RT4G150_ES die.
- **Minimum number of data pins** – specifies the threshold value for data pin promotion to global resources. It is the minimum fanout of data (non-clock) nets to be kept on globals (no demotion). The default is 5000 (must be between 1000 and 200000).

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.

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

- **High speed** – RAM Optimization is geared toward Speed. The resulting synthesized design achieves better performance (higher speed) at the expense of more FPGA resources.
- **Low power** – RAM Optimization is geared toward 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. Libero does not perform any syntax checks for these options. All 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 EDIF, *.edn, file. Your EDIF netlist is then automatically translated by the software into an HDL netlist. The resulting *edn and *.vhd 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.

Precision RTL

Libero SoC Classic Constraint Flow supports Precision RTL from Mentor Graphics. Libero SoC Enhanced Constraint Flow does not support Precision RTL.

To run synthesis with Precision RTL default settings, set Precision RTL as the synthesis tool for your project (as outlined below), right-click **Synthesize** and choose **Run**.

To run synthesis with custom settings, right-click **Synthesize** and choose **Open Interactively**. Precision RTL opens and enables you to change settings before you run synthesis.

If your design is not ready for synthesis then Open does not appear in your right-click menu.

To set Precision RTL as the synthesis tool for your project:

1. From the **Project** menu, choose **Tool Profiles**. The Tool Profiles dialog box appears.
2. Click **Synthesis** to choose the synthesis tool profile.
3. Click the **Add** button. The Add Profile dialog box appears.
4. Enter a name. This is the name that appears in the Tool Profile dialog box.
5. In the **Tool integration** dropdown menu choose **Precision RTL**.
6. Enter the location of Precision RTL and any additional parameters.
7. Click **OK**.
8. Select **Precision RTL** in the Tool Profile dialog box and click **OK**.
9. Double-click **Synthesize** in the Design Flow window to start Precision RTL and run synthesis.

Compile - SmartFusion, IGLOO, ProASIC3, Fusion

See the SmartFusion2/IGLOO2 Compile options if you are using those devices.

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

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

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

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

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

Configure Options in Compile

The configuration dialog provides options that enable you to control merge behavior during Compile .

Merge SDC file(s) with Existing Timing Constraints

Select **Merge SDC file(s) with existing timing constraints** to preserve all existing timing constraints that you have entered using either the constraints editor or in a previously imported SDC file.

If you import new SDC files and you have this checkbox selected, the software merges the existing constraints and the constraints from the new SDC files. In case of a conflict, the new constraints have priority over the existing constraints.

This option is **On** by default. When this option is **Off**, all the existing timing constraints are replaced by the constraints in the newly imported SDC files.

Merge PDC file(s) with Existing Physical Constraints

Select **Merge PDC file(s) with existing physical constraints** to preserve all existing physical constraints that you have entered using either one of the MVN tools or in a previously imported PDC file.

If you import new PDC files and you have this checkbox selected, the software merges the existing constraints and the constraints from the new PDC files. In case of a conflict, the new constraints have priority over the existing constraints.

This option is **On** by default. When this option is **Off**, all the existing physical constraints are replaced by the constraints in the newly imported PDC files.

Compile - SmartFusion2, IGLOO2, and RTG4

See the Compile options for SmartFusion, IGLOO, ProASIC3, Fusion if you are designing for those families.

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

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

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

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

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

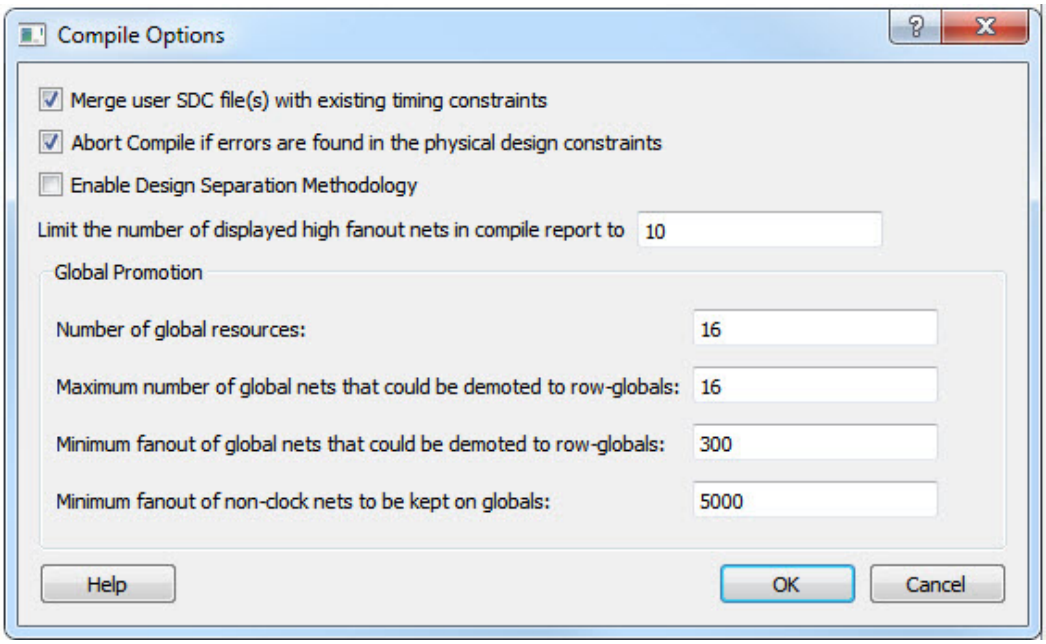


Figure 34 · Compile Options Dialog Box

Configure Options

The Compile Options dialog box enables you to control SDC file merge behavior, PDC error reporting, and limit the number of high fanout nets in the Compile Report.

Merge User SDC file(s) with Existing Timing Constraints

Select **Merge User SDC file(s) with existing timing constraints** to preserve all existing timing constraints that you have entered using either the constraints editor or in a previously imported SDC file.

If you import new SDC files and you have this checkbox selected, the software merges the existing constraints and the constraints from the new SDC files. In case of a conflict, the new constraints have priority over the existing constraints.

This option is **On** by default. When this option is **Off**, all the existing timing constraints are replaced by the constraints in the newly imported SDC files.

Abort Compile if errors are found in the physical design constraints

Controls the compile flow behavior if errors are encountered in the physical design constraints (PDC) file. Select this option to stop the flow if any error is reported in reading your PDC file. If you deselect this option, the tool skips errors when reading your PDC file and reports them as warnings. The default is ON.

This option is useful if you do not wish to debug your PDC commands before you run Compile.

Note: Compile fails even if this option is deselected if there is a PDC command syntax error (for example, the command does not exist or the syntax of the command is incorrect)

Note: Every time you invoke this dialog box, this option is reset to its default value ON. This is to ensure that your PDC file is correct.

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.

Enable Design Separation Methodology (SmartFusion2 and IGLOO2 only**) Checkbox** – Check this box if your design is for security and safety critical applications and you want to make your design's individual

subsystems (design blocks) separate and independent (in terms of physical layout and programming) to meet your design separation requirements. When checked, Libero generates a parameter file (MSVT.param) that details design blocks present in the design and the number of signals entering and leaving a design block. Microsemi provides a separate tool, known as Microsemi Separation Verification Tool (MSVT), which checks the final design place and route result against the MSVT.param file and determines whether the design separation meets your requirements.

Limit the number of displayed high fanout nets in compile report to - The number of high fanout nets to be displayed is controlled using the **Limit the number of displayed high fanout nets**; the default value is 10. This means the top 10 nets with the highest fanout will appear in the Compile Report.

Global Promotion/Demotion Options

Number of global resources - The number of available global resources for the die is reported in this field. It varies with the die size you have selected for the Libero project.

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

Maximum Number of global nets that could be demoted to row-globals – Specifies the maximum number of global nets that could be demoted to row-globals. The default is 16. A higher value may negatively impact the tool placing essential globals.

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. The default is 300. If you run out of global routing resources for your design, reduce this number (to allow more globals to be demoted) or select a bigger die for your design.

Minimum fanout of non-clock nets to be kept on globals – Specifies the minimum fanout of non-clock (data) nets to be kept on globals (no demotion). The default is 5000. If you run out of global routing resource for your design, increase this number or select a bigger die for your design. A valid range is between 300 and 20,000.

Block Instantiation Compile Options

If there are multiple blocks instantiated in your design, the software uses the Compile Options to resolve the conflicts. These options appear only if you are using Blocks in your design.

Placement

Error if conflict - Compile errors out if any instance from a designer block is unplaced. This is the default option.

Resolve conflict

- **Keep non-conflicting placement** - If some instances get unplaced for any reason, the non-conflicting elements remaining are preserved but not locked (you can move them).
- **Keep and lock non-conflicting placement** - If some instances get unplaced for any reason, the remaining non-conflicting elements are preserved and locked.
- **Discard placement from all blocks** – Placement information will be discarded from all blocks even if there is no conflict.

Routing

Error if conflict - Compile errors out if any preserved net routing in a designer block is deleted.

Resolve conflict

- **Keep non-conflicting routing**- If a nets' routing is removed for any reason, the routing for the non-conflicting nets is preserved but not locked (so that they can be rerouted). This is the default option.
- **Keep and lock non-conflicting routing**- If the routing is removed for any reason, the remaining non-conflicting nets are preserved and locked; they cannot be rerouted. This is the default option.

- **Discard routing from all blocks** – Routing information will be discarded from all blocks even if there is no conflict.

Compile Options (SmartFusion, IGLOO, ProASIC3, and Fusion)

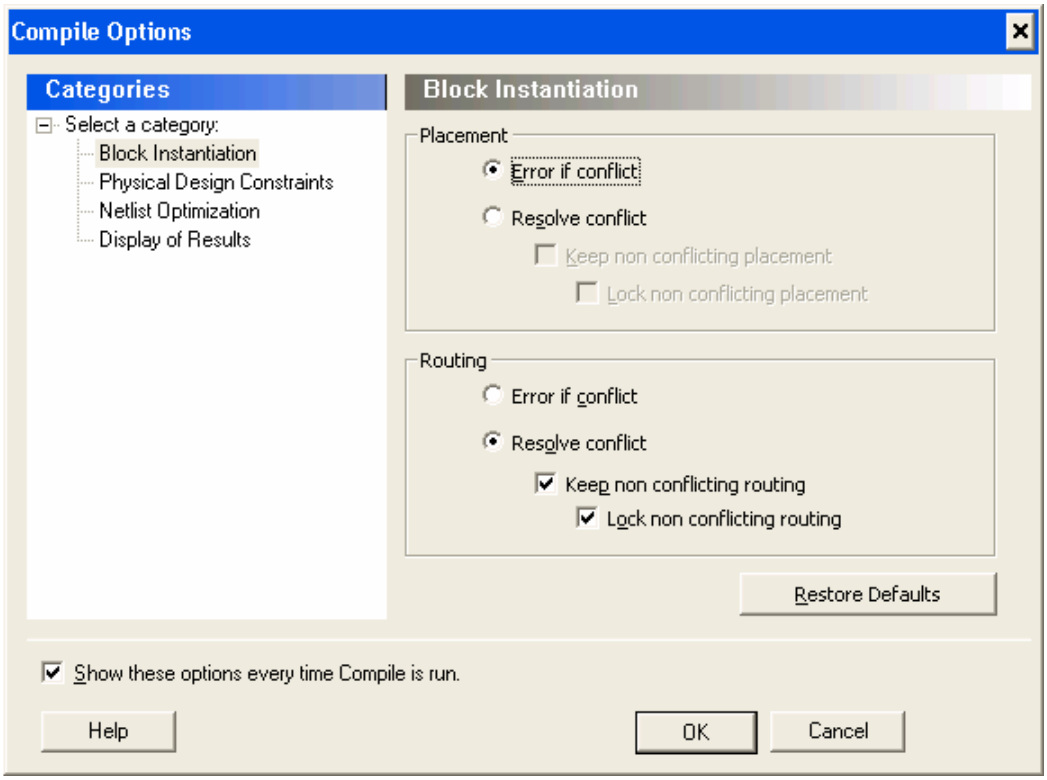
To set custom compile options:

1. Right-click **Compile** and choose **Open Interactively**. Designer opens.
2. Click the **Compile** button. The Compile Options dialog box opens. The Options available are family specific.
3. Select your options, and click **OK**.

The Compile Options dialog box enables you to do the following:

- Set your Block Instantiation options (used for conflict resolution when you instantiate multiple blocks)
- Verify Physical Design Constraints
- Perform Globals Management
- Netlist Optimization
- Generate a Compile report in Display of Results
- Set Block Creation options (available only if you are creating a block)

Block Instantiation



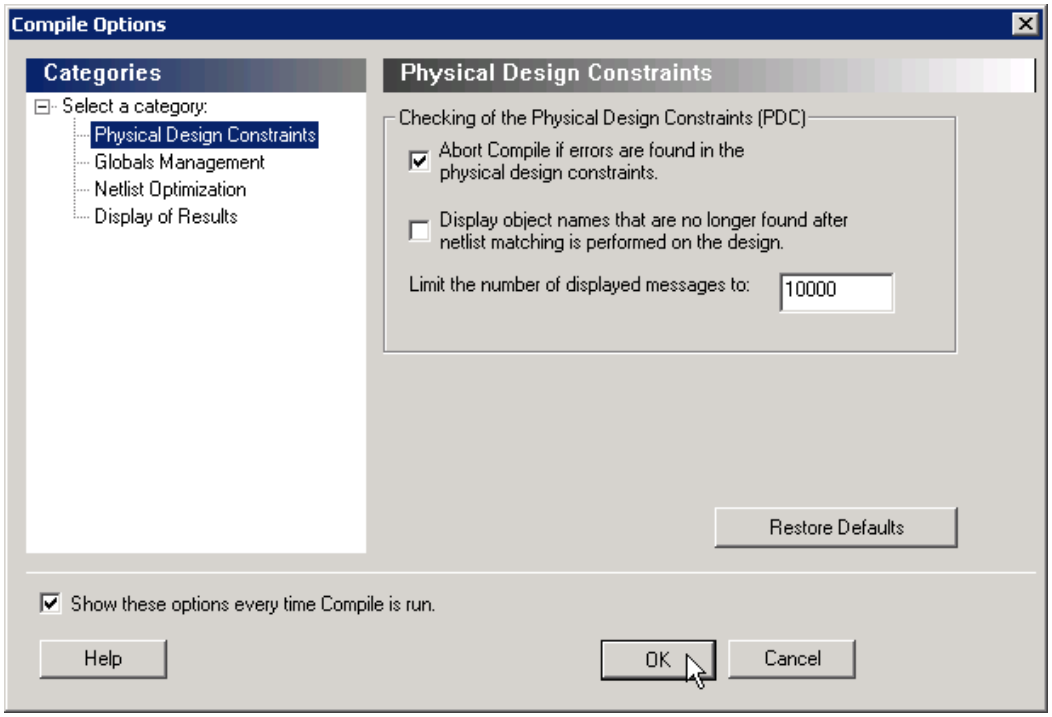
Designer uses the Block Instantiation options to resolve conflicts between multiple blocks in your design. The default options is to return an error if there is overlapping placement between the blocks and resolve any conflict for nets.

This ensures you are aware that the blocks overlap; you can go back and set the placement to resolve the conflicts and it will Compile.

See Conflict resolution in Designer Blocks for more information.

Physical Design Constraints

This interface enables you to verify the Physical Design Constraints (PDC) file.



Checking the Physical Design Constraint (PDC)

Abort Compile if errors are found in the physical design constraints: Changes the Abort on PDC error behavior. Select this option to stop the flow if any error is reported in reading your PDC file. If you deselect this option, the tool skips errors in reading your PDC file and just reports them as warnings. The default is ON.

Note: The flow always stops even if this option is deselected in the following two cases:

- If there is a Tcl error (for example, the command does not exist or the syntax of the command is incorrect)
- The assign_local_clock command for assigning nets to LocalClocks fails. This may happen if any floor planning DRC check fails, such as, region resource check, fix macro check (one of the load on the net is outside the local clock region). If such an error occurs, then the Compile command fails. Correct your PDC file to proceed.

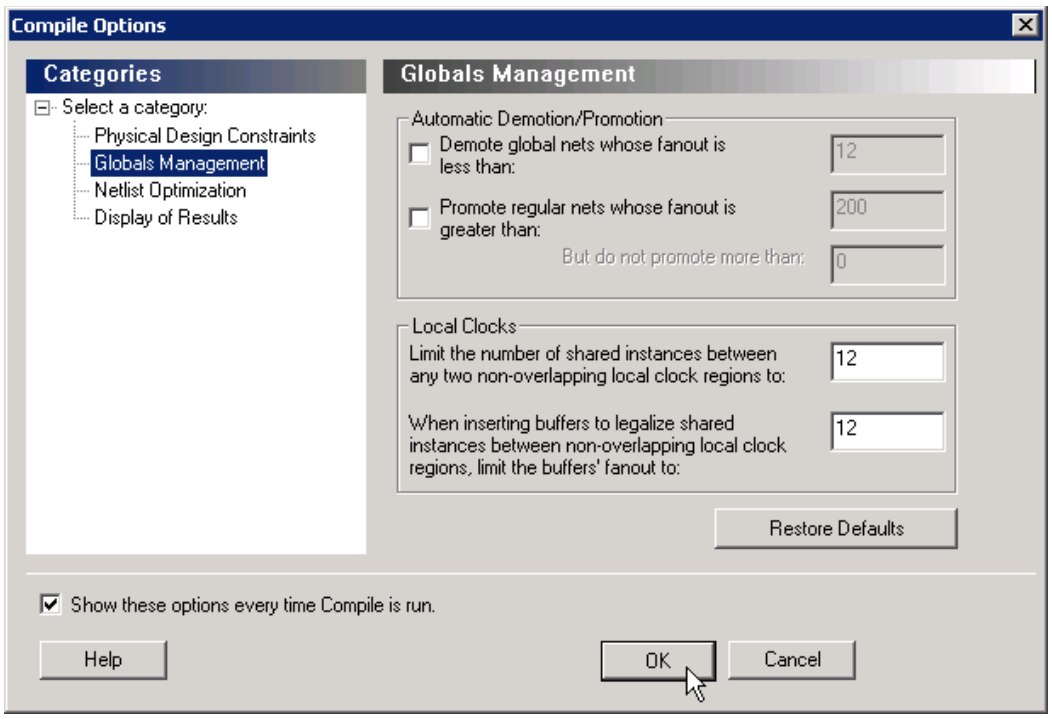
Note: Every time you invoke this dialog box, this option is reset to its default value ON. This is to ensure that your PDC file is correct.

Display object names that are no longer found after netlist matching is performed on the design: Displays netlist objects in the PDC that are not found in the imported netlist during the Compile ECO mode. Select this option to report netlist objects not found in the current netlist when reading the internal ECO PDC constraints. The default is OFF.

Limit the number of displayed messages to: Defines the maximum number of errors/warnings to be displayed in the case of reading ECO constraints. The default is 10000 messages.

Globals Management

The interface provides a global control to the Compile component of the design flow.



Automatic Demotion/Promotion

Demote global nets whose fanout is less than: Enables the global clock demotion of global nets to regular nets.

By default, this option is OFF. The maximum fanout of a demoted net is 12.

Note: A global net is not automatically demoted (assuming the option is selected) if the resulting fanout of the demoted net is greater than the max fanout value. Microsemi recommends that the automatic global demotion only act on small fanout nets. Microsemi recommends that you drive high fanout nets with a clock network in the design to improve timing and routability.

Promote regular nets whose fanout is greater than: Enables global clock promotion of nets to global clock network. By default, this option is OFF. The minimum fanout of a promoted net is 200.

But do not promote more than: Defines the maximum number of nets to be automatically promoted to global. The default value is 0. This is not the total number as nets need to satisfy the minimum fanout constraint to be promoted. The promote_globals_max_limit value does not include globals that may have come from either the netlist or PDC file (quadrant clock assignment or global promotion).

Note: Demotion of globals through PDC or Compile is done before automatic global promotion is done.

Note: You may exceed the number of globals present in the device if you have nets already assigned to globals or quadrants from the netlist or by using a PDC file. The automatic global promotion adds globals on what already exists in the design.

Local clocks

Limit the number of shared instances between any two non-overlapping local clock regions to: Defines the maximum number of shared instances allowed to perform the legalization. It is also for quadrant clocks.

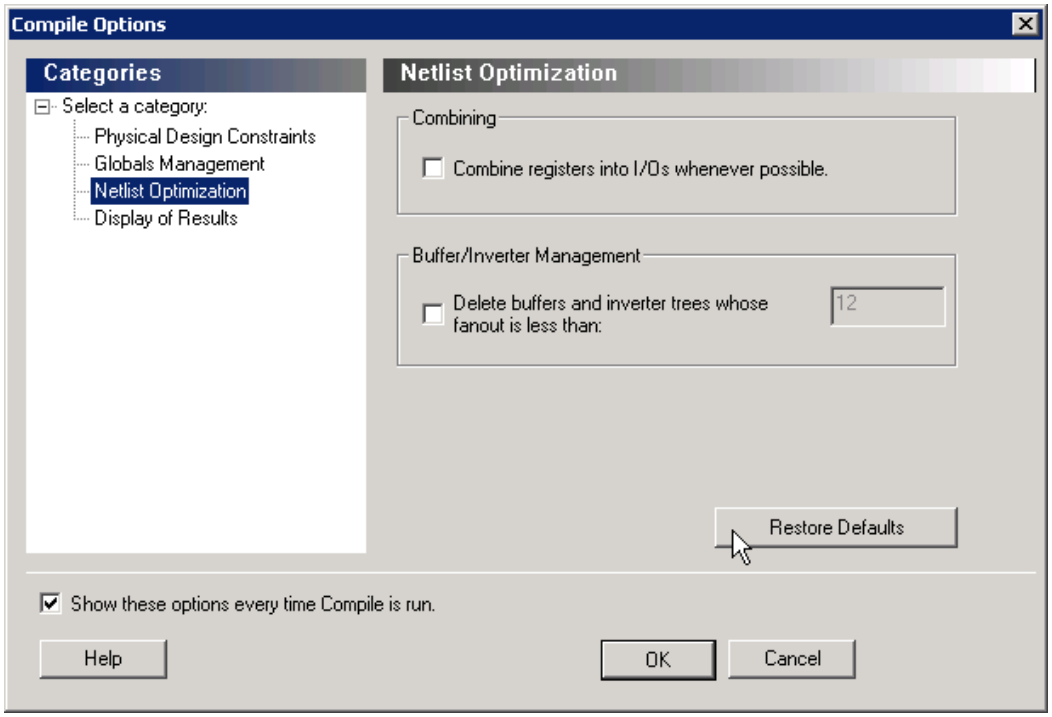
The maximum number of instances allowed to be shared by 2 local clock nets assigned to disjoint regions to perform the legalization (default is 12, range is 0-1000). If the number of shared instances is set to 0, no legalization is performed.

When inserting buffers to legalize shared instances between non-overlapping local clock regions, limit the buffers' fanout to: Defines the maximum fanout value used during buffer insertion for clock legalization. Set the value to 0 to disable this option and prevent legalization (default value is 12, range is 0-1000). If the value is set to 0, no buffer insertion is performed. If the value is set to 1, there will be one buffer inserted per pin.

Note: If you assign quadrant clock to nets using MultiView Navigator, no legalization is performed.

Netlist Optimization

This interface allows you to perform netlist optimization.



Combining

Combine registers into I/O wherever possible: Combines registers at the I/O into I/O-Registers. Select this option for optimization to take effect. By default, this option is OFF.

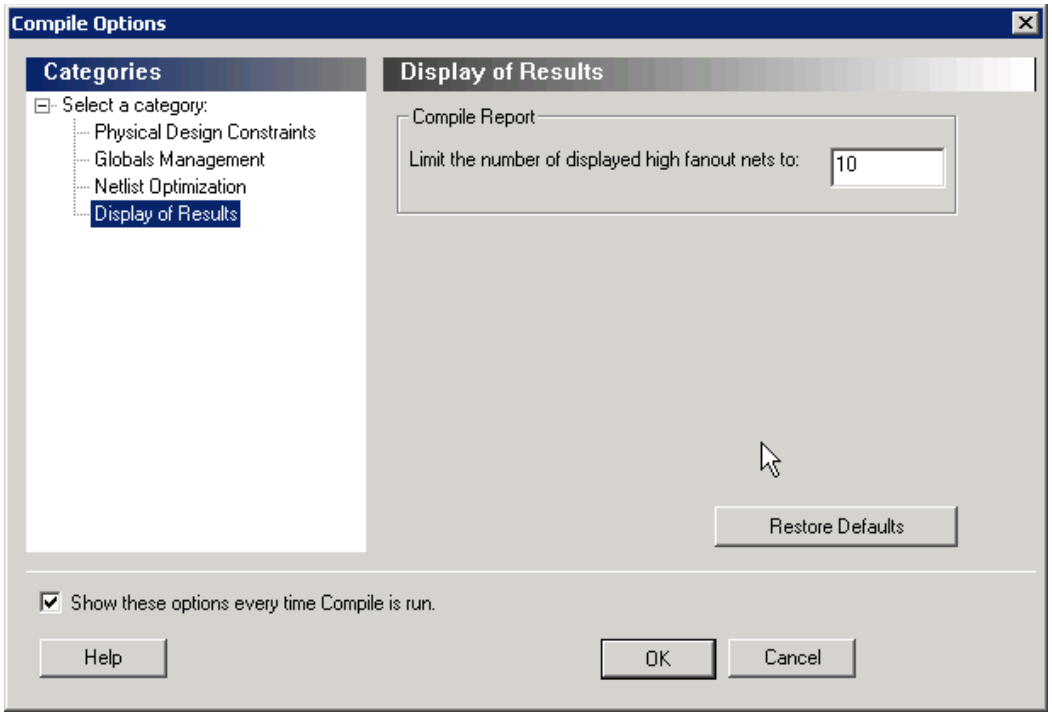
Buffer/Inverter Management

Delete buffers and inverter trees whose fanout is less than: Enables buffer tree deletion on the global signals from the netlist. The buffer and inverter are deleted. By default, this option is OFF. The maximum fanout of a net after buffer tree deletion is 12.

Note: A net does not automatically remove its buffer tree (assuming the option is on) if the resulting fanout of the net (if the buffer tree was removed) is greater than the max fanout value. Microsemi recomends that the automatic buffer tree deletion should only act on small fanout nets. From a routability and timing point of view, it is not recommended to have high fanout nets not driven by a clock network in the design.

Display of Results

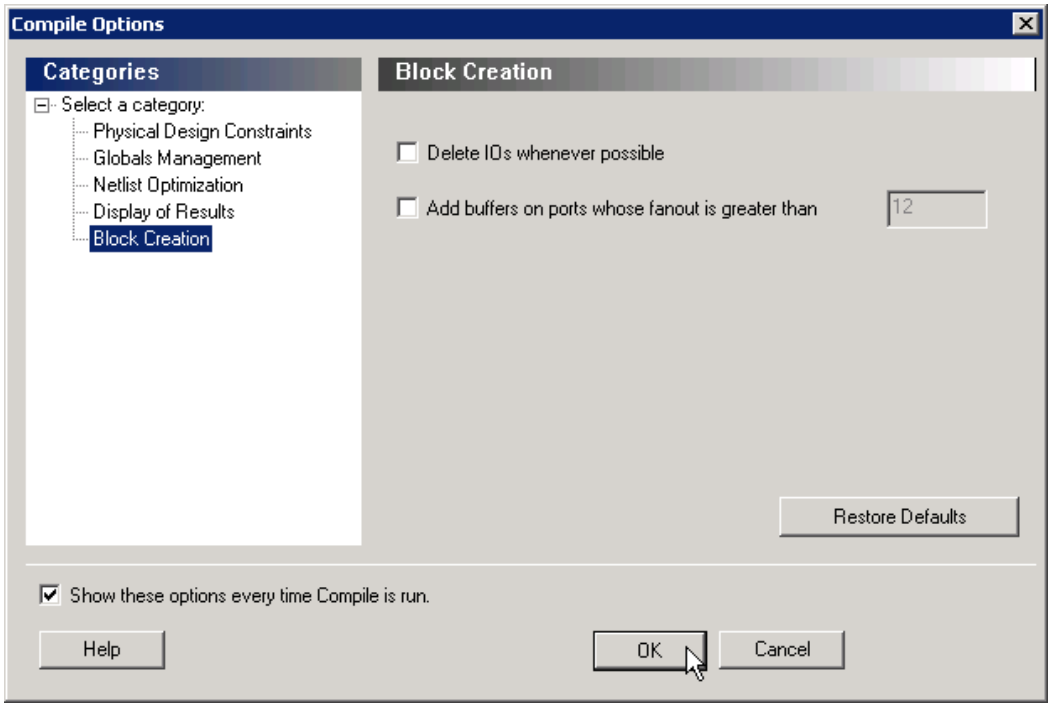
This interface lets you generate a Compile report.



Compile Report

Limit the number of displayed high fanout nets to: Enables flip-flop net sections in the compile report and defines the number of nets to be displayed in the high fanout. The default value is 10.

Block Creation (Available only when creating Designer Blocks)



Delete I/Os whenever possible - Deletes I/Os in the block during compile (except TRIBUFF and BIBUFF, because they cannot be removed). Useful if you have I/Os in your design but want to create a block anyway.

Add buffers on ports whose fanout is greater than <value> - Adds buffers on ports with a fanout greater than a value you specify. This option enables more predictable block timing. For example, if you have a net with a fanout of 100 the net will be unrouted. If you add a buffer, the output of the buffer is routed and the routing is preserved.

See Also

`compile`

Configure Flash*Freeze

Opens the Flash*Freeze Hardware Settings dialog box. For more information on the Flash*Freeze mode for SmartFusion2 see the SmartFusion2 Low Power 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

Place and Route - SmartFusion2, IGLOO2, RTG4

This topic describes Place and Route options for SmartFusion2, IGLOO2, RTG4 devices. See the topic for SmartFusion, IGLOO, ProASIC3 and Fusion for information about Place and Route options for those devices.

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 35 · Layout Options Dialog Box

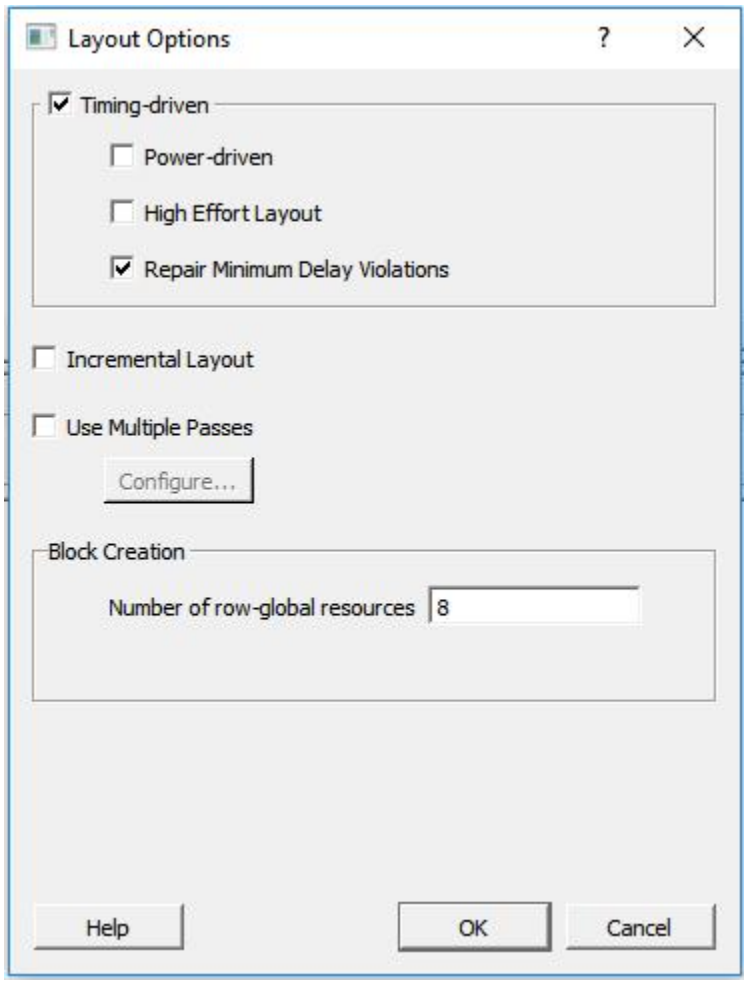


Figure 36 · Layout Options Dialog Box

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.

If you have set multiple timing constraint Scenarios in SmartTime, the Scenario selected for TDPR is used in Timing-Driven layout.

Power-Driven

Select this option to run Power-Driven layout. The primary goal of power-driven layout is to reduce dynamic power while still maintaining timing constraints.

High Effort Layout

Enable this option to optimize performance; layout runtime will increase if you select 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)
- Has the sink and driver pins located in the same cluster

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.

Alternatively, you can employ block-based design methodologies. Small designs can be placed, routed, tuned and then turned into blocks after timing constraints have been met. These blocks can then be imported into larger designs while ensuring that timing characteristics of the individual blocks are preserved. See SmartFusion2, IGLOO2, and RTG4 Block Flow 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).

See Also

SmartTime Constraint Scenario - SmartFusion2, IGLOO2, and RTG4

SmartFusion2, IGLOO2, and RTG4 Block Flow

Mutiple Pass Layout Configuration (SmartFusion2, IGLOO2, RTG4)

extended_run_lib

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 37 · 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_RNI0T36/U0_YWn	1848
5	GB[14]	(740, 156)	pll_i/GL2_INST/U0_YWn_GEast	1104
6	GB[2]	(724, 156)	serdes_i/SERDES_IF_0/EPCS_1_RX_CLK_keep_RNIEDL3/U0_YWn	514
7	GB[0]	(722, 156)	serdes_i/SERDES_IF_0/EPCS_0_RX_CLK_keep_RNID1J5/U0_YWn	513
8	GB[3]	(725, 156)	serdes_i/SERDES_IF_0/EPCS_2_RX_CLK_keep_RNIFPN1/U0_YWn	511
9	GB[7]	(729, 156)	serdes_i/pcs_gl_0_pcs_0/rx_rst_n_i_0_RNI9T3C/U0_YWn	312
10	GB[11]	(737, 156)	serdes_i/pcs_gl_1_pcs_0/rx_rst_n_i_0_RNIAGFA/U0_YWn	310
11	GB[1]	(723, 156)	serdes_i/SERDES_IF_0/EPCS_0_TX_CLK_keep_RNIFLD8/U0_YWn	178
12	GB[5]	(727, 156)	serdes_i/SERDES_IF_0/EPCS_2_TX_CLK_keep_RNIHDI4/U0_YWn	178
13	GB[6]	(728, 156)	serdes_i/SERDES_IF_0/EPCS_1_TX_CLK_keep_RNIG1G6/U0_YWn	178
14	GB[10]	(736, 156)	SPI_SCLK_ibuf_RNIT4T6/U0_YWn_GEast	149
15	GB[9]	(735, 156)	SRAM_CQ_ibuf_RNIB7IC/U0_YWn_GEast	111
16	GB[12]	(738, 156)	SRAM_CQn_ibuf_RNIPMM6/U0_YWn_GEast	18

Figure 38 · Gloabl 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_IOIN:Y	(6, 307)	GB[10]	SPI_SCLK_ibuf	ROUTED	1
2	SRAM_CQ	G16	DDRIO120NB2/MDDR_DQ_ECC0/CCC_NE1_CLKI3	SRAM_CQ_ibuf/U0/U_IOIN: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_IOPAD:Y	North IO #6 (1002, 313)	GB[12]	SRAM_CQn_ibuf	HARDWIRED	1

Figure 39 · 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 40 · 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 41 · 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	MSIO35PB5/CCC_NE0_CLK0	pll_i/CLK0_PAD_INST/U_IOPADP:IOUT_P	East IO #0 (1455, 112)	pll_i/CCC_INST/INST_CCC_IP:CLK0_PAD	CCC-NE0 (1428, 302)	fixed_clk	HARDWIRED	1

Figure 42 · 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 43 · 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 **Fanou**t 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_resetfifo:Y	(1313, 201)	I_ARRAY_CORE_ABC/un1_resetfifo_i	10	1	(1166, 187)	5
					2	(1166, 190)	5
2	I_ARRAY_CORE_DEF/I_DATABUS/I_DB_CONTROL/un1_resetfifo:Y	(1394, 228)	I_ARRAY_CORE_DEF/un1_resetfifo_i	10	1	(1166, 220)	5
					2	(1166, 223)	5
3	I_ARRAY_CORE_ABC/I_TGSLICE_CMP/I_C1_RESIDUE_FIFO/comb_reset:Y	(1176, 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_TGSLICED2/I_D2_RESIDUE_FIFO/comb_reset:Y	(1179, 291)	I_ARRAY_CORE_ABC/I_TGSLICED2/I_D2_RESIDUE_FIFO/comb_reset_Z	1		(1166, 181)	1
5	I_ARRAY_CORE_ABC/I_TGSLICED2/I_MATCH_FIFO/un1_resetoutl:Y	(1312, 201)	I_ARRAY_CORE_ABC/I_TGSLICED2/I_MATCH_FIFO/un1_resetoutl_i	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_i	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_i	1		(1166, 199)	1
8	I_ARRAY_CORE_ABC/I_OUTPUT_FIFO_WR_CTL/un1_resetsyncl:Y	(1191, 240)	I_ARRAY_CORE_ABC/I_OUTPUT_FIFO_WR_CTL/un1_resetsyncl_i	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_i	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_i	1		(1166, 247)	1
12	I_ARRAY_CORE_DEF/I_TGSLICED2/I_MATCH_FIFO/un1_resetoutl:Y	(1181, 252)	I_ARRAY_CORE_DEF/I_TGSLICED2/I_MATCH_FIFO/un1_resetoutl_i	1		(1166, 250)	1
13	I_ARRAY_CORE_DEF/I_OUTPUT_FIFO_WR_CTL/un1_resetsyncl:Y	(1081, 252)	I_ARRAY_CORE_DEF/I_OUTPUT_FIFO_WR_CTL/un1_resetsyncl_i	1		(1166, 253)	1
14	I_ARRAY_CORE_DEF/I_TGSLICED2/I_D2_RESIDUE_FIFO/comb_reset:Y	(1179, 255)	I_ARRAY_CORE_DEF/I_TGSLICED2/I_D2_RESIDUE_FIFO/comb_reset_Z	1		(1166, 265)	1

Figure 44 · 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_TGSLICED2/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_TGSLICED2/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 45 · 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 46 · 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_IOIN:Y	(3, 151)	MSIOD	GBR[16]	clk16_ibuf_Z	1
2	clktbout_ibuf/U0/U_IOIN:Y	(6, 151)	MSIOD	GBR[14]	clktbout_ibuf_Z	1
3	clkdary_a_c_ibuf/U0/U_IOIN:Y	(1524, 145)	MSIOD	GBR[15]	clkdary_a_c_ibuf_Z	2

Figure 47 · 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_TGSLICED2/I_D2_RESIDUE_FIFO/comb_reset:Y	(1179, 291)	RGRESET	I_ARRAY_CORE_ABC/I_TGSLICED2/I_D2_RESIDUE_FIFO/comb_reset_Z	1
5	I_ARRAY_CORE_ABC/I_TGSLICED2/I_MATCH_FIFO/un1_resetoutl:Y	(1312, 201)	RGRESET	I_ARRAY_CORE_ABC/I_TGSLICED2/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_TGSLICED2/I_MATCH_FIFO/un1_resetoutl:Y	(1181, 252)	RGRESET	I_ARRAY_CORE_DEF/I_TGSLICED2/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_TGSLICED2/I_D2_RESIDUE_FIFO/comb_reset:Y	(1179, 255)	RGRESET	I_ARRAY_CORE_DEF/I_TGSLICED2/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 48 · 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

Place and Route

The following topic applies to SmartFusion, IGLOO, ProASIC3 and Fusion families. See the SmartFusion2 and IGLOO2 Place and Route topic for information specific to those families.

Place and Route runs automatically using Timing-Driven Place and Route as the default during the push-button design flow in Libero SoC.

Custom Layout options are saved when you save your ADB after place and route. You must invoke Layout in interactive mode from Libero SoC and run Layout from Designer to view your custom Layout options. When your options are saved you can run Layout from Libero, but you must set your additional Layout Options in Designer. See below for more information on setting additional Layout Options when you open Interactively.

The I/O Bank Assigner and Global Planner run automatically after you click **OK** in the **Layout Options** dialog box. The I/O Bank Assigner automatically assigns technologies to all I/O banks that have not been assigned a technology. The Global Planner automatically assigns global nets to clock conditioning circuit (CCC) locations on the chip in the design.

Note: All I/O technologies assigned to I/O banks by the I/O Bank Assigner in Layout are unlocked.

To change your Place and Route settings:

Expand **Implement Design**, right-click **Place and Route** and choose **Configure Options**.

Place and Route Options

Timing-Driven

Timing-Driven place and route is selected by default. The primary goal of timing-driven layout is to meet timing constraints, specified by you or generated automatically. Timing-driven layout typically delivers better performance than Standard layout.

If you do not select Timing-driven layout, Designer runs Standard layout. Standard layout targets efficient usage of the chip resources. Chip performance is not optimized. Timing constraints are not considered by the layout in standard mode, although a delay report based on delay constraints entered in SmartTime can still be generated for the design. This is helpful to determine if Timing-Driven layout is required.

If you have set multiple timing constraint Scenarios in SmartTime, you can select a scenario from the pull-down list to perform timing-driven layout. Timing constraints from the Scenario you select will be used in Timing-Driven layout.

Place and Route Incrementally

Select this option to use previous placement data as the initial placement for next placement run. Additionally, this will preserve previous placement data during the next incremental placement run.

Router will also be run incrementally. Select to fully route a design when some nets failed to route during a previous run. Incremental routing should only be used if a low number of nets fail to route (less than 50 open nets or shorted segments). A high number of failures usually indicates a less than optimal placement (if using manual placement through macros, for example) or a design that is highly connected and does not fit in the device. If a high number of nets fail, relax constraints, remove tight placement constraints, deactivate timing-driven mode, or select a bigger device and rerun Layout. Also, see the Advanced Layout options for your device.

You can also use it when the post-synthesis netlist has undergone a minor or incremental change.

Additional Layout Options Available if you Open Interactively

You must invoke Layout in interactive mode from Libero SoC and run Layout from Designer to view your custom Layout options. To open Additional Layout Options interactively, in the **Design Flow** window right-click **Layou**t and choose **Open Interactively**.

Lock Existing Placement (Fix)

Locks your existing placement. Use this option if you do not want any changes in your layout.

Power-Driven

Select this option to run Power-Driven Layout. The primary goal of power-driven layout is to reduce dynamic power while still maintaining timing constraints. This option is available when you select Timing Driven Layout.

To get the most out of Power-Driven Layout:

1. Enter maximum delay, minimum delay, setup, and hold constraints in SmartTime's constraint editor or in SDC.
2. Set false paths on any paths that have a constraint, but do not need one (this will help layout meet the constraints that are needed).
3. Perform Layout with **Timing-Driven**, **Run Place**, and **Run Route** options checked.
4. Resolve worst case setup and maximum delay violations.
5. Generate an SDF back-annotation file.
6. Perform a post layout back-annotated simulation using this SDF file, and export a VCD (Value Change Dump) file that will capture real activities for each net.
7. Open smartPower and import the VCD file using **Simulation > Import VCD File**.
8. Perform Layout with **Timing-Driven** and **Power-Driven** checked. Run Place and Route.
9. Verify that your timing constraints are still met with SmartTime.
10. Analyze your power with SmartPower.

In case you do not have simulation vectors for your design, the following alternative flow is recommended:

1. Enter maximum delay, minimum delay, setup, and hold constraints in SmartTime's constraint editor or in SDC.
2. Set false paths on any paths that have a constraint, but do not need one (this will help layout to meet the constraints that are needed).
3. Perform Layout with **Timing-Driven**, **Run Place**, and **Run Route** options checked.
4. Resolve worst case setup and maximum delay violations.
5. Verify that your timing constraints are still met with SmartTime.
6. Open SmartPower and set clock frequencies and toggle rates for the different clocks. Clock frequencies can be imported from your timing constraints. Refer to Initialize Frequencies for more information.
7. Perform Layout with **Timing-Driven**, and **Power-Driven** options checked. Run Place and Route.
8. Verify that your timing constraints are still met with SmartTime.
9. Analyze your power with SmartPower

Run Place

Select this option to run the placer during Layout. By default, it reflects the current Layout state. If you have not run Layout before, Run Place is selected by default. If your design has already been placed but not routed, this box is cleared by default. You can also select the following incremental placement options.

- **Incrementally**: Select to use previous placement data as the initial placement for the next place run.
- **Lock Existing Placement (fix):** Select to preserve previous placement data during the next incremental placement run.

Incremental options apply to the entire design. For more detailed control of the placer behavior (such as, to fix placement of a portion of the design), use the MultiView Navigator tools or set fixed attributes on the placed instances via PDC constraint files.

Run Route

Select to run the router during Layout. By default, it reflects the current Layout state. If you have not run Layout before, Run Route is checked. Run Route is also checked if your previous Layout run completed with routing failures. If your design has been routed successfully, this check box is cleared.

- **Incrementally**: Select to fully route a design when some nets failed to route during a previous run. You can also use it when the incoming netlist has undergone an ECO. (Engineering Change Order). Incremental routing should only be used if a low number of nets fail to route (less than 50 open nets or shorted segments). A high number of failures usually indicates a less than optimal placement (if using manual placement through macros, for example) or a design that is highly connected and does not fit in the device. If a high number of nets fail, relax constraints, remove tight placement constraints, deactivate timing-driven mode, or select a bigger device and rerun Layout. Also, see the Advanced Layout options for your device.

There is no "Fix" option for the router. In incremental mode the router tries to preserve the existing routing; there is no guarantee that it will be preserved. Therefore the timing characteristics of the previously routed portion of the design may change, even if the placement was fixed for that portion of the design. The chance of this is quite small, and the router will print the list of nets that have fixed terminals (i.e. those nets whose every pin's macro has the placement FIX attribute).

Use Multiple Passes

Select to run layout multiple times with different seeds. Multiple Pass Layout attempts to improve layout quality by selecting from a greater number of layout results. Click **Configure** to set your Multiple Pass Configuration.

Click the Advanced button to set Timing-Driven options.

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.

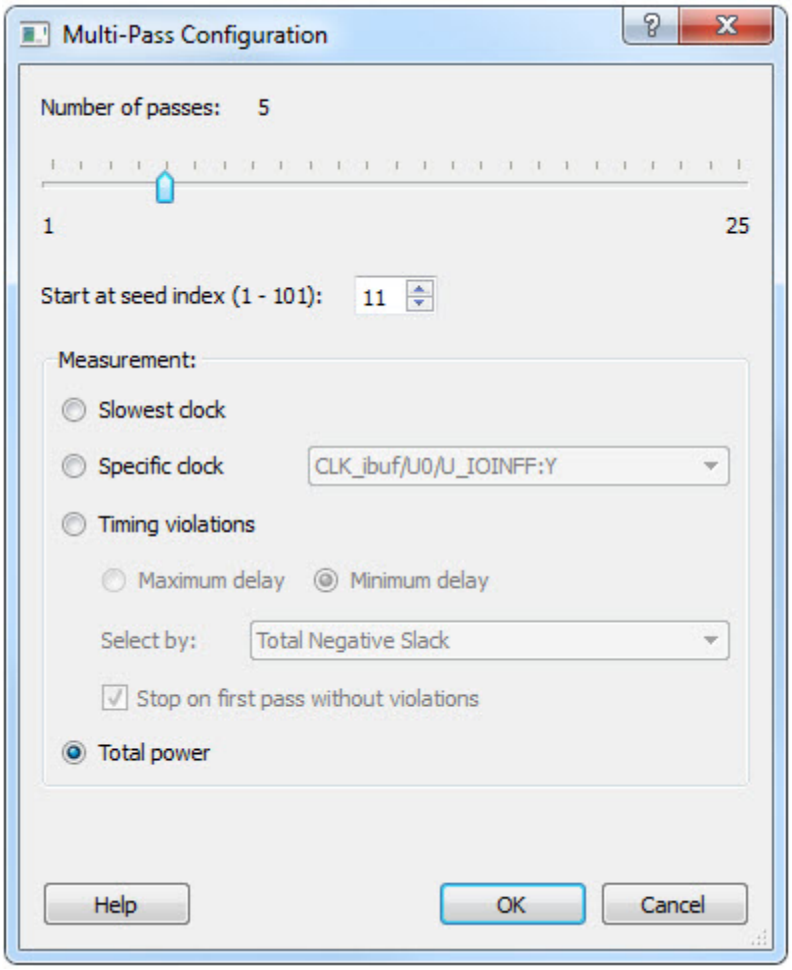


Figure 49 · Multi-Pass Configuration Dialog Box

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.

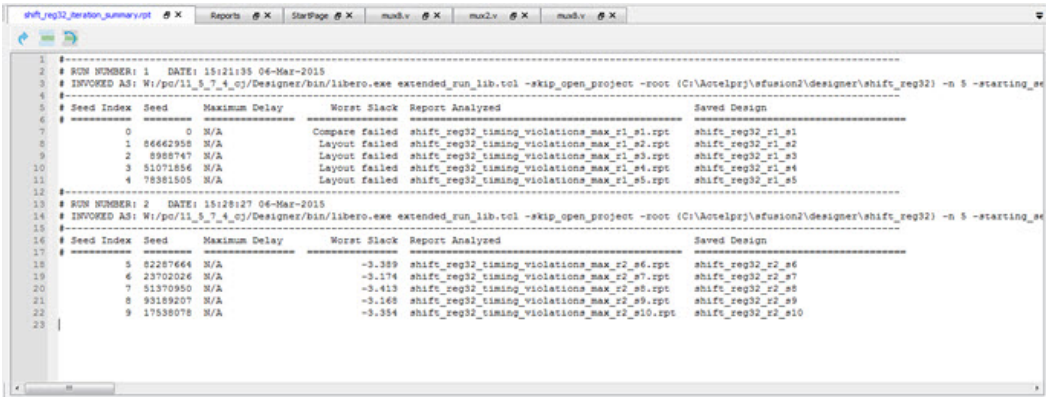
Note: You must enter your own timing constraints through SmartTime or SDC.

- **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.



```
#-----------------------------------------------------------------------------------------------------------------------
# RUN NUMBER: 1   DATE: 15:21:35 04-Mar-2015
# INVOKED AS: W:/pc/11_5_7_4_cj/Designer/bin/libero.exe extended_run_lib.tcl -skip_open_project -root (C:\Actelprj\sfusion2\designer\shift_reg32) -n 5 -starting_se
#-----------------------------------------------------------------------------------------------------------------------
# Seed Index  Seed      Maximum Delay        Worst Slack  Report Analyzed                                  Saved Design
# ==========  =======   ===============      ===========  ===============================================  ================================
          0        0   N/A                   Compare failed  shift_reg32_timing_violations_max_r1_s1.rpt    shift_reg32_r1_s1
          1  86662958   N/A                    Layout failed  shift_reg32_timing_violations_max_r1_s2.rpt    shift_reg32_r1_s2
          2   8988747   N/A                    Layout failed  shift_reg32_timing_violations_max_r1_s3.rpt    shift_reg32_r1_s3
          3  51071856   N/A                    Layout failed  shift_reg32_timing_violations_max_r1_s4.rpt    shift_reg32_r1_s4
          4  78381505   N/A                    Layout failed  shift_reg32_timing_violations_max_r1_s5.rpt    shift_reg32_r1_s5
#-----------------------------------------------------------------------------------------------------------------------
# RUN NUMBER: 2   DATE: 15:28:27 04-Mar-2015
# INVOKED AS: W:/pc/11_5_7_4_cj/Designer/bin/libero.exe extended_run_lib.tcl -skip_open_project -root (C:\Actelprj\sfusion2\designer\shift_reg32) -n 5 -starting_se
#-----------------------------------------------------------------------------------------------------------------------
# Seed Index  Seed      Maximum Delay        Worst Slack  Report Analyzed                                  Saved Design
# ==========  =======   ===============      ===========  ===============================================  ================================
          5  82287664   N/A                       -3.389  shift_reg32_timing_violations_max_r2_s6.rpt    shift_reg32_r2_s6
          6  23702026   N/A                       -3.174  shift_reg32_timing_violations_max_r2_s7.rpt    shift_reg32_r2_s7
          7  51370950   N/A                       -3.413  shift_reg32_timing_violations_max_r2_s8.rpt    shift_reg32_r2_s8
          8  93189207   N/A                       -3.168  shift_reg32_timing_violations_max_r2_s9.rpt    shift_reg32_r2_s9
          9  17538078   N/A                       -3.354  shift_reg32_timing_violations_max_r2_s10.rpt   shift_reg32_r2_s10
```

Figure 50 · Iteration Summary Report

See Also

Place and Route - SmartFusion2, IGLOO2, RTG4

extended_run_lib

SmartFusion, IGLOO, ProASIC3 and Fusion Place and Route Advanced Options

To set these advanced options during Layout, click **Advanced** in the Layout dialog box. The Advanced Layout options are only available in timing-driven Layout mode.

High Effort Layout Mode

This option turns on netlist optimizations to obtain better performance. Layout runtime will increase when this option is selected. You can also combine this option with the Multi-Pass mode to achieve the best possible performance.

In the regular flow the compile step in Designer would modify the netlist to make use of efficient resources on the chip, such as global networks and special macros. When the **High Effort Layout** option is turned on, the placer could further change the mapping of the logic components, preserving the original functionality of the design. The changed netlist is then used in all post-layout Designer tools including back-annotation.

The names and types of the combinational core logic primitives may change. All other logic cells (such as registers, memory, I/Os or clocks) or combinational logic primitives that are assigned a physical constraint (locked at a location, assigned to a region, or part of a block component), referred in a timing constraint, or have a preserve property, will remain unchanged.

When the **Lock Existing Placement** option is also turned on, the placer runs in regular effort mode.

Note: If you change the High Effort Setting you must rerun Place and Route to complete Layout.

Sequential Optimization

This option turns on optimization of sequential cells in the High Effort Layout mode. This typically enables register retiming without disturbing timing latency. The names of registers may change unless they are assigned a physical constraint (locked at a location, assigned to a region, or part of a block component), referred in a timing constraint, or have a preserve property. Other restrictions may also apply.

The following cases are excluded from sequential optimization:

- Registers that have any timing constraint other than global FMAX, TSU (setup time) or TCO (clock to out). Registers referred by multi-cycle or exception timing constraints are not moved.
- Registers that feed asynchronous control signals on another register.
- Registers feeding the clock of another register.
- Registers feeding a register in another clock domain.
- Registers that are fed by a register in another clock domain.
- Registers connected to PLL.
- Registers that have PDC attribute “preserve”, assigned a physical constraint (locked at a location, assigned to a region, or part of a block component).
- Both registers in a direct connection from input I/O-to-register-to-register if both registers have the same clock and the first register does not fan out to anywhere else. These registers are considered synchronization registers.
- Both registers in a direct connection from register-to-register if both registers have the same clock, the first register does not fanout anywhere else, and the first register is fed by another register in a different clock domain. These registers are considered synchronization registers.

Router

Repair Minimum Delay Violations

With this option selected, layout will perform an additional route that will attempt 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 paths. Since placement will remain unchanged and no additional tiles or modules will be inserted, the amount of delay inserted is limited. As a result, this function is best suited to repair paths with small (0 to 3 ns) hold and minimum delay violations. Paths with large violations will likely improve, but for a complete repair of these paths, manual placement or source code modification may be necessary. Every effort will be made to avoid creating max-delay timing violations on worst case paths.

To get the most out of repair minimum delay violations:

1. Enter max-delay, min-delay, setup and hold constraints in SmartTime's constraint editor or in SDC.
2. Set false paths on any paths that have a constraint, but do not need one (this will help layout to meet the constraints that are needed).
3. Perform Layout with **Timing Driven**, **Run Place**, **Run Route** and optionally **Run incrementally** enabled.
4. Resolve worst case setup and max-delay violations before running minimum delay violations repair.
5. After worst case max-delay timing is resolved, evaluate timing in SmartTime's Timing Analyzer in minimum delay analysis mode to check for hold time and minimum delay violations.
6. Run repair minimum delay violations with incremental route enabled.
   The repair minimum delay violations tool will attempt to fix all hold time and minimum delay violations by lengthening routing delay paths and inserting routing buffers. As delay is added to paths, worst case max-delay timing is verified to avoid creating new max-delay timing violations. Designer will report the worst minimum slack and the number of violating paths in the log window. In some cases, additional improvement can occur by running repair minimum delay violations multiple times with **Run Incrementally** enabled.
7. Perform both maximum and minimum delay timing analysis to check the timing. Manual placement or source code modification may be necessary to repair all minimum delay violations.
8. After making placement or source code changes, run incremental route and repair minimum delay violations, and then analyze timing again.

Additional Factors

Runtime may vary greatly with the number of paths that need repair, the number of nets in those paths, and the resources available for the tool to insert delay. Over-constraining paths will increase runtime, but will not likely improve results .

The tool will only work on paths that have min delay and hold time constraints. However, other paths that share common nets to the constrained paths may be inadvertently affected.

It is recommended to run minimum delay violations repair with incremental route. This will ensure that paths which do not have minimum delay violations are preserved.

Repair will be performed on:

- Register to register paths where both registers are on the same global or non-global clock
- Register to register paths where the registers are on different clock networks and a minimum delay constraint exists
- Input to register, register to output, clock to out, input to output paths with minimum delay or hold constraint.

You may select programmable input delays to increase delay on input to register paths for devices that support the feature.

Restore Defaults

Click Restore Defaults to run the factory default settings for Advanced options.

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 µs and the Wave window opens to display the simulation results.
5. Scroll in the Wave window to verify the logic works as intended. Use the cursor and zoom buttons to zoom in and out and measure timing delays. 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**.

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.

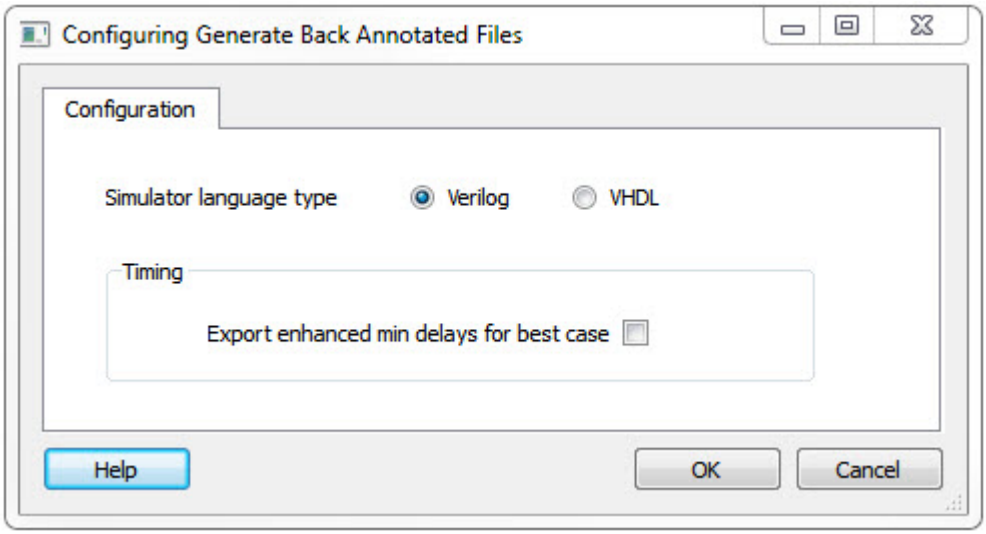


Figure 51 · Configuring Generate Back Annotated Files Dialog Box

Generate Bitstream

Generates the bitstream for use with the Run PROGRAM Action tool.

The tool incorporates the Fabric design, eNVM configuration (if configured) and security settings (if configured) to generate the bitstream file. You need to configure the bitstream before you generate the bitstream. 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.

This operation is completed automatically as the last step if you use the Build button.

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.

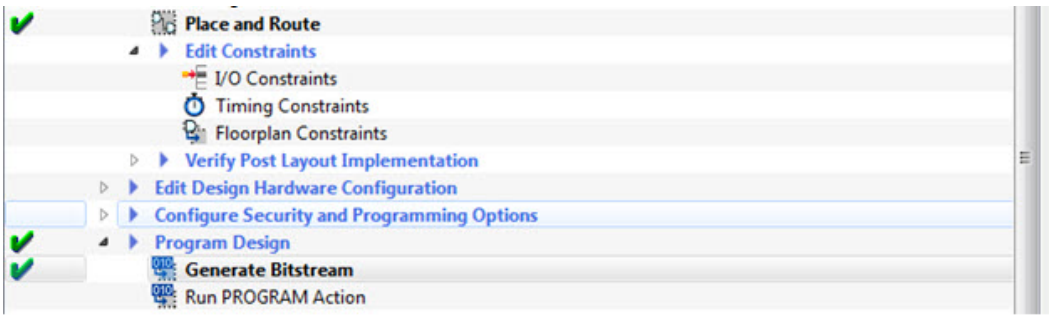


Figure 52 · Generate Bitstream (Complete)

See also

Configure Bitstream Dialog Box

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.

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 figures show examples of the Verify Timing Configuration dialog box for various operating conditions and report selections.

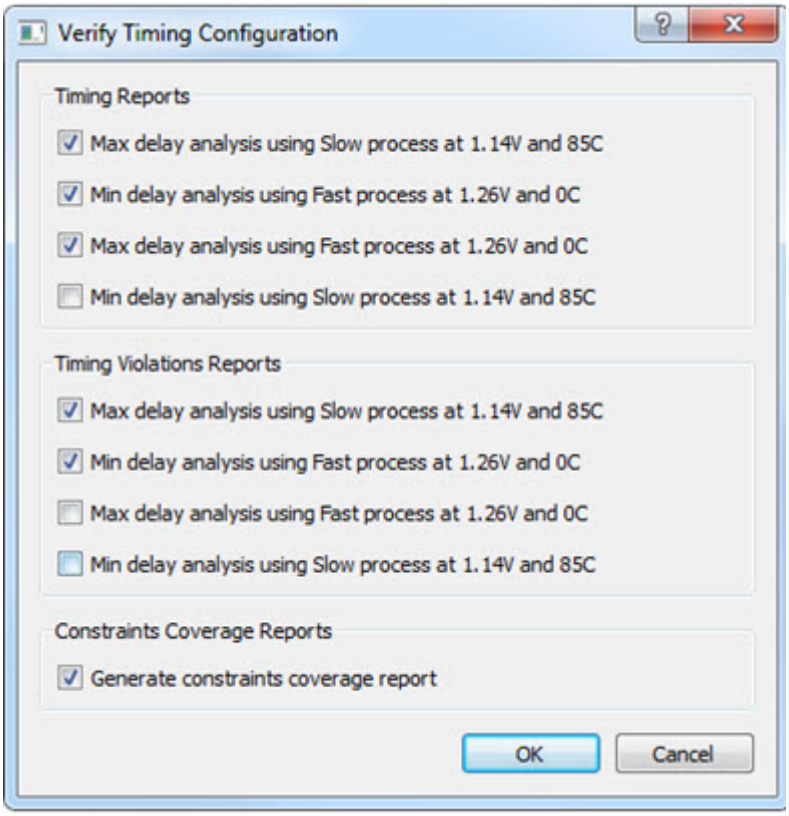


Figure 53 · Temperature/Voltage Value for Operating Conditions: COM (Commercial)

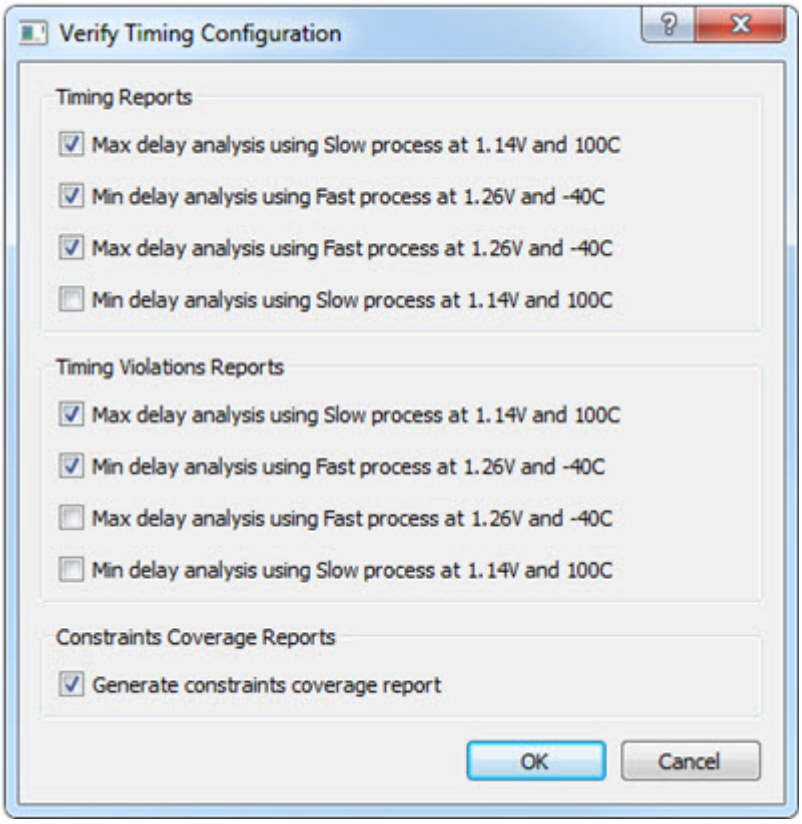


Figure 54 · Temperature/Voltage Value for Operating Conditions: IND (Industrial)

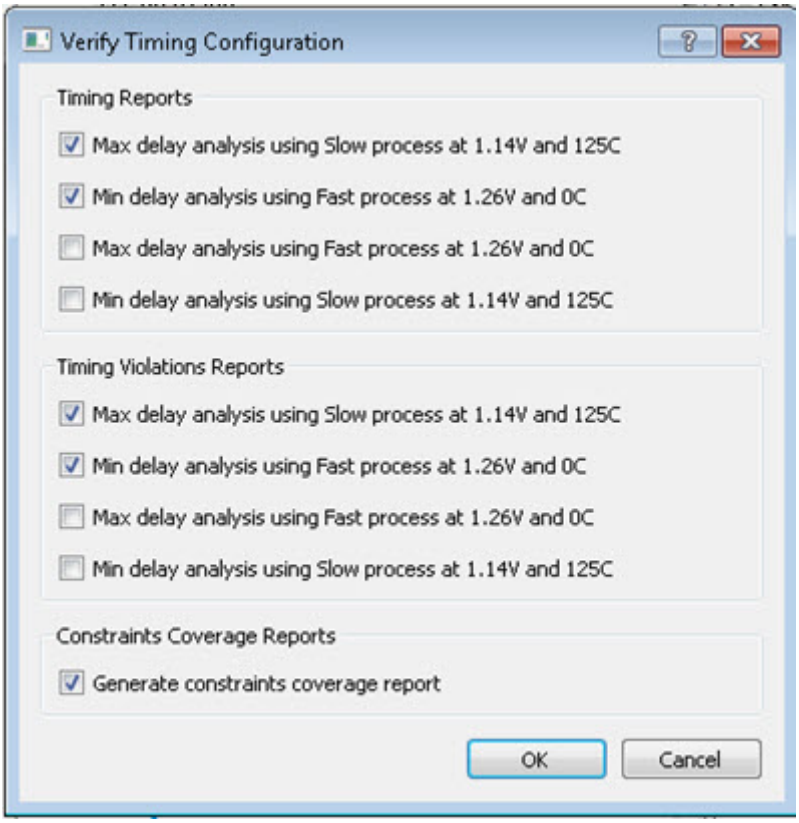


Figure 55 · Temperature/Voltage value for CUSTOM Temperature (0°C to 125°C) and IND Voltage

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)

- o 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 boxare 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.

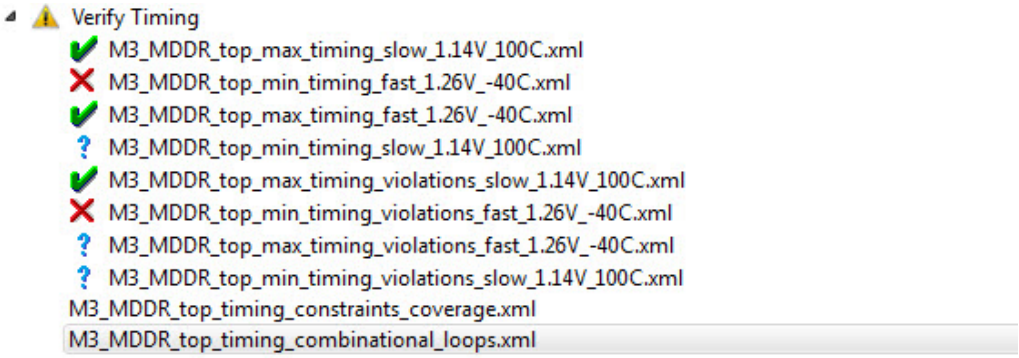


Figure 56 · Reports Example

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

See Also

Generating timing report

Generating timing violation report

Generating a datasheet report

Generating a bottleneck report

Generating a constraints coverage report

Generating a Combinational Loop Report

Verify Timing with SmartTime

SmartTime is a gate-level static timing analysis (STA) tool for SmartFusion, IGLOO, ProASIC3, Fusion, SmartFusion2, IGLOO2, RTG4. With SmartTime, you can enter timing constraints and perform complete timing analysis (Min Delay and Max Delay Analysis) of your design to ensure that you meet all timing constraints and that your design operates at the desired speed with the right amount of margin across all operating conditions.

For detailed information about SmartTime, refer to the following documents:

SmartTime User Guide (Classic Constraint Flow) - SmartFusion2, IGLOO2, and RTG4

SmartTime User Guide (Classic Constraint Flow) - SmartFusion, IGLOO, ProSAIC3, and Fusion

Verify Power with 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 Model*Sim* or other simulators, and then utilize the switching activity-over-time results to evaluate average and peak power consumption for your design.

For detailed information about SmartPower, refer to the 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 for the Classic Constraint Flow, from Libero SoC's Design Flow window, right-click **IO Advisor** and choose **Open Interactively** (**IO Advisor > Open Interactively**) or double-click **IO Advisor**.

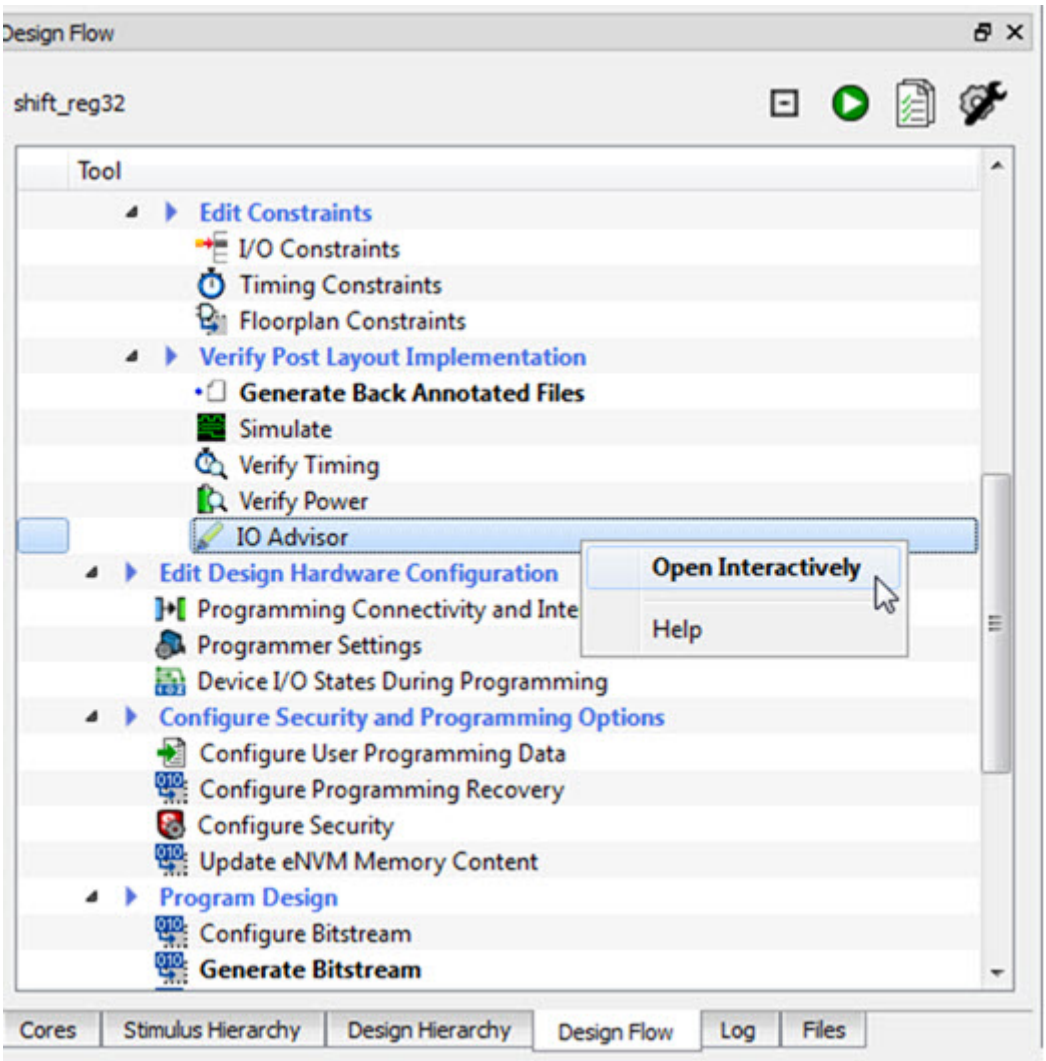


Figure 57 · IO Advisor (Classic Constraint Flow)

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.

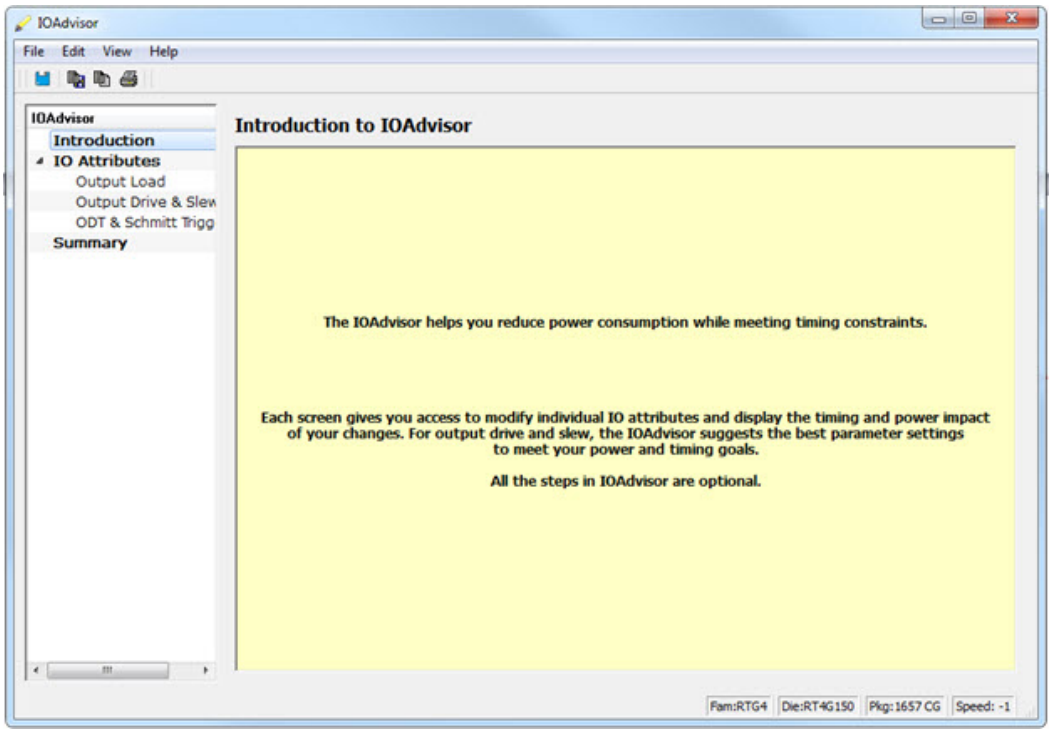


Figure 58 · IO Advisor - Introduction

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.

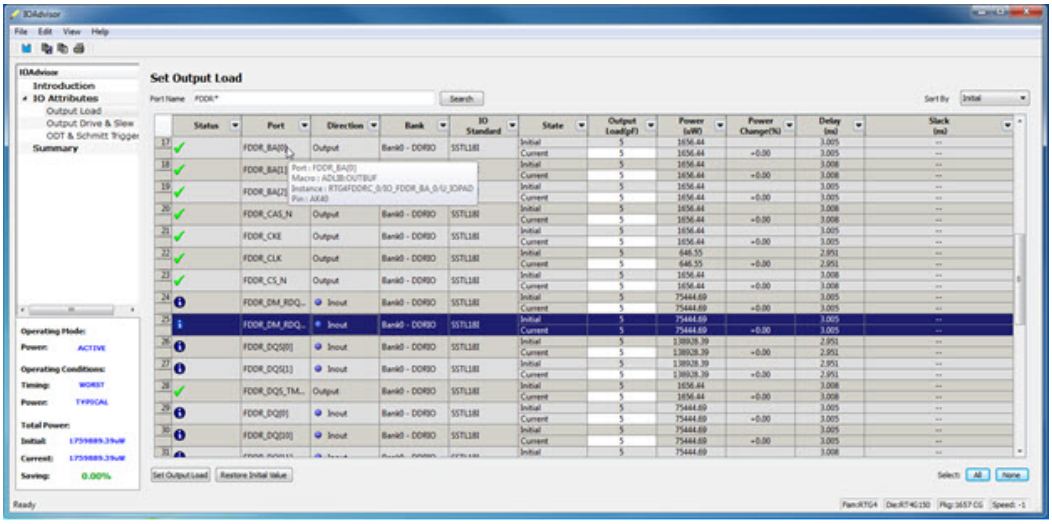


Figure 59 · IO Advisor - Output Load Panel

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.

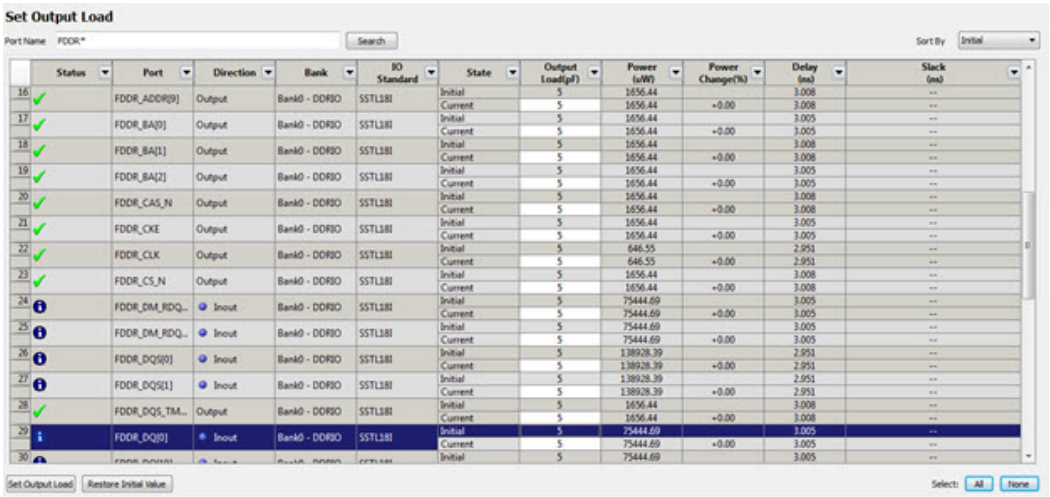


Figure 60 · Search Field and Regular Expressions

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.

Icon	Status and Explanation
	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.

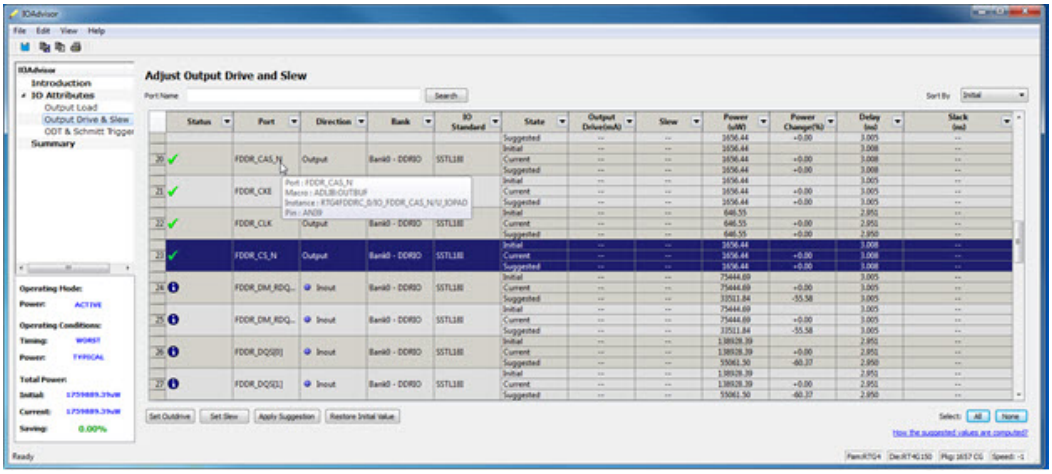


Figure 61 · IO Advisor – Output Drive and Slew

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 mulitple 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.

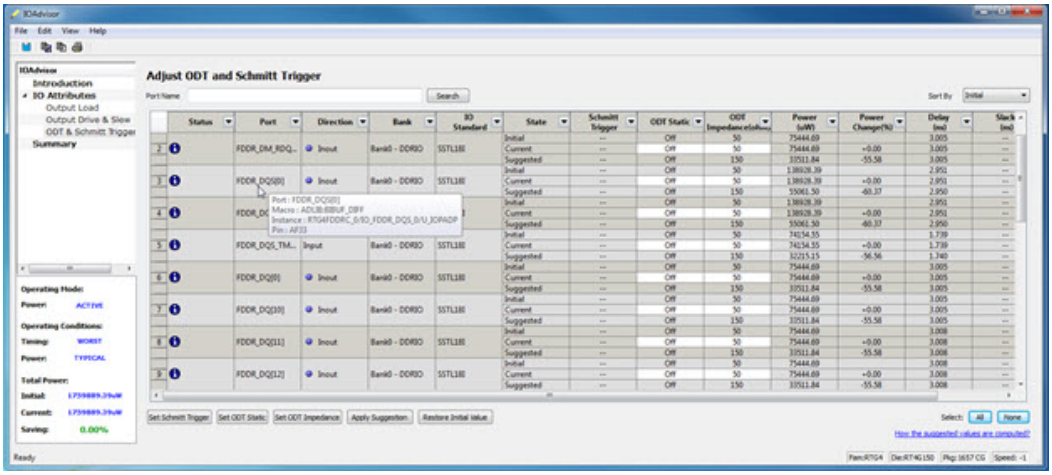


Figure 62 · IO Advisor – Adjust ODT and Schmitt Trigger

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.

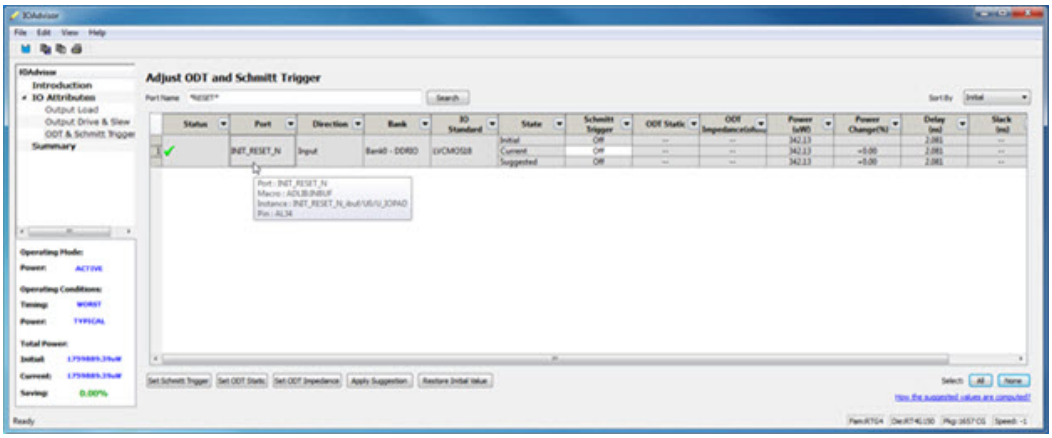


Figure 63 · Search Field and Regular Expressions

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.

Icon	Status and Explanation
	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 mulitple 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.

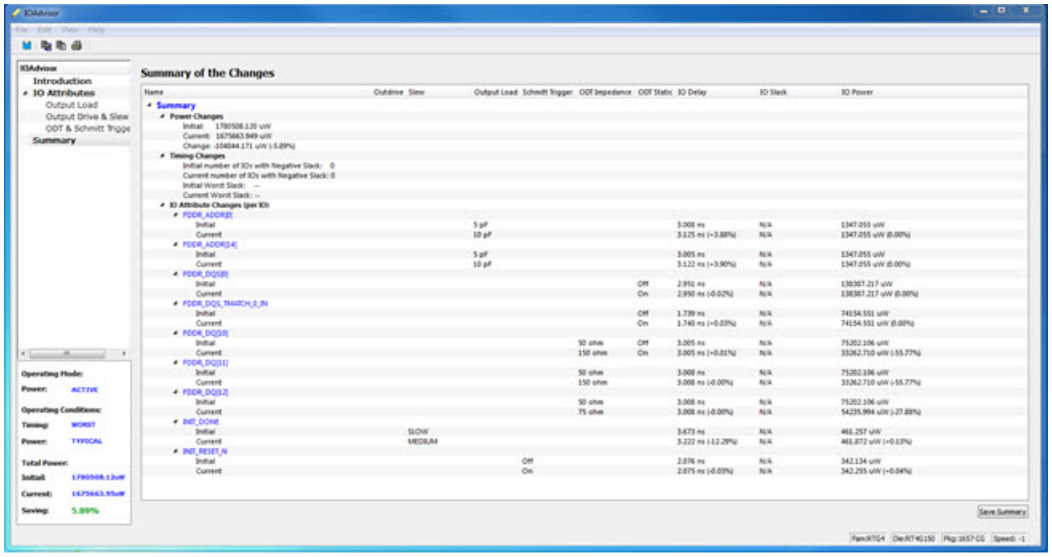


Figure 64 · IO Advisor – Summary

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 and IGLOO2 families.

Supported Die/Package

Family	Die	Package
IGLOO2	M2GL090	FG676
	M2GL060	FG676
	M2GL050	FG896
	M2GL025	FG484
	M2GL010	FG484
SmartFusion2	M2S090	FG676
	M2S060	FG676
	M2S050	FG896
	M2S025	FG484
	M2S010	FG484

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

Supported I/O Standard

The SSN Analyzer supports the following I/O Standards:

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

Supported I/O Types

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

SSN Analyzer

Three tabs are available in the SSN Analyzer:

- Noise Report
- Excluded 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 LVTTL.
- **Drive Strength (mA)** – Drive Strength selections are available from 2 to 12.
- **Static** – When this checkbox is checked, the I/O is considered neither as an Aggressor nor as a Victim. It is excluded from SSN Analysis.
- **Don't Care** – When this checkbox is checked, the I/O is excluded from consideration as a Victim for Noise Margin computation. However, it is considered as an Aggressor for Noise Margin computation of other I/Os.

Note: Static and Don't Care are mutually exclusive.

- **Noise Margin (%)** – This is the Noise Margin number computed by the SSN Analyzer. A negative number (shown in red) indicates that it is outside the guideline of SSN analysis.
- **Within Guideline**– Either Yes (Positive Noise Margin) or No (Negative Noise Margin). The Yes (within guideline) or No (outside guideline) guideline is different for different I/O standards:
    - LVTTL/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 VIH_{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 VIL_{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 VIH_{min} and a pulse width of less than or equal to 1 ns
    - Noise margin violating the criteria for "Yes" is considered to fall outside the specified guidelines, and is reported as a "No"

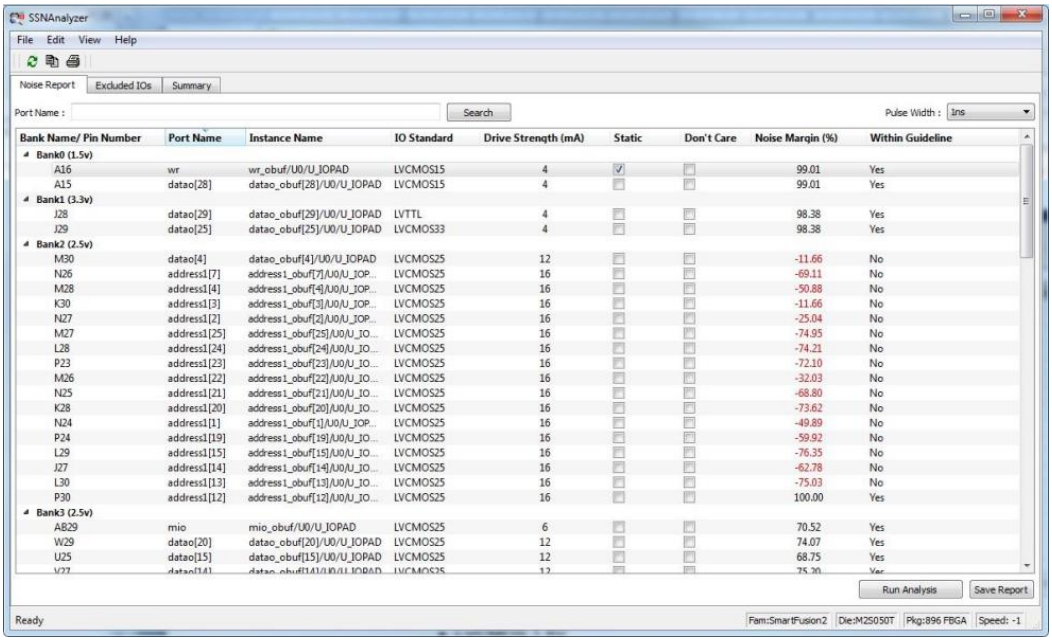


Figure 65 · SSN Analyzer – Noise Report Tab

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 to 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/0s 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

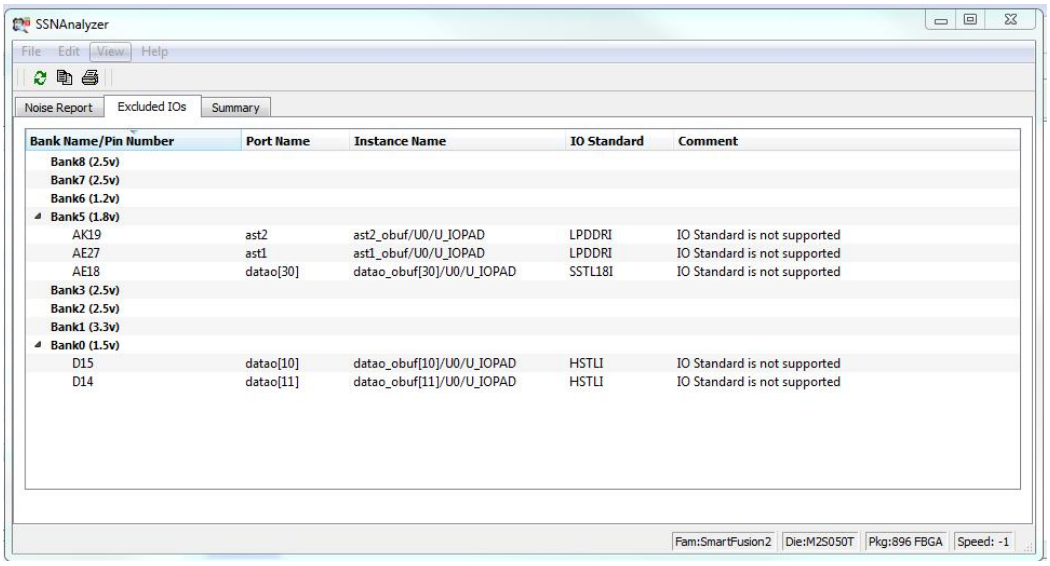


Figure 66 · SSN Analyzer – Excluded I/Os Tab

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.

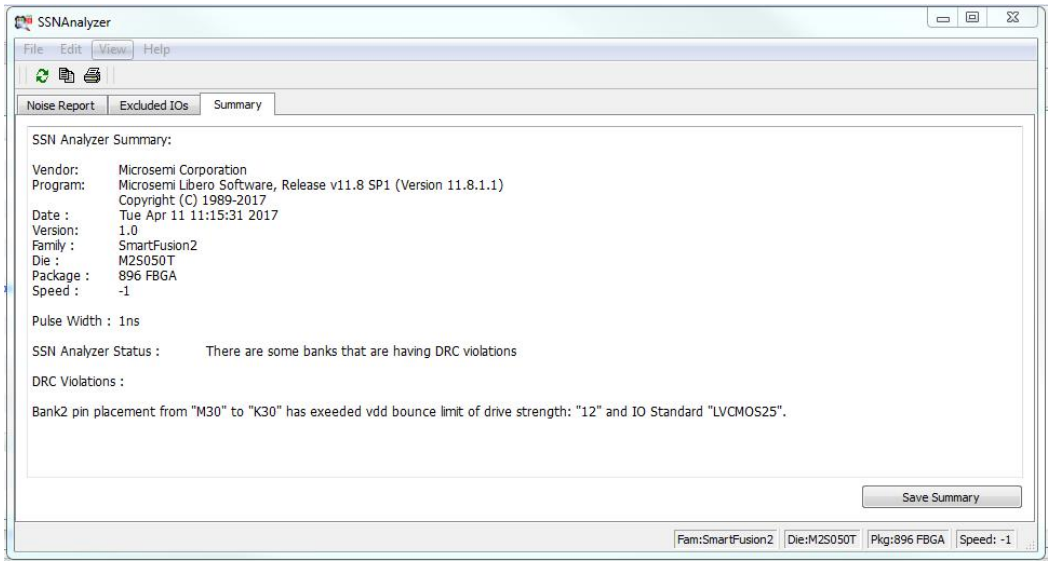


Figure 67 · SSN Analyzer - Summary

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.

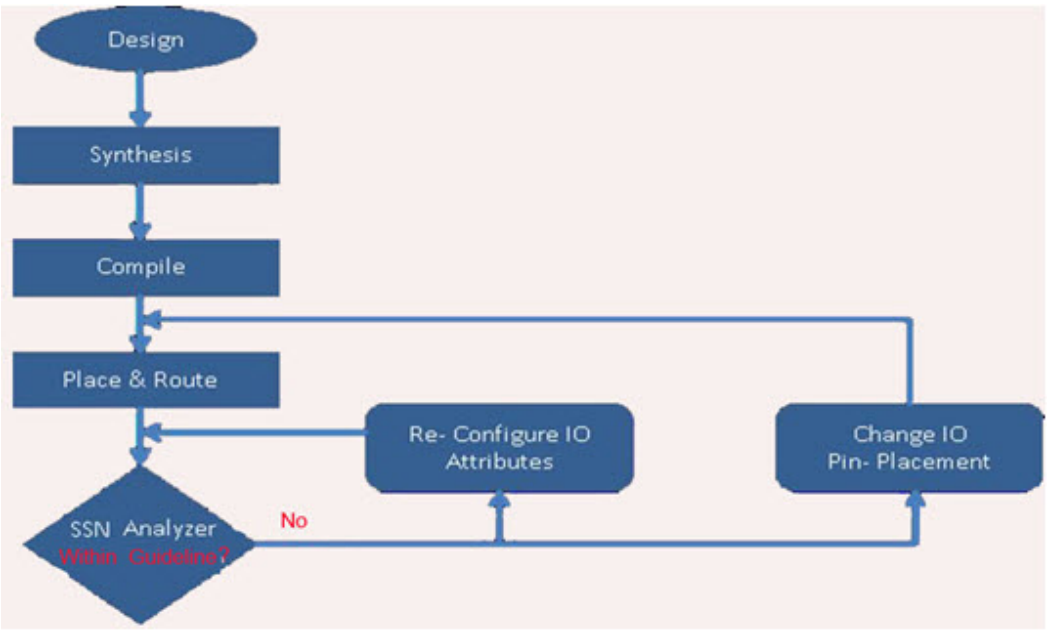


Figure 68 · SSN Analyzer in the Design Flow

See Also

Simultaneous Switching Noise and Signal Integrity Application Notes

Generate Back Annotated Files - SmartFusion2, IGLOO2, and RTG4

Generates Back Annotated files for your design.

Back Annotated files include:

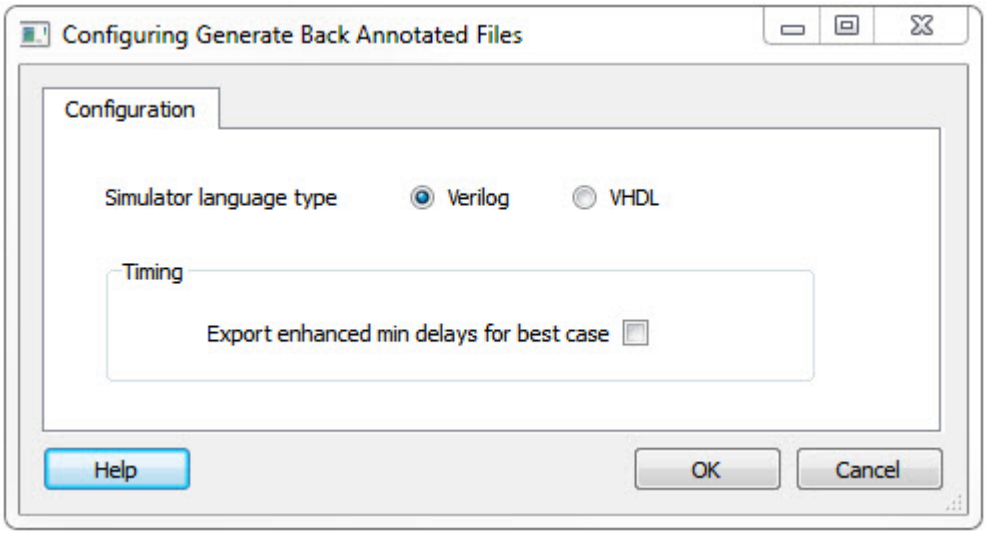
- *ba.sdf - Standard Delay Format for back-annotation to the simulator.
- *ba.v/.vhd - Post-layout flattened netlist used exclusively for back-annotated timing simulation. May contain low level macros not immediately recognizable to you; these were added by the software to improve your design performance.

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.


Configuring Generate Back Annotated Files
Configuration
Simulator language type
Verilog
VHDL
Timing
Export enhanced min delays for best case
Help
OK
Cancel

Device Programming

Default Programming Data File (*.pdb file) is generated automatically as part of the Libero SoC push-button design flow.

SmartFusion, IGLOO, ProASIC3 and Fusion devices use the FlashPoint program file generator to create a programming file. For SmartFusion, IGLOO, and ProASIC3, the FlashPoint interface also enables the advanced security features.

To open FlashPoint, expand **Implement Design**, right-click **Generate Programming Data** and choose **Open Interactively**, which will open Designer. Click **Programming File** within Designer to open FlashPoint. You can generate the following programming files:

- IEEE 1532 files (*.bsd, *.isc)
- DirectC files (*.dat)
- Programming Data File (*.pdb)
- STAPL file (*.stp)
- Serial Vector Files (*.svf)

You must have completed your design to generate your programming (*.stp or STAPL) file.

See Also

Generate a DAT file

Note: The instructions in this section are for PC only, and are not supported for SmartFusion on Linux.

Default Programming Data (*.fdb) is generated automatically as part of the Libero SoC push-button design flow.

This step generates a FDB file that contains the FPGA array only. To add security to the design, you must use the FlashPro application. Right-click **Program Device** or **Export Programming File** to open FlashPro. When FlashPro opens, choose **Modify PDB** to add the security settings.

To generate other programming files for the Fusion devices, open FlashPro.

1. When FlashPro opens, click **File > Export > Export Single Programming File** to open the Export Programming Files dialog box.

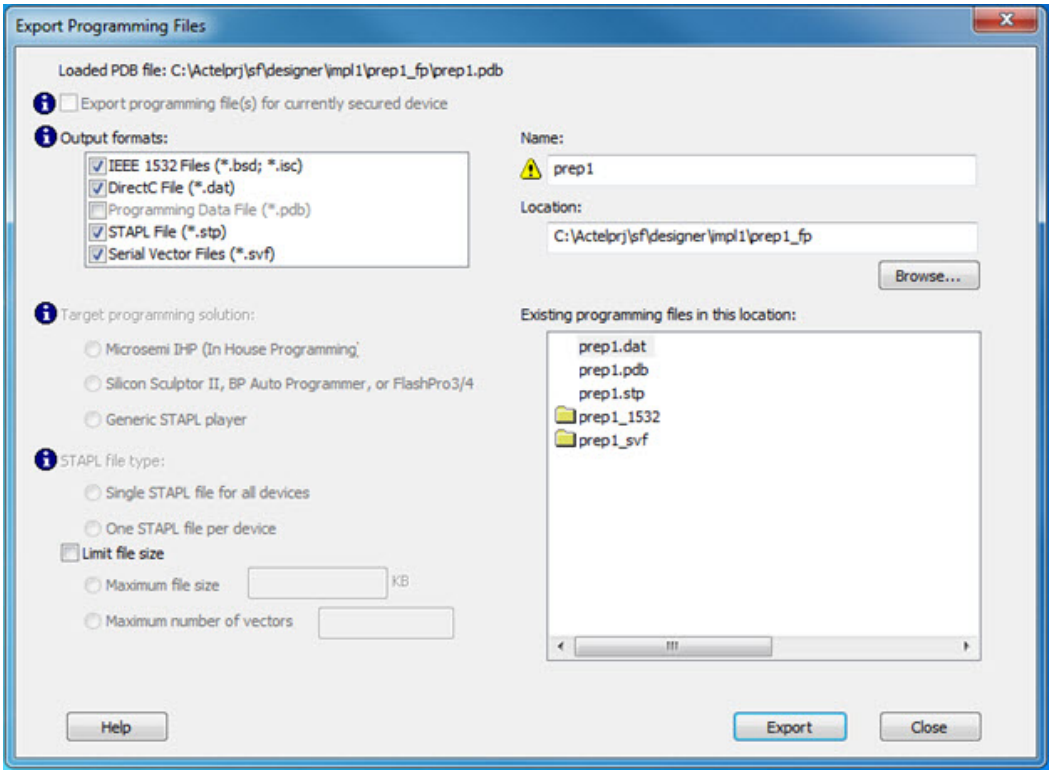


Figure 69 · Export Programming Files Dialog Box

2. Select the Programming File output format you want to generate:

- IEEE 1532 Files (*.bsd; *.isc)
- DirectC File (*.dat)
- STAPL File (*.stp)
- Serial Vector Files (*.svf)

3. Click **Export** to generate the programming files and then click **Close**.

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:

- **Select Programming Mode** – Select JTAG or SPI Slave mode. SPI Slave mode is only supported by FlashPro5.
- **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 Device** – 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 information if you hover your pointer over a device in the grid:

- **Name** - Editable field for a 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 Length** - Device instruction length.
- **TCK** - Maximum clock frequency in MHz 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.

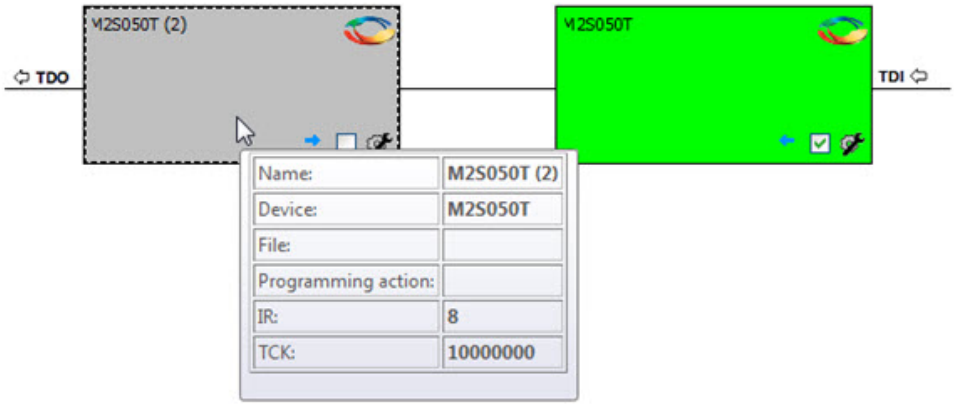


Figure 70 · Device Information

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 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.
- **Enable Device for Programming** - Select to enable the device for programming; enabled devices are green, disabled devices are gray.
- **HIGH-Z** - Sets disabled Microsemi SoC ProAsic3, IGLOO, Fusion, and SmartFusion devices in the chain to HIGH-Z (tri-states all the I/Os) during chain programming of enabled Microsemi devices in the daisy chain (not supported for Libero SoC target design device)
- **Configure Device** – Ability to reconfigure the device (for a Libero SoC target device the dialog appears but only the device name is editable)
- **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 Program Procedure/Actions** (Not supported for Libero SoC target design device):
    - **Actions** - List of programming actions for your device.
    - **Procedures** - Advanced option; enables you to customize the list of recommended and optional procedures for the selected Action.
- **Move Device Left/Right** – Move device in the chain to left or right.

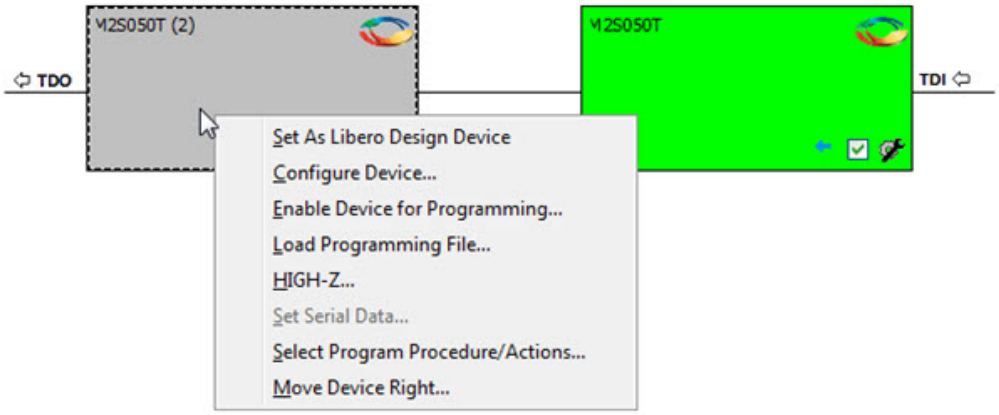


Figure 71 · Right-click Properties

Programmer Settings

In the Libero SoC Design Flow window, expand **Edit Design Hardware Configuration** and double-click **Programmer Settings** to view the name, type, and port. The dialog box displays information about your programmer if it is connected.

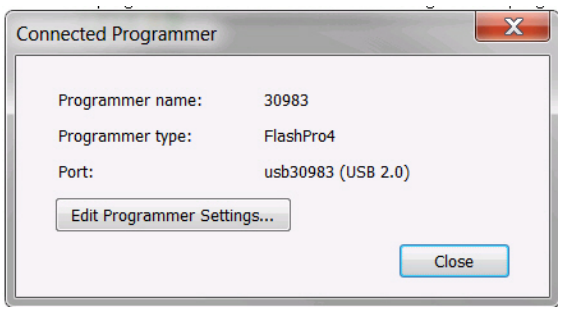


Figure 72 · Programmer Settings for Connected Programmer

Click **Edit Programmer Settings** to view the Programmer Settings dialog box. You can set specific voltage and force TCK frequency values for your programmer in this dialog box.

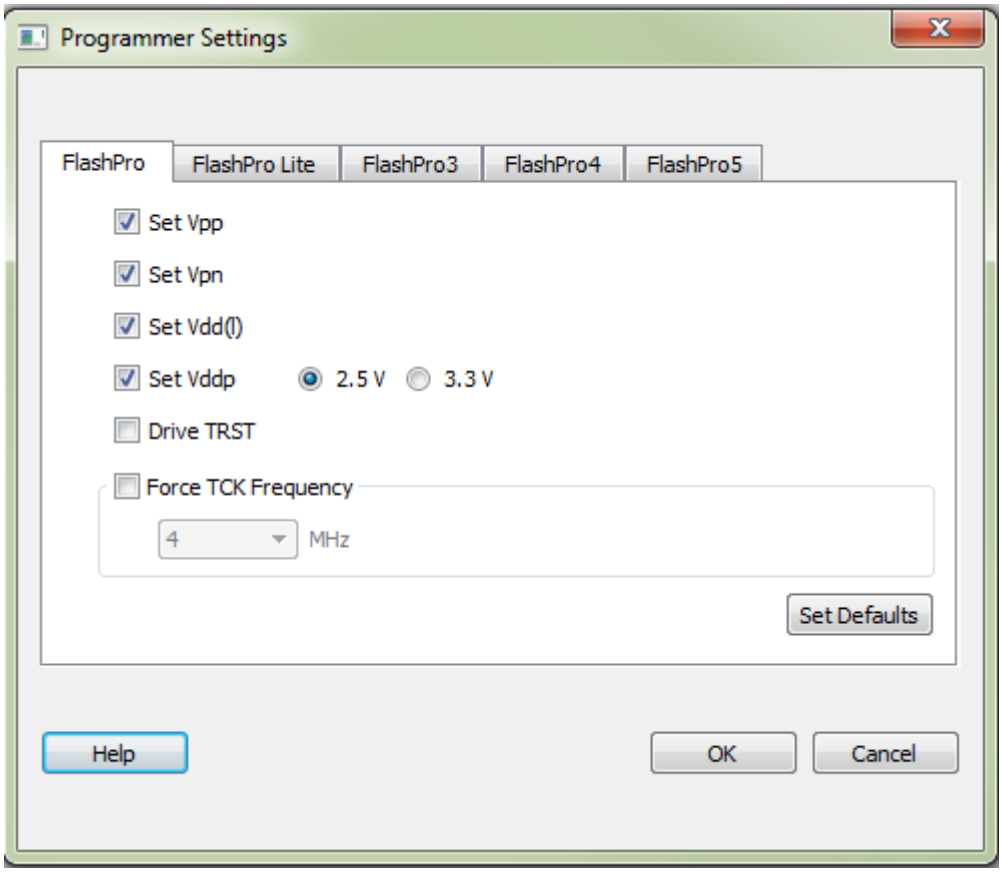


Figure 73 · Programmer Settings Dialog Box

The Programmer Settings dialog box includes setting options for FlashPro5/4/3/3X, FlashPro Lite and FlashPro.

Limitation of the TCK frequency for the selected programmer:

- FlashPro supports 1-4 MHz
- FlashPro Lite is limited to 1, 2, or 4 MHz only.
- 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:

• IGLOO, ProASIC3, Fusion, SmartFusion and SmartFusion2 – 10MHz to 20MHz

• ProASICPLUS and ProASIC – 10 MHz.

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.

FlashPro Programmer Settings

Choose your programmer settings for FlashPro (see the above figure). If you choose to add the Force TCK Frequency, select the appropriate MHz frequency. After you have made your selection(s), click **OK**.

Default Settings

- The Vpp, Vpn, Vdd(l), and Vddp options are checked (Vddp is set to 2.5V) to instruct the FlashPro programmer(s) to supply Vpp, Vpn, Vdd(l) and Vddp.
- The Driver TRST option is unchecked to instruct the FlashPro programmer(s) NOT to drive the TRST pin.
- The Force TCK Frequency option is unchecked to instruct FlashPro to use the TCK frequency specified by the Frequency statement in the STAPL file(s).

FlashPro Lite Programmer Settings

If you choose to add the Force TCK Frequency, select the appropriate MHz frequency. After you have made your selection(s), click **OK**.

Default Settings

- The Vpp and Vpn options are checked to instruct the FlashPro Lite programmer(s) to supply Vpp and Vpn.
- The Driver TRST option is unchecked to instruct the FlashPro Lite programmer(s) NOT to drive the TRST pin.
- The Force TCK Frequency option is unchecked to instruct the FlashPro Lite to use the TCK frequency specified by the Frequency statement in the STAPL file(s).

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. 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.
- 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.

Device I/O States During Programming -- JTAG Mode Only

In the Libero SoC Design Flow window expand **Edit Design Hardware Configuration** and double-click **Device I/O states during programming** to specify the I/O states prior to programming. In Libero SoC, 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

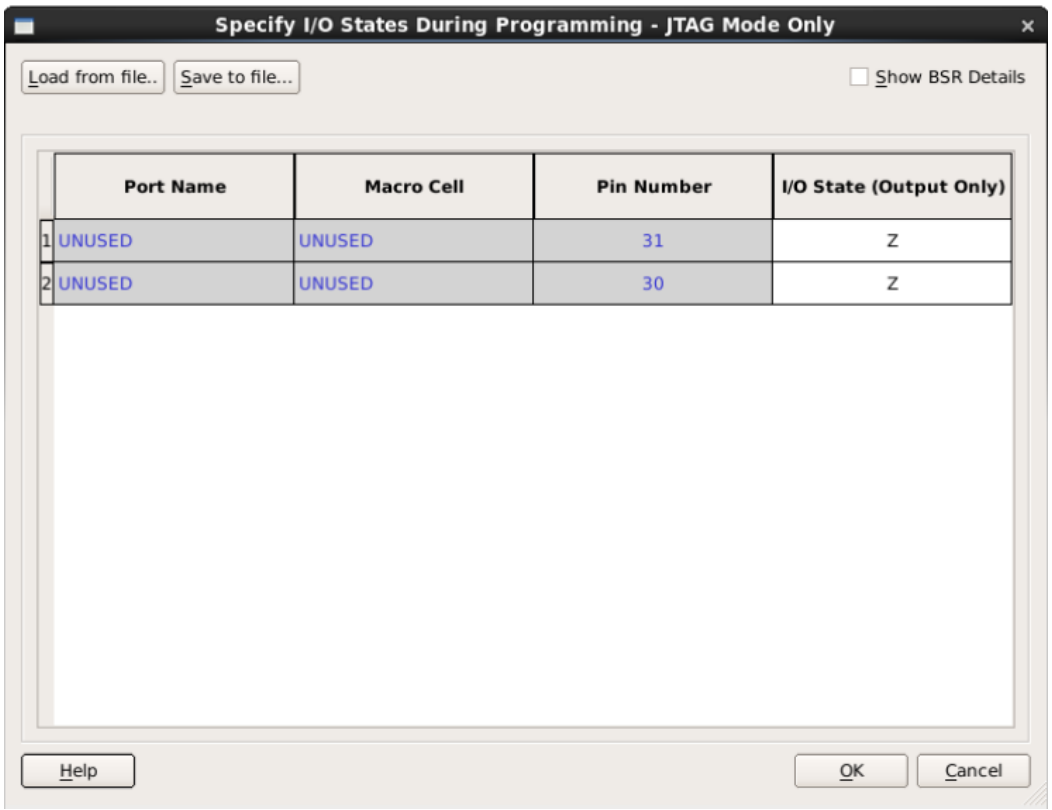


Figure 74 · I/O States During Programming Window

6. 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**.

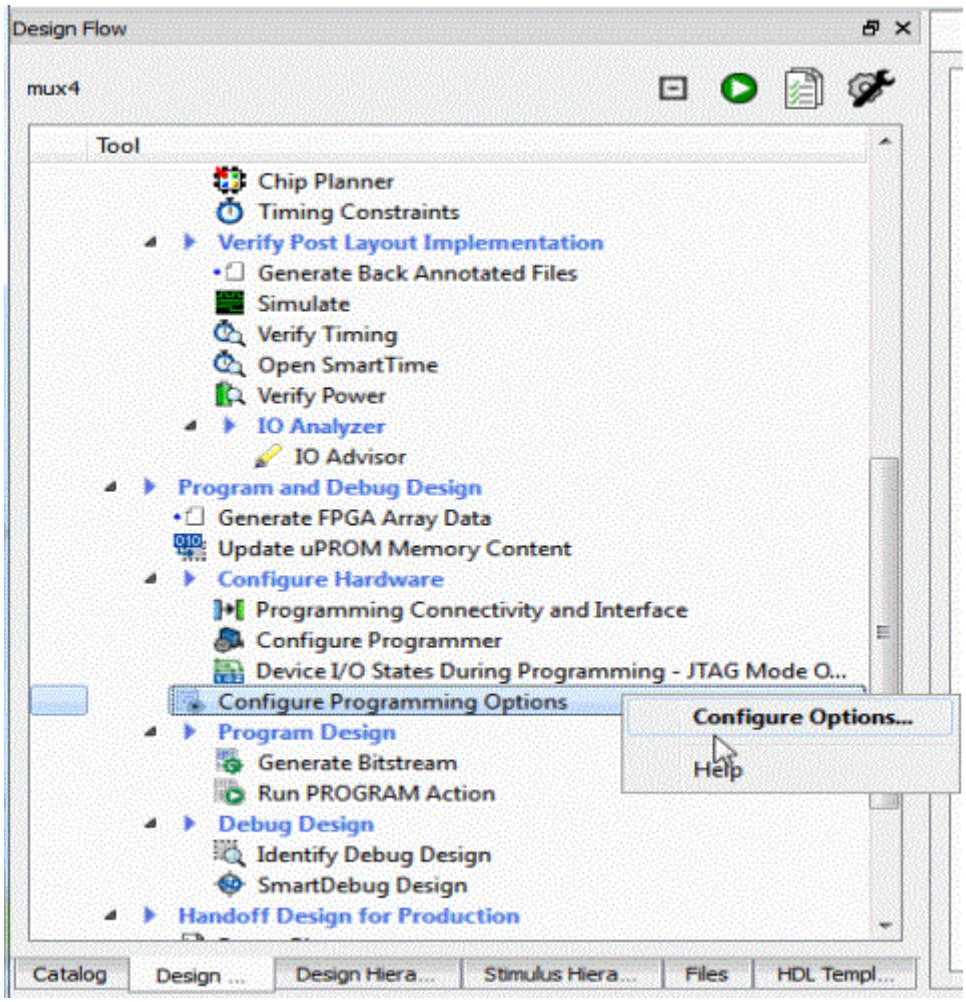


Figure 75 · Configure Programming Bitstream Settings - Configure Options

The Configure Programming Options dialog box appears for you to configure the programming options.

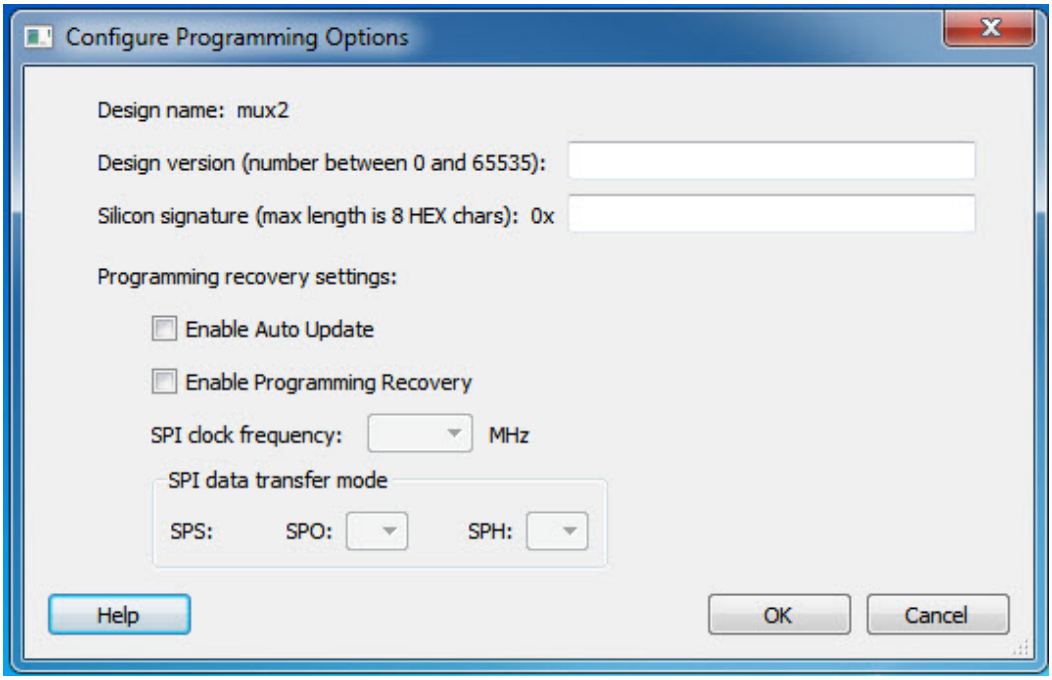


Figure 76 · Configure Programming Options Dialog Box

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 enables 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**.

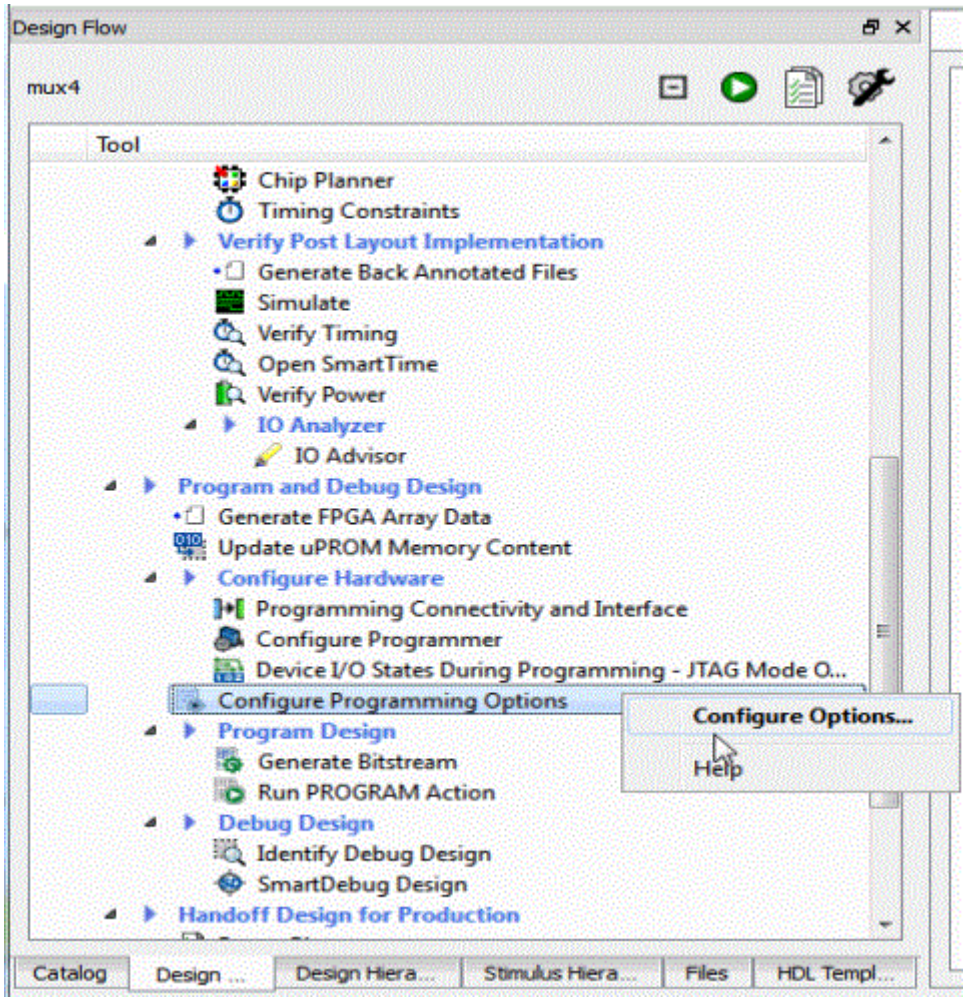


Figure 77 · Configure Programming Bitstream Settings - Configure Options

The Configure Programming Options dialog box appears for you to configure the programming options.

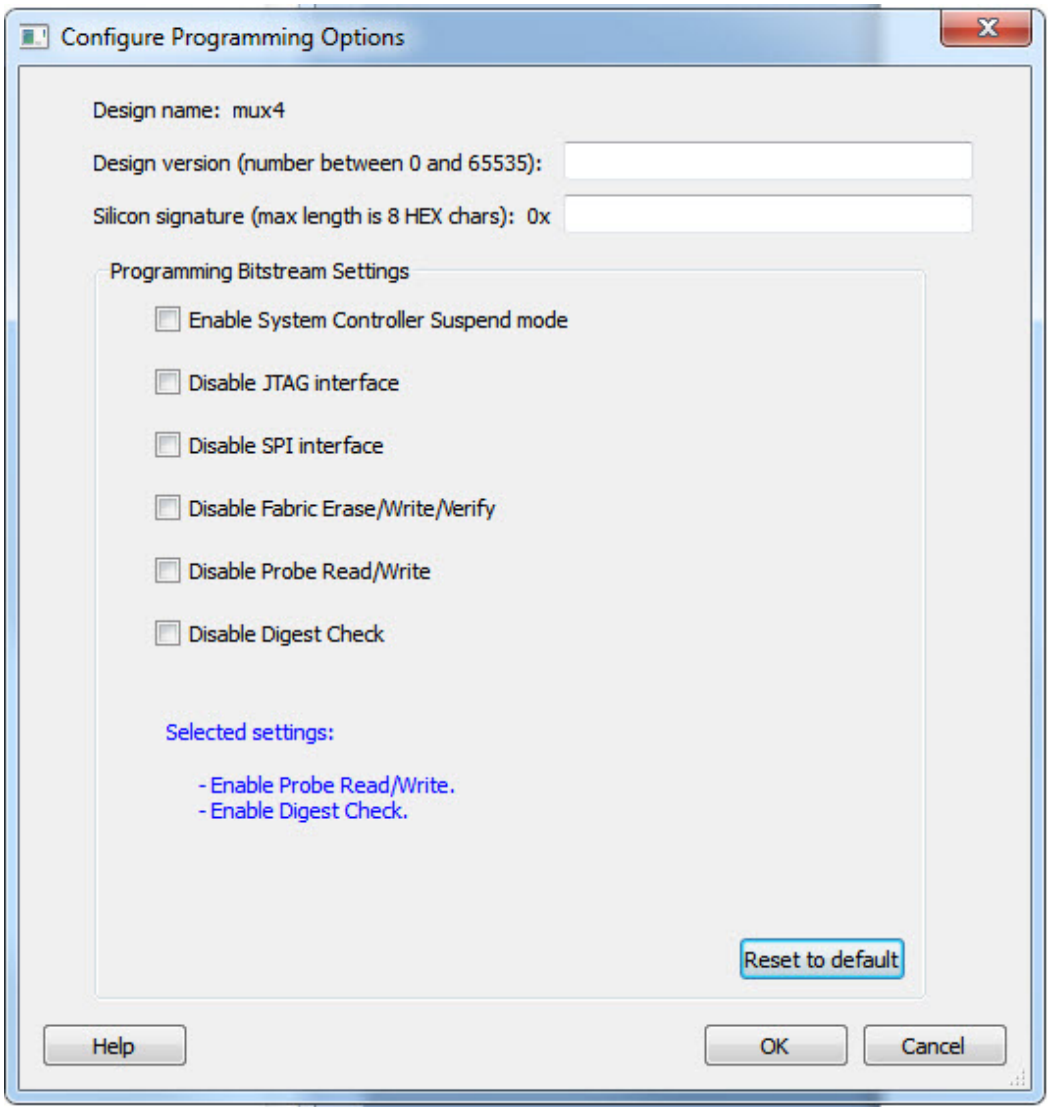


Figure 78 · Programming Bitstream Settings Dialog Box

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

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 – Check this box to disable the SPI interface when TRSTB is low during device power up. You can enable the SPI interface 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 SPI in the project settings of the Libero project (Project > Project Settings > Device Settings)

Note: If SPI and JTAG interface are 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.

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.

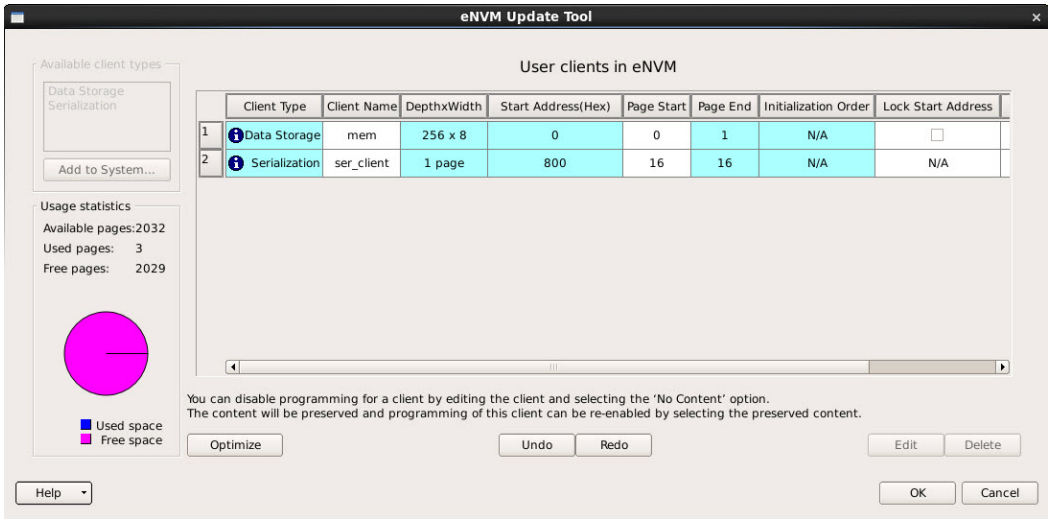


Figure 79 · eVNM Update Dialog Box

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

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.

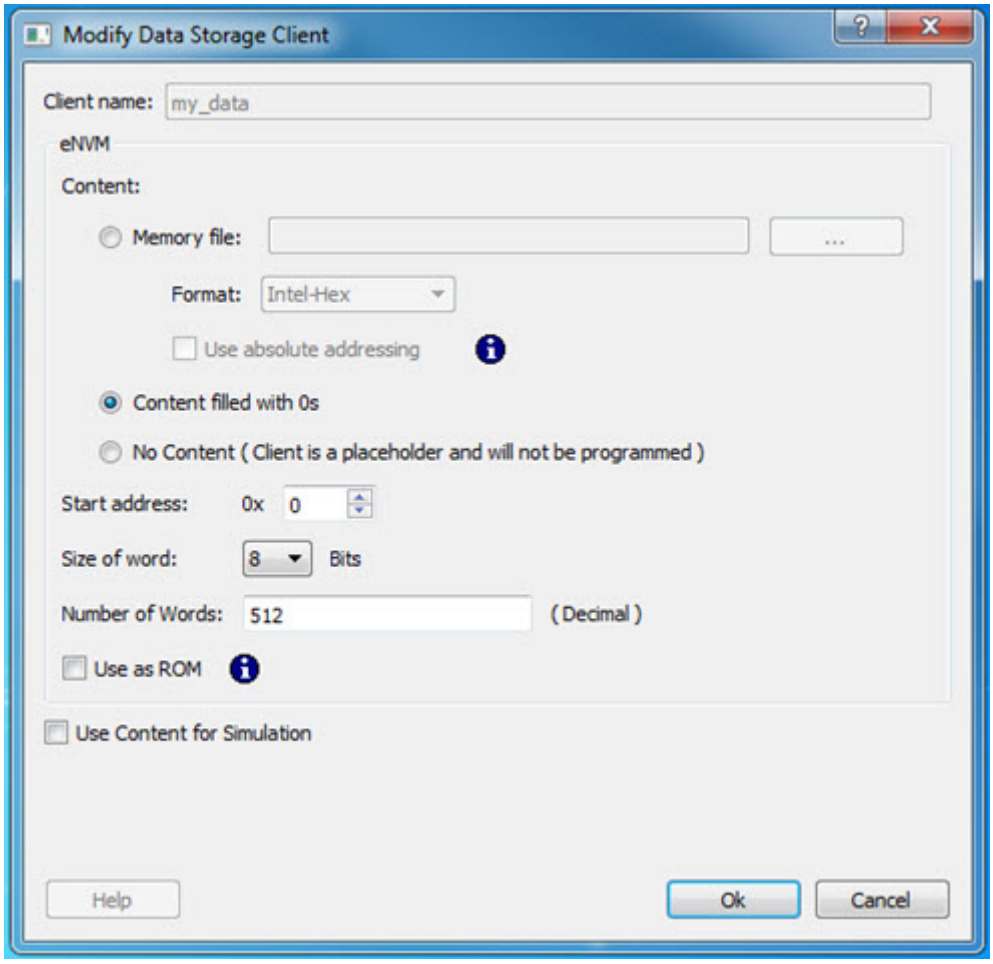


Figure 80 · Modify Data Storage Client Dialog Box

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.

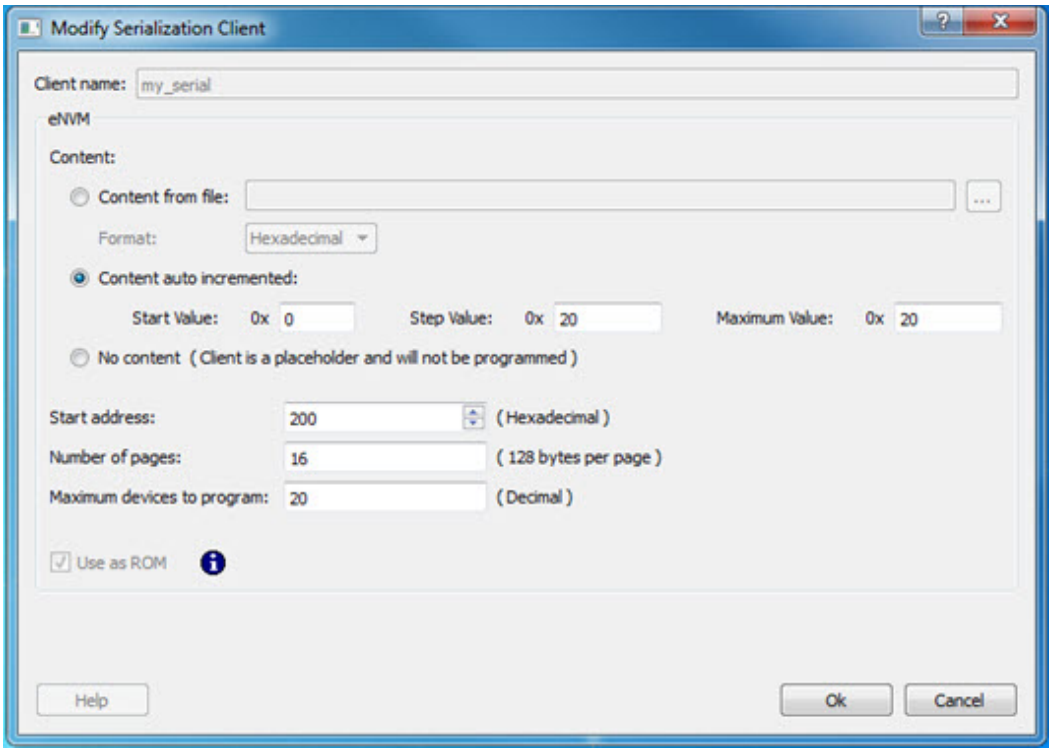


Figure 81 · Modify Serialization Client Dialog Box

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's Guide for more information.

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**.

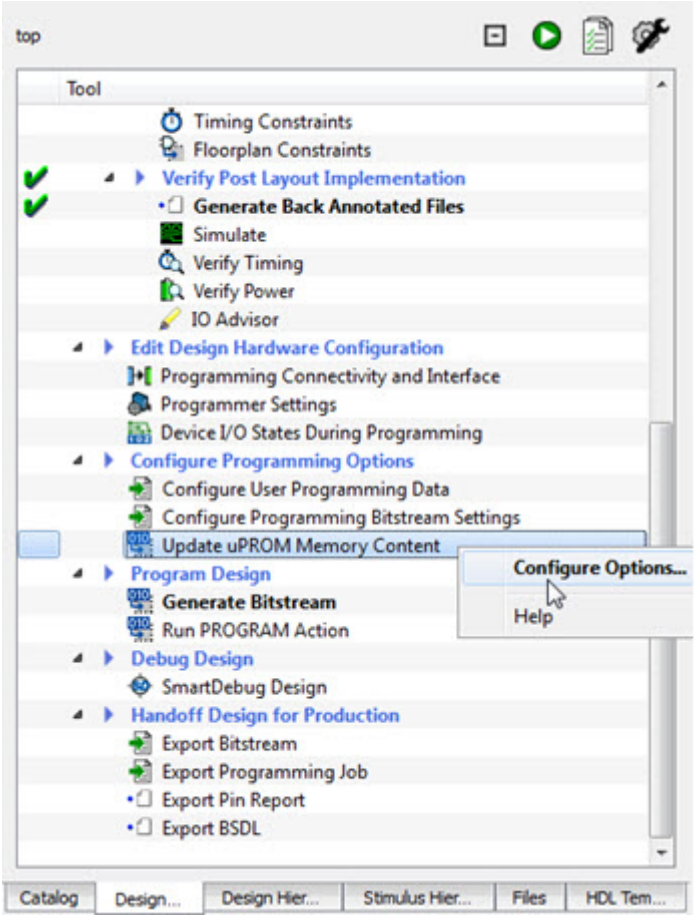


Figure 82 · Update uPROM Memory Content

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

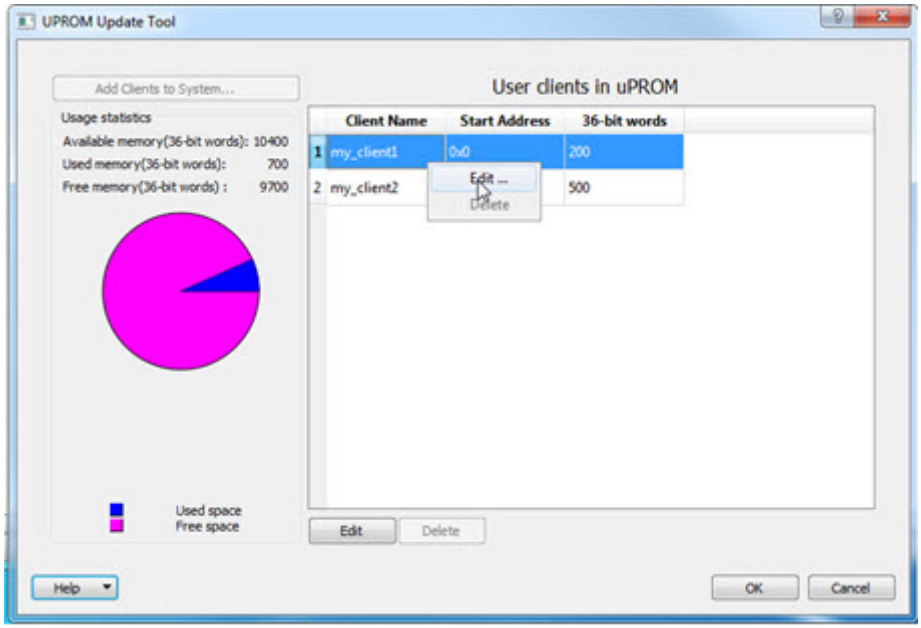


Figure 83 · uPROM Update Tool

The Edit Data Storage Client dialog box appears.

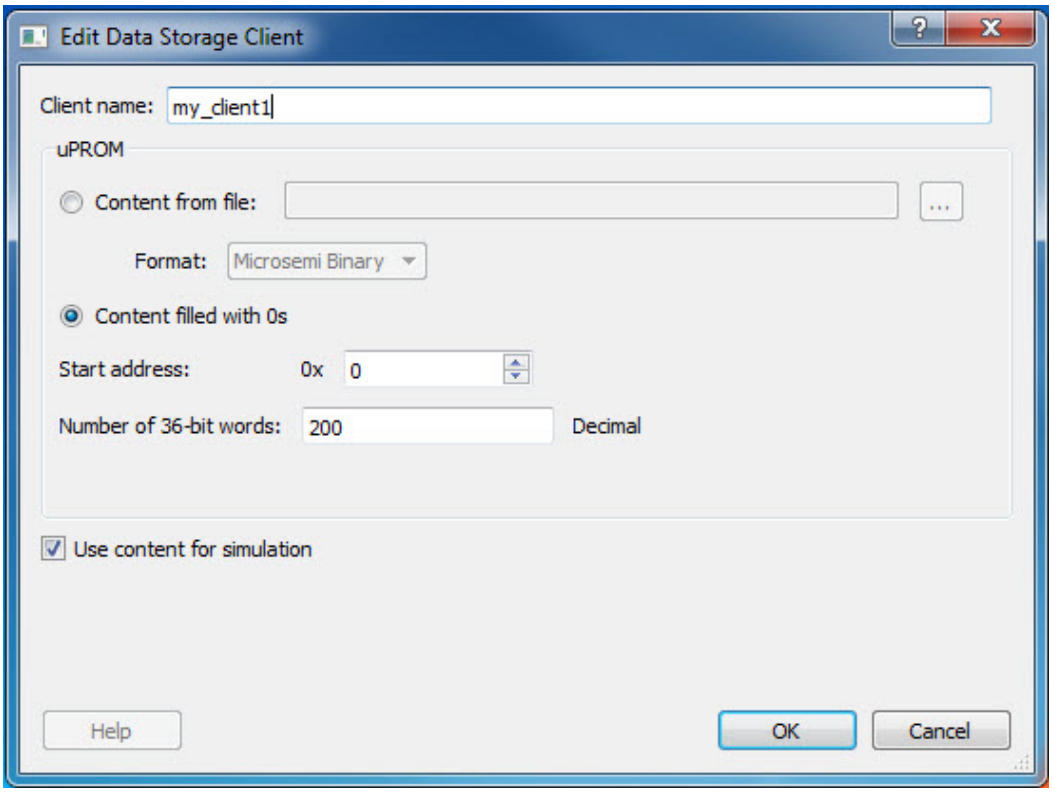


Figure 84 · Edit Data Storage Client Dialog Box

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.

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.

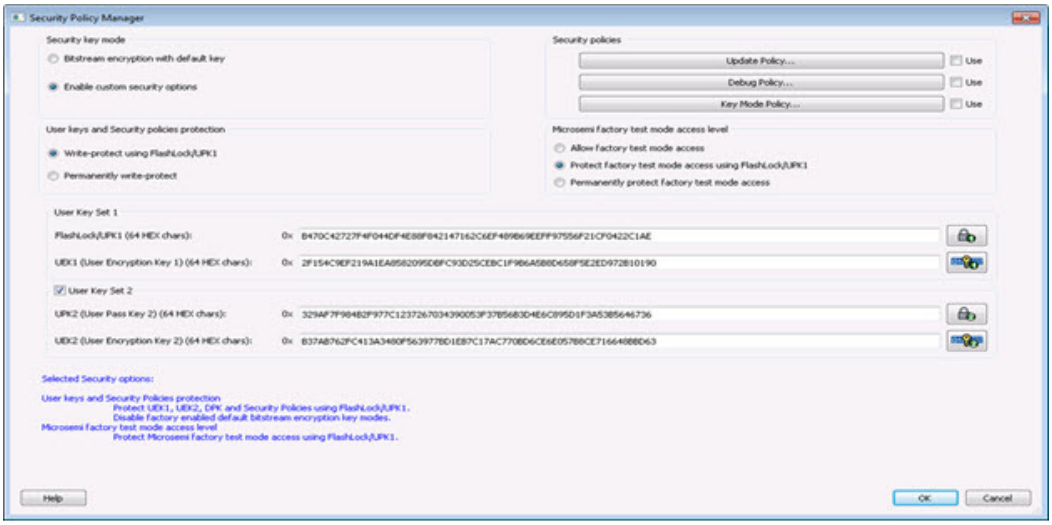


Figure 85 · Security Policy Manager Dialog Box

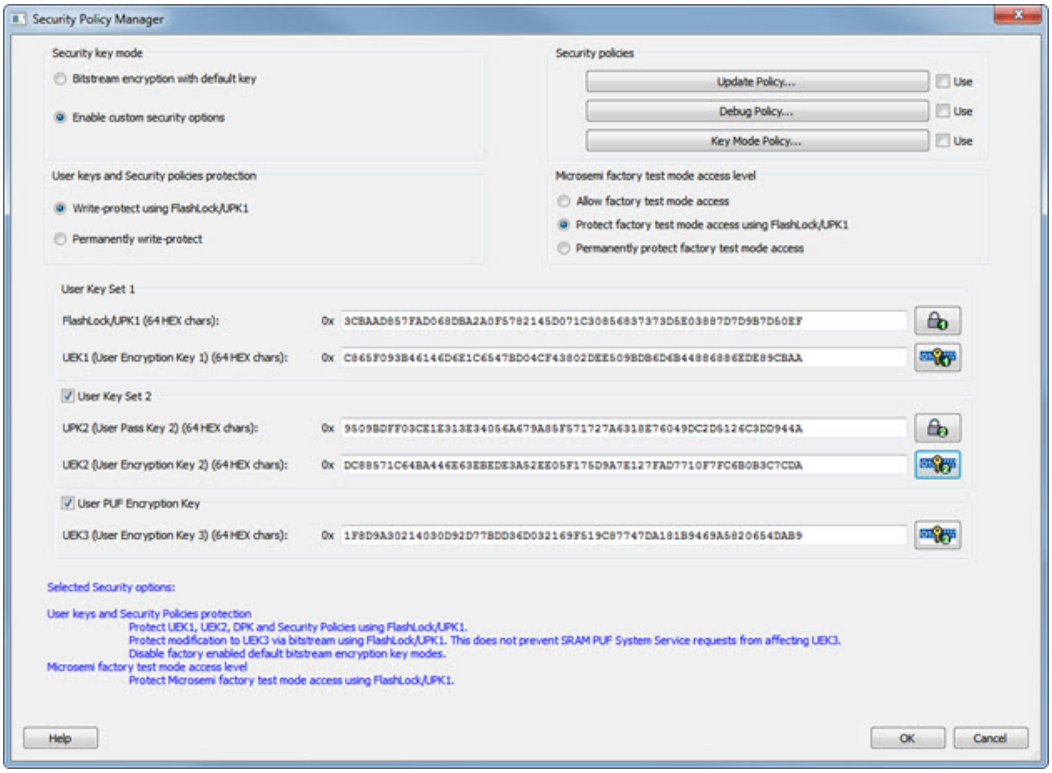


Figure 86 · Security Policy Manager Dialog Box (for devices supporting UEK3)

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

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.

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

Security Policies

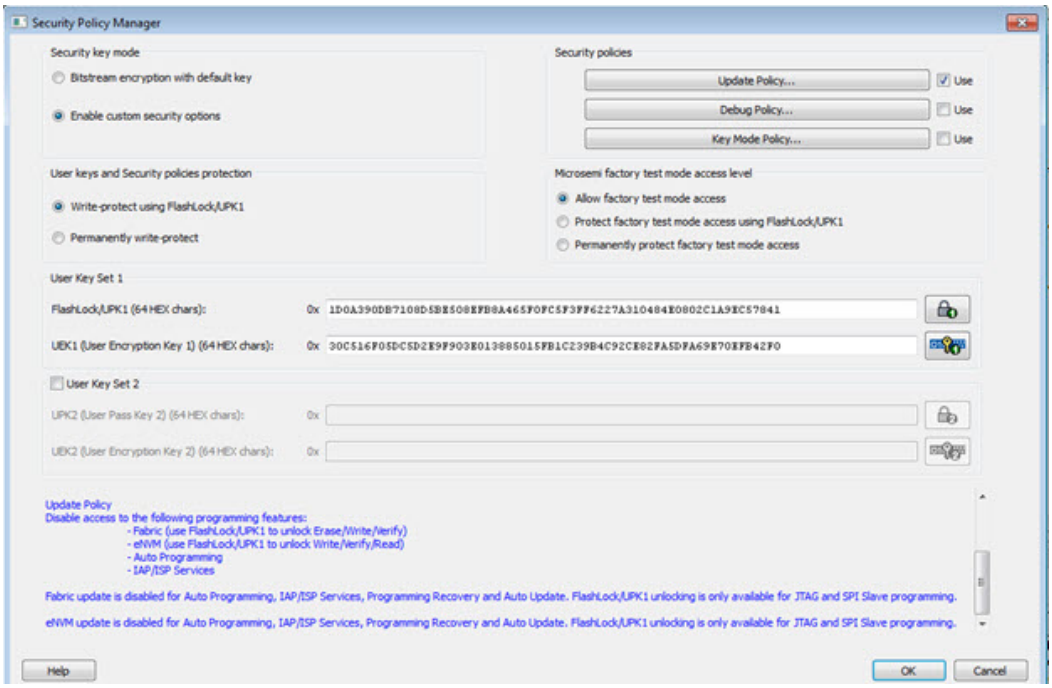
Update Policy - Sets your Fabric, eNVM and Back Level protections. See the Update Policy topic for more information.

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

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

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.

See the following example.



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 User Programming Data tool). 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.

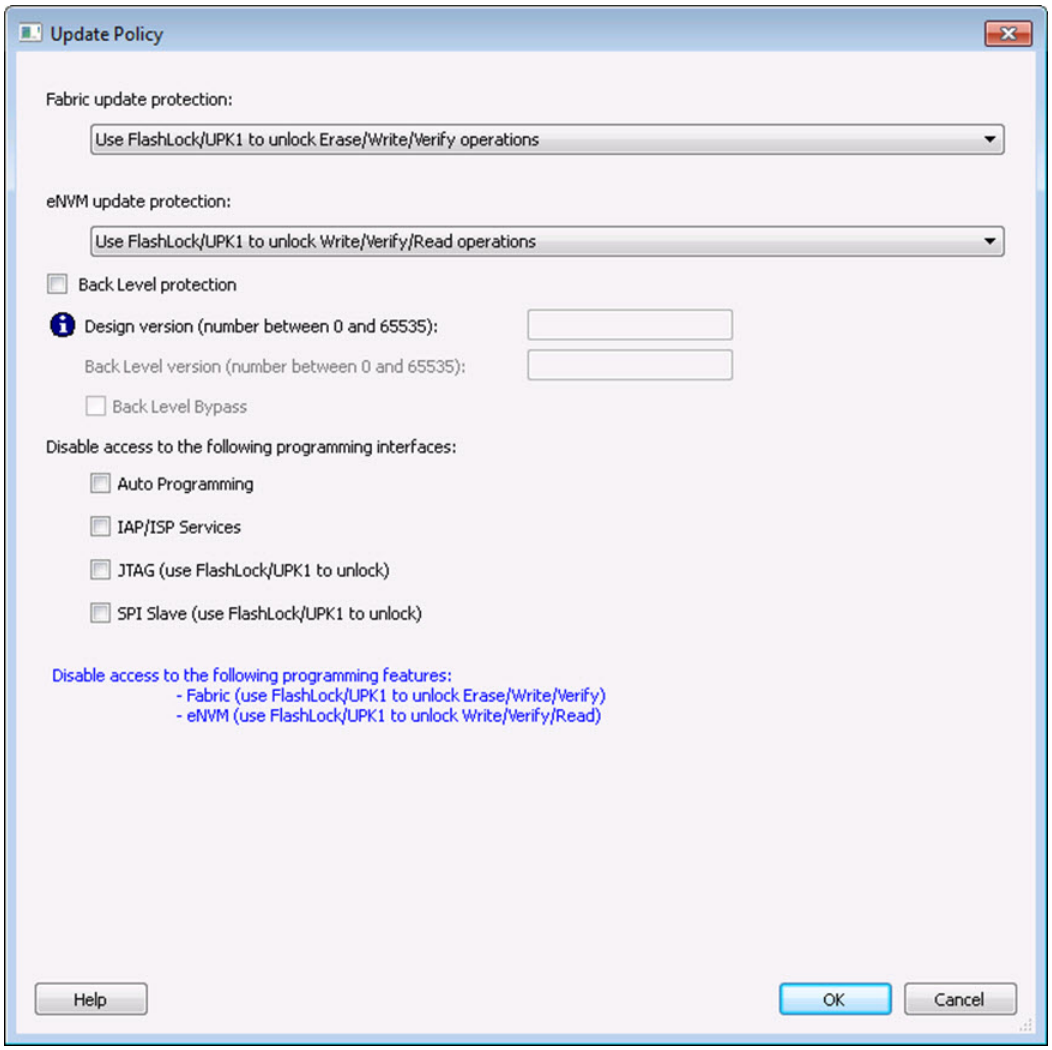


Figure 87 · Update Policy Dialog Box

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.

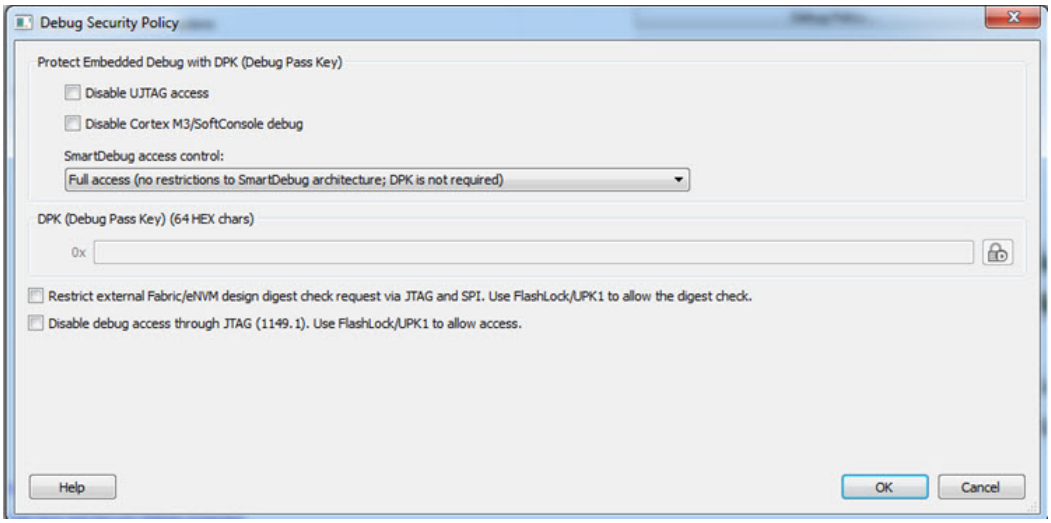


Figure 88 · Debug Security Policy Dialog Box

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.

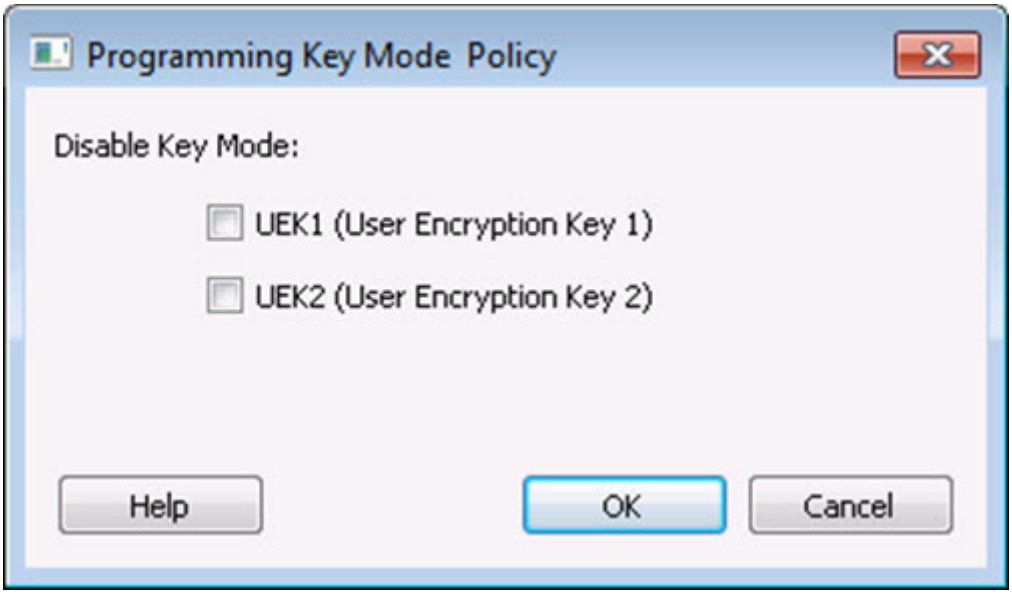


Figure 89 · Programming Key Mode Policy Dialog Box

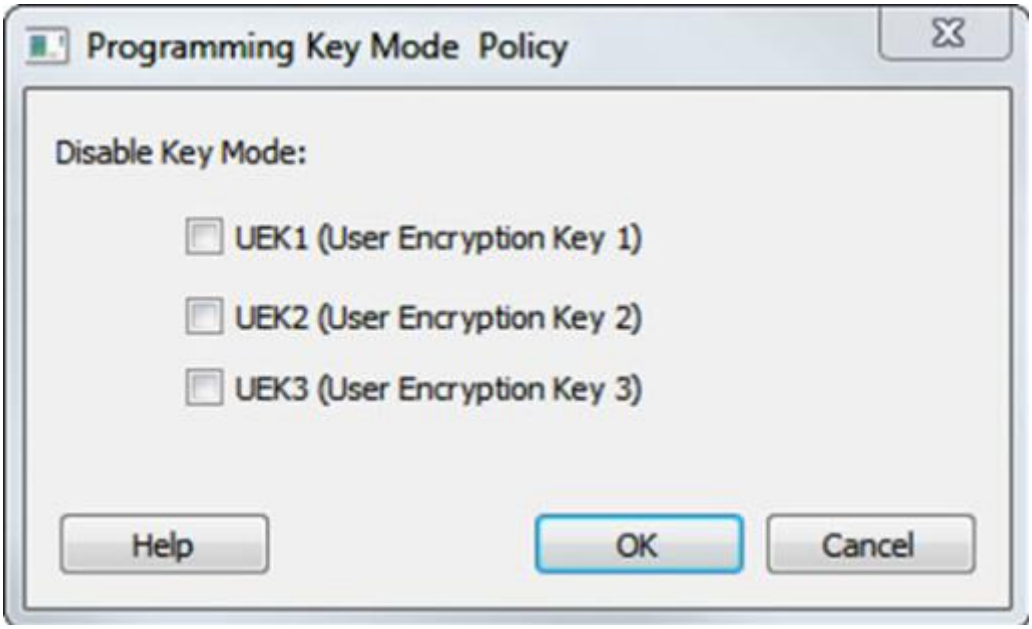


Figure 90 · Programming Key Mode Policy Dialog Box (for devices supporting UEK3)

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 Error! Bookmark not defined. **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 **Error! Bookmark not defined.** 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 M2S060, M2GL060, M2S090, M2GL090, M2S150, and M2GL150 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 **Error! Bookmark not defined.** . 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 **Error! Bookmark not defined.**. 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 M2S060, M2GL060, M2S090, M2GL090, M2S150, and M2GL150 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.

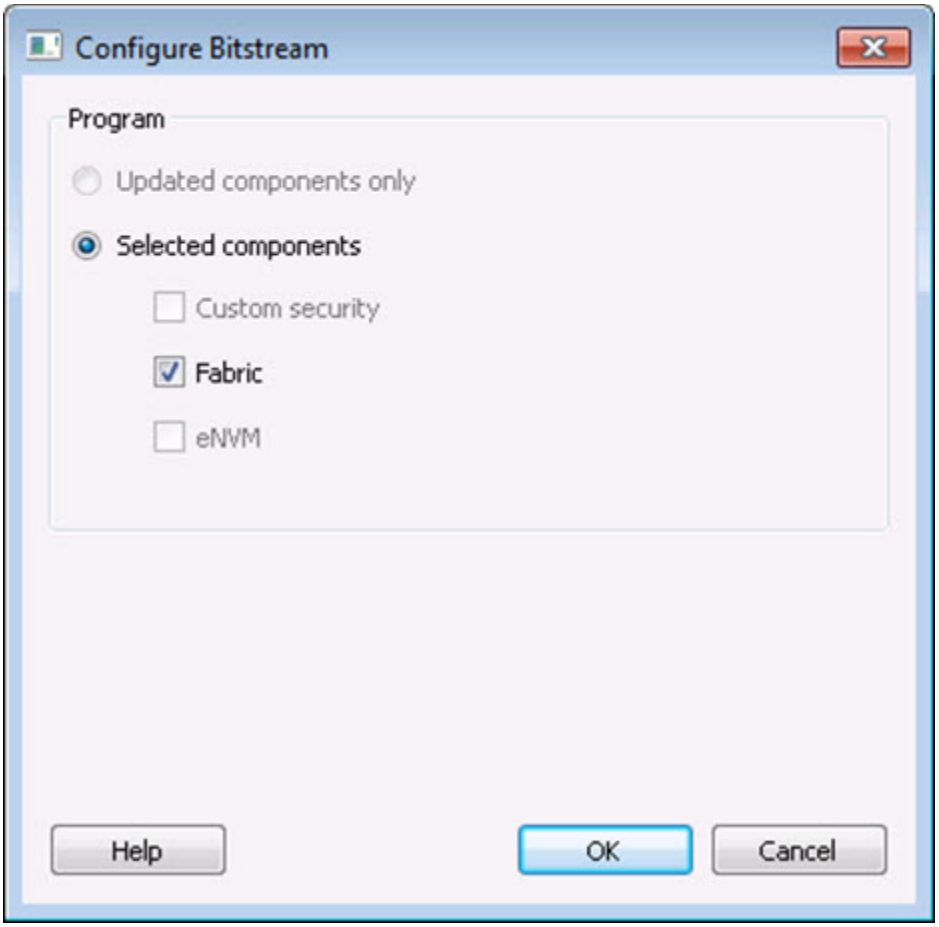


Figure 91 · Configure Bitstream Dialog Box - SmartFusion2 and IGLOO2

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 security settings (if configured) to generate the bitstream file. You need to configure the bitstream before you generate the

bitstream. 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.

This operation is completed automatically as the last step if you use the Build button.

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.

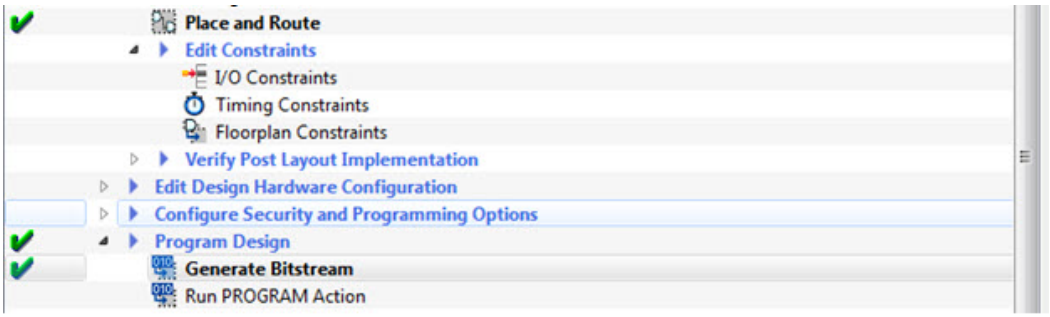


Figure 92 · Generate Bitstream (Complete)

See also

Configure Bitstream Dialog Box

Run PROGRAM Action - SmartFusion2, IGLOO2, and RTG4 Only

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).

Right-click **Run PROGRAM Action** and choose **Configure Action/Procedures** to open the Select Action and Procedures dialog box.

Program Device - SmartFusion, IGLOO, ProASIC3, and Fusion

Double-click **Program Device** in the Design Flow window to create a programming file (if necessary) and program your device with default settings. A programming file will be generated prior to programming, if it has not already been generated.

To change the programming action, configure the JTAG chain, modify the programmer settings, or make any other programming changes, you will need to configure these changes with the FlashPro tool.

1. Right-click **Program Device** and choose **Open Interactively** to open FlashPro to change the programming action, configure the JTAG chain, modify the programmer settings, or make any other programming changes.
2. Make your changes.
3. Save your changes and continue programming with Libero, or you can program directly from FlashPro.

For the SmartFusion device, Libero only exports the FDB file. Inside the FlashPro tool , you can add the eNVM or FlashROM if it has not been added or modify the eNVM or FlashROM if it has been added. For Security, you must add it in FlashPro. To open FlashPro interactively, right-click **Program Device** or **Export Programming File** and choose **Open Interactively**.

SmartFusion2 and IGLOO2 Programming - Default Settings

To view your default settings, from the **Project** menu choose **Project Settings**.

To program your SmartFusion2 or IGLOO2 device with default settings:

1. Create a Libero Soc project using any SmartDesign component. For example, you can create a project using a SmartDesign component, such as a simple fabric module and a MSS block with Flash Memory module.
2. Click the Build button to complete Synthesis, Place and Route and program the device with default settings. The default settings do not contain any security settings; use the Security Policy Manager (SPM) to manage your settings prior to programming your device.

SmartFusion2 and IGLOO2 Programming - Custom Settings

Custom Programming Settings enable you to build the JTAG chain, define programmer settings, set I/O states during programming and run scan chain.

1. To create a JTAG chain, in the Design Flow window expand **Edit Design Hardware Configuration**, right-click **Programming Connectivity and Interface** and choose **Open Interactively**. It opens a schematic view of the devices connected in a JTAG chain; all the devices are targeted by default.

   The Programming and Connectivity Interface detects and constructs the JTAG chain automatically. Use the interface to add devices manually.

   When you add Microsemi devices you can either load the STP or PDB file or add the device from a drop-down list. You must provide the IR length and Max TCK frequency OR load the BSDL file for non-Microsemi devices.
2. Right-click **Programmer Settings** and choose **Open Interactively** to view your programmer settings. If necessary, click Edit Programmer Settings to specify custom settings for your programmer.
3. Right-click **Device I/O States During Programming** and choose **Open Interactively** to open the Specify I/O States During Programming dialog box and set your device I/O states. Click OK to save your settings and continue.
4. Expand Configure Security, right-click **Security Policy Manager** and choose **Open Interactively** to specify your Secured Programming Use Model, User Key Entry and Security Policies.

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

Error Code	Exit Code	Exit Message	Possible Cause	Possible Solution
				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 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	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

Error Code	Exit Code	Exit Message	Possible Cause	Possible Solution
				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. 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.
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	>Unstable	Monitor related power supplies that

Error Code	Exit Code	Exit Message	Possible Cause	Possible Solution
		digest	voltage level Signal integrity issues on JTAG pins	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	-20	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.
0x8016	-20	eNVM_0 digest verification: FAIL	Programming bitstream components do not match components programmed eNVM_0 data has been corrupted or tampered with	Use the same programming file that was used to program the device.
0x8017	-20	eNVM_1 digest verification: FAIL	Programming bitstream components do not match components programmed eNVM_1 data has been corrupted or tampered with	Use the same programming file that was used to program the device.
0x8018	-20	User security policies segment	Programming bitstream	Use the same programming file that was used to program the device.

Error Code	Exit Code	Exit Message	Possible Cause	Possible Solution
		digest verification: FAIL	components do not match components programmed User security policy segment 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.

Error Code	Exit Code	Exit Message	Possible Cause	Possible Solution
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.
0x8021 0x8022	-23	Authentication Error Invalid/Corrupted	File contains an encrypted key that does not match the device	Provide a programming file with an encryption key that matches that on the device.

Error Code	Exit Code	Exit Message	Possible Cause	Possible Solution
		encryption key	Attempting to erase a device with no security using master security file File contains user encryption key, but device has not been programmed with the user encryption key Device has user encryption key 1/2 enforced and you are attempting to reprogram security settings	Run DEVICE_INFO action to verify that the device has no security. If the device does not have secuirty, you cannot erase it. First program security with master programming file, then program with user encryption 1/2 field update programming files. You must first ERASE security with the master security file, then you can reprogram new security settings.
0x8041	-23	Authentication Error Invalid/Corrupted encryption key	File contains an encrypted key that does not match the device File contains user encryption key, but device has not been programmed with the user encryption key Attempting to erase a device with no security using master security file Device has user encryption key 1/2 enforced and you are attempting to reprogram security settings	Provide a programming file with an encryption key that matches that on the device. Run DEVICE_INFO action to verify that the device has no security. If the device does not have secuirty, you cannot erase it. First program security with master programming file, then program with user encryption 1/2 field update programming files. You must first ERASE security with the master security file, then you can reprogram new security settings.

Error Code	Exit Code	Exit Message	Possible Cause	Possible Solution
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	Device is in System Controller Suspend Mode Check board connections	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.
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	Programming file version is out of date	Generate programming file with latest version of Libero SoC.

Error Code	Exit Code	Exit Message	Possible Cause	Possible Solution
		regenerate		
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. Plaintext pass key match is disabled. This occurs if HSM was used to program the	Provide a programming file with a pass key that matches pass key programmed into the device. Match pass key using HSM.

Error Code	Exit Code	Exit Message	Possible Cause	Possible Solution
			device.	
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.

Bitstream Encryption with Default Key in the Security Policy Manager

See the Export Bitstream topic for more information on exporting your bitstream.

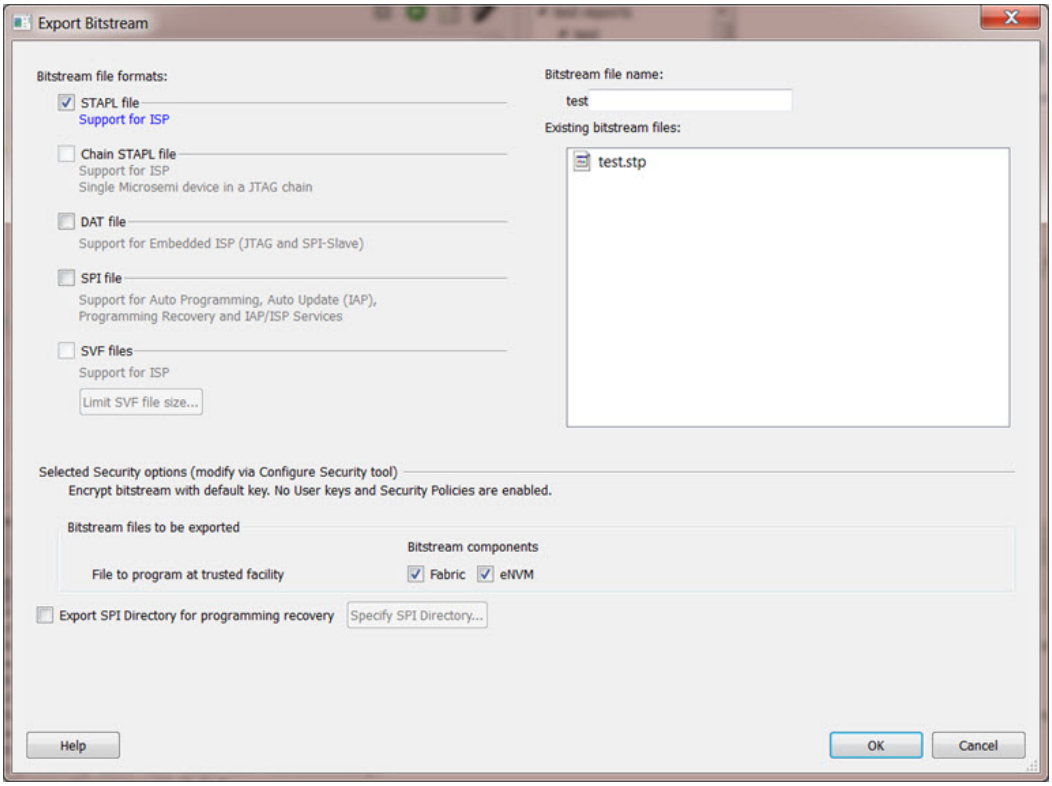


Figure 93 · Export Bitstream Dialog Box with Default Key

Bitstream file name - Sets the name of your bitstream file. The prefix varies depending on the name of your top-level design.

Existing bitstream files - Lists bitstream files you created already.

Bitstream File Formats:

Select the Bitstream File format you want to export:

- STAPL file
- Chain STAPL file (Enabled only when there are two or more devices in the chain)
- DAT file
- SPI file

Selected Security options (modify via Security Policy Manager) – Gives a brief description of current security options.

Bitstream files to be exported – Lists all the bitstream files that will 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 an 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).

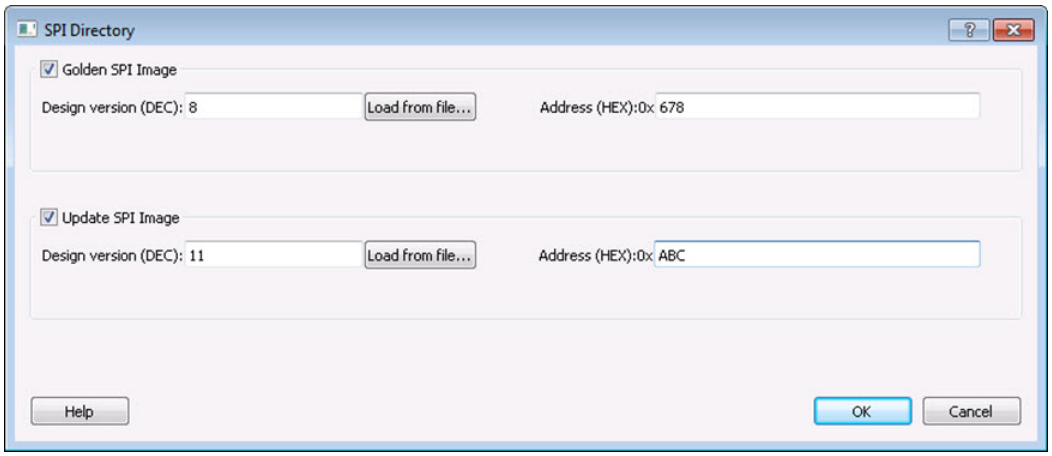


Figure 94 · SPI Directory Dialog Box

Enable Custom Security Options in the Security Policy Manager - SmartFusion2 and IGLOO2

See the Export Bitstream topic for information on exporting your bitstream.

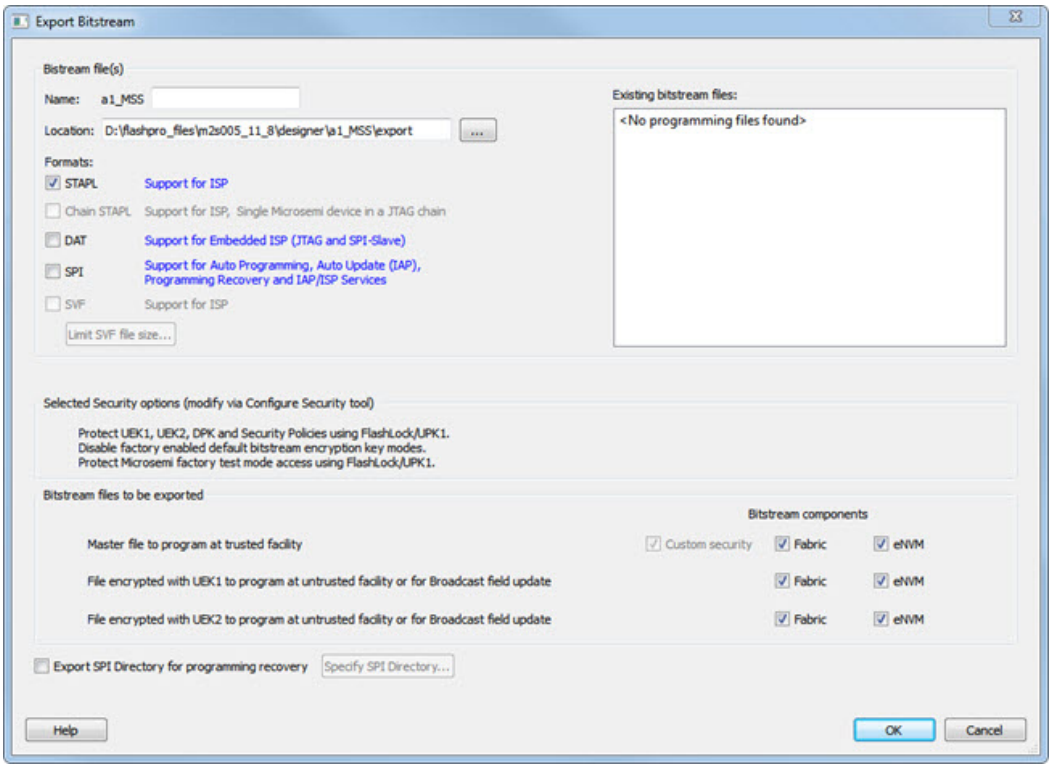


Figure 95 · Export Bitstream Dialog Box with Enable Custom Security Options in the Security Policy Manager

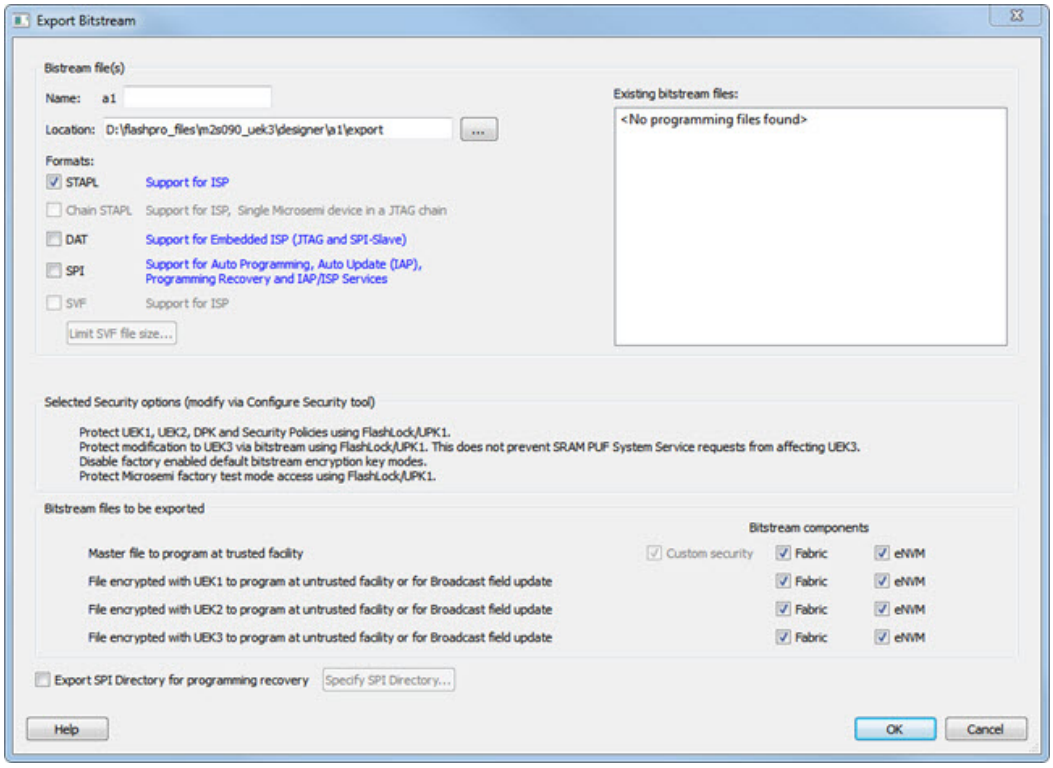


Figure 96 · Export Bitstream Dialog Box with Enable Custom Security Options in the Security Policy Manager (for devices with UEK3)

Bitstream file - Sets the name of your bitstream file. The prefix varies depending on the name of your top-level design.

Existing files: - Lists bitstream files you created already.

Formats:

Select the Bitstream file format you want to export:

- STAPL file
- Chain STAPL file (Enabled only when there are two or more devices in the chain)
- DAT file
- SPI file

Selected Security options (modify via Configure Security tool) – Gives a brief description of current security options.

Bitstream files to be exported – Lists all the bitstream files that will be exported.

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

Master file to program at trusted facility – Click to include Fabric and/or eNVM into the bitsream files to be programmed at a trusted facility. Note that Security is always programmed in Master file.

File encrypted with UEK1 to program at untrusted facility or for Broadcast field update – Click to include Fabric and/or eNVM into the bitsream files to be programmed. If the selected features are not protected by UPK1, the bitstream can be programmed at 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 into the bitsream files to be programmed. If the selected features are not

protected by UPK1, the bitstream can be programmed at 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 into the bitsream files to be programmed. If the selected features are not protected by UPK1, the bitstream can be programmed at untrusted location, since it is encrypted with UEK3 that is preprogrammed into the device.

Note: If the eNVM/Fabric is protected with UPK1 and included in the bitstream, UPK1 will be added to the STAPL and DAT file, and cannot be used at untrusted location.

Note: If eNVM/Fabric is One Time Programmable, it precluded from 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.

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).

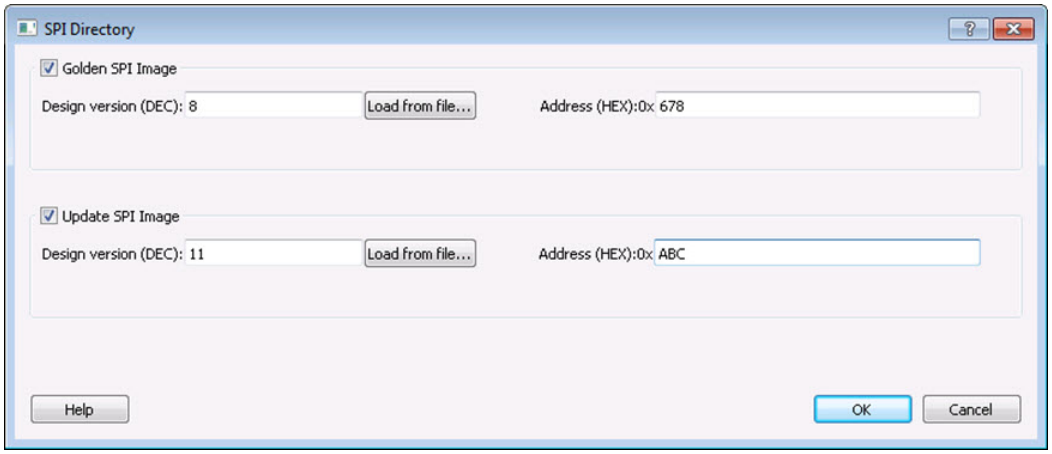


Figure 97 · SPI Directory Dialog Box

Programming SmartFusion in the Libero SoC

Double-click **Program Device** to create a programming file (if necessary) and program your device with default settings.

Right-click **Program Device** and choose **Open Interactively** to open FlashPro.

SmartFusion only exports the FDB file via Libero SoC. Inside the FlashPro tool, you can add the eNVM or FlashROM if it has not been added or modify the eNVM or FlashROM if it has been added. For Security, you must add it in FlashPro.

To open FlashPro interactively, right-click **Program Device** or **Export Programming File** and choose **Open Interactively**.

1. When FlashPro opens, click **File > Export > Export Single Programming File** to open the Export Programming Files dialog box.

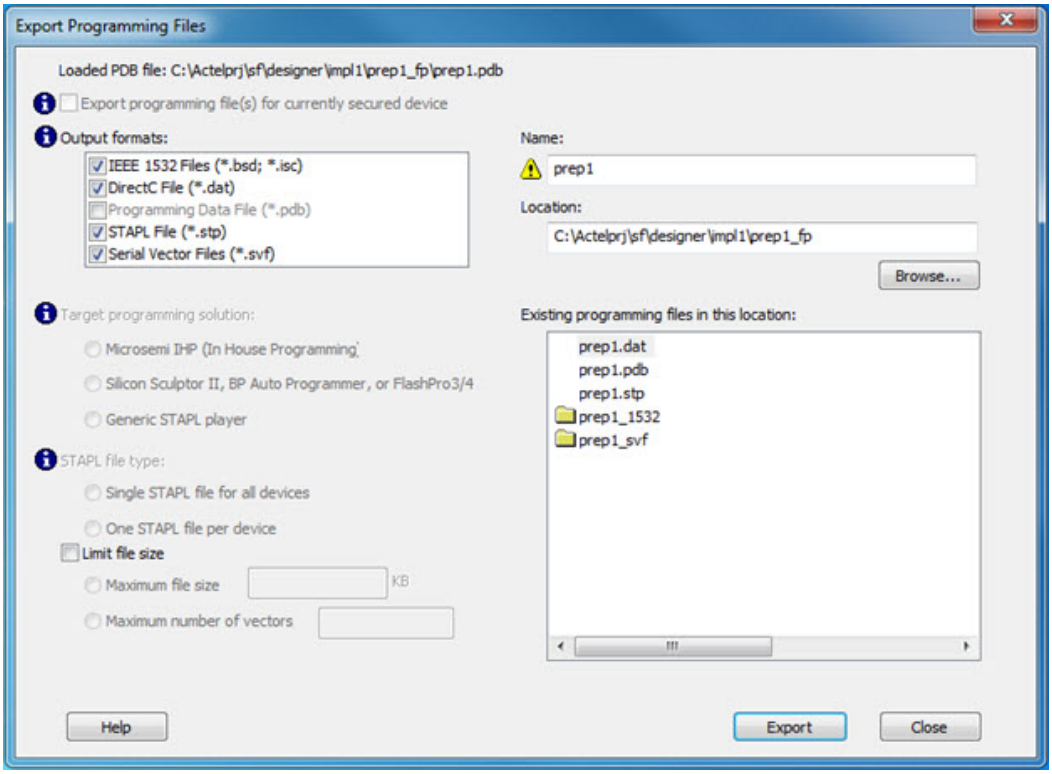


Figure 98 · Export Programming Files Dialog Box

2. Select the Programming File output format you want to generate:

- IEEE 1532 Files (*.bsd; *.isc)
- DirectC File (*.dat)
- STAPL File (*.stp)
- Serial Vector Files (*.svf)

3. Click **Export** to generate the programming files and then click **Close**.

Generate a Programming File in FlashPoint

FlashPoint enables you to program security settings, FPGA Array, and FlashROM features for SmartFusion, IGLOO, ProASIC3, Fusion family devices. You can program these features separately using different programming files or you can combine them into one programming file. Each feature is listed as a silicon feature in the GUI.

Note: You can generate a programming file with one, two, or all of the silicon features from the Programming File Generator first page.

To generate a programming file:

1. Select the **Silicon feature(s)** you want to program.

- Security settings
- FPGA Array
- FlashROM

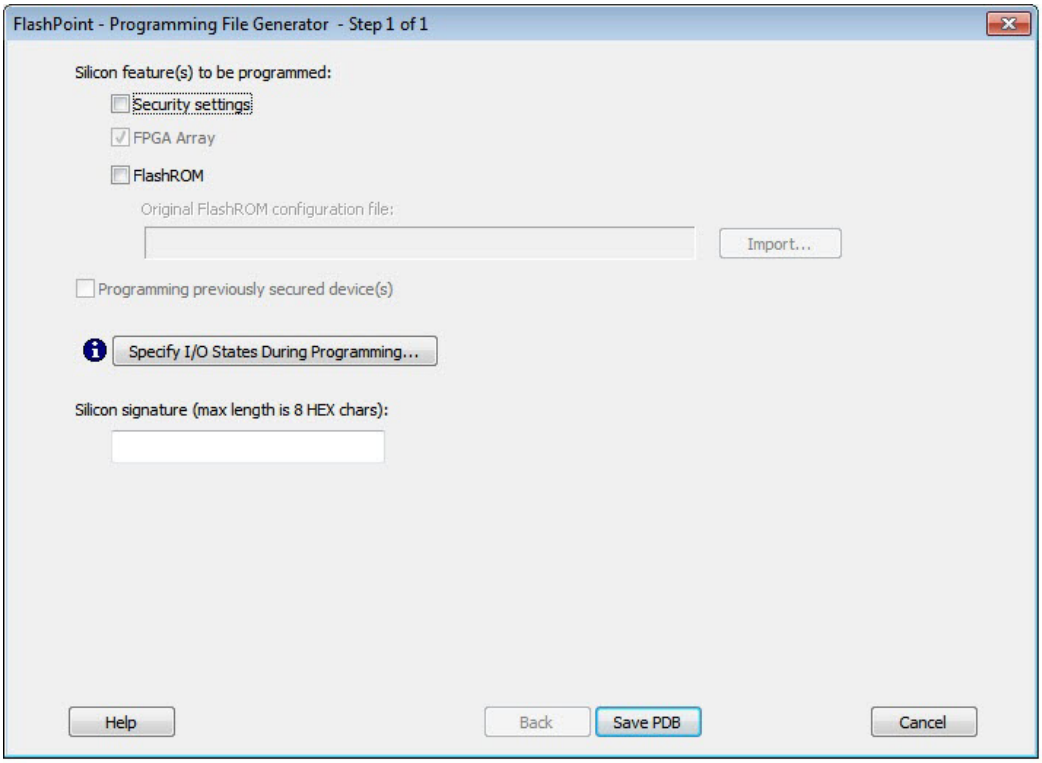


Figure 99 · Programming File Generator – Step 1 of 1

Note: When FlashPoint is invoked for the first time, after netlist files are imported and the design is in post-layout state, the software retrieves the FlashROM and EFM blocks configuration files from the imported netlists and imports the configuration files. Otherwise, you need to import configuration files.

2. Click the **Programming previously secured device(s)** check box if you are reprogramming a device that has been secured.

   Because the SmartFusion, IGLOO, ProASIC3, Fusion families enable you to program the Security Settings separately from the FPGA Array and/or FlashROM, you must indicate if the Security Settings were previously programmed into the target device. This requirement also applies when you generate programming files for reprogramming.
3. Enter the silicon signature (0-8 HEX characters). See Silicon Signature for more information.
4. Click **Save PDB**.

Programming File Types

The table below summarizes the Microsemi SoC programming file types and programmers.

Unless otherwise noted, listing an individual device indicates the device family and all its derivatives. For example, IGLOO indicates IGLOO, IGLOOe, IGLOO nano and IGLOO plus. See the Supported Families topic for more information. See the list of programming file type descriptions below for more details.

Programming File Type	Device Support	Programmer
PDB (*.pdb)	See device	FlashPro 4/3/3x

Programming File Type	Device Support	Programmer
STAPL (*.stp)	specifications	FlashPro 4/3/3x, FlashPro Lite, FlashPro, Silicon Sculptor III/II
SVF (*.svf)		Third party programmer
IEEE 1532 (*.isc or *.bsd)		Third party programmer

The following programming-related files are required if you use the related functional block elements in your enabled devices. See the appropriate sections of the FlashPro help for more information on creating these files.

File Type	Device Support	Function
FDB (*.fdb)	See device specifications	Contains your FPGA array data
UFC (*.ufc)		Contains your FlashROM data
EFC (*.efc)		Contains your Embedded Flash Memory file

PDB Files

A proprietary Microsemi programming data file.

STAPL Files

The Standard Test And Programming Language (STAPL) is designed to support the programming of programmable devices and testing of electronic systems, using the IEEE Standard 1149.1: “Standard Test Access Port and Boundary Scan Architecture” (commonly referred to as JTAG) interface. As a STAPL file is executed, signals are produced on the IEEE 1149.1 interface, as described in the STAPL file. STAPL operates on a single IEEE 1149.1 chain. STAPL supports the programming of any IEEE 1149.1-compliant programmable device.

STAPL has support for programming and test systems with user interface features. A single STAPL file may perform several different functions, such as programming, verifying, and erasing a programmable device.

Bitstream Files

Proprietary Microsemi programming data file.

SVF Files

Courtesy Serial Vector Format Specification from ASSET InterTech, 1999:

Serial Vector Format (SVF) is the media for exchanging descriptions of high-level IEEE 1149.1 bus operations. In general, IEEE 1149.1 bus operations consist of scan operations and movements between different stable states on the IEEE 1149.1 state diagram. SVF does not explicitly describe the state of the IEEE 1149.1 bus at every Test Clock.

The SFV file is defined as an ASCII file that consists of a set of SVF statements. The maximum number of characters on a line is 256, although one SVF statement can span more htan one line. Each statement consists of a command and associated parameters. Each SVF statement is terminated by a semicolon. SVF is not case sensitive.

IEEE 1532 Files

Courtesy ieee.org:

The IEEE 1532 files implement programming capabilities within programmable integrated circuit devices, utilizing (and compatible with) the 1149.1 communication protocol. This standard allows the programming of one or more compliant devices concurrently, while mounted on a board or embedded in a system, known as In-System Configuration.

Generate a Programming File for SmartFusion

You can configure and generate a new PDB file from FlashPoint.

If you are using Single Mode, click **Create** to add a new PDB, or click **Modify** to make changes to a loaded PDB.

In Chain Mode, if you have not already done so, construct a chain and click **Create PDB** to create a new PDB for programming, or click **Modify PDB** to make changes to a loaded PDB.

FlashPoint enables you to specify your security settings and silicon features when you generate your programming file in SmartFusion. You can specify your FPGA Array, FlashROM, and Embedded Flash Memory by importing FDB, UFC and EFC files, respectively (as shown in the figure below). If you have imported a FlashROM and Embedded Flash Memory file you can click **Modify** to configure these feature before saving your PDB file.

Click Specify I/O States During Programming to set custom I/O states.

Note: You must import an FDB to populate Port Name and Macro Cell columns.

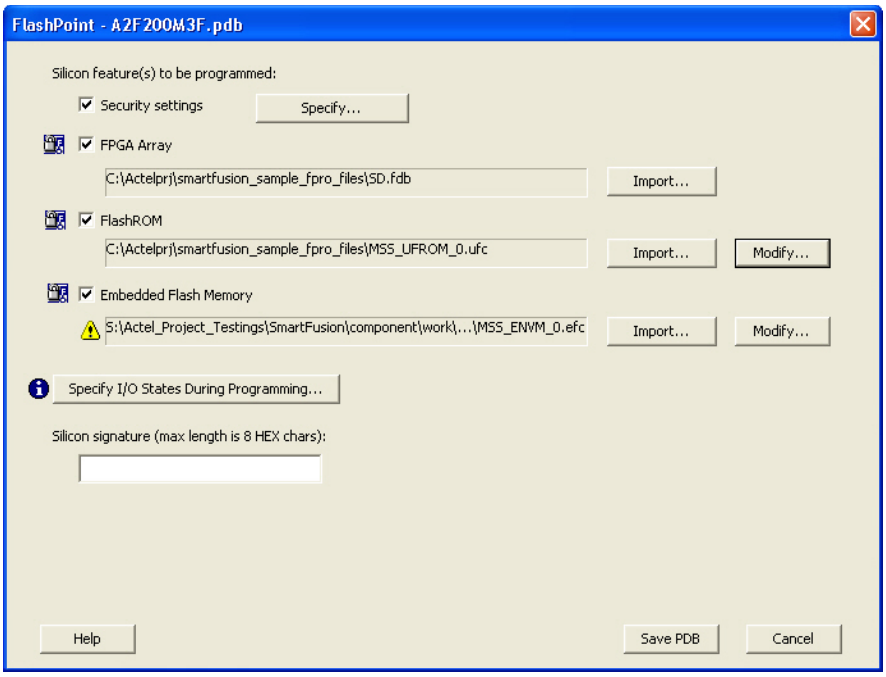


Figure 100 · FlashPoint Programming Settings for SmartFusion

Generate a Programming File for CoreMP7/Cortex-M1 Device Support

FlashPoint enables you to program FPGA Array and FlashROM features for CoreMP7/Cortex-M1 devices. You can program these features separately using different programming files or you can combine them into one programming file. Each feature is listed as a silicon feature in the GUI. You can generate a programming file with one, two, or all of the silicon features from the **Programming File Generator** first

page. For CoreMP7/Cortex-M1 device support, you cannot select your own security settings. The generated programming file always has the encrypted FPGA Array content. The programming file generation is the same as the ProASIC3 family devices.

To generate a programming file:

1. Select the Silicon feature(s) you want to program.

   FPGA Array

   FlashROM

2. Click **Next** or **Finished** depending on the silicon features you selected.

   If you click **Next**, follow the instructions in the appropriate dialog box. If you click **Finish**, the **Generate Programming Files** dialog box appears. Use this dialog box box to specify the programming file name, location, and output format (STAPL file, SVF file, PDB file, DirectC DAT file, 1532 file).

   For more information on DAT files, refer to the Data File Generator (DatGen) section of the *DirectC User's Guide*.

CoreMP7/Cortex-M1 Device Security

CoreMP7/Cortex-M1 devices are shipped with the following security enabled:

- FPGA Array enabled for AES encrypted programming and verification.
- FlashROM enabled for plain text read and write.

You cannot select your own security settings. The generated programming file includes the encrypted FPGA Array content.

Programming FlashROM and FPGA Array

For CoreMP7/Cortex-M1 device support, the programming generation for FlashROM and FPGA Array is the same as the programming generation for ProASIC3 and ProASIC family devices.

Generate a Programming File for AFS Device Support - Designer Only

FlashPoint enables you to program Security Settings, FPGA Array, Embedded Flash Memory Blocks, and FlashROM features for AFS device support. You can program these features separately using different programming files or you can combine them into one programming file. Each feature is listed as a silicon feature in the GUI. You can generate a programming file with one, two, or all of the silicon features from the **Programming File Generator** first page.

AFS Programming

In addition to FPGA Array, FlashROM and security setting, the Fusion devices provide Embedded Flash Memory

Blocks (FB) for both Analog configuration initialization and regular memory storage. Depending on the targeted AFS device, you may have one, two, or four FBs available to you. FlashPoint enables you to initialize the FB Instance(s), as desribed in the Embedded Flash Memory help.

To generate a programming file:

1. Select the **Silicon feature(s)** you want to program.

   Security Settings

   FPGA Array

   FlashROM

Embedded Flash Memory Block

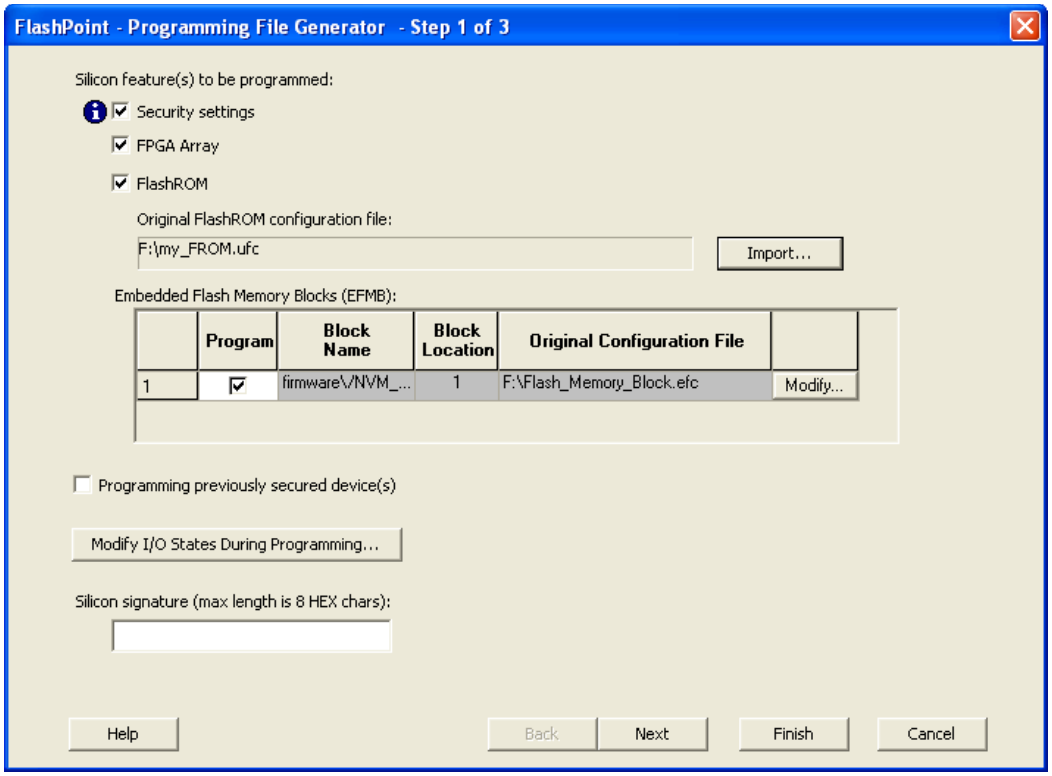


Figure 101 · FlashPoint- Programming File Generator for AFS

Note: Check the check box in the Program column to enable block modification.

2. Check the **Programming previously secured devices(s)** box if you want to program previously secured devices.
3. Enter the **Silicon signature**.
4. Depending upon the Silicon features you selected, click **Finish** or **Next**.

   If you click **Next**, follow the instructions in the appropriate dialog box. If you click **Finish**, the **Generate Programming Files** dialog box appears. Use this dialog box box to specify the programming file name, location, and output format (STAPL file, SVF file, PDB file, DirectC DAT file, 1532 file).

   For more information on DAT files, refer to the Data File Generator (DatGen) section of the *DirectC User's Guide*.

Programming Security Settings, FlashROM, and FPGA Array

For AFS device support, the programming generation for Security Settings, FlashROM and FPGA Array is the same as the programming generation for ProASIC3 family devices.

Generate a Programming File for Serialization Support in In House Programming (IHP)

FlashPoint allows you to program security settings, FPGA Array, and FlashROM features for SmartFusion, IGLOO, ProASIC3, Fusion family devices. You can program these features separately using different programming files or you can combine them into one programming file. Each feature is listed as a silicon feature in the GUI.

SVF Serialization Support in IHP

In addition to FPGA Array, FlashROM, and security setting, FlashPoint supports generating SVF files with serialization support in IHP.

To generate SVF with serialization support:

1. Select the **Silicon feature(s)** you want to program.

- Security settings
- FPGA Array
- FlashROM
- Programming Embedded Flash Memory Block

2. Import the UFC file which contains serialization data to FlashROM. Click **Next**.
3. Type in the number of devices to program (as shown in the figure below).

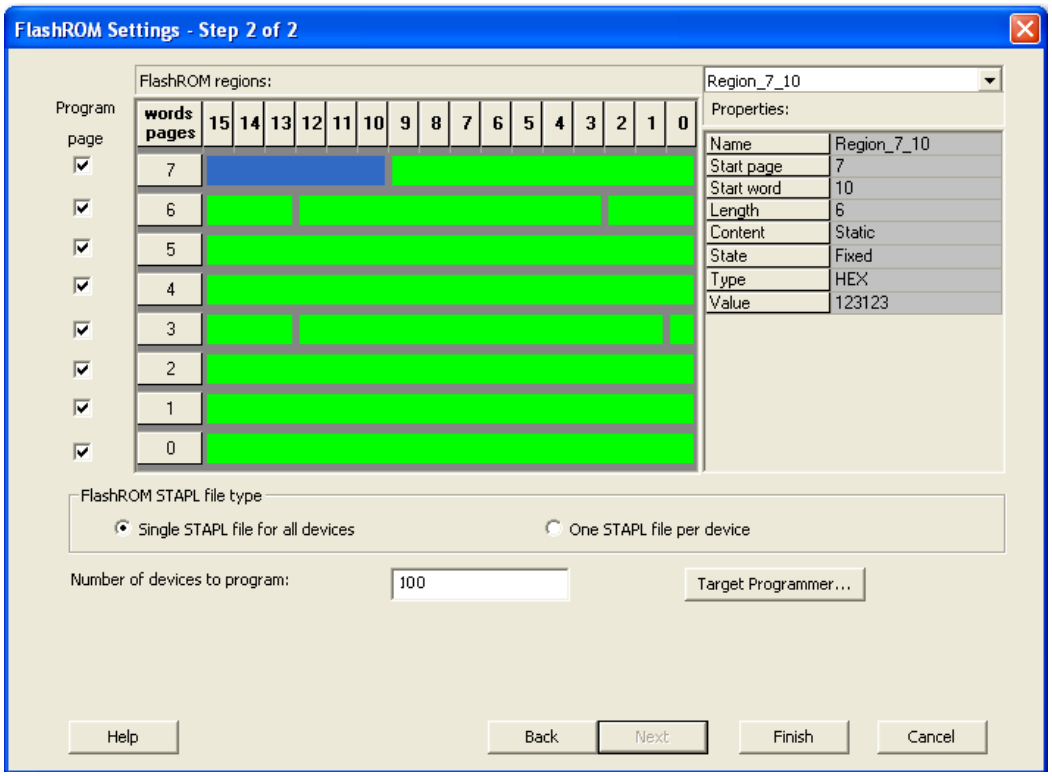


Figure 102 · Type Number of Devices

4. Click **Target Programmer** and select **Microsemi IHP**.

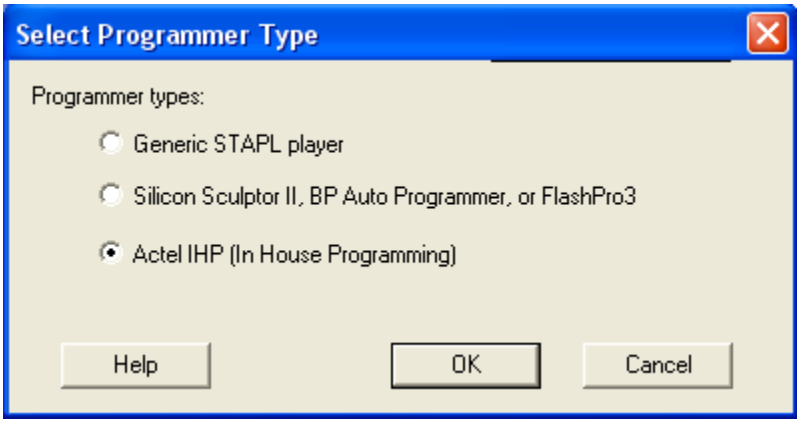


Figure 103 · Select Microsemi IHP

5. Click **OK**. The Generate Programming Files window appears (as shown in the figure below). Select **Serial Vector Files (*.svf)**.

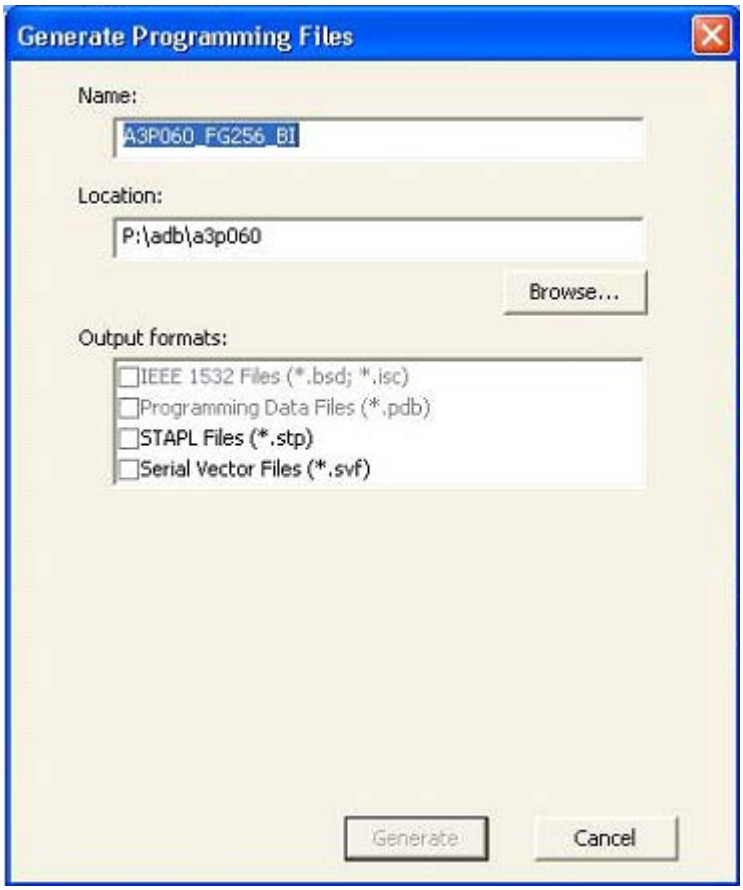


Figure 104 · Select Serial Vector Files

6. Click **Generate**. An Microsemi-specific SVF file will be generated with a corresponding serialization data file.

Note: Generated SVF files will only work with IHP.

Creating a Programming Database (PDB) File in Designer

The programming database (PDB) file supports SmartFusion, IGLOO, ProASIC3 and Fusion devices only. This allows reconfiguration of the security settings, FlashROM, FPGA Array, and Embedded Flash Memory Blocks. You create the file in Designer using FlashPoint and you modify the file in FlashPro.

You must create programming files for SmartFusion in FlashPro; see the Generate a Programming File for SmartFusion topic for more information.

1. From the Designer main window, click the **Programming File** button. This brings up FlashPoint (see figure below).

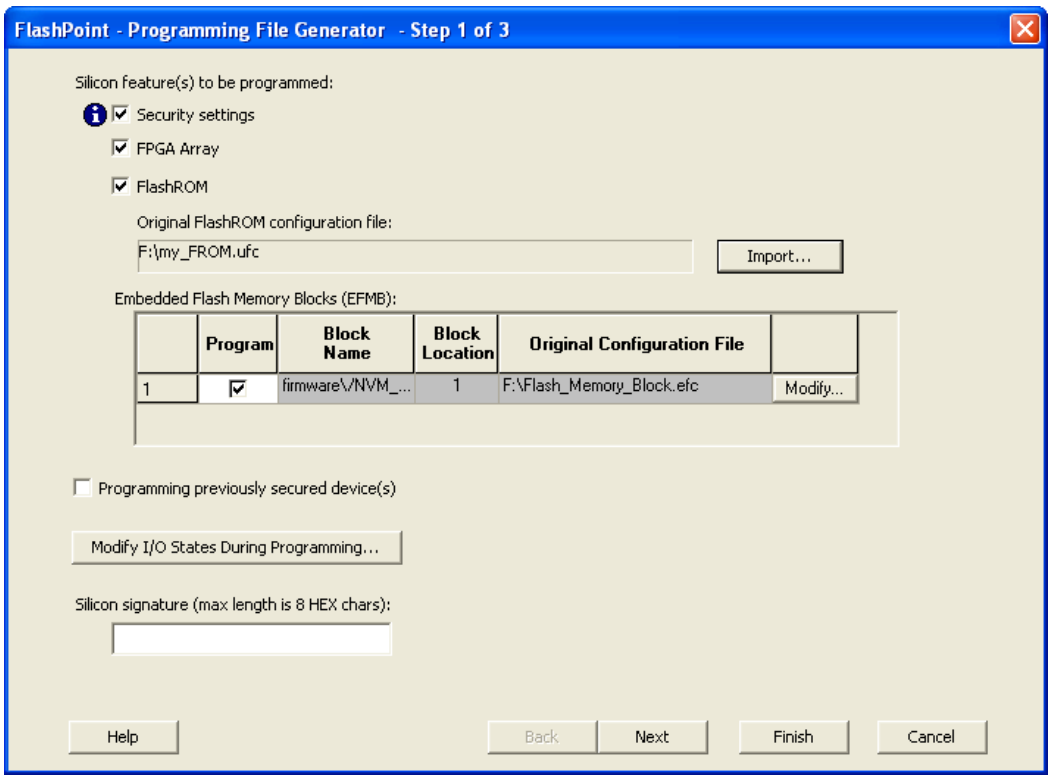


Figure 105 · FlashPoint Programming File Generator - PDB File

2. Select the silicon feature(s) to be programmed: Security Settings, FPGA array, FlashROM, and Embedded Flash Memory Block. If you are programming a previously secured device, check the Programming previously secured device(s) and enter the silicon signature.
3. Click **Finish** to create the PDB file.

See Also

Configuring security and FlashROM settings in FlashPro

Configuring security settings in FlashPro

Configuring FPGA array settings

Configuring FlashROM settings in FlashPro

Configuring Embedded Flash Memory Block settings in FlashPro

Programming Embedded Flash Memory Block

For more information about the Embedded Flash Memory Block, see the Flash Memory System Builder online help.

To program the Embedded Flash Memory Block:

1. Check the **Program** box to enable Embedded Flash Memory Block modification.
2. Click the **Modify** button to import Embedded Flash Memory Block configuration and memory content.

   The **Modify Embedded Flash Memory Block** dialog box appears.

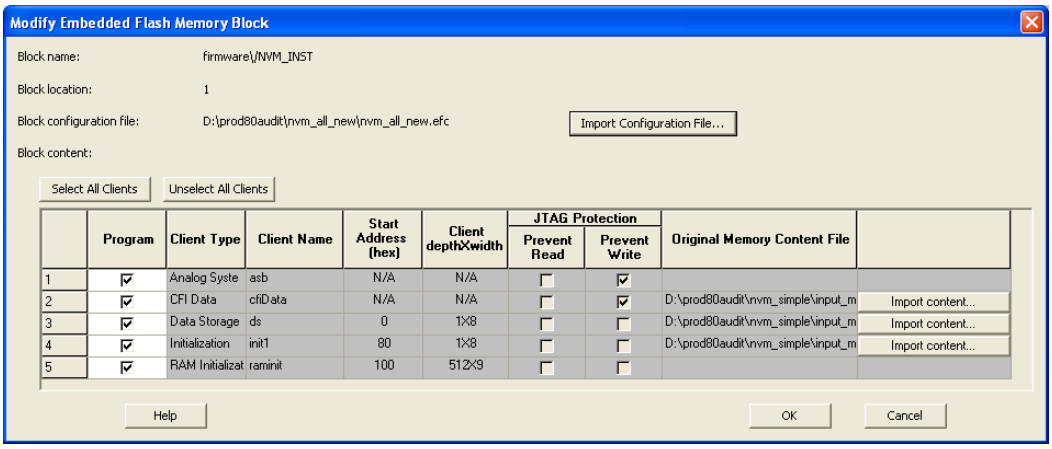


Figure 106 · Modify Embedded Flash Memory Block Content Dialog Box

3. Click the **Import Configuration File** button (if available) to import the Embedded Flash Memory Block configuration and memory content from the EFC file. This will populate the client table below. All clients that belong to this block will be selected by default.
4. Click the **Import content** button if you want to change the client memory content.
5. Click **OK**.

Note: FlashPoint audits original configuration and memory content files and warns you if the files cannot be located or if they have been updated.

Programming the FlashROM

You can program selected memory pages and specify the region values of the FlashROM.

- **Single STAPL file for all devices**: generates one programming file with all the generated increment values or with values in the custom serialization file.
- **One STAPL file per device**: generates one programming file for each generated increment value or for each value in the custom serialization file.

1. Select your target **Programmer type**.
   - Select Generic STAPL Player when generating STAPL files for generic STAPL players.
   - Select Silicon Sculptor II, BP Auto Programmer, or FlashPro5/4/3x/3 when generating programming files for those programmers.
   - Select Microsemi IHP (In House Programming) when generating STAPL or SVF files for Microsemi SoC (formerly Actel) IHP.
2. Click **OK**.
   FlashPoint generates your programming file.

Note: You cannot change the FlashROM region configuration from FlashPoint. You can only change the configuration from the FlashROM core generator.

For more information, click the Help button in FlashROM.

To program FlashROM:

1. Select **FlashROM** from the **Generate Programming File** page.
2. Enter the location of the FlashROM configuration file. The **FlashROM Settings** page appears (see figure below).

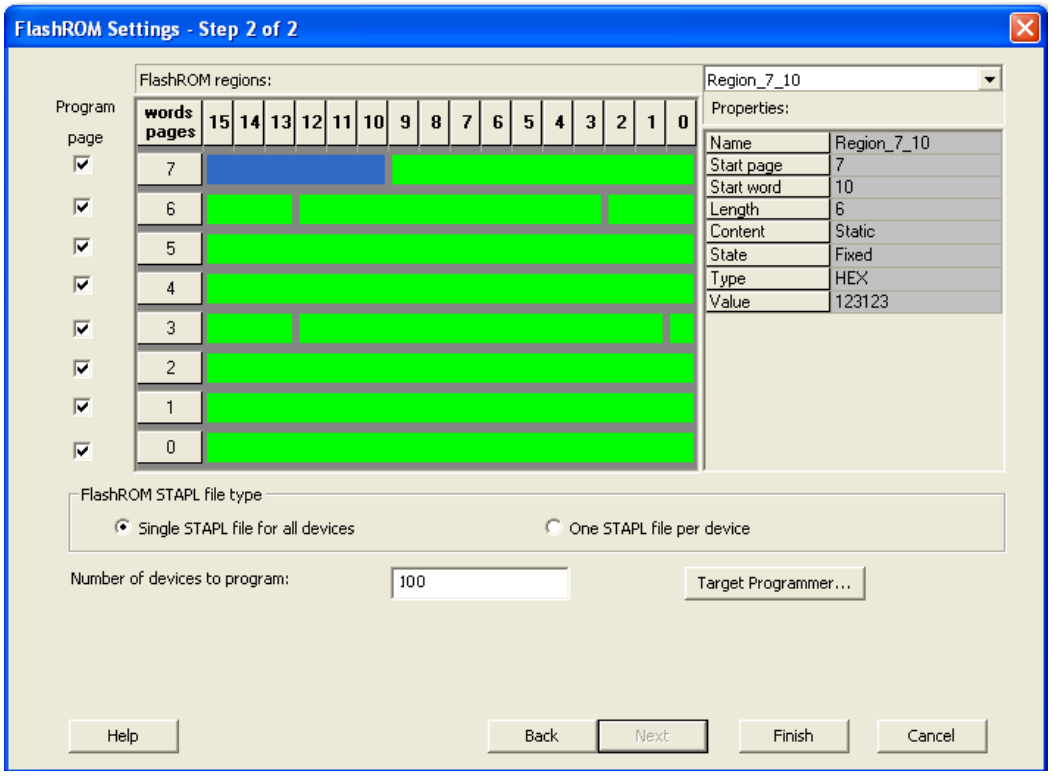


Figure 107 · FlashROM Settings

3. Select the FlashROM memory page that you want to program.
4. Enter the data value for the configured regions.
5. If you selected the region with a **Read From File**, specify the file location.
6. If you selected the **Auto Increment** region, specify the **Start** and **Max** values.

   Enter the number of devices you want to program.

   Select your target Programmer Type.

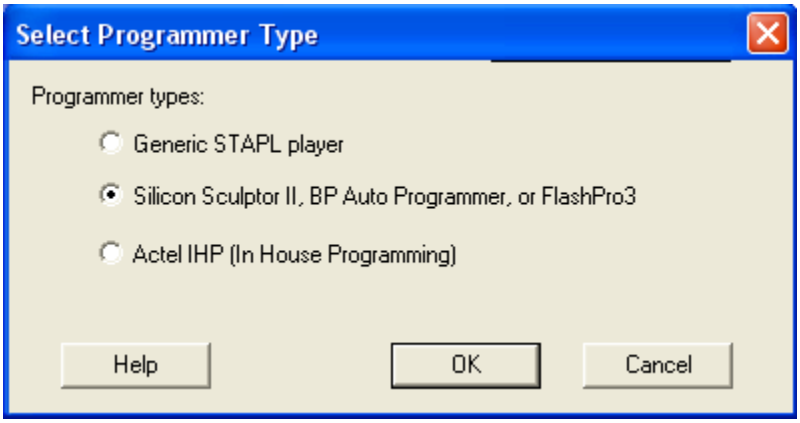


Figure 108 · Select Programmer

7. Click **Finish**.
   FlashPoint generates your programming file.

Note: You cannot change the FlashROM region configuration from FlashPoint. You can only change the configuration from the FlashROM core generator.

Silicon Signature

With Libero SoC tools, you can use the silicon signature to identify and track Microsemi designs and devices. When you generate a programming file, you can specify a unique silicon signature to program into the device. This signature is stored in the design database and in the programming file, and programmed into the device during programming.

The silicon signature is accessible through the USERCODE JTAG instruction.

Note: If you set the security level to high, medium, or custom, you must program the silicon signature along with the Security Setting. If you have already programmed the Security Setting into the target device, you cannot reprogram the silicon signature without reprogramming the Security Setting.

Note: The previously programmed silicon signature will be erased if:

- You have already programmed the silicon signature and
- You are programming the security settings, but you do not have an entry in the silicon signature field

Programming Security Settings

FlashPoint allows you to set a security level of high, medium, or none (SmartFusion uses radio buttons and the option Clear Security instead of None).

To program Security Settings on the device:

1. If you choose to program Security Settings on the device from the **Generate Programming File** page, the wizard takes you to the **Security Settings** page.

   Your Security Settings page depends on your family.
2. Set the security level for FPGA, FlashROM, and EFMB (see the table below for a description of the security levels).

Table 2 · FPGA, FlashROM, and EFMB Security Settings

Security Level	Security Option	Description
High	Protect with a 128-bit Advanced Encryption Standard (AES) key and a Pass Key	Access to the device is protected by an AES Key and the Pass Key. The Write and Verify operations of the FPGA Array use a 128-bit AES encrypted bitstream. From the JTAG interface, the Write and Verify operations of the FlashROM use a 128-bit AES encrypted bitstream. Read back of the FlashROM content via the JTAG interface is protected by the Pass Key. Read back of the FlashROM content is allowed from the FPGA Array. The Read and Verify operations of the EFMB module are protected by Pass Key from the JTAG interface. The Write operations of the EFMB module use a 128-bit AES encrypted bitstream.
Medium	Protect with Pass Key	The Write and Verify operations of the FPGA Array require a Pass Key. From the JTAG interface, the Read and Write operations on the FlashROM content require a Pass Key. You can Verify the FlashROM content via the JTAG interface without a Pass Key. Read back of the FlashROM content is allowed from the FPGA Array. The Read, Write, and Verify operations of the EFMB module are protected by Pass Key from the JTAG interface.
None	No security	The Write and Verify operations of the FPGA Array do not require keys. The Read, Write, and Verify operations of the FlashROM content also do not require keys. The Read, Write, and Verify operations of the EFMB module content do not require keys. This option is available for SmartFusion; to choose it, de-select the Security Settings checkbox.

Note: When a Device is programmed with a Pass key and AES key, only the Pass key is required for reprogramming since re-entering the correct Pass key unlocks the bits that restrict programming to require AES encryption and also unlocks the bits that prohibit reprogramming altogether (if locked); thus both plaintext and encrypted programming are [re-] enabled.

3. **Enable eNVM client JTAG protection -** Enables eNVM client JTAG protection in the event you have not set Medium or High security. Enables you to protect specific clients with a user pass key and then leave others unprotected. This can be advantageous if you want to protect your IP, but give another user access to the rest of the eNVM for storage. You can also set custom security levels for your eNVM.
   NOTE: EFMB (Fusion) is called eNVM for SmartFusion devices.

4. Enter the **Pass Key** and/ or the **AES Key** as appropriate. You can generate a random key by clicking the **Generate random key** button.

   The **Pass Key** protects all the Security Settings for the FPGA Array, FlashROM, and/or EFMB.
   The **AES Key** decrypts the FPGA Array, FlashROM, and/or EFMB programming file content. Use the AES Key if you intend to program the device at an unsecured site or if you plan to update the design at a remote site in the future.

You can also customize the security levels by clicking the **Custom Level** button. For more information, see the Custom Security Levels section.

To change or disable your security keys you must run the ERASE_SECURITY action code. This erases your security settings and enables you to generate the programming file with new keys and reprogram, or to generate a programming file that has no security key.

Custom Security Levels

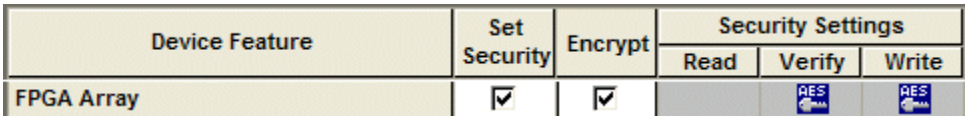
For advanced use, you can customize your security levels.

To set custom security levels:

1. Click the **Custom Level** button in the **Security Settings** page. The **Custom Security Level** dialog box appears.
2. Select the **FPGA Array Security** and the **FlashROM Security** levels. For SmartFusion and Fusion devices, you can also choose the Embedded Flash Memory Block level of security. The FPGA Array and the FlashROM can have different Security Settings. See the tables below for a description of the custom security option levels for FPGA Array and FlashROM.

Table 3 · FPGA Array

Security Option	Description
Lock for both writing and verifying Device Feature: FPGA Array; Set Security: ☑; Encrypt: ☐; Security Settings: Read: ; Verify: 🔒; Write: 🔒	Allows writing/erasing and verification of the FPGA Array via the JTAG interface only with a valid Pass Key.
Lock for writing Device Feature: FPGA Array; Set Security: ☑; Encrypt: ☐; Security Settings: Read: ; Verify: 🔒; Write: 🔒	Allows the writing/erasing of the FPGA Array only with a valid Pass Key. Verification is allowed without a valid Pass Key.
Use the AES Key for both writing and verifying	Allows the writing/erasing and

Security Option	Description
 Device Feature | Set Security | Encrypt | Security Settings: Read | Verify | Write FPGA Array | ☑ | ☑ | | AES | AES	verification of the FPGA Array only with a valid AES Key via the JTAG interface. This configures the device to accept an encrypted bitstream for reprogramming and verification of the FPGA Array. Use this option if you intend to complete final programming at an unsecured site or if you plan to update the design at a remote site in the future. Accessing the device security settings requires a valid Pass Key.
Allow write and verify Device Feature | Set Security | Encrypt | Security Settings: Read | Verify | Write FPGA Array | ☐ | ☐ | | |	Allows writing/erasing and verification of the FPGA Array with plain text bitstream and without requiring a Pass Key or an AES Key. Use this option when you develop

Security Option	Description
	your product in-house.

Note: The ProASIC3 family FPGA Array is always read protected regardless of the Pass Key or the AES Key protection.

Table 4 · FlashROM

Security Option	Description
Lock for both reading and writing Device Feature: FlashROM; Set Security: ☑; Encrypt: ☐; Security Settings: Read, Verify, Write	Allows the writing/erasing and reading of the FlashROM via the JTAG interface only with a valid Pass Key. Verification is allowed without a valid Pass Key.
Lock for writing Device Feature: FlashROM; Set Security: ☑; Encrypt: ☐; Security Settings: Read, Verify, Write	Allows the writing/erasing of the FlashROM via the JTAG interface only with a valid Pass Key. Reading and verification is allowed without a valid Pass Key.
Use the AES Key for both writing and verifying Device Feature: FlashROM; Set Security: ☑; Encrypt: ☑; Security Settings: Read, Verify, Write	Allows the writing/erasing and verification of the FlashROM via the JTAG interface only with a valid AES Key. This configures the device to accept an encrypted bitstream for

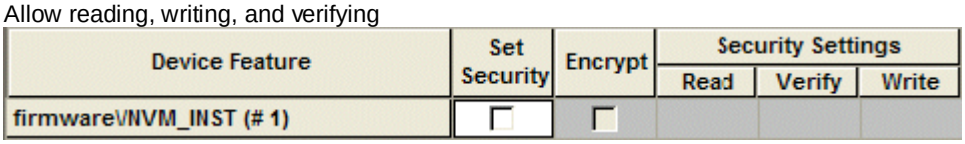
Security Option	Description
	reprogrammin g and verification of the FlashROM. Use this option if you complete final programming at an unsecured site or if you plan to update the design at a remote site in the future. Note: The bitstream that is read back from the FlashROM is always unencrypted (plain text).
Allow reading, writing, and verifying Device Feature: FlashROM; Set Security: ☐; Encrypt: ☐; Security Settings: Read, Verify, Write	Allows writing/erasing , reading and verification of the FlashROM content with a plain text bitstream and without requiring a valid Pass Key or an AES Key.

Note: The FPGA Array can always read the FlashROM content regardless of these Security Settings.

Table 5 · Embedded Flash Memory Block

Security Option	Description
Lock for reading, verifying, and writing Device Feature: firmware\NVM_INST (# 1); Set Security: ☑; Encrypt: ☐; Security Settings: Read, Verify, Write	Allows the writing and reading of the Embedded Flash Memory Block via the

Security Option	Description
	JTAG interface only with a valid Pass Key. Verification accomplished by reading back and compare.
Lock for writing Device Feature | Set Security | Encrypt | Security Settings: Read | Verify | Write firmware\NVM_INST (# 1) | ☑ | ☐ | Read | Verify | Write	Allows the writing of the Embedded Flash Memory Block via the JTAG interface only with a valid Pass Key. Reading and verification is allowed without a valid Pass Key.
Use AES Key for writing Device Feature | Set Security | Encrypt | Security Settings: Read | Verify | Write firmware\NVM_INST (# 1) | ☑ | ☑ | Read | Verify | Write	Allows the writing of the Embedded Flash Memory Block via the JTAG interface only with a valid AES Key. This configures the device to accept an encrypted bitstream for reprogrammin g of the Embedded Flash Block. Use this option if you complete final programming at an unsecured site or if you plan

Security Option	Description
	to update the design at a remote site in the future. The bitstream that is read back from the Embedded Flash Memory Block is always unencrypted (plain text), when a valid pass key is provided.
Allow reading, writing, and verifying  Device Feature: firmware\NVM_INST (# 1) Set Security: ☐ Encrypt: ☐ Security Settings: Read, Verify, Write	Allows writing, reading and verification of the Embedded Flash Memory Block content with a plain text bitstream and without requiring a valid Pass Key or an AES Key.

3. To make the Security Settings permanent, select **Permanently lock the security settings** check box. This option prevents any future modifications of the Security Setting of the device. A Pass Key is not required if you use this option.

   Note: When you make the Security Settings permanent, you can never reprogram the Silicon Signature. If you Lock the write operation for the FPGA Array or the FlashROM, you can never reprogram the FPGA Array or the FlashROM, respectively. If you use an AES key, this key cannot be changed once you permanently lock the device.

4. (SmartFusion Only) Enable M3 Debugger option enables access to the M3 debugger even if security is enforced. Select the **Enable M3 debugger** checkbox if you want to access the M3 debugger after programming.
5. To use the Permanent FlashLock™ feature (One-time programmable or OTP), select **Lock for both writing and verifying** for FPGA Array, **Lock for both reading and writing** for FlashROM, **Lock for reading, writing, and verifying** each Embedded Flash Memory Block (for Fusion and SmartFusion), if present, and select the **Permanently lock the security settings** checkbox as shown in the figure below. This will make your device one-time programmable.

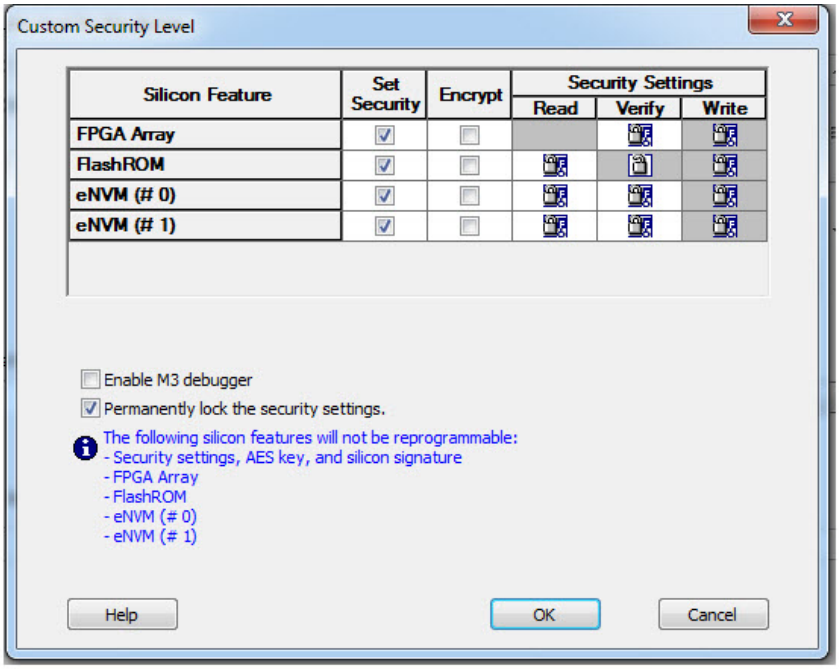


Figure 109 · Custom Security Level

6. Click the **OK** button. The **Security Settings** page appears with the **Custom security settings** information as shown in the figure below.

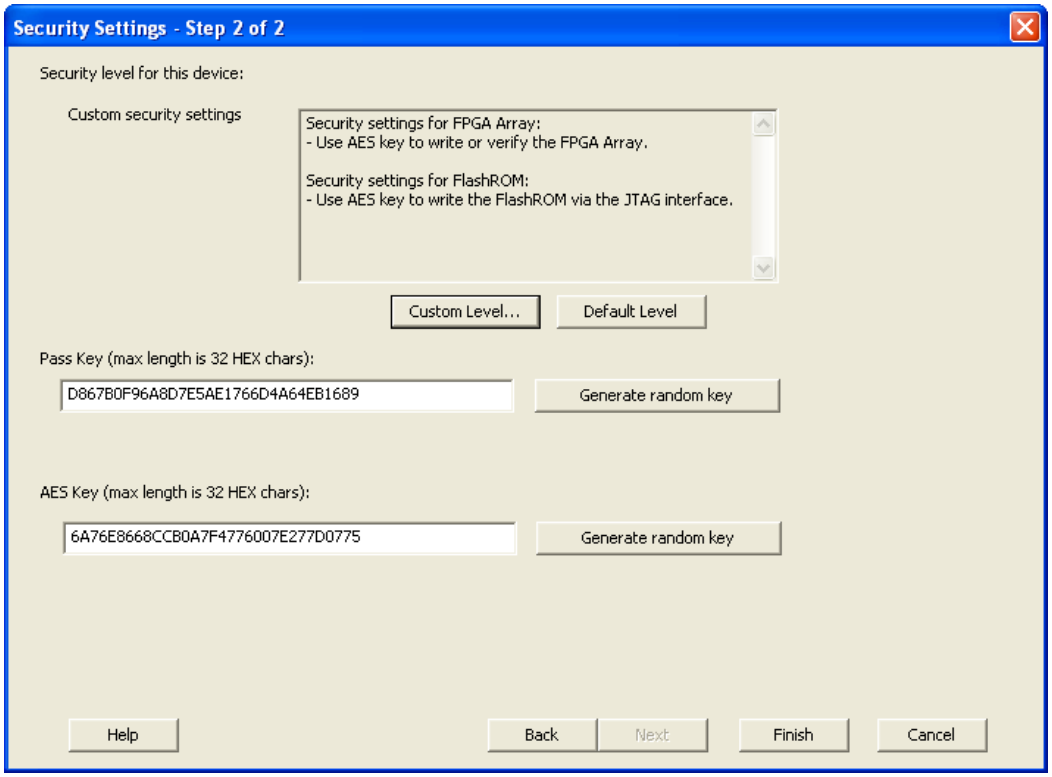


Figure 110 · Security Settings

Reprogramming a Secured Device

You must know the previous Security Settings of the device before you can reprogram a device with Security Settings.

To program a secured device:

1. In the Generate Programming File window, click the **Programming previously secured devices(s)** check box (see figure below).

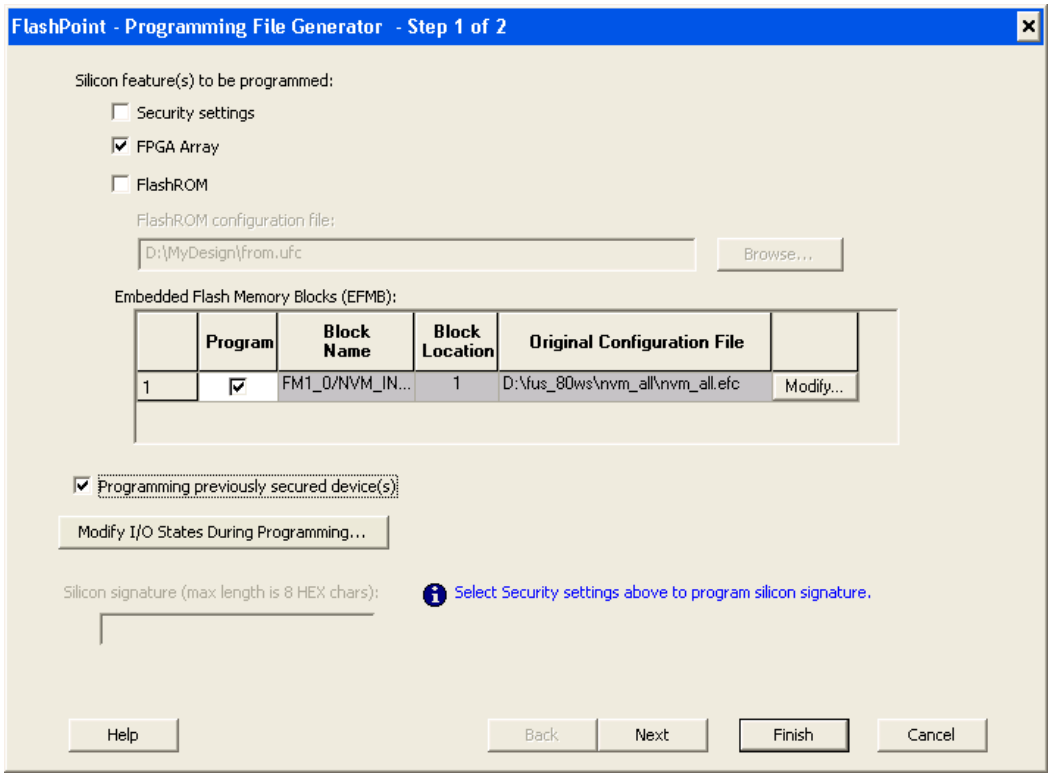


Figure 111 · Generate Programming File

2. Specify the previously programmed security setting for the FlashROM and/or the FPGA Array. To generate a programming file for encrypted programming please ensure that the Security settings checkbox is unchecked.
3. If you programmed the device with a custom security level, click the **Custom Level** button to open the Custom security dialog box, and select the **Security Settings for the FPGA Array** or the FlashROM that you programmed (see figure below).

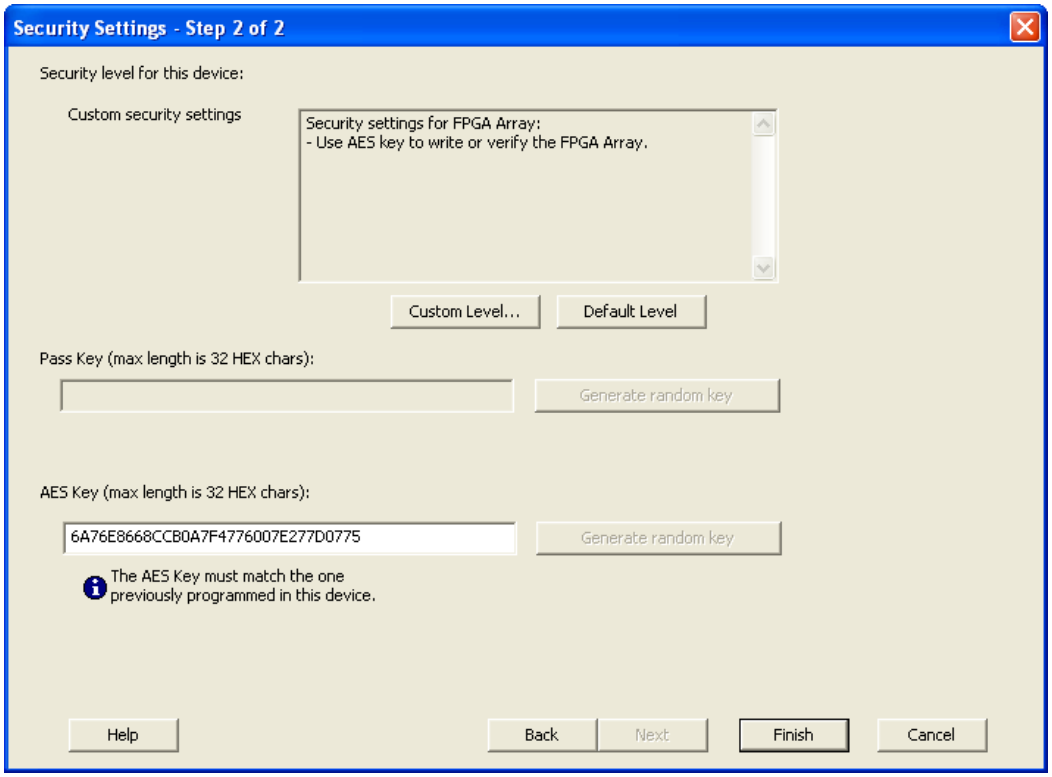


Figure 112 · Security Settings

4. Enter the previously programmed Pass Key and/or the AES Key.
5. Click **Finish**.

Note: Enter the AES Key only if you want to perform encrypted programming.

Programming a Secured SmartFusion Device

After you create a PDB you may wish to export a programming file for a secured device. To do so:

1. Create a PDB file (as explained above) with security set to **High** or **Medium**. Save the PDB file.
2. From the **File** menu, choose **Export Single Programming File**. The Export Programming Files dialog box appears.
3. Click the **Export programming file(s) for currently secured device** checkbox. This exports programming files for devices that already have security settings programmed.
4. Choose your outputs and enter your output file **Name** and **Location**.
5. Click **Export** to create the file(s). Your updated secured programming files are in the directory you specified.

Custom Serialization Data for FlashROM Region

FlashPoint enables you to specify a custom serialization file as a source to provide content for programming into a Read from file FlashROM region. You can use this feature for serializing the target device with a custom serialization scheme.

To specify a FlashROM region:

1. From the Properties section in the FlashROM Settings page, select the file name of the custom serialization file (see figure below). For more information on custom serialization files, see Custom Serialization Data File Format.

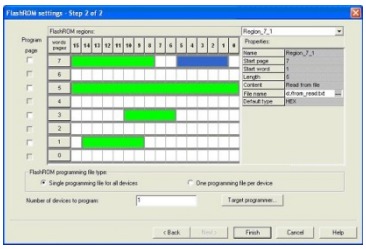


Figure 113 · FlashROM Settings

2. Select the FlashROM programming file type you want to generate from the two options below:

- Single programming file for all devices option: generates one programming file with all the values in the custom serialization file.
- One programming file per device: generates one programming file for each value in the custom serialization file.

3. Enter the number of devices you want to program.
4. Click the **Target Programmer** button.
5. Select your target Programmer type.
6. Click **OK**.

Custom Serialization Data File Format

FlashPoint supports custom serialization data files that specify the data in binary, HEX, decimal, or ASCII text. The custom serialization data files may contain multiple data with the Line Feed (LF) character as the delimiter. You can create a file by entering serialization data into any type of text editor. Depending on the serialization data format (hex, ASCII, binary, decimal), input the serialization data according to the size of the region you specified in the FlashROM settings page.

Semantics

Each custom serialization file has only one type of data format (binary, decimal, Hex or ASCII text). For example, if a file contains two different data formats (i.e. binary and decimal) it is considered an invalid file.

The length of each data file must be shorter or equal to the selected region length. If the data is shorter then the selected region length, the most significant bits shall be padded with 0's. If the specified region length is longer then the selected region length, it is considered an invalid file.

The digit / character length is as follows:

```
-Binary digit: 1 bit
-Decimal digit: 4 bits
-Hex digit: 4 bits
-ASCII Character: 8 bits
```

Note: Note the standard example below:

If you wanted to use, for example, device serialization for three devices with serialization data 123, 321, and 456, you would create file name from_read.txt. Each line in from_read.txt corresponds to the serialization data that will be programmed on each device. For example, the first line corresponds to the first device to be programmed, the second line corresponds to the second device to be programmed, and so on.

Hex serialization data file example

The following example is a Hex serialization data file for a 40-bit region. Enter the serialization data below into file created by any text editor:

```
123AEd210
AeB1
0001242E
```

Note: If you enter an invalid Hex digit such as 235SedF1, an error occurs. An error will also occur if you enter data that is out of range, i.e. 4300124EFE.

The following is an example of programming "AeB1" into Region_7_1 located on page 7, from Word 5 to Word 1 in the FlashROM settings page. See Custom serialization data for FlashROM region for more information.

	Table 15	...		Word 5	Word 4	Word 3	Word 2	Word 1	Word 0
Page 7	...	...	...	00	00	00	AE	B1	...

Binary serialization data file example

The following example is a binary serialization data file for a 16-bit region:

```
1100110011010001
100110011010011
11001100110101111 (This is an error: data out of range)
1001100110110111
1001100110110112 (This is an error: invalid binary digit)
```

Decimal serialization data file example

The following example is a decimal serialization data file for a 16-bit region:

```
65534
65535
65536 (This is an error: data out of range)
6553A (This is an error: invalid decimal digit)
```

Text serialization data file example

The following example is a text serialization data file for a 32-bit region:

```
AESB
A )e
ASE3 23 (This is an error: data out of range)
65A~
1234
AEbF
```

Syntax

Indentations in the syntax below indicate a wrapped line. If a line wraps and is not indented, then it should appear on one line; you may need to expand your help window to view the syntax correctly.

```
Custom serialization data file =
        <hex region data list> | <decimal region data list> |
```

```
        <binary region data list> | <ascii text data list>
Hex region data list = <hex data> <new line> { < hex data> <new line> }
Decimal region data list = <decimal data> <new line> {<decimal data><new line> }
Binary region data list = <binary data> <new line> { <binary data> <new line> }
ASCII text region data list = < ascii text data> <new line> { < ascii text data> <new
line> }
hex data = <hex digit> {<hex digit>}
decimal data = < decimal digit> {< decimal digit>}
binary data = < binary digit> {< binary digit>}
ASCII text data = <ascii character> {< ascii character >}
new line = LF
binary digit = ‘0’|‘1’
decimal digit = ‘0’|‘1’|‘2’|‘3’|‘4’|‘5’|‘6’|‘7’|‘8’| ‘9’
hex digit = ‘0’|‘1’|‘2’|‘3’|‘4’|‘5’|‘6’|‘7’|‘8’|‘9’|‘A’|‘B’|‘C’|‘D’ | ‘E’| ‘F’ |
        ‘a’| ‘b’ | ‘c’| ‘d’ | ‘e’| ‘f’
ascii character = characters from SP(0x20) to‘~’(0x7E).
```

File Format Limitations

The read from file data size cannot exceed the size of the region. The maximum size supported for each format is described below:

HEX - limited to the size of the FlashROM page. Maximum size of 128-bits

DEC - 32-bit unsigned numbers. Maximum decimal value is: 4294967295

BIN - limited to the size of the FlashROM page. Maximum size of 128-bits

TEXT - limited to the size of the FlashROM page. Maximum size of 128-bits

Specifying I/O States During Programming

In Libero SoC, the I/O states can be set prior to programming, and held at the set values during programming. This feature is only available after layout is completed.

1. From the Designer GUI, click **Modify I/O States During Programming** . The Programming File Generator window appears.
2. Click **Specify I/O States During Programming** to display the Specify I/O States During Programming dialog box.
3. 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).
4. 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 Specifying I/O States During Programming - I/O States and BSR Details for more information about 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

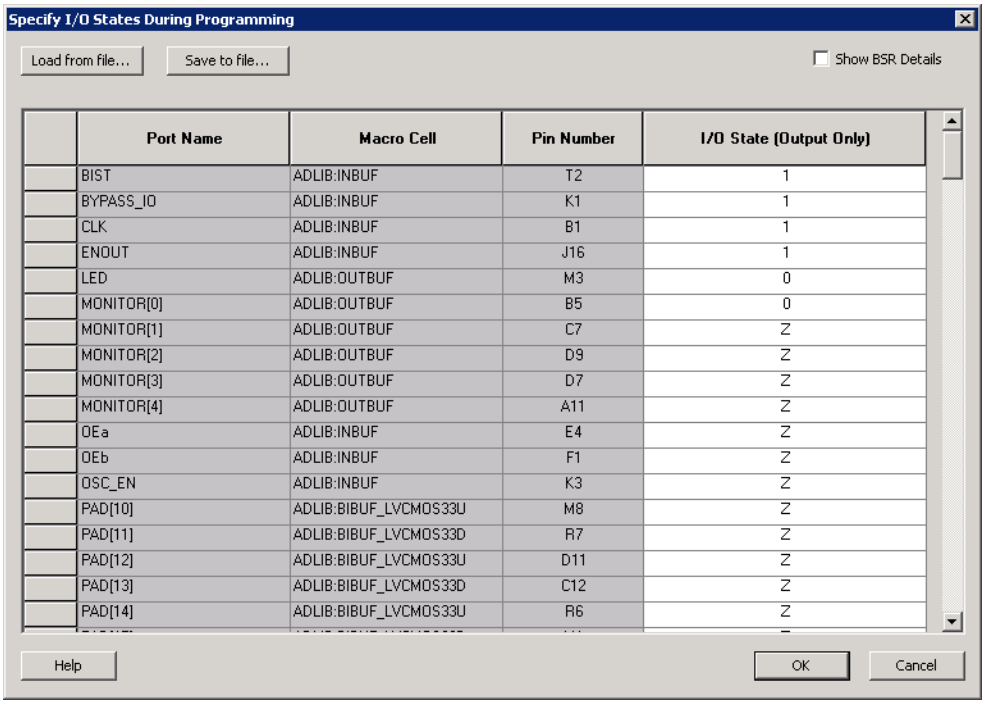


Figure 114 · I/O States During Programming Window

6. Click **OK** to return to the FlashPoint – Programming File Generator window.

Note: I/O States During Programming are saved to the ADB and resulting programming files after completing programming file generation.

Custom I/O Settings and Boundary Scan Registers

Each I/O in your device is comprised of an Input, Output and Output Enable Boundary Scan Register (BSR) cell..

The BSR cells enable you to define I/O states during programming and control the individual states for each Input, Output, and Output Enable register.

The Specify I/O States During Programming dialog box enables access to each of these BSR cells for control over the individual states. You can use the I/O State (Output Only) settings to set a specific output state and ignore the other values for the individual BSR elements, or you can click the Show BSR Details checkbox for control over the settings for each Input, Output Enable, and Output as you exit programming.

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 6 · 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 7 · BSR Details I/O Output Settings

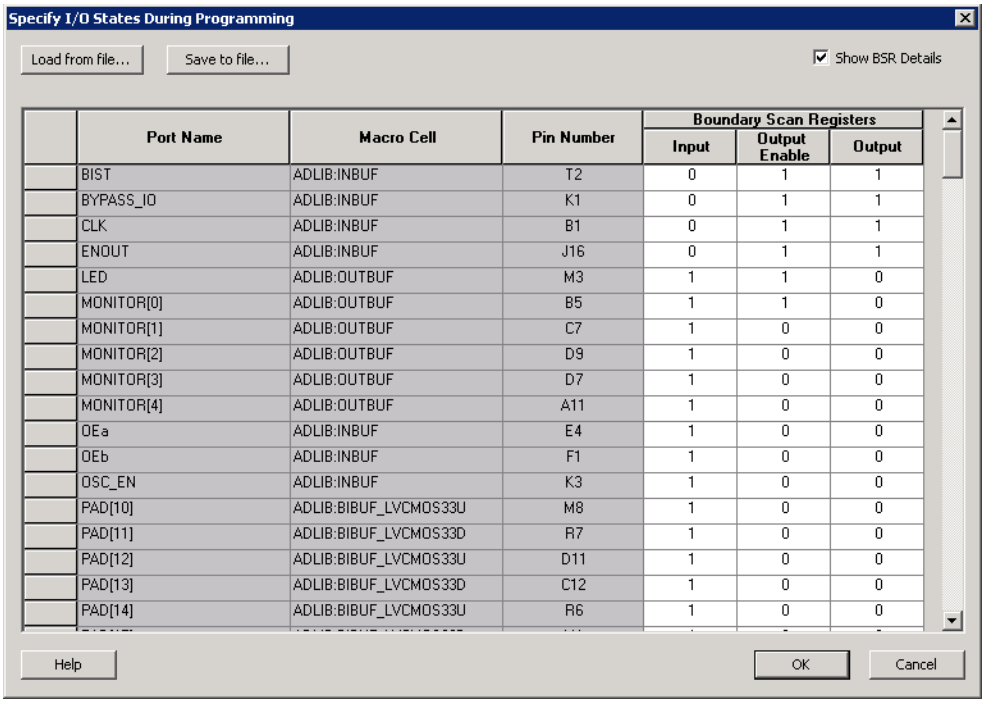
Output State	Settings		
	Input	Output Enable	Output
Z (Tri-State)	Don't Care	0	Don't Care
0 (Low)	Don't Care	1	0
1 (High)	Don't Care	1	1
Last Known State	Last State	Last State	Last State

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 115 · Boundary Scan Registers

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

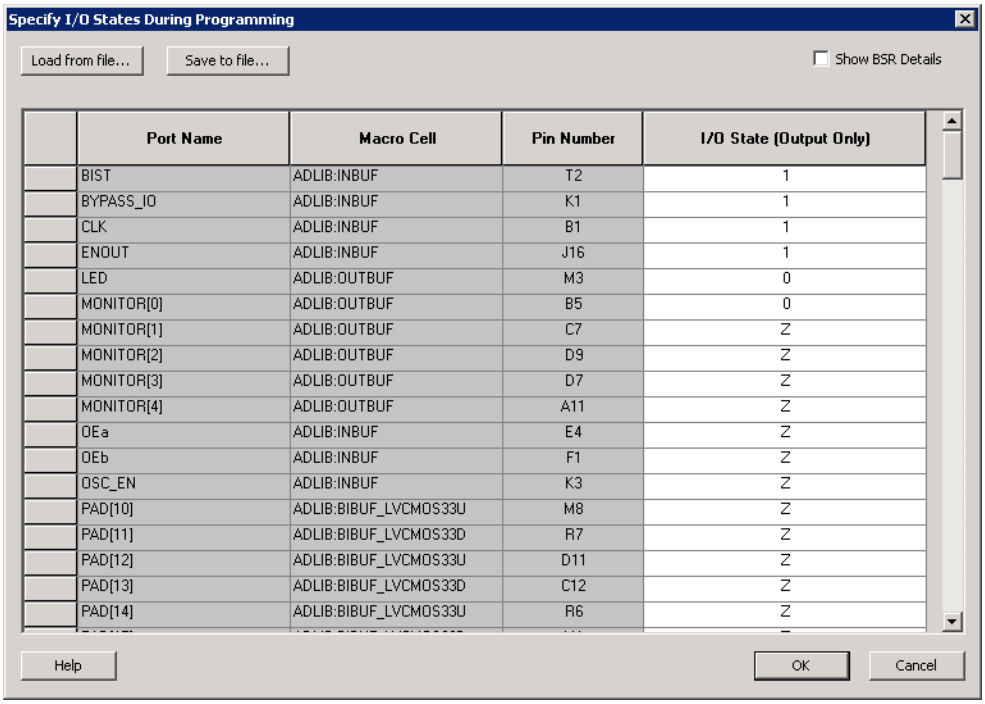
Lists the I/O type, such as INBUF, OUTBUF, PLLs, etc.

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.



Port Name	Macro Cell	Pin Number	I/O State (Output Only)
BIST	ADLIB:INBUF	T2	1
BYPASS_IO	ADLIB:INBUF	K1	1
CLK	ADLIB:INBUF	B1	1
ENOUT	ADLIB:INBUF	J16	1
LED	ADLIB:OUTBUF	M3	0
MONITOR[0]	ADLIB:OUTBUF	B5	0
MONITOR[1]	ADLIB:OUTBUF	C7	Z
MONITOR[2]	ADLIB:OUTBUF	D9	Z
MONITOR[3]	ADLIB:OUTBUF	D7	Z
MONITOR[4]	ADLIB:OUTBUF	A11	Z
OEa	ADLIB:INBUF	E4	Z
OEb	ADLIB:INBUF	F1	Z
OSC_EN	ADLIB:INBUF	K3	Z
PAD[10]	ADLIB:BIBUF_LVCMOS33U	M8	Z
PAD[11]	ADLIB:BIBUF_LVCMOS33D	R7	Z
PAD[12]	ADLIB:BIBUF_LVCMOS33U	D11	Z
PAD[13]	ADLIB:BIBUF_LVCMOS33D	C12	Z
PAD[14]	ADLIB:BIBUF_LVCMOS33U	R6	Z

Figure 116 · I/O States During Programming Dialog Box

Generate a DAT file

DAT files are generated via the Generate Programming Files dialog box.

To access the Generate Programming Files dialog box from Libero SoC and generate a DAT file:

1. In the Design Flow window, expand **Implement Design**, right-click **Generate Programming Data** and choose **Open Interactively**. This opens Designer.
2. Click **Programming File** to start FlashPoint.
3. Set your feature and I/O options if necessary. Click **Finish**. This opens the Generate Programming File dialog box, as shown in the figure below.

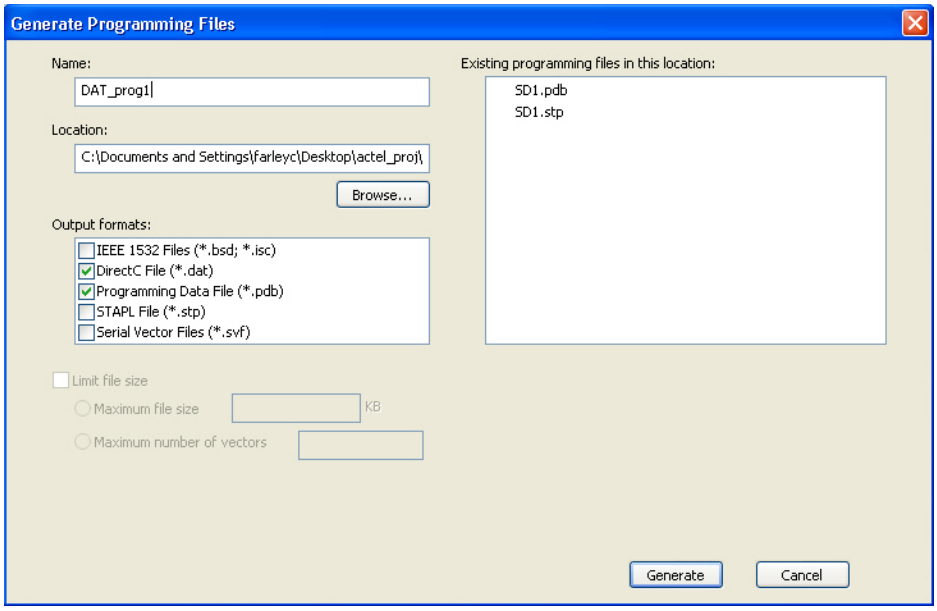


Figure 117 · Generate Programing Files Dialog Box - DirectC File (*.dat)

4. Set your output file **Name** and **Location**.
5. Set your Output Formats to **DirectC file (*.dat)** and **Programming Data File (*.pdb)**.
6. Click **Generate** to create your file.

FlashLock®

Microsemi's SmartFusion devices contain FlashLock circuitry to lock the device by disabling the programming and readback capabilities after programming. Care has been taken to make the locking circuitry very difficult to defeat through electronic or direct physical attack.

FlashLock has three security options: No Lock, Permanent Lock, and Keyed Lock.

No Lock

Creates a programming file which does not secure your device.

Permanent Lock

The permanent lock makes your device one time programmable. It cannot be unlocked by you or anyone else.

Keyed Lock

Within each device, there is a multi-bit security key user key. The number of bits depends on the size of the device. Once secured, read permission and write permission can only be enabled by providing the correct user key to first unlock the device. The maximum security key for the device is shown in the dialog box.

Generating Bitstream and STAPL files

Bitstream allows you to generate a STAPL file for SmartFusion, IGLOO, ProASIC3, Fusion devices. Please consult the Program Files table to find out which file type you should choose.

To generate a STAPL file:

1. From the **Tools** menu, choose **Programming File**.
2. Select **Bitstream** or **STAPL** from the **File Type** drop-down list box. Bitstream files are not available for SmartFusion, IGLOO, ProASIC3 and Fusion devices.
3. **FlashLock**. Select one of the following options:

- **No Locking:** Creates a programming file which does not secure your device.
- **Use Keyed Lock:** Creates a programming file which secures your device with a FlashLock key. The maximum security key for the device is shown in the dialog box. The maximum security key for the device is shown in the dialog box.
- **Use Permanent Lock:** Creates a one-time programmable device.

4. Click **OK**. Designer validates the security key and alerts you to any concerns.

Note: The bitstream file header contains the security key.

SmartFusion2 and IGLOO2 Programming Tutorials

SPI Programming Tutorial Overview

SmartFusion2 and IGLOO2 devices can be programmed using multiple programming methods.

Refer to the Programming User Guide for your respective device for details on different programming methods.

The following tutorials describe programming methods that involve the MSS/HPMS SPI_0 ports:

- Auto Programming
- In Application Programing (IAP) Tutorial
- Programming Recovery - Auto Update

Auto Programming

Auto programming uses the Microcontroller Subsystem SPI port (SPI_0) to fetch the bitstream from the external SPI flash (pre-programmed) connected to the same port and then program the FPGA Fabric and eNVM. The transaction between the System controller and the external SPI flash happens in SPI master mode.

The SPI_0 port is enabled to operate as a SPI master at power-on reset or DEVRST_N assertion if the FLASH_GOLDEN_N pin is pulled low. The Auto Programming can be protected in the Security Policy Manager > Update Policy.

1. Right-click **Configure Security and Programming > Configure Security** and choose **Configure Options** (as shown in the figure below).

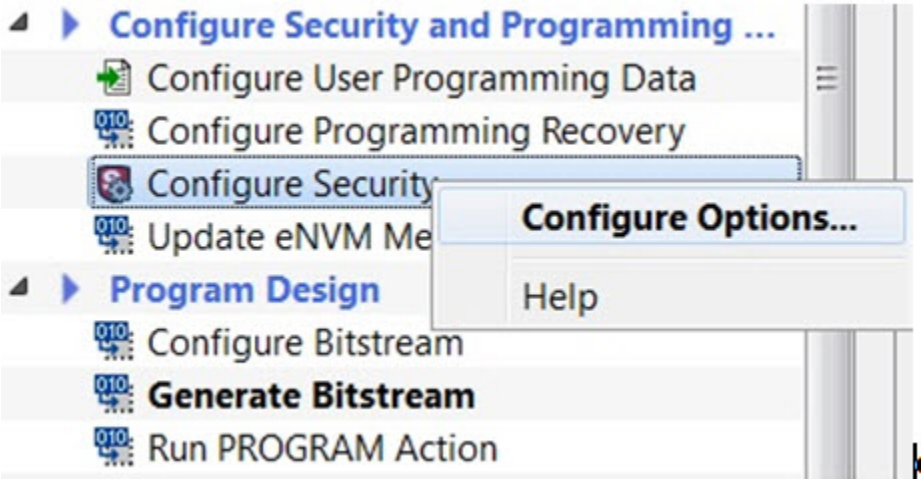


Figure 118 · Launch Security Policy Manager

2. In Security key mode select **Enable customer security options**.
3. In Security policies click the checkbox to enable **Update Policy**.
4. Click the **Update Policy** button and click to enable **Auto Programming**.

Auto programming uses a *.spi programming file. You must export a *.spi file using Export Bitstream (under Handoff Design for Production).

Note: 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.

DEVICE_INFO programming

When the DEVICE_INFO programming action is run with a STAPL file for SmartFusion2 and IGLOO2 devices, you will see the following message:

"Info: Algorithm Version, Programmer, Software Version, Programming Software, Programming Interface Protocol, and Programming File Type information do not get programmed with Auto Programming, Auto Update (IAP), Programming Recovery, or IAP/ISP Services and should be ignored."

In Application Programming (IAP) Tutorial

In Application Programming (IAP) is a two-step process. In the first step, the intended bitstream is written to an external SPI flash connected to SPI_0 port of MSS. In the second step, Cortex-M3 (SmartFusion2) or user logic in Fabric (IGLOO2) calls the IAP programming service of the system controller to program the FPGA fabric and eNVM with the bitstream fetched from the external SPI flash.

In order to perform IAP, the required functional blocks must be configured by pre-programming the chip.

To enable SmartFusion2/IGLOO2 device for IAP, you must configure the following components in Libero SoC:

- **Source of application image**: eSRAM, eNVM or DDR/SDR memories or Fabric
- **Source of programming bitstream**: MSS peripherals including SPI_0 port
- **Interface between System Controller and Cortex-M3**: COMM_BLK

To create a programming file with IAP configuration (SmartFusion2 steps shown):

1. Create a Libero SoC project targeting the desired SmartFusion2 device. Select **SmartFusion2 Microcontroller**. Click the checkbox to enable **Use Design Tool** and select the **SmartFusion2 Microcontroller Subsystem (MSS)**. Click OK to continue. A dialog box appears to name your new component.
2. Name the component **IAP_Setup_MSS_0**.
3. Double-click **IAP_Setup_MSS_0** to configure the blocks:

- **ENVM** stores the two step IAP application code.
- **USB** is the interface used to read the programming bitstream and download it to the External SPI flash.
- **MSS_CCC** is the Cortex-M3 Clock
- **RESET** Controller is the Chip reset
- **SPI_0** is connected with External SPI flash
- **MMUART_1** is the Host PC communication to get status

  See the SmartFusion2 Microcontroller Subsystem User Guide section on How to Use Blocks for details on how to configure them.

4. Click **SmartDesign > Generate Component**.
5. Complete the design flow up to Place and Route and export the firmware to enable the MSS component.
6. Click Generate Bitstream or Export Bitstream to export a programming file you can use to program your device.

Note: Libero SoC will error out if SPI_0 routs to the fabric instead of a package pin.

DEVICE_INFO programming

When the DEVICE_INFO programming action is run with a STAPL file for SmartFusion2 and IGLOO2 devices, you will see the following message:

"Info: Algorithm Version, Programmer, Software Version, Programming Software, Programming Interface Protocol, and Programming File Type information do not get programmed with Auto Programming, Auto Update (IAP), Programming Recovery, or IAP/ISP Services and should be ignored."

Programming Recovery Tutorial

If programming recovery is enabled, SmartFusion2/IGLOO2 device automatically recovers from a power failure during a programming operation. Programming recovery requires an external SPI flash to be connected to the MSS SPI_0 port. The FPGA must be pre-programmed with the programming recovery settings. Programming recovery settings define the location of the image and how to recovery will take place.

For a blank device the image location is set to MSS SPI_0 port where an external SPI flash must be connected.

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.

To configure Programming Recovery:

1. Under **Configure Security and Programming Options** double-click **Configure Programming Recovery**.
2. Select **Enable Programming Recovery**. If Auto update is selected then Design version is mandatory. You must enter the design version in the Configure User Programming Data menu.
3. Check **Enable Programming Recovery** and set the SPI clock frequency and data transfer mode. Refer to the table below for SPI data transfer options. For a blank device SPO and SPH are defaulted to 1.

Table 8 · SPI Signal Polarity Modes

SPO	SPH	SPI Clock in Idle	Sampl e Edge	Shift Edge	SPI Select in Idle	SPI Select Between Frames
0	0	Low	Rising	Falling	High	Stays active until all the frames set by frame counter have been transmitted
0	1	Low	Falling	Rising	High	
1	0	High	Falling	Rising	High	
1	1	High	Rising	Falling	High	

Note: SPO = SPI clock polarity; SPH = SPI clock phase

If Enable Auto Update is checked, Programming recovery is automatically enabled. In this scenario when the device is powering up, it will be auto programmed with the bitstream stored in SPI_0 port if

the update SPI image design version is greater than the design version currently programmed in the device.

During programming recovery, if there is a programming failure due to power failure, then the the device will be recovered with the golden image.

Back Level protection - If you enable Back Level protection it provides bitstream replay protection. The BACKLEVEL value limits the design versions that the device can update. So, only programming bitstreams with DESIGNER > BACKLEVEL are allowed for programming

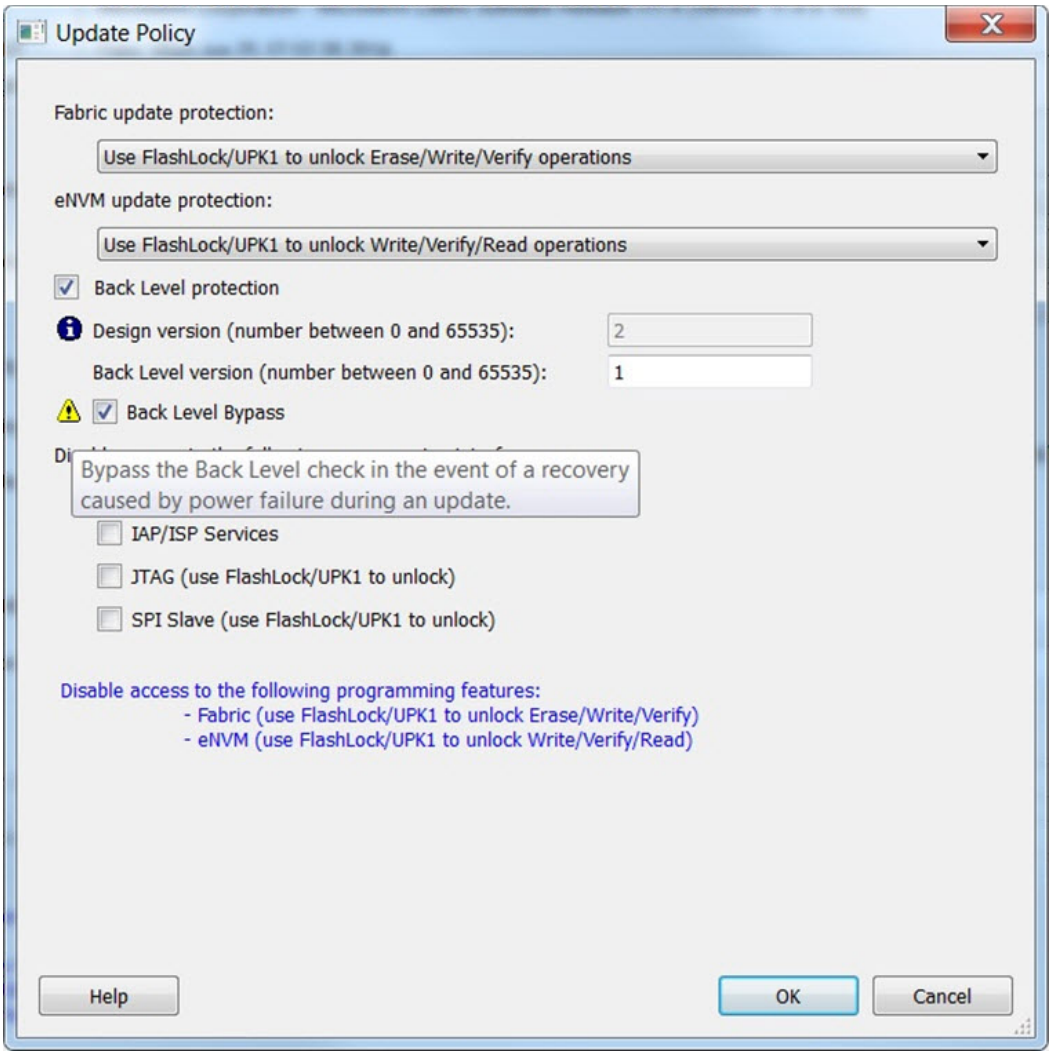


Figure 119 · Bypass Back Level Version Check

4. In the Update Policy dialog box click the checkbox to enable **Back Level Protection**, as shown in the figure above. Note that in order to use Back Level Protection you must enter the Design version in the Configure User Programming Data dialog box. By default the Back Level bypass is checked in order to continue with programming recovery with current/golden image in the event of power failure. This allows the golden SPI image to be programmed even if the design version is less than the backlevel version without the need of a pass key.

   Consider an instance where the device is programmed with a golden image first which has DESIGNVER = 2 and BACKLEVEL = 1. Now you want to program the device with an update image that has DESIGNVER = 3 and BACKLEVEL = 2. If the Back Level bypass is not checked and power

fails, the programming recovery will stop and the device will be left inoperable because the golden image DESIGNVER is not greater than the BACKLEVEL of update image.

5. Export the SPI directory for Programming recovery. The flash device on the MSS SPI_0 port contains a directory at address 0 with the information shown in the table below.

Offset	Name	Description
0	GOLDEN_IMAGE_ADDRESS[3:0]	Contains the address where the golden image starts
4	GOLDEN_IMAGE_DESIGNVER[1:0]	Contains the design version of the golden image
6	UPDATE_IMAGE_ADDRESS[3:0]	Contains the address where the update image starts
10	UPDATE_IMAGE_DESIGNVER[1:0]	Contains the design version of the update image

6. Right-click **Export Programming File** and choose **Configure Options**.
7. In the Export Bitstream dialog box, click the checkbox to enable **Export SPI directory for programming recovery** and click the **Specify SPI Directory** button.
8. In the SPI Directory dialog box enter the **Design version** and browse to the location of the Golden SPI Image and/or Update SPI Image. You must enter the **Address** of the bitstream manually (as shown in the figure below).

DEVICE_INFO programming

When the DEVICE_INFO programming action is run with a STAPL file for SmartFusion2 and IGLOO2 devices, you will see the following message:

"Info: Algorithm Version, Programmer, Software Version, Programming Software, Programming Interface Protocol, and Programming File Type information do not get programmed with Auto Programming, Auto Update (IAP), Programming Recovery, or IAP/ISP Services and should be ignored."

SmartFusion2 Programming Tutorial Overview

The SmartFusion2 Programming Tutorial describes the basic steps for SmartFusion2 programming.

Only the bold steps in the Design Flow window are required to complete and program your design. Note that the bold steps are completed automatically if you use the Build button.

1. MSS Configuration - eNVM

   eNVM configuration enables you to configure eNVM as a ROM so that it can be included in the eNVM digest calculations.
   Data Security Configuration controls which masters have access to which memory region within the MSS.

2. Generate Bitstream

   Generate bitstream for programming within Libero.

3. Edit Design Hardware Configuration

   Configures Device I/O States During Programming.

4. Configure Security and Programming Options

- Security Policy Manager
- Configure Bitstream
- Update eNVM Memory Content

5. Program Design
   Configure Actions/Procedures (sets programming options) and programs your device.
6. Handoff Design for Production
   - Export Bitstream
   - Export BSDL

MSS Configuration - eNVM

eNVM Configuration

You must create a MSS to configure your eNVM. Use System Builder to configure your eNVM for IGLOO2.

eNVM configuration enables you to configure eNVM as a ROM so that it can be included in the eNVM digest calculations. To do so:

1. Open your MSS and double-click the **eNVM block** to open the **eNVM configuration dialog box**, as shown in the figure below.

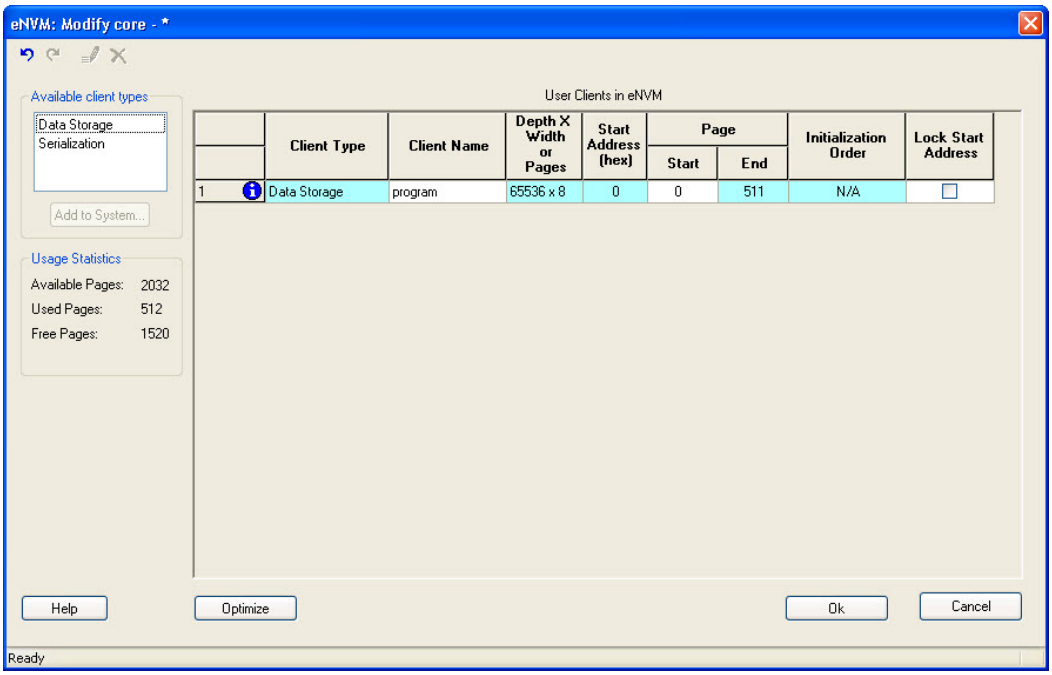
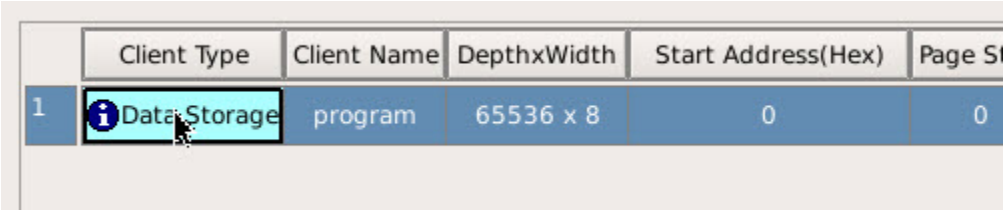


Figure 120 · eNVM Configuration Dialog Box

The example design shown in the figure above already has a Data Storage Client; it can be modified.

2. Double-click the **Data Storage Client**



This will open the **Modify Data Storage Client dialog box**, as shown in the figure below.

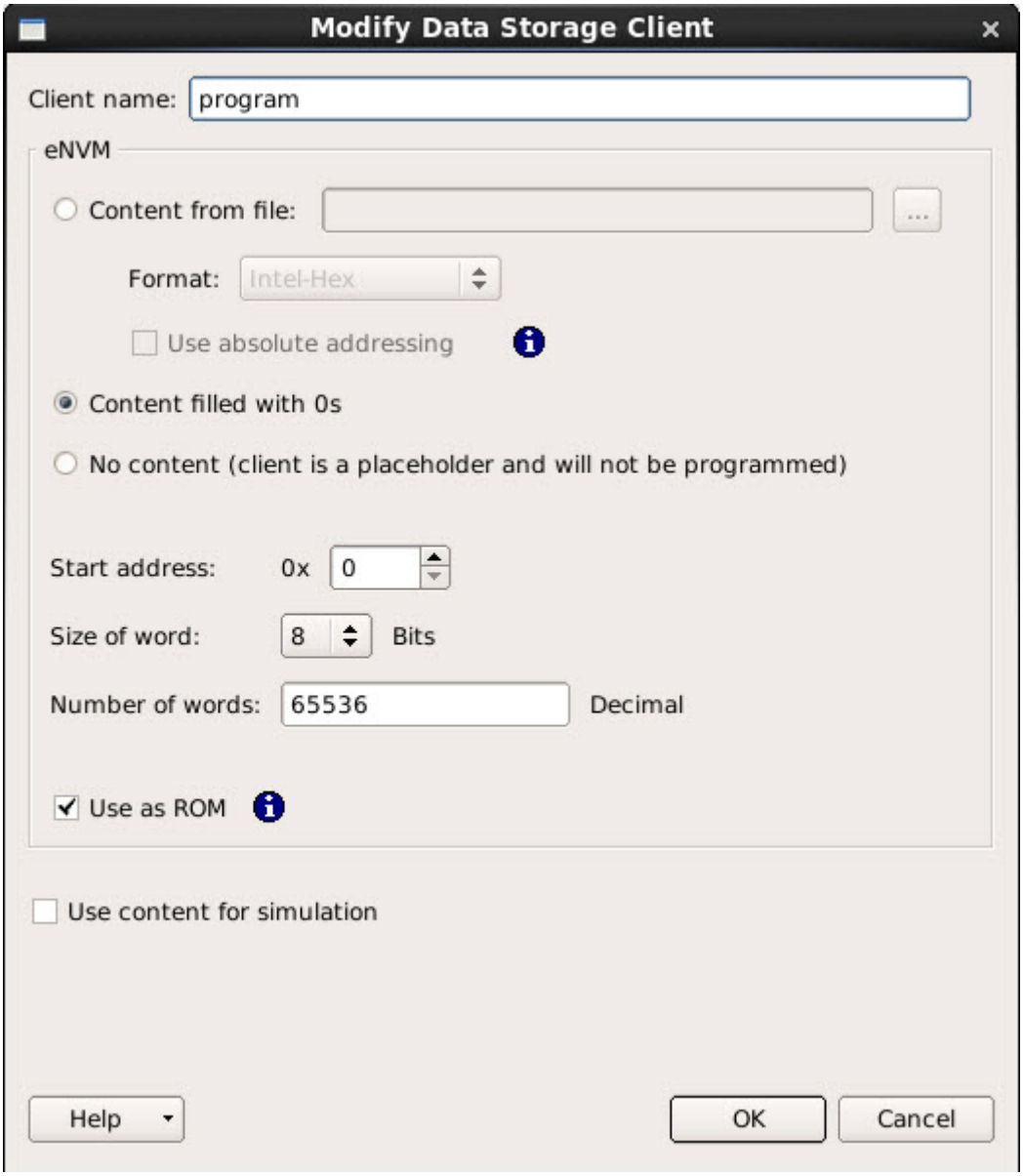


Figure 121 · Modify Data Storage Client Dialog Box - Use as ROM Selected

3. Click **Use as ROM** (as shown in the figure above) to configure the memory region as a ROM and include the eNVM digest calculations.
4. Click **OK** in the Modify Data Storage Client dialog box to continue.

5. Double-click **Serialization** to open the **Add Serialization Client dialog box** (as shown in the figure below) and reserve a client for serialization. Add your Client name and eNVM Start address and Number of pages, as appropriate.

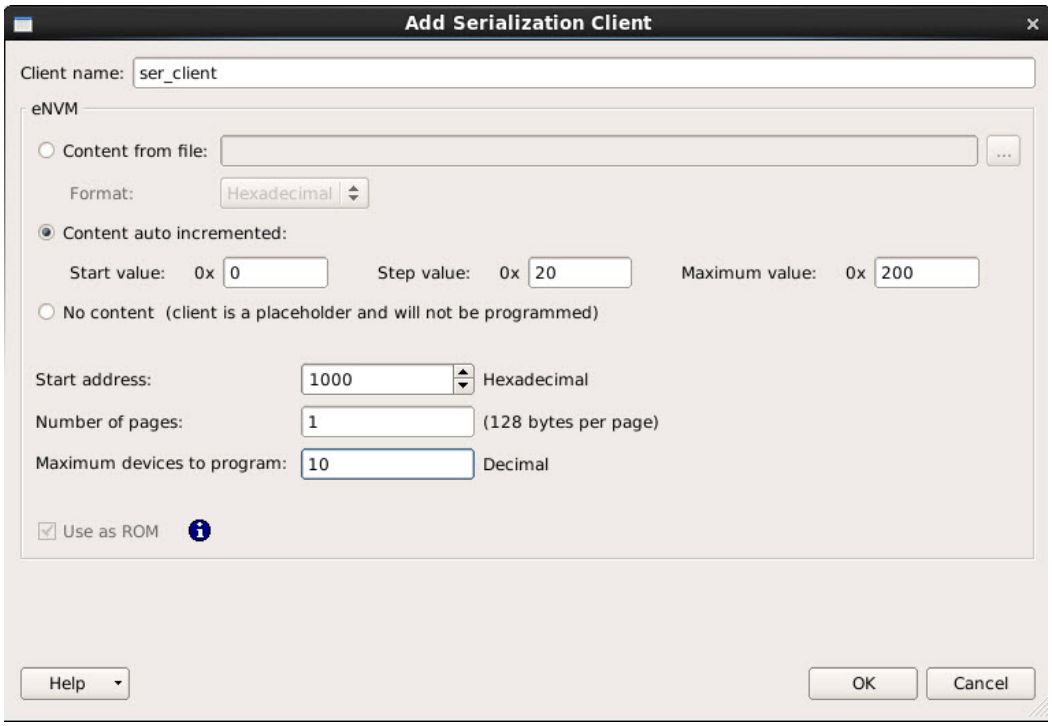


Figure 122 · Add Serialization Client Dialog Box

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's Guide for more information.

6. Click **OK** in the Add Serialization Client dialog box to continue.
7. Click **OK** in the eNVM Configuration dialog box to return to the MSS.

To re-configure the Clients after Synthesis and Compile, you can double-click Update eNVM Memory Content in the Design Flow window. See Update eNVM Memory Content

Enable Data Security

The Data Security Configuration controls which masters have access to which memory region within the MSS. To configure your data security:

Double-click the **Security** block in the MSS to open the **MSS Security Policies Configurator**, as shown in the figure below.

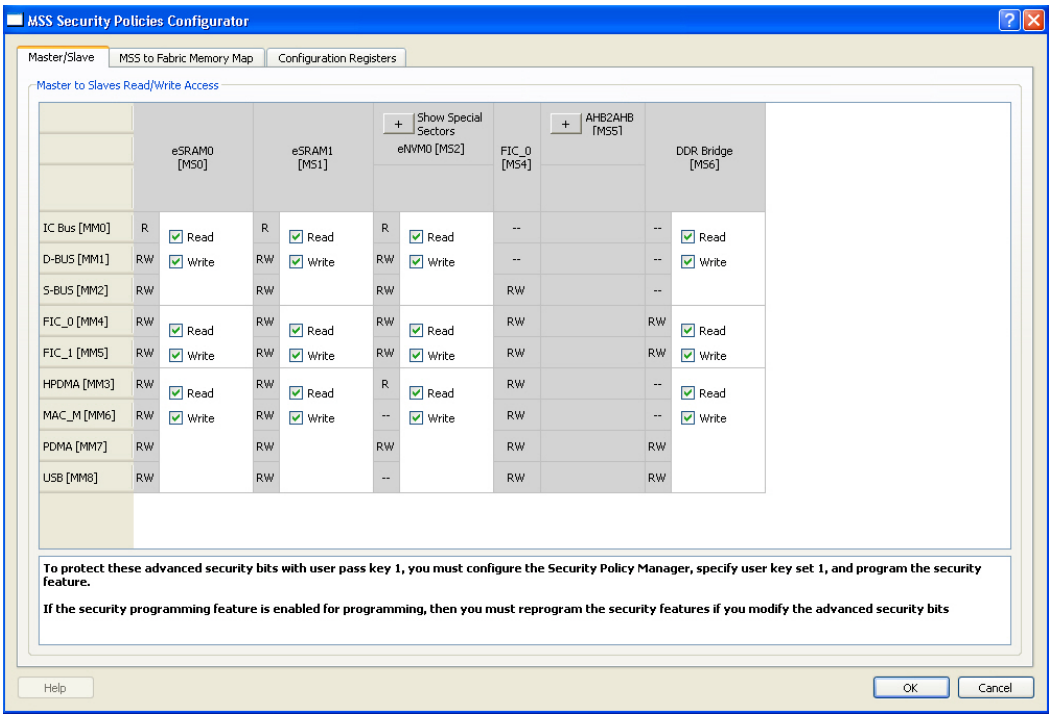


Figure 123 · MSS Security Policies Configurator

Masters are listed on the left, and slaves are shown at the top. All masters have access to all slaves by default; click to enable or disable Read/Write access for specific masters and slaves.

Restrict your master/slave Read/Write access according to your preference and click **OK** to continue.

Note: 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.

Generate Bitstream

Generates the bitstream for use with the Run PROGRAM Action tool.

The tool incorporates the Fabric design, eNVM configuration (if configured) and security settings (if configured) to generate the bitstream file. You need to configure the bitstream before you generate the bitstream. 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.

This operation is completed automatically as the last step if you use the Build button.

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.

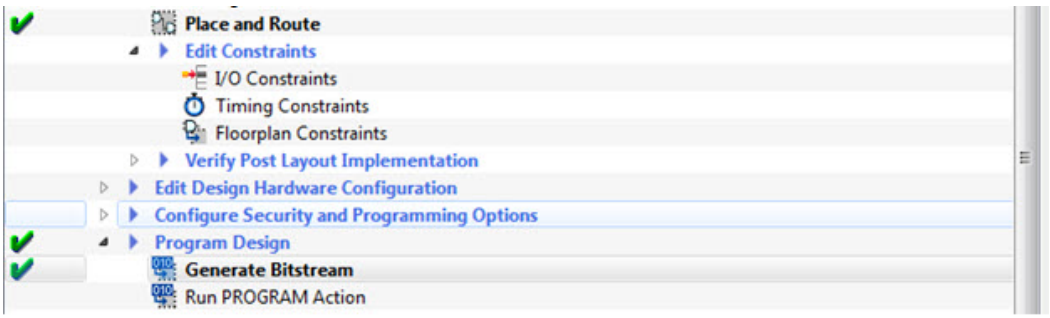


Figure 124 · Generate Bitstream (Complete)

See also

Configure Bitstream Dialog Box

Edit Design Hardware Configuration - Device I/O States During Programming

You can configure your FPGA I/Os while the device is being programmed using Device I/O States During Programming.

In the Design Flow window, expand **Edit Design Hardware Configuration** and double-click **Device I/O States During Programming**.

The Device I/O States During Programming dialog box appears.

Click a value in the I/O State (Output Only) column to set your I/O State options according to your preference, as shown in the figure below.

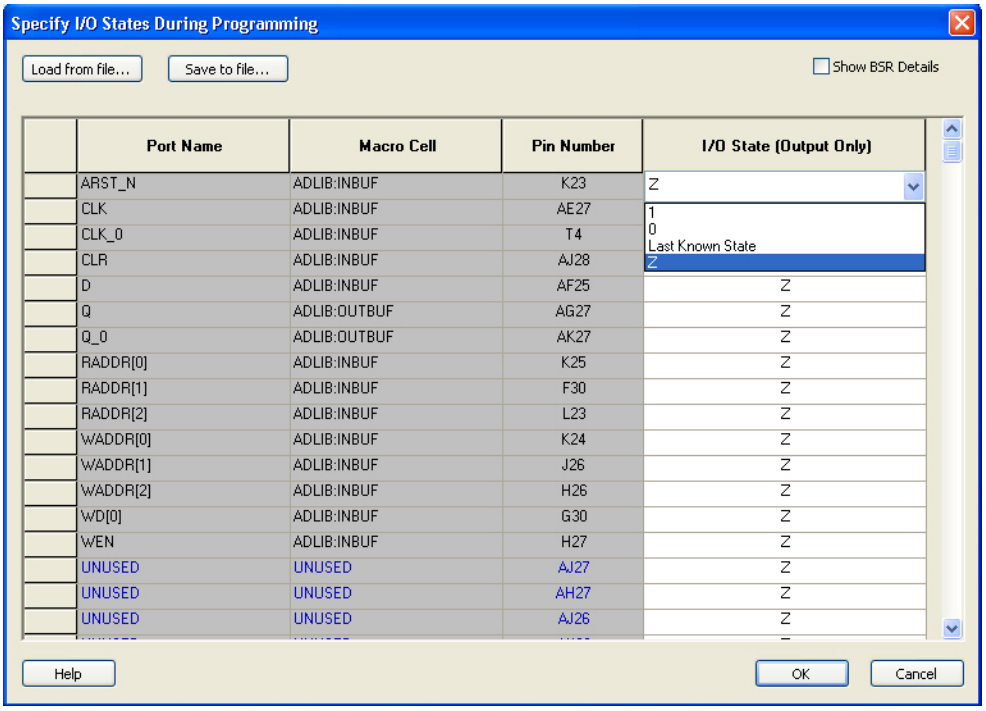


Figure 125 · Set I/O State

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.

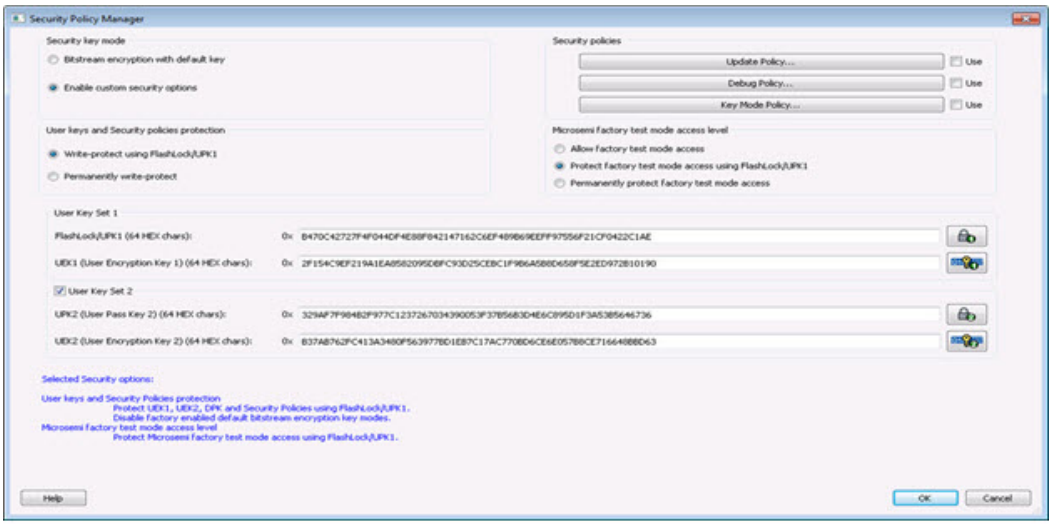


Figure 126 · Security Policy Manager Dialog Box

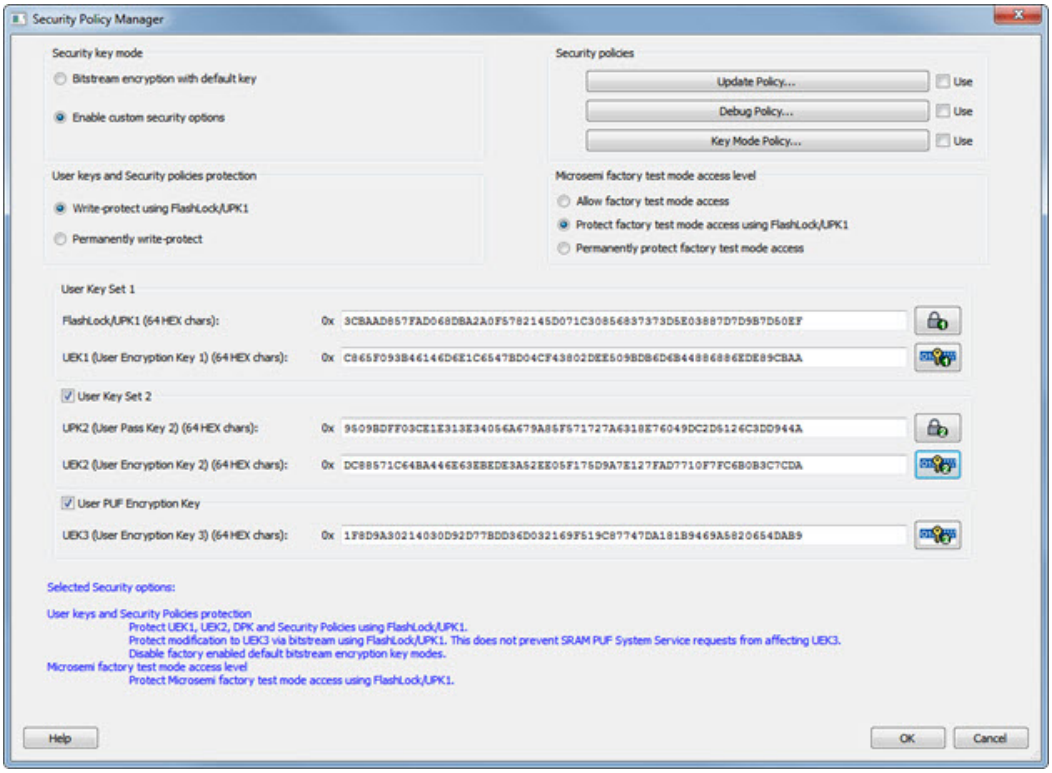


Figure 127 · Security Policy Manager Dialog Box (for devices supporting UEK3)

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

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.

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

Security Policies

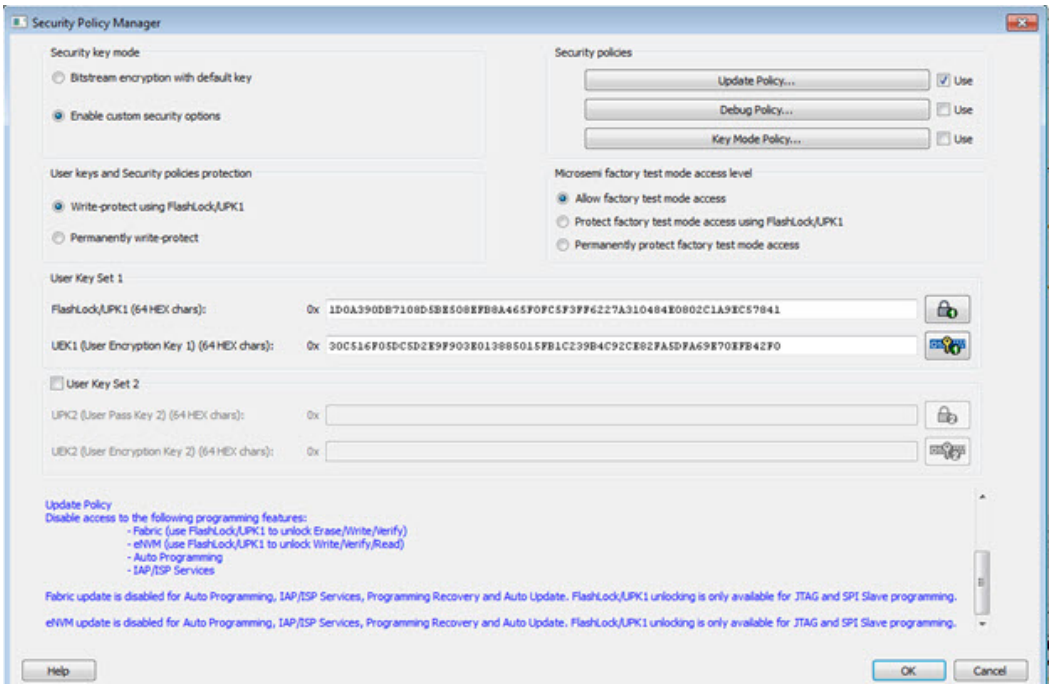
Update Policy - Sets your Fabric, eNVM and Back Level protections. See the Update Policy topic for more information.

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

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

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.

See the following example.



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 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.

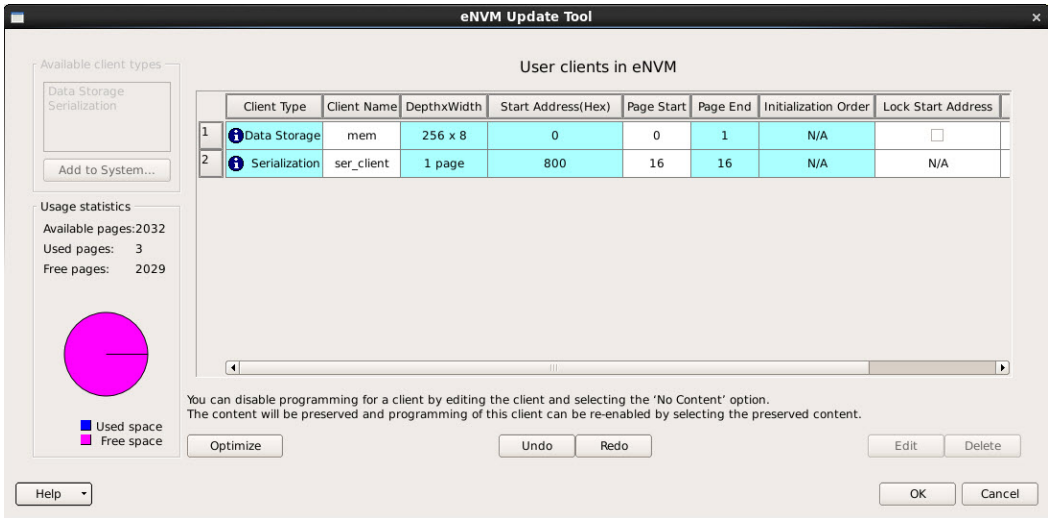


Figure 128 · eVNM Update Dialog Box

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

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.

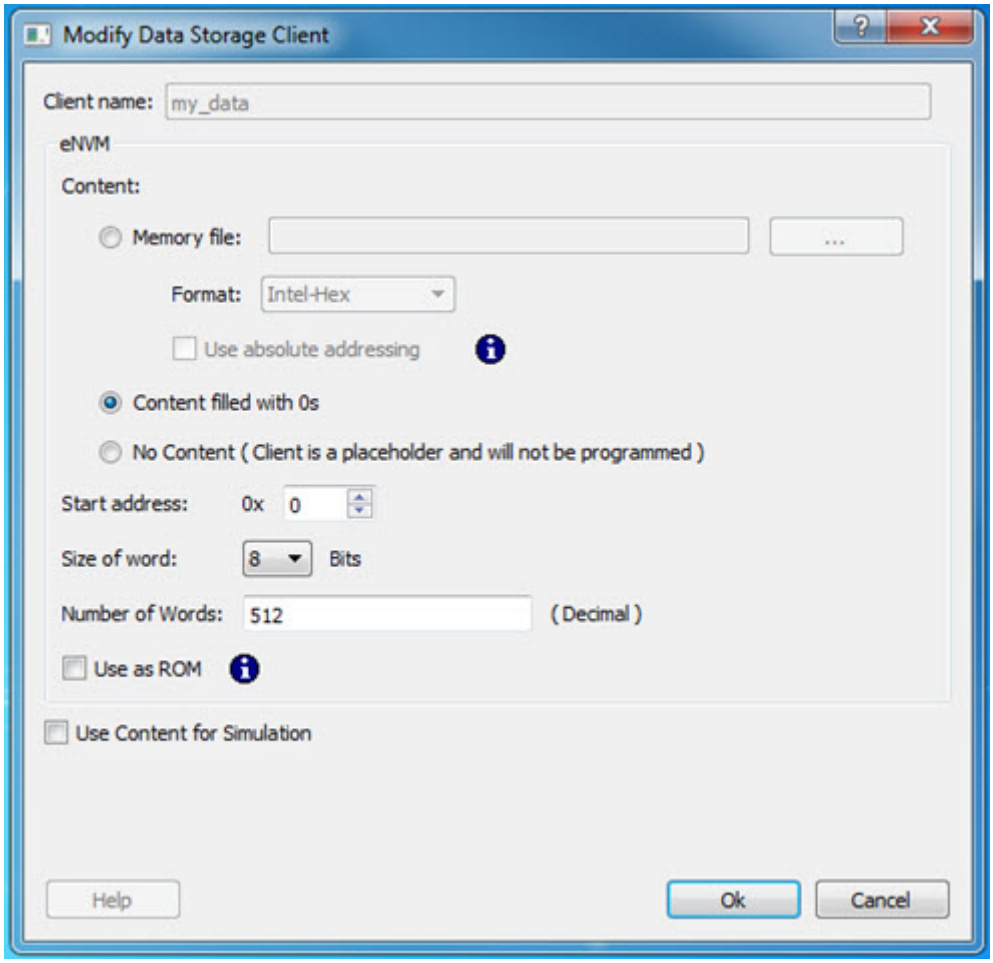


Figure 129 · Modify Data Storage Client Dialog Box

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.

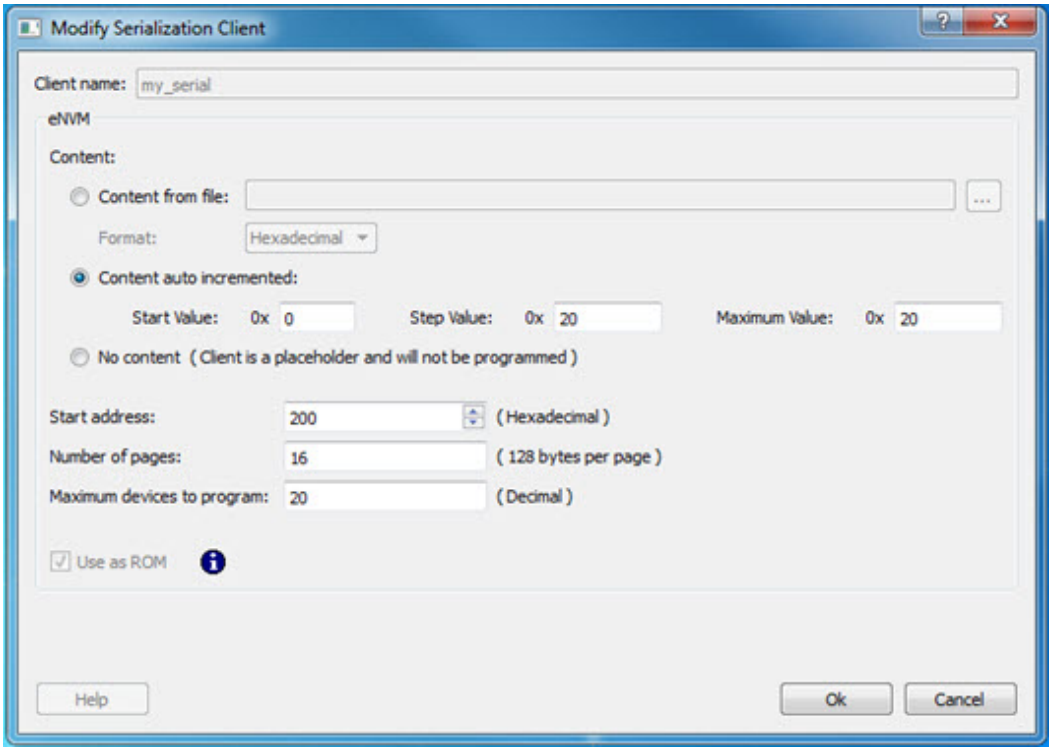


Figure 130 · Modify Serialization Client Dialog Box

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's Guide for more information.

Program Design - Run PROGRAM Action

Expand Program Design and double-click **Run PROGRAM Action** to program your device with default settings, as shown in the figure below.

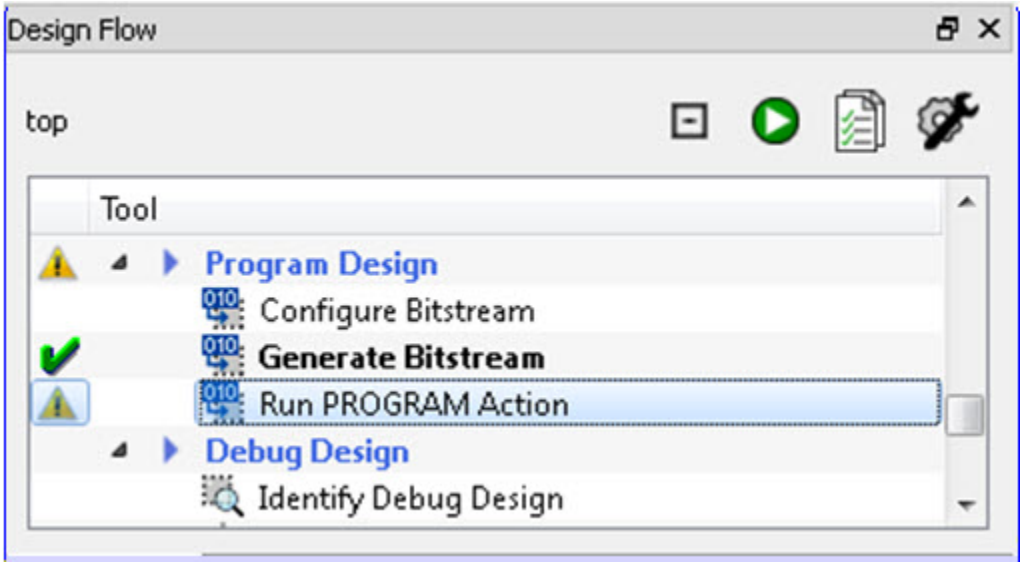


Figure 131 · Program Device in the Design Flow Window

Right-click **Run PROGRAM Action** and choose from the following menu options:

- **Clean and Run All** - Cleans all tools, deletes all reports and output files and runs through programming. All logs will be updated with new files.
- **Clean** - deletes only reports and output files associated with this tool; the other tool files and reports are unaffected.
- **Configure Actions/Procedures** - Enables you to set the specific Action you wish to program. Select your programming action from the dropdown menu.

Handoff Design for Production

- In order to handoff your design for production you must export a programming file or generate a BSDL file.
- See the Export Bitstream - SmartFusion2, IGLOO2, and RTG4 topic for complete instructions on how to handoff your design for production.
- You can also export your programming job.

Debug Design

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 file
7. From the Instumentor 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, select the edif netlist of the Identify implementation you want to use in the flow. Right-click **Compile** and choose **Organize Input Files > Organize Source Files** and select the edif netlist of your Identify implementation.
9. In Libero SoC, run Compile, Place and Route and Generate a Programming File with the edif netlist you created with the Identify implementation.
10. 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.

SmartDebug

Microsemi's SmartDebug tool complements design simulation by allowing verification and troubleshooting at the hardware level.

SmartDebug can be run in the following modes:

- Integrated mode from the Libero Design Flow
- Standalone mode
- Demo 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, 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. When launched in standalone mode, you must to go through SmartDebug project creation and import a Design Debug Data Container (DDC) file, exported from Libero, to access all debug features in the supported devices.

Note: In standalone mode, the Probe Insertion feature is not available in FPGA Array Debug, as it requires incremental routing to connect the user net to the specified I/O.

Demo Mode

Demo mode allows you to experience SmartDebug features (Active Probe, Live Probe, Memory Blocks, SERDES) without connecting a board to the system running SmartDebug.

Note: SmartDebug demo mode is for demonstration purposes only, and does not provide the functionality of integrated mode or standalone mode.

Note: You cannot switch between demo mode and normal mode while SmartDebug is running.

For detailed information about SmartDebug, refer to the SmartDebug User Guide.

Handoff Design for Production

Export Bitstream – SmartFusion2 and IGLOO2

Export Bitstream Files enables you to export 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 Securiy Policy Manager 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, or **SPI file** if you need support for IAP or SPI Directory if you need support for Programming Recovery.
3. Select the bitstream components (Fabric/eNVM) that you want to program.
4. Enter your **Bitstream file name** and click **OK** to export the selected bitstream files.

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

- Export Bitstream - RTG4

Note: Security Programming is not supported in RTG4.

Export Bitstream - RTG4

Bitstream File Name – Sets the name of your bitstream file. The prefix varies depending on the name of your top-level design.

Choose the Bitstream file format:

- STAPL file
- Chain STAPL file (Enabled only when there are two or more devices in the chain)
- DAT file
- SPI File
- SVF file (Serial Vector Format) specifications from ASSET InterTech, 1999.

Note: Security Programming is not supported for RTG4.

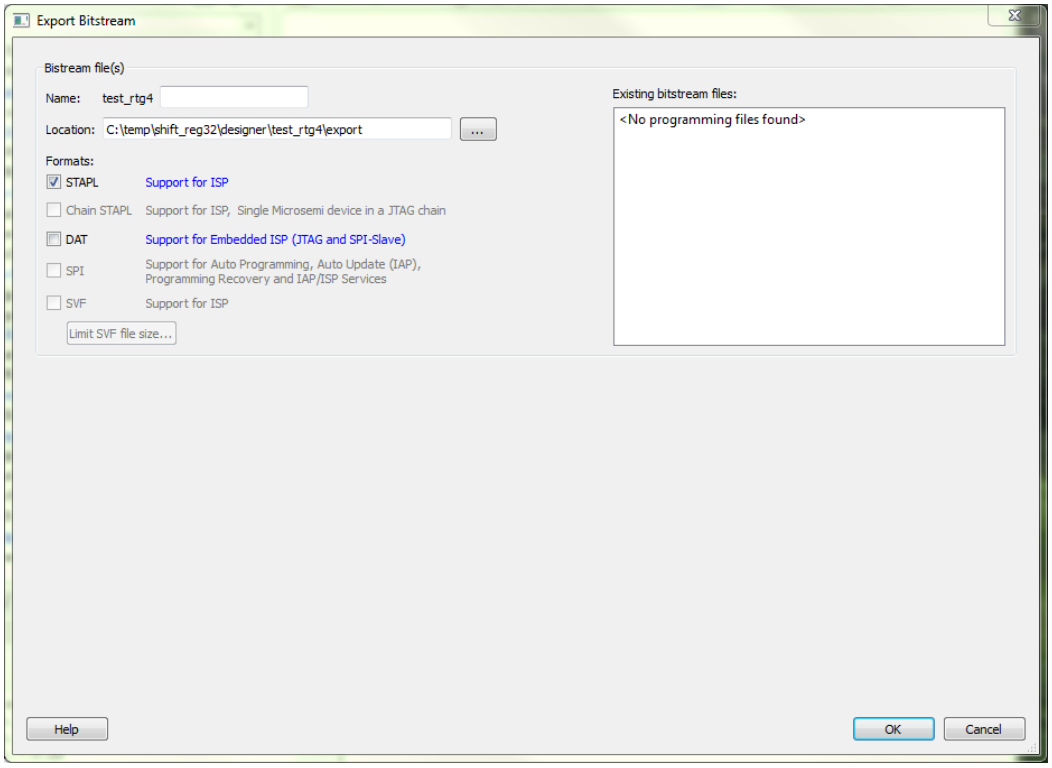


Figure 132 · Export Bitstream Dialog Box

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 the Browse Button to navigate to a disk location where you want the pin report to be saved to.

Check the checkbox to make your selections:

- Pin Report sorted by Port Name
- Pin Report sorted by Package Pin Name
- I/O Bank Report
- I/O Register Combining Report

The Pin Report lists the pins in your device sorted according to your preference: sort by Port Name or Sorted by Package Pin Name. The Pin Report generates two files:

- <design>_pinrpt_name.rpt - Pin report sorted by name.
- <design>_pinrpt_number.rpt - Pin report sorted by pin number.

You must select at least one report.

Export Pin Report generates a Bank Report by default; the filename is <design>-bankrpt.rpt. Export Pin Report also generates an I/O Register Combining Report listing the I/Os which have been combined into a Register for betting timing performance.

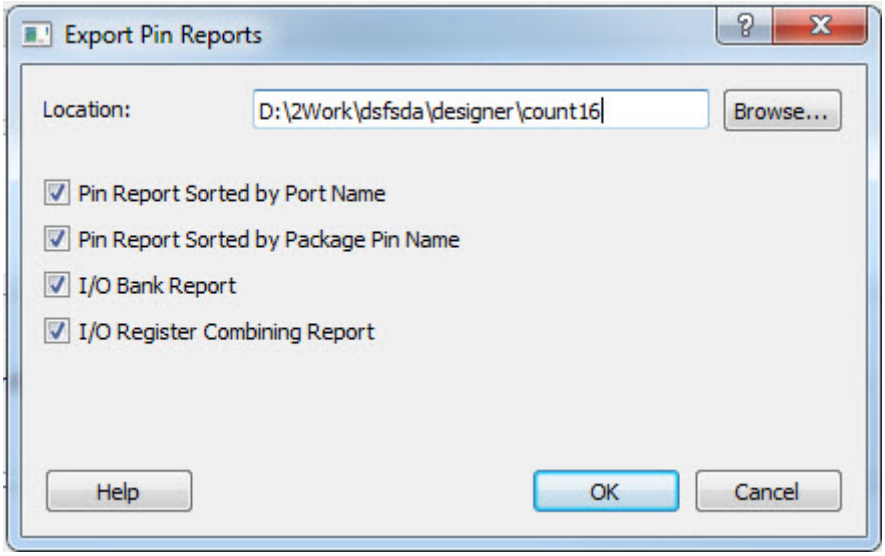


Figure 133 · Export Pin Report Dialog Box

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 Datasheet/Report.

For SmartFusion, IGLOO, ProASIC3, and Fusion devices, do the following:

1. Open Designer interactively.
2. Click **Compile**.
3. Click **Layout**.
4. Click **Programming File** and click **Finish** when done.
5. In the Generate Programming Files dialog box, under Output formats, select **IEEE 1532 Files (*.bsd, *.isc)**.
6. Click **Generate** to generate the related BSDL file.

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.

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, IGOO2 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].

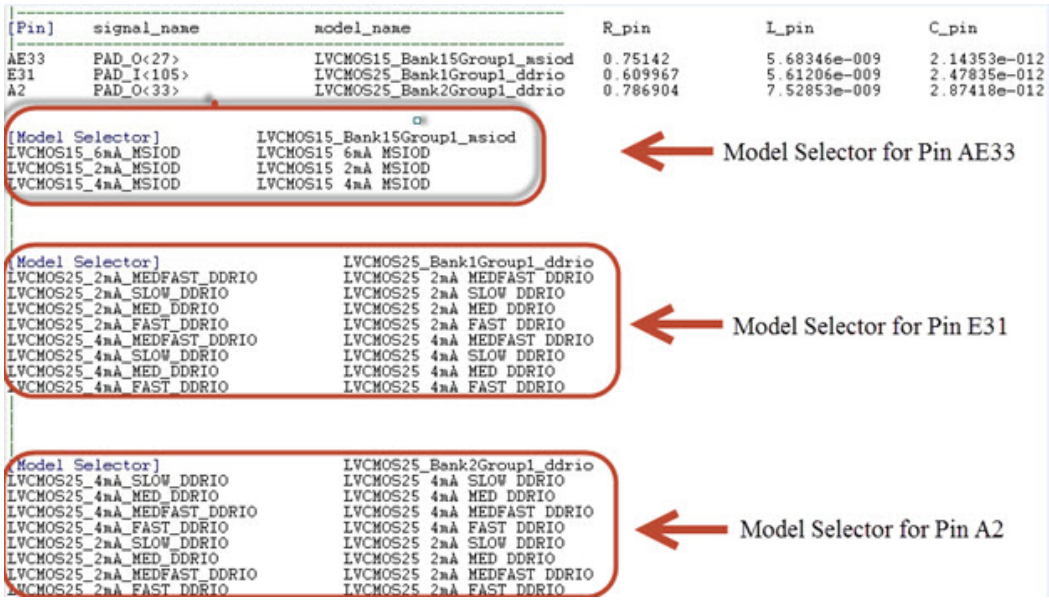


Figure 134 · Model Selector *.ibs File

With this Model Selector feature support, users 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 IBIS model application note for more information on IBIS models.

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.

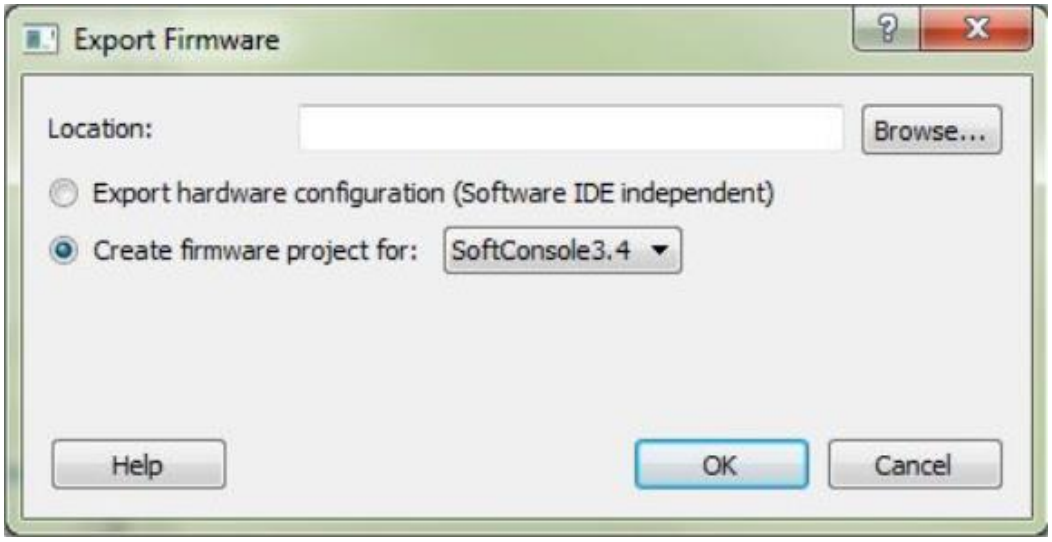


Figure 135 · Export Firmware Dialog Box

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.

Export hardware configuration (Software IDE independent):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 firmware project for: <selected Software Tool Chain>

This option is checked by default. 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.

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 supports the following versions of third-party development tools:

- SoftConsole v3.4
- IAR v7.4
- Keil V5.16

Reference Topics, Menus and Dialog Boxes

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).

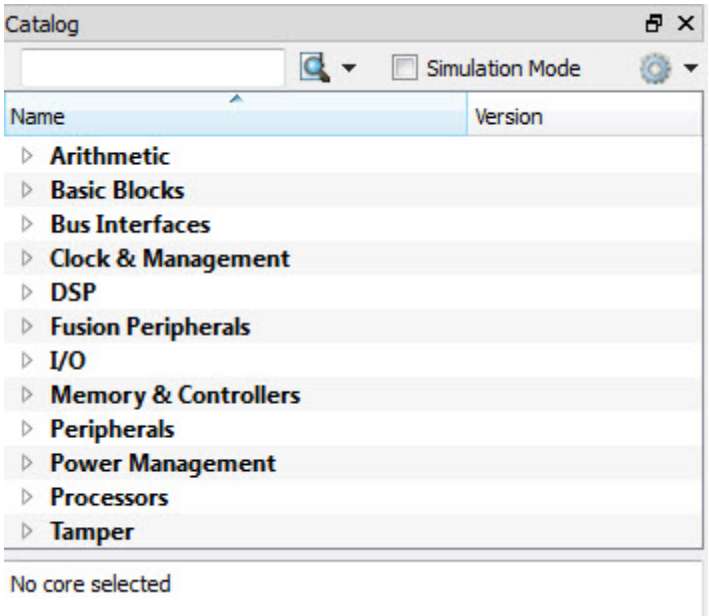


Figure 136 · Libero SoC Catalog

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)

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 .

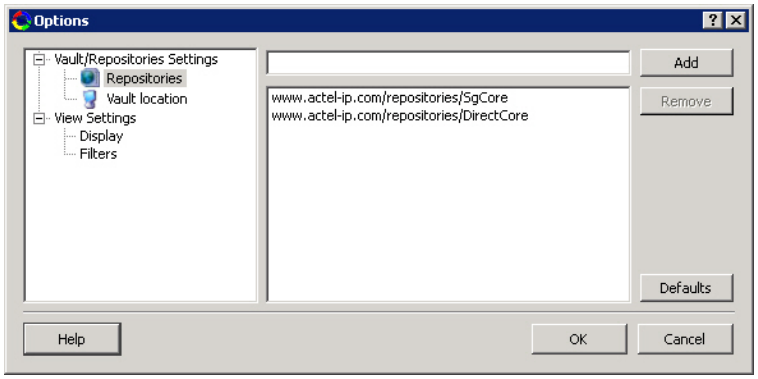


Figure 137 · Catalog Display Options Dialog Box

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.

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 Device Information

Device and package information, device variations, and operating conditions are set when you import a netlist and compile a new design. However, you can change this information for existing designs.

To change device information for existing designs:

1. In the **Project** menu, choose **Project Settings**. The Project Settings dialog box opens.
2. Select your updated options, such as Die, Package, and Speed.
3. Click **Close**.

Refer to the Microsemi FPGA Data Book or call your local Microsemi Sales Representative for information about device, package, speed grade, variations, and operating conditions.

Compatible Die Change

When you change the device, some design information can be preserved depending on the type of change.

Changing Die Revisions

If you change the die from one technology to another, all information except timing is preserved. An example is changing an A1020A (1.2μm) to an A1020B (1.0μm) while keeping the package the same.

Device Change Only

Constraint and pin information is preserved, when possible. An example is changing an A1240A in a PL84 package to an A1280A in a PL84 package.

Repackager Function

When the package is changed (for the same device), the Repackager automatically attempts to preserve the existing pin and Layout information by mapping external pin names based on the physical bonding diagrams. This always works when changing from a smaller package to a larger package (or one of the same size). When changing to a smaller package, the Repackager determines if any of the currently assigned I/Os are mapped differently on the smaller package. If any of the I/Os are mapped differently, then the layout is invalidated and the unassigned pins identified.

Core Manager

The Core Manager only lists cores that are in your current project. If any of the cores in your current project are not in your vault, you can use the Core Manager to download them all at once.

For example, if you download a sample project and open it, you may not have all the cores in your local vault. In this instance you can use the Core Manager to view and download them with one click. Click **Download All** to add any missing cores to your vault. To add any individual core, click the green download button.

To view the Core Manager, from the **View** menu choose **Windows > Cores**.

The column headings in the Core Manager are:

- **Name** - Core name.
- **Vendor** - Source of the core.
- **Core Type** - Core type.
- **Version** - Version of the core used in your project; it may be a later version than you have in your vault. If so, click **Download All** to download the latest version.

Deleting Files

Files can be deleted from the current project or from the disk.

To delete a file from the project:

1. Select the **Files** tab in the Design Explorer window.
2. **Right-click** the file and choose **Delete from Project**. The file remains on your disk.

To delete a file from your project and the disk:

1. Select the **Files** tab in the Design Explorer window.
2. **Right-click** the file and choose **Delete from Disk and Project**. The file is deleted from your disk and is no longer part of any project.

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. The software continuously analyzes and updates 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.

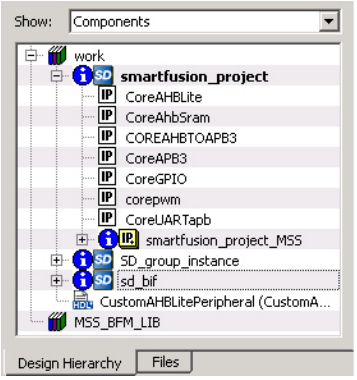


Figure 138 · Design Hierarchy

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.

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 pathname, 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 Design 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 Canvas in SmartDesign.

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
	IP core was instantiated into SmartDesign but the HDL netlist has not been generated
	Core
	Error during core validation
	Updated core available for download

Icon	Description
	HDL netlist

Design Menu - Libero SoC

Command	Icon	Function
Place and Route		Runs the push-button flow for place and route.
Reports		Creates and/or opens the Datasheet Reports for your project.
Design Summary		Generate a design summary for your project.

Designer in Libero SoC

Microsemi's Designer software is integrated with the Libero SoC Project Manager. The Designer interface opens only when you choose not to use the default settings in the push-button design flow.

To implement your design, click the Build button in the Libero SoC Design Flow window. If you wish to change the default settings for any element in the design flow, right-click the function and choose **Open Interactively**.

The following tools are available to run interactively:

SmartTime

SmartPower

NetlistViewer

PinEditor

ChipPlanner

I/O Attribute Editor

Edit Core Definition - Ports and Parameters Dialog Box

This dialog box appears when you add a core you created with HDL+.

Click to select any Extracted Parameter and click the Delete button to remove it from the list. Extracted Parameters may be configured if you add the HDL+ core to the Canvas.

If you delete an Extracted Parameter and want to re-add it to the list click the **Re-extract ports and parameters from HDL button**.

Click **Add/Edit bus interfaces** to open the Edit Core Definition - Bus Interfaces dialog box.

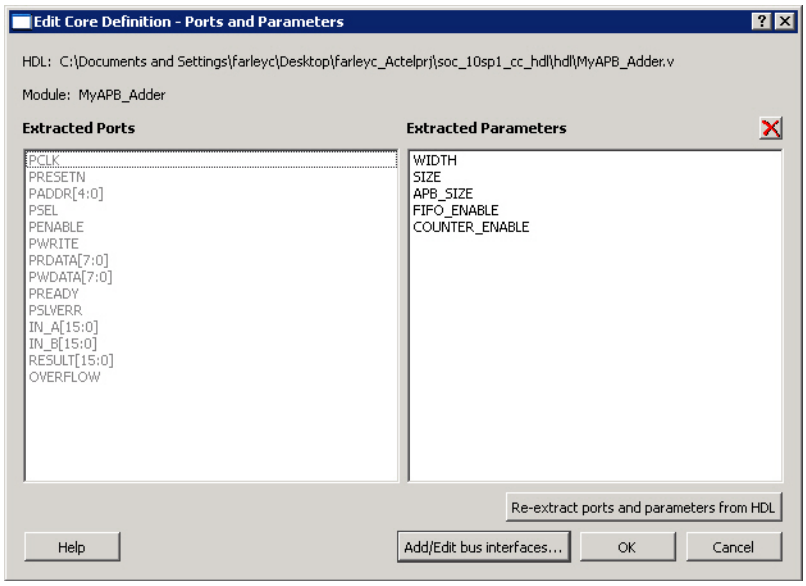


Figure 139 · Edit Core Definition - Ports and Parameters Dialog Box

Edit Menu - Libero SoC

Command	Icon	Shortcut	Function
Undo		CTRL + Z	Reverses your last action
Redo		CTRL + Y	Reverses the action of your last Undo command
Find		CTRL + F	Displays the Find dialog box, which you use to locate instances, nets, ports, and regions
Find Next		F3	Finds the next occurrence of the text in the Find field

Execute Script Dialog Box

You can use the Execute Script dialog box to run Tcl scripts from within Libero SoC. You do not need to have a design open to run a script.

Specify a script file, enter Arguments (if necessary), and click **Run** to execute.

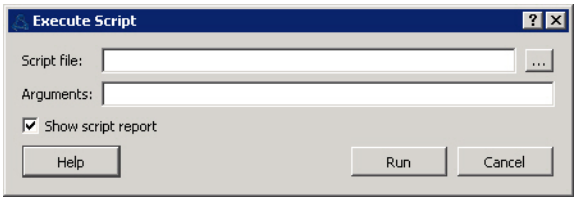


Figure 140 · Execute Script Dialog Box

Script file

Specify a script file. Browse to select a script file with a valid extension (*.tcl or *.dsf).

Arguments

Input your arguments for your script file (if necessary).

Export Script Dialog Box

The Export Script Files dialog box enables you to export Tcl script file, which is useful if you want to run Libero SoC in batch mode or run operations from the command line.

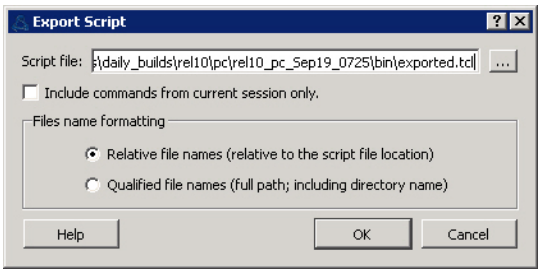


Figure 141 · Export Script Dialog Box

Script file

Specifies the location of the file you are about to save.

Include commands from current session only limits your commands to the current session. Deselect if you wish to include commands from other sessions.

File name formatting

Relative file names (relative to the script file location) truncates all the directories in the script with relative filenames. Select this option if you do not plan to move the script file.

Qualified file names (full path; including directory name) includes the full pathname for all the files and directories. Select this option if you want to move the file to a different directory.

File Menu - Libero SoC

Command	Icon	Shortcut	Sub-menu	Function
New			SmartDesign	Opens the appropriate New file dialog box and prompts you to enter a name and specify additional options (if necessary)
			HDL	
			SmartDesign Testbench	
			HDL Testbench	
			SDC (sdc)	
			Physical Design Constraint (PDC)	
			Simulation Script (do)	
Open				Opens the Open dialog box; enables you to select a file to open
Close <filename>				Closes the current file; the Project Manager remains open
Save <filename>		Ctrl + S		Saves the current file
Save <filename> As				Saves the current file as a different type (such as a TXT file)
Import Files				Opens the Import Files dialog

Command	Icon	Shortcut	Sub-menu	Function
				box; enables you to import project files into the Project Manager. Types include HDL Source Files, HDL Stimulus Files, Blocks, I/O Constraint (PDC) Files, Timing Constraint files, and more.
Link Files			Create Link	Opens the Create Link dialog box; browse to select the file you wish to link. Linked files are added to the Design Explorer in the Modules defined in multiple files list.
			Change All Links	Opens the Change All Links dialog box; enables you to update/change all the links for the files in your project at once.
			Unlink All: Copy Files Locally	Copies all linked files to your local project.
Export				Allows you to export I/O PDC, Floorplan PDC, Netlist, Back-Annotated Data, and Design Summary.
VHDL Library >			Add Library	Adds VHDL library to your Design Hiearchy
			Rename Library	Renames an existing VHDL library
			Remove Library	Removes an existing VHDL library from your project
Print Reports				Displays the Print dialog box (if available); allows you to print a report that contains project information and device details.

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 also delete files from the project by selecting **Delete from Project** from the right-click menu. You can delete files from the project and the disk by selecting **Delete from Disk and Project** 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 **-** ADB files (Microsemi Designer project files), *_ba.sdf, *_ba.v(hd), PRB (for Silicon Explorer), TCL (used to run designer), impl.prj_des (local project file relative to revision), designer.log (logfile)

Note: The Microsemi ADB file memory requirement is equivalent to 2x the size of the ADB file. If your computer does not have 2x the size of your ADB file's memory available, please make memory available on your hard drive.

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), *.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)

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

HDL Templates in Libero SoC

Use the templates in the Libero SoC Project Manager to create HDL.

To use the templates included with the Project Manager, from the View menu, choose Windows > HDL Templates. Find the template you want to use and double-click to add it to your HDL file.

Place the cursor where you want to add the template, browse the list of VHDL and Verilog templates, and double-click the template to add it to your design.

The VHDL and Verilog templates are useful if you want to modify your netlist but are unfamiliar with the language. The templates facilitate the writing of HDL files by inserting predefined language constructs.

To bring a non-standard template into Libero:

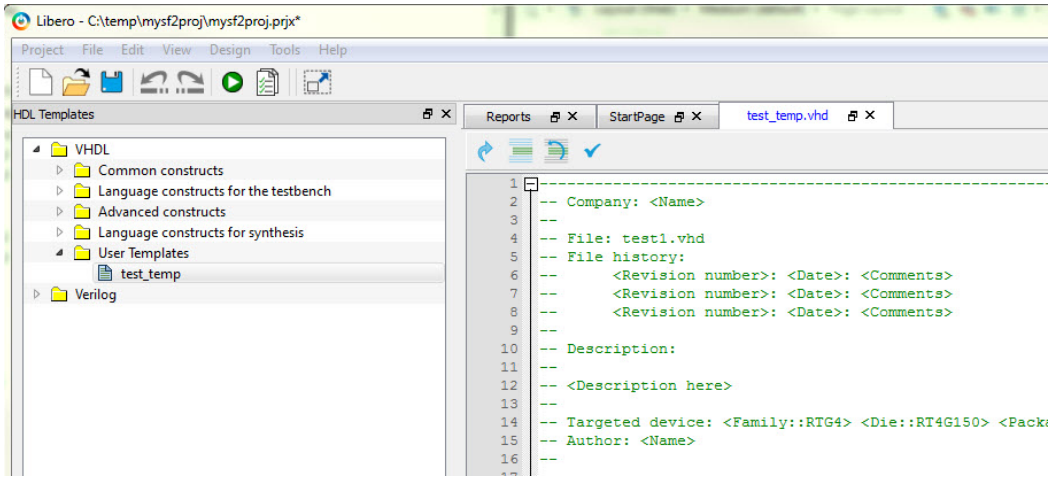
You can also save your own template files to reuse in other designs (for example, if you wanted to add the same header in all your files).

To import the file as a template:

1. Save the file to be used as a .vhd or .v file.
2. Click **File** -> **Import** -> **Others**.
3. Browse to the location of the file and choose **HDL Templates (*.vhdl *.vhd *.v *.sv)** for Files of type:.

Important Note: You must close the project, exit Libero, and reopen Libero and the project and to see your template under User Templates in Libero.

In the following example, test_temp.vhd has been saved as a VHDL User Template.



Help Menu - Libero SoC

The Help menu enables you to access the Libero SoC online help, reference manuals, check for updates, and view your license and version information.

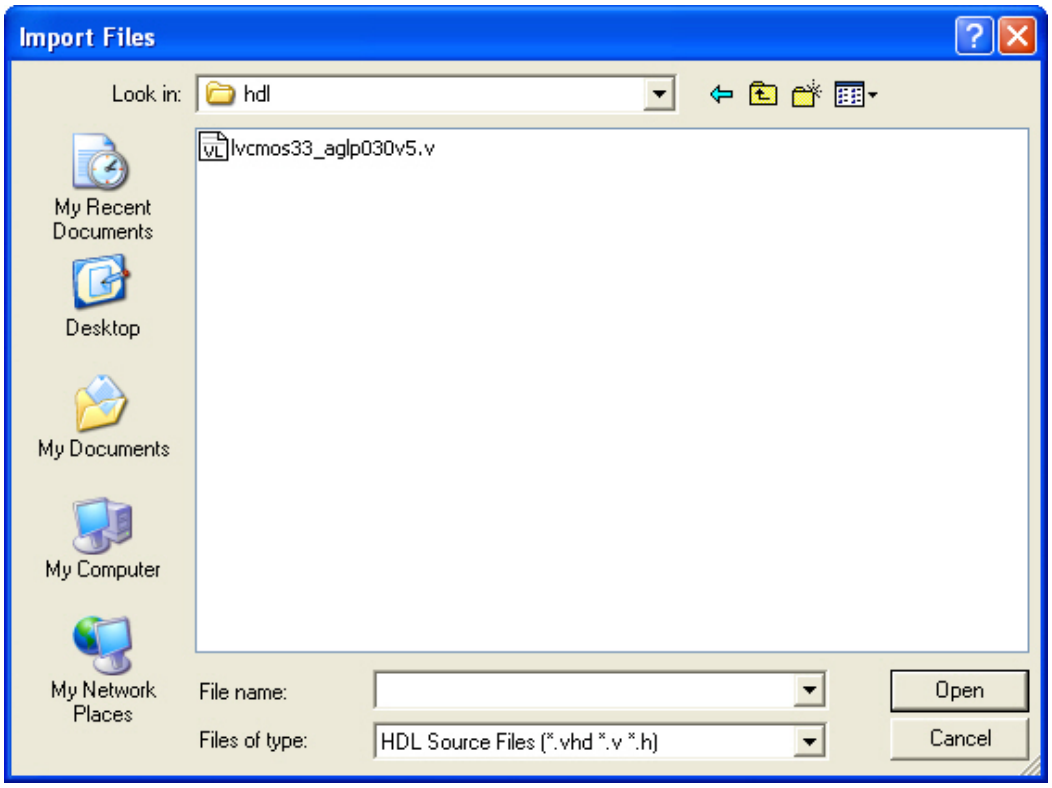
Command	Function
Help Topics	Opens the Libero Project Manager online help
Reference Manuals	Opens the Libero SoC Documentation Catalog
Microsemi Technical Support	Displays the Microsemi customer support web page in your default browser
Microsemi SoC Web Site	Displays the Microsemi SoC page in your default browser
Release Notes	Opens the Libero SoC Design Suite web page in your default browser
Check for Software Updates	Checks for updates to the Libero Project Manager software
License Details	Displays detailed license information for your version of Libero SoC
About Libero	Displays version and release numbers for Libero SoC

Import Files Dialog Box (Project Manager)

Use the Import Files dialog box to add new files to your project in the Libero SoC Project Manager.

You can import schematics, VHDL or Verilog source files, stimulus files, SDC, PDC, VCD, and SAIF files, cores, and even tool profiles (from other Libero SoC projects).

Browse to and select the file you wish to add and click **Import**, or click **Cancel** to return to the Project Manager.



Look in

Specifies your current directory. Browse to find your file if it is not listed here. If you are in the correct directory and your file is not listed here, select the **File of type** extension to match it.

File name

Type the file name, or browse to its location and select it.

File of type

Specify the file type displayed in the dialog box.

To access this dialog: from the **File** menu, choose **Import Files**.

License Details

To display information about your license:

From the **Help** menu, choose **License Details**. The software displays your complete license configuration, all Microsemi-installed software and versions, as well as your HostID and disk volume serial number.

Link Files

You can add or change links for individual files in your project, or change all the links in your files at once.

To add a link to an individual file, right-click the file in the Files list and choose **Create Link From File**. Navigate to the file you wish to link to your project and click **Create Link**. The Project Manager adds the file to your Files list; a small link icon indicates that the source file is not stored with the project.

If you have a single project file with a broken link , right-click the file and choose **Change Link**. This opens the Change Link dialog box and enables you to specify a new file location.

You can update all the links in your project at once. This is useful when you are linking to shared network folders that may have been renamed or moved. To change links for your entire project, from the **File** menu, choose **Change All Links**. This opens the Change All Links dialog box. Enter (or browse) your old and new paths to update the links for your project.

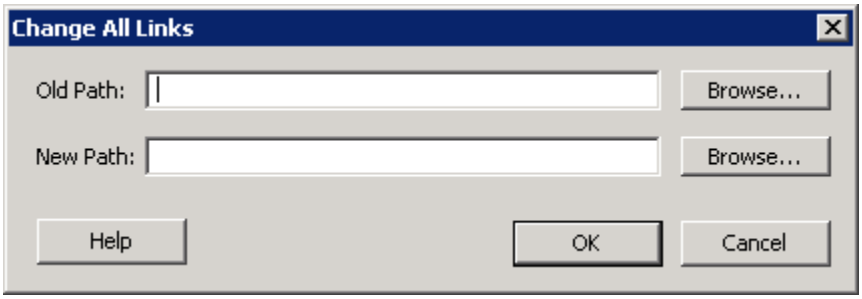


Figure 142 · Change All Links Dialog Box

To unlink a file, right-click the file in the Files tab and choose **Unlink: copy file locally**. This copies the file to the directory in your project folder that corresponds to the file type.

To unlink all files and copy them to your local project, from the **File** menu choose **Unlink All: Copy files locally**.

You can also change/remove links from the Design Explorer; to do so, right-click the file in the **Design Explorer > Modules defined in multiple files** and choose **Change Link**.

Log Window

Colors and Symbols

The log window displays Messages, Errors, Warnings, and Information. Messages are represented by symbols and color-coded. The default colors are:

Type	Color
Error	Red
Warning	Blue
Information	Black

The colors can be changed by using the Preferences dialog box.

Linked Messages

Error and warning messages that are dark blue and underlined are linked to online help to provide you with more details or helpful workarounds. Click them to open online help.

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
- Import HDL source files and/or design constraint files into your project

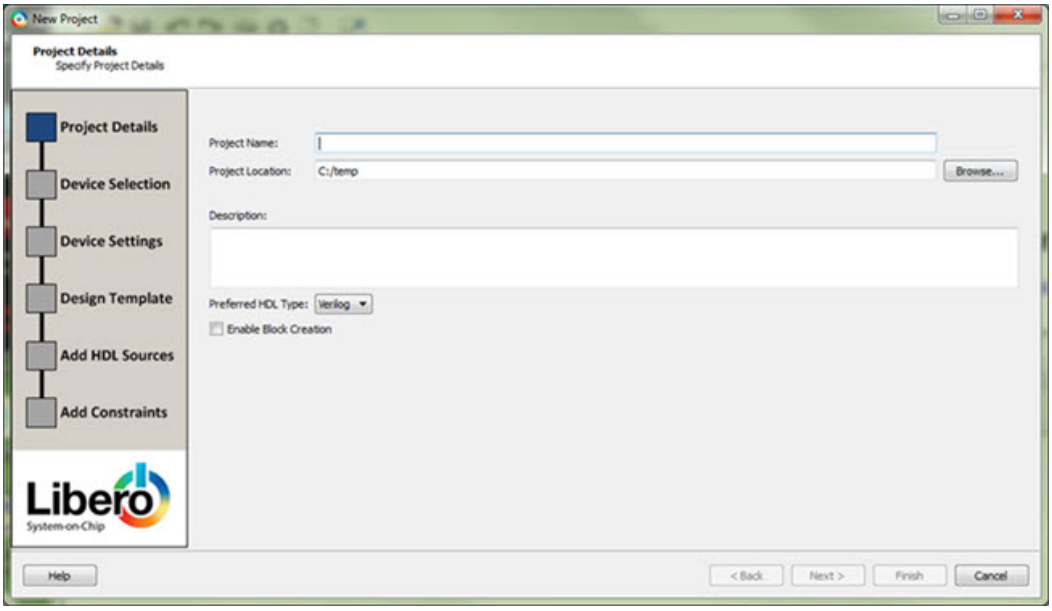


Figure 143 · Libero SoC New Project Creation Wizard

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.

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

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.

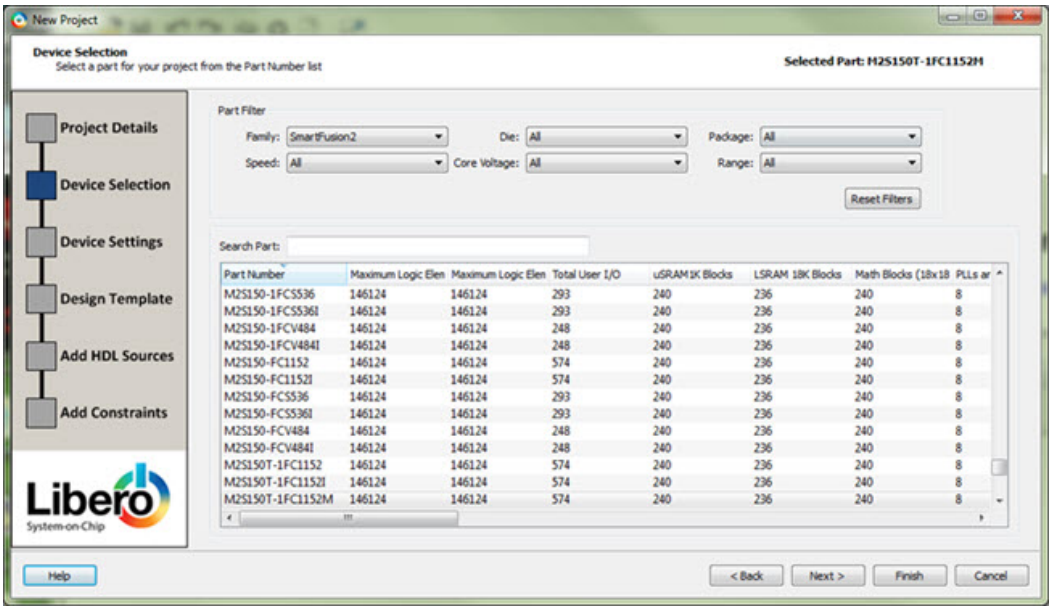


Figure 144 · New Project Creation Wizard - Device Selection Page

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)
- IND (Industrial)
- TGrade1 (Automotive)
- TGrade2 (Automotive)
- 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. Depending on your chosen device, the Constraint Flow Choice Dialog box may pop-up enabling you to select either Classic Constraint Flow or Enhanced Constraint Flow for the project.

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 condtions".

New Project Creation Wizard – Device Settings

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.

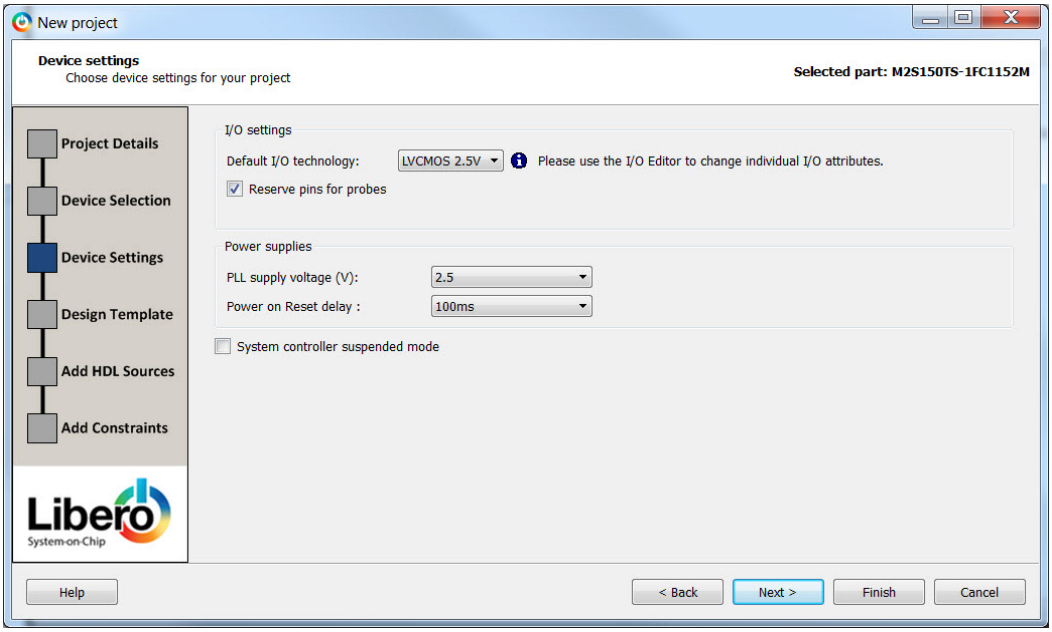


Figure 145 · New Project Creation Wizard – Device Settings Page

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 (SmartFusion2, IGLOO2 and RTG4) - Reserve your pins for probing if you intend to debug using SmartDebug.

Unused MSS I/O Resistor Pull (SmartFusion only)

Reserve Pins for SPI (RTG4 only) – Check this box to reserve pins for SPI functionality in Programming. This option is displayed in the Compile Report when the compile process completes.

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 µs, 1 ms, 10 ms, and 100 ms. Each selection represents the maximum ramp rate to apply to VDD and VPP.

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

System Controller 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, refer to the SmartFusion2 or IGLOO2 System Controller User's Guide for your device on the .

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

- **Next** to proceed to the next page.
  OR
- **Finish** to complete New Project Creation. Depending on your chosen device, the Constraint Flow Choice Dialog box may pop-up enabling you to select either Classic Constraint Flow or Enhanced Constraint Flow for the project.

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 condtions".

New Project Creation Wizard – Design Template

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.

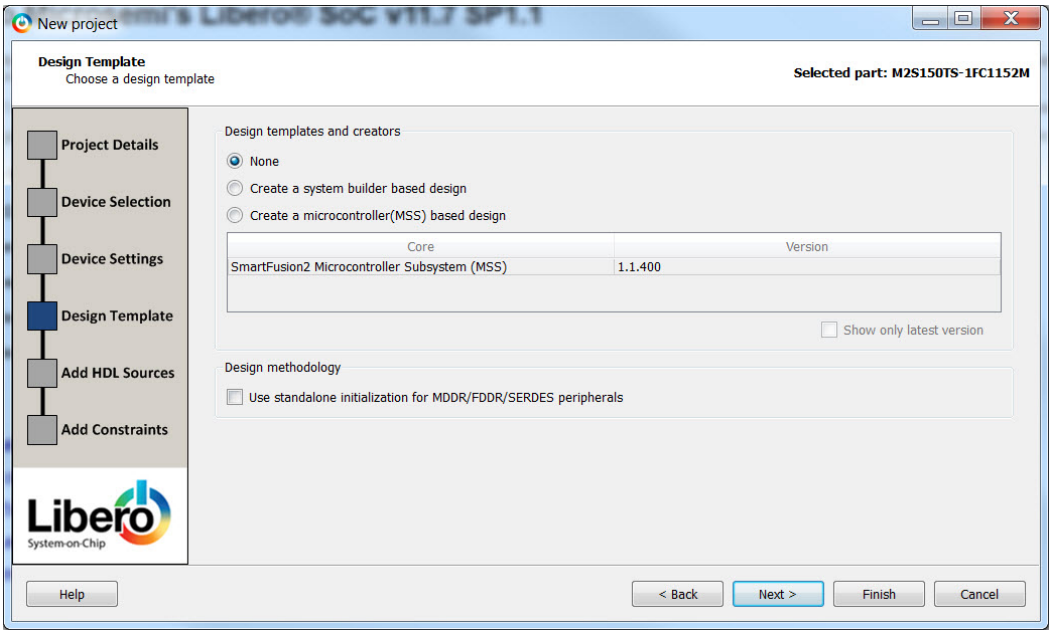


Figure 146 · New Project Creation Wizard – Design Template Page

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

Create a System Builder based design (SmartFusion2 and IGLOO2 only) – Use System Builder to generate your top-level design.

Create a Microcontroller (MSS) based design (SmartFusion2 and IGLOO2 only) – 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 (SmartFusion2 and IGLOO2 only) – 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.

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

- **Next** to proceed to the Add HDL Sources page
  OR
- **Finish** to complete New Project Creation. Depending on your chosen device, the Constraint Flow Choice Dialog box may pop-up enabling you to select either Classic Constraint Flow or Enhanced Constraint Flow for the project.

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.

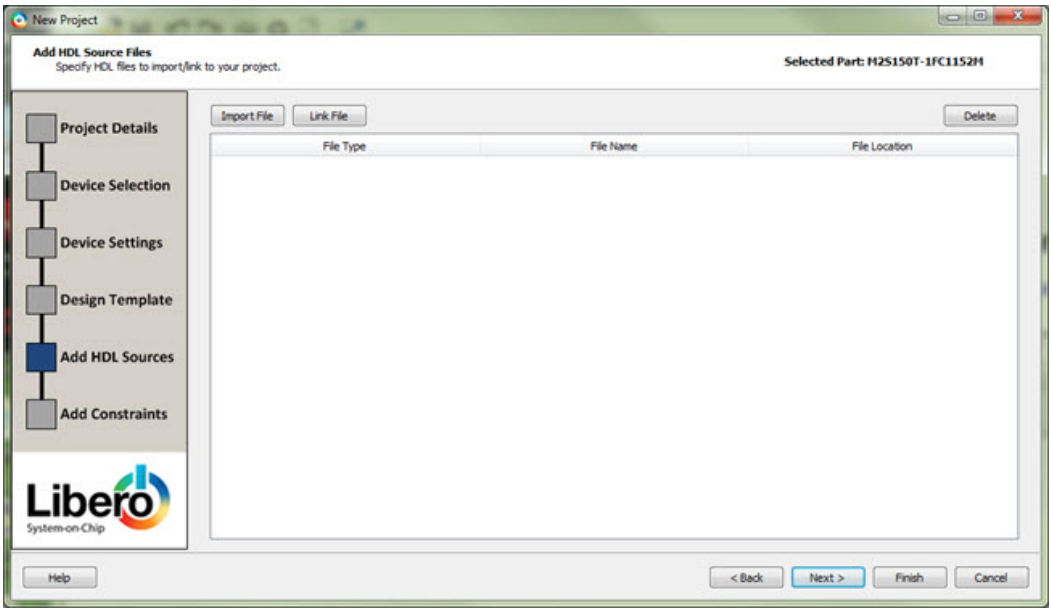


Figure 147 · New Project Creation Wizard - Add HDL Source Files Page

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. Depending on your chosen device, the Constraint Flow Choice Dialog box may pop-up enabling you to select either Classic Constraint Flow or Enhanced Constraint Flow for the project.

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.

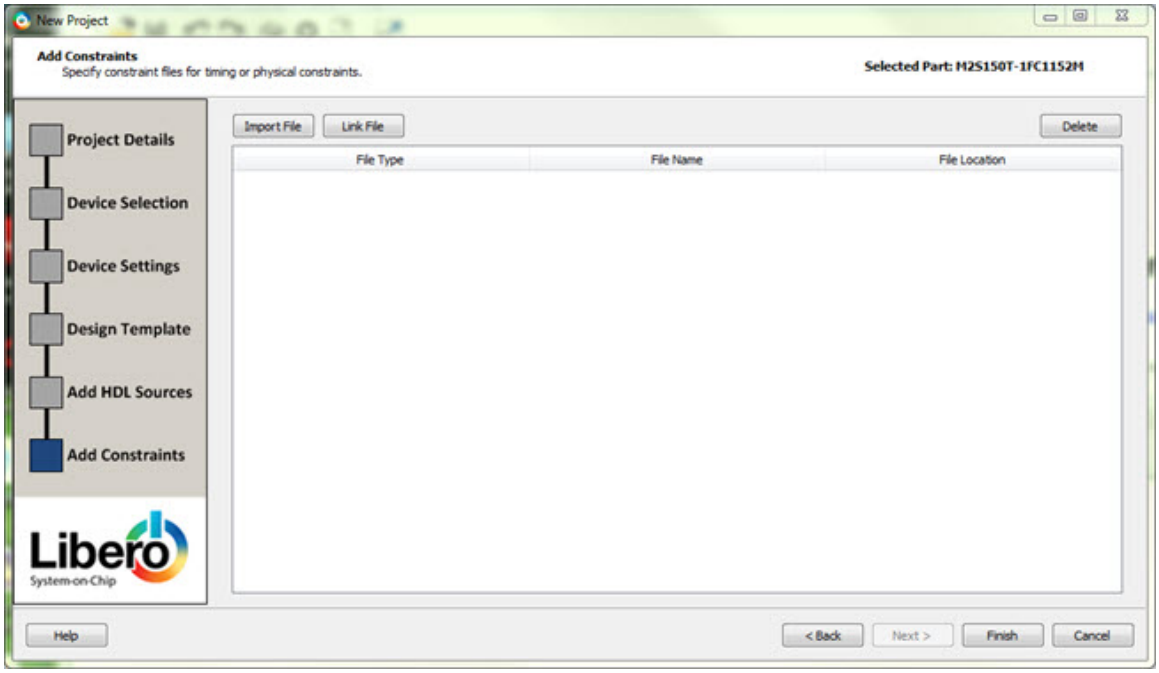


Figure 148 · New Project Creation Wizard – Add Constraints Page

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 Constriants** is completed, click on:

- **Finish** to complete New Project Creation. Depending on your chosen device, the Constraint Flow Choice Dialog box may pop-up enabling you to select either Classic Constraint Flow or Enhanced Constraint Flow for the project.

The **Reports** tab displays the result of the New Project creation.

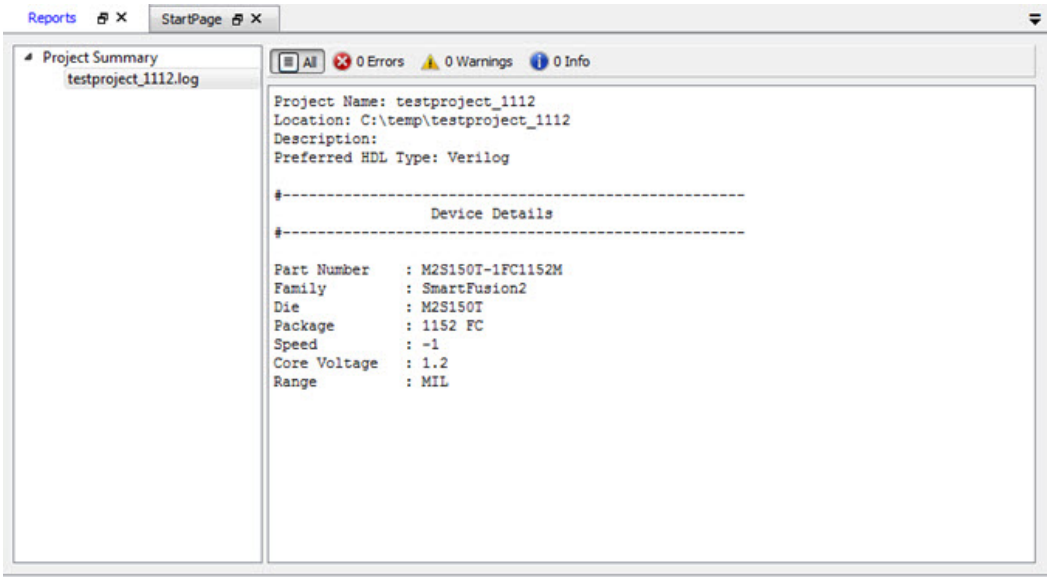


Figure 149 · Reports Tab

New File Dialog Box

The New File dialog box opens when you choose to create any of the following new files:

- SmartDesign
- SmartDesign Testbench – Use a SmartDesign to instantiate and connect stimulus cores or modules to drive your Root design.
- HDL
- HDL Testbench – Creates a new HDL testbench in your project. You can use a testbench to apply stimulus, analyze results or to compare the results of two different simulations.
- SDC (sdc) – SDC Timing Constraint File
- I/O Constraint (PDC) File – I/O Constraint File
- Synplify Netlist Constraint (*.fdc) File - Non-timing constraint file for Synplify
- Compile Netlist Constraint (*.ndc) File - Non-timing constraint file for Compile
- Floorplan Constraint (PDC) File – Floorplanning PDC Constraint File
- Simulation Script (do) – Libero creates the “run.do” script by default and passes it to ModelSim to run the simulation. The user can create a customized *.do script (different from the default “run.do” script) and use the Project Settings (Project > Project Settings > Simulation Options > Do file > User Definded DO file) to pass it to ModelSim to run the simulation.

To create a new file:

1. From the **File** menu, choose **New File > New**.
2. Select the file type from the submenu.
3. Enter a name for the new file.
4. Set any additional options specific to the file type.
5. Click **OK** and then click **Save** to create the new file.

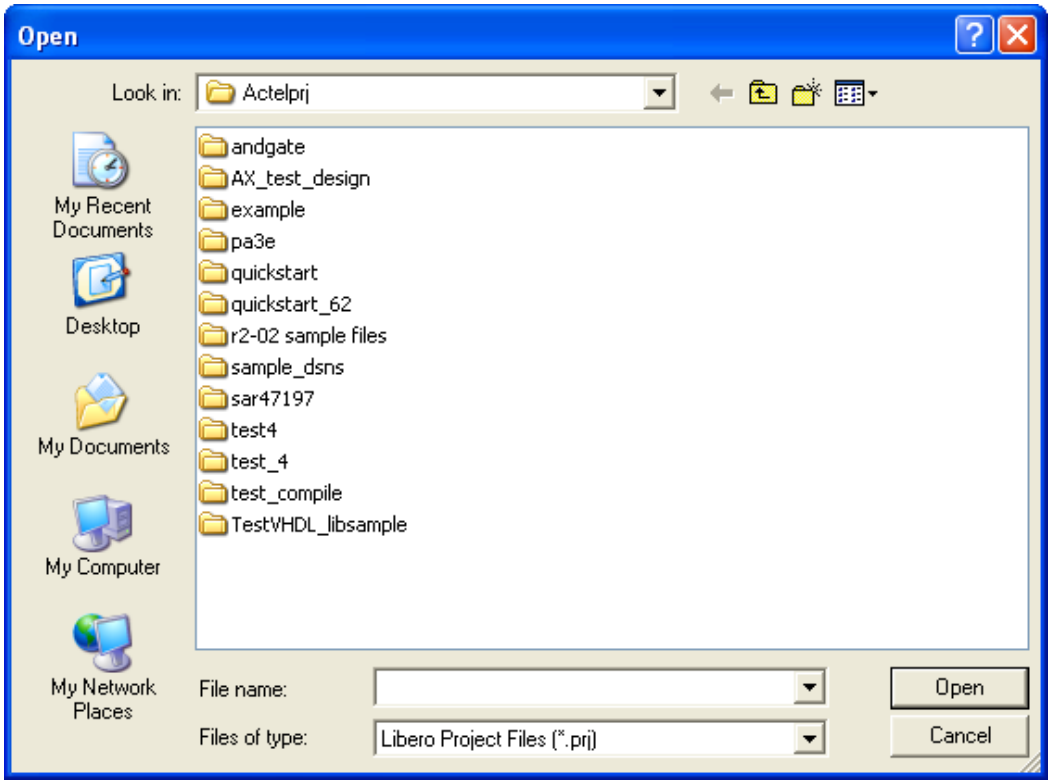
For SmartDesign or SmartDesign Testbench creation, Libero opens the SmartDesign canvas for you to create the design graphically. When the design or testbench is generated and saved, a Verilog or VHDL file is created to capture the design/testbench, and the file is added to the project location.

For HDL, HDL testbench, SDC, PDC, or Simulation *.do file creation, Libero opens the HDL Editor for you to create the file. When the file is saved, it is added to the Libero project.

File Type	File Extension	File Location	Opens/Editable with Libero Tool
HDL Design File	*.v (Verilog) *.vhd (VHDL)	<prj_folder>/hdl	HDL Editor
HDL Testbench	*.v (Verilog) *.vhd (VHDL)	<prj_folder>/stimulus	HDL Editor
SDC Timing Constraint File	*.sdc	<prj_folder>/constraints	HDL Editor Constraint Editor
I/O PDC Constraint	*.pdc	<prj_folder>/constraints/io	HDL Editor I/O Editor
Floorplan PDC Constraint	*.pdc	<prj_folder>/constraints/fp	HDL Editor Chip Planner
Simulation scipt (Do)	*.do	<prj_folder>/simulation	HDL Editor

Open Project Dialog Box

Use the Open Project dialog box to navigate to and open existing projects in the Project Manager. Browse to your project and click **Open**, or click **Cancel** to return to the Project Manager.



Look in

Specifies the directory that contains your project.

File name

Type the file name, or browse to its location and select it.

File of type

Specify the file type displayed in the dialog box.

To access this dialog: from the **Project** menu, select **Open Project**.

Opening your Libero SoC project

Libero SoC does not open your most recent project automatically. You can change your default startup preferences in the Startup tab.

To open a project in Libero SoC:

From the **File** menu, choose **Open Project** or **New Project**. If you create a new project the Project Manager opens the New Project Creation Wizard.

Tip: Recent saved projects are available from the Project menu. From the **Project** menu, choose **Recent Projects**, and then select the project to open.

You can open an existing project by double-clicking the *.prj file or dragging the *.prj file over the Libero SoC desktop icon.

See Also

`open_project`

Organize Constraint Files

The Organize Constraint Files dialog box enables you to set the constraint file and order in the Libero SoC.

Click the **Use list of files organized by User** radio button to add or remove Associated Constraint files.

To specify the constraint file order:

1. In the Design Flow window under Implement Design, right-click **Compile** and choose **Organize Input Files > Organize Constraint Files**. The Organize Constraint 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.
4. Click **OK**.The files appear in the Design Flow window under **Implement Design > Compile > Constraints** with a green check mark to indicate that they are being used in the project.

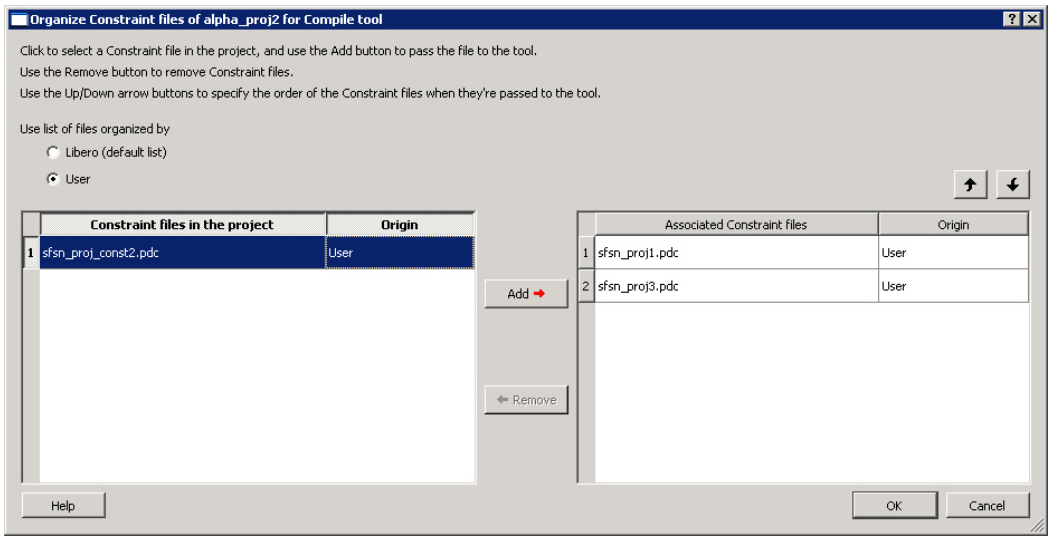


Figure 150 · Organize Constraint Files Dialog Box

Organize Simulation Files Dialog Box

The Organize Simulation files dialog box enables you to set the constraint file order in Libero SoC.

Click the **Use list of files organized by User** radio button to add or remove Associated Simulation files.

To specify the simulation file order:

1. In the Design Flow window under Implement Design > Verify Post Layout Implementation, right-click **Simulate** and choose **Organize Input Files > Organize Simulation Files**. The Organize Simulation 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.
4. Click **OK**.

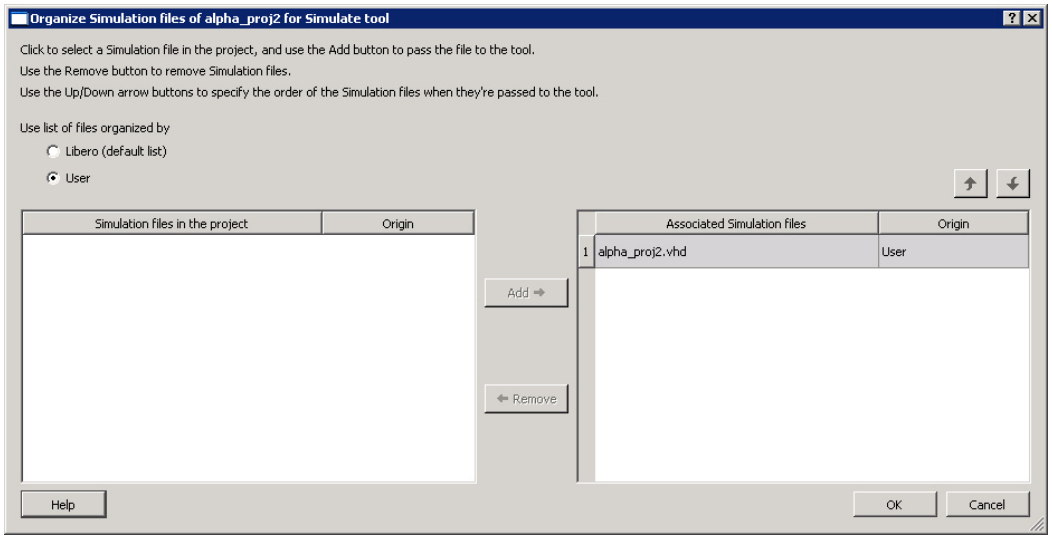


Figure 151 · Organize Simulation Files Dialog Box

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**.

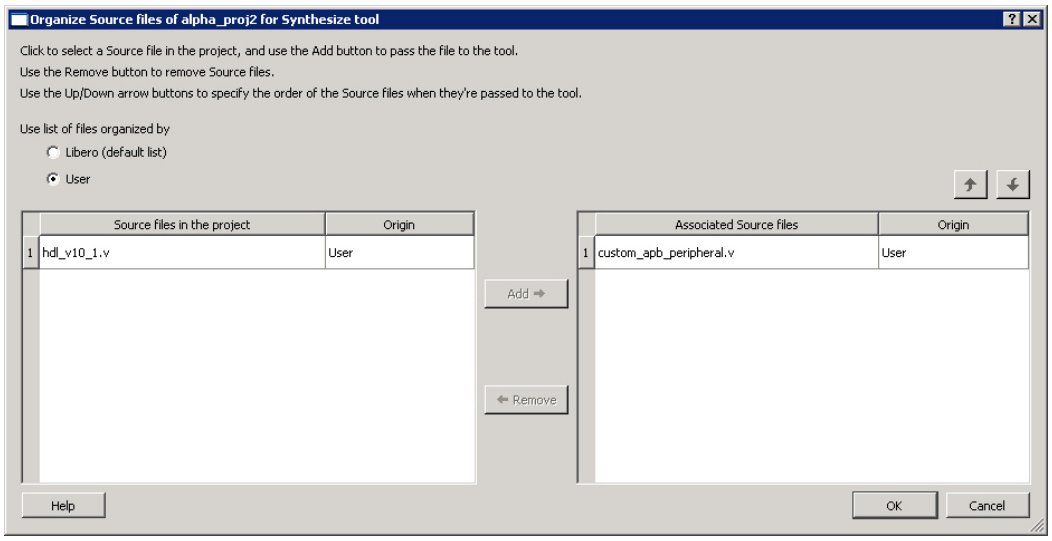


Figure 152 · Organize Source Files Dialog Box

Organize Stimulus Files Dialog Box

The Organize Stimulus files dialog box enables you to set the stimulus file order in Libero SoC.

Click the **Use list of files organized by User** radio button to add or remove Associated Stimulus files.

To specify the stimulus file order:

1. In the Design Flow window under Create Design > Verify Pre-Synthesized Design, right-click **Simulate** and choose **Organize Input Files > Organize Stimulus Files**. The Organize Stimulus 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.
4. Click **OK**.

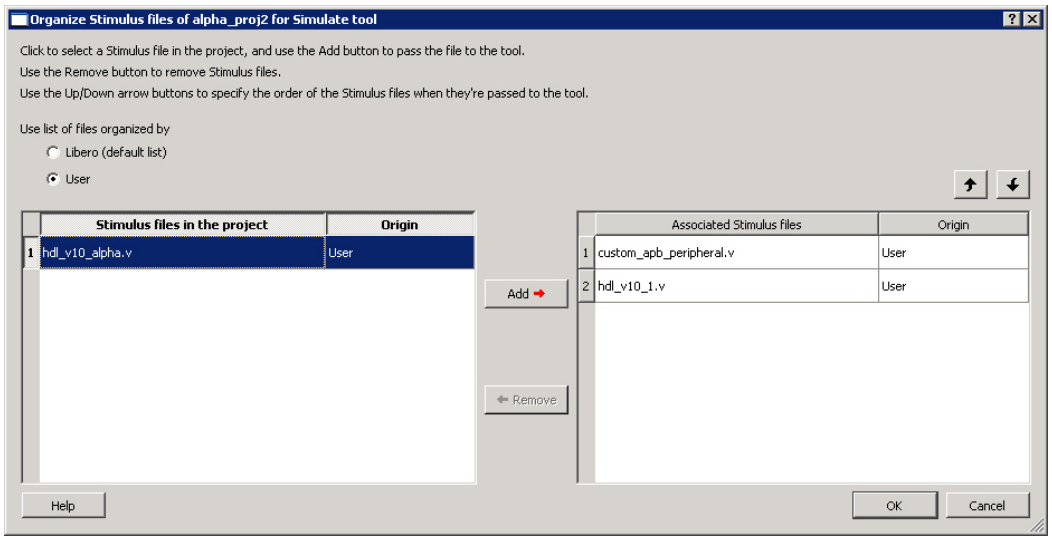


Figure 153 · Organize Stimulus Files Dialog Box

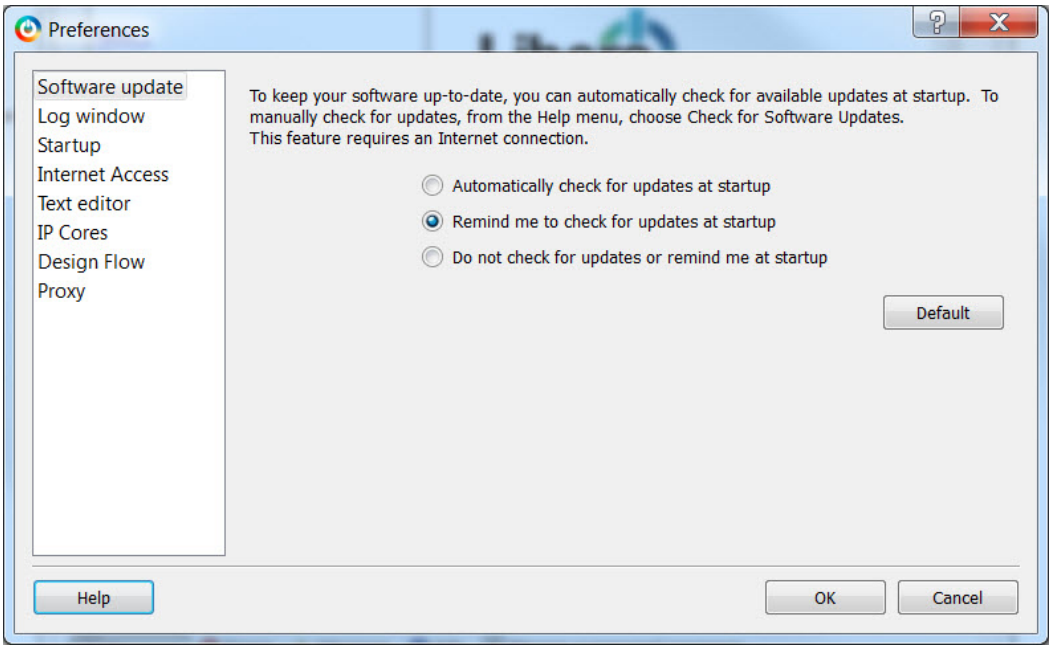
Physical Synthesis and the Libero SoC

If you want to run physical synthesis on your design (such as with PALACE) you must run it manually.

Automatic physical synthesis is not supported from within the Libero SoC.

Preferences

User Preferences Dialog Box



Use the Preferences dialog to customize the Libero SoC Project Manager.

To set your preferences:

1. From the **Project** menu, choose **Preferences**.
2. Specify your preferences.

   Software update

   Log window

   Startup (File association)

   Internet Access

   Text Editor

   IP Cores

   Design Flow

   Proxy

   PDF reader (Linux only)

   Web browser (Linux only)
3. Click **OK**.

Note: These preferences are stored on a per-user basis; they are not project specific.

Project Menu - Libero SoC

Command	Sub-menu	Icon	Function
New Project			Starts the New Project Wizard
Open Project			Opens the Open Project dialog box
Close <project name>			Closes the current active project; the Project Manager remains open
Save <project name>			Saves the current project
Save As <project name>			Saves the current project in a new directory; prompts you to enter a new project name
Archive Project			Creates an archive (*.zip file) of your existing project.
Project Settings			Opens the Project Settings dialog box, enables you to set your Device, HDL Type, Design Flow, Simulation and Simulation Library options.
Tool Profiles			Opens the Project Profile dialog box; enables you to specify locations for your third-party synthesis, stimulus, and simulation tools. Libero SoC includes tools for synthesis, stimulus, and simulation.
"Vault/Repositories Settings Dialog Box" on page **Error! Bookmark not defined.**			Opens the Vault/Repositories Settings dialog box; enables you to view/change the location of your vault and repositories.
Preferences			Opens the Preferences dialog box
Execute Script			Opens Execute Script dialog box; enables you to run Tcl script from the Project Manager
Export Script File			Opens the Export Script dialog box; enables you to export a Tcl script
Recent Projects			Opens list of recent projects.
Exit			Closes Libero SoC

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.

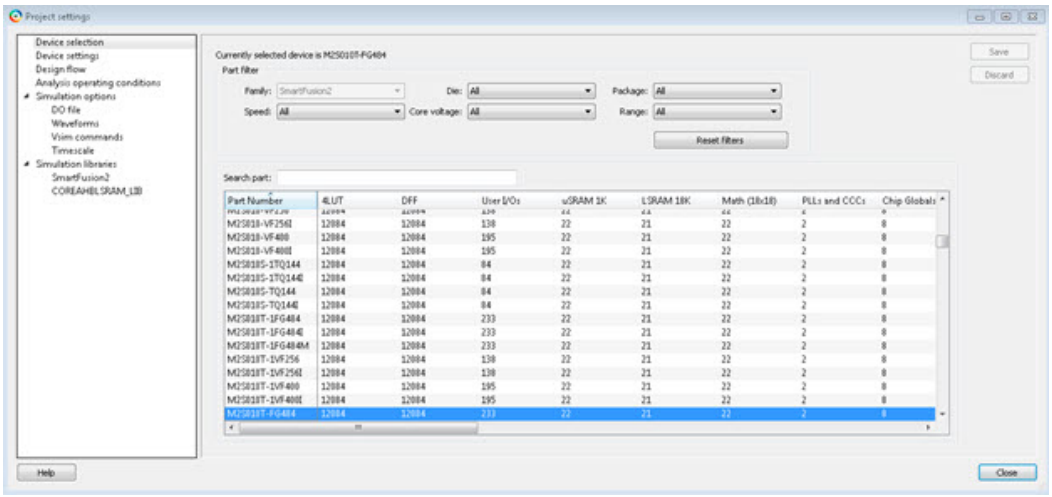


Figure 154 · Project Settings Dialog Box

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

Reserve Pins for Probes (SmartFusion2, IGLOO2, and RTG4 only) - Reserve your pins for probing if you intend to debug using SmartDebug.

Reserve Pins for SPI (RTG4 only) – Check this box to reserve pins for SPI functionality in Programming. This reserved SPI pin option is displayed in the Compile Report when the compile process completes.

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.**

Power-on Reset Delay (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

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

If your Verilog files contain System Verilog constructs, select the System Verilog option.

Note: Libero SoC supports the following Verilog and VHDL IEEE standards:

- 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)

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 < >

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.

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 (for SmartFusion2 and IGLOO2 only)

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.

Reports (Available only in Enhanced Constraint Flow)

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 if Errors are found in Physical Design Constraints (PDC) – (Enhanced Constraint Flow only) 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) – (Enhanced Constraint Flow only) Check this checkbox to abort Place and Route if the Timing Constraint SDC file contains errors.

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.

Note: For SmartFusion, IGLOO, ProAsic3, and Fusion projects, The Temperature and Voltage Range tables are disabled. To perform Timing/Power analysis with different operating conditions, invoke Designer and make the operating condition settings in the Project Settings page.

Simulation Options and Simulation Libraries

Sets your simulation options. See the Project Settings: Simulation Options topic for more information.

Project Settings: Simulation

To access this dialog box, from the **Project** menu choose **Project Settings** and click **Simulation options > DO File**.

Use the Simulation tab to set your simulation values in your project. 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 in this dialog box.

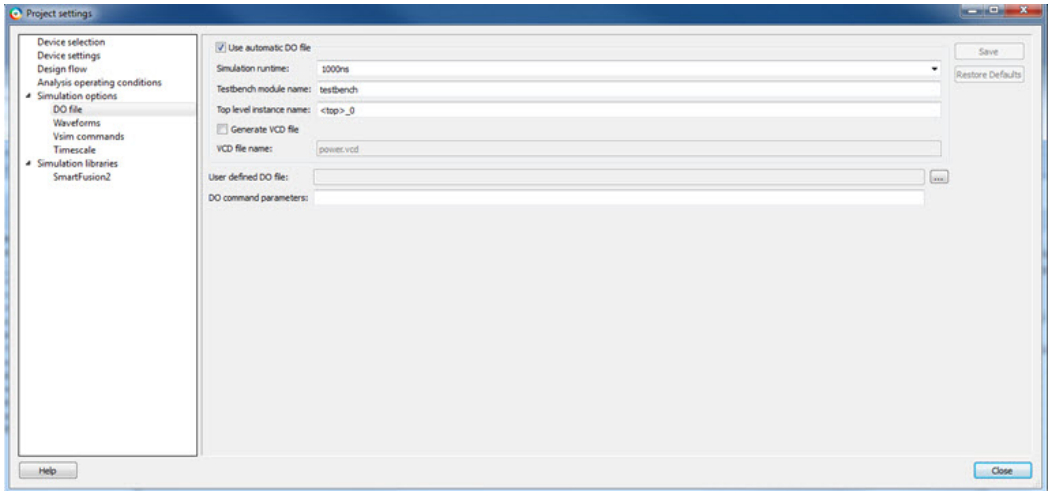


Figure 155 · Project Settings: - DO File

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 Model*Sim*.

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

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.

Additional options - Text entered in this field is added to the vsim command.

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

Use default library path - Sets the library path to default from your Libero SoC installation.

Library path - Enables you to change the mapping for your simulation library (both Verilog and VHDL). Type the pathname or click the Browse button to navigate to your library directory.

Project Sources

Project sources are any design files that make up your design. These can include schematics, HDL files, simulation files, testbenches, etc. Anything that describes your design or is needed to program the device is a project source.

Source files appear in the Project Flow window. The Design Hierarchy tab displays the structure of the design modules as they relate to each other, while the Files tab displays all the files that make up the project.

The design description for a project is contained within the following types of sources:

- Schematics
- HDL Files (VHDL or Verilog)
- SmartDesign components

One source file in the project is the top-level source for the design. The top-level source defines the inputs and outputs that will be mapped into the devices, and references the logic descriptions contained in lower-level sources. The referencing of another source is called an *instantiation*. Lower-level sources can also instantiate sources to build as many levels of logic as necessary to describe your design.

File Linking

The Project Manager enables you to link to files not managed in your Libero project. Linked files are useful if you want to preserve a file in an archive, or if more than one person is using a file and it is impractical to store it on your local machine. If you link to external files and rename your project, the Project Manager asks if you want to copy the external files into your project or continue using the link. Note that some files (such as schematics) cannot be linked.

Some project sources can be imported.

Sources for your project can include:

Source	File Extension
Schematic	*.1-9
Verilog Module	*.v
VHDL Entity	*.vhd
SmartDesign Component	*.vhd
Testbench	*.vhd
Stimulus	*.tim
Programming Files	*.afm; *.prb

See Also

Creating HDL Sources

Generating a Bitstream file

Generating Programming files

Reserved Microsemi Keywords

See the online help for a complete list of reserved Microsemi keywords.

Right-Click (Shortcut) Menu Options in Libero SoC Design Hierarchy

Right-click menu options vary depending on your design state.

The option in bold the right-click menu is the action performed when you double-click the tool. For example, if you expand Implement Design and right-click **Synthesize**, Run is bold, indicating that it is the default action when you double-click the tool in the Design Hierarchy.

- **Run** - Runs the current tool. If any predecessor tools are required to be in the PASSED state, then they will be run as well.
- **Clean and Run All**- Clean all predecessor tools (deletes Report and output files) and run up to this tool.
- **Clean** - Delete report and output files of this tool. Subsequent tools become OUT OF DATE.
- **Open Interactively** - Open the tool to set/change the tool options.
- **Update and Run** -- Available if a tool is in the OUT OF DATE state; it cleans all predecessor tools that are in the OUT OF DATE state and runs up to this tool.
- **Run Synthesize > Compile > Place and Route > Verify Timing > Generate Programming Data > Program Device** - Enables you to bypass the Fabric portion of the design flow.

  For example, in SmartFusion you can go directly from MSS configuration to Program Device by just using the .EFC file. For users who are not using any of the FPGA fabric, this is useful because you can skip the entire FPGA flow. In that instance you can select Run MSS Configurator > Program Device.
- **Organize Input Files** - Enables you to customize which project files are used by the tool.
- **Import Files** - Shortcut to import files that are relevant to that tool. For example, the relevant files for the Compile tool are PDC and SDC files, so the dialog is pre-filtered to only allow importing of those types
- **Edit Profile** - Shortcut to open the Tool Profiles dialog box.
- **View Report** - Opens the report of that tool in the Reports view.
- **Configure Options**- - Opens the Libero SoC tool options specific to that tool.

Save Project As Dialog Box

The Save Project As dialog box enables you to save your entire project with a new name and location.

Enter the name and location for your modified project and click OK to continue.

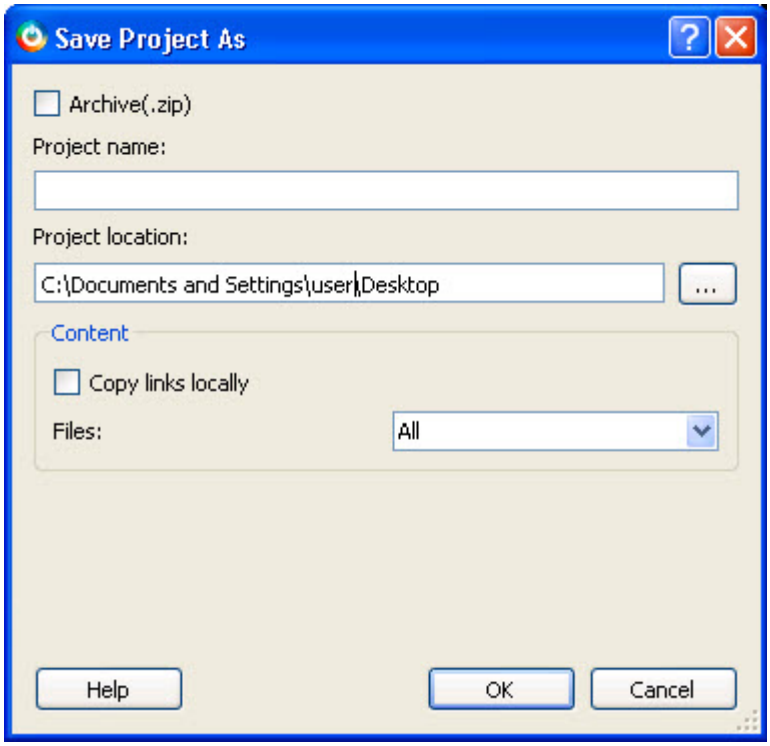


Figure 156 · Save Project As Dialog Box

Archive (*.zip) - Creates a ZIP file of your project and saves it at the specified location. This is useful if you want to create a quick zip of your project.

Project Name

Type the project name for your modified project.

Project Location

Accept the default location or **Browse** to the new location where you can save and store your project. All files for your project are saved in this directory.

Content

Copy Links locally - Select this checkbox to copy the links from your current project into your new project. If you do not select this checkbox, the links will not be copied and you must add them manually.

Files

- **All** - Includes all your project and source files; the state of the project is retained.
- **Project files only** - Copies only the project-related information required to retain the state of the project.
- **Source files only** - Copies all the source files into the specified location. This means the configuration of all the tools in the tool chain is retained but the states are not. Source files means constraint information and component information available in the component, hdl and smartgen directories.

Files are saved as shown in the table below.

Folder Name	Files		
	All	Project	Source
component	All	All Files	All Files

Folder Name	Files		
	All	Project	Source
	Files		
constraint	All Files	All Files	All Files
hdl	All Files	All Files	All Files
stimulus	All Files	All Files	All Files
smartgen	All Files	All Files	All Files
firmware	All Files	All Files	All Files
CoreConsole	All Files	All Files	All Files
SoftConsole/Keil/IAR	All Files	All Files	All Files
Phy_Synthesis	All Files	All Files	Not Copied
simulation	All Files	*.ini, *.bfm, *.do., *.vec	*.ini, *.bfm, *.do., *.vec
synthesis	All Files	*.edn, *.sdc, *.so, *.prj, *.srr, *.v, run_options.txt, synplify.log	*.prj files
Designer/impl1	All Files	All Files	*.ide_des files
Designer/<root>	All Files	All Files	Not Copied
tooldata	All Files	All Files	All Files

To access this dialog, from the **Project** menu, choose **Save Project As.**

Saving Files

Files and projects are saved when you close them.

To save an active file:

- From the **Project** menu, choose **Save** or **Save As**.
- Click the **Save** button in the toolbar.

Script Export Options Dialog Box

If you export a Tcl script in the Project Manager, the Script Export Options dialog box appears.

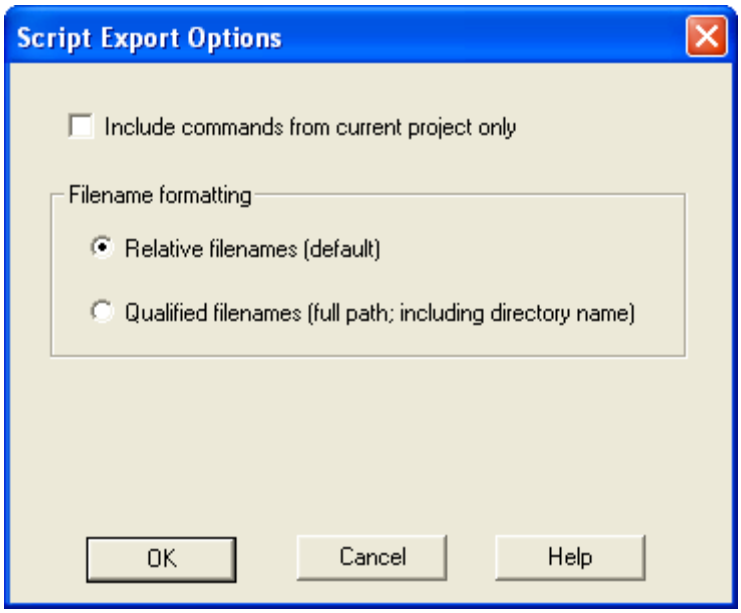


Figure 157 · Script Export Options Dialog Box

Include commands from current project only - Select this option if you want to include all the commands from your current project.

Filename Formatting - Choose **Relative filenames** if you do not intend to move the Tcl script from the saved location, or **Qualified filenames** if you plan to move the Tcl script on your machine.

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.

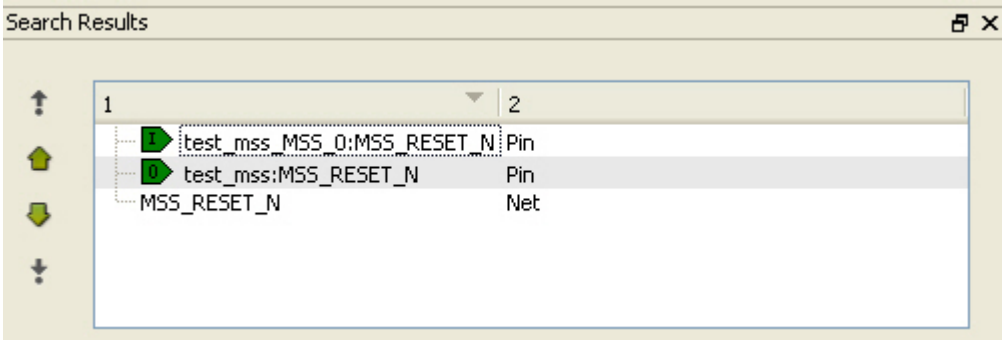


Figure 158 · Search Results

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 a Workspace Dialog Box

This dialog box enables you to choose which processor you want to open when you have two or more processors in your design.

It is only available if you have two or more processors and double-click **Develop Firmware > Write Application Code**.

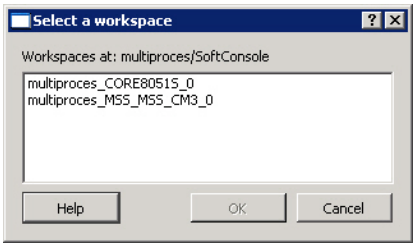


Figure 159 · Select a Workspace Dialog Box

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**.

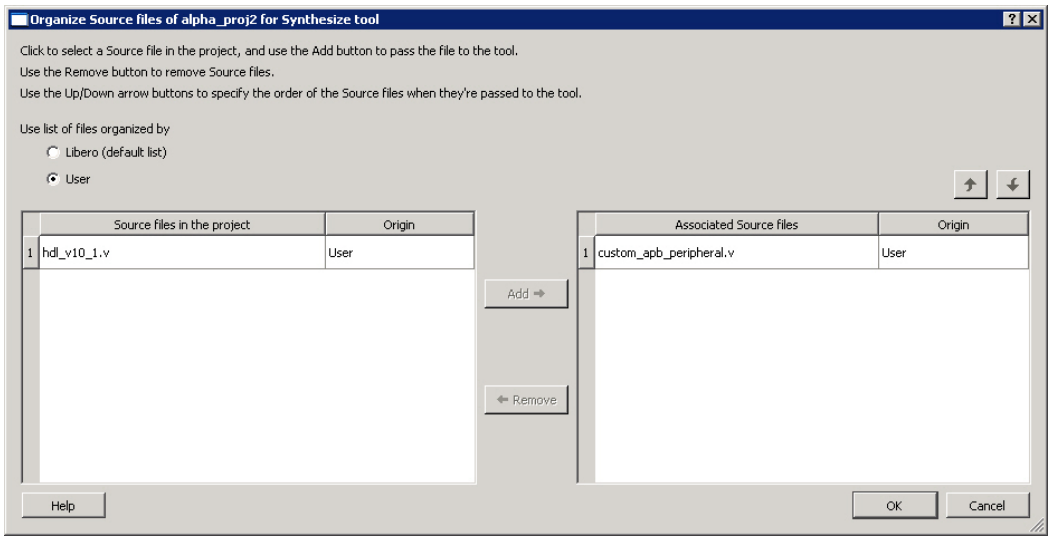


Figure 160 · Organize Source Files Dialog Box

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.

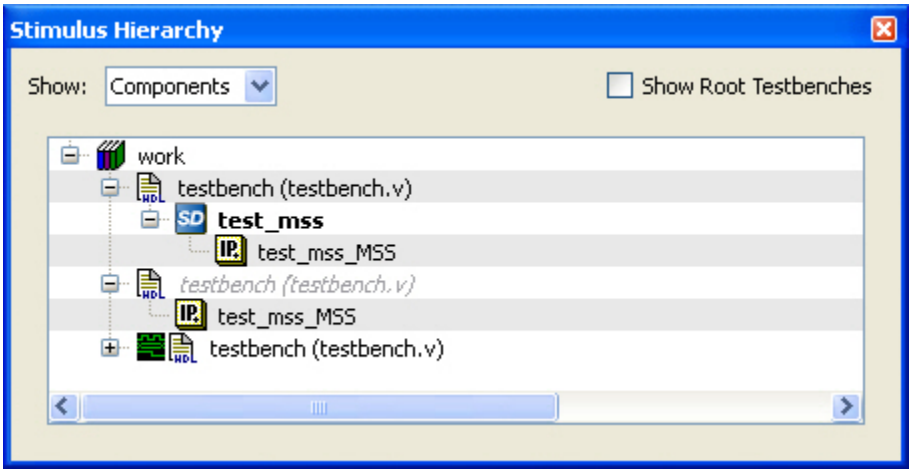


Figure 161 · Stimulus Hierarchy Dialog Box

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 10 · Design Hierarchy Icons

Icon	Description
	SmartDesign component
	SmartDesign component with HDL netlist not generated
	SmartDesign testbench
	SmartDesign testbench with HDL netlist not generated
	IP core was instantiated into SmartDesign but the HDL netlist has not been generated
	HDL netlist

Text Editor

You can use the Libero IDE HDL text editor or another text editor.

To set your text editor preferences:

1. From the **Project** menu, choose **Preferences**.
2. Click **Text editor**.
2. Set your options and click **OK**.

Libero SoC text editor options:

- **Use Libero text editor**: Select to use the Libero HDL text editor.
- **Enable block folding**: This option lets you fold (hide) portions of your text.
- **Enable line numbers**: This option enables you to see line numbers in the text editor.
- **Open programming/debugging files as read-only:**Select to specify read-only permission to .stp and .prb files.

User defined text editor

- **User defined text editor**: Deselect Use Libero text editor to activate this area. Enter the location of the the EXE for your alternative text editor.
- **Additional parameters:** Use to specify other settings to pass to the text editor. Typically, it is not necessary to modify this field.

User Template Location - Sets the path where your user templates are exported.

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.

To set or change your tool profile:

1. From the **Project** menu, choose **Tool Profiles**. Select the type of tool you wish to add.

- **To add a tool**: Select the tool type and click the **Add** button . Fill out the tool profile and click **OK**.
- **To change a tool profile**: After selecting the tool, click the **Edit** button to select another tool, change the tool name, or change the tool location.
- **To remove a tool from the project**:After selecting a tool, click the **Remove** button.

2. When you are done, click **OK**.

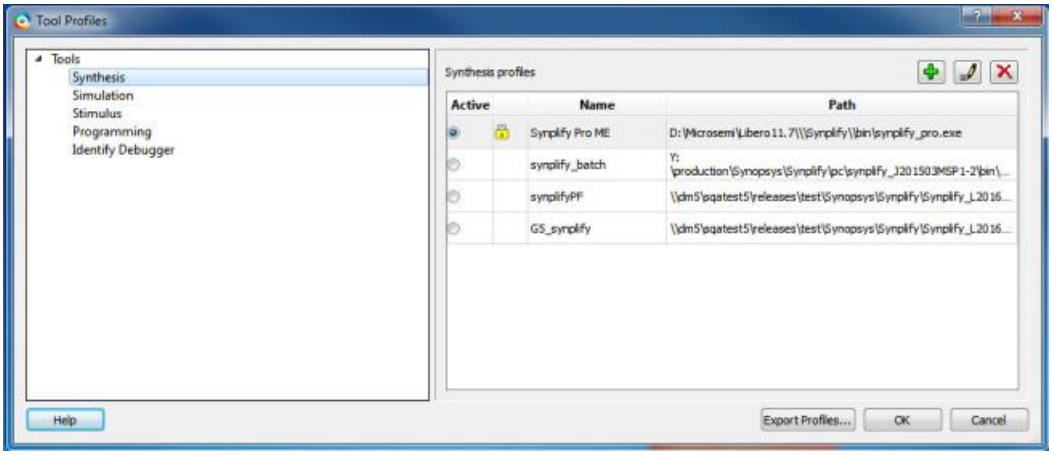


Figure 162 · Libero SoC Tool Profiles Dialog Box

The tool profile with the padlock icon indicates that it is a pre-defined tool profile (the default tool that comes with the Libero SoC Installation.)

To export the tool profile and save it for future use, click the **Export Tool Profiles** dialog box and save the tool profile file as a tool profile *.ini file. The tool profile *.ini file can be imported into a Libero SoC project (**File > Import > Others**) and select Tool Profiles (*.ini) in the File Type pull-down list.

Tools Menu - Libero SoC

Command	Function
Waveformer Lite	Opens the Waverformer Lite stimulus tool, if configured.

Vault/Repositories Settings Dialog Box

The repositories are the web locations where Microsemi IP cores and Configuration cores are uploaded.

The vault is the physical disk location where cores are stored when they are downloaded from the repositories.

Use the Repository dialog box to add, remove, or reset your repositories to default settings.

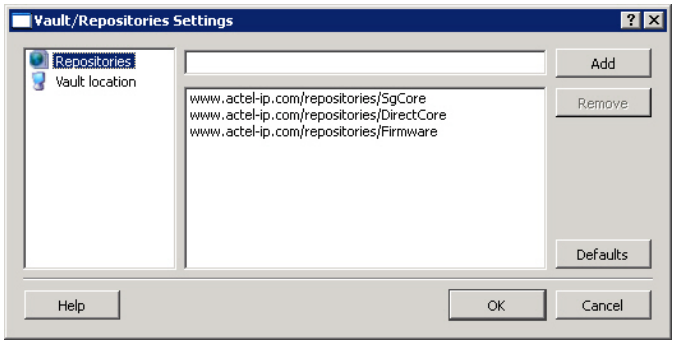


Figure 163 · Repositories Settings Dialog Box

Use the Vault location dialog box to specify a new location for your local vault.

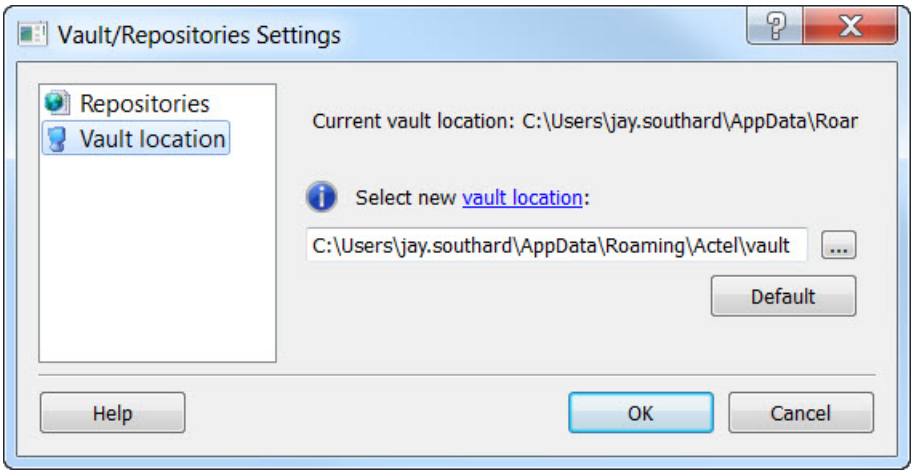


Figure 164 · Vault location Settings Dialog Box

The Vault location is common to all Microsemi software. Changing your vault location here updates the vault location for all the Microsemi tools you use on this machine.

Videos - Libero SoC

There are short videos available that explain a variety of elements in Libero SoC. The maximum video length is 60 seconds, unless otherwise noted. See the SoC website for a complete list of the latest video content, as well as tutorials and online training.

Video Links

SoC Work Area Description - The SoC Work Window displays the HDL Editor, Report view, and SmartDesign Canvas.

Design Hierarchy Tab and Files Tab - Introduces the Design Hierarchy and Files tabs in the SoC GUI.

Design Flow Tab and Catalog - Introduces the Design Flow tab and the Catalog.

AutoConnect in SmartDesign - Demonstrates the Autoconnect feature in SmartDesign.

Connection Mode in SmartDesign - Demonstrates the manual Connection mode feature in SmartDesign.

View Design Datasheet/Report

The Design Datasheet/Report lists all the reports available for your design.

Reports are added automatically when 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 a report is not listed you may have to open it manually. For example, you must double-click **Export IBIS Model** to display the IBIS Model report in the Design Datasheet.

You can view the following reports from here:

- Analyze Timing - Lists the following delay reports:
    - Timing violations report - Flat Slack report provides information about constraint violations.
    - Timing Report - Displays the timing information organized by clock domain.
- Compile - Summarizes your compile parameters and lists any related warnings, errors, PDC commands, device utilization and net information.
- Synthesize - Lists the following synthesis reports:
    - synplify.log - Outputs the Synplify log file output; identical to log file content in Synplify Pro AE if you run synthesis manually.
    - datasheet.srr - Lists the Pin Description, DC Electrical Characteristics, and AC electrical characteristics.
    - run_options.txt - Lists all the run options organized by category: project files; implementation; device options; compile/mapping options; mapper options.
- Export Pin Report - Lists the pins in your device sorted by I/O signal name and by package number.
- Place-and-Route - Lists the following reports:
    - Place-and-Route - Lists Compile and netlist information.
    - Global Net and Global Usage- Contains information about the net(s) that are assigned or routed using Global or LocalClock resources
    - I/O bank reports - Provides information on the I/O functionality, I/O technologies, I/O banks and I/O voltages.
- Export IBIS Model - Exports the IBIS model report, which provides a standard file format for recording parameters like driver output impedance, rise/fall time, and input loading, which may then be used by any software application.
- Programming - Lists the programming information for your design.

View Menu - Libero SoC

Command	Sub-menu	Shortcut	Function
Windows >	Catalog		Shows/hides the Catalog
	Cores		Shows/hides the list of cores used in your design
	Design Flow		Shows/hides the Design Flow window

Command	Sub-menu	Shortcut	Function
	Design Hierarchy		Shows/hides the Design Hierarchy
	Files		Shows/hides the Files window
	HDL Templates		Shows/hides the HDL Templates window
	Log		Shows/hides the Log window
	Message		Shows/hides the Message window
	Search Results		Shows/hides Search results
	Stimulus Hierarchy		Shows/hides the Stimulus Hierarchy
Start Page			Displays the Welcome to Libero SoC page; the page includes links to help and other pages that may be helpful for new users.
Toolbars			Shows the Find Bar or Project Toolbar
Refresh Design Hierarchy		F5	Updates the Hierarchy tab. Useful if you add files to the project and the software does not show them in the Hierarchy.
Maximize Work Area		CTRL+W	Hides the Catalog, Log Window, and Design Explorer windows (if open) and expands the selected tab in the Project Flow or SmartDesign work area.
Reset Layout			Returns the Libero SoC window layout to default.

VHDL Library - Add, Remove, or Rename

Libero SoC enables you to manage your VHDL libraries from within the Project Manager.

From the File menu, select **VHDL Library** and **Add**, **Rename**, or **Remove** to update your library.

When you add a library it appears in your Hierarchy.

Application Notes

Application notes are available for all Microsemi SoC devices. A full list of application notes is available at the Microsemi SoC website.

Application notes are organized by product or type. For example, you can view a full list of application notes for SmartFusion, or you can view a list of application notes on Design Entry that includes documents for all available families.

The following is a short list of popular application notes covering a range of applications and devices.

- AC333: Connecting User Logic to the SmartFusion Microcontroller Subsystem App Note (design files required - 23 MB) - Describes how to create AHB Lite or APB3 wrapper on custom logic and how to connect it to the MSS System via the Fabric Interface Controller.
- AC225 Programming Antifuse Devices App Note - Provides an overview of the programming options available for the antifuse families.
- AC362: SmartFusion cSoC: Programming FPGA Fabric and eNVM Using In-Application Programming Interface App Note (design files required - 50 MB)
- AC335: Building an APB3 Core for SmartFusion cSoC FPGAs App Note (design files required - 13 MB) - Describes how to create an APB3 wrapper interface for your logic or IP and connect it to the MSS via the Fabric Interface Controller.
- AC265: Clock Generation and Distribution Design Example App Note (design files required - 1 MB) - Demonstrates the use of the IGLOO and ProASIC3 clock conditioning circuits and phase-locked loops (PLLs) to generate multiple clock signals with different phases and frequencies.

Tutorials and Training Modules

Software tutorials, webcasts and online training modules are available on the Microsemi website. See the website for a full list.

The following list is an example of the tutorials available. Training modules may require you to register to enter the Microsemi Training Portal. Registration is free.

Product Support

Microsemi SoC Products Group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, electronic mail, and worldwide sales offices. This appendix contains information about contacting Microsemi SoC Products Group and using these support services.

Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From North America, call **800.262.1060**
From the rest of the world, call **650.318.4460**
Fax, from anywhere in the world **650. 318.8044**

Customer Technical Support Center

Microsemi SoC Products Group staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions about Microsemi SoC Products. The Customer Technical Support Center spends a great deal of time creating application notes, answers to common design cycle questions, documentation of known issues and various FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

Technical Support

For Microsemi SoC Products Support, visit http://www.microsemi.com/products/fpga-soc/design-support/fpga-soc-support.

Website

You can browse a variety of technical and non-technical information on the Microsemi SoC Products Group home page, at http://www.microsemi.com/soc/.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center. The Technical Support Center can be contacted by email or through the Microsemi SoC Products Group website.

Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is soc_tech@microsemi.com.

My Cases

Microsemi SoC Products Group customers may submit and track technical cases online by going to My Cases.

Outside the U.S.

Customers needing assistance outside the US time zones can either contact technical support via email (soc_tech@microsemi.com) or contact a local sales office. Visit About Us for sales office listings and corporate contacts.

ITAR Technical Support

For technical support on RH and RT FPGAs that are regulated by International Traffic in Arms Regulations (ITAR), contact us via soc_tech@microsemi.com. Alternatively, within My Cases, select **Yes** in the ITAR drop-down list. For a complete list of ITAR-regulated Microsemi FPGAs, visit the ITAR web page.