
Libero SoC for Enhanced Constraint Flow v11.7 SP1

User's Guide

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



Table of Contents

Libero SoC Introduction	7
Welcome to Microsemi's Libero® SoC v11.7 SP1	7
What's New in Libero SoC v11.7 SP1	7
Libero SoC Design Flow (Enhanced Constraint Flow) - SmartFusion2, IGLOO2, RTG4.....	7
Enhanced Constraint Flow and Design Sources.....	11
Microsemi License Utility	12
Supported Families (Enhanced Constraint Flow)	13
File Types in Libero SoC.....	14
Software Tools - Libero SoC	14
Frequently Asked Questions - Libero SoC	15
Firmware Cores Frequently Asked Questions.....	16
Software IDE Integration.....	16
Libero Design Flow Using Enhanced Constraint Flow for SmartFusion2, IGLOO2, and RTG4.....	16
Starting the Libero GUI	17
Design Report	18
Using the Libero SoC New Project Wizard to Start an Enhanced Constraint Flow Project.	19
Create and Verify Design	25
System Builder.....	25
MSS - SmartFusion2 only.....	25
SmartDesign.....	27
Designing with HDL.....	75
Simulation and Verification	78
Libero SoC Constraint Management (Enhanced Constraint Flow)	82
Invocation of Constraint Manager From the Design Flow Window	83
Libero SoC Design Flow.....	83
Synthesis Constraints	84
Place and Route Constraints.....	85
Timing Verifications Constraints	85
Constraint Manager Components.....	86
Constraint File and Tool Association.....	86
User Actions on Constraint File	87
Derive Constraints in Timing Tab.....	87
Create New Constraints	88
Constraint File Order	89
Import a Constraint File	89
Link a Constraint File	89
Check a Constraint File.....	89

Edit a Constraint File	92
Timing Constraints Editor.....	94
Implement Design in Enhanced Constraint Flow for SmartFusion2, IGLOO2, and RTG4	151
Synthesize (Enhanced Constraint Flow)	151
Verify Post-Synthesis Implementation - Simulate	156
Compile Netlist (Enhanced Constraint Flow)	157
Configure Flash*Freeze	158
Resource Usage (SmartFusion2, IGLOO2, RTG4).....	158
Enhanced Constraint Flow in Implementation.....	159
Place and Route - SmartFusion2, IGLOO2, RTG4.....	166
Multiple Pass Layout Configuration (SmartFusion2, IGLOO2, RTG4).....	167
Export Back Annotated Files	169
Verify Post Layout Implementation.....	170
Generate Back Annotated Files - SmartFusion2, IGLOO2, and RTG4 Only	170
Simulate - Opens ModelSim AE.....	170
Verify Timing	171
SmartTime (Enhanced Constraint Flow).....	175
Verify Power.....	176
IO Advisor (SmartFusion2, IGLOO2, and RTG4).....	177
Program and Debug.....	184
SmartFusion2 and IGLOO2 Programming Tutorials	184
Generate FPGA Array Data	197
Update uPROM Memory Content - RTG4 Only	198
Update eNVM Memory Content (SmartFusion2 and IGLOO2).....	200
Configure Hardware.....	202
Configure Security and Programming Options	207
Program Design	219
Debug Design.....	234
Handoff Design for SmartFusion2, IGLOO2, and RTG4.....	299
Export Bitstream - SmartFusion2, IGLOO2	299
Export Bitstream - RTG4.....	299
Export FlashPro Express Job - SmartFusion2, IGLOO2, RTG4.....	300
Export Job Manager Data - SmartFusion2, IGLOO2.....	303
Export Pin Report.....	304
Export BSDL File	305
Export IBIS Model.....	305
Handoff Design for Firmware Development	306
View/Configure Firmware Cores.....	306
Export Firmware – SmartFusion2.....	308
Product Support.....	409

Customer Service	409
Customer Technical Support Center	409
Technical Support.....	409
Website.....	409
Contacting the Customer Technical Support Center	409
ITAR Technical Support.....	410



Libero SoC Introduction

Welcome to Microsemi's Libero[®] SoC v11.7 SP1

Libero[®] System-on-Chip (SoC) is a comprehensive and powerful FPGA design and development software available from Microsemi, providing start-to-finish design flow guidance and support for novice and experienced users alike. Libero SoC combines Microsemi SoC Products Group tools with such EDA powerhouses as Synplify Pro[®] and ModelSim[®].

Libero SoC v11.7 SP1 includes enhancements to RTG4 timing data, and adds RTG4 Dynamic CCC configuration support. In addition, the Lock Bit Configuration tool is introduced, enabling certain design security use models.

Use Libero SoC v11.7 SP1 for designing with Microsemi's RTG4 Rad-Tolerant FPGAs, SmartFusion2 and SmartFusion SoC FPGAs, and IGLOO2, IGLOO, ProASIC3, and Fusion FPGA families.

What's New in Libero SoC v11.7 SP1

HDL Language Mode Unification – HDL language mode selections have been centralized in Libero SoC Project Settings.

Configure Register Lock Bits Tool – Allows users to protect the MSS/HPMS, FDDR, and SERDES configurations for SmartFusion2 and IGLOO2 devices.

Synthesis Tcl Flow – Users can now specify advanced Synplify options in Libero SoC using the Synthesis tool Configure Options dialog box. Libero passes the options in the Tcl file to Synplify Pro for processing.

RTG4 Dynamic CCC Solution – RTG4 devices now support dynamic configuration of the CCC, via a dedicated APB interface.

RTG4 RAM Tie-off – Enhancements have been made to improve power in the radiation environment.

System Verilog Support – Libero can now process Verilog files that contain System Verilog constructs.

SET Mitigation – SET Mitigation per flipflop instance is supported for RTG4.

Enhanced Constraint Flow – A number of enhancements have been made for this release, including:

- SmartFusion2/IGLOO2 - CoreConfig/CoreReset constraint flow enhancements to improve timing closure
- RTG4 EPCS constraint generation
- RTG4 XAUI constraint generation
- Encrypted VHDL flow support
- Constraint coverage reporting enhancements

More Information

- Libero – [Learn more about Libero SoC](#) including 11.7 SP1 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](#)
- Development Kits – [Learn more about development kits to accelerate your design](#)
- Licensing – [Learn more about Libero licensing](#)

Libero SoC Design Flow (Enhanced Constraint Flow) - SmartFusion2, IGLOO2, RTG4

The Libero SoC Enhanced Constraint Design Flow is shown in the figure below.

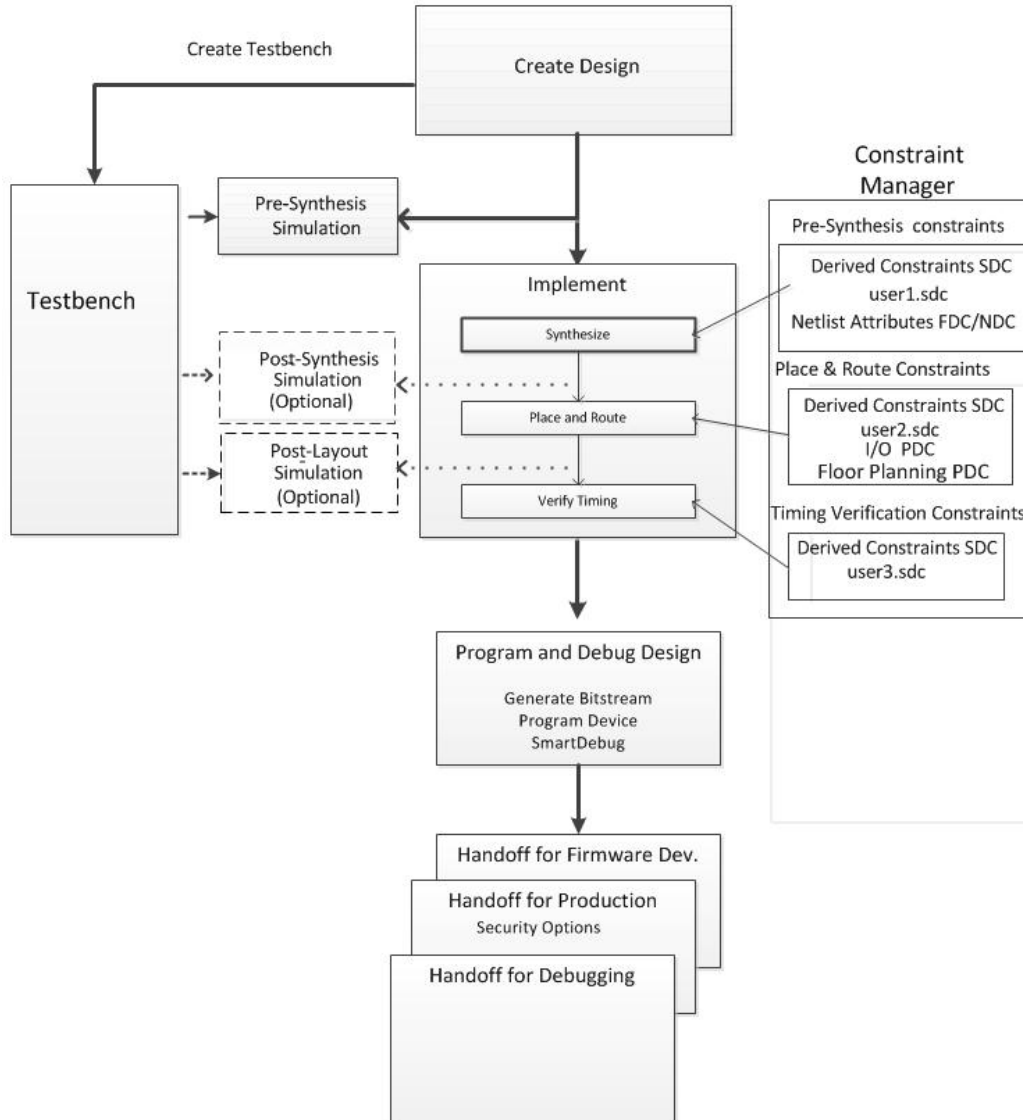



Figure 1 · Libero SoC Enhanced Constraint Design Flow for SmartFusion2, IGLOO2, and RTG4 Devices


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

Create Design

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

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



After the design is created, click the  button, and the Libero SoC software executes the design flow from Synthesis (for HDL flow) or Compile Netlist (for EDIF flow) through Place and Route with default settings.

Constraint Manager

Create Design Constraints

SmartFusion2 I/O constraint PDC files are separate from Floorplan constraint PDC files. The I/O constraints and Floorplan constraints must be in separate files.

- **I/O Constraints** - To add an I/O constraint, in the Design Flow window, double-click **Manage Constraints**. Choose the I/O Attributes tab and click:

New – to create new I/O constraints in a *.pdc file.

Import – to import an I/O constraints into the project. The imported I/O constraints file is copied into the project's constraints folder.

Link – to create a link from the project to an existing I/O constraint file located and maintained outside the Libero project.

New I/O constraint files have the *.pdc file extension and are stored in the <project_location>\constraints\io folder.

- **Timing Constraints** – In the Constraint Manager, click **Timing** to add Timing SDC files into the project

New – to create new timing constraints in an *.sdc file.

Import – to import a timing constraint *.sdc file into the project. The imported timing constraint *.sdc file is copied into the project's constraints folder.

Link – to create a link from the project to an existing timing constraint *.sdc file located and maintained outside the Libero project.

New timing constraint files have the *.sdc file extension and are stored in the <project_location>\constraints folder.

- **Floorplan Constraints** – In the Constraint Manager, click **Floor Planner** to add Floorplanning *.pdc files into the project.

New – to create new floorplanning constraints in a *.pdc file.

Import – to import a floorplanning constraint *.pdc file into the project. The imported floorplanning constraint *.pdc file is copied into the <project_location>\constraints\fp folder.

Link – to create a link from the project to an existing floorplanning constraint *.pdc file located and maintained outside the Libero project.

New floorplanning constraint files have the *.pdc file extension and are stored in the <project_location>\constraints\fp folder.

- **Netlist Attributes Constraints**

New – to create new Netlist Attribute constraints file.

Import – to import a Netlist Attribute constraints file into the project. The imported Netlist Attribute file is copied into the <project_location>\constraints folder.

Link – to create a link from the project to an existing Netlist Attribute constraint file located and maintained outside the Libero project.

New Netlist Attributes constraint files have the *.fdc (for Synplify Netlist Constraint file) or *.ndc (for Compile Netlist Constraint file) extension and are stored in the <project_location>\constraints folder.

Edit Constraints with the [Constraint Manager](#)

I/O constraints, Timing Constraints, and Floorplanning Constraints are editable from the Constraint Manager tabs.

I/O Constraints – From the Constraint Manager, click the **I/O Attributes** tab. Click **Edit with I/O Editor**. The I/O Editor opens and loads all I/O Constraint PDC files which you have associated with Place and Route. Use the I/O Editor to view, sort, select, edit, change Output Load, lock and unlock assigned attributes. When you make changes and save the changes in the I/O Editor, the modified I/O constraints are written back to the original I/O Constraint PDC file(s).


Timing Constraints – From the Constraint Manager, click the **Timing** tab. From the pull-down menu, select any one of the following to open and load the [Timing Constraints Editor](#):

- **Edit Synthesis Constraints** – The Constraint Editor opens and loads the SDC Constraint file(s) you have associated with Synthesis.
- **Edit Place and Route Constraints** – The Constraint Editor opens and loads the SDC Constraint file(s) you have associated with Place and Route.
- **Edit Timing Verifications Constraints** – The Constraint Editor opens and loads the SDC Constraint file(s) you have associated with Timing Verifications.

When you make changes and save the changes in the Constraint Editor, the modified constraints are written back to the original SDC file(s).


Floorplan Constraints - From the Constraint Manager, click the **Floor Planner** tab. Click **Edit with Floor Planner**. Chip Planner opens and loads the Floorplanning PDC constraint file(s) you have associated with Place and Route. When you make and save changes in Chip Planner, the modified floorplanning PDC constraint(s) are written back to the original floorplanning PDC file(s).

Implement

- **Synthesize**
Double-click **Synthesize** to run synthesis on your design with the default settings. The constraints associated with Synthesis in the Constraint Manager are passed to Synplify.
- **Place and Route**
Place and Route takes the design constraints from the Constraint Manager and runs with default settings. This is the last step in the push-button  design flow execution.

Program and Debug Design

Generate Bitstream

The design flow process initiated by the  button ends after Place and Route. Click **Generate Bitstream** to generate the bitstream file for programming.

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.

See Also

[Constraint Manager](#)

[New Project Wizard to import/link design constraints when creating new projects](#)

Enhanced Constraint Flow and Design Sources

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

Enhanced Constraint Flow for HDL designs

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

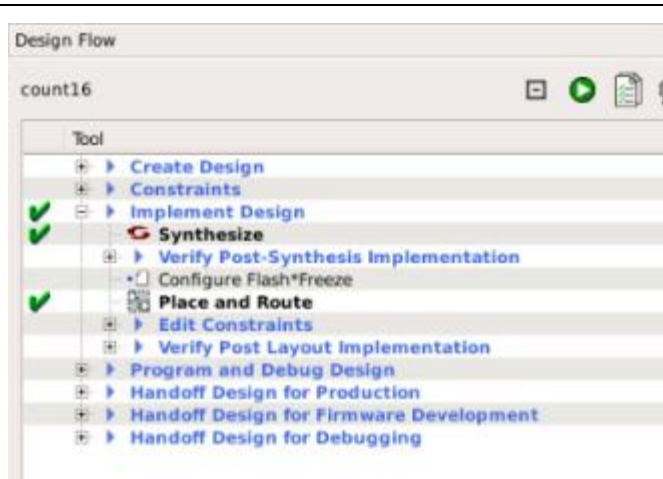
Enhanced Constraint Flow for EDIF designs

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

The options to promote or demote global resources of the chip are set in the Compile Netlist options.

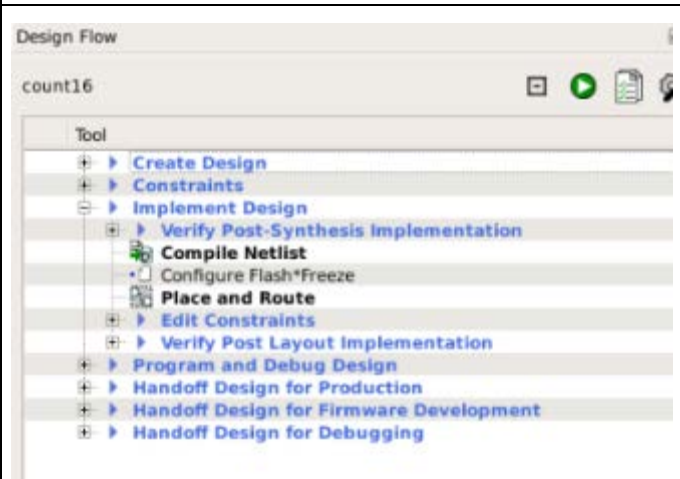
The HDL flow versus the EDIF Flow is compared and contrasted below.

HDL Flow

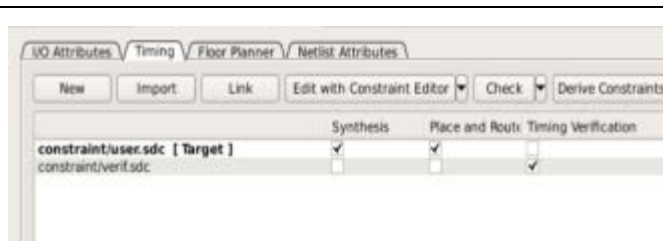


Design Flow Window

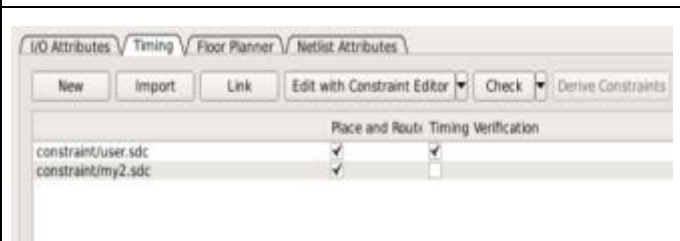
EDIF Flow



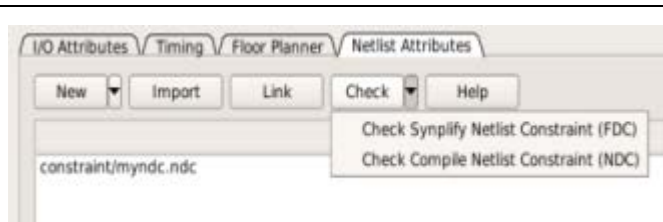
Design Flow Window



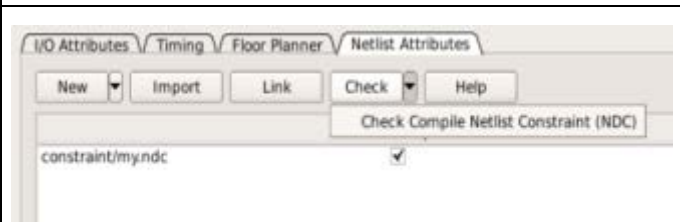
Constraint Manager



Constraint Manager

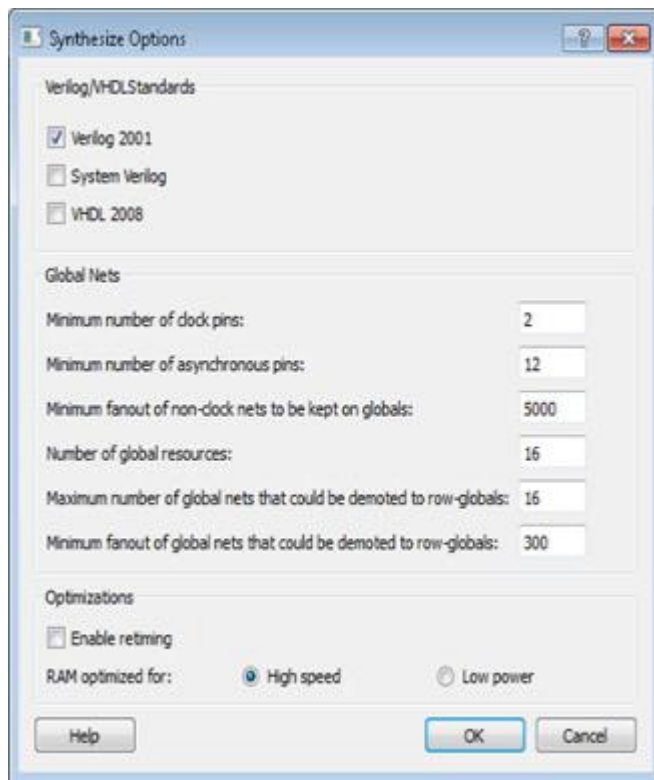


Constraint Manager - Check *.fdc and *.ndc



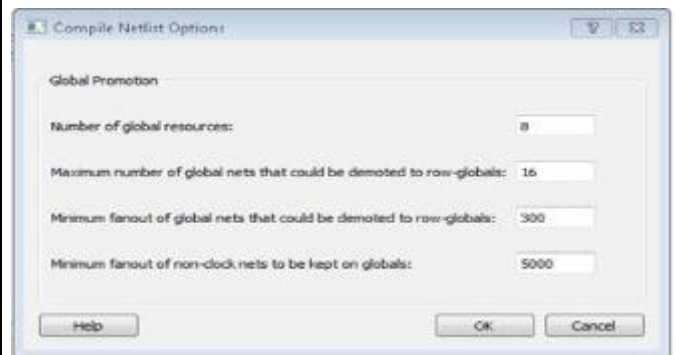
Constraint Manager - Check *.ndc only

HDL Flow



Global Promotion/Demotion Options set in Synthesis Options Dialog Box

EDIF Flow



Global Promotion/Demotion Options set in Compile Netlist Options Dialog Box

Microsemi License Utility

The [Microsemi SoC Products Group License Troubleshooting 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.

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.

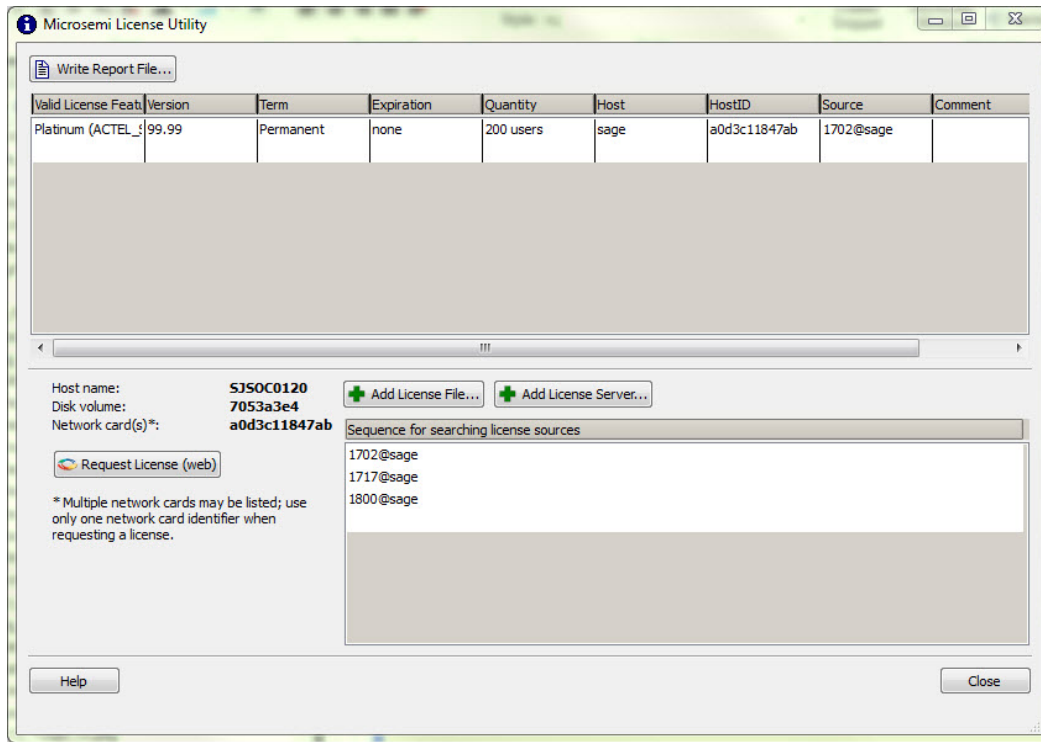


Figure 2 · Microsemi License Utility

Supported Families (Enhanced Constraint Flow)

Microsemi's Libero SoC software supports the following families of devices in Enhanced Constraint Flow:

- SmartFusion2
- IGLOO2
- RTG4

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

Table 1 · Product Families and Derivatives

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

File Types in Libero SoC

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. When you [import](#) files from outside your current project, the files are [copied into your local project folder](#).

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

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

The top level directory (<project_name>) contains your PRJ file; only one PRJ file is enabled for each Libero SoC project.

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), 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

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

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, please 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, NetlistViewer, Floorplanning, Package Editing, Place-and-Route, Debugging	Libero SoC	Microsemi SoC
Programming Software	FlashPro	Microsemi SoC

Function	Tool	Company
Programming Software	FlashPro Express	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:

- **Chip Planner** displays I/O and logic macros in your design for floorplanning
- **NetlistViewer** 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

How do I set my Multi-Pass 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 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 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](#)

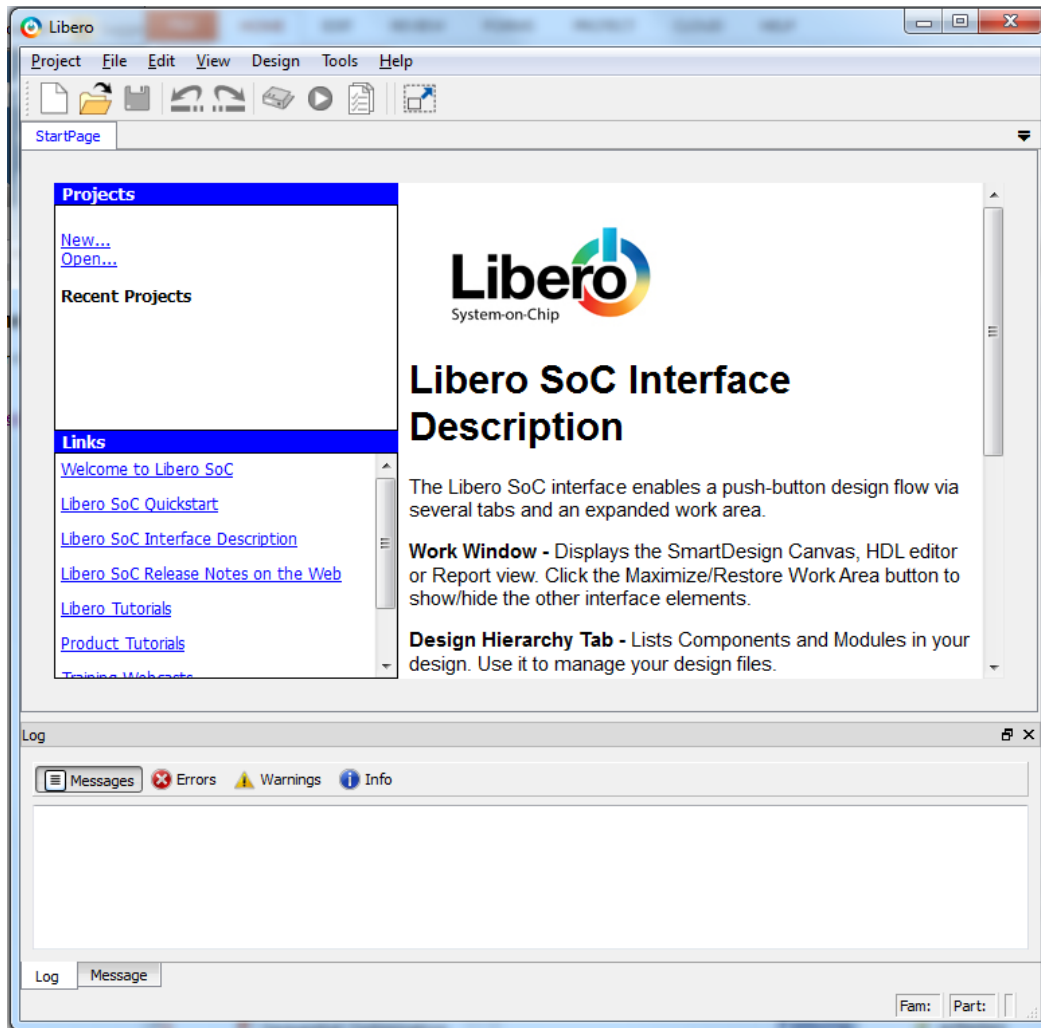
Libero Design Flow Using Enhanced Constraint Flow for SmartFusion2, IGLOO2, and RTG4

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

If you do not create or open an Enhanced Constraint Flow project for SmartFusion2, IGLOO2, and RTG4, please refer to the generic Libero SoC User Guide [ed -- insert link]

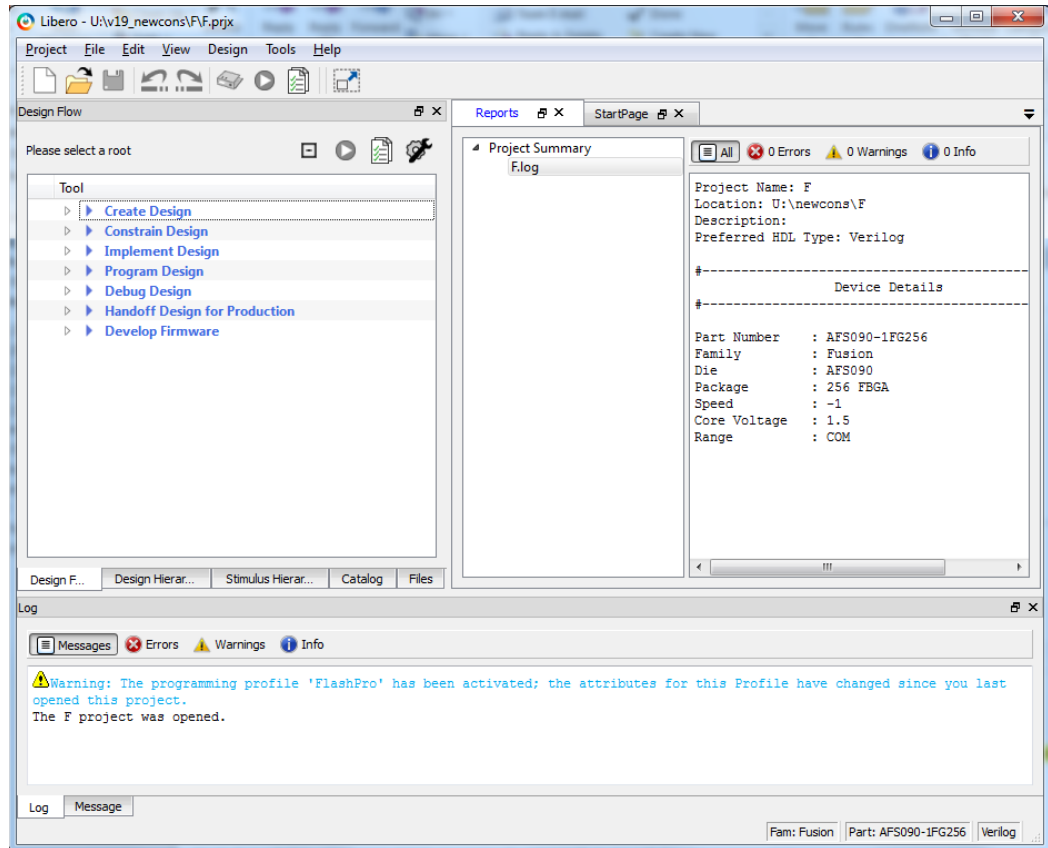
Starting the Libero GUI

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



- Clicking on $\text{Open} \dots$ opens an already existing Libero SoC project.
- Clicking on $\text{New} \dots$ starts the [New Project Wizard](#). Upon completion of the wizard, a new Libero SoC project is created and opened.

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



The Design Flow Window

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

- Create
- Constrain
- Implement
- Program and Debug
- Handoff

Design Report

The Design Report Tab lists all the reports available for your design, and displays the selected report.

Reports are added automatically as you move through design development. For example Timing reports are added when you run timing analysis on your design. The reports are updated each time you run timing analysis.

If the Report Tab is not visible, you may expose it at any time by clicking on the main menu item **Design > Reports**

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

Reports for the following steps are available for viewing here:

- Project Summary
- [Synthesize](#)
- [Place and Route](#)
- [Verify Timing](#)

- [Verify Power](#)
- Programming
 - [Generate FPGA Array Data](#)
 - [Generate Bitstream](#)
- Export
 - [Export Bitstream](#)
 - [Export Pin Report](#)
 - [Export IBIS Model](#)

Using the Libero SoC New Project Wizard to Start an Enhanced Constraint Flow Project.

Libero SoC's Enhanced Constraint Flow provides a centralized user interface provided for you to better manage design constraints (I/O Constraints, Floorplanning Constraints, Timing Constraints and Netlist Constraints) throughout the design process. It also derives SDC Timing Constraints for IP cores, such as CCC, SERDES, MSS, and OSC, based on the configuration of those cores and the component SDC. This new feature is available for SmartFusion2, IGLOO2, and RTG4.

Because this is the first release of the Enhanced Constraint Flow, it has the following limitations:

- block flow is not enabled
- The design separation methodology is not enabled

When you click **Finish** in the New Project Wizard at the end of project creation, the New Project Information dialog box appears and gives you the following choices:

- Use Classic Constraint Flow
- Use Enhanced Constraint Flow

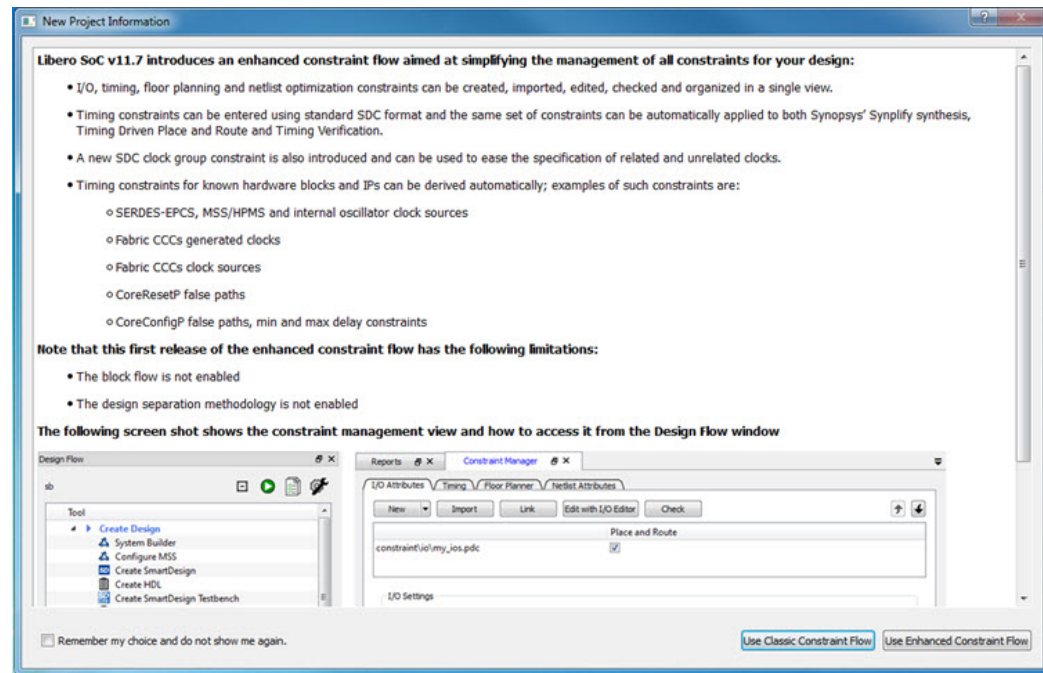


Figure 3 · New Project Information Dialog Box

Click the **Use Enhanced Constraint Flow** button to activate this feature for the design flow.

Use the **Remember my choice and do not show me again** check-box to preserve your choice in your [Libero Design Flow Preferences](#).

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

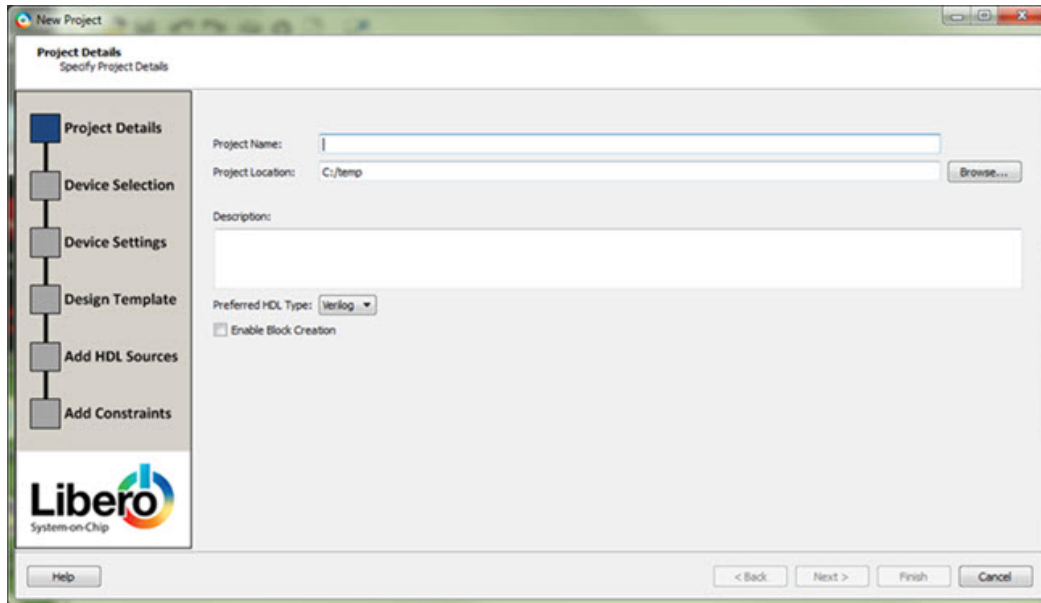


Figure 4 · Libero SoC New Project Creation Wizard

Project

Project Name - Identifies your project name; do not use spaces or reserved Verilog or VHDL keywords.

Project Location – Identifies your project location on disk.

Description – General information about your design and project. The information entered appears in your Datasheet Report View.

Preferred HDL type - Sets your HDL type: Verilog or VHDL. Libero-generated files (SmartDesigns, SmartGen cores, etc.) are created in your specified HDL type. Libero SoC supports mixed-HDL designs.

Enable Block Creation - Enables you to build blocks for your design. These blocks can be assembled in other designs, and may have already completed Layout and been optimized for timing and power performance for a specific Microsemi device. Once optimized, the same block or blocks can be used in multiple designs.

When you are finished, click **Next** to proceed to the next 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 Constraints](#)

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

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.

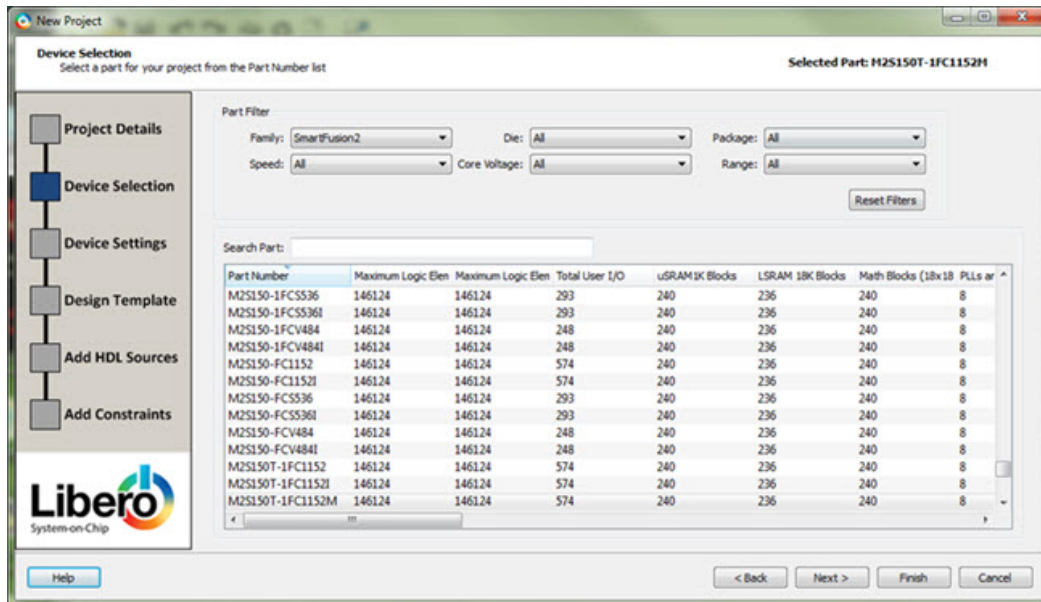


Figure 5 · New Project Creation Wizard - Device Selection Page

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. Wide range voltage for die voltage (VCCA) and VCCI is available for ProASIC3L and 1.2V IGLOO devices. 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 - Define the voltage and temperature ranges a device may encounter in your application. Tools such as SmartTime, SmartPower, timing-driven layout, power-driven layout, the timing report, and back-annotated simulation are affected by operating conditions.

Supported ranges include:

- Commercial (COM) - Junction temperature is a function of ambient temperature, air flow, and power consumption.
- Industrial (IND) - Junction temperature is a function of ambient temperature, air flow, and power consumption.
- Military (MIL) - Junction temperature is a function of the case temperature, air flow, and power consumption.

Note: Supported operating condition ranges vary according to your device and package.

Refer to your device datasheet to find your recommended temperature range.

Set your I/O operating voltages in [Project Settings](#) > Device I/O Settings.

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

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

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.

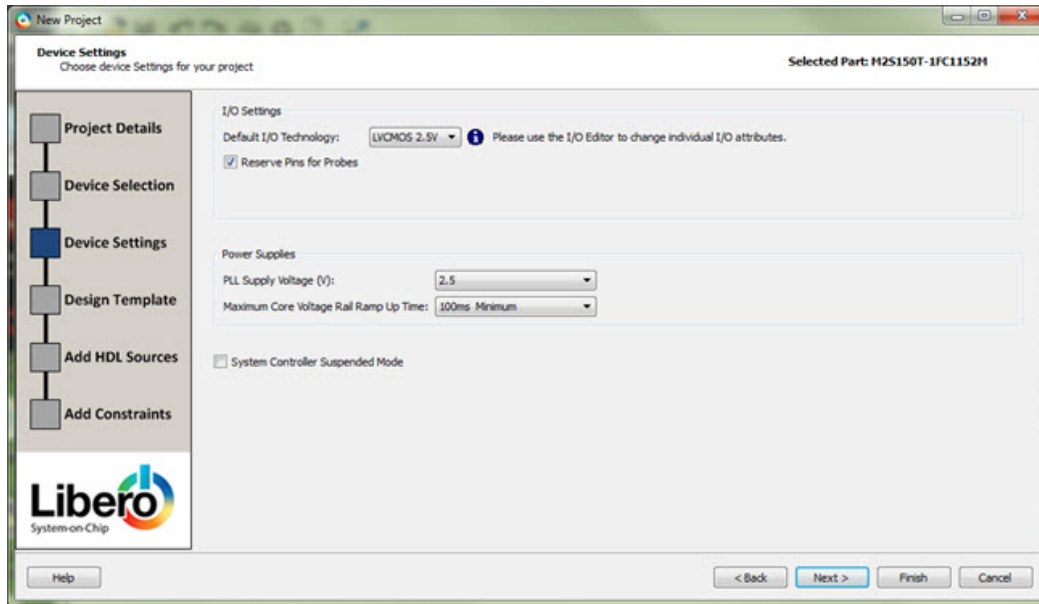


Figure 6 · New Project Creation Wizard – Device Settings Page

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.

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.

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

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

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

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

System Controller Suspended Mode (SmartFusion2, IGLOO2 only) - Enables designers to suspend operation of the System Controller. Enabling this bit instructs the System Controller to place itself in a reset state when the device is powered up. This effectively suspends all system services from being performed.

For a list of system services, refer to the SmartFusion2 or IGLOO2 System Controller User's Guide for your device on the [Microsemi website](#).

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

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.

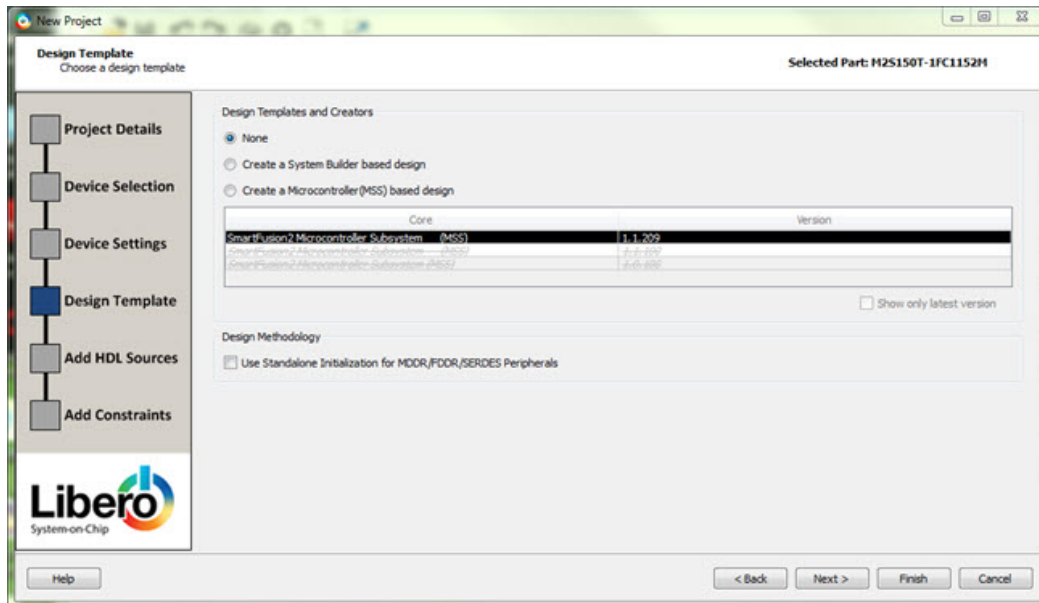


Figure 7 · New Project Creation Wizard – Design Template Page

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.

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.

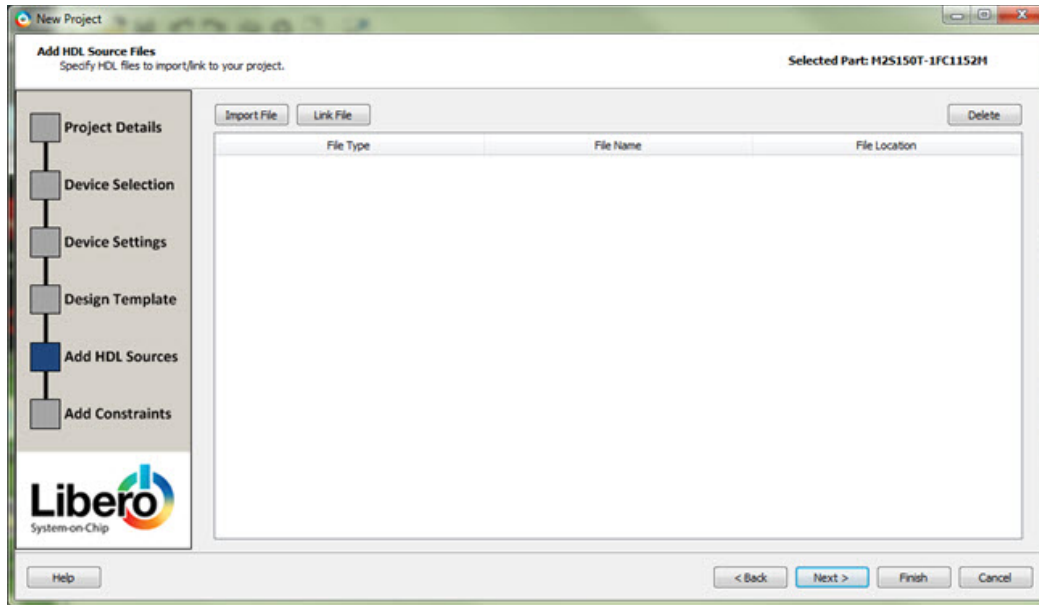


Figure 8 · New Project Creation Wizard - Add HDL Source Files Page

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.

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.

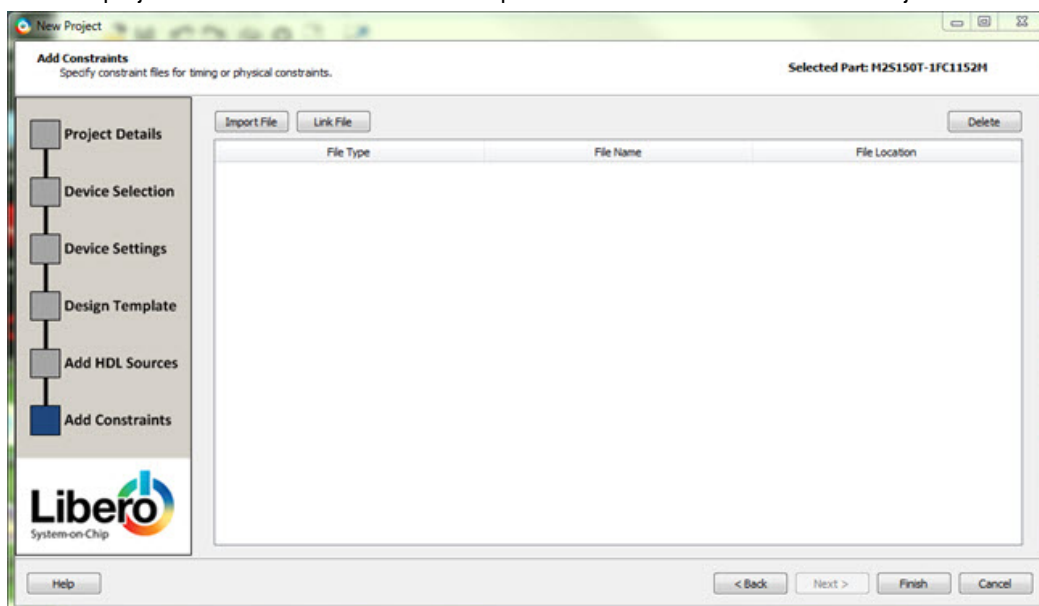


Figure 9 · New Project Creation Wizard – Add Constraints Page

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.

Click **Finish** to complete New Project Creation.

The **Reports** tab displays the result of the New Project creation.

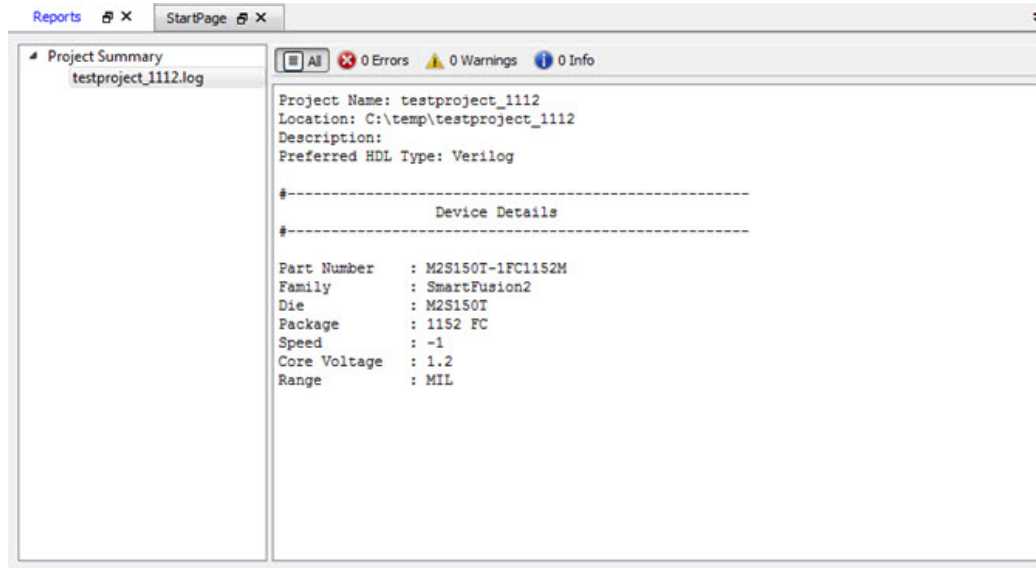


Figure 10 · Reports Tab

Create and Verify 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.

MSS - SmartFusion2 only

Instantiate a SmartFusion2 MSS in your Design

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

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

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

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

Configure the SmartFusion2 MSS

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

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

MSS ARM® Cortex™-M3

Peripherals

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

Fabric Interfaces

- MSS Fabric Interface Controllers (FICs)

Additional Information

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

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

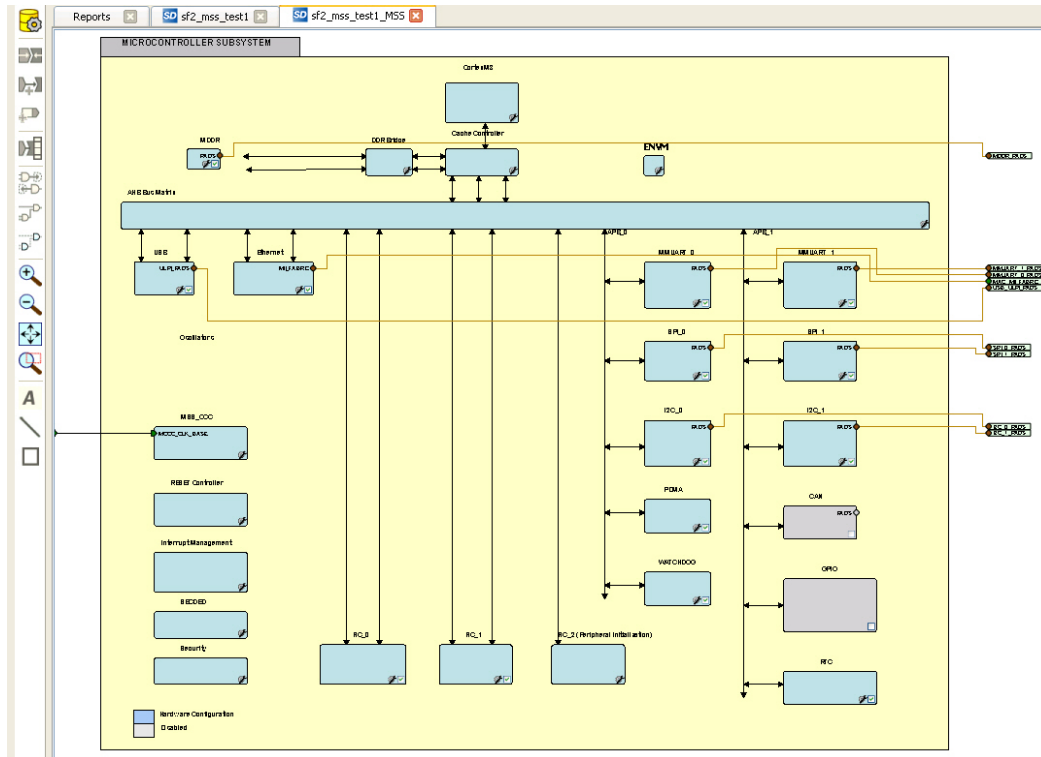


Figure 11 · Microcontroller Subsystem Configurator

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.

SmartDesign

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.
 - [View a Memory Map / Datasheet](#) - 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.

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

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 [Project Manager 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 [Project Manager 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. Hit 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**.

Getting Started With SmartDesign

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).

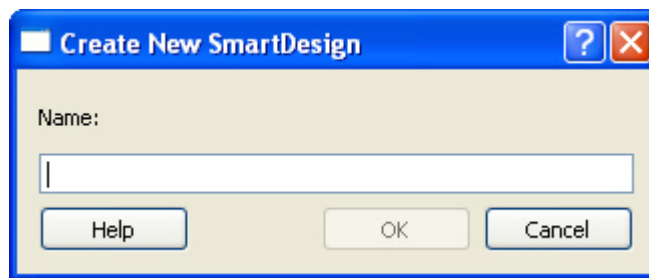


Figure 12 · Create New SmartDesign Dialog Box

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 **Import** 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.

Memory Maps / Data Sheet

If your design contains standard Bus Instances such as the DirectCore AMBA bus cores, CoreAPB or CoreAHB, then 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.

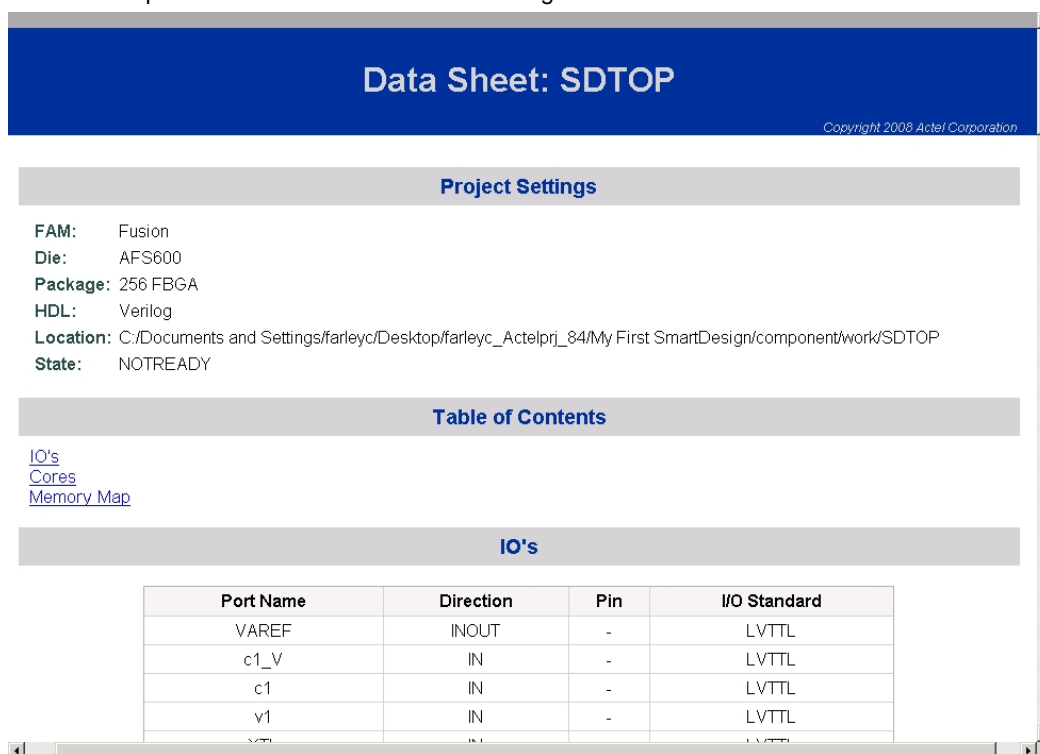
The datasheet reports the memory map of the different subsystems of your design, where a subsystem is any independent bus structure with a Master and Slave peripheral attached.

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.



Data Sheet: SDTOP
Copyright 2008 Actel Corporation

Project Settings

FAM: Fusion
Die: AFS600
Package: 256 FBGA
HDL: Verilog
Location: C:/Documents and Settings/farleyc/Desktop/farleyc_Actelprj_84/My First SmartDesign/component/work/SDTOP
State: NOTREADY

Table of Contents

[IO's](#)
[Cores](#)
[Memory Map](#)

IO's

Port Name	Direction	Pin	I/O Standard
VAREF	INOUT	-	LVTTTL
c1_V	IN	-	LVTTTL
c1	IN	-	LVTTTL
v1	IN	-	LVTTTL

Figure 13 · 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 simplifies connecting peripherals to specific base addresses on the bus. The dialog 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.

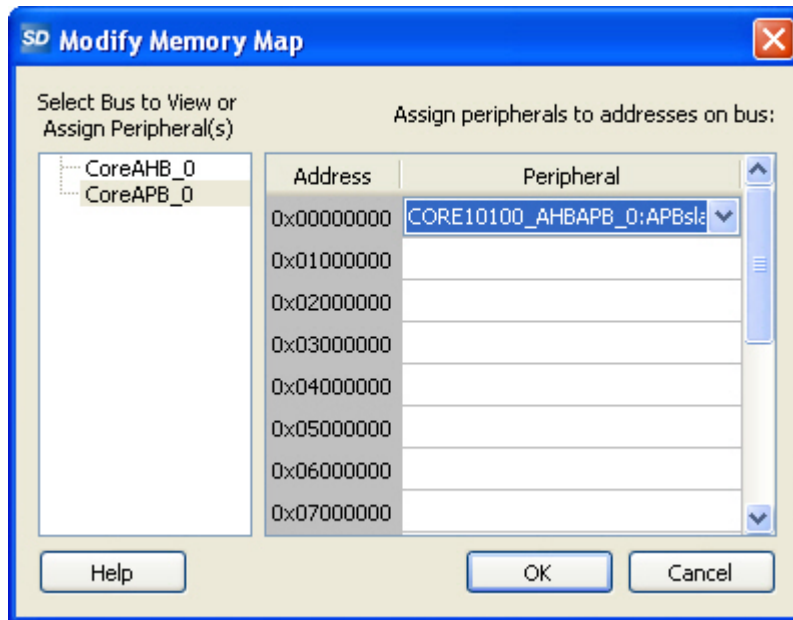


Figure 14 · Modify Memory Map Dialog Box

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.

Finding Ports/Nets/Instances in SmartDesign

Use the [Find Window](#) to search for ports, nets, or instances in SmartDesign. Searching for ports / nets / instances in SmartDesign highlights the objects on the Canvas.

Canvas View

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.

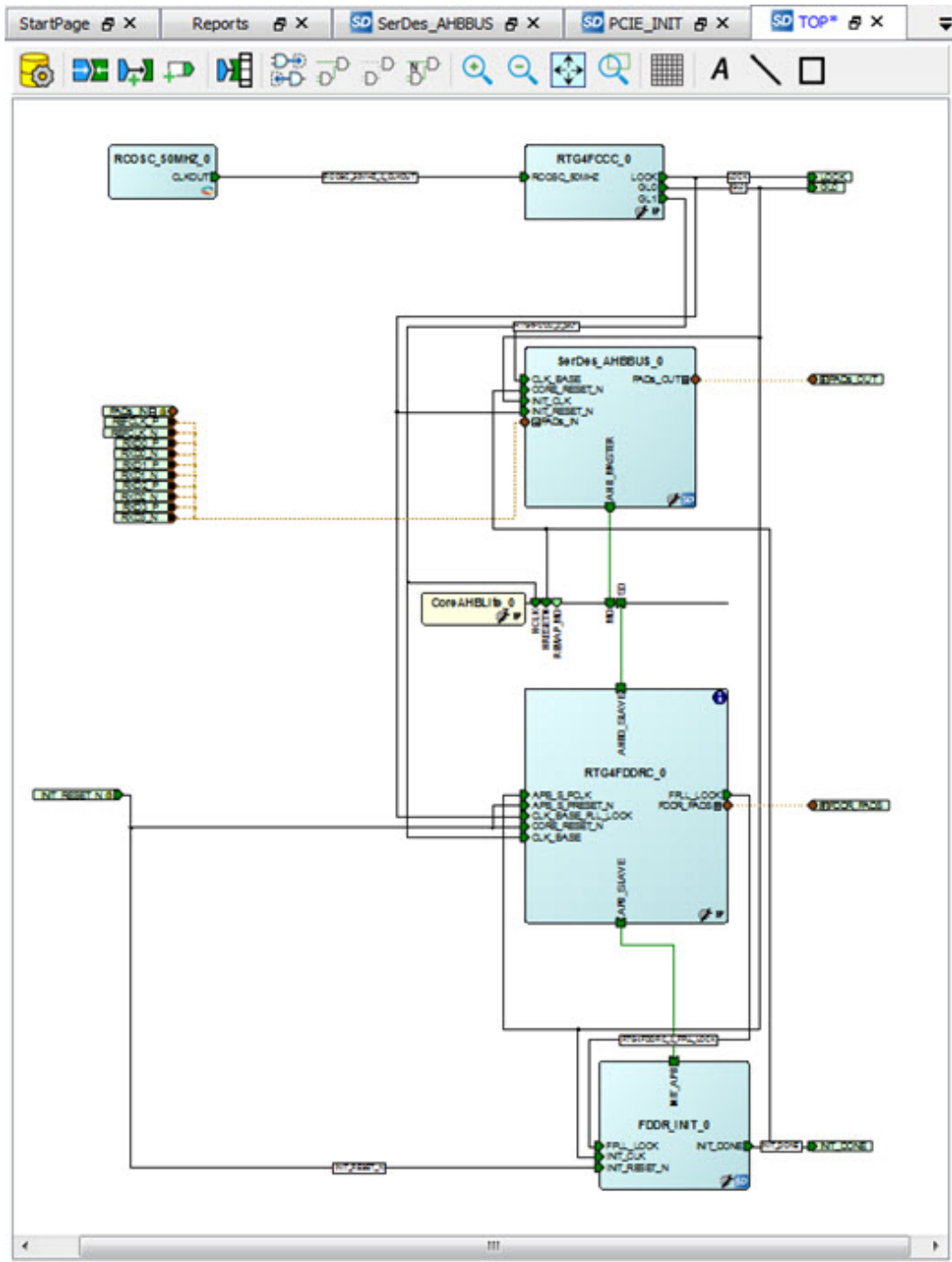


Figure 15 · SmartDesign Canvas

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.

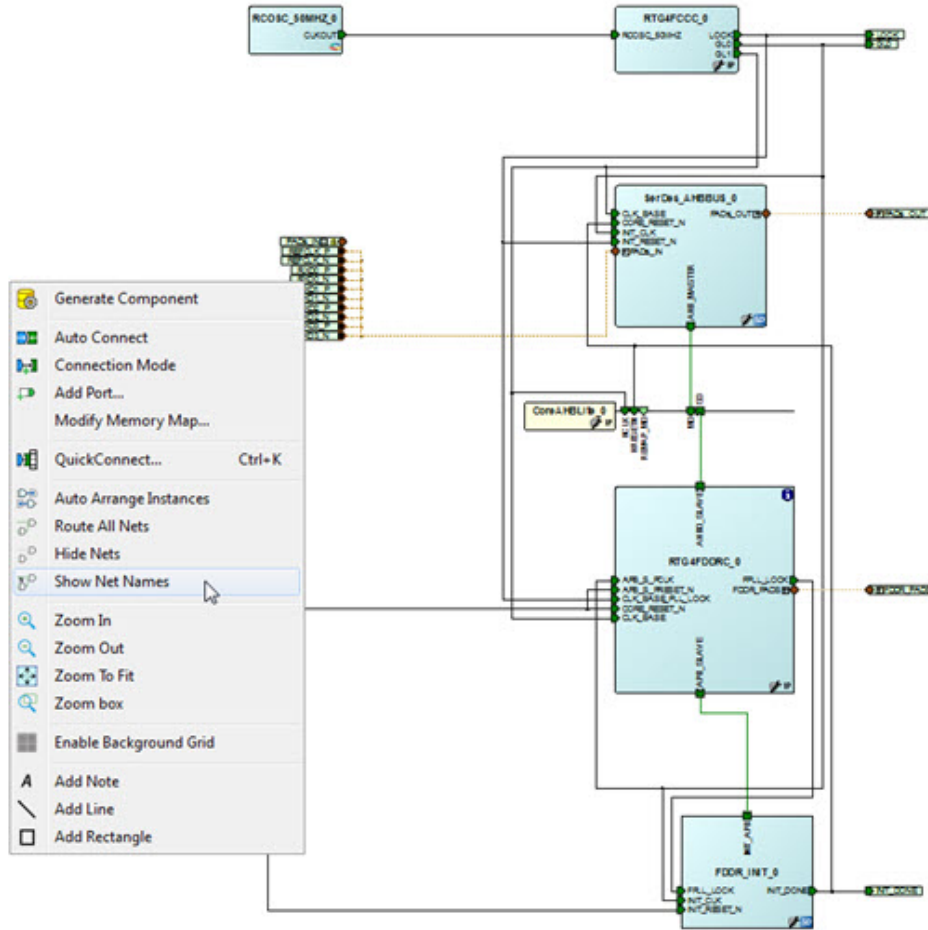


Figure 16 · Components in SmartDesign

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.

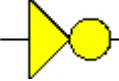


Connected pins are green.

Right-click a pin to assign an attribute.

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

Table 2 · Pin Attribute Icons

Attribute	Icon
Tie Low	
Tie High	

Attribute	Icon
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:	AHBslave1
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).

Scalar:	smartfusion_project_M55_0_FAB_CLK
smartfusion_project_M55_0	FAB_CLK
COREAHBTOAPB3_0	HCLK
CoreAHBLite_0	HCLK
CoreAhbSram_0	HCLK
CoreGPIO_0	PCLK
CoreUARTapb_0	PCLK
CustomAHBLitePeripheral_0	HCLK
corepwm_0	PCLK

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>i</i> where $i = 1 \dots n$.

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**.

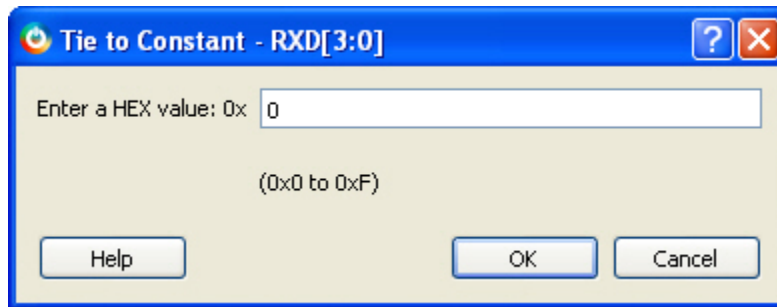


Figure 17 · Tie to Constant Dialog Box

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).

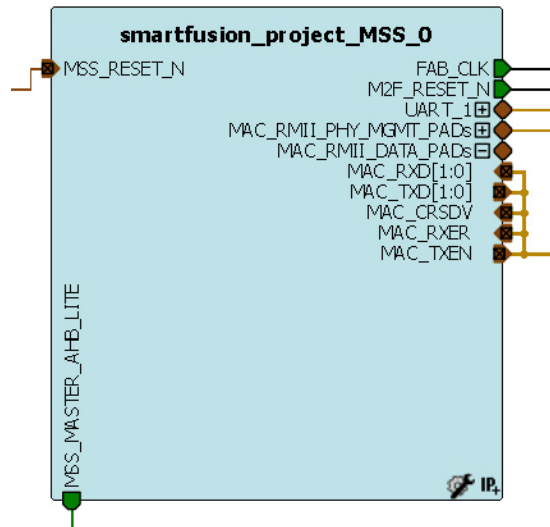


Figure 18 · Groups in an Instance on the Canvas

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.

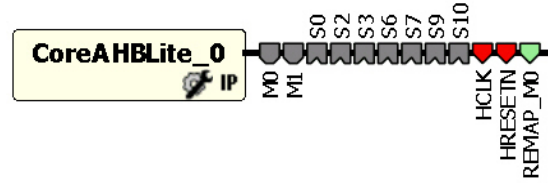


Figure 19 · Bus Instance in SmartDesign

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
	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.

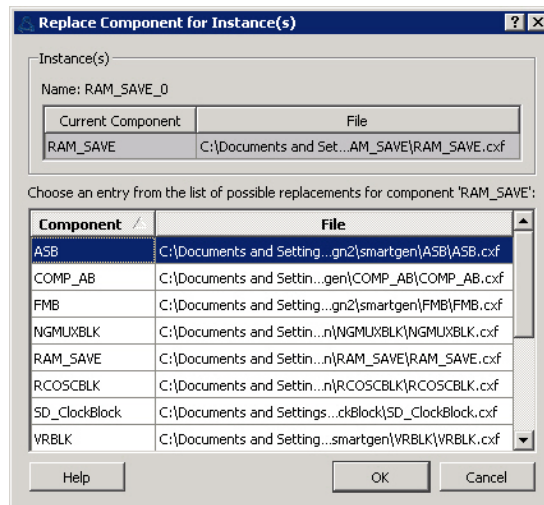


Figure 20 · Replace Component for Instance Dialog Box

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.

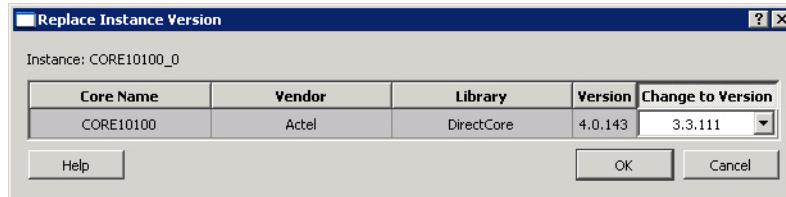


Figure 21 · Replace Instance Version Dialog Box

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).

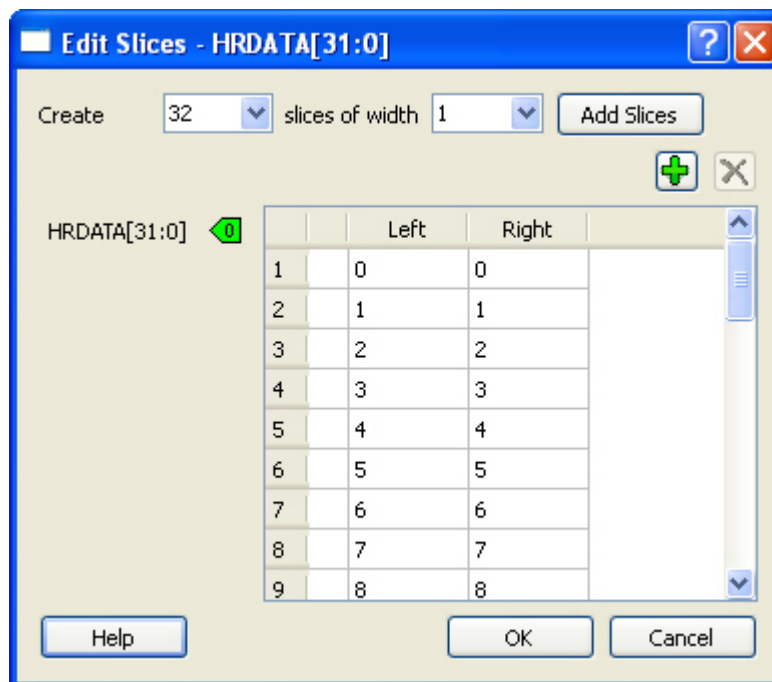


Figure 22 · Edit Slices Dialog Box

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'.

Creating a SmartDesign

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).

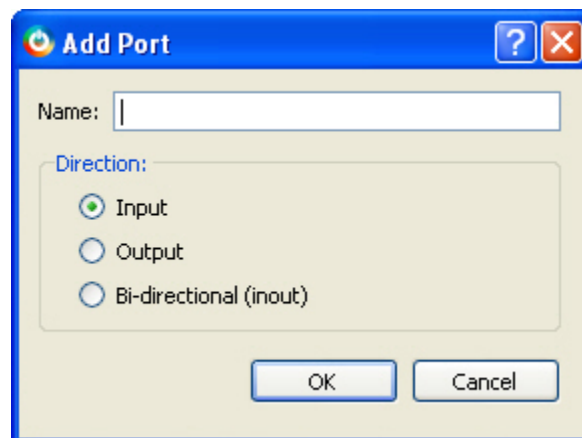


Figure 23 - Add New Port Dialog Box

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](#)

Connecting Instances

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.

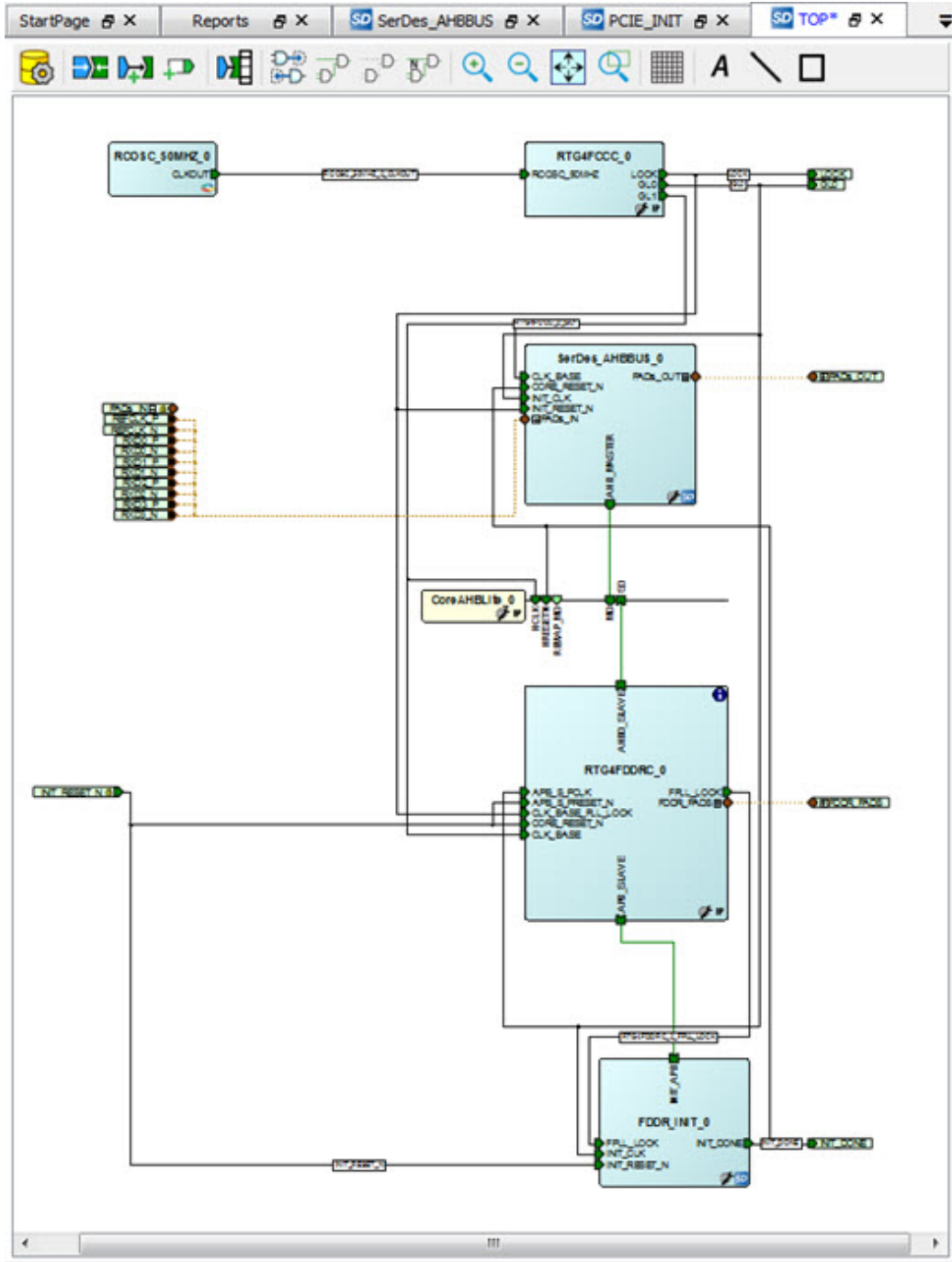


Figure 24 · Auto-Connected Cores

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: If the connection is invalid, 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 indicated 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

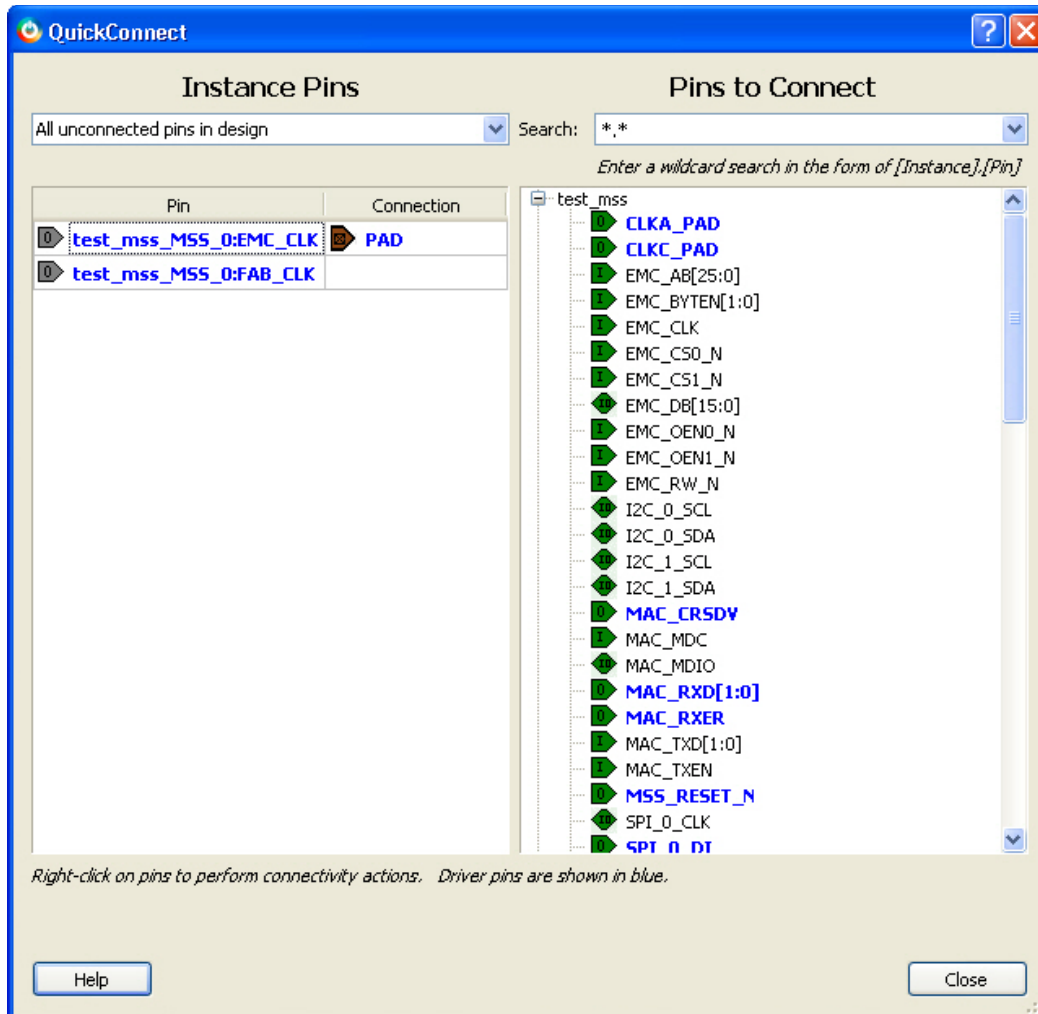


Figure 25 · QuickConnect Dialog Box

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.

Bus Interfaces

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.

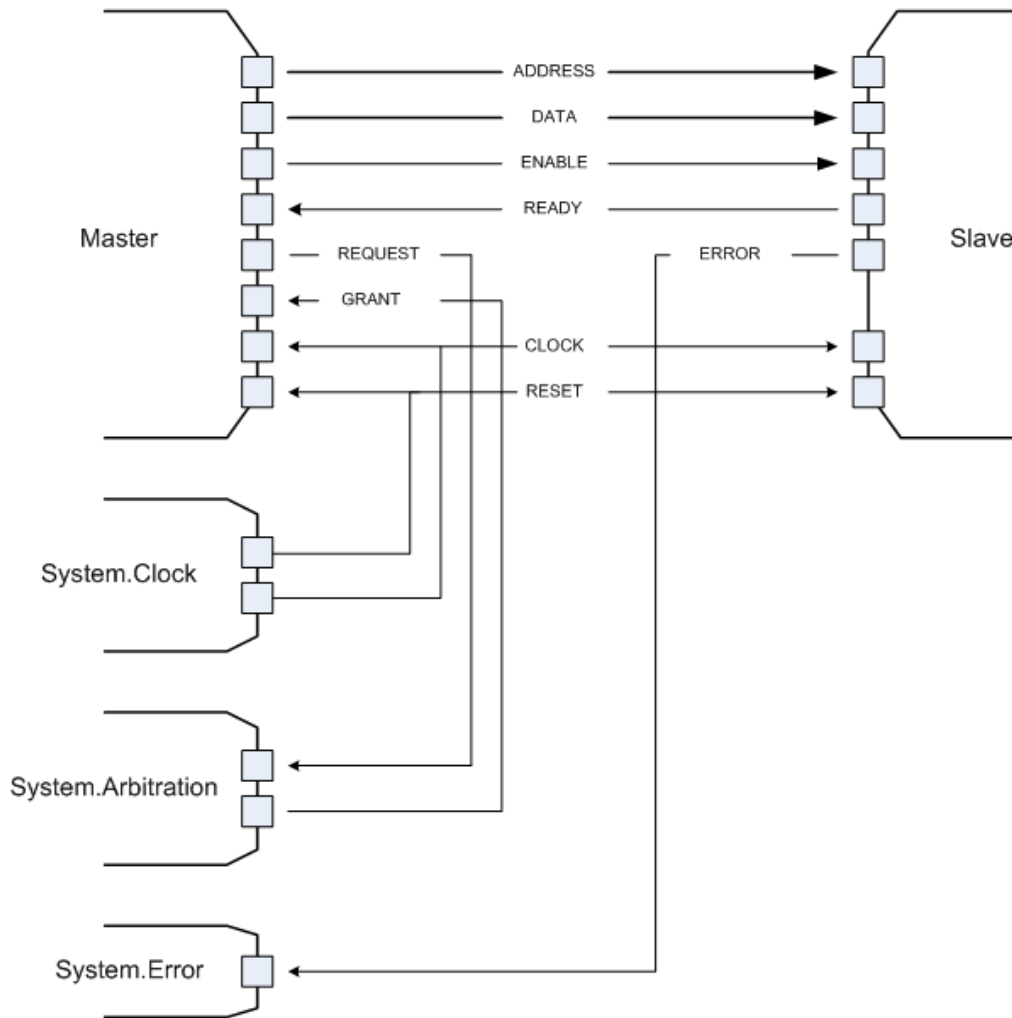


Figure 26 · Bus Definition

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.

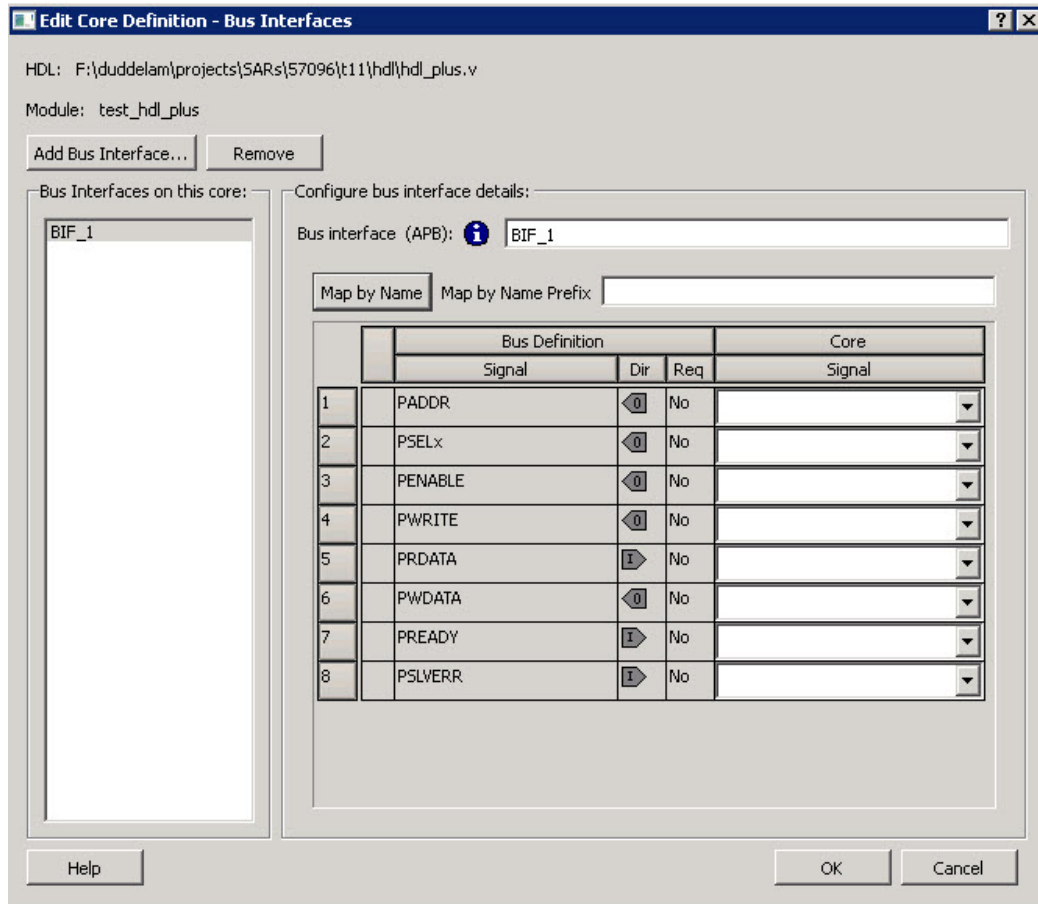


Figure 27 · Edit Core Definition Dialog Box

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:

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.

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

InitCfgAnalog

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.

CPIInterface

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).

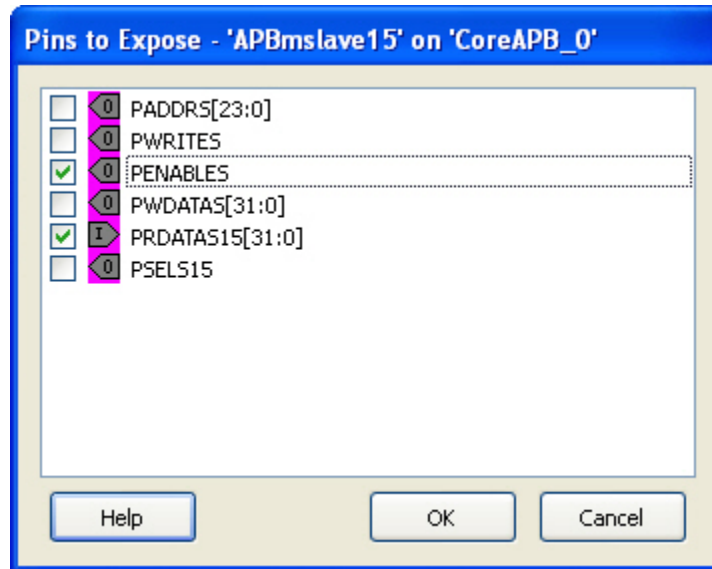


Figure 28 · Expose Driver Pin Dialog Box

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. For example, when using the Real Time Counter in a Fusion design you must also connect it up to a Crystal Oscillator core.

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

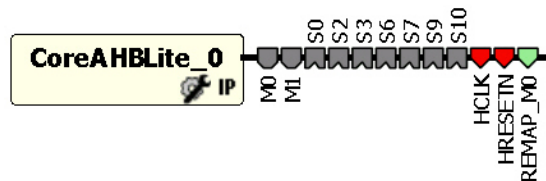


Figure 29 · Required Unconnected, Optional Unconnected, and Connected Bus Interfaces

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.

Incremental Design

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

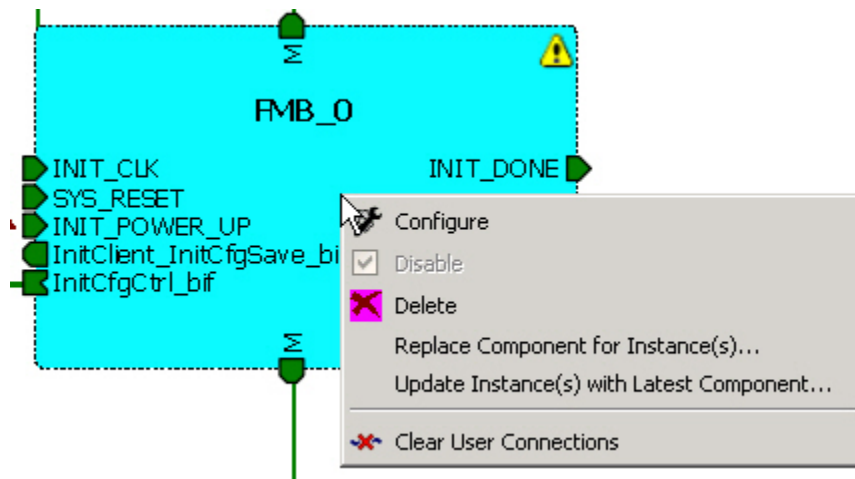


Figure 30 · Right-Click Menu - Replace Component for this Instance

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).

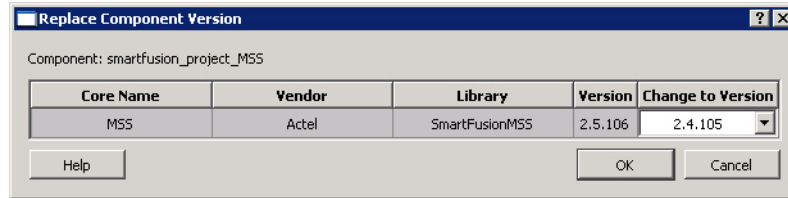


Figure 31 · Replace Component Version Dialog Box

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 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 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.

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

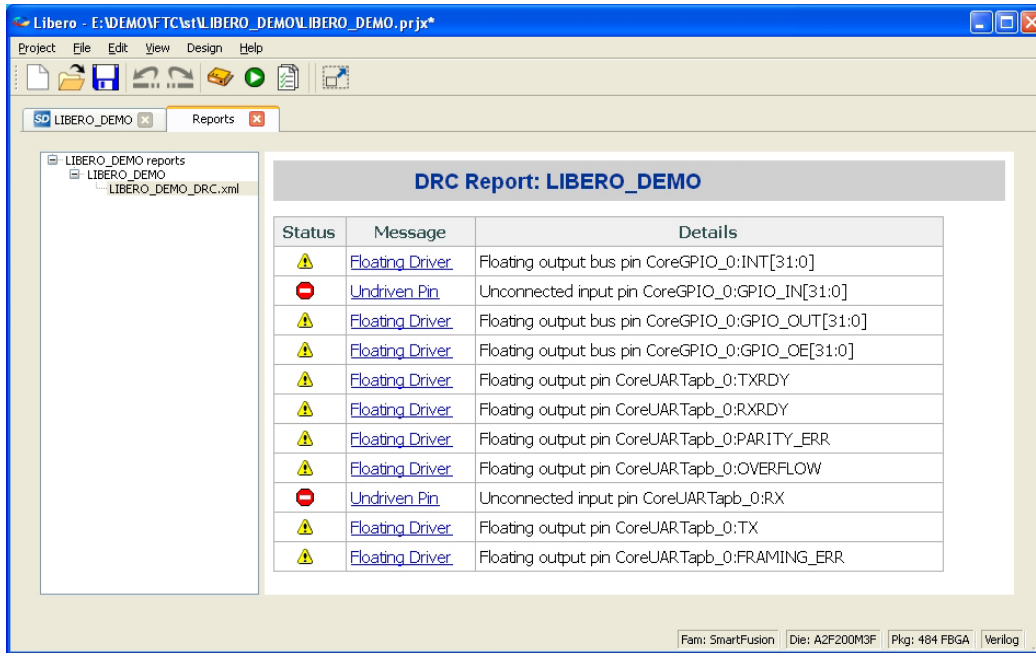


Figure 32 · Design Rules Check Results

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 rules. 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 Reference

SmartDesign Menu

Command	Icon	Function
Generate Component		Generates the SmartDesign component
Auto Connect		Auto-connects instances
Connection Mode		Toggles connection mode on or off

Command	Icon	Function
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
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

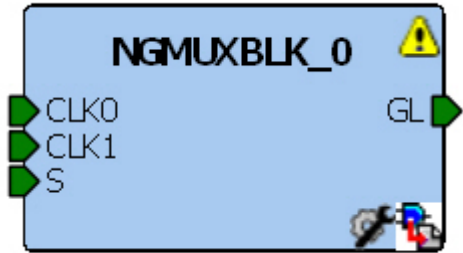
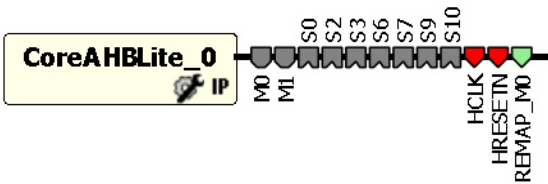
Term	Description
BIF	Abbreviation for bus interface.
bus	An array of scalar ports or pins, where all scalars have a common base name and have unique indexes in the bus.






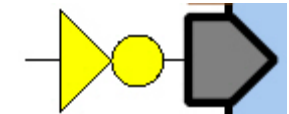


Term	Description
Bus Definition	Defines the signals that comprise a bus interface. Includes which signals are present on a master, slave, or system interface, signal direction, width, default value, etc. A bus definition is not specific to a logic or design component but is a type or protocol.
Bus Interface	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	<p>Design element with a specific functionality that is used as a building block to create a SmartDesign core.</p> <p>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.</p>
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	<p>A specific reference to a component/module that you have added to your design.</p> <p>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.</p>
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.

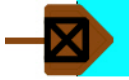
Icon	Description
 <p>The icon shows a blue rectangular component labeled 'NGMUXBLK_0'. On the left side, there are three green arrow-shaped ports labeled 'CLK0', 'CLK1', and 'S'. On the right side, there is a green arrow-shaped port labeled 'GL'. A yellow warning triangle icon is in the top right corner. At the bottom right, there is a small icon of a bicycle with a person riding it.</p>	<p>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.</p> <p>The instance type is indicated by an icon inside the instance. There are specific icons for instances from SmartDesign, HDL, and ViewDraw. The instance icon at left indicates a Microsemi SoC core.</p>
 <p>The icon shows a yellow rectangular component labeled 'CoreAHBLite_0' with a small 'IP' icon. To its right is a bus structure consisting of a series of ports labeled 'M0', 'M1', 'S0', 'S2', 'S3', 'S6', 'S7', 'S9', 'S10'. Further right are three ports labeled 'HCLK', 'HRESETN', and 'REMAP_M0'.</p>	<p>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.</p>

Icon	Description
	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																										
<div style="border: 1px solid black; padding: 5px;"> <p>Name: CoreAhbSram_0 Instance of: CoreAhbSram Type: IP Class: Regular Vendor: Actel Library: DirectCore Core Name: CoreAhbSram Version: 1.4.104 Number of ports: 13</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td>Pin: HCLK</td><td>IN</td></tr> <tr><td>Pin: HRESETn</td><td>IN</td></tr> <tr><td>Pin: HSEL</td><td>IN</td></tr> <tr><td>Pin: HWRITE</td><td>IN</td></tr> <tr><td>Pin: HREADYIN</td><td>IN</td></tr> <tr><td>Pin: HREADY</td><td>OUT</td></tr> <tr><td>Pin: HTRANS[1:0]</td><td>IN</td></tr> <tr><td>Pin: HSIZE[2:0]</td><td>IN</td></tr> <tr><td>Pin: HWDATA[31:0]</td><td>IN</td></tr> <tr><td>Pin: HADDR[14:0]</td><td>IN</td></tr> <tr><td>Pin: HRESP[1:0]</td><td>OUT</td></tr> <tr><td>Pin: HRDATA[31:0]</td><td>OUT</td></tr> <tr><td>Pin: AHBslave</td><td>SLAVE</td></tr> </table> </div>	Pin: HCLK	IN	Pin: HRESETn	IN	Pin: HSEL	IN	Pin: HWRITE	IN	Pin: HREADYIN	IN	Pin: HREADY	OUT	Pin: HTRANS[1:0]	IN	Pin: HSIZE[2:0]	IN	Pin: HWDATA[31:0]	IN	Pin: HADDR[14:0]	IN	Pin: HRESP[1:0]	OUT	Pin: HRDATA[31:0]	OUT	Pin: AHBslave	SLAVE	<p>Instance details. If there are less than twenty ports, they are listed in the details.</p>
Pin: HCLK	IN																										
Pin: HRESETn	IN																										
Pin: HSEL	IN																										
Pin: HWRITE	IN																										
Pin: HREADYIN	IN																										
Pin: HREADY	OUT																										
Pin: HTRANS[1:0]	IN																										
Pin: HSIZE[2:0]	IN																										
Pin: HWDATA[31:0]	IN																										
Pin: HADDR[14:0]	IN																										
Pin: HRESP[1:0]	OUT																										
Pin: HRDATA[31:0]	OUT																										
Pin: AHBslave	SLAVE																										
<div style="border: 1px solid black; padding: 5px;"> <p>Bus Net: DataB[1:0]</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td style="background-color: #cccccc;">sd_acc</td><td>DataB[1:0]</td></tr> <tr><td style="background-color: #99cc99;">subtr_1_0</td><td>DataB[1:0]</td></tr> </table> </div>	sd_acc	DataB[1:0]	subtr_1_0	DataB[1:0]	<p>Bus Net details.</p>																						
sd_acc	DataB[1:0]																										
subtr_1_0	DataB[1:0]																										
	<p>Master bus interface icon. A master is a bus interface that initiates a transaction on a bus interface net.</p> <p>An unconnected master BIF with REQUIRED connection is red (shown at left).</p> <p>A master BIF with unconnected OPTIONAL connection is gray.</p>																										

Icon	Description																				
<div style="border: 1px solid black; padding: 5px;"> <p>Name: RTCVR_bif Role: master State: Unconnected - required</p> <p>*This pin is a required connection, you must connect it for a valid design.</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th colspan="2" style="text-align: center;">Pin Map</th> </tr> <tr> <th>Formal</th> <th>Actual</th> </tr> </thead> <tbody> <tr> <td>RTCPMMATCH</td> <td>RTCPMMATCH</td> </tr> </tbody> </table> </div>	Pin Map		Formal	Actual	RTCPMMATCH	RTCPMMATCH	<p>Master BIF details, showing name, role, and state.</p> <p>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).</p>														
Pin Map																					
Formal	Actual																				
RTCPMMATCH	RTCPMMATCH																				
	<p>Slave BIF (shown at left).</p> <p>Unconnected slave icons with REQUIRED connections are red.</p> <p>Unconnected slave icons with OPTIONAL connections are gray.</p>																				
<div style="border: 1px solid black; padding: 5px;"> <p>Name: ExtSeqCtrl_bif Role: slave State: Unconnected</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th colspan="2" style="text-align: center;">Pin Map</th> </tr> <tr> <th>Formal</th> <th>Actual</th> </tr> </thead> <tbody> <tr> <td>ASSC_SEQIN</td> <td>ASSC_SEQIN[5:0]</td> </tr> <tr> <td>ASSC_SEQJUMP</td> <td>ASSC_SEQJUMP</td> </tr> <tr> <td>ASSC_MODE</td> <td>ASSC_XMODE</td> </tr> <tr> <td>ASSC_XTRIG</td> <td>ASSC_XTRIG</td> </tr> <tr> <td>ASSC_DONE</td> <td>ASSC_DONE</td> </tr> <tr> <td>ASSC_SEQOUT</td> <td>ASSC_SEQOUT[5:0]</td> </tr> <tr> <td>ASSC_SEQCHANGE</td> <td>ASSC_SEQCHANGE</td> </tr> <tr> <td>ASSC_SAMPFLAG</td> <td>ASSC_SAMPFLAG</td> </tr> </tbody> </table> </div>	Pin Map		Formal	Actual	ASSC_SEQIN	ASSC_SEQIN[5:0]	ASSC_SEQJUMP	ASSC_SEQJUMP	ASSC_MODE	ASSC_XMODE	ASSC_XTRIG	ASSC_XTRIG	ASSC_DONE	ASSC_DONE	ASSC_SEQOUT	ASSC_SEQOUT[5:0]	ASSC_SEQCHANGE	ASSC_SEQCHANGE	ASSC_SAMPFLAG	ASSC_SAMPFLAG	<p>Slave BIF details, showing name, role, and state.</p> <p>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 (CLKA).</p>
Pin Map																					
Formal	Actual																				
ASSC_SEQIN	ASSC_SEQIN[5:0]																				
ASSC_SEQJUMP	ASSC_SEQJUMP																				
ASSC_MODE	ASSC_XMODE																				
ASSC_XTRIG	ASSC_XTRIG																				
ASSC_DONE	ASSC_DONE																				
ASSC_SEQOUT	ASSC_SEQOUT[5:0]																				
ASSC_SEQCHANGE	ASSC_SEQCHANGE																				
ASSC_SAMPFLAG	ASSC_SAMPFLAG																				
	<p>Master-slave bus interface connection</p>																				

Icon	Description																												
<div data-bbox="344 262 570 346"> <p>Name: AHBslave2 Role: mirroredSlave State: Connected</p> </div> <div data-bbox="344 352 570 766"> <p>Pin Map</p> <table border="1"> <thead> <tr> <th>Formal</th> <th>Actual</th> </tr> </thead> <tbody> <tr><td>HADDR</td><td>HADDR_S2[31:0]</td></tr> <tr><td>HTRANS</td><td>HTRANS_S2[1:0]</td></tr> <tr><td>HWRITE</td><td>HWRITE_S2</td></tr> <tr><td>HSIZE</td><td>HSIZE_S2[2:0]</td></tr> <tr><td>HWDATA</td><td>HWDATA_S2[31:0]</td></tr> <tr><td>HSELx</td><td>HSEL_S2</td></tr> <tr><td>HRDATA</td><td>HRDATA_S2[31:0]</td></tr> <tr><td>HREADY</td><td>HREADY_S2</td></tr> <tr><td>HMASTLOCK</td><td>HMASTLOCK_S2</td></tr> <tr><td>HREADYOUT</td><td>HREADYOUT_S2</td></tr> <tr><td>HRESP</td><td>HRESP_S2[1:0]</td></tr> <tr><td>HBURST</td><td>HBURST_S2[2:0]</td></tr> <tr><td>HPROT</td><td>HPROT_S2[3:0]</td></tr> </tbody> </table> </div>	Formal	Actual	HADDR	HADDR_S2[31:0]	HTRANS	HTRANS_S2[1:0]	HWRITE	HWRITE_S2	HSIZE	HSIZE_S2[2:0]	HWDATA	HWDATA_S2[31:0]	HSELx	HSEL_S2	HRDATA	HRDATA_S2[31:0]	HREADY	HREADY_S2	HMASTLOCK	HMASTLOCK_S2	HREADYOUT	HREADYOUT_S2	HRESP	HRESP_S2[1:0]	HBURST	HBURST_S2[2:0]	HPROT	HPROT_S2[3:0]	<p>Master-slave bus interface connection details.</p>
Formal	Actual																												
HADDR	HADDR_S2[31:0]																												
HTRANS	HTRANS_S2[1:0]																												
HWRITE	HWRITE_S2																												
HSIZE	HSIZE_S2[2:0]																												
HWDATA	HWDATA_S2[31:0]																												
HSELx	HSEL_S2																												
HRDATA	HRDATA_S2[31:0]																												
HREADY	HREADY_S2																												
HMASTLOCK	HMASTLOCK_S2																												
HREADYOUT	HREADYOUT_S2																												
HRESP	HRESP_S2[1:0]																												
HBURST	HBURST_S2[2:0]																												
HPROT	HPROT_S2[3:0]																												
	<p>Groups of pins in an instance.</p> <p>Fully connected groups are solid green.</p> <p>Partially connected groups are gray with a green outline.</p> <p>Unconnected groups (no connections) are gray with a black outline.</p>																												
	<p>A system BIF is the bus interface that does not have a simple input/output relationship on both master/slave.</p> <p>This could include signals that only drive the master interface, or only drive the slave interface, or drive both the master and slave interfaces.</p>																												
<div data-bbox="344 1558 695 1843"> <p>Name: InitCfgSave_bif Role: system State: Connected</p> <p>Pin Map</p> <table border="1"> <thead> <tr> <th>Formal</th> <th>Actual</th> </tr> </thead> <tbody> <tr> <td>CLIENTAVAILx0</td> <td>ramrd</td> </tr> </tbody> </table> </div>	Formal	Actual	CLIENTAVAILx0	ramrd	<p>System BIF details, showing name, role, and state.</p> <p>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).</p>																								
Formal	Actual																												
CLIENTAVAILx0	ramrd																												

Icon	Description
	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)
MetaEnumeratedItem : **enum** <enum_name> (Item_name{,Item_name})
Range : [NumericRange | MetaEnumeratedItem]
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.

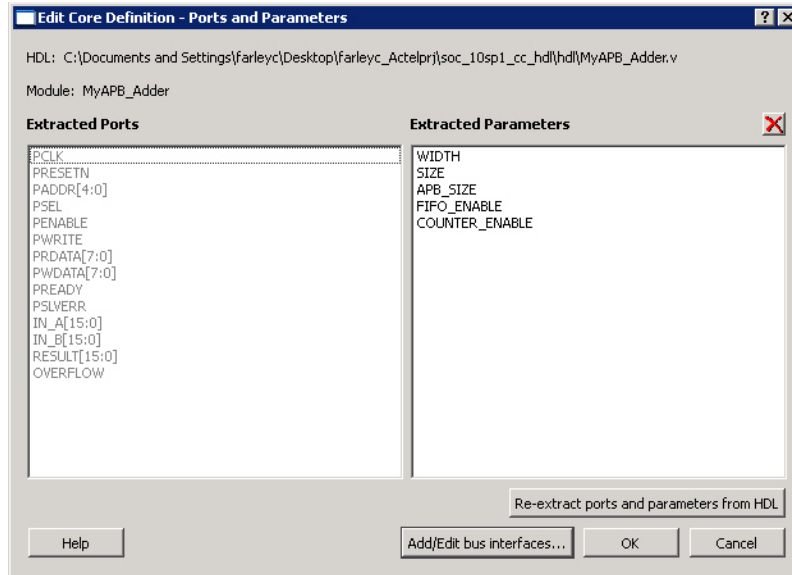


Figure 33 · Edit Core Definition - Ports and Parameters Dialog Box

- (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 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.

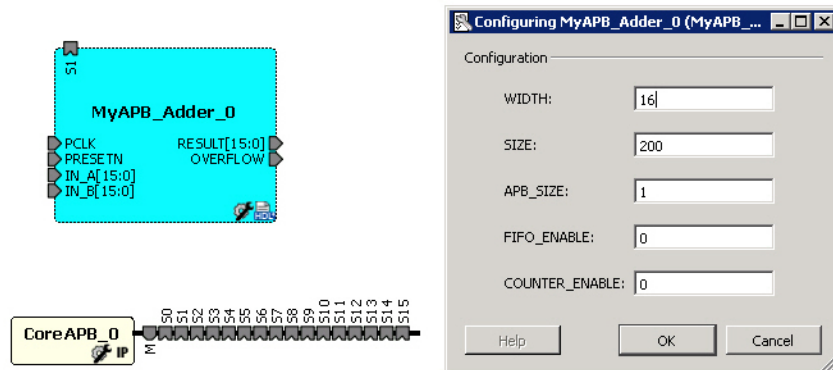


Figure 34 · HDL+ Instance and Configuration Dialog Box

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**.

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.

Designing with HDL

Create HDL

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

You can use VHDL and Verilog to implement your design.

To create an HDL file:

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

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

Using the HDL Editor

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

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

Editing

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

Saving

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

Printing

Print is available from the File menu and the toolbar.

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

HDL Syntax Checker

To run the syntax checker:

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

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

Commenting Text

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

To comment or uncomment out text:

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

Find

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

You can search for:

- Match Case
- Match whole word
- Regular Expression

The Find to Replace function is also supported.

Column Editing

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

Importing HDL Source Files

To import an HDL source file:

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

Mixed-HDL Support in Libero SoC

You must have ModelSim PE or SE to use mixed HDL in the Libero SoC. Also, you must 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. (E.g. if tb_top is in verilog, <fam>.v is compiled.)

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

HDL Testbench

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

HDL Testbench automatically instantiates the selected SmartDesign into the Canvas.

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.

Set your HDL Type, specify a name, select the data options and click OK to create a new testbench.

Initialize file with standard template populates the new HDL file with basic headers and Clock/Reset driver, as in the header of the example file below.

Instantiate Root Design includes your root design information in the new file. It includes architectural, constant, signal, component, clock, and port information.

Set as Active Stimulus sets the HDL Testbench as the stimulus file to use for simulations.

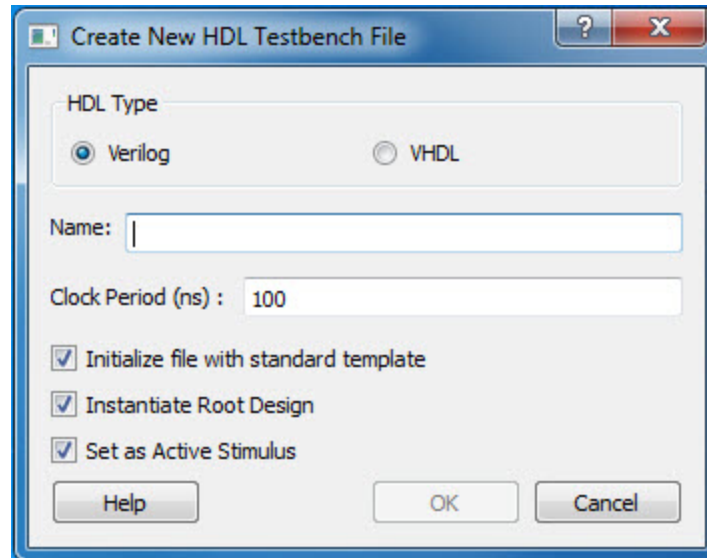


Figure 35 · Create HDL Testbench Dialog Box

```

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 36 · HDL Testbench Example - Standard Template and Root Design Enabled

Simulation and Verification

RTL Simulation

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

If you wish 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, we recommend 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 ModelSim without a testbench.

To associate a stimulus:

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

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

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

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

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

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

Selecting Additional Modules for Simulation

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

If you need additional modules in simulation, in the Design Flow window right-click **Simulate** and choose **Organize Input Files > Organize Source Files**. The Organize Files for Simulation dialog box appears. Select the HDL modules you wish to add from the Simulation Files in the Project list and click **Add** to add them to the Associated Stimulus Files list

Performing Functional Simulation

To perform functional simulation:

1. Create your testbench.
2. Right-click **Simulate** (in Design Flow window, Implement Design > Verify Post-Synthesis Implementation > Simulate) and choose **Organize Input Files > Organize Source 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 AE, right-click **Simulate** in the Design Hierarchy window and choose **Open Interactively**.
ModelSim starts and compiles the appropriate source files. When the compilation completes, the simulator runs for 1 μ s and the Wave window opens to display the simulation results.
5. Scroll in the Wave window to verify that the logic of your design functions as intended. Use the zoom buttons to zoom in and out as necessary.
6. From the **File** menu, select **Quit**.

Performing DirectCore Functional Simulation

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

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

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

To perform functional simulation of a DirectCore project:

1. Right-click a stitched module of the DirectCore project and select **Set as root**.
2. To start ModelSim AE, right-click **Simulate** in the Design Hierarchy window and choose **Open Interactively**.

ModelSim starts and compiles the appropriate source files. When the compilation completes, the simulator runs for 1 μ s and the Wave window opens to display the simulation results.

3. Scroll in the Wave window to verify that the logic of your design functions as intended. Use the zoom buttons to zoom in and out as necessary.
4. From the **File** menu, select **Quit**.

Libero SoC Constraint Management (Enhanced Constraint Flow)

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

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

To enable the Constraint Manager in the Design Flow window, check the Enhanced Constraint Flow checkbox in the pop-up window at the end of project creation.

Note: The Enhanced Constraint Flow option is available only for SmartFusion2, IGLOO2, RTG4.

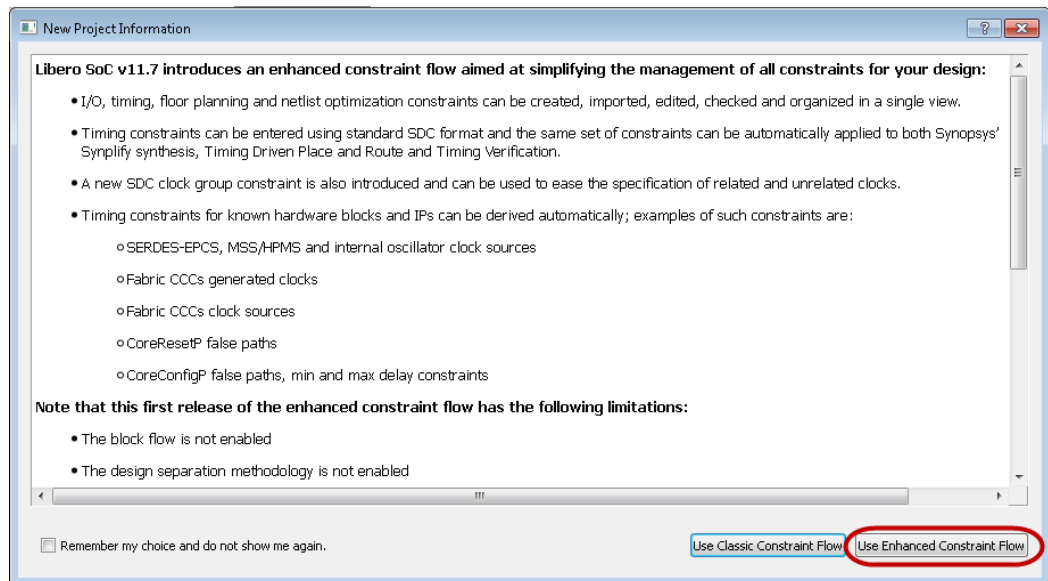


Figure 37 · New Project Information Dialog Box

Note: If you plan to always use the Enhanced Constraint Flow for your new projects and you don't want to be prompted to make a selection for Classic Constraint Flow versus Enhanced Constraint Flow every time you create a new project, you may set your personal preferences in the Project > Preferences > Design Flow dialog box as follows:

- Check the box Use Enhanced Constraint Flow when creating a new project.
- Uncheck the box Show Classic/Enhanced Constraint Flow choice dialog when creating a new project.

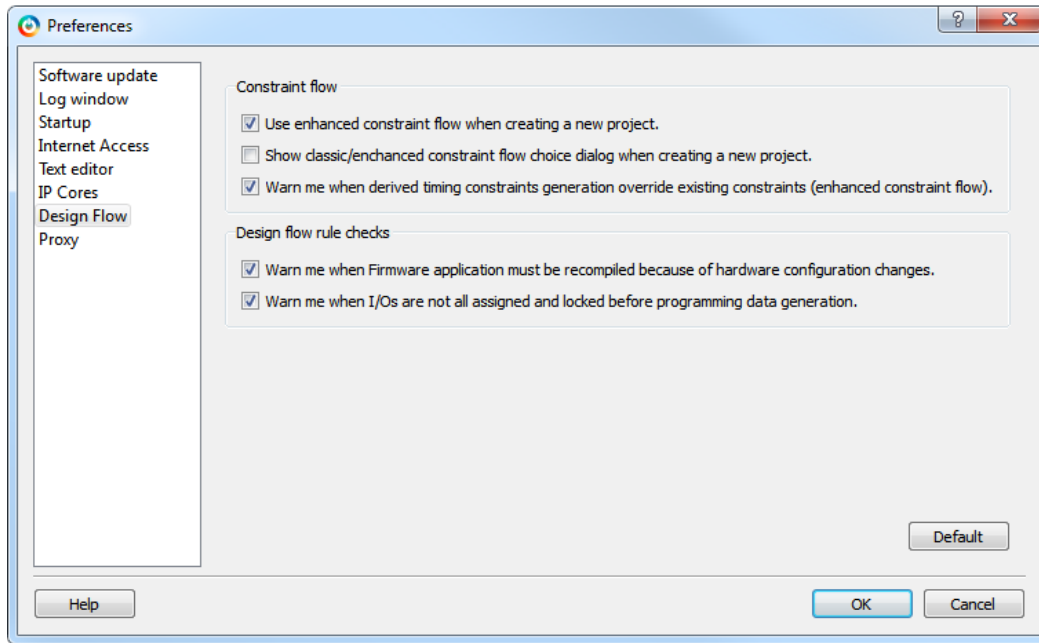


Figure 38 · Project > Preferences > Design Flow Dialog

Invocation of Constraint Manager From the Design Flow Window

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

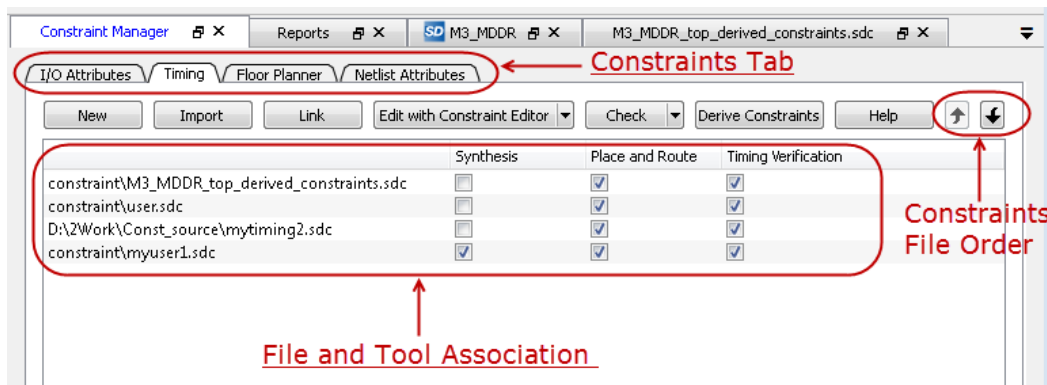


Figure 39 · Constraint Manager

Libero SoC Design Flow

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

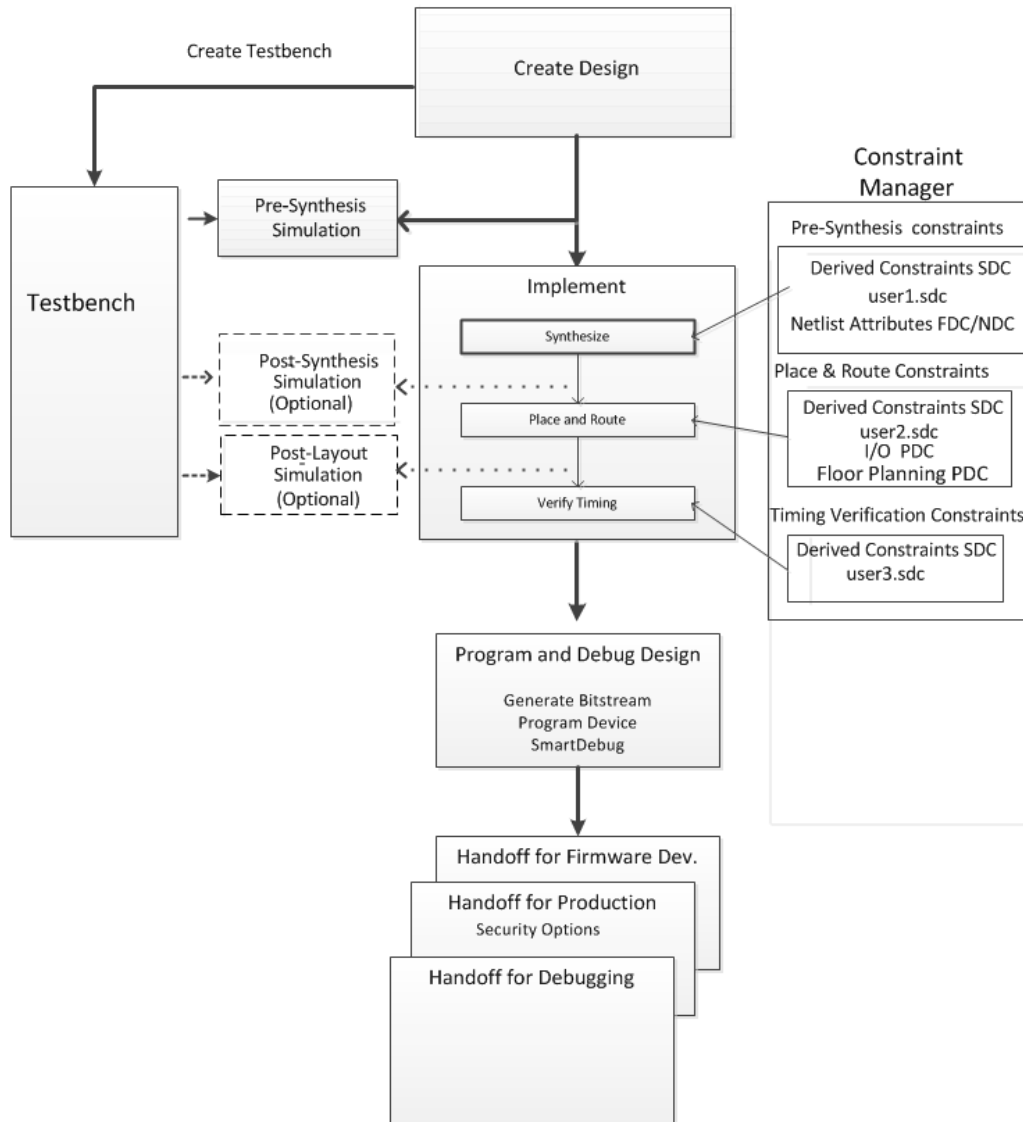


Figure 40 · Constraint Manager in Libero SoC Design Flow

Synthesis Constraints

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

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

Synplify Netlist Constraints (*.fdc)

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

Compile Netlist Constraints (*.ndc)

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

SDC Timing Constraints (*.sdc)

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

After the synthesis step, you may click **Edit with Constraint Editor > Edit Synthesis Constraints** to open the [Timing Constraints Editor](#) to edit existing constraints or add new SDC constraints.

Derived Timing Constraints (*.sdc)

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

Place and Route Constraints

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

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

I/O PDC Constraints

These are I/O Physical Design Constraints in an *.io.pdc file. From the Constraint Manager I/O Attribute tab, you may import (**I/O Attributes > Import**) or create in the Text Editor (**I/O Attributes > New**) an *.io.pdc file. Click the checkbox under Place and Route to associate the file with Place and Route.

Floorplanning PDC Constraints

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

Timing SDC Constraint file (*.sdc)

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

Timing Verifications Constraints

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

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

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

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

Constraint Manager Components

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

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

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

Constraint File and Tool Association

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

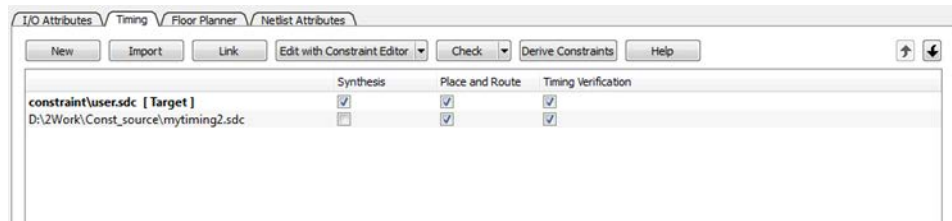




Figure 41 · Constraint File and Tool Association

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

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

- I/O Editor
- Chip Planner
- Constraint Editor

The file type, file extension, location, tool association and Interactive Tool (for editing) of the constraints are listed below.

Constraint Type	Constraint File Extension	Location inside Project	Associated with Design Tool	Interactive Tool (For Editing)
I/O Attributes	PDC (*.pdc)	<proj>\constraints\io*.pdc	Place and Route	I/O Editor
	PDC (*.pdc)		Place and	Chip

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

User Actions on Constraint File

Right-click a Constraint File and drop-down menu items appear:

- Set/Unset as Target – Set/Unset a constraint file as Target. A constraint file, when set as target, receives and stores new constraints from the Interactive Tools: I/O Editor, Chip Planner, and Constraint Editor. When a constraint is added, and then saved in the Interactive Tool, the constraint file set as target is updated to store the new constraints added in the Interactive Tool. When no constraint file is associated with a tool or no constraint file is set as target, the Constraint Manager creates a new constraint file (user.sdc or user.pdc) to store the newly-added constraints from the Interactive Tool and set the new constraint file as Target.

Note: Netlist Attribute constraint (*.fdc and *.ndc) file cannot be Set as Target. Only SDC (*.sdc) and PDC (*.pdc) files can be Set as Target.

- Open in Text Editor – Opens the selected Constraint File in Libero SoC's Text Editor for Editing.
- Clone – Save the selected constraint file with a new name. The original file remains intact.
- Rename – Rename the selected constraint file to a new name. The original file is deleted.
- Copy File Path – Copy the file path to the clipboard.
- Delete from Project and Disk – Delete the file physically from Disk and Project.

Note: Deleting a constraint file associated with a tool invalidates the tool state in the Design Flow Window.

Note: Using the Text Editor to change a constraint file associated with a tool invalidates the tool state in the Design Flow Window.

Derive Constraints in Timing Tab

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

To generate SDC timing constraints for IP cores:

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

click **Yes** to accept automatic association or **No** and then check or uncheck the appropriate checkbox for tool association.

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

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

Note: The Derive Constraints button is available for HDL-based, SmartDesign-based, and System Builder-based design flows. It is not available if the design source is EDIF/EDN.

Create New Constraints

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

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

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

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

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

- I/O Attributes constraint file (<proj>\io*.pdc) in the I/O Attributes tab
- Floorplanning constraints (<proj>\fp*.pdc) in the Floor Planner tab
- Timing constraints (<proj>\constraints*.sdc) in the Timing tab

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

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

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

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

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

1. Select the Tab that corresponds to the type of constraint you want to create.
2. Click Edit to open the Interactive Tools. The Interactive Tool that Libero SoC opens varies with the constraint type:
 - I/O Editor to edit/create I/O Attribute Constraints
 - Chip Planner to edit/create Floorplanning constraints
 - Constraint Editor to edit/create Timing Constraints
3. Create the Constraints in the Interactive Tool. Click **Commit and Save**.
4. Check that Libero SoC creates these file to store the new constraints:

- Constraints\io\user.pdc file when I/O constraints are added in I/O Editor.
- Constraints\fp\user.pdc file when floorplanning constraints are added in Chip Planner.
- Constraints\user.sdc file when Timing Constraints are added in Constraint Editor

Constraint File Order

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

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

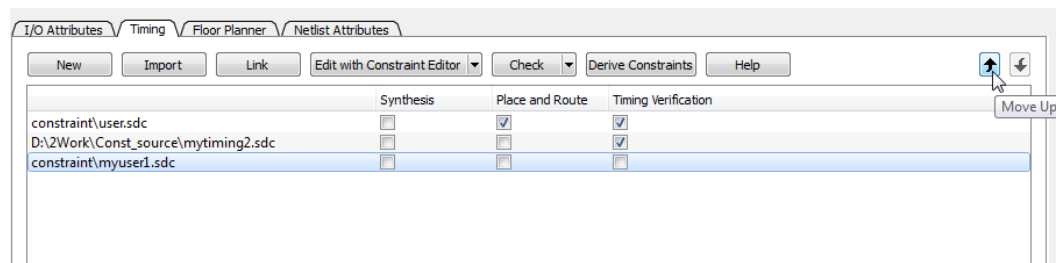


Figure 42 · Move constraint file Up or Down

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

Import a Constraint File

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

To import a constraint file:

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

Link a Constraint File

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

To link a constraint file:

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

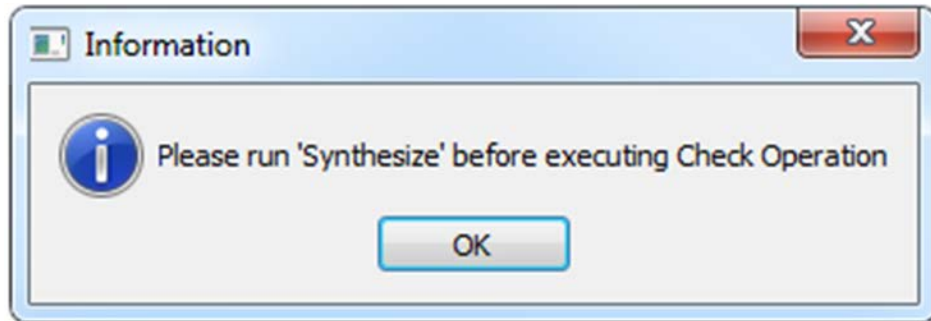
Check a Constraint File

Use the Constraint Manager to check a constraint file.

To check a constraint file:

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

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



All constraint files associated with the tool are checked. Files not associated with a tool are not checked. For Timing Constraints, select from the Check drop-down menu one of the following:

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

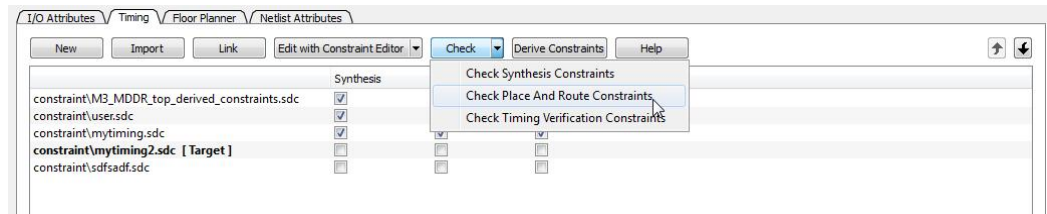


Figure 43 · Check Constraints

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

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

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

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

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

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

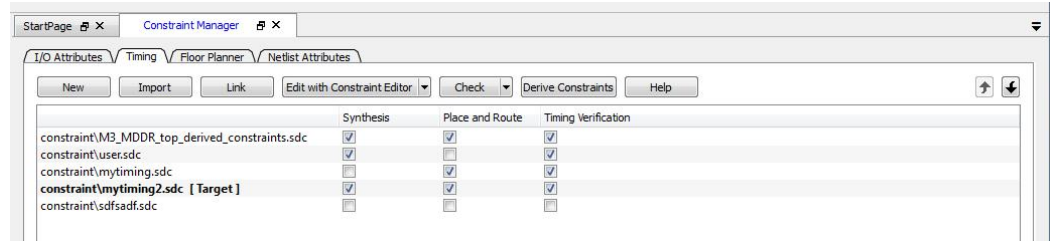


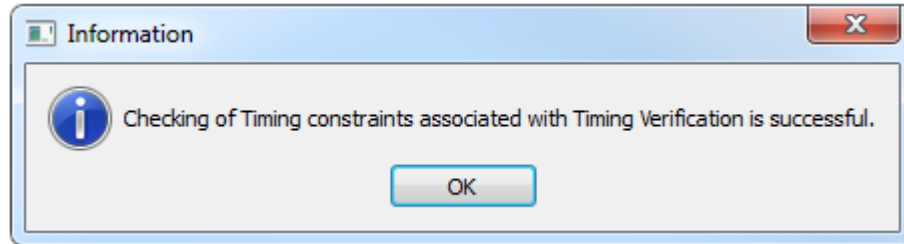
Figure 44 · Timing Constraints SDC file and Tool Association

When a constraint file is checked, the Constraint Manager:

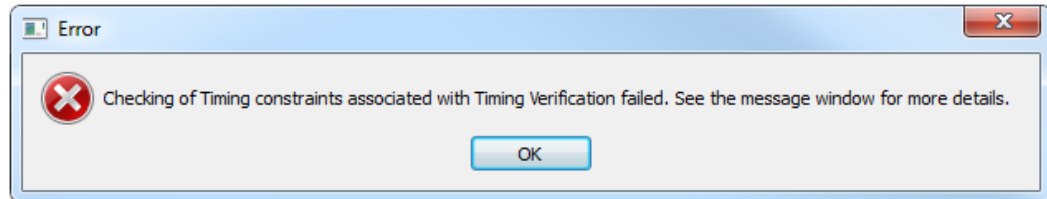
- Checks the SDC or PDC syntax.
- Compares the design objects (pins, cells, nets, ports) in the constraint file versus the design objects in the netlist (RTL or post-layout ADL netlist). Any discrepancy (e.g. constraints on a design object which does not exist in the netlist) are flagged as errors and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

Check Result

If the check is successful, this message pops up.



If the check fails, this error message pops up.



Constraint Type	Check for Tools	Required Design State Before Checks	Netlist Used for Design Objects Checks	Check Result Details In
I/O Constraints	Place and Route	Post-Synthesis	ADL Netlist	Libero Message Window
Floorplanning Constraints	Place and Route	Post-Synthesis	ADL Netlist	Libero Message Window
Timing Constraints	Synthesis	Pre-Synthesis	RTL Netlist	synthesis_sdc_check.log
	Place and	Post-	ADL Netlist	placer_sdc_check.log

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

Edit a Constraint File

The Edit button in the Constraint Manager allows you to:

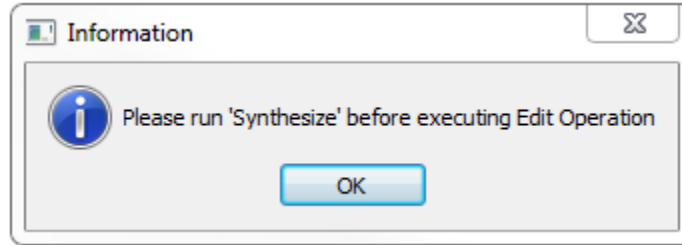
- Create new constraint files. See [To create new constraints from the Constraint Manager using the Text Editor](#) for details.
- Edit existing constraint files.

To edit a constraint file

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

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

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



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

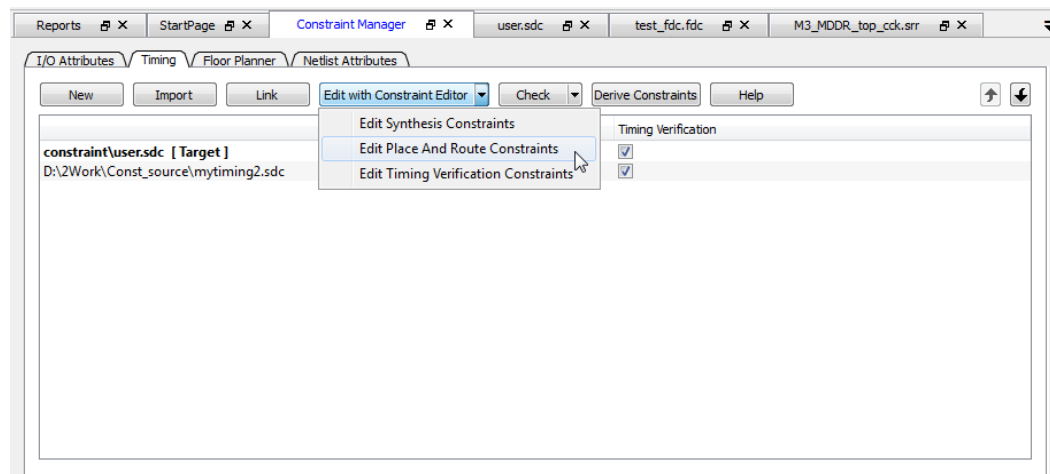


Figure 45 · Edit Drop-down Menu

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

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

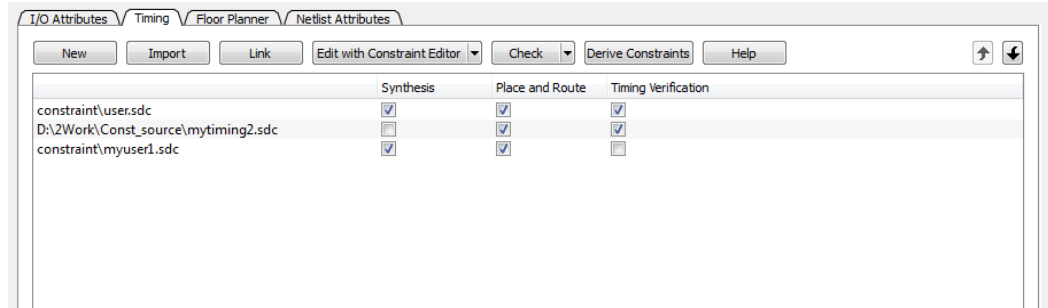


Figure 46 · Timing Constraint File and Tool Association

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

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

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

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

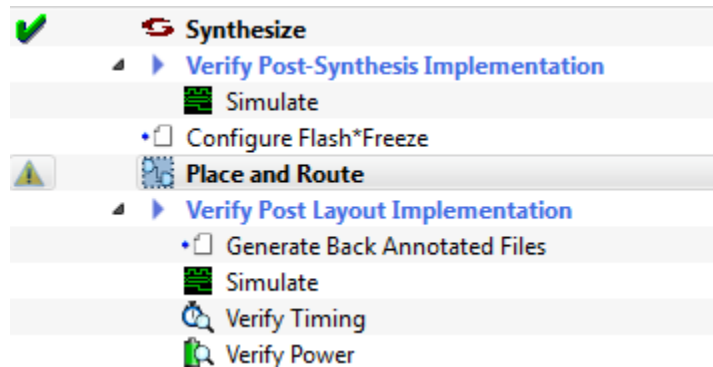


Figure 47 · Place and Route Invalidation

Timing Constraints Editor

The Timing Constraints Editor enables you to create, view, and edit timing constraints. This editor includes powerful visual dialogs that guide you toward capturing your timing requirements and timing exceptions quickly and correctly.

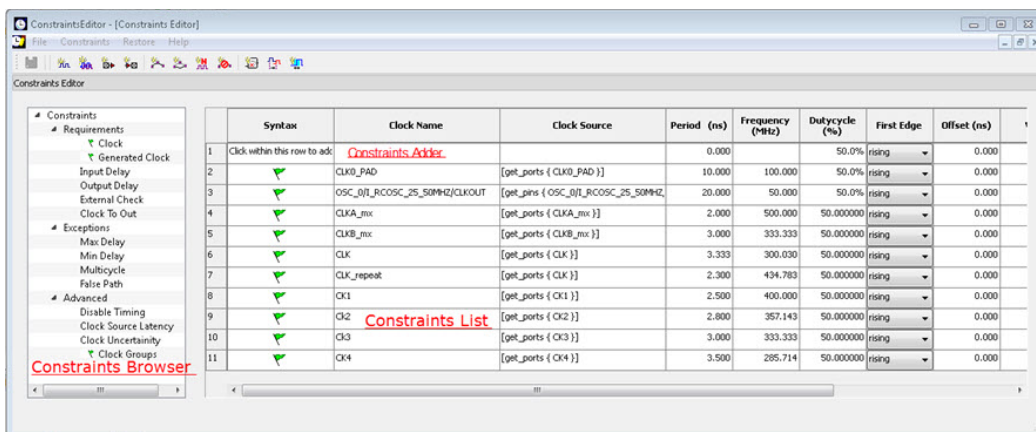


Figure 48 · Constraints Editor

The Constraints Editor window is divided into a Constraint Browser, Constraint List, and a Constraint Adder.

Constraints Browser

The Constraint Browser categorizes constraints based on three types of Constraints:

- **Requirements** – General constraints to meet the design’s timing requirements and specifications. Examples are clock constraints and generated clock constraints.
- **Exceptions** – Constraints on certain timing paths for special considerations by SmartTime. Examples are false path constraints and multicycle path constraints.
- **Advanced** – Special timing constraints such as clock latency and clock groups

Constraints List

This is a spreadsheet-like list of the constraints with detailed values and parameters of the constraint displayed in individual cells. You may click on individual cells of the spreadsheet to change the values of the constraint parameters.

Constraints Adder

This is the first row of the spreadsheet-like constraint list. There are 2 ways of adding a constraint from this row. User can right click on the row, and select Add Constraint to add a constraint of the same type to the Constraint List. This method will invoke the specific add constraint dialog.

Alternatively, user can select a cell by clicking in it. Then follow by double-clicking and start typing text. This method of creating a constraint is targeted at the experienced user who knows the design well, and need not rely on the dialog box for guidance.



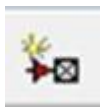







You can perform the following tasks in the Constraints View:

- Select a constraint type from the Constraint Browser and create or edit the constraint.
- Add a new constraint and check the syntax.
- Right-click a constraint in the Constraints List to edit or delete.
- Use the first row to create a constraint (as described above), and add it to the main table (list)

Constraint Icons

Across the top of the Constraint Editor is a list of icons you can click to add constraints. Tooltips are available to identify the constraints.

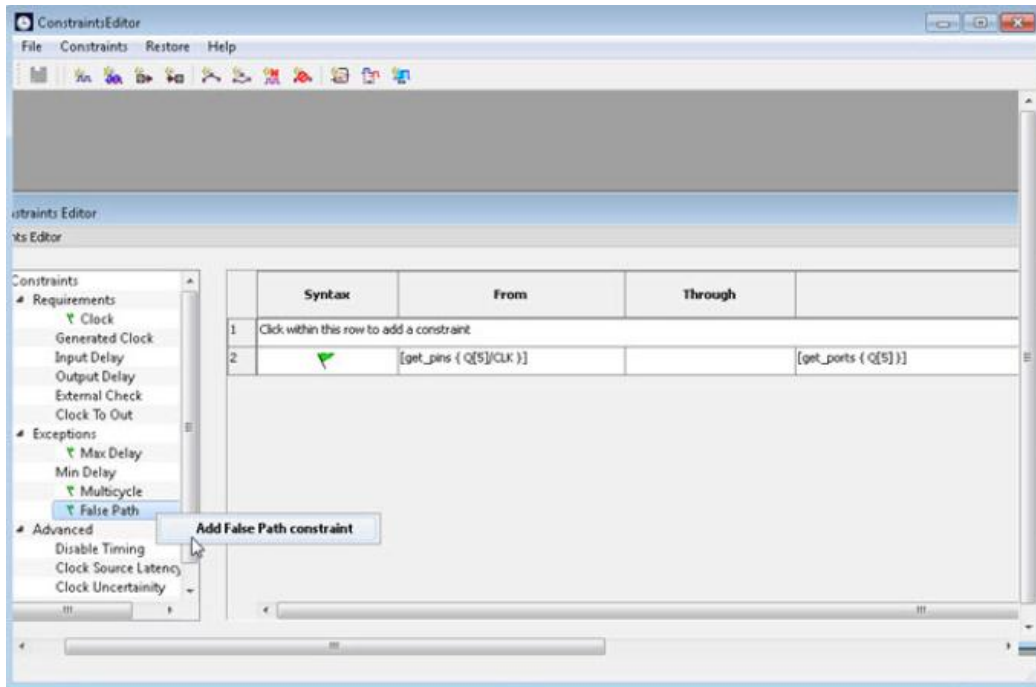
Icon	Name
	Add Clock Constraint

Icon	Name
	Add Generated Clock Constraint
	Add Input Delay Constraint
	Add Output Delay Constraint
	Add Maximum Delay Constraint
	Add Minimum Delay Constraint
	Add Multicycle Path Constraint
	Add False Path Constraint
	Add Disable Timing Constraint
	Add Clock Source Latency
	Add Clock to Clock Uncertainty

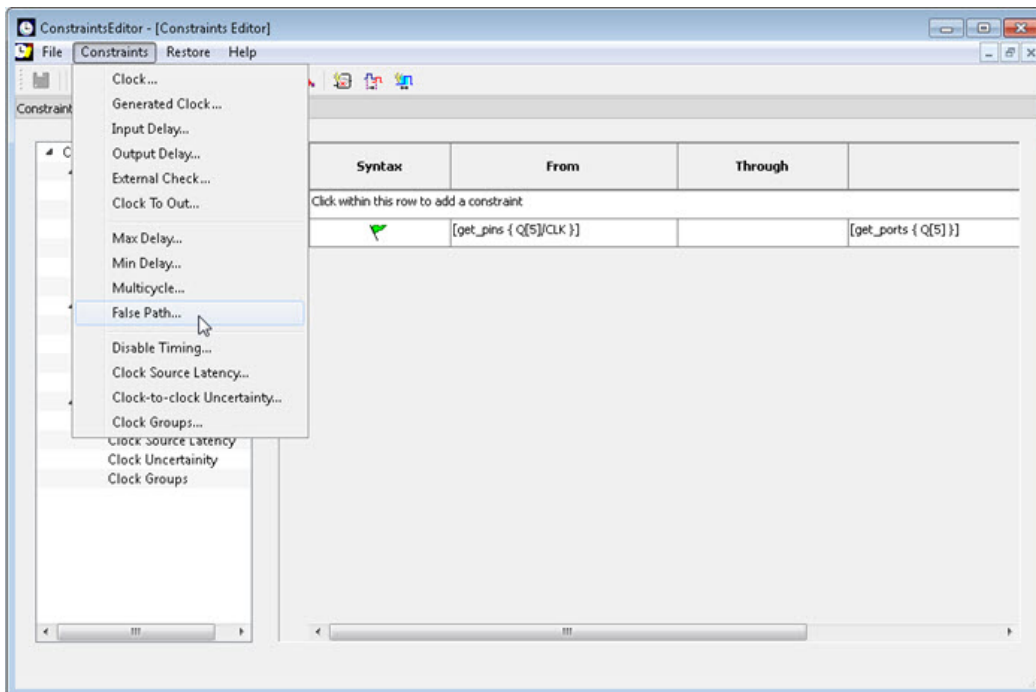
Adding Constraints

The Constraints Editor provides four ways to add Constraints. The Add Constraints dialog box appears when you add constraints in one of the following four ways:

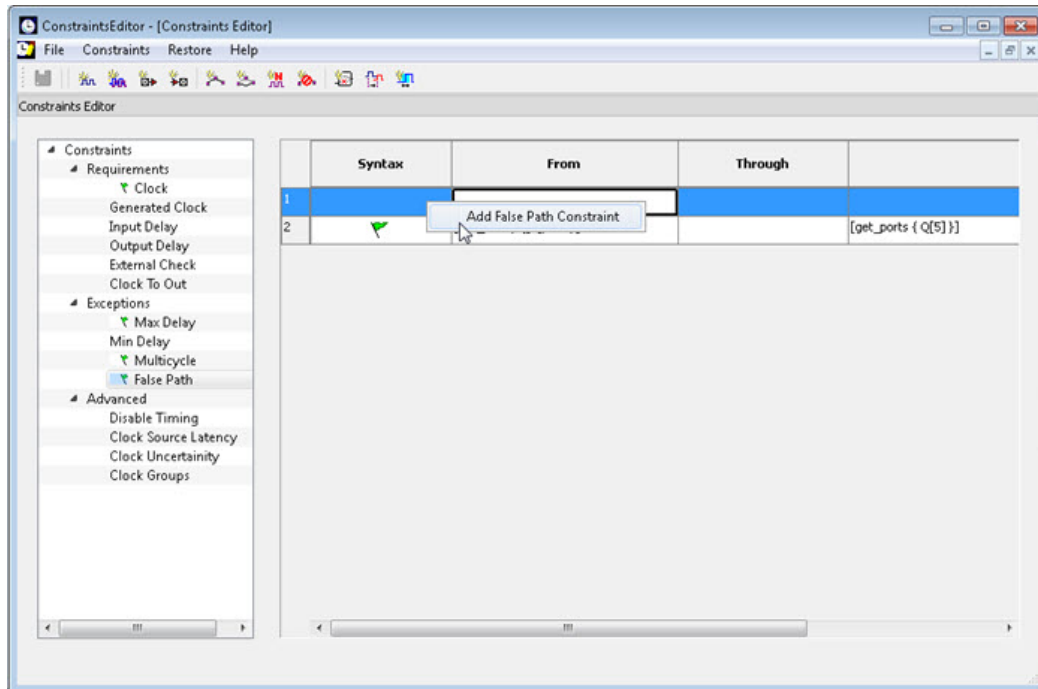
- Click the Add Constraint icon. Example: Click  to add False Path Constraints.
- From the Constraints Browser, choose the type of Constraints to add. Example: False Path



- Choose **False Path** from the Constraints drop-down menu (**Constraints > False Path**).



- Right-click the first row and choose **Add False Path Constraint**.




See Also

- [Set Clock Constraints](#)
- [Set Generated Clock Constraints](#)
- [Set Input Delay Constraints](#)
- [Set Output Delay Constraints](#)
- [Set External Check Constraints](#)
- [Set Clock to Out Constraints](#)
- [Set False Path Constraints](#)
- [Set Multicycle Path Constraints](#)
- [Set Minimum Delay Constraints](#)
- [Set Maximum Delay Constraints](#)
- [Set Disable Timing Constraint](#)
- [Set Clock to Clock Uncertainty Constraint](#)
- [Set Clock Source Latency Constraint](#)
- [Set Clock Groups Constraint](#)

Set Clock Constraints

Adding a clock constraint is the most effective way to constrain and verify the timing behavior of a sequential design. Use clock constraints to meet your performance goals.

To set a clock constraint, open the Create Clock Constraint dialog box in one of the following four ways:

- From the Constraints Browser, choose **Clock**.
- Double-click the Add Clock Constraint icon  .
- Choose **Clock** from the Constraints drop-down menu (**Constraints > Clock**).
- Right-click the first row or any other row (if they exist) in the Clock Constraints Table and choose **Add Clock Constraint**.

The Create Clock Constraint dialog box appears.

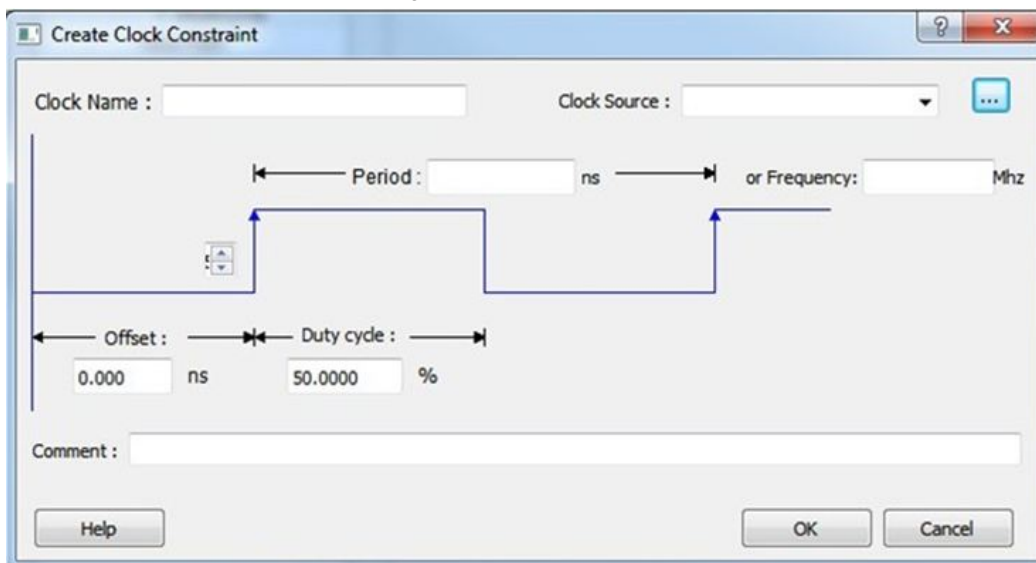


Figure 49 · Create Clock Constraint Dialog Box

Clock Name

Specifies the name of the clock constraint.

Clock Source

Select the pin to use as clock source. You can click the Browse button to display the [Select Source Pins for Clock Constraint Dialog Box](#).

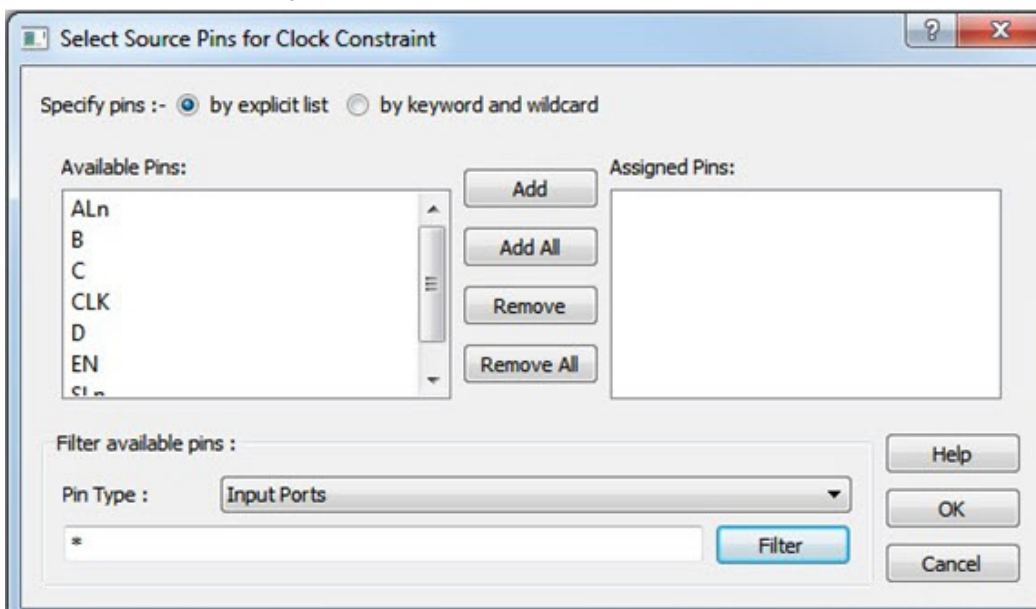


Figure 50 · Select Source Pin for Clock Constraint Dialog Box

The Pin Type options are:

- Input Ports
- All Pins

- All Nets

Use the Select Source Pin for Clock Constraint dialog box to display a list of source pins from which you can choose. By default, it displays the Input Ports of the design.

To choose other pin types in the design as clock source pins, click the drop-down and choose **Input Ports**, **All Pins**, or **All Nets**. To display a subset of the displayed clock source pins, you can create and apply a filter. The default filter is * (wild-card for all).

Click **OK** to save these dialog box settings.

Period/Frequency

Specifies the Period in nanoseconds (ns) or Frequency in MegaHertz (MHz). When you edit the period, the tool automatically updates the frequency value and vice versa. The frequency must be a positive real number. Accuracy is up to 3 decimal places.

Starting Clock Edge Selector

Click the Up or Down arrow to use the rising or falling edge as the starting edge for the created clock.

Offset

Indicates the shift (in nanoseconds) of the first clock edge with respect to instant zero common to all clocks in the design.

The offset value must be a positive real number. Accuracy is up to 2 decimal places. Default value is 0.

Duty Cycle

This number specifies the percentage of the overall period that the clock pulse is high. The duty cycle must be a positive real number. Accuracy is up to 4 decimal places. Default value is 50%.

Comment

Enter a single line of text that describes the clock constraints purpose.


See Also

[create_clock \(SDC\)](#)

Set Generated Clock Constraints

Use the generated clock constraint to define an internally generated clock for your design and verify its timing behavior. Use generated clock constraints and clock constraints to meet your performance goals.

To set a generated clock constraint, open the Create Generated Clock Constraint dialog box in one of the following four ways:

- From the Constraints Browser, choose **Generated Clock**.
- Double-click the Add Generated Clock Constraint icon .
- Choose **Generated Clock** from the Constraints drop-down menu (**Constraints > Generated Clock**).
- Right-click any row in the Generated Clock Constraints Table and choose **Add Generated Clock Constraint**.

The Create Generated Clock Constraint dialog box appears.

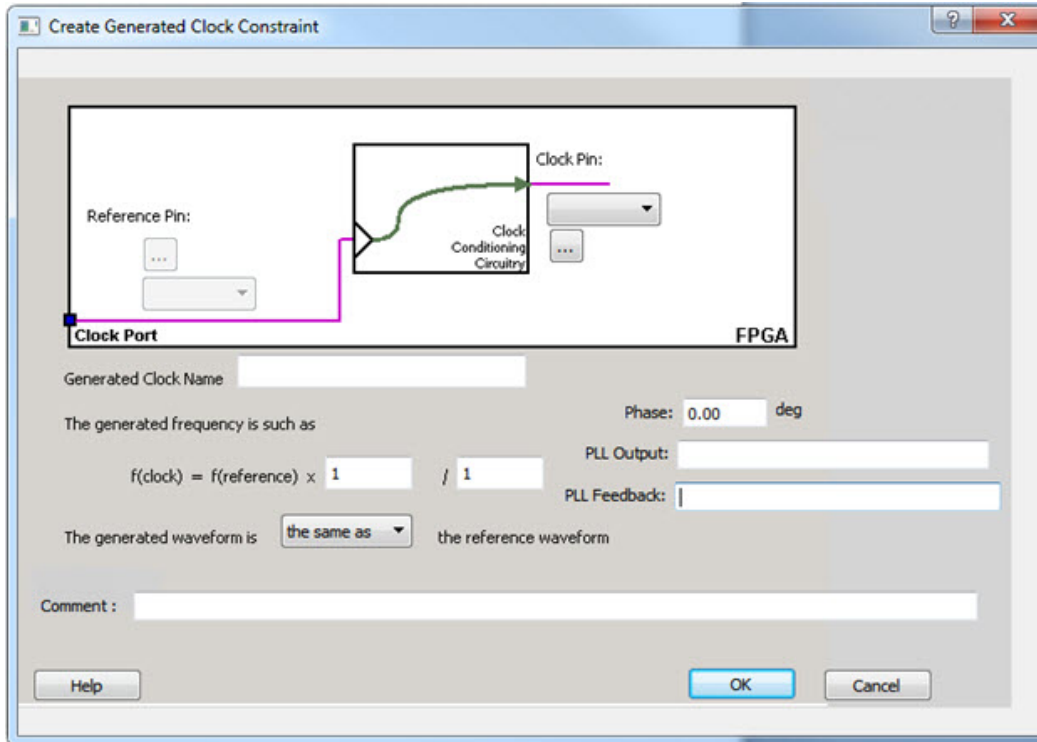


Figure 51 · Create Generated Clock Constraint Dialog Box

Clock Pin

Select a Clock Pin to use as the generated clock source. To display a list of the available generated clock source pins, click the Browse button. The Select Generated Clock Source dialog box appears.

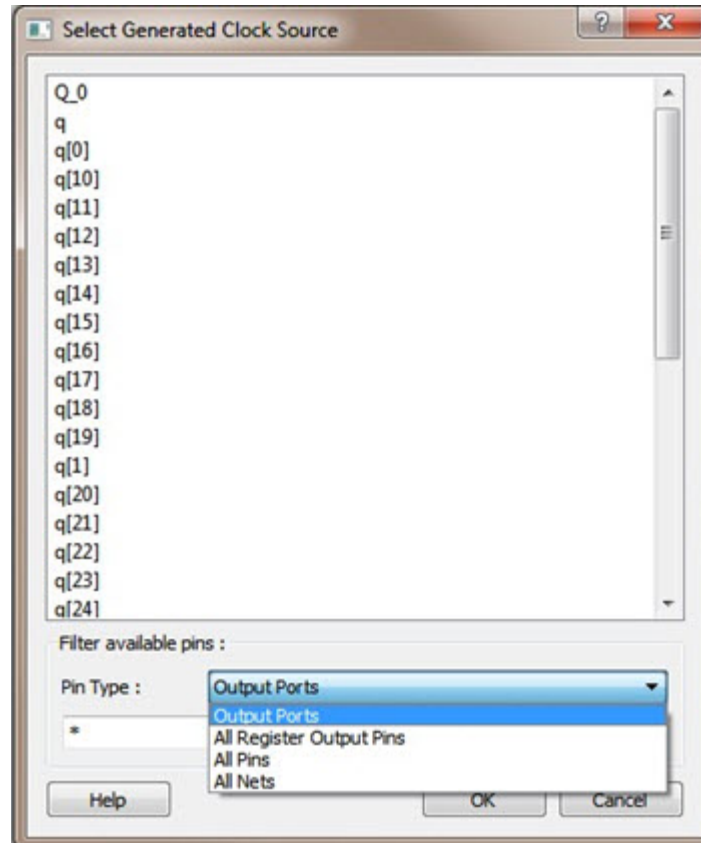


Figure 52 · Select Generated Clock Source Dialog Box

The Pin Type options for Generated Clock Source are:

- Output Ports
- All Register Output Pins
- All Pins
- All Nets

Click **OK** to save the dialog box settings.

Modify the Clock Name if necessary.

Reference Pin

Specify a Clock Reference. To display the list of available clock reference pins, click the Browse button. The Select Generated Clock Reference dialog box appears.

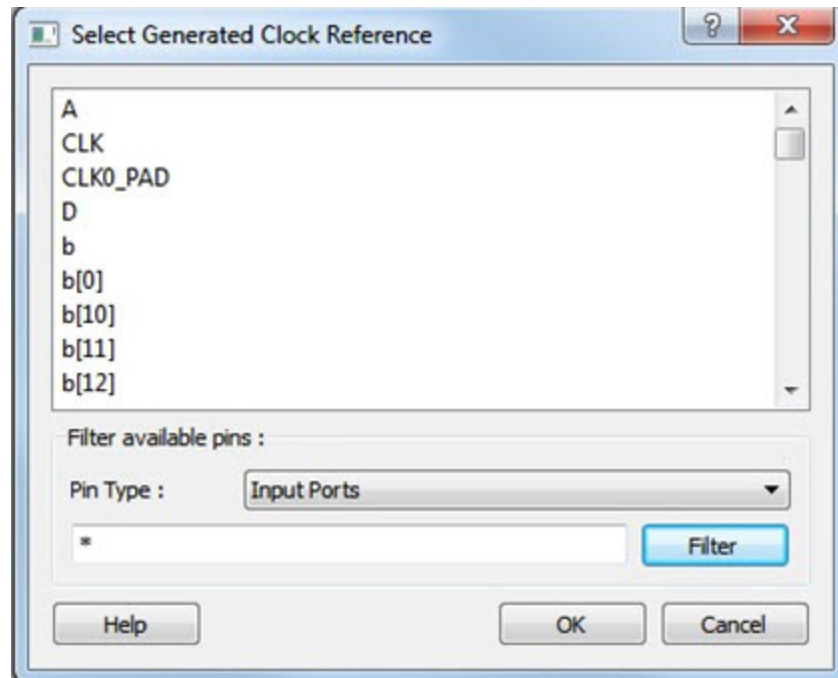


Figure 53 · Select Generated Clock Reference Dialog Box

The Pin Type options for Generated Clock Reference are:

- Input Ports
- All Pins

Click **OK** to save the dialog box settings.

Generated Clock Name

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

Generated Frequency

Specify the values to calculate the generated frequency: a multiplication factor and/or division factor (must be positive integers) is applied to the reference clock to compute the generated clock.

Generated Waveform

Specify whether the generated waveform is the same or inverted with respect to the reference waveform. Click **OK**.

Phase

This field is primarily used to report the information captured from the CCC configuration process, and when constraint is auto-generated. Meaningful phase values are: 0, 45, 90, 135, 180, 225, 270, and 315. This field is used to report the information captured from the CCC configuration process, and when the constraint is auto-generated.

PLL Output

This field refers to the CCC GL0/1/2/3 output that is fed back to the PLL (in the CCC). This field is primarily used to report the information captured from the CCC configuration process, and when constraint is auto-generated.

PLL Feedback

This field refers to the manner in which the GL/0/1/2/3 output signal of the CCC is connected to the PLL's FBCLK input. This field is primarily used to report the information captured from the CCC configuration process, and when constraint is auto-generated.

Comment

Enter a single line of text that describes the generated clock constraints purpose.


See Also

- [create_generated_clock \(SDC\)](#)
- [Specifying Generated Clock Constraints](#)
- [Select Generated Clock Source](#)

Set an Input Delay Constraint

Use the input delay constraint to define the arrival time of an input relative to a clock.

To specify an input delay constraint, open the Add Input Delay Constraint dialog box in one of the following four ways:

- From the Constraints Browser, choose **Input Delay**.
- Double-click the Add Input Delay Constraint icon .
- Choose **Input Delay** from the Constraints drop-down menu (**Constraints > Input Delay**).
- Right-click any row in the Input Delay Constraints Table and choose **Add Input Delay Constraint**.

The Add Input Delay Constraint dialog box appears.

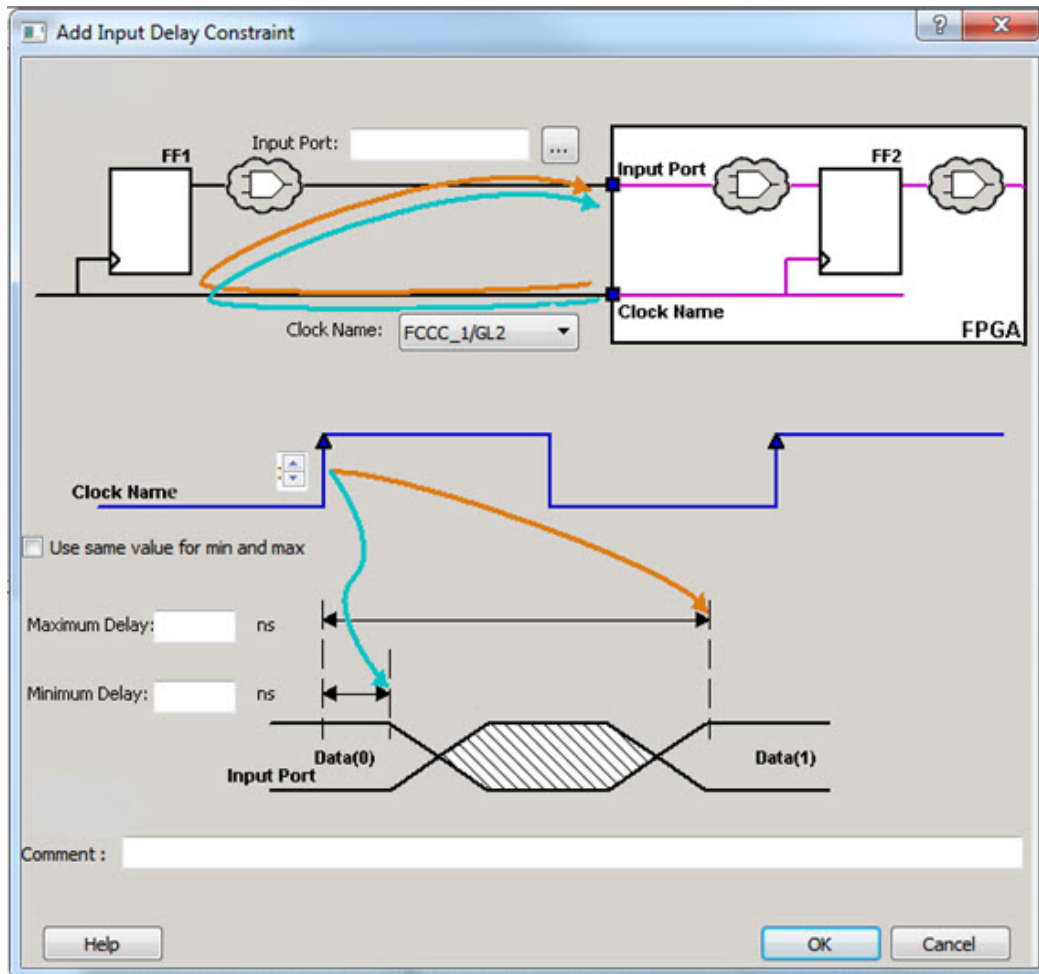


Figure 54 · Add Input Delay Constraint Dialog Box

The Input Delay Dialog Box enables you to enter an input delay constraint by specifying the timing budget outside the FPGA. You can enter the Maximum Delay, the Minimum Delay, or both.

Input Port

Specify the Input Port or click the browse button next to Input Port to display the Select Ports for Input Delay dialog box. You can select multiple input ports on which to apply the input delay constraint.

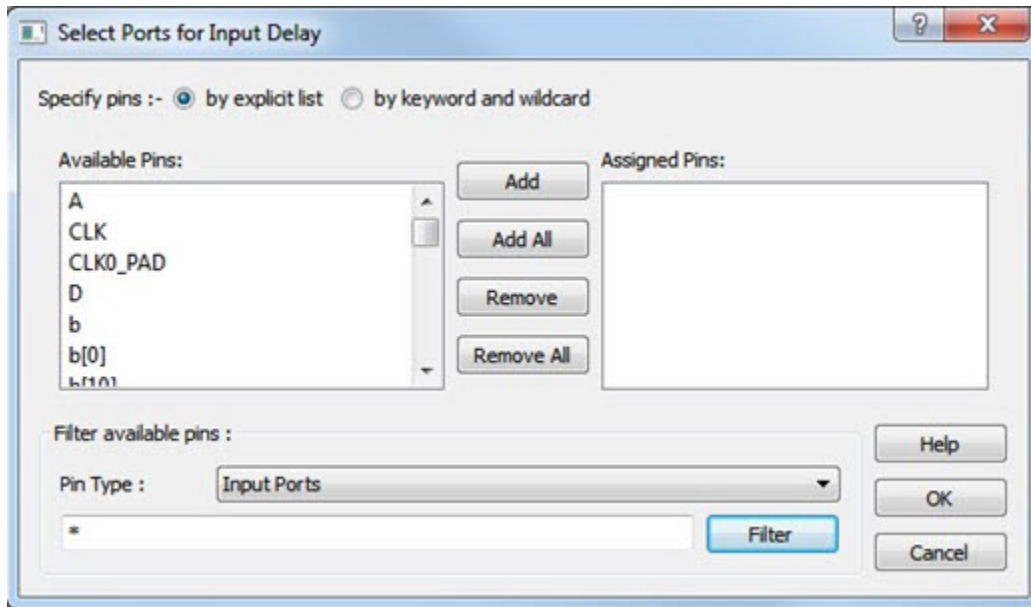


Figure 55 - Select Ports for Input Delay Dialog Box

There is only 1 Pin Type available for Input Delay: Input Ports.

Clock Name

Specifies the clock reference to which the specified input delay is based.

Clock edge

Select rising or falling as the launching edge of the clock.

Use same value for min and max

Specifies that the minimum input delay uses the same value as the maximum input delay.

Maximum Delay

Specifies that the delay refers to the longest path arriving at the specified input.

Minimum Delay

Specifies that the delay refers to the shortest path arriving at the specified input.

Comment

Enter a one-line comment for this constraint.


See Also

[set input delay \(SDC\)](#)

Set an Output Delay Constraint

Use the output delay constraints to define the output delay of an output relative to a clock.

To specify an output delay constraint, open the Add Output Delay Constraint Dialog box in one of the following four ways:

- From the Constraints Browser, choose **Output Delay**.
- Double-click the Add Output Delay Constraint icon  .
- Choose Output Delay from the Constraints drop-down menu (**Constraints > Output Delay**).
- Right-click any row in the Output Delay Constraints Table and choose **Add Output Delay Constraint**.

The Add Output Delay Constraint dialog box appears.

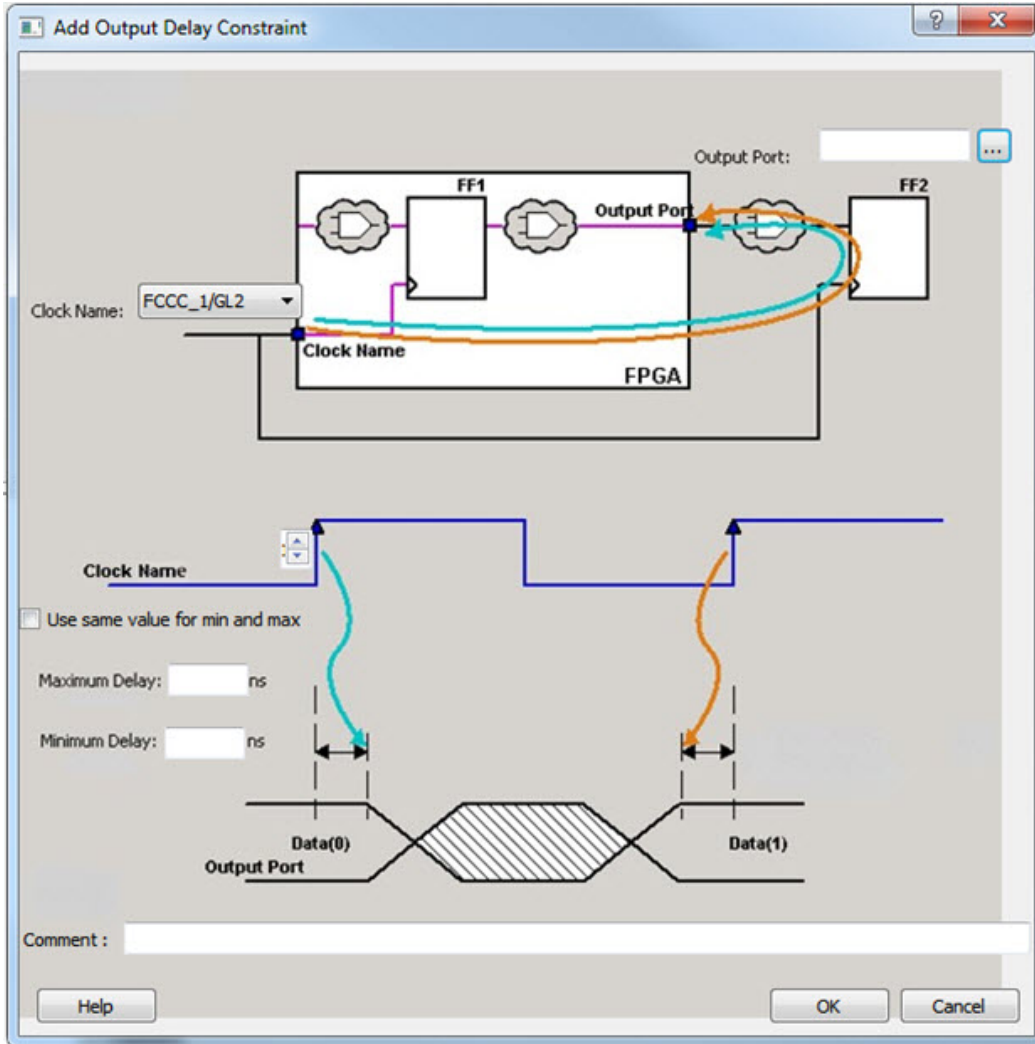


Figure 56 · Add Output Delay Constraint Dialog Box

The Output Delay dialog box enables you to enter an output delay constraint by specifying the timing budget outside the FPGA. You can enter the Maximum Delay, the Minimum Delay, or both.

Enter the name of the Output Port or click the browse button to display the Select Ports for Output Delay dialog box.

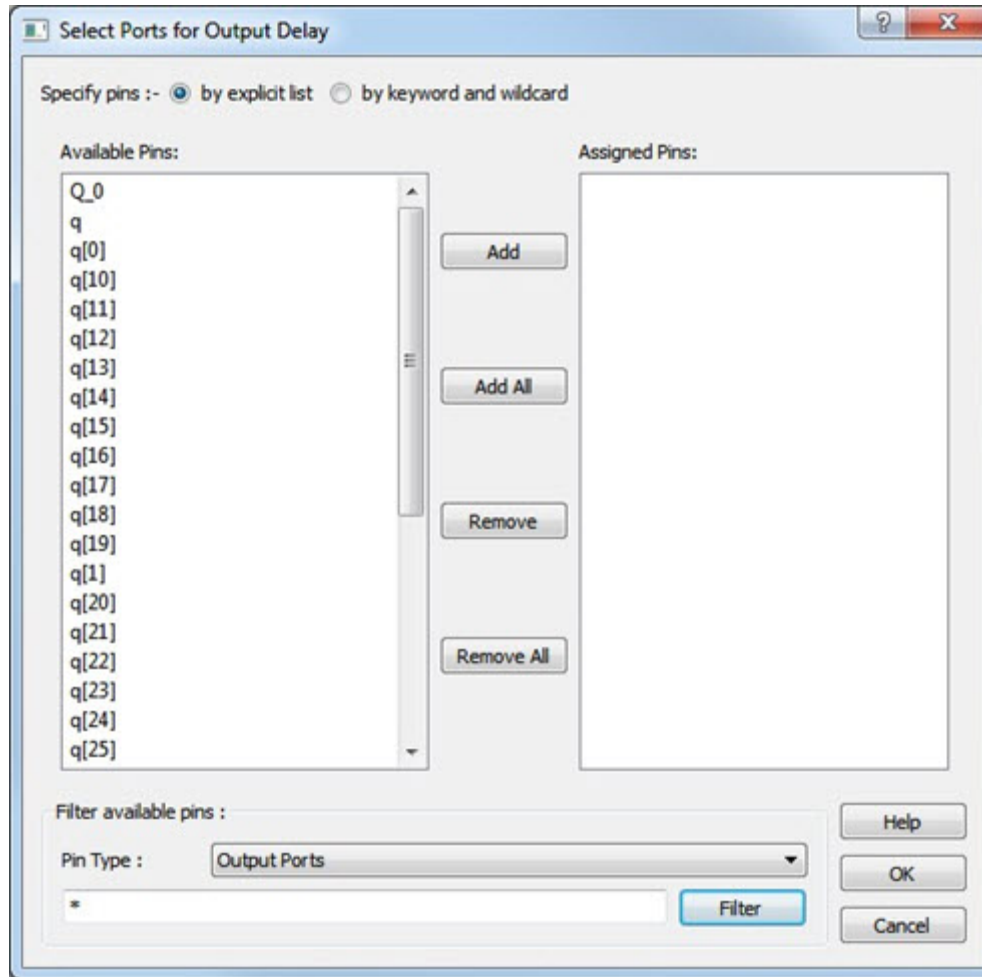


Figure 57 · Output Delay Dialog Box

There is only 1 Pin Type available for Output Delay: Output Ports

Output Port

Specifies a list of output ports in the current design to which the constraint is assigned. You can select multiple output ports to apply the output delay constraints.

Clock Name

Specifies the clock reference to which the specified output delay is related.

Clock edge Selector

Use the Up or Down arrow to select the rising or falling edge as the launching edge of the clock.

Use Same Value for Min and Max

Check this checkbox to use the same delay value for Min and Max delay.

Maximum Delay

Specifies the delay in nanoseconds for the longest path from the specified output to the captured edge. This represents a combinational path delay to a register outside the current design plus the library setup time.

Minimum Delay

Specifies the delay in nanoseconds for the shortest path from the specified output to the captured edge. This represents a combinational path delay to a register outside the current design plus the library hold time.

Comment

Enter a one-line comment for the constraint.

See Also

[set_output_delay \(SDC\)](#)

Set an External Check Constraint

Use the Add External Check Constraint to specify the timing budget inside the FPGA.

To specify an External Check constraint, open the Add External Check Constraint dialog box in one of the following three ways:

- From the Constraints Browser, choose **External Check**.
- Choose **External Check** from the Constraints drop-down menu (**Constraints > External Check**).
- Right-click any row in the External Check Constraints Table and choose **Add External Check Constraint**.

The Add External Check Constraint dialog box appears.

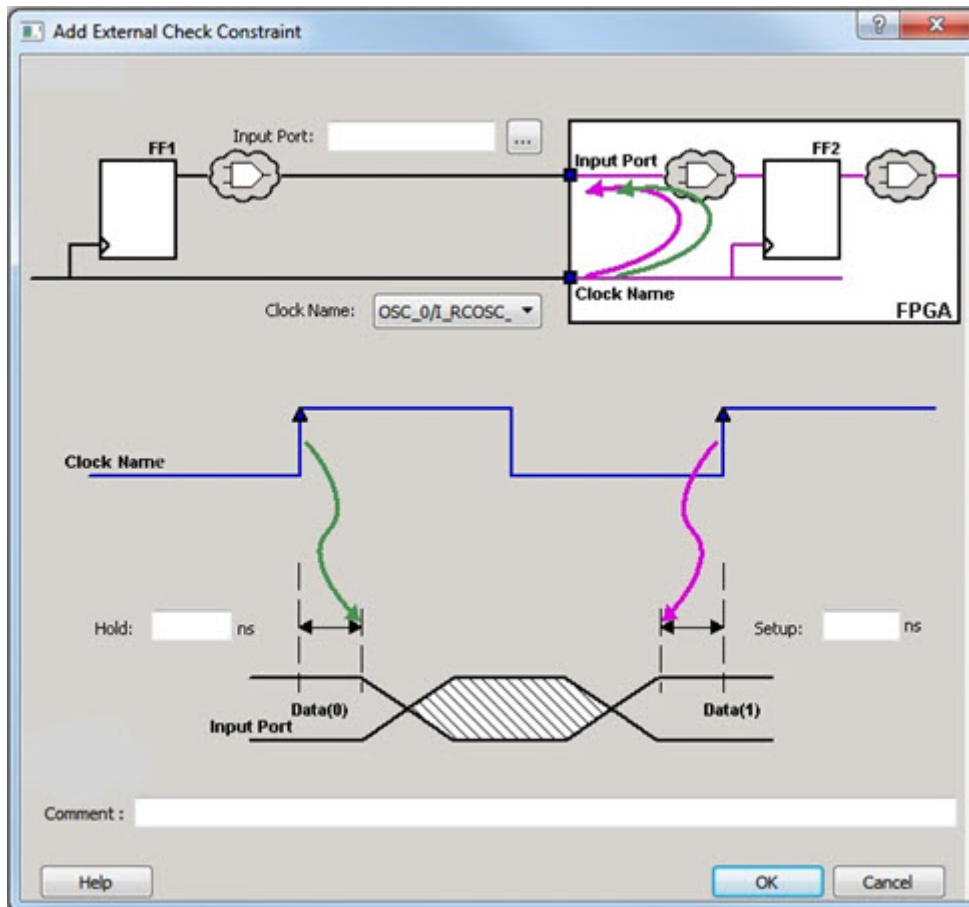


Figure 58 · Add External Check Constraint Dialog Box

Input Port

Specify the Input Port or click the browse button next to Input Port to display the Select Ports for External Check dialog box. You can select multiple input ports on which to apply the External Check constraint.

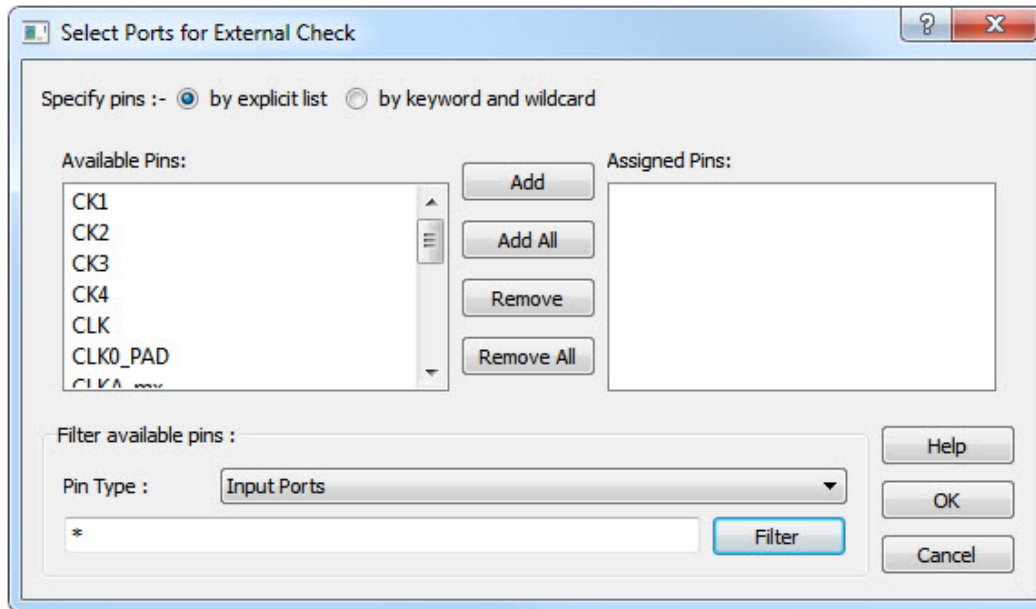


Figure 59 · Select Ports for External Check Dialog Box

Clock Name

Specifies the clock reference to which the specified External Check is related.

Hold

Specifies the external hold time requirement in nanoseconds for the specified input ports.

Setup

Specifies the external setup time requirement in nanoseconds for the specified input ports.

Comment

Enter a one-line comment for this constraint.

See Also

[set_external_check](#)

Set Clock To Out Constraint

Enter a clock to output constraint by specifying the timing budget inside the FPGA.

To specify a Clock to Out constraint, open the Add Clock to Out Constraint Dialog box in one of the following three ways:

- From the Constraints Browser, choose **Clock to Out**.
- Choose **Clock to Out** from the Constraints drop-down menu (**Constraints > Clock to Out**).
- Right-click any row of the Clock To Out Constraints Table and choose **Add Clock to Out Constraint**.

The Add Clock To Out Constraint dialog appears.

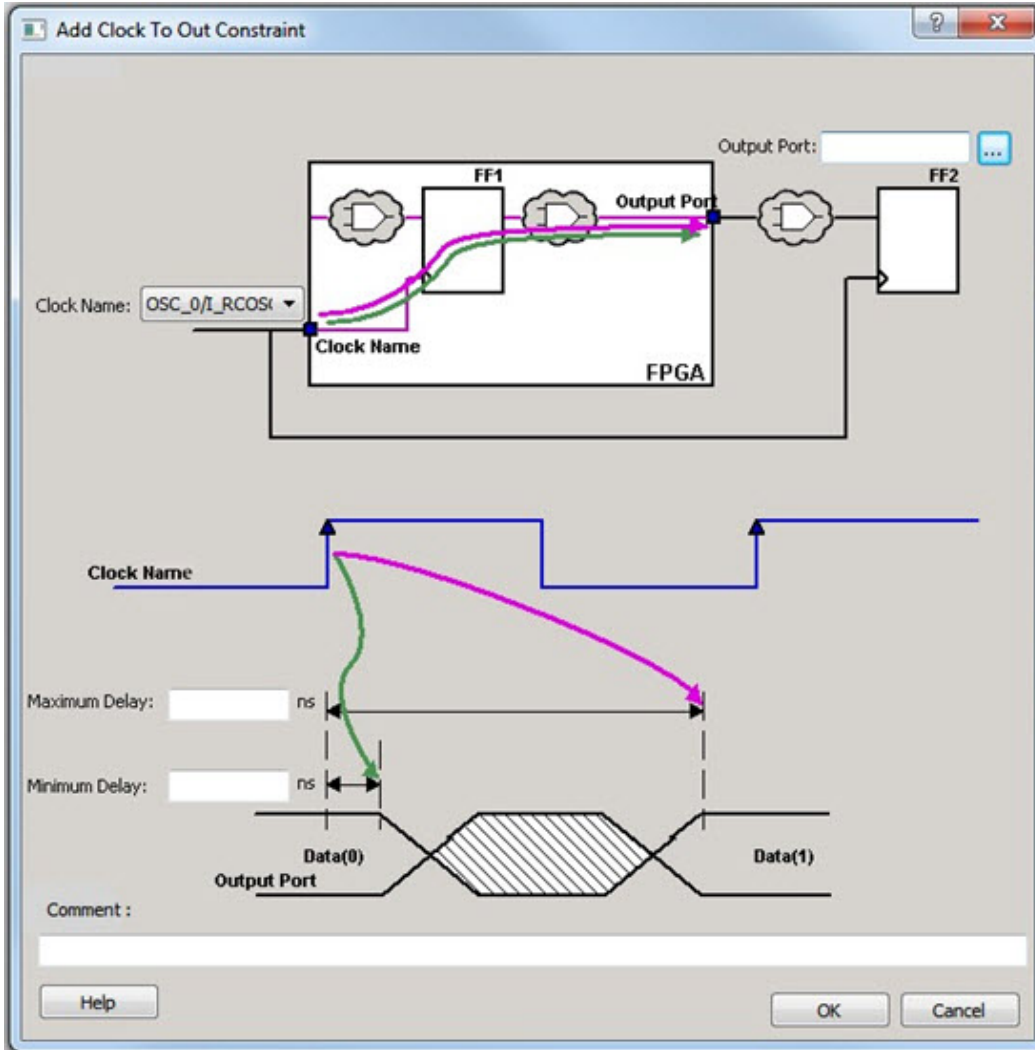


Figure 60 · Add Clock to Out Constraint Dialog Box

Specify the Output Port or click the browse button to display the Select Ports for Clock to Output dialog box. You can select multiple output ports on which to apply the Clock to Output constraint. Click the browse button next to Output Port to open the Select Ports for Clock To Output dialog box.

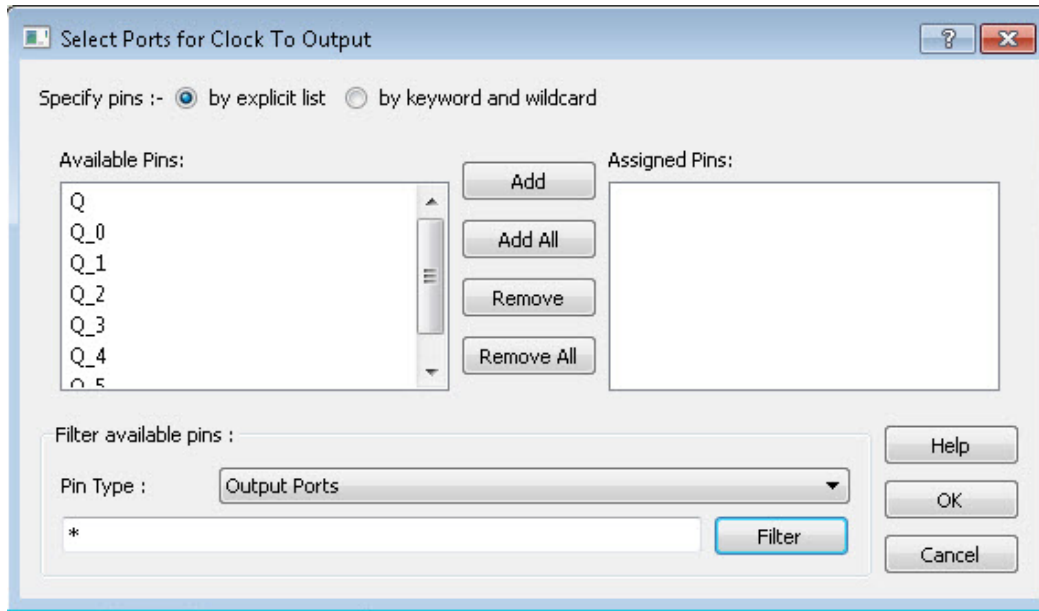


Figure 61 · Select Ports for Clock To Output Dialog Box

Clock Name

Specifies the clock reference to which the specified Clock to Out delay is related.

Maximum Delay

Specifies the delay in nanoseconds for the longest path from the specified output to the captured edge. This represents a combinational path delay to a register outside the current design plus the library setup time.

Minimum Delay

Specifies the delay in nanoseconds for the shortest path from the specified output to the captured edge. This represents a combinational path delay to a register outside the current design plus the library hold time.

Comment

Enter a one-line comment for this constraint.

See Also

[set_clock_to_output](#)

Set a Maximum Delay Constraint

Set the options in the Maximum Delay Constraint dialog box to relax or to tighten the original clock constraint requirement on specific paths.

SmartTime automatically derives the individual maximum delay targets from clock waveforms and port input or output delays. So the maximum delay constraint is a timing exception. This constraint overrides the default single cycle timing relationship for one or more timing paths. This constraint also overrides a multiple cycle path constraint.

Note: When the same timing path has more than one timing exception constraint, SmartTime honors the timing constraint with the highest precedence and ignores the other timing exceptions according to the order of precedence shown.

Timing Exception Constraints	Order of Precedence
set_disable_timing	1
set_false_path	2
set_maximum_delay/set_minimum_delay	3

Timing Exception Constraints	Order of Precedence
set_multicycle_path	4

Note: The set_maximum_delay_constraint has a higher precedence over set_multicycle_path constraint and therefore the former overrides the latter when both constraints are set on the same timing path.

To set a Maximum Delay constraint, open the Set Maximum Delay Constraint Dialog box in one of the following four ways:

- From the Constraints Browser, choose **Max Delay**.
- Double-click the Add Max Delay Constraint icon 
- Choose **Max Delay** from the Constraints drop-down menu (**Constraints > Max Delay**).
- From the Max Delay Constraints Table, right-click any row and choose **Add Maximum Delay Constraint**.

The Set Maximum Delay Constraint dialog box appears.

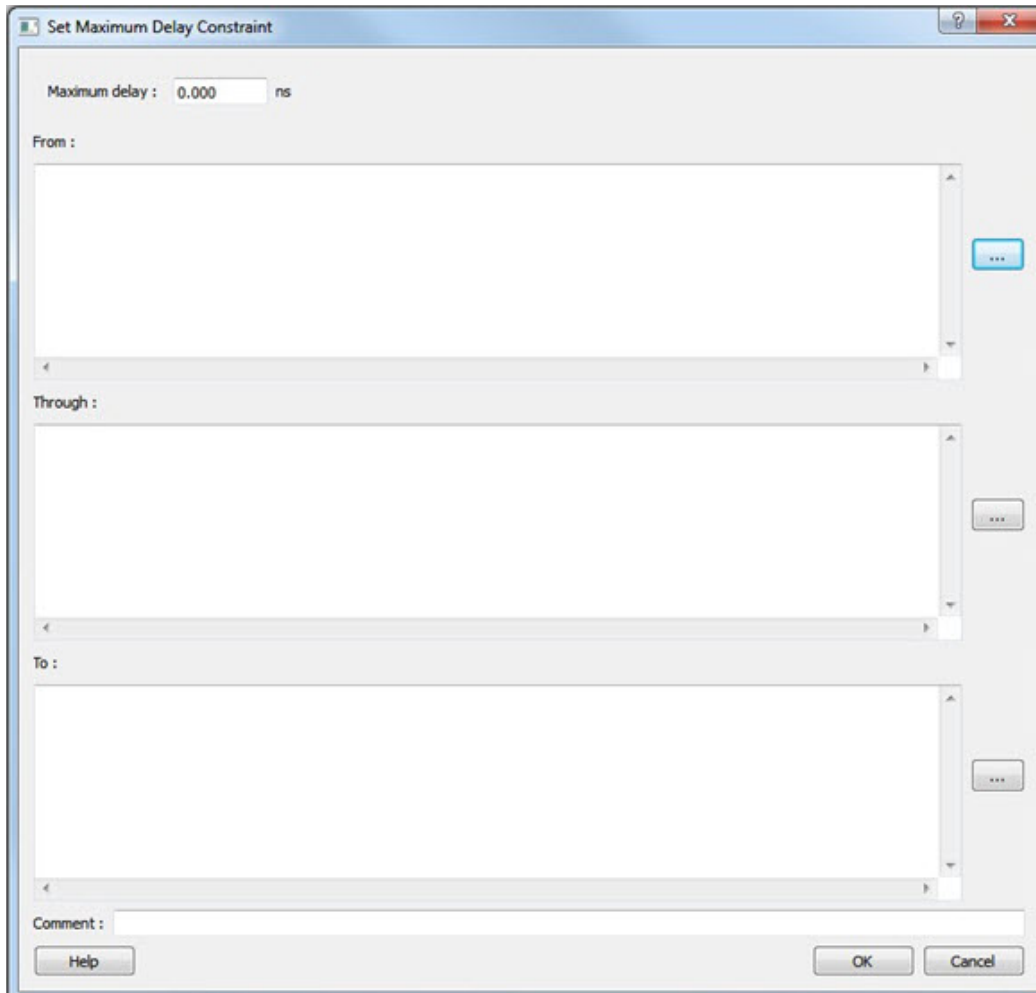


Figure 62 · Set Maximum Delay Constraint Dialog Box

Maximum Delay

Specifies a floating point number in nanoseconds that represents the required maximum delay value for specified paths.

If the path starting point is on a sequential device, SmartTime includes clock skew in the computed delay.

If the path starting point has an input delay specified, SmartTime adds that delay value to the path delay.

If the path ending point is on a sequential device, SmartTime includes clock skew and library setup time in the computed delay.

If the ending point has an output delay specified, SmartTime adds that delay to the path delay.

Source/From Pins

Specifies the starting points for max delay constraint path. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

To specify the Source pins(s), click on the Browse button to open the Select Source Pins for Max Delay Constraint dialog box.

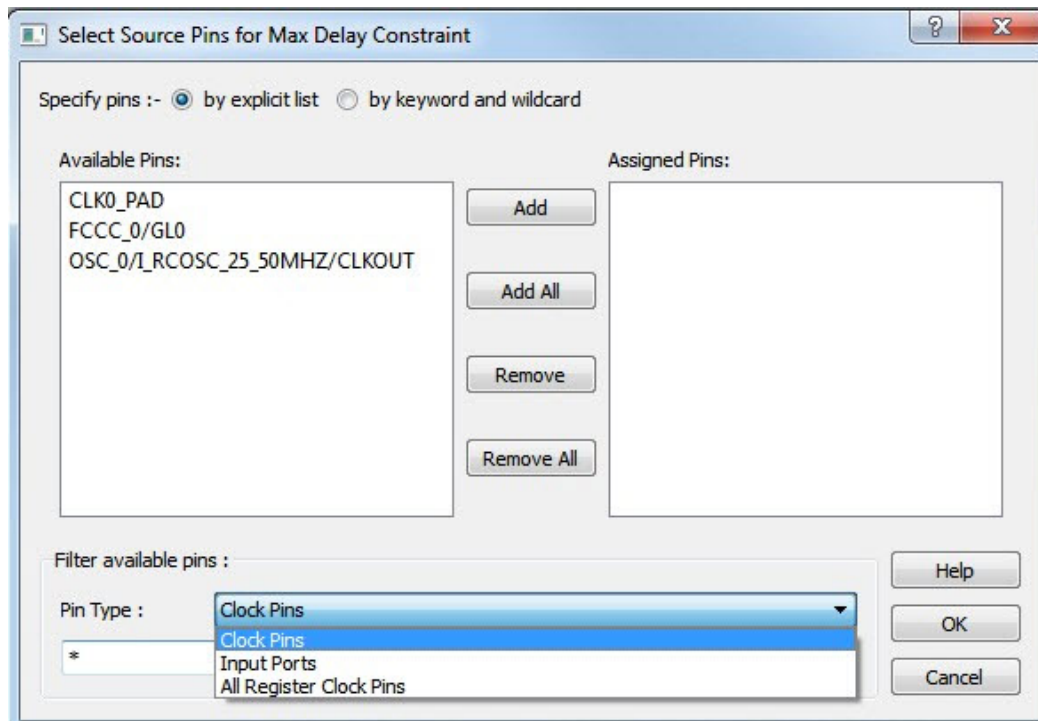


Figure 63 · Select Source Pins for Max Delay Constraint Dialog Box

The Pin Type options for Source Pins are:

- Clock Pins
- Input Ports
- All Register Clock Pins

Through Pins

Specifies the through pins in the specified path for the Maximum Delay constraint.

To specify the Through pin(s), click on the browse button next to the “Through” field to open the Select Through Pins for Max Delay Constraint dialog box.

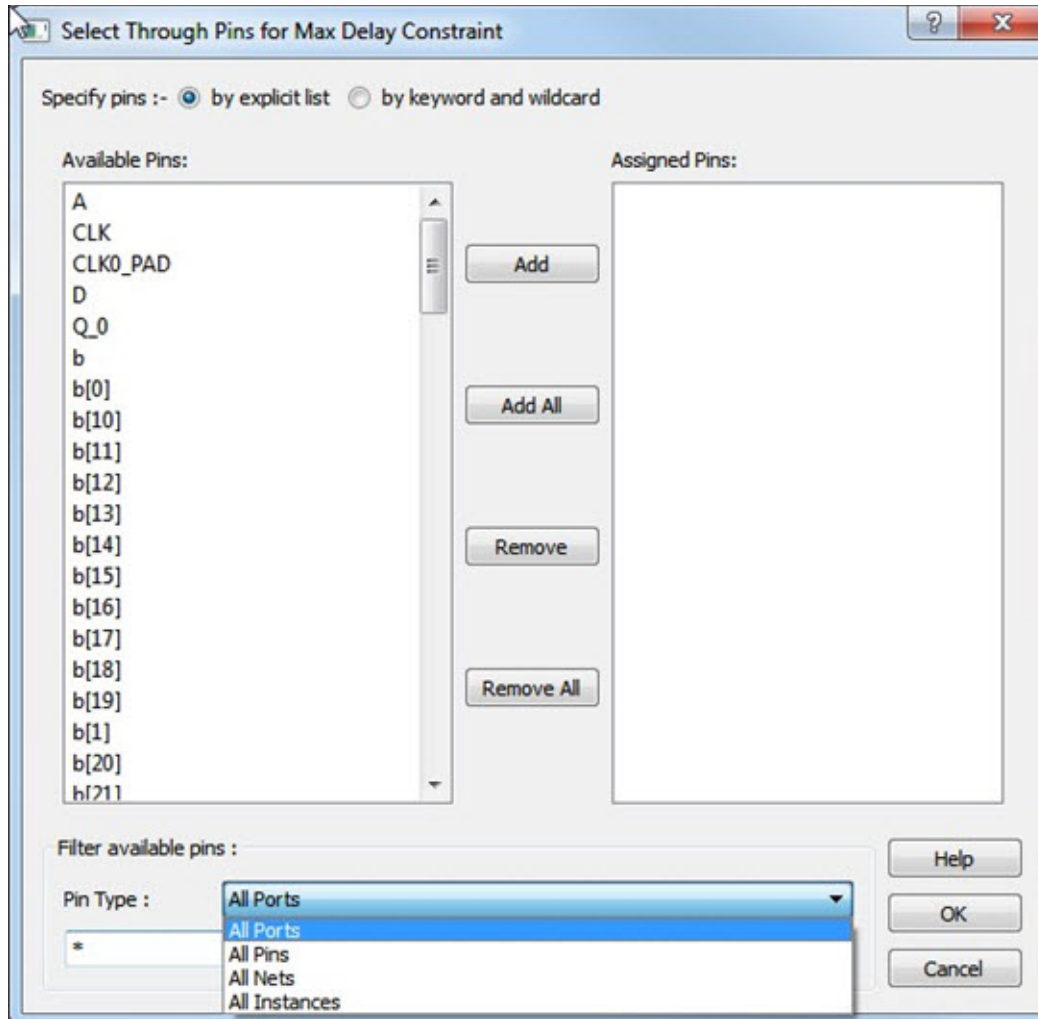


Figure 64 · Select Through Pins for Max Delay Constraint Dialog Box

The available Pin Type options are:

- All Ports
- All Pins
- All Nets
- All Instances

Destination/To Pins

Specifies the ending points for maximum delay constraint. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

To specify the Destination pin(s), click on the browse button next to the “To” field to open the Select Destination Pins for Max Delay Constraint dialog box.

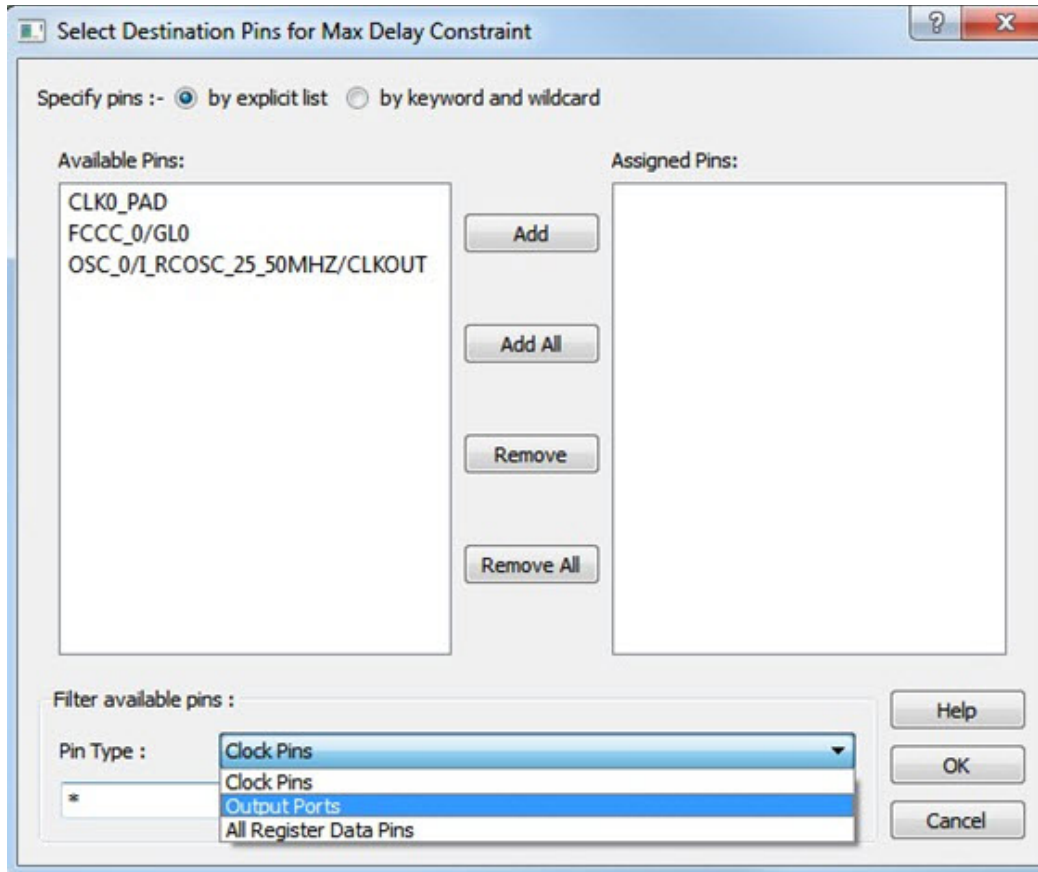


Figure 65 · Select Destination Pins for Max Delay Constraint Dialog Box

The available Pin Type options are:

- Clock Pins
- Output Ports
- All Register Data Pins

Comment

Enter a one-line comment for the constraint.

See Also

[Timing Exceptions Overview](#)

Set a Minimum Delay Constraint

Set the options in the Minimum Delay Constraint dialog box to relax or to tighten the original clock constraint requirement on specific paths.

SmartTime automatically derives the individual minimum delay targets from clock waveforms and port input or output delays. So the minimum delay constraint is a timing exception. This constraint overrides the default single cycle timing relationship for one or more timing paths. This constraint also overrides a multiple cycle path constraint.

Note: When the same timing path has more than one timing exception constraint, SmartTime honors the timing constraint with the highest precedence and ignores the other timing exceptions according to the order of precedence shown.

Timing Exception Constraints	Order of Precedence
set_disable_timing	1

Timing Exception Constraints	Order of Precedence
set_false_path	2
set_maximum_delay/set_minimum_delay	3
set_multicycle_path	4

Note: The set_maximum_delay_constraint has a higher precedence over set_multicycle_path constraint and therefore the former overrides the latter when both constraints are set on the same timing path.

To set a Minimum Delay constraint, open the Set Minimum Delay Constraint dialog box in one of the following four ways:

- From the Constraints Browser, choose Min Delay.
- Double-click the Add Min Delay Constraint icon  .
- Choose Min Delay from the Constraints drop-down menu (Constraints > Min Delay).
- Right click on any row in the Min Delay Constraints Table and select Add Minimum Delay Constraint.

The Set Minimum Delay Constraint dialog box appears.

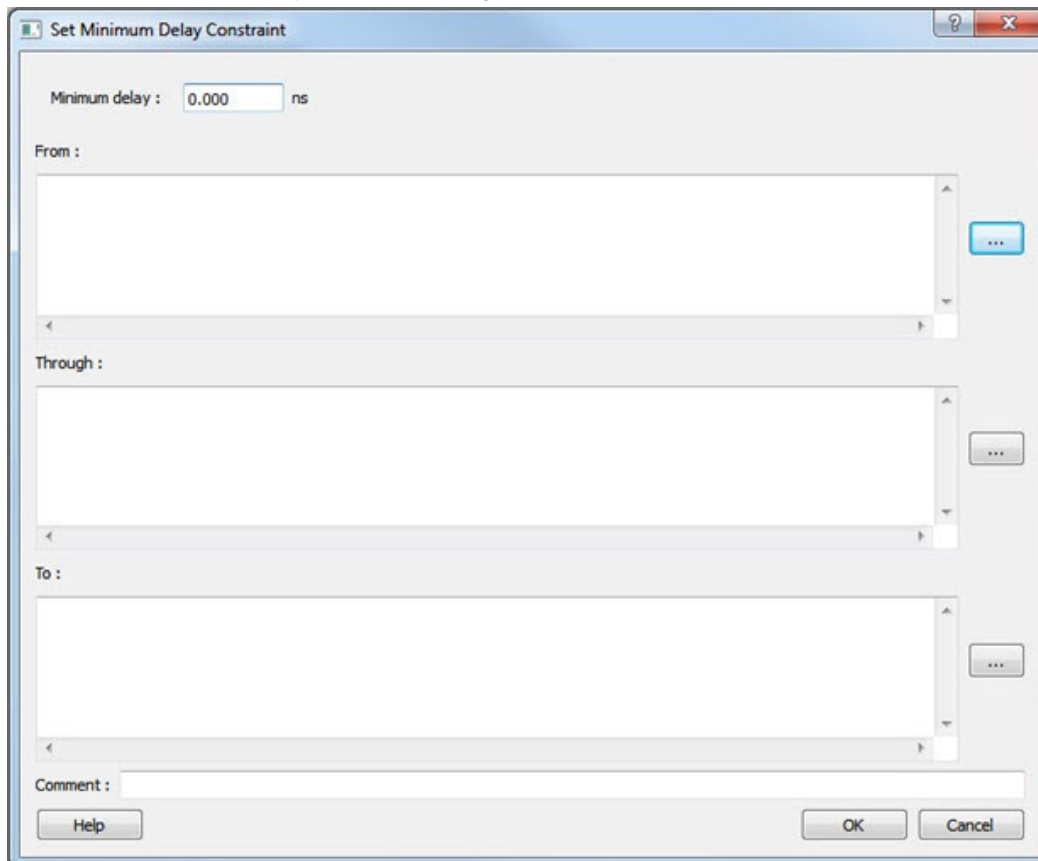


Figure 66 · Set Minimum Delay Constraint Dialog Box

Minimum Delay

Specifies a floating point number in nanoseconds that represents the required minimum delay value for specified paths.

If the path starting point is on a sequential device, SmartTime includes clock skew in the computed delay.

If the path starting point has an input delay specified, SmartTime adds that delay value to the path delay.

If the path ending point is on a sequential device, SmartTime includes clock skew and library setup time in the computed delay.

If the ending point has an output delay specified, SmartTime adds that delay to the path delay.

Source Pins/From

Specifies the starting point for minimum delay constraint. A valid timing starting point is a clock, a primary input, an input port, or a clock pin of a sequential cell.

Click the browse button next to the “From” field to open the Select Source Pins for Min Delay Constraint dialog box.

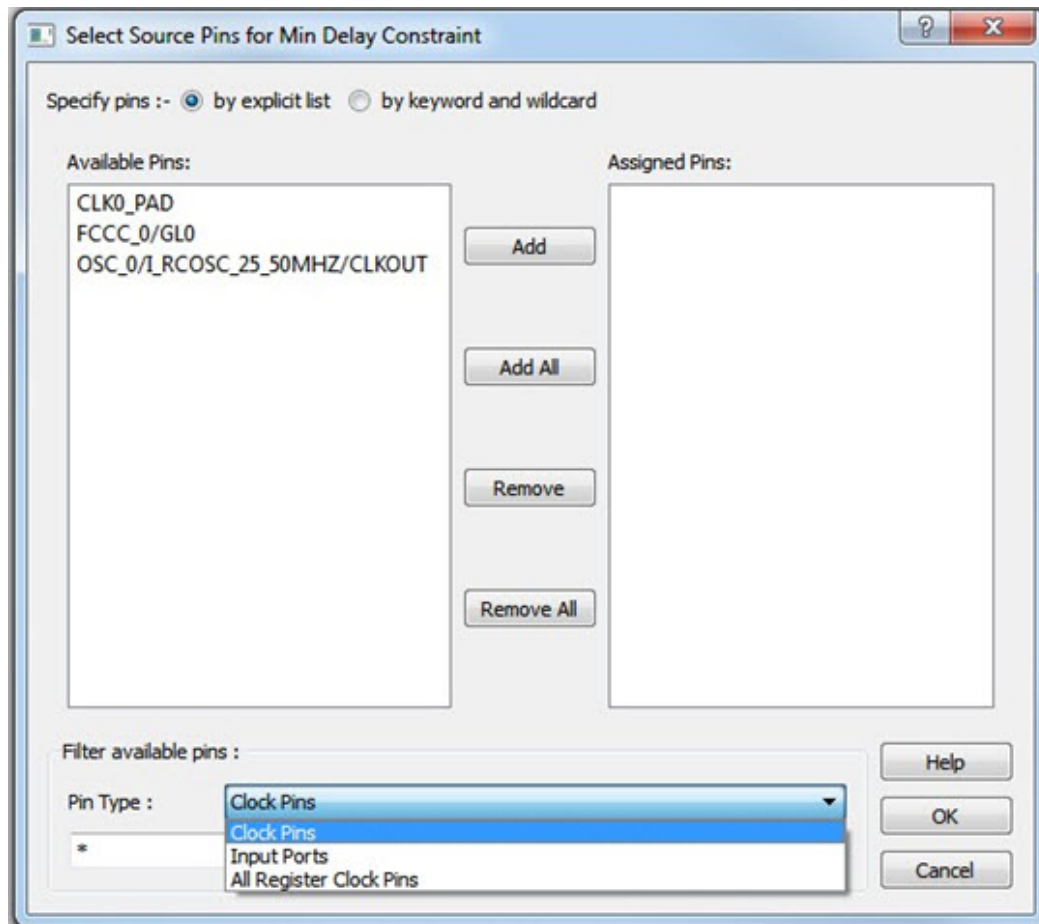


Figure 67 · Select Source Pins for Min Delay Constraint Dialog Box

The available Pin Type options are:

- Clock Pins
- Input Ports
- All Register Clock Pins

Through Pins

Specifies the through points for the Minimum Delay constraint.

Click the browse button next to the “Through” field to open the Select the Through Pins for Min Delay dialog box.

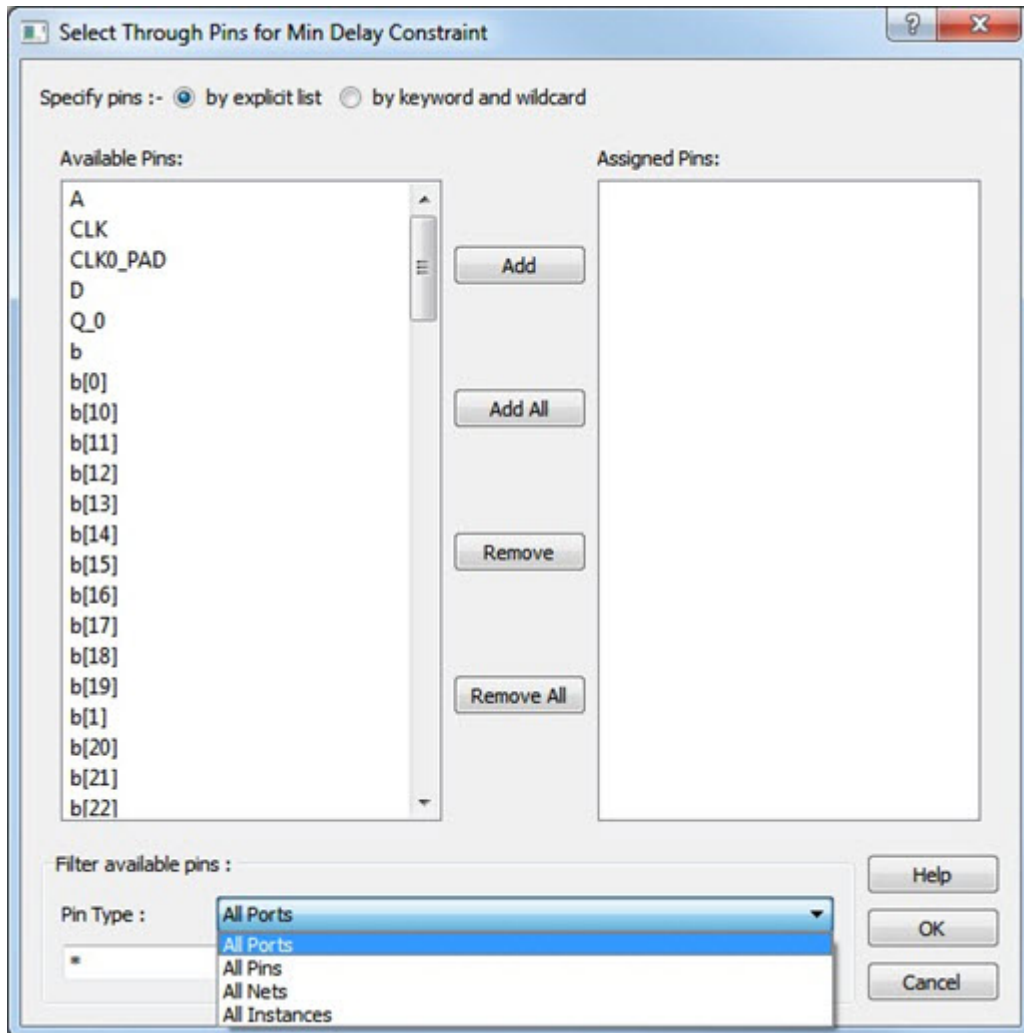


Figure 68 · Select the Through Pins for Min Delay Dialog Box

The available Pin Type options are:

- All Ports
- All Pins
- All Nets
- All Instances

Destination Pins

Specifies the ending points for minimum delay constraint. A valid timing ending point is a clock, a primary output, or a data pin of a sequential cell.

Click the browse button next to the “To” field to open the Select the Destination Pins for Min Delay Constraint dialog box.

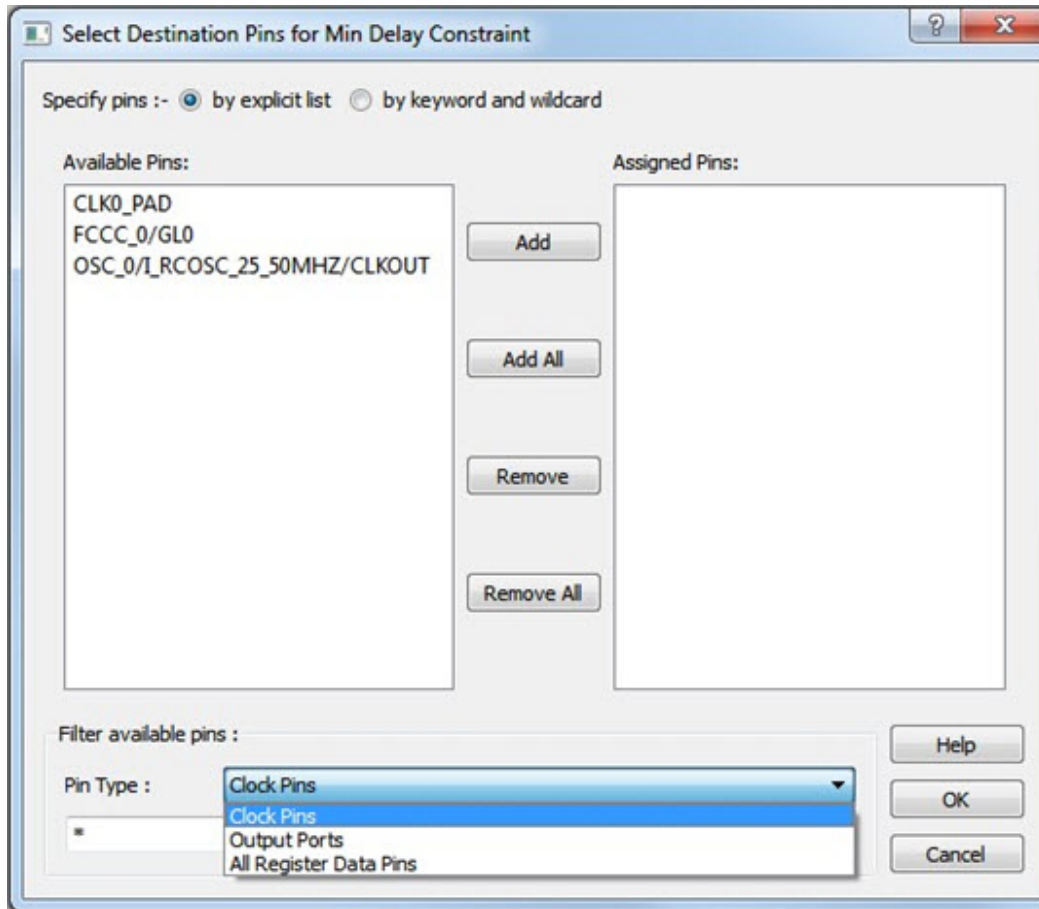


Figure 69 · Select the Destination Pins for Min Delay Constraint Dialog Box

The available Pin Type options are:

- Clock Pins
- Output Ports
- All Register Data Pins

Comment

Enter a one-line comment for the Constraint.

See Also

[Timing Exceptions Overview](#)

[Specifying Minimum Delay Constraints](#)

[set_min_delay \(SDC\)](#)


Set a Multicycle Constraint

Set the options in the Set Multicycle Constraint dialog box to specify paths that take multiple clock cycles in the current design.

Setting the multiple-cycle path constraint overrides the single-cycle timing relationships (the default) between sequential elements by specifying the number of cycles (two or more) that the data path must have for setup or hold checks.

Note: The false path information always takes precedence over multiple cycle path information. A specific maximum delay constraint overrides a general multiple cycle path constraint.

To set a multicycle constraint, open the Set Multicycle Constraint dialog box in one of the following four ways:

- From the Constraints Browser, choose **Multicycle**.
- Double-click the Add Multicycle Constraint icon  .
- Choose Multicycle from the Constraints drop-down menu (**Constraints > Multicycle**).
- Right-click any row in the Multicycle Constraints Table and choose **Add Multicycle Path Constraint**.

The Set Multicycle Constraint dialog box appears.

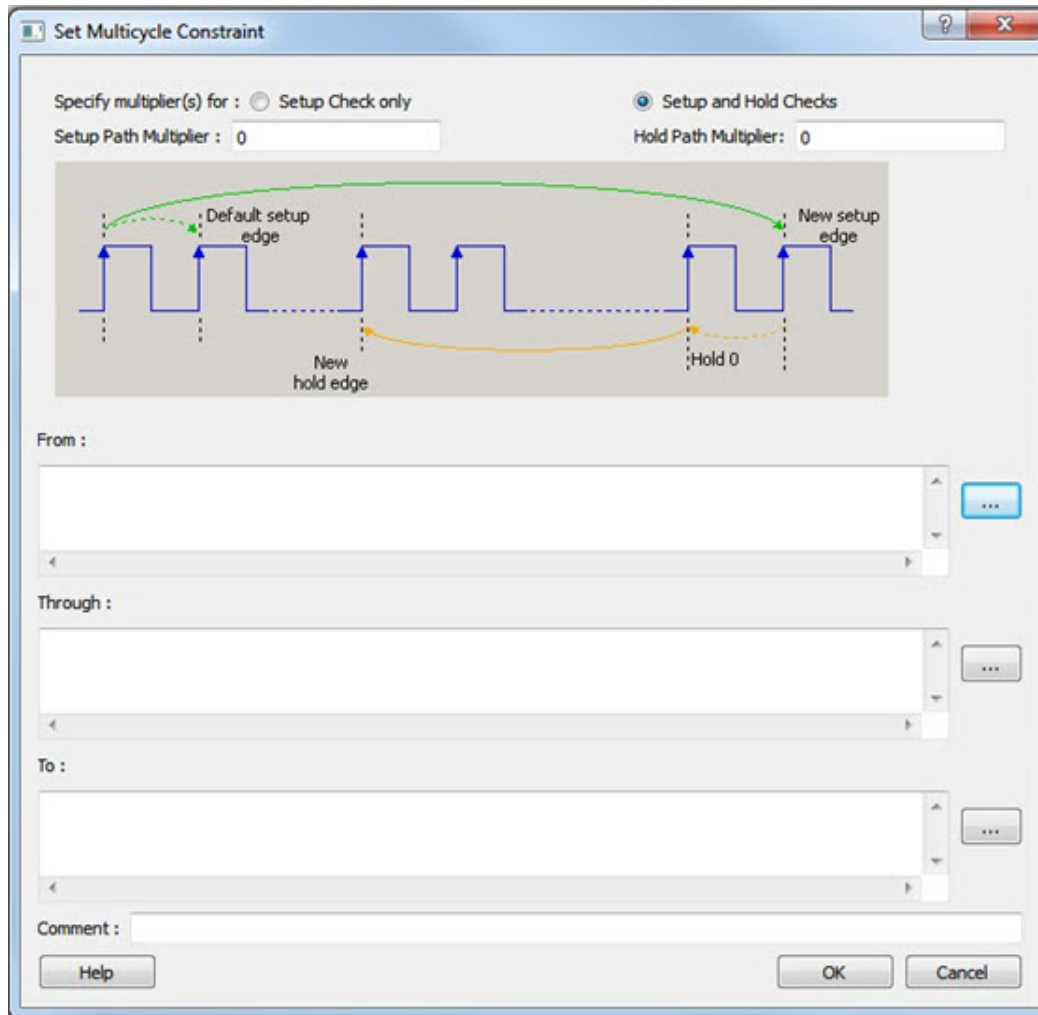


Figure 70 · Set Multicycle Constraint Dialog Box

Setup Check Only

Check this box to apply multiple clock cycle timing consideration for Setup Check only.

Setup and Hold Checks

Check this box to apply multiple clock cycle timing consideration for both Setup and Hold Checks.

Setup Path Multiplier

Specifies an integer value that represents the number of clock cycles (more than one) the data path must have for a setup check.

Hold Path Multiplier

Specifies an integer value that represents the number of clock cycles (more than one) the data path must have for a Hold check.

Source Pins/From Pins

Specifies the starting points for the multiple cycle path. A valid starting point is a clock, a primary input, an inout port, or the clock pin of a sequential cell.

Click the browse button next to the “From” field to open the Select Source Pins for Multicycle Constraint dialog box.

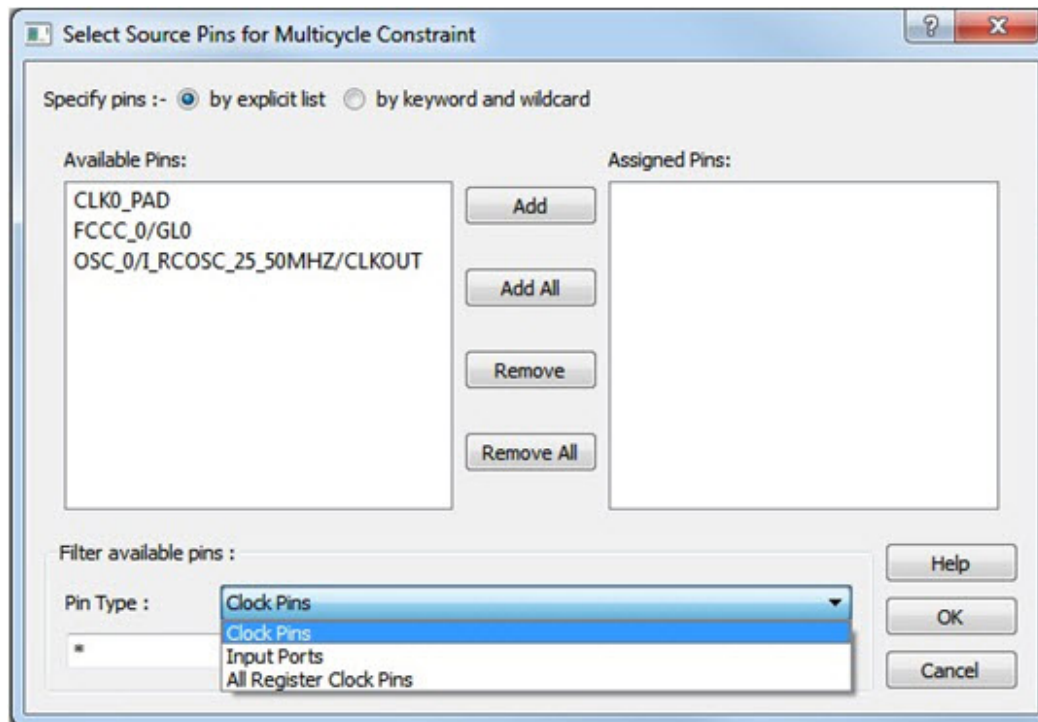


Figure 71 · Select Source Pins for Multicycle Constraint Dialog Box

The available Pin Type options are:

- Clock Pins
- Input Ports
- All Register Clock Pins

Through Pins

Click the browse button next to the “Through” field to open the Select Through Pins for Multicycle Constraint dialog box. The Select Through Pins for Multicycle Constraint dialog box appears.

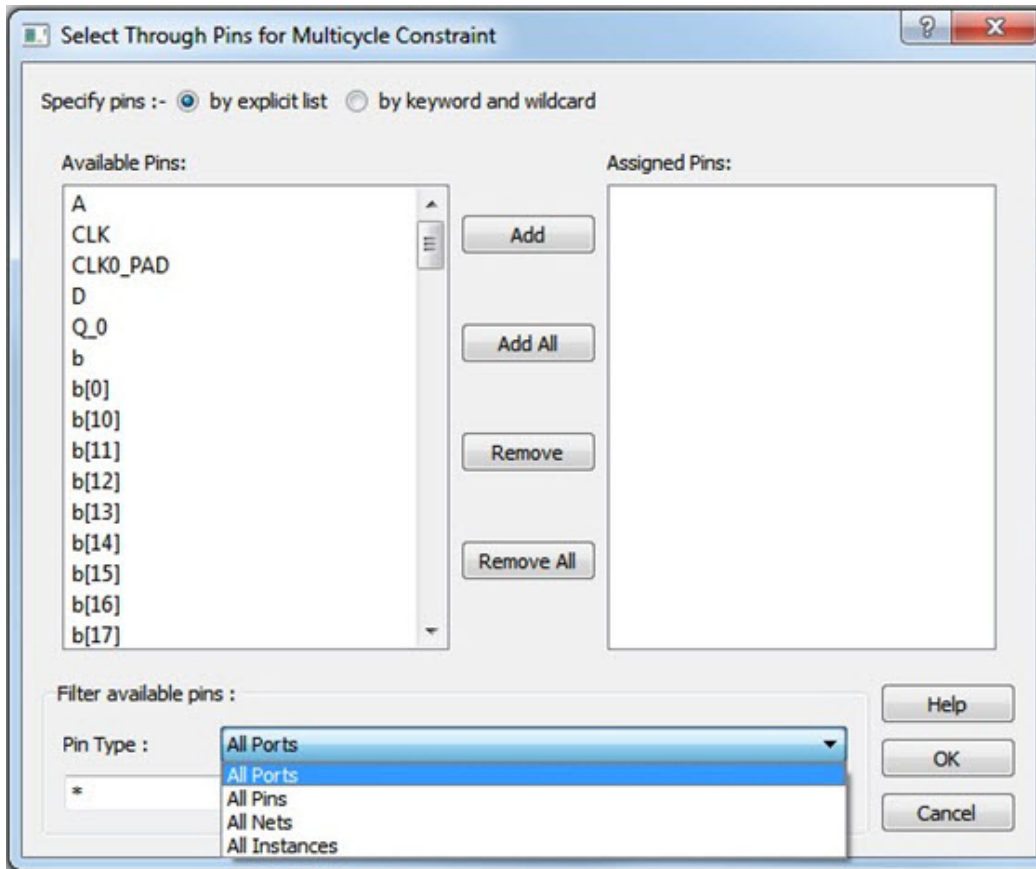


Figure 72 · Select Through Pins for Multicycle Constraint Dialog Box

The available Pin Type options are:

- All Ports
- All Pins
- All Nets
- AllInstances

Destination/To Pins

Click the browse button next to the “To” field to open the Select Destination Pins for Multicycle Constraint dialog box.

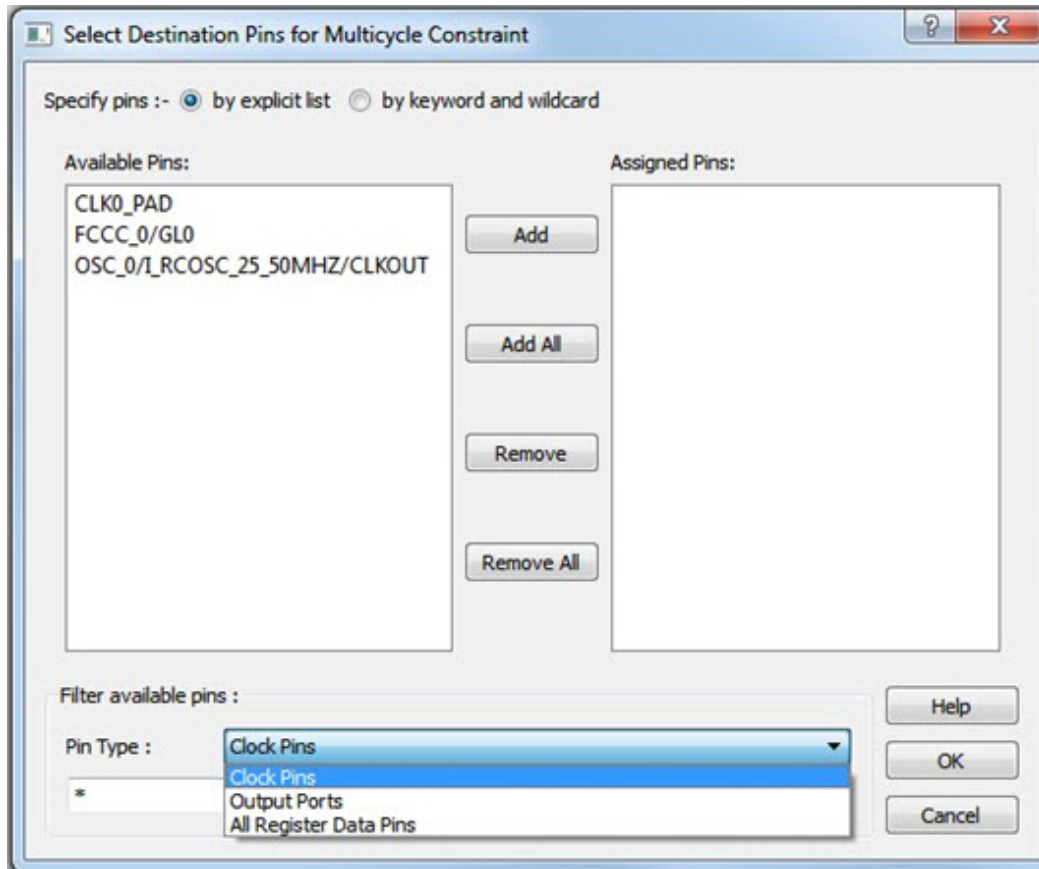


Figure 73 · Select Destination Pins for Multicycle Constraint Dialog Box

The available Pin Type options are:

- Clock Pins
- Output Ports
- All Register Data Pins

Comment

Enter a one-line comment for the constraint.

See Also

[Specifying a Multicycle Constraint](#)

Set a False Path Constraint

Set options in the Set False Path Constraint dialog box to define specific timing paths as false path.

This constraint removes timing requirements on these false paths so that they are not considered during the timing analysis. The path starting points are the input ports or register clock pins and path ending points are the register data pins or output ports. This constraint disables setup and hold checking for the specified paths.

Note: When the same timing path has more than one timing exception constraint, SmartTime honors the timing constraint with the highest precedence and ignores the other timing exceptions according to the order of precedence shown below.

Timing Exception Constraints	Order of Precedence
set_disable_timing	1

Timing Exception Constraints	Order of Precedence
set_false_path	2
set_maximum_delay/set_minimum_delay	3
set_multicycle_path	4

Note: The set_false_path constraint has the second highest precedence and always overrides the set_multicycle_path constraints and set_maximum/minimum_delay constraints.

To set a false path constraint, open the Set False Path Constraint dialog box in one of the following four ways:

- From the Constraints Browser, choose **False Path**.
- Double-click the Add False Path Constraint icon .
- Choose **False Path** from the Constraints drop-down menu (**Constraints > False Path**).
- Right-click any row in the False Path Constraints Table and choose **Add False Path Constraint**.

The Set False Path Constraint dialog box appears.

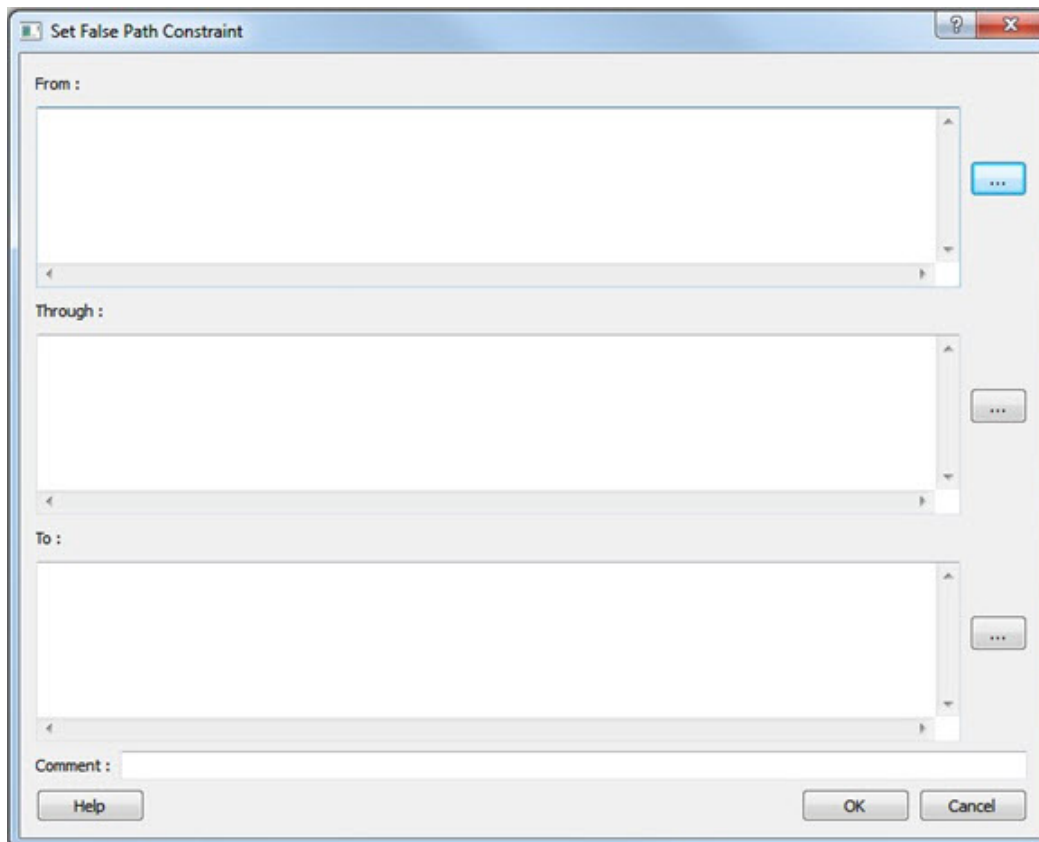


Figure 74 · Set False Path Constraint Dialog Box

Source/From Pins

To select the Source Pin(s), click the browse button next to the “From” field and open the Select Source Pins for False Path Constraint dialog box.

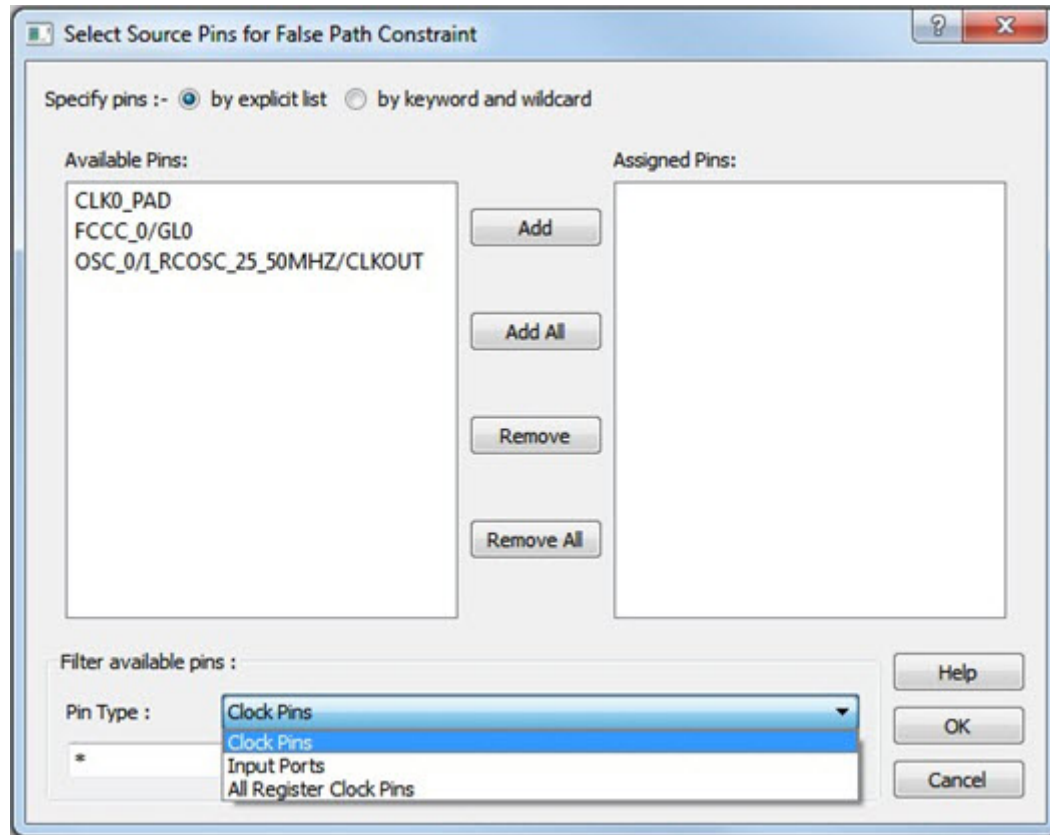


Figure 75 · Select Source Pins for False Path Constraint Dialog Box

The available options for Pin Type are:

- Clock Pins
- Input Ports
- All Register Clock Pins

Through Pins

Specifies a list of pins, ports, cells, or nets through which the false paths must pass.

To select the Through pin(s), click the browse button next to the “Through” field to open the Select Through Pins for False Path Constraint dialog box.

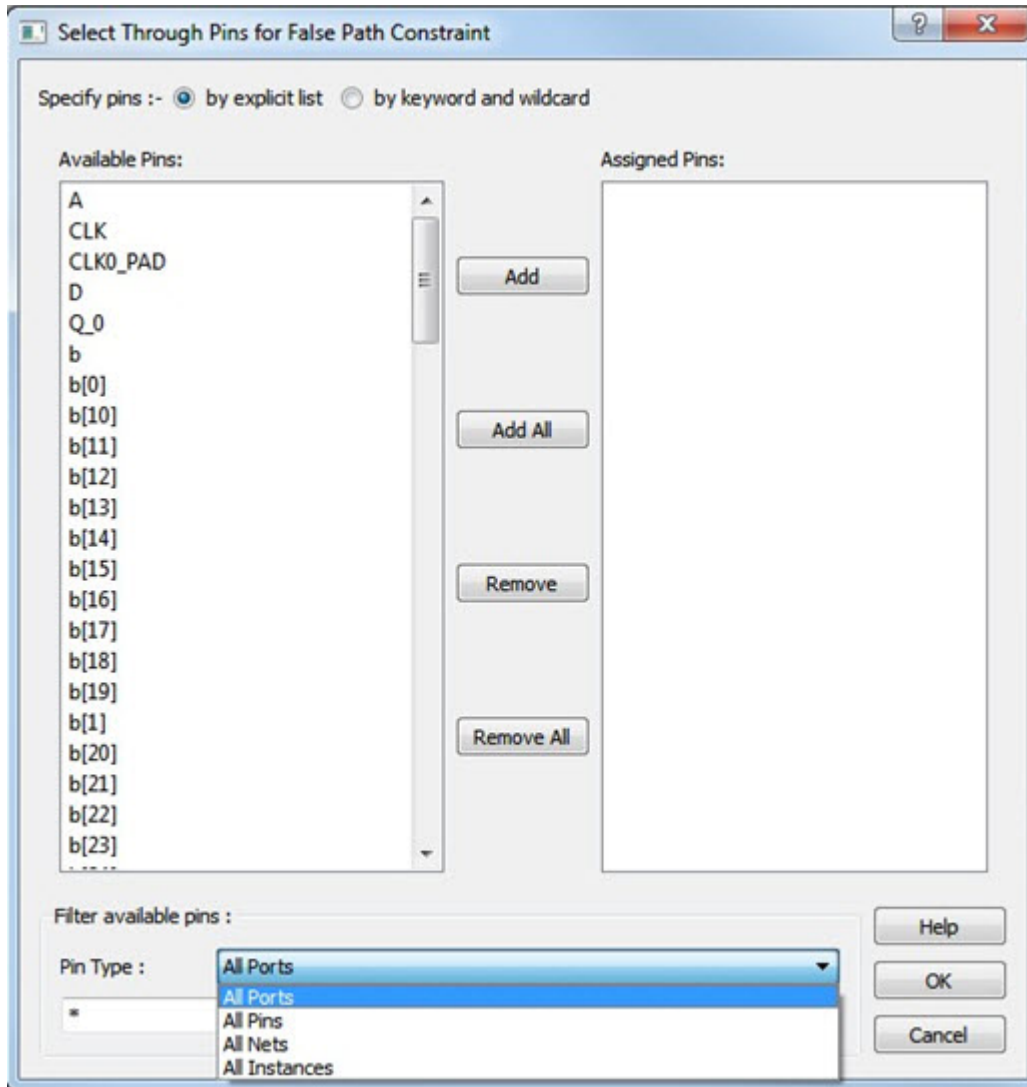


Figure 76 · Select Through Pins for False Path Constraint Dialog Box

The available options for Pin Type are:

- All Ports
- All Pins
- All Nets
- All Instances

Destination/To Pins

To select the Destination Pin(s), click the browse button next to the “To” field to open the Select Destination Pins for False Path Constraint dialog box.

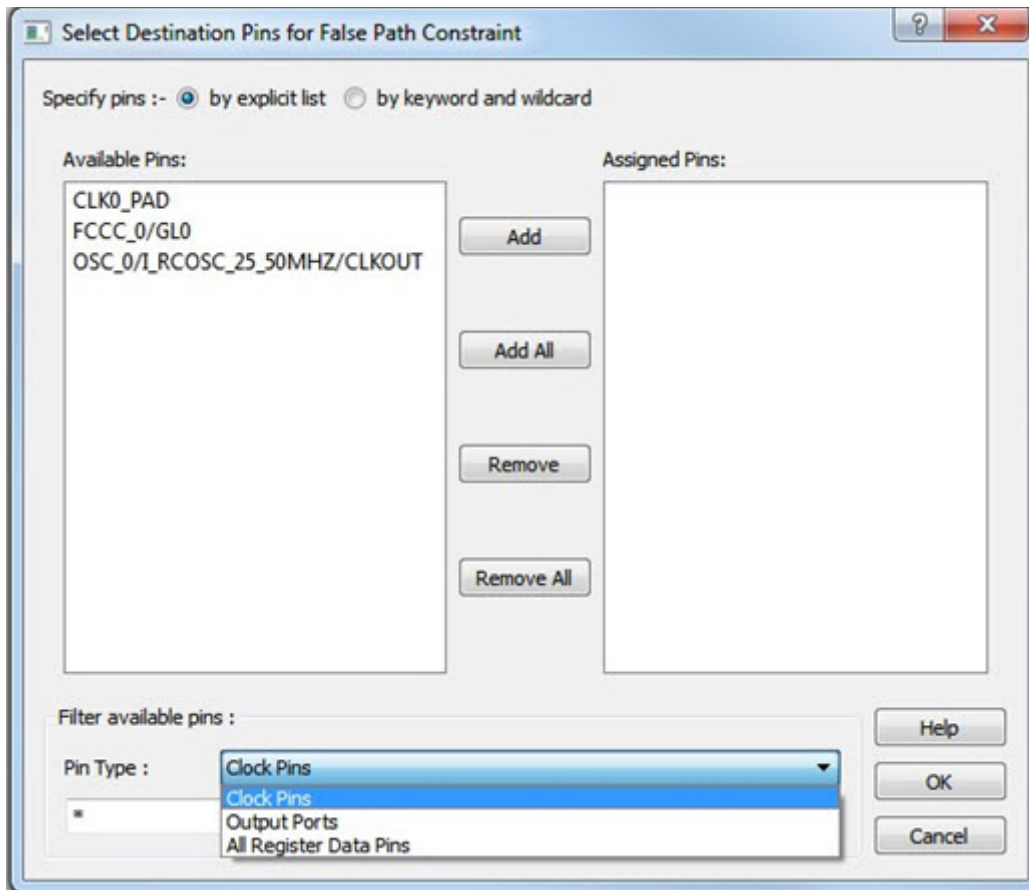


Figure 77 · Select Destination Pins for False Path Constraint Dialog Box

The available options for Pin Type are:

- Clock Pins
- Output Ports
- All Register Data Pins

Comment

Enter a one-line comment for the constraint.

See Also

[Specifying False Path Constraints](#)

Set a Disable Timing Constraint

Use disable timing constraint to specify the timing arcs to be disabled for timing consideration.

Note: This constraint is for the Place and Route tool and the Verify Timing tool. It is ignored by the Synthesis tool.

To specify a Disable Timing constraint, open the Set Constraint to Disable Timing Arcs dialog box in one of the following four ways:

- From the Constraints Browser, choose **Advanced > Disable Timing**.



- Double-click the Add Disable Timing Constraint icon.
- Choose **Disable Timing** from the Constraints drop-down menu (**Constraints > Disable Timing**).
- Right-click any row in the Disable Timing Constraints Table and choose **Add Constraint to Disable Timing**.

The Set Constraint to Disable Timing Arcs dialog box appears.

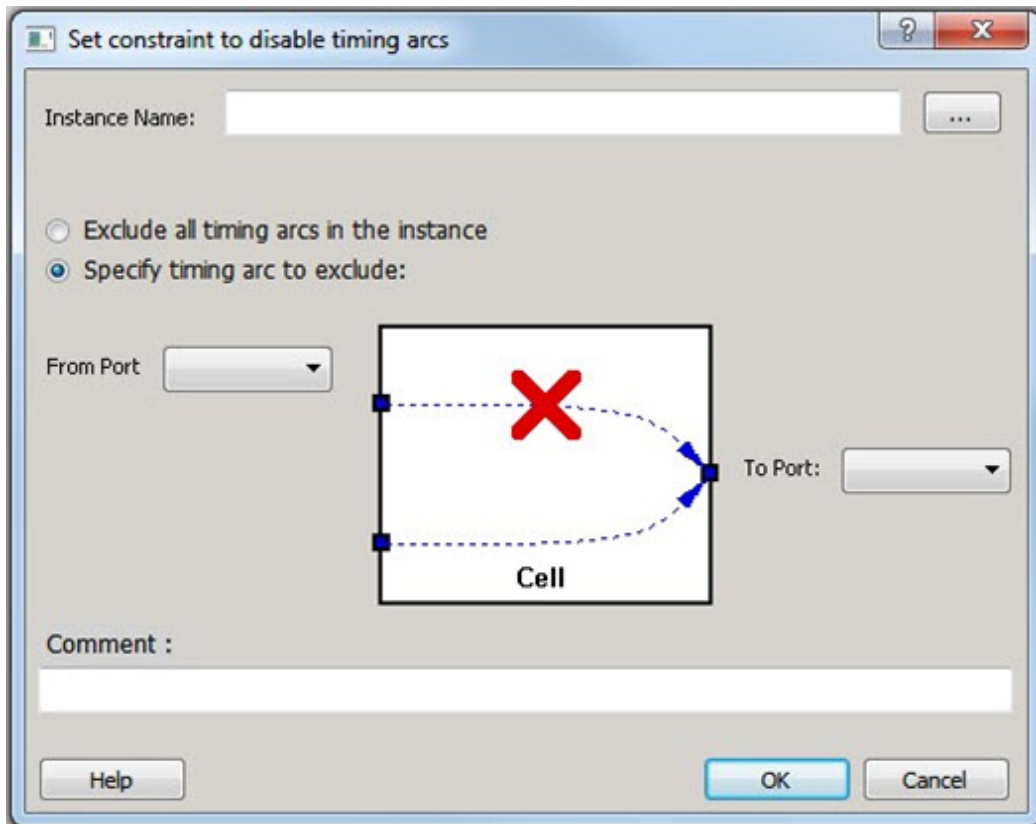


Figure 78 · Set Constraint to Disable Timing Arcs Dialog Box

Instance Name

Specifies the instance name for which the disable timing arc constraint will be created.

Click the browse button next to the Instance Name field to open the Select instance to constrain dialog box.

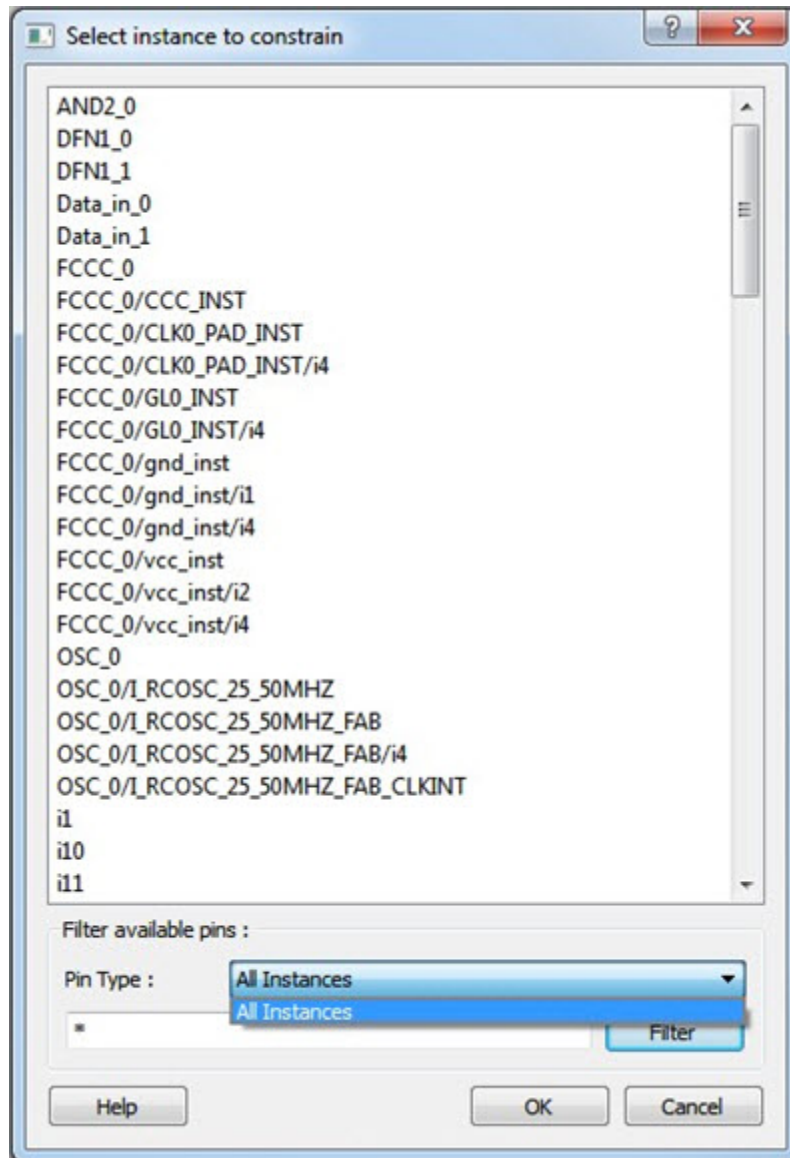


Figure 79 · Set Instance to Constrain Dialog Box

The Pin Type selection is limited to All Instances only.

Exclude All Timing Arcs in the Instance

This option enables you to exclude all timing arcs in the specified instance.

Specify Timing Arc to Exclude

This option enables you to specify the timing arc to exclude. In this case, you need to specify the from and to ports:

From Port

Specifies the starting point for the timing arc.

To Port

Specifies the ending point for the timing arc.

Comment

Enter a one-line comment for the constraint.

Set Clock Source Latency Constraint

Use clock source latency constraint to specify the delay from the clock generation point to the clock definition point in the design.

Clock source latency defines the delay between an external clock source and the definition pin of a clock. It behaves much like an input delay constraint.

You can specify both an "early" delay and a "late" delay for this latency, providing an uncertainty which the timing analyzer can use for propagating through its calculations. Rising and falling edges of the same clock can have different latencies. If only one value is provided for the clock source latency, it is taken as the exact latency value, for both rising and falling edges.

To specify a Clock Source Latency constraint, open the Set Clock Source Latency Constraint dialog box in one of the following four ways:

- From the Constraints Browser, choose **Clock Source Latency**.
- Double-click the Clock Source Latency Constraint icon .
- Choose **Clock Source Latency** from the Constraints drop-down menu (**Constraints > Advanced > Clock Source Latency**).
- Right-click any row of the Clock Latency Constraints Table and choose **Add Clock Source Latency**.

The Set Clock Source Latency Constraint dialog box appears.

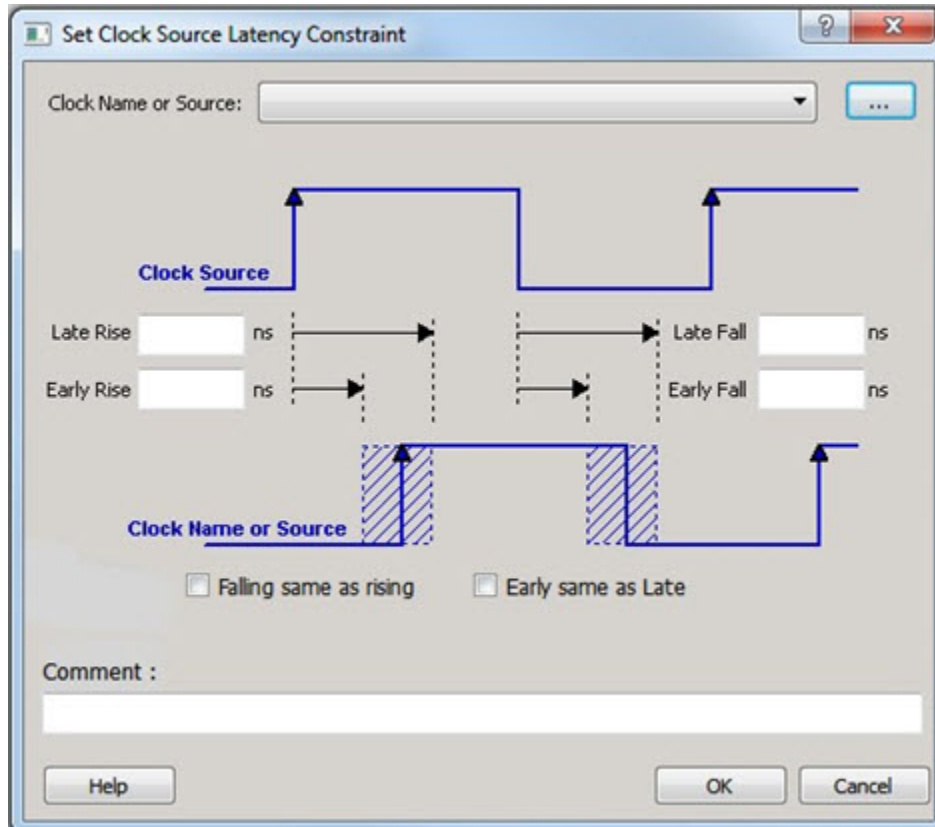


Figure 80 · Set Clock Source Latency Constraint Dialog Box

To select the Clock Source, click on the browser button to open the Choose the Clock Source Pin dialog box:

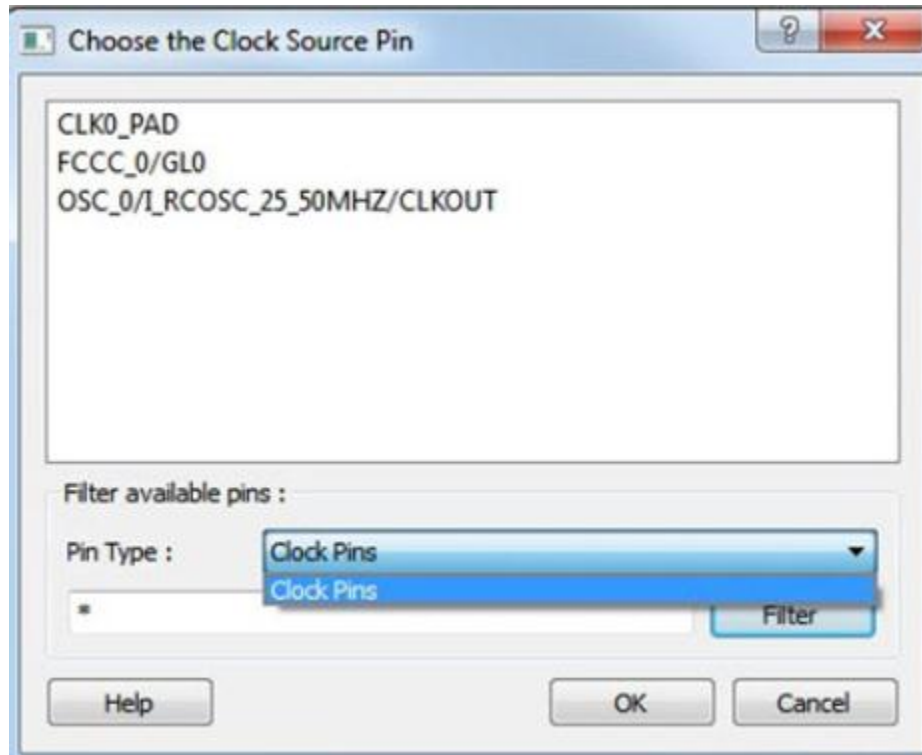


Figure 81 · Choose the Clock Source Pin Dialog Box

The only choice available for Pin Type is Clock Pins.

Late Rise

Specifies the largest possible latency, in nanoseconds, of the rising clock edge at the clock port or pin selected, with respect to its source. Negative values are acceptable, but may lead to overly optimistic analysis.

Early Rise

Specifies the smallest possible latency, in nanoseconds, of the rising clock edge at the clock port or pin selected, with respect to its source. Negative values are acceptable, but may lead to overly optimistic analysis.

Late Fall

Specifies the largest possible latency, in nanoseconds, of the falling clock edge at the clock port or pin selected, with respect to its source. Negative values are acceptable, but may lead to overly optimistic analysis.

Early Fall

Specifies the smallest possible latency, in nanoseconds, of the falling clock edge at the clock port or pin selected, with respect to its source. Negative values are acceptable, but may lead to overly optimistic analysis.

Clock Edges

Select the latency for the rising and falling edges:

Falling same as rising: Specifies that Rising and Falling clock edges have the same latency.

Early same as late: Specifies that the clock source latency should be considered as a single value, not a range from "early" to "late".

Comment

Enter a one-line comment to describe the clock source latency.

See Also


[Specifying Clock Constraints](#)

[Set Clock Latency Constraint](#)

Set Clock-to-Clock Uncertainty Constraint

Use the clock-to-clock uncertainty constraint to model tracking jitter between two clocks in your design.

To specify a Clock-to-Clock Uncertainty constraint, open the Set Clock-to-Clock Uncertainty Constraint dialog box in one of the following four ways:

- From the Constraints Browser, choose **Clock Uncertainty**.
- Double-click the Clock-to-Clock Uncertainty icon  .
- Choose **Clock-to-Clock Uncertainty** from the Constraints drop-down menu (**Constraints > Advanced > Clock-to-Clock Uncertainty**).
- Right-click any row in the Clock Uncertainty Constraints Table and choose **Add Clock-to-Clock Uncertainty**.

The Set Clock-to-Clock Uncertainty Constraint dialog box appears.

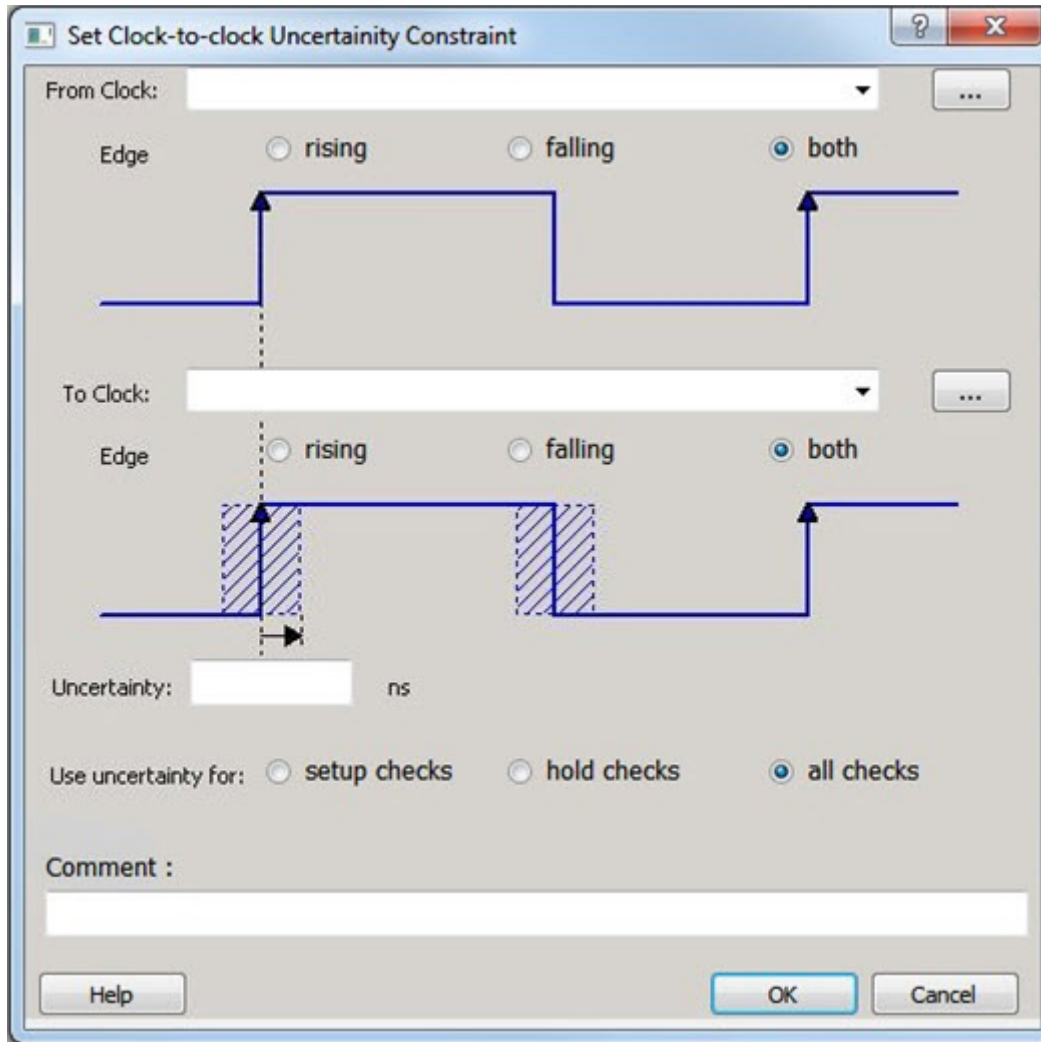


Figure 82 · Set Clock-to-Clock Uncertainty Dialog Box

From Clock

Specifies clock name as the uncertainty source.

To set the From Clock, click the browser button to open the Select Source Clock List for Clock-to-clock Uncertainty dialog box.

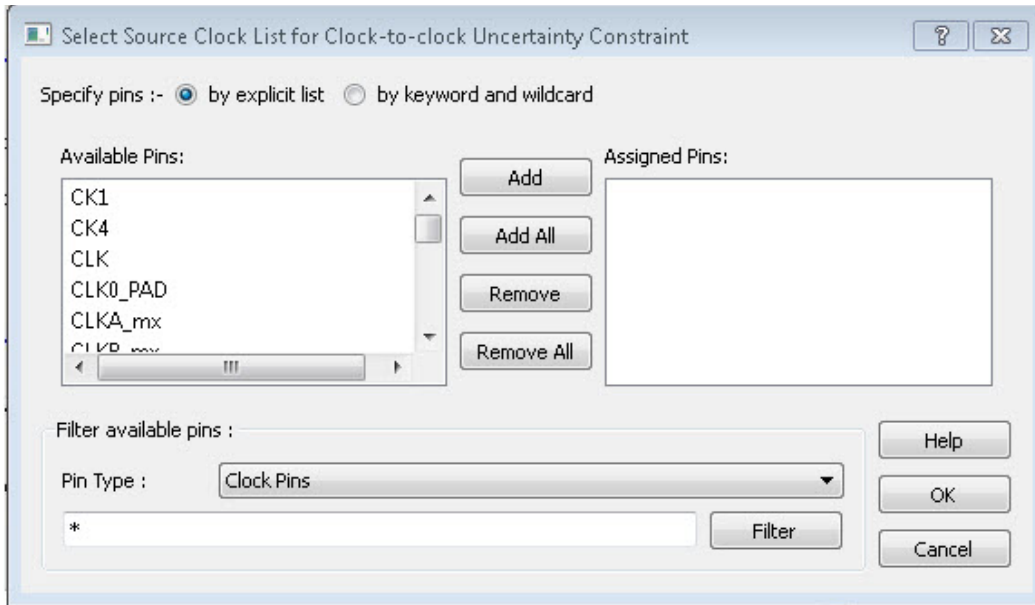


Figure 83 · Select Source Clock List for Clock-to-Clock Uncertainty Dialog Box

The Pin Type selection is for Clock Pins only.

Edge

This option enables you to select if the clock-to-clock uncertainty applies to rising, falling, or both edges.

To Clock

Specifies clock name as the uncertainty destination.

To set the To Clock, click the browser button to open the Select Destination Clock List for Clock-to-clock Uncertainty Constraint dialog box.

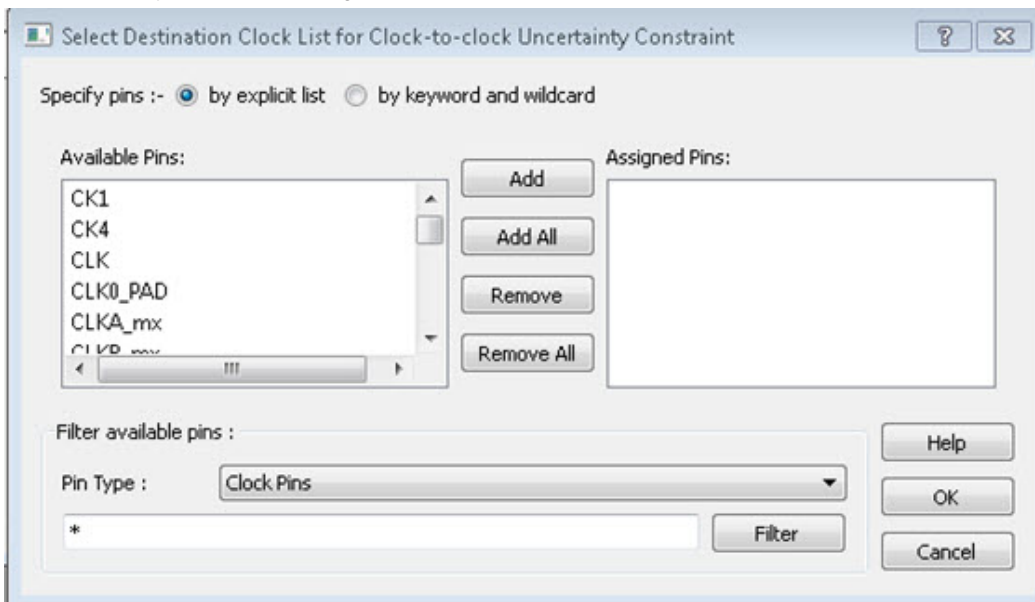


Figure 84 · Select Destination Clock List for Clock-to-Clock Uncertainty Constraint Dialog Box

Edge

This option enables you to select if the clock-to-clock uncertainty applies to rising, falling, or both edges.

Uncertainty

Enter the time in nanoseconds that represents the amount of variation between two clock edges.

Use Uncertainty For

This option enables you select whether the uncertainty constraint applies to setup, hold, or all checks.

Comment

Enter a single line of text that describes this constraint.

To set the Destination Clock, click the browser button to open the Select Destination Clock List for Clock-to-clock Uncertainty Constraint dialog box.

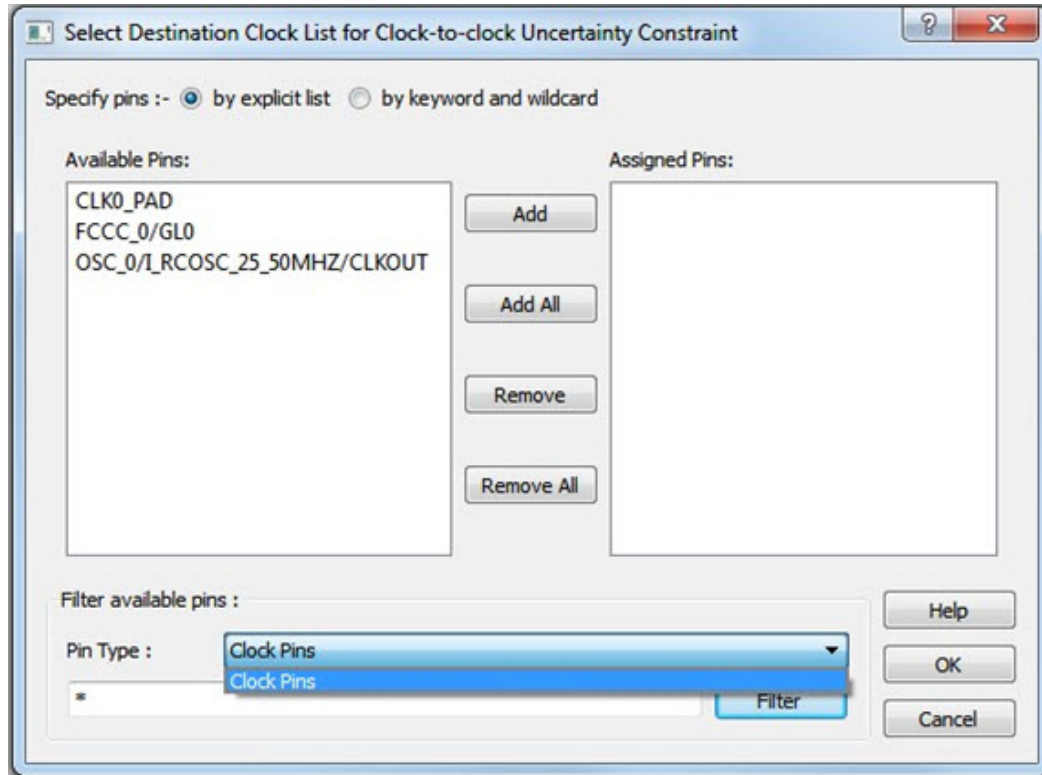


Figure 85 · Select Destination Clock List for Clock-to-Clock Uncertainty Dialog Box
The Pin Type selection is for Clock Pins only.

See Also

[Specifying Disable Timing Constraints](#)

[set clock uncertainty](#)

Set Clock Groups

To add or delete a Clock Group constraint, open the Add Clock Groups Constraint dialog box in one of three ways:

- Select **Clock Groups** from the Constraints drop-down menu (**Constraints > Clock Groups**).
- Double-click **Clock Groups** in the Constraints Browser.
- Right-click any row in the Clock Groups Constraints Table and choose **Add Clock Groups**.

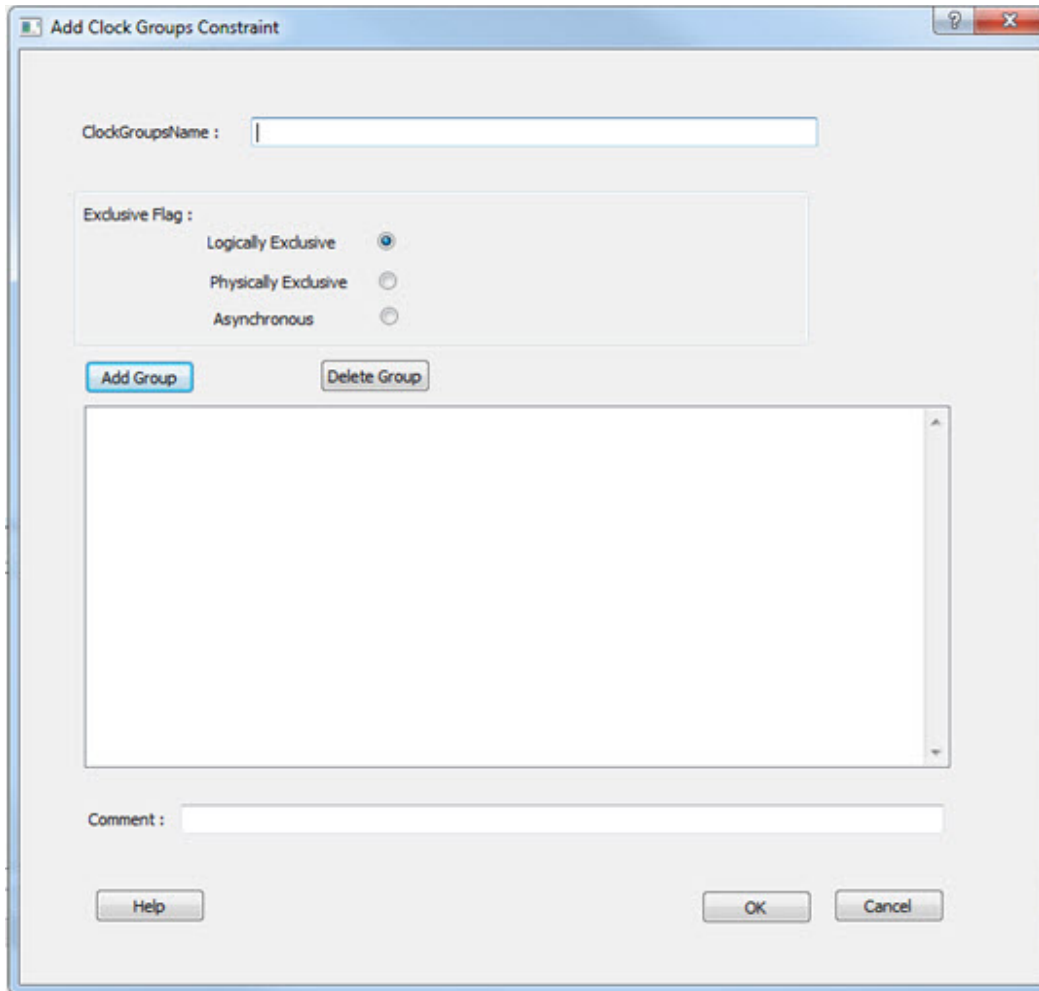


Figure 86 · Add Clock Group Constraints Dialog Box

ClockGroupsName – Enter a name for the Clock Groups to be added.

Exclusive Flag - Choose one of the three clock group attributes for the clock group:

- **Logically Exclusive** - Use this setting for clocks that can exist physically on the device at the same time but are logically exclusive (e.g., multiplexed clocks).
- **Physically Exclusive** - Use this setting for clocks that cannot exist physically on the device at the same time (e.g., multiple clocks defined on the same pin).
- **Asynchronous** – Use this setting when there are valid timing paths between the two clock groups but the two clocks do not have any frequency or phase relationship and therefore these timing paths can be excluded from timing analysis.

Add Group – Click **Add** to open a dialog to add clocks to a clock group. Select the clocks from the Available Pins list and click **Add** to move them to Assigned Pins list. Click **OK**.

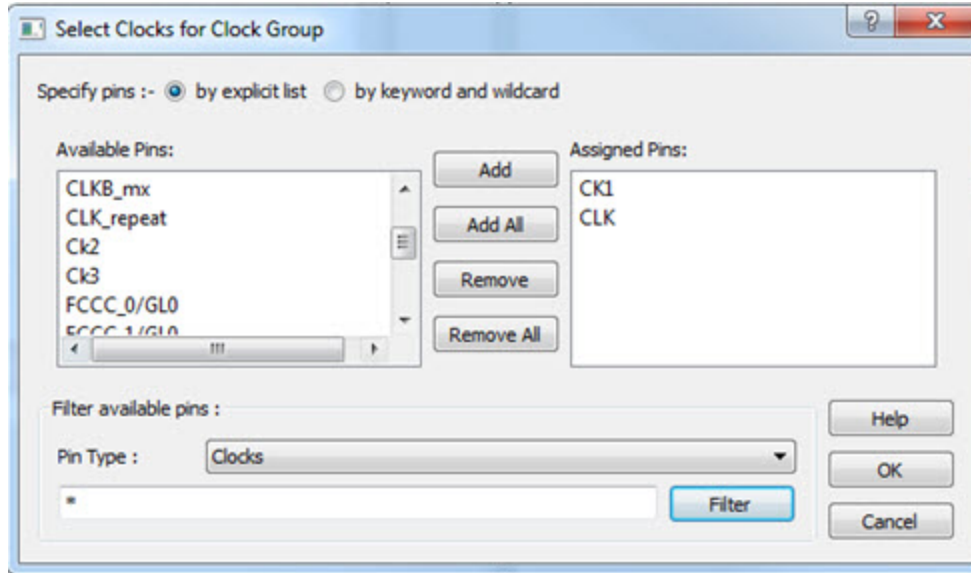


Figure 87 · Add Clocks For Clock Group Dialog Box

Delete Group – Delete the clocks from the Clock Group. Select the group of clock to be deleted and click **Delete Group**. This will delete the clock group.

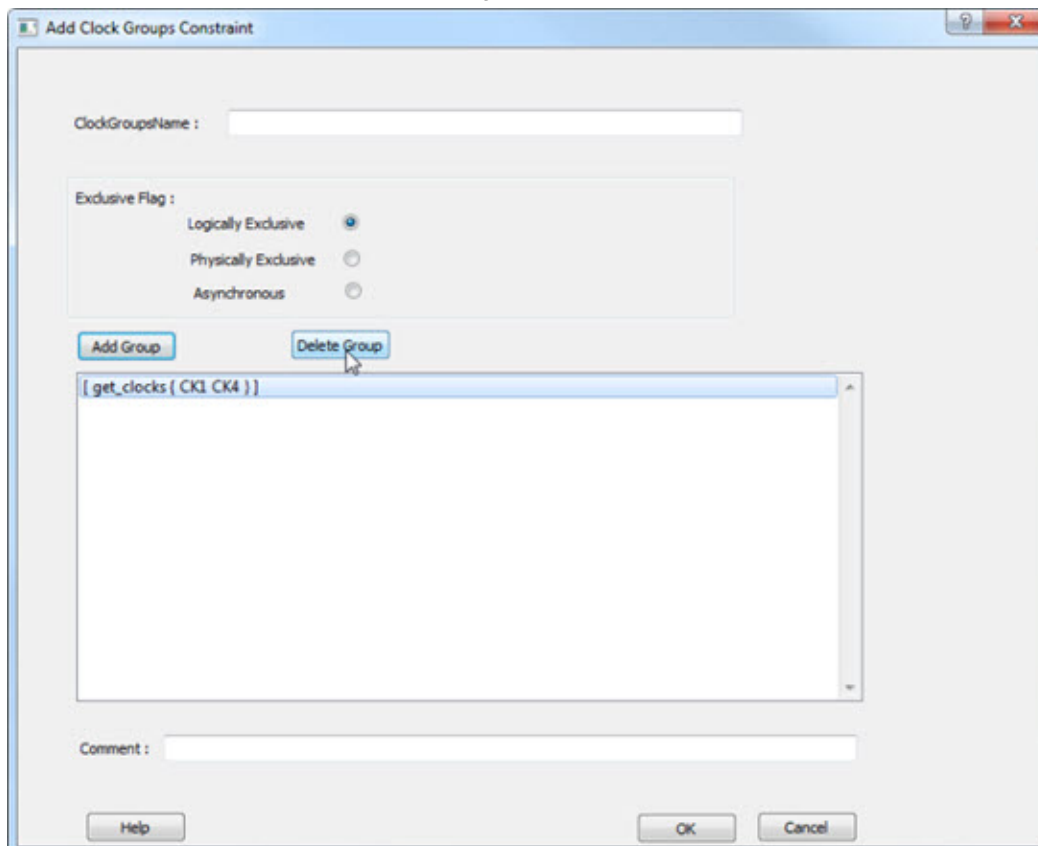


Figure 88 · Delete Group

See Also

[set_clock_groups](#)

[list_clock_groups](#)

[remove clock groups](#)

Select Destination Clock for Clock-to-clock Uncertainty Constraint Dialog Box

This dialog box opens when you select the browse button for Destination/To Clock for Clock-to-clock Uncertainty Constraints dialog box.

Use this dialog box to select Clock Pins:

- By explicit list
- By keyword and wildcard

To open the Select Destination Clock dialog box, double-click **Constraint > Advanced > Clock Uncertainty**. Click the browse button next to the To Clock field to select the Destination Clock Pin.

By Explicit List

This is the default. This mode stores the actual Clock Pin names. The following figure shows an example dialog box for Select Destination Clock.

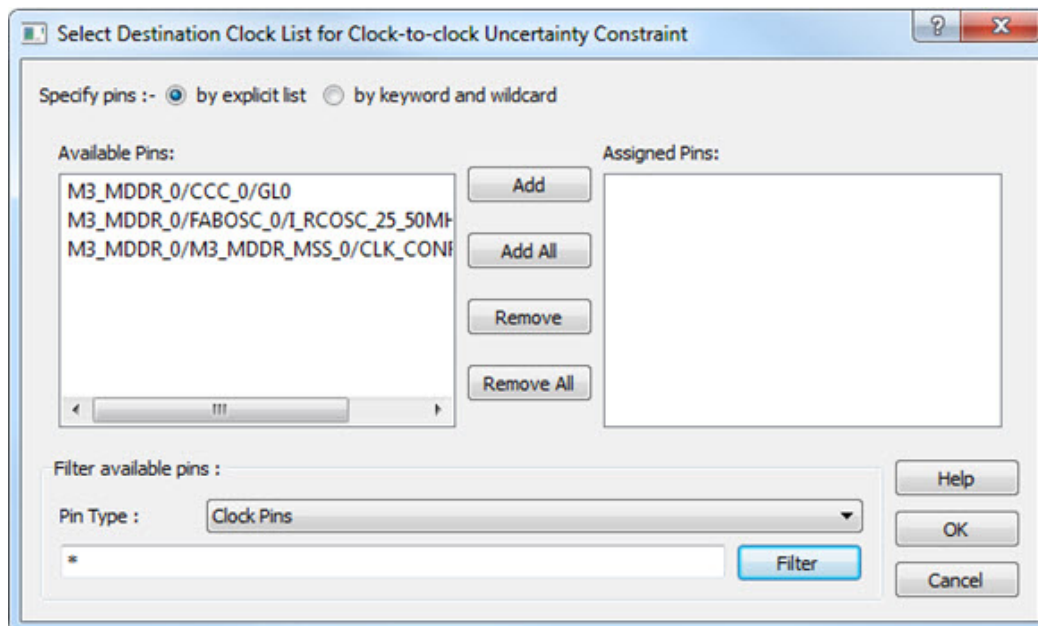


Figure 89 · Select Destination Clock Pins for Clock-to-Clock Uncertainty Dialog Box – by Explicit List

Available Pins

The list box displays the available Clock Pins. If you change the filter value, the list box shows the available pins based on the filter.

Use Add, Add All, to add Clock Pins from the Available Pins List or Remove, Remove All to delete Clock Pins from the Assigned Pins list.

Filter Available Pins

Pin type – Specifies the filter on the available Clock Pins.

Filter

Specifies the filter based on which the Available Pins list shows the Clock Pin names. The default is *, which is a wild-card match for all. You can specify any string value.

By Keyword and Wildcard

This mode stores the filter only. It does not store the actual pin names. The constraints created using this mode get exported with the SDC accessors (get_pins) and the wildcard filter. The following figure shows an example dialog box for Select Destination Clock Pins by keyword and the *CCC* filter.

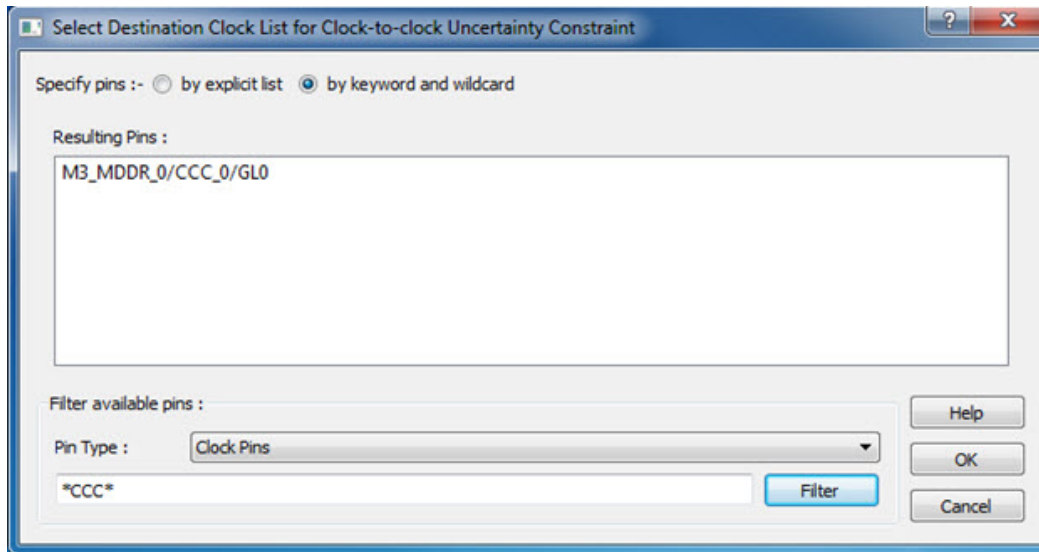


Figure 90 · Select Destination Clock Pins for Clock-to-Clock Uncertainty Dialog Box – By Keyword and Wildcard

Pin Type

Specifies the filter on the available pins. The only valid selection is Clock Pins.

Filter

Specifies the filter based on which the Available Pins list shows the pin names. The default is *, which is a wild-card match for all. You can specify any string value.

Resulting Pins

Displays pins from the available pins based on the filter.

Select Instance to Constrain Dialog Box

This dialog box appears when you click the browse button next to the Instance Name field in the Set Constraint to Disable Timing Arcs Dialog Box.

The list box displays the available Pins. If you change the filter value, the list box shows the available pins based on the filter.

Filter Available Pins

Pin type – Specifies the filter on the available Pin Types: All Instances is the only valid type.

Filter

Specifies the filter based on which the Available Pins list shows the Pin names. The default is *, which is a wild-card match for all. You can specify any string value.

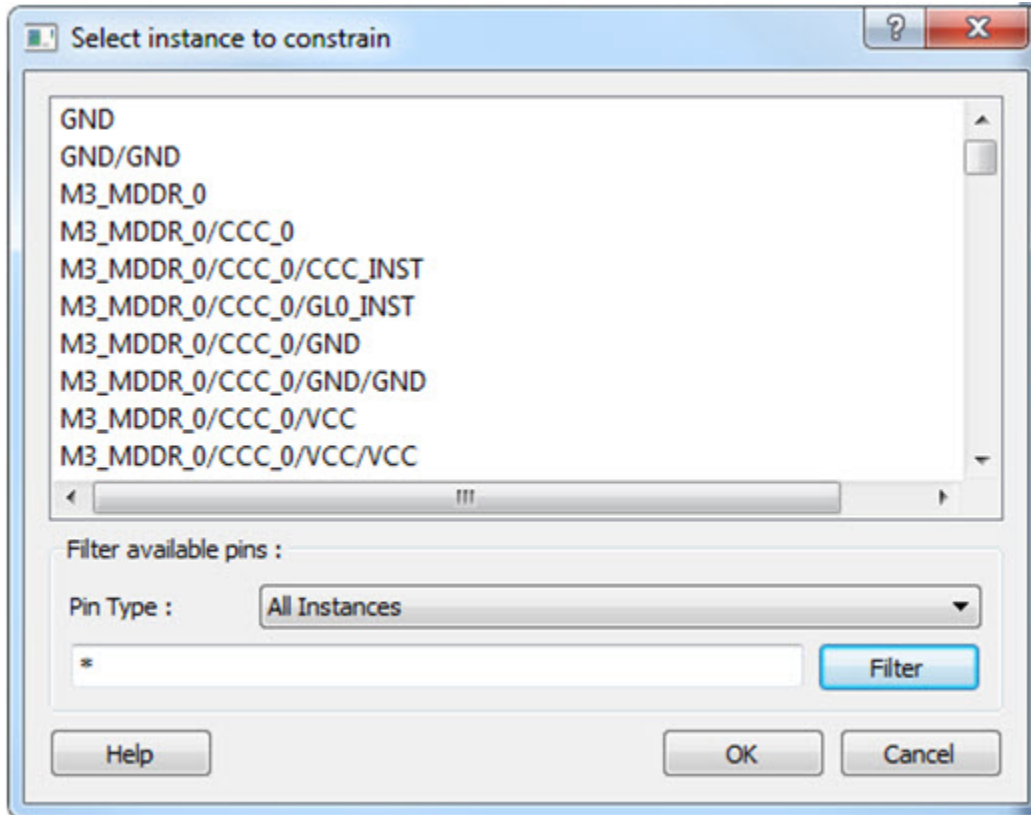


Figure 91 · Select Instance to Constrain Dialog Box

Select Generated Clock Reference Dialog Box

Use this dialog box to find and choose the generated clock reference pin from the list of available pins.

To open the Select Select Generated Clock Reference dialog box (shown below) from the Constraints Editor, open the Create Generated Clock Constraint Dialog Box dialog box and click the browse button for the Reference Pin.

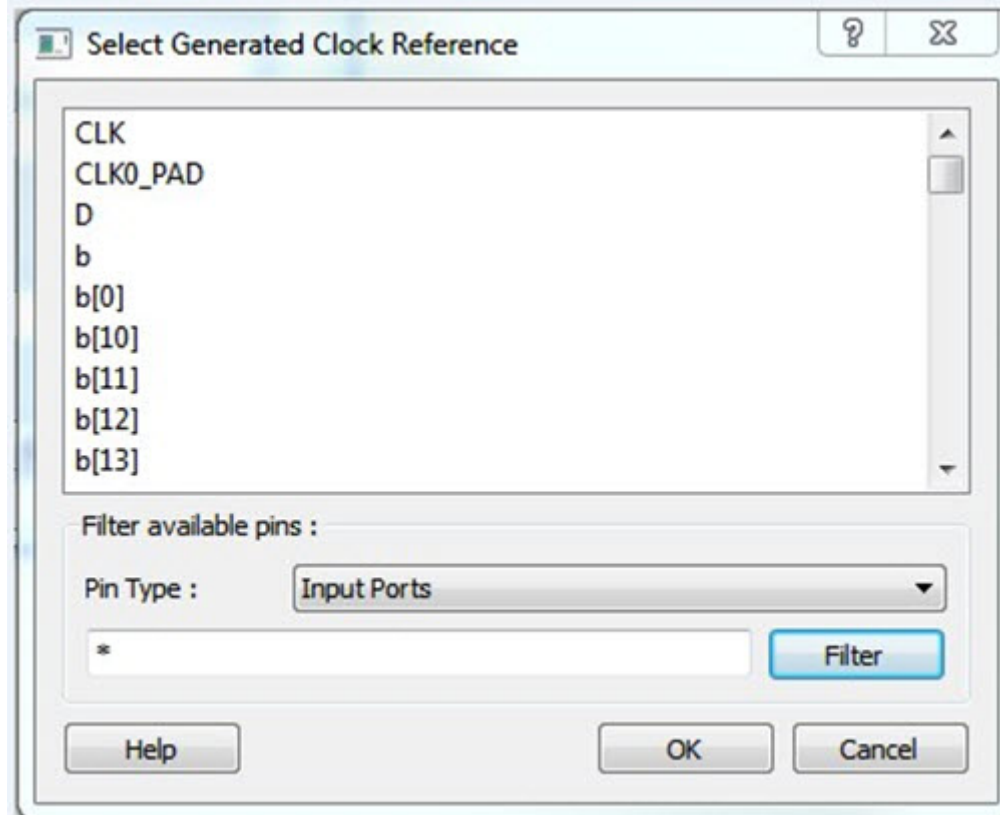


Figure 92 · Select Generated Clock Reference Dialog Box

Filter Available Pins

To identify any other pins in the design as the generated master pin, under **Filter available pins**, for Pin Type, select **Input Ports** or **All Pins**. You can also click filter the generated reference clock pin name in the displayed list. The default filter is *, which is a wild-card match for all.

See Also

[Specifying Generated Clock Constraints](#)

Select Generated Clock Source Dialog Box

Use this dialog box to find and choose the generated clock source from the list of available pins.

To open the Select Generated Clock Source dialog box (shown below) from the Constraints Editor, open the Create Generated Clock Constraint dialog box and click the browse button for the Clock Pin. The Selected Generated Clock Source dialog box appears.

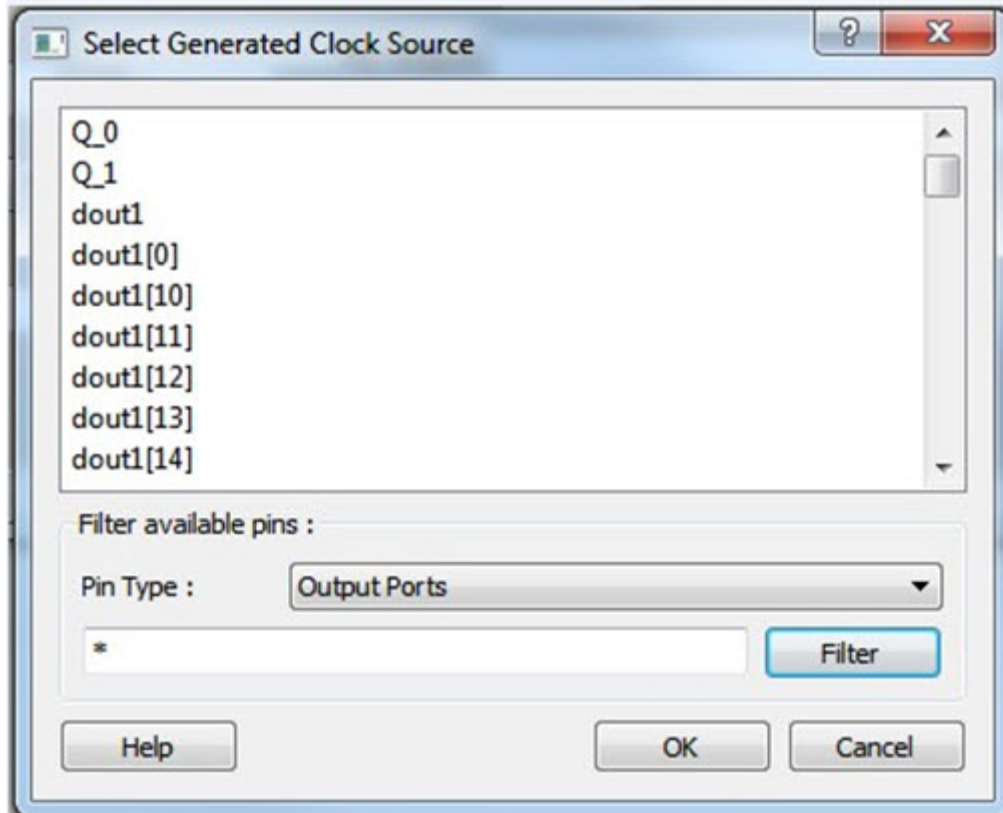


Figure 93 · Selected Generated Clock Source Dialog Box

Filter Available Pins

Explicit clock pins for the design is the default value. To identify any other pins in the design as the generated clock source pins, from the Pin Type pull-down list, select **All Ports**, **All Pins**, **All Nets**, or **All Register Output Pins**. You can also filter the generated clock source pin name in the displayed list. The default filter is *, which is a wild-card match for all.

See Also

[Specifying Generated Clock Constraint](#)

Select Ports Dialog Box

This dialog box appears when you click the browse button next to the Input Port field in the Set Input Delay Dialog Box or the Output Port field in the Set Output Delay Dialog Box. It also applies to the Set External Check & Set Clock To Output constraints.

The list box displays the available Pins. If you change the filter value, the list box shows the available pins based on the filter.

Use this dialog box to select the Input or Output Port:

- By explicit list
- By keyword and wildcard

By Explicit List

This is the default. This mode stores the actual Input/Out Port names. The following figure shows an example dialog box for the Select Input Port for Input Delay.

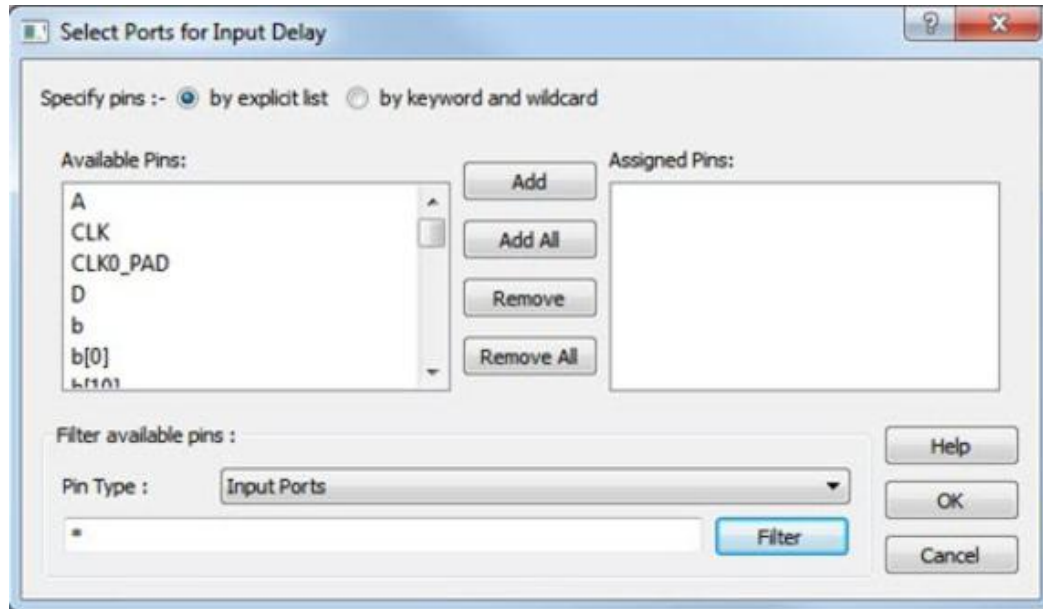


Figure 94 · Select Input Port for Input Delay Dialog Box

Available Pins

The list box displays the available Pins. If you change the filter value, the list box shows the available pins based on the filter.

Use **Add**, **Add All**, to add Pins from the Available Pins List or **Remove**, **Remove All** to delete Pins from the Assigned Pins list.

Filter Available Pins

Pin type – Specifies the filter on the available Pin Types: Input Port is the only valid type for Input Delay and Output Port is the only valid type for Output Delay.

Filter

Specifies the filter based on which the Available Pins list shows the Pin names. The default is *, which is a wild-card match for all. You can specify any string value.

By Keyword and Wildcard

This mode stores the filter only. It does not store the actual pin names. The constraints created using this mode get exported with the SDC accessors (`get_ports`) and the wildcard filter. The following figure shows an example dialog box for Select Output Ports by keyword and the *DM* filter.

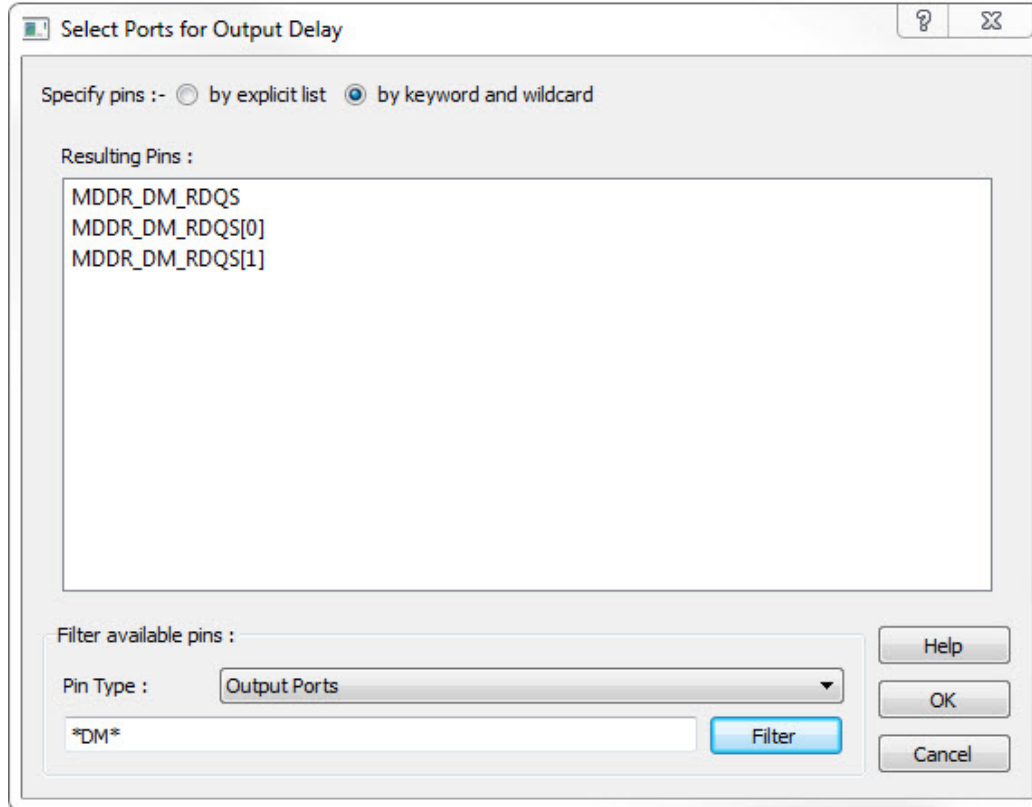


Figure 95 · Select Ports for Output Delay Dialog Box – By Keyword and Wildcard

Pin Type

Specifies the filter on the available pins. The valid values are Input Ports for Input Delay and Output Ports for Output Delay.

Filter

Specifies the filter based on which the Available Pins list shows the pin names. The default is *, which is a wild-card match for all. You can specify any string value.

Available Pins

Displays pins from the Pin Type based on the filter.

Select Source Clock for Clock-to-clock Uncertainty Constraint Dialog Box

This dialog box opens when you click the browse button for Source/From Clock for Clock-to-clock Uncertainty Constraints dialog box.

Use this dialog box to select Clock Pins:

- By explicit list
- By keyword and wildcard

To open the Select Source Clock dialog box, double-click **Constraint > Advanced > Clock Uncertainty**. Click the browse button to select the source.

By Explicit List

This is the default. This mode stores the actual Clock Pin names. The following figure shows an example dialog box for Select Source Clock List for Clock-to-Clock Uncertainty Constraint Dialog Box .

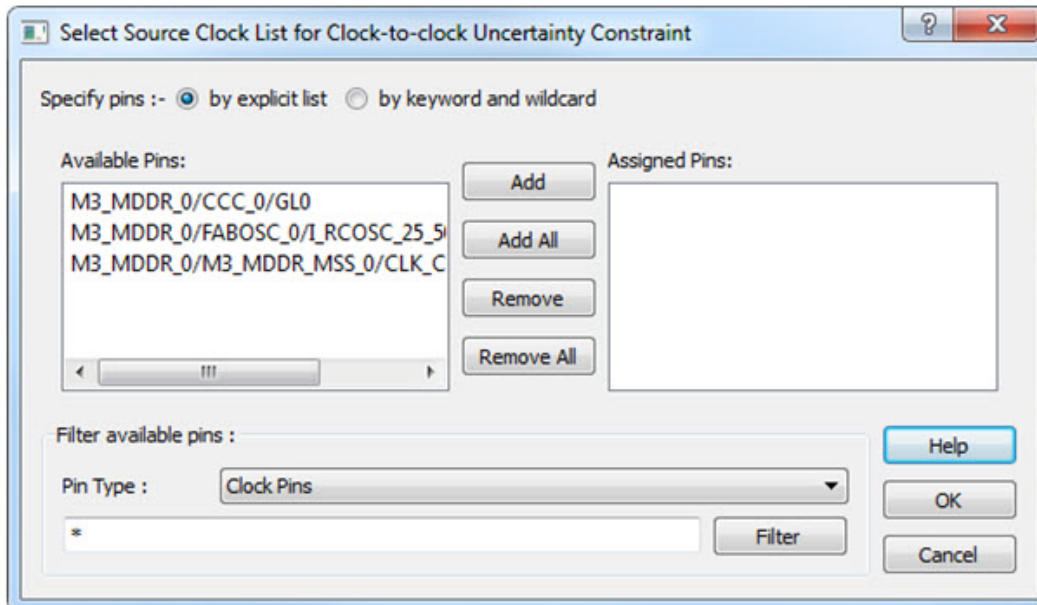


Figure 96 · Select Source Clock List for Clock-to-Clock Uncertainty Constraint Dialog Box – By Explicit List

Available Pins

The list box displays the available Clock Pins. If you change the filter value, the list box shows the available pins based on the filter.

Use Add, Add All, to add Clock Pins from the Available Pins List or Remove, Remove All to delete Clock Pins from the Assigned Pins list.

Filter Available Pins

Pin type – Specifies the filter on the available Clock Pins.

Filter

Specifies the filter based on which the Available Pins list shows the Clock Pin names. The default is *, which is a wild-card match for all. You can specify any string value.

By Keyword and Wildcard

This mode stores the filter only. It does not store the actual pin names. The constraints created using this mode get exported with the SDC accessors (get_pins) and the wildcard filter. The following figure shows an example dialog box for Select Source Clock Pins by keyword and the *CCC* filter.

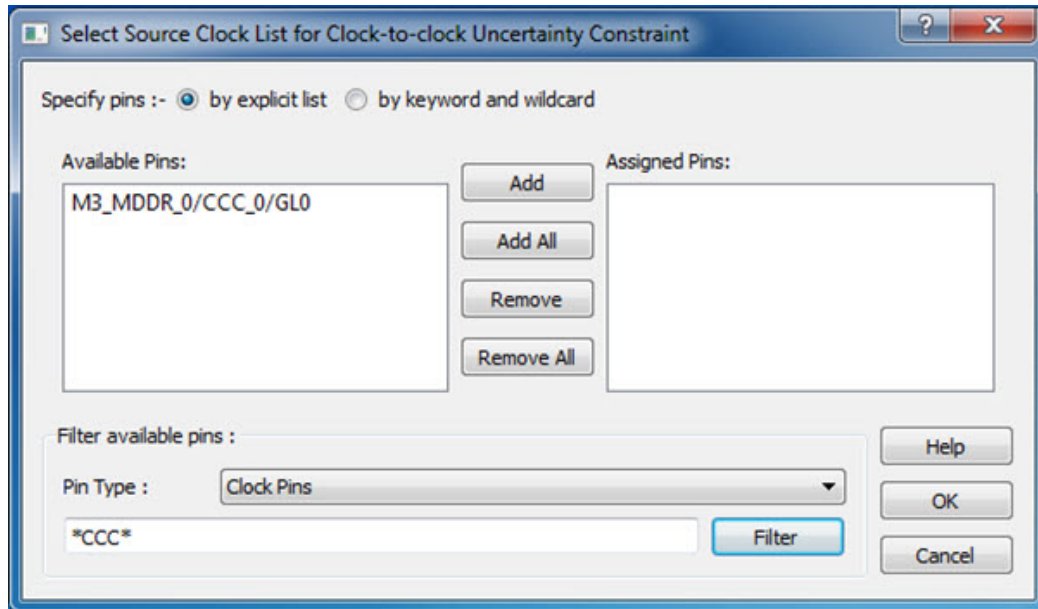


Figure 97 · Select Source Clock List for Clock-to-Clock Uncertainty Constraint Dialog Box – By Keyword and Wildcard

Pin Type

Specifies the filter on the available pins. The only valid selection is Clock Pins.

Filter

Specifies the filter based on which the Available Pins list shows the pin names. The default is *, which is a wild-card match for all. You can specify any string value.

Resulting Pins

Displays pins from the available pins based on the filter.

Select Source or Destination Pins for Constraint Dialog Box

This dialog box opens when you select the browse button for Source/From, Intermediate/Through and Destination/To pins for Timing Exception Constraints: False Path Constraints, Multicycle Path Constraints, and Maximum/Minimum Delay Constraints.

Use this dialog box to select pins or ports:

- By explicit list
- By keyword and wildcard

To open the Select Source or Destination Pins for Constraint dialog box from the Constraints Editor, choose **Constraint > Timing Exception Constraint Name**. Click the browse button to select the source.

By Explicit List

This is the default. This mode stores the actual pin names. The following figure shows an example dialog box for Select Source Pins for Multicycle Constraint (specify by explicit list).

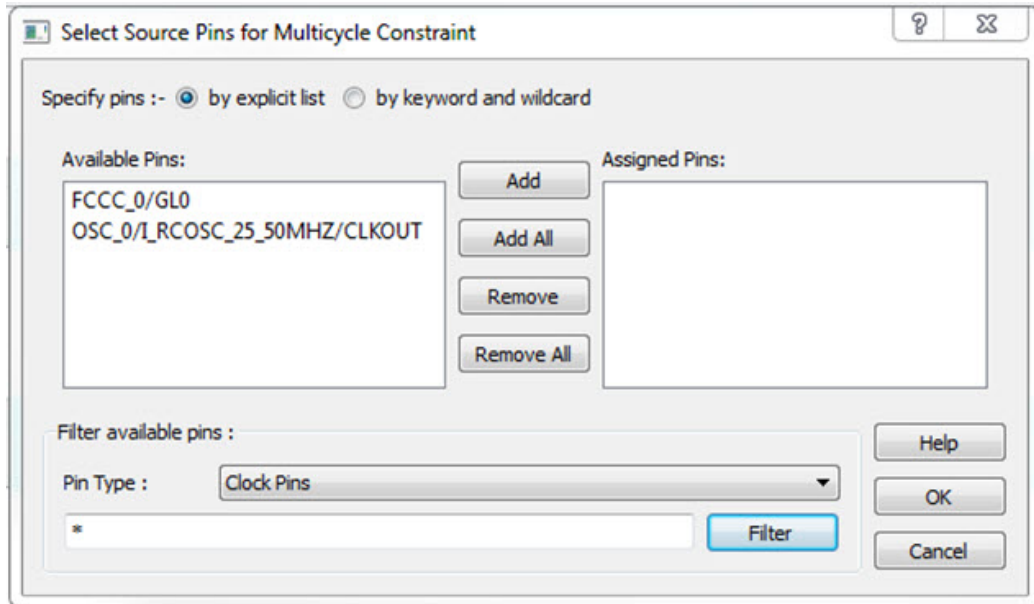


Figure 98 · Select Source Pins for Multicycle Constraint Dialog Box (specify by explicit list)

Available Pins

The list box displays the available pins. If you change the filter value, the list box shows the available pins based on the filter.

Click **Add**, **Add All**, **Remove**, and **Remove All** to add or delete pins from the Assigned Pins list.

Filter Available Pins

Pin Type – Specifies the filter on the available pins. You can specify Input Ports, Clock Pins, All Register Clock Pins.

Filter

Specifies the filter based on which the Available Pins list shows the pin names. The default is *, which is a wild-card match for all. You can specify any string value.

By Keyword and Wildcard

This mode stores the filter only. It does not store the actual pin names. The constraints created using this mode get exported with the SDC accessors (`get_ports`, `get_pins`, etc.) and the wildcard filter. The following figure shows an example dialog box for Select Source Pins for Multicycle Constraint (specified by keyword and wildcard).

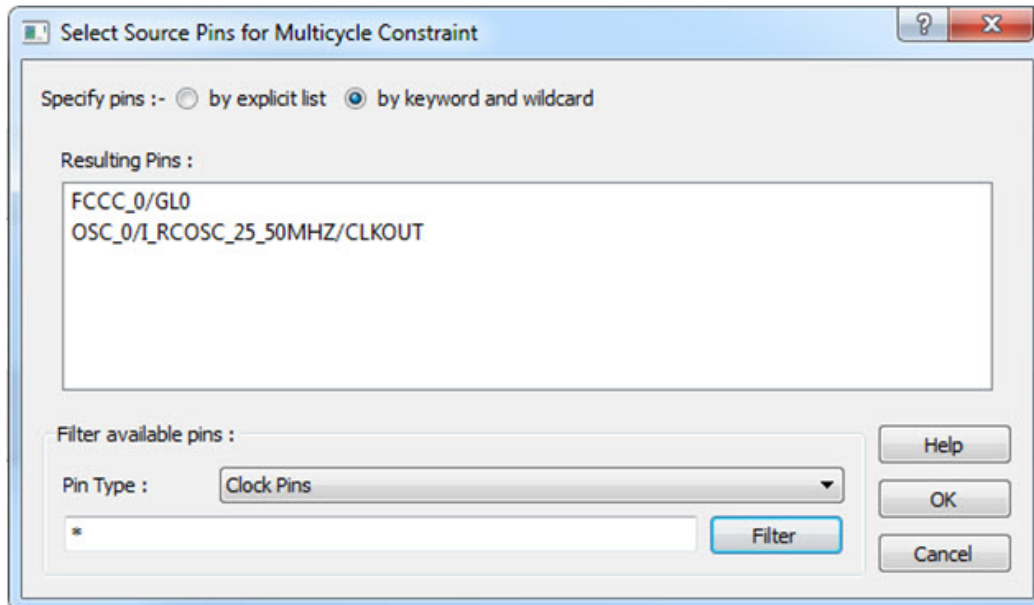


Figure 99 · Select Source Pins for Multicycle Constraint Dialog Box (specified by keyword and wildcard)

Pin Type

Specifies the filter on the available pins. The source pins can be Clock Pins, Input Ports, All Register Clock Pins. The default pin type is Clock Pins. The available Pin Type varies with Source Pins, Through Pins, and Destination Pins.

Filter

Specifies the filter based on which the Available Pins list shows the pin names. The default is *, which is a wild-card match for all. You can specify any string value.

Resulting Pins

Displays pins from the available pins based on the filter.

Select Source Pins for Clock Constraint Dialog Box

Use this dialog box to find and choose the clock source from the list of available pins.

To open the Select Source Pins for the Clock Constraint dialog box (shown below) from the Constraints Editor, click the browse button to the right of the Clock source field in the Create Clock Constraint dialog box.

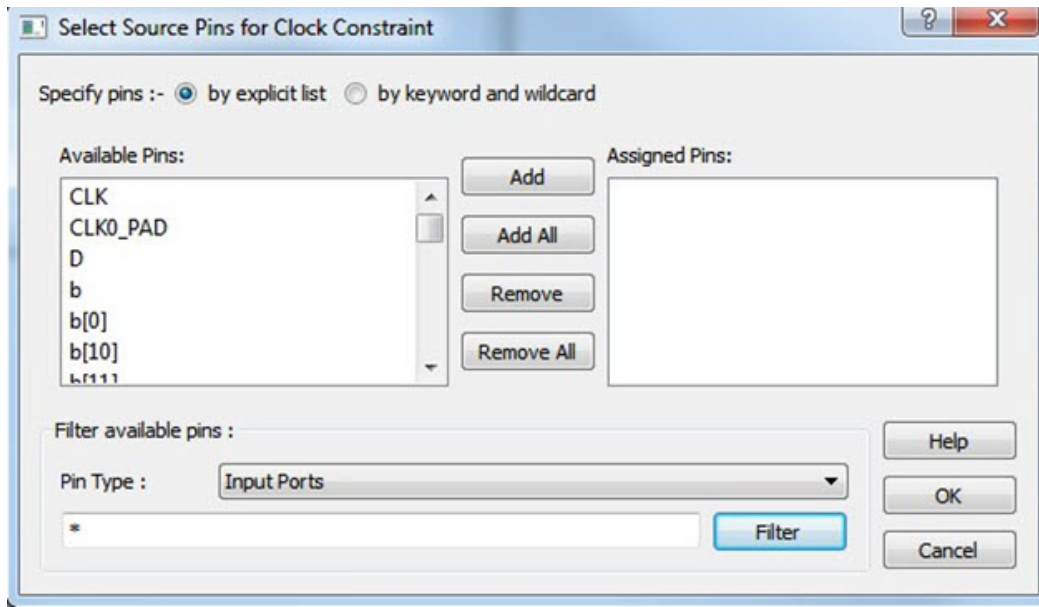


Figure 100 · Select Source Pins for Clock Constraint Dialog Box

Select a Pin

Displays all available pins.

Filter Available Objects

Explicit clock pins for the design is the default value. To identify any other pins in the design as clock pins, under **Filter available pins**, for Pin Type, select **Input Ports**, **All Pins**, or **All Nets**. You can also filter the clock source pin name in the displayed list. The default filter is *, which is a wild-card match for all.

See Also

[Specifying Clock Constraints](#)

Select Through Pins for Timing Exception Constraint Dialog Box

This dialog box opens when you select the Browse button for Intermediate/Through Pins for False Path, Multicycle Path, Min/Max Delay Constraints dialog box.

Use this dialog box to select the Intermediate Pin:

- By explicit list
- By keyword and wildcard

To open the Select Through Pins dialog box, double-click **Constraint > Exceptions > Max/Min Delay/False Path/Multicycle Path**. Click the browse button next to the To the Through field to select the Intermediate/Through Pin.

By Explicit List

This is the default. This mode stores the actual Intermediate/Through Pin names. The following figure shows an example dialog box for Select Through Pins for Multicycle Path Constraint.

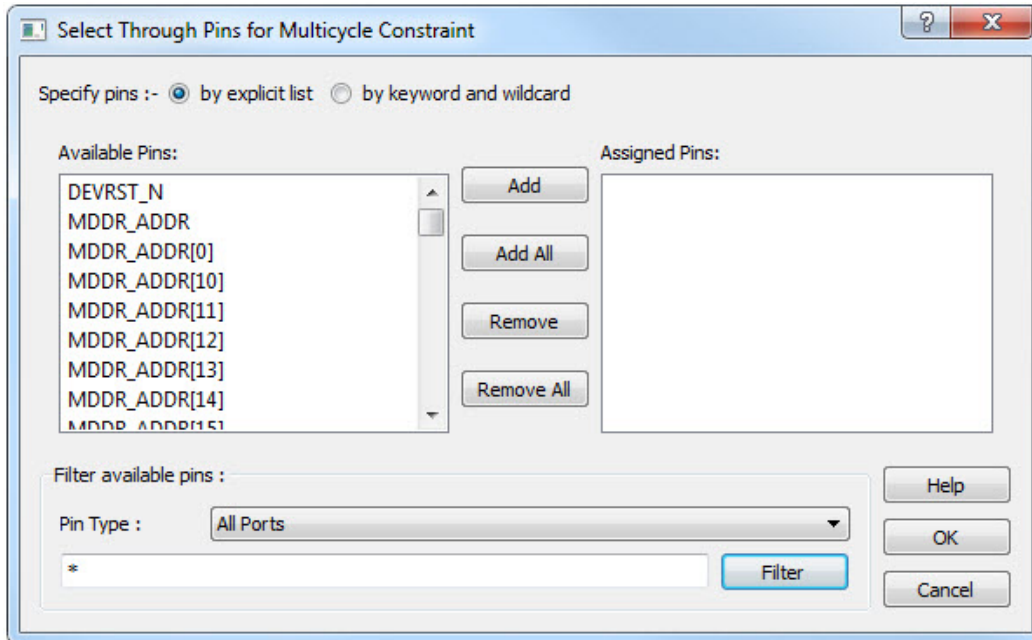


Figure 101 · Select Through Pins for Multicycle Path Constraint Dialog Box – By Explicit List

Available Pins

The list box displays the available Pins. If you change the filter value, the list box shows the available pins based on the filter.

Use **Add**, **Add All**, to add Pins from the Available Pins List or **Remove**, **Remove All** to delete Pins from the Assigned Pins list.

Filter Available Pins

Pin type – Specifies the filter on the available Pin Types: All Ports, All Nets, All Pins and All Instances.

Filter

Specifies the filter based on which the Available Pins list shows the Pin names. The default is *, which is a wild-card match for all. You can specify any string value.

By Keyword and Wildcard

This mode stores the filter only. It does not store the actual pin names. The constraints created using this mode get exported with the SDC accessors (`get_pins`) and the wildcard filter. The following figure shows an example dialog box for Select Through Pins by keyword and the *DM* filter.

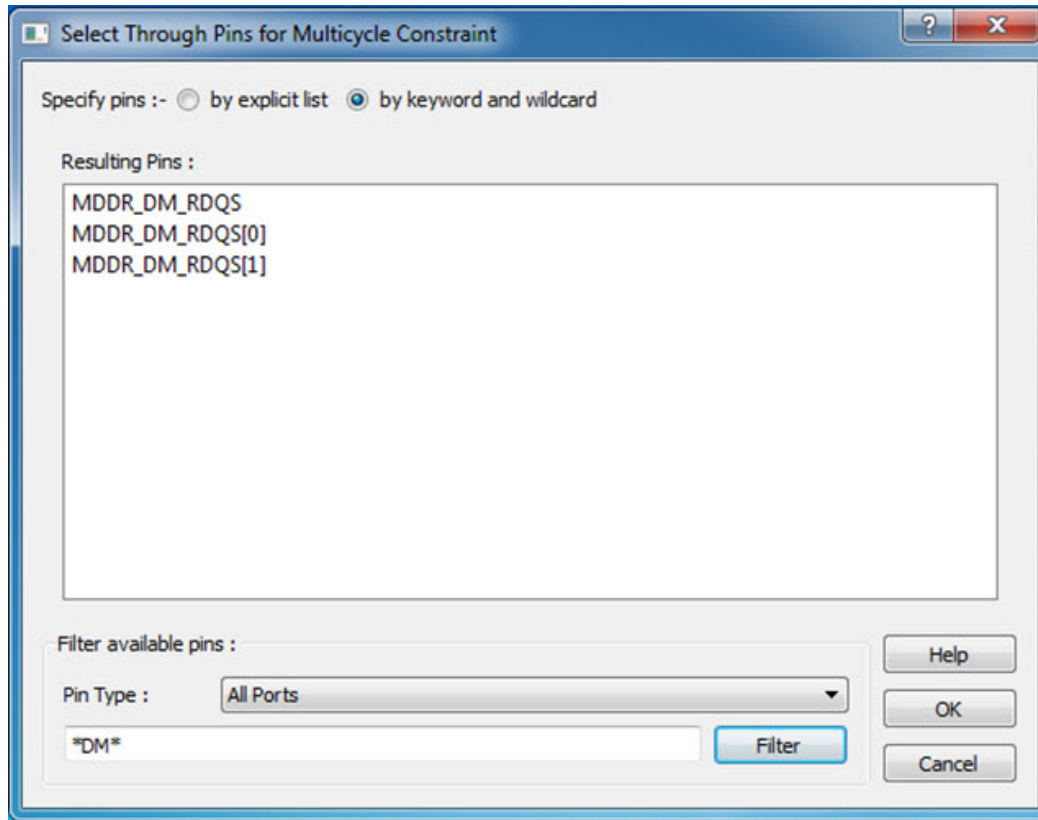


Figure 102 · Select Through Pins for Multicycle Path Constraint Dialog Box – By Keyword and Wildcard

Pin Type

Specifies the filter on the available pins. The valid values are All Ports, All Nets, All Pins and All Instances.

Filter

Specifies the filter based on which the Available Pins list shows the pin names. The default is *, which is a wild-card match for all. You can specify any string value.

Resulting Pins

Displays pins from the available pins based on the filter.

Implement Design in Enhanced Constraint Flow for SmartFusion2, IGLOO2, and RTG4

Synthesize (Enhanced 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.

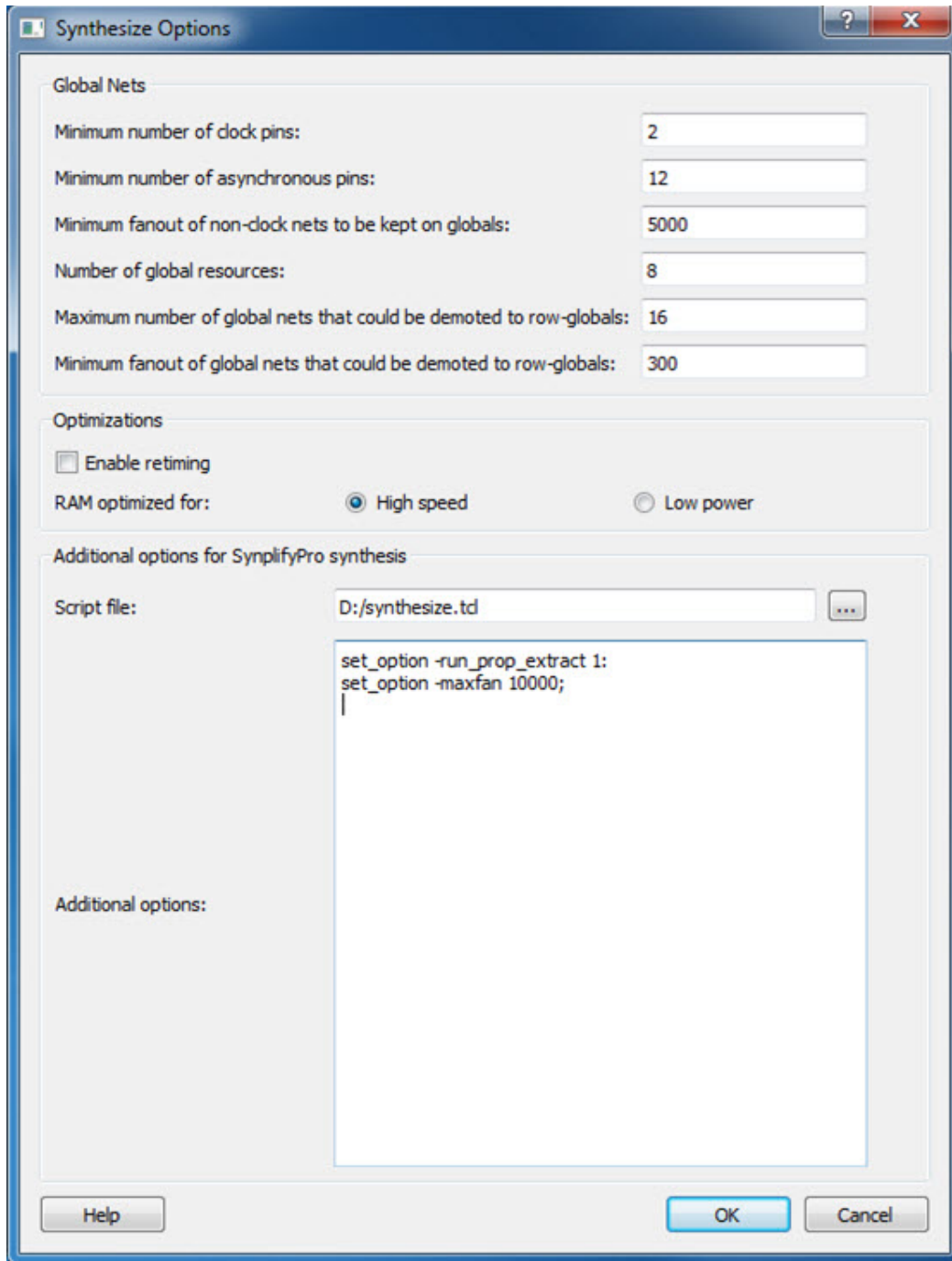


Figure 103 · Synthesize Options Dialog Box

HDL Synthesis Language Settings

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

Global Nets (Promotions and 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 RT4G150 die of the RTG4 family. This option is not available for RT4G150_ES die.
- **Minimum fanout of non-clock nets to be kept on globals** – Specifies the threshold value for data pin promotion to global resources. It is the minimum fanout of non-clock (data) nets to be kept on globals (no demotion). The default is 5000 (must be between 1000 and 200000).
- **Number of global resources** – This can be used to control number of Global resources you want to use in your design. By default this displays the number of available global resources for the die you have selected for the project and varies with different die sizes.
- **Maximum number of global nets that could be demoted to row-globals** – Specifies the maximum number of global nets that could be demoted to row-globals. The default is 16 (must be between 0 to 50).
- **Minimum fanout of global nets that could be demoted to row-globals** – Specifies the minimum fanout of global nets that could be demoted to row-global. It is undesirable to have high fanout nets demoted using row globals because it may result in high skew. The default is 300. (Must be between 25 to 5000). If you run out of global routing resources for your design, reduce this number (to allow more globals to be demoted to Row Globals) or select a bigger die for your design.

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

Optimizations


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

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

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

Additional options for Synplify Pro synthesis

Script File

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

Additional Options

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

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

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

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

```
set_option -use_fsm_explorer 0/1
set_option -frequency 200.000000
```

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

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

Refer to the Synplify Pro Reference Manual for detailed information about the options and supported families.

The following options are already set by Libero. Do not include them in the additional options field or Script File:

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

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

Synplify Pro ME

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

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

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

To run synthesis using Synplify Pro ME with custom settings:

1. If you have set Synplify as your default synthesis tool, right-click **Synthesize** in the Libero SoC Design Flow 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.

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 Instrumentor UI specify the sample clock, the breakpoints, and other signals to probe. Synplify creates a new synthesis implementation. Synthesize the design.
8. In Libero SoC, 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. Run Compile 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.

Verify Post-Synthesis Implementation - Simulate

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

To perform functional simulation:

1. If you have not done so, back-annotate your design and create your testbench.
2. Right-click **Simulate** (in Design Flow window, Implement Design > Verify Post-Synthesis Implementation > Simulate) and choose **Organize Input Files > Organize Source 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 AE, right-click **Simulate** in the Design Hierarchy window and choose **Open Interactively**. ModelSim starts and compiles the appropriate source files. When the compilation completes, the simulator runs for 1 μ s and the Wave window opens to display the simulation results.
5. Scroll in the Wave window to verify the logic works as intended. Use the cursor and zoom buttons to zoom in and out and measure timing delays.
6. When you are done, from the **File** menu, choose **Quit**.

Compile Netlist (Enhanced Constraint Flow)

The Compile Netlist step appears in the Design Flow window for Enhanced Constraint Flow only when the design source is EDIF (EDIF design flow). To enable the EDIF design flow, turn off the Enable Synthesis option in the Project Settings > Design Flow page.

When the design source is HDL/SmartDesign, this Compile Netlist step is not available in the Design Flow window.

Options

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

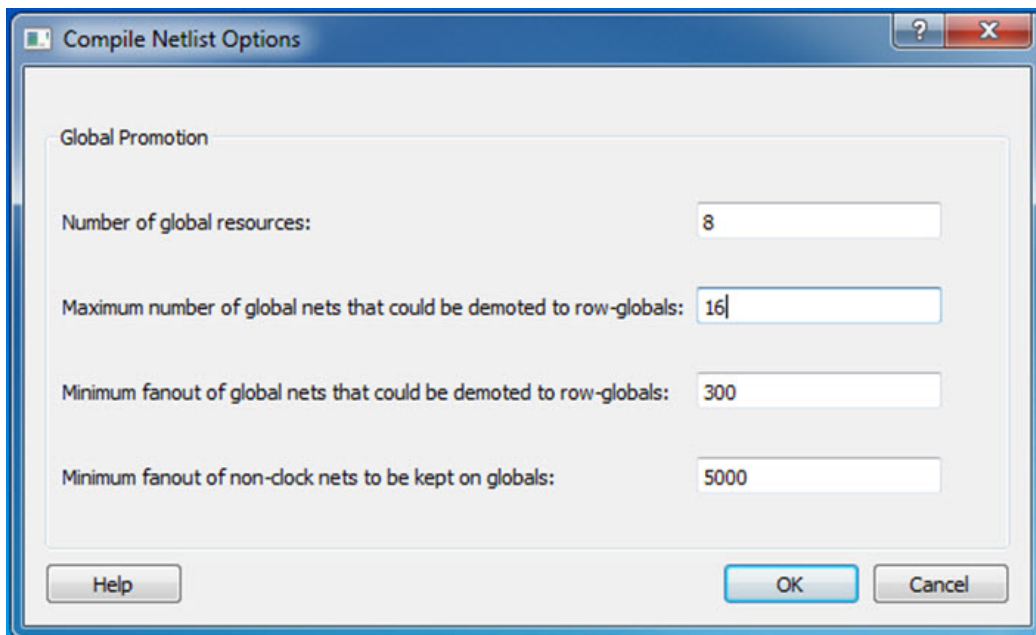


Figure 104 · Compile Netlist Options Dialog Box

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

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

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

Minimum fanout of global nets that could be demoted to row-globals – Specifies the minimum fanout of global nets that can be demoted to row-global. The default is 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 larger die for your design.

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

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

Resource Usage (SmartFusion2, IGLOO2, RTG4)

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

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

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

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

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

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

Overlapping of Resource Reporting

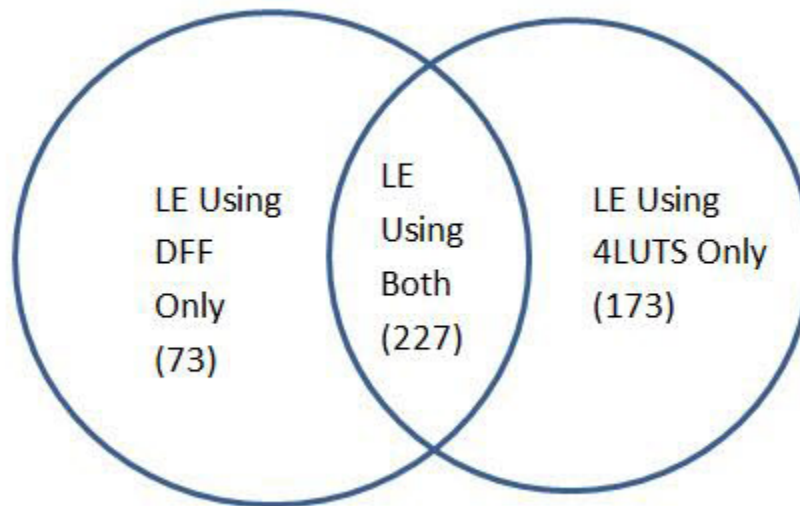
The number of 4LUTs in the report are the total number used for your design, regardless of whether or not they are combined with the DFFs. Similarly, the number of DFFs in the report are the total number used for your design, regardless of whether or not they are combined with 4LUT's.

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

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

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

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








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

Enhanced Constraint Flow in Implementation


Design State Invalidation – Enhanced Constraint Flow

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

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

Constraints and Design Invalidation

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

The out-of-date design state is identified by the  icon in the Design Flow window.

Sources and/or settings are defined as:

- HDL sources (for Synthesis), gate level netlist (for Compile), and Smart Design and System Builder components
- Design Blocks (*.cxz files) – low-level design units which may have completed Place and Route and re-used as components in a higher-level design
- Constraint files associated with a tool
- Upstream tools in the Design Flow:
 - o If the tool state of a Design Flow tool changes from PASS to OUT OF DATE, the tool states of all the tools below it in the Design Flow, if already run and are in PASS state, also change to OUT OF DATE with appropriate tooltips. For example, if the Synthesis tool state changes from PASS to OUT OF DATE, the tool states of Place and Route tool as well as all the tools below it in the Design Flow change to OUT OF DATE.
 - o If a Design Flow tool is CLEANED, the tool states of all the tools below it in the Design Flow, if already run, change from PASS to OUT OF DATE.
 - o If a Design Flow tool is rerun, the tool states of all the tools below it in the Design Flow, if already run, are CLEANED.
- Tool Options

Implement Design in Enhanced Constraint Flow for SmartFusion2, IGLOO2, and RTG4

- o If the configuration options of a Design Flow tool (right-click the tool and choose **Configure Options**) are modified, the tool states of that tool and all the other tools below it in the Design Flow, if already run, are changed to OUT OF DATE with appropriate tooltips.
- Project Settings:
 - o Device selection
 - o Device settings
 - o Design Flow
 - o Analysis operating conditions

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

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

Setting Changed	Applicable Families	Note	Design Flow Tools Affected	New State of the Affected Design Flow Tools
Abort flow if errors are found in SDC	SmartFusion2, IGLOO2, RTG4		None	N/A
Temperature range(C)	SmartFusion2, IGLOO2, RTG4		Verify Timing, Post Layout Simulate, and Verify Power	OUT OF DATE
Core voltage range(V)	SmartFusion2, IGLOO2, RTG4		Verify Timing, Post Layout Simulate, and Verify Power	OUT OF DATE
Default I/O voltage range	SmartFusion2, IGLOO2, RTG4		Verify Timing, Post Layout Simulate, and Verify Power	OUT OF DATE
Radiation (krad)	RTG4		Verify Timing, Post Layout Simulate, and Verify Power	OUT OF DATE
Enable Single Event Transient mitigation	RTG4		Verify Timing, Post Layout Simulate, Verify Power, Generate FPGA Array Data and all other "Program and Debug Design" tools below it	OUT OF DATE

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

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

Edit Constraints

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

The Interactive Tool to edit I/O Attributes is the I/O Editor. The Interactive Tool to edit Floorplanning Constraints is the Chip Planner. The Interactive Tool to edit Timing Constraints is the Constraint Editor.

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

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

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

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

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

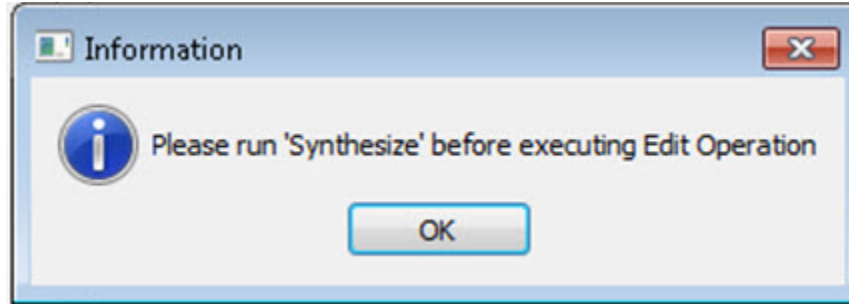
Constraints in the SDC constraint file that are read by the Constraint Editor and subsequently modified by you will be written back to the SDC file when you save the edits and close the Constraint Editor.

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

Constraint Type and Interactive Tool

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

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



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

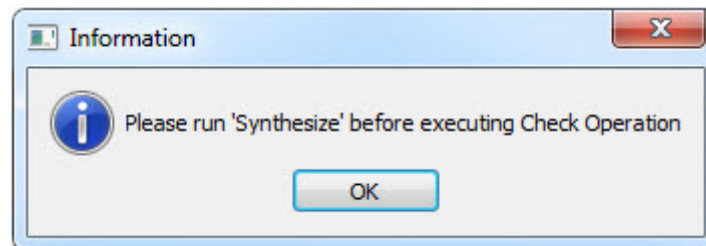
Check Constraints

When a constraint file is checked, the Constraint Checker:

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

Design State and Constraints Check

Constraints can be checked only when the design is in the right state. A pop-up message appears when the check is made and the design flow has not reached the right state,



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

Constraint Type	Check for Tools	Required Design State Before Checking	Netlist Used for Design Objects Checks	Check Result
Attributes				Window
Netlist Attributes	NDC Check	Pre-Synthesis	RTL Netlist	Reported in Libero Log Window

Place and Route - SmartFusion2, IGLOO2, RTG4

To change your Place and Route settings:

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

Timing-Driven

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

If you do not select Timing-driven Place and Route, timing constraints are not considered by the software, although a delay report based on delay constraints entered in SmartTime can still be generated for the design.

Power-Driven

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, and 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.

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

[Multiple Pass Layout Configuration \(SmartFusion2, IGLOO2, RTG4\)](#)
[extended_run_lib - Libero SoC Only](#)

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.

- Click **Configure**. The Multi-Pass Configuration dialog box appears.

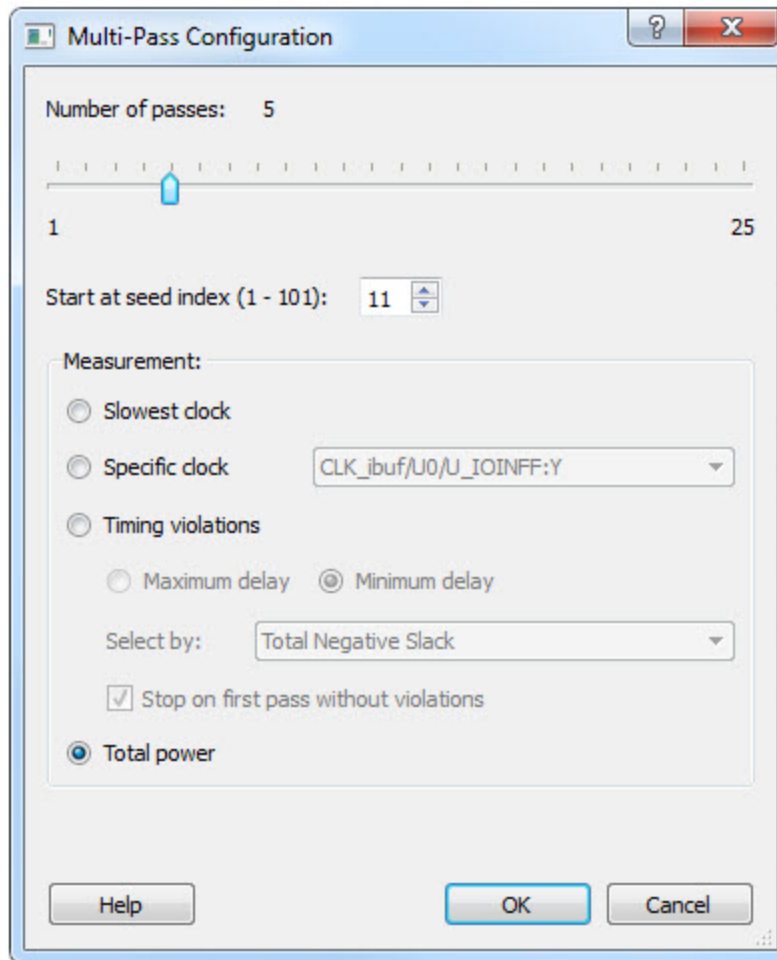


Figure 105 · Multi-Pass Configuration Dialog Box

- Set the options and click **OK**.

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

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

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

- **Slowest clock:** Select to use the slowest clock frequency in the design in a given pass as the performance reference for the layout pass.
- **Specific clock:** Select to use a specific clock frequency as the performance reference for all layout passes.

Timing violations: This is the default. Select Timing Violations to use the pass that best meets the slack or timing-violations constraints.

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

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

- When Worst Slack (default) is selected, the largest amount of negative slack (or least amount of positive slack if all constraints are met) for each pass is identified, and the largest value of all passes determines the best pass.
- When Total Negative Slack is selected, the sum of negative slacks from the first 100 paths in the Timing Violations report for each pass is identified, and the largest value of all the passes determines the best pass. If no negative slacks exist for a pass, the worst slack is used to evaluate that pass.
- Stop on first pass without violations: Select to stop performing remaining passes if all timing constraints have been met (when there are no negative slacks reported in the timing violations report).
- **Total power:** Select to determine the best pass to be the one that has the lowest total power (static + dynamic) of all layout passes.

Iteration Summary Report

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

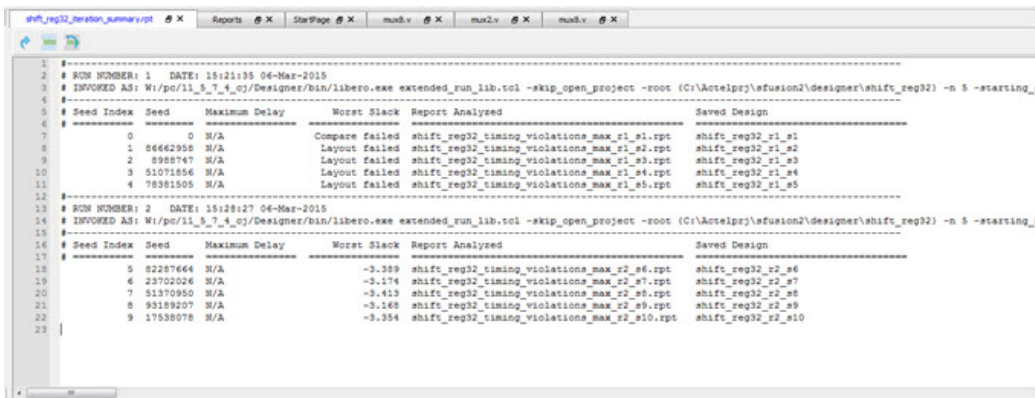


Figure 106 - Iteration Summary Report

See Also

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

Export Back Annotated Files

Libero SoC uses post-layout files for back-annotated timing simulation.

Post-layout files include:

- *.ba.sdf - Standard Delay Format for back-annotation to the simulator.
- *.ba.v/hvd - 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 a post-layout file, in the Design Flow window click **Implement Design** and double-click **Export Back Annotated Files**.

If you wish to export the Back Annotated files with options that are different than the default, right-click **Export Back Annotated Files** and choose **Open Interactively**.

Verify Post Layout Implementation

Generate Back Annotated Files - SmartFusion2, IGLOO2, and RTG4 Only

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-placement netlist used for back-annotated timing simulation.

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.

Simulate - Opens ModelSim AE

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 AE. 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 Source 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 AE, right-click **Simulate** in the Design Hierarchy window and choose **Open Interactively**. ModelSim starts and compiles the appropriate source files. When the compilation completes, the simulator runs for 1 μ s and the Wave window opens to display the simulation results.
5. Scroll in the Wave window to verify the logic works as intended. Use the cursor and zoom buttons to zoom in and out and measure timing delays. If you did not create a testbench with WaveFormer Pro, you may get error messages with the vsim command if the instance names of your testbench do not follow the same conventions as WaveFormer Pro. Ignore the error message and type the correct vsim command.
6. When you are done, from the **File** menu, choose **Quit**.

Verify Timing

Verify Timing Configuration

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

For the timing and timing violation reports you can select:

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

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 configuration dialog box for various operating conditions and report selections.

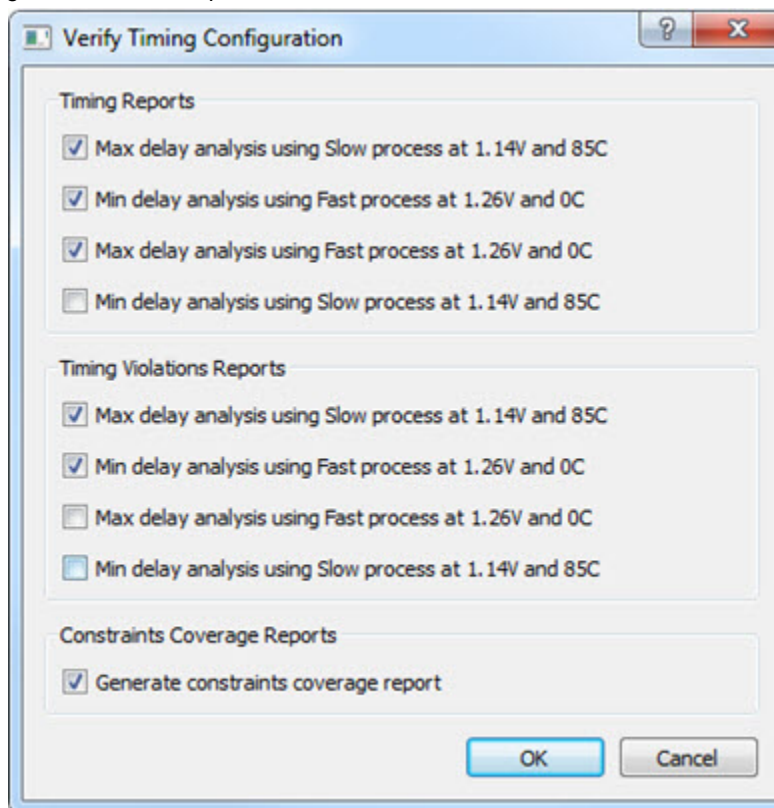


Figure 107 · Temperature/Voltage Value for Operating Conditions: COM (Commercial)

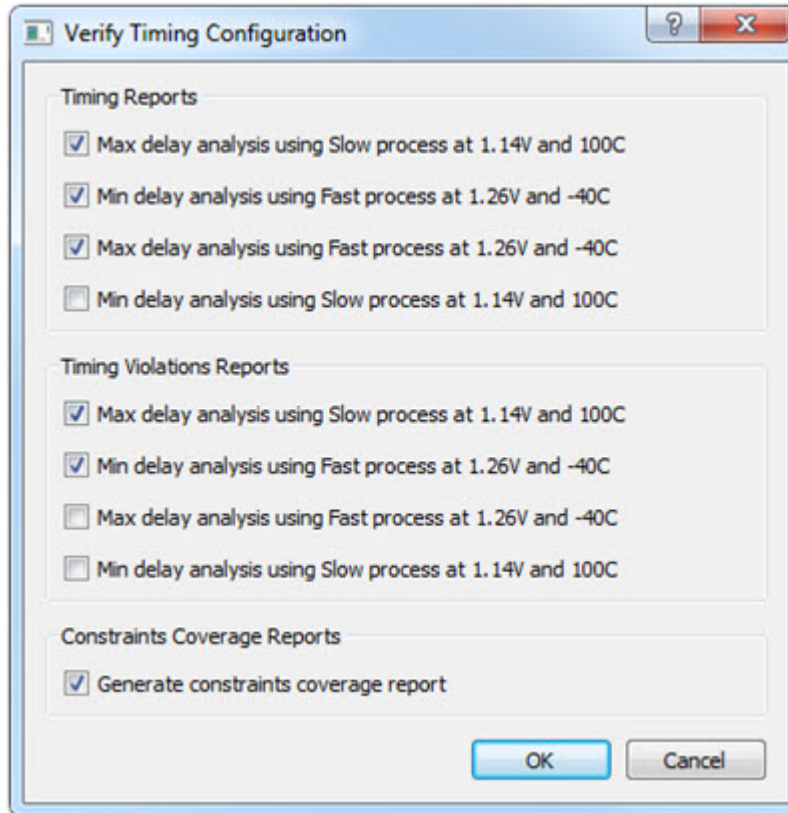


Figure 108 · Temperature/Voltage Value for Operating Conditions: IND (Industrial)

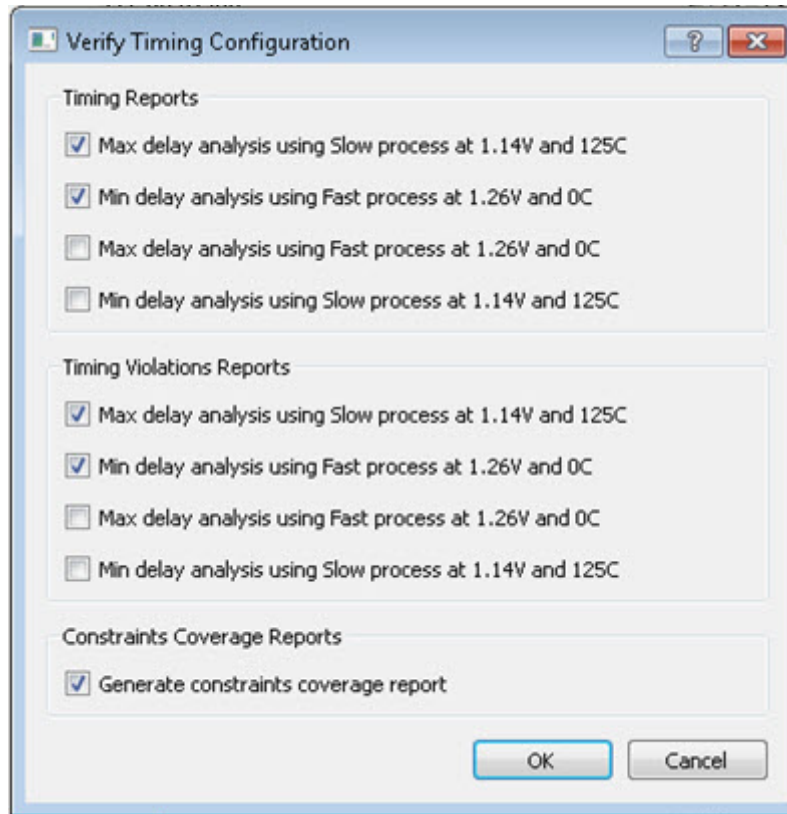


Figure 109 · Temperature/Voltage value for CUSTOM Temperature (0°C to 125°C) and IND Voltage

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>_min_timing_fast_<hv>_<lt>.xml
- Max Delay Static Timing Analysis report based on Fast process, High Voltage and Low Temperature operating conditions.
<root>_max_timing_fast_<hv>_<lt>.xml (generated by default)
- Min Delay Static Timing Analysis report based on Slow process, Low Voltage and High Temperature operating conditions.
<root>_min_timing_slow_<lv>_<ht>.xml

Timing violations reports – These reports display timing information organized by clock domain. Four types of timing violations reports are available. You can configure which reports to generate using the 'Verify Timing' configuration dialog (**Design Flow > Verify Timing > Configure Options**). The following reports can be generated:

- Max Delay Static Timing Analysis report based on Slow process, Low Voltage and High Temperature operating conditions.
`<root>_max_timing_slow_violations_<lv>_<ht>.xml` (generated by default)
- Min Delay Static Timing Analysis report based on Fast process, High Voltage and Low Temperature operating conditions.
`<root>_max_timing_violations_fast_<hv>_<lt>.xml`
- Max Delay Static Timing Analysis report based on Fast process, High Voltage and Low Temperature operating conditions.
`<root>_min_timing_fast_violations_<hv>_<lt>.xml` (generated by default)
- Min Delay Static Timing Analysis report based on Slow process, Low Voltage and High Temperature operating conditions.
`<root>_min_timing_slow_violations_<lv>_<ht>.xml`

Constraints coverage report – This report displays the overall coverage of the timing constraints set on the current design.

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

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

`<root>_timing_combinational_loops.xml` (always generated)

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

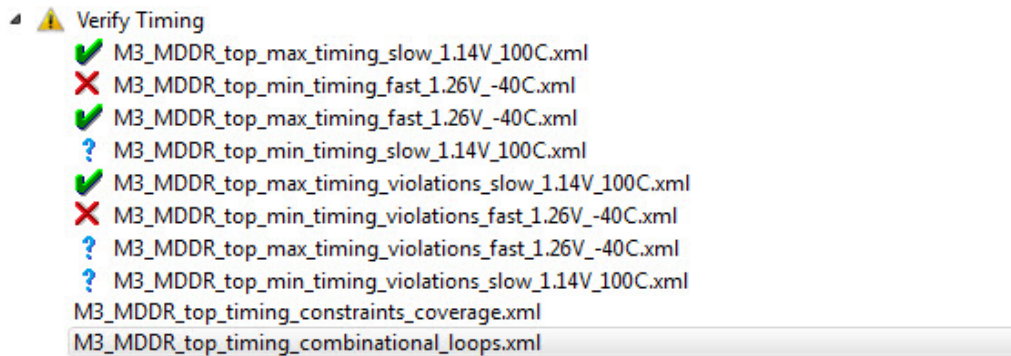


Figure 110 · Reports Example

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

SmartTime (Enhanced Constraint Flow)

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

Note: SmartTime in the Enhanced Constraint Flow has changed. See the [Timing Constraints Editor](#) for help with creating and editing timing constraints in the Enhanced Constraints Flow.

Static Timing Analysis (STA)

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

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

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

Timing Constraints

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

Timing Analysis

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

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

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

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

SmartTime and Place and Route

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

SmartTime and Timing Reports

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

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

SmartTime and Cross-Probing into Chip Planner

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

- Ports
- Macros
- Timing Paths

Refer to the SmartTime User's Guide for details ([Libero SoC > Help > Reference Manual > SmartTime User's Guide](#)).

SmartTime and Cross-Probing into Constraint Editor

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

Refer to the [SmartTime for Libero In Enhanced Constraint Flow User's Guide](#) for details.

Verify Power

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

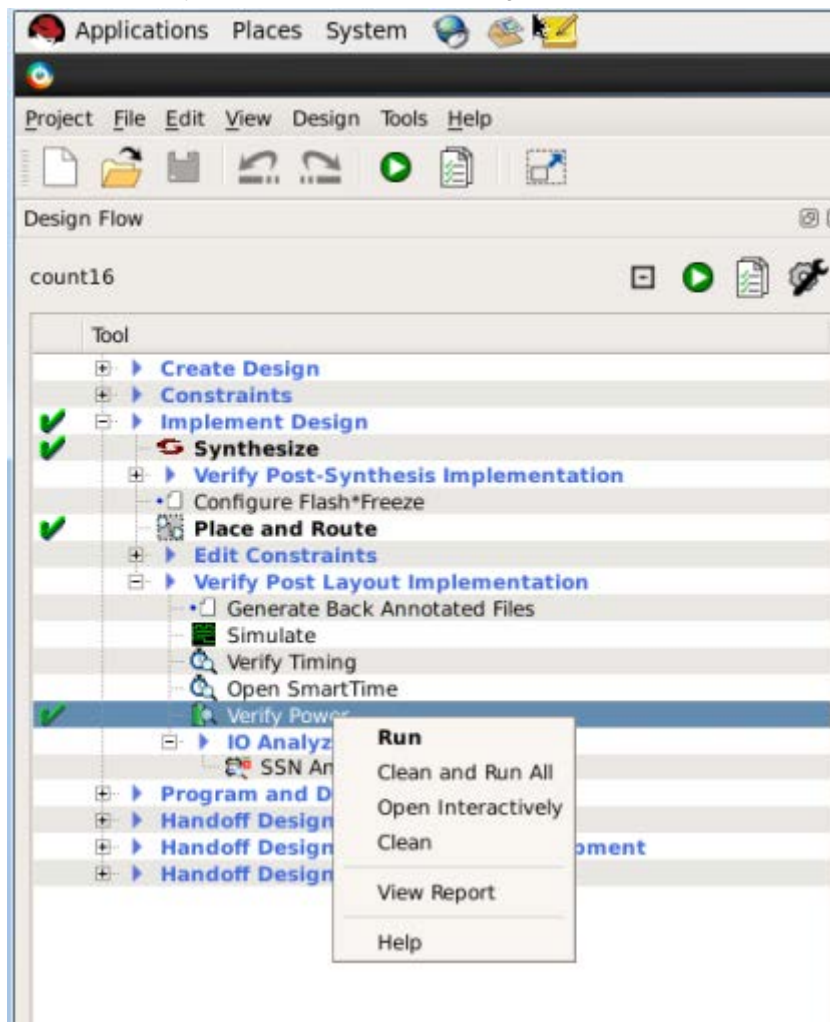



Figure 111 · Verify Power right-click menu

Verify Power sub-commands

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

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

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

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

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

SmartPower

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

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

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

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

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

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

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

See [SmartPower For Libero SoC User Guide](#)

IO Advisor (SmartFusion2, IGLOO2, and RTG4)

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

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

To open the IO Advisor, from Libero SoC's Design Flow window, right-click **IO Advisor** and choose **Open Interactively (IO Advisor > Open Interactively)** or double-click **IO Advisor**.

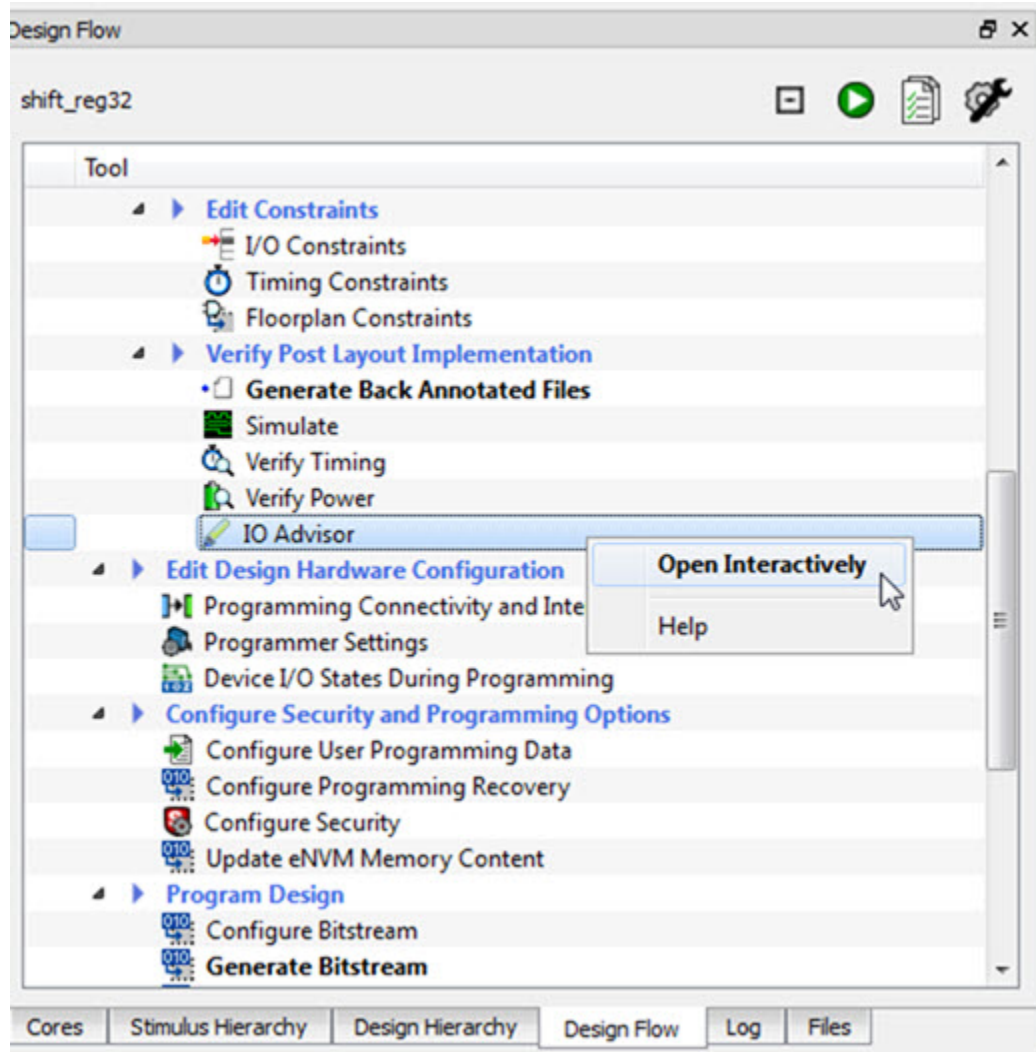


Figure 112 · IO Advisor > Open Interactively

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.

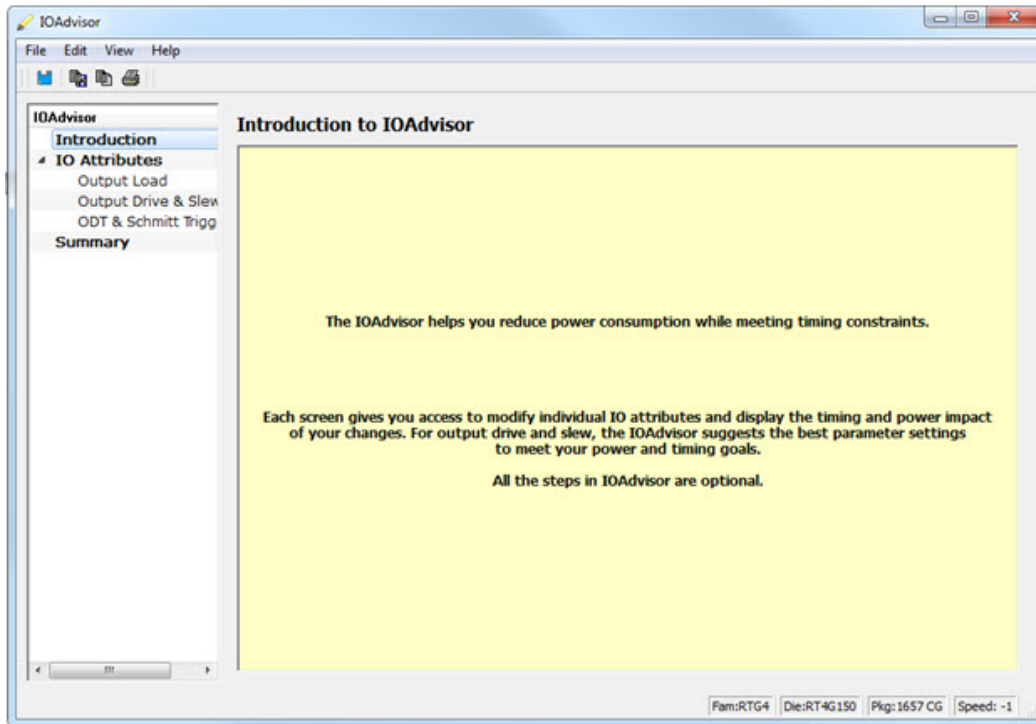


Figure 113 · IO Advisor - Introduction

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.

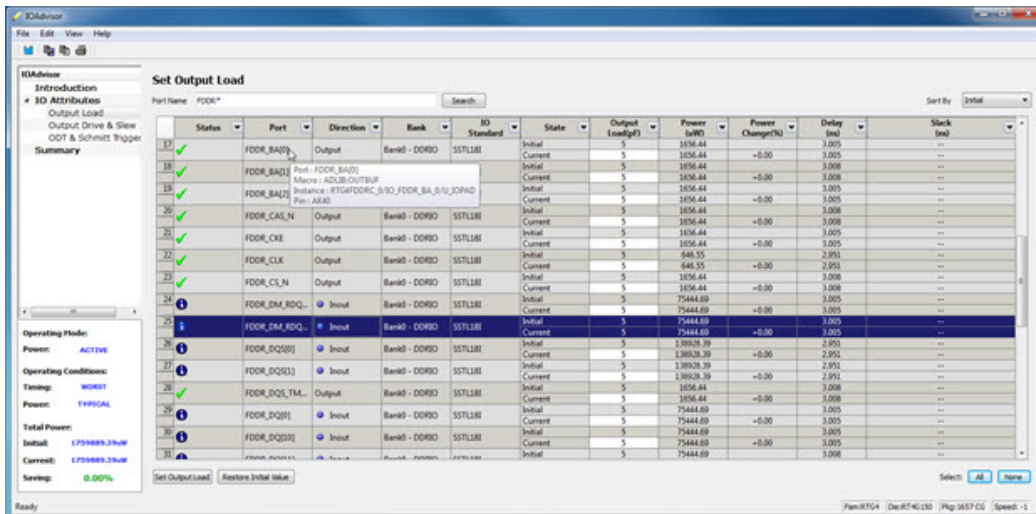


Figure 114 · IO Advisor - Output Load Panel

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.

Status	Port	Direction	Bank	IO Standard	State	Output Load(p)	Power (uW)	Power Change(%)	Delay (ns)	Slack (ns)
✓	FDDR_ADD[0]	Output	Bank0 - DDRIO	SSTL18I	Initial	5	1656.44		3.008	...
✓	FDDR_ADD[0]	Output	Bank0 - DDRIO	SSTL18I	Current	5	1656.44	+0.00	3.005	...
✓	FDDR_BA[0]	Output	Bank0 - DDRIO	SSTL18I	Initial	5	1656.44		3.005	...
✓	FDDR_BA[0]	Output	Bank0 - DDRIO	SSTL18I	Current	5	1656.44	+0.00	3.008	...
✓	FDDR_BA[1]	Output	Bank0 - DDRIO	SSTL18I	Initial	5	1656.44		3.005	...
✓	FDDR_BA[1]	Output	Bank0 - DDRIO	SSTL18I	Current	5	1656.44	+0.00	3.008	...
✓	FDDR_BA[2]	Output	Bank0 - DDRIO	SSTL18I	Initial	5	1656.44		3.005	...
✓	FDDR_BA[2]	Output	Bank0 - DDRIO	SSTL18I	Current	5	1656.44	+0.00	3.005	...
✓	FDDR_CAS_N	Output	Bank0 - DDRIO	SSTL18I	Initial	5	1656.44		3.008	...
✓	FDDR_CAS_N	Output	Bank0 - DDRIO	SSTL18I	Current	5	1656.44	+0.00	3.005	...
✓	FDDR_CKE	Output	Bank0 - DDRIO	SSTL18I	Initial	5	1656.44		3.005	...
✓	FDDR_CKE	Output	Bank0 - DDRIO	SSTL18I	Current	5	1656.44	+0.00	3.005	...
✓	FDDR_CLK	Output	Bank0 - DDRIO	SSTL18I	Initial	5	646.55		2.951	...
✓	FDDR_CLK	Output	Bank0 - DDRIO	SSTL18I	Current	5	646.55	+0.00	2.951	...
✓	FDDR_CS_N	Output	Bank0 - DDRIO	SSTL18I	Initial	5	1656.44		3.008	...
✓	FDDR_CS_N	Output	Bank0 - DDRIO	SSTL18I	Current	5	1656.44	+0.00	3.008	...
i	FDDR_DM_RDQ...	Inout	Bank0 - DDRIO	SSTL18I	Initial	5	75444.69		3.005	...
i	FDDR_DM_RDQ...	Inout	Bank0 - DDRIO	SSTL18I	Current	5	75444.69	+0.00	3.005	...
i	FDDR_DM_RDQ...	Inout	Bank0 - DDRIO	SSTL18I	Initial	5	75444.69		3.005	...
i	FDDR_DM_RDQ...	Inout	Bank0 - DDRIO	SSTL18I	Current	5	75444.69	+0.00	3.005	...
i	FDDR_DQ[0]	Inout	Bank0 - DDRIO	SSTL18I	Initial	5	138928.39		2.951	...
i	FDDR_DQ[0]	Inout	Bank0 - DDRIO	SSTL18I	Current	5	138928.39	+0.00	2.951	...
i	FDDR_DQ[1]	Inout	Bank0 - DDRIO	SSTL18I	Initial	5	138928.39		2.951	...
i	FDDR_DQ[1]	Inout	Bank0 - DDRIO	SSTL18I	Current	5	138928.39	+0.00	2.951	...
✓	FDDR_DQS_TM...	Output	Bank0 - DDRIO	SSTL18I	Initial	5	1656.44		3.008	...
✓	FDDR_DQS_TM...	Output	Bank0 - DDRIO	SSTL18I	Current	5	1656.44	+0.00	3.008	...
i	FDDR_DQ[0]	Inout	Bank0 - DDRIO	SSTL18I	Initial	5	75444.69		3.005	...
i	FDDR_DQ[0]	Inout	Bank0 - DDRIO	SSTL18I	Current	5	75444.69	+0.00	3.005	...
i	FDDR_DQ[0]	Inout	Bank0 - DDRIO	SSTL18I	Initial	5	75444.69		3.005	...
i	FDDR_DQ[0]	Inout	Bank0 - DDRIO	SSTL18I	Current	5	75444.69	+0.00	3.005	...

Figure 115 · Search Field and Regular Expressions

Status Column

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

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

Column Display and Sorting

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

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

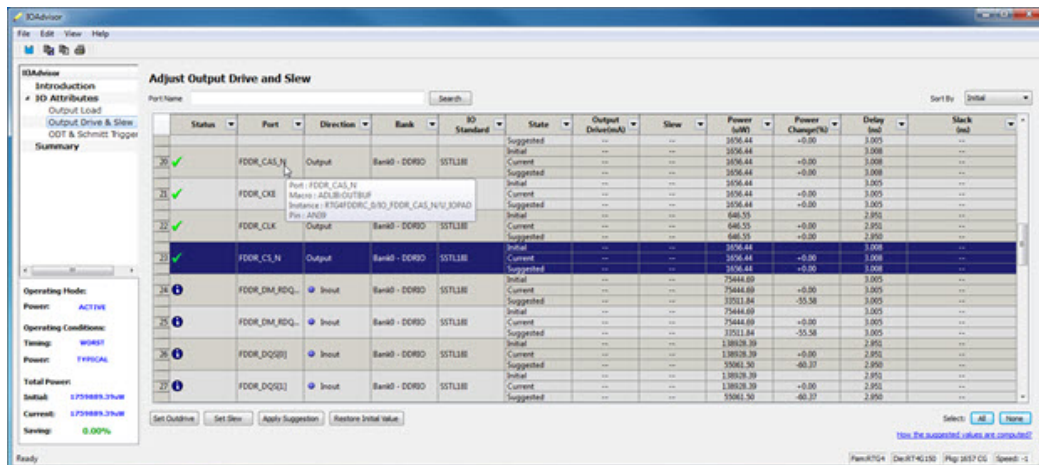


Figure 116 · IO Advisor – Output Drive and Slew

How the Suggested Values Are Computed

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

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

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

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

Apply Suggestion

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

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

Adjust ODT and Schmitt Trigger

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

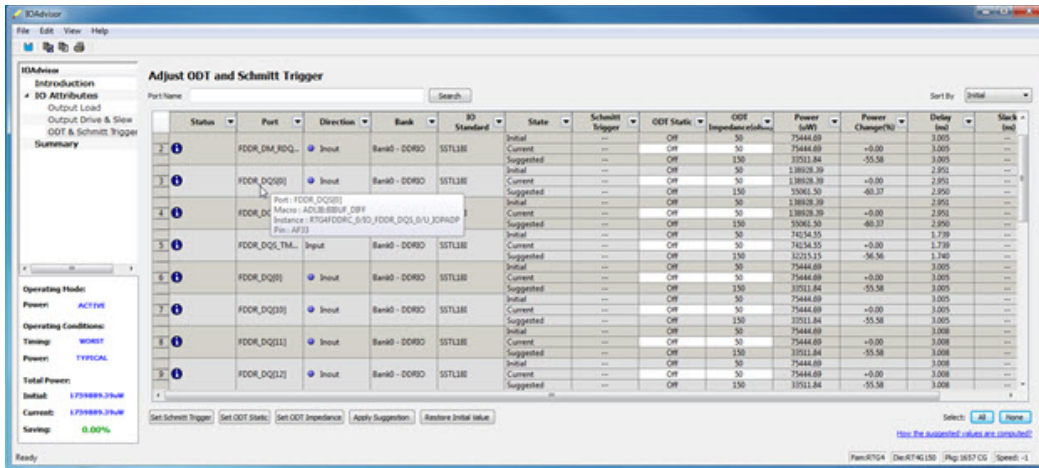


Figure 117 · IO Advisor – Adjust ODT and Schmitt Trigger

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.

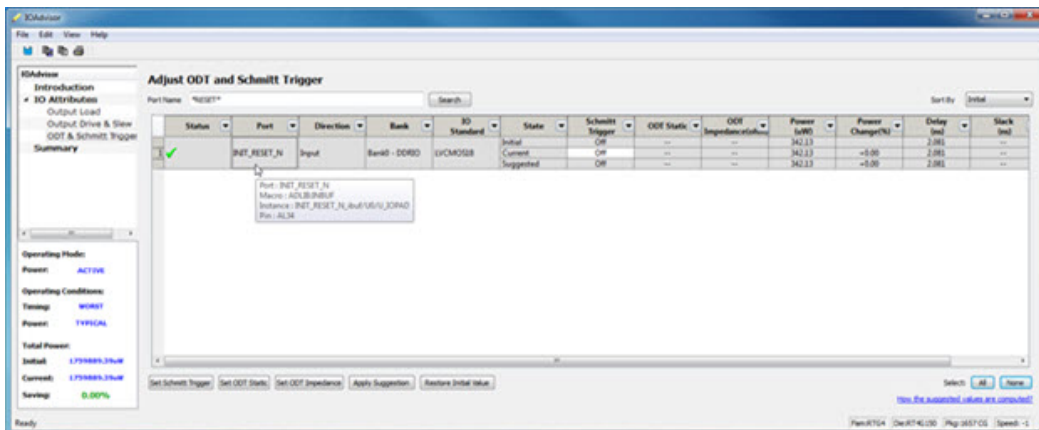


Figure 118 · Search Field and Regular Expressions

Status Column

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

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

Column Display and Sorting

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

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

Set Schmitt Trigger

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

Set ODT Static

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

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

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

Set ODT Impedance (Ohm)

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

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

Apply Suggestion

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

Restore Initial Value

To restore an input/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.

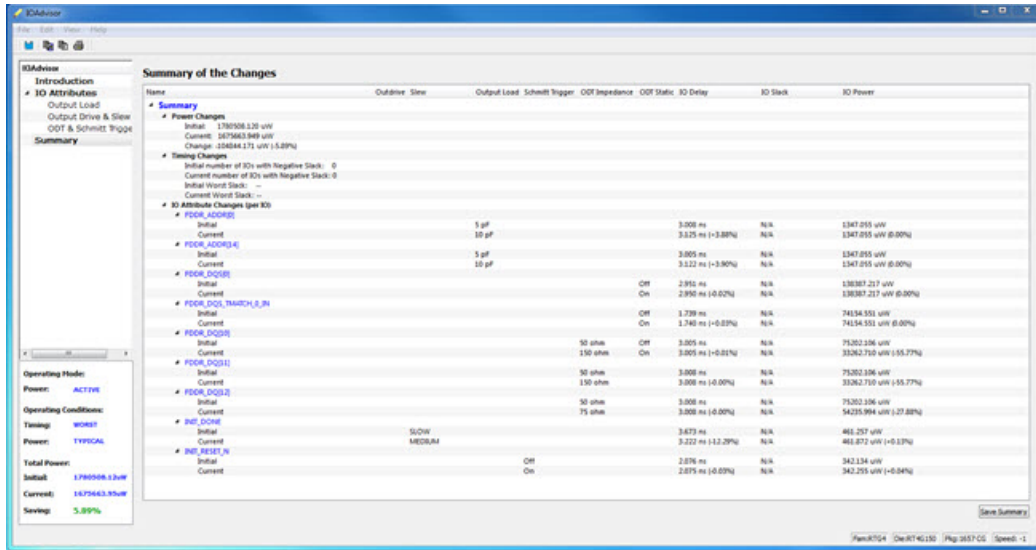


Figure 119 · IO Advisor – Summary

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.

Program and Debug

SmartFusion2 and IGLOO2 Programming Tutorials

SPI 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 Programming \(IAP\) 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).

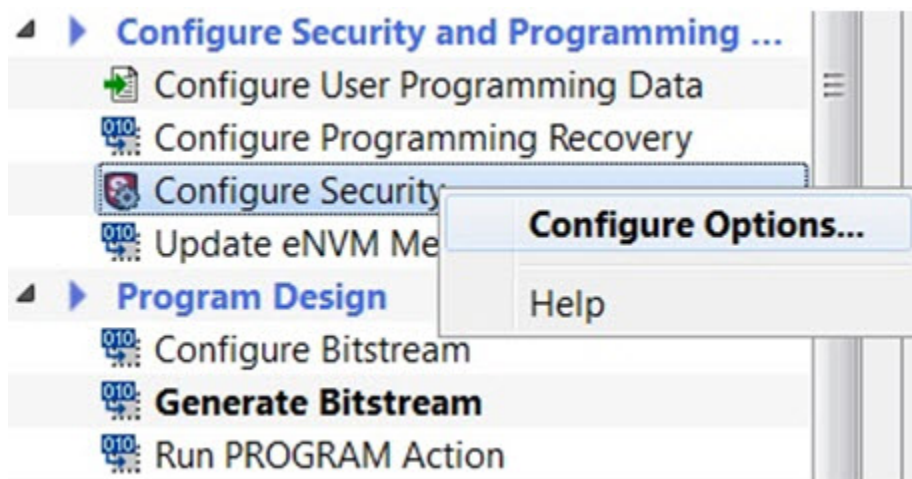


Figure 120 · Launch Security Policy Manager

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 3 · SPI Signal Polarity Modes

SPO	SPH	SPI Clock in Idle	Sample 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

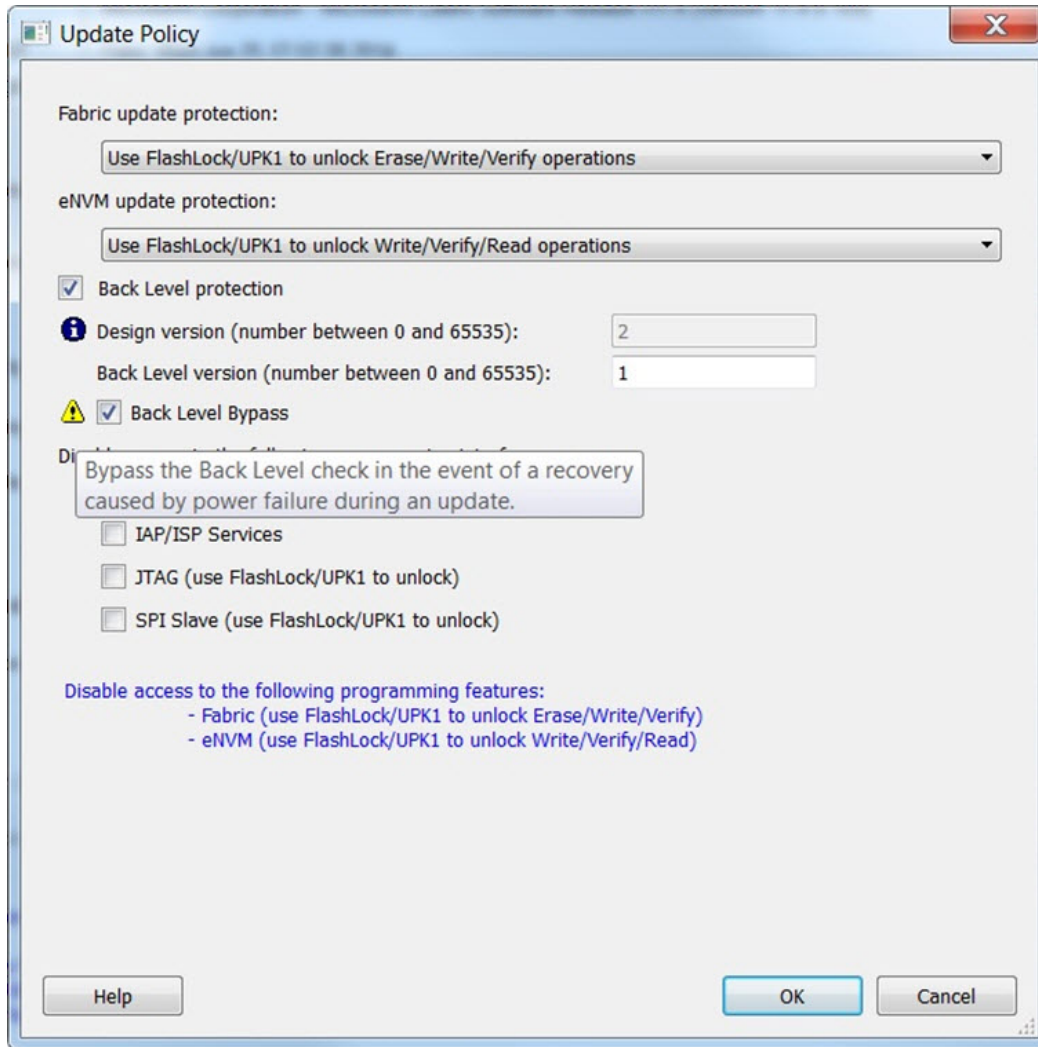


Figure 121 · Bypass Back Level Version Check

- 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.

- 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

Offset	Name	Description
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

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.

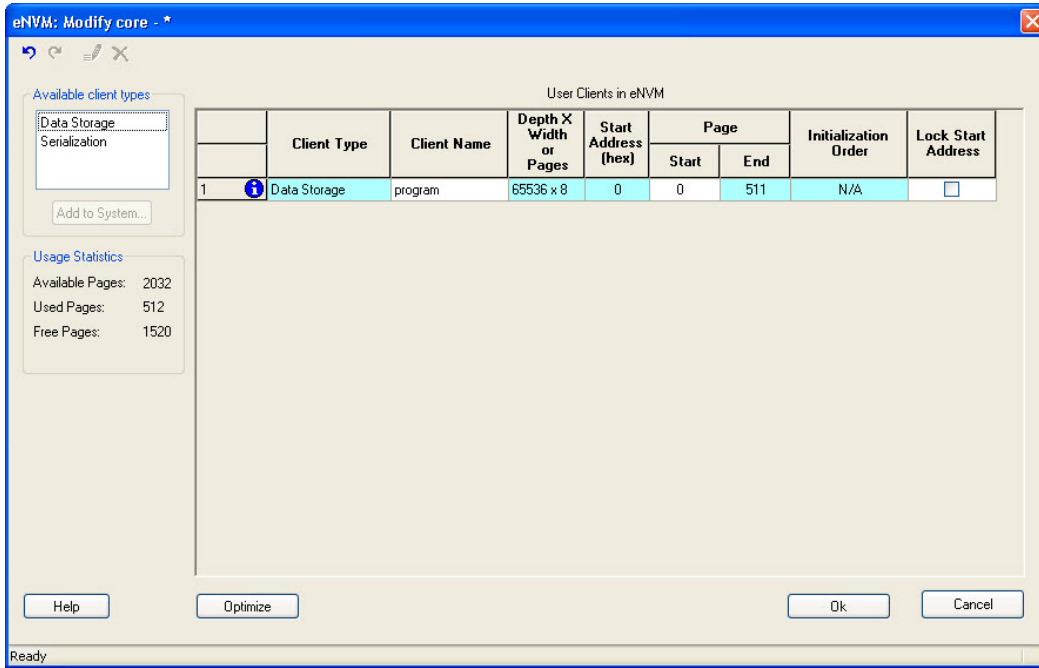


Figure 122 · eNVM Configuration Dialog Box

The example design shown in the figure above already has a Data Storage Client.

2. Double-click the **Data Storage Client** to open the **Modify Data Storage Client dialog box**, as shown in the figure below.

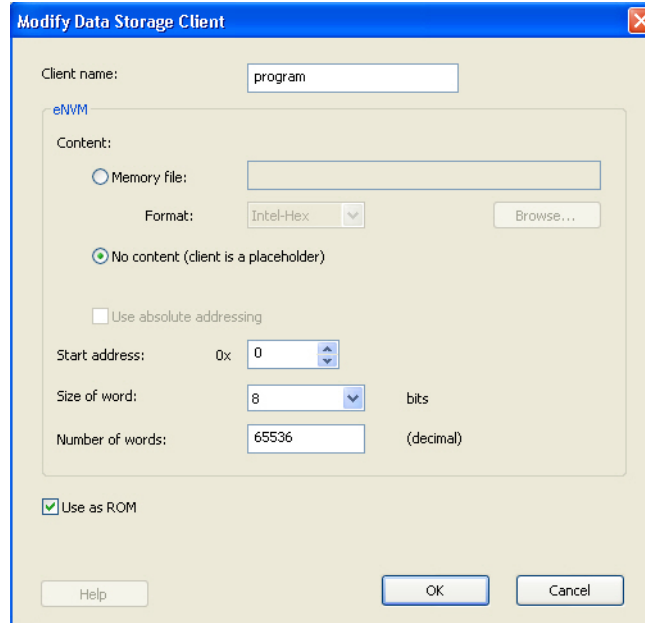


Figure 123 · Modify Data Storage Client Dialog Box - Use as ROM Selected

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. You can configure the content of your serialization region client in the [Serialization Client Editor](#), available from the [Update eNVM Memory Content dialog box](#).

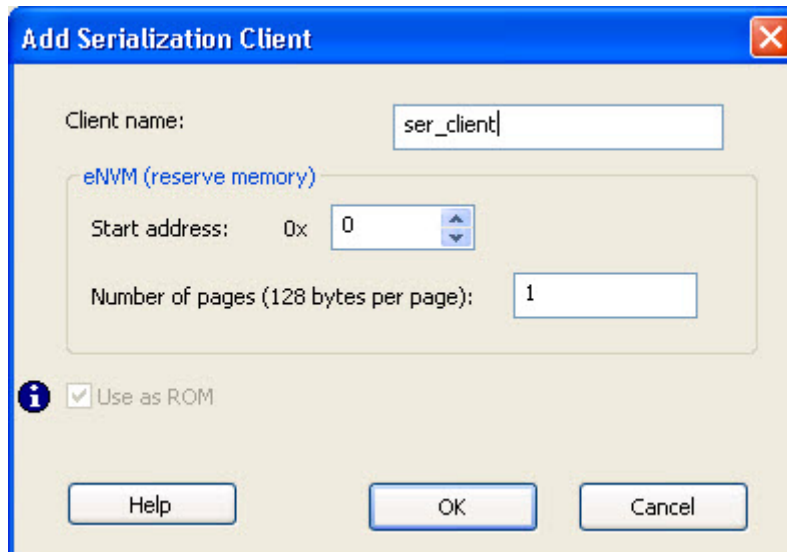


Figure 124 · Add Serialization Client Dialog Box

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.

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.

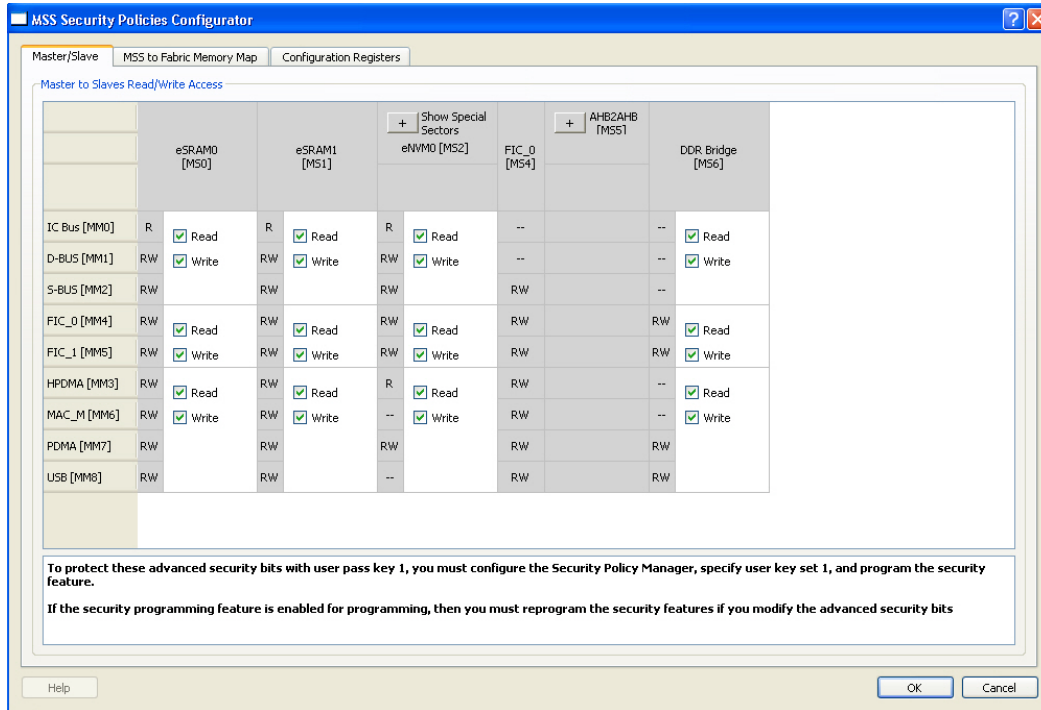


Figure 125 · MSS Security Policies Configurator

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 - SmartFusion2, IGLOO2, and RTG4

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.

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.

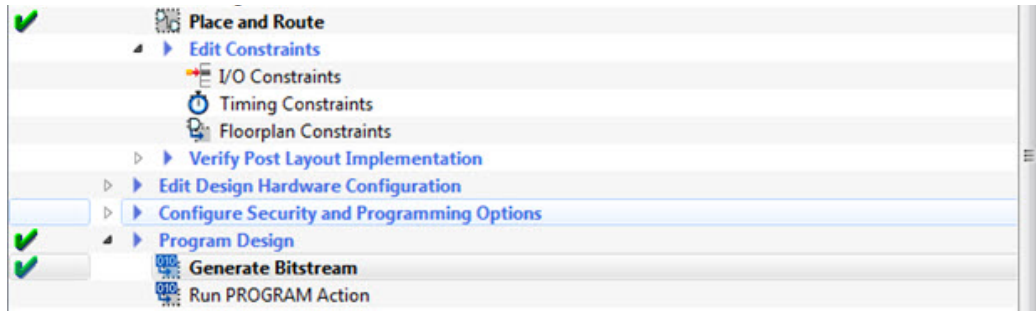


Figure 126 · Generate Bitstream (Complete)

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.

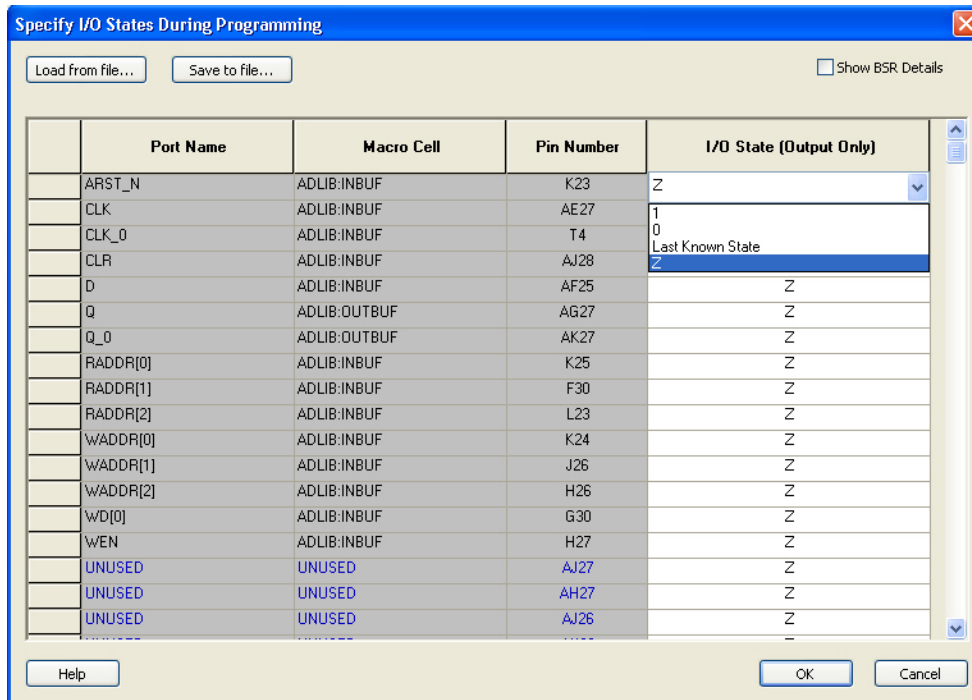


Figure 127 · Set I/O State

Bitstream Configuration

Expand Configure Security and Programming Options and double-click Bitstream Configuration to open the [Bitstream Configuration dialog box](#), as shown in the figure below.

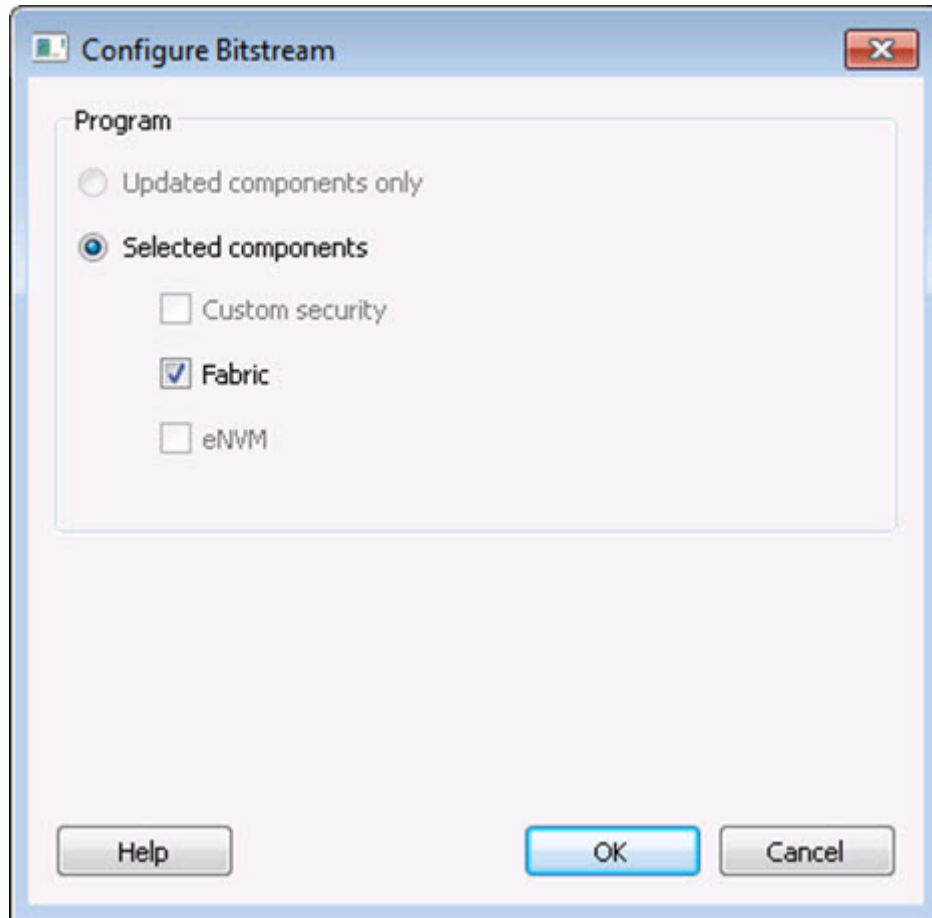


Figure 128 · Configure Bitstream Dialog Box

You can program your Security, Fabric, eNVM or any combination.

Features (Security, Fabric, eNVM) are enabled for programming by default when you add them to your design. If you manually disable a feature in this dialog box then you must re-enable it here if you want it included during programming.

Update eNVM Memory Content (SmartFusion2 and IGLOO2)

Right-click **Update eNVM Memory Content** and choose **Configure Options (eNVM Memory Content > Configure Options)** or double-click **Update eNVM Memory Content** to open the dialog box and modify your eNVM content.

The Update eNVM Memory Content dialog box enables you to update your eNVM content for programming 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 Update eNVM Memory Content 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.

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 in the Modify Data Storage Client dialog box. To make these changes, go to the eNVM configurator inside the MSS/HPMS Configurator or navigate to the System Builder's Memory page (eNVM tab).

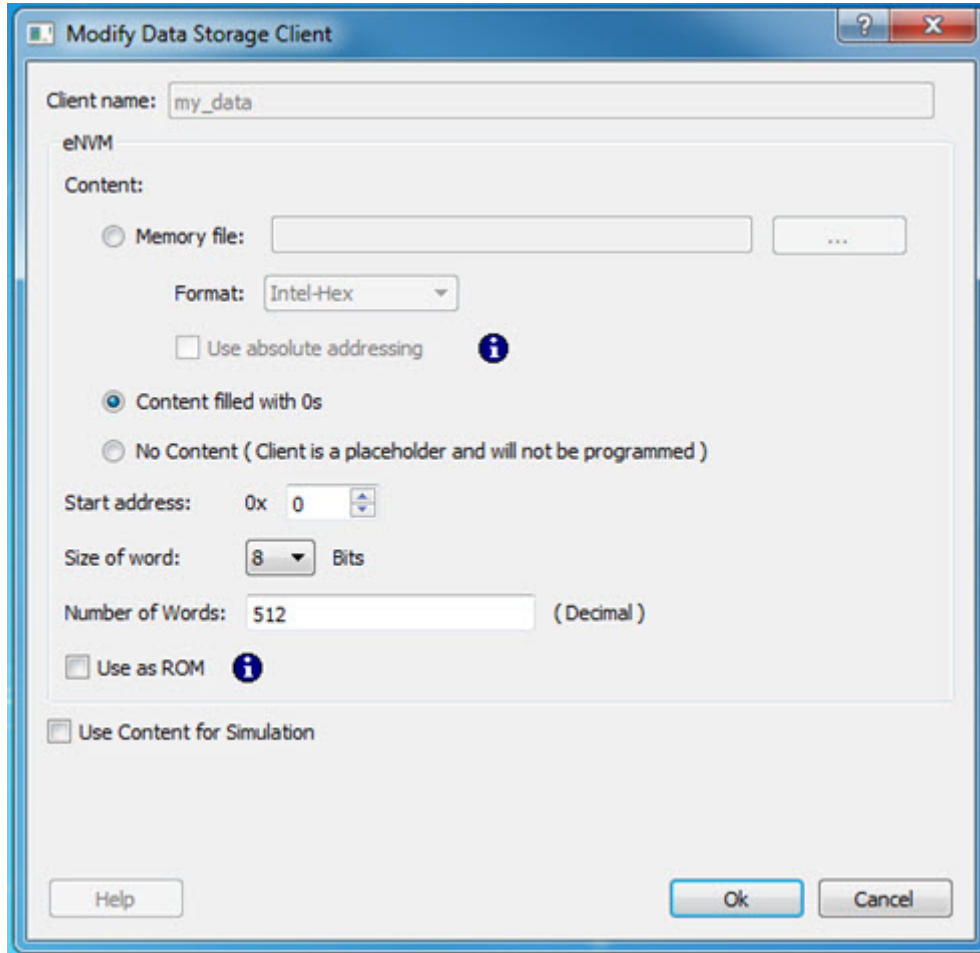


Figure 129 · Modify Data Storage Client Dialog Box

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.

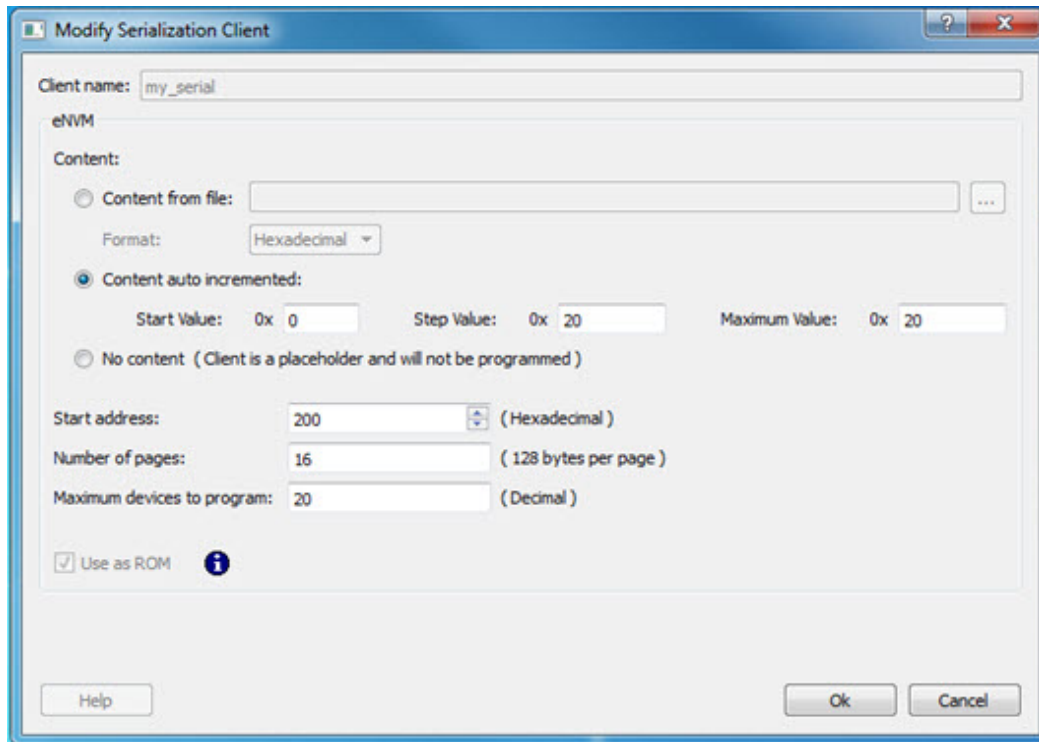


Figure 130 · Modify Serialization Client Dialog Box

You have three options to specify the eNVM content:

- Import a Memory File
- Increment values automatically
- 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.

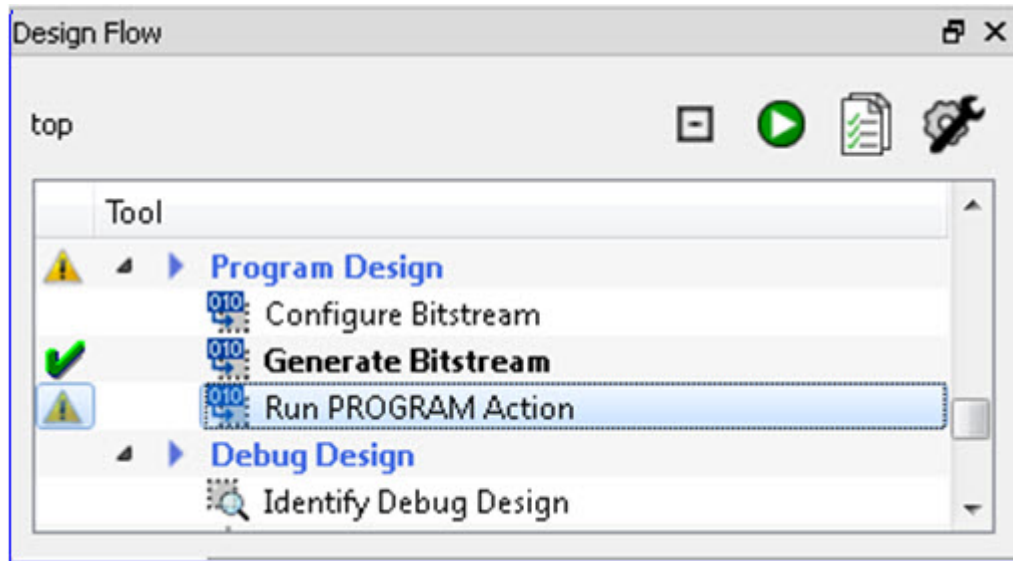


Figure 131 · Program Device in the Design Flow Window

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](#).

Export BSDL File

Double-click **Export BSDL File** to generate a BSDL file for your project.

Right-click **Export BSDL File** and choose **Clean and Run All** to remove all data and output from tools run previously and rerun the Design Flow up through this point.

Generate FPGA Array Data

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

- *.db used for debugging FPGA Fabric in SmartDebug
- *.map and *.dca files used for Programming

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

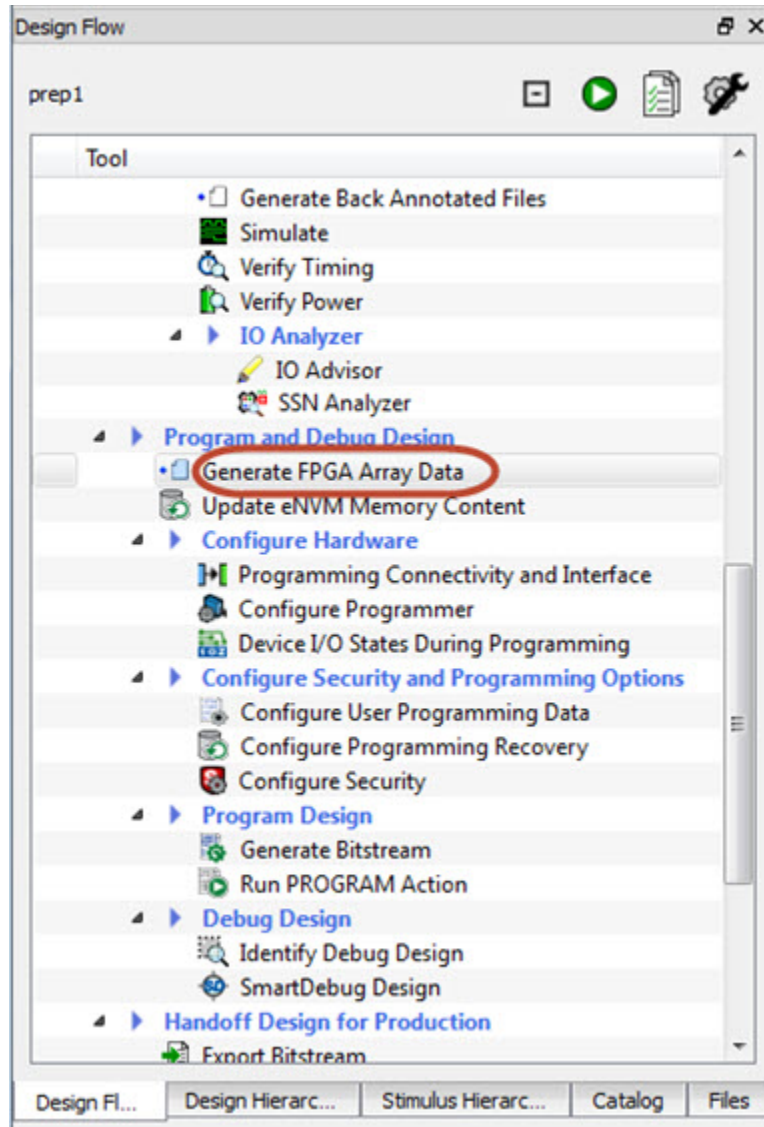


Figure 132 · Design Flow Window – Generate FPGA Array Data

Note: Generate FPGA Array Data is the upstream tool for Export SmartDebug Data. Libero SoC implicitly runs the Generate FPGA Array Data step (if not already completed) before it exports the SmartDebug data files.

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**.

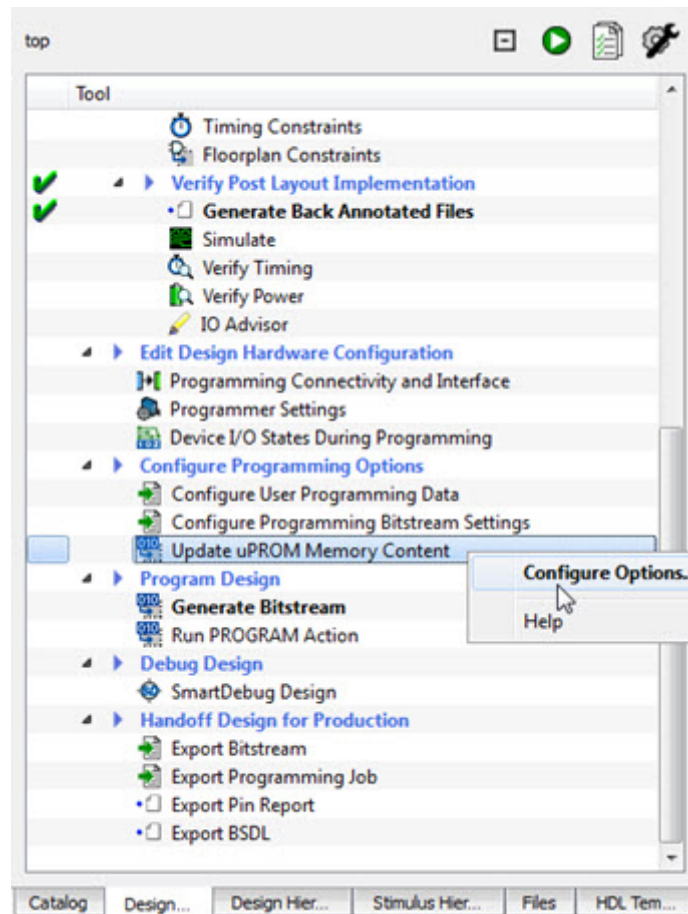


Figure 133 · Update uPROM Memory Content

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

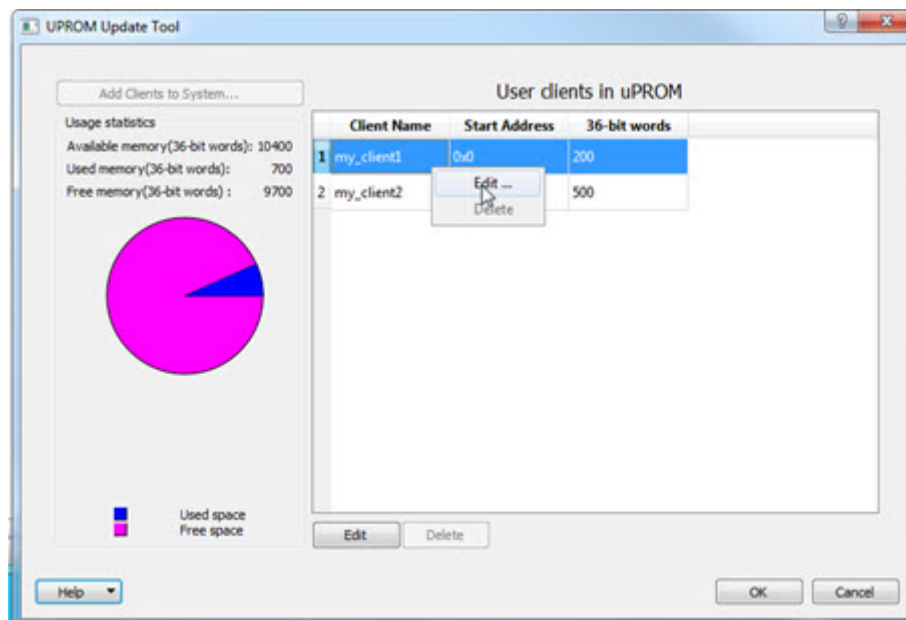


Figure 134 · uPROM Update Tool

The Edit Data Storage Client dialog box appears.

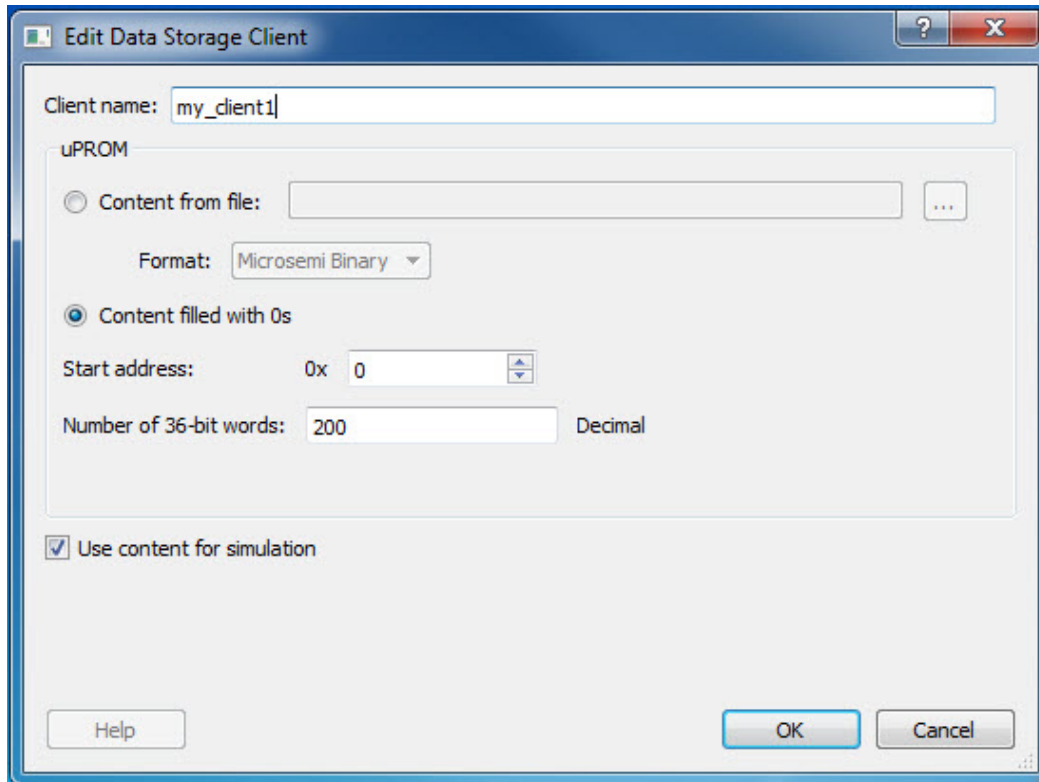


Figure 135 · Edit Data Storage Client Dialog Box

You can make the following changes to the uPROM client:

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

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

Update eNVM Memory Content (SmartFusion2 and IGLOO2)

Right-click **Update eNVM Memory Content** and choose **Configure Options (eNVM Memory Content > Configure Options)** or double-click **Update eNVM Memory Content** to open the dialog box and modify your eNVM content.

The Update eNVM Memory Content dialog box enables you to update your eNVM content for programming 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 Update eNVM Memory Content 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.

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 in the Modify Data Storage Client dialog box. To make these changes, go to the eNVM configurator inside the MSS/HPMS Configurator or navigate to the System Builder's Memory page (eNVM tab).

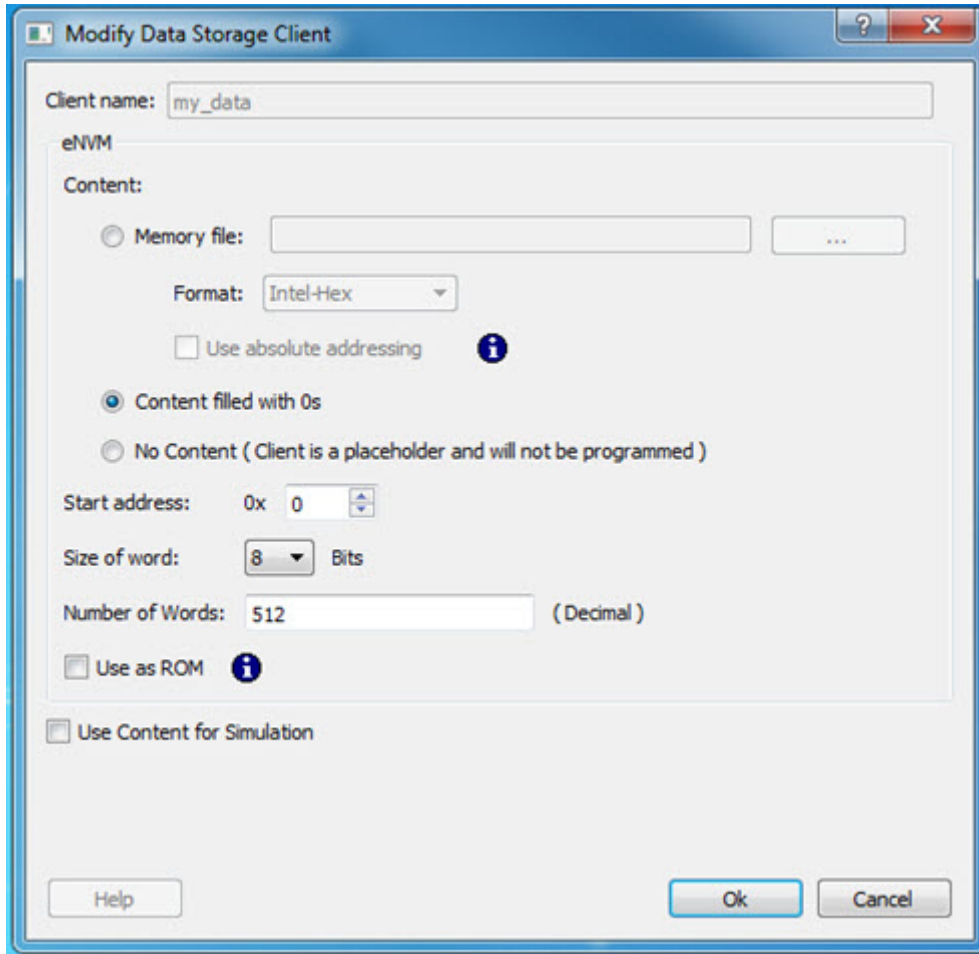


Figure 136 · Modify Data Storage Client Dialog Box

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.

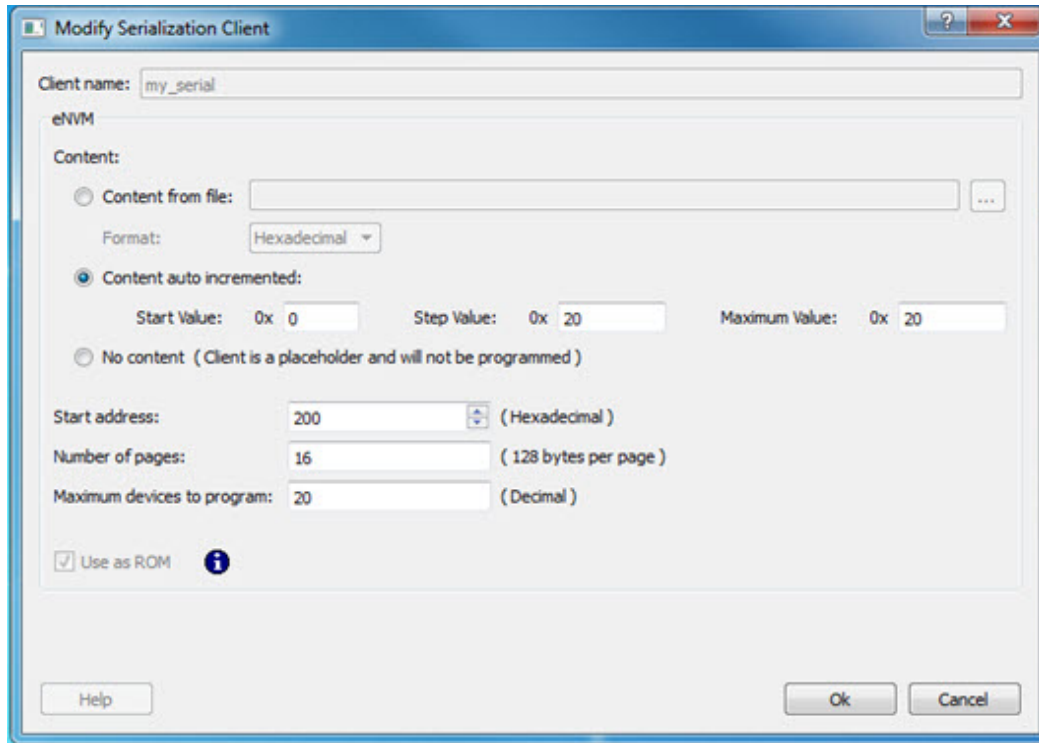


Figure 137 · Modify Serialization Client Dialog Box

You have three options to specify the eNVM content:

- Import a Memory File
- Increment values automatically
- 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.

Configure Hardware

Programming Connectivity and Interface - SmartFusion2, IGLOO2, and RTG4 Only

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

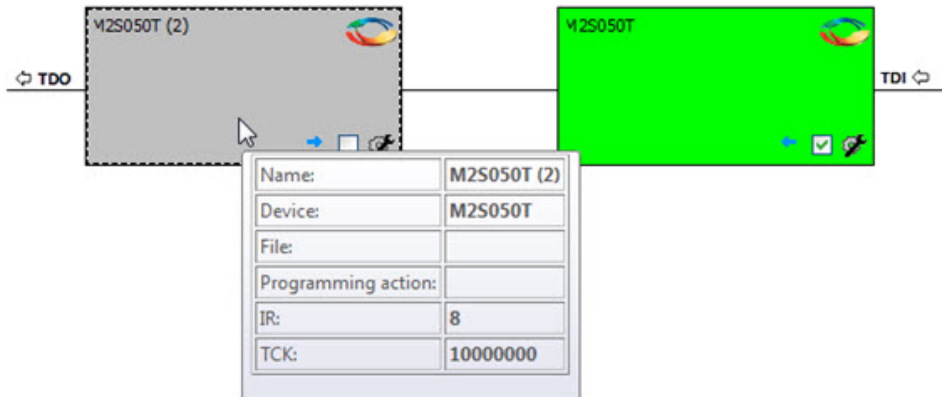


Figure 138 · Device Information

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** - 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)
- **Enable Serial** - Select to enable serialization when you have loaded a serialization programming file (not supported in software version 11.0)
- **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.

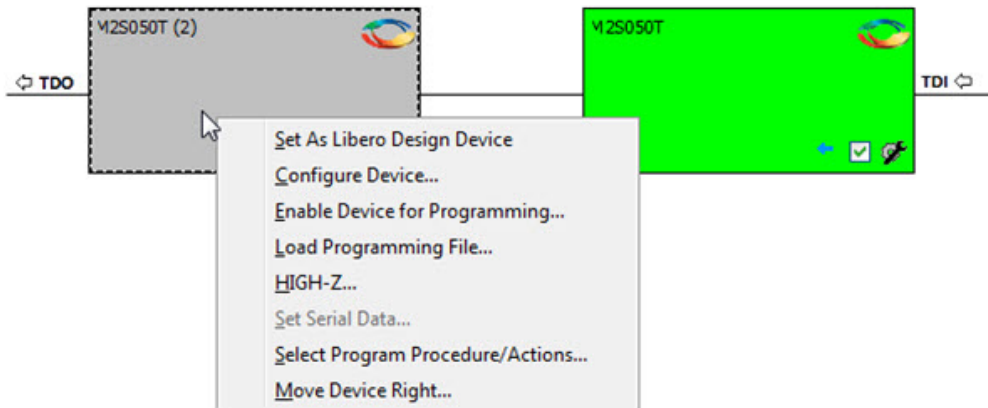


Figure 139 · Right-click Properties

Programmer Settings - SmartFusion2, IGLOO2, and RTG4 Only

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.

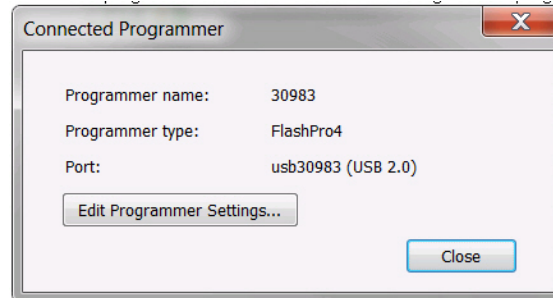


Figure 140 · Programmer Settings for Connected Programmer

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.

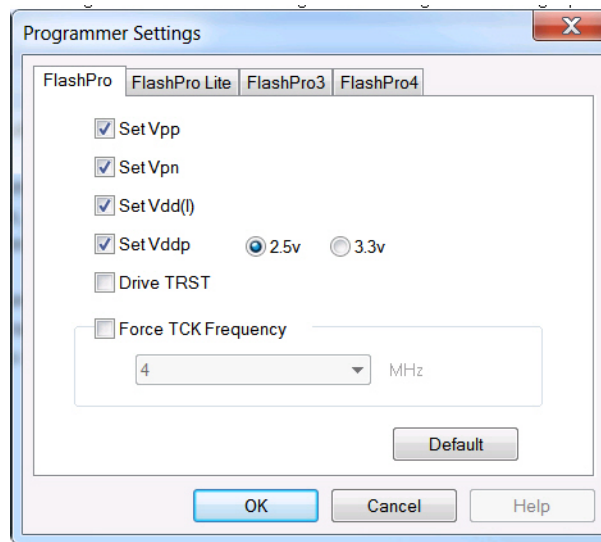


Figure 141 · Programmer Settings Dialog Box

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

Set the TCK setting in your PDB/STAPL file by selecting the TCK frequency in the Programmer Settings dialog box. TCK frequency limits by programmer:

- FlashPro supports 1-4 MHz
- FlashPro Lite is limited to 1, 2, or 4 MHz only.
- FlashPro5/4/3/3X supports 1-4 MHz.

TCK frequency limits by target device:

- SmartFusion2, IGLOO2, and RTG4 - 20MHz

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

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

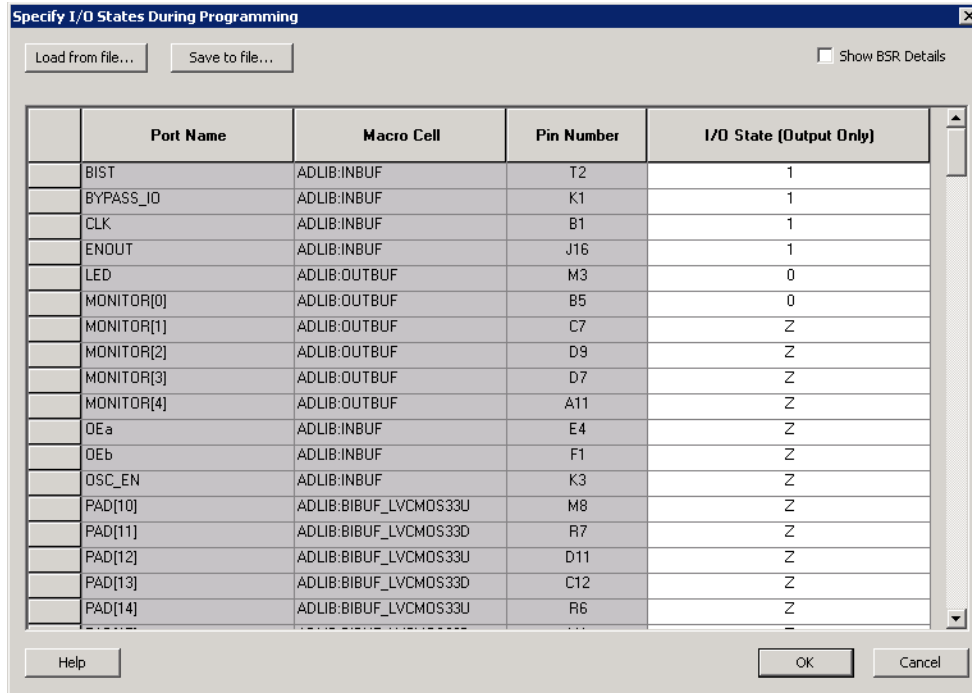


Figure 142 · I/O States During Programming Window

6. Click **OK** to save your settings.

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

Configure Security and Programming Options

Configure User Programming Data

Sets your Design Version and Silicon Signature.

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

Design Version (number between 0 and 65535) - Specifies the design version to be programmed to the device. This field is also used for Back-level protection in [Update Policy](#) of the Security Policy Manager and in Enable Auto Update in the [Programming Recovery tool](#).

Silicon signature (max length is 8 HEX chars) - 32-bit user configurable silicon signature to be programmed into the device. This field can be read from the device using the JTAG (IEEE 1149-1) USERCODE instruction or by running the [DEVICE_INFO](#) programming action.

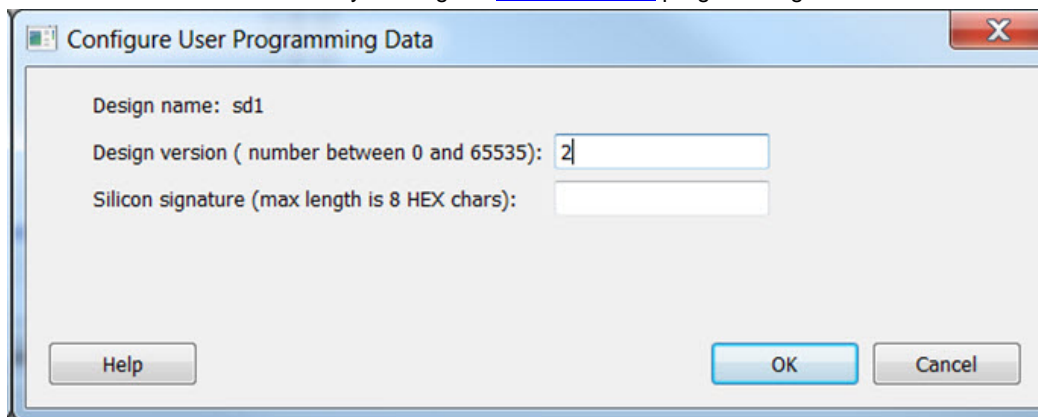


Figure 143 · Configure User Programming Data Dialog Box

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 Bitstream Settings (RTG4 Only)

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

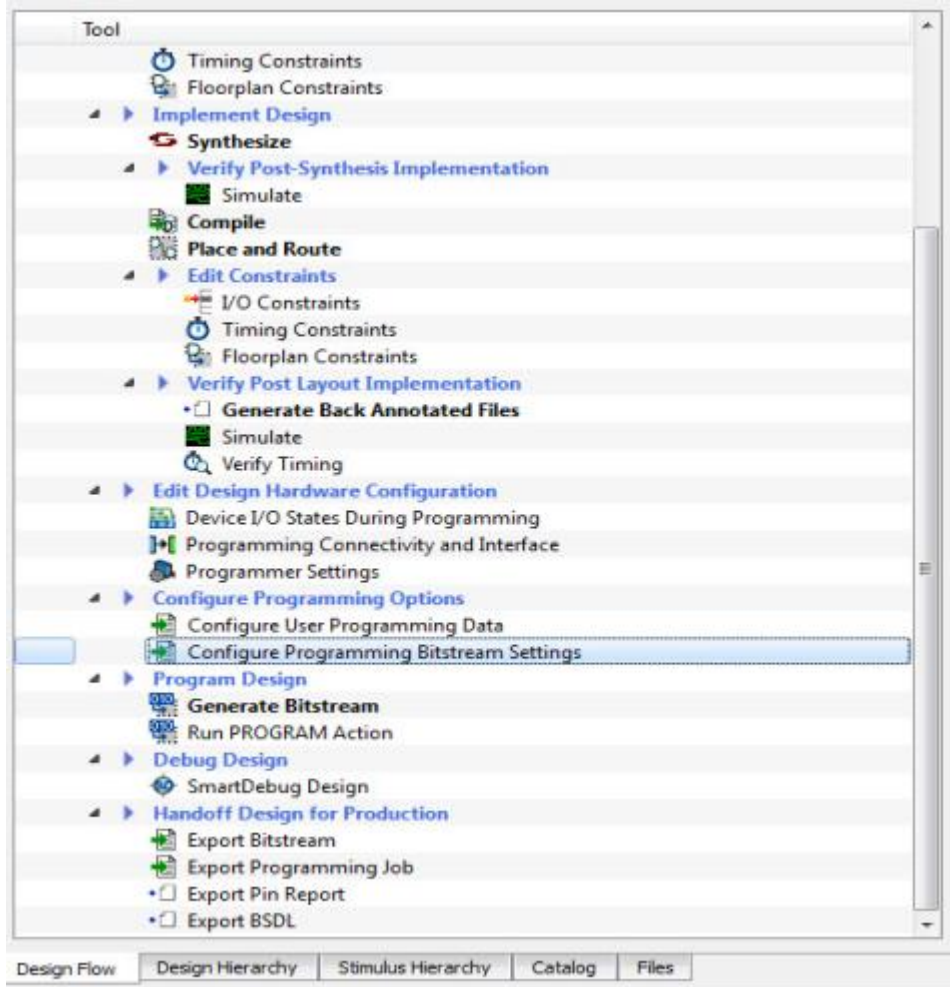


Figure 144 · Configure Programming Bitstream Settings - Configure Options

The Programming Bitstream Settings dialog box appears for you to configure the Bitstream Settings.

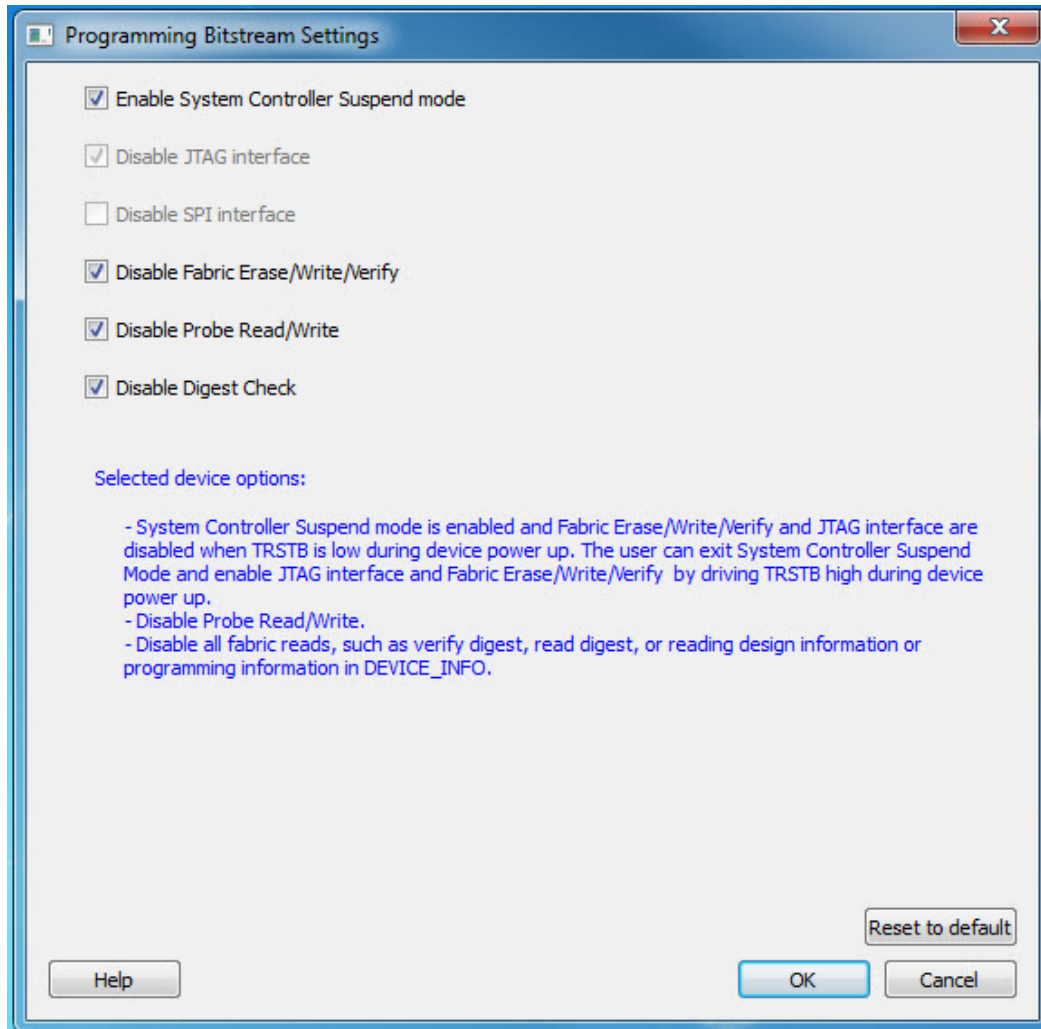


Figure 145 · Programming Bitstream Settings Dialog Box

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.

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.

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 Bitstream 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.

Configure Programming Recovery

The Programming Recovery dialog box 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.

Design version - The Design Version used for Auto Update Programming or for Backlevel protection within the SPM update policy. This is a read-only field that must be configured within the tool [Configure User Programming Data](#).

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.

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.

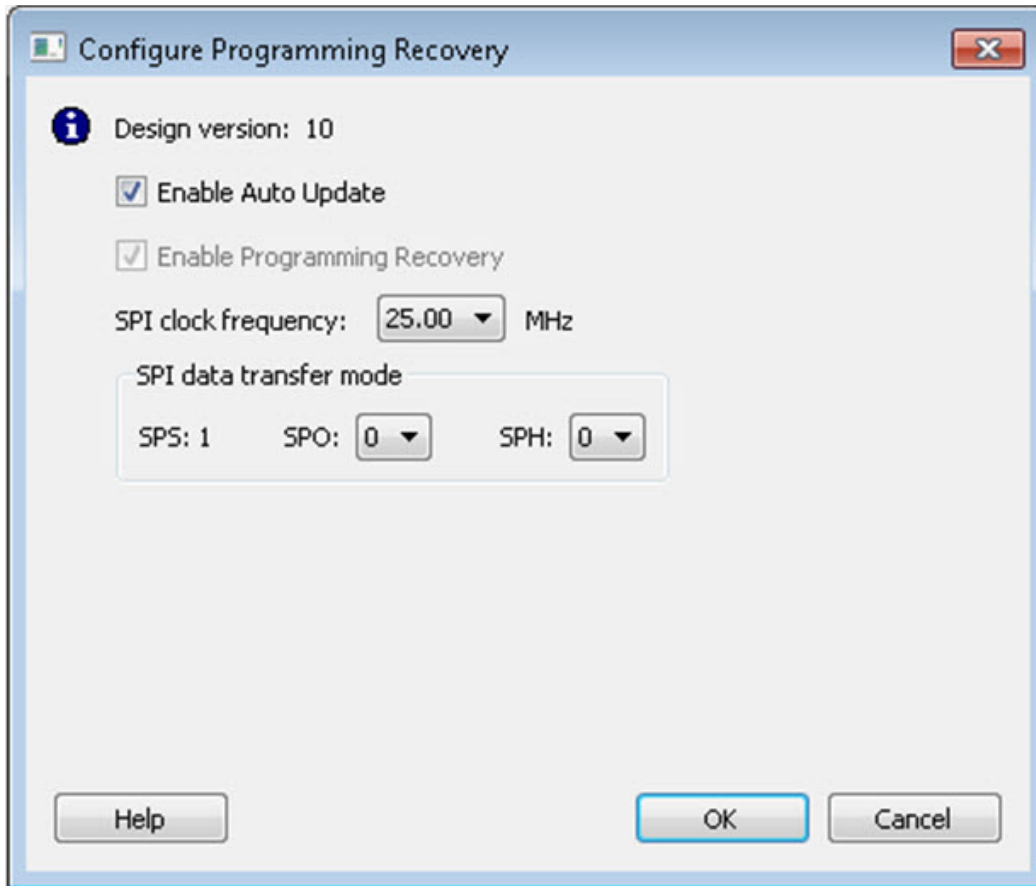
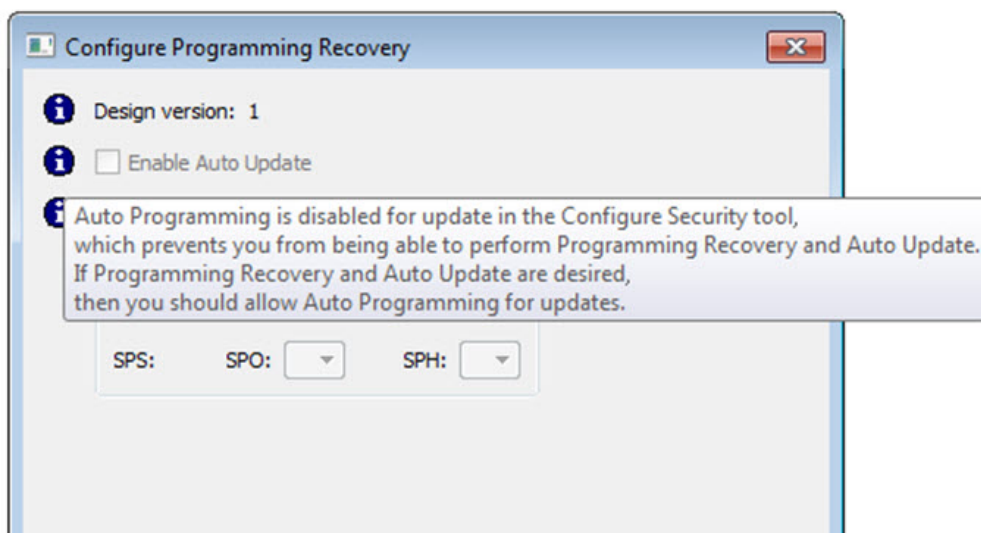


Figure 146 · Configure Programming Recovery Dialog Box

SPM - 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 unlocking 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.



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 [Security Policy Manager](#) 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 un-check 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?

- Configure the [Security Policy Manager](#) and specify a User Key Set 1 and User Key Set 2 (User Key Set 2 is available if you select Field Update Broadcast mode). 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) and <filename>_uek2.(stp/svf/spi/dat) files are encrypted with UEK1 and UEK2 respectively. See Security Programming File Descriptions below for more information on programming files.

What are Security Programming Files?

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

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 [Security Policy Manager](#). 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 [Security Policy Manager](#). 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.

Configure Security

Security Policy Manager (SPM)

Expand **Configure Security and Programming Options**, double-click **Configure Security** to 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.

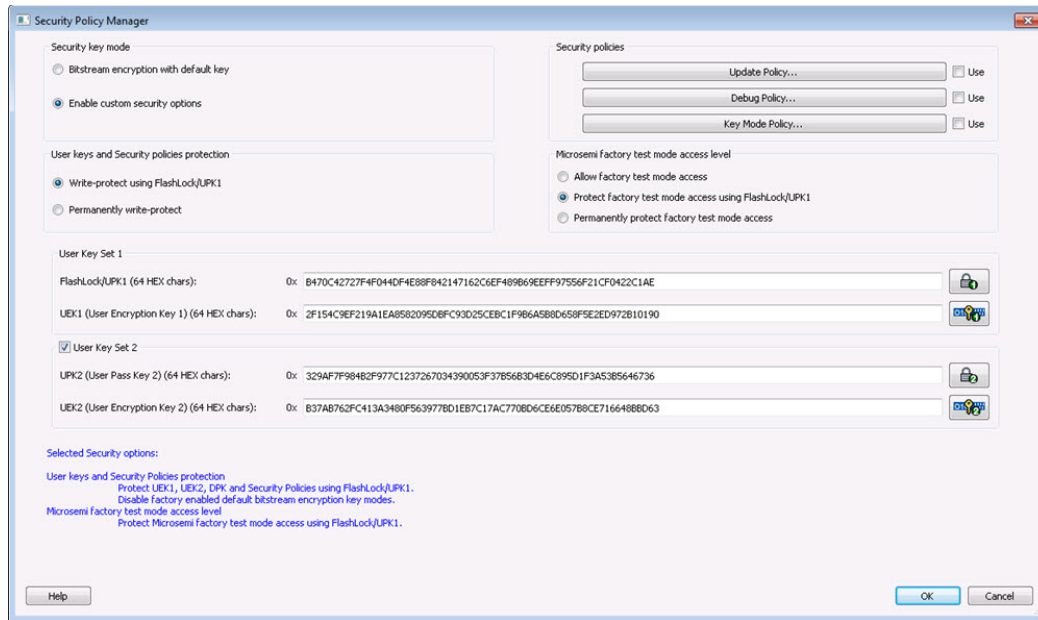


Figure 147 · Security Policy Manager Dialog Box

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), DPK (Debug Pass Key) and Security Policies using FlashLock/ UPK1.

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

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. 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.

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

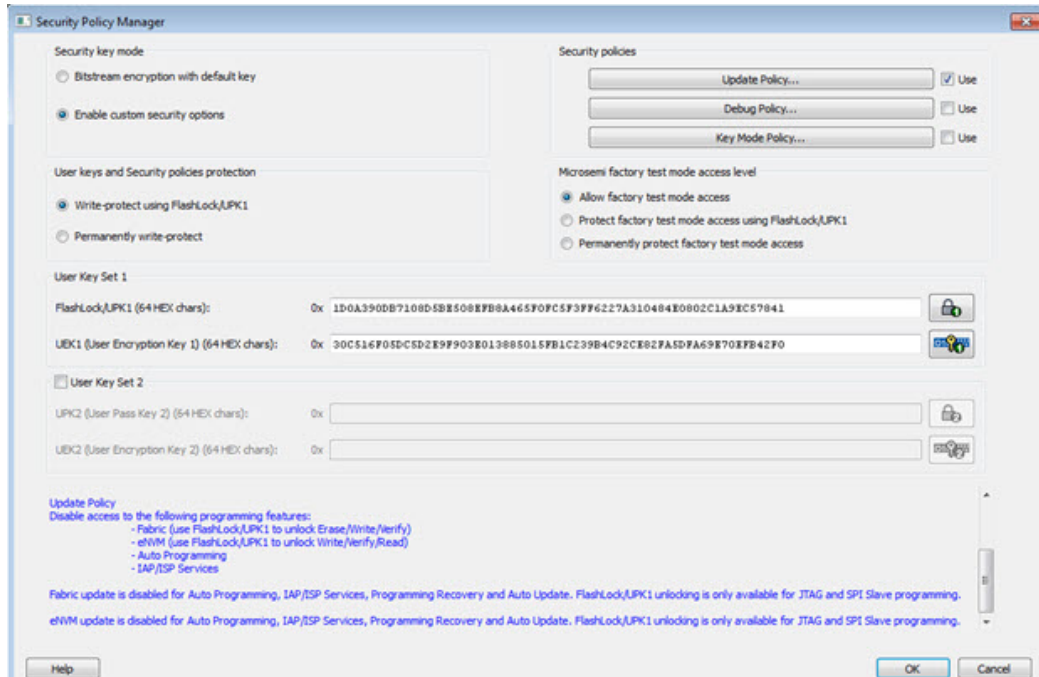
Update Policy - Sets your Fabric, eNVM and Back Level protections. See the [Update Policy topic](#) for more information.

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).

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

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

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.

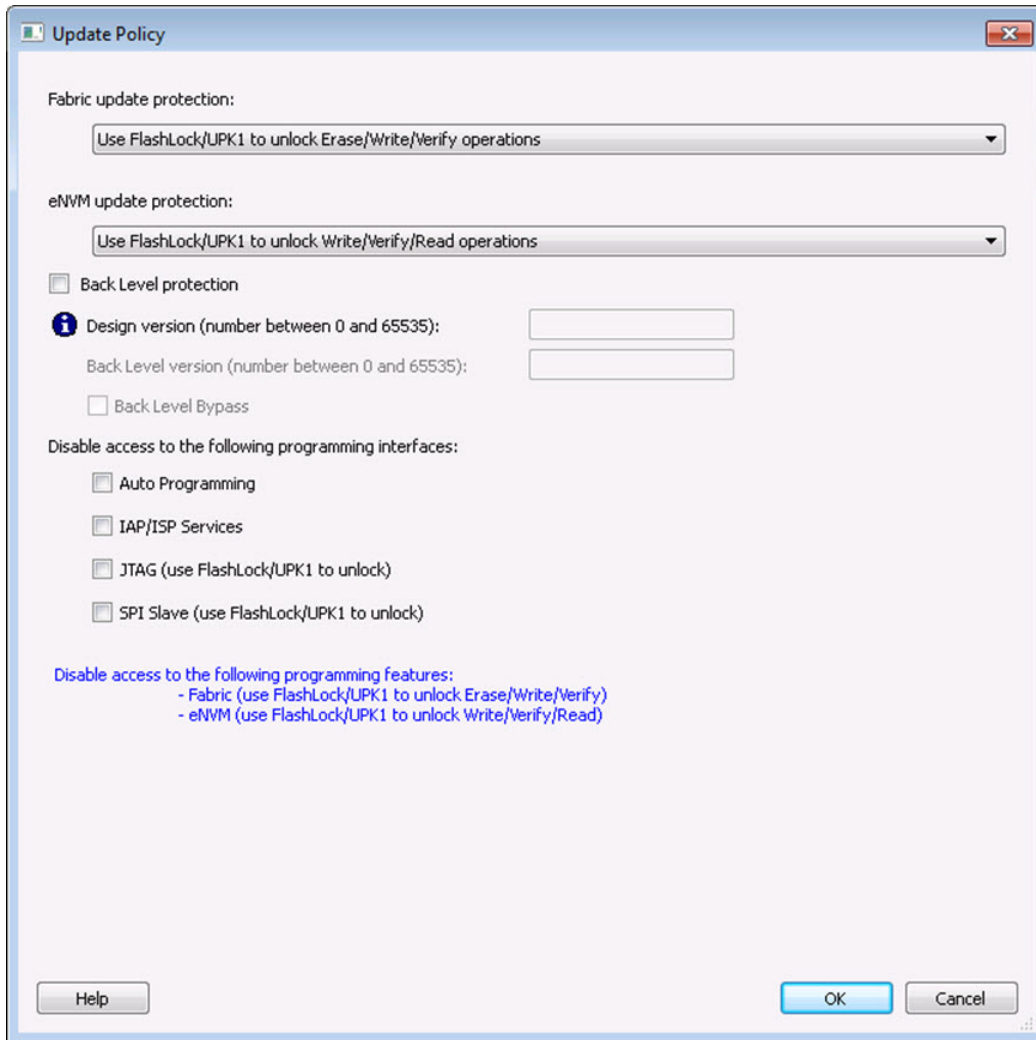


Figure 148 · Update Policy Dialog Box

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.

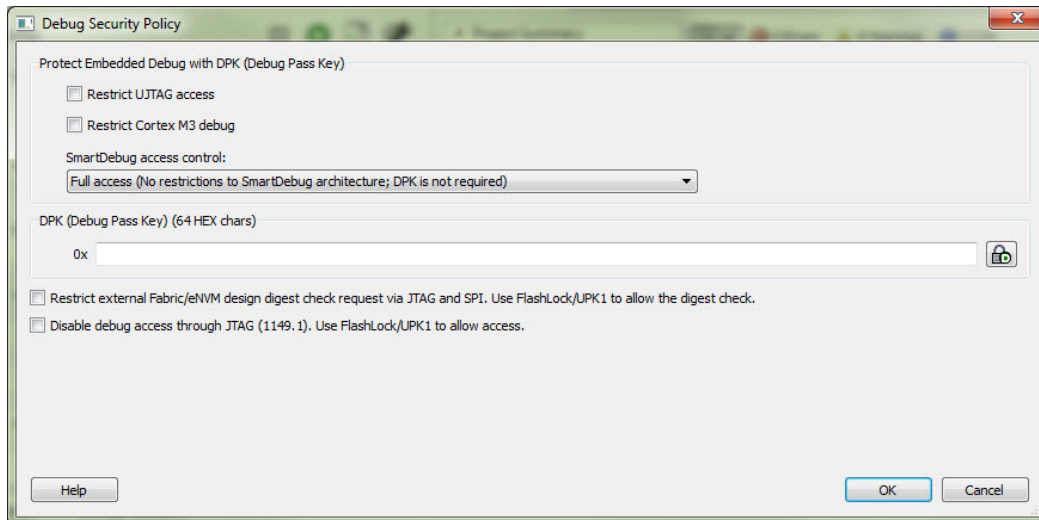


Figure 149 · Debug Security Policy Dialog Box

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.

Two key modes can be disabled:

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

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

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



Figure 150 · Programming Key Mode Policy Dialog Box

Protocol Policy

Protect programming protocols with UPK1. If a programming protocol is disabled, then UPK1 is required to program with that programming protocol.

Two protocols can be disabled:

- User Encryption Key 1
- User Encryption Key 2

If both protocols are disabled then device update is impossible.

Operational Integrity Policy

Add digest calculation and verification on power-up. Only features in the design can be selected. The available features are:

- Fabric
- eNVM

Note: Only clients with Use as ROM selected are included in this check. Verify/Read eNVM must not be protected by UPK1 within the Update Policy to enable eNVM digest check.

Serialization Client Editor

Serialization enables you to have a client in the eNVM whose value is different for each device that is programmed. You can program n number of devices with values that are configured as either Auto Incremented (AUTO_INC) or Read from File (READ_FROM_FILE).

The Serialization Client Editor is available from within the Update eNVM Memory Core dialog box. Double-click **eNVM** in your MSS (SmartFusion) or System Builder (IGLOO2) to view the eNVM Memory Core dialog box.

The Serialization Client Editor enables you to specify your serialization type. You must use eNVM Configuration to reserve a client for serialization - see the [eNVM Configuration help](#) and the [tutorial topic](#) for more information.

Content Type

Content from file

The number of lines in the file is the number of devices desired for serialization programming. The first line is the first serial index to be used for programming, the second line is the second serial index to be used for programming and the last line is the last serial index to be used for programming. Blank lines and comments (lines that begin with a # character) will be ignored.

Two file formats:

- **DEC:** An unsigned 64-bit decimal value.
- **HEX:** Hexadecimal value to be programmed into the device. If one page is specified for the serialization client, then a maximum of 256HEX characters can be placed on each line. The data orientation is MSB -> LSB, where the least significant byte is all the way to the right. If the data does not complete a page, then the page will be padded with 0's. If serialization client is larger than one page then the data format is as follows:

<Page N><PageN-1>.....<Page1><Page0>

Where each Page X is a maximum of 256HEX characters

Content auto incremented

- **Start Value (Hex)** - The first 64-bit unsigned value to program to the device.
- **Step Value (Hex)** - The step value to use for each subsequent device to be programmed.
- **Maximum Value (Hex)** - The maximum value to be programmed on the last device.

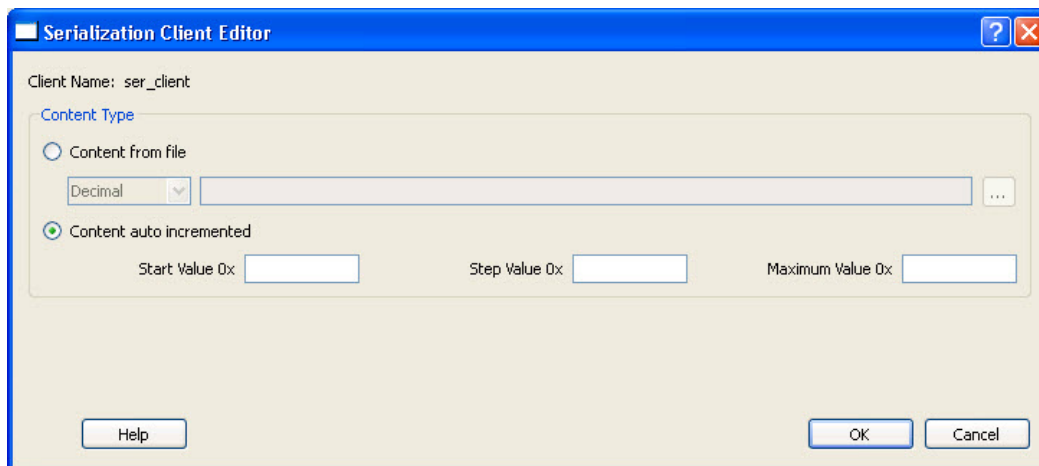


Figure 151 · Serialization Client Editor Dialog Box

Program Design

Generate Bitstream - SmartFusion2, IGLOO2, and RTG4

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.

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.

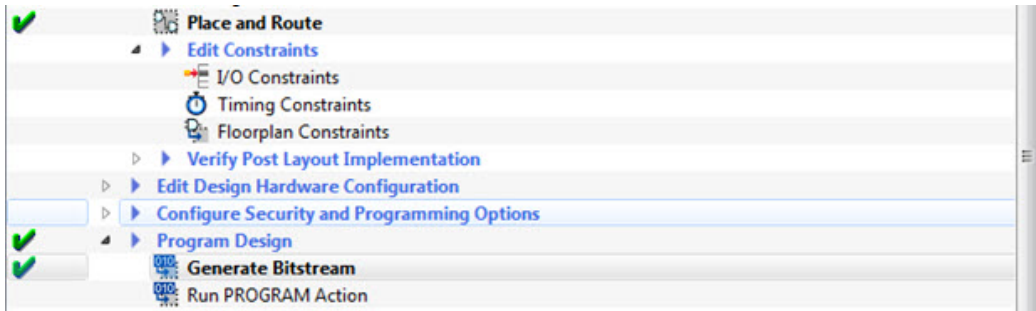


Figure 152 · Generate Bitstream (Complete)

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](#).

SmartFusion2 and IGLOO2 Programming in Libero SoC

SmartFusion2 and IGLOO2 Programming - Default Settings

To view your default settings, from the **Project** menu choose **Project Settings**.

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. (SmartFusion2 and IGLOO2)

Error Code	Exit Code	Exit Message	Possible Cause	Possible Solution
	0	Passed (no error)	-	-
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".
0x8002	5	Failure to configure device programming at 1.2/1.0 VCC voltage	Unstable voltage level Signal integrity issues on JTAG pins	Monitor related power supplies that cause the issue during programming; check for transients outside of Microsemi specifications. See your device datasheet for more information on transient specifications. Monitor JTAG supply pins during programming; measure JTAG signals for noise or reflection.
0x8032	5	Device is busy	Unstable VDDIx voltage level	Monitor related power supplies that cause the issue during programming; check for transients outside of Microsemi specifications. See your device datasheet for more information on transient specifications.
0x8003	5	Failed to enter programming mode	Unstable voltage level Signal integrity issues on JTAG pins DEV_RST_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 DEV_RST_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-

Error Code	Exit Code	Exit Message	Possible Cause	Possible Solution
				up to pin. Reduce the length of Ground connection.
0x8005 0x8006	10	Failed to program eNVM	Unstable voltage level. Signal integrity issues on JTAG pins.	Monitor related power supplies that cause the issue during programming; check for transients outside of Microsemi specifications. See your device datasheet for more information on transient specifications. Monitor JTAG supply pins during programming; measure JTAG signals for noise or reflection.
0x8007	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	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.
0x8010	-35	Failed to unlock User Pass Key 1	Pass key in file does not match device	Provide a programming file with a pass key that matches pass key

Error Code	Exit Code	Exit Message	Possible Cause	Possible Solution
				programmed into the device.
0x8011	-35	Failed to unlock User Pass Key 2	Pass key in file does not match device	Provide a programming file with a pass key that matches pass key programmed into the device.
0x8012	-35	Failed to unlock debug pass key	Pass key in file does not match device	Provide a programming file with a pass key that matches pass key programmed into the device.
0x8013	-18	Digest request from SPI/JTAG is protected by User Pass Key 1	Digest request from SPI/JTAG is protected by user pass key 1. Lock bit has been configured in the Debug Policy within SPM (Security Policy Manager)	Provide a programming file with a pass key that matches pass key programmed into the device.
0x8014	-19	Failed to verify digest	Unstable voltage level Signal integrity issues on JTAG pins	Monitor related power supplies that cause the issue during programming; check for transients outside of Microsemi specifications. See your device datasheet for more information on transient specifications. Monitor JTAG supply pins during programming; measure JTAG signals for noise or reflection.
0x8015	-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.

Error Code	Exit Code	Exit Message	Possible Cause	Possible Solution
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 digest verification: FAIL	Programming bitstream components do not match components programmed User security policy segment data has been corrupted or tampered with	Use the same programming file that was used to program the device.
0x8019	-20	User key set 1 segment digest verification: FAIL	Programming bitstream components do not match components programmed User key set 1 segment data has been corrupted or tampered with	Use the same programming file that was used to program the device.
0x801A	-20	User key set 2 segment digest verification: FAIL	Programming bitstream components do not match components programmed User key set 2 segment data has been corrupted or tampered with	Use the same programming file that was used to program the device.
0x801B	-20	Factory row and factory key segment digest verification: FAIL	Programming bitstream components do not match components programmed Factory row and factory key segment data has been corrupted or tampered with	Use the same programming file that was used to program the device.
0x801C	-20	Fabric configuration segment digest verification: FAIL	Programming bitstream components do not match components programmed. Fabric configuration segment data has been corrupted or tampered with	Use the same programming file that was used to program the device.

Error Code	Exit Code	Exit Message	Possible Cause	Possible Solution
0x801D 0x801E	-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	-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 encryption key	File contains an encrypted key that does not match the device 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	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 security, 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.
0x8023 0x8024	-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.
0x8025 0x8026	-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.
0x8027	-26	Authentication Error	Device does not support	Generate a programming

Error Code	Exit Code	Exit Message	Possible Cause	Possible Solution
0x8028		Insufficient device capabilities	the capabilities specified in programming file	file with the correct capabilities for the target device.
0x8029 0x802A	-27	Authentication Error Incorrect DEVICEID	Incorrect programming file Incorrect device in chain Signal integrity issues on JTAG pins	Choose the correct programming file and select the correct device in chain. Measure JTAG pins and noise or reflection. If TRST is left floating, then add pull-up to pin. Reduce the length of ground connection.
0x802B 0x802C	-28	Authentication Error Programming file is out of date, please regenerate	Programming file version is out of date	Generate programming file with latest version of Libero SoC.
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	-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.
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	Release the lock on the eNVM after your master has completed its access operations. Write 0x00 to "REQACCESS" register in eNVM Control Registers

Error Code	Exit Code	Exit Message	Possible Cause	Possible Solution
			corrupted Bitstream was incorrectly generated	(address 0x600801FC) to release the access. Program the device prior to running VERIFY action Regenerate bitstream file.
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 security, 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.
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.
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.
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.
0x8046	-28	Authentication Error	Old programming file	Generate programming file

Error Code	Exit Code	Exit Message	Possible Cause	Possible Solution
		Unsupported bitstream protocol version		with latest version of Libero SoC.
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.
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.
8x804A	10	Failed to program eNVM	Unstable voltage level. Signal integrity issues on JTAG pins.	Monitor related power supplies that cause the issue during programming; check for transients outside of Microsemi specifications. See your device datasheet for more information on transient specifications. Monitor JTAG supply pins during programming; measure JTAG signals for noise or reflection.
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.

Error Code	Exit Code	Exit Message	Possible Cause	Possible Solution
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.
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.

Programming File Actions - SmartFusion2 and IGLOO2

Libero SoC enables you to program security settings, FPGA Array, and eNVM features for SmartFusion2 and IGLOO2 device support.

You can program these features separately using different programming files or you can combine them into one programming file.

In the Design Flow window, expand **Program Design**, click **Run PROGRAM Action**, and right-click **Configure Actions/Procedures**.

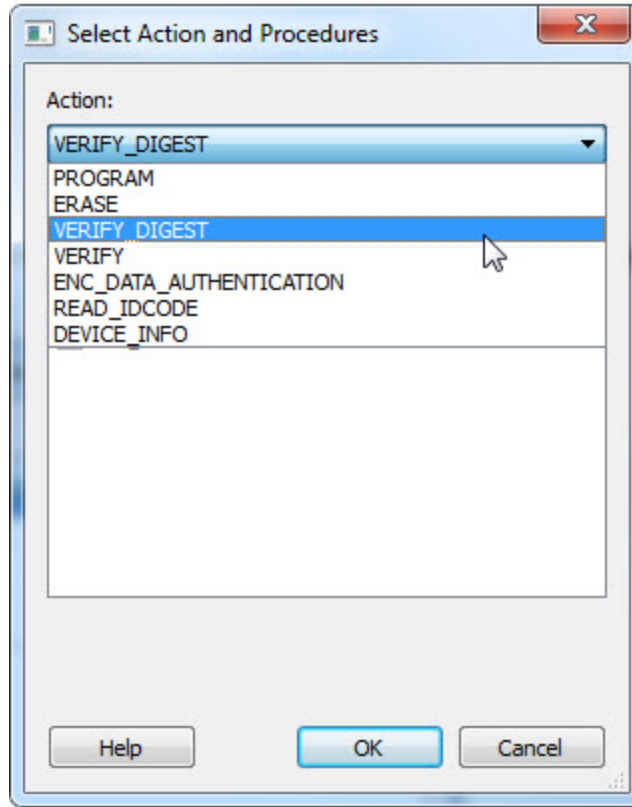


Figure 153 · Select Action and Procedures Dialog Box

Table 4 · Programming File Actions

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

Options Available in Programming Actions

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

Table 5 · Programming File Actions - Options

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

Export Bitstream - SmartFusion2 and IGLOO2

Bitstream Encryption with Default Key in Security Policy Manager - SmartFusion2 and IGLOO2

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

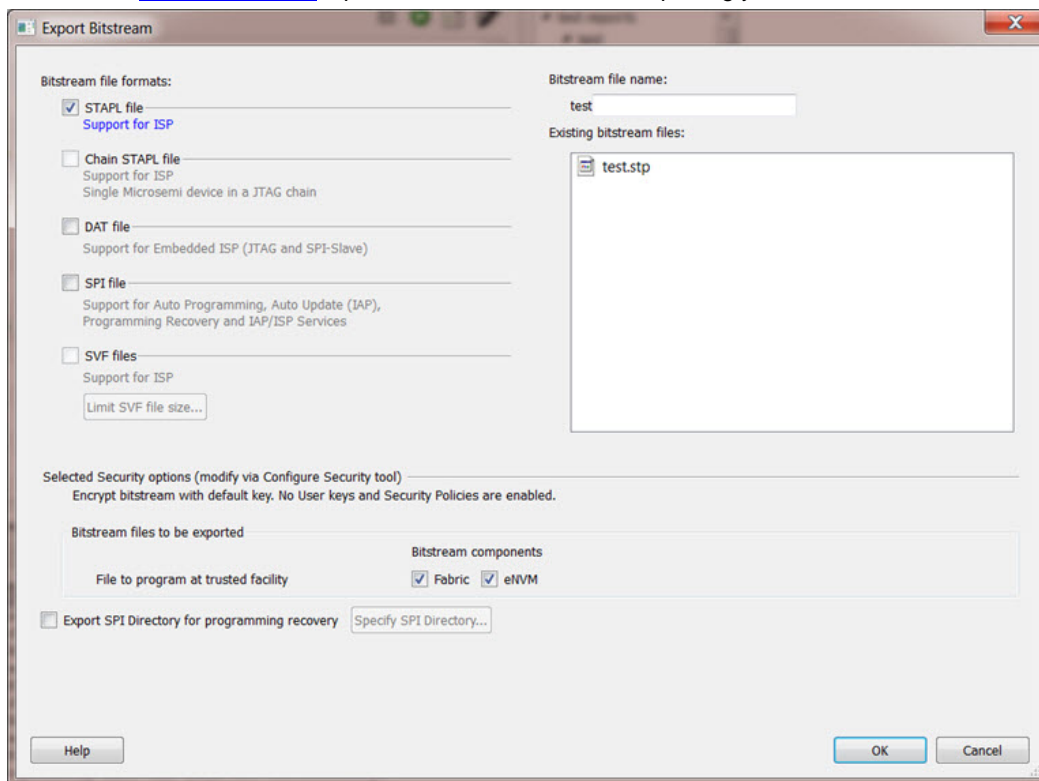


Figure 154 · Export Bitstream Dialog Box

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).

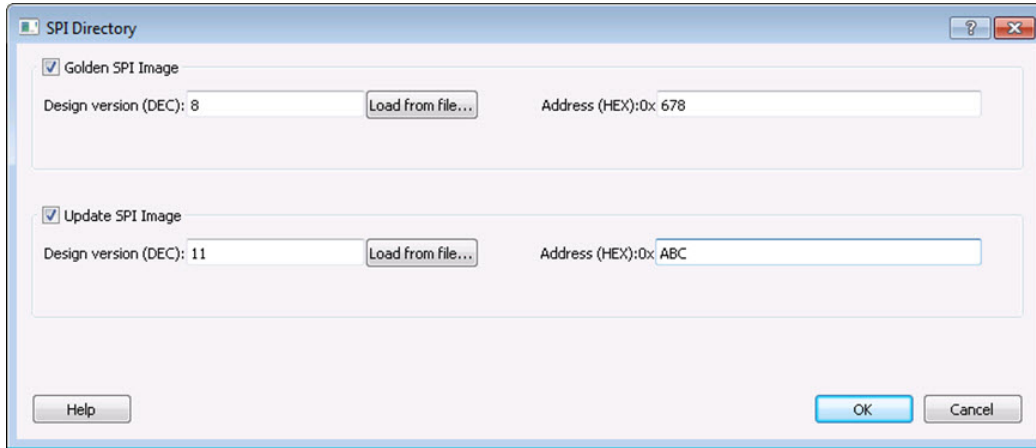


Figure 155 · SPI Directory Dialog Box

Enable Custom Security Options in the Security Policy Manager - SmartFusion2 and IGLOO2

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

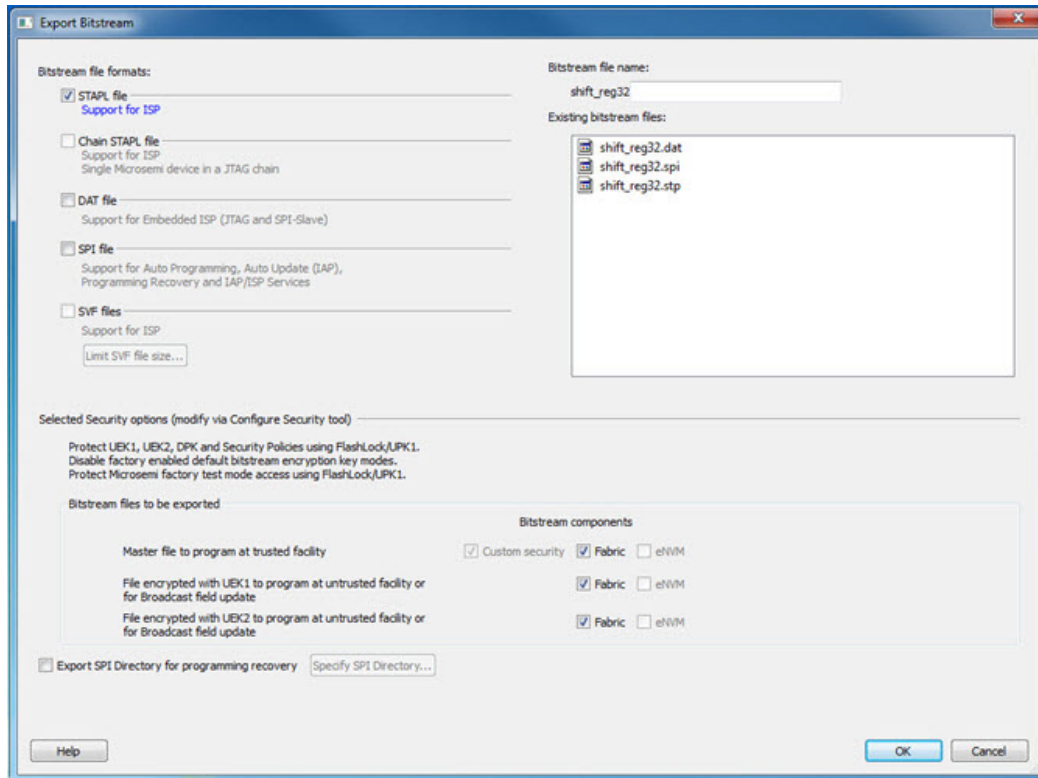


Figure 156 · Export Bitstream Dialog Box with Enable Custom Security Options in the Security Policy Manager

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 Format:

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 bitstream 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 bitstream files to be programmed. If the selected features is 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 bitstream files to be programmed. If the selected features is not protected by UPK1, the bitstream can be programmed at untrusted location, since it is encrypted with UEK2 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.

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).

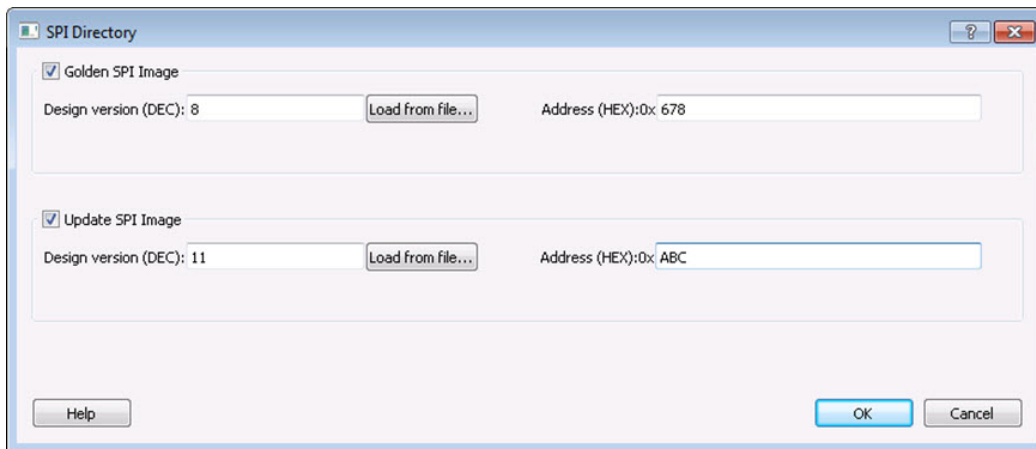


Figure 157 · SPI Directory Dialog Box

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. 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

Introduction to SmartDebug

Design debug is a critical phase of FPGA design flow. Microsemi's SmartDebug tool complements design simulation by allowing verification and troubleshooting at the hardware level. SmartDebug provides access to non-volatile memory (eNVM), SRAM, SERDES, DDR controller, and probe capabilities. Microsemi SmartFusion2 System-on-chip (SoC) field programmable gate array (FPGA), IGLOO2 FPGA, and RTG4 FPGA devices have built-in probe logic that greatly enhance the ability to debug logic elements within the device. SmartDebug accesses the built-in probe points through the Active Probe and Live Probe features, which enables designers to check the state of inputs and outputs in real-time without re-layout of the design.

Use Models

SmartDebug can be run in two modes:

- Integrated mode from the Libero Design Flow
- Standalone mode

Integrated Mode

When run in integrated mode from Libero, SmartDebug can access all design and programming hardware information. No extra setup step is required. In addition, the Probe Insertion feature is available in Debug FPGA Array.

To open SmartDebug in the Libero Design Flow window expand **Debug Design** and double-click **SmartDebug Design**.

Standalone Mode

SmartDebug can be installed separately in the setup containing FlashPro, 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 go through SmartDebug project creation and import a Design Debug Data Container (DDC) file, exported from Libero, to access all the 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.

Standalone Mode Use Model Overview

The main use model for standalone SmartDebug requires users to generate the DDC file from Libero and import it into a SmartDebug project to gain full access to the device debug features. Alternatively, SmartDebug can be used without a DDC file with a limited feature set.

Supported Families, Programmers, and Operating Systems

Programming and Debug: SmartFusion2, IGLOO2, and RTG4

Programmers: FlashPRO3, FlashPRO4, and FlashPRO5

Operating Systems: Windows XP, Windows 7, and RHEL 6.x

Getting Started with SmartDebug

This topic introduces the basic elements and features of SmartDebug. If you are already familiar with the user interface, proceed to the Solutions to Common Issues Using SmartDebug or Frequently Asked Questions sections.

SmartDebug enables you to use JTAG to interrogate and view embedded silicon features and device status (FlashROM, Security Settings, Embedded Flash Memory (NVM)). SmartDebug is available as a part of the FlashPro programming tool.

See [Using SmartDebug with SmartFusion2, IGLOO2, and RTG4](#) for an overview of the use flow.

You can use the debugger to:

- [Get device status and view diagnostics](#)
- [Use the Embedded Flash Memory Debug GUI to read out and compare your content with your original files](#)

Using SmartDebug with SmartFusion2, IGLOO2, and RTG4

The most common flow for SmartDebug is:

1. [Create your design](#). You must have a FlashPro programmer connected to use SmartDebug.
2. Expand **Debug Design** and double-click **Smart Debug Design** in the Design Flow window. SmartDebug opens for your target device.
3. Click **View Device Status** to view the device status report and check for issues.
4. Examine individual silicon features, such as FPGA debug.

Create Standalone SmartDebug Project

A standalone SmartDebug project can be configured in two ways:

- Import DDC files exported from Libero
- Construct Automatically

From the SmartDebug main window, click **Project** and choose **New Project**. The Create SmartDebug Project dialog box opens.

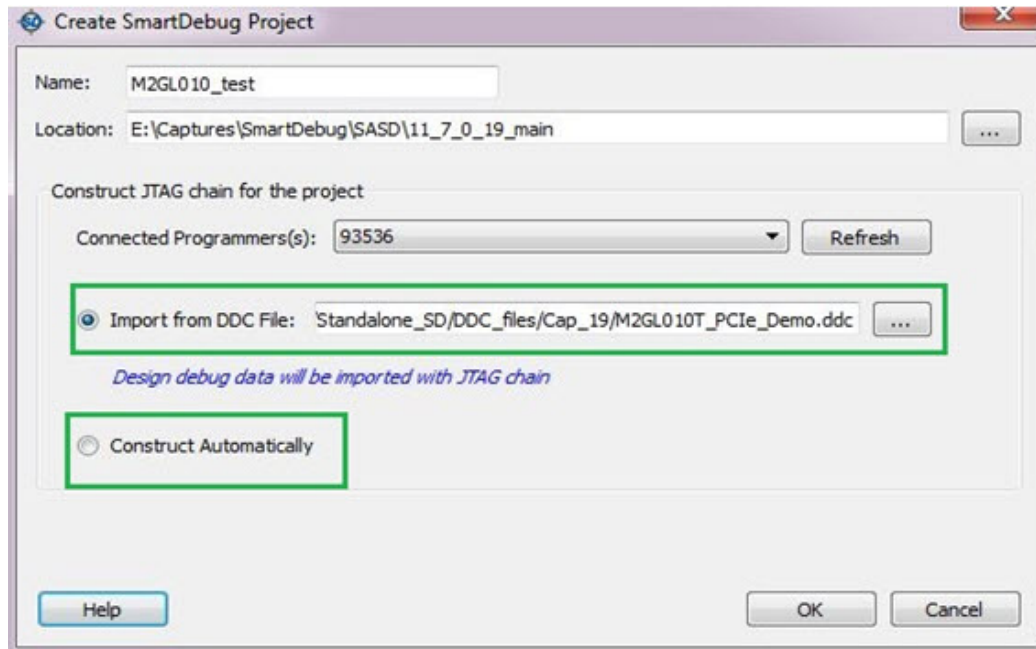


Figure 158 · Create SmartDebug Project Dialog Box

Import from DDC File (created from Libero)

When you select the **Import from DDC File** option in the Create SmartDebug Project dialog box, the Design Debug Data of the target device and all hardware and JTAG chain information present in the DDC file exported in Libero are automatically inherited by the SmartDebug project. The programming file information loaded onto other Microsemi devices in the chain, including ProAsic3/E, SmartFusion, and Fusion devices, are also transferred to the SmartDebug project.

Debug data is imported from the DDC file (created through Export SmartDebug Data in Libero) into the debug project, and the devices are configured using data from the DDC file.

Construct Automatically

When you select the **Construct Automatically** option, a debug project is created with all the devices connected in the chain for the selected programmer. This is equivalent to Construct Chain Automatically in FlashPRO.

Configuring a Generic Device

For Microsemi devices having the same JTAG IDCODE (i.e., multiple derivatives of the same Die—for example, M2S090T, M2S090TS, and so on), the device type must be configured for SmartDebug to enable relevant features for debug. The device can be configured by loading the programming file, by manually selecting the device using Configure Device, or by importing DDC files through Programming Connectivity and Interface. When the device is configured, all debug options are shown.

For debug projects created using Construct Automatically, you can use the following options to debug the devices:

- Load the programming file – Right-click the device in Programming Connectivity and Interface.
- Import Debug Data from DDC file – Right-click the device in Programming Connectivity and Interface.

The appropriate debug features of the targeted devices are enabled after the programming file or DDC file is imported.

Connected FlashPRO Programmers

The drop-down lists all FlashPro programmers connected to the device. Select the programmer connected to the chain with the debug device. At least one programmer must be connected to create a standalone SmartDebug project.

Before a debugging session or after a design change, program the device through Programming Connectivity and Interface.

See Also

[Programming Connectivity and Interface](#)

[View Device Status](#)

[Export SmartDebug Data \(from Libero\)](#)

Import from DDC File (created from Libero)

When you select the **Import from DDC File** option in the Create SmartDebug Project dialog box, the Design Debug Data of the target device and all hardware and JTAG chain information present in the DDC file exported in Libero are automatically inherited by the SmartDebug project. The programming file information loaded onto other Microsemi devices in the chain, including ProAsic3/E, SmartFusion, and Fusion devices, are also transferred to the SmartDebug project.

Debug data is imported from the DDC file (created through Export SmartDebug Data in Libero) into the debug project, and the devices are configured using data from the DDC file.

Construct Automatically

When you select the **Construct Automatically** option, a debug project is created with all the devices connected in the chain for the selected programmer. This is equivalent to Construct Chain Automatically in FlashPRO.

Configuring a Generic Device

For Microsemi devices having the same JTAG IDCODE (i.e., multiple derivatives of the same Die—for example, M2S090T, M2S090TS, and so on), the device type must be resolved for SmartDebug to enable relevant features for debug. The device can be resolved by loading the programming file, manually selecting the device using Configure Device, or by importing DDC files through Programming Connectivity and Interface. When the device is resolved, all debug options are shown.

For debug projects created using Construct Automatically, you can use the following options to debug the devices:

- Load the programming file – Right-click the device in Programming Connectivity and Interface.
- Import Debug Data from DDC file – Right-click the device in Programming Connectivity and Interface.

The appropriate debug features of the targeted devices are enabled after the programming file or DDC file is imported.

Smartdebug User Interface

Standalone SmartDebug User Interface

You can start standalone SmartDebug from the Libero installation folder or from the FlashPRO installation folder.

Windows:

<Libero Installation folder>/Designer/bin/sdebug.exe

<FlashPRO Installation folder>/bin/sdebug.exe

Linux :

<Libero Installation folder>/ bin/sdebug

<FlashPRO Installation folder>/bin/sdebug

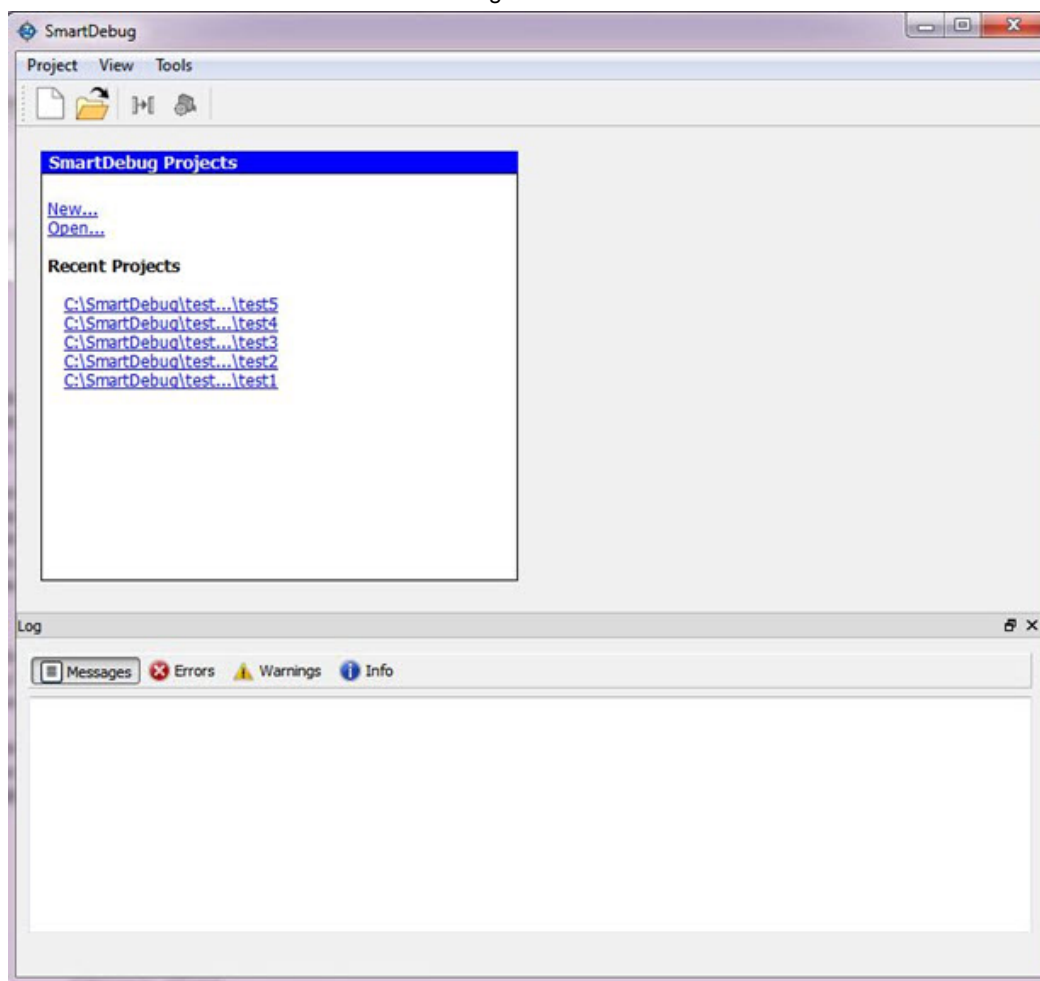


Figure 159 · Standalone SmartDebug Main Window

Project Menu

The Project menu allows you do the following:

- Create new SmartDebug projects (**Project > New Project**)
- Open existing debug projects (**Project > Open Project**)
- Execute SmartDebug-specific Tcl scripts (**Project > Execute Script**)
- Export SmartDebug-specific commands to a script file (**Project > Export Script File**)
- See a list of recent SmartDebug projects (**Project > Recent Projects**).

Log Window

SmartDebug displays the Log window by default when it is invoked. To suppress the Log window display, click the View menu and toggle **View Log**.

The Log window has four tabs:

- Messages** – displays standard output messages
- Errors** – displays error messages
- Warnings** – displays warning messages
- Info** – displays general information

Tools Menu

The Tools menu includes Programming Connectivity and Interface and Programmer Settings options, which are enabled after creating or opening a SmartDebug project.

Programming Connectivity and Interface

To open the Programming Connectivity and Interface dialog box, from the standalone SmartDebug Tools menu, choose **Programming Connectivity and Interface**. The Programming Connectivity and Interface dialog box displays the physical chain from TDI to TDO.

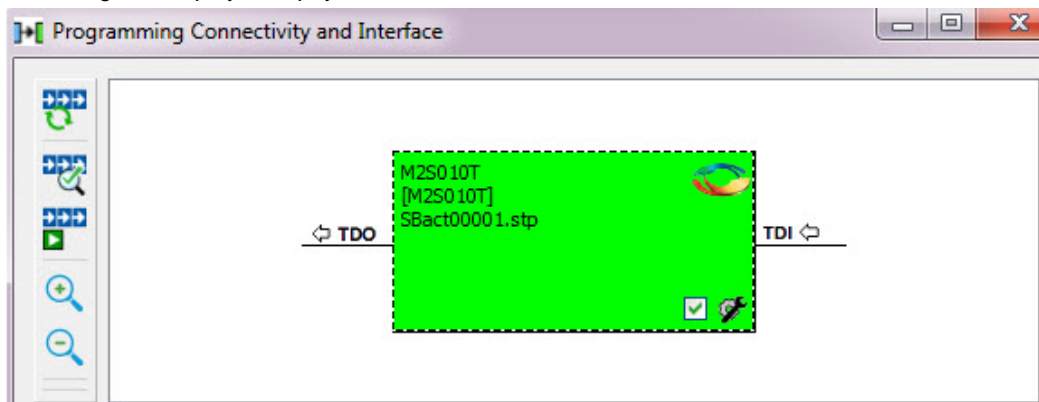


Figure 160 · Programming Connectivity and Interface Dialog Box – Project created using Import from DDC File

All devices in the chain are disabled by default when a standalone SmartDebug project is created using the **Construct Automatically** option in the Create SmartDebug Project dialog box.

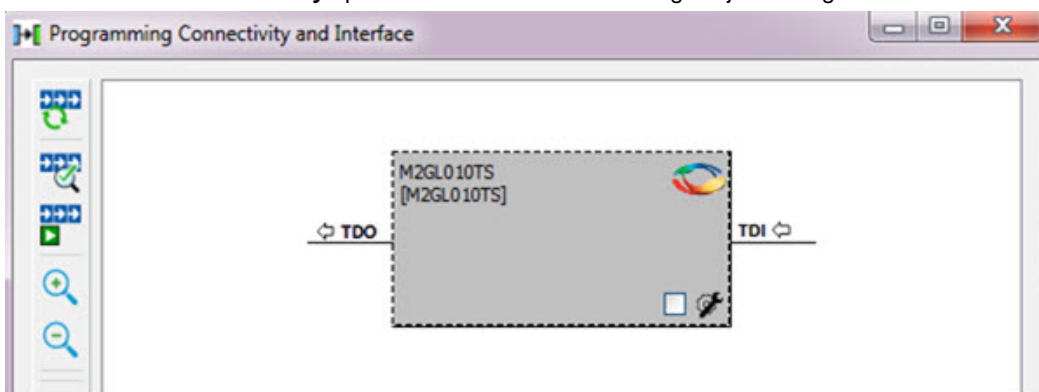


Figure 161 · Programming Connectivity and Interface window – Project created using Construct Automatically

The Programming Connectivity and Interface dialog box includes the following actions:

- **Construct Chain Automatically** - Automatically construct the physical chain.

Running Auto-Construct in Programming Connectivity and Interface removes all existing debug/programming data included using DDC/programming files. The project is the same as a new project created using the Construct Automatically option.

- **Scan and Check Chain** – Scan the physical chain connected to the programmer and check if it matches the chain constructed in the scan chain block diagram.
- **Run Programming Action** – Option to program the device with the selected programming procedure.

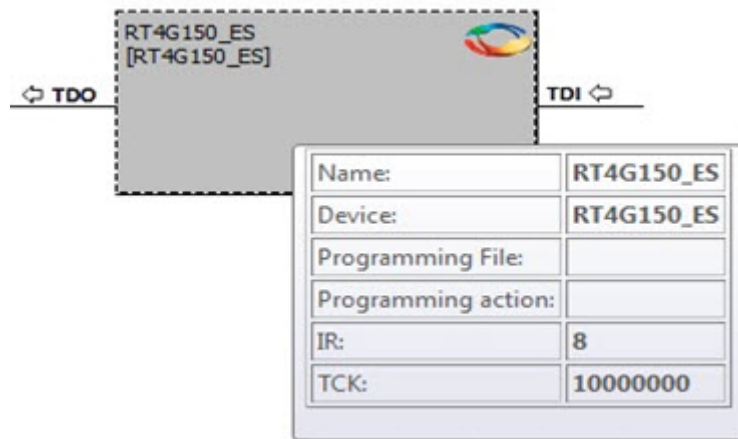
When two devices are connected in the chain, the programming actions are independent of the device. For example, if M2S090 and M2GL010 devices are connected in the chain, and the M2S090 device is to be programmed and the M2GL010 device is to be erased, both actions can be done at the same time using the Run Programming Action option.

- **Zoom In** – Zoom into the scan chain block diagram.
- **Zoom Out** – Zoom out of the scan chain block diagram.

Hover Information

The device tooltip displays the following information if you hover your cursor over a device in the scan chain block diagram:

- **Name:** User-specified device name. This field indicates the unique name specified by the user in the Device Name field in Configure Device (right-click **Properties**).
- **Device:** Microsemi device name.
- **Programming File:** Programming file name.
- **Programming action:** The programming action selected for the device in the chain when a programming file is loaded.
- **IR:** Device instruction length.
- **TCK:** Maximum clock frequency in MHz to program a specific device; standalone SmartDebug uses this information to ensure that the programmer operates at a frequency lower than the slowest device in the chain.

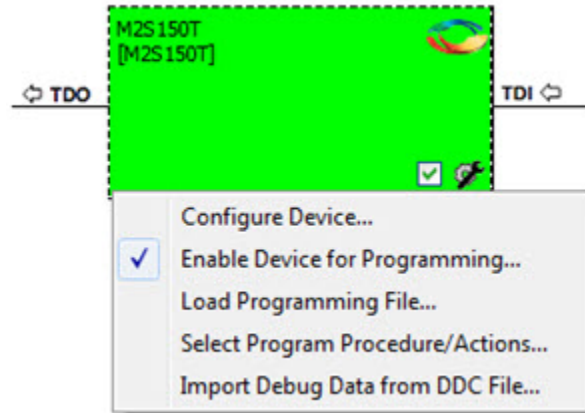


Device Chain Details

The device within the chain has the following details:

- User-specified device name
- Device name
- Programming file name
- Programming action – Select **Enable Device for Programming** to enable the device for programming. Enabled devices are green, and disabled devices are grayed out.

Right-click Properties



Configure Device - Ability to reconfigure the device.

- **Family and Die:** The device can be explicitly configured from the Family, Die drop-down.
- **Device Name:** Editable field for providing user-specified name for the device.

Enable Device for Programming - Select to enable the device for programming. Enabled devices are shown in green, and disabled devices are grayed out.

Load Programming File - Load the programming file for the selected device.

Select Programming Procedure/Actions- Option to select programming action/procedures for the devices connected in the chain.

- **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.

Import Debug Data from DDC File - Option to import debug data information from the DDC file.

The DDC file selected for import into device must be created for a compatible device. When the DDC file is imported successfully, all current device debug data is removed and replaced with debug data from the imported DDC file.

The JTAG Chain configuration from the imported DDC file is ignored in this option.

If a programming file is already loaded into the device prior to importing debug data from the DDC file, the programming file content is replaced with the content of the DDC file (if programming file information is included in the DDC file).

Debug Context Save

Debug context refers to the user selections in debug options such as Debug FPGA Array, Debug SERDES, and View Flash Memory Content. In standalone SmartDebug, the debug context of the current session is saved or reset depending on the user actions in Programming Connectivity and Interface.

The debug context of the current session is retained for the following actions in Programming Connectivity and Interface:

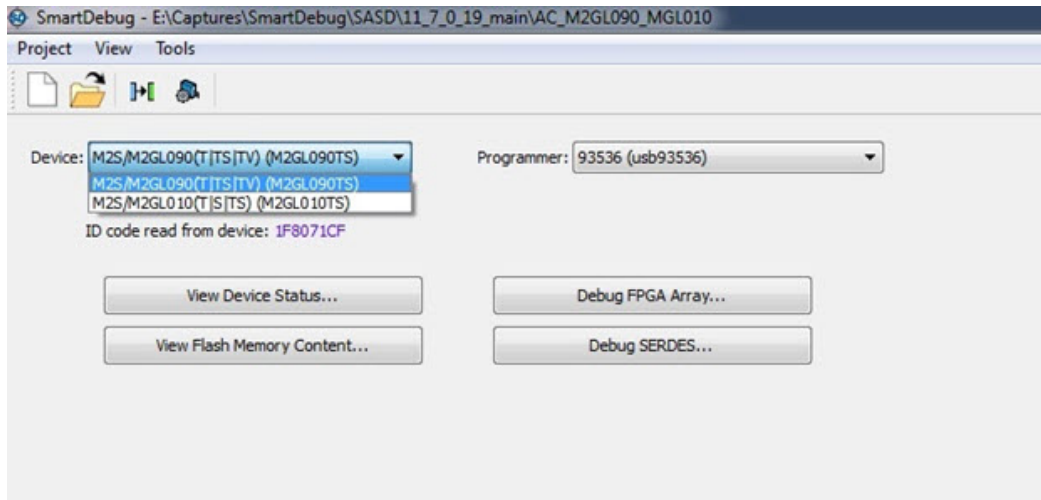
- Enable Device for Programming
- Select Programming Procedure/Actions
- Scan and Check Chain
- Run Programming Action

The debug context of the current session is reset for the following actions in Programming Connectivity and Interface:

- Auto Construct – Clears all the existing debug data. You need to reimport the debug data from DDC file.
- Import Debug Data from DDC file
- Configure Device – Renaming the device in the chain
- Configure Device – Family/Die change
- Load Programming File

Selecting Devices for Debug

Standalone SmartDebug provides an option to select the devices connected in the JTAG chain for debug. The device debug context is not saved when another debug device is selected.



View Device Status (SmartFusion2, IGLOO2, and RTG4 Only)

Click **View Device Status** in the standalone SmartDebug main window to display the Device Status Report. The Device Status Report is a complete summary of IDCode, device certificate, design information, programming information, digest, and device security information. Use this dialog box to save or print your information for future reference.

Note: This information is available for SmartFusion2 and IGLOO2 devices only. For RTG4 devices, View Device Status displays IDCode information only.

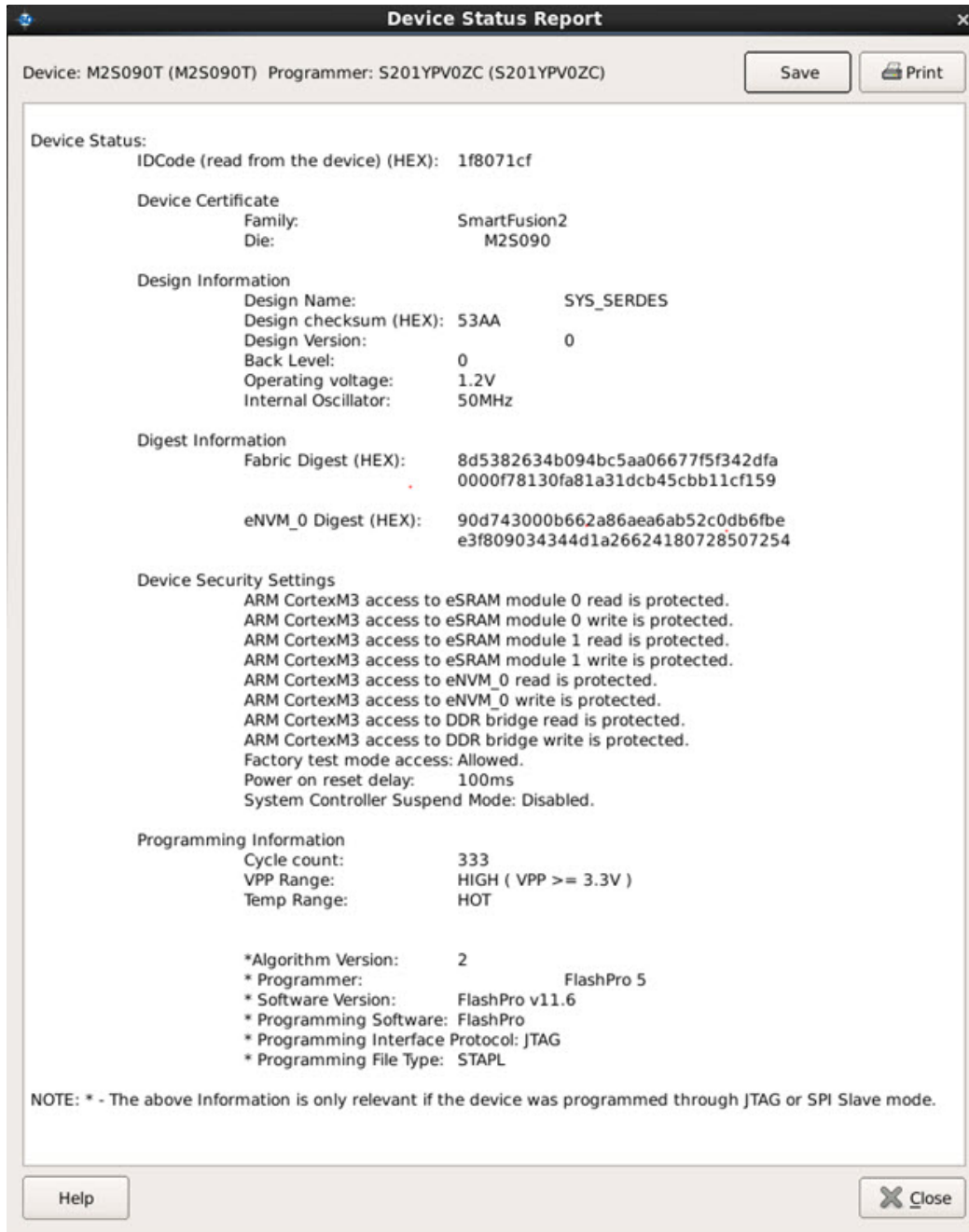


Figure 162 · Device Status Report

IdCode

IDCode read from the device under debug.

Device Certificate

Device certificate displays Family and Die information if device certificate is installed on the device.

If the device certificate is not installed on the device, a message indicating that the device certificate may not have been installed is shown.

Design Information

Design Information displays the following:

- Design Name
- Design Checksum
- Design Version
- Back Level
- Operating Voltage
- Internal Oscillator

Digest Information

Digest Information displays Fabric Digest, eNVM_0 Digest and eNVM_1 Digest (for M2S090 and M2S150 devices only) computed from the device during programming. eNVM digest is shown when eNVM is used in the design.

Device Security Settings

Device Security Settings indicate the following:

- Factory test mode access
- Power on reset delay
- System Controller Suspend Mode

In addition, if custom security options are used, Device Security Settings indicate:

- User Lock segment is protected
- User Pass Key 1/2 encrypted programming is enforced for the FPGA Array
- User Pass Key 1/2 encrypted programming is enforced for the eNVM_0 and eNVM_1
- SmartDebug write access to Active Probe and AHB mem space
- SmartDebug read access to Active Probe, Live Probe & AHB mem space
- UJTAG access to fabric

Programming Information

Programming Information displays the following:

- Cycle Count
- VPP Range
- Temp Range
- Algorithm Version
- Programmer
- Software Version
- Programming Software
- Programming Interface Protocol
- Programming File Type

Embedded Flash Memory (NVM) Content Dialog Box (SmartFusion2 and IGLOO2 Only)

The NVM content dialog box is divided into two sections:

- View content of Flash Memory pages (as shown in the figure below)

- Check page status and identify if a page is corrupted or if the write count limit has exceeded the 10-year retention threshold

Choose the eNVM page contents to be viewed by specifying the page range (i.e., start page and the end page) and click **Read from Device** to view the values.

You must click **Read from Device** each time you specify a new page range to update the view.

Specify a page range if you wish to examine a specific set of pages. In the Retrieved Data View, you can enter an Address value (such as 0010) in the Go to Address field and click the corresponding button to go directly to that address. Page Status information appears to the right.

Contents of Page Status

- ECC1 detected and corrected
- ECC2 detected
- Write count of the page
- If write count has exceeded the threshold
- If the page is used as ROM (first page lock)
- Overwrite protect (second page lock)
- Flash Freeze state (deep power down)

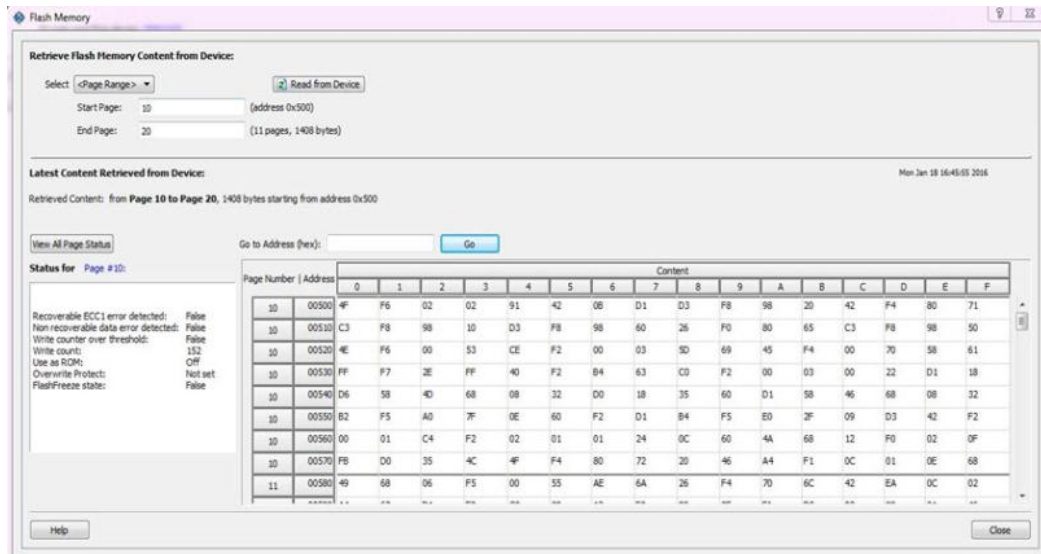


Figure 163 - Flash Memory Dialog Box for a SmartFusion2 Device (SmartDebug)

The page status gets updated when you:

- Click Page Range
- Click a particular cell in the retrieved eNVM content table
- Scroll pages from the keyboard using the Up and Down arrow keys
- Click Go to Address (hex)

The retrieved data table displays the content of the page range selection. If content cannot be read (for example, pages are read-protected, but security has been erased or access to eNVM private sectors), Read from Device reports an error.

Click **View Detailed Status** for a detailed report on the page range you have selected.

For example, if you want to view a report on pages 1-3, set the Start Page to 1, set the End Page to 3, and click **Read from Device**. Then click **View Detailed Status**. The figure below is an example of the data for a specific page range.

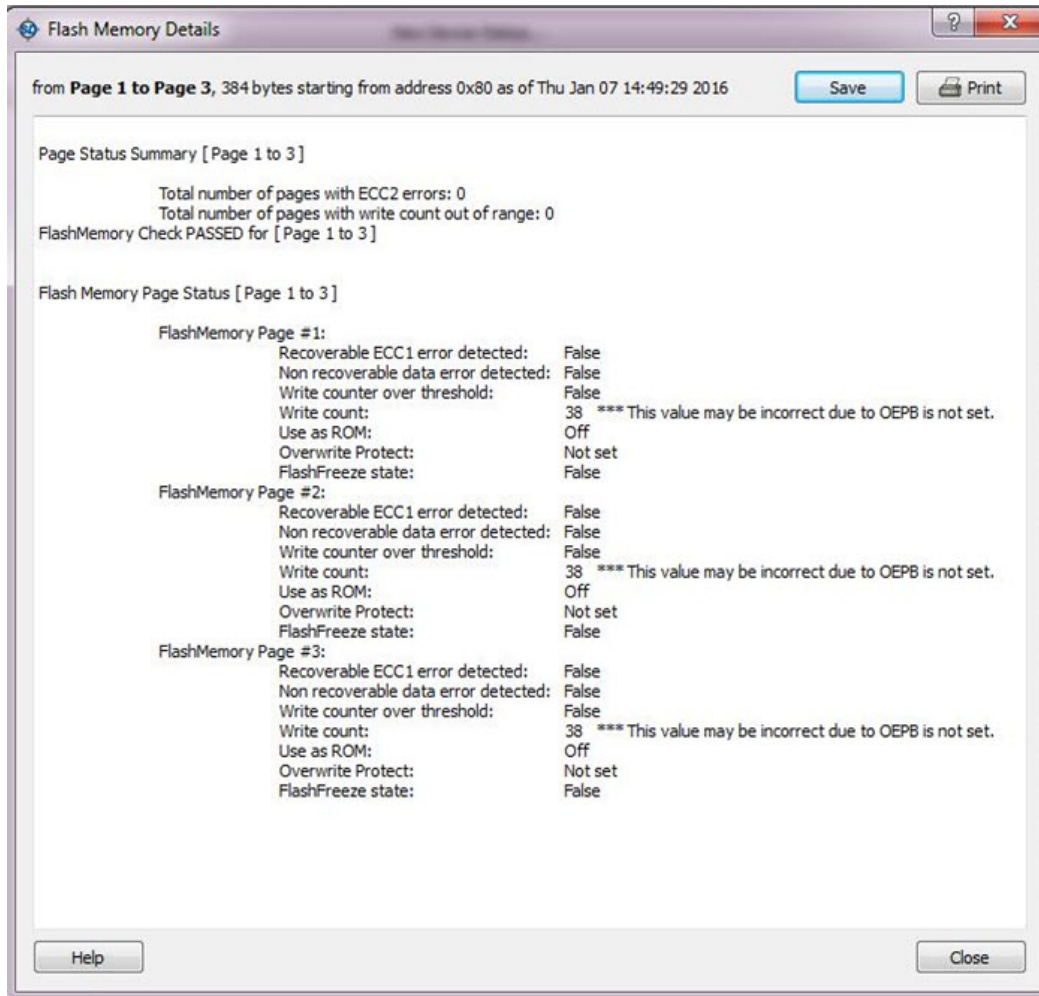


Figure 164 · Flash Memory Details Dialog Box (SmartDebug)

Debugging

Debug FPGA Array (SmartFusion2, IGLOO2, and RTG4 Only)

In the Debug FPGA Array dialog box, you can view your Live Probes, Active Probes, and Memory Blocks, and Insert Probes (Probe Insertion).

The Debug FPGA Array dialog box includes the following four tabs:

- [Live Probes](#)
- [Active Probes](#)
- [Memory Blocks](#)
- [Probe Insertion](#)

Hierarchical View

The Hierarchical View lets you view the instance level hierarchy of the design programmed on the device and select the signals to add to the Live Probes, Active Probes, and Probe Insertion tabs in the Debug FPGA Array dialog box.

- **Instance** – Displays the probe points available at the instance level.

- **Primitives** – Displays the lowest level of probeable points in the hierarchy for the corresponding component —i.e., leaf cells (hard macros on the device).

You can expand the hierarchy tree to see lower level logic.

Signals with the same name are grouped automatically into a bus that is presented at instance level in the instance tree.

The probe points can be added by selecting any instance or the leaf level instance in the Hierarchical View. Adding an instance adds all the probe able points available in the instance to Live Probes, Active Probes, and Probe Insertion.

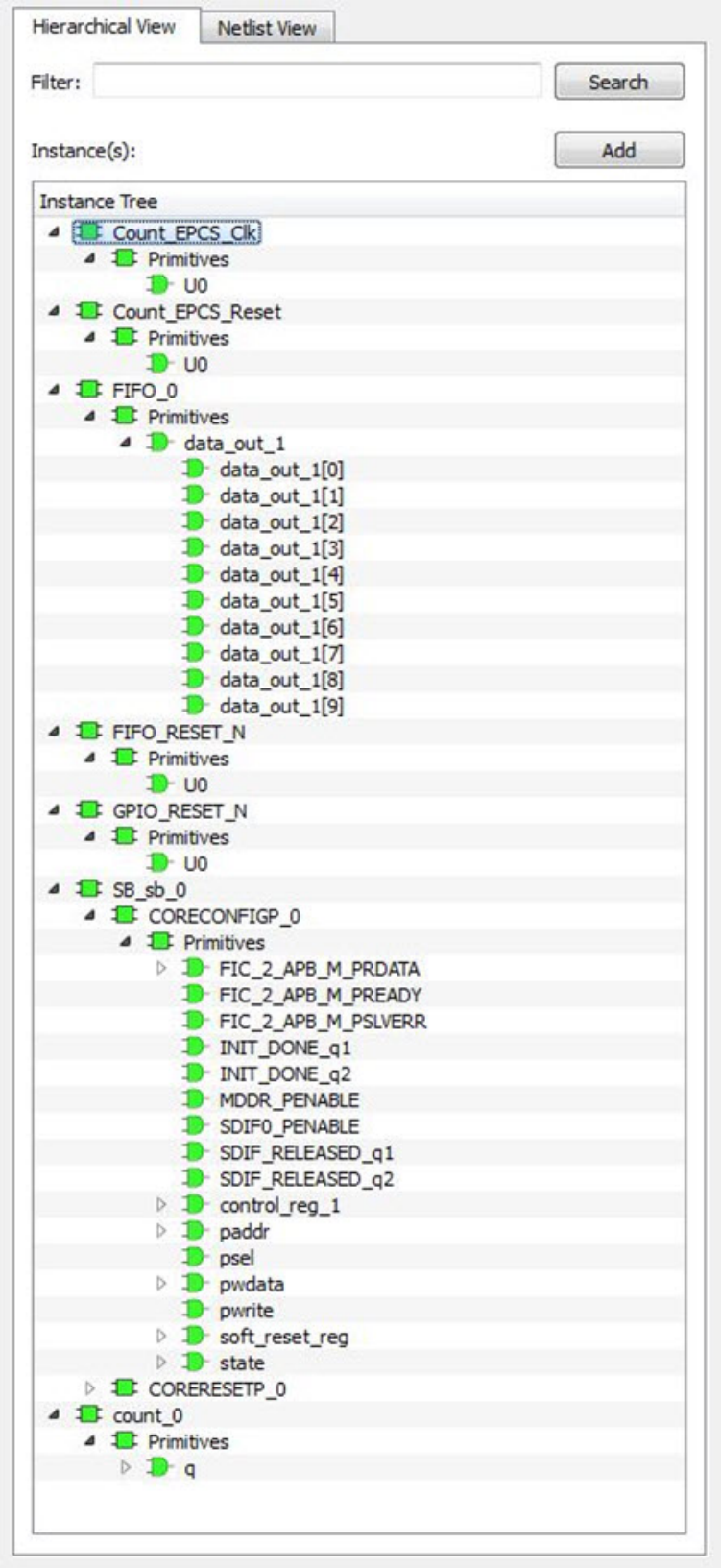


Figure 165 · Hierarchical View

Search

In Live Probes, Active Probes, and the Probe Insertion UI, a search option is available in the Hierarchical View. You can use wildcard characters such as * or ? in the search column for wildcard matching.

Probe points of leaf level instances resulting from a search pattern can only be added to Live Probes, Active Probes, and the Probe Insertion UI. You cannot add instances of search results in the Hierarchical View.

Netlist View

The Netlist View displays a flattened net view of all the probe able points present in the design, along with the associated cell type.

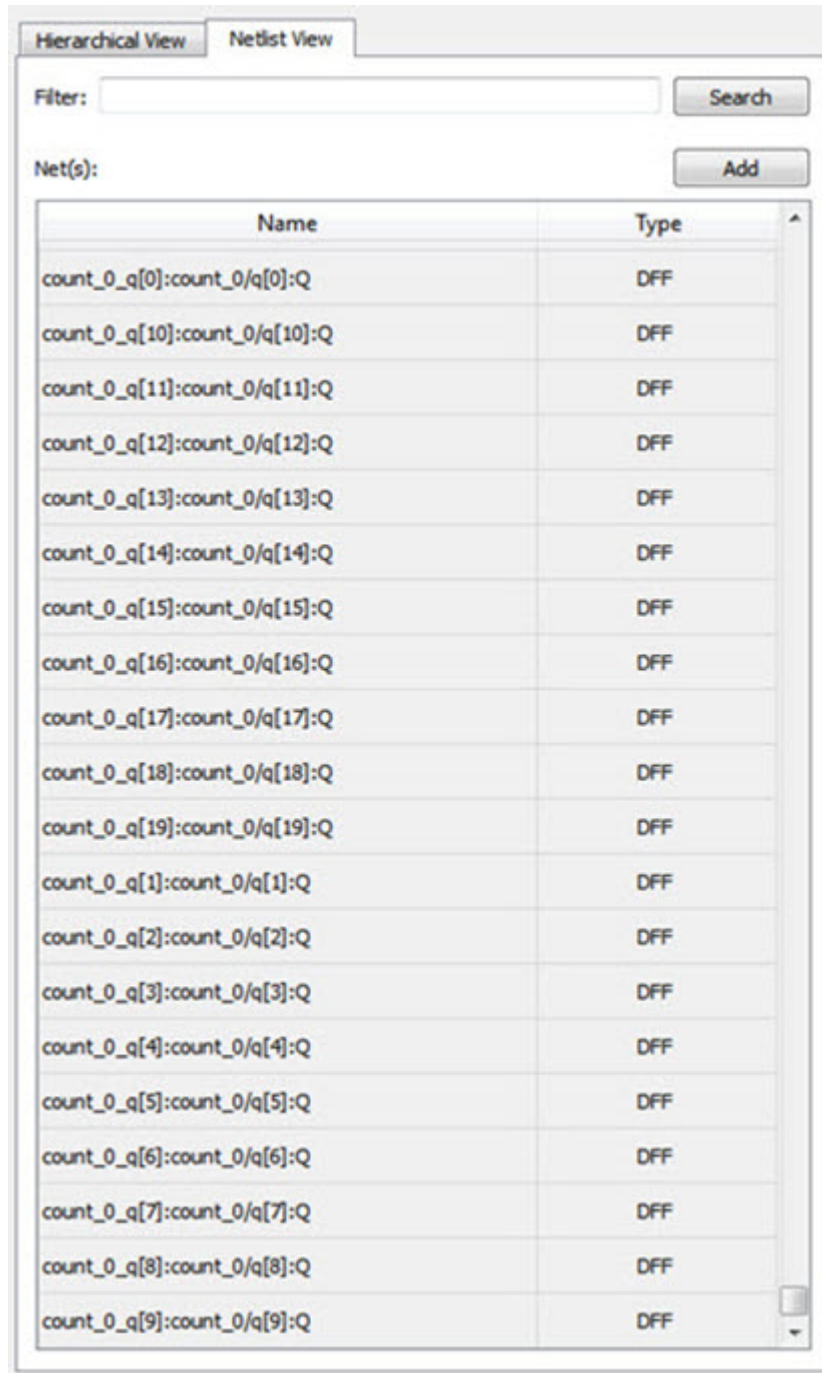


Figure 166 · Netlist View

Search

A search option is available in the Netlist View for Live Probes, Active Probes, and Probe Insertion. You can use wildcard characters such as * or ? in the search column for wildcard matching.

Live Probes (SmartFusion2, IGLOO2, and RTG4)

The Live Probes tab displays a table with the probe name and pin type.

Note: SmartFusion2 and IGLOO2 support two probe channels, and RTG4 supports one probe channel.

SmartFusion2 and IGLOO2

Two probe channels (ChannelA and ChannelB) are available. When a probe name is selected, it can be assigned to either ChannelA or ChannelB.

You can assign a probe to a channel by doing either of the following:

- Right-click a probe in the table and choose **Assign to Channel A** or **Assign to Channel B**.
- Click the **Assign to Channel A** or **Assign to Channel B** button to assign the probe selected in the table to the channel. The buttons are located below the table.

When the assignment is complete, the probe name appears to the right of the button for that channel, and SmartDebug configures the ChannelA and ChannelB I/Os to monitor the desired probe points. Because there are only two channels, a maximum of two internal signals can be probed simultaneously.

Click the **Unassign Channels** button to clear the live probe names to the right of the channel buttons and discontinue the live probe function during debug.

Note: At least one channel must be set; if you want to use both probes, they must be set at the same time.

The Active Probes READ/WRITE overwrites the settings of Live Probe channels (if any).

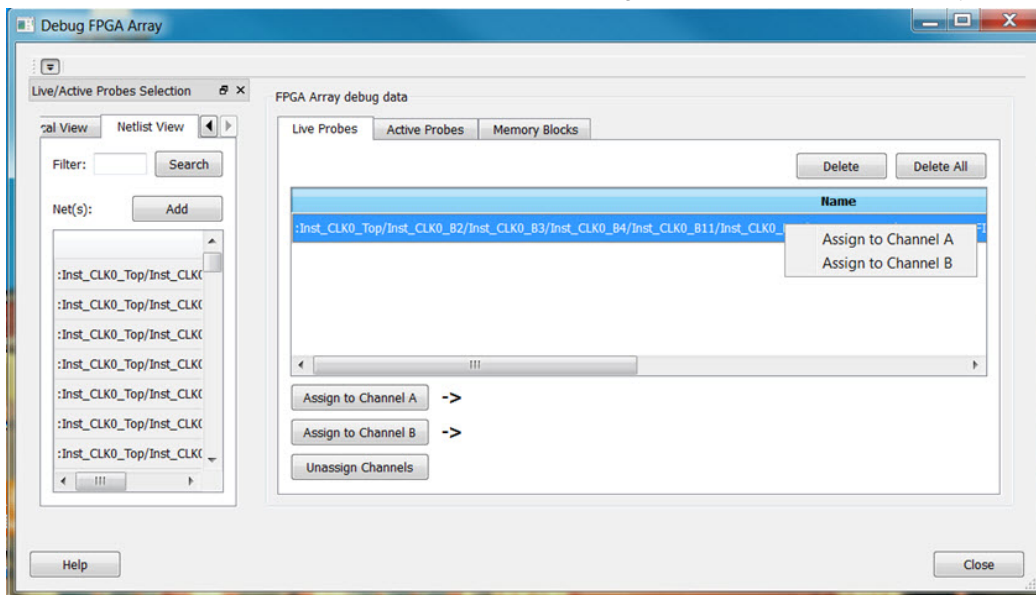


Figure 167 · Live Probes Tab (SmartFusion2 and IGLOO2) in SmartDebug FPGA Array Dialog Box

RTG4

One probe channel (Probe Read Data Pin) is available for RTG4 for debug. When a probe name is selected, it can be assigned to the Probe Channel (Probe Read Data Pin).

You can assign a probe to a channel by doing either of the following:

- Right-click a probe in the table and choose **Assign to Probe Read Data Pin**.
- Click the **Assign to Probe Read Data Pin** button to assign the probe selected in the table to the channel. The button is located below the table.

Click the **Unassign probe read data pin** button to clear the live probe name to the right of the channel button and discontinue the live probe function during debug.

The Active Probes READ/WRITE overwrites the settings of Live Probe channels (if any).

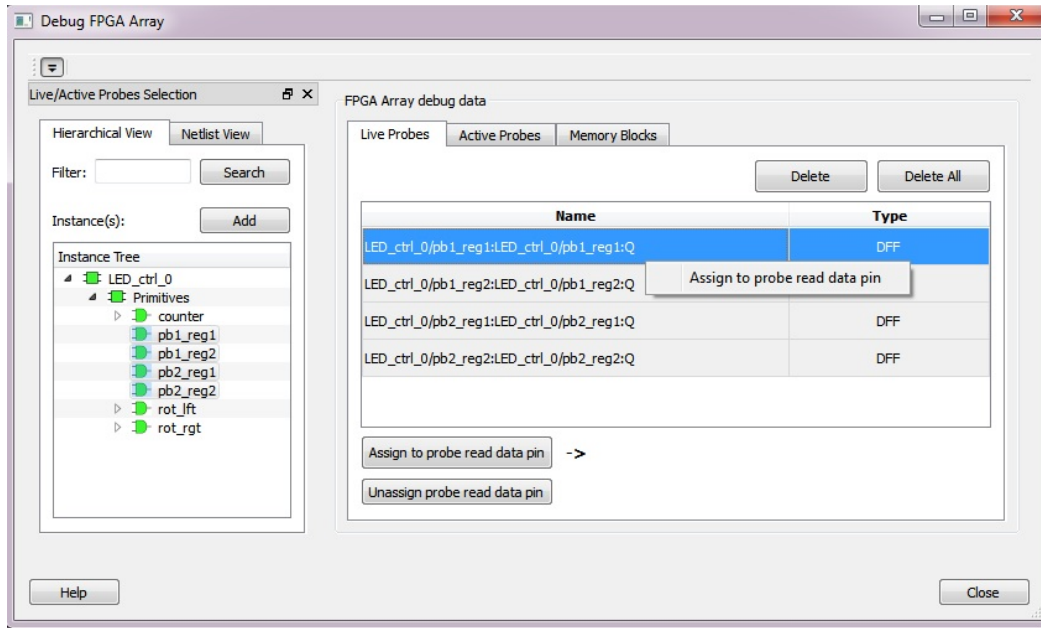


Figure 168 · Live Probes Tab (RTG4) in SmartDebug FPGA Array Dialog Box

Active Probes (SmartFusion2, IGLOO2, and RTG4)

In the left pane of the Active Probes tab, all available Probe Points are listed in instance level hierarchy in the Hierarchical View. All Probe Names are listed with the Name and Type (which is the physical location of the flip-flop) in the Netlist View.

Select probe points from the Hierarchical View or Netlist View, right-click and choose **Add** to add them to the Active Probes UI. You can also add the selected probe points by clicking the **Add** button. The probes list can be filtered by using the Filter box.

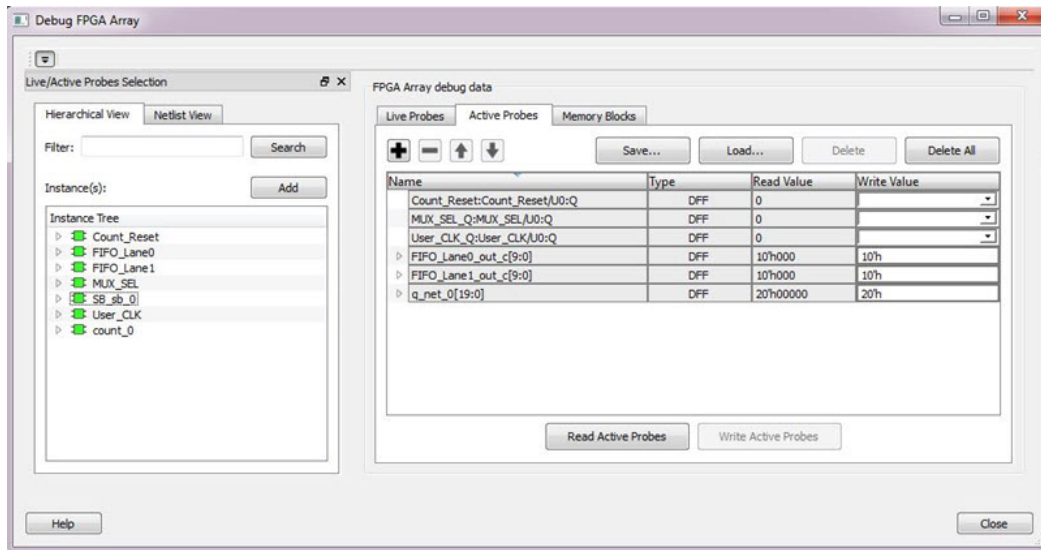


Figure 169 · Active Probes Tab in SmartDebug FPGA Array Dialog Box

When you have selected the desired probe, points appear in the Active Probe Data chart and you can read and write multiple probes (as shown in the figure below).

You can use the following options in the Write Value column to modify the probe signal added to the UI:

- Drop-down menu with values '0' and '1' for individual probe signals

- Editable field to enter data in hex or binary for a probe group or a bus

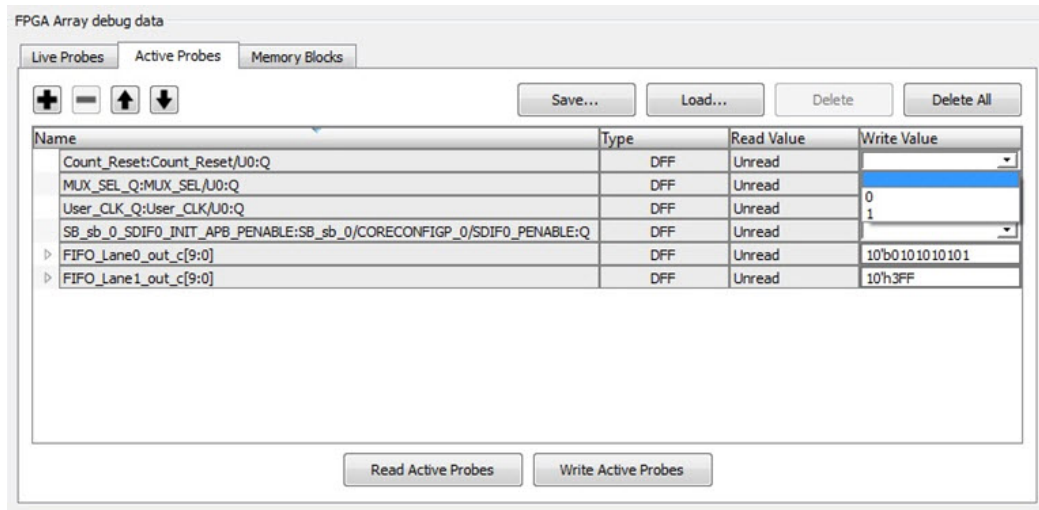


Figure 170 · Active Probes Tab - Write Value Column Options

Probe Grouping (Active Probes Only)

During the debug cycle of the design, designers often want to examine the different signals. In large designs, there can be many signals to manage. The Probe Grouping feature assists in comprehending multiple signals as a single entity. This feature is applicable to Active Probes only. Probe nets with the same name are automatically grouped in a bus when they are added to the Active Probes tab. Custom probe groups can also be created by manually selecting probe nets of a different name and adding them into the group

The Active Probes tab provides the following options for probe points that are added from the Hierarchical View/Netlist View:

- Display of bus name. An automatically generated bus name cannot be modified. Only custom bus names can be modified.
- Expand/collapse of bus or probe group
- Move Up/Down the signal or bus or probe group
- Save (Active Probes list)
- Load (already saved Active Probes list)
- Delete (applicable to a single probe point added to the Active Probes tab)
- Delete All (deletes all probe points added to the Active Probes tab)
- In addition, the context (right-click) menu provides the following operations:
 - o Create Group, Add/Move signals to Group, Remove signals from Group,
 - o Ungroup
 - o Reverse bit order, Change Radix for a bus or probe group
 - o Read, Write, or Delete the signal or bus or probe group

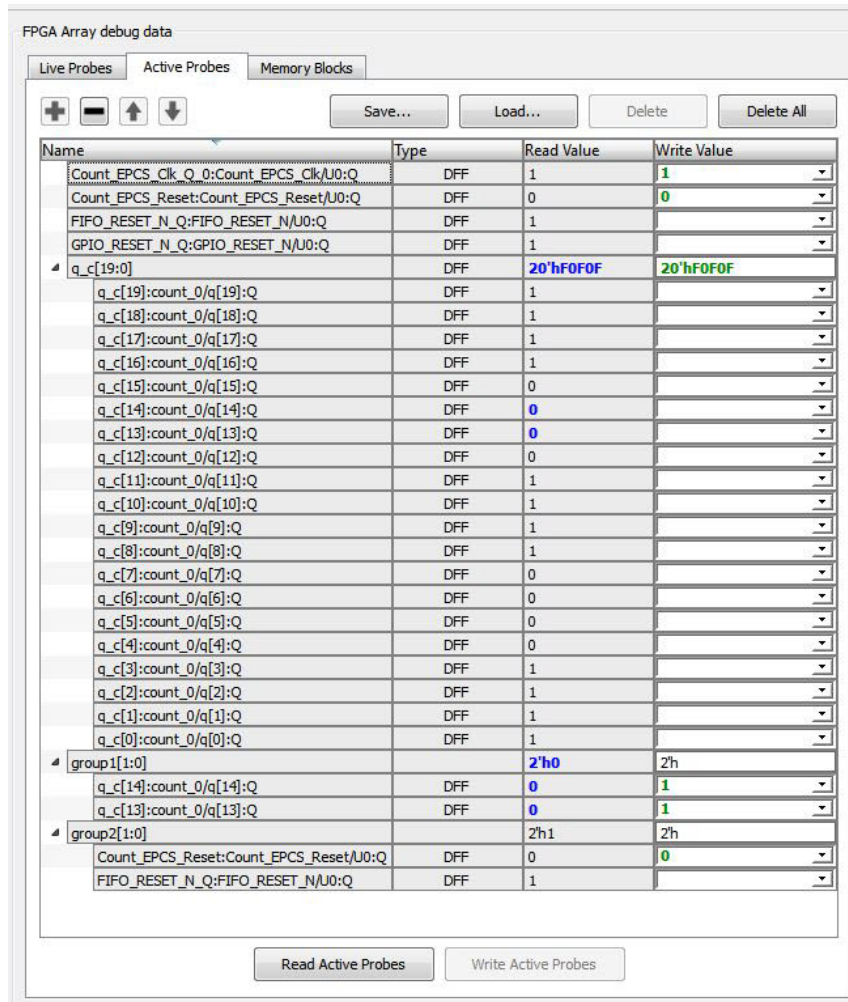


Figure 171 · Active Probes Tab

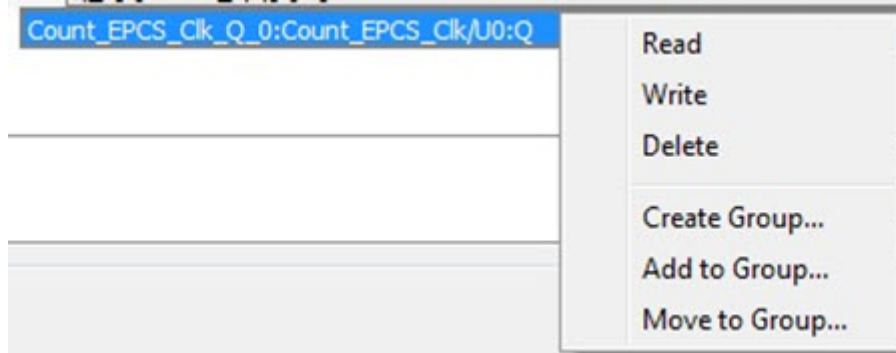
- Green entries in the “Write Value” column indicate that the operation was successful.
- Blue entries in the “Read Value” column show indicate values that have changed since the last read.

Context Menu of Probe Points Added to the Active Probes UI

When you right-click a signal or bus, you will see the following menu options:

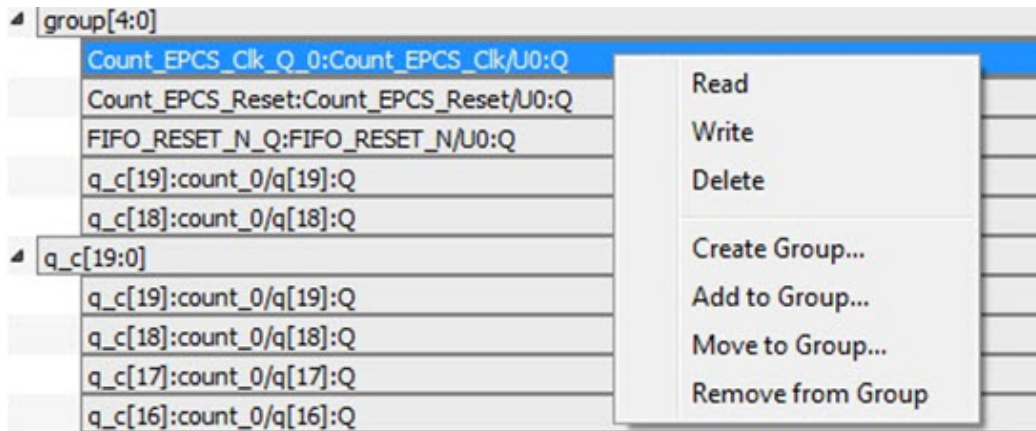
For individual signals that are not part of a probe group or bus:

- Read
- Write
- Delete
- Create Group
- Add to Group
- Move to Group



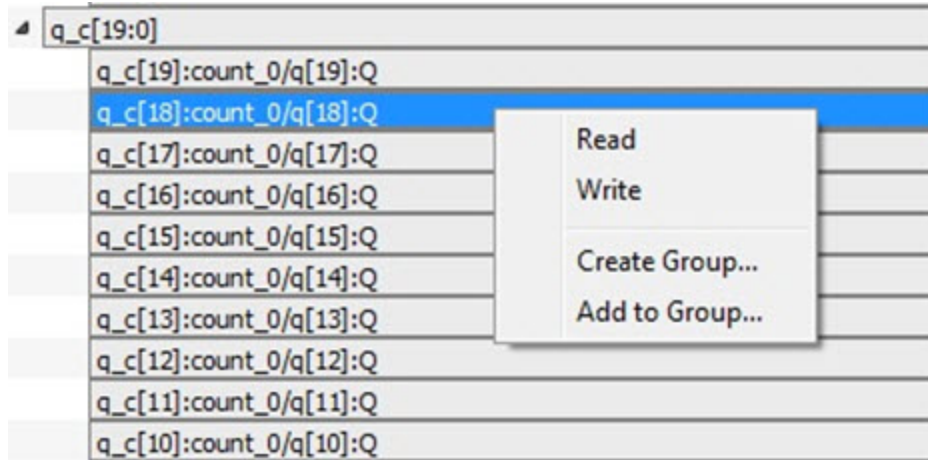
For individual signals in a probe group:

- Read
- Write
- Delete
- Create Group
- Add to Group
- Move to Group
- Remove from Group



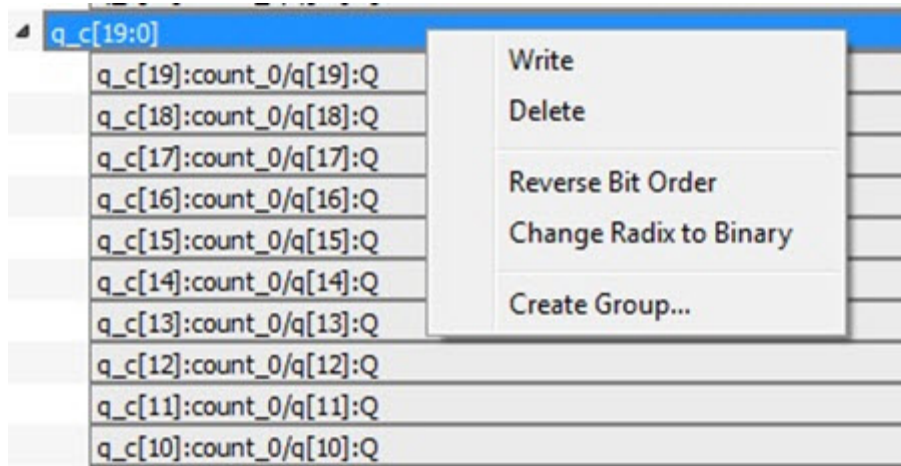
For individual signals in a bus:

- Read
- Write
- Create Group
- Add to Group



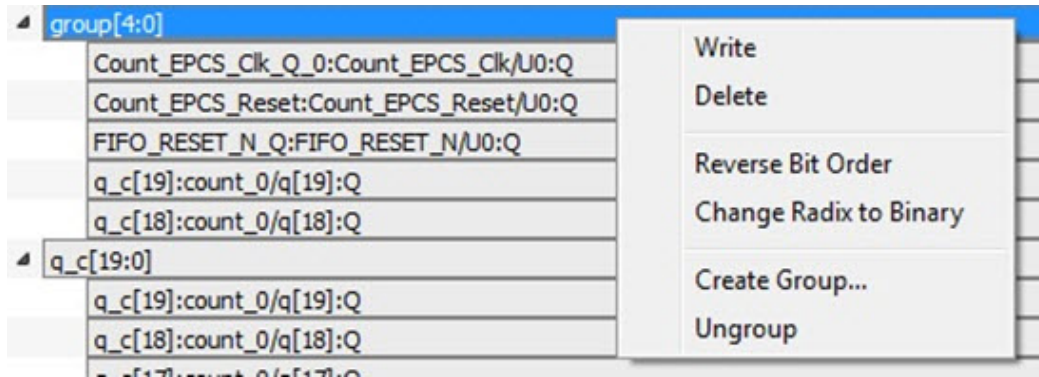
For a bus:

- Read
- Write
- Delete
- Reverse Bit Order
- Change Radix to Hex/Binary
- Create Group



For a probe group:

- Write
- Delete
- Reverse Bit Order
- Change Radix to Hex/Binary
- Create Group
- Ungroup



Differences Between a Bus and a Probe Group

A bus is created automatically by grouping selected probe nets with the same name into a bus. A bus *cannot* be ungrouped.

A Probe Group is a custom group created by adding a group of signals in the Active Probe tab into the group. The members of a Probe Group are not associated by their names. A Probe Group *can* be ungrouped.

In addition, the certain operations are also restricted to the member of a bus, whereas they are allowed in a probe group.

The following operations are not allowed in a bus:

- **Delete:** Deleting an individual signal in a bus
- **Move to Group:** Moving a signal to a probe group
- **Remove from Group:** Removing a signal from a probe group

Memory Blocks (SmartFusion2, IGLOO2, and RTG4)

The left pane in the Memory Blocks tab shows the list of defined memory blocks that are specified in your design. Select a memory block and add it to the UI. After you select a memory block, you can click **Read Block** to show the current content of the memory or write to individual memory locations. Each field is editable, and multiple memory locations can be written at the same time.

Each field is a 9-bit memory word (for SmartFusion2 and IGLOO2 devices) and 9-bit or 12-bit memory word (for RTG4 devices, depending on the memory configuration). Valid inputs are hexadecimal values between 0x0 and 0x1FF (for SmartFusion2, IGLOO2, and RTG4 devices in 9-bit mode) and 0x0 and 0xFFFF (for RTG4 devices only in 12-bit mode), as shown in the figure below.

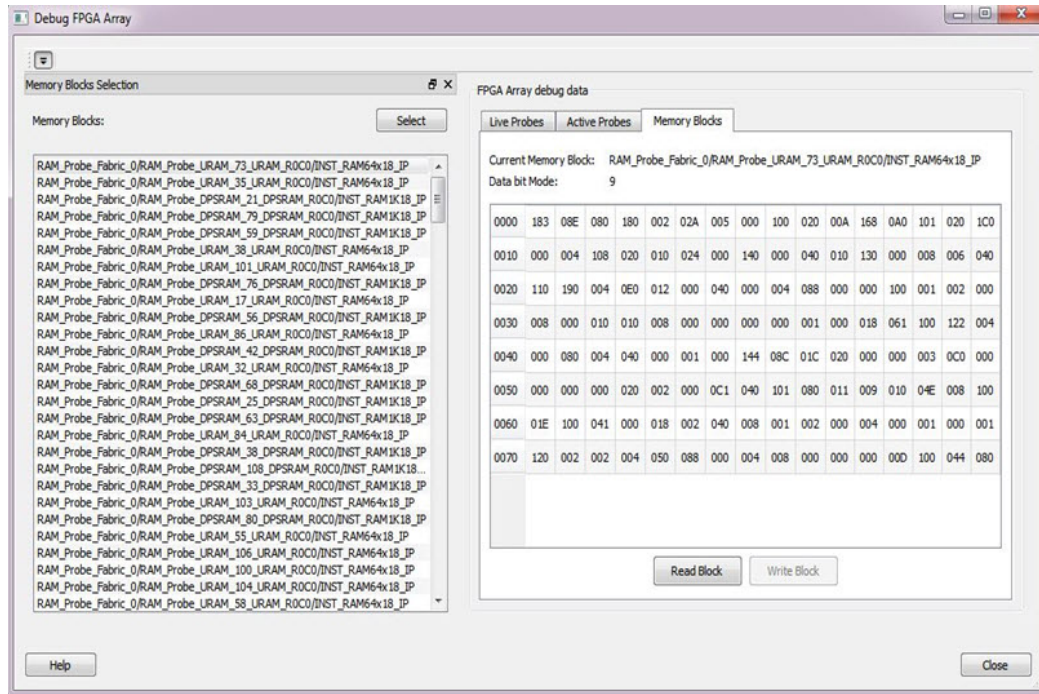


Figure 172 · Debug FPGA Array - Memory Blocks Tab

Fields that have changed but have not yet been written appear in red text until you click **Write Block** to initiate a memory write (as shown in the figure below).

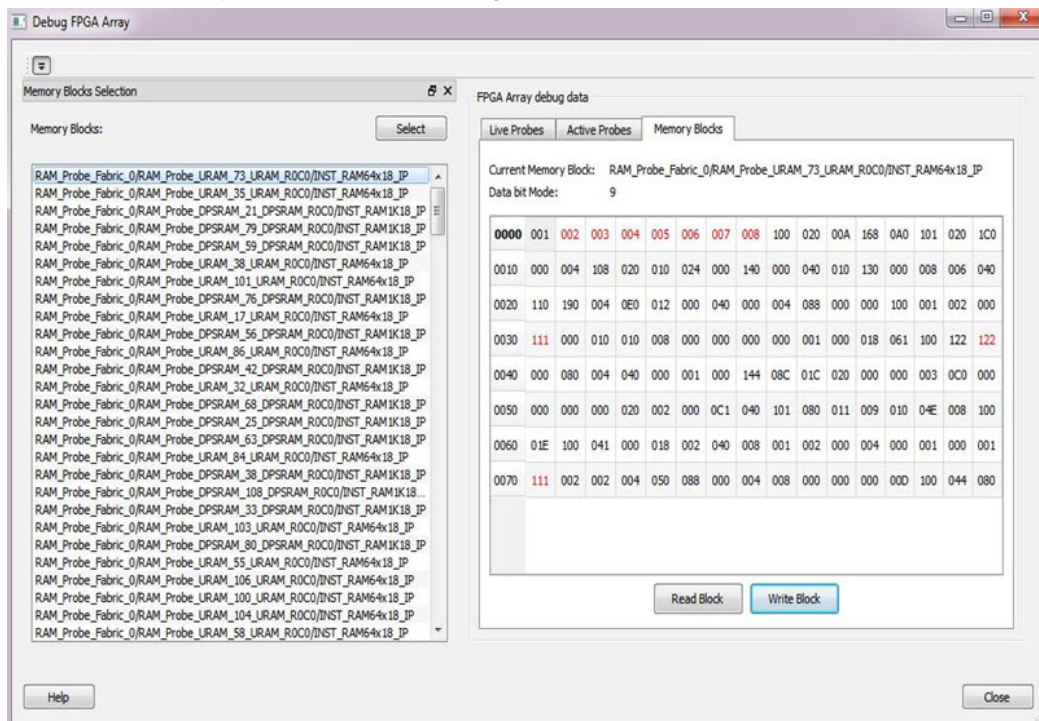


Figure 173 · Memory Blocks with Updated Values

Probe Insertion (Post-Layout) - SmartFusion2, IGLOO2, and RTG4

Introduction

Probe insertion is a post-layout debug process that enables internal nets in the FPGA design to be routed to unused I/Os. Nets are selected and assigned to probes using the Probe Insertion window in SmartDebug. The rerouted design can then be programmed into the FPGA, where an external logic analyzer or oscilloscope can be used to view the activity of the probed signal.

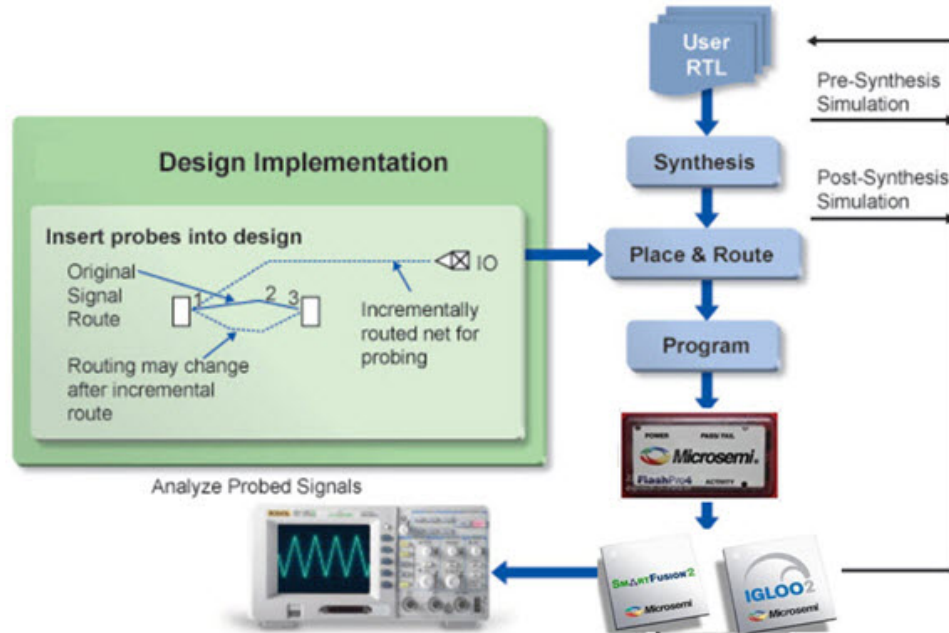


Figure 174 · Probe Insertion in the Design Process

The Probe Insertion debug feature is complementary to Live Probes and Active Probes. Live Probes and Active Probes use a special dedicated probe circuitry.

Probe Insertion

1. Double-click **SmartDebug Design** in the Design Flow window to open the SmartDebug main window.
Note: FlashPro Programmer must be connected for SmartDebug.
2. Select **Debug FPGA Array** and then select the Probe Insertion tab.

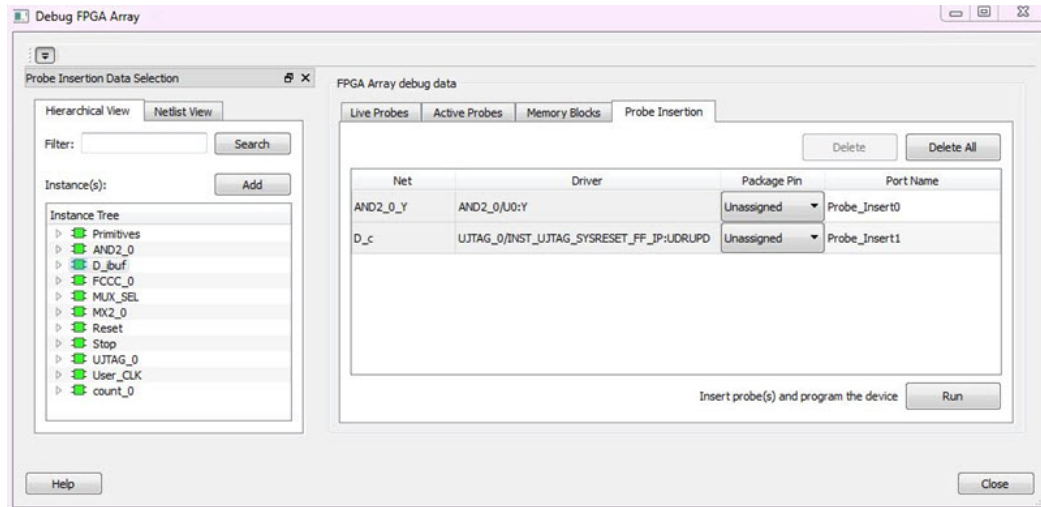


Figure 175 · Probe Insertion Tab

In the left pane of the Probe Insertion tab, all available Probe Points are listed in instance level hierarchy in the Hierarchical View. All Probe Names are shown with the Name and Type in the Netlist View.

3. Select probe points from the Hierarchical View or Netlist View, right-click and choose **Add** to add them to the Active Probes UI. You can also add the selected probe points by clicking the **Add** button. The probes list can be filtered by using the Filter box.

Each entry has a Net and Driver name which identifies that probe point.

The selected net(s) appear in the Probes table in the Probe Insertion tab, as shown in the figure below. SmartDebug automatically generates the Port Name for the probe. You can change the Port Name from the default if desired.

4. Assign a package pin to the probe using the drop-down list in the Package Pin column. You can assign the probe to any unused package pin (spare I/O).

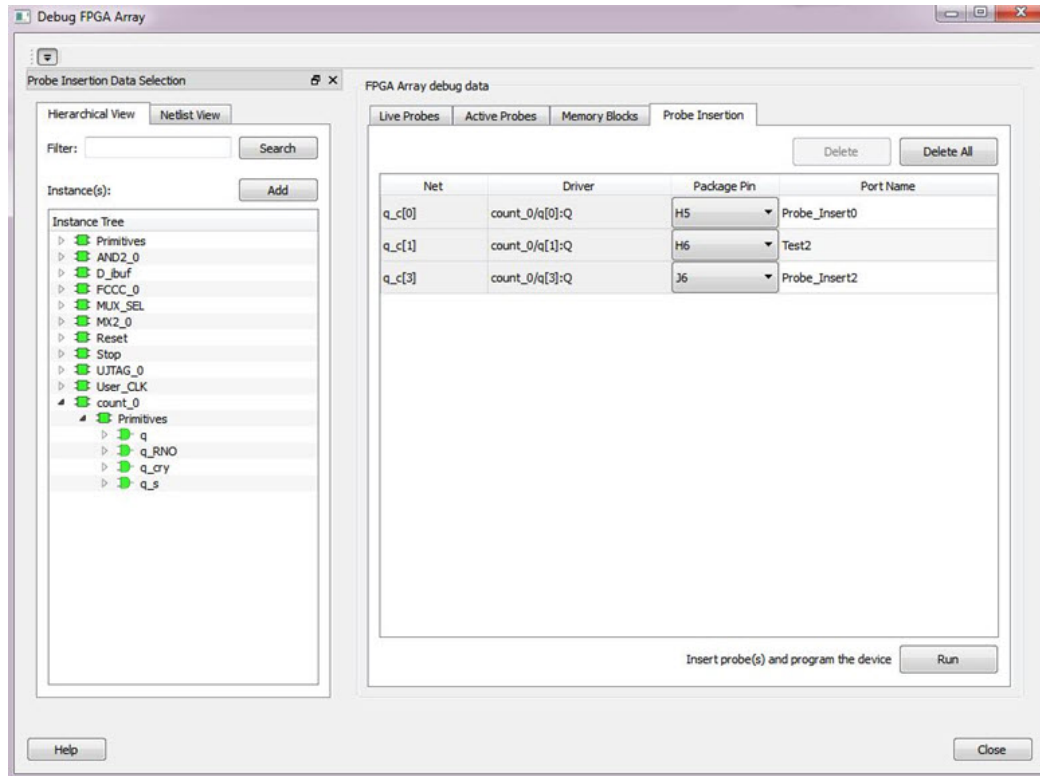


Figure 176 · Debug FPGA Array > Probe Insertion > Add Probe

5. Click **Run**.

This triggers Place and Route in incremental mode, and the selected probe nets are routed to the selected package pin. After incremental Place and Route, Libero automatically reprograms the device with the added probes.

The log window shows the status of the Probe Insertion run.

Probe Deletion

To delete a probe, select the probe and click **Delete**. To delete all the probes, click **Delete All**.

Note: Deleting probes from the probes list without clicking **Run** does not automatically remove the probes from the design.

Reverting to the Original Design

To revert to the original design after you have finished debugging:

1. In SmartDebug, click **Delete All** to delete all probes.
2. Click **Run**.
3. Wait until the action has completed by monitoring the activity indicator (spinning blue circle). Action is completed when the activity indicator disappears.
4. Close SmartDebug.

Debug SERDES (SmartFusion2, IGLOO2, and RTG4)

You can examine and debug the SERDES blocks in your design in the Debug SERDES dialog box (shown in the figure below).

To Debug SERDES, expand **SmartDebug** in the Design Flow window and double-click **Debug SERDES**.

Debug SERDES Configuration is explained below. See the [PRBS Test](#) and [Loopback Test](#) topics for information specific to those procedures.

SERDES Block identifies which SERDES block you are configuring. Use the drop-down menu to select from the list of SERDES blocks in your design.

Debug SERDES - Configuration

Configuration Report

The Configuration Report output depends on the options you select in your [PRBS Test](#) and [Loopback Tests](#). The default report lists the following for each Lane in your SERDES block:

Lane mode - Indicates the programmed mode on a SERDES lane as defined by the SERDES system register.

PMA Ready - Indicates whether PMA has completed its internal calibration sequence for the specific lane and whether the PMA is operational. See the [SmartFusion2](#) or [IGLOO2](#) High Speed Serial Interfaces User Guide on the Microsemi website for details.

TxPLL status - Indicates the loss-of-lock status for the TXPLL is asserted and remains asserted until the PLL reacquires lock.

RxPLL status - Indicates the CDR PLL frequency is not grossly out of range of with incoming data stream.

Click **Refresh Report** to update the contents of your SERDES Configuration Report. Changes to the specified SERDES register programming can be read back to the report.

SERDES Register Read or Write

Script - Runs Read/Write commands to access the SERDES control/status register map using a script. Enter the full pathname for the script location or click the browse button to navigate to your script file. Click **Execute** to run the script.

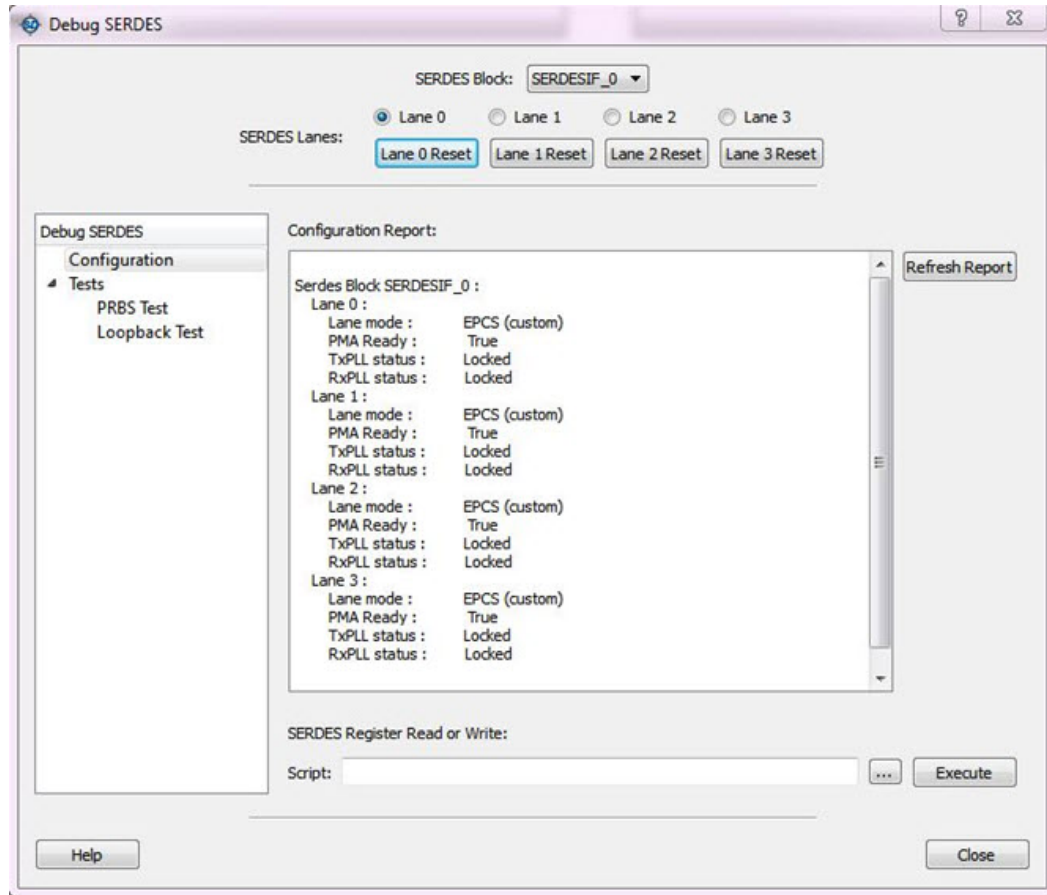


Figure 177 · Debug SERDES - Configuration

Note: The PCIe and XAUI protocols only support PRBS7. The EPCS protocol supports PRBS7/11/23/31.

Debug SERDES – Loopback Test

Loopback data stream patterns are generated and checked by the internal SERDES block. These are used to self-test signal integrity of the device. You can switch the device through predefined tests.

See the [PRBS Test topic](#) for more information about the PRBS test options.

SERDES Block identifies which SERDES block you are configuring. Use the drop-down menu to select from the list of SERDES blocks in your design.

SERDES Lanes

Select the **Lane** and **Lane Status** on which to run the Loopback test. Lane mode indicates the programmed mode on a SERDES lane as defined by the SERDES system register.

Test Type

PCS Far End PMA RX to TX Loopback- This loopback brings data into the device and deserializes and serializes the data before sending it off-chip. This loopback requires 0PPM clock variation between the TX and RX SERDES clocks.

See the [SmartFusion2](#) or [IGLOO2](#) High Speed Serial Interfaces User's Guide on the Microsemi website for details.

Near End Loopback (On Die) - To enable, select the Near End Loopback (On Die) option and click **Start**. Click **Stop** to disable. Using this option allows you to send and receive user data without sending traffic off-chip. You can test design functionality without introducing other issues on the PCB.

See the [SmartFusion2](#) or [IGLOO2](#) High Speed Serial Interfaces User's Guide on the Microsemi website for details.

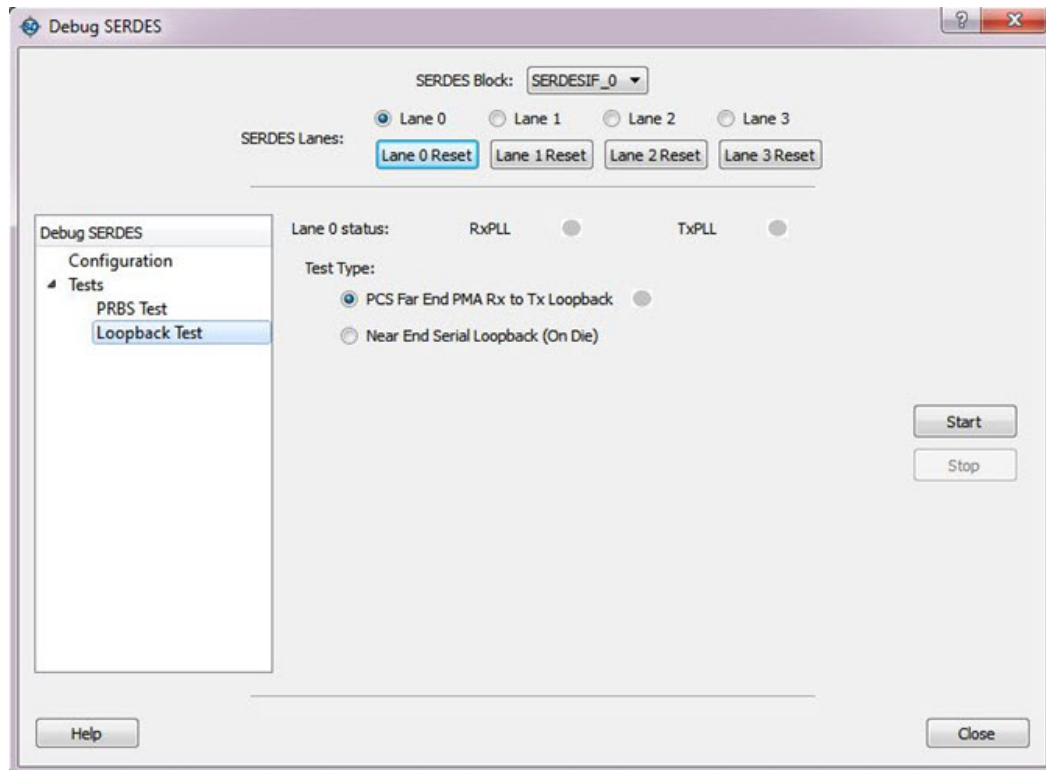


Figure 178 · Debug SERDES - Loopback Test

Debug SERDES – PRBS Test

PRBS data stream patterns are generated and checked by the internal SERDES block. These are used to self-test signal integrity of the device. You can switch the device through several predefined patterns.

View Loopback Test settings in the [Debug SERDES - Loopback Test topic](#).

SERDES Block identifies which SERDES block you are configuring. Use the drop-down menu to select from the list of SERDES blocks in your design.

SERDES Lanes

Select the **Lane** and **Lane Status** on which to run the PRBS test. Lane mode indicates the programmed mode on a SERDES lane as defined by the SERDES system register.

Test Type

Near End Serial Loopback (On-Die) enables a self test of the device. The serial data stream is sent from the SERDES TX output and folded back onto the SERDES RX input.

Serial Data (Off-Die) is the normal system operation where the data stream is sent off chip from the TX output and must be connected to the RX input via a cable or other type of electrical interconnection.

Pattern

The SERDESIF includes an embedded test pattern generator and checker used to perform serial diagnostics on the serial channel, as shown in the table below.

Pattern	Type
PRBS7	Pseudo-Random data stream of 2 ⁷ polynomial sequences
PRBS11	Pseudo-Random data stream of 2 ¹¹ polynomial sequences
PRBS23	Pseudo-Random data stream of 2 ²³ polynomial sequences
PRBS31	Pseudo-Random data stream of 2 ³¹ polynomial sequences

Cumulative Error Count

Lists the number of cumulative errors after running your PRBS test. Click **Reset** to reset to zero. 1. By default, Cumulative Error Count = 0, the Data Rate text box is blank, and Bit Error Rate = NA.

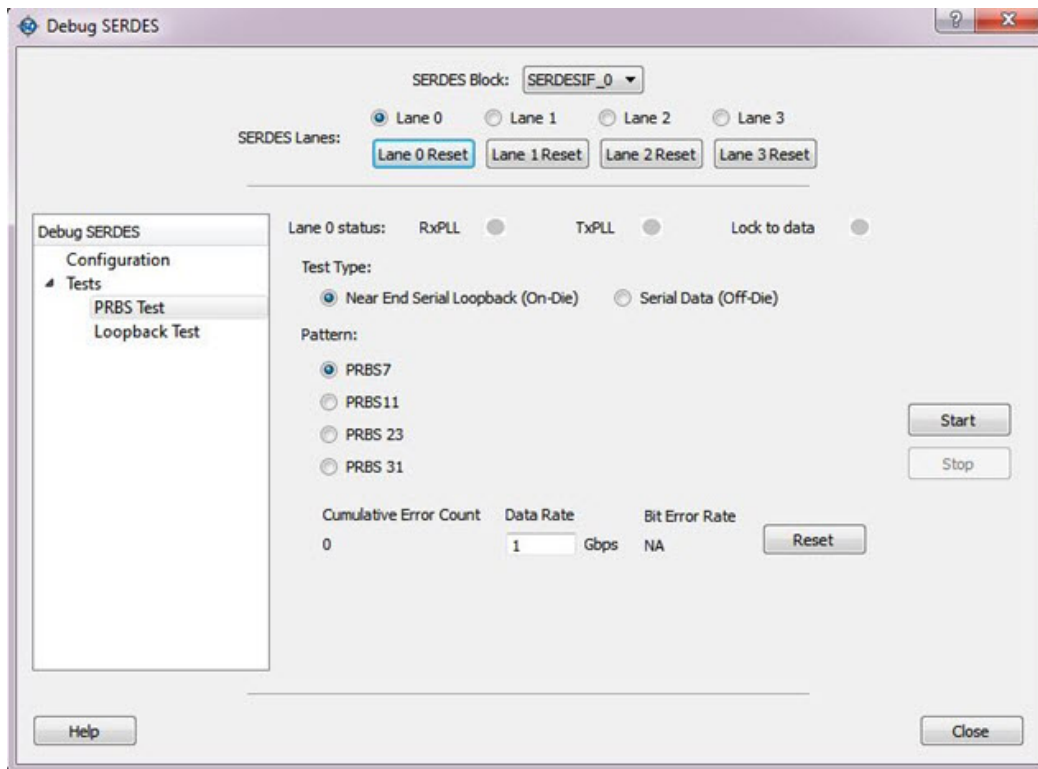


Figure 179 · Debug SERDES - PRBS Test

Note: If the design uses SERDES PCIe, PRBS7 will be the only option shown for PRBS tests..

Bit Error Rate

Displays the Bit Error Rate (BER) for the PRBS test in progress.

The formula for calculating the BER is as follows:

$$\text{BER} = (\text{\#bit errors} + 1) / \text{\#bits sent}$$

#bits sent = Elapsed time/bit period

When clicked on Start:

- The BER is updated every second for the entered data rate and errors observed.
- If no data rate is entered by the user, the BER is set to the default.

When clicked on Stop:

- The BER resets to default.

When clicked on Reset:

- The BER is reset to default.
- If no test is in progress, the BER remains in the default value.
- If the PRBS test is in progress, the BER calculation restarts.

Debug SERDES – PHY Reset

SERDES PMA registers (for example, TX_AMP_RATIO) modified using a TCL script from the Configuration tab require a soft reset for the new values to be updated. Lane Reset for individual lanes achieves this functionality. Depending on the SERDES lanes used in the design, the corresponding Lane Reset buttons are enabled.

Lane Reset Behavior for SERDES Protocols Used in the Design

- EPCS: Reset is independent for individual lanes. Reset to Lane X (where X = 0,1,2,3) resets the Xth lane.
- PCIe: Reset to Lane X (where X = 0,1,2,3) resets all lanes present in the PCIe link and PCIe controller.

For more information about soft reset, refer to the [SmartFusion2 and IGLOO2 High Speed Serial Interfaces User Guide](#).

SmartDebug Tcl Support (SmartFusion2, IGLOO2, and RTG4)

The following table lists the Tcl commands related to SmartDebug for SmartFusion2, IGLOO2, and RTG4. Click the command to view more information.

Table 6 · SmartDebug Tcl Commands

Command	Action
DDR/MDDR	
ddr_read	Reads the value of specified configuration registers pertaining to the DDR memory controller (MDDR/FDDR)
ddr_write	Writes the value of specified configuration registers pertaining to the DDR memory controller (MDDR/FDDR)
Probe	
add_to_probe_group	Adds the specified probe points to the specified probe group
create_probe_group	Creates a new probe group
delete_active_probe	Deletes either all or the selected active probes.
load_active_probe_list	Loads the list of probes from the file.
move_to_probe_group	Moves the specified probe points to the specified probe group.

Command	Action
DDR/MDDR	
set_live_probe	Set Live probe channels A and/or B to the specified probe point (or points).
select_active_probe	Manages the current selection of active probe points to be used by active probe READ operations.
read_active_probe	Reads active probe values from the device.
remove_from_probe_group	Move out the specified probe points from the group.
save_active_probe_list	Saves the list of active probes to a file.
select_active_probe	Manages the current selection of active probe points to be used by active probe READ operations.
ungroup	Disassociates the probes as group.
unset_live_probe	Discontinues the debug function and clears live probe channels.
write_active_probe	Sets the target probe point on the device to the specified value.
LSRAM	
read_lsram	Reads a specified block of large SRAM from the device.
write_lsram	Writes a seven bit word into the specified large SRAM location.
uSRAM	
read_usram	Reads a uSRAM block from the device.
write_usram	Writes a seven bit word into the specified uSRAM location.
SERDES	
prbs_test	Starts, stops, resets the error counter and reads the error counter value in PRBS tests.
loopback_test	Starts and stops the loopback tests.
serdes_lane_reset	In EPCS mode, this command resets the lane. In PCI mode, this command resets the lane, all other lanes in the link, and the corresponding PCIe controller.
serdes_read_register	Reads the SERDES register value and displays the result in the log window/console.
serdes_write_register	Writes the value to the SERDES register.
Additional Commands	

Command	Action
DDR/MDDR	
export_smart_debug_data	Exports debug data for the SmartDebug application.

add_to_probe_group (SmartFusion2, IGLOO2, and RTG4)

Tcl command; adds the specified probe points to the specified probe group.

```
add_to_probe_group -name probe_name -group group_name
```

Arguments

- name *probe_name*
Specifies one or more probes to add.
- group *group_name*
Specifies name of the probe group.

Supported Families

SmartFusion2, IGLOO2, and RTG4

Example

```
add_to_probe_group -name out[5]:out[5]:Q \  
                  -name grp1.out[3]:out[3]:Q \  
                  -name out.out[1].out[1]:Q \  
                  -group my_new_grp
```

compare_flashrom_client

Compares the content of the FlashROM configurations in your design against the actual values in the selected device.

```
compare_flashrom_client [-name {device_name}] [-file {filename}]
```

Arguments

- name {*device_name*}
Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the [set_debug_device](#) command.
- file {*filename*}
Optional file name for FlashROM compare log.

Supported Families

SmartFusion, IGLOO, ProASIC3 and Fusion

Exceptions

None

Example

The following command saves the FlashROM data to the file 'FlashRomCompReport.log':

```
compare_flashrom_client -file {FlashRomCompReport.log}
```

The following command compares the data in the device 'A3P250' and saves the data in the logfile 'FlashRomCompReport.log':

```
compare_flashrom_client -name {A3P250} -file {FlashRomCompReport.log}
```

compare_memory_client

Compares the memory client in a specific device and block.

```
compare_memory_client [-name {device_name}] [-block integer_value] -client {client_name} [-file {filename}]
```

Arguments

-name { *device_name* }

Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the [set_debug_device](#) command.

-block { *integer_value* }

(Optional argument; you must set -client.) Specifies location of block for memory compare.

-client { *client_name* }

Name of client for memory compare.

-file { *filename* }

Optional file name.

Supported Families

SmartFusion and Fusion

Exceptions

None

Example

The following command compares the memory in the client 'DS32' on the device 'AFS600':

```
compare_memory_client -client DS32 -name AFS600
```

The following command compares the data at block '0' to the client 'DS8bit':

```
compare_memory_client -block 0 -client {DS8bit}
```

The following command compares the memory in the device 'AFS600' at block '0' to the memory client 'DS8bit':

```
compare_memory_client -name {AFS600} -block 0 -client {DS8bit}
```

The following command compares the memory at block '1' to the memory client 'DS8bit' and saves the information in a log file to F:/tmp/NVMCompReport.log:

```
compare_memory_client -block 1 -client {DS8bit} -file {F:/tmp/NVMCompReport.log}
```

create_probe_group (SmartFusion2, IGLOO2, and RTG4)

Tcl command; creates a new probe group.

```
create_probe_group -name group_name
```

Arguments

-name *group_name*

Specifies the name of the new probe group.

Supported Families

SmartFusion2, IGLOO2, and RTG4

Example

```
create_probe_group -name my_new_grp
```

delete_active_probe

Tcl command; deletes either all or the selected active probes.

Note: You cannot delete an individual probe from the Probe Bus.

```
Delete_active_probe -all | -name probe_name
```

Arguments

-all

Deletes all active probe names.

-name *probe_name*

Deletes the selected probe names.

Supported Families

SmartFusion2, IGLOO2, and RTG4

Example

```
Delete -all          <- deletes all active probe names
Delete -name out[5]:out[5]:Q \
      -name my_grp1.out[1]:out[1]:Q          <- deletes the selected probe names
Delete -name my_grp1 \
      -name my_bus          <- deletes the group, bus and their members.
```

ddr_read (SmartFusion2, IGLOO2, and RTG4)

Tcl command; reads the value of specified configuration registers pertaining to the DDR memory controller (MDDR/FDDR).

```
ddr_read -block ddr_name -name reg_name
```

Arguments

`-block <fddr || mddr || east_fddr || west_fddr>`

- Specifies which DDR configurator is used in the Libero design.
- SmartFusion2 and IGLOO2 - fddr and mddr
- RTG4 - east_fddr and west_fddr

`-name register_name`

- Specifies which configuration registers need to be read.
- A complete list of registers is available in the DDR Interfaces User Guides for the respective families.

Supported Families

SmartFusion2, IGLOO2, and RTG4

Example

Read DDR Controller register `DDRC_DYN_REFRESH_1_CR` for a configured FDDR block on a SmartFusion2 or IGLOO2 device:

```
ddr_read -block fddr -name DDRC_DYN_REFRESH_1_CR
```

Returns

Returns 16-bit hexadecimal value.

The result of the command in the example above will be:

```
Register Name: DDRC_DYN_REFRESH_1_CR Value: 0x1234  
"ddr_read" command succeeded.
```

ddr_write (SmartFusion2, IGLOO2, and RTG4)

Tcl command; writes the value of specified configuration registers pertaining to the DDR memory controller (MDDR/FDDR).

```
ddr_write-block ddr_name -name reg_name -value hex_value
```

Arguments

`-block <fddr || mddr || east_fddr || west_fddr>`

- Specifies which DDR configurator is used in the Libero design.
- SmartFusion2 and IGLOO2 - fddr and mddr
- RTG4 - east_fddr and west_fddr

`-name register_name`

- Specifies which configuration registers need to be read.
- A complete list of registers is available in the DDR Interfaces User Guides for the respective families.

`-value hex_value`

- Specifies the value to be written into the specified register of a given block.
- Hex_value in the form of "0x12FA".

Supported Families

SmartFusion2, IGLOO2, and RTG4

Example

Write a 16-bit value DDR Controller register DDRC_DYN_REFRESH_1_CR for a configured FDDR block on a SmartFusion2 or IGLOO2 device:

```
ddr_write -block fddr -name DDRC_DYN_REFRESH_1_CR -value 0x123f
```

Returns

Returns if the command succeeded or failed to execute.

```
"ddr_write" command succeeded
```

export_smart_debug_data (SmartFusion2, IGLOO2, and RTG4)

Tcl command; exports debug data for the SmartDebug application.

```
export_smart_debug_data [device_components] [bitstream_components] [-file_name {file} [-  
export_dir {dir}]
```

The command corresponds to the Export SmartDebug Data tool in Libero. The command creates a file with the extension "ddc" that contains data based on selected options. This file is used by SmartDebug to create a new SmartDebug project, or it can be imported into a device in SmartDebug.

- If you not specify any design components, all components available in the design will be included by default.
- The generate_bitstream parameter is required if you want to generate bitstream file and include it in the exported file.
 - o You must specify the bitstream components you want to include in the generated bitstream file or all available components will be included.
 - o If you choose to include bitstream, and the design has custom security, the custom security bitstream component must be included.

Arguments

device_components

The following device components can be selected. Specify "1" to include the component, and "0" if you do not want to include the component.

```
-probes <1|0>  
-package_pins <1|0>  
-memory_blocks <1|0>  
-envm_data <1|0>  
-security_data <1|0>  
-chain <1|0>  
-programmer_settings <1|0>  
-io_states <1|0>
```

bitstream_components

The following bitstream components can be selected. Specify "1" to include the component, and "0" if you do not want to include the component.

```
-generate_bitstream <1|0>  
-bitstream_security <1|0>  
-bitstream_fabric <1|0>
```

```
-bitstream_envm <1|0>
-file_name file
Name of exported file with extension "ddc".
-export_dir dir
Location where DDC file will be exported. If omitted, design export folder will be used.
```

Supported Families

SmartFusion2, IGLOO2, and RTG4

Example

The following example shows the `export_smart_debug_data` command with all parameters:

```
export_smart_debug_data \
-file_name {sd1} \
-export_dir {d:\sd_prj\test3T\designer\sd1\export} \
-probes 1 \
-package_pins 0 \
-memory_blocks 1 \
-envm_data 0 \
-security_data 1 \
-chain 1 \
-programmer_settings 1 \
-ios_states 1 \
-generate_bitstream 0 \
-bitstream_security 0 \
-bitstream_fabric 0 \
-bitstream_envm 0
```

The following example shows the command with no parameters:

```
export_smart_debug_data
```

load_active_probe_list

Tcl command; loads the list of probes from the file.

```
load_active_probe_list -file file_path
```

Arguments

```
-file file_path
The input file location.
```

Supported Families

SmartFusion2, IGLOO2, and RTG4

Example

```
load_active_probe_list -file "./my_probes.txt"
```

loopback_test (SmartFusion2, IGLOO2, RTG4)

Tcl command; used to start and stop the loopback tests.

```
loopback_test [-deviceName device_name] -start -serdes num -lane num -type LoopbackType  
loopback_test [-deviceName device_name] -stop -serdes num -lane num
```

Arguments

-deviceName *device_name*

Parameter is optional if only one device is available in the current configuration or set for debug (see the SmartDebug User's Guide for details).

-start

Starts the loopback test.

-stop

Stops the loopback test.

-serdes *num*

Serdes block number. Must be between 0 and 4 and varies between dies.

-lane *num*

Serdes lane number. Must be between 0 and 4

-type *LoopbackType*

Specifies the loopback test type. Must be *meso* (PCS Far End PMA RX to TX Loopback)

Supported Families

SmartFusion2, IGLOO2, RTG4

Example

```
loopback_test -start -serdes 1 -lane 1 -type meso  
loopback_test -start -serdes 0 -lane 0 -type plesio  
loopback_test -start -serdes 1 -lane 2 -type parallel  
loopback_test -stop -serdes 1 -lane 2
```

move_to_probe_group (SmartFusion2, IGLOO2, and RTG4)

Tcl command; moves the specified probe points to the specified probe group.

Note: Probe points related to a bus cannot be moved to another group.

```
move_to_probe_group -name probe_name -group group_name
```

Arguments

-name *probe_name*

Specifies one or more probes to move.

-group *group_name*

Specifies name of the probe group.

Supported Families

SmartFusion2, IGLOO2, and RTG4

Example

```
move_to_probe_group -name out[5]:out[5]:Q \  
                    -name grp1.out[3]:out[3]:Q \  
                    -group my_grp2
```

prbs_test (SmartFusion2, IGLOO2, RTG4)

Tcl command; used in PRBS test to start, stop, reset the error counter and read the error counter value.

```
prbs_test [-deviceName device_name] -start -serdes num -lane num [-near] -pattern PatternType  
prbs_test [-deviceName device_name] -stop -serdes num -lane num  
prbs_test [-deviceName device_name] -reset_counter -serdes num -lane num  
prbs_test [-deviceName device_name] -read_counter -serdes num -lane num
```

Arguments

-deviceName *device_name*

Parameter is optional if only one device is available in the current configuration or set for debug (see the SmartDebug User's Guide for details).

-start

Starts the prbs test.

-stop

Stops the prbs test.

-reset_counter

Resets the prbs error count value to 0.

-read_counter

Reads and prints the error count value.

-serdes *num*

Serdes block number. Must be between 0 and 4 and varies between dies.

-lane *num*

Serdes lane number. Must be between 0 and 4.

-near

Corresponds to near-end (on-die) option for prbs test. Not specifying implies off-die.

-pattern *PatternType*

The pattern sequence to use for PRBS test. It can be one of the following:

prbs7, *prbs11*, *prbs23*, or *prbs31*

Supported Families

SmartFusion2, IGLOO2, RTG4

Example

```
prbs_test -start -serdes 1 -lane 0 -near -pattern prbs11  
prbs_test -start -serdes 2 -lane 2 -pattern custom -value all_zeros
```

```
prbs_test -start -serdes 0 -lane 1 -near -pattern user -value 0x0123456789ABCDEF0123
```

read_active_probe (SmartFusion2, IGLOO2, and RTG4)

Tcl command; reads active probe values from the device. The target probe points are selected by the [select_active_probe](#) command.

```
read_active_probe [-deviceName device_name] [-name probe_name] [-file file_path]
```

Arguments

-deviceName *device_name*

Parameter is optional if only one device is available in the current configuration.

-name *probe_name*

Instead of all probes, read only the probes specified. The probe name should be prefixed with bus or group name if the probe is in the bus or group.

-file *file_path*

Optional. If specified, redirects output with probe point values read from the device to the specified file.

Note: When the user tries to read at least one signal from the bus/group, the complete bus or group is read. The user is presented with the latest value for all the signals in the bus/group.

Supported Families

SmartFusion2, IGLOO2, and RTG4

Example

```
Read_active_probe -file output.txt
```

```
Read_active_probe -name grp1.out[3]:out[3]:Q -file output.txt
```

read_device_status

Displays the Device Information report; the Device Information report is a complete summary of your device state, analog block test values, user information, factory serial number and security information..

```
read_device_status [-name {device_name}] [-file {filename}]
```

Arguments

-name *device_name*

Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the [set_debug_device](#) command.

-file {*filename*}

(Optional) Identifies the name of the file to which read results will be saved.

Supported Families

SmartFusion, IGLOO, ProASIC3 and Fusion

Exceptions

None

Example

The following reads device info from the 'AFS600' device.

```
read_device_status -name AFS600
```

read_id_code

The command reads IDCode from the device without masking any IDCode fields. This is the raw IDcode from the silicon.

Note: Being able to read the IDCode is an indication that the JTAG interface is working correctly.

```
read_id_code [-name {device_name}]
```

Arguments

-name *device_name*

Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the [set_debug_device](#) command.

Supported Families

SmartFusion, IGLOO, ProASIC3 and Fusion

Exceptions

None

Example

The following command reads the IDCODE from the device 'AFS600':

```
read_id_code -name {AFS600}
```

read_flashrom

Reads the content of the FlashROM from the selected device.

```
read_flashrom [-name {device_name}] [-mapping {logical / physical}] [-file {filename}]
```

Arguments

-name *device_name*

Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the [set_debug_device](#) command.

-mapping {*logical* / *physical*}

(Optional) Specifies how the data read from the UFROM is mapped. Values are explained in the table below.

Value	Description
logical	Logical mapping (default)
physical	Physical mapping

`-file {filename}`

(Optional) Identifies the name of the file to which read results will be saved.

Supported Families

SmartFusion, IGLOO, ProASIC3 and Fusion

Exceptions

None

Example

The following reads the FROM content on the device 'AFS600' and sets to physical mapping:

```
read_flashrom -name {AFS600} -mapping {physical}
```

read_flash_memory

The command reads information from the NVM modules. There are two types of information that can be read:

- Page Status – includes ECC2 status, write count, access protection
- Page Data

```
read_flash_memory
[-name {device_name}]
[-block {integer_value}]
[-client {client_name}]
[-startpage {integer_value}]
[-endpage {integer_value}]
[-access {all | status | data}]
[-file {filename}]
```

At a minimum you must specify `-client <name>` OR

```
-startpage <page_number> -endpage <page_number> -block <number>
```

Arguments

`-name {device_name}`

Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the [set debug device](#) command.

`-block {integer_value}`

(Optional argument; you must set `-client` or `-startpage` and `-endpage` before use.) Specifies location of block for memory read.

`-client {client_name}`

Name of client for memory read.

`-startpage {integer_value}`

Startpage for page range; value must be an integer. You must specify a `-endpage` and `-block` along with this argument.

`-endpage` {*integer_value*}

Endpage for page range; value must be an integer. You must specify a `-startpage` and `-block` along with this argument.

`-access` {*all* | *status* | *data*}

(Optional argument; you must set `-client` or `-startpage`, `-endpage` and `-block` before use.) Specifies what eNVM information to check: page status, data or both.

Value	Description
all	Shows the number of pages with corruption status, data corruption and out-of-range write count (default)
status	Shows the number of pages with corruption status and the number of pages with out-of-range write count
data	Shows only the number of pages with data corruption

`-file` {*filename*}

(Optional argument; you must set `-client` or `-startpage`, `-endpage` and `-block` before use.) Name of output file for memory read.

Supported Families

SmartFusion, Fusion

Exceptions

None

Example

The following command reads the flash memory for the client 'DS8bit' and reports the data in a logfile 'readFlashMemoryReport.log':

```
read_flash_memory -client {DS8bit} -file {readFlashMemoryReport.log}
read_flash_memory -startpage 0 -endpage 2 -block 0 -access {data}
```

read_lsr (SmartFusion2, IGLOO2, and RTG4)

Tcl command; reads a specified block of large SRAM from the device.

```
read_lsr [-deviceName device_name] -block block_name [-file filename]
```

`-deviceName` *device_name*

Parameter is optional if only one device is available in the current configuration or set for debug (see SmartDebug help for details).

`-block` *block_name*

Specifies the name for the target block.

`-file` *filename*

Optional; specifies the output file name for the data read from the device.

Supported Families

SmartFusion2, IGLOO2, and RTG4

Exceptions

- You must set a debug file
- Array must be programmed and active
- Security locks may disable this function

Example

Reads the SRAM Block `sram_block1` from the `sf2` device and writes it to the file `sram_block_output`.

```
read_lsram [-deviceName sf2] -block sram_block1 [-file sram_block_output]
```

read_usram (SmartFusion2 and IGLOO2)

Tcl command; reads a uSRAM block from the device.

```
read_lsram [-deviceName device_name] -block block_name [-file filename]
```

-deviceName *device_name*

Parameter is optional if only one device is available in the current configuration or set for debug (see SmartDebug help for details).

-block *block_name*

Specifies the name for the target block.

-file *filename*

Optional; specifies the output file name for the data read from the device.

Supported Families

SmartFusion2, IGLOO2, and RTG4

Exceptions

- You must set a debug file
- Array must be programmed and active
- Security locks may disable this function

Example

Reads the uSRAM Block `usram_block2` from the `sf2` device and writes it to the file `sram_block_output`.

```
read_usram [-deviceName sf2] -block usram_block2 [-file sram_block_output]
```

recover_flash_memory

The command removes ECC2 errors due to memory corruption by reprogramming specified flash memory (NVM) pages and initializing all pages to zeros. The recovery affects data blocks and auxiliary blocks.

The write counters of the corrupted pages might not be accurate due to corruption. The recovery operation will not change state of the page write counters.

Use the `check_flash_memory` command to detect flash memory errors.

```
recover_flash_memory
[-name {device_name}]
[-block {integer_value}]
[-client {client_name}]
[-startpage {integer_value}]
[-endpage {integer_value}]
```

At a minimum you must specify `-client <name>` OR
`-startpage <page_number> -endpage <page_number> -block <number>`

Arguments

`-name {device_name}`

Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the [set_debug_device](#) command.

`-block {integer_value}`

(Optional argument; you must set `-client` or `-startpage` and `-endpage` before use.) Specifies location of block for memory recovery.

`-client {client_name}`

Name of client for memory recovery.

`-startpage {integer_value}`

Startpage for page range; value must be an integer. You must specify a `-endpage` and `-block` along with this argument.

`-endpage {integer_value}`

Endpage for page range; value must be an integer. You must specify a `-startpage` and `-block` along with this argument.

Supported Families

SmartFusion, Fusion

Exceptions

None

Example

The following command recovers flash memory data in the client 'DS8bit':

```
recover_flash_memory -client {DS8bit}
```

The following command recovers flash memory from block 0, startpage 0, and endpage 3:

```
recover_flash_memory -block 0 -startpage 0 -endpage 3
```

remove_from_probe_group (SmartFusion2, IGLOO2, and RTG4)

Tcl command; moves out the specified probe points from the group. That is, the moved out probe points won't be associated with any probe group.

Note: Probes cannot be moved out from the bus.

```
remove_from_probe_group -name probe_name
```

Arguments

-name *probe_name*
Specifies one or more probe points to move out.

Supported Families

SmartFusion2, IGLOO2, and RTG4

Example

The following command moves out two probes from my_grp2.

```
Move_out_of_probe_group -name my_grp2.out[3]:out[3]:Q \  
                        -name my_grp2.out[3]:out[3]:Q
```

save_active_probe_list

Tcl command; saves the list of active probes to a file.

```
save_active_probe_list -file file_path
```

Arguments

-file *file_path*
The output file location.

Supported Families

SmartFusion2, IGLOO2, and RTG4

Example

```
save_active_probe_list -file "./my_probes.txt"
```

select_active_probe (SmartFusion2, IGLOO2, and RTG4)

Tcl command; manages the current selection of active probe points to be used by active probe READ operations. This command extends or replaces your current selection with the probe points found using the search pattern.

```
select_active_probe [-deviceName device_name] [-name probe_name_pattern] [-reset true/false]
```

Arguments

-deviceName *device_name*
Parameter is optional if only one device is available in the current configuration..

-name *probe_name_pattern*
Specifies the name of the probe. Optionally, search pattern string can specify one or multiple probe points. The pattern search characters "*" and "?" also can be specified to filter out the probe names.

-reset *true | false*

Optional parameter; resets all previously selected probe points. If name is not specified, empties out current selection.

Supported Families

SmartFusion2, IGLOO2, and RTG4

Example

The following command selects three probes. In the below example, “grp1” is a group and “out” is a bus..

```
Select_active_probe -name out[5]:out[5]:Q
Select_active_probe -name out.out[1]:out[1]:Q \
                    -name out.out[3]:out[3]:Q \
                    -name out.out[5]:out[5]:Q
```

serdes_lane_reset

Tcl command. In EPCS mode, this command resets the lane. In PCI mode, this command resets the lane, all other lanes in the link, and the corresponding PCIe controller. The result is shown in the log window/console.

```
serdes_lane_reset -serdes num -lane num
```

Arguments

-serdes *num*

The SERDES block number. It must be between 0 and varies between dies. It must be one of the SERDES blocks used in the design.

lane *num*

The SERDES lane number. It must be between 0 and 3. It must be one of the lanes enabled for the block in the design.

Supported Families

SmartFusion2, IGLOO2, and RTG4

Example

```
serdes_lane_reset -serdes 0 -lane 0
```

In EPCS mode, resets Lane 0, for block 0. In PCI mode, resets Lane 0 for block 0, all other lanes in the same link for block 0

```
serdes_lane_reset -serdes 5 -lane 3
```

Errors

The following errors result in the failure of the Tcl command and the corresponding message on the smart debug log window:

When the “-serdes” parameter is not specified:

```
Error: Required parameter 'serdes' is missing.
```

```
Error: Failure when executing Tcl script. [Line 26: Error in command serdes_lane_reset]
```

```
Error: The Execute Script command failed.
```

When the “lane” parameter is not specified:

```
Error: Required parameter 'lane' is missing.  
Error: Failure when executing Tcl script. [Line 26: Error in command serdes_lane_reset]  
Error: The Execute Script command failed.
```

When “block number” is not specified:

```
Error: Parameter 'serdes' has illegal value.  
Error: Failure when executing Tcl script. [Line 26: Error in command serdes_lane_reset]  
Error: The Execute Script command failed.
```

When “lane number” is not specified:

```
Error: Required parameter 'lane' is missing.  
Error: Failure when executing Tcl script. [Line 26: Error in command serdes_lane_reset]  
Error: The Execute Script command failed.
```

When “block number” is invalid:

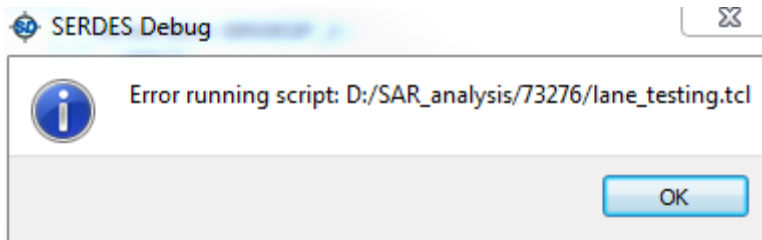
```
Error: Phy Reset: Serdes block number should be one of the following: 0  
Error: The command 'serdes_lane_reset' failed.  
Error: Failure when executing Tcl script. [Line 26]  
Error: The Execute Script command failed.
```

Note: Only the SERDES blocks used the design will be mentioned in the above list.

When “lane number” is invalid:

```
Error: Phy Reset: Serdes lane number should be between 0 and 3.  
Error: The command 'serdes_lane_reset' failed.  
Error: Failure when executing Tcl script. [Line 26]  
Error: The Execute Script command failed.
```

For all the above scenarios, the following message appears:



serdes_read_register (SmartFusion2, IGLOO2, and RTG4)

Tcl command; reads the SERDES register value and displays the result in the log window/console.

```
serdes_read_register -serdes num [ -lane num ] -name REGISTER_NAME
```

Arguments

-serdes *num*

SERDES block number. Must be between 0 and and varies between dies.

-lane *num*

SERDES lane number. Must be between 0 and 3.

The lane number must be specified when the lane register is used. Otherwise, the command will fail.

When the lane number is specified along with the SYSTEM or PCIe register, the command will fail with an error message, as the lane is not applicable to them.

-name *REGISTER_NAME*

Name of the SERDES register.

Supported Families

SmartFusion2, IGLOO2, and RTG4

Example

```
serdes_read_register -serdes 0 -name SYSTEM_SER_PLL_CONFIG_HIGH  
serdes_read_register -serdes 0 -lane 0 -name CRO
```

[serdes_write_register](#)

[UG0567: RTG4 High-Speed Serial Interfaces User Guide](#) (includes all SERDES register names)

[UG0447: SmartFusion2 and IGLOO2 FPGA High-Speed Serial Interfaces User Guide](#)

serdes_write_register (SmartFusion2, IGLOO2, and RTG4)

Tcl command; writes the value to the SERDES register. Displays the result in the log window/console.

```
serdes_write_register -serdes num [-lane num ] -name REGISTER_NAME -value 0x1234
```

Arguments

-serdes *num*

SERDES block number. Must be between 0 and 5 and varies between dies.

-lane *num*

SERDES lane number. Must be between 0 and 3.

The lane number should be specified when the lane register is used. Otherwise, the command will fail.

When the lane number is specified along with the SYSTEM or PCIe register, the command will fail with an error message, as the lane is not applicable to them.

-name *REGISTER_NAME*

Name of the SERDES register.

-value

Specify the value in hexadecimal format.

Supported Families

SmartFusion2, IGLOO2, and RTG4

Example

```
serdes_write_register -serdes 0 -name SYSTEM_SER_PLL_CONFIG_HIGH -value 0x5533
```

See Also

[serdes_read_register.htm](#)

[UG0567: RTG4 High-Speed Serial Interfaces User Guide](#) (includes all SERDES register names)

[UG0447: SmartFusion2 and IGLOO2 FPGA High-Speed Serial Interfaces User Guide](#)

set_debug_device

Identifies the device you intend to debug.

```
set_debug_device -name {device_name}
```

Arguments

name {device_name}

Device name. The device name is not required if there is only one device in the current configuration.

Supported Families

SmartFusion2, IGLOO2, SmartFusion, IGLOO, ProASIC3 and Fusion

Exceptions

None

Example

The following example identifies the device 'A3P250' for debugging:

```
set_debug_device -name {A3P250}
```

set_debug_programmer

Identifies the programmer you want to use for debugging (if you have more than one). The name of the programmer is the serial number on the bar code label on the FlashPro programmer.

```
set_debug_programmer -name {programmer_name}
```

Arguments

-name {programmer_name}

Programmer name is the serial number on the bar code label of the FlashPro programmer.

Supported Families

SmartFusion, IGLOO, ProASIC3 and Fusion

Exceptions

None

Example

The following example selects the programmer 10841

```
set_debug_programmer -name {10841}
```

set_live_probe (SmartFusion2, IGLOO2, RTG4)

Tcl command; set_live_probe channels A and/or B to the specified probe point(s). At least one probe point must be specified. Only exact probe name is allowed (i.e. no search pattern that may return multiple points).

```
set_live_probe [-deviceName device_name] [-probeA probe_name] [-probeB probe_name]
```

Arguments

-deviceName *device_name*

Parameter is optional if only one device is available in the current configuration or set for debug (see SmartDebug user guide for details).

-probeA *probe_name*

Specifies target probe point for the probe channel A.

-probeB *probe_name*

Specifies target probe point for the probe channel B.

Supported Families

SmartFusion2, IGLOO2, RTG4

Exceptions

- The array must be programmed and active
- Active probe read or write operation will affect current settings of Live probe since they use same probe circuitry inside the device
- Setting only one Live probe channel affects the other one, so if both channels need to be set, they must be set from the same call to set_live_probe
- Security locks may disable this function
- In order to be available for Live probe, ProbeA and ProbeB I/O's must be reserved for Live probe respectively

Example

Sets the Live probe channel A to the probe point A12 on device sf2.

```
set_live_probe [-deviceName sf2] [-probeA A12]
```

ungroup (SmartFusion2, IGLOO2, and RTG4)

Tcl command; disassociates the probes as a group.

```
nngroup -name group_name
```

Arguments

-name *group_name*

Name of the group.

Supported Families

SmartFusion2, IGLOO2, and RTG4

Example

```
ungroup -name my_grp4
```

unset_live_probe

Tcl command; discontinues the debug function and clears both live probe channels (Channel A and Channel B). An all zeros value is shown for both channels in the oscilloscope.

Note: For RTG4, only one probe channel (Probe Read Data Pin) is available.

```
unset_live_probe [-deviceName device_name]
```

Arguments

-deviceName *device_name*

Parameter is optional if only one device is available in the current configuration or set for debug (see the SmartDebug User's Guide for details).

Supported Families

SmartFusion2, IGLOO2, and RTG4

Exceptions

- The array must be programmed and active.
- Active probe read or write operation affects current of Live Probe settings, because they use the same probe circuitry inside the device.
- Security locks may disable this function.

Example

The following example unsets both live probe channels (Channel A and Channel B) from the device sf2.

```
unset_live_probes [-deviceName sf2]
```

write_active_probe (SmartFusion2, IGLOO2, and RTG4)

Tcl command; sets the target probe point on the device to the specified value. The target probe point name must be specified.

```
write_active_probe [-deviceName device_name] -name probe_name -value true/false  
-group_name group_bus_name -group_value "hex-value" | "binary-value"
```

Arguments

-deviceName *device_name*

Parameter is optional if only one device is available in the current configuration.

-name *probe_name*

Specifies the name for the target probe point. Cannot be a search pattern.

-value *true* | *false* *hex-value* | *binary-value*

Specifies values to be written.

True = High

False = Low

-group_name *group_bus_name*

Specify the group or bus name to write to complete group or bus.

-group_value "*hex-value*" | "*binary-value*"

Specify the value for the complete group or bus.

Hex-value format : "*<size>'h<value>*"

Binary-value format: "*<size>'b<value>*"

Supported Families

SmartFusion2, IGLOO2, and RTG4

Example

```
write_active_probe -name out[5]:out[5]:Q -value true <-- write to a single probe
write_active_probe -name grp1.out[3]:out[3]:Q -value low <-- write to a probe in the group
write_active_probe -group_name grp1 -group_value "8'hF0" <-- write the value to complete group
write_active_probe -group_name out -group_value "8'b11110000" \
    -name out[2]:out[2]:Q -value true <-- write multiple probes at the same time.
```

write_Isram (SmartFusion2, IGLOO2, and RTG4)

Tcl command; writes a seven bit word into the specified large SRAM location.

```
write_lsram [-deviceName device_name] -block block_name] -offset offset_value -value value
```

-deviceName *device_name*

Parameter is optional if only one device is available in the current configuration or set for debug (see SmartDebug help for details).

-block *block_name*

Specifies the name for the target block.

-offset *offset_value*

Offset (address) of the target word within the memory block.

-value *value*

Value to be written to the target location.

Supported Families

SmartFusion2, IGLOO2, and RTG4

Exceptions

- You must set a debug file
- Array must be programmed and active
- The maximum value that can be written is 0x1FF
- Security locks may disable this function

Example

Writes a value of 0x1A to the device sf2 in the block sram_block1 with an offset of 16.

```
write_lsrām [-deviceName sf2] -block sram_block1 -offset 16 -value 0x1A
```

write_usram (SmartFusion2, IGLOO2, and RTG4)

Tcl command; writes a seven bit word into the specified uSRAM location.

```
write_usram [-deviceName device_name] -block block_name] -offset offset_value -value value
```

-deviceName *device_name*

Parameter is optional if only one device is available in the current configuration or set for debug (see SmartDebug help for details).

-block *block_name*

Specifies the name for the target block.

-offset *offset_value*

Offset (address) of the target word within the memory block.

-value *value*

Seven-bit value to be written.

Supported Families

SmartFusion2, IGLOO2, and RTG4

Exceptions

- You must set a debug file
- Array must be programmed and active
- The maximum value that can be written is 0x1FF
- Security locks may disable this function

Example

Writes a value of 0x1A to the device sf2 in the block usram_block2 with an offset of 16.

```
write_usram [-deviceName sf2] -block usram_block2 -offset 16 -value 0x1A
```

Embedded Flash Memory (NVM) - Failure when Programming/Verifying

If the Embedded Flash Memory failed verification when executing the PROGRAM_NVM, VERIFY_NVM or PROGRAM_NVM_ACTIVE_ARRAY action, the failing page may be [corrupted](#). To confirm and address this issue:

1. In the **Inspect Device** window click **View Flash Memory Content**.
2. Select the Flash Memory block and client (or page range) to retrieve from the device.
3. Click **Read from Device**; the retrieved data appears in the lower part of the window.
4. Click **View Detailed Status**.

Note: You can use the `check_flash_memory` and `read_flash_memory` Tcl commands to perform diagnostics similar to the commands outlined above.

5. To reset the corrupted NVM pages, either re-program the pages with your original data or 'zero-out' the pages by using the Tcl command [recover flash memory](#).

If the Embedded Flash Memory failed verification when executing a VERIFY_NVM or VERIFY_NVM_ACTIVE_ARRAY action, the failure may be due to the change of content in your design. To confirm this, repeat steps 1-3 above.

Note: NVM corruption is still possible when writing from user design. Check NVM status for confirmation.

SmartDebug Frequently Asked Questions

How do I unlock the device security so I can debug?

You must provide the PDB file with a User Pass Key in order to unlock the device and continue debugging. If you do not have a PDB with User Pass Key, [see the FlashPro User Guide](#) to create a PDB file (if you know the Pass Key value).

How do I export a report?

You can export three reports from the SmartDebug GUI: Device Status, Client Detailed Status from the NVM, or the Compare Client Content report from the NVM. Each of those reports can be saved and printed. For more information about Tcl commands supported by SmartDebug, see [SmartDebug Tcl Commands](#).

Where can I find files to compare my contents/settings?

FlashROM

You can compare the FlashROM content in the device with the data in the PDB file. You can find the PDB in the <Liberio IDE project>/Designer/Impl directory.

Embedded Flash Memory (NVM)

You can compare the Embedded Flash Memory content in the device with the data in the PDB file. You can find the PDB in the <Liberio IDE project>/Designer/Impl directory.

What is a UFC file? What is an EFC file?

UFC is the User FlashROM Configuration file, generated by the FlashROM configurator; it contains the partition information set by the user. It also contains the user-selected data for region types with static data. However, for AUTO_INC and READ_FROM_FILE, regions the UFC file contains only:

- Start value, end value, and step size for AUTO_INC regions, and
- File directory for READ_FROM_FILE regions

EFC is the Embedded Flash Configuration file, generated by the Flash Memory Builder in the Project Manager [Catalog](#); it contains the partition information and data set by the user.

Both UFC and EFC information is embedded in the PDB when you generate the PDB file.

Is my FPGA fabric enabled?

When your FPGA fabric is programmed, you will see the following statement under Device State in the Device Status report:

```
FPGA Array Status: Programmed and Enabled
```

If the FPGA fabric is not programmed, the Device State shows:

```
FPGA Array Status: Not Enabled
```

Embedded Flash Memory (NVM) Frequently Asked Questions

How do I display Embedded Flash Memory (NVM) content in the Client partition?

You must load your PDB into your FlashPro project in order to view the Embedded Flash Memory content in the Client partition. To view NVM content in the client partition:

1. Load your PDB into your FlashPro project.
2. Click **Inspect Device**.
3. Click **View Flash Memory Content**.
4. Choose a block from the drop-down menu.
5. Select a client.
6. Click **Read from Device**. The Embedded Flash Memory content from the device appears in the Flash Memory dialog box.

How do I know if I have Embedded Flash Memory (NVM) corruption?

When Embedded Flash Memory is [corrupted](#), [checking Embedded Flash Memory](#) may return with any or all of the following page status:

- ECC1/ECC2 failure
- Page write count exceeds the 10-year retention threshold
- Page write count is invalid
- Page protection is set illegally (set when it should not be)

See the [How do I interpret data in the Flash Memory \(NVM\) Status Report?](#) topic for details.

If your Embedded Flash Memory is corrupted, you can recover by reprogramming with original design data. Alternatively, you can 'zero-out' the pages by using the Tcl command [recover_flash_memory](#).

Why does Embedded Flash Memory (NVM) corruption happen?

Embedded Flash Memory corruption occurs when Embedded Flash Memory programming is interrupted due to:

- Supply brownout; monitor power supplies for brownout conditions. For SmartFusion monitor the VCC_ENVM/VCC_ROSC voltage levels; for Fusion, monitor VCC_NVM/VCC_OSC.
- Reset signal is not properly tied off in your design. Check the Embedded Memory reset signal.

How do I recover from Embedded Flash Memory corruption?

Reprogram with original design data or 'zero-out' the pages by using the Tcl command [recover_flash_memory](#).

What is a JTAG IR-Capture value?

JTAG IR-Capture value contains private and public device status values. The public status value in the value read is ISC_DONE, which indicates if the FPGA Array is programmed and enabled.

The ISC_DONE signal is implemented as part of IEEE 1532 specification.

What does the ECC1/ECC2 error mean?

ECC is the Error Correction Code embedded in each Flash Memory page.

ECC1 – One bit error and correctable.

ECC2 – Two or more errors found, and not correctable.

FlashROM Frequently Asked Questions

Can I compare serialization data?

To compare the serialization data, you can read out the FlashROM content and visually check data in the serialization region. Note that a serialization region can be an AUTO_INC or READ_FROM_FILE region.

For serialization data in the AUTO_INC region, check to make sure that the data is within the specified range for that region.

For READ_FROM_FILE region, you can search for a match in the source data file.

Can I tell what security options are programmed in my device?

To determine the programmed security settings, run the Device Status option from the Inspect Device dialog and examine the Security Section in the report.

This section lists the security status of the FlashROM, FPGA Array and Flash Memory blocks.

How do I interpret data in the Device Status report?

The Device Status Report generated from the FlashPro SmartDebug Feature contains the following sections:

- IDCode (see below)
- [User Information](#)
- [Device State](#)
- [Factory Data](#)
- [Security Settings](#)

Device Status Report: IDCode

The IDCode section shows the raw IDCode read from the device. For example, in the Device Status report for an AFS600 device, you will find the following statement:

```
IDCode (HEX): 233261cf
```

The IDCode is compliant to IEEE 1149.1. The following table lists the IDCode bit assignments:

Table 7 · IDCode Bit Assignments

Bit Field (little endian)	Example Bit Value for AFS600 (HEX)	Description
Bit [31-28] (4 bits)	2	Silicon Revision
Bit [27-12] (16 bits)	3326	Device ID
Bit [11-0] (12 bits)	1cf	IEEE 1149.1 Manufacturer ID for Microsemi

Device Status Report: User Info

The User Information section reports the information read from the User ROW (UROW) of IGLOO, ProASIC3, SmartFusion and Fusion devices. The User Row includes user design information as well as troubleshooting information, including:

- Design name (10 characters max)
- Design check sum (16-bit CRC)
- Last programming setup used to program/erase any of the silicon features.
- FPGA Array / Fabric programming cycle count

For example:

```
User Information:
UROW data (HEX): 603a04e0a1c2860e59384af926fe389f
Programming Method: STAPL
Programmer: FlashPro3
Programmer Software: FlashPro vX.X
Design Name: ABCBASICTO
Design Check Sum: 603A
Algorithm Version: 19
Array Prog. Cycle Count: 19
```

Table 8 · Device Status Report User Info Description

Category	Field	Description
User Row Data	(Example) UROW data (HEX): 603a04e0a1c2860e59384af926fe389f	Raw data from User Row (UROW)
Programming Troubleshooting Info	(Example) Programming Method: STAPL Programmer: FlashPro3 Programmer Software: FlashPro v8.6 Algorithm Version: 19	Known programming setup used. This includes: Programming method/file, programmer and software. It also includes programming Algorithm version used.
Design Info	(Example) Design Name: ABCASICTO Design Check Sum: 603A	Design name (limited to 10 characters) and check sum. Design check sum is a 16-bit CRC calculated from the fabric (FPGA Array) datastream generated for programming. If encrypted datastream is generated selected, the encrypted datastream is used for calculating the check sum.

Device Status Report: Device State

The device state section contains:

- IR-Capture register value, and
- The FPGA status

The IR-Capture is the value captured by the IEEE1149.1 instruction register when going through the IR-Capture state of the IEEE 1149.1 state machine. It contains information reflecting some of the states of the devices that is useful for troubleshooting.

One of the bits in the value captured is the ISC_DONE value, specified by IEEE 1532 standard. When the value is '1' it means that the FPGA array/fabric is programmed and enabled. This is available for IGLOO, ProASIC3, SmartFusion and Fusion devices.

For example:

```
Device State:
IRCapture Register (HEX): 55
FPGA Array Status: Programmed and enabled
```

For a blank device:

```
Device State:
IRCapture Register (HEX): 51
FPGA Array Status: Not enabled
```

Device Status Report: Factory Data

The Factory Data section lists the Factory Serial Number (FSN).

Each of the IGLOO, ProASIC3, SmartFusion and Fusion devices has a unique 48-bit FSN.

Device Status Report: Security

The security section shows the security options for the FPGA Array, FlashROM and Flash Memory (NVM) block that you programmed into the device.

For example, using a Fusion AFS600 device:

```
Security:
Security Register (HEX): 0000000088c01b
FlashROM
Write/Erase protection: Off
Read protection: Off
Encrypted programming: Off
FPGA Array
Write/Erase protection: Off
Verify protection: Off
Encrypted programming: Off
FlashMemory Block 0
Write protection: On
Read protection: On
Encrypted programming: Off
FlashMemory Block 1
Write protection: On
Read protection: On
Encrypted programming: Off
```

Table 9 · Device Status Report - Security Description

Security Status Info	Description
Security Register (HEX)	Raw data captured from the device's security status register
Write/Erase Protection	Write protection is applicable to FlashROM, FPGA Array (Fabric) and Flash Memory (NVM) blocks. When On, the Silicon feature is write/erase protected by user passkey.

Security Status Info	Description
Read Protection	Read protection is applicable to FlashROM and Flash Memory (NVM) blocks. When On, the Silicon feature is read protected by user passkey.
Verify Protection	Verify Protection is only applicable to FPGA Array (Fabric) only. When On, the FPGA Array require user passkey for verification. Reading back from the FPGA Array (Fabric) is not supported. Verification is accomplished by sending in the expected data for verification.
Encrypted Programming	Encrypted Programming is supported for FlashROM, FPGA Array (Fabric) and Flash Memory (NVM) blocks. When On, the silicon feature is enable for encrypted programmed. This allows field design update with encrypted datastream so the user design is protected.

Encrypted Programming

To allow encrypted programming of the features, the target feature cannot be Write/Erase protected by user passkey.

The security settings of each silicon feature when they are enabled for encrypted programming are listed below.

FPGA Array (Fabric)

Write/Erase protection: Off
Verify protection: Off
Encrypted programming: On

Set automatically by Designer or FlashPro when you select to enable encrypted programming of the FPGA Array (Fabric). This setting allows the FPGA Array (Fabric) to be programmed and verified with an encrypted datastream.

FlashROM

Write/Erase protection: Off
Read protection: On
Encrypted programming: On

Set automatically by Designer or FlashPro when you select to enable encrypted programming of the FlashROM. This setting allows the FlashROM to be programmed and verified with an encrypted datastream.

FlashROM always allows verification. If encrypted programming is set, verification has to be performed with encrypted datastream.

Designer and FlashPro automatically set the FlashROM to be read protected by user passkey when encrypted programming is enabled. This protects the content from being read out of the JTAG port after encrypted programming.

Flash Memory (NVM) Block

Write/Erase protection: Off
Read protection: On
Encrypted programming: On

The above setting is set automatically set by Designer or FlashPro when you select to enable encrypted programming of the Flash Memory (NVM) block. This setting allows the Flash Memory (NVM) block to be programmed with an encrypted datastream.

The Flash Memory (NVM) block does not support verification with encrypted datastream.

Designer and FlashPro automatically set the Flash Memory (NVM) block to be read protected by user passkey when encrypted programming is enabled. This protects the content from being read out of the JTAG port after encrypted programming.

How do I interpret data in the Flash Memory (NVM) Status Report?

The Embedded Flash Memory (NVM) Status Report generated from the FlashPro SmartDebug feature consists of the page status of each NVM page. For example:

```
Flash Memory Content [ Page 34 to 34 ]
FlashMemory Page #34:
Status Register(HEX): 00090000
Status ECC2 check: Pass
Data ECC2 Check: Pass
Write Count: Pass (2304 writes)
Total number of pages with status ECC2 errors: 0
Total number of pages with data ECC2 errors: 0
Total number of pages with write count out of range: 0
FlashMemory Check PASSED for [ Page 34 to 34 ]
The 'check_flash_memory' command succeeded.
The Execute Script command succeeded.
```

Table 10 · Embedded Flash Memory Status Report Description

Flash Memory Status Info	Description
Status Register (HEX)	Raw page status register captured from device
Status ECC2 Check	Check for ECC2 issue in the page status
Data ECC2 Check	Check for ECC2 issue in the page data
Write Count	<p>Check if the page-write count is within the expected range.</p> <p>The expected write count is greater than or equal to:</p> <p>6,384 - SmartFusion devices 2,288 - Fusion devices</p> <p>Note: Write count, if corrupted, cannot be reset to a valid value within the customer flow; invalid write count will not prevent device from being programmed with the FlashPro tool.</p> <p>The write count on all good eNVM pages is set to be 2288 instead of 0 in the manufacturing flow. The starting count of the eNVM is 2288. Each time the page is programmed or erased the count increments by one. There is a Threshold that is set to 12288, which equals to 3 * 4096.</p> <p>Since the threshold can only be set in multiples of 4096 (2¹²), to set a 10,000 limit, the Threshold is set to 12288 and the start count is set to</p>

Flash Memory Status Info	Description
	2288; and thus the eNVM has a 10k write cycle limit. After the write count exceeds the threshold, the STATUS bit goes to 11 when attempting to erase/program the page.

Handoff Design for SmartFusion2, IGLOO2, and RTG4

Export Bitstream - SmartFusion2, 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 Security Policy Manager settings:
 - [Bitstream Encryption with the Default Key in the Security Policy Manager - SmartFusion2 and IGLOO2](#)
 - [Enable Custom Security Options in the Security Policy Manager - SmartFusion2 and IGLOO2](#)
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. Set your **eNVM Serialization options**, if any. eNVM Serialization must be enabled via the [eNVM Configurator in your MSS](#).
4. Select the bitstream components (Fabric/eNVM) that you want to program.
5. 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.

See Also

[Digest File](#)

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.

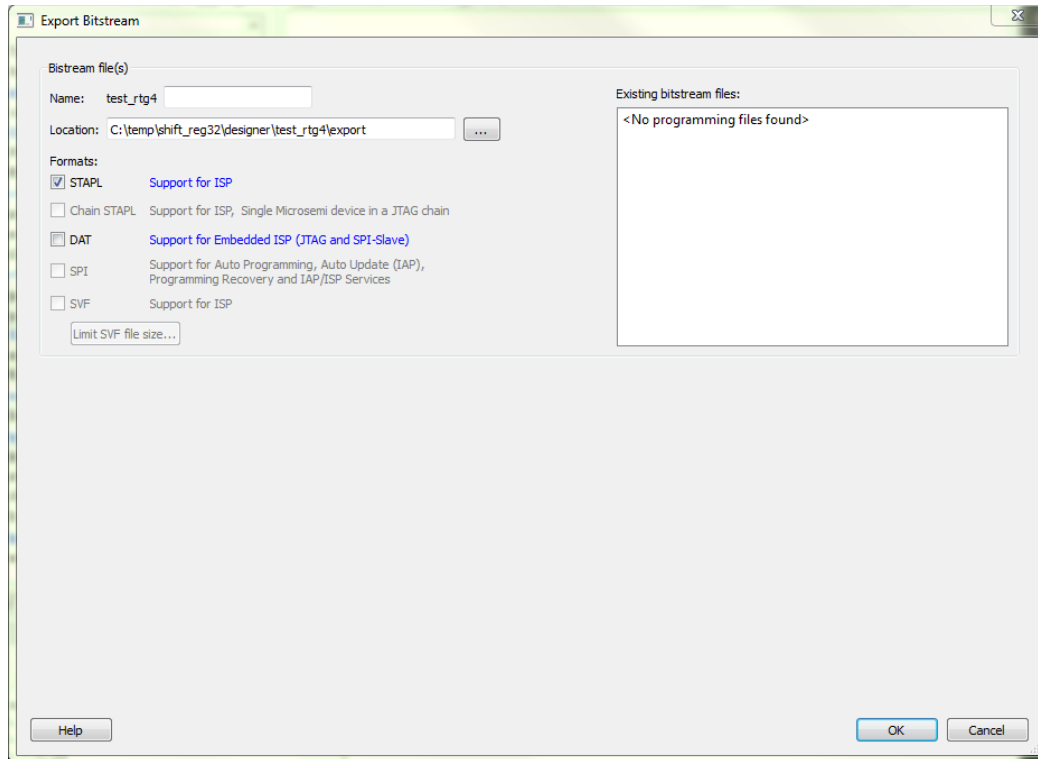


Figure 180 · Export Bitstream Dialog Box

Export FlashPro Express Job - SmartFusion2, IGLOO2, RTG4

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

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

SmartFusion2 and IGLOO2

The Export FlashPro Express Job dialog box options vary depending on the Security Key Mode you select in the [Security Policy Manager](#).

Select Bitstream Encryption with Default Key in the [Security Policy Manager](#)

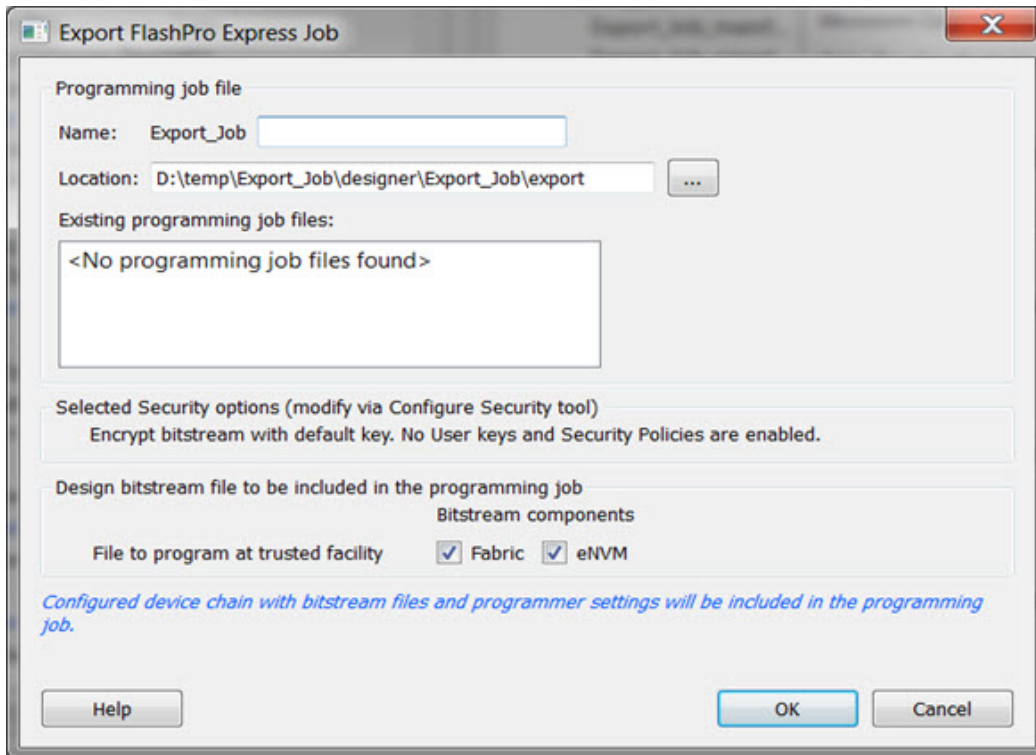


Figure 181 · Export FlashPro Express Job Dialog Box -- SmartFusion2 and IGLOO2

Programming job file

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

Location - Location of the file to be exported.

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

Selected Security Level (Modify via [Security Policy Manager](#)) - Gives a brief description of current security options.

Design bitstream file to be included in the programming job - Lists all the available bitstream files, one of which will be included in the programming job for the current target device.

File to program at trusted facility - Click to enable programming for Fabric and/or eNVM bitstream components at a trusted facility.

Enable Custom Security Options in the [Security Policy Manager](#)

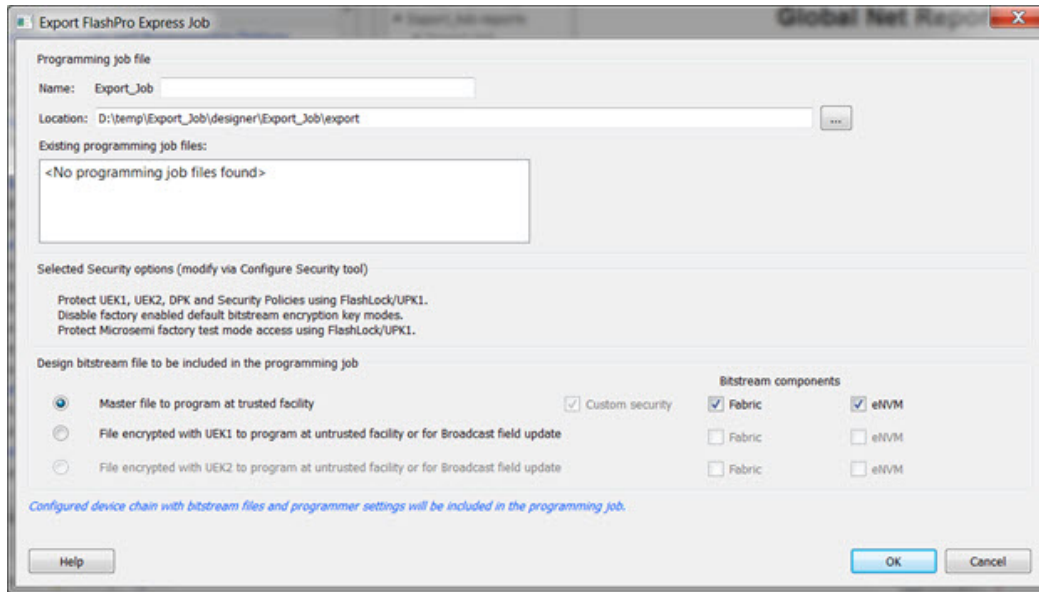


Figure 182 · Export FlashPro Express Job Dialog Box - Enable Custom Security Options

Programming job file name - All names use a prefix as shown in your software.

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

Selected Security Level (Modify via [Security Policy Manager](#)) - Gives a brief description of current security options.

Design bitstream file to be included in the programming job - Lists all the available bitstream files, one of which will be included in the programming job for the current target device.

Master 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 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 bitstream files to be programmed. If the selected components are not protected by UPK1, the bitstream can be programmed at an untrusted location, since it is encrypted with UEK1 that is pre-programmed 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 bitstream files to be programmed. If the selected components are not protected by UPK1, the bitstream can be programmed at an untrusted location, since it is encrypted with UEK2 that is pre-programmed 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 an untrusted location.

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

RTG4

Programming job file name - All names use a prefix as shown in your software.

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

The configured device chain with bitstream files and programmer settings will be included in the programming job.

Note: Security Programming is not supported for RTG4.

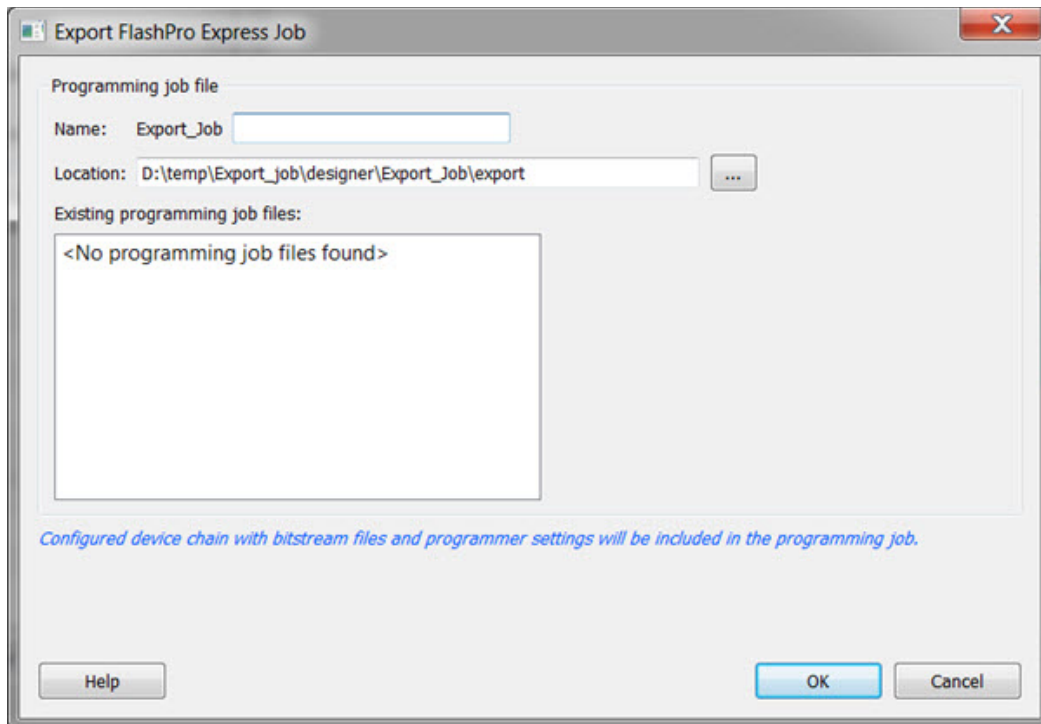


Figure 183 · Export FlashPro Express Job Dialog Box - RTG4

Prepare Design for Production Programming in FlashPro Express

After you have exported a programming job you can handoff this programming job to the FlashPro Express tool for production programming. To do so:

In FlashPro Express, from the **File** menu choose **Create Job Project From a Programming Job**. You will be prompted to specify the Programming Job location that you just exported from Libero and the location of where to store the Job Project. The Job Project name automatically uses the programming job name and cannot be changed. Click OK and a new Job Project will be created and opened for production programming.

Export Job Manager Data - SmartFusion2, IGLOO2

Job Manager is Microsemi's HSM-based security software for job management.

As a part of the SPPS flow, the Export Job Manager Data dialog box allows a design engineer (user) to export Libero design data to Job Manager. Exported data is used by an operation engineer (OE) using Job Manager to prepare the manufacturing process for HSM or non-HSM flow.

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

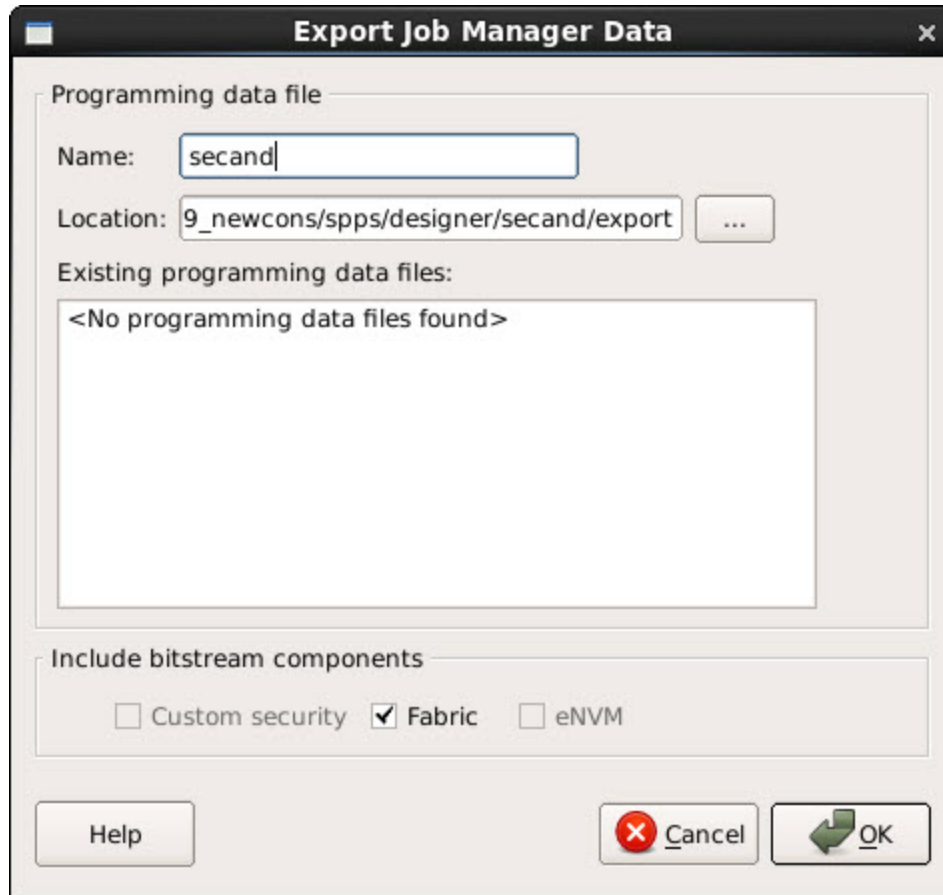


Figure 184 · Export Job Manager Data Dialog Box

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

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

Location - Location of the file to be exported.

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

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

Export Pin Report

Double-click **Export Pin Report** to display the pin report in your [Design Report](#)

The Pin Report lists the pins in your device. Double-click **Export Pin Report** and choose specific report types from the **Export Pin Reports** dialog box (see below). Your options are:

- <design>_pinrpt_name.rpt - Pin report sorted by name.
- <design>_pinrpt_number.rpt - Pin report sorted by pin number.
- <design>_bankrpt.rpt - I/O Bank Report
- <design>_ioff.xml - I/O Register Combining Report

You must select at least one report.

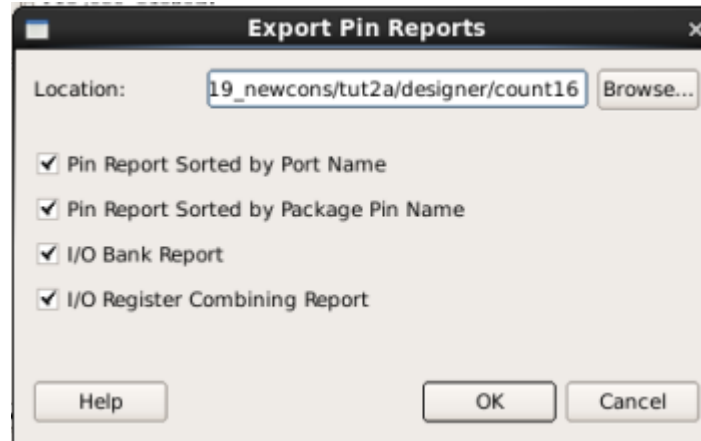


Figure 185 · Export Pin Reports Dialog Box

Export BSDL File

Double-click Export BSDL File (in the Libero SoC Design Flow window, **Handoff Design for Production > Export BSDL File**) to generate the BSDL File report to your [Design Report](#).

The BSDL file provides a standard file format for electronics testing using JTAG. It describes the boundary scan device package, pin description and boundary scan cell of the input and output pins. BSDL models are available as downloads for many Microsemi SoC devices; see the [Microsemi website for more information](#).

Export IBIS Model

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

The IBIS model report provides an industry-standard file format for recording parameters like driver output impedance, rise/fall time, and input loading, which may then be used by software applications such as Signal Integrity tools or IBIS simulators.

The exported IBIS file has the file extension *.ibs (named <root>.ibs) and is displayed in the Files tab.

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

In the [Pin] section of the IBIS *.ibs file, listed under the model_name are the Model Selector tag. The IBIS *.ibs file has a [Model Selector] section that describes the model selector and its list of models. The Model Selector tag in the [Pin] section establishes the relationship between the pin and the [Model Selector].

[Pin]	signal_name	model_name	R_pin	I_pin	C_pin
AE33	PAD_O<27>	LVCNOS15_Bank15Group1_asiod	0.75142	5.68346e-009	2.14353e-012
E31	PAD_I<105>	LVCNOS25_Bank1Group1_ddrio	0.609967	5.61206e-009	2.47835e-012
A2	PAD_O<33>	LVCNOS25_Bank2Group1_ddrio	0.786904	7.52853e-009	2.87418e-012

[Model Selector]	model_name
LVCNOS15_6mA_MSIOD	LVCNOS15_6mA_MSIOD
LVCNOS15_2mA_MSIOD	LVCNOS15_2mA_MSIOD
LVCNOS15_4mA_MSIOD	LVCNOS15_4mA_MSIOD

← Model Selector for Pin AE33

[Model Selector]	model_name
LVCNOS25_2mA_MEDFAST_DDRIO	LVCNOS25_2mA_MEDFAST_DDRIO
LVCNOS25_2mA_SLOW_DDRIO	LVCNOS25_2mA_SLOW_DDRIO
LVCNOS25_2mA_MED_DDRIO	LVCNOS25_2mA_MED_DDRIO
LVCNOS25_2mA_FAST_DDRIO	LVCNOS25_2mA_FAST_DDRIO
LVCNOS25_4mA_MEDFAST_DDRIO	LVCNOS25_4mA_MEDFAST_DDRIO
LVCNOS25_4mA_SLOW_DDRIO	LVCNOS25_4mA_SLOW_DDRIO
LVCNOS25_4mA_MED_DDRIO	LVCNOS25_4mA_MED_DDRIO
LVCNOS25_4mA_FAST_DDRIO	LVCNOS25_4mA_FAST_DDRIO

← Model Selector for Pin E31

[Model Selector]	model_name
LVCNOS25_4mA_SLOW_DDRIO	LVCNOS25_4mA_SLOW_DDRIO
LVCNOS25_4mA_MED_DDRIO	LVCNOS25_4mA_MED_DDRIO
LVCNOS25_4mA_MEDFAST_DDRIO	LVCNOS25_4mA_MEDFAST_DDRIO
LVCNOS25_4mA_FAST_DDRIO	LVCNOS25_4mA_FAST_DDRIO
LVCNOS25_2mA_SLOW_DDRIO	LVCNOS25_2mA_SLOW_DDRIO
LVCNOS25_2mA_MED_DDRIO	LVCNOS25_2mA_MED_DDRIO
LVCNOS25_2mA_MEDFAST_DDRIO	LVCNOS25_2mA_MEDFAST_DDRIO
LVCNOS25_2mA_FAST_DDRIO	LVCNOS25_2mA_FAST_DDRIO

← Model Selector for Pin A2

Figure 186 · 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.

Handoff Design for Firmware Development

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](#)

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.

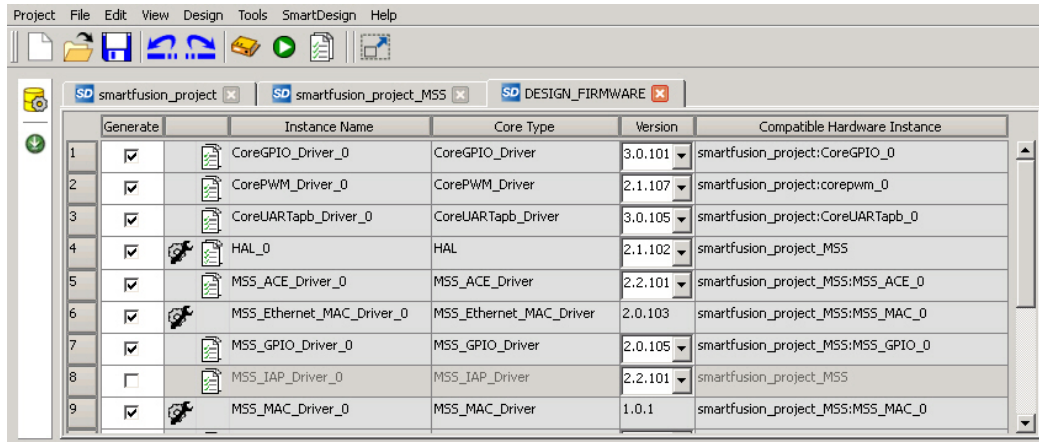


Figure 187 · Firmware Cores Tab (DESIGN_FIRMWARE)

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](#)

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.

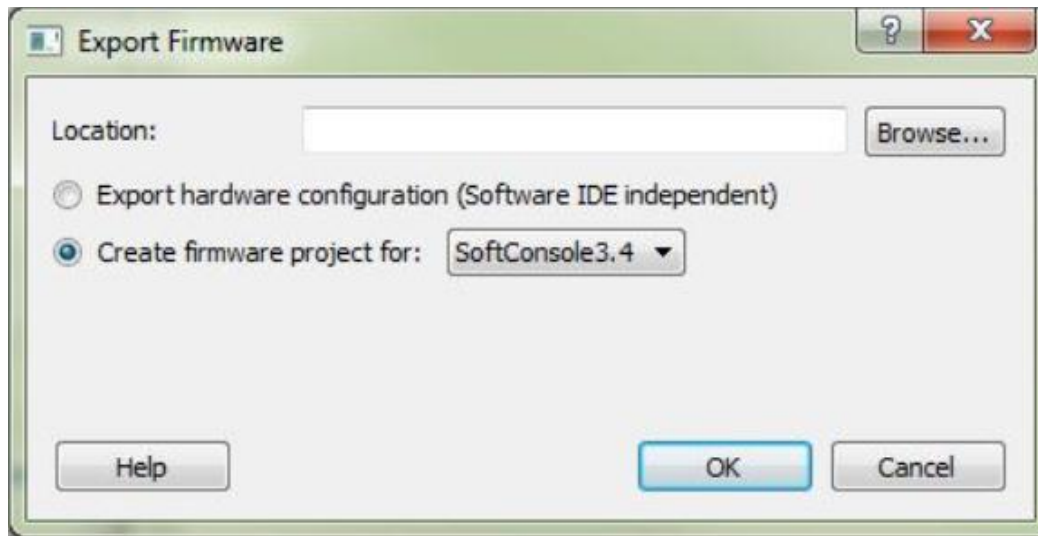


Figure 188 · Export Firmware Dialog Box

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

Export SmartDebug Data (Libero SoC)

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

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

See the following figure for an example.

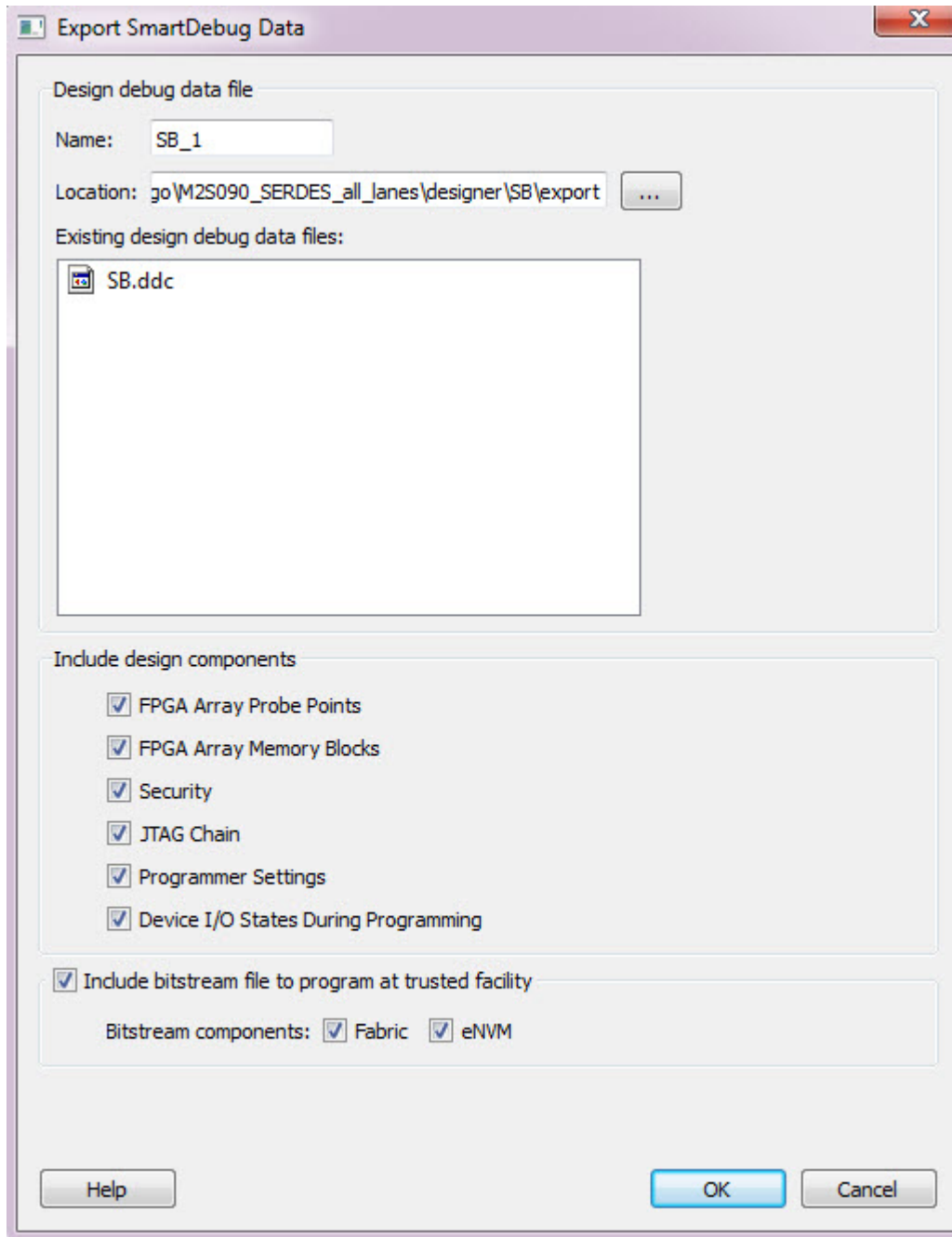


Figure 189 · Export SmartDebug Data Dialog Box

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

Design debug data file (*.ddc)

Name

The name of the design.

Location

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

Existing Design Debug Data Files

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

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

Include design components

A DDC file can contain the following components:

- **FPGA Array Probe Points** – When checked, Libero SoC exports Live and Active probes information (<design>_probe.db file) into the *.ddb container file.
- **FPGA Array Memory Blocks** – When checked, Libero SoC exports information about FPGA memories (<design>_sii_block.db) into the *.ddb container file:
 - o names and addresses of the memory blocks instantiated by the design
 - o data formats selected by the user in the design
- **Security** – This contains the security locks, keys, and security policy information needed for debug. This may be default or custom security (<design>.spm file). It is hidden if security is not supported for the device; for example, RTG4 devices.
- **JTAG Chain** (device chain information configured using Programming Connectivity and Interface in Libero) – When checked, Libero SoC exports chain data including devices, their programming files if loaded, device properties, and so on (<design>.pro file). If JTAG chain is unchecked, the default JTAG chain with Libero design device only is added to the *.ddc file.
- **Programmer Settings** (<design>.pro file) – If Programmer Settings is unchecked, the default programmer settings are added to the *.ddc file.
- **Device I/O States During Programming** (<design>.ios file) – This setting is used by some SmartDebug features, for example, for programming eNVM. It is NOT used during device programming in SmartDebug; programming files used to program devices already have I/O states data.

In addition, you can include bitstream file information, which can be used for programming the device in standalone SmartDebug.

Include Bitstream file to program at trusted facility

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

The default location of the DDC file is: <Libero_Project_directory>/designer/<design_name>/export.

The DDC file can be exported to any user-specified location if the location has read and write permission.

References

Catalog

In the Libero SoC, from the **View** menu choose **Windows > Catalog**.

The Catalog displays a list of available cores, busses and macros (see image below).

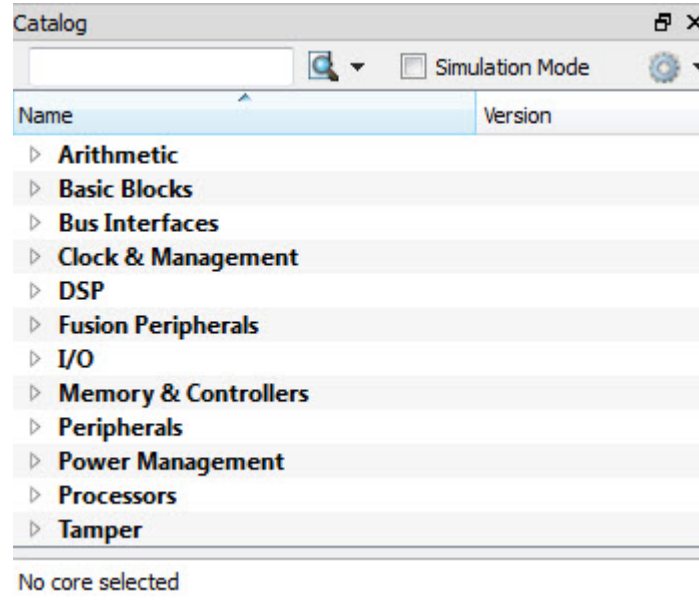


Figure 190 · Libero SoC Catalog

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 [Catalog Display 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  to customize the [Catalog Display Options](#). Click the Catalog Options drop-down arrow to import a core, reload the Catalog, or [enter advanced download mode](#).


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)

See Also

[Project Manager - Cores Dialog Box \(Advanced Download Mode\)](#)

Catalog Options Dialog Box

The Catalog Options dialog box (as shown below) enables you to customize your [Catalog display](#). 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 .

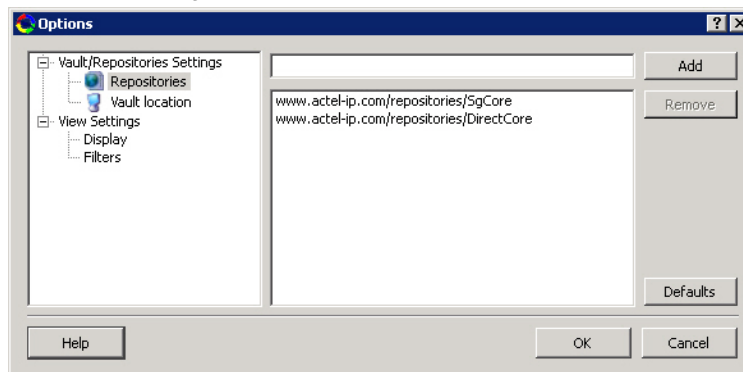


Figure 191 · Catalog Display Options Dialog Box

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 Output Port Capacitance

Output propagation delay is affected by both the capacitive loading on the board and the I/O standard. The I/O Attribute Editor in ChipPlanner provides a mechanism for setting the expected capacitance to improve the propagation delay model. SmartTime automatically uses the modified delay model for delay calculations.

To change the output port capacitance and view the effect of this change in SmartTime Timing Analyzer, refer to the following example. The figure below shows the delay from FF3 to output port OUT2. It shows a delay of 6.603 ns based on the default loading of 35 pF.

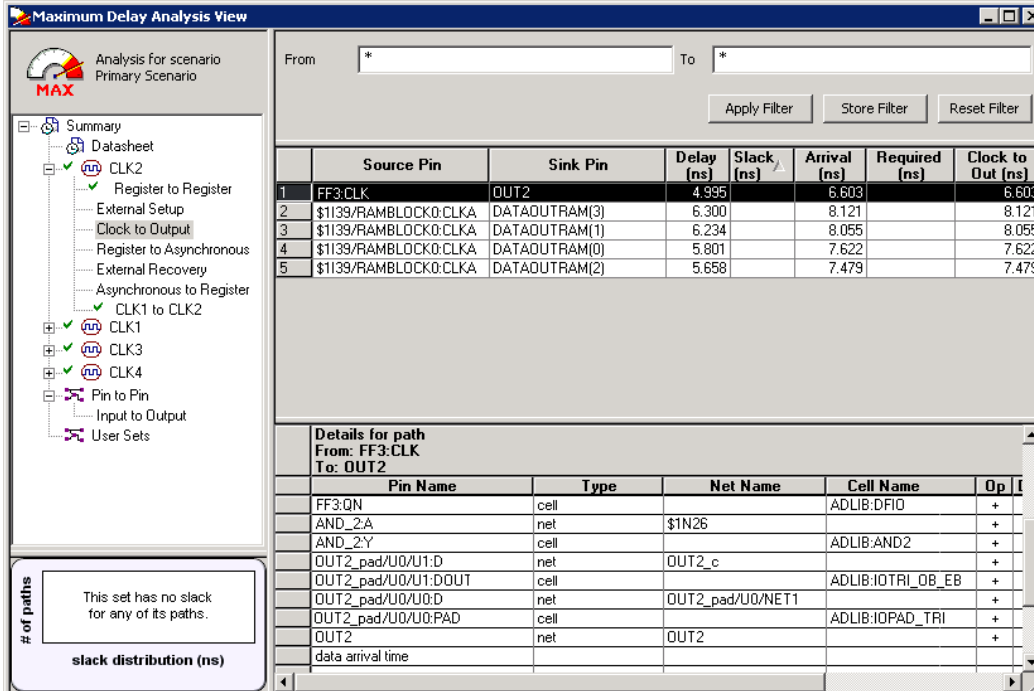


Figure 192 · Maximum Delay Analysis View

If your board has output capacitance of 75pf on OUT2, you must perform the following steps to update the timing number:

1. Open the I/O Attribute Editor and change the output load to 75pf.

Port Name	Macro Cell	Pin #	Locked	Bank Name	IO Standard	Output Drive (mA)	Slew	Resistor Pull	Skew	Output Load	Use I/O Reg
1	CLK2	ADLB:CLKBUF	13	<input type="checkbox"/>	Bank1	LVTTTL	--	None	--		<input type="checkbox"/>
2	CLK4	ADLB:INBUF	15	<input type="checkbox"/>	Bank1	LVTTTL	--	None	--		<input type="checkbox"/>
3	WADDR(3)	ADLB:INBUF	85	<input type="checkbox"/>	Bank0	LVTTTL	--	None	--		<input type="checkbox"/>
4	DATAOUTRAM(2)	ADLB:OUTBUF	86	<input type="checkbox"/>	Bank0	LVTTTL	12	High	<input type="checkbox"/>	35	<input type="checkbox"/>
5	OUT2	ADLB:OUTBUF	16	<input type="checkbox"/>	Bank1	LVTTTL	12	High	<input type="checkbox"/>	75	<input type="checkbox"/>

Figure 193 · I/O Attribute Editor View

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

You can see that the Clock to Output delay changed to 7.723 ns.

Configure Bitstream Dialog Box - SmartFusion2, IGLOO2, and RTG4

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 an eNVM in your design.

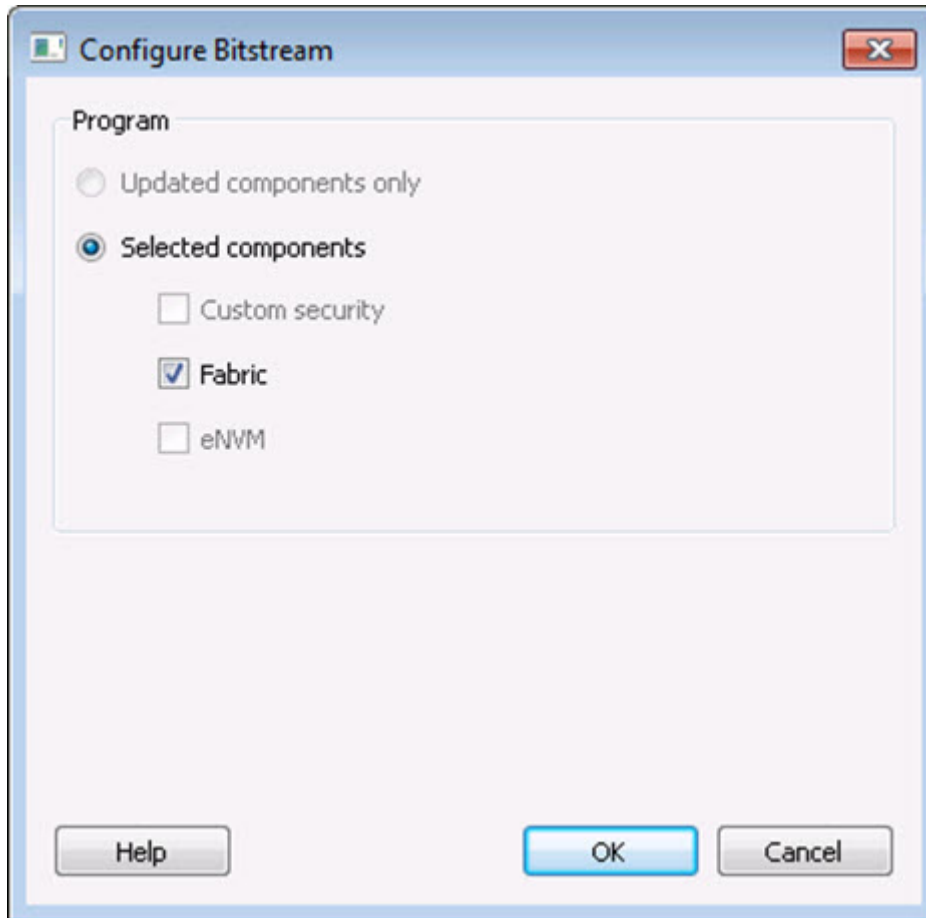


Figure 194 · Configure Bitstream Dialog Box - SmartFusion2 and IGLOO2

Updated components only (not supported in this release) - Updates only the components that have changed since your last programming.

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

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

Importing Source Files – Copying Files Locally

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

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

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

Create Clock Constraint Dialog Box

Use this dialog box to enter a clock constraint setting.

It displays a typical clock waveform with its associated clock information. You can enter or modify this information, and save the final settings as long as the constraint information is consistent and defines the clock waveform completely. The tool displays errors and warnings if information is missing or incorrect. To open the Create Clock Constraint dialog box (shown below) from the SmartTime Constraints Editor, choose **Constraints > Clock**.

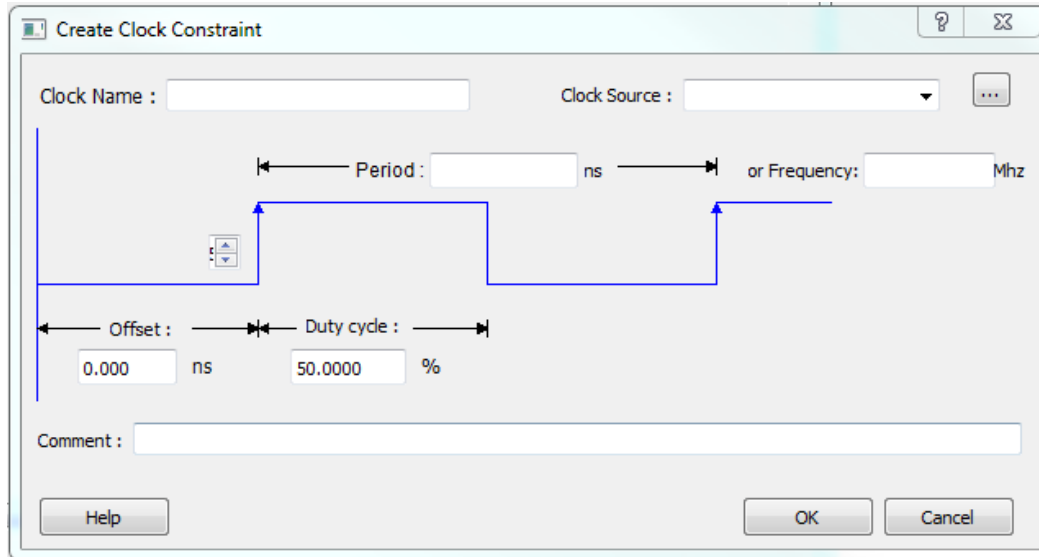


Figure 195 · Create Clock Constraint Dialog Box

Clock Source

Enables you to choose a pin from your design to use as the clock source.

The drop-down list is populated with all explicit clocks. You can also select the Browse button to access all potential clocks. The **Browse** button displays the [Select Source Pins for Clock Constraint Dialog Box](#).

Clock Name

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

T(zer) Label

Instant zero used as a common starting time to all clock constraints.

Period

When you edit the period, the tool automatically updates the frequency value.

The period must be a positive real number. Accuracy is up to 3 decimal places.

Frequency

When you edit the frequency, the tool automatically updates the period value.

The frequency must be a positive real number. Accuracy is up to 3 decimal places.

Offset (Starting Edge Selector)

Enables you to switch between rising and falling edges and updates the clock waveform.
If the current setting of starting edge is rising, you can change the starting edge from rising to falling.
If the current setting of starting edge is falling, you can change the starting edge from falling to rising.

Duty Cycle

This number specifies the percentage of the overall period that the clock pulse is high.
The duty cycle must be a positive real number. Accuracy is up to 4 decimal places. Default value is 50%.

Offset

The offset must be a positive real number. Accuracy is up to 2 decimal places. Default value is 0.

Comment

Enables you to save a single line of text that describes the clock constraints purpose.

See Also

[Specifying Clock Constraints](#)

Create Generated Clock Constraint Dialog Box

Use this dialog box to specify generated clock constraint settings.

It displays a relationship between the clock source and its reference clock. You can enter or modify this information, and save the final settings as long as the constraint information is consistent. The tool displays errors and warnings if the information is missing or incorrect.

To open the Create Generated Clock Constraint dialog box (shown below) from the SmartTime Constraints Editor, choose **Constraints > Generated Clock**.

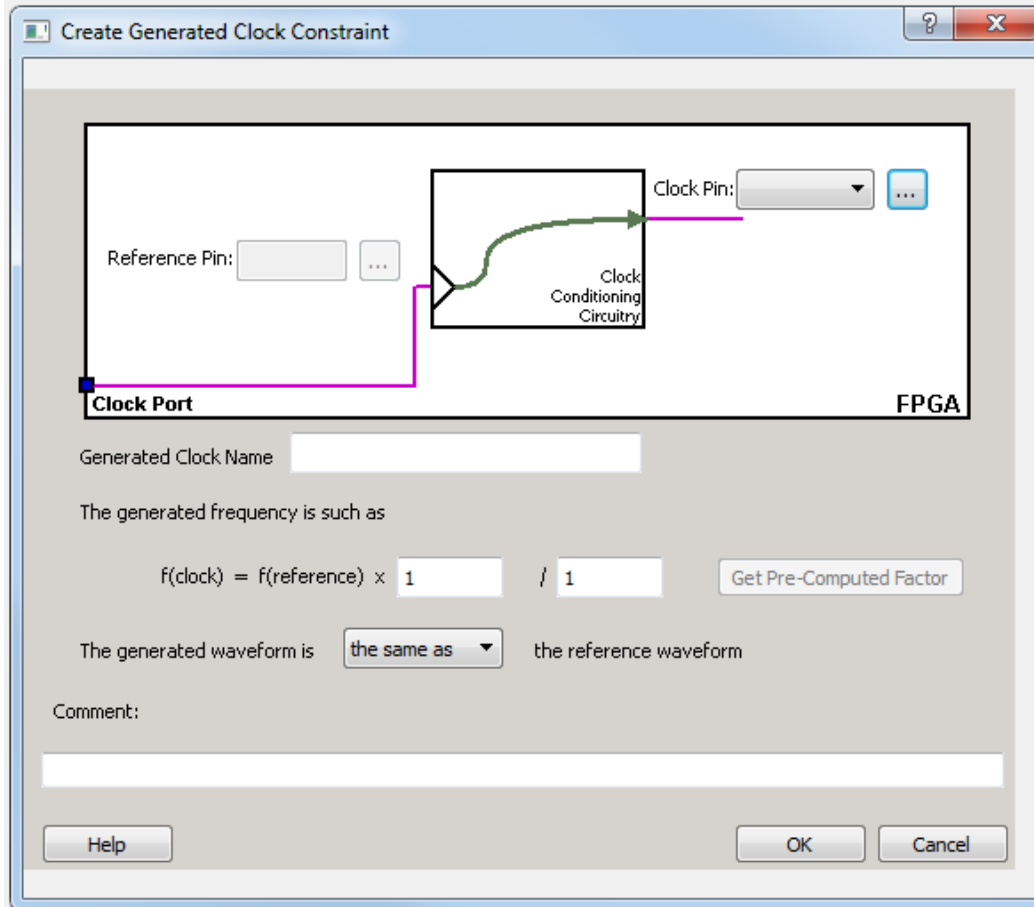


Figure 196 · Create Generated Clock Constraint

Clock Pin

Enables you to choose a pin from your design to use as a generated clock source.

The drop-down list is populated with all unconstrained explicit clocks. You can also select the Browse button to access all potential clocks and pins from the clock network. The Browse button displays the [Select Generated Clock Source](#) dialog box.

Reference Pin

Enables you to choose a pin from your design to use as a generated reference pin.

Generated Clock Name

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

Generated Frequency

The generated frequency is a factor of reference frequency defined with a multiplication element and/or a division element.

Generated Waveform

The generated waveform could be either the same as or inverted w.r.t. the reference waveform.

Comment

Enables you to save a single line of text that describes the generated clock constraints purpose.

See Also

- [create_generated_clock \(SDC\)](#)
- [Specifying Generated Clock Constraints](#)
- [Select Generated Clock Source](#)

Design Hierarchy in the Design Explorer

The Design Hierarchy tab displays a hierarchical representation of the design based on the source files in the project. 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.

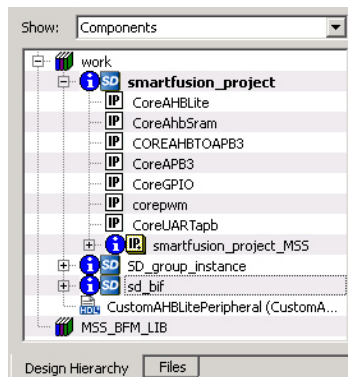


Figure 197 · Design Hierarchy

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 11 · Design Hierarchy Icons

Icon	Description
	SmartDesign component

Icon	Description
	SmartDesign component with HDL netlist not generated
	IP core was instantiated into SmartDesign but the HDL netlist has not been generated
	Core
	Error during core validation
	Updated core available for download
	HDL netlist

Digest File

Users can verify which bitstream file was programmed onto their devices by running the VERIFY or VERIFY_DIGEST actions on each device that was programmed. This is a costly and time-consuming process. To speed up the verification process, digests are printed during bitstream generation and bitstream programming. These digests can be compared to verify that all of the devices were programmed with the correct bitstream file.

The bitstream file is divided into three major component sections: FPGA fabric, eNVM, and Security. A valid bitstream will contain a combination of any of the three primary bitstream components.

Use Case

When a customer creates a design in Libero and then exports the STAPL file (for FlashPro) or programming job (for FlashPro Express), the digest of each of the primary components is printed in the Libero log window and saved in a digest file under the export folder. The digest file is a text file containing the bitstream component name with its corresponding digest. The name of the digest file will match the name of the STAPL/programming job exported, and will be appended with a “.digest” extension.

The customer then sends the STAPL/programming job to a production programming house. Now, when the devices are programmed, the digest of each of the primary components is printed in the log window. The production programming house saves the log files and sends the devices along with log files back to the customer. The customer can then verify that the correct design was programmed on the device by matching the digests in the log file with that in the *.digest file under the Libero export folder.

Example Using STAPL File

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

Libero log:

```
Opened 'D:\flashpro_files\m2s005_digest1\designer\al_MSS\al_MSS_fp\al_MSS.pro'
The 'open_project' command succeeded.
PDB file
'D:\flashpro_files\m2s005_digest1\designer\al_MSS\4a8552f8-57ee-4baa-97ee-2baa57ee2baa.pdb' has
been loaded successfully.
DESIGN : al_MSS; CHECKSUM : DE15; PDB_VERSION : 1.9
The 'load_programming_data' command succeeded.
Sucessfully exported STAPL file:
```

```
'D:\flashpro_files\m2s005_digest1\designer\al_MSS\export\al_MSS.stp'; file programs
Fabric
and eNVM.
Fabric component digest:
276fbefb0a18cc0de1d45efc84589745ee02fc2adbcc1259fbeb674094754014
eNVM component digest:
6b2c2353e25c5982643c32640ac16c581874c8950300135622c126ee22d8b1de
Finished: Thu Jan 22 12:37:32 2015 (Elapsed time 00:00:06)
The 'export_single_stapl' command succeeded.
The 'set_programming_file' command succeeded.
Project saved.
The 'save_project' command succeeded.
Project closed.
```

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

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

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

```
programmer '73207' : Scan Chain...
Warning: programmer '73207' : Vpump has been selected on programmer AND an externally
provided Vpump has also been detected. Using externally provided Vpump voltage source.
programmer '73207' : Check Chain...
programmer '73207' : Scan and Check Chain PASSED.
programmer '73207' : device 'M2S/M2GL005(S)' : Executing action PROGRAM
programmer '73207' : device 'M2S/M2GL005(S)' : Family: SmartFusion2
programmer '73207' : device 'M2S/M2GL005(S)' : Product: M2S005
programmer '73207' : device 'M2S/M2GL005(S)' : EXPORT ISC_ENABLE_RESULT[32] = 007c6b44
programmer '73207' : device 'M2S/M2GL005(S)' : EXPORT CRCERR: [1] = 0
programmer '73207' : device 'M2S/M2GL005(S)' : EXPORT EDCERR: [1] = 0
programmer '73207' : device 'M2S/M2GL005(S)' : TEMPGRADE: ROOM
programmer '73207' : device 'M2S/M2GL005(S)' : EXPORT VPPRANGE: [3] = 2
programmer '73207' : device 'M2S/M2GL005(S)' : VPPRANGE: HIGH
programmer '73207' : device 'M2S/M2GL005(S)' : EXPORT TEMP: [8] = 6b
programmer '73207' : device 'M2S/M2GL005(S)' : EXPORT VPP: [8] = 7c
programmer '73207' : device 'M2S/M2GL005(S)' : Programming FPGA Array and eNVM...
programmer '73207' : device 'M2S/M2GL005(S)' : EXPORT Fabric component
digest[256] =
276fbefb0a18cc0de1d45efc84589745ee02fc2adbcc1259fbeb674094754014
programmer '73207' : device 'M2S/M2GL005(S)' : EXPORT eNVM component digest[256]
= 6b2c2353e25c5982643c32640ac16c581874c8950300135622c126ee22d8b1de
programmer '73207' : device 'M2S/M2GL005(S)' :
=====
programmer '73207' : device 'M2S/M2GL005(S)' : EXPORT DSN[128] =
c6e99c2d1a992f13cf8231c4be847acb
programmer '73207' : device 'M2S/M2GL005(S)' :
=====
programmer '73207' : device 'M2S/M2GL005(S)' : Finished: Thu Jan 22 17:57:37 2015
(Elapsed time 00:00:19)
programmer '73207' : device 'M2S/M2GL005(S)' : Executing action PROGRAM PASSED.
programmer '73207' : Chain programming PASSED.
Chain Programming Finished: Thu Jan 22 17:57:37 2015 (Elapsed time 00:00:19)
o - o - o - o - o - o
```

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

Example Using Programming Job

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

Libero log:

```
Software Version: 11.5.1.5
Opened 'D:/flashpro_files/m2s005_digest1/designer/a1_MSS/a1_MSS_fp/a1_MSS.pro'
The 'open_project' command succeeded.
PDB file
'D:\flashpro_files\m2s005_digest1\designer\al_MSS\83ce6816-1e56-496b-9e56-
d96ble56d96b.pdb' has
been loaded successfully.
DESIGN : a1_MSS; CHECKSUM : DE15; PDB_VERSION : 1.9
The 'load_programming_data' command succeeded.
Successfully exported STAPL file:
'D:\flashpro_files\m2s005_digest1\designer\al_MSS\export\al_MSS_M2S005.stp'; file
programs
Fabric and eNVM.
Fabric component digest:
276fbefb0a18cc0de1d45efc84589745ee02fc2adbcc1259fbeb674094754014
eNVM component digest:
6b2c2353e25c5982643c32640ac16c581874c8950300135622c126ee22d8b1de
Finished: Wed Jan 28 16:48:56 2015 (Elapsed time 00:00:06)
The 'export_single_stapl' command succeeded.
The 'set_programming_file' command succeeded.
Project saved.
The 'save_project' command succeeded.
Project closed.
```

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

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

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

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

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

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

See Also

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

Editable Constraints Grid

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

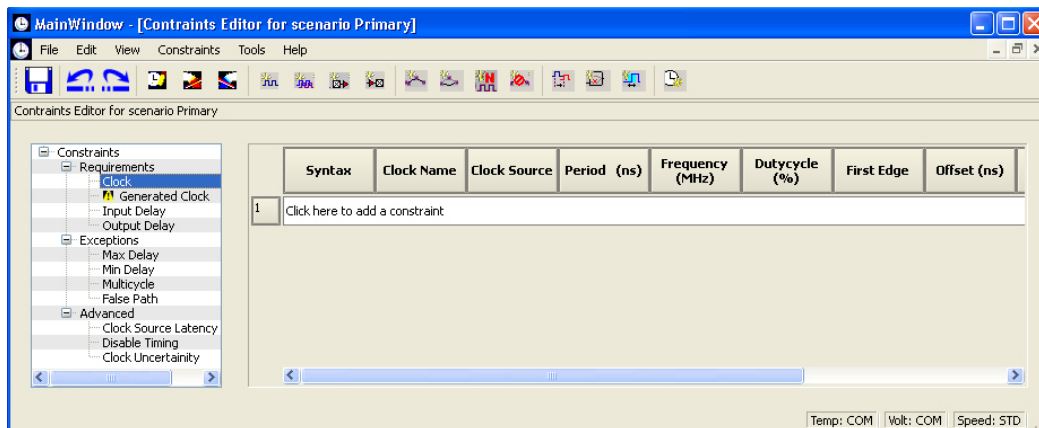


Figure 198 - Constraints Editor

To add a new constraint:

1. Select a constraint type from the constraint browser.
2. Enter the constraint values in the first row and click the green check mark to apply your changes. To cancel the changes press the red cancel mark.
3. The new constraint is added to the Constraint List. The green syntax flag indicates that the constraint was successfully checked.

To edit a constraint:

1. Select a constraint type from the constraint browser.

2. Select the constraint, edit the values and click the green check mark to apply your changes. To cancel the changes press the red cancel mark. The green syntax flag indicates that the constraint was successfully checked.

To delete a constraint:

1. Select a constraint type from the constraint browser.
2. Right-click the constraint you want to delete and choose **Delete Constraint**.

extended_run_lib - Libero SoC Only

Note: This is not a Tcl command; it is a shell script that can be run from the command line. The extended_run_lib Tcl script enables you to run the multiple pass layout in batch mode from a command line.

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

Note:

- There is no option to save the design files from all the passes. Only the (Timing or Power) result reports from all the passes are saved.
- This script supports only SmartFusion2, IGLOO2 and RTG4 designs.

Arguments

-root *path/designer/module_name*

The path to the root module located under the designer directory of the Libero project.

[-n *numPasses*]

Sets the number of passes to run. The default number of passes is 5.

[-starting_seed_index *numIndex*]

Indicates the specific index into the array of random seeds which is to be the starting point for the passes. Value may range from 1 to 100. If not specified, the default behavior is to continue from the last seed index that was used.

[-compare_criteria *value*]

Sets the criteria for comparing results between passes. The default value is set to frequency when the -c option is given or timing constraints are absent. Otherwise, the default value is set to violations.

Value	Description
frequency	Use clock frequency as criteria for comparing the results between passes. This option can be used in conjunction with the -c option (described below).
violations	Use timing violations as criteria for comparing the results between passes. This option can be used in conjunction with the -analysis, -slack_criteria and -stop_on_success options (described below).
power	Use total power as criteria for comparing the results between passes, where lowest total power is the goal.

[-c *clockName*]

Applies only when the clock frequency comparison criteria is used. Specifies the particular clock that is to be examined. If no clock is specified, then the slowest clock frequency in the design in a given pass is

used. The clock name should match with one of the Clock Domains in the Summary section of the Timing report.

`[-analysis value]`

Applies only when the timing violations comparison criteria is used. Specifies the type of timing violations (the slack) to examine. The following table shows the acceptable values for this argument:

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

`[-slack_criteria value]`

Applies only when the timing violations comparison criteria is used. Specifies how to evaluate the timing violations (slack). The type of timing violations (slack) is determined by the -analysis option. The following table shows the acceptable values for this argument:

Value	Description
worst	Sets the timing violations criteria to Worst slack. For each pass obtains the most amount of negative slack (or least amount of positive slack if all constraints are met) from the timing violations report. The largest value out of all passes will determine the best pass. This is the default.
tns	Sets the timing violations criteria to Total Negative Slack (tns). For each pass it obtains the sum of negative slack values from the first 100 paths from the timing violations report. The largest value out of all passes determines the best pass. If no negative slacks exist for a pass, then the worst slack is used to evaluate that pass.

`[-stop_on_success]`

Applies only when the timing violations comparison criteria is used. The type of timing violations (slack) is determined by the -analysis option. Stops running the remaining passes if all timing constraints have been met (when there are no negative slacks reported in the timing violations report).

`[-timing_driven|-standard]`

Sets layout mode to timing driven or standard (non-timing driven). The default is -timing_driven or the mode used in the previous layout command.

`[-power_driven value]`

Enables or disables power-driven layout. The default is off or the mode used in the previous layout command. The following table shows the acceptable values for this argument:

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

`[-placer_high_effort value]`

Sets placer effort level. The default is off or the mode used in the previous layout command. The following table shows the acceptable values for this argument:

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

Return

A non-zero value will be returned on error.

Supported Families

SmartFusion2, IGLOO2, RTG4

Exceptions

None

Example

```
D:/Libero_11_3_SP1/Designer/bin/libero
script:D:/Libero_11_3_SP1/Designer/scripts/extended_run_lib.tcl logfile:extended_run.log
"script_args:-root E:/designs/centralfpga/designer/centralfpga -n 3 -slack_criteria tns -
stop_on_success"
```

See Also

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

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

Files Tab and File Types

The Files tab displays all the files associated with your project, listed in the directories in which they appear.

Right-clicking a file in the Files tab provides a menu of available options specific to the file type. You can 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), 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

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)

viewdraw directory - viewdraw.ini files

How do I interpret data in the Flash Memory (NVM) Status Report?

The Embedded Flash Memory (NVM) Status Report generated from the FlashPro SmartDebug feature consists of the page status of each NVM page. For example:

```
Flash Memory Content [ Page 34 to 34 ]
FlashMemory Page #34:
Status Register(HEX): 00090000
Status ECC2 check: Pass
Data ECC2 Check: Pass
Write Count: Pass (2304 writes)
Total number of pages with status ECC2 errors: 0
Total number of pages with data ECC2 errors: 0
Total number of pages with write count out of range: 0
FlashMemory Check PASSED for [ Page 34 to 34 ]
The 'check_flash_memory' command succeeded.
The Execute Script command succeeded.
```

Table 12 - Embedded Flash Memory Status Report Description

Flash Memory Status Info	Description
Status Register (HEX)	Raw page status register captured from device
Status ECC2 Check	Check for ECC2 issue in the page status
Data ECC2 Check	Check for ECC2 issue in the page data
Write Count	Check if the page-write count is within the expected range.

Flash Memory Status Info	Description
	<p>The expected write count is greater than or equal to:</p> <p>6,384 - SmartFusion devices 2,288 - Fusion devices</p> <p>Note: Write count, if corrupted, cannot be reset to a valid value within the customer flow; invalid write count will not prevent device from being programmed with the FlashPro tool.</p> <p>The write count on all good eNVM pages is set to be 2288 instead of 0 in the manufacturing flow. The starting count of the eNVM is 2288. Each time the page is programmed or erased the count increments by one. There is a Threshold that is set to 12288, which equals to $3 * 4096$.</p> <p>Since the threshold can only be set in multiples of 4096 (2^{12}), to set a 10,000 limit, the Threshold is set to 12288 and the start count is set to 2288; and thus the eNVM has a 10k write cycle limit. After the write count exceeds the threshold, the STATUS bit goes to 11 when attempting to erase/program the page.</p>

Importing Files

Anything that describes your design, or is needed to program the device, is a project source. These may include schematics, HDL files, simulation files, testbenches, etc. Import these source files.

To import a file:

1. From the **File** menu, choose **Import Files**.
2. In **Files of type**, choose the file type.
3. In **Look in**, navigate to the drive/folder where the file is located.
4. Select the file to import and click **Open**.

Note: You cannot import a Verilog File into a VHDL project and vice versa.

File Types for Import

File Type	File Extension
ViewDraw Symbol	*.1-9
ViewDraw Schematic	*.1-9
Behavioral and Structural VHDL; VHDL Package	*.vhd, *.vhdl
Design Block Core	*.gen
Verilog Include	*.h
Behavioral and Structural Verilog	*.v
Stimulus	*.vhd, *.vhdl,

File Type	File Extension
	*.v
EDIF Netlist	*.edn
Memory file	*.mem
Components (Designer Blocks, Synplify DSP)	*.cxf

list_clock_groups

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

```
list_clock_groups
```

Arguments

None

Supported Families

SmartFusion2, IGLOO2, RTG4

Example

```
list_clock_groups
```

See Also

[set_clock_groups](#)

[remove_clock_groups](#)

Project Manager - Cores Dialog Box (Advanced Download Mode)

This dialog box (shown below) enables you to download cores from a [web repository](#) into a Vault.

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

The Catalog displays all the cores in your Vault.

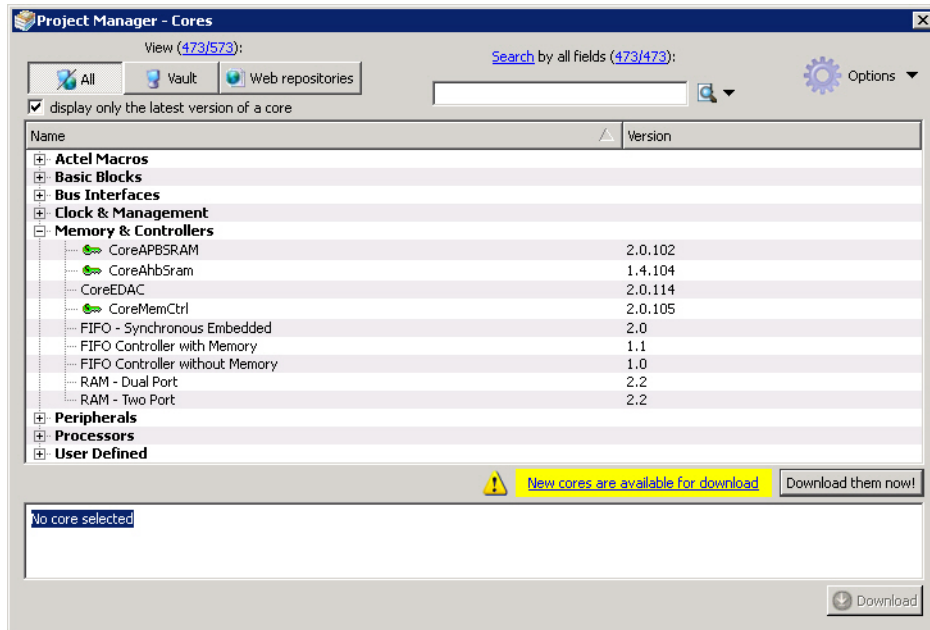


Figure 199 · Project Manager - Cores Dialog Box

Display only the latest version of a core is checked by default. This option, if checked, shows the latest versions of cores that are not in the Vault, and also filters out any duplicate cores that have the same Vendor, Library, and Name, with an earlier version number.

If the checkbox is de-selected the dialog box displays all cores, including those already loaded into the Vault. The status column indicates if a core has already been loaded.

Use the Filter to find any string that exists either in the core name or the Core Description. By default the filter contains a beginning and ending "*", so if you type 'controller' you get all cores with controller in the core name (case insensitive search) or in the core description.

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

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

Yellow Key - Has a limited or evaluation license only. Precompiled simulation libraries are provided, enabling the core to be instantiated and simulated within the Microsemi Libero SoC Using the Evaluation version of the core it is possible to create and simulate the complete design in which the core is being included. The design is not synthesizable (RTL code is not provided). No license feature in the license.dat file is needed to run the core in evaluation mode. You can purchase a license to generate an obfuscated or RTL netlist.

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

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

Project Settings Dialog Box

The Project Settings dialog box enables you to modify your Device, HDL, and Design Flow settings and your Simulation Options.

Device Selection

Sets the device Family, Die, and Package for your project. See the [New Project Creation Wizard - Device Settings](#) 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.

Design Flow

Preferred Language

Sets your HDL to VHDL or Verilog. You can import both Verilog and VHDL files into your project. However, the HDL files generated in the flow (such as the post-layout netlist for simulation) are created in the HDL (Verilog or VHDL) you set as the Preferred Language here.

When Verilog is selected, both System Verilog and Verilog 2001 (default) are supported. The default standard (Verilog 2001) is used when a project is first created for Verilog.

When VHDL is selected, both VHDL 2008 (default) and VHDL 93 are supported. The default standard (VHDL 2008) is used when a project is first created for VHDL.

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 [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.

Analysis Operating Conditions (For SmartFusion2, IGLOO2, and RTG4)

Sets the Operating Temperature Range (COM/IND/MIL/Custom), the Core Voltage Range (COM/IND/MIL/Custom) and Default I/O Voltage Range (COM/IND/MIL/Custom).

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

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.

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.

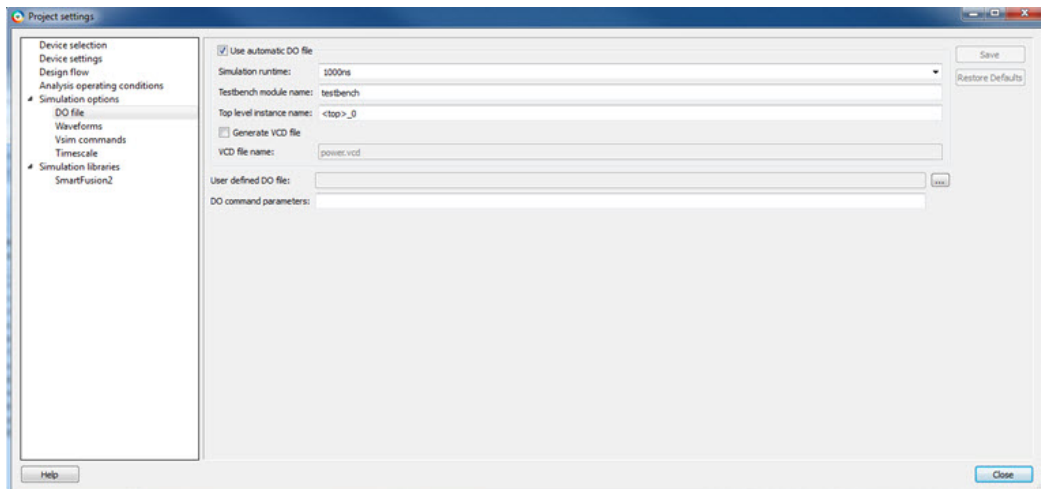


Figure 200 · Project Settings: - DO File

DO file

Use automatic DO file

Select if you want the Project Manager to automatically create a DO file that will enable you to simulate your design.

Simulation Run Time - Specify how long the simulation should run. If the value is 0, or if the field is empty, there will not be a run command included in the run.do file.

Testbench module name - Specify the name of your testbench entity name. Default is “testbench,” the value used by WaveFormer Pro.

Top Level instance name - Default is <top_0>, the value used by WaveFormer Pro. The Project Manager replaces <top> with the actual top level macro when you run simulation (presynth/postsynth/postlayout).

Generate VCD file - Click the checkbox to generate a VCD file.

VCD file name - Specifies the name of your generated VCD file. The default is power.vcd; click power.vcd and type to change the name.

User defined DO file - Enter the DO file name or click the browse button to navigate to it.

DO command parameters - Text in this field is added to the DO command.

Waveforms

Include DO file - Including a DO file enables you to customize the set of signal waveforms that will be displayed in ModelSim.

Display waveforms for - You can display signal waveforms for either the top-level testbench or for the design under test. If you select **top-level testbench** then Project Manager outputs the line 'add wave /testbench/*' in the DO file run.do. If you select **DUT** then Project Manager outputs the line 'add wave /testbench/DUT/*' in the run.do file.

Log all signals in the design - Saves and logs all signals during simulation.

Vsim Commands

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.

remove_clock_groups

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

```
remove_clock_groups [-id id# | -name groupname] \  
[-physically_exclusive | -logically_exclusive | -asynchronous]
```

Note: The exclusive flag is not needed when removing a clock group by ID.

Arguments

`-id id#`

Specifies the clock group by the ID.

`-name groupname`

Specifies the clock group by name (to be always followed by the exclusive flag).

`[-physically_exclusive | -logically_exclusive | - asynchronous]`

Supported Families

SmartFusion2, IGLOO2, RTG4

Example

Removal by group name

```
remove_clock_groups -name mygroup3 -physically_exclusive
```

Removal by group ID

```
remove_clock_groups -id 12
```

See Also

[set_clock_groups](#)

[list_clock_groups](#)

Running Libero SoC from your Software Tool Chain

When launched from your software toolchain, Libero SoC becomes solely an MSS configurator. This can be useful if you are responsible for the embedded code development for the SmartFusion device and are more comfortable in your existing software tool chain.

Any FPGA fabric development needs to be done using the regular Libero® SoC tool flow. Using the Libero SoC in the software toolchain mode only enables you to configure the SmartFusion Microcontroller Subsystem (MSS) and not the FPGA fabric.

The MSS Configurator can be integrated in any software development IDE that supports external tools. Configure your IDE to start the Libero SoC executable and use the parameters below to customize your interface. For SoftConsole, Keil and IAR the parameters are:

```
"PROJECT_LOCATION:<path>" //Project directory location, and the location of your generated MSS files.
```

```
"DESIGN_NAME:<name>" //Name of your design.
```

```
"STARTED_BY:<tool>" //Identifies which tool invoked the MSS Configurator; can be SoftConsole, Keil, or IAR EWARM
```

See Also

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

[Libero SoC Frequently Asked Questions](#)

[Software IDE Integration](#)

[View/Configure Firmware Cores](#)

Search in Libero SoC

Search options vary depending on your search type.

To find a file:

1. Use CTRL + F to open the Search window.

2. Enter the name or part of name of the object you wish to find in the Find field. '*' indicates a wildcard, and [*-] indicates a range, such as if you search for a1, a2, ... a5 with the string a[1-5].
3. Set the Options for your search (see below for list); options vary depending on your search type.
4. Click **Find All** (or **Next** if searching Text).

Searching an open text file, Log window or Reports highlights search results in the file itself. All other results appear in the Search Results window (as shown in the figure below).

Match case: Select to search for case-sensitive occurrences of a word or phrase. This limits the search so it only locates text that matches the upper- and lowercase characters you enter.

Match whole word: Select to match the whole word only.

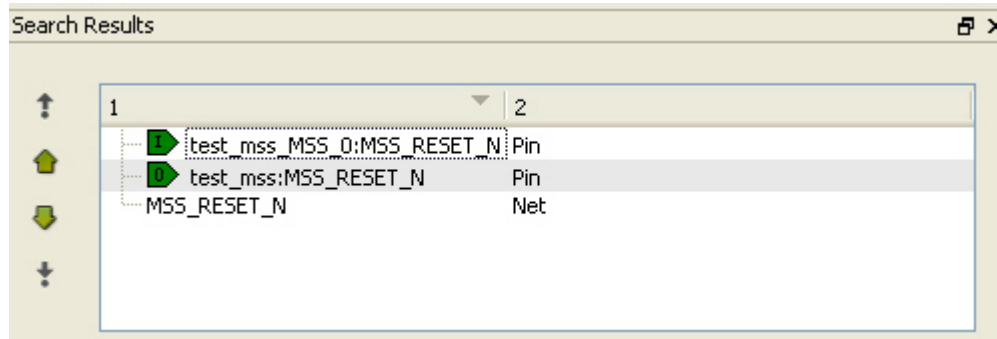


Figure 201 · Search Results

Current Open SmartDesign

Searches your open SmartDesign, returns results in the Search window.

Type: Choose Instance, Net or Pin to narrow your search.

Query: Query options vary according to Type.

Type	Query Option	Function
Instance	Get Pins	Search restricted to all pins
	Get Nets	Search restricted to all nets
	Get Unconnected Pins	Search restricted to all unconnected pins
Net	Get Instances	Searches all instances
	Get Pins	Search restricted to all pins
Pin	Get Connected Pins	Search restricted to all connected pins
	Get Associated Net	Search restricted to associated nets
	Get All Unconnected Pins	Search restricted to all unconnected pins

Current Open Text Editor

Searches the open text file. If you have more than one text file open you must place the cursor in it and click CTRL + F to search it.

Find All: Highlights all finds in the text file.

Next: Proceed to next instance of found text.

Previous: Proceed to previous instance of found text.

Replace with: Replaces the text you searched with the contents of the field.

Replace: Replaces a single instance.

Replace All: Replaces all instances of the found text with the contents of the field.

Design Hierarchy

Searches your Design Hierarchy; results appear in the Search window.

Find All: Displays all finds in the Search window.

Stimulus Hierarchy

Searches your Stimulus Hierarchy; results appear in the Search window.

Find All: Displays all finds in the Search window.

Log Window

Searches your Log window; results are highlighted in the Log window - they do not appear in the Search Results window.

Find All: Highlights all finds in the Log window.

Next: Proceed to next instance of found text.

Previous: Proceed to previous instance of found text.

Reports

Searches your Reports; returns results in the Reports window.

Find All: Highlights all finds in the Reports window.

Next: Proceed to next instance of found text.

Previous: Proceed to previous instance of found text.

Files

Searches your local project file names for the text in the Search field; returns results in the Search window.

Find All: Lists all search results in the Search window.

Files on disk

Searches the files' content in the specified directory and subdirectories for the text in the Search field; returns results in the Search window.

Find All: Lists all finds in the Search window.

File type: Select a file type to limit your search to specific file extensions, or choose *.* to search all file types.

Select Generated Clock Reference Dialog Box

Use this dialog box to find and choose the generated clock reference pin from the list of available pins.

To open the Select Generated Clock Reference dialog box (shown below) from the SmartTime Constraints Editor, open the [Create Generated Clock Constraint Dialog Box](#) dialog box and click the **Browse** button for the **Clock Reference**.

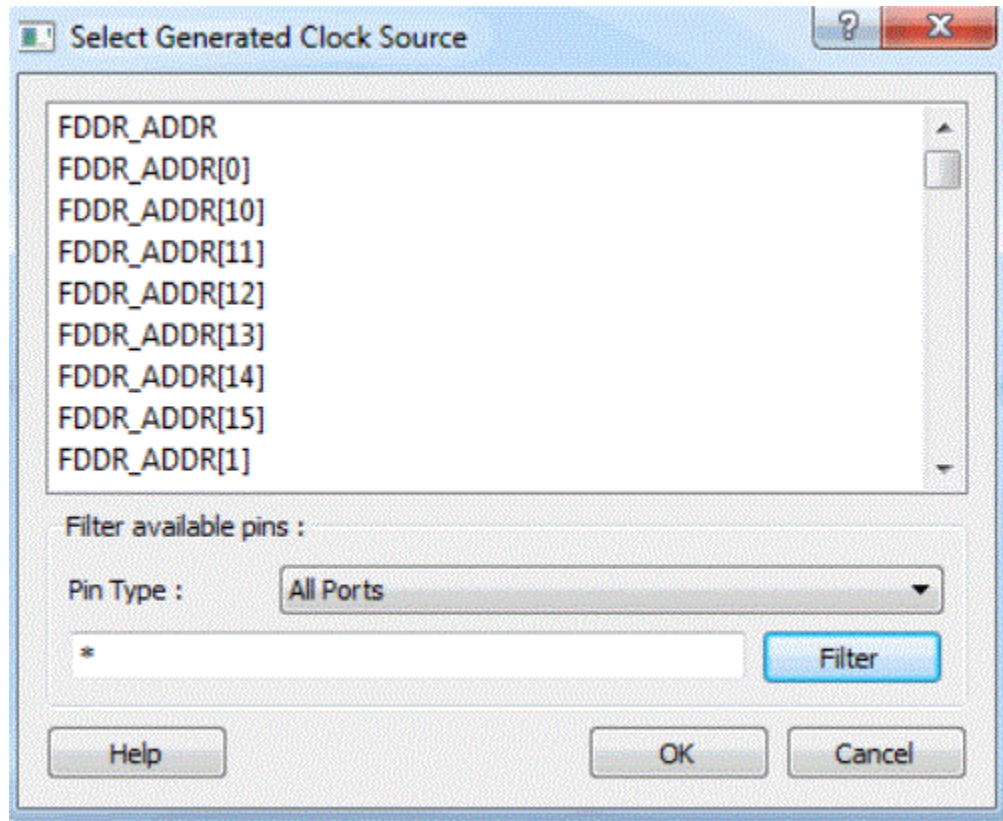


Figure 202 · Select Generated Clock Reference Dialog Box

Filter Available Pins

To identify any other pins in the design as the generated master pin, select **Filter available objects - Type** as **Clock Network**. You can also use the **Filter** to filter the generated reference clock pin name in the displayed list.

See Also

[Specifying generated clock constraints](#)

Select Generated Clock Source Dialog Box

Use this dialog box to find and choose the generated clock source from the list of available pins.

To open the Select Generated Clock Source dialog box (shown below) from the SmartTime Constraints Editor, open the [Create Generated Clock Constraint](#) dialog box and click the **Browse** button for the **Clock Pin**.

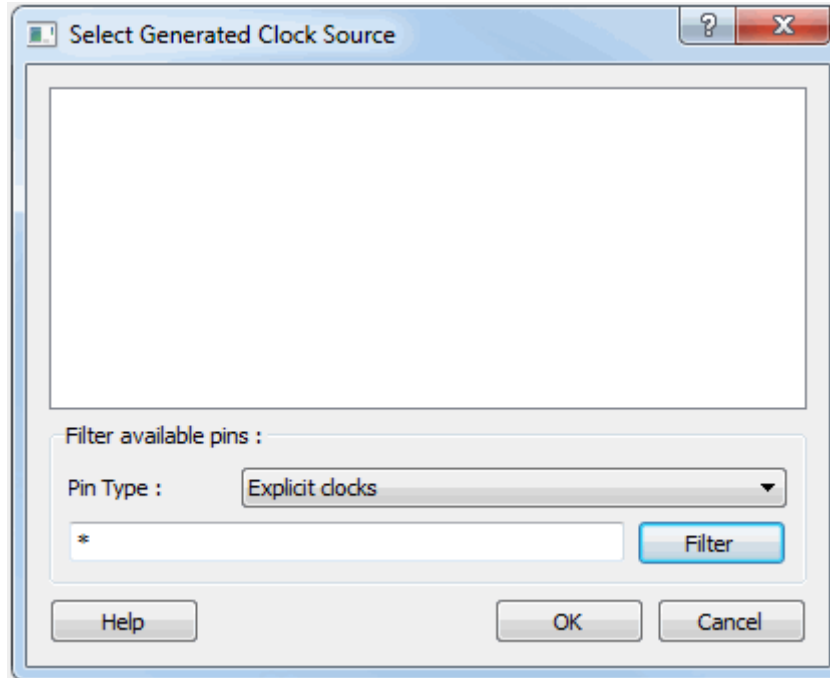


Figure 203 · Select Generated Clock Source Dialog Box

Filter Available Pins

Explicit clock pins for the design is the default value. To identify any other pins in the design as the generated clock source pins, from the **Pin Type** pull-down list, select **Explicit clocks**, **Potential clocks**, **All Ports**, **All Pins**, **All Nets**, **Pins on clock network**, or **Nets in clock network**. You can also use the **Filter** to filter the generated clock source pin name in the displayed list.

Select Source or Destination Pins for Constraint Dialog Box

Use this dialog box to select pins or ports:

- By explicit list
- By keyword and wildcard

To open the Select Source or Destination Pins for Constraint dialog box from the SmartTime Constraints Editor, right-click the Constraint Type in the Constraint Browser to open the Add Constraint Dialog Box. From this dialog box, click the **Browse** button to open the Select Source or Destination Pins for the Constraint dialog box.

By Explicit List

This is the default. This mode stores the actual pin names. The following figure shows an example dialog box for **Select Source Pins for Multicycle Constraint > by explicit list**.

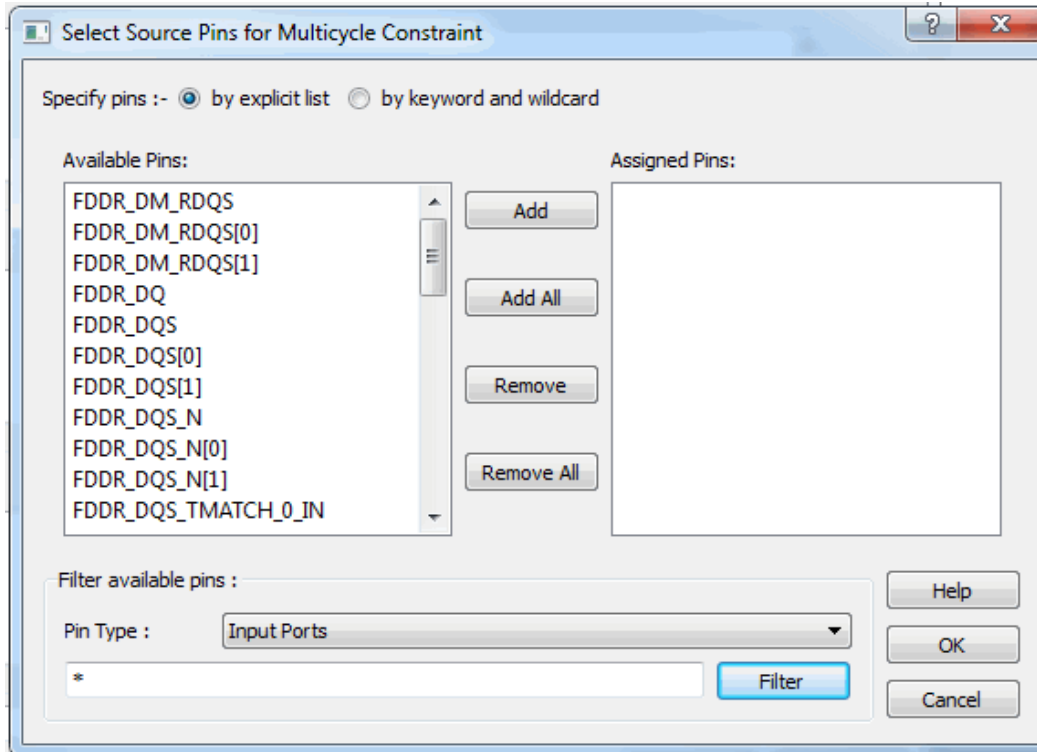


Figure 204 · Select Source Pins for Multicycle (Specify pins by explicit list) Dialog Box

Available Pins

The list box displays the available valid objects. If you change the filter value, the list box shows the available objects based on the filter.

Use **Add**, **Add All**, **Remove**, **Remove All** to add or delete pins from the **Assigned Pins** list.

Filter Available Pins

Pin type – Specifies the filter on the available object. This can be by **Explicit clocks**, **Potential clocks**, **Input ports**, **All Pins**, **All Nets**, **Pins on clock network**, or **Nets in clock network**

Filter

Specifies the filter based on which the **Available Pins** list shows the pin names. The default is *. You can specify any string value.

By Keyword and Wildcard

This mode stores the filter only. It does not store the actual pin names. The constraints created using this mode get exported with the SDC accessors (get_ports, get_pins, etc.). The following figure shows an example dialog box for **Select Source Pins for Multicycle Constraint > by keyword and wildcard**.

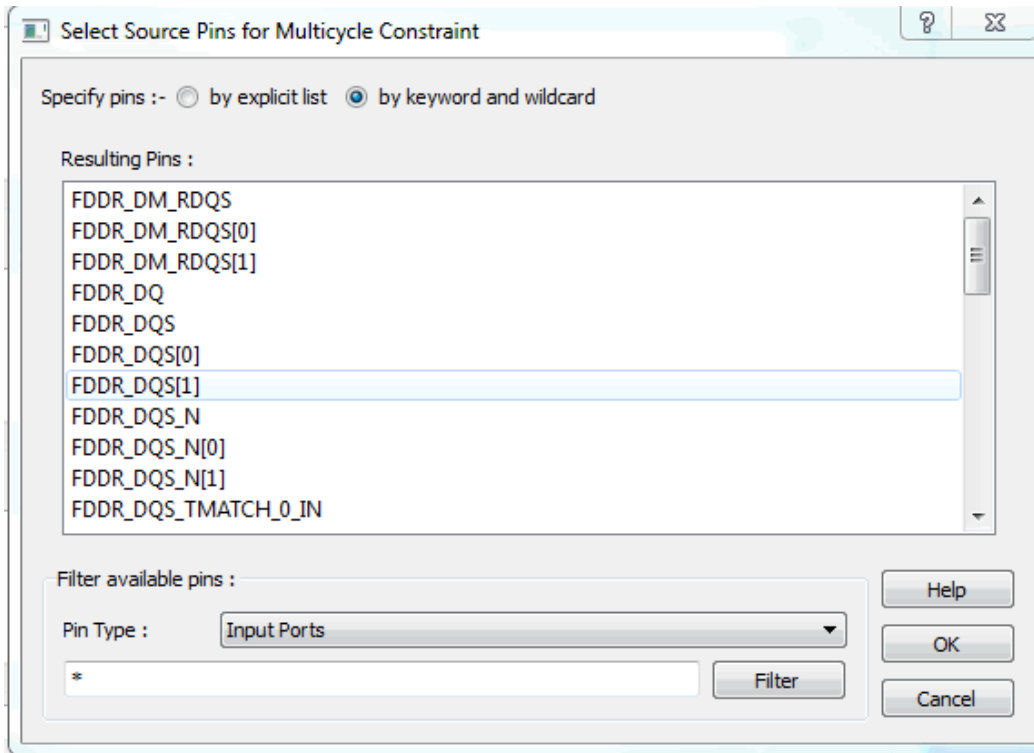


Figure 205 · Select Source Pins for Multicycle (Specify pins by keyword and wildcard) Dialog Box

Pin Type

Specifies the filter on the available pins. This can be Registers by pin names, Registers by clock names, Input Ports, or Output Ports. The default pin type is Registers by pin names.

Filter

Specifies the filter based on which the Available Pins list shows the pin names. The default is *. You can specify any string value.

Resulting Pins

Displays pins from the available pins based on the filter.

Select Source Pins for Clock Constraint Dialog Box

Use this dialog box to find and choose the clock source from the list of available pins.

To open the Select Source Pins for the Clock Constraint dialog box (shown below) from the SmartTime Constraints Editor, click the **Browse** button to the right of the Clock source field in the [Create Clock Constraint](#) dialog box.

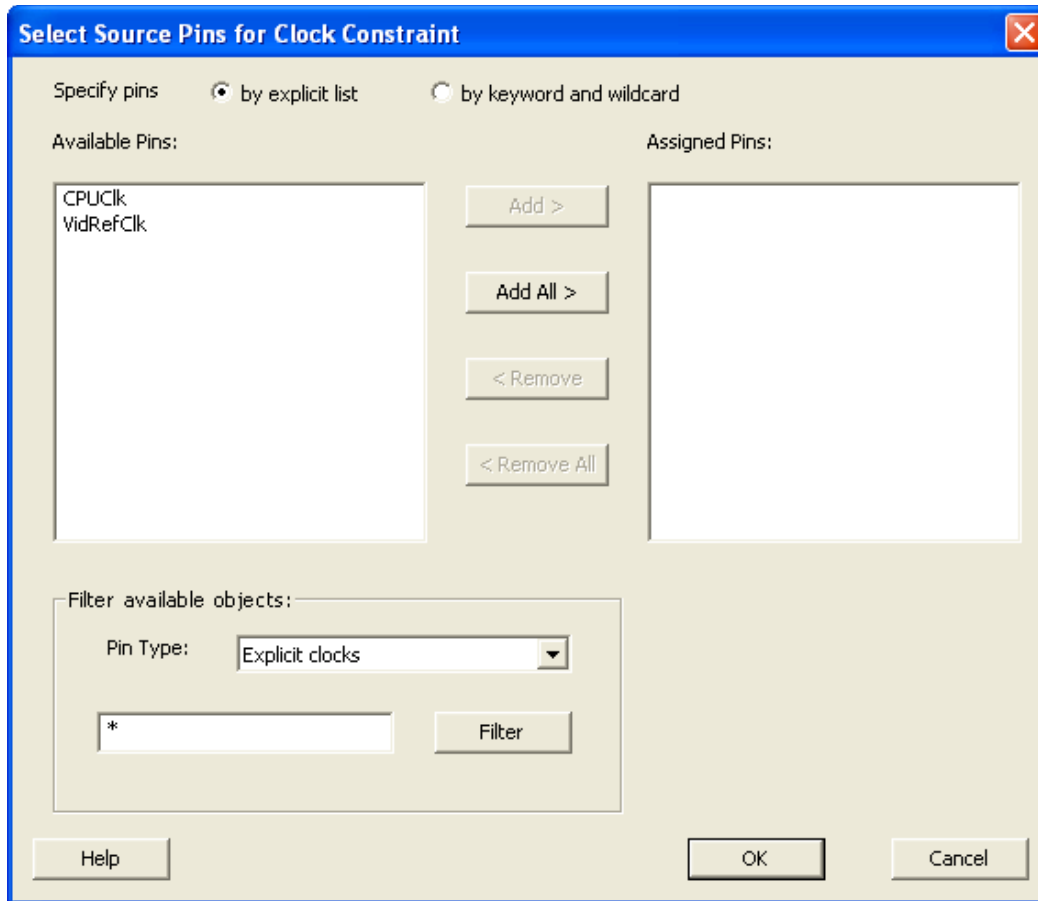


Figure 206 · Select Source Pins for Clock Constraint Dialog Box

Available Pins

Displays all available pins.

Filter Available Pins

Explicit clock pins for the design is the default value. To identify any other pins in the design as clock pins, right-click the **Pin Type** pull-down menu and select one of the following:

- Explicit clocks
- Potential clocks
- Input ports
- All Pins
- All Nets
- Pins on clock network
- Nets in clock network

You can also use the **Filter** to filter the clock source pin name in the displayed list.

See Also

[Specifying clock constraints](#)

Set a Disable Timing Constraint

Use disable timing constraint to specify the timing arcs to be disabled for timing consideration.

Note: This constraint is for the Place and Route tool and the Verify Timing tool. It is ignored by the Synthesis tool.

To specify a Disable Timing constraint, open the Set Constraint to Disable Timing Arcs dialog box in one of the following four ways:

- From the Constraints Browser, choose **Advanced > Disable Timing**.



- Double-click the Add Disable Timing Constraint icon.
- Choose **Disable Timing** from the Constraints drop-down menu (**Constraints > Disable Timing**).
- Right-click any row in the Disable Timing Constraints Table and choose **Add Constraint to Disable Timing**.

The Set Constraint to Disable Timing Arcs dialog box appears.

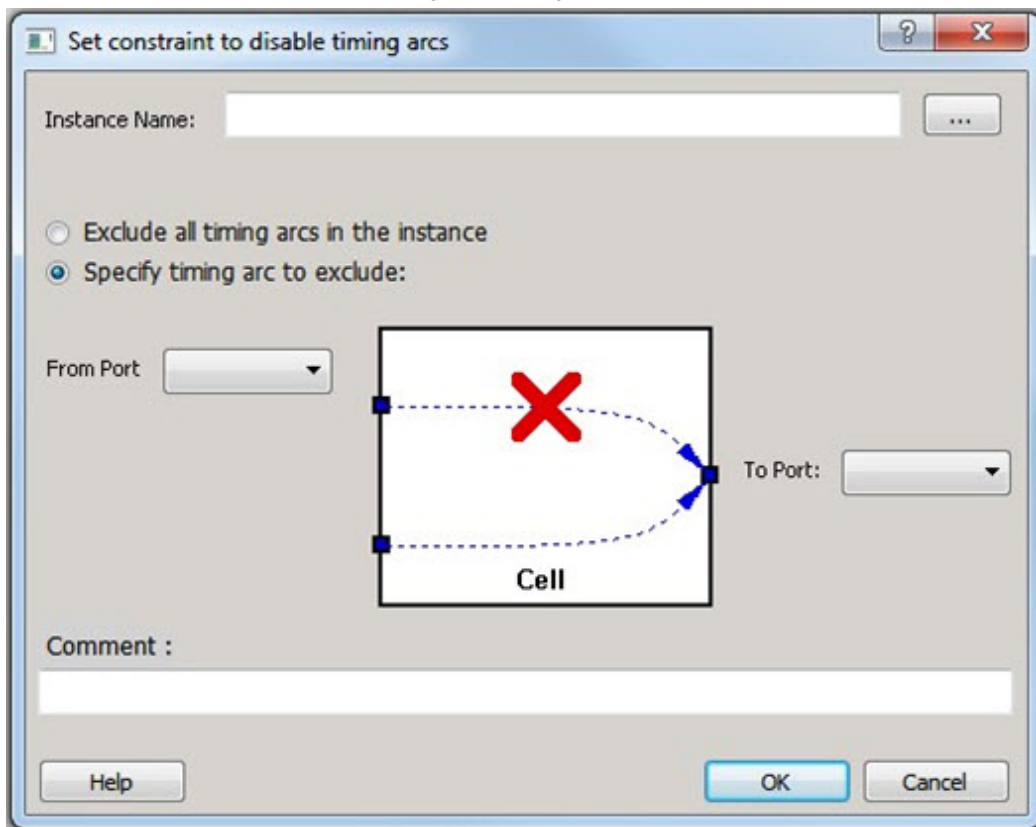


Figure 207 · Set Constraint to Disable Timing Arcs Dialog Box

Instance Name

Specifies the instance name for which the disable timing arc constraint will be created.

Click the browse button next to the Instance Name field to open the Select instance to constrain dialog box.

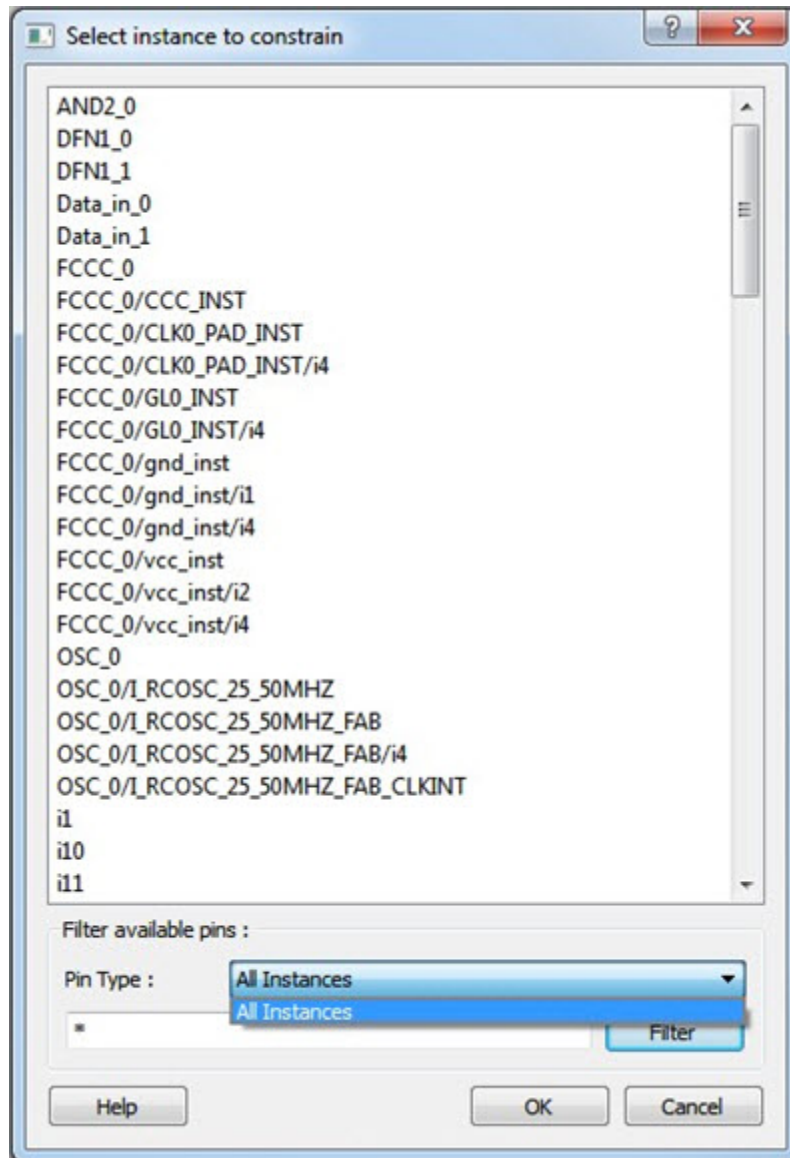


Figure 208 · Set Instance to Constrain Dialog Box

The Pin Type selection is limited to All Instances only.

Exclude All Timing Arcs in the Instance

This option enables you to exclude all timing arcs in the specified instance.

Specify Timing Arc to Exclude

This option enables you to specify the timing arc to exclude. In this case, you need to specify the from and to ports:

From Port

Specifies the starting point for the timing arc.

To Port

Specifies the ending point for the timing arc.

Comment

Enter a one-line comment for the constraint.

Set Clock Source Latency Dialog Box

Use this dialog box to define the delay between an external clock source and the definition pin of a clock within SmartTime.

To open the Set Clock Source Latency dialog box (shown below) from the Timing Analysis View, you must first [create a clock constraint](#). From the **Constraints** menu, choose **Clock Source Latency**.

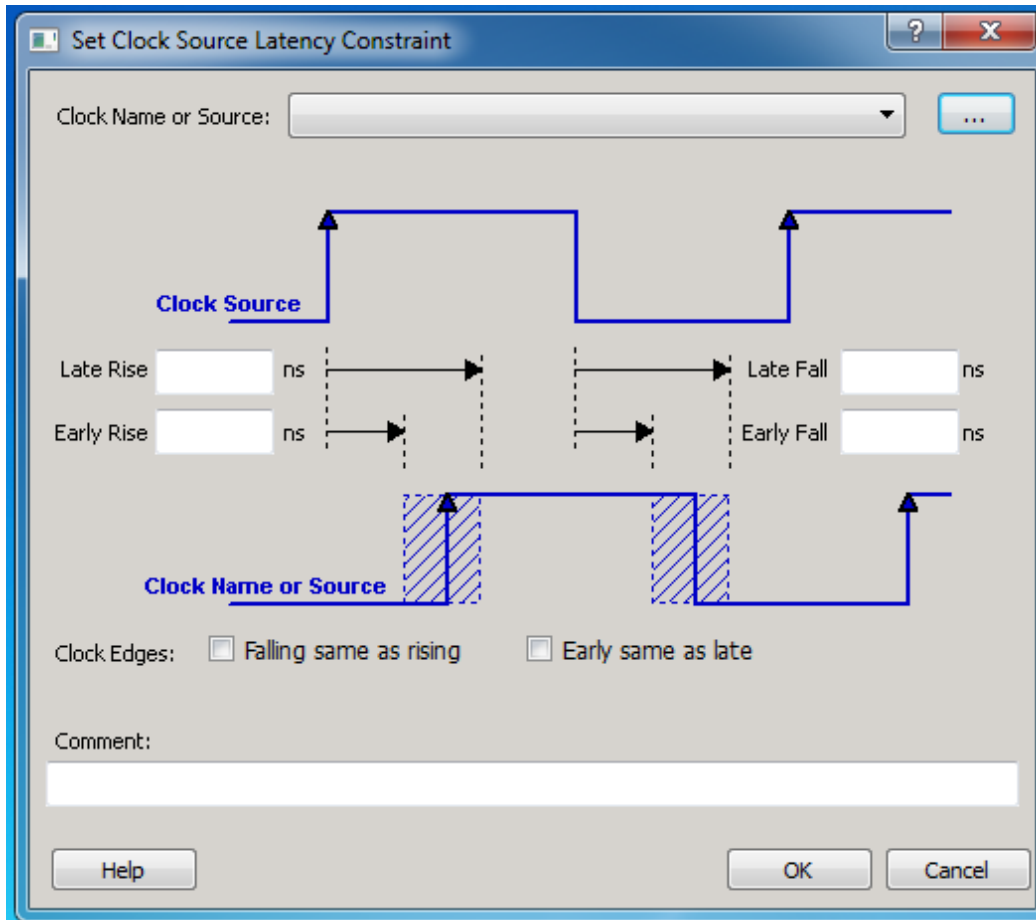


Figure 209 · Set Clock Source Latency Dialog Box

Clock Name or Source

Displays a list of clock ports or pins that do not already have a clock source latency specified. Select the clock name or source for which you are specifying the clock source latency.

Late Rise

Specifies the largest possible latency, in nanoseconds, of the rising clock edge at the clock port or pin selected, with respect to its source. Negative values are acceptable, but may lead to overly optimistic analysis.

Early Rise

Specifies the smallest possible latency, in nanoseconds, of the rising clock edge at the clock port or pin selected, with respect to its source. Negative values are acceptable, but may lead to overly optimistic analysis.

Late Fall

Specifies the largest possible latency, in nanoseconds, of the falling clock edge at the clock port or pin selected, with respect to its source. Negative values are acceptable, but may lead to overly optimistic analysis.

Early Fall

Specifies the smallest possible latency, in nanoseconds, of the falling clock edge at the clock port or pin selected, with respect to its source. Negative values are acceptable, but may lead to overly optimistic analysis.

Clock Edges

Select the latency for the rising and falling edges:

Falling same as rising: Specifies that Rising and Falling clock edges have the same latency.

Early same as late : Specifies that the clock source latency should be considered as a single value, not a range from "early" to "late".

Comment

Enables you to save a single line of text that describes the clock source latency.

See Also

[Specifying Clock Constraints](#)

Set Constraint to Disable Timing Arcs Dialog Box

Use this dialog box to specify the timing arcs being disabled to fix the combinational loops in the design.

To open the Set Constraint to Disable Timing Arcs dialog box (shown below) from the Timing Analysis View, from the **Constraints** menu, choose **Disable Timing**.

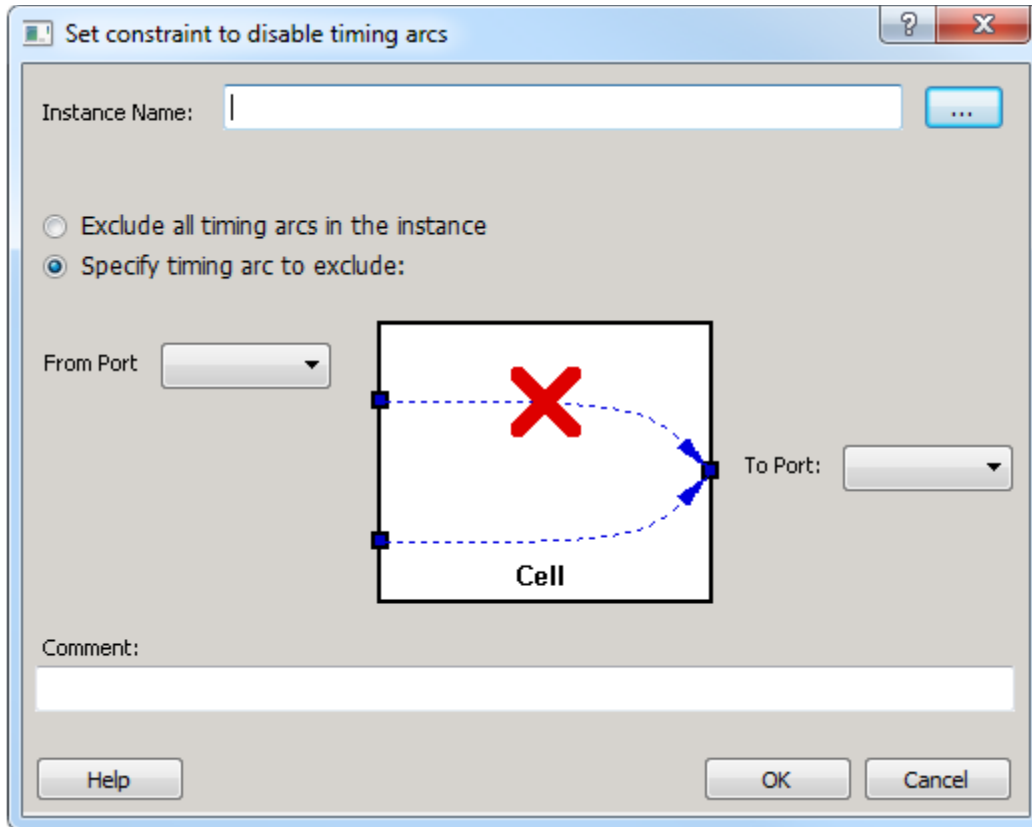


Figure 210 · Set Constraint to Disable Timing Arcs Dialog Box

Instance Name

Specifies the instance name for which the disable timing arc constraint will be created.

Exclude All Timing Arcs in the Instance

This option enables you to exclude all timing arcs in the specified instance.

Specify Timing Arc to Exclude

This option enables you to specify the timing arc to exclude. In this case, you need to specify the from and to ports:

From Port

Specifies the starting point for the timing arc.

To Port

Specifies the ending point for the timing arc.

Comment

Enables you to save a single line of text that describes the disable timing arc.

See Also

[Specifying Disable Timing Constraint](#)

Set False Path Constraint Dialog Box

Use this dialog box to define specific timing paths as being false.

This constraint removes timing requirements on these false paths so that they are not considered during the timing analysis. The path starting points are the input ports or register clock pins and path ending points are the register data pins or output ports. This constraint disables setup and hold checking for the specified paths.

Note: The false path information always takes precedence over multiple cycle path information and overrides maximum delay constraints.

To open the Set False Path Constraint dialog box (shown below) from the SmartTime Constraints Editor, choose **Constraints > False Path**.

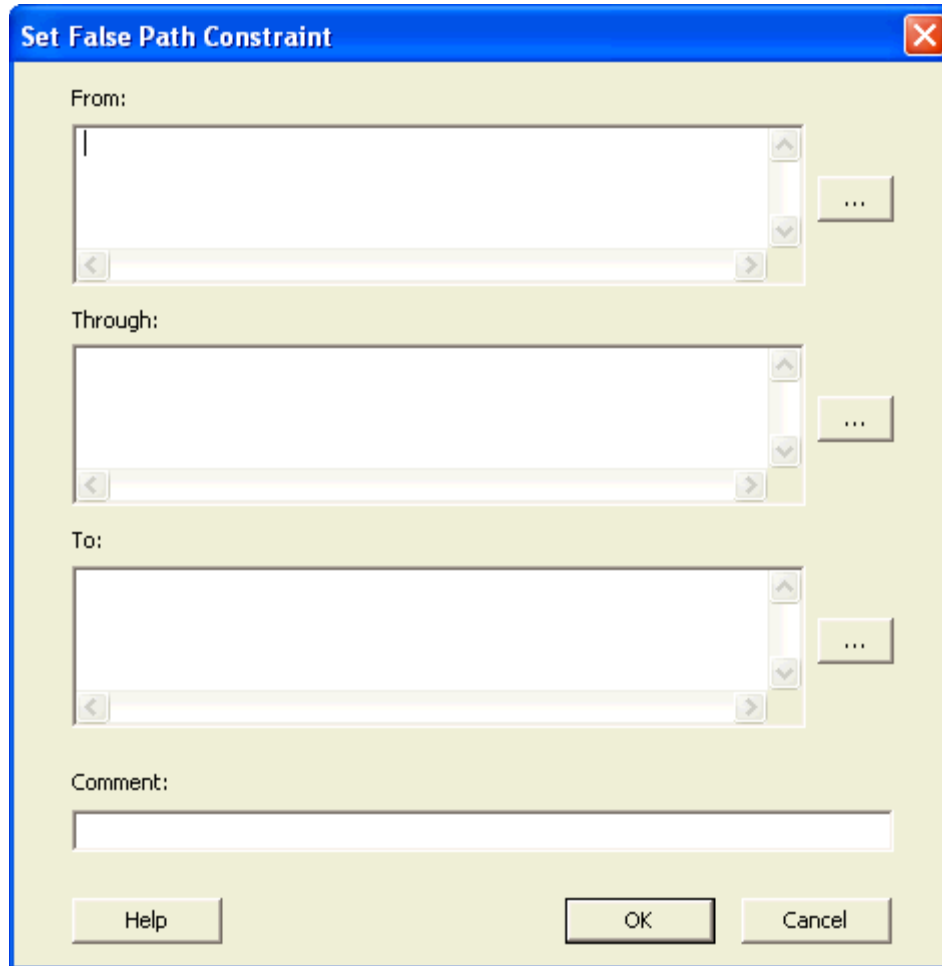


Figure 211 · Set False Path Constraint Dialog Box

From

Specifies the starting points for false path. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

Through

Specifies a list of pins, ports, cells, or nets through which the disabled paths must pass.

To

Specifies the ending points for false path. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

Comment

Enables you to provide comments for this constraint.

[See Also](#)

Set Maximum Delay Constraint Dialog Box

Use this dialog box to specify the required maximum delay for timing paths in the current design.

SmartTime automatically derives the individual maximum delay targets from clock waveforms and port input or output delays. So the maximum delay constraint is a timing exception. This constraint overrides the default single cycle timing relationship for one or more timing paths. This constraint also overrides a multiple cycle path constraint.

To open the Set Maximum Delay Constraint dialog box (shown below) from the SmartTime Constraints Editor, click the **Constraints** menu and choose **Max Delay** (**Constraints > Max Delay**).

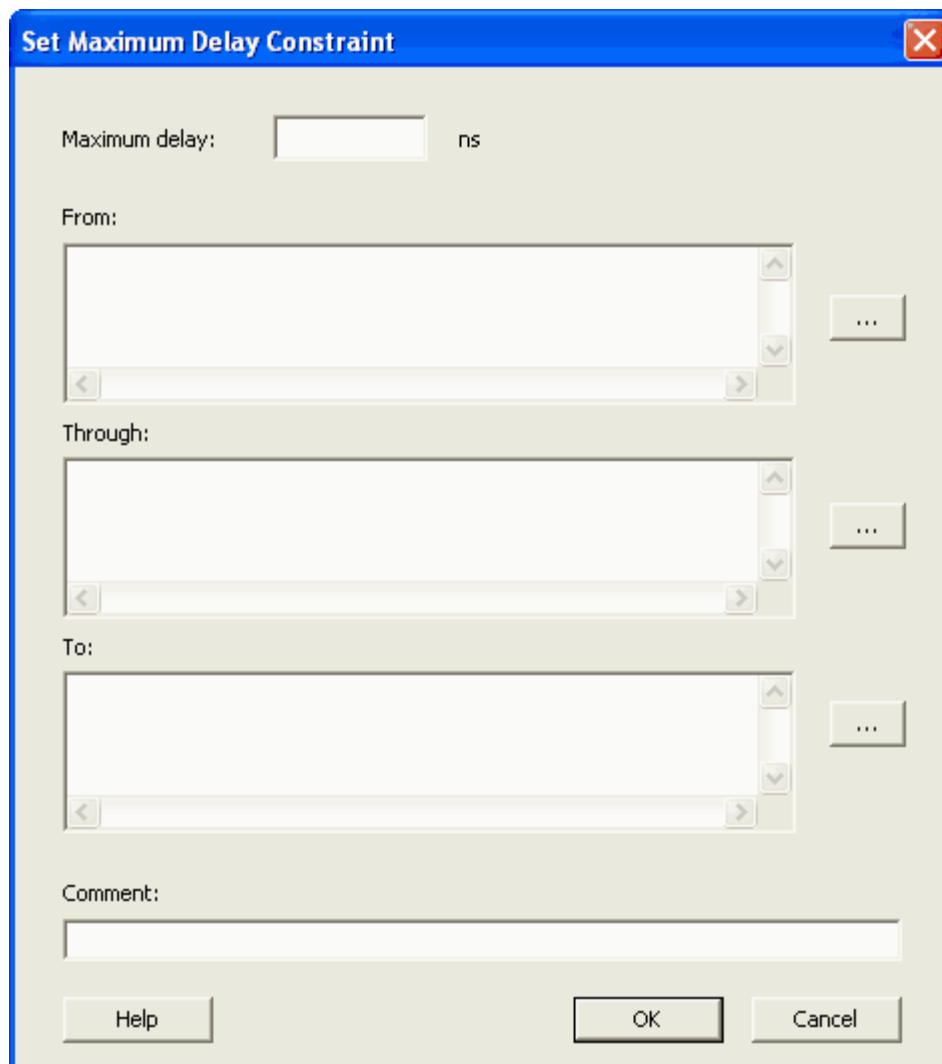


Figure 212 · Set Maximum Delay Constraint Dialog Box

Maximum Delay

Specifies a floating point number in nanoseconds that represents the required maximum delay value for specified paths.

If the path starting point is on a sequential device, SmartTime includes clock skew in the computed delay.

If the path starting point has an input delay specified, SmartTime adds that delay value to the path delay.

If the path ending point is on a sequential device, SmartTime includes clock skew and library setup time in the computed delay.

If the ending point has an output delay specified, SmartTime adds that delay to the path delay.

From

Specifies the starting points for max delay constraint. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

Through

Specifies the through points for the multiple cycle constraint.

To

Specifies the ending points for maximum delay constraint. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

Comment

Enables you to provide comments for this constraint.

See Also

[Specifying a Maximum Delay Constraint](#)

Set Minimum Delay Constraint Dialog Box

Use this dialog box to specify the required minimum delay for timing paths in the current design.

SmartTime automatically derives the individual minimum delay targets from clock waveforms and port input or output delays. So the minimum delay constraint is a timing exception. This constraint overrides the default single cycle timing relationship for one or more timing paths. This constraint also overrides a multiple cycle path constraint.

To open the Set Minimum Delay Constraint dialog box (shown below) from the SmartTime Constraints Editor, click the **Constraints** menu and choose **Min Delay** (**Constraints > Min Delay**).

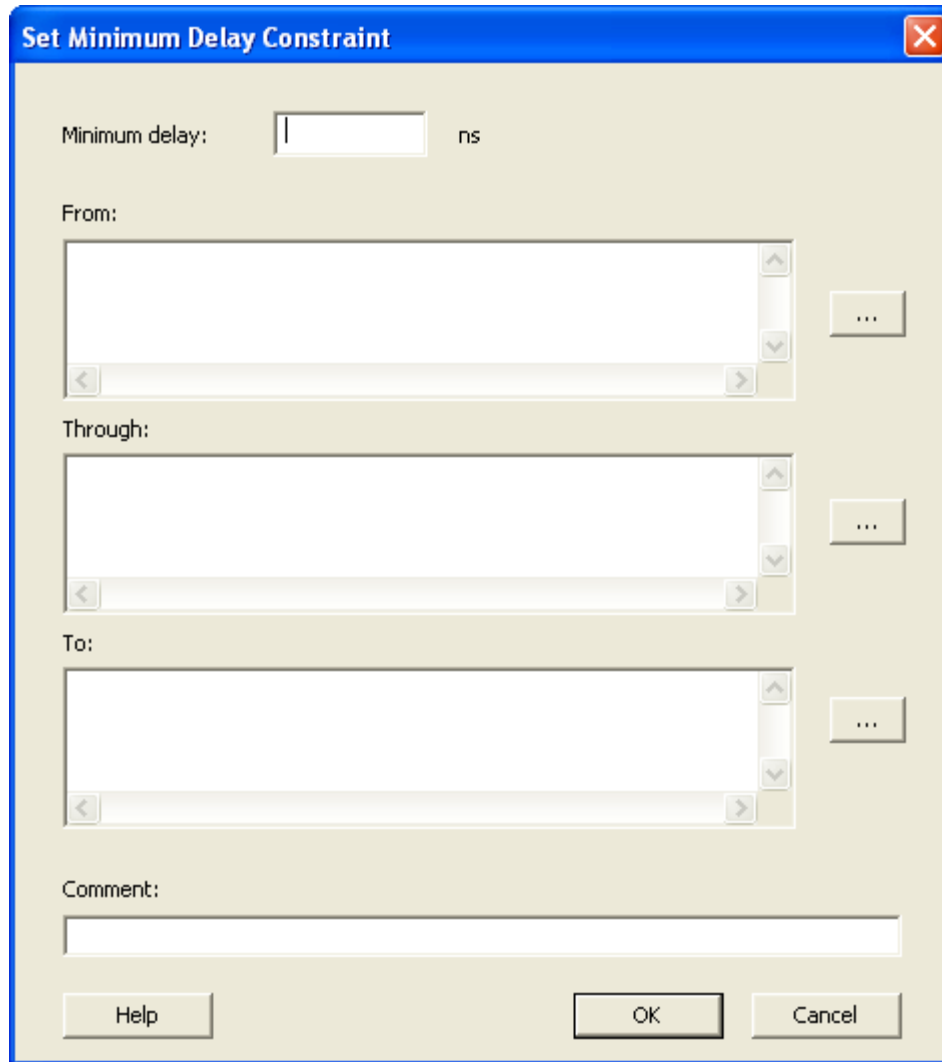


Figure 213 · Set Minimum Delay Constraint Dialog Box

Minimum Delay

Specifies a floating point number in nanoseconds that represents the required minimum delay value for specified paths.

If the path starting point is on a sequential device, SmartTime includes clock skew in the computed delay.

If the path starting point has an input delay specified, SmartTime adds that delay value to the path delay.

If the path ending point is on a sequential device, SmartTime includes clock skew and library setup time in the computed delay.

If the ending point has an output delay specified, SmartTime adds that delay to the path delay.

From

Specifies the starting points for minimum delay constraint. A valid timing starting point is a clock, a primary input, an input port, or a clock pin of a sequential cell.

Through

Specifies the through points for the multiple cycle constraint.

To

Specifies the ending points for minimum delay constraint. A valid timing ending point is a clock, a primary output, an input port, or a data pin of a sequential cell.

Comment

Enables you to provide comments for this constraint.

See Also

[Specifying a Minimum Delay Constraint](#)

Set Multicycle Constraint Dialog Box

Use this dialog box to specify the paths that take multiple clock cycles in the current design.

Setting the multiple-cycle paths constraint overrides the single-cycle timing relationships between sequential elements by specifying the number of cycles that the data path must have for setup or hold checks.

Note: When multiple timing constraints are set on the same timing path, the false path constraint has the highest priority and always takes precedence over multiple cycle path constraint. A specific maximum delay constraint overrides a general multicycle path constraint.

To open the Set Multicycle Constraint dialog box (shown below) from the SmartTime Constraints Editor, choose **Constraints > Multicycle**.

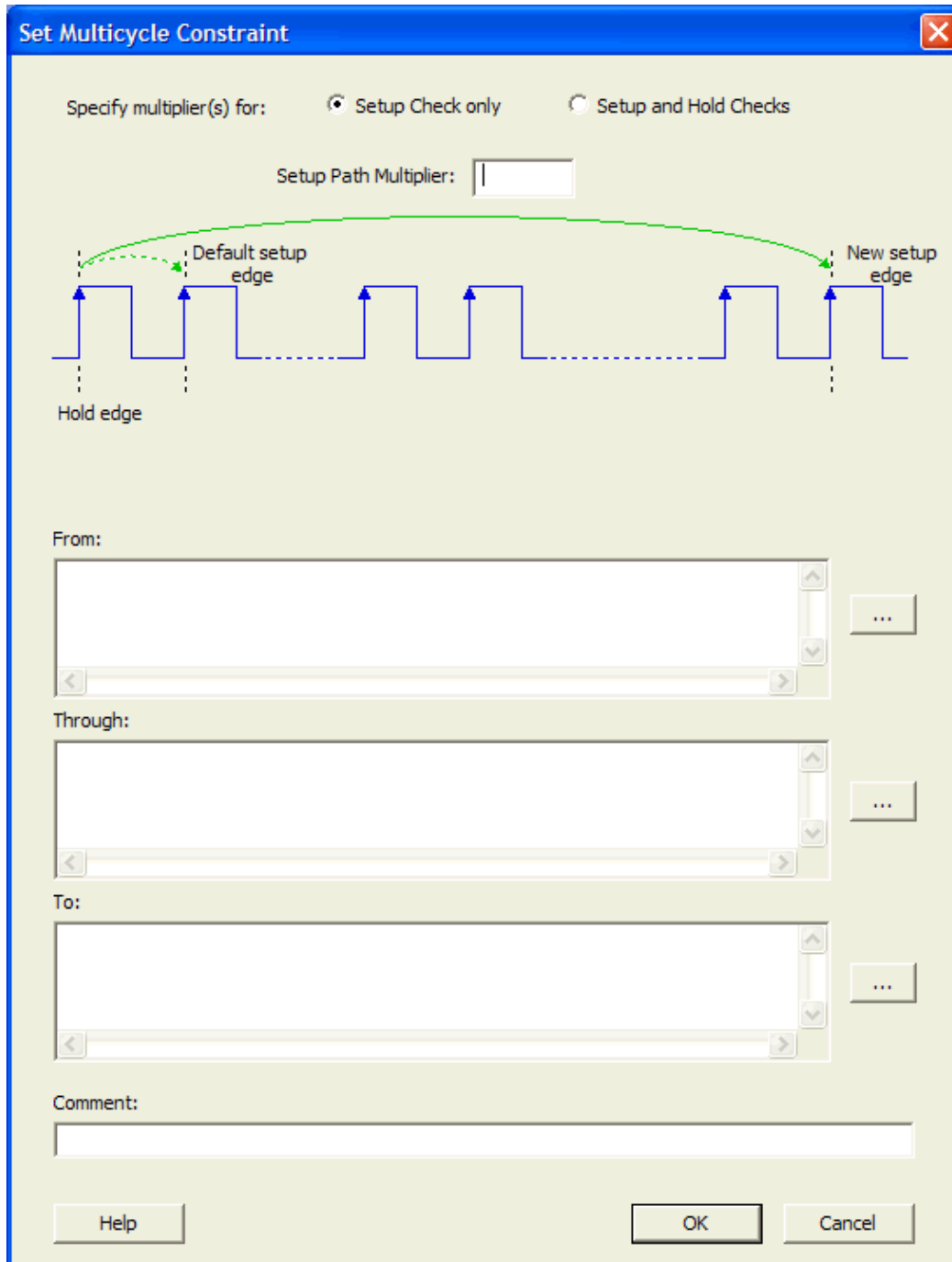


Figure 214 · Set Multicycle Constraint Dialog Box

Setup Path Multiplier

Specifies an integer value that represents a number of cycles the data path must have for a setup check. No hold check will be performed.

From

Specifies the starting points for the multiple cycle constraint. A valid timing starting point is a clock, a primary input, an inout port, or the clock pin of a sequential cell.

Through

Specifies the through points for the multiple cycle constraint.

To

Specifies the ending points for the multiple cycle constraint. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

Comment

Enables you to provide comments for this constraint.

When you select the Setup and Hold Checks option, an additional field appears in this dialog box: **Hold Path Multiplier**.

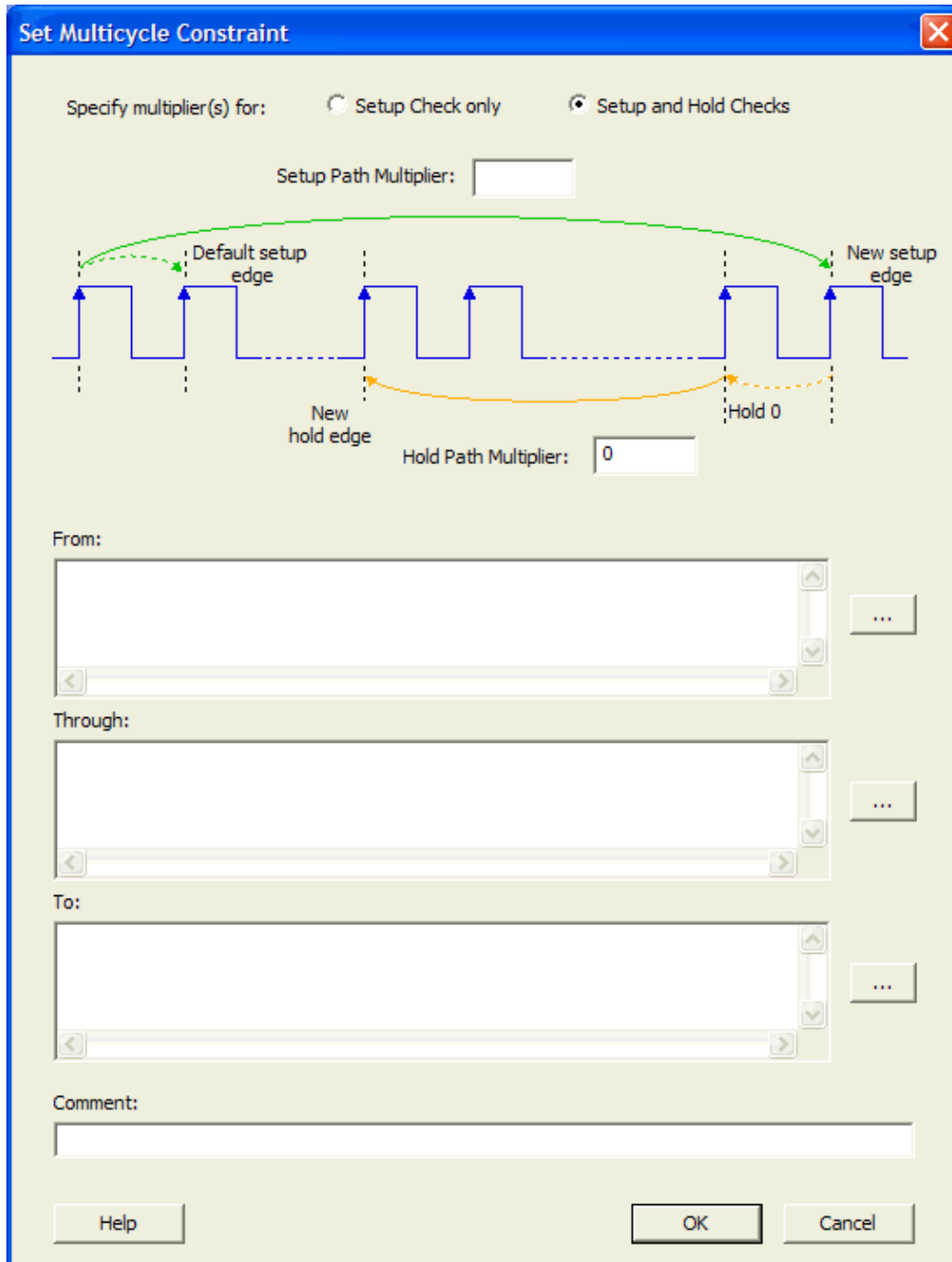


Figure 215 · Set Multicycle Constraint Dialog Box with Setup and Hold Checks Selected

Hold Path Multiplier

Specifies an integer value that represents a number of cycles the data path must have for a hold check, starting from one cycle before the setup check edge.

See Also

[Specifying a Multicycle Constraint](#)

set_clock_groups

set_clock_groups is an SDC command which disables timing analysis between the specified clock groups. No paths are reported between the clock groups in both directions. Paths between clocks in the same group continue to be reported.

```
set_clock_groups [-name name]  
                 [-physically_exclusive | -logically_exclusive | -asynchronous]  
                 [-comment comment_string]  
                 -group clock_list
```

Note: If you use the same name and the same exclusive flag of a previously defined clock group to create a new clock group, the previous clock group is removed and a new one is created in its place.

Arguments

-name *name*

Name given to the clock group. Optional.

-physically_exclusive

Specifies that the clock groups are physically exclusive with respect to each other. Examples are multiple clocks feeding a register clock pin. The exclusive flags are all mutually exclusive. Only one can be specified.

-logically_exclusive

Specifies that the clocks groups are logically exclusive with respect to each other. Examples are clocks passing through a mux.

-asynchronous

Specifies that the clock groups are asynchronous with respect to each other, as there is no phase relationship between them. The exclusive flags are all mutually exclusive. Only one can be specified.

Note: The exclusive flags for the arguments above are all mutually exclusive. Only one can be specified.

-group *clock_list*

Specifies a list of clocks. There can any number of groups specified in the set_clock_groups command.

Supported Families

SmartFusion2, IGLOO2, RTG4

Example

```
set_clock_groups -name mygroup3 -physically_exclusive \  
-group [get_clocks clk_1] -group [get_clocks clk_2]
```

See Also

[list_clock_groups](#)

[remove_clock_groups](#)

set_clock_to_output

SDC command; defines the timing budget available inside the FPGA for an output relative to a clock.

```
set_clock_to_output delay_value -clock clock_ref [-max] [-min] [-clock_fall] output_list
```

Arguments

delay_value

Specifies the clock to output delay in nanoseconds. This time represents the amount of time available inside the FPGA between the active clock edge and the data change at the output port.

-clock *clock_ref*

Specifies the reference clock to which the specified clock to output is related. This is a mandatory argument.

-max

Specifies that *delay_value* refers to the maximum clock to output at the specified output. If you do not specify -max or -min options, the tool assumes maximum and minimum clock to output delays to be equal.

-min

Specifies that *delay_value* refers to the minimum clock to output at the specified output. If you do not specify -max or -min options, the tool assumes maximum and minimum clock to output delays to be equal.

-clock_fall

Specifies that the delay is relative to the falling edge of the reference clock. The default is the rising edge.

output_list

Provides a list of output ports in the current design to which *delay_value* is assigned. If you need to specify more than one object, enclose the objects in braces ({}).

set_clock_uncertainty

Tcl command; specifies a clock-to-clock uncertainty between two clocks (from and to) and returns the ID of the created constraint if the command succeeded.

```
set_clock_uncertainty uncertainty -from | -rise_from | -fall_from from_clock_list -to | -  
rise_to | -fall_to to_clock_list -setup {value} -hold {value}
```

Arguments

uncertainty

Specifies the time in nanoseconds that represents the amount of variation between two clock edges.

-from

Specifies that the clock-to-clock uncertainty applies to both rising and falling edges of the source clock list. Only one of the -from, -rise_from, or -fall_from arguments can be specified for the constraint to be valid.

-rise_from

Specifies that the clock-to-clock uncertainty applies only to rising edges of the source clock list. Only one of the -from, -rise_from, or -fall_from arguments can be specified for the constraint to be valid.

-fall_from

Specifies that the clock-to-clock uncertainty applies only to falling edges of the source clock list. Only one of the -from, -rise_from, or -fall_from arguments can be specified for the constraint to be valid.

from_clock_list

Specifies the list of clock names as the uncertainty source.

-to

Specifies that the clock-to-clock uncertainty applies to both rising and falling edges of the destination clock list. Only one of the -to, -rise_to, or -fall_to arguments can be specified for the constraint to be valid.

-rise_to

Specifies that the clock-to-clock uncertainty applies only to rising edges of the destination clock list. Only one of the -to, -rise_to, or -fall_to arguments can be specified for the constraint to be valid.

-fall_to

Specifies that the clock-to-clock uncertainty applies only to falling edges of the destination clock list. Only one of the -to, -rise_to, or -fall_to arguments can be specified for the constraint to be valid.

to_clock_list

Specifies the list of clock names as the uncertainty destination.

-setup

Specifies that the uncertainty applies only to setup checks. If none or both -setup and -hold are present, the uncertainty applies to both setup and hold checks.

-hold

Specifies that the uncertainty applies only to hold checks. If none or both -setup and -hold are present, the uncertainty applies to both setup and hold checks.

Description

The `set_clock_uncertainty` command sets the timing uncertainty between two clock waveforms or maximum clock skew. Timing between clocks have no uncertainty unless you specify it.

Examples

```
set_clock_uncertainty 10 -from Clk1 -to Clk2
set_clock_uncertainty 0 -from Clk1 -fall_to { Clk2 Clk3 } -setup
set_clock_uncertainty 4.3 -fall_from { Clk1 Clk2 } -rise_to *
set_clock_uncertainty 0.1 -rise_from [ get_clocks { Clk1 Clk2 } ] -fall_to { Clk3 Clk4 }
-setup
set_clock_uncertainty 5 -rise_from Clk1 -to [ get_clocks { * } ]
```

set_external_check

SDC command; defines the external setup and hold delays for an input relative to a clock.

```
set_external_check delay_value -clock clock_ref [-setup] [-hold] [-clock_fall] input_list
```

Arguments

delay_value

Specifies the external setup or external hold delay in nanoseconds. This time represents the amount of time available inside the FPGA for the specified input after a clock edge.

-clock *clock_ref*

Specifies the reference clock to which the specified external check is related. This is a mandatory argument.

-setup

Specifies that *delay_value* refers to the setup check at the specified input. This is a mandatory argument if -hold is not used. You must specify either -setup or -hold option.

-clock_fall

Specifies that the delay is relative to the falling edge of the reference clock. The default is the rising edge.

input_list

Provides a list of input ports in the current design to which *delay_value* is assigned. If you need to specify more than one object, enclose the objects in braces ({}).

Supported Families

SmartFusion2, IGLOO2, RTG4, SmartFusion, IGLOO, ProASIC3, Fusion

Description

The `set_external_check` command specifies the external setup and hold times on input ports relative to a clock edge. This usually represents a combinational path delay from the input port to the clock pin of a register internal to the current design. For in/out (bidirectional) ports, you can specify the path delays for both input and output modes. The tool uses external setup and external hold times for paths starting at primary inputs.

A clock is a singleton that represents the name of a defined clock constraint. This can be an object accessor that will refer to one clock. For example:

```
[get_clocks {system_clk}]  
[get_clocks {sys*_clk}]
```

Examples

The following example sets an external setup check of 12 ns and an external hold check of 6 ns for port `data_in` relative to the rising edge of `CLK1`:

```
set_external_check 12 -clock [get_clocks CLK1] -setup [get_ports data_in]  
set_external_check 6 -clock [get_clocks CLK1] -hold [get_ports data_in]
```

See Also

[SDC Syntax Conventions](#)

`set_min_delay`

SDC command; specifies the minimum delay for the timing paths.

```
set_min_delay delay_value [-from from_list] [-to to_list]
```

Arguments

delay_value

Specifies a floating point number in nanoseconds that represents the required minimum delay value for specified paths.

- If the path starting point is on a sequential device, the tool includes clock skew in the computed delay.
- If the path starting point has an input delay specified, the tool adds that delay value to the path delay.
- If the path ending point is on a sequential device, the tool includes clock skew and library setup time in the computed delay.
- If the ending point has an output delay specified, the tool adds that delay to the path delay.

-from *from_list*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

-to *to_list*

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

Supported Families

SmartFusion2, IGLOO2, RTG4, SmartFusion, IGLOO, ProASIC3, Fusion

Description

This command specifies the required minimum delay for timing paths in the current design. The path length for any startpoint in *from_list* to any endpoint in *to_list* must be less than *delay_value*.

The tool automatically derives the individual minimum delay targets from clock waveforms and port input or output delays. For more information, refer to the [create_clock](#), [set_input_delay](#), and [set_output_delay](#) commands.

The minimum delay constraint is a timing exception. This constraint overrides the default single cycle timing relationship for one or more timing paths. This constraint also overrides a multicycle path constraint.

Examples

The following example sets a minimum delay by constraining all paths from *ff1a:CLK* or *ff1b:CLK* to *ff2e:D* with a delay less than 5 ns:

```
set_min_delay 5 -from {ff1a:CLK ff1b:CLK} -to {ff2e:D}
```

The following example sets a minimum delay by constraining all paths to output ports whose names start by "out" with a delay less than 3.8 ns:

```
set_min_delay 3.8 -to [get_ports out*]
```

Microsemi Implementation Specifics

The `-through` option in the `set_min_delay` SDC command is not supported.

See Also

[SDC Syntax Conventions](#)

Organize Source Files

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**.

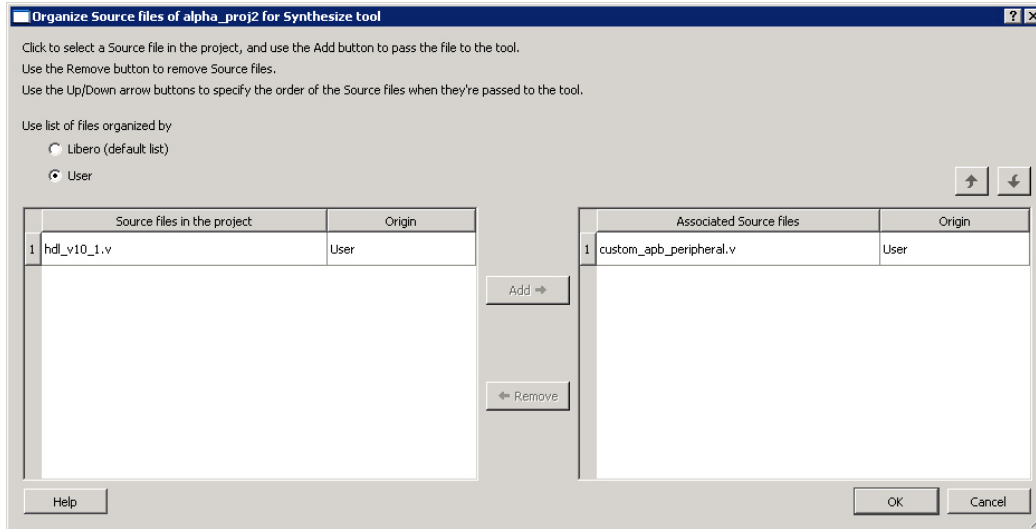


Figure 216 · Organize Source Files Dialog Box

Specify I/O States During Programming Dialog Box

The I/O States During Programming dialog box enables you to specify [custom settings](#) for I/Os in your programming file. This is useful if you want to set an I/O to drive out specific logic, or if you want to use a custom I/O state to manage settings for each Input, Output Enable, and Output associated with an I/O.

Load from file

Load from file enables you to load an I/O Settings (*.ios) file. You can use the IOS file to import saved custom settings for all your I/Os. The exported IOS file have the following format:

- Used I/Os have an entry in the IOS file with the following format:

```
set_prog_io_state -portName {<design_port_name>} -input <value> -outputEnable <value> -output <value>
```

- Unused I/Os have an entry in the IOS file with the following format:

```
set_prog_io_state -pinNumber {<device_pinNumber>} -input <value> -outputEnable <value> -output <value>
```

Where <value> is:

- 1 – I/O is set to drive out logic High
- 0 – I/O is set to drive out logic Low
- Last_Known_State: I/O is set to the last value that was driven out prior to entering the programming mode, and then held at that value during programming
- Z - Tri-State: I/O is tristated

Save to file

Saves your I/O Settings File (*.ios) for future use. This is useful if you set custom states for your I/Os and want to use them again later in conjunction with a PDC file.

Port Name

Lists the names of all the ports in your design.

Macro Cell

Lists the I/O type, such as INBUF, OUTBUF, PLLs, etc.

Pin Number

The package pin associate with the I/O.

I/O State (Output Only)

Your custom I/O State set during programming. This heading changes to Boundary Scan Register if you select the BSR Details checkbox; see the [Specifying I/O States During Programming - I/O States and BSR Details](#) help topic for more information on the BSR Details option.

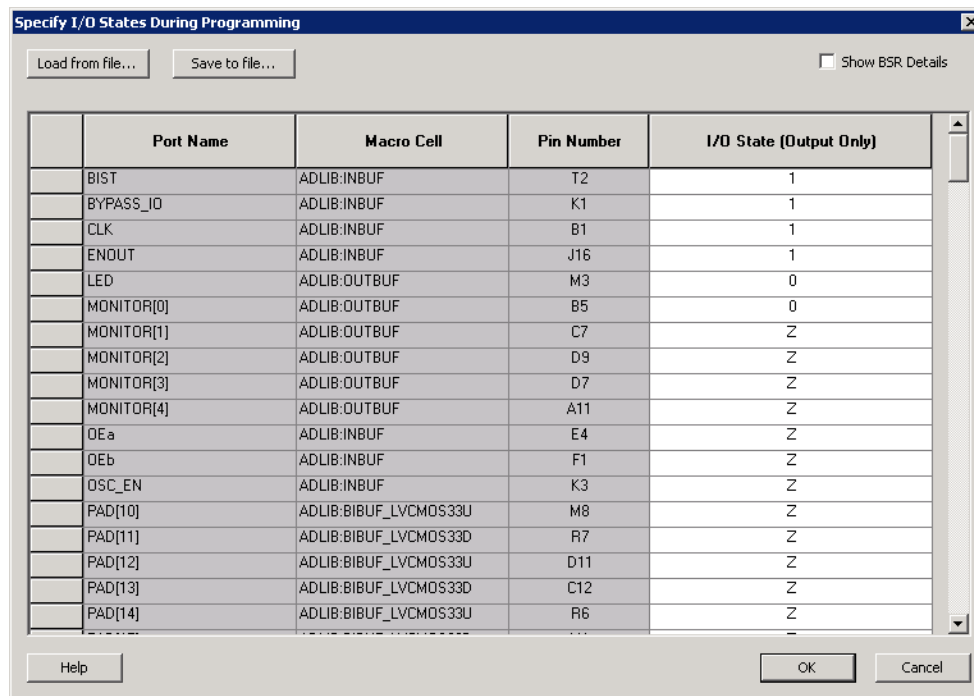



Figure 217 · I/O States During Programming Dialog Box

Specifying a False Path Constraint

You set options in the [Set False Path Constraint](#) dialog box to define specific timing paths as false.

To specify False Path constraints:

1. Add the constraint in the [Editable Constraints Grid](#) or open the [Set False Path Constraint](#) dialog box. You can do this by using one of the following methods:
 - From the SmartTime **Constraints** menu, choose **False Path**.
 - Click the  icon.
 - Right-click **False Path** in the Constraint Browser and choose **Add False Path Constraint**.

The Set False Path Constraint dialog box appears (as shown below).

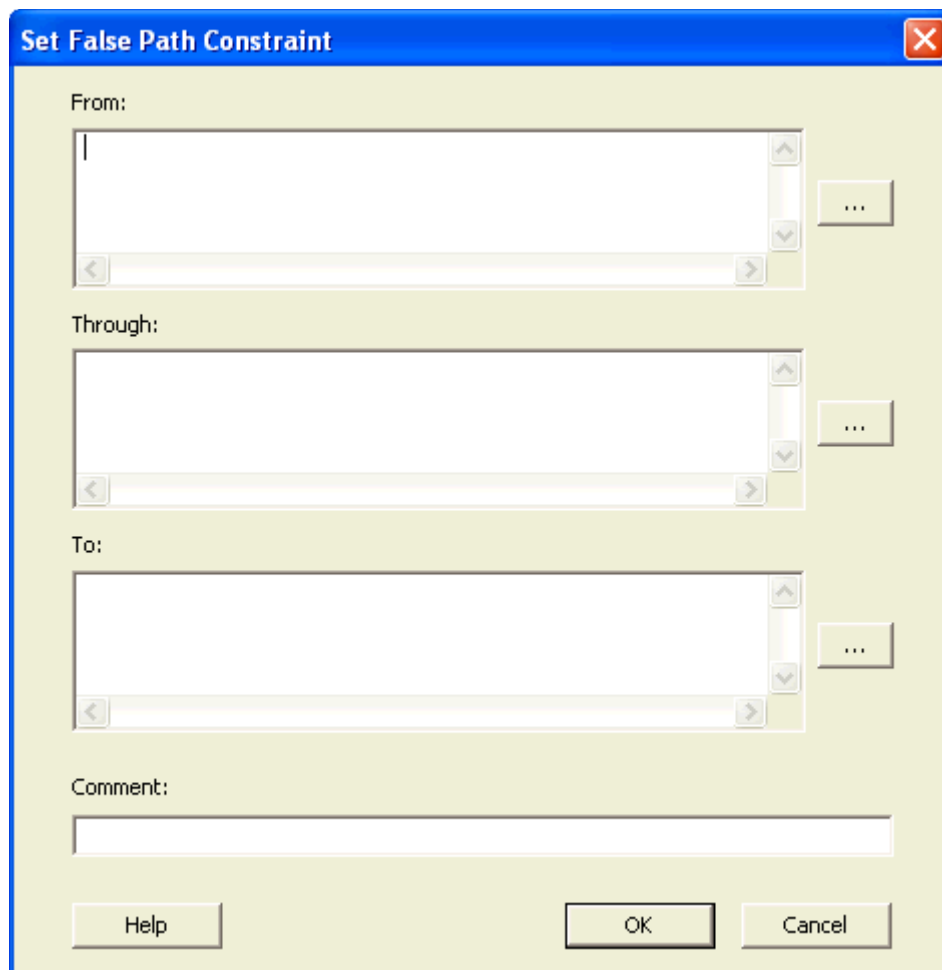


Figure 218 · Set False Path Constraint Dialog Box

2. Specify the **From** pin(s). Click the **Browse** button next to **From** to open the Select Source Pins for False Path Constraint dialog box (as shown below).

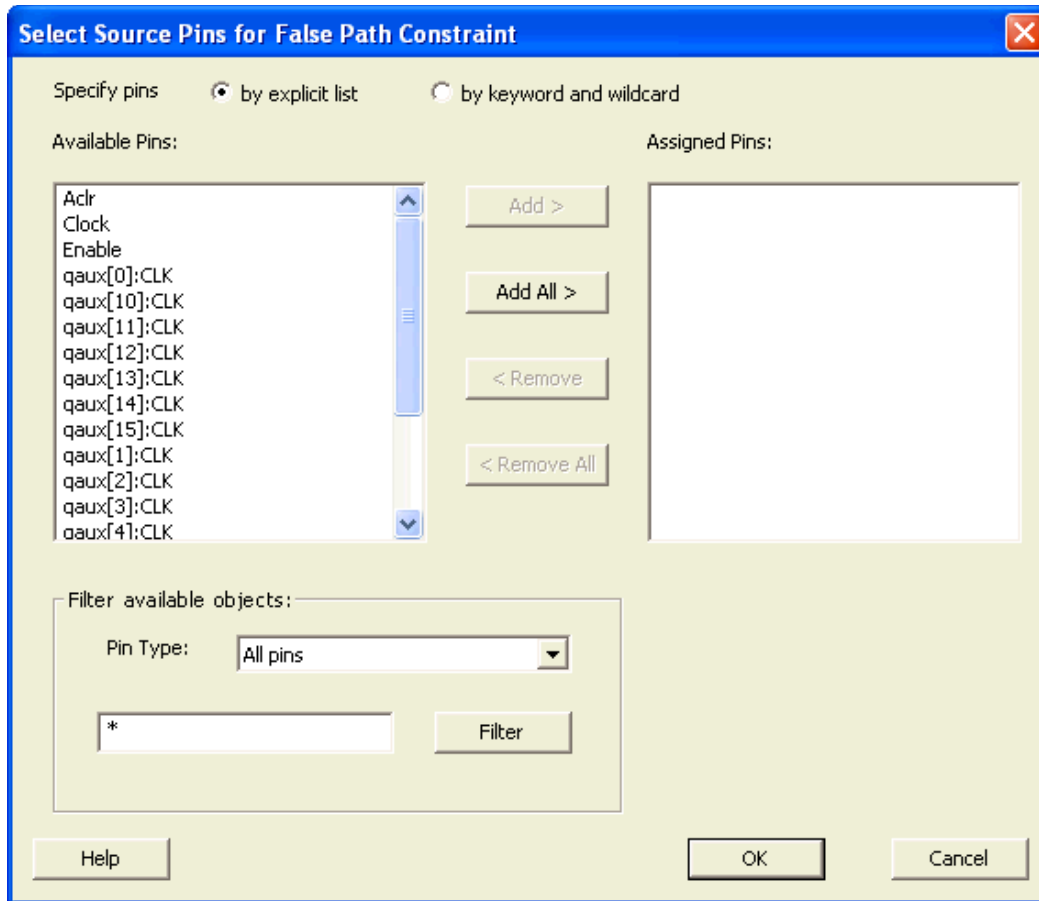


Figure 219 · Select Source Pins for False Path Constraint Dialog Box

3. Select **by explicit list**. (Alternatively, you can select **by keyword and wildcard**. For details, refer to [Select Source or Destination Pins for Constraint Dialog Box](#).)
4. Select the input pin(s) from the **Available Pin** list. You can use **Filter available objects** to narrow the pin list. You can select multiple ports in this window.
5. Click **Add** or **Add All**. The input pin(s) move from the **Available Pins** list to the **Assigned Pins** list.
6. Click **OK**.

The [Set False Path Constraint](#) dialog box displays the updated representation of the **From** pin(s).

7. Click the **Browse** button for **Through** and **To** and add the appropriate pin(s). The displayed list shows the pins reachable from the previously selected pin(s) list.
8. Enter comments in the **Comment** section.
9. Click **OK**.


SmartTime adds the False Path constraints to the Constraints List in the SmartTime Constraints Editor.

Specifying a Maximum Delay Constraint

You set options in the [Set Maximum Delay Constraint](#) dialog box to relax or to tighten the original clock constraint requirement on specific paths.

To specify Max delay constraints:

1. Add the constraint in the [Editable Constraints Grid](#) or open the [Set Maximum Delay Constraint](#) dialog box using one of the following methods:

- Click the  icon in the Constraints Editor.
- From the Constraints Editor, right-click the Constraints menu and choose **Max delay**.

The Set Maximum Delay Constraint dialog box appears (as shown below).

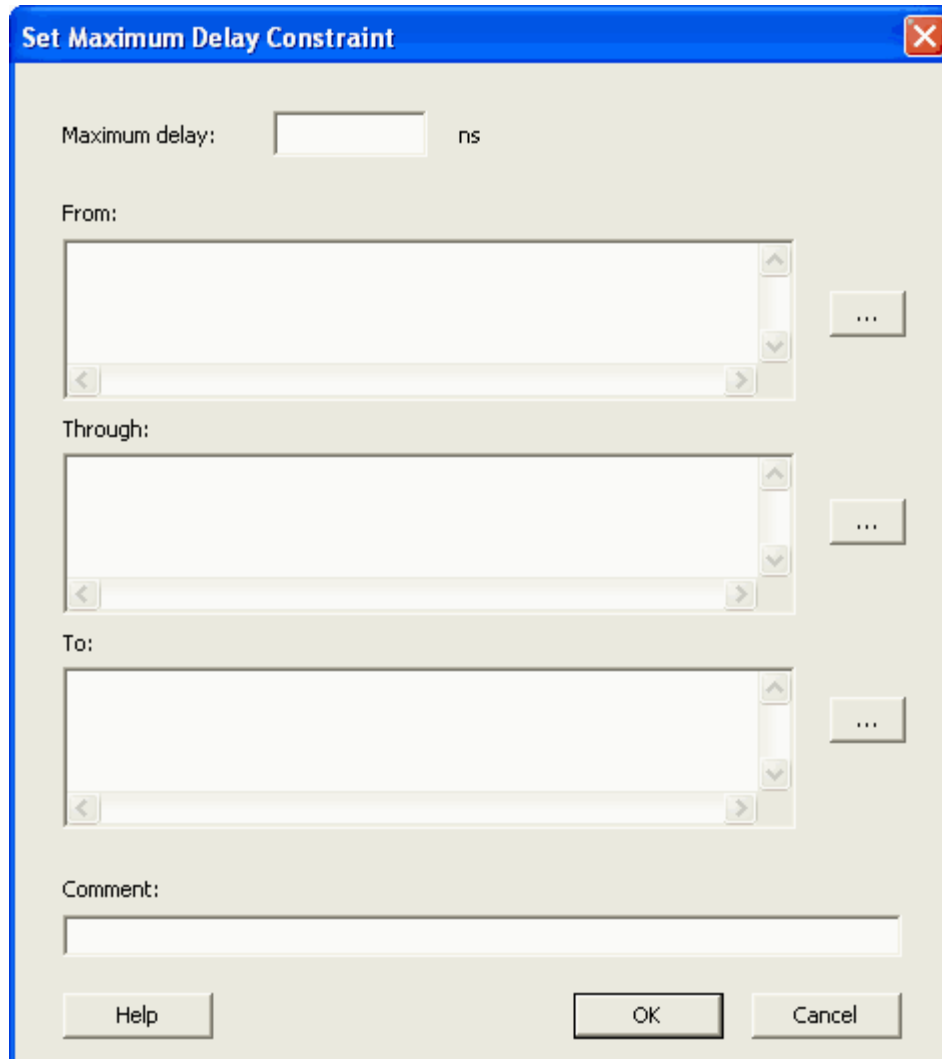


Figure 220 · Set Maximum Delay Constraint Dialog Box

2. Specify the delay in the **Maximum delay** field.
3. Specify the **From** pin(s). Click the **Browse** button next to **From** to open the Select Source Pins for Max Delay Constraint dialog box (as shown below).

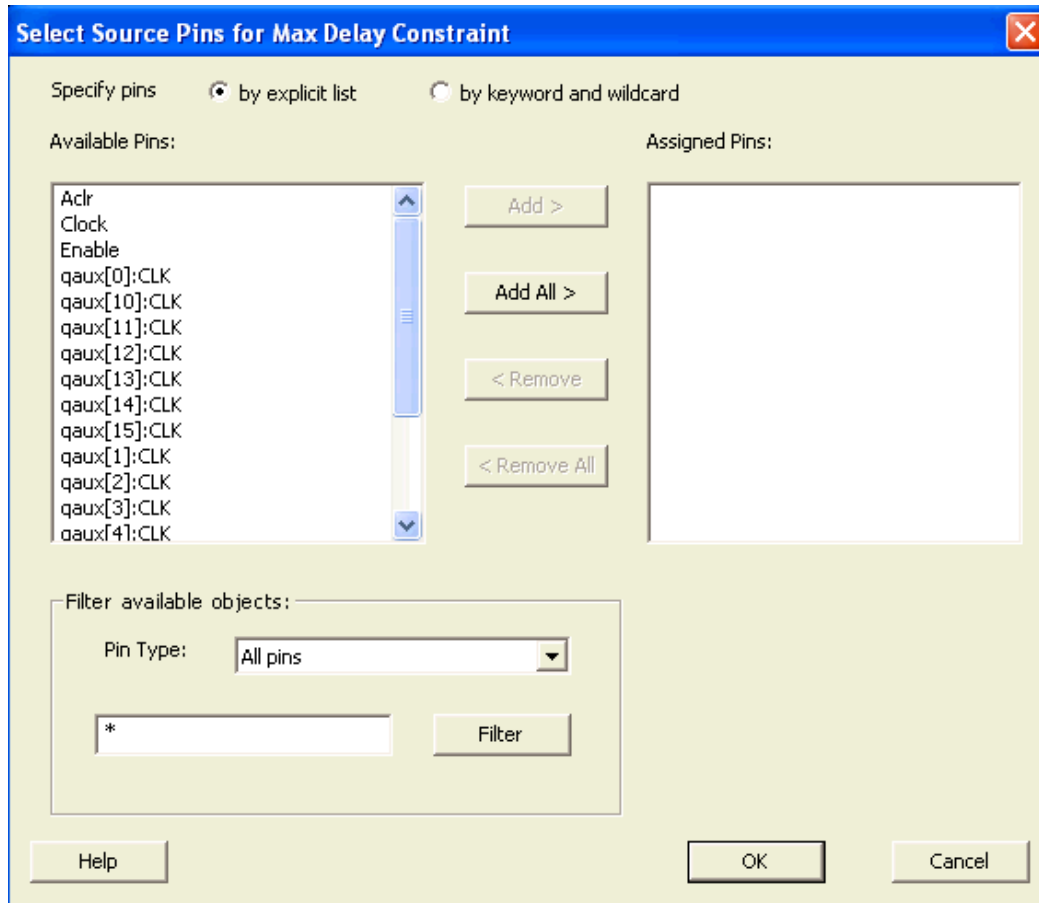


Figure 221 · Select Source Pins for Max Delay Constraint

4. Select **by explicit list**. (Alternatively, you can select **by keyword and wildcard**.)
5. Select the input pin(s) from the **Available Pins** list. You can also use **Filter available objects** to narrow the pin list. You can select multiple ports in this window.
6. Click **Add** or **Add All**. The input pin(s) move from the **Available Pins** list to the **Assigned Pins** list.
7. Click **OK**. The **Set Maximum Delay Constraint** dialog box displays the updated **From** pin(s) list.
8. Click the **Browse** button for **Through** and **To** and add the appropriate pin(s). The displayed list shows the pins reachable from the previously selected pin(s) list.
9. Enter comments in the **Comment** section.
10. Click **OK**.

SmartTime adds the maximum delay constraints to the Constraints List in the SmartTime Constraints Editor.

See Also


[Timing Exceptions Overview](#)

Specifying a Minimum Delay Constraint

You set options in the [Set Minimum Delay Constraint](#) dialog box to relax or to tighten the original clock constraint requirement on specific paths.

To specify *Min* delay constraints:

1. Open the [Set Minimum Delay Constraint](#) dialog box using one of the following methods:

- Click the  icon in the Constraints Editor.

- From the Constraints Editor, right-click the Constraints Menu and choose **Min delay**.

The Set Minimum Delay Constraint dialog box appears (as shown below).

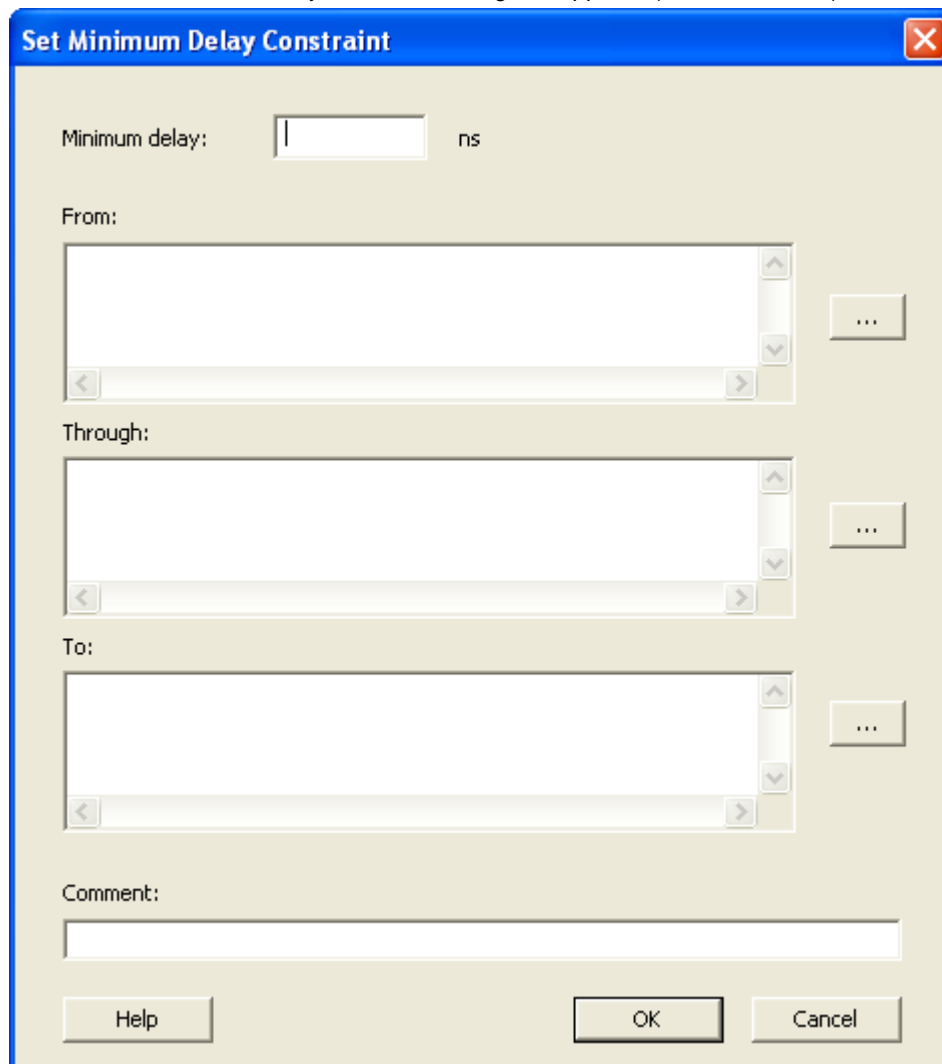


Figure 222 · Set Minimum Delay Constraint Dialog Box

2. Specify the delay in the **Minimum delay** field.
3. Specify the **From** pin(s). Click the **Browse** button next to **From** to open the Select Source Pins for Min Delay Constraint dialog box (as shown below).

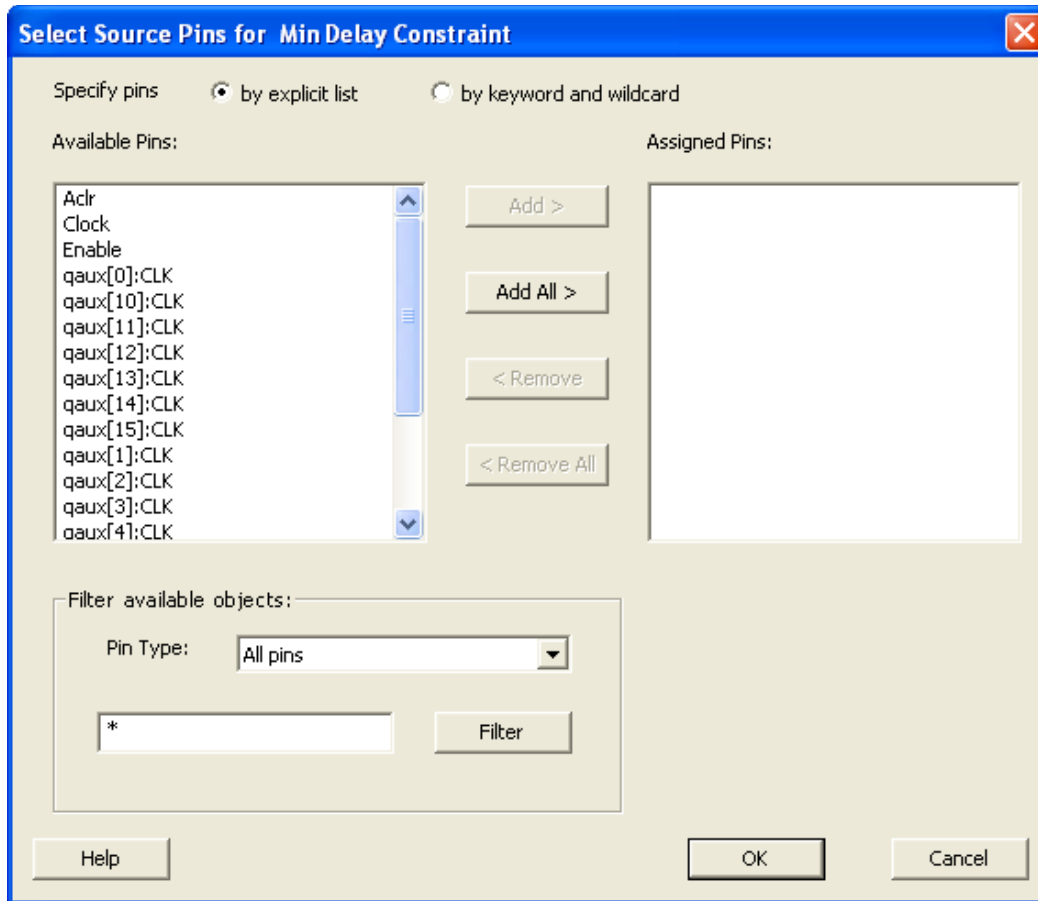


Figure 223 · Select Source Pins for Min Delay Constraint

4. Select **by explicit list**. (Alternatively, you can select **by keyword and wildcard**.)
5. Select the input pin(s) from the **Available Pins** list. You can also use **Filter available objects** to narrow the pin list. You can select multiple ports in this window.
6. Click **Add** or **Add All**. The input pin(s) move from the **Available Pins** list to the **Assigned Pins** list.
7. Click **OK**. The Set Minimum Delay Constraint dialog box displays the updated **From** pin(s) list.
8. Click the **Browse** button for **Through** and **To** and add the appropriate pin(s). The displayed list shows the pins reachable from the previously selected pin(s) list.
9. Enter comments in the **Comment** section.
10. Click **OK**.

SmartTime adds the minimum delay constraints to the Constraints List in the SmartTime Constraints Editor.

See Also


[Timing Exceptions Overview](#)

Specifying a Multicycle Constraint

You set options in the [Set Multicycle Constraint](#) dialog box to specify paths that take multiple clock cycles in the current design.

To specify multicycle constraints:

1. Add the constraint in the [Editable Constraints Grid](#) or open the [Set Multicycle Constraint](#) dialog box using one of the following methods:
 - From the SmartTime Constraints Editor, choose **Constraint > MultiCycle**.

- Click the  icon.
- Right-click the **Multicycle** option in the Constraint Browser and select **Add Multicycle Path Constraint**.

The Set Multicycle Constraint dialog box appears (as shown below).

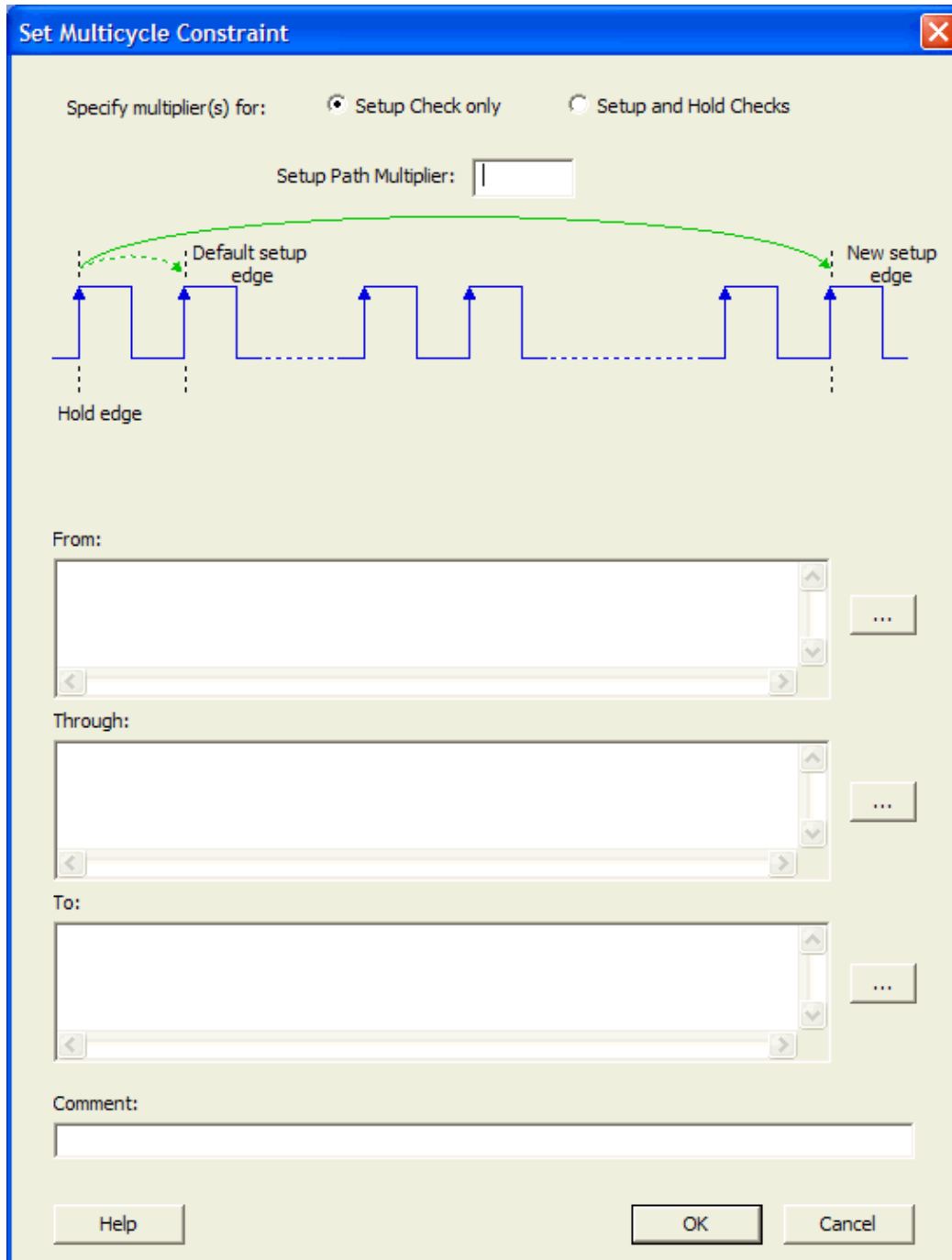


Figure 224 · Set Multicycle Constraint Dialog Box

2. Specify the number of cycles in the **Setup Path Multiplier**.
3. Specify the **From** pin(s). Click the **Browse** button next to **From** to open the Select Source Pins for Constraint dialog box (as shown below).

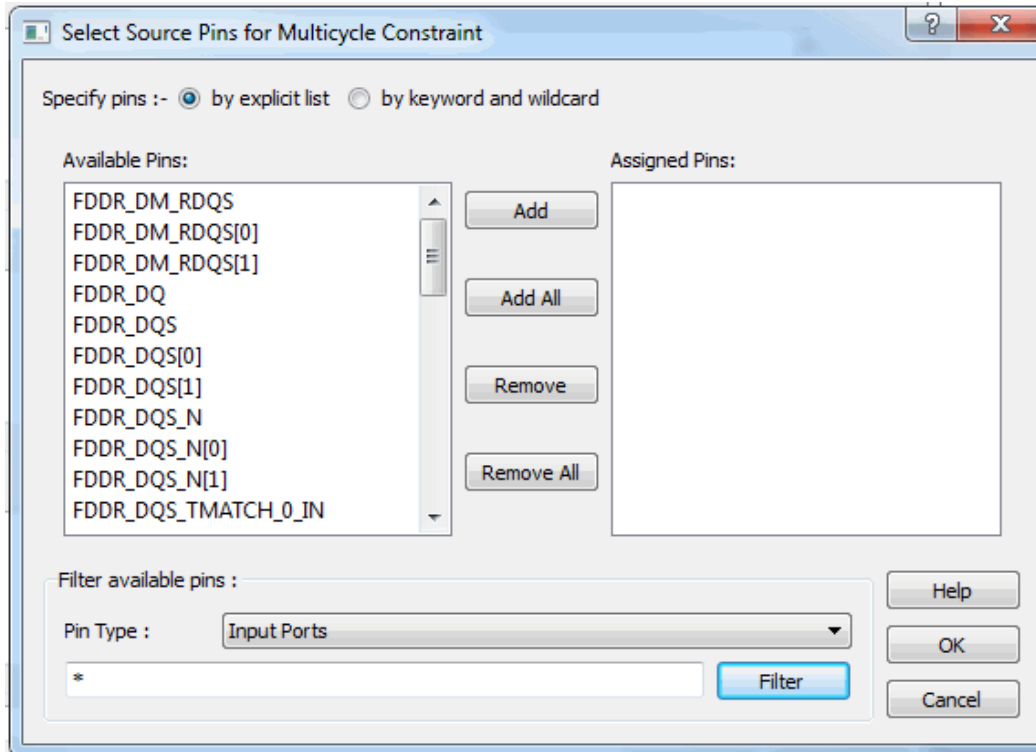


Figure 225 · Select Source Pins for Multicycle Constraint

4. Select **by explicit list**. (Alternatively, you can select **by keyword and wildcard**. For details, refer to the [Select Source or Destination Pins for Constraint Dialog Box](#).)
5. Select the input pin(s) from the **Available Pin** list. You can use **Filter available objects** to narrow the pin list. You can select multiple ports in this window.
6. Click **Add** or **Add All** to move the input pin(s) from the **Available pins** list to the **Assigned Pins** list.
7. Click **OK**.

The Set Multicycle Constraint dialog box displays the updated representation of the **From** pin(s) (as shown below).

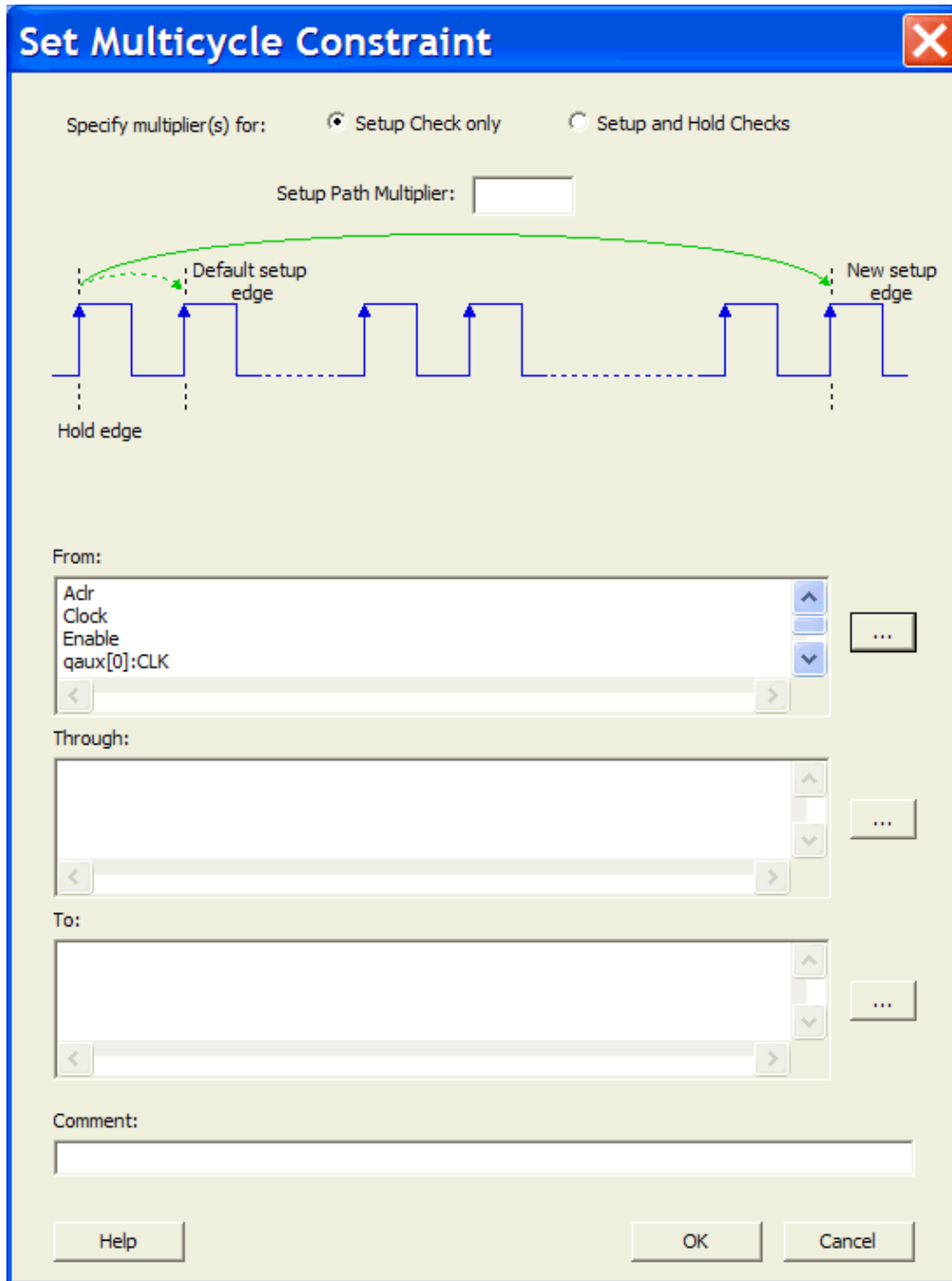


Figure 226 · Set Multicycle Constraint Dialog Box

8. Click the browse button for **Through** and **To** and add the appropriate pins. The displayed list shows the pins reachable from the previously selected pin(s) list.
9. Enter comments in the **Comment** section.
10. Click **OK**. SmartTime adds the multicycle constraints to the Constraints List in the SmartTime Constraints Editor.


See Also

[Set Multicycle Constraint Dialog Box](#)

Specifying Disable Timing Constraint

Use disable timing constraint to specify the timing arcs being disabled.

To specify the disable timing constraint:

1. Add the constraint in the [Editable Constraints Grid](#) or open the [Set Constraint to Disable Timing Arcs Dialog Box](#) using one of the following methods:
 - From the SmartTime Constraints Editor, choose **Constraints > Disable Timing**.
 - Click the  icon in the Constraints Editor.
 - In the Constraints Editor, right-click **Disable Timing** and choose **Add Constraints** to disable timing ..
2. Select an instance from your design.
3. Select whether you want to exclude all timing arcs in the instance or if you want to specify the timing arc to exclude. If you selected specify timing arc to exclude, select a from and to port for the timing arc.
4. Enter any comments to be attached to the constraint.
5. Click **OK**. The new constraint appears in the constraints list.

Note: When you choose Save from the File menu, SmartTime saves the newly-created constraint in the database.


See Also

[Set Constraint to Disable Timing Arcs Dialog Box](#)

Specifying Clock Constraints

Specifying clock constraints is the most effective way to constrain and verify the timing behavior of a sequential design. Use clock constraints to meet your performance goals.

To specify a clock constraint:

1. Add the constraint in the [editable constraints grid](#) or open the [Create Clock Constraint](#) dialog box using one of the following methods:
 - Click the  icon in the Constraints Editor.
 - Right-click the **Clock** in the Constraint Browser and choose **Add Clock Constraint**.
 - Double-click **Clock** in the Constraint Browser.

The Create Clock Constraint dialog box appears (as shown below).

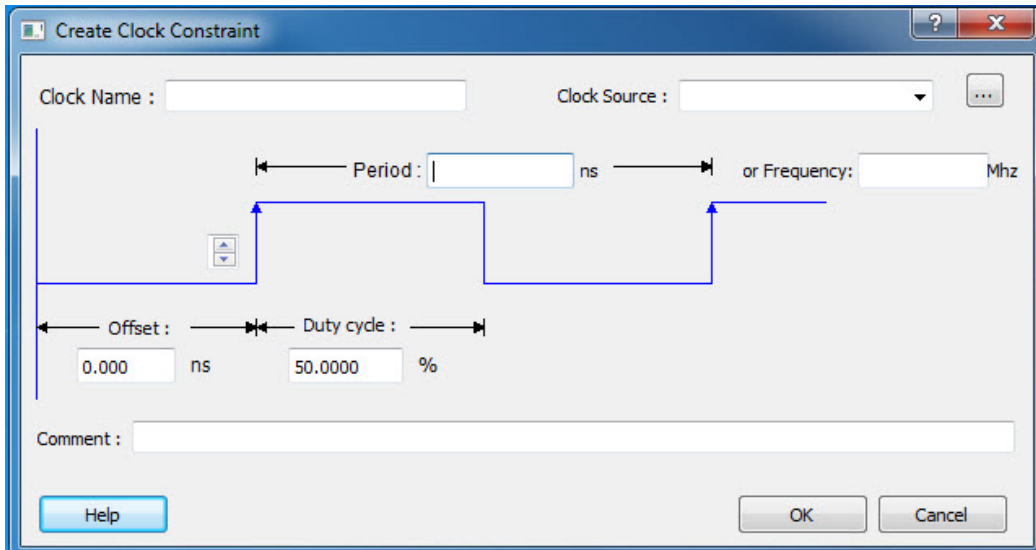


Figure 227 · Create Clock Constraint Dialog Box

2. Select the pin to use as the clock source. You can click the **Browse** button to display the [Select Source Pins for Clock Constraint Dialog Box](#) (as shown below).

Note: Do not select a source pin when you specify a virtual clock. Virtual clocks can be used to define a clock outside the FPGA that it is used to synchronize I/Os.

Use the Choose the Clock Source Pin dialog box to display a list of source pins from which you can choose. By default, it displays the explicit clock sources of the design. To choose other pins in the design as clock source pins, select **Filter available objects - Pin Type** as **Explicit clocks, Potential clocks, All Ports, All Pins, All Nets, Pins on clock network, or Nets in clock network**. To display a subset of the displayed clock source pins, you can create and apply a filter.

Multiple source pins can be specified for the same clock when a single clock is entering the FPGA using multiple inputs with different delays.

Click **OK** to save these dialog box settings.

3. Specify the **Period** in nanoseconds (ns) or **Frequency** in megahertz (MHz).
4. Modify the **Clock Name**. The name of the first clock source is provided as default.
5. Modify the **Duty cycle**, if needed.
6. Modify the **Offset** of the clock, if needed.
7. Modify the first edge direction of the clock, if needed.
8. Click **OK**. The new constraint appears in the Constraints List.

Note: When you choose File > Save, SmartTime saves the newly created constraint in the database.

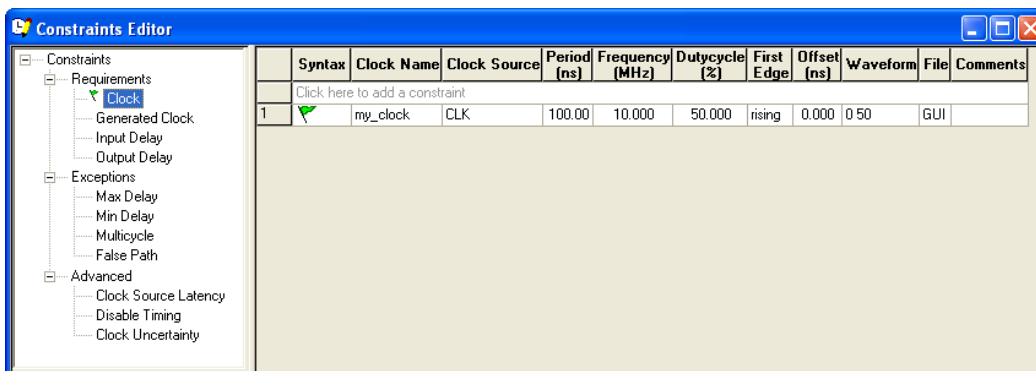



Figure 228 · SmartTime Timing Constraint View

Specifying Generated Clock Constraints

Specifying a generated clock constraint enables you to define an internally generated clock for your design and verify its timing behavior. Use generated clock constraints and [clock constraints](#) to meet your performance goals.

To specify a generated clock constraint:

- Open the [Create Generated Clock Constraint](#) dialog box using one of the following methods:
 - Click the  icon.
 - Right-click the **GeneratedClock** in the Constraint Browser and choose **Add Generated Clock**.
 - Double-click the Generated Clock Constraints grid. The Create Generated Clock Constraint dialog box appears (as shown below).

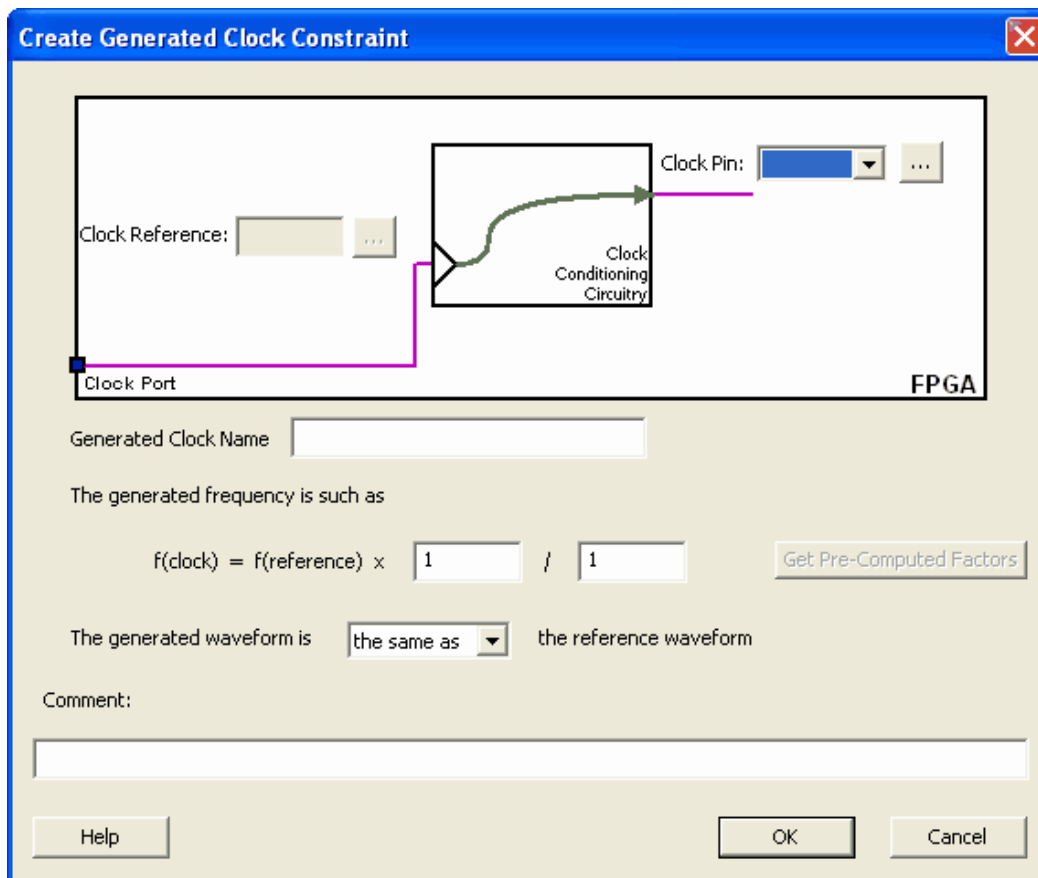


Figure 229 - Create Generated Clock Constraint

- Select a **Clock Pin** to use as the generated clock source. To display a list of available generated clock source pins, click the **Browse** button. The [Select Generated Clock Source](#) dialog box appears (as shown below).

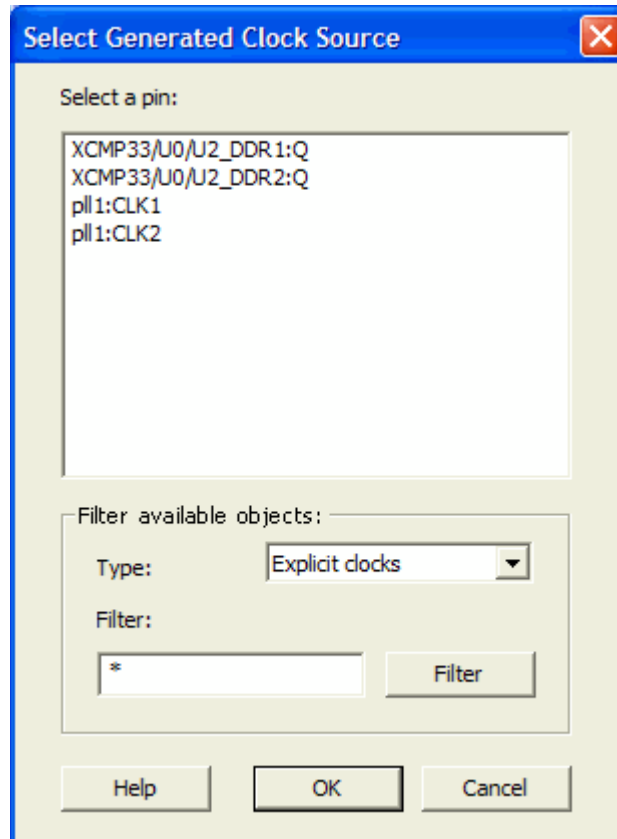


Figure 230 · Select Generated Clock Source Dialog Box

3. Modify the **Clock Name** if necessary.
4. Click **OK** to save these dialog box settings.
5. Specify a **Clock Reference**. To display a list of available clock reference pins, click the **Browse** button. The [Select Generated Clock Reference](#) dialog box appears.
5. Click **OK** to save this dialog box settings.
6. Specify the values to calculate the generated frequency: a multiplication factor and/or a division factor (both positive integers).
7. Specify the first edge of the generated waveform either same as or inverted with respect to the reference waveform.
8. Click **OK**. The new constraint appears in the Constraints List.

Tip: From the **File** menu, choose **Save** to save the newly created constraint in the database.

Specifying I/O States During Programming - I/O States and BSR Details

The I/O States During Programming dialog box enables you to set custom I/O states prior to programming.

I/O State (Output Only)

Sets your I/O states during programming to one of the values shown in the list below.

- 1 – I/Os are set to drive out logic High
- 0 – I/Os are set to drive out logic Low
- Last Known State: I/Os are set to the last value that was driven out prior to entering the programming mode, and then held at that value during programming
- Z - Tri-State: I/Os are tristated

When you set your I/O state, the Boundary Scan Register cells are set according to the table below. Use the Show BSR Details option to set custom states for each cell.

Table 13 · 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 14 · 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.

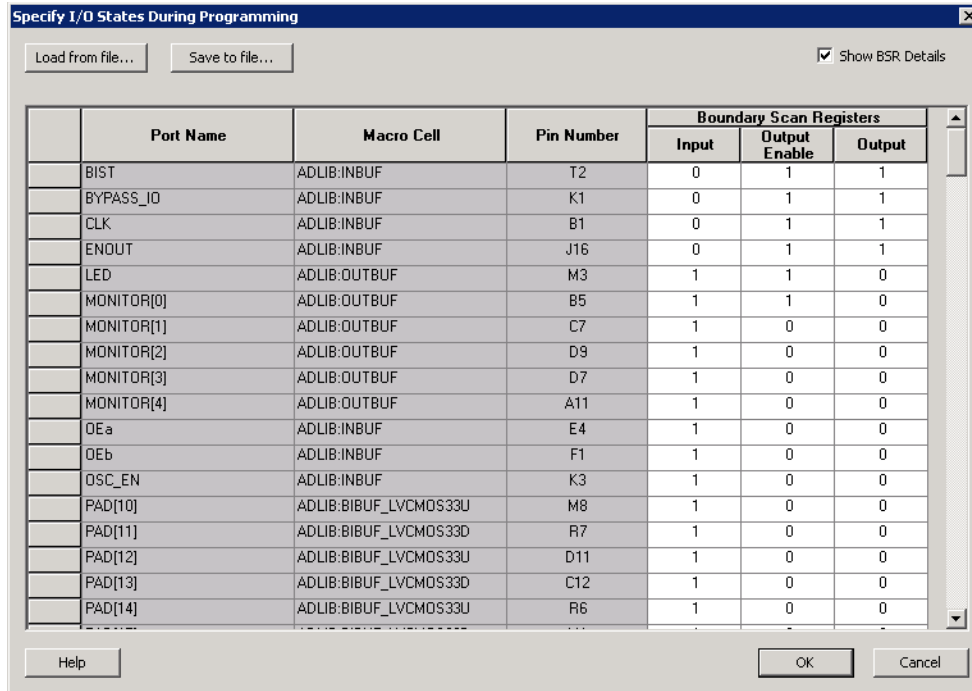


Figure 231 · Boundary Scan Registers

Stimulus Hierarchy

To view the Stimulus Hierarchy, from the **View** menu choose **Windows > Stimulus Hierarchy**.

The Stimulus Hierarchy tab displays a hierarchical representation of the stimulus and simulation files in the project. The software continuously analyzes and updates files and content. The tab (see figure below) displays the structure of the modules and component stimulus files as they relate to each other.

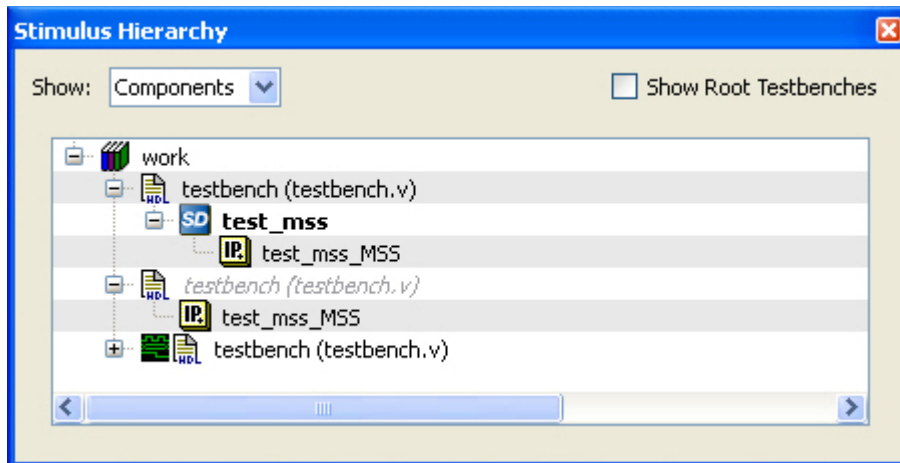


Figure 232 · Stimulus Hierarchy Dialog Box

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** 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 15 · 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

Timing Exceptions Overview

Use timing exceptions to overwrite the default behavior of the design path. Timing exceptions include:

- Setting multicycle constraint to specify paths that (by design) will take more than one cycle.
- Setting a false path constraint to identify paths that must not be included in the timing analysis or the optimization flow.
- Setting a maximum delay constraint on specific paths to relax or to tighten the original clock constraint requirement.

Tool Profiles Dialog Box

The Tool Profiles dialog box enables you to add, edit, or delete your project tool profiles.

Each Libero SoC project can have a different profile, enabling you to integrate different tools with different projects.

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**.

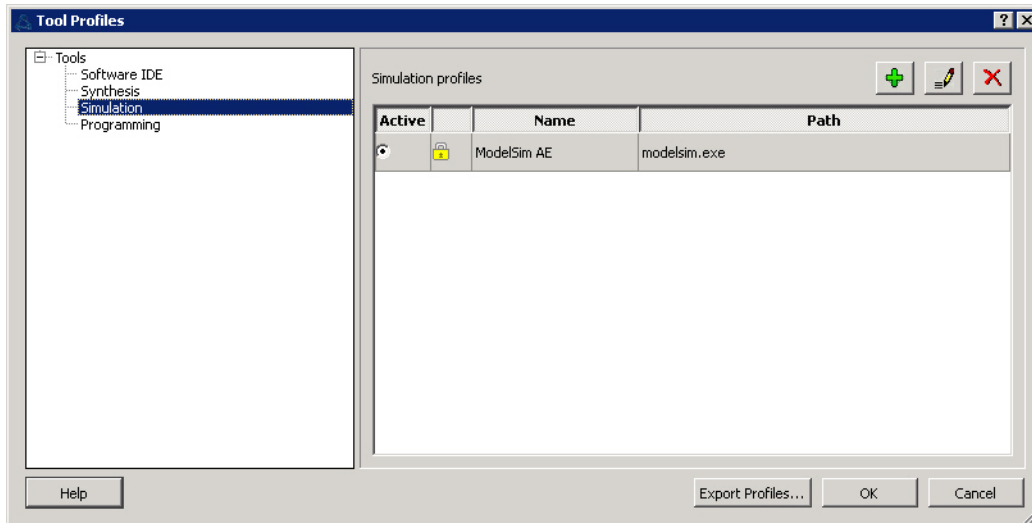


Figure 233 · Libero SoC Tool Profiles Dialog Box

User Preferences Dialog Box – Design Flow Preferences

This dialog box allows you to set your personal preferences for how Libero SoC manages the design flow across the projects you create.

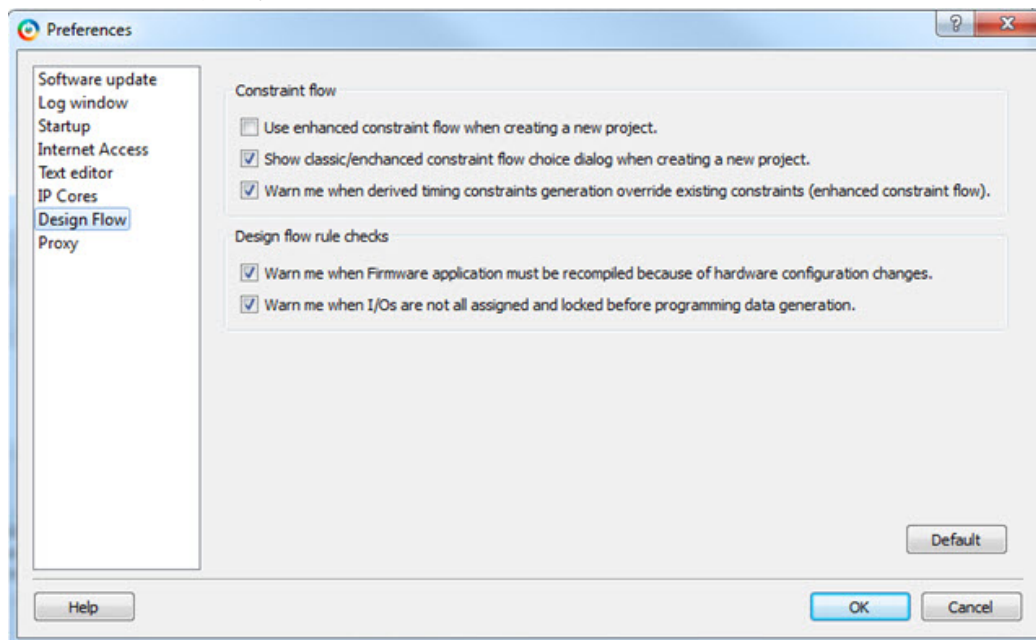


Figure 234 · Preferences Dialog Box – Design Flow Preferences

Constraint Flow

- **Use enhanced constraint flow when creating a new project.**

Check this box to use the Enhanced Constraint Flow when you create a new project. Uncheck this box to use the Classic Constraint Flow (default). Make your Classic Constraint Flow or Enhanced Constraint Flow selection here for new project creation if you do not want a New Project Information

dialog to appear and prompt you to make a selection every time you create a new project. This value is in sync with your most recent selection in the New Project Information dialog box. This box is checked by default.

The Enhanced Constraint Flow provides a centralized graphical interface for you to better manage all your design constraint files in the design process.

- **Show classic/enhanced constraint flow choice dialog when creating a new project.**

When checked, a dialog/informational message appears at the end of project creation and prompts you to select Classic Constraint Flow or Enhanced Constraint Flow. Uncheck this box if you do not want the dialog/informational message to appear at the end of project creation. This box is checked by default.

- **Warn me when derived timing constraints generation override existing constraints (enhanced constraint flow).**

Libero SoC can generate/derive timing constraints for known hardware blocks and IPs such as SERDES, CCC, MSS/HPMS. Check this box to have Libero SoC pop up a warning message when the generated timing constraints for these blocks override the timing constraints you set for these blocks. This box is checked by default.

Design Flow Rule Checks

- **Warn me when Firmware applications must be recompiled because of hardware configuration changes.**

Check this box if you want Libero SoC to display a warning message. This box is checked by default.

- **Warn me when I/Os are not all assigned and locked before programming data generation.**

I/Os should always be assigned and locked before programming data generation. Check this box if you want Libero SoC to display a warning message. This box is checked by default.

Note: These preferences are stored on a per-user basis across multiple projects; they are not project-specific.

What does the ECC1/ECC2 error mean?

ECC is the Error Correction Code embedded in each Flash Memory page.

ECC1 – One bit error and correctable.

ECC2 – Two or more errors found, and not correctable.

Synopsys Design Constraints (SDC)

Synopsys Design Constraints (SDC) is a Tcl-based format used by Synopsys tools to specify the design intent, including the timing and area constraints for a design. Microsemi tools use a subset of the SDC format to capture supported timing constraints. Any timing constraint that you can enter using Designer tools can also be specified in an SDC file.

Use the SDC-based flow to share timing constraint information between Microsemi tools and third-party EDA tools.

Command	Action
create_clock	Creates a clock and defines its characteristics
create_generated_clock	Creates an internally generated clock and defines its characteristics
remove_clock_uncertainty	Removes a clock-to-clock uncertainty from the current timing scenario.

Command	Action
set_clock_latency	Defines the delay between an external clock source and the definition pin of a clock within SmartTime
set_clock_uncertainty	Defines the timing uncertainty between two clock waveforms or maximum skew
set_false_path	Identifies paths that are to be considered false and excluded from the timing analysis
set_input_delay	Defines the arrival time of an input relative to a clock
set_load	Sets the load to a specified value on a specified port
set_max_delay	Specifies the maximum delay for the timing paths
set_min_delay	Specifies the minimum delay for the timing paths
set_multicycle_path	Defines a path that takes multiple clock cycles
set_output_delay	Defines the output delay of an output relative to a clock

See Also

[SDC Syntax Conventions](#)

SDC Syntax Conventions

The following table shows the typographical conventions that are used for the SDC command syntax.

Syntax Notation	Description
command - argument	Commands and arguments appear in Courier New typeface.
<i>variable</i>	Variables appear in blue, italic Courier New typeface. You must substitute an appropriate value for the variable.
[-argument <i>value</i>]	Optional arguments begin and end with a square bracket.

Note: SDC commands and arguments are case sensitive.

Example

The following example shows syntax for the create_clock command and a sample command:

```
create_clock -period period_value [-waveform edge_list] source
create_clock -period 7 -waveform {2 4}{CLK1}
```

Wildcard Characters

You can use the following wildcard characters in names used in the SDC commands:

Wildcard	What it does
\	Interprets the next character literally
*	Matches any string

Note: The matching function requires that you add a backslash (\) before each slash in the pin names in case the slash does not denote the hierarchy in your design.

Special Characters ([], { }, and \)

Square brackets ([]) are part of the command syntax to access ports, pins and clocks. In cases where these netlist objects names themselves contain square brackets (for example, buses), you must either enclose the names with curly brackets ({}), or precede the open and closed square brackets ([]) characters with a backslash (\). If you do not do this, the tool displays an error message.

For example:

```
create_clock -period 3 clk\[0\]
set_max_delay 1.5 -from [get_pins ff1\[5\]:CLK] -to [get_clocks {clk[0]}]
```

Although not necessary, Microsemi recommends the use of curly brackets around the names, as shown in the following example:

```
set_false_path -from {data1} -to [get_pins {reg1:D}]
```

In any case, the use of the curly bracket is mandatory when you have to provide more than one name.

For example:

```
set_false_path -from {data3 data4} -to [get_pins {reg2:D reg5:D}]
```

Entering Arguments on Separate Lines

If a command needs to be split on multiple lines, each line except the last must end with a backslash (\) character as shown in the following example:

```
set_multicycle_path 2 -from \
[get_pins {reg1*}] \
-to {reg2:D}
```

See Also

[About SDC Files](#)

create_clock

SDC command; creates a clock and defines its characteristics.

```
create_clock -name name -period period_value [-waveform edge_list] source
```

Arguments

-name *name*

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

-period *period_value*

Specifies the clock period in nanoseconds. The value you specify is the minimum time over which the clock waveform repeats. The `period_value` must be greater than zero.

`-waveform` [edge_list](#)

Specifies the rise and fall times of the clock waveform in ns over a complete clock period. There must be exactly two transitions in the list, a rising transition followed by a falling transition. You can define a clock starting with a falling edge by providing an edge list where fall time is less than rise time. If you do not specify `-waveform` option, the tool creates a default waveform, with a rising edge at instant 0.0 ns and a falling edge at instant $(\text{period_value}/2)$ ns.

[source](#)

Specifies the source of the clock constraint. The source can be ports or pins in the design. If you specify a clock constraint on a pin that already has a clock, the new clock replaces the existing one. Only one source is accepted. Wildcards are accepted as long as the resolution shows one port or pin.

Supported Families

SmartFusion2, IGLOO2, RTG4, SmartFusion, IGLOO, ProASIC3, Fusion

Description

Creates a clock in the current design at the declared source and defines its period and waveform. The static timing analysis tool uses this information to propagate the waveform across the clock network to the clock pins of all sequential elements driven by this clock source.

The clock information is also used to compute the slacks in the specified clock domain that drive optimization tools such as place-and-route.

Exceptions

- None

Examples

The following example creates two clocks on ports CK1 and CK2 with a period of 6, a rising edge at 0, and a falling edge at 3:

```
create_clock -name {my_user_clock} -period 6 CK1
create_clock -name {my_other_user_clock} -period 6 -waveform {0 3} {CK2}
```

The following example creates a clock on port CK3 with a period of 7, a rising edge at 2, and a falling edge at 4:

```
create_clock -period 7 -waveform {2 4} [get_ports {CK3}]
```

Microsemi Implementation Specifics

- The `-waveform` in SDC accepts waveforms with multiple edges within a period. In Microsemi design implementation, only two waveforms are accepted.
- SDC accepts defining a clock on many sources using a single command. In Microsemi design implementation, only one source is accepted.
- The source argument in SDC `create_clock` command is optional. This is in conjunction with the `-name` argument in SDC to support the concept of virtual clocks. In Microsemi implementation, source is a mandatory argument as `-name` and virtual clocks concept is not supported.
- The `-domain` argument in the SDC `create_clock` command is not supported.

See Also

[SDC Syntax Conventions](#)

create_generated_clock

SDC command; creates an internally generated clock and defines its characteristics.

```
create_generated_clock -name {name} -source reference_pin [-divide_by divide_factor] [-multiply_by multiply_factor] [-invert] source -pll_output pll_feedback_clock -pll_feedback pll_feedback_input
```

Arguments

-name *name*

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

-source *reference_pin*

Specifies the reference pin in the design from which the clock waveform is to be derived.

-divide_by *divide_factor*

Specifies the frequency division factor. For instance if the *divide_factor* is equal to 2, the generated clock period is twice the reference clock period.

-multiply_by *multiply_factor*

Specifies the frequency multiplication factor. For instance if the *multiply_factor* is equal to 2, the generated clock period is half the reference clock period.

-invert

Specifies that the generated clock waveform is inverted with respect to the reference clock.

source

Specifies the source of the clock constraint on internal pins of the design. If you specify a clock constraint on a pin that already has a clock, the new clock replaces the existing clock. Only one source is accepted. Wildcards are accepted as long as the resolution shows one pin.

-pll_output *pll_feedback_clock*

Specifies the output pin of the PLL which is used as the external feedback clock. This pin must drive the feedback input pin of the PLL specified using the `-pll_feedback` option. The PLL will align the rising edge of the reference input clock to the feedback clock. This is a mandatory argument if the PLL is operating in external feedback mode.

-pll_feedback *pll_feedback_input*

Specifies the feedback input pin of the PLL. This pin must be driven by the output pin of the PLL specified using the `-pll_output` option. The PLL will align the rising edge of the reference input clock to the external feedback clock. This is a mandatory argument if the PLL is operating in external feedback mode.

Supported Families

SmartFusion2, IGLOO2, RTG4, SmartFusion, IGLOO, ProASIC3, Fusion

Description

Creates a generated clock in the current design at a declared source by defining its frequency with respect to the frequency at the reference pin. The static timing analysis tool uses this information to compute and propagate its waveform across the clock network to the clock pins of all sequential elements driven by this source.

The generated clock information is also used to compute the slacks in the specified clock domain that drive optimization tools such as place-and-route.

Examples

The following example creates a generated clock on pin U1/reg1:Q with a period twice as long as the period at the reference port CLK.

```
create_generated_clock -name {my_user_clock} -divide_by 2 -source [get_ports  
{CLK}] U1/reg1:Q
```

The following example creates a generated clock at the primary output of myPLL with a period $\frac{3}{4}$ of the period at the reference pin clk.

```
create_generated_clock -divide_by 3 -multiply_by 4 -source clk [get_pins {myPLL:CLK1}]
```

The following example creates a generated clock named system_clk on the GL2 output pin of FCCC_0 with a period equal to half the period of the source clock. The constraint also identifies GL2 output pin as the external feedback clock source and CLK2 as the feedback input pin for FCCC_0.

```
create_generated_clock -name { system_clk } \  
-multiply_by 2 \  
-source { FCCC_0/CCC_INST/INST_CCC_IP:CLK3_PAD } \  
-pll_output { FCCC_0/CCC_INST/INST_CCC_IP:GL2 } \  
-pll_feedback { FCCC_0/CCC_INST/INST_CCC_IP:CLK2 } \  
{ FCCC_0/CCC_INST/INST_CCC_IP:GL2 }
```

Microsemi Implementation Specifics

- SDC accepts either `-multiply_by` or `-divide_by` option. In Microsemi design implementation, both are accepted to accurately model the PLL behavior.
- SDC accepts defining a generated clock on many sources using a single command. In Microsemi design implementation, only one source is accepted.
- The `-duty_cycle`, `-edges` and `-edge_shift` options in the SDC `create_generated_clock` command are not supported in Microsemi design implementation.

See Also

[SDC Syntax Conventions](#)

remove_clock_uncertainty

SDC command; Removes a clock-to-clock uncertainty from the current timing scenario.

```
remove_clock_uncertainty -from | -rise_from | -fall_from from_clock_list -to | -rise_to | -  
fall_to to_clock_list -setup {value} -hold {value}  
remove_clock_uncertainty -id constraint_ID
```

Arguments

`-from`

Specifies that the clock-to-clock uncertainty applies to both rising and falling edges of the source clock list. You can specify only one of the `-from`, `-rise_from`, or `-fall_from` arguments for the constraint to be valid.

`-rise_from`

Specifies that the clock-to-clock uncertainty applies only to rising edges of the source clock list. You can specify only one of the `-from`, `-rise_from`, or `-fall_from` arguments for the constraint to be valid.

`-fall_from`

Specifies that the clock-to-clock uncertainty applies only to falling edges of the source clock list. You can specify only one of the `-from`, `-rise_from`, or `-fall_from` arguments for the constraint to be valid.

from_clock_list

Specifies the list of clock names as the uncertainty source.

`-to`

Specifies that the clock-to-clock uncertainty applies to both rising and falling edges of the destination clock list. You can specify only one of the `-to`, `-rise_to`, or `-fall_to` arguments for the constraint to be valid.

`-rise_to`

Specifies that the clock-to-clock uncertainty applies only to rising edges of the destination clock list. You can specify only one of the `-to`, `-rise_to`, or `-fall_to` arguments for the constraint to be valid.

`-fall_to`

Specifies that the clock-to-clock uncertainty applies only to falling edges of the destination clock list. You can specify only one of the `-to`, `-rise_to`, or `-fall_to` arguments for the constraint to be valid.

`to_clock_list`

Specifies the list of clock names as the uncertainty destination.

`-setup`

Specifies that the uncertainty applies only to setup checks. If none or both `-setup` and `-hold` are present, the uncertainty applies to both setup and hold checks.

`-hold`

Specifies that the uncertainty applies only to hold checks. If none or both `-setup` and `-hold` are present, the uncertainty applies to both setup and hold checks.

`-id constraint_ID`

Specifies the ID of the clock constraint to remove from the current scenario. You must specify either the exact parameters to set the constraint or its constraint ID.

Supported Families

SmartFusion2, IGLOO2, RTG4, SmartFusion, IGLOO, ProASIC3, Fusion

Description

Removes a clock-to-clock uncertainty from the specified clock in the current scenario. If the specified arguments do not match clocks with an uncertainty constraint in the current scenario, or if the specified ID does not refer to a clock-to-clock uncertainty constraint, this command fails.

Do not specify both the exact arguments and the ID.

Exceptions

None

Examples

```
remove_clock_uncertainty -from Clk1 -to Clk2
remove_clock_uncertainty -from Clk1 -fall_to { Clk2 Clk3 } -setup
remove_clock_uncertainty 4.3 -fall_from { Clk1 Clk2 } -rise_to *
remove_clock_uncertainty 0.1 -rise_from [ get_clocks { Clk1 Clk2 } ] -fall_to { Clk3
Clk4 } -setup
remove_clock_uncertainty 5 -rise_from Clk1 -to [ get_clocks {*} ]
remove_clock_uncertainty -id $clockId
```

See Also

[SDC Syntax Conventions](#)

[set_clock_uncertainty](#)

set_clock_latency

SDC command; defines the delay between an external clock source and the definition pin of a clock within SmartTime.

```
set_clock_latency -source [-rise][-fall][-early][-late] delay clock
```

Arguments

`-source`

Specifies a clock source latency on a clock pin.

`-rise`

Specifies the edge for which this constraint will apply. If neither or both rise are passed, the same latency is applied to both edges.

`-fall`

Specifies the edge for which this constraint will apply. If neither or both rise are passed, the same latency is applied to both edges.

`-invert`

Specifies that the generated clock waveform is inverted with respect to the reference clock.

`-late`

Optional. Specifies that the latency is late bound on the latency. The appropriate bound is used to provide the most pessimistic timing scenario. However, if the value of "-late" is less than the value of "-early", optimistic timing takes place which could result in incorrect analysis. If neither or both "-early" and "-late" are provided, the same latency is used for both bounds, which results in the latency having no effect for single clock domain setup and hold checks.

`-early`

Optional. Specifies that the latency is early bound on the latency. The appropriate bound is used to provide the most pessimistic timing scenario. However, if the value of "-late" is less than the value of "-early", optimistic timing takes place which could result in incorrect analysis. If neither or both "-early" and "-late" are provided, the same latency is used for both bounds, which results in the latency having no effect for single clock domain setup and hold checks.

delay

Specifies the latency value for the constraint.

clock

Specifies the clock to which the constraint is applied. This clock must be constrained.

Supported Families

SmartFusion2, IGLOO2, RTG4, SmartFusion, IGLOO, ProASIC3, Fusion

Description

Clock source latency defines the delay between an external clock source and the definition pin of a clock within SmartTime. It behaves much like an input delay constraint. You can specify both an "early" delay and a "late" delay for this latency, providing an uncertainty which SmartTime propagates through its calculations. Rising and falling edges of the same clock can have different latencies. If only one value is provided for the clock source latency, it is taken as the exact latency value, for both rising and falling edges.

Exceptions

None

Examples

The following example sets an early clock source latency of 0.4 on the rising edge of `main_clock`. It also sets a clock source latency of 1.2, for both the early and late values of the falling edge of `main_clock`. The late value for the clock source latency for the falling edge of `main_clock` remains undefined.

```
set_clock_latency -source -rise -early 0.4 { main_clock }
set_clock_latency -source -fall 1.2 { main_clock }
```

Microsemi Implementation Specifics

SDC accepts a list of clocks to `-set_clock_latency`. In Microsemi design implementation, only one clock pin can have its source latency specified per command.

See Also

[SDC Syntax Conventions](#)

set_clock_to_output

SDC command; defines the timing budget available inside the FPGA for an output relative to a clock.

```
set_clock_to_output delay_value -clock clock_ref [-max] [-min] [-clock_fall] output_list
```

Arguments

delay_value

Specifies the clock to output delay in nanoseconds. This time represents the amount of time available inside the FPGA between the active clock edge and the data change at the output port.

`-clock clock_ref`

Specifies the reference clock to which the specified clock to output is related. This is a mandatory argument.

`-max`

Specifies that *delay_value* refers to the maximum clock to output at the specified output. If you do not specify `-max` or `-min` options, the tool assumes maximum and minimum clock to output delays to be equal.

`-min`

Specifies that *delay_value* refers to the minimum clock to output at the specified output. If you do not specify `-max` or `-min` options, the tool assumes maximum and minimum clock to output delays to be equal.

`-clock_fall`

Specifies that the delay is relative to the falling edge of the reference clock. The default is the rising edge.

output_list

Provides a list of output ports in the current design to which *delay_value* is assigned. If you need to specify more than one object, enclose the objects in braces ({}).

set_clock_uncertainty

SDC command; defines the timing uncertainty between two clock waveforms or maximum skew.

```
set_clock_uncertainty uncertainty (-from | -rise_from | -fall_from) from_clock_list (-to | -rise_to | -fall_to) to_clock_list [-setup | -hold]
```

Arguments

uncertainty

Specifies the time in nanoseconds that represents the amount of variation between two clock edges. The value must be a positive floating point number.

`-from`

Specifies that the clock-to-clock uncertainty applies to both rising and falling edges of the source clock list. You can specify only one of the `-from`, `-rise_from`, or `-fall_from` arguments for the constraint to be valid. This option is the default.

`-rise_from`

Specifies that the clock-to-clock uncertainty applies only to rising edges of the source clock list. You can specify only one of the `-from`, `-rise_from`, or `-fall_from` arguments for the constraint to be valid.

`-fall_from`

Specifies that the clock-to-clock uncertainty applies only to falling edges of the source clock list. You can specify only one of the `-from`, `-rise_from`, or `-fall_from` arguments for the constraint to be valid.

from_clock_list

Specifies the list of clock names as the uncertainty source.

`-to`

Specifies that the clock-to-clock uncertainty applies to both rising and falling edges of the destination clock list. You can specify only one of the `-to`, `-rise_to`, or `-fall_to` arguments for the constraint to be valid.

`-rise_to`

Specifies that the clock-to-clock uncertainty applies only to rising edges of the destination clock list. You can specify only one of the `-to`, `-rise_to`, or `-fall_to` arguments for the constraint to be valid.

`-fall_to`

Specifies that the clock-to-clock uncertainty applies only to falling edges of the destination clock list. You can specify only one of the `-to`, `-rise_to`, or `-fall_to` arguments for the constraint to be valid.

to_clock_list

Specifies the list of clock names as the uncertainty destination.

`-setup`

Specifies that the uncertainty applies only to setup checks. If you do not specify either option (`-setup` or `-hold`) or if you specify both options, the uncertainty applies to both setup and hold checks.

`-hold`

Specifies that the uncertainty applies only to hold checks. If you do not specify either option (`-setup` or `-hold`) or if you specify both options, the uncertainty applies to both setup and hold checks.

Supported Families

SmartFusion2, IGLOO2, RTG4, SmartFusion, IGLOO, ProASIC3, Fusion

Description

Clock uncertainty defines the timing between an two clock waveforms or maximum clock skew.

Both **setup** and **hold** checks must account for clock skew. However, for setup check, SmartTime looks for the smallest skew. This skew is computed by using the maximum insertion delay to the launching sequential component and the shortest insertion delay to the receiving component.

For **hold** check, SmartTime looks for the largest skew. This skew is computed by using the shortest insertion delay to the launching sequential component and the largest insertion delay to the receiving component. SmartTime makes this distinction automatically.

Exceptions

None

Examples

The following example defines two clocks and sets the uncertainty constraints, which analyzes the inter-clock domain between clk1 and clk2.

```
create_clock -period 10 clk1
create_generated_clock -name clk2 -source clk1 -multiply_by 2 sclk1
set_clock_uncertainty 0.4 -rise_from clk1 -rise_to clk2
```

Microsemi Implementation Specifics

- SDC accepts a list of clocks to -set_clock_uncertainty.

See Also

[SDC Syntax Conventions](#)
[remove_clock_uncertainty](#)

set_disable_timing

SDC command; disables timing arcs within the specified cell and returns the ID of the created constraint if the command succeeded.

```
set_disable_timing [-from from_port] [-to to_port] cell_name
```

Arguments

-from *from_port*

Specifies the starting port.

-to *to_port*

Specifies the ending port.

cell_name

Specifies the name of the cell in which timing arcs will be disabled.

Supported Families

SmartFusion2, IGLOO2, RTG4, SmartFusion, IGLOO, ProASIC3, Fusion

Description

This command disables the timing arcs in the specified cell, and returns the ID of the created constraint if the command succeeded. The -from and -to arguments must either both be present or both omitted for the constraint to be valid.

Examples

The following example disables the arc between a2:A and a2:Y.


```
set_disable_timing -from port1 -to port2 cellname
```

This command ensures that the arc is disabled within a cell instead of between cells.

Microsemi Implementation Specifics

- None

See Also

[SDC Syntax Conventions](#)

set_external_check

SDC command; defines the external setup and hold delays for an input relative to a clock.

```
set_external_check delay_value -clock clock_ref [-setup] [-hold] [-clock_fall] input_list
```

Arguments

delay_value

Specifies the external setup or external hold delay in nanoseconds. This time represents the amount of time available inside the FPGA for the specified input after a clock edge.

-clock *clock_ref*

Specifies the reference clock to which the specified external check is related. This is a mandatory argument.

-setup

Specifies that *delay_value* refers to the setup check at the specified input. This is a mandatory argument if -hold is not used. You must specify either -setup or -hold option.

-clock_fall

Specifies that the delay is relative to the falling edge of the reference clock. The default is the rising edge.

input_list

Provides a list of input ports in the current design to which *delay_value* is assigned. If you need to specify more than one object, enclose the objects in braces ({}).

Supported Families

SmartFusion2, IGLOO2, RTG4, SmartFusion, IGLOO, ProASIC3, Fusion

Description

The `set_external_check` command specifies the external setup and hold times on input ports relative to a clock edge. This usually represents a combinational path delay from the input port to the clock pin of a register internal to the current design. For in/out (bidirectional) ports, you can specify the path delays for both input and output modes. The tool uses external setup and external hold times for paths starting at primary inputs.

A clock is a singleton that represents the name of a defined clock constraint. This can be an object accessor that will refer to one clock. For example:

```
[get_clocks {system_clk}]  
[get_clocks {sys*_clk}]
```

Examples

The following example sets an external setup check of 12 ns and an external hold check of 6 ns for port `data_in` relative to the rising edge of `CLK1`:

```
set_external_check 12 -clock [get_clocks CLK1] -setup [get_ports data_in]
set_external_check 6 -clock [get_clocks CLK1] -hold [get_ports data_in]
```

See Also

[SDC Syntax Conventions](#)

set_false_path

SDC command; identifies paths that are considered false and excluded from the timing analysis.

```
set_false_path [-from from_list] [-through through_list] [-to to_list]
```

Arguments

-from *from_list*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

-through *through_list*

Specifies a list of pins, ports, cells, or nets through which the disabled paths must pass.

-to *to_list*

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

Supported Families

SmartFusion2, IGLOO2, RTG4, SmartFusion, IGLOO, ProASIC3, Fusion

Description

The `set_false_path` command identifies specific timing paths as being false. The false timing paths are paths that do not propagate logic level changes. This constraint removes timing requirements on these false paths so that they are not considered during the timing analysis. The path starting points are the input ports or register clock pins, and the path ending points are the register data pins or output ports. This constraint disables setup and hold checking for the specified paths.

The false path information always takes precedence over multiple cycle path information and overrides maximum delay constraints. If more than one object is specified within one `-through` option, the path can pass through any objects.

Examples

The following example specifies all paths from clock pins of the registers in clock domain `clk1` to data pins of a specific register in clock domain `clk2` as false paths:

```
set_false_path -from [get_clocks {clk1}] -to reg_2:D
```

The following example specifies all paths through the pin `U0/U1:Y` to be false:

```
set_false_path -through U0/U1:Y
```

Microsemi Implementation Specifics

SDC accepts multiple -through options in a single constraint to specify paths that traverse multiple points in the design. In Microsemi design implementation, only one -through option is accepted.

See Also

[SDC Syntax Conventions](#)

set_input_delay

SDC command; defines the arrival time of an input relative to a clock.

```
set_input_delay delay_value -clock clock_ref [-max] [-min] [-clock_fall] input_list
```

Arguments

delay_value

Specifies the arrival time in nanoseconds that represents the amount of time for which the signal is available at the specified input after a clock edge.

-clock *clock_ref*

Specifies the clock reference to which the specified input delay is related. This is a mandatory argument. If you do not specify -max or -min options, the tool assumes the maximum and minimum input delays to be equal.

-max

Specifies that *delay_value* refers to the longest path arriving at the specified input. If you do not specify -max or -min options, the tool assumes maximum and minimum input delays to be equal.

-min

Specifies that *delay_value* refers to the shortest path arriving at the specified input. If you do not specify -max or -min options, the tool assumes maximum and minimum input delays to be equal.

-clock_fall

Specifies that the delay is relative to the falling edge of the clock reference. The default is the rising edge.

input_list

Provides a list of input ports in the current design to which *delay_value* is assigned. If you need to specify more than one object, enclose the objects in braces ({}).

Supported Families

SmartFusion2, IGLOO2, RTG4, SmartFusion, IGLOO, ProASIC3, Fusion and IGLOOe, except ProASIC3 nano and ProASIC3L

Description

The `set_input_delay` command sets input path delays on input ports relative to a clock edge. This usually represents a combinational path delay from the clock pin of a register external to the current design. For in/out (bidirectional) ports, you can specify the path delays for both input and output modes. The tool adds input delay to path delay for paths starting at primary inputs.

A clock is a singleton that represents the name of a defined clock constraint. This can be:

- a single port name used as source for a clock constraint
- a single pin name used as source for a clock constraint; for instance `reg1:CLK`. This name can be hierarchical (for instance `toplevel/block1/reg2:CLK`)
- an object accessor that will refer to one clock: `[get_clocks {clk}]`

Examples

The following example sets an input delay of 1.2ns for port data1 relative to the rising edge of CLK1:

```
set_input_delay 1.2 -clock [get_clocks CLK1] [get_ports data1]
```

The following example sets a different maximum and minimum input delay for port IN1 relative to the falling edge of CLK2:

```
set_input_delay 1.0 -clock_fall -clock CLK2 -min {IN1}
```

```
set_input_delay 1.4 -clock_fall -clock CLK2 -max {IN1}
```

Microsemi Implementation Specifics

In SDC, the `-clock` is an optional argument that allows you to set input delay for combinational designs. Microsemi Implementation currently requires this argument.

See Also

[SDC Syntax Conventions](#)

set_load

SDC command; sets the load to a specified value on a specified port.

```
set_load capacitance port_list
```

Arguments

capacitance

Specifies the capacitance value that must be set on the specified ports.

port_list

Specifies a list of ports in the current design on which the capacitance is to be set.

Supported Families

SmartFusion2, IGLOO2, RTG4, SmartFusion, IGLOO, ProASIC3, Fusion

Description

The load constraint enables the Designer software to account for external capacitance at a specified port. You cannot set load constraint on the nets. When you specify this constraint on the output ports, it impacts the delay calculation on the specified ports.

Examples

The following examples show how to set output capacitance on different output ports:

```
set_load 35 out_p
```

```
set_load 40 {01 02}
```

```
set_load 25 [get_ports out]
```

Microsemi Implementation Specifics

- In SDC, you can use the `set_load` command to specify capacitance value on nets. Microsemi Implementation only supports output ports.

See Also

[SDC Syntax Conventions](#)

set_max_delay (SDC)

SDC command; specifies the maximum delay for the timing paths.

```
set_max_delay delay_value [-from from_list] [-to to_list]
```

Arguments

delay_value

Specifies a floating point number in nanoseconds that represents the required maximum delay value for specified paths.

- If the path starting point is on a sequential device, the tool includes clock skew in the computed delay.
- If the path starting point has an input delay specified, the tool adds that delay value to the path delay.
- If the path ending point is on a sequential device, the tool includes clock skew and library setup time in the computed delay.
- If the ending point has an output delay specified, the tool adds that delay to the path delay.

-from *from_list*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

-to *to_list*

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

Supported Families

SmartFusion2, IGLOO2, RTG4, SmartFusion, IGLOO, ProASIC3, Fusion

Description

This command specifies the required maximum delay for timing paths in the current design. The path length for any startpoint in *from_list* to any endpoint in *to_list* must be less than *delay_value*.

The tool automatically derives the individual maximum delay targets from clock waveforms and port input or output delays. For more information, refer to the [create clock](#), [set input delay](#), and [set output delay](#) commands.

The maximum delay constraint is a timing exception. This constraint overrides the default single cycle timing relationship for one or more timing paths. This constraint also overrides a multicycle path constraint.

Examples

The following example sets a maximum delay by constraining all paths from ff1a:CLK or ff1b:CLK to ff2e:D with a delay less than 5 ns:

```
set_max_delay 5 -from {ff1a:CLK ff1b:CLK} -to {ff2e:D}
```

The following example sets a maximum delay by constraining all paths to output ports whose names start by "out" with a delay less than 3.8 ns:

```
set_max_delay 3.8 -to [get_ports out*]
```

Microsemi Implementation Specifics

The `-through` option in the `set_max_delay` SDC command is not supported.

See Also

[SDC Syntax Conventions](#)

set_min_delay

SDC command; specifies the minimum delay for the timing paths.

```
set_min_delay delay_value [-from from_list] [-to to_list]
```

Arguments

delay_value

Specifies a floating point number in nanoseconds that represents the required minimum delay value for specified paths.

- If the path starting point is on a sequential device, the tool includes clock skew in the computed delay.
- If the path starting point has an input delay specified, the tool adds that delay value to the path delay.
- If the path ending point is on a sequential device, the tool includes clock skew and library setup time in the computed delay.
- If the ending point has an output delay specified, the tool adds that delay to the path delay.

`-from` *from_list*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-to` *to_list*

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

Supported Families

SmartFusion2, IGLOO2, RTG4, SmartFusion, IGLOO, ProASIC3, Fusion

Description

This command specifies the required minimum delay for timing paths in the current design. The path length for any startpoint in `from_list` to any endpoint in `to_list` must be less than `delay_value`.

The tool automatically derives the individual minimum delay targets from clock waveforms and port input or output delays. For more information, refer to the [create_clock](#), [set_input_delay](#), and [set_output_delay](#) commands.

The minimum delay constraint is a timing exception. This constraint overrides the default single cycle timing relationship for one or more timing paths. This constraint also overrides a multicycle path constraint.

Examples

The following example sets a minimum delay by constraining all paths from ff1a:CLK or ff1b:CLK to ff2e:D with a delay less than 5 ns:

```
set_min_delay 5 -from {ff1a:CLK ff1b:CLK} -to {ff2e:D}
```

The following example sets a minimum delay by constraining all paths to output ports whose names start by “out” with a delay less than 3.8 ns:

```
set_min_delay 3.8 -to [get_ports out*]
```

Microsemi Implementation Specifics

The `-through` option in the `set_min_delay` SDC command is not supported.

See Also

[SDC Syntax Conventions](#)

set_multicycle_path

SDC command; defines a path that takes multiple clock cycles.

```
set_multicycle_path ncycles [-setup] [-hold] [-from from_list] [-through through_list] [-to to_list]
```

Arguments

ncycles

Specifies an integer value that represents a number of cycles the data path must have for setup or hold check. The value is relative to the starting point or ending point clock, before data is required at the ending point.

`-setup`

Optional. Applies the cycle value for the setup check only. This option does not affect the hold check. The default hold check will be applied unless you have specified another `set_multicycle_path` command for the hold value.

`-hold`

Optional. Applies the cycle value for the hold check only. This option does not affect the setup check.

Note: If you do not specify `-setup` or `-hold`, the cycle value is applied to the setup check and the default hold check is performed (*ncycles* -1).

`-from from_list`

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-through through_list`

Specifies a list of pins or ports through which the multiple cycle paths must pass.

`-to to_list`

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

Supported Families

SmartFusion2, IGLOO2, RTG4, SmartFusion, IGLOO, ProASIC3, Fusion

Description

Setting multiple cycle paths constraint overrides the single cycle timing relationships between sequential elements by specifying the number of cycles that the data path must have for setup or hold checks. If you change the multiplier, it affects both the setup and hold checks.

False path information always takes precedence over multiple cycle path information. A specific maximum delay constraint overrides a general multiple cycle path constraint.

If you specify more than one object within one -through option, the path passes through any of the objects.

Examples

The following example sets all paths between reg1 and reg2 to 3 cycles for setup check. Hold check is measured at the previous edge of the clock at reg2.

```
set_multicycle_path 3 -from [get_pins {reg1}] -to [get_pins {reg2}]
```

The following example specifies that four cycles are needed for setup check on all paths starting at the registers in the clock domain ck1. Hold check is further specified with two cycles instead of the three cycles that would have been applied otherwise.

```
set_multicycle_path 4 -setup -from [get_clocks {ck1}]
set_multicycle_path 2 -hold -from [get_clocks {ck1}]
```

Microsemi Implementation Specifics

- SDC allows multiple priority management on the multiple cycle path constraint depending on the scope of the object accessors. In Microsemi design implementation, such priority management is not supported. All multiple cycle path constraints are handled with the same priority.

See Also

[SDC Syntax Conventions](#)

set_output_delay

SDC command; defines the output delay of an output relative to a clock.

```
set_output_delay delay_value -clock clock_ref [-max] [-min] [-clock_fall] output_list
```

Arguments

delay_value

Specifies the amount of time before a clock edge for which the signal is required. This represents a combinational path delay to a register outside the current design plus the library setup time (for maximum output delay) or hold time (for minimum output delay).

-clock *clock_ref*

Specifies the clock reference to which the specified output delay is related. This is a mandatory argument. If you do not specify -max or -min options, the tool assumes the maximum and minimum input delays to be equal.

-max

Specifies that `delay_value` refers to the longest path from the specified output. If you do not specify `-max` or `-min` options, the tool assumes the maximum and minimum output delays to be equal.

`-min`

Specifies that `delay_value` refers to the shortest path from the specified output. If you do not specify `-max` or `-min` options, the tool assumes the maximum and minimum output delays to be equal.

`-clock_fall`

Specifies that the delay is relative to the falling edge of the clock reference. The default is the rising edge.

`output_list`

Provides a list of output ports in the current design to which `delay_value` is assigned. If you need to specify more than one object, enclose the objects in braces (`{}`).

Supported Families

SmartFusion2, IGLOO2, RTG4, SmartFusion, IGLOO, ProASIC3, Fusion

Description

The `set_output_delay` command sets output path delays on output ports relative to a clock edge. Output ports have no output delay unless you specify it. For in/out (bidirectional) ports, you can specify the path delays for both input and output modes. The tool adds output delay to path delay for paths ending at primary outputs.

Examples

The following example sets an output delay of 1.2ns for port OUT1 relative to the rising edge of CLK1:

```
set_output_delay 1.2 -clock [get_clocks CLK1] [get_ports OUT1]
```

The following example sets a different maximum and minimum output delay for port OUT1 relative to the falling edge of CLK2:

```
set_output_delay 1.0 -clock_fall -clock CLK2 -min {OUT1}
```

```
set_output_delay 1.4 -clock_fall -clock CLK2 -max {OUT1}
```

Microsemi Implementation Specifics

- In SDC, the `-clock` is an optional argument that allows you to set the output delay for combinational designs. Microsemi Implementation currently requires this option.

See Also

[SDC Syntax Conventions](#)

Design Object Access Commands

Design object access commands are SDC commands. Most SDC constraint commands require one of these commands as command arguments.

Microsemi software supports the following SDC access commands:

Design Object	Access Command
Cell	get_cells
Clock	get_clocks
Net	get_nets

Design Object	Access Command
Port	get_ports
Pin	get_pins
Input ports	all_inputs
Output ports	all_outputs
Registers	all_registers

See Also

[About SDC Files](#)

all_inputs

[Design object access command](#); returns all the input or inout ports of the design.

```
all_inputs
```

Arguments

- None

Supported Families

SmartFusion2, IGLOO2, RTG4, SmartFusion, IGLOO, ProASIC3, Fusion

Exceptions

- None

Example

```
set_max_delay -from [all_inputs] -to [get_clocks ck1]
```

Microsemi Implementation Specifics

- None

See Also

[SDC Syntax Conventions](#)

all_outputs

[Design object access command](#); returns all the output or inout ports of the design.

```
all_outputs
```

Arguments

- None

Supported Families

SmartFusion2, IGLOO2, RTG4, SmartFusion, IGLOO, ProASIC3, Fusion

Exceptions

- None

Example

```
set_max_delay -from [all_inputs] -to [all_outputs]
```

Microsemi Implementation Specifics

None

See Also

[SDC Syntax Conventions](#)

all_registers

[Design object access command](#); returns either a collection of register cells or register pins, whichever you specify.

```
all_registers [-clock clock_name] [-cells] [-data_pins ]  
[-clock_pins] [-async_pins] [-output_pins]
```

Arguments

-clock *clock_name*

Creates a collection of register cells or register pins in the specified clock domain.

-cells

Creates a collection of register cells. This is the default. This option cannot be used in combination with any other option.

-data_pins

Creates a collection of register data pins.

-clock_pins

Creates a collection of register clock pins.

-async_pins

Creates a collection of register asynchronous pins.

-output_pins

Creates a collection of register output pins.

Supported Families

SmartFusion2, IGLOO2, RTG4, SmartFusion, IGLOO, ProASIC3, Fusion

Description

This command creates either a collection of register cells (default) or register pins, whichever is specified. If you do not specify an option, this command creates a collection of register cells.

Exceptions

- None

Examples

```
set_max_delay 2 -from [all_registers] -to [get_ports {out}]
set_max_delay 3 -to [all_registers -async_pins]
set_false_path -from [all_registers -clock clk150]
set_multicycle_path -to [all_registers -clock c* -data_pins
-clock_pins]
```

Microsemi Implementation Specifics

- None

See Also

[SDC Syntax Conventions](#)

get_cells

[Design object access command](#); returns the cells (instances) specified by the pattern argument.

```
get_cells pattern
```

Arguments

pattern

Specifies the pattern to match the instances to return. For example, "get_cells U18*" returns all instances starting with the characters "U18", where "*" is a wildcard that represents any character string.

Supported Families

SmartFusion2, IGLOO2, RTG4, SmartFusion, IGLOO, ProASIC3, Fusion

Description

This command returns a collection of instances matching the pattern you specify. You can only use this command as part of a `-from`, `-to`, or `-through` argument for the following constraint exceptions: `set_max_delay`, `set_multicycle_path`, and `set_false_path` design constraints.

Exceptions

None

Examples

```
set_max_delay 2 -from [get_cells {reg*}] -to [get_ports {out}]
set_false_path -through [get_cells {Rblock/muxA}]
```

Microsemi Implementation Specifics

- None

See Also

[SDC Syntax Conventions](#)

get_clocks

[Design object access command](#); returns the specified clock.

```
get_clocks pattern
```

Arguments

pattern

Specifies the pattern to match to the SmartTime on which a clock constraint has been set.

Supported Families

SmartFusion2, IGLOO2, RTG4, SmartFusion, IGLOO, ProASIC3, Fusion

Description

- If this command is used as a –from argument in maximum delay ([set_max_path_delay](#)), false path ([set_false_path](#)), and multicycle constraints ([set_multicycle_path](#)), the clock pins of all the registers related to this clock are used as path start points.
- If this command is used as a –to argument in maximum delay ([set_max_path_delay](#)), false path ([set_false_path](#)), and multicycle constraints ([set_multicycle_path](#)), the synchronous pins of all the registers related to this clock are used as path endpoints.

Exceptions

- None

Example

```
set_max_delay -from [get_ports data1] -to \
[get_clocks ck1]
```

Microsemi Implementation Specifics

None

See Also

[SDC Syntax Conventions](#)

get_pins

[Design object access command](#); returns the specified pins.

```
get_pins pattern
```

Arguments

pattern

Specifies the pattern to match the pins.

Supported Families

SmartFusion2, IGLOO2, RTG4, SmartFusion, IGLOO, ProASIC3, Fusion

Exceptions

None

Example

```
create_clock -period 10 [get_pins clock_gen/reg2:Q]
```

Microsemi Implementation Specifics

- None

See Also

[SDC Syntax Conventions](#)

get_nets

[Design object access command](#); returns the named nets specified by the pattern argument.

```
get_nets pattern
```

Arguments

pattern

Specifies the pattern to match the names of the nets to return. For example, "get_nets N_255*" returns all nets starting with the characters "N_255", where "*" is a wildcard that represents any character string.

Supported Families

SmartFusion2, IGLOO2, RTG4, SmartFusion, IGLOO, ProASIC3, Fusion

Description

This command returns a collection of nets matching the pattern you specify. You can only use this command as source objects in create clock ([create_clock](#)) or create generated clock ([create_generated_clock](#)) constraints and as -through arguments in set false path ([set_false_path](#)), set

minimum delay (`set_min_delay`), set maximum delay ([set_max_delay](#)), and set multicycle path ([set_multicycle_path](#)) constraints.

Exceptions

None

Examples

```
set_max_delay 2 -from [get_ports RDATA1] -through [get_nets {net_chkp1 net_chkqi}]
set_false_path -through [get_nets {Tblk/rm/n*}]
create_clock -name mainCLK -per 2.5 [get_nets {cknet}]
```

Microsemi Implementation Specifics

None

See Also

[SDC Syntax Conventions](#)

get_ports

[Design object access command](#); returns the specified ports.

```
get_ports pattern
```

Argument

pattern

Specifies the pattern to match the ports. This is equivalent to the macros `$in()[<pattern>]` when used as –from argument and `$out()[<pattern>]` when used as –to argument or `$ports()[<pattern>]` when used as a –through argument.

Supported Families

SmartFusion2, IGLOO2, RTG4, SmartFusion, IGLOO, ProASIC3, Fusion

Exceptions

None

Example

```
create_clock -period 10[get_ports CK1]
```

Microsemi Implementation Specifics

None

See Also

[SDC Syntax Conventions](#)

Exceptions

None

Example

```
create_clock -period 10[get_ports CK1]
```

Microsemi Implementation Specifics

None

See Also[SDC Syntax Conventions](#)

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.



Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA

Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

E-mail: sales.support@microsemi.com

© 2016 Microsemi Corporation. All rights reserved. Microsemi and the Microsemi logo are 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 communications, defense & security, aerospace 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, Calif., and has approximately 4,800 employees globally. Learn more at www.microsemi.com.

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.