
PolarFire Design Flow User 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.**





Microsemi Corporate Headquarters
 One Enterprise, Aliso Viejo,
 CA 92656 USA
 Within the USA: +1 (800) 713-4113
 Outside the USA: +1 (949) 380-6100
 Fax: +1 (949) 215-4996
 Email:
sales.support@microsemi.com
www.microsemi.com

©2017 Microsemi Corporation. All rights reserved. Microsemi and the Microsemi logo are registered trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners.

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

About Microsemi

Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions; security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, California, and has approximately 4,800 employees globally. Learn more at www.microsemi.com.

Table of Contents

Table of Contents.....	4
Libero SoC Introduction.....	7
Welcome to Microsemi's Libero® SoC PolarFire™ Release.....	7
Libero SoC PolarFire Design Flow	7
Constraint Flow and Design Sources	11
Microsemi License Utility.....	12
File Types in Libero SoC.....	13
Software Tools - Libero SoC.....	14
Libero Design Flow.....	16
Starting the Libero GUI.....	16
Design Report	17
Using the New Project Wizard to Start a Project.....	18
Create and Verify Design	23
SmartDesign.....	23
Designing with HDL.....	69
Simulation and Verification.....	72
Libero SoC Constraint Management.....	76
Invocation of Constraint Manager From the Design Flow Window	76
Libero SoC Design Flow.....	76
Synthesis Constraints	77
Place and Route Constraints	78
Timing Verifications Constraints.....	79

Constraint Manager Components79

Constraint File and Tool Association.....79

Derive Constraints in Timing Tab.....80

Create New Constraints80

Constraint File Order81

Import a Constraint File.....81

Link a Constraint File82

Check a Constraint File82

Edit a Constraint File.....84

Constraint Types86

Constraint Manager – I/O Attributes Tab87

Constraint Manager – Timing Tab89

Derived Constraints.....91

Timing Constraints Editor.....92

Constraint Manager – Floor Planner Tab.....148

Constraint Manager – Netlist Attributes Tab150

Implement Design152

Synthesize152

Compile Netlist.....157

Resource Usage.....157

Constraint Flow in Implementation159

Place and Route.....165

Multiple Pass Layout Configuration.....167

Verify Post Layout Implementation.....170

SmartTime170

Verify Power	171
Handoff Design	174
Export Pin Report	174
Export BSDL File	174
References	175

Libero SoC Introduction

Welcome to Microsemi's Libero® SoC PolarFire™ Release

Microsemi's Libero® SoC PolarFire™ software is specifically for designing with PolarFire FPGAs, the fifth generation family of non-volatile FPGA devices from Microsemi, built on state-of-the-art 28nm non-volatile process technology. Cost-optimized PolarFire FPGAs deliver the lowest power at mid-range densities.

For documentation about PolarFire FPGAs, see the [PolarFire product information](#) page on the Microsemi website.

Microsemi's Libero® SoC is a comprehensive and powerful FPGA design and development software suite, providing start-to-finish design flow guidance and support for novice and experienced users alike. Libero SoC combines Microsemi's tools with EDA tools such as Synplify Pro® and ModelSim®.

Complete FPGA Design Flow

The Libero SoC PolarFire release offers a complete FPGA design flow, including:

- Create Design: SmartDesign, VHDL, Verilog, and SystemVerilog for design and test bench development
- Verify Pre-Synthesized Design: Simulate using ModelSim ME Pro featuring mixed-language simulation
- Synthesize using Synopsys Synplify Pro ME
- Constraint Management: I/O Constraints, Timing Constraints, Floorplan Constraints, and Netlist Attributes Constraints with enhanced support for Transceiver and Memory Interface (DDR) assignments
- Chip Planner – For floorplanning and cross-probing from the Chip Planner to the post-compile flattened netlist view (only in Netlist Viewer embedded inside Chip Planner) or from SmartTime
- Place and Route: Timing Driven including Min Delay Repair and Advanced Multiple Pass options
- Static Timing Analysis with SmartTime using Advanced timing data
- Power Analysis with SmartPower using Advanced power data
- Netlist Viewer – For probing into the netlist at various design stages: pre-synthesis (RTL view), post-synthesis (hierarchical view) and post-compile (flattened view)

More Information

For more information about the Libero® SoC PolarFire release, see the [Microsemi website](#).

Libero SoC PolarFire Design Flow

The Libero SoC PolarFire Flow is shown in the figure below.

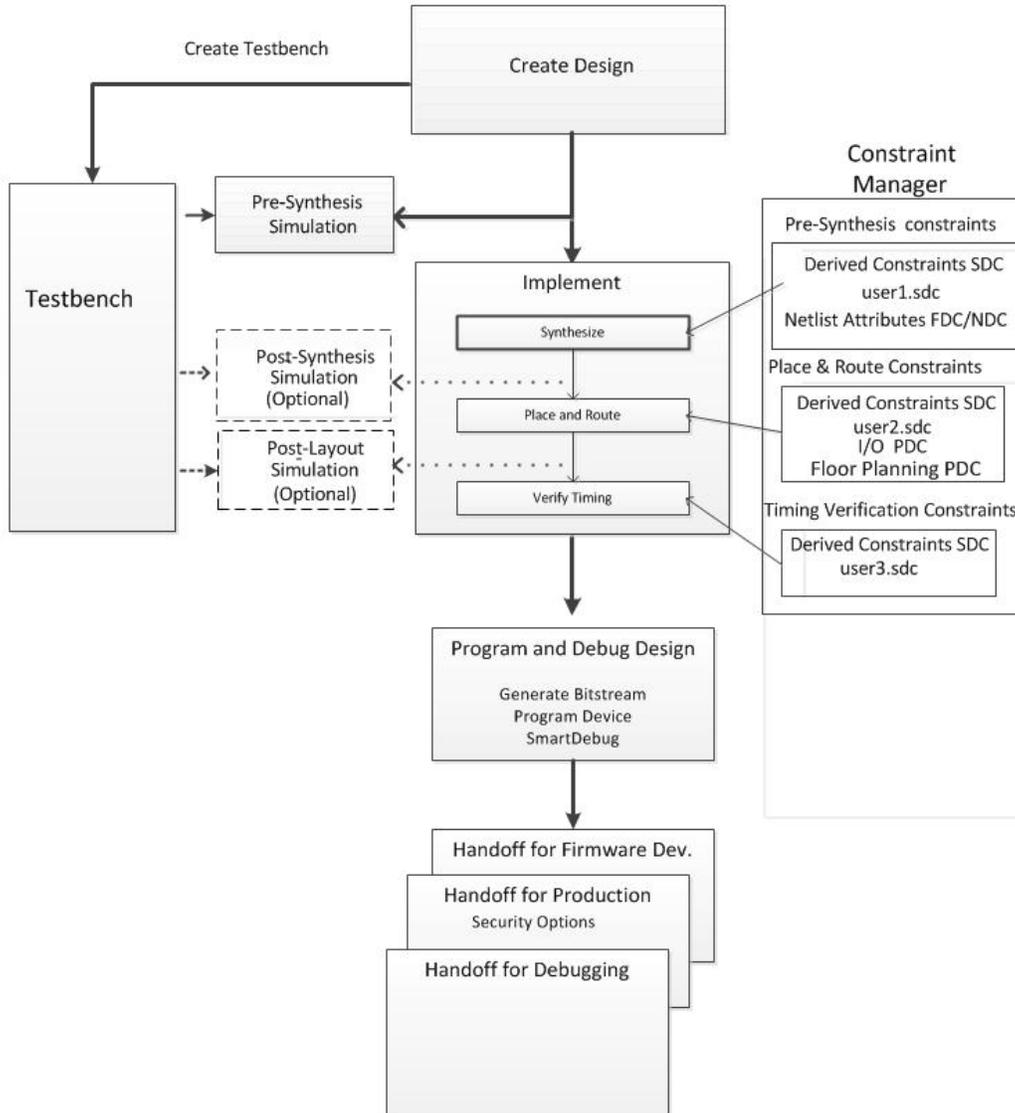


Figure 1 · Libero SoC Enhanced Constraint Design Flow for PolarFire Devices
 (NOTE: "PolarFire Flow" is often called "Enhanced Constraint Flow" in the documentation).

Create Design

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

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

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

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

Programming

You do not have to open FlashPro or FlashPoint to program your 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](#)

Constraint Flow and Design Sources

The 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.

Constraint Flow for HDL designs

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

Constraint Flow for EDIF designs

When the design source is 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.

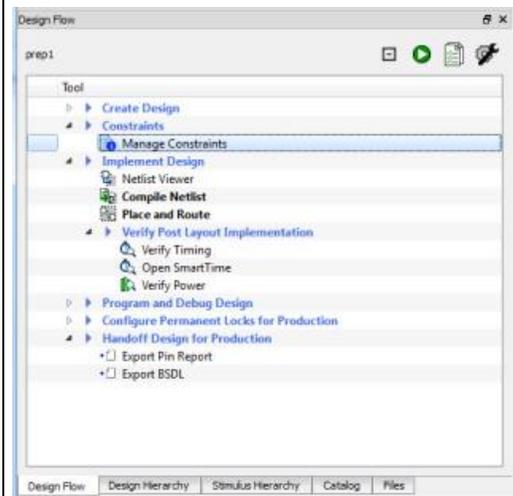
The HDL flow versus the EDIF Flow is compared and contrasted below.

HDL Flow



Design Flow Window

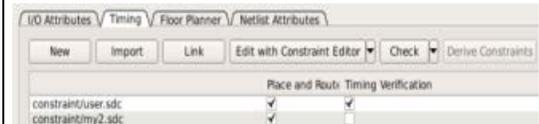
EDIF Flow



Design Flow Window



Constraint Manager



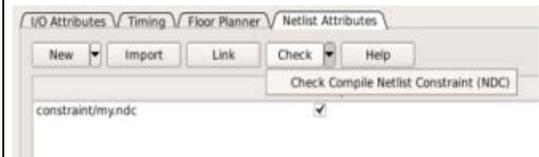
Constraint Manager

HDL Flow

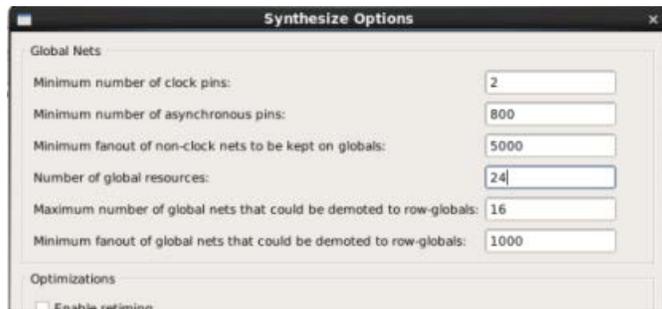


Constraint Manager - Check *.fdc and *.ndc

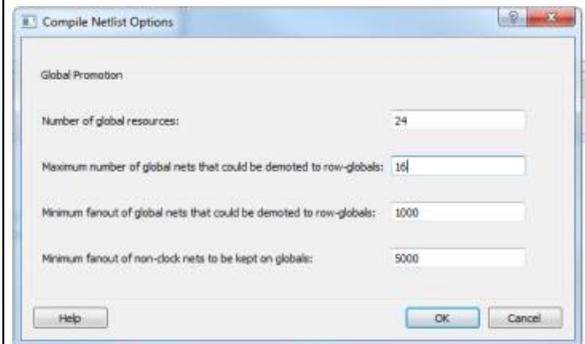
EDIF Flow



Constraint Manager - Check *.ndc only



Global Promotion/Demotion Options set in Synthesize Options Dialog Box



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.

The following licenses are available:

- 1-year Platinum - Purchased license that supports all devices
- 1-year Gold - Purchased licenses that supports a smaller set of devices than Platinum
- 1-year Silver - Free license that supports a smaller set of devices than Gold
- 30-day Evaluation - Free license that supports all devices but programming is disabled

When you have received your license file, follow the instructions and save the license to your local disk. In the Microsemi License Utility window, click **Add License File** and browse/select the license file from your disk. If you are using a floating license, click **Add License Server** and enter the Port Number and Name of the license server host.

The list of features for which you are licensed will show all versions, but your license must have a version equal to or greater than your design tools release version in order for the `libero.exe` and `designer.exe` tools to run.

The list at the lower right shows the order in which the license files are read, with the first file read at the top of the list.

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

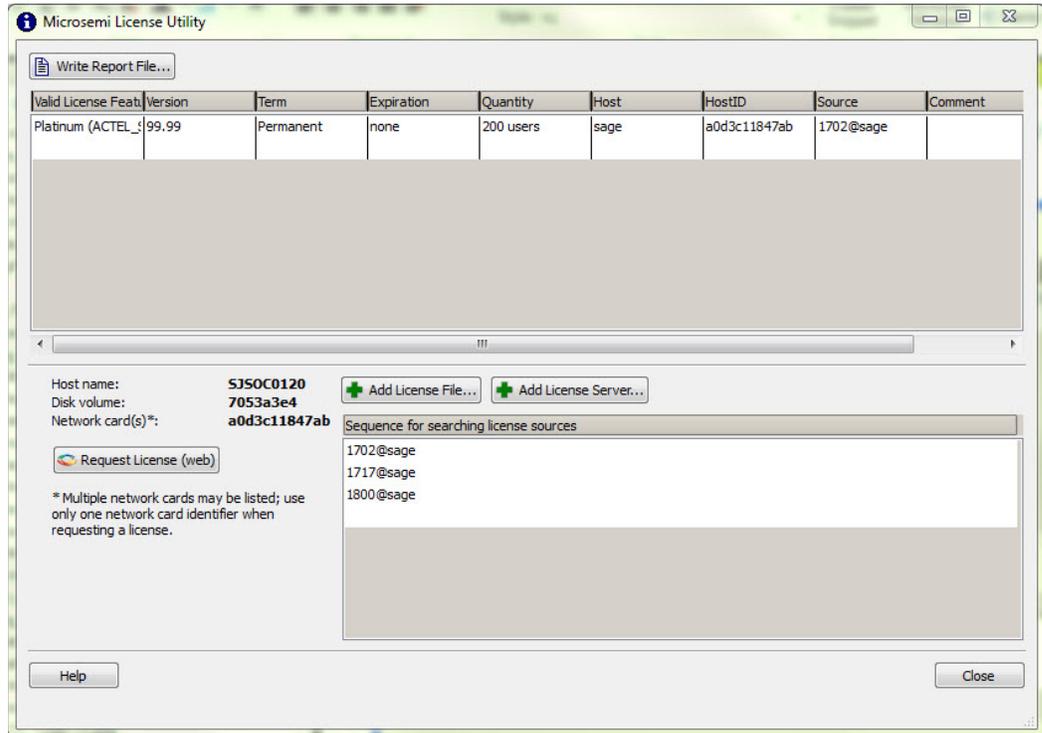


Figure 2 · Microsemi License Utility

File Types in Libero SoC

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

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

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

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

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

Libero SoC generates a <component_name>_manifest.txt file which stores the file name and location of:

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

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

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

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

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

smartgen directory - GEN files and LOG files from generated cores

stimulus directory - BTIM, Verilog, and VHDL stimulus files

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

viewdraw directory - viewdraw.ini files

Internal Files

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

File	File Extension	Remarks
Routing Segmentation File	*.seg	
Combiner Info	*.cob	
Hierarchical Netlist	*.adl	
Flattened Netlist	*.afl	
map file	*.map	Fabric Programming File

Software Tools - Libero SoC

The Libero SoC integrates design tools, streamlines your design flow, manages design and log files, and passes design data between tools.

For more information on Libero SoC tools, visit:

<http://www.microsemi.com/products/fpga-soc/design-resources/design-software/libero-soc#overview>

Function	Tool	Company
Project Manager, HDL Editor, Core Generation	Libero SoC	Microsemi SoC

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

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

Synplify Pro ME from Synopsys is integrated as part of the design package, enabling designers to target HDL code to specific devices.

Microsemi SoC software package includes:

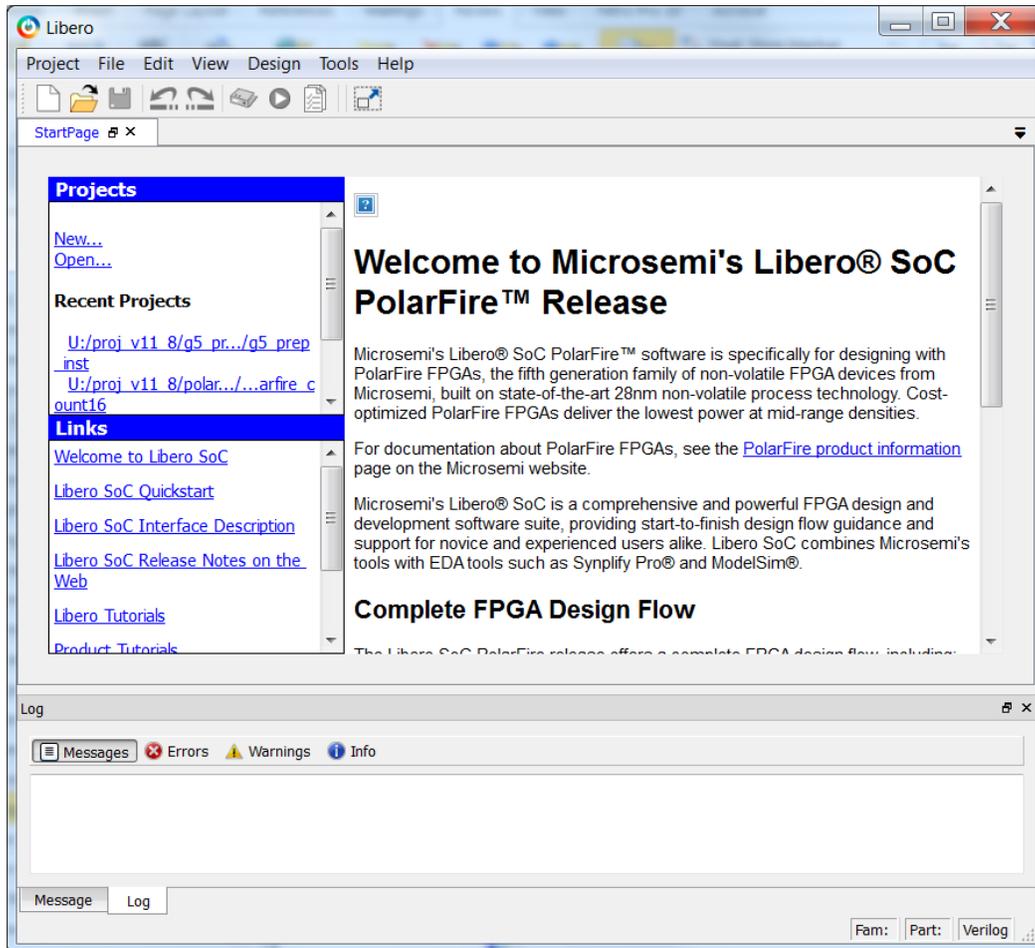
- **Chip Planner** displays I/O and logic macros in your design for floorplanning
- **Netlist Viewer** design schematic viewer
- **SmartPower** power analysis tool
- **SmartTime** static timing analysis and constraints editor

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

Libero Design Flow

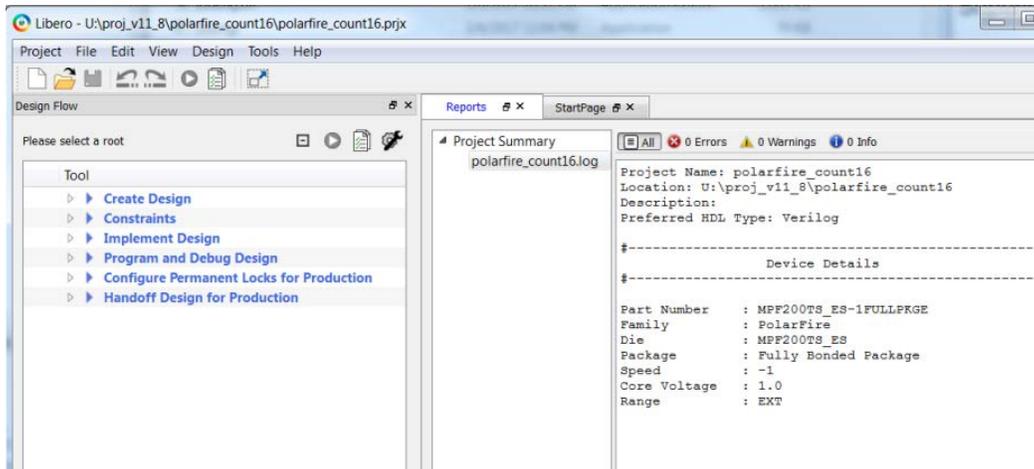
Starting the Libero GUI

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



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

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



The Design Flow Window

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

- Create
- Constrain
- Implement

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](#)
- Export
 - [Export Pin Report](#)
 - [Export IBIS Model](#)

Using the New Project Wizard to Start a Project

New Project Creation Wizard – Project Details

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

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

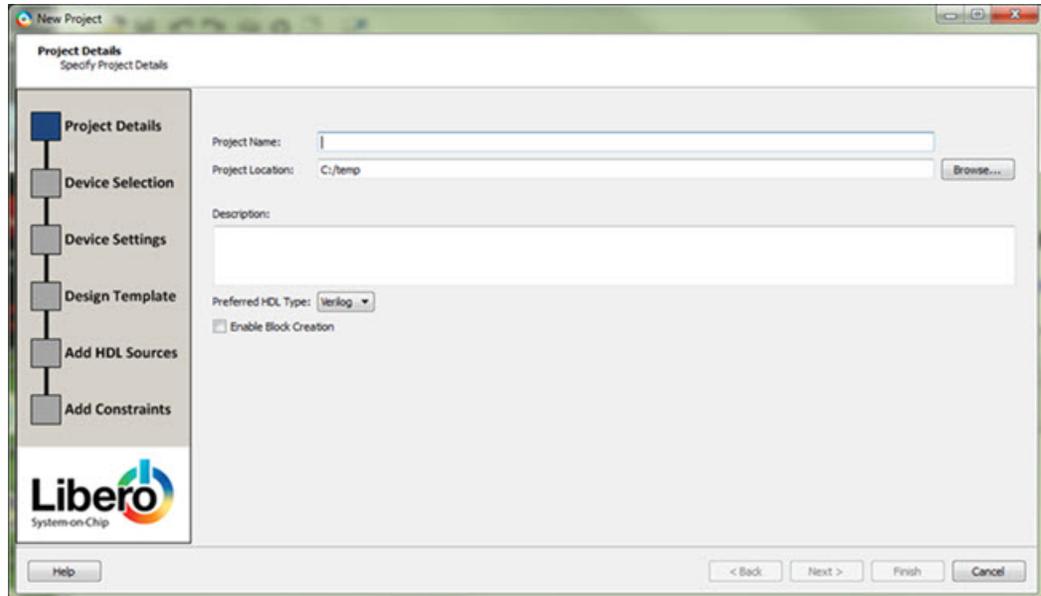


Figure 3 · Libero SoC New Project Creation Wizard

Project

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

Project Location – Identifies your project location on disk.

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

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

When you are finished, click **Next** to proceed to the [Device Selection](#) page.

See Also

[New Project Creation Wizard - Device Selection](#)

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

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

[New Project Creation Wizard - Add Constraints](#)

New Project Creation Wizard – Device Selection

The Device Selection page is where you specify the Microsemi device for your project. Use the filters and drop-down lists to refine your search for the right part to use for your design.

This page contains a table of all parts with associated FPGA resource details generated as a result of a value entered in a filter.

When a value is selected for a filter:

- The parts table is updated to reflect the result of the new filtered value.
- All other filters are updated, and only relevant items are available in the filter drop-down lists.

For example, when PolarFire is selected in the Family filter:

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

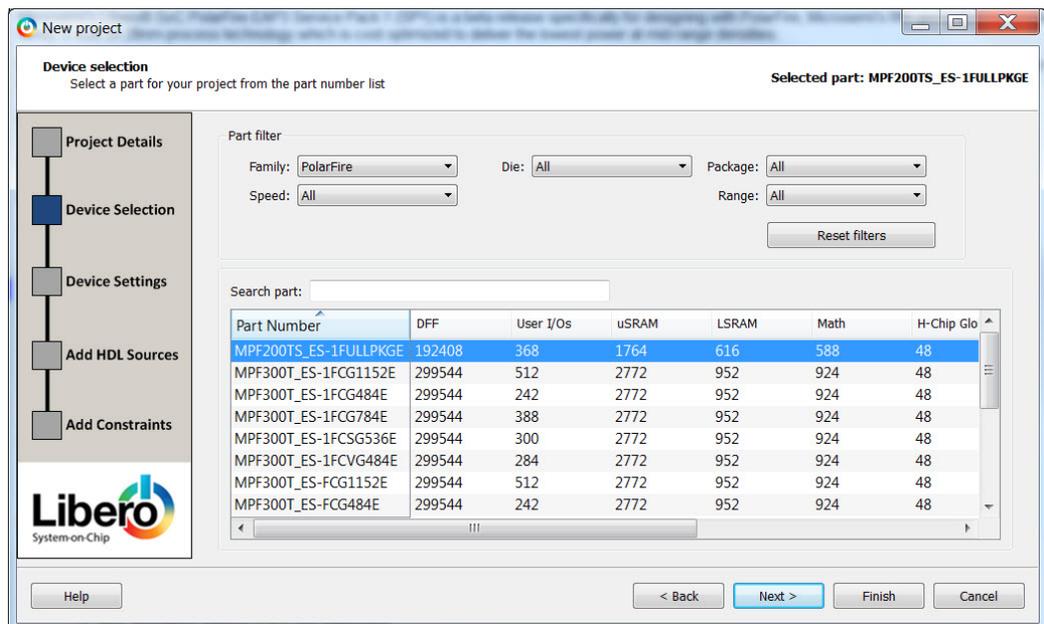


Figure 4 · New Project Creation Wizard - Device Selection Page

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

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

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

Range - Define the voltage and temperature ranges a device may encounter in your application. Tools such as SmartTime, SmartPower, timing-driven layout, power-driven layout, the timing report, and back-annotated simulation are affected by operating conditions.

Supported ranges include:

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

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.

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

- **Next** to proceed to the [Device Settings](#) page
OR
- **Finish** to complete New Project Creation with all remaining defaults.

New Project Creation Wizard – Device Settings

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

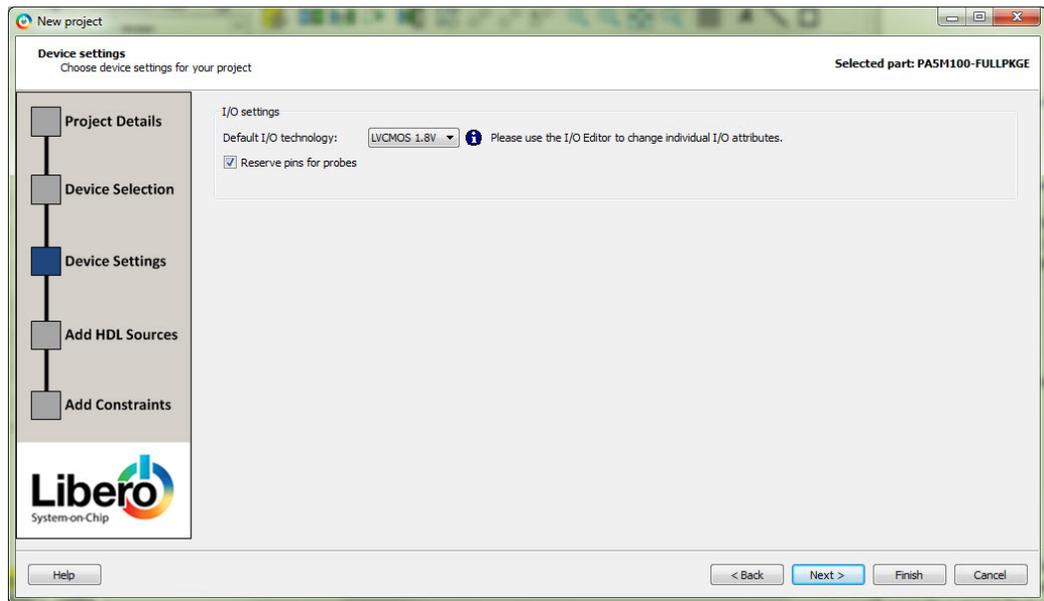


Figure 5 · New Project Creation Wizard – Device Settings Page

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 - Reserve your pins for probing if you intend to debug using SmartDebug.

When you are finished, click **Next** to proceed to the next page, or click **Finish** to complete New Project Creation with all remaining defaults.

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.

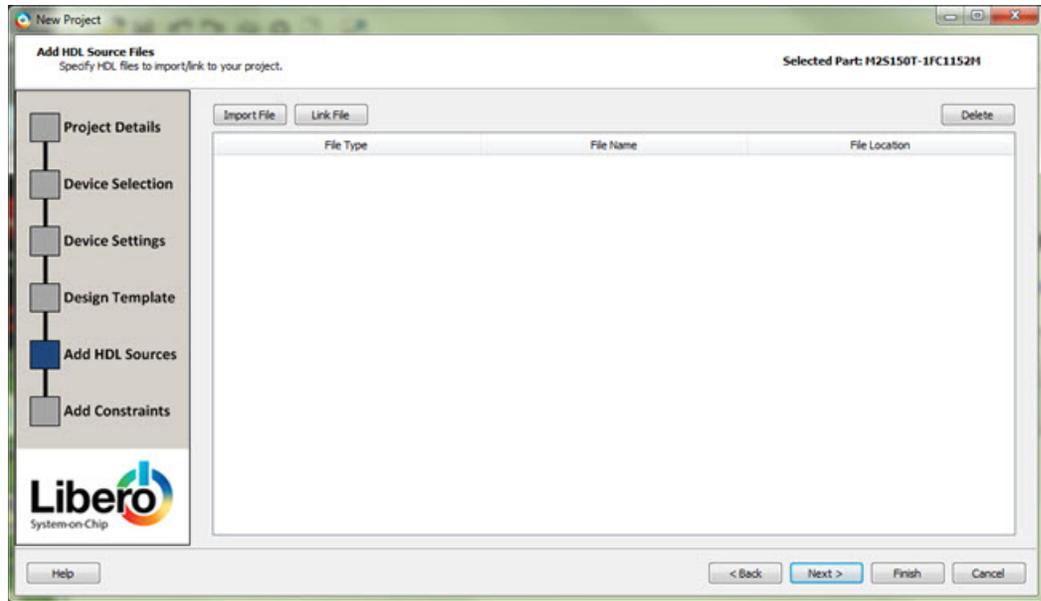


Figure 6 · New Project Creation Wizard - Add HDL Source Files Page

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

Link File – Navigate to the disk location of the HDL source. Select the HDL file and click **Open**. The HDL file is linked to the Libero Project. Use this option if the HDL source file is located and maintained outside of the Libero project.

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

When **Add HDL Sources** is completed, click on:

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

New Project Creation Wizard - Add Constraints

The Add Constraints page is where you add Timing constraints and Physical Constraints files to your Libero SoC project. The constraints file can be imported or linked to the Libero SoC Project.

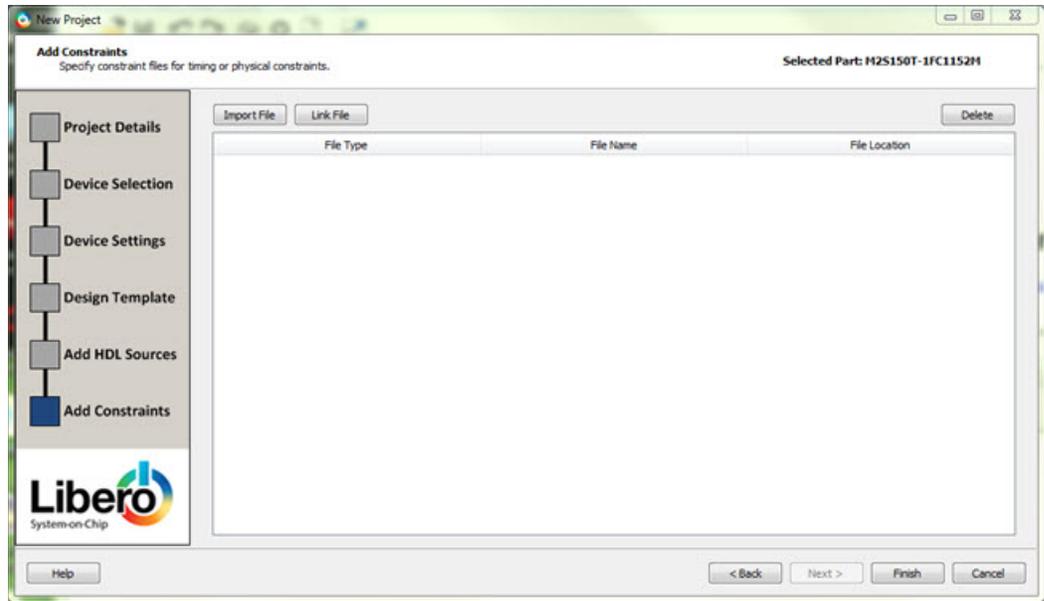


Figure 7 · New Project Creation Wizard – Add Constraints Page

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

Link File – Navigate to the disk location of the constraints file. Select the constraints file and click **Open**. The constraints file is linked to the Libero Project. Use this option if the constraint file is located and maintained outside of the Libero project.

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

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

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

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

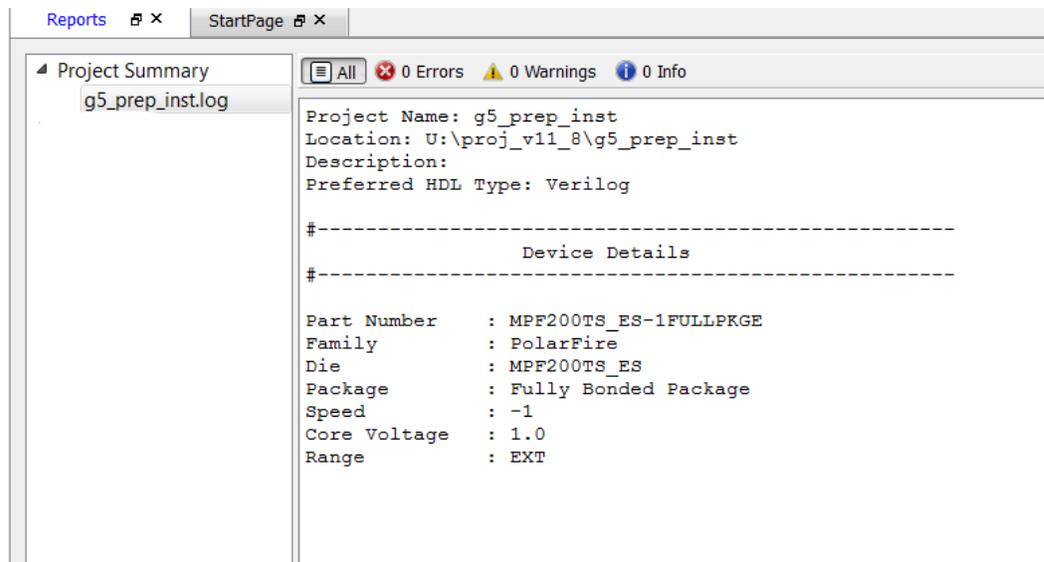


Figure 8 · Reports Tab

Create and Verify Design

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

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

SmartDesign

Introduction To 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](#) - The memory map shows 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. Press CTRL-W again if you need to see your Hierarchy or Catalog.

Working with Processor-Based Designs in SmartDesign

How do I connect my peripherals to the bus?

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

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

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

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

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

How do I view the Memory Map of my design?

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

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

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.

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).

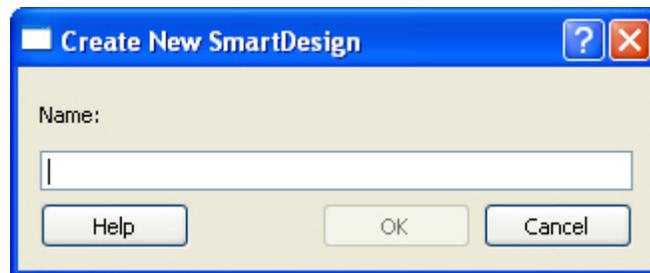


Figure 9 · Create New SmartDesign Dialog Box

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

Note: 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 t

he 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.

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.

Modify Memory Map Dialog Box

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

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

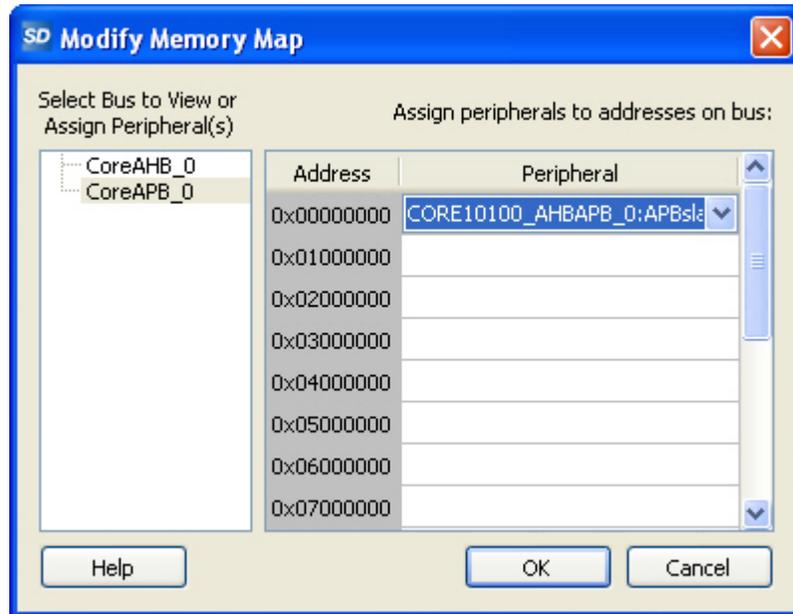


Figure 10 · Modify Memory Map Dialog Box

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

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

Canvas 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.

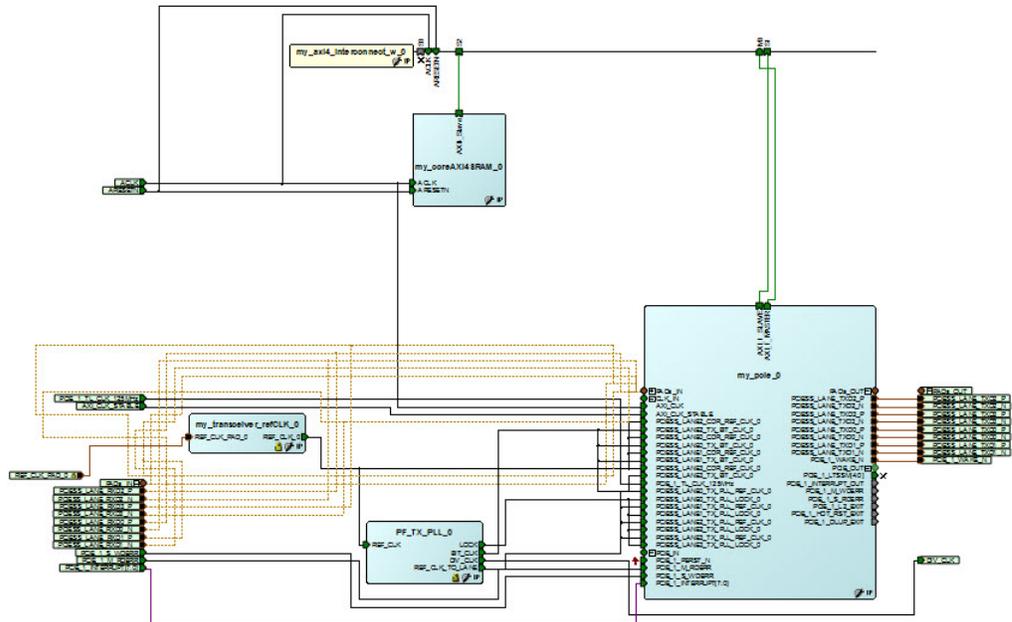


Figure 11 · SmartDesign Canvas

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.

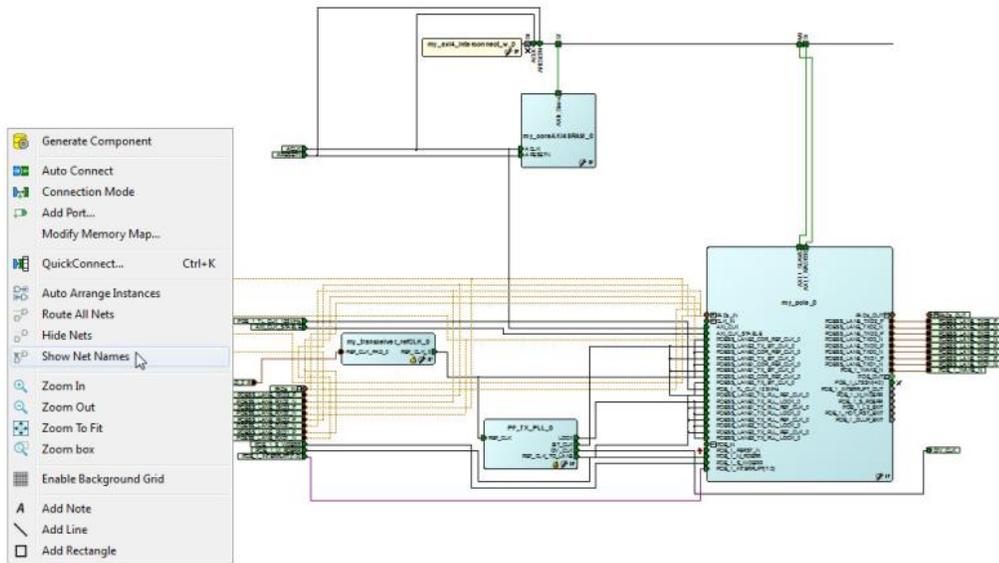


Figure 12 · Components in SmartDesign

The net names are shown in the Canvas.

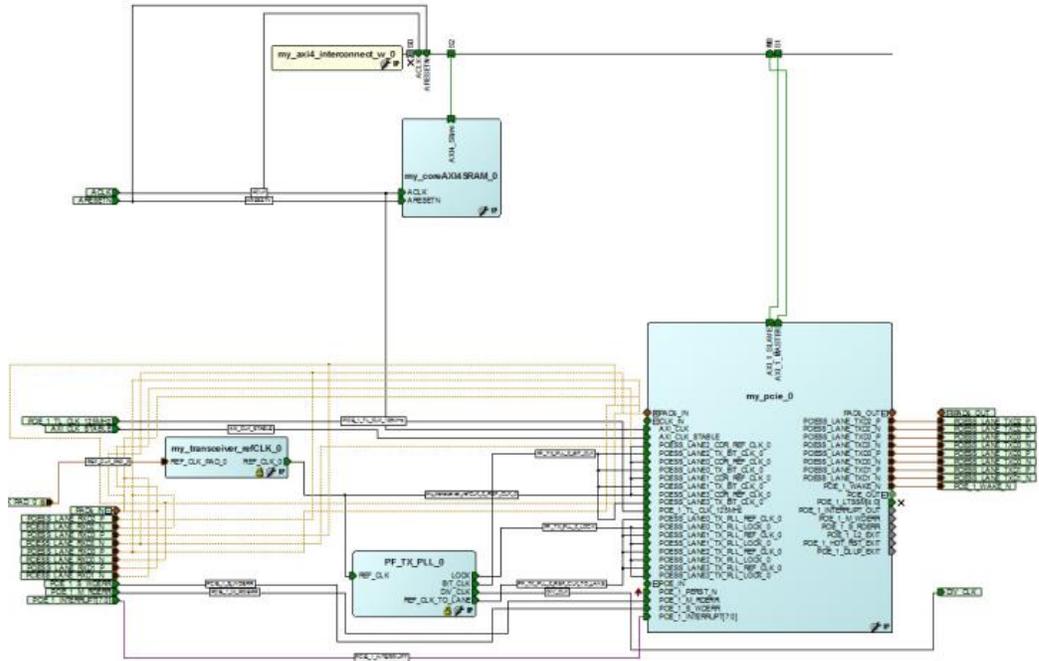


Figure 13 · Net Names shown in Canvas

Pin and Attribute Icons

Unconnected pins that do not require a connection are gray. 
 Unconnected pins that require a connection are red.

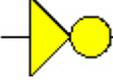
Unconnected pins that have a default tie-off are pale green.

Connected pins are green. 

Right-click a pin to assign an attribute.

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

Table 1 - Pin Attribute Icons

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

See the [Canvas Icons reference page](#) for definitions for each element on the Canvas.

Each connection made using a [bus interface](#) is shown in a separate connection known as a bus-interface net.

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

Name:	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_MSS_0_FAB_CLK	
smartfusion_project_MSS_0	FAB_CLK
COREAHBTOAPB3_0	HCLK
CoreAHBLite_0	HCLK
CoreAhbStram_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...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**.

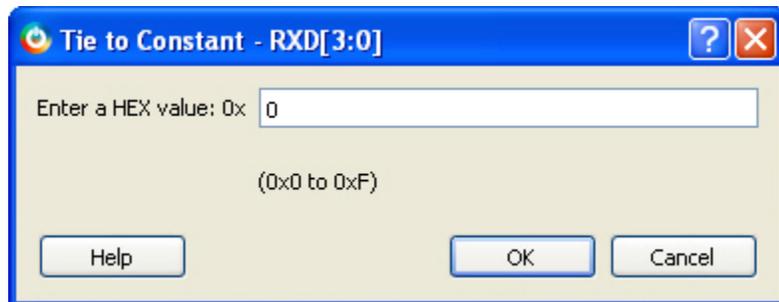


Figure 14 · Tie to Constant Dialog Box

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).

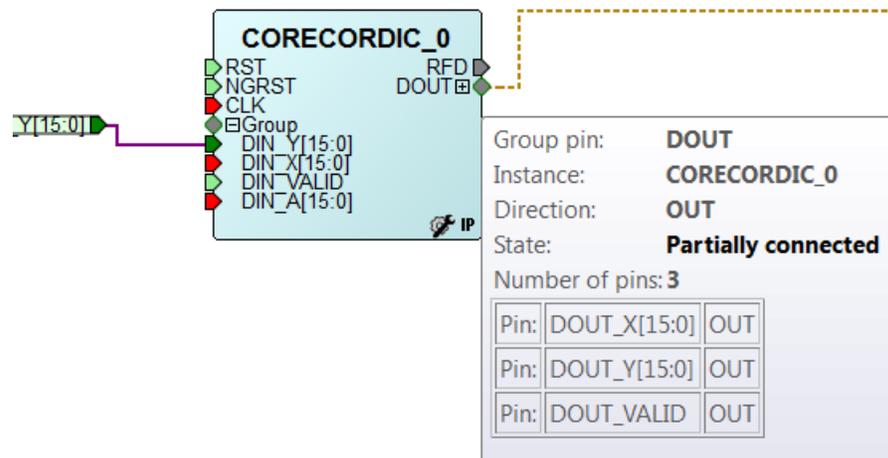


Figure 15 · Groups in an Instance on the Canvas

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.

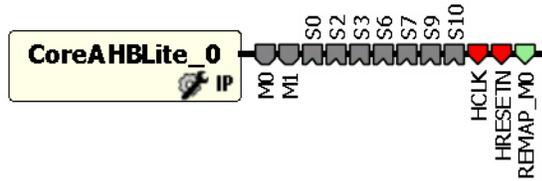


Figure 16 · Bus Instance in SmartDesign

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

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.

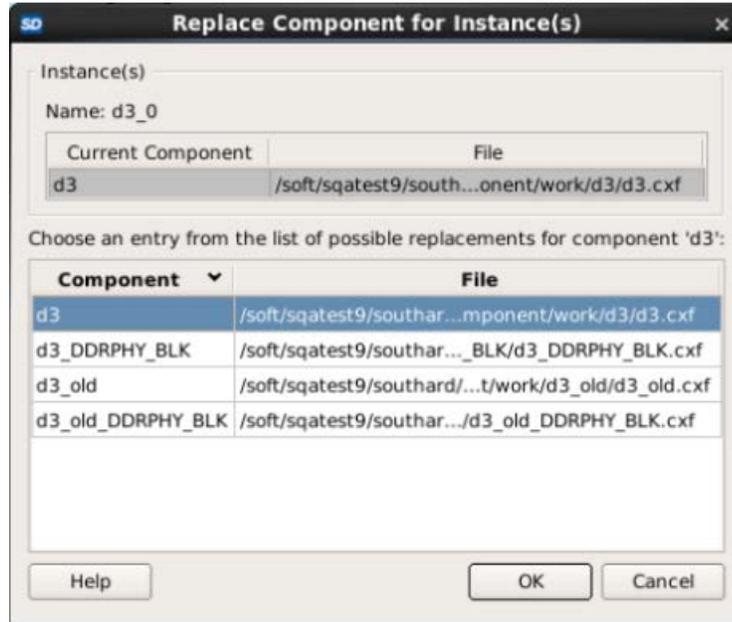


Figure 17 · Replace Component for Instance Dialog Box

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 18 · 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).

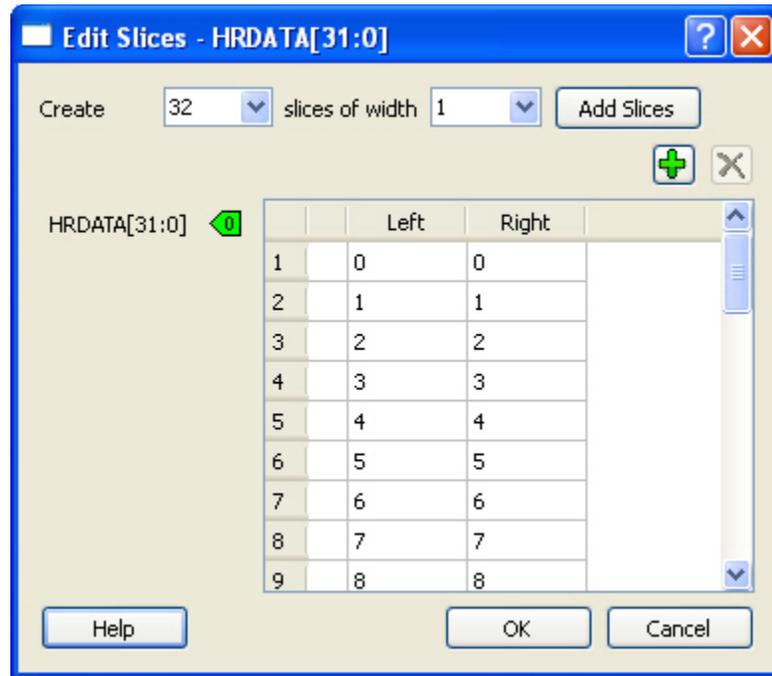


Figure 19 - Edit Slices Dialog Box

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 overwritten 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).

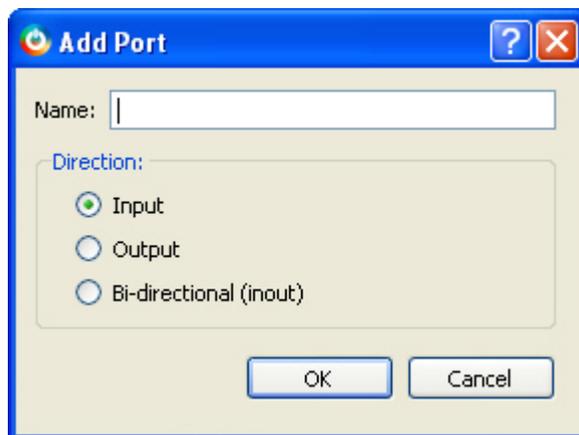


Figure 20 - Add New Port Dialog Box

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.

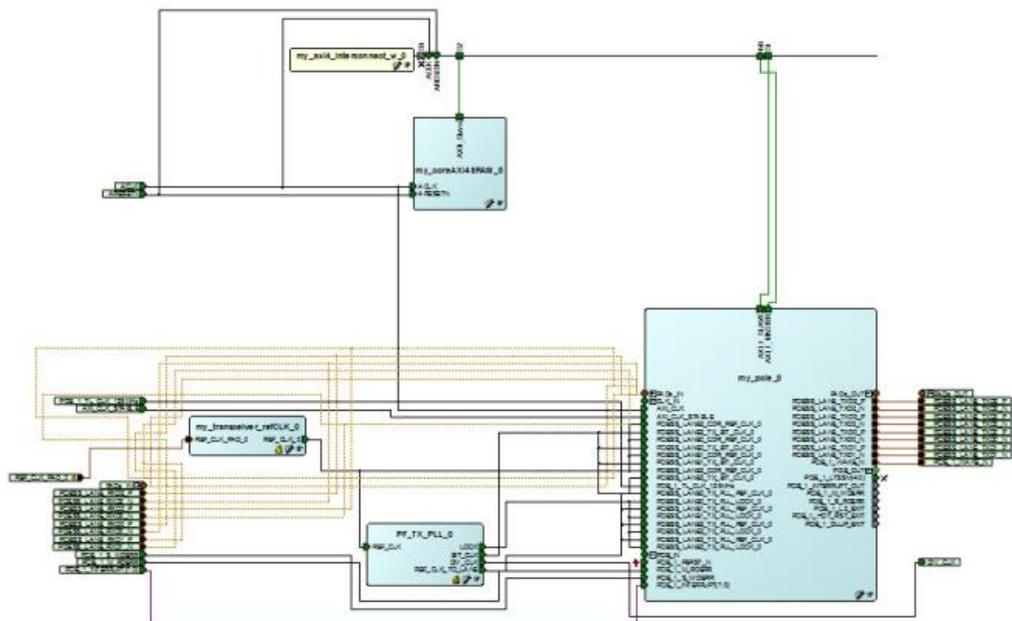


Figure 21 · Auto-Connected Cores

QuickConnect

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

To connect pins using QuickConnect:

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

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

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

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

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

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

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

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

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

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

- *UART*.*: show all pins of any instances that contain UART in the name
- MyUART_0.*: only show the pins of the "MyUART_0" instance
- *.p: show all pins in the design that contain the letter 'p'

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

The pin letters and icons in the QuickConnect dialog box are the same as the [Canvas icons](#) and communicate information about the pin. Inputs, Outputs and I/Os are 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

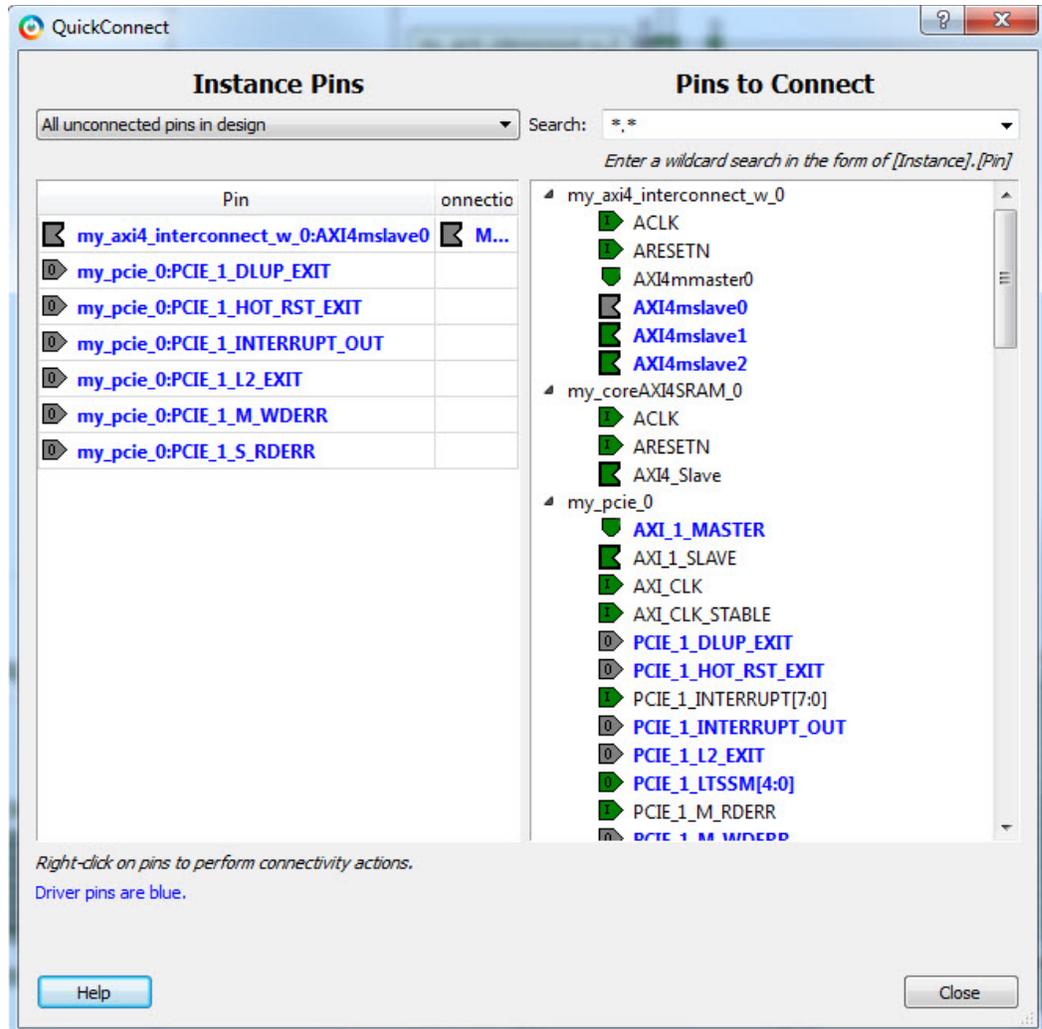


Figure 22 · QuickConnect Dialog Box

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 two 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.

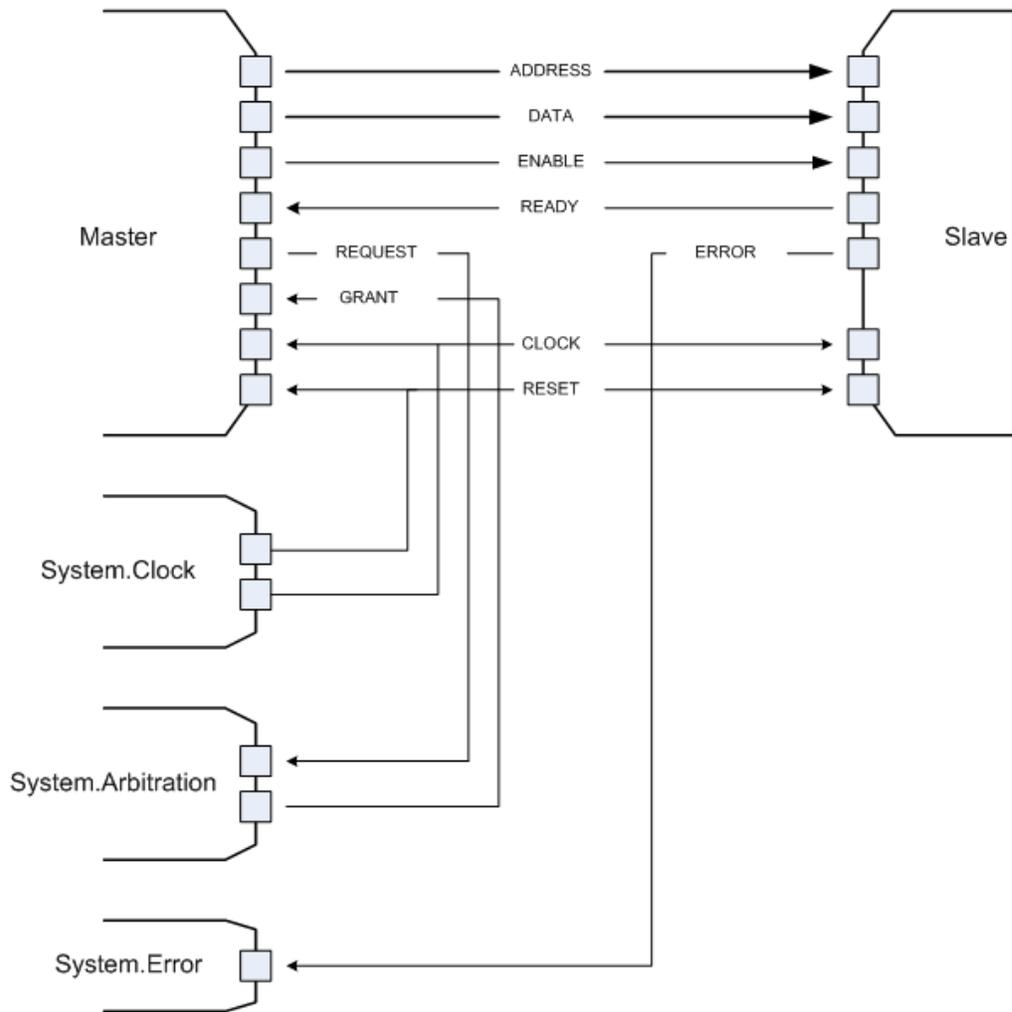


Figure 23 · Bus Definition

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.

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.

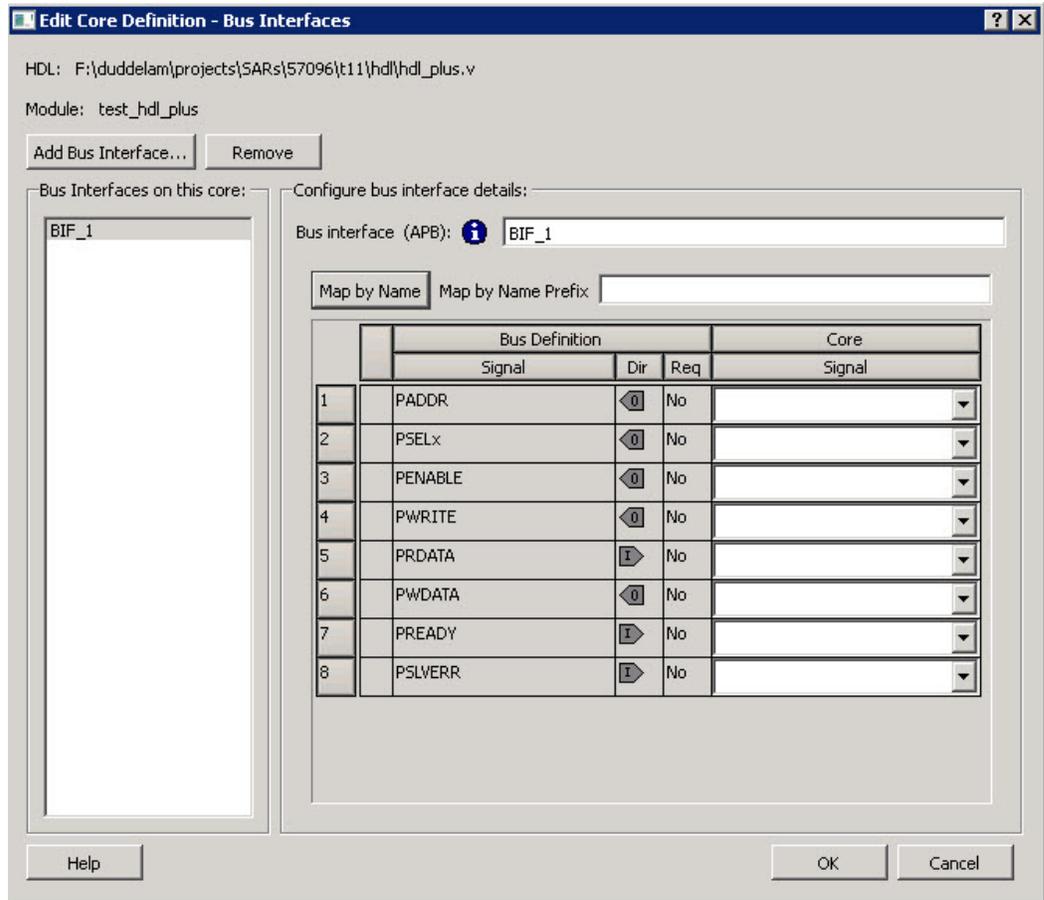


Figure 24 · Edit Core Definition Dialog Box

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:

- AHB – Master, Slave, Mirrored Master, MirroredSlave
- APB – Master, Slave, Mirroredmaster, MirroredSlave
- AXI – Master, Slave, MirroredMaster, MirrorSlave, System
- AXI 4 - Master, Slave, MirroredMaster, MirrorSlave

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).

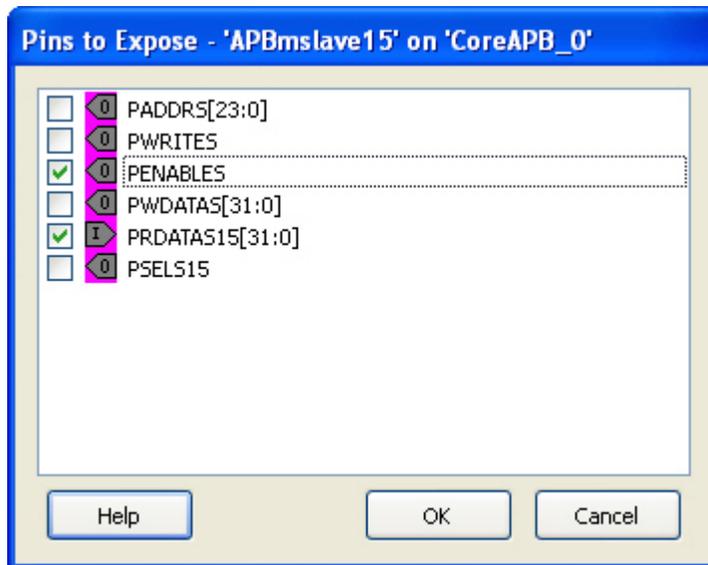


Figure 25 · Expose Driver Pin Dialog Box

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

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

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

Default Tie-offs with Bus Interfaces

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

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

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

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

Tying Off (Disabling) Unused Bus Interfaces

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

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

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

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

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

Required vs. Optional Bus Interfaces

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

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

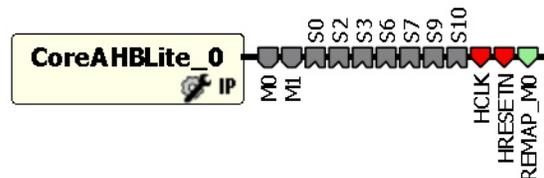


Figure 26 · Required Unconnected, Optional Unconnected, and Connected Bus Interfaces

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 $i = 1 \dots n$.

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

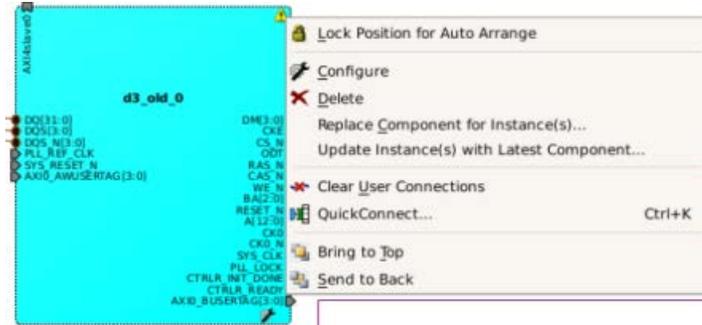


Figure 27 · Right-Click Menu - Replace Component for this Instance

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).

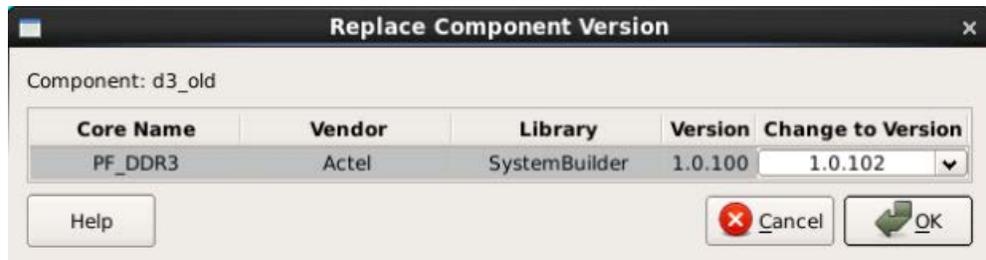


Figure 28 · Replace Component Version Dialog Box

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

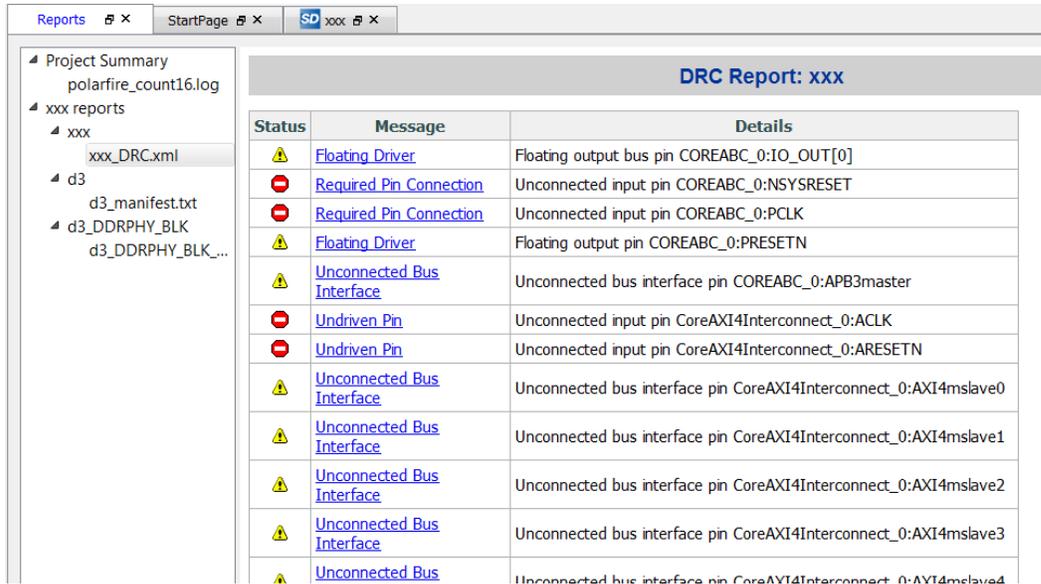
When you modify the memory content of a core such as RAM with Initialization that is used by a Flash Memory core, the Flash Memory core indicates that one of its dependent components has changed and that it needs to be regenerated. This indication will be shown in the Hierarchy or Files Tab .

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

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



Status	Message	Details
	Floating Driver	Floating output bus pin COREABC_0:IO_OUT[0]
	Required Pin Connection	Unconnected input pin COREABC_0:NSYSRESET
	Required Pin Connection	Unconnected input pin COREABC_0:PCLK
	Floating Driver	Floating output pin COREABC_0:PRESETN
	Unconnected Bus Interface	Unconnected bus interface pin COREABC_0:APB3master
	Undriven Pin	Unconnected input pin CoreAXI4Interconnect_0:ACLK
	Undriven Pin	Unconnected input pin CoreAXI4Interconnect_0:ARESETN
	Unconnected Bus Interface	Unconnected bus interface pin CoreAXI4Interconnect_0:AXI4mslave0
	Unconnected Bus Interface	Unconnected bus interface pin CoreAXI4Interconnect_0:AXI4mslave1
	Unconnected Bus Interface	Unconnected bus interface pin CoreAXI4Interconnect_0:AXI4mslave2
	Unconnected Bus Interface	Unconnected bus interface pin CoreAXI4Interconnect_0:AXI4mslave3
	Unconnected Bus Interface	Unconnected bus interface pin CoreAXI4Interconnect_0:AXI4mslave4

Figure 29 · 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 because of silicon limitations. You must remove the extra instances.
- Incompatible Family Configuration** – The instance is not configured to work with this project’s Family setting. Either it is not supported by this family or you need to re-instantiate the core.
- Incompatible Die Configuration** – The instance is not configured to work with this project’s Die setting. Either it is not supported or you need to reconfigure the Die configuration.
- No RTL License, No Obfuscated License, No Evaluation License** – You do not have the proper license to generate this core. [Contact Microsemi SoC](#) to obtain the necessary license.
- No Top level Ports** - There are no ports on the top level. To auto-connect top-level ports, right-click the Canvas and choose Auto-connect

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.

SmartDesign Reference

SmartDesign Menu

Command	Icon	Function
Generate Component		Generates the SmartDesign component
Auto Connect		Auto-connects instances

Command	Icon	Function
Connection Mode		Toggles connection mode on or off
Add Port		Opens the Add Port dialog box, adds a port to the top SmartDesign component
QuickConnect		Opens the QuickConnect dialog box, enables you to view, find and connect pins
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	A	Adds text to your Canvas
Add Line		Enables you to add a line to the Canvas
Add Rectangle		Enables you to add a rectangle to the Canvas

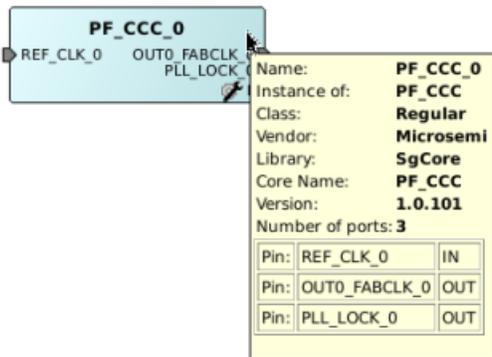
SmartDesign Glossary

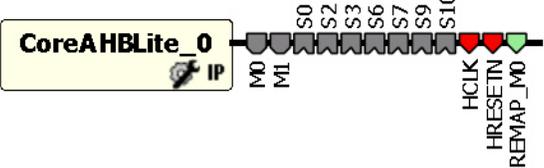
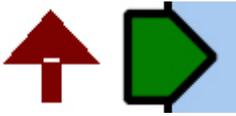
Term	Description
BIF	Abbreviation for bus interface.
bus	An array of scalar ports or pins, where all scalars have a common base name and have unique indexes in the bus.
Bus Definition	Defines the signals that comprise a bus interface. Includes which signals are present on a master, slave, or system interface, signal direction, width, default value, etc. A bus definition is not specific to a logic or design component but is a type or protocol.
Bus Interface	Logical grouping of ports or pins that represent a single functional purpose. May contain both input and output, scalars or busses. A bus interface is a specific mapping of a bus definition onto a component instance.
Bus Interface Net	A connection between 2 or more compatible bus interfaces.
Canvas	Block diagram, connections represent data flow; enables you to connect instances of components in your design.
Component	Design element with a specific functionality that is used as a building block to create a SmartDesign core. A component can be an HDL module, non-IP core generated from the Catalog, SmartDesign core, Designer Block, or IP core. When you add a component to your design, SmartDesign creates a specific instance of that component.
Component Declaration	VHDL construct that refers to a specific component.
Component Port	An individual port on a component definition.
Driver	A driver is the origin of a signal on a net. The input and slave BIF ports of the top-level or the output and Master BIF ports from instances are drivers.
Instance	A specific reference to a component/module that you have added to your design. You may have multiple instances of a single component in your design. For each specific instance, you usually will have custom connections that differ from other instances of the same component.
Master Bus Interface	The bus interface that initiates a transaction (such as a read or write) on a bus.

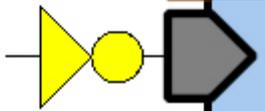
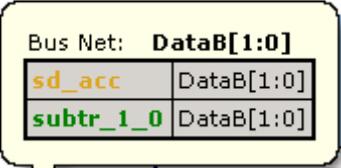
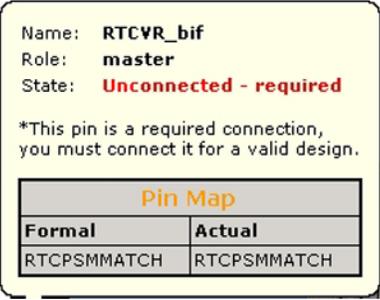
Term	Description
Net	Connection between individual pins. Each net contains a single output pin and one or more input pins, or one or more bi-directional pins. Pins on the net must have the same width.
PAD	The property of a port that must be connected to a design's top level port. PAD ports will eventually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

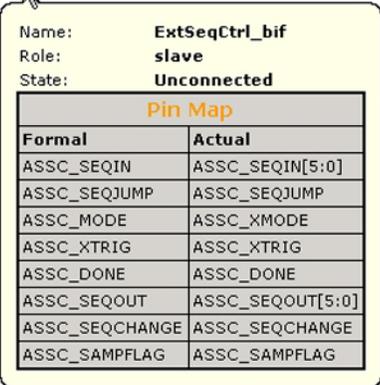
Canvas Icons

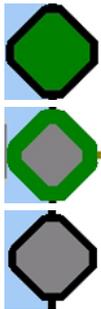
Hover your pointer over any icon in the SmartDesign Canvas view to display details.

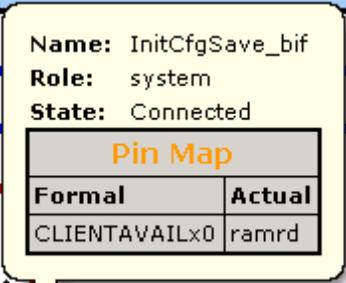
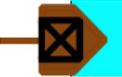
Icon	Description
	Representation of an instance in your design. An instance is a component that has been added to your SmartDesign component. The name of the instance appears at the top and the name of the

Icon	Description
	<p>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, and HDL. The instance icon at left indicates a Microsemi SoC core.</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>
	<p>Optional unconnected pin. Required pins are red.</p>
	<p>Connected pin</p>
	<p>Pin with default Tie Off</p>
	<p>Pin tied low</p>
	<p>Pin tied high</p>

Icon	Description						
	Pin inverted						
	Pin marked as unused						
	Pin tied to constant						
 <table border="1" data-bbox="342 682 641 766"> <tr> <td colspan="2">Bus Net: DataB[1:0]</td> </tr> <tr> <td>sd_acc</td> <td>DataB[1:0]</td> </tr> <tr> <td>subtr_1_0</td> <td>DataB[1:0]</td> </tr> </table>	Bus Net: DataB[1:0]		sd_acc	DataB[1:0]	subtr_1_0	DataB[1:0]	Bus Net details.
Bus Net: DataB[1:0]							
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>						
 <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" data-bbox="342 1675 690 1766"> <thead> <tr> <th colspan="2">Pin Map</th> </tr> <tr> <th>Formal</th> <th>Actual</th> </tr> </thead> <tbody> <tr> <td>RTCPSMMATCH</td> <td>RTCPSMMATCH</td> </tr> </tbody> </table>	Pin Map		Formal	Actual	RTCPSMMATCH	RTCPSMMATCH	<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)</p>
Pin Map							
Formal	Actual						
RTCPSMMATCH	RTCPSMMATCH						

Icon	Description																				
	and the Actual, or representative name assigned by the user (CLKOUT).																				
	<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>																				
 <table border="1" data-bbox="342 989 690 1266"> <thead> <tr> <th colspan="2">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>	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, ASSC_MODE) and the Actual, or representative name assigned by the user (ASSC_XMODE).</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 style="border: 1px solid black; padding: 5px;"> <p>Name: AHBslave2</p> <p>Role: mirroredSlave</p> <p>State: Connected</p> <p style="text-align: center;">Pin Map</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%;">Formal</th> <th style="width: 50%;">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</p>																												

Icon	Description
	<p>master and slave interfaces.</p> <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>
	<p>Pad port icon; indicates a hardwired chip-level pin</p>

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
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.

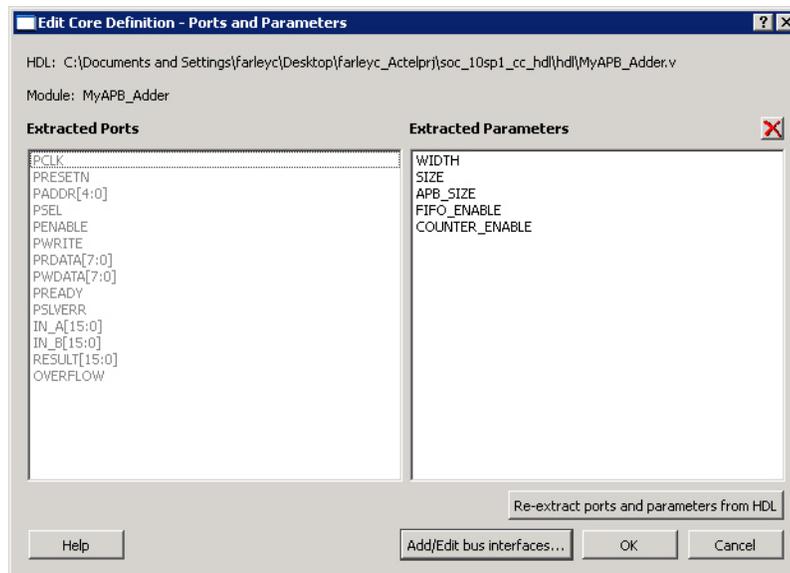


Figure 30 · Edit Core Definition - Ports and Parameters Dialog Box

4. (Optional) Click **Add/Edit Bus Interfaces** to [add bus interfaces](#) to your core.

After you have specified the information, your HDL turns into an HDL+ icon in the Design Hierarchy. Click and drag your HDL+ module from the Design Hierarchy to the **Canvas**.

If you added bus interfaces to your HDL+ core, then it will show up in your SmartDesign with a bus interface pin that can be used to easily connect to the appropriate bus IP core.

If your HDL+ has configurable parameters then double-clicking the object on the Canvas (or right-click and select **Configure**) invokes a configuration dialog that enables you to set these values. On generation, the specific configuration values per instance are written out to the SmartDesign netlist.

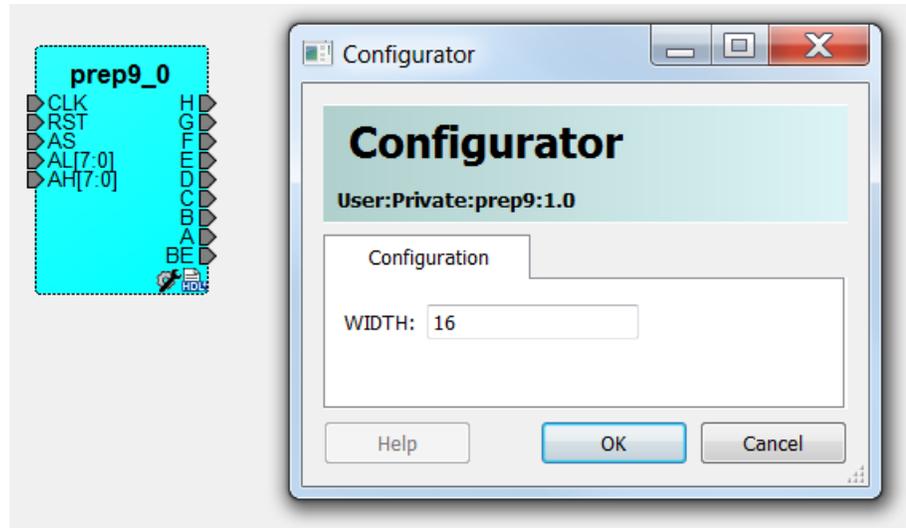


Figure 31 · HDL+ Instance and Configuration Dialog Box

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**.

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

Mixed-HDL Support in Libero SoC

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

When you [create a project](#), you must select a preferred language. The HDL files generated in the flow (such as the post-layout netlist for simulation) are created in the preferred language.

The language used for simulation is the same language as the last compiled testbench. (For example, if tb_top is in Verilog, <fam>.v is compiled.)

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

HDL Testbench

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

HDL Testbench automatically instantiates the selected SmartDesign into the Component.

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

HDL Type

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

Name

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

Clock Period (ns)

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

Set as Active Stimulus sets the HDL Testbench as the stimulus file to use for simulations. The active stimulus file/testbench is included in the run.do file that Libero generates to drive the simulation. Setting one testbench as the Active Stimulus is necessary when there are multiple testbenches in the stimulus hierarchy.

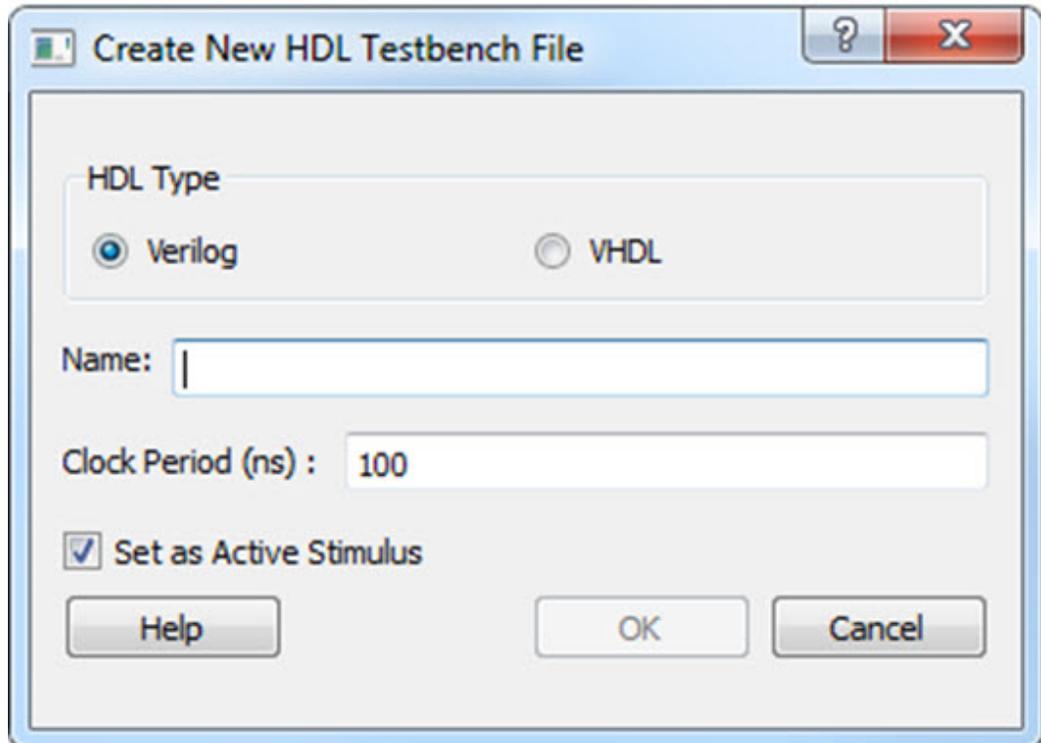


Figure 32 · Create New HDL Testbench File Dialog Box

```

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

```

Figure 33 · 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 Pro.

ModelSim™ ME Pro 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 Pro 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>. 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 resolution is 1ps.
- **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 the Design Flow window, Implement Design > Verify Post-Synthesis Implementation > Simulate) and choose **Organize Input Files > Organize Simulation Files** from the right-click menu.

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

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

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

To remove a testbench: To remove or change the file(s) in the Associated Source Files list box, select the file(s) and click **Remove**.
3. When you are satisfied with the Associated Simulation Files list, click **OK**.
4. To start ModelSim ME Pro, right-click **Simulate** in the Design Hierarchy window and choose **Open Interactively**.

ModelSim starts and compiles the appropriate source files. When the compilation completes, the simulator runs for 1 s and the Wave window opens to display the simulation results.
5. Scroll in the Wave window to verify that the logic of your design functions as intended. Use the zoom buttons to zoom in and out as necessary.
6. From the **File** menu, select **Quit**.

Libero SoC Constraint Management

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

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

Invocation of Constraint Manager From the Design Flow Window

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

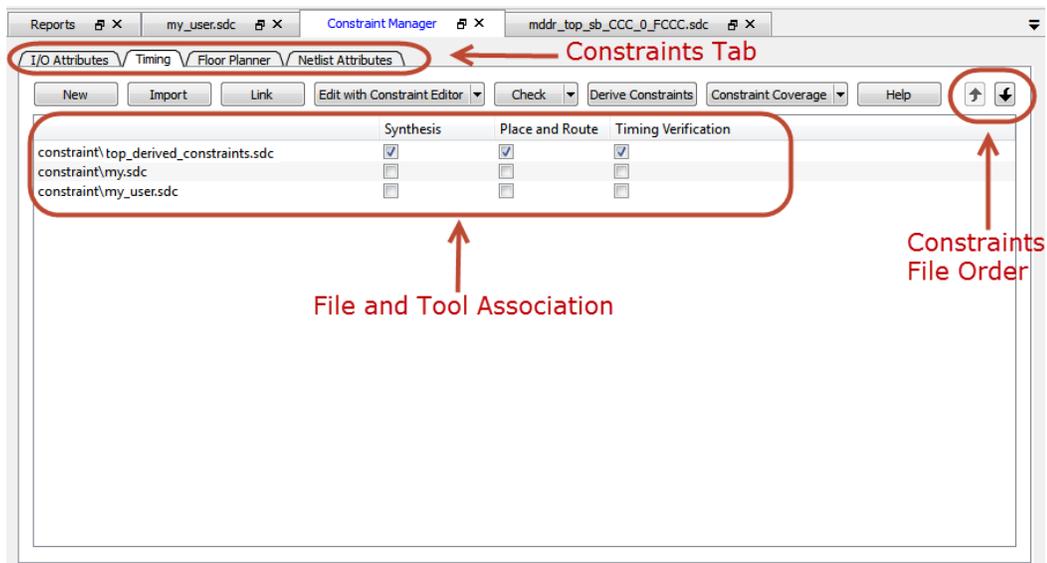


Figure 34 · Constraint Manager

Libero SoC Design Flow

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

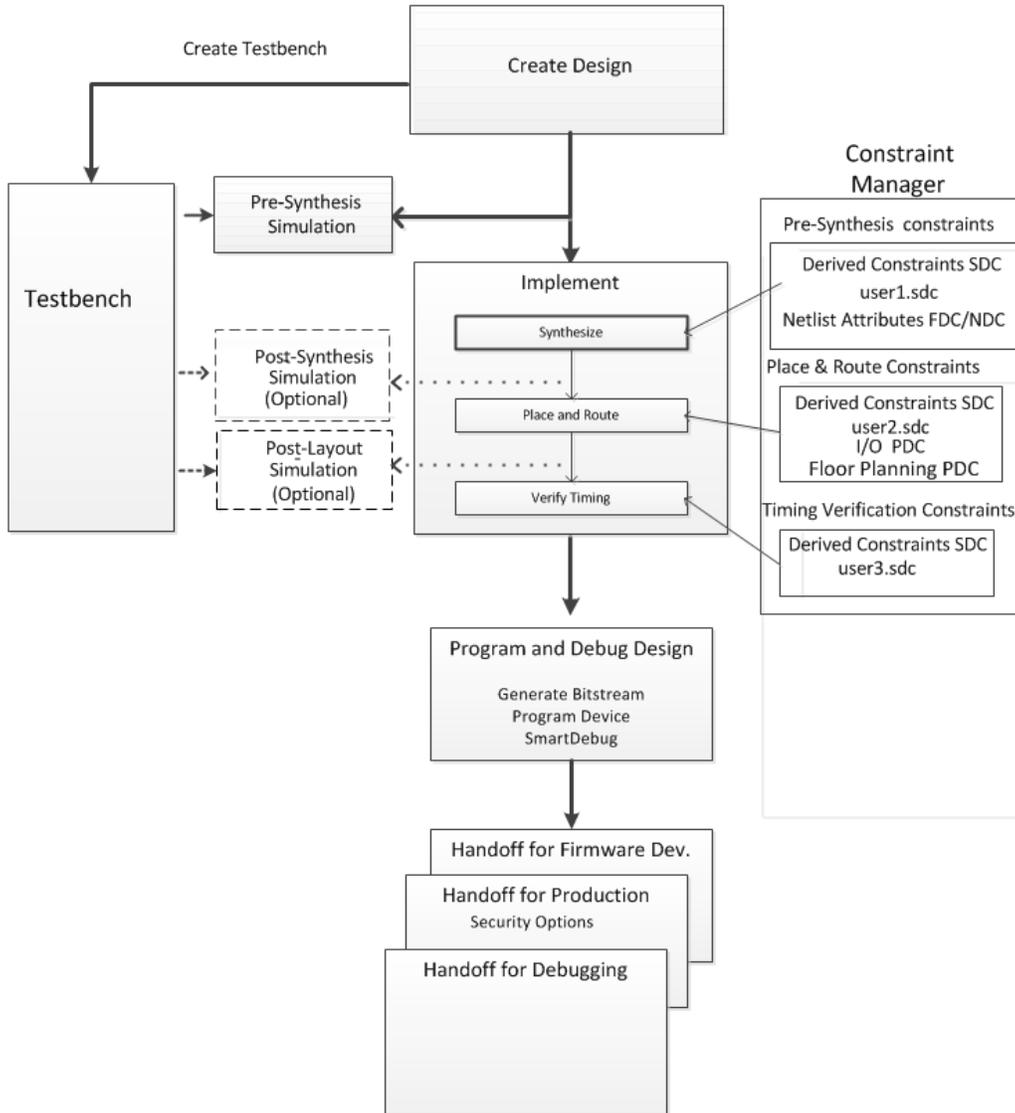


Figure 35 - Constraint Manager in Libero SoC Design Flow

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 or create a new FDC file in the Text Editor (**Netlist Attributes > New > Create**)

New Synplify Netlist Constraint). After the FDC file is created or imported, click the checkbox under synthesis to associate the FDC file with Synthesis.

Compile Netlist Constraints (*.ndc)

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

SDC Timing Constraints (*.sdc)

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

After the synthesis step, you may click **Edit with Constraint Editor > Edit Synthesis Constraints** to 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 (*.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.

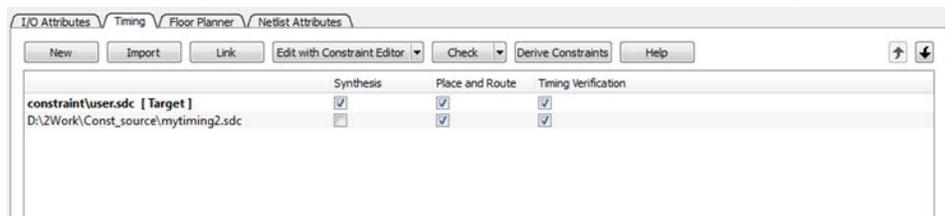


Figure 36 · Constraint File and Tool Association

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

Derive Constraints in Timing Tab

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

Create New Constraints

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

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

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

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

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

- I/O Attributes constraint file (<proj>\io*.pdc) in the I/O Attributes tab
 - Floorplanning constraints (<proj>\fp*.pdc) in the Floor Planner tab
 - Timing constraints (<proj>\constraints*.sdc) in the Timing tab
6. (Optional) Double-click the constraint file in the Constraint Manager to open and add more constraints to the file.

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

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

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

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

1. Select the Tab that corresponds to the type of constraint you want to create.
2. Click Edit to open the Interactive Tools. The Interactive Tool that Libero SoC opens varies with the constraint type:
 - I/O Editor to edit/create I/O Attribute Constraints
 - 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.

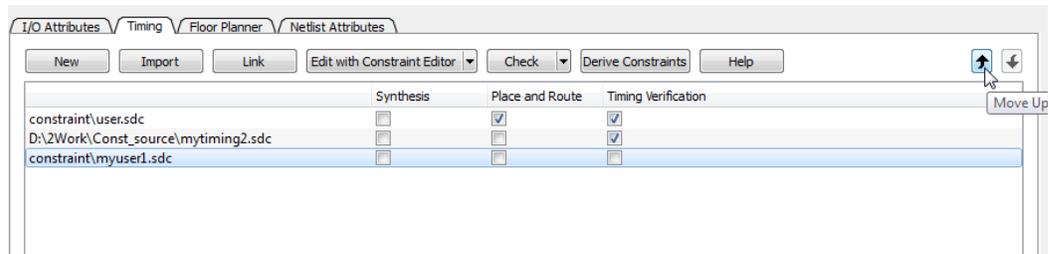


Figure 37 · Move constraint file Up or Down

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.

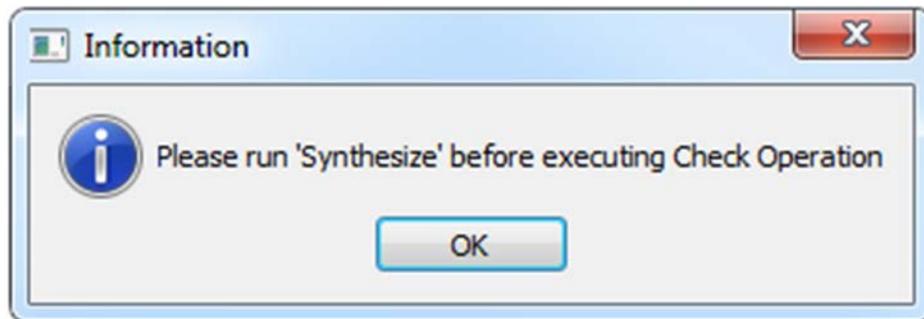
Check a Constraint File

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

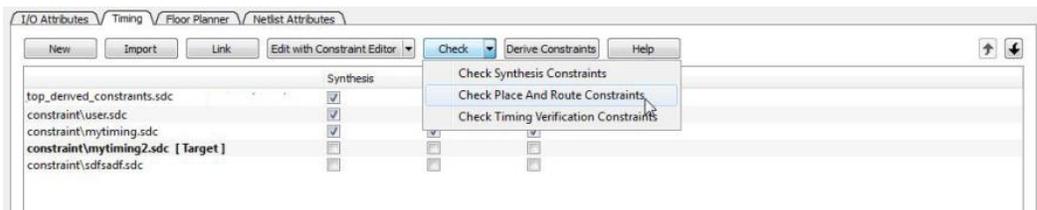


Figure 38 · Check Constraints

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

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

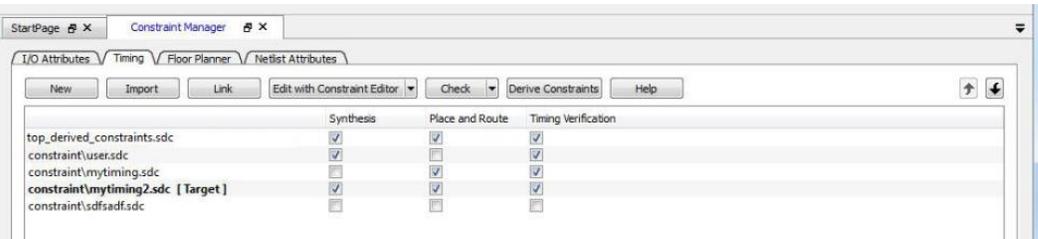


Figure 39 · Timing Constraints SDC file and Tool Association

When a constraint file is checked, the Constraint Manager:

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

Check Result

If the check is successful, this message pops up.

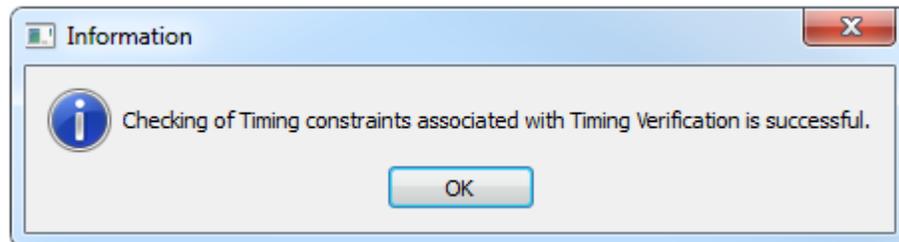


Figure 40 · Check Successful Message

If the check fails, this error message pops up.

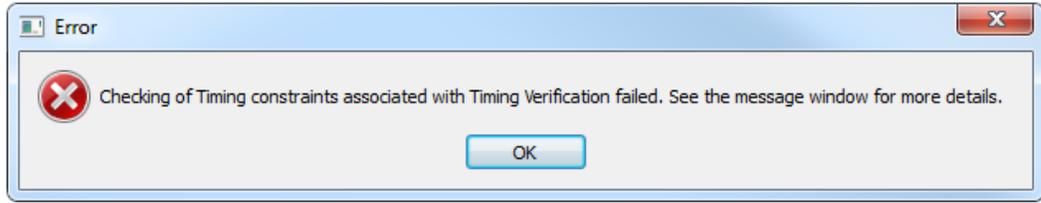


Figure 41 · Check Fails Message

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 Route	Post-Synthesis	ADL Netlist	placer_sdc_check.log
	Timing Verifications	Post-Synthesis	ADL Netlist	timing_sdc_check.log
Netlist Attributes (*.fdc)	Synthesis	Pre-Synthesis	RTL Netlist	*cck.srr file
Netlist Attributes (*.ndc)	Synthesis	Pre-Synthesis	RTL Netlist	Libero Log Window

Edit a Constraint File

The Edit button in the Constraint Manager allows you to:

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

To edit a constraint file

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

1. Select the tab for the constraint type to edit. An Interactive Tool is opened to make the edits.
 2. Click Edit.
- All constraint files associated with the tool are edited. Files not associated with the tool are not edited.

- When a constraint file is edited, the constraints in the file are read into the Interactive Tool.
- Different Interactive Tools are used to edit different constraints/different files:
 - I/O Editor to edit I/O Attributes (<proj>\io*.pdc). For details, refer to the I/O Editor 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.

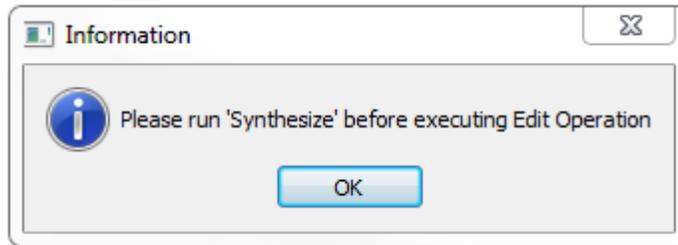


Figure 42 · Pop-up Message

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

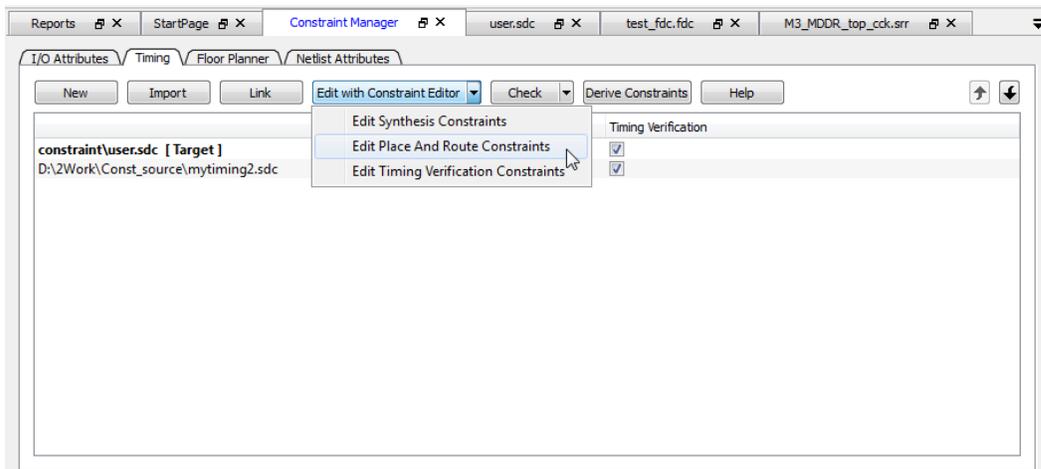


Figure 43 · Edit Drop-down Menu

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

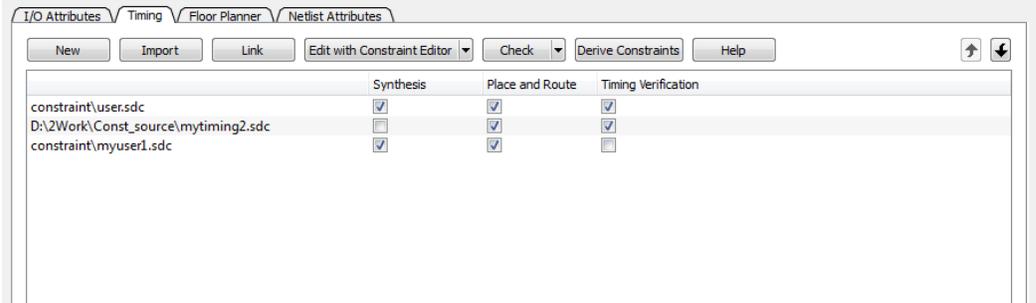


Figure 44 · Timing Constraint File and Tool Association

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.

Constraint Types

Libero SoC manages four different types of constraints:

- **I/O Attributes Constraints** – Used to constrain placed I/Os in the design. Examples include setting I/O standards, I/O banks, and assignment to Package Pins, output drive, and so on. These constraints are used by Place and Route.
- **Timing Constraints** – Specific to the design set to meet the timing requirements of the design, such as clock constraints, timing exception constraints, and disabling certain timing arcs. These constraints are passed to Synthesis, Place and Route, and Timing Verification.
- **Floor Planner Constraints** – Non-timing floorplanning constraints created by the user or Chip Planner and passed to Place and Route to improve Quality of Routing.
- **Netlist Attributes** - Microsemi-specific attributes that direct the Synthesis tool to synthesize/optimize the, leveraging the architectural features of the Microsemi devices. Examples include setting the fanout limits, specifying the implementation of a RAM, and so on. These constraints are passed to the Synthesis tool only.

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

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

Constraint Manager – I/O Attributes Tab

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

Available actions are:

- **New** – Creates a new I/O PDC file and saves it into the <Project_location>\constraint\io folder. There are two options:
 - Create New I/O Constraint
 - Create New I/O Constraint From Root Module -- This will pre-populate the PDC file with information from the Root Module

Having selected the create method:

- When prompted, enter the name of the constraint file.
- The file is initially opened in the text editor for user entry.
- **Import** – Imports an existing I/O PDC file into the Libero SoC project. The I/O PDC file is copied into the <Project_location>\constraint\io folder.
- **Link** – Creates a link in the project's constraint folder to an existing I/O PDC file (located and maintained outside of the Libero SoC project).
- **Edit with I/O Editor** – Opens the I/O Editor tool to modify the I/O PDC file(s) associated with the Place and Route tool.

- **Check** – Checks the legality of the PDC file(s) associated with the Place and Route tool against the gate level netlist.

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

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

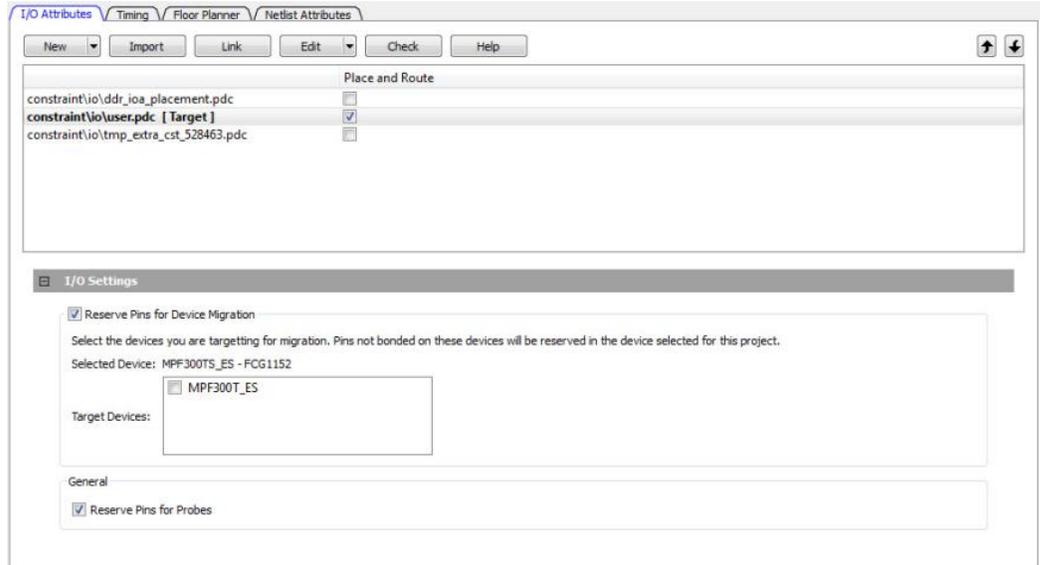


Figure 45 · Constraint Manager – I/O Attributes Tab

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

- **Set/UnSet as Target** – Sets or clears the selected file as the target to store new constraints created in the I/O Editor tool. Newly created constraints only go into the target constraint file. Only one file can be set as target.
- **Open in Text Editor** – Opens the selected constraint file in the Libero Text Editor.
- **Clone** – Copies the file to a file with a different name. The original file name and its content remain intact.
- **Rename** – Renames the file to a different name.
- **Copy File Path** - Copies the file path to the clipboard.
- **Delete From Project and Disk** – Deletes the file from the project and from the disk.
- **Unlink: Copy file locally** – Removes the link and copies the file into the <Project_location>\constraint\io folder. This selection is available only for linked constraints files.

File and Tool Association

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

I/O Settings

Reserve Pins for Device Migration – This option allows you to reserve pins in the currently selected device that are not bonded in a device or list of devices you may later decide to migrate your design to. Select the target device(s) you may migrate to later to ensure that there will be no device/package incompatibility if you migrate your design to that device.

Reserve Pins for Probes – Check this box if you plan to use live probes when debugging your design with SmartDebug.

Constraint Manager – Timing Tab

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

Available actions are:

- **New** – Creates a new timing SDC file and saves it into the <Project_location>\constraint folder.
 - When prompted, enter the name of the constraint file.
 - The file is initially opened in the text editor for user entry.
- **Import** – Imports an existing timing SDC file into the Libero SoC project. The timing SDC file is copied into the <Project_location>\constraint folder.
- **Link** – Creates a link in the project's constraint folder to an existing timing SDC file (located and maintained outside of the Libero SoC project).
- **Edit with Constraint Editor** – Opens the [SmartTime Timing Constraint Editor](#) to modify the SDC file(s) associated with one of the three tools:
 - **Synthesis** – When selected, the timing SDC file(s) associated with the Synthesis tool is loaded in the constraints editor for editing.
 - **Place and Route** - When selected, the timing SDC file(s) associated with the Place and Route tool is loaded in the constraints editor for editing.
 - **Timing Verification** - When selected, the timing SDC file(s) associated with the Timing Verification tool is loaded in the constraints editor for editing.
- **Check** – Check the legality of the SDC file(s) associated with one of the three tools described below:
 - **Synthesis** – The check is performed against the pre-synthesis HDL design.
 - **Place and Route** – The check is performed against the post-synthesis gate level netlist.
 - **Timing Verification** – The check is performed against the post-synthesis gate level netlist.
- **Derive Constraints** – When clicked, Libero generates a timing SDC file based on user configuration of IP core, components and component SDC. It generates the create_clock and create_generated_clock SDC timing constraints. This file is named <top_level_> derived_constraints.sdc. The component SDC and the generated <root>_derived_constraint.sdc files are dependent on the IP cores and vary with the device family.

Examples:

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

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

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

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

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

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

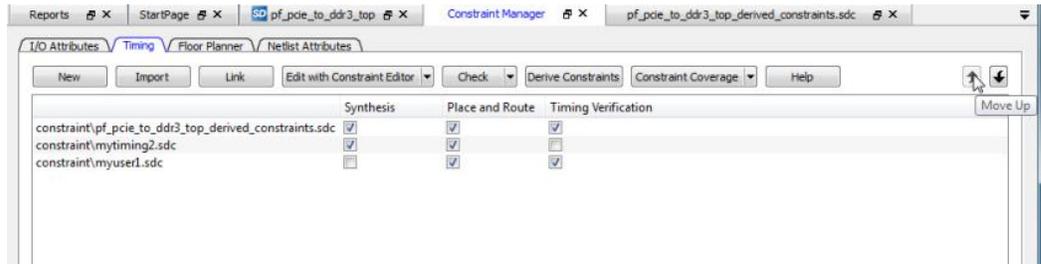


Figure 46 · Constraint Manager – Timing Tab

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

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

File and Tool Association

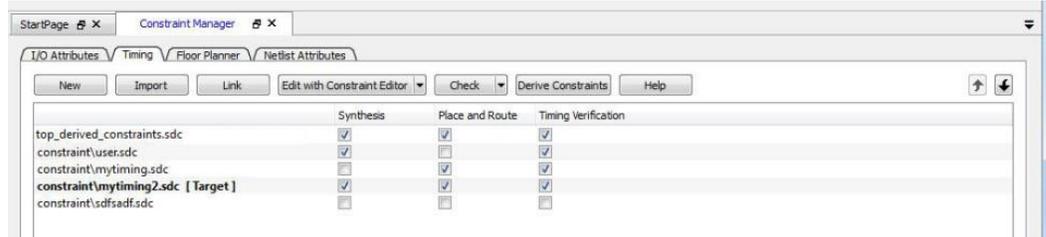
Each timing constraint file can be associated or disassociated with any one, two, or all three of the following tools:

- Synthesis
- Place and route
- Timing Verification

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

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

Example



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

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

Derived Constraints

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

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

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

To generate SDC timing constraints for IP cores:

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

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

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

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

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.

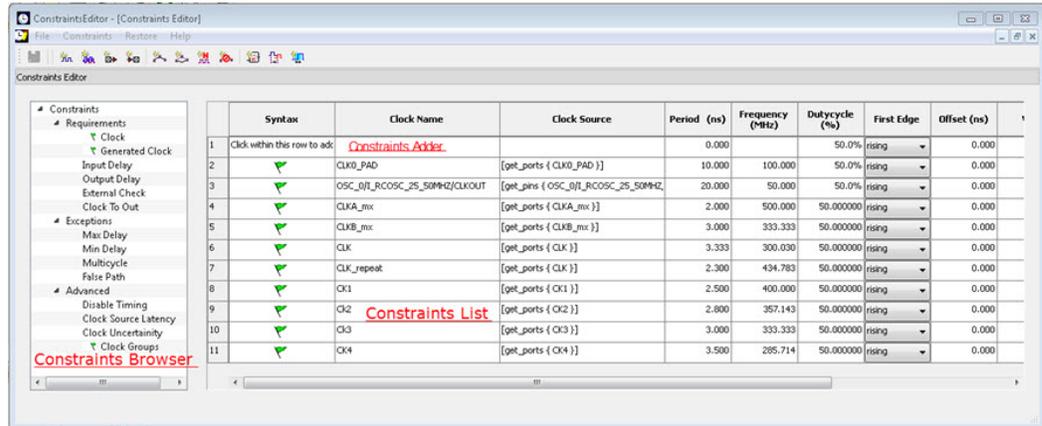


Figure 47 · Constraints Editor

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

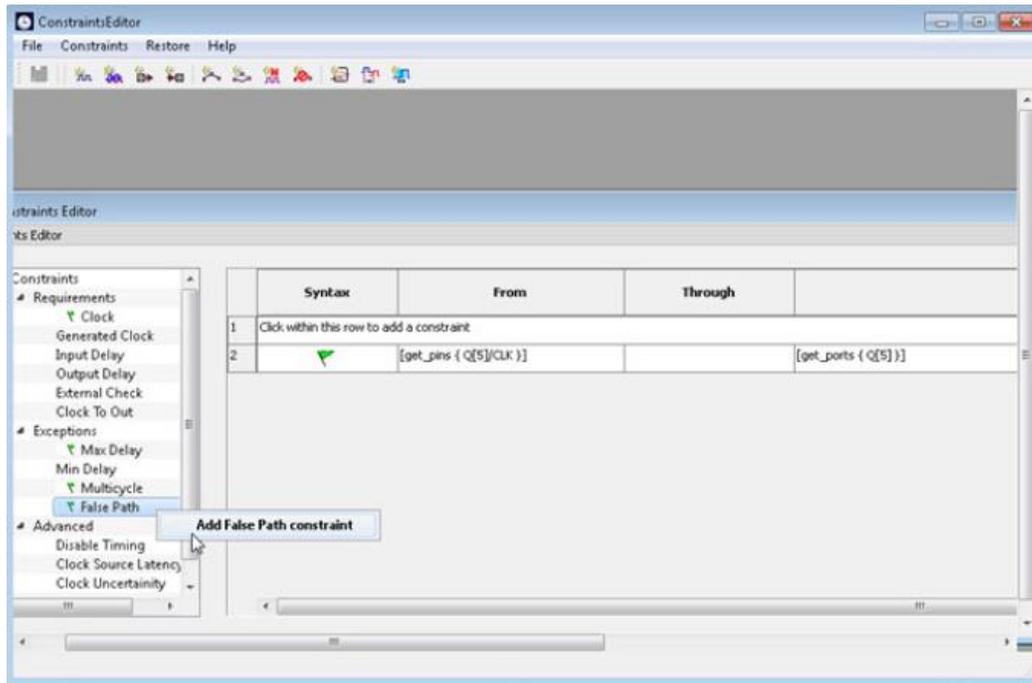
Across the top of the Constraint Editor is a list of icons you can click to add constraints. Tooltips are available to identify the constraints.

Icon	Name
	Add Clock Constraint
	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

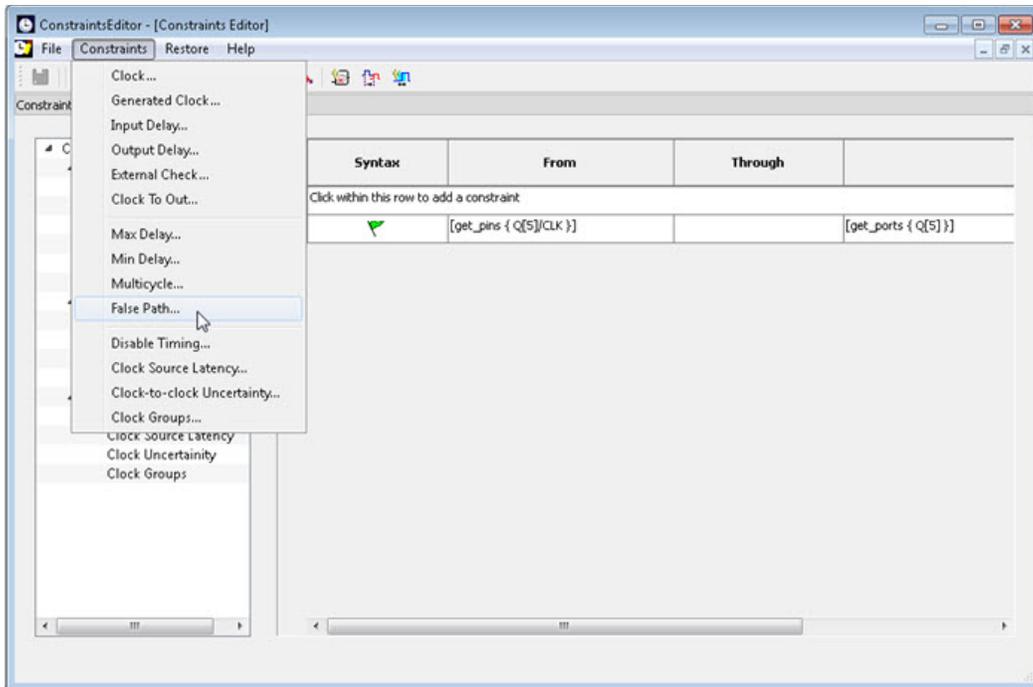
Adding Constraints

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:

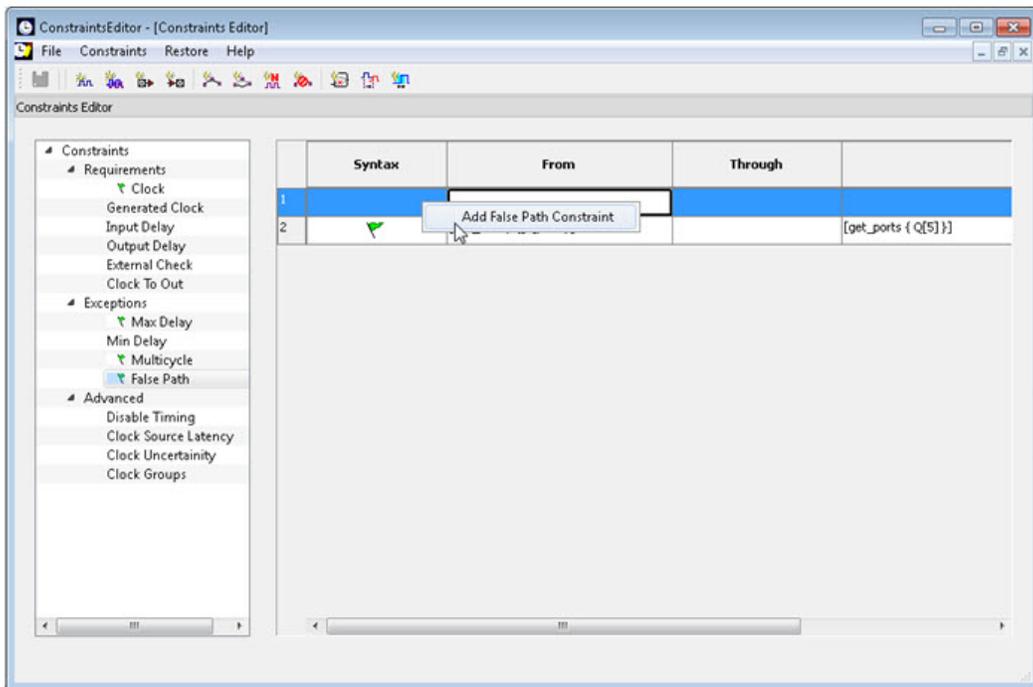
- Click the Add Constraint icon. Example: Click  to add False Path Constraints.
- From the Constraints Browser, choose the type of Constraints to add. Example: False Path



- 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.

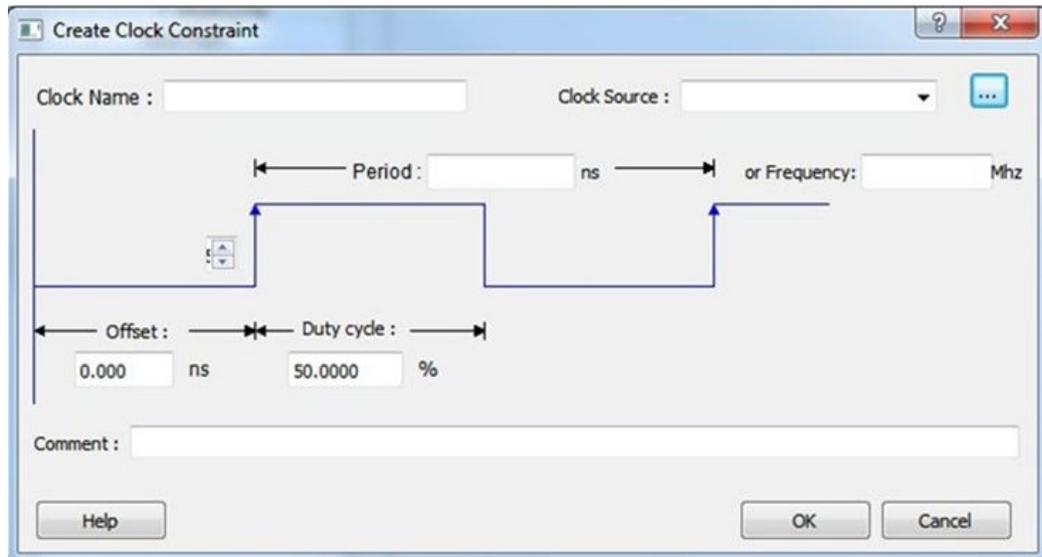


Figure 48 · Create Clock Constraint Dialog Box

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](#).

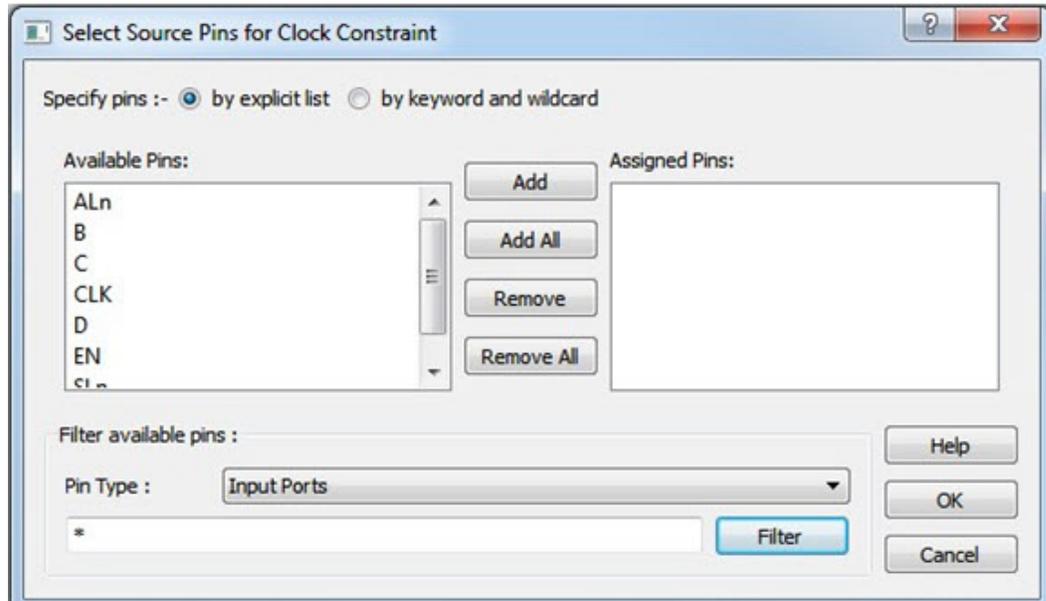


Figure 49 · Select Source Pin for Clock Constraint Dialog Box

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.

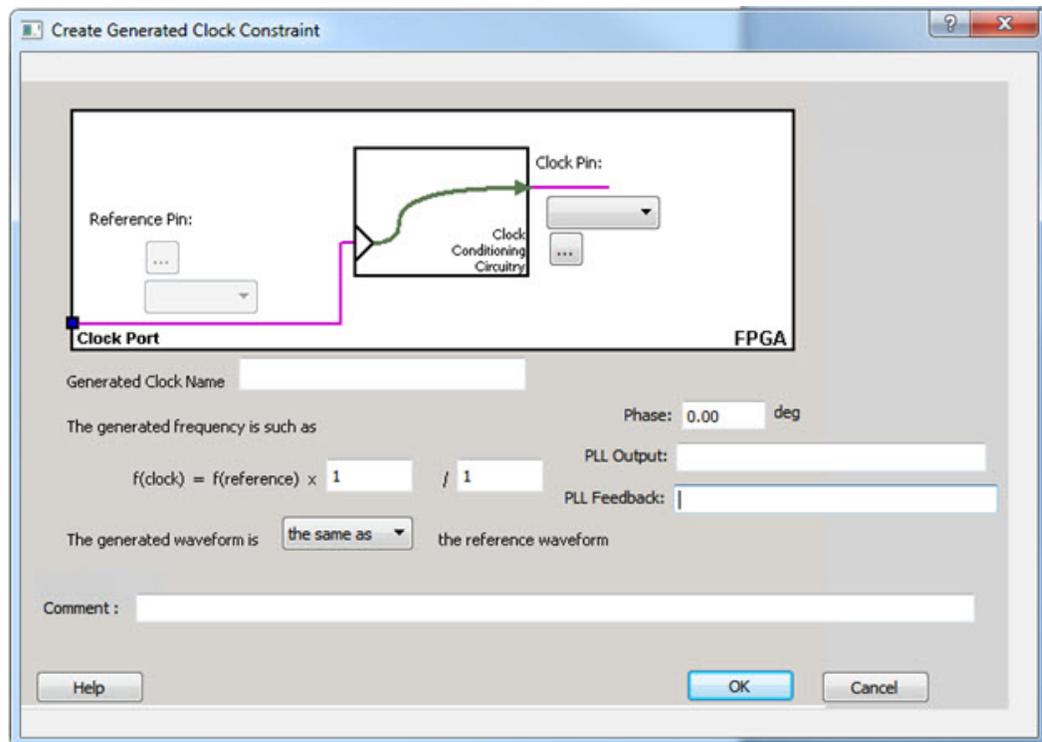


Figure 50 · Create Generated Clock Constraint Dialog Box

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.

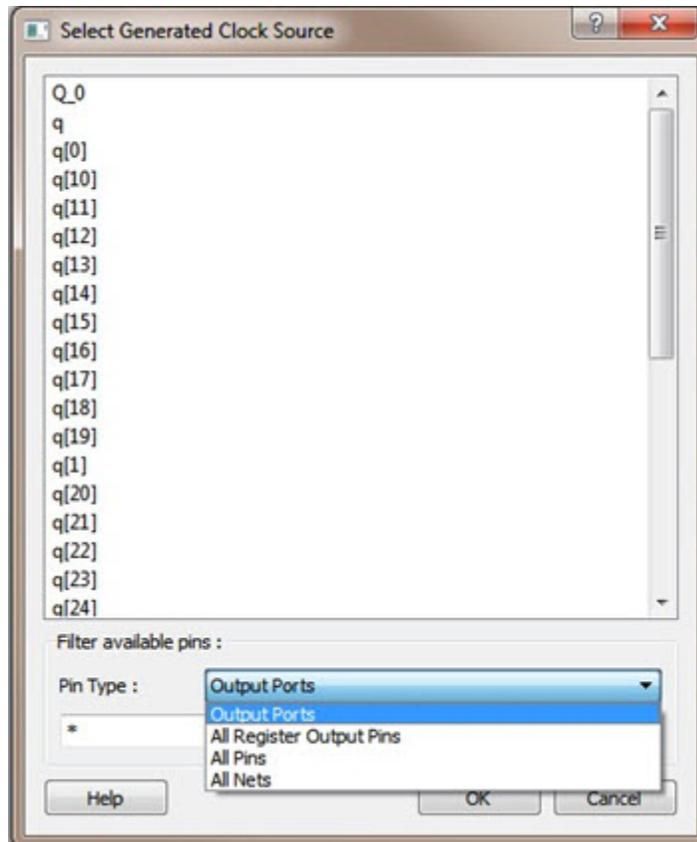


Figure 51 · Select Generated Clock Source Dialog Box

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.

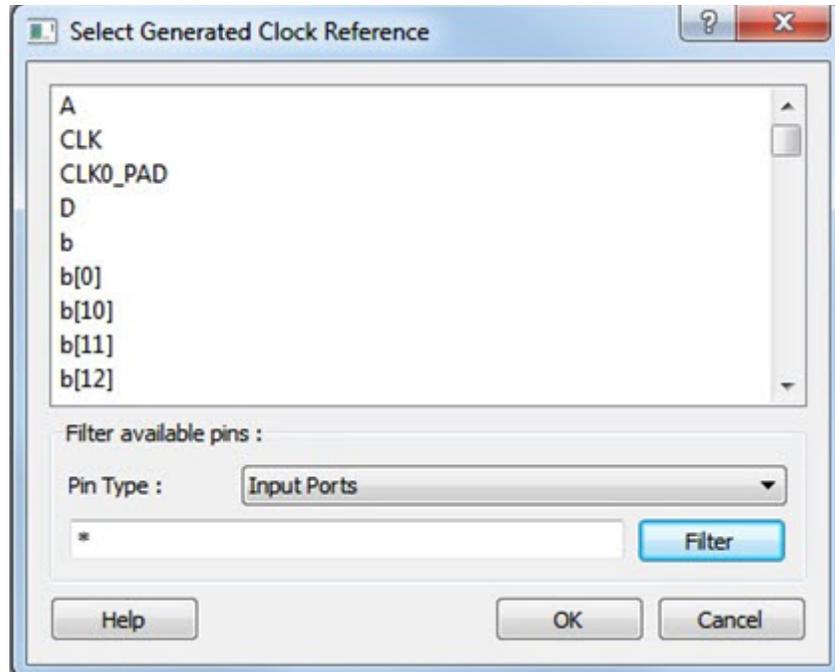


Figure 52 · Select Generated Clock Reference Dialog Box

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.

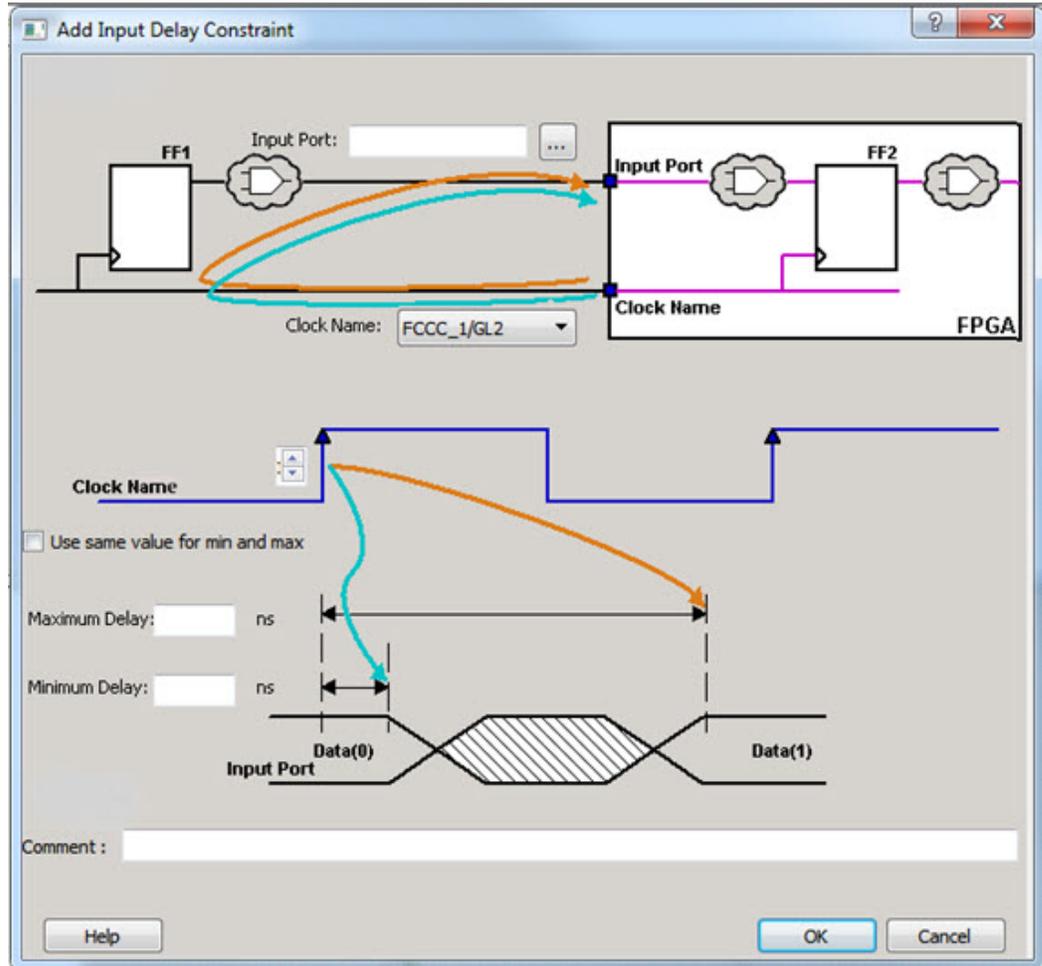


Figure 53 · Add Input Delay Constraint Dialog Box

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.

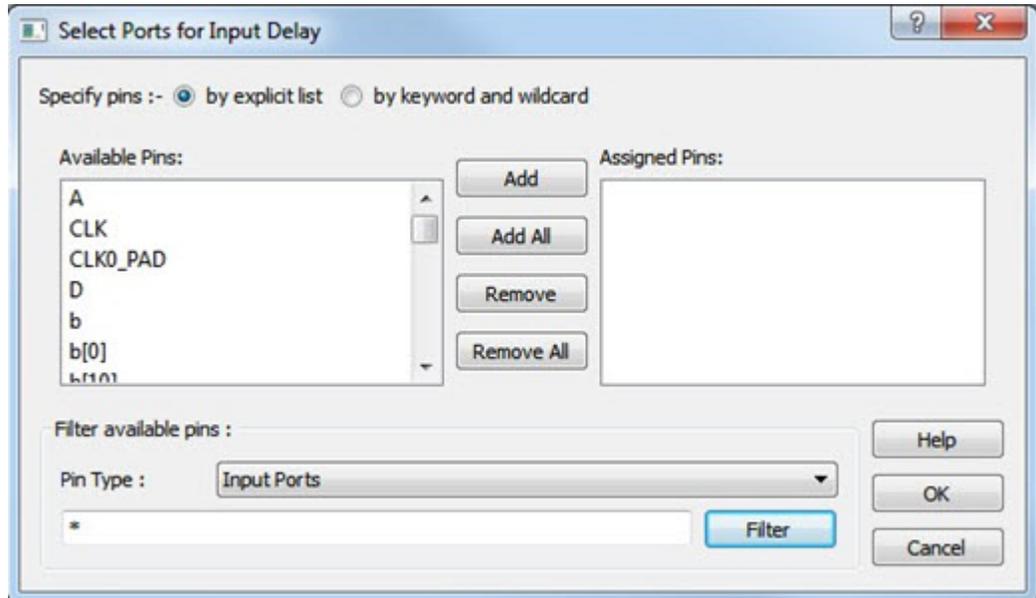


Figure 54 · Select Ports for Input Delay Dialog Box

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.

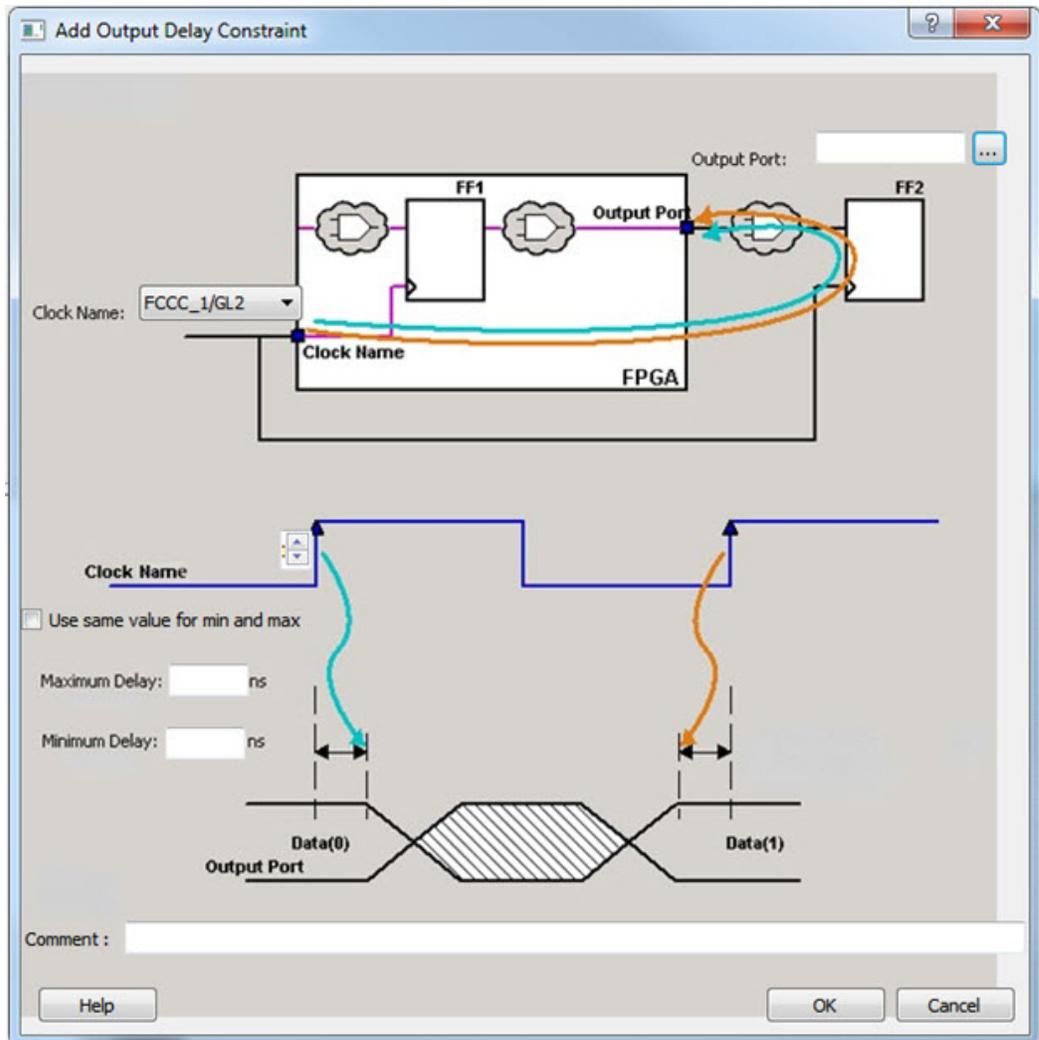


Figure 55 · Add Output Delay Constraint Dialog Box

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.

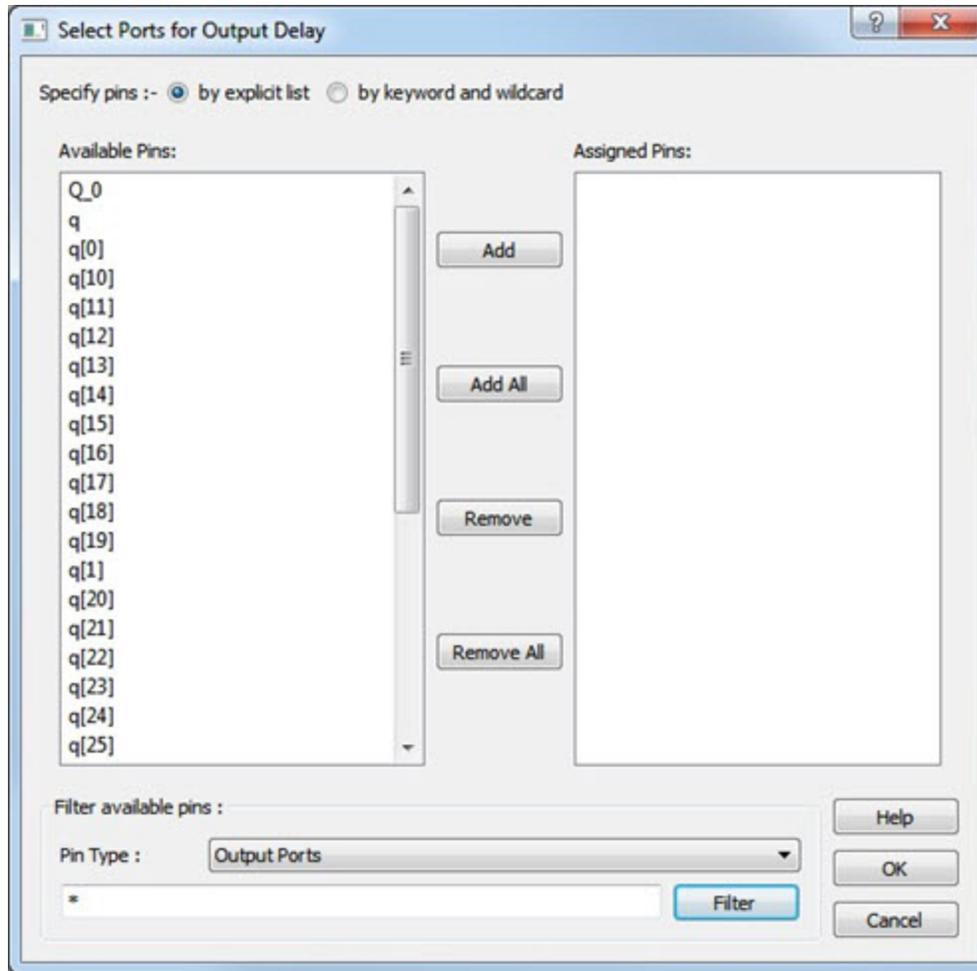


Figure 56 · Output Delay Dialog Box

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.

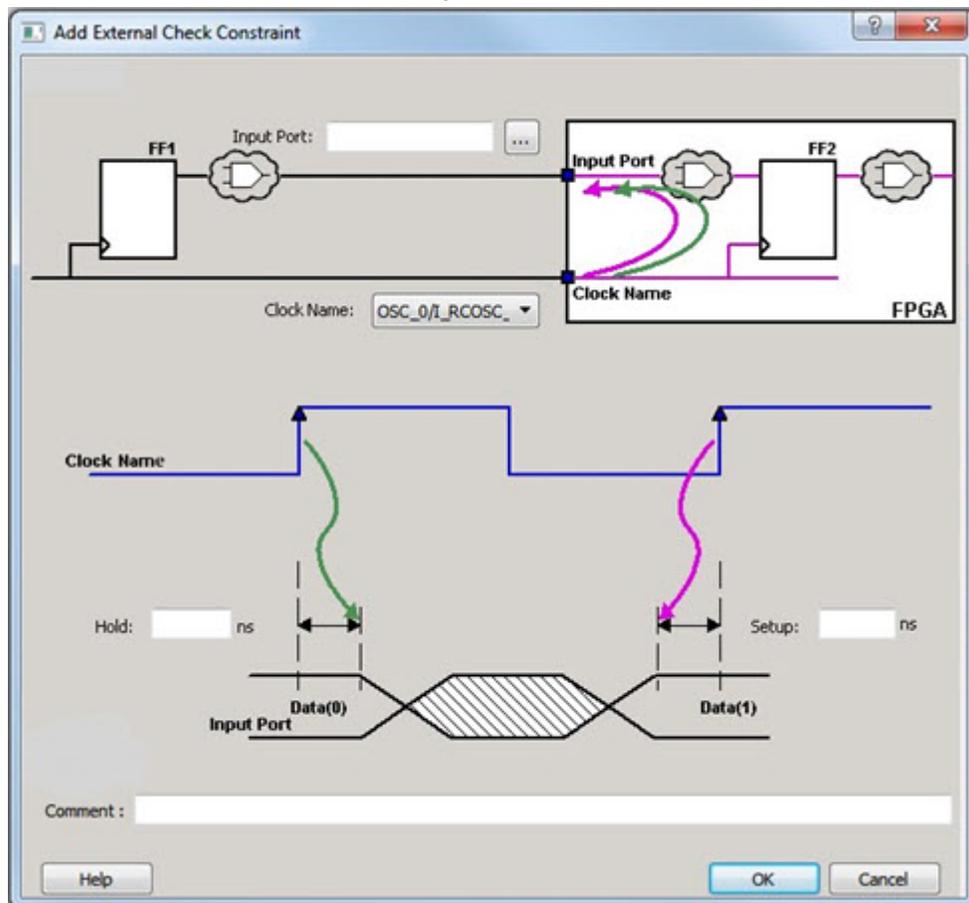


Figure 57 · Add External Check Constraint Dialog Box

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.

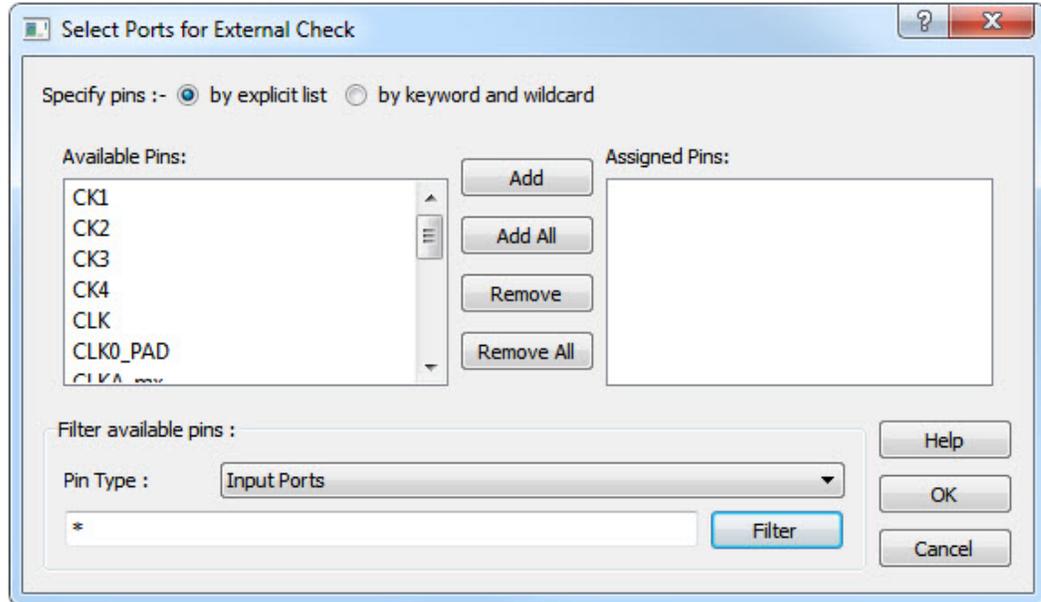


Figure 58 · Select Ports for External Check Dialog Box

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 box appears.

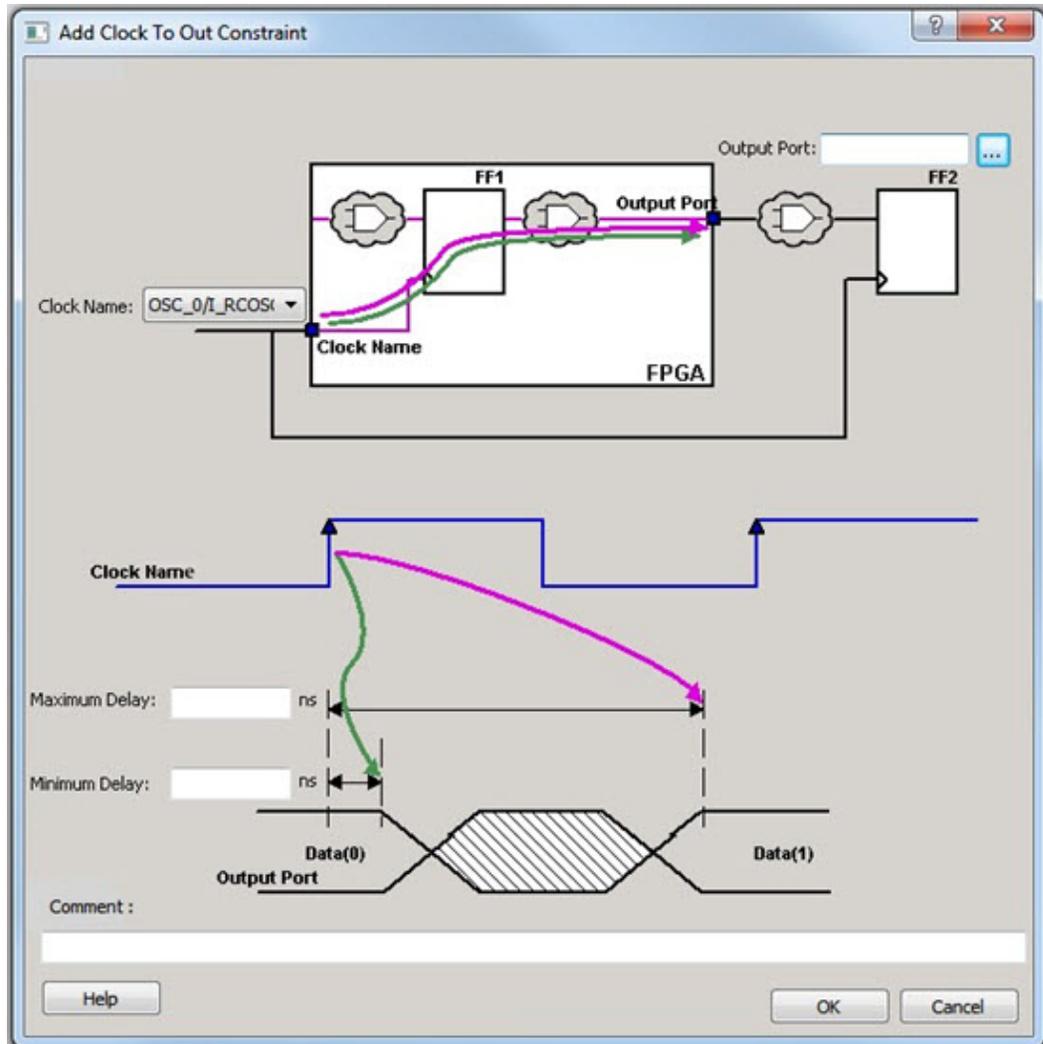


Figure 59 · Add Clock to Out Constraint Dialog Box

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 Out constraint.

Click the browse button next to Output Port to open the Select Ports for Clock To Output dialog box.

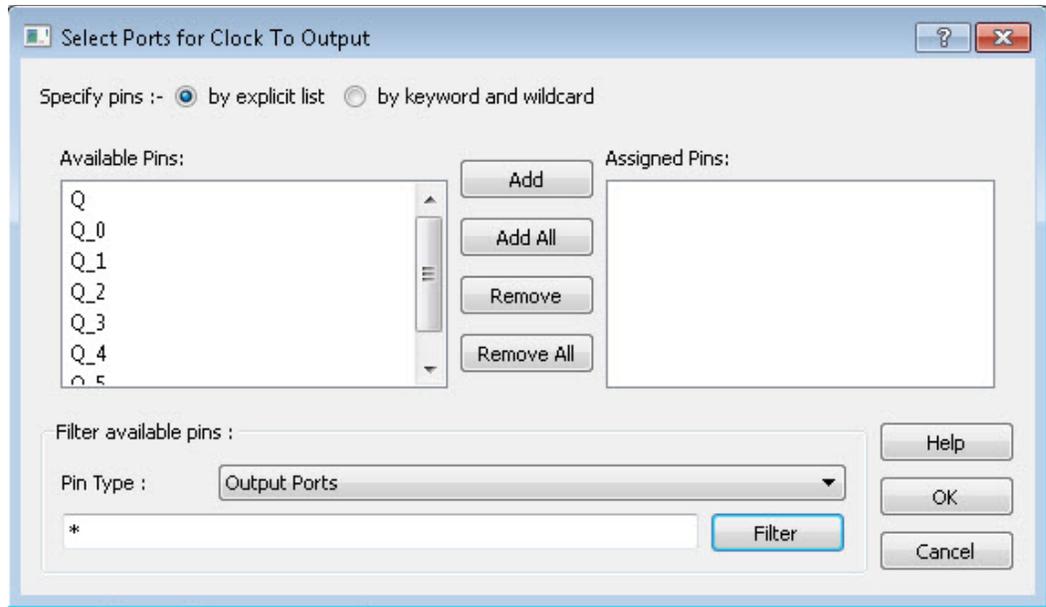


Figure 60 · Select Ports for Clock To Output Dialog Box

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

Timing Exception Constraints	Order of Precedence
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 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.

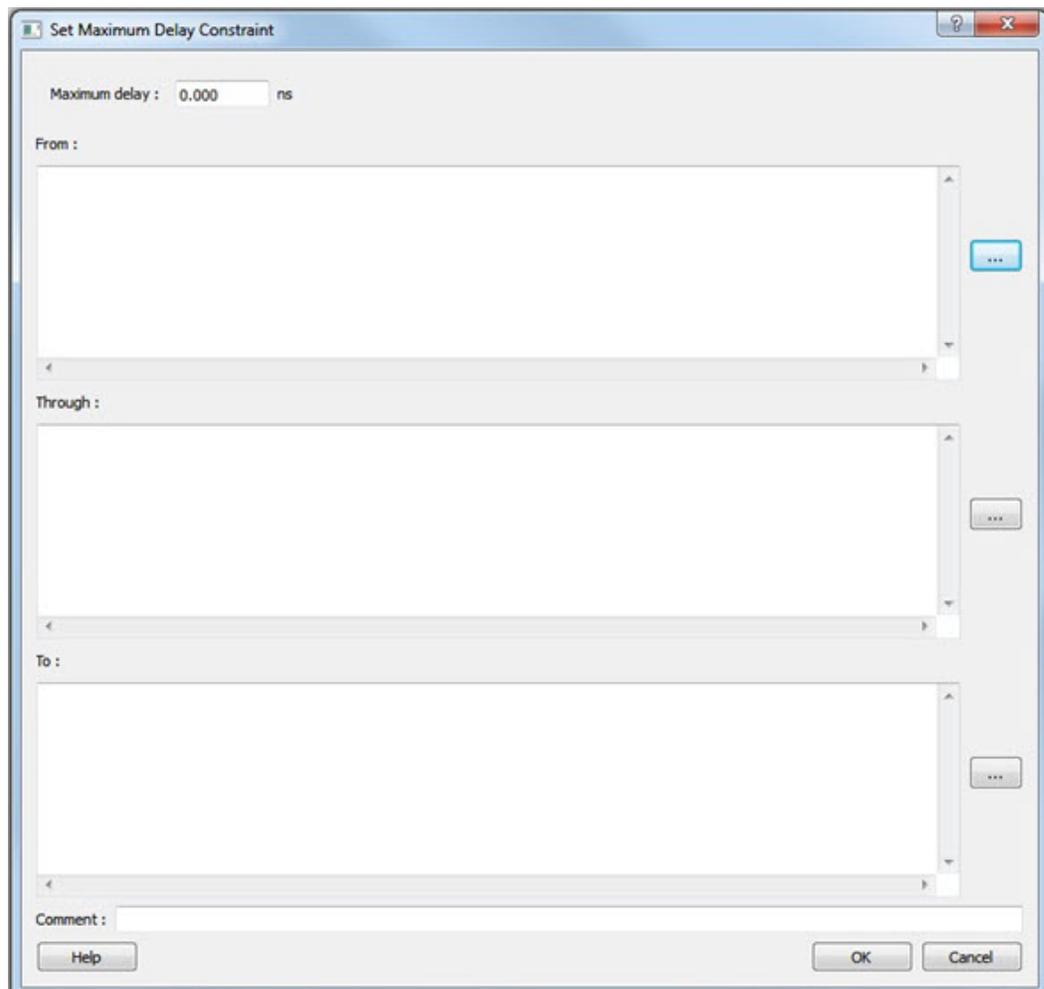


Figure 61 · Set Maximum Delay Constraint Dialog Box

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.

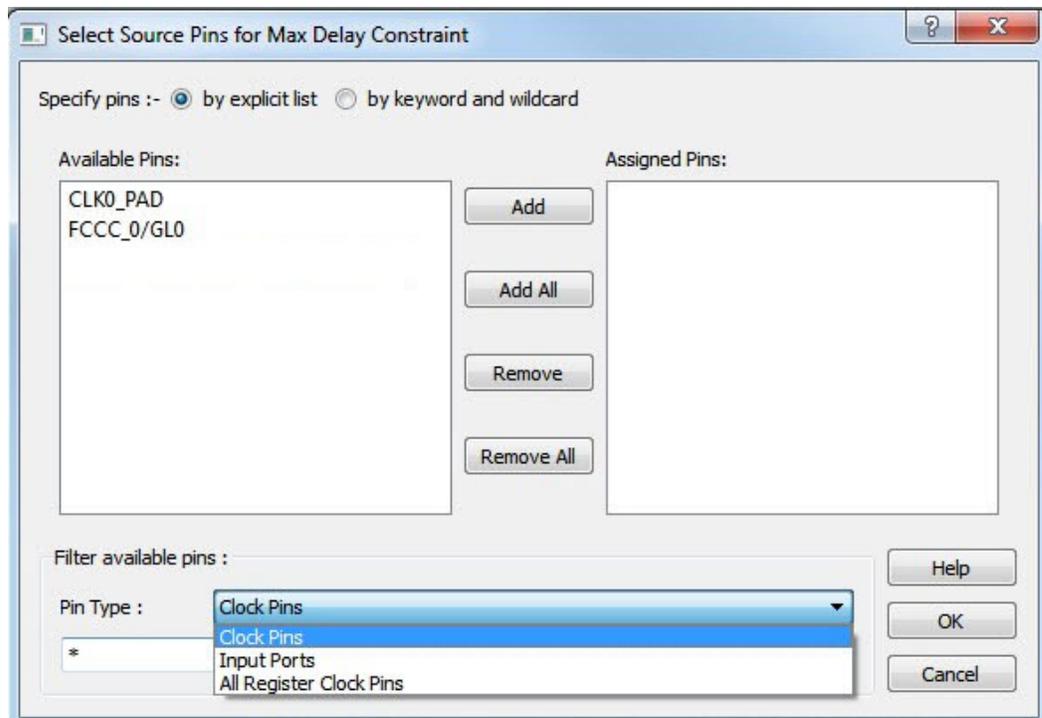


Figure 62 · Select Source Pins for Max Delay Constraint Dialog Box

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.

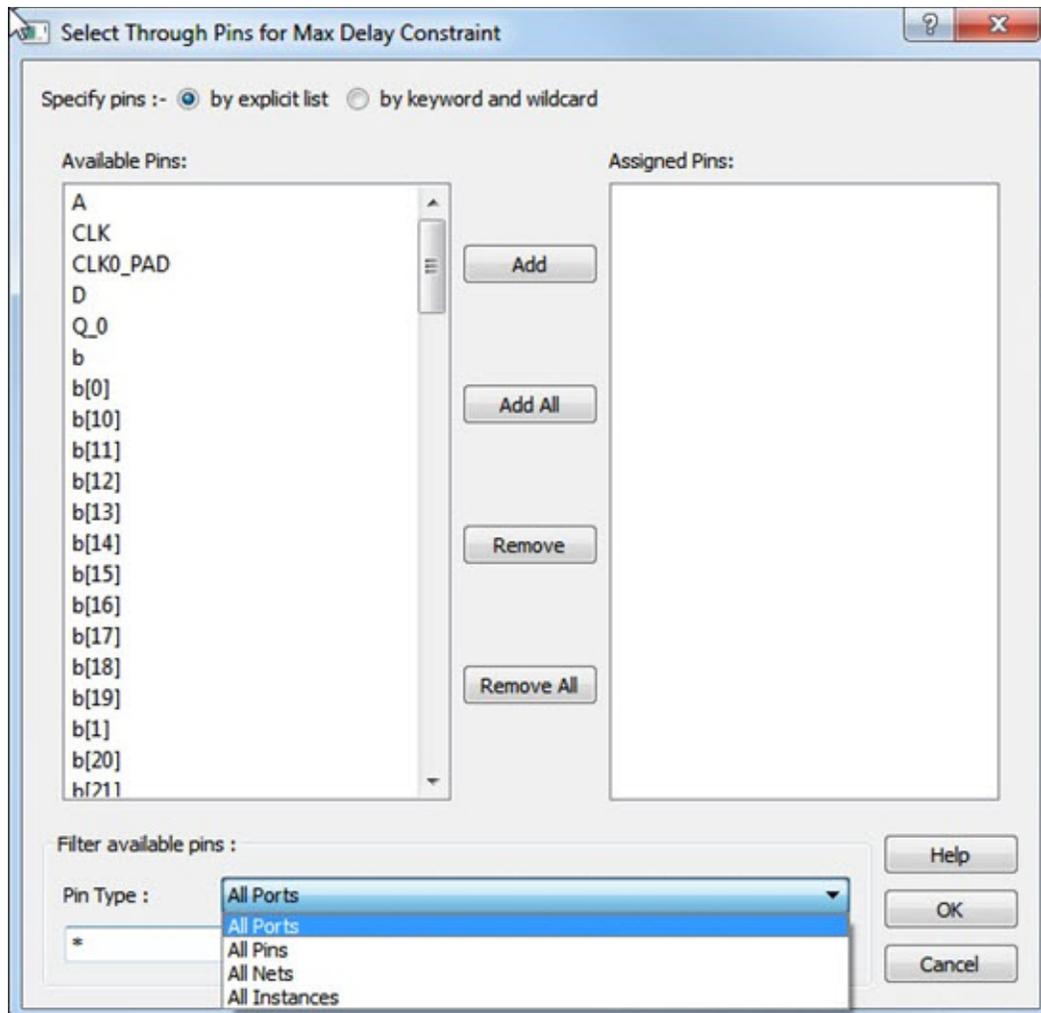


Figure 63 · Select Through Pins for Max Delay Constraint Dialog Box

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.

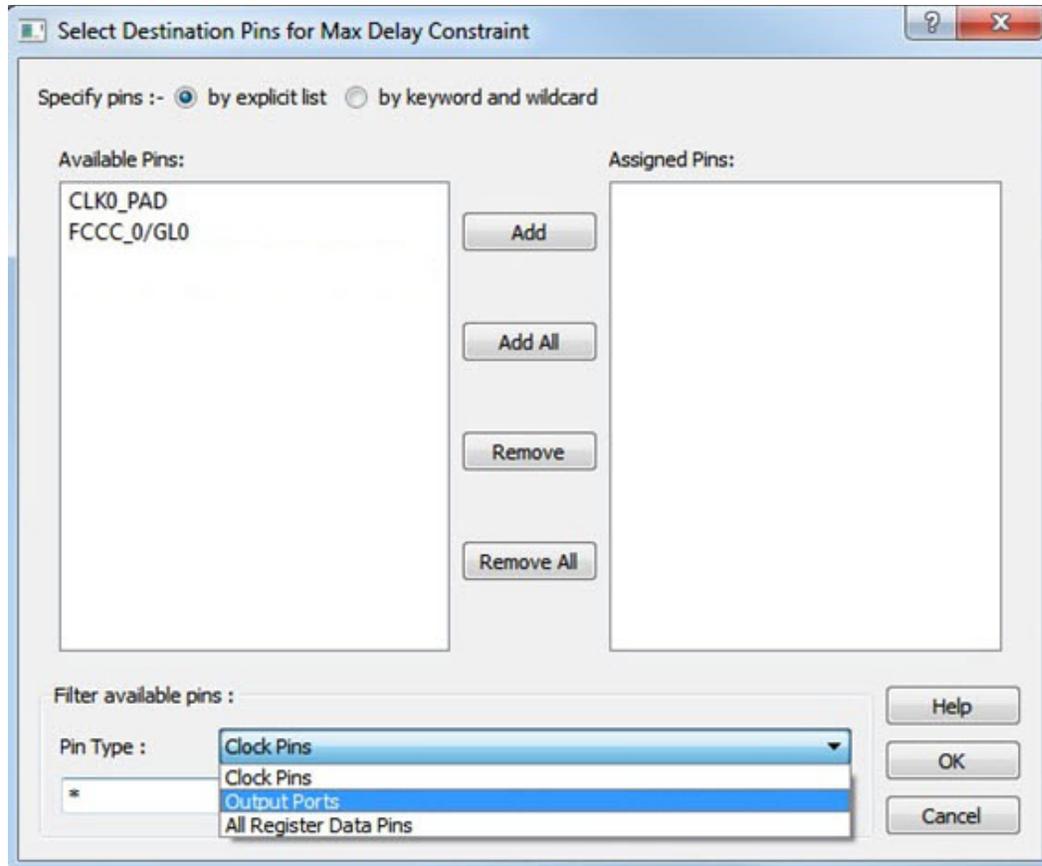


Figure 64 · Select Destination Pins for Max Delay Constraint Dialog Box

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
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.

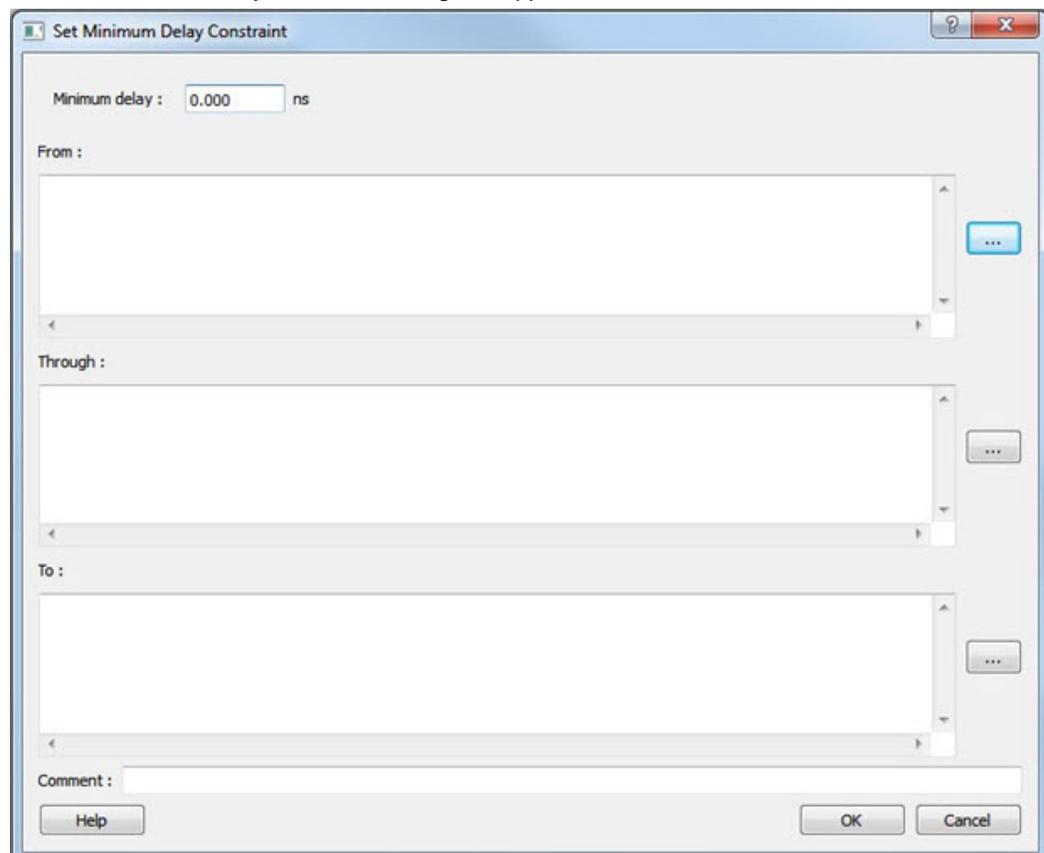


Figure 65 · Set Minimum Delay Constraint Dialog Box

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.

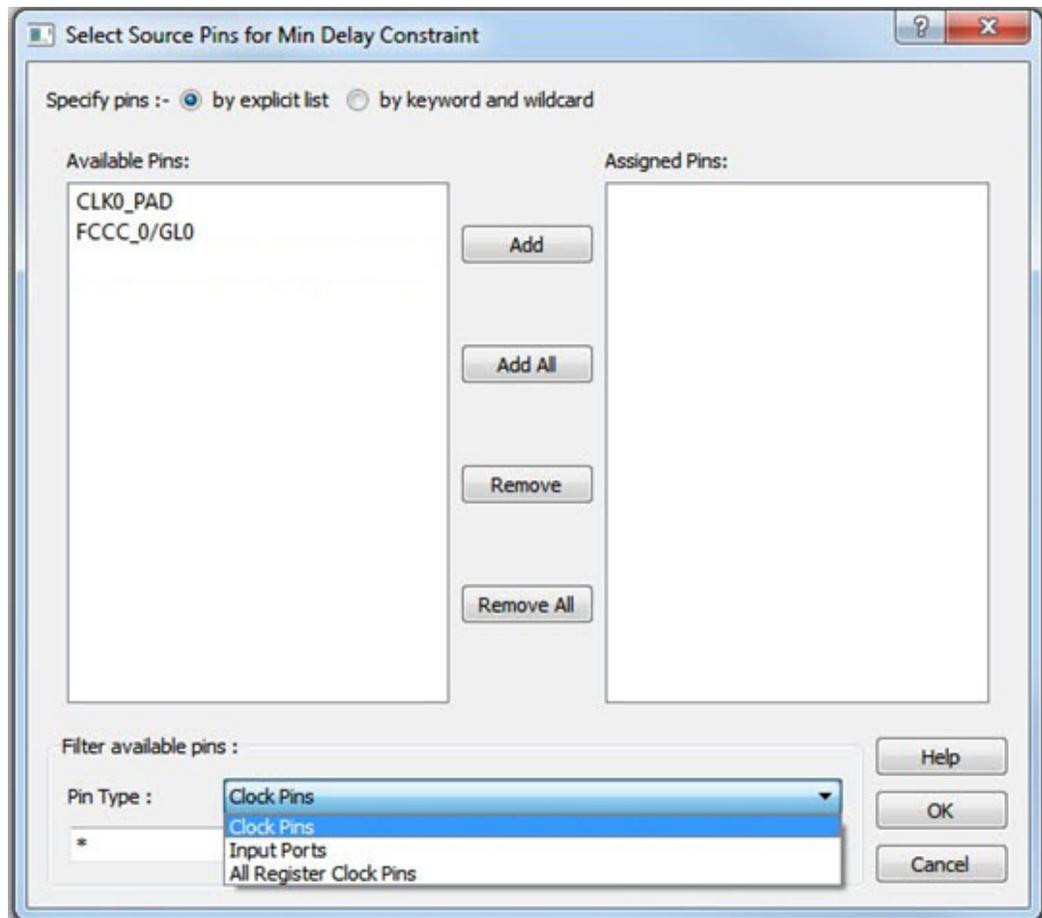


Figure 66 · Select Source Pins for Min Delay Constraint Dialog Box

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.

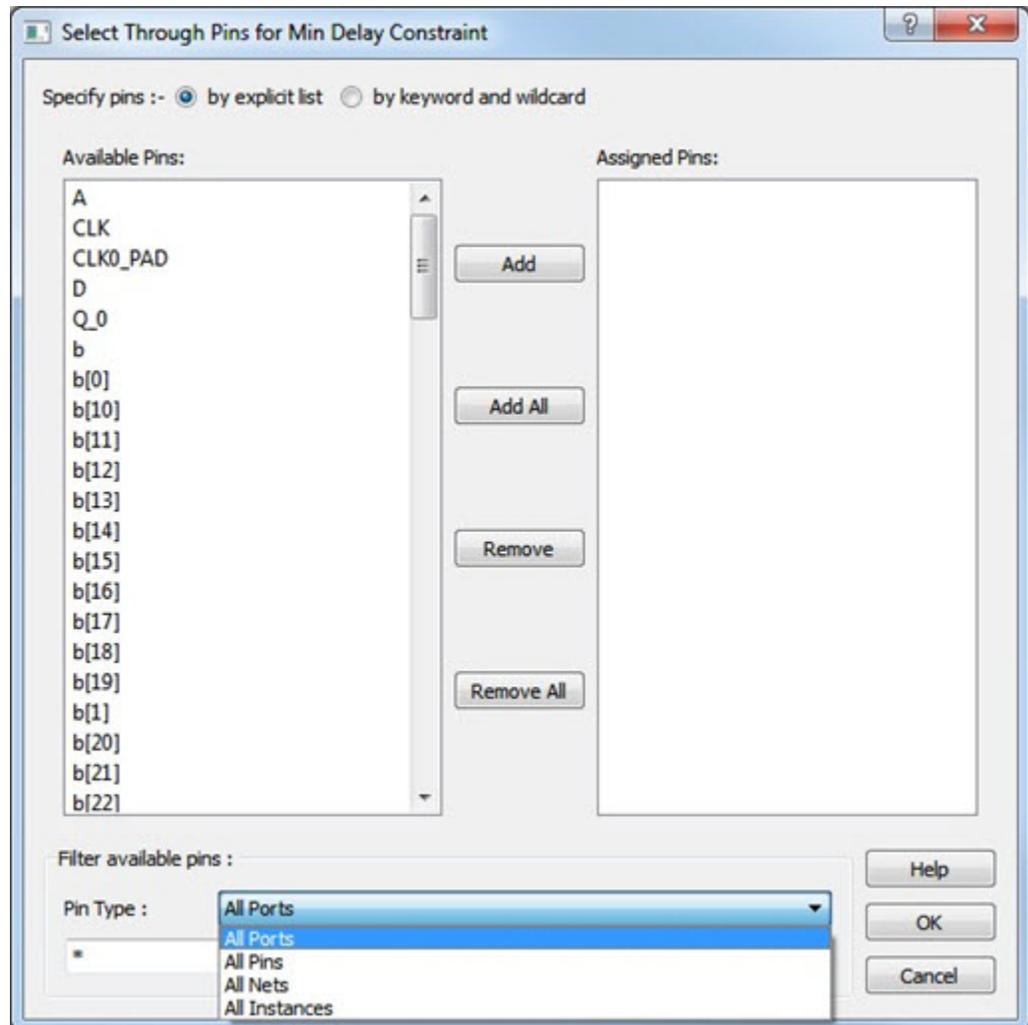


Figure 67 · Select the Through Pins for Min Delay Dialog Box

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.

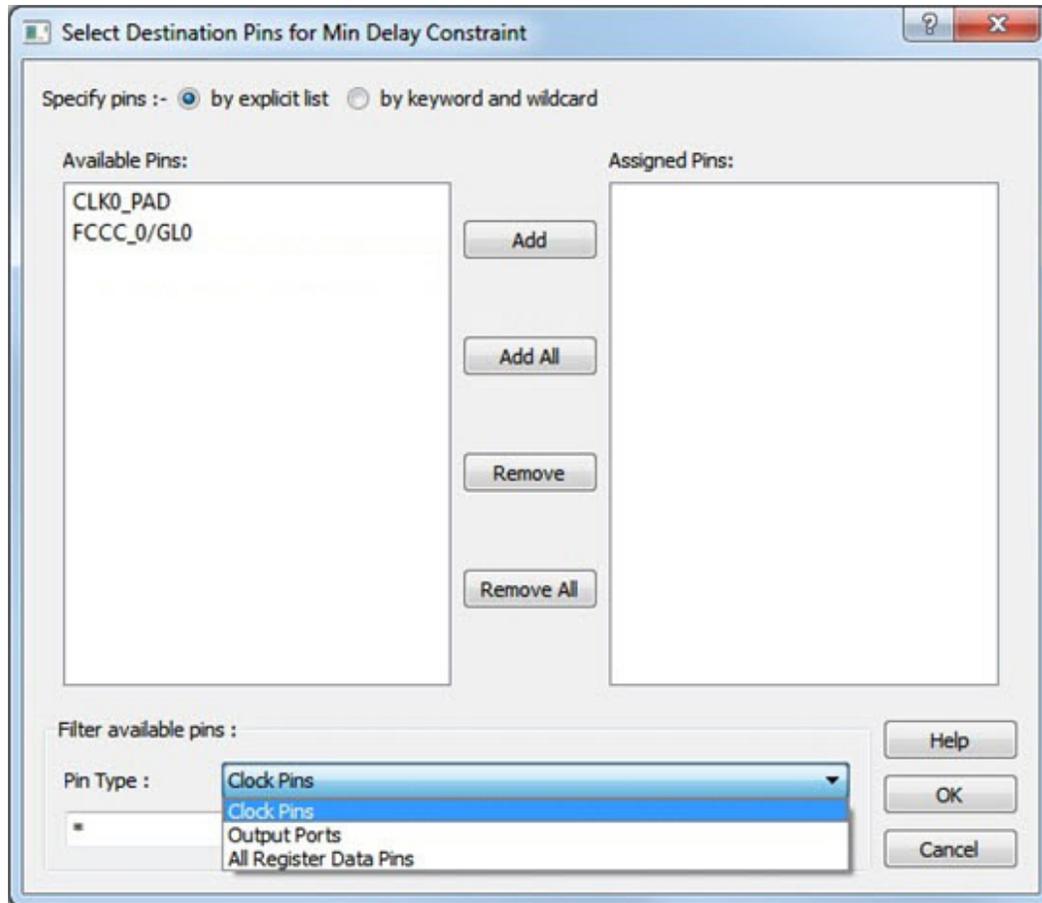


Figure 68 · Select the Destination Pins for Min Delay Constraint Dialog Box

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.

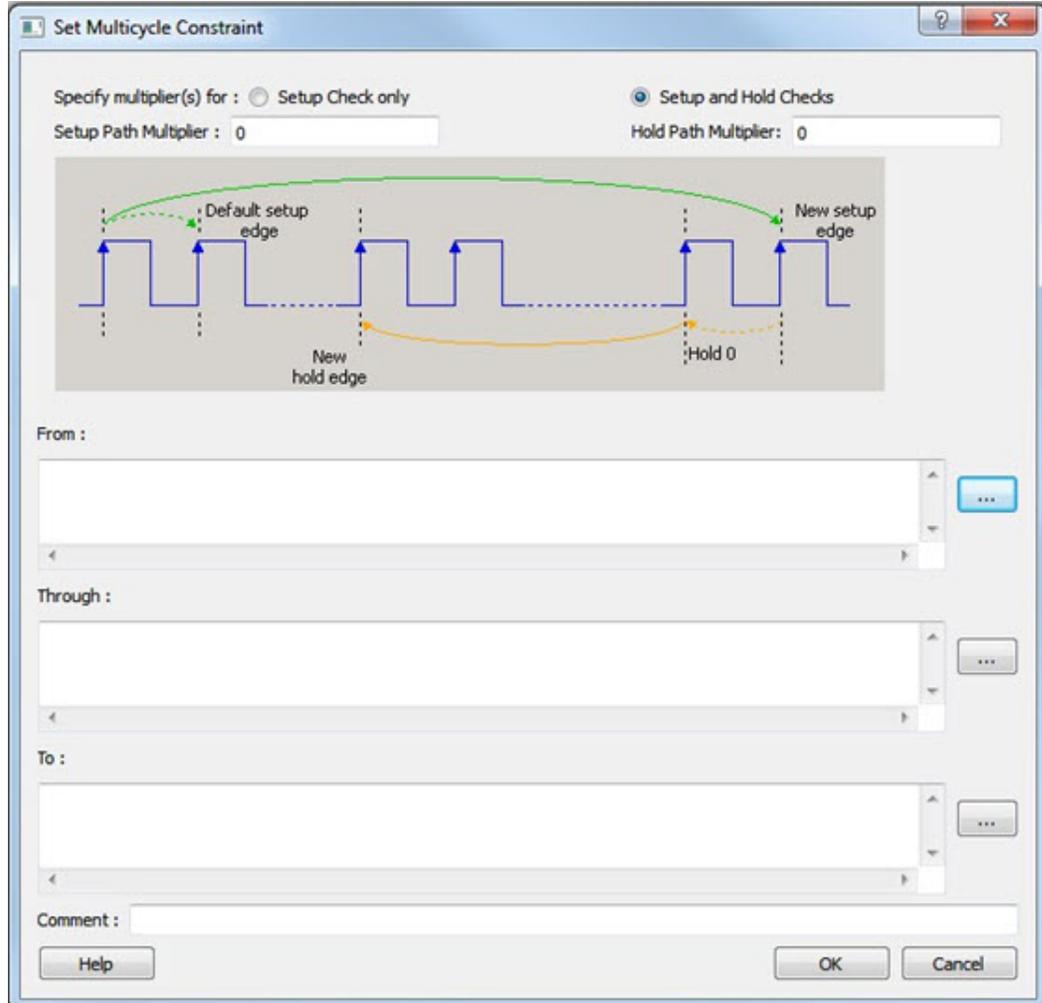


Figure 69 · Set Multicycle Constraint Dialog Box

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.

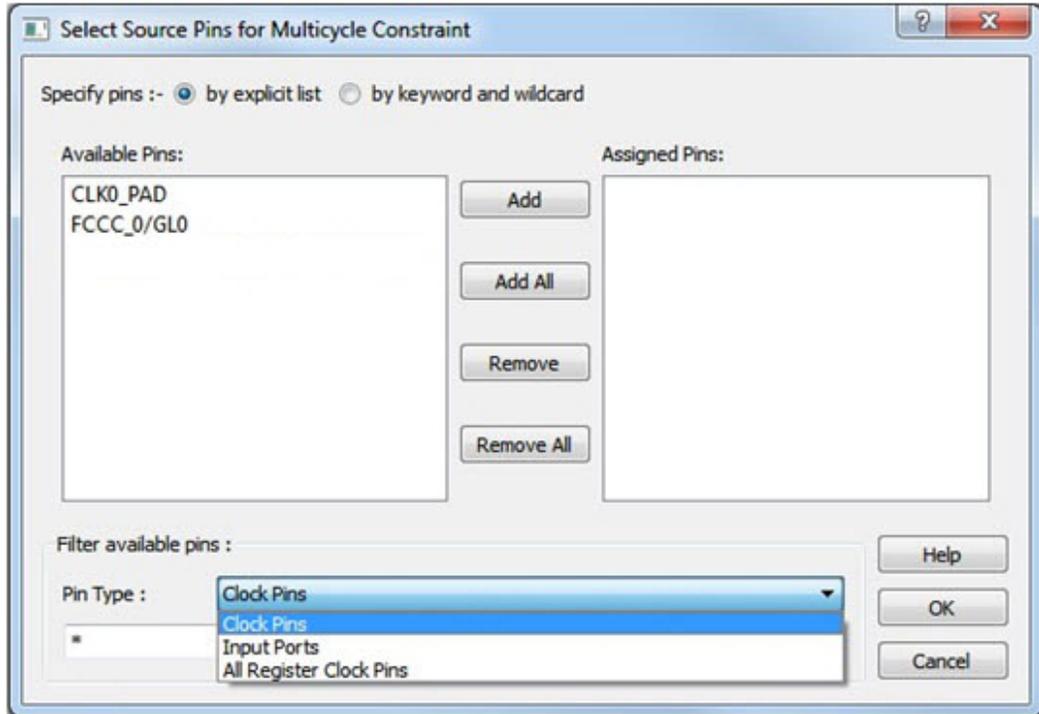


Figure 70 · Select Source Pins for Multicycle Constraint Dialog Box

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.

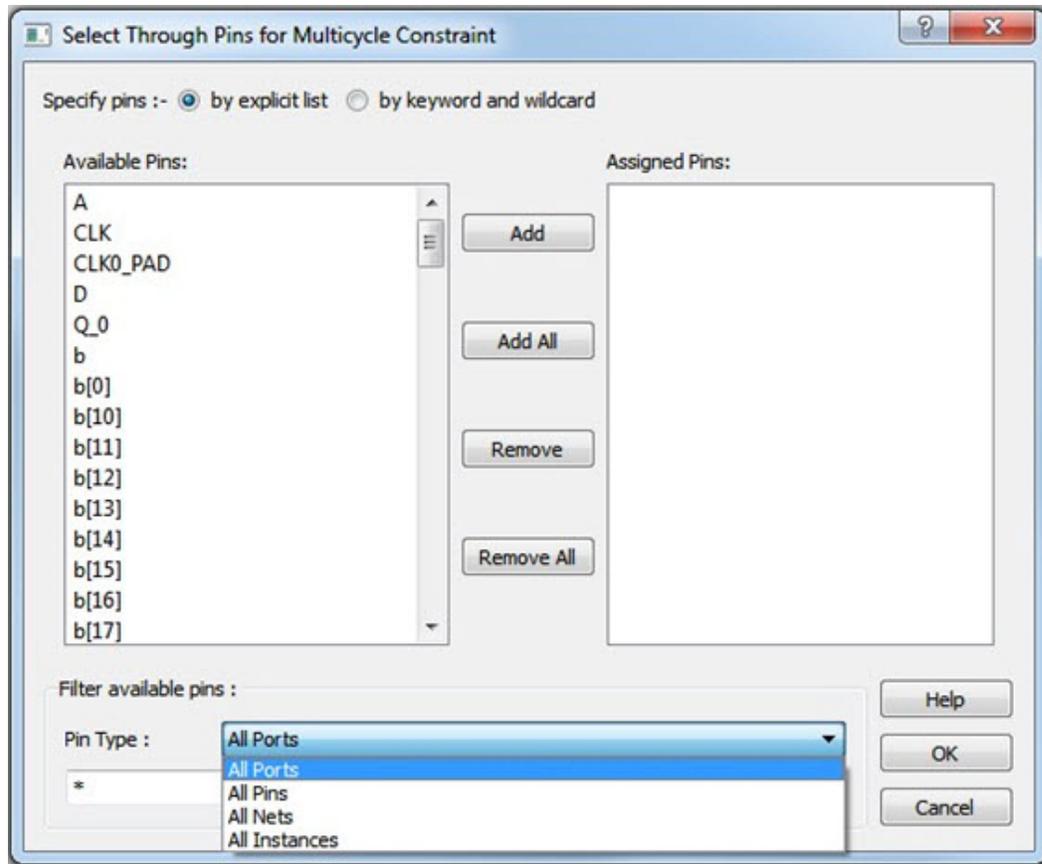


Figure 71 · Select Through Pins for Multicycle Constraint Dialog Box

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 72 · 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
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.

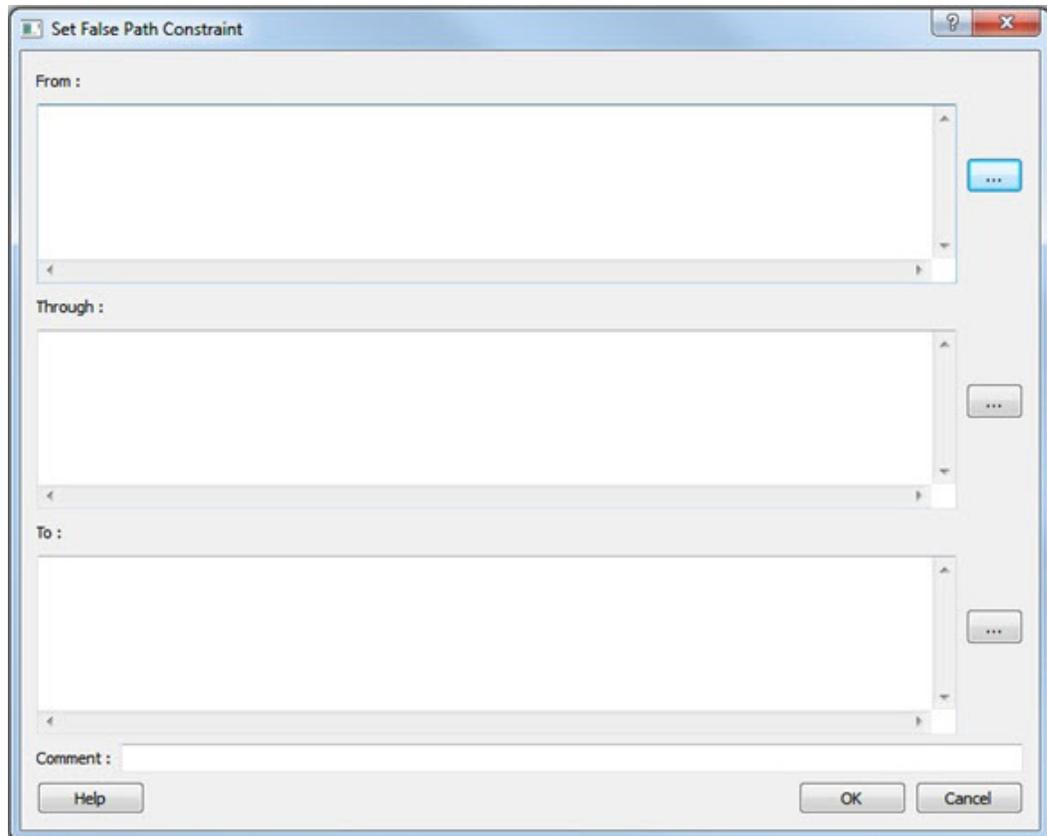


Figure 73 · Set False Path Constraint Dialog Box

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.

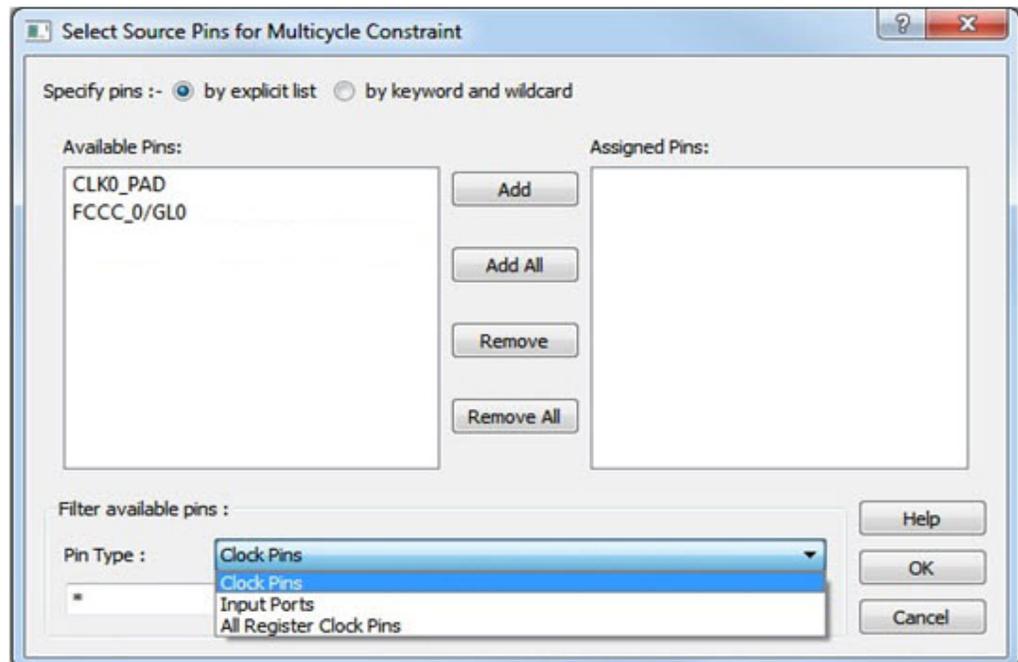


Figure 74 · Select Source Pins for False Path Constraint Dialog Box

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.

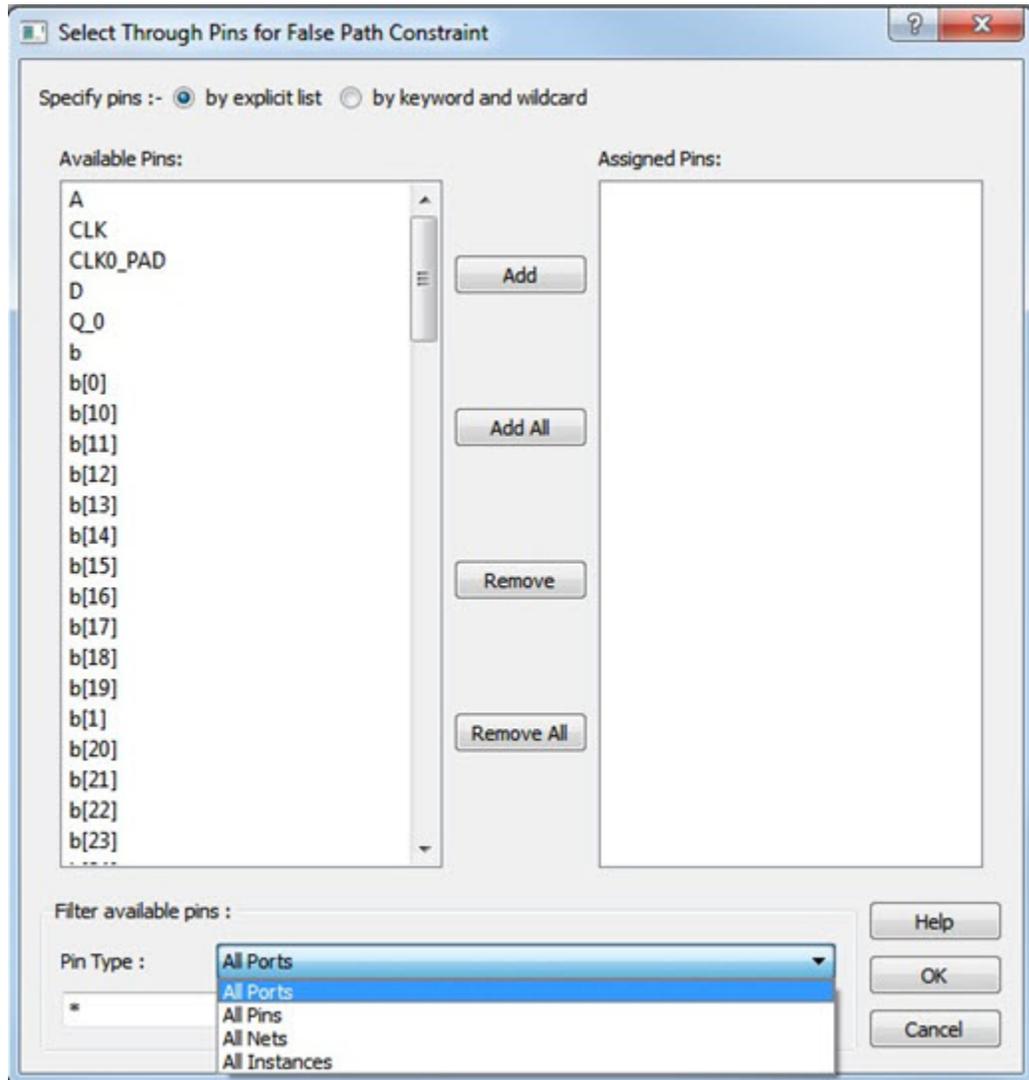


Figure 75 · Select Through Pins for False Path Constraint Dialog Box

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.

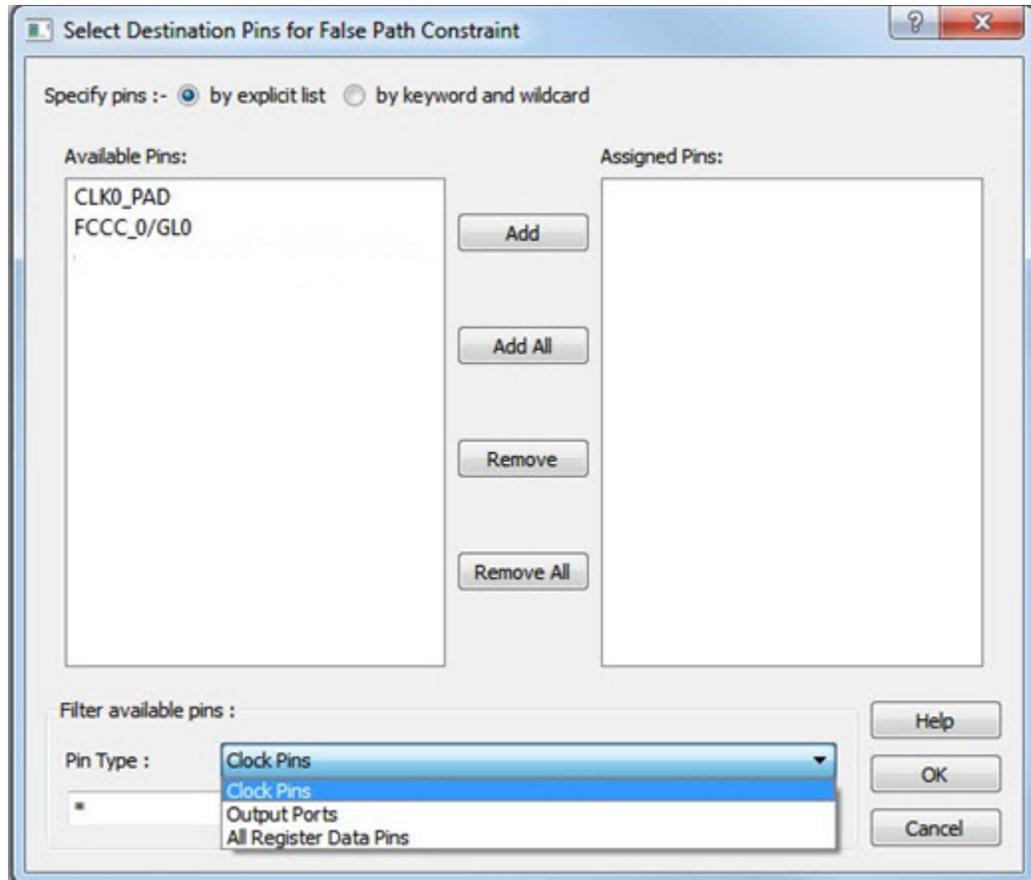


Figure 76 · Select Destination Pins for False Path Constraint Dialog Box

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.

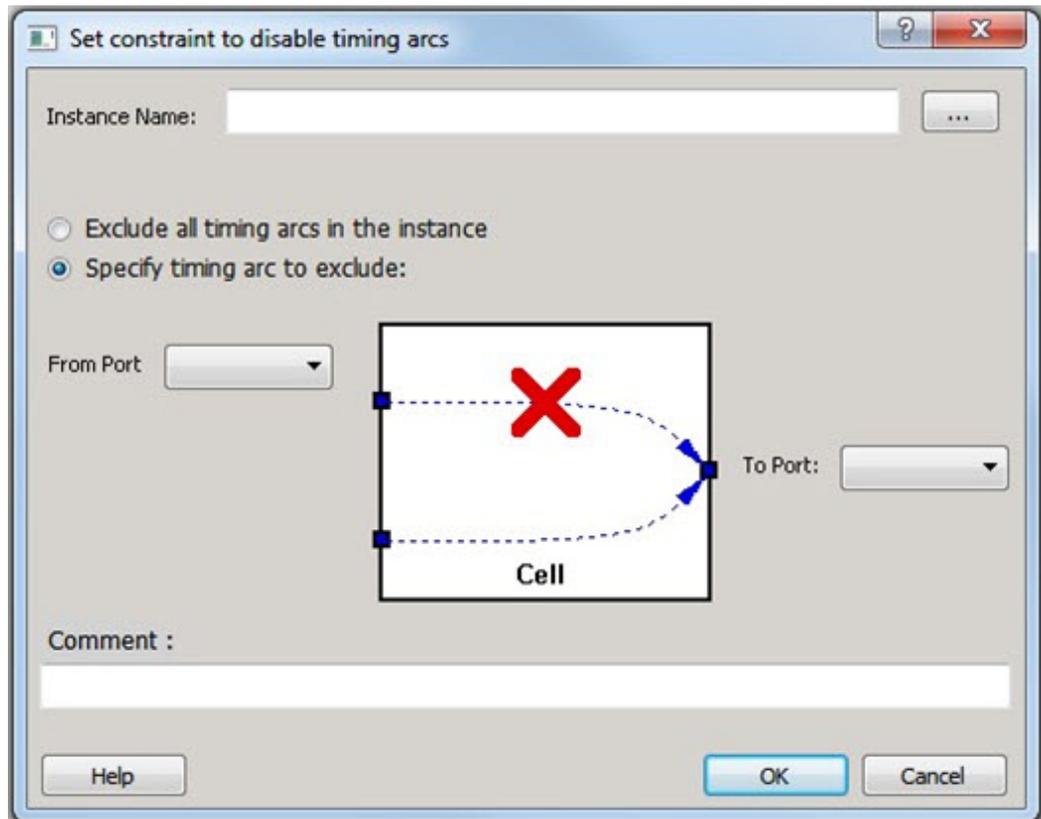


Figure 77 · Set Constraint to Disable Timing Arcs Dialog Box

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.

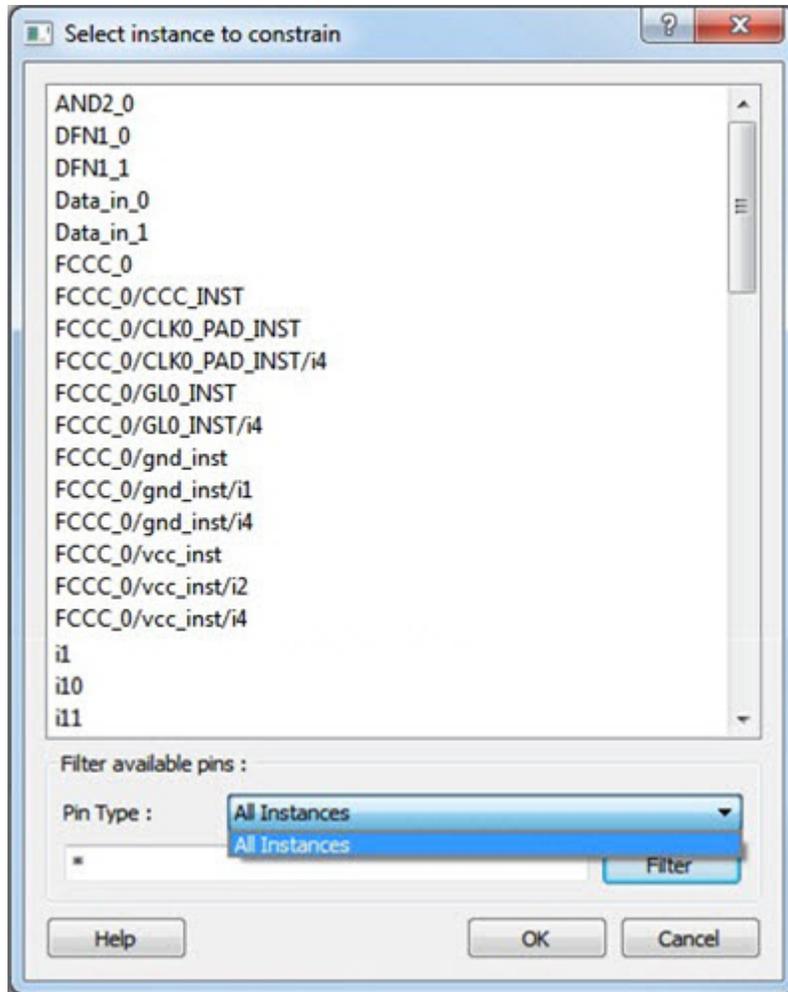


Figure 78 · Set Instance to Constrain Dialog Box

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.

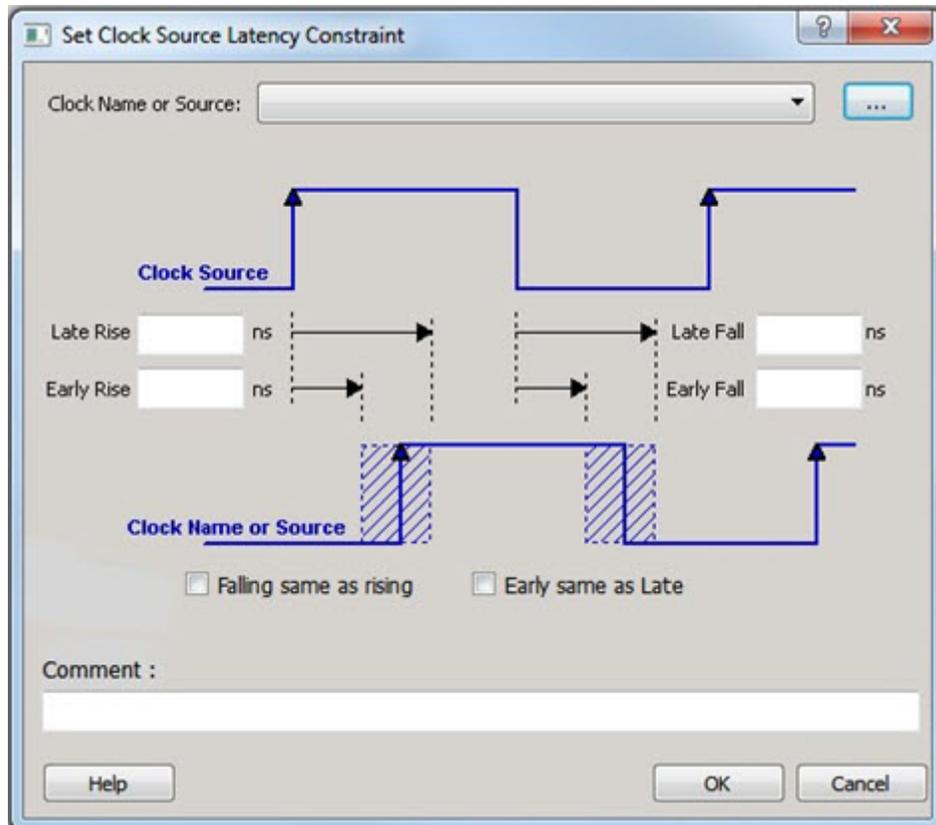


Figure 79 · Set Clock Source Latency Constraint Dialog Box

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

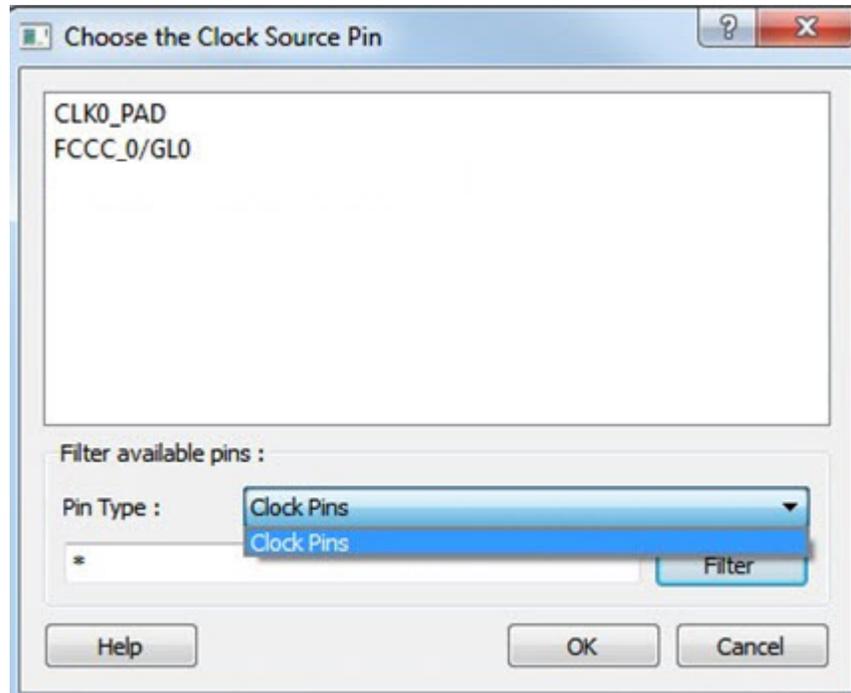


Figure 80 · Choose the Clock Source Pin Dialog Box

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.

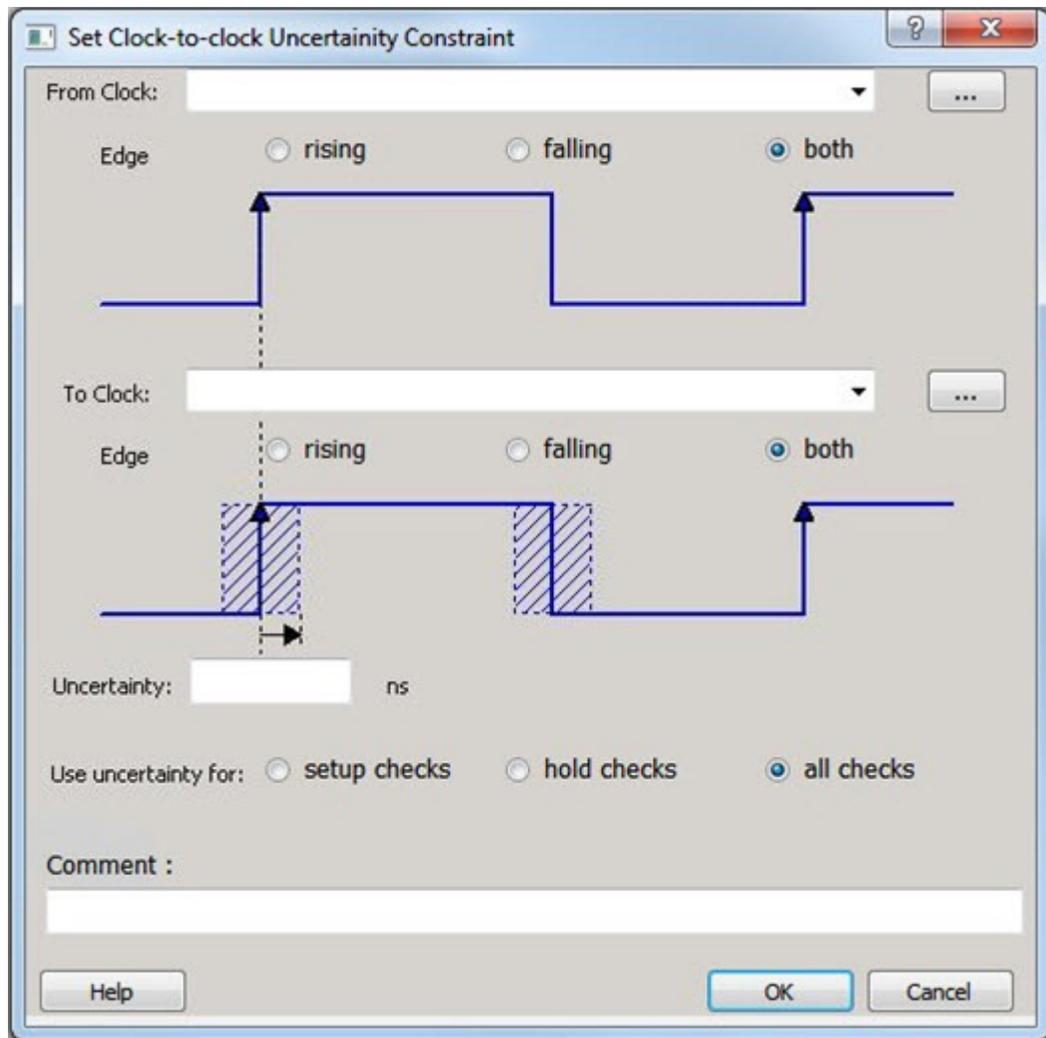


Figure 81 · Set Clock-to-Clock Uncertainty Dialog Box

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.

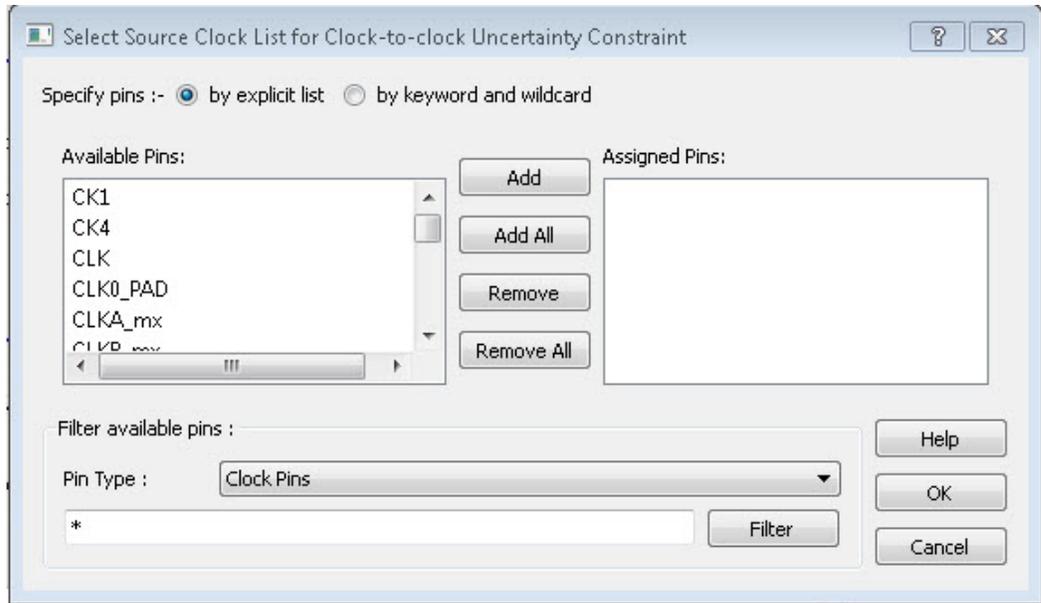


Figure 82 · Select Source Clock List for Clock-to-Clock Uncertainty Dialog Box

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.

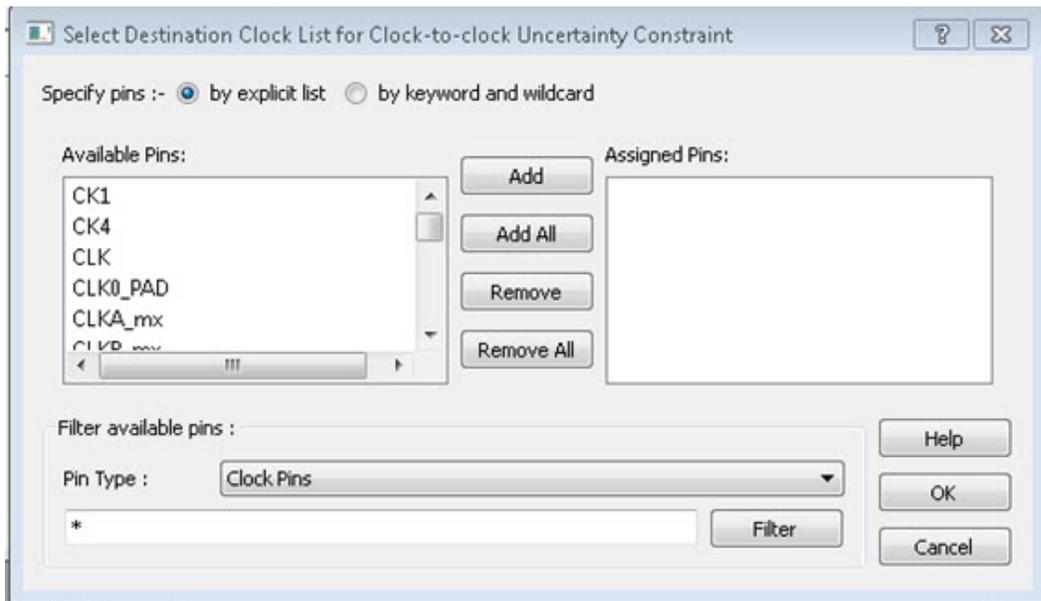


Figure 83 · Select Destination Clock List for Clock-to-Clock Uncertainty Constraint Dialog Box

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.

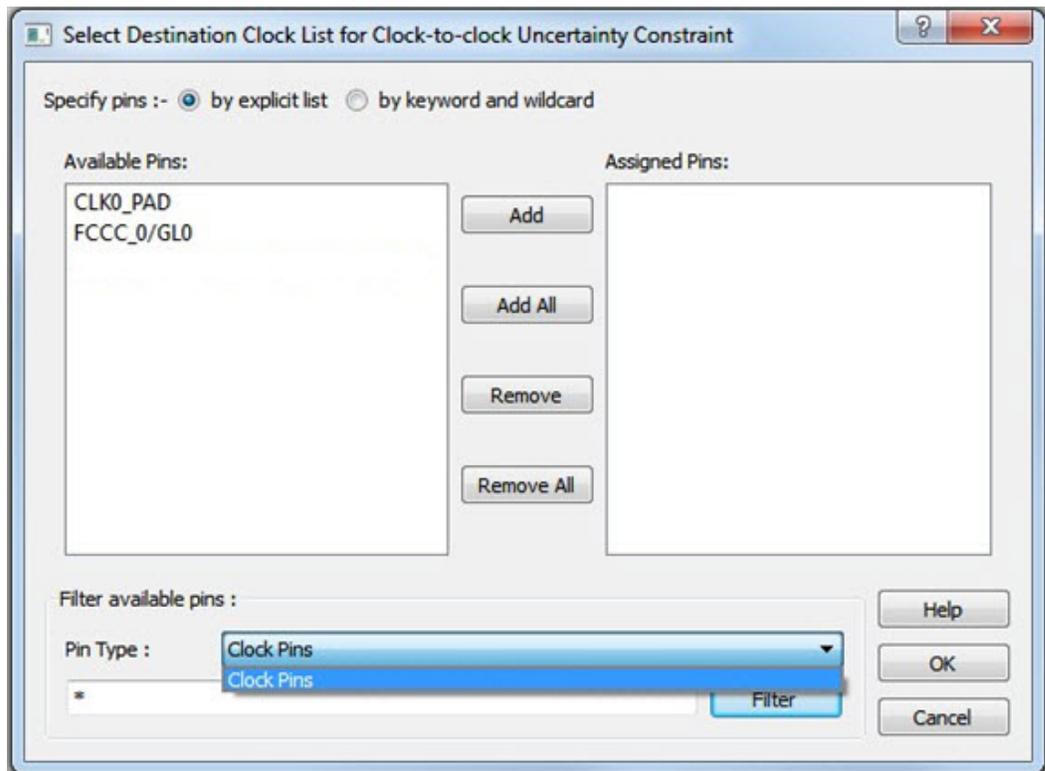


Figure 84 · Select Destination Clock List for Clock-to-Clock Uncertainty Dialog Box

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**.

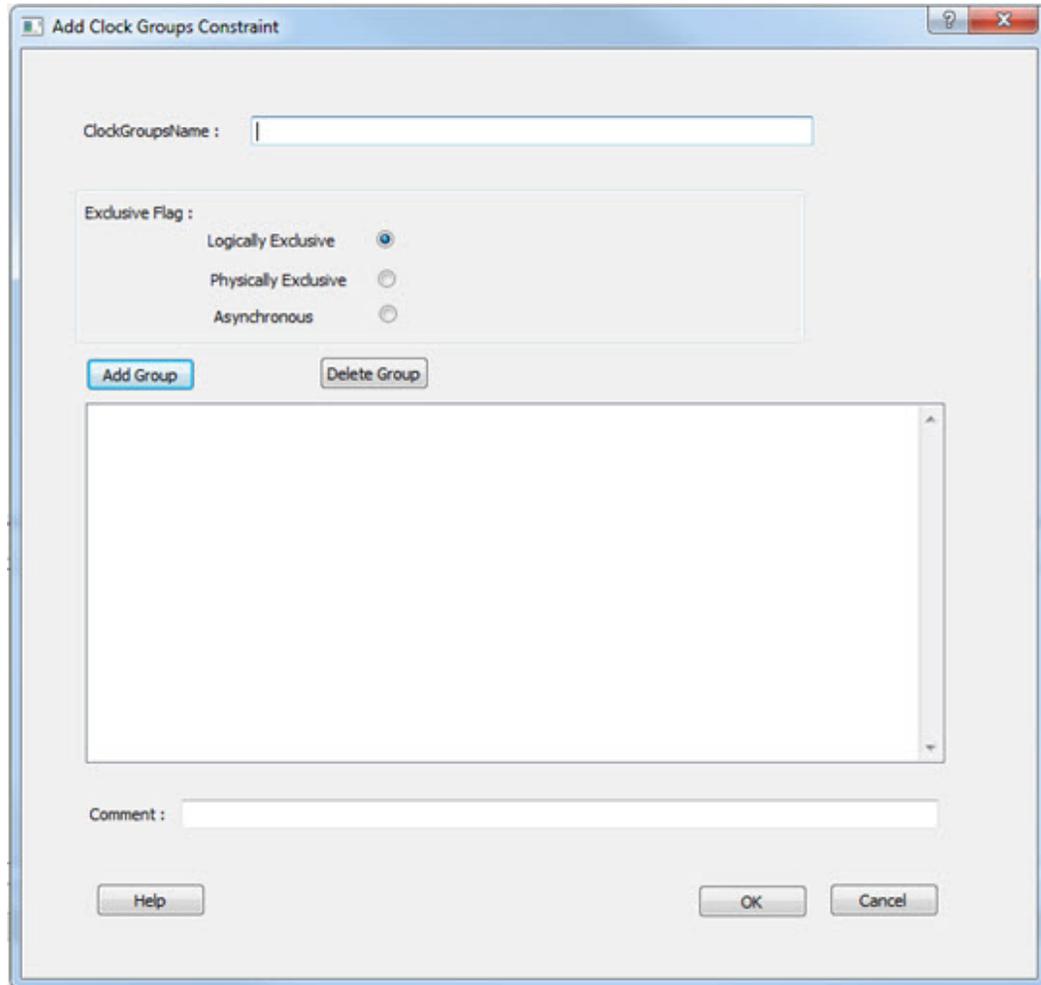


Figure 85 · Add Clock Group Constraints Dialog Box

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**.

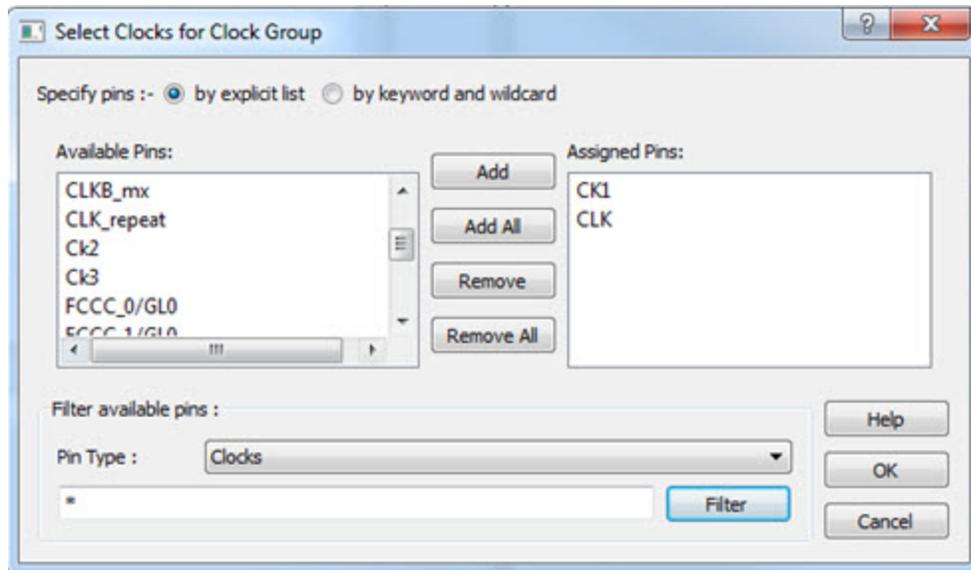


Figure 86 · Add Clocks For Clock Group Dialog Box

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.

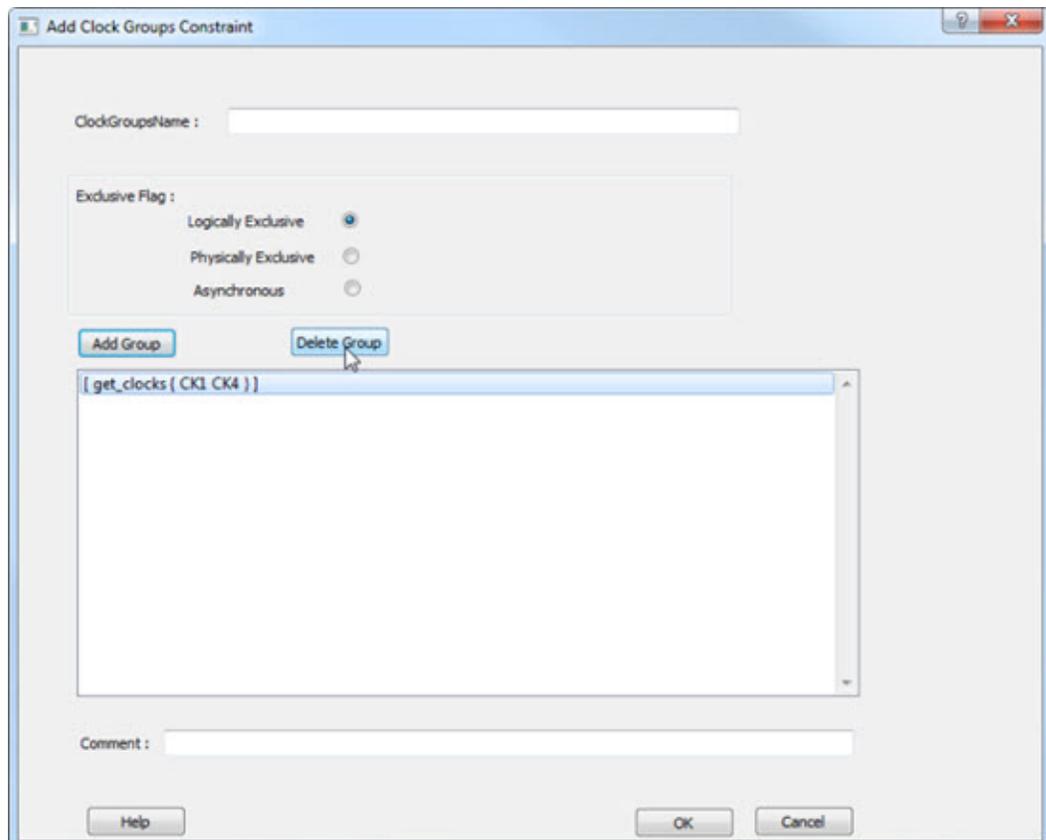


Figure 87 · Delete Group

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.

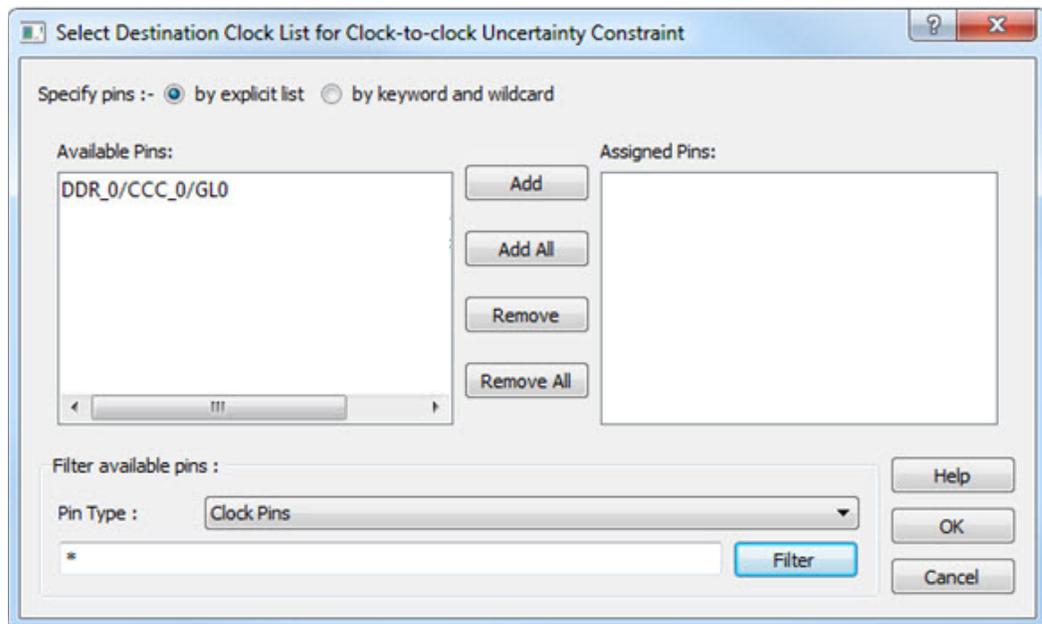


Figure 88 · Select Destination Clock Pins for Clock-to-Clock Uncertainty Dialog Box – by Explicit List

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.

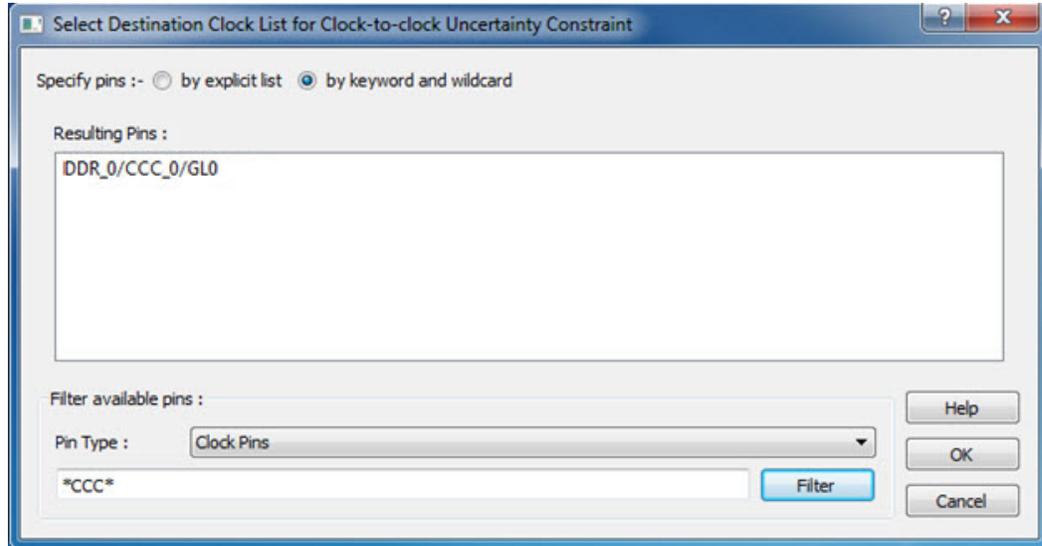


Figure 89 · Select Destination Clock Pins for Clock-to-Clock Uncertainty Dialog Box – By Keyword and Wildcard

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.

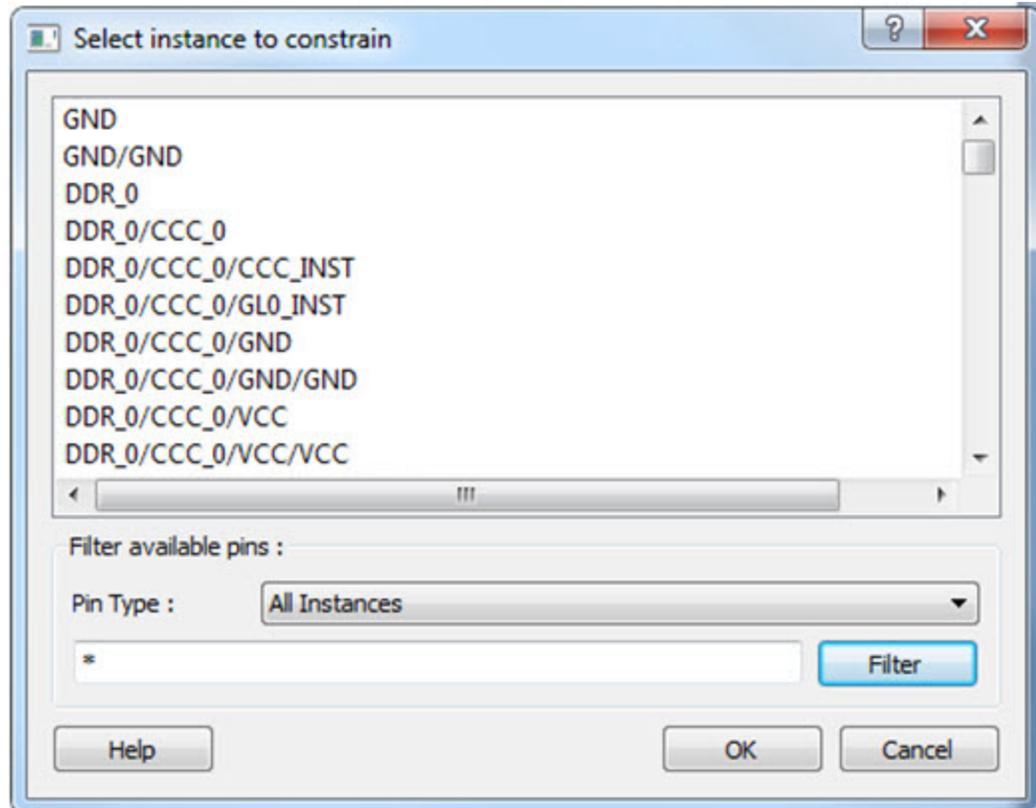


Figure 90 · Select Instance to Constrain Dialog Box

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.

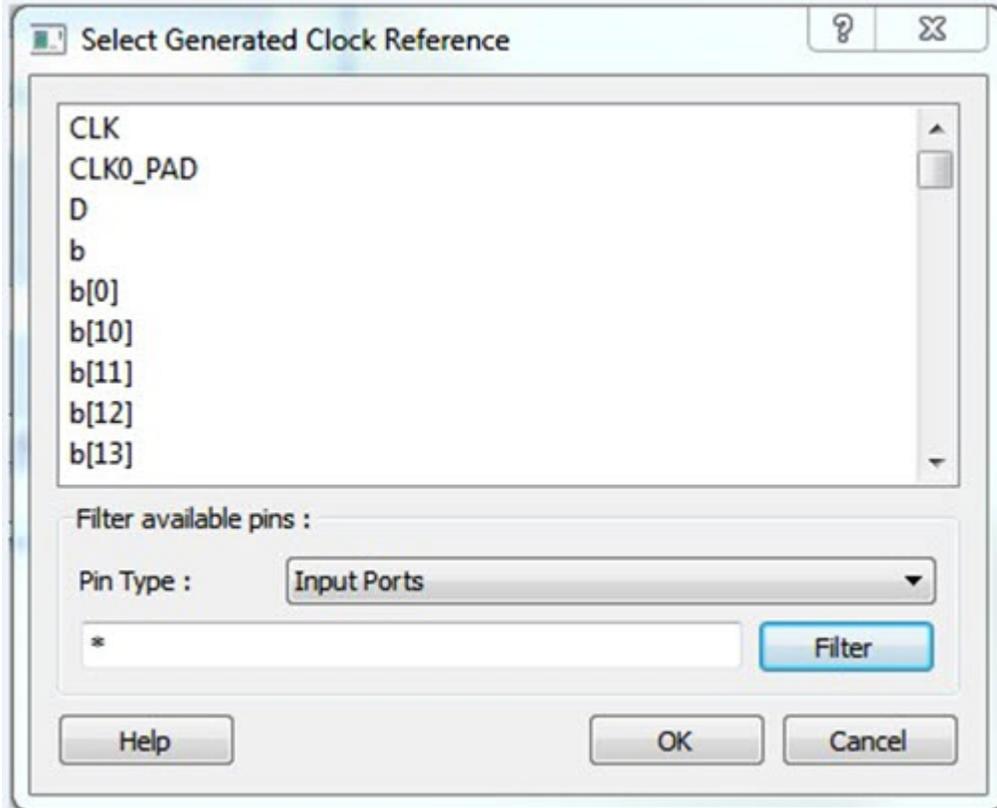


Figure 91 · Select Generated Clock Reference Dialog Box

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.

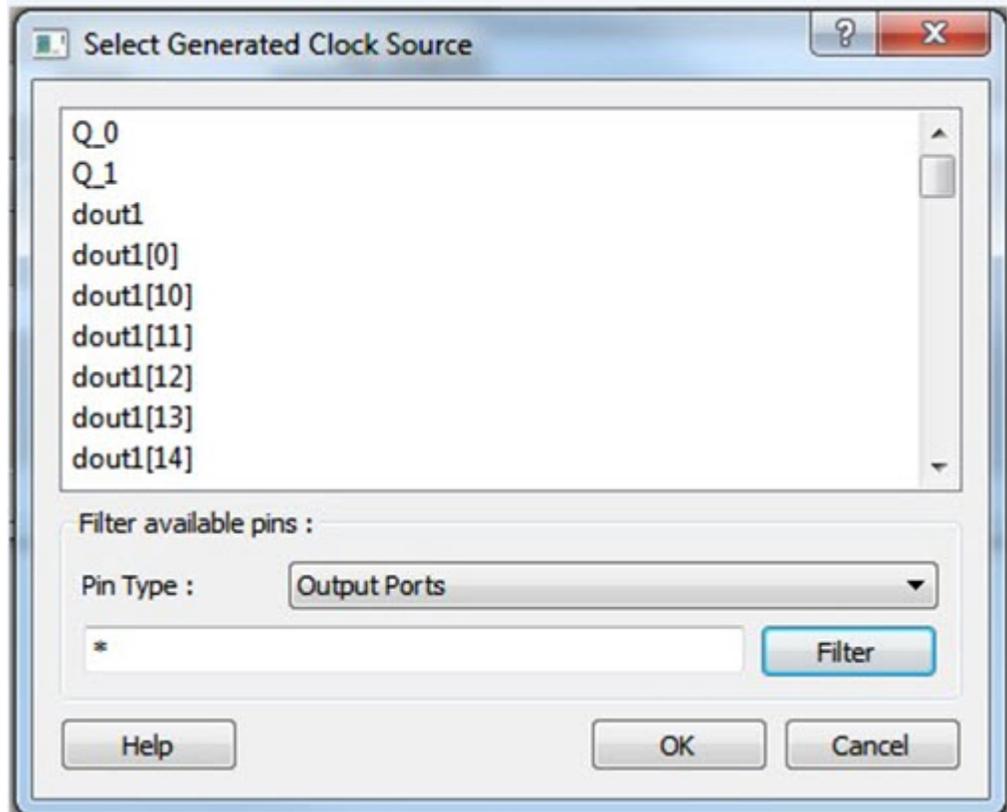


Figure 92 · Selected Generated Clock Source Dialog Box

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.

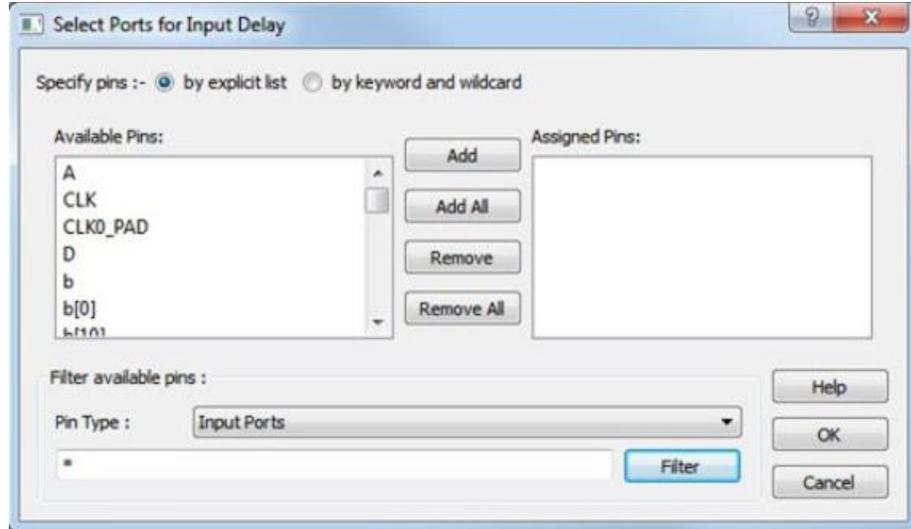


Figure 93 · Select Input Port for Input Delay Dialog Box

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.

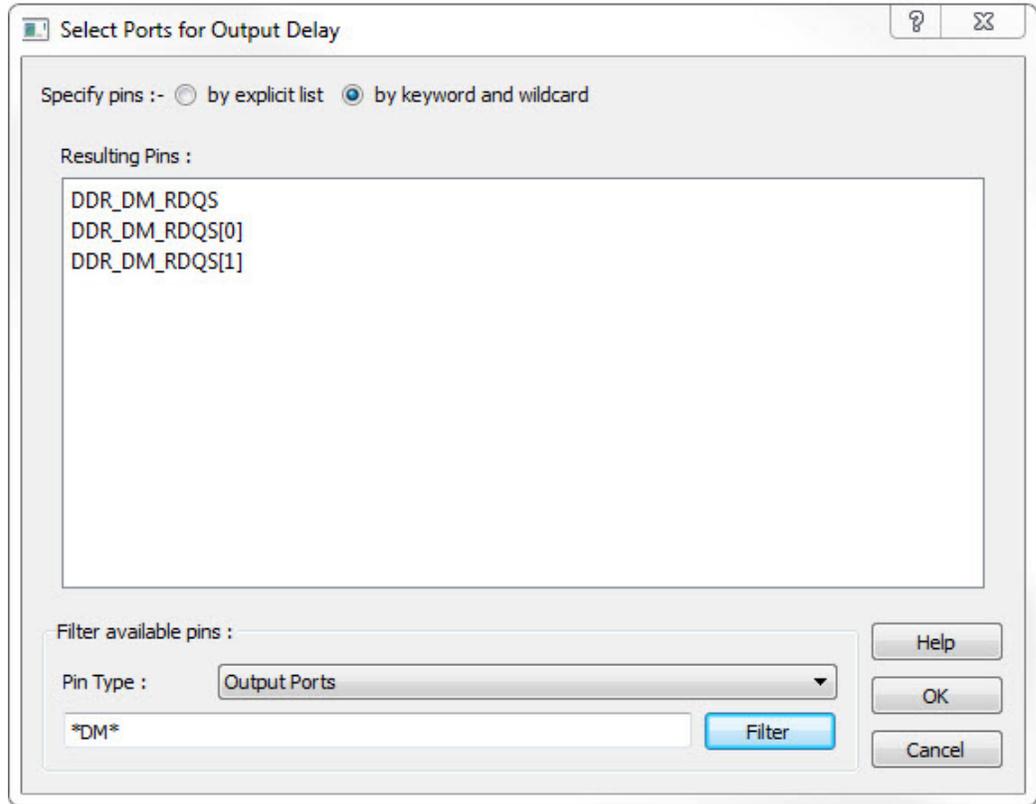


Figure 94 · Select Ports for Output Delay Dialog Box – By Keyword and Wildcard

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 .

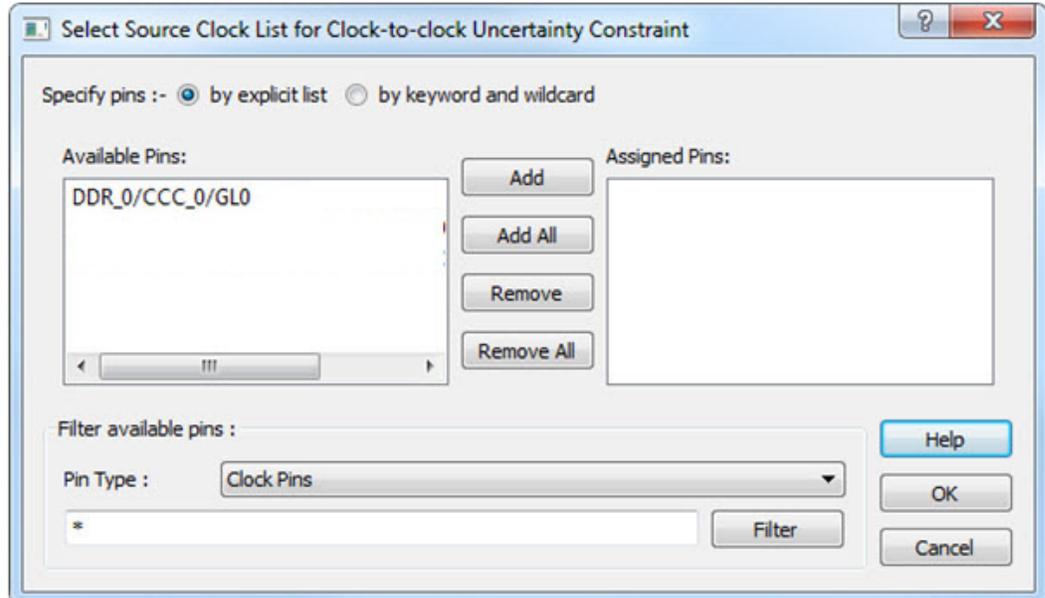


Figure 95 · Select Source Clock List for Clock-to-Clock Uncertainty Constraint Dialog Box – By Explicit List

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.

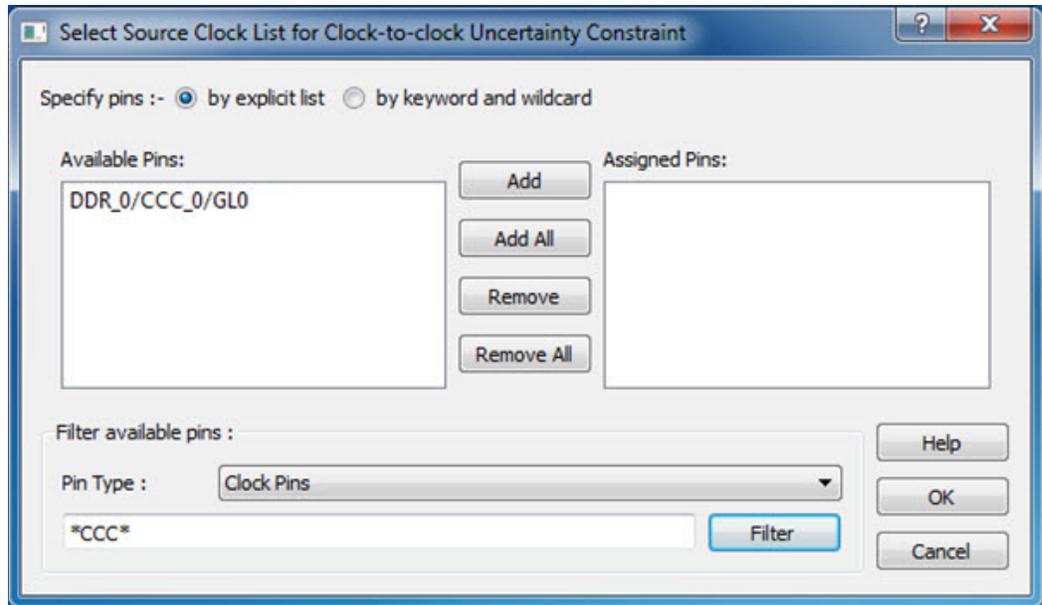


Figure 96 · Select Source Clock List for Clock-to-Clock Uncertainty Constraint Dialog Box – By Keyword and Wildcard

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).

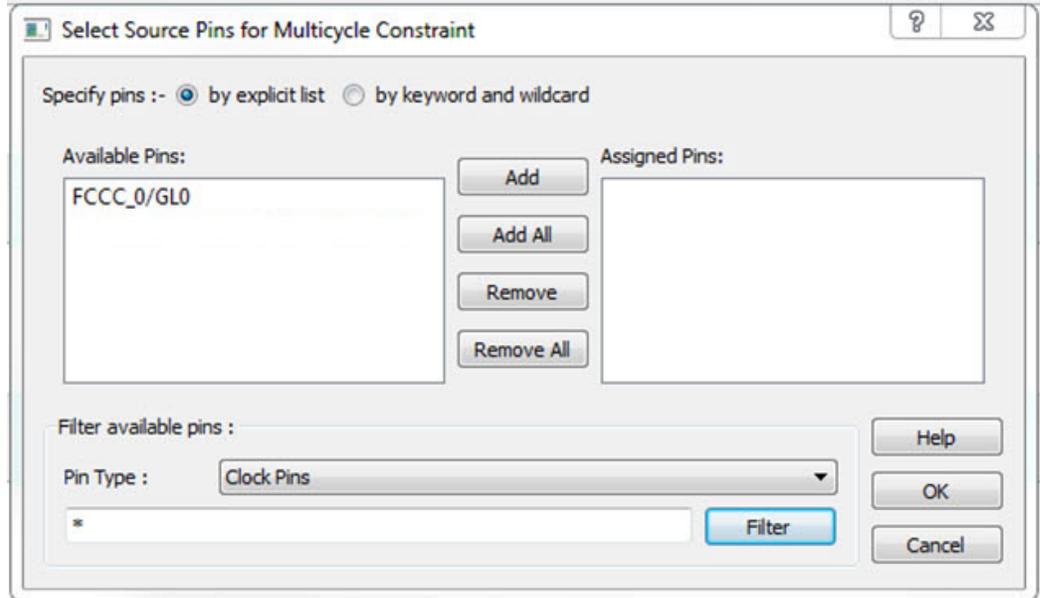


Figure 97 · Select Source Pins for Multicycle Constraint Dialog Box (specify by explicit list)

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).

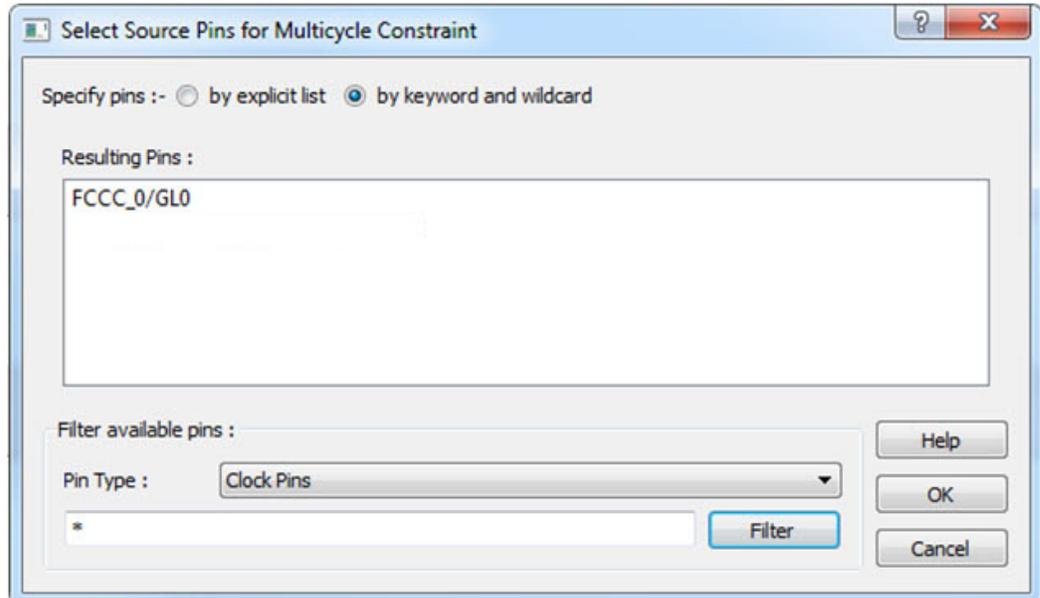


Figure 98 · Select Source Pins for Multicycle Constraint Dialog Box (specified by keyword and wildcard)

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.

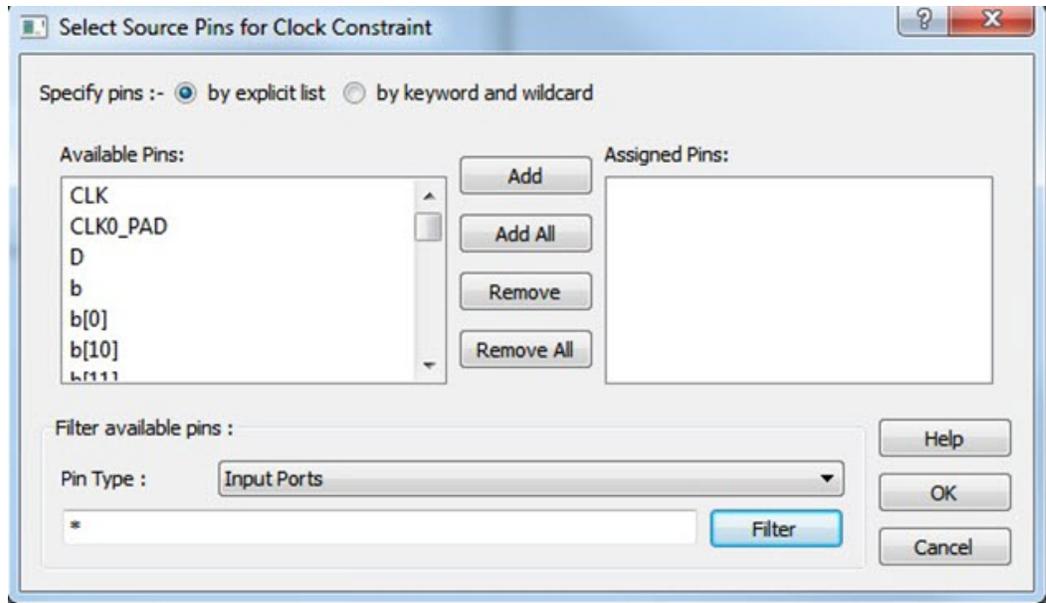


Figure 99 · Select Source Pins for Clock Constraint Dialog Box

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.

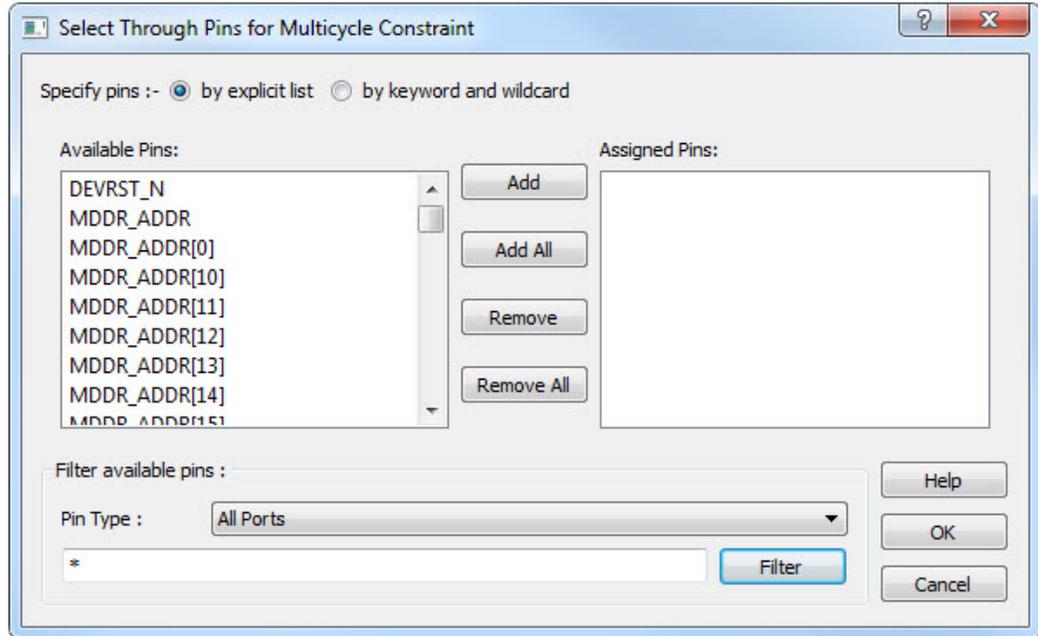


Figure 100 · Select Through Pins for Multicycle Path Constraint Dialog Box – By Explicit List

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.

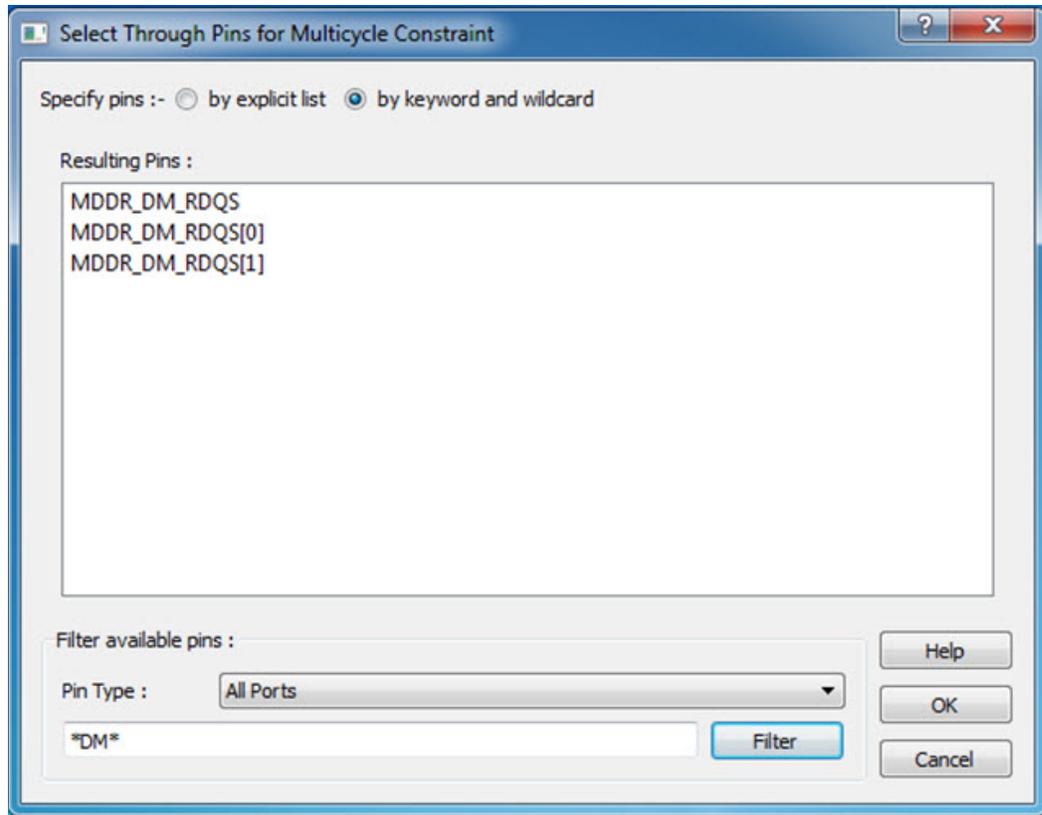


Figure 101 · Select Through Pins for Multicycle Path Constraint Dialog Box – By Keyword and Wilcard

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.

Constraint Manager – Floor Planner Tab

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

Available actions are:

- **New** – Creates a new floorplanning PDC file and saves it into the <Project_location>\constraint\fp folder.
- **Import** – Imports an existing floorplanning PDC file into the Libero SoC project. The floorplanning PDC file is copied into the <Project_location>\constraint\fp folder.
- **Link** – Creates a link in the project's constraint folder to an existing floorplanning PDC file (located and maintained outside of the Libero SoC project).

- **Edit with Chip Planner** – Opens the Chip Planner tool to modify the floorplanning PDC file(s) associated with the Place and Route tool.
- **Check** – Checks the legality of the PDC file(s) associated with the Place and Route tool against the gate level netlist.

When the Chip Planner tool is invoked or the constraint check is performed, all files associated with the Place and Route tool are passed for processing.

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

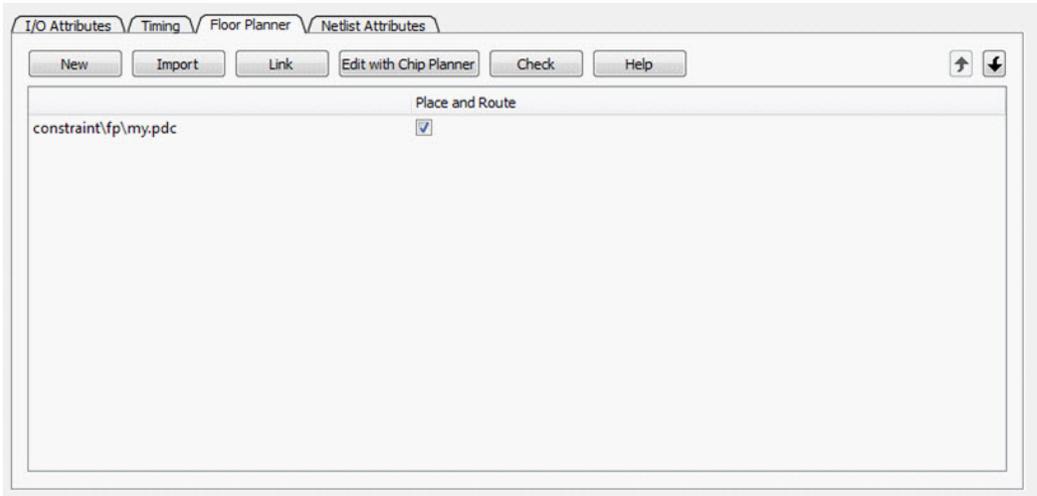


Figure 102 · Constraint Manager – Floor Planner Tab

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

- **Set/Unset as Target** – Sets or clears the selected file as the target to store new constraints created in the Chip Planner tool. Newly created constraints only go into the target constraint file. Only one file can be set as target.
- **Open in Text Editor** – Opens the selected constraint file in the Libero Text Editor.
- **Clone** - Copies the file to a file with a different name. The original file name and its content remain intact.
- **Rename** - Renames the file to a different name.
- **Copy File Path** - Copies the file path to the clipboard.
- **Delete From Project and Disk** - Deletes the selected file from the project and from the disk.
- **Unlink: Copy file locally** – Removes the link and copies the file into the <Project_location>\constraint\fp folder. The selection is available only for linked constraint files.

File and Tool Association

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

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

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

Constraint Manager – Netlist Attributes Tab

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

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

Available operations are:

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

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

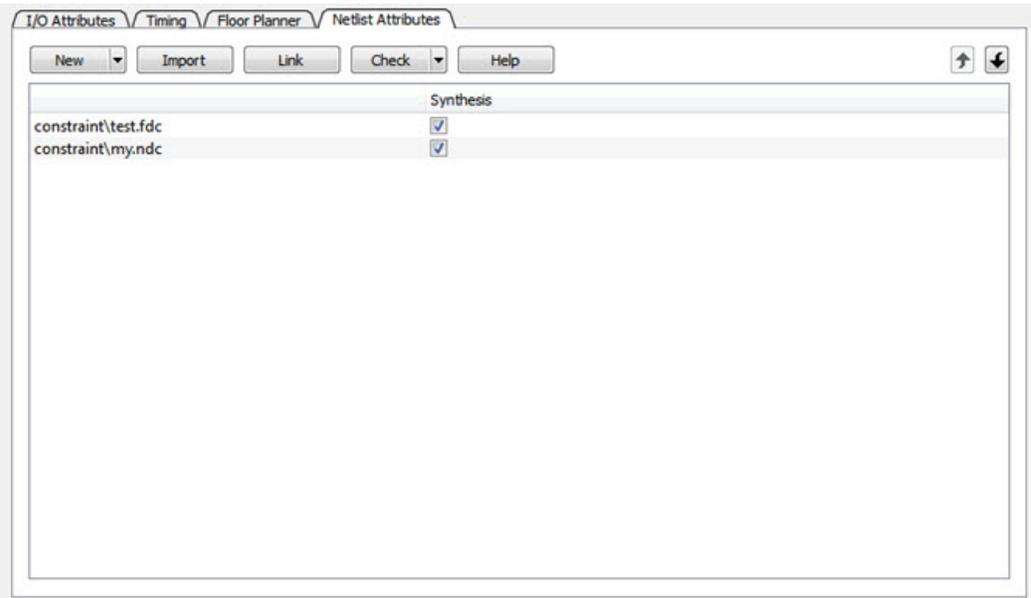


Figure 103 · Constraint Manager – Netlist Attributes Tab

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

- **Open in Text Editor** – Opens the selected constraint file in the Libero SoC Text Editor.
- **Clone** - Copies the file to a file with a different name. The original file name and its content remain intact.
- **Rename** - Renames the file to a different name.
- **Copy File Path** - Copies the file path to the clipboard.
- **Delete From Project and Disk** – Deletes the file from the project and from the disk.

- **Unlink: Copy file locally** – Removes the link and copies the file into the <Project_location>\constraint folder. This menu item is available only for linked constraint files.

File and Tool Association

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

Click the checkbox under **Synthesis** (Compile) to associate/disassociate the file from Synthesis (Compile).

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

When Synthesis is ON for a project, the Design Flow Synthesis action runs both the synthesis engine and the post-synthesis compile engine.

When Synthesis is OFF for a project, the Design Flow Synthesis action is replaced by the Compile action and runs the compile engine on the gate-level netlist (EDIF or Verilog) available in the project.

Implement Design

Synthesize

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

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

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

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

Synthesize Options

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

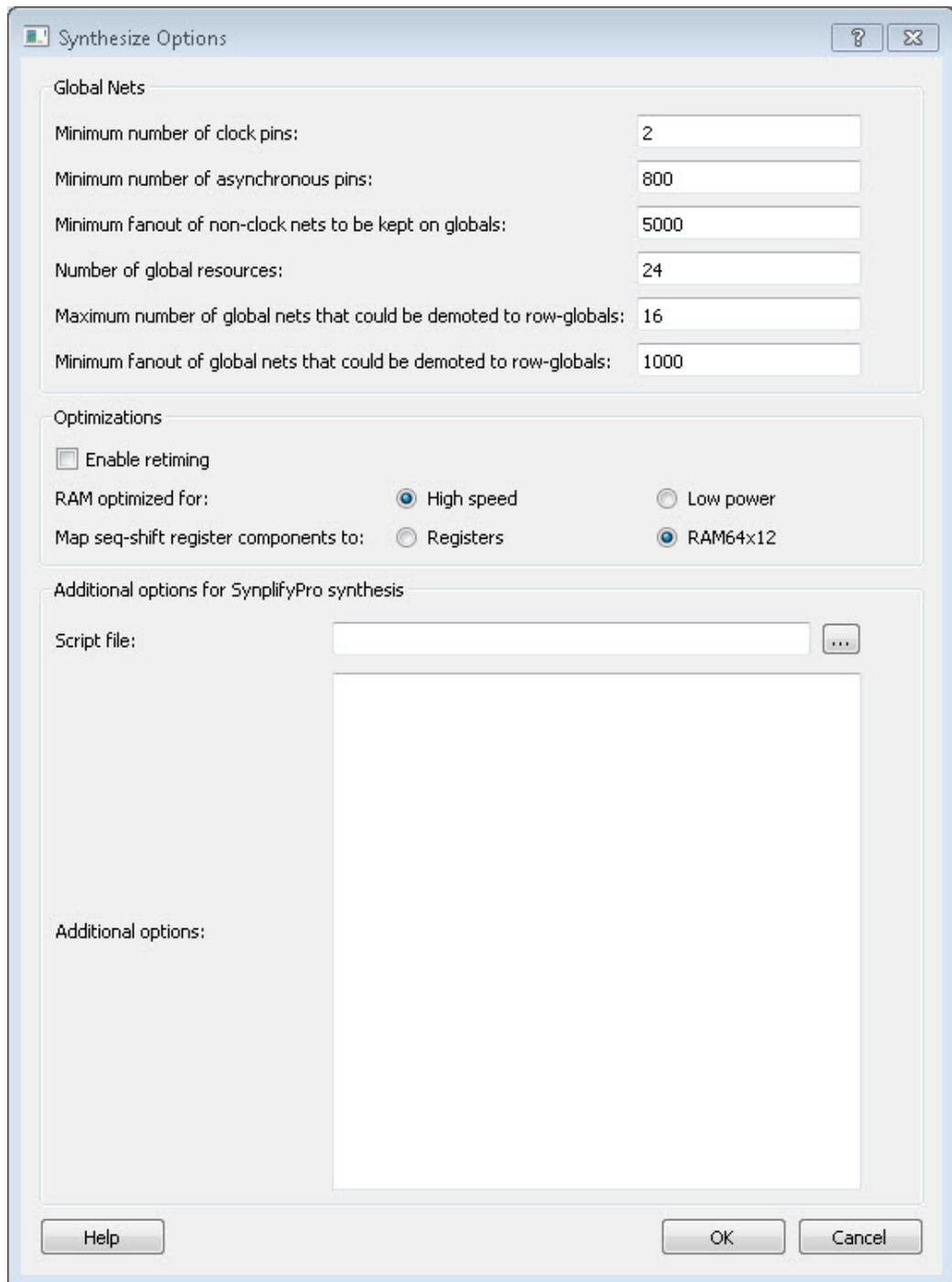


Figure 104 · Synthesize Options Dialog Box

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

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

Optimizations

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

RAM optimized for:

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

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

Map seq-shift register components to:

Use this option to select the mapping of sequential logic:

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

Additional options for Synplify Pro synthesis

Script File

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

Additional Options

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

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

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

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

```

set_option -use_fsm_explorer 0/1
set_option -frequency 200.000000
set_option -write_verilog 0/1
set_option -write_vhdl 0/1
set_option -resolve_multiple_driver 1/0
set_option -rw_check_on_ram 0/1
set_option -auto_constrain_io 0/1
set_option -run_prop_extract 1/0
set_option -default_enum_encoding default/onehot/sequential/gray
set_option -maxfan 30000
set_option -report_path 5000
set_option -update_models_cp 0/1
set_option -preserve_registers 1/0
set_option -continue_on_error 1/0
set_option -symbolic_fsm_compiler 1/0
set_option -compiler_compatible 0/1
set_option -resource_sharing 1/0
set_option -write_apr_constraint 1/0
set_option -dup 1/0
set_option -enable64bit 1/0
set_option -fanout_limit 50
set_option -frequency auto
set_option -hdl_define SLE_INIT=2
set_option -hdl_param -set "width=8"
set_option -looplimit 3000
set_option -fanout_guide 50
set_option -maxfan_hard 1/0
set_option -num_critical_paths 10
set_option -safe_case 0/1

```

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

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

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

```

add_file <*>
impl <*>
project_folder <*>
add_folder <*>
constraint_file <*>
project <*>

```

```

project_file <*>
open_file <*>
set_option -part
set_option -package
set_option -speed_grade
set_option -top_module
set_option -technology
set_option -opcond
set_option -vlog_std
set_option -vhdl2008
set_option -disable_io_insertion
set_option -async_globalthreshold
set_option -clock_globalthreshold
set_option -globalthreshold
set_option -low_power_ram_decomp
set_option -retiming

```

Synplify Pro ME

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

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

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

To run synthesis using Synplify Pro ME with custom settings:

1. If you have set Synplify as your default synthesis tool, right-click **Synthesize** in the Libero SoC Design Flow window and choose **Open Interactively**. Synplify starts and loads the appropriate design files, with a few pre-set default values.
2. From Synplify's **Project** menu, choose **Implementation Options**.
3. Set your specifications and click **OK**.
4. Deactivate synthesis of the defparam statement. The defparam statement is only for simulation tools and is not intended for synthesis. Embed the defparam statement in between **translate_on** and **translate_off** synthesis directives as follows :

```

/* synthesis translate_off */
defparam M0.MEMORYFILE = "meminit.dat"

```

```

/*synthesis translate_on */
// rest of the code for synthesis

```

5. Click the **RUN** button. Synplify compiles and synthesizes the design into an 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.

Compile Netlist

The Compile Netlist step appears in the Design Flow window 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.

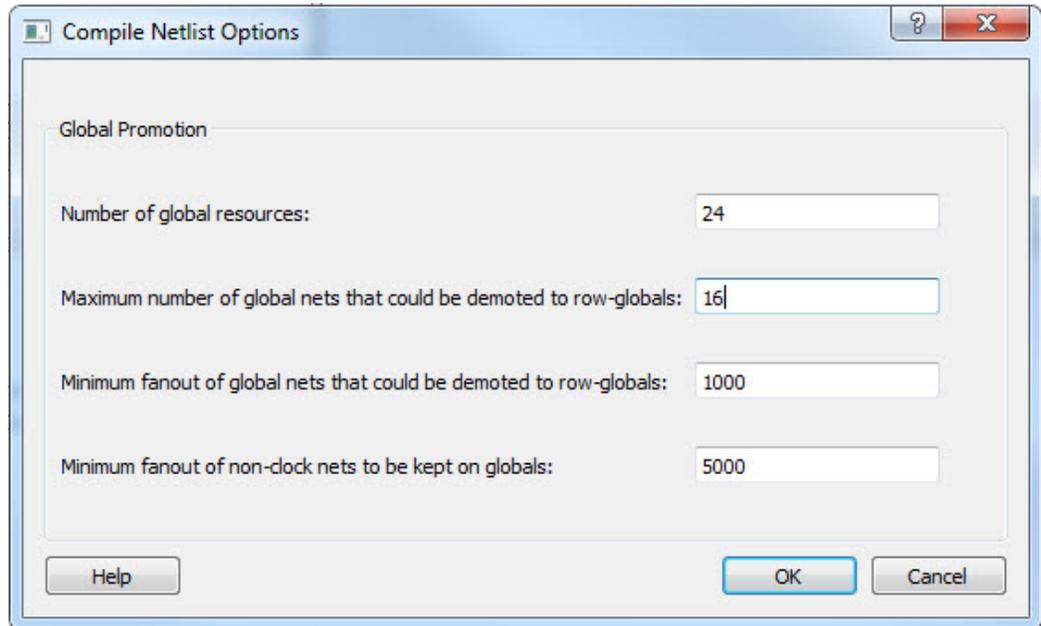


Figure 105 · Compile Netlist Options Dialog Box

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 1000. If you run out of global routing resources for your design, reduce this number (to allow more globals to be demoted) or select a larger die for your design.

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

Resource Usage

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

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

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

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

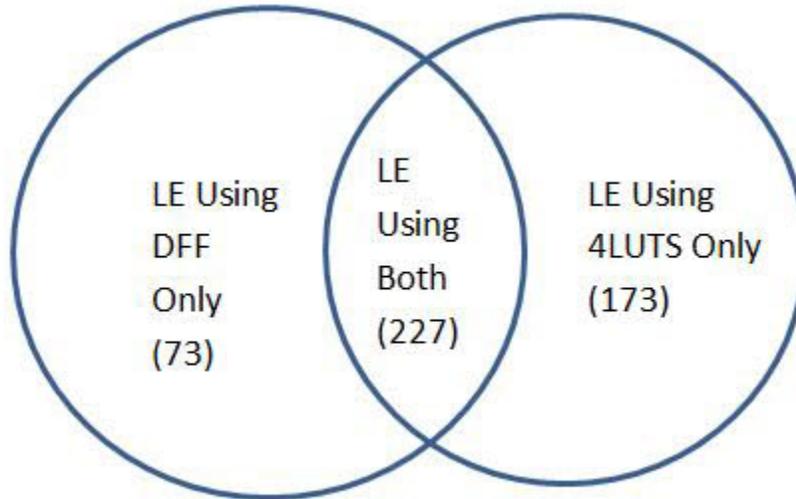
The number of 4LUTs in the report are the total number used for your design, regardless of whether or not they are combined with the DFFs. Similarly, the number of DFFs in the report are the total number used for your design, regardless of whether or not they are combined with 4LUT's.

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.

Constraint Flow in Implementation

Design State Invalidation

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

Status Icon	Tooltip	Description	Possible Causes
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 (for example, 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 fails to run.	Invalid command options or switches, invalid design objects, invalid design constraints.
	Design State is Out of Date.	Tool state changes from PASS to OUT OF DATE.	Since the last successful run, design source design files, constraint files or constraint file/tool association, constraint files order, tool options, and/or project settings have changed.
	Timing	Timing Verification runs	Design fails Timing Analysis. Design has either set-up or hold

Status Icon	Tooltip	Description	Possible Causes
	Constraints have not been met.	successfully but the design fails to meet timing requirements.	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 components
- Constraint files associated with a tool
- Upstream tools in the Design Flow:
 - If the tool state of a Design Flow tool changes from PASS to OUT OF DATE, the tool states of all the tools below it in the Design Flow, if already run and are in PASS state, also change to OUT OF DATE with appropriate tooltips. For example, if the Synthesis tool state changes from PASS to OUT OF DATE, the tool states of Place and Route tool as well as all the tools below it in the Design Flow change to OUT OF DATE.
 - If a Design Flow tool is CLEANED, the tool states of all the tools below it in the Design Flow, if already run, change from PASS to OUT OF DATE.
 - If a Design Flow tool is rerun, the tool states of all the tools below it in the Design Flow, if already run, are CLEANED.
- Tool Options
 - If the configuration options of a Design Flow tool (right-click the tool and choose **Configure Options**) are modified, the tool states of that tool and all the other tools below it in the Design Flow, if already run, are changed to OUT OF DATE with appropriate tooltips.
- Project Settings:
 - Device selection
 - Device settings
 - Design Flow
 - Analysis operating conditions

Setting Changed	Note	Design Flow Tools Affected	New State of the Affected Design Flow Tools
Die	Part# is changed	All	CLEANED/NEW
Package	Part# is changed	All	CLEANED/NEW
Speed	Part# is changed	All	CLEANED/NEW

Setting Changed	Note	Design Flow Tools Affected	New State of the Affected Design Flow Tools
Core Voltage	Part# is changed	All	CLEANED/NEW
Range	Part# is changed	All	CLEANED/NEW
Default I/O Technology		Synthesize, and all tools below it	OUT OF DATE
Reserve Pins for Probes		Place and Route, and all tools below it	OUT OF DATE
PLL Supply Voltage (V)		Verify Power, Generate FPGA Array Data and all other "Program and Debug Design" tools below it	OUT OF DATE
Power On Reset Delay		Generate FPGA Array Data and all other "Program and Debug Design" tools below it	OUT OF DATE
System controller suspended mode		Generate FPGA Array Data and all other "Program and Debug Design" tools below it	OUT OF DATE
Preferred Language		None	N/A
Enable synthesis		All	OUT OF DATE
Synthesis gate level netlist format		Synthesize	CLEANED/NEW

Setting Changed	Note	Design Flow Tools Affected	New State of the Affected Design Flow Tools
Reports(Maximum number of high fanout nets to be displayed)		None	N/A
Abort flow if errors are found in PDC		None	N/A
Abort flow if errors are found in SDC		None	N/A
Temperature range(C)		Verify Timing, Post Layout Simulate, and Verify Power	OUT OF DATE
Core voltage range(V)		Verify Timing, Post Layout Simulate, and Verify Power	OUT OF DATE
Default I/O voltage range		Verify Timing, Post Layout Simulate, and Verify Power	OUT OF DATE

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

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

Edit Constraints

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

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

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

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

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

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

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

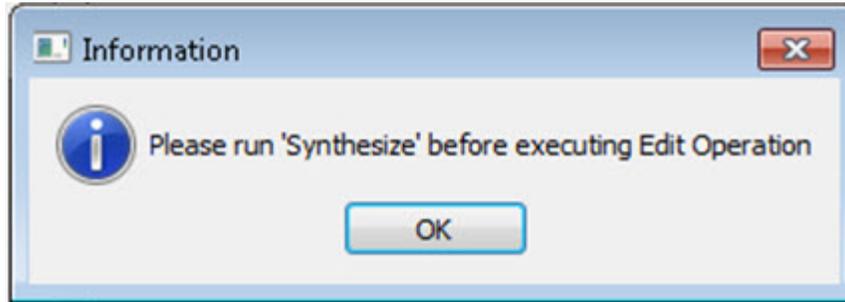
Constraints in the SDC constraint file that are read by the Constraint Editor and subsequently modified by you will be written back to the SDC file when you save the edits and close the Constraint Editor.

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

Constraint Type and Interactive Tool

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

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



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

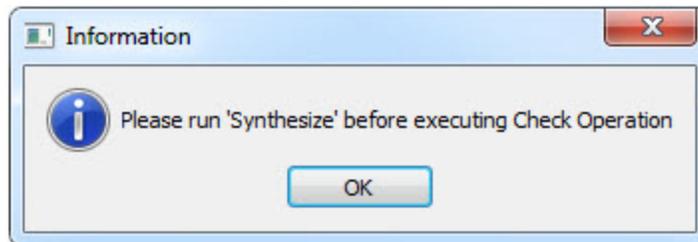
Check Constraints

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

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

Design State and Constraints Check

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

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

Place and Route

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

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.

See Also

[Multiple Pass Layout Configuration](#)
[extended_run.lib](#)

Multiple Pass Layout Configuration

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

- Before running Multiple Pass Layout, save your design.
- Multiple Pass Layout is supported by all families.
- Multiple Pass Layout saves your design file with the pass that has the best layout results. If you want to preserve your existing design state, you should save your design file with a different name before proceeding. To do this, from the File menu, choose **Save As**.
- Four types of reports (timing, maximum delay timing violations, minimum delay timing violations, and power) for each pass are written to the working directory to assist you in later analysis:
 - `<root_module_name>_timing_r<runNum>_s<seedIndex>.rpt`
 - `<root_module_name>_timing_violations_r<runNum>_s<seedIndex>.rpt`
 - `<root_module_name>_timing_violations_min_r<runNum>_s<seedIndex>.rpt`
 - `<root_module_name>_power_r<runNum>_s<seedIndex>.rpt`
 - `<root_module_name>_iteration_summary.rpt` provides additional details about the saved files.

To configure your multiple pass options:

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

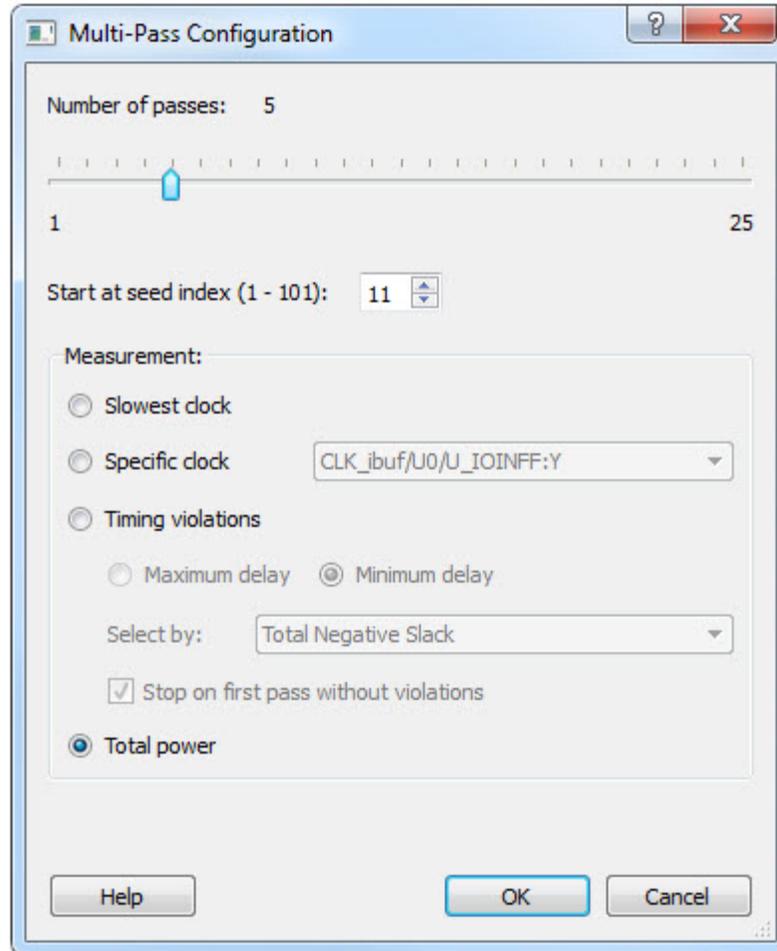


Figure 106 · Multi-Pass Configuration Dialog Box

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

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

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

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

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

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

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

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

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

Iteration Summary Report

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

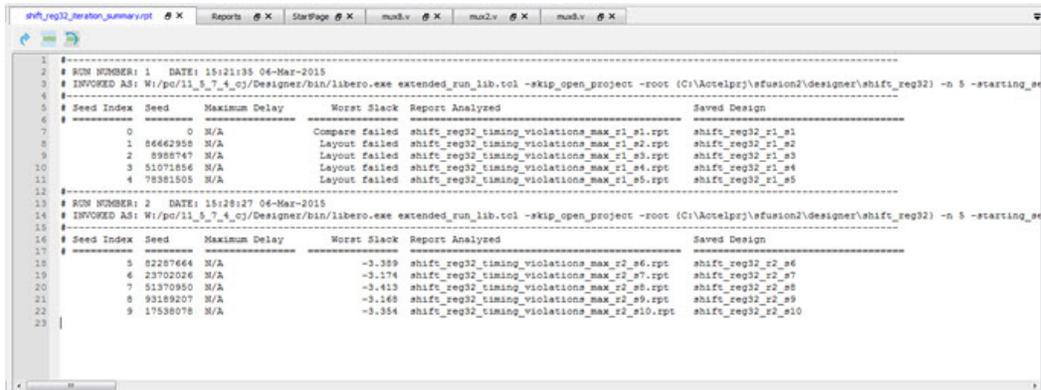


Figure 107 · Iteration Summary Report

See Also

- [Place and Route](#)
- [extended_run_lib](#)

Verify Post Layout Implementation

SmartTime

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

Note: See the [Timing Constraints Editor](#) for help with creating and editing timing constraints.

Static Timing Analysis (STA)

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

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

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

Timing Constraints

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

Timing Analysis

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

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

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

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

SmartTime and Place and Route

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

SmartTime and Timing Reports

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

- Timing Report (for both Max and Min Delay Analysis)

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

SmartTime and Cross-Probing into Chip Planner

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

- Ports
- Macros
- Timing Paths

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 Static Timing Analyzer User Guide](#) for details.

Verify Power

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

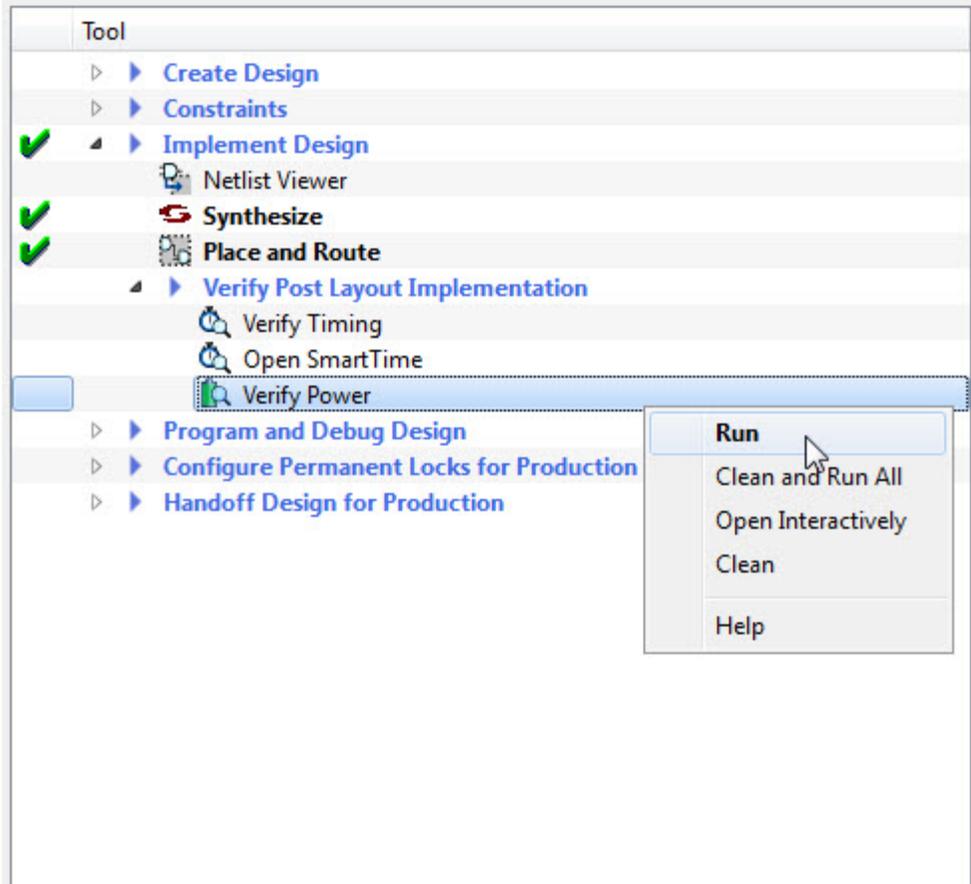


Figure 108 · Verify Power right-click menu

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 User Guide](#)

Handoff Design

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.

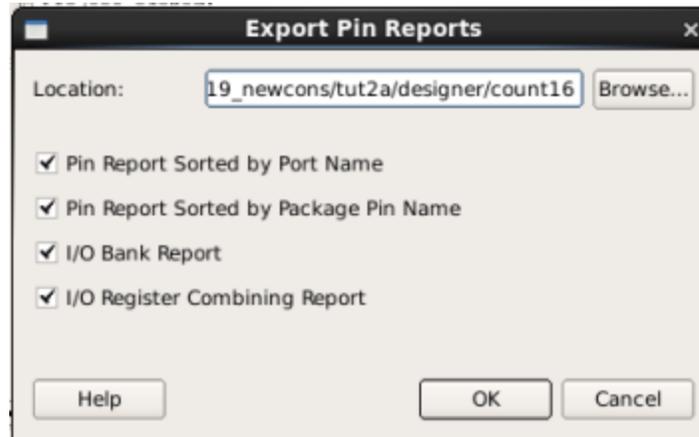


Figure 109 · Export Pin Reports Dialog Box

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](#).

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 110 · 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 .

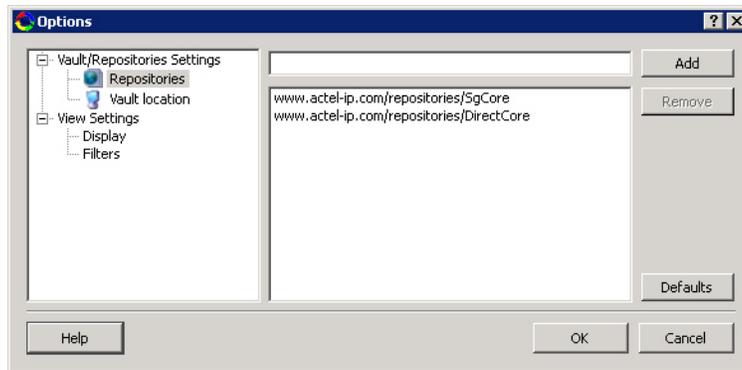


Figure 111 · Catalog Display Options Dialog Box

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.

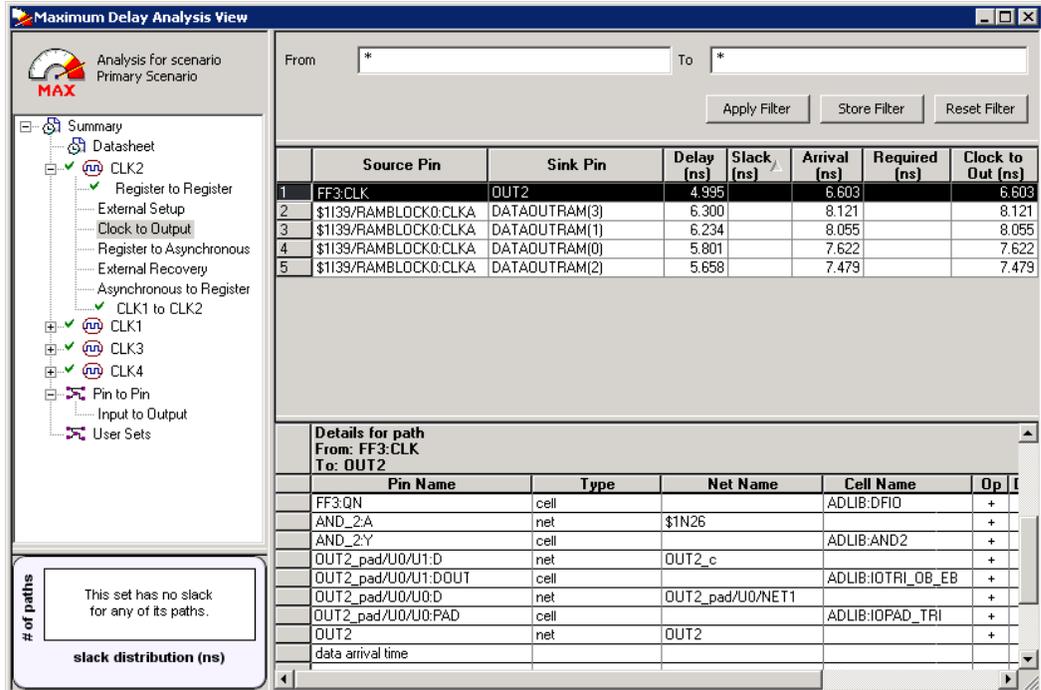


Figure 112 · Maximum Delay Analysis View

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	I/O Standard	Output Drive (mA)	Stew	Resistor Pull	Skew	Output Load	Use I/O Reg
1	CLK2	ADLIB:CLKBUF	13	<input type="checkbox"/>	Bank1	LVTTL	--	--	None	--		<input type="checkbox"/>
2	CLK4	ADLIB:IBUF	15	<input type="checkbox"/>	Bank1	LVTTL	--	--	None	--		<input type="checkbox"/>
3	MACOR(3)	ADLIB:IBUF	85	<input type="checkbox"/>	Bank0	LVTTL	--	--	None	--		<input type="checkbox"/>
4	DATAOUTRAM(2)	ADLIB:OUTBUF	86	<input type="checkbox"/>	Bank0	LVTTL	12	High	None	<input type="checkbox"/>	35	<input type="checkbox"/>
5	OUT2	ADLIB:OUTBUF	16	<input type="checkbox"/>	Bank1	LVTTL	12	High	None	<input type="checkbox"/>	75	<input type="checkbox"/>

Figure 113 · 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.

Core Manager

The Core Manager only lists cores that are in your current project. If any of the cores in your current project are not in your vault, you can use the Core Manager to download them all at once.

For example, if you download a sample project and open it, you may not have all the cores in your local vault. In this instance you can use the Core Manager to view and download them with one click. Click **Download All** to add any missing cores to your vault. To add any individual core, click the green download button.

To view the Core Manager, from the **View** menu choose **Windows > Cores**.

The column headings in the Core Manager are:

- **Name** - Core name.
- **Vendor** - Source of the core.
- **Core Type** - Core type.
- **Version** - Version of the core used in your project; it may be a later version than you have in your vault. If so, click **Download All** to download the latest version.

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**.

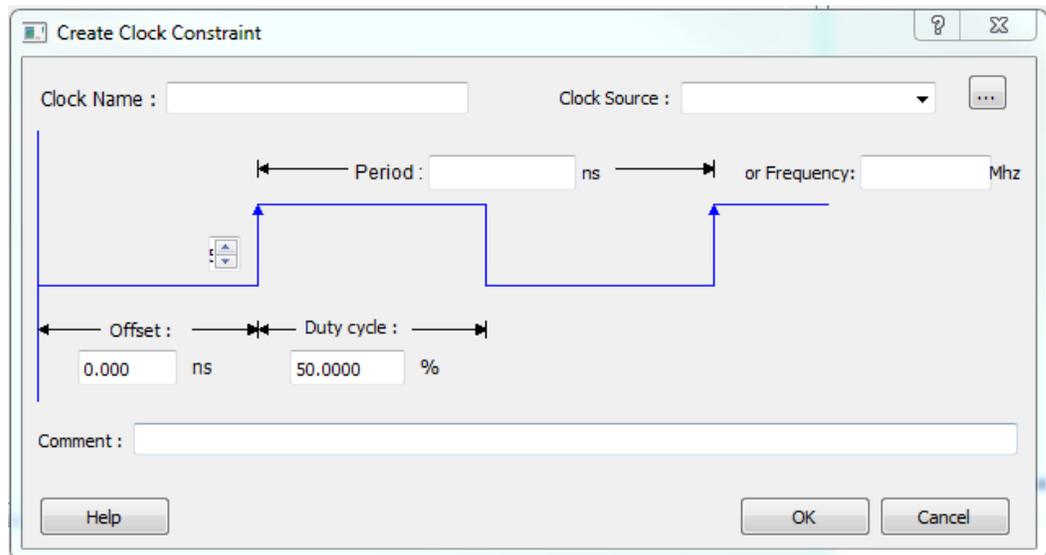


Figure 114 · Create Clock Constraint Dialog Box

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(zero) 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**.

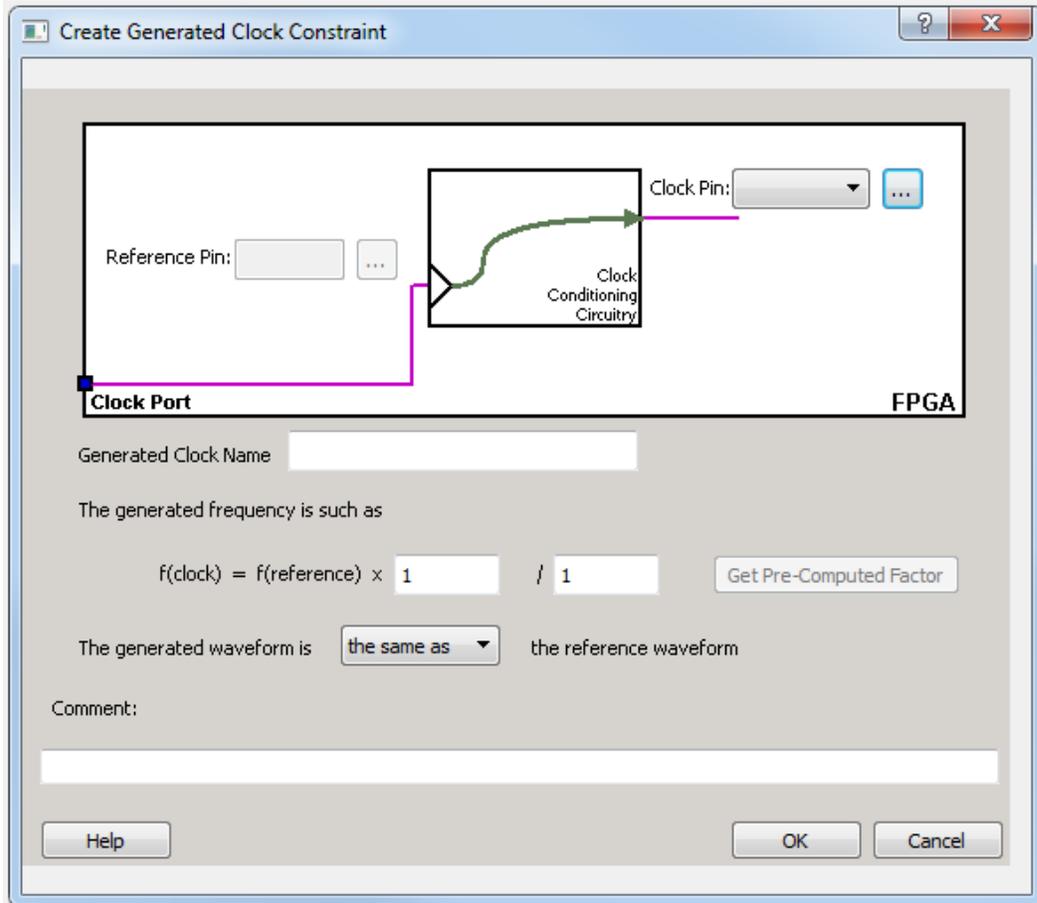


Figure 115 · Create Generated Clock Constraint

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.

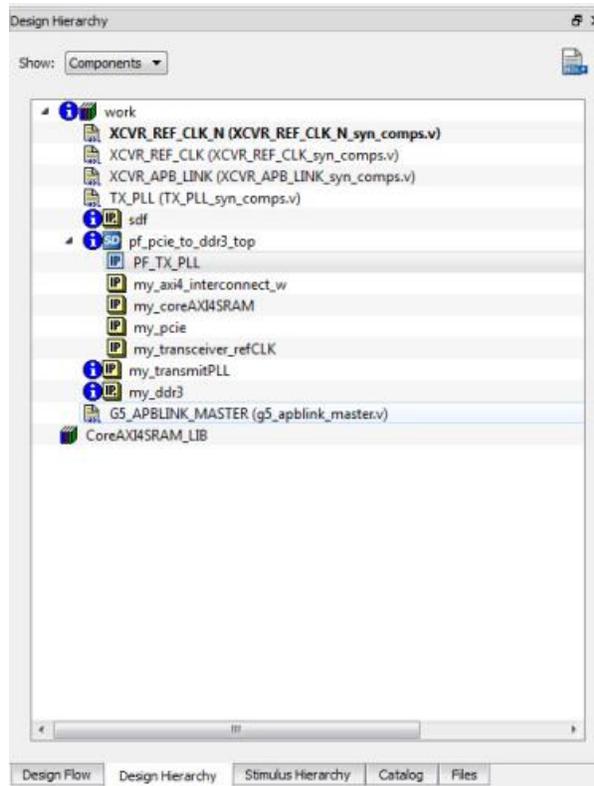


Figure 116 · Design Hierarchy

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 2 · Design Hierarchy Icons

Icon	Description
	SmartDesign component
	SmartDesign component with HDL netlist not generated
	IP core was instantiated into SmartDesign but the HDL netlist has not been generated
	Core
	Error during core validation
	Updated core available for download
	HDL netlist

Editable Constraints Grid

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

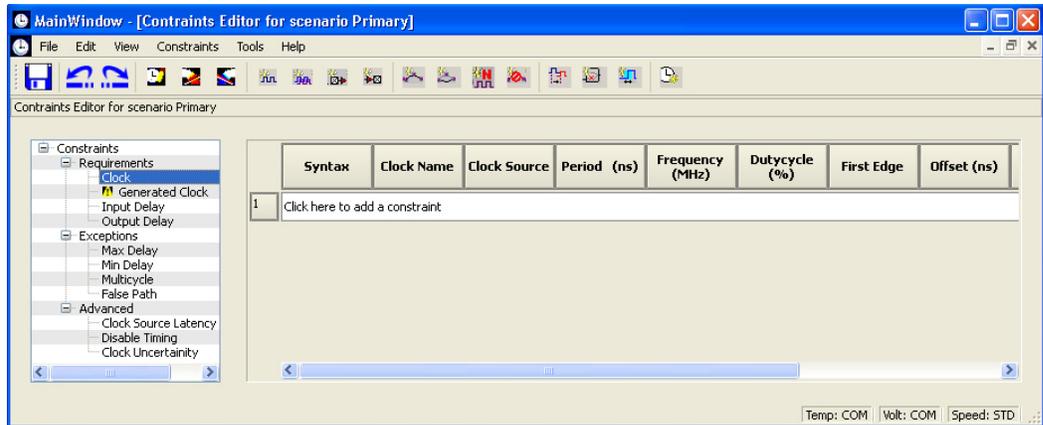


Figure 117 · Constraints Editor

To add a new constraint:

1. Select a constraint type from the constraint browser.
2. Enter the constraint values in the first row and click the green check mark to apply your changes. To cancel the changes press the red cancel mark.
3. The new constraint is added to the Constraint List. The green syntax flag indicates that the constraint was successfully checked.

To edit a constraint:

1. Select a constraint type from the constraint browser.
2. Select the constraint, edit the values and click the green check mark to apply your changes. To cancel the changes press the red cancel mark. The green syntax flag indicates that the constraint was successfully checked.

To delete a constraint:

1. Select a constraint type from the constraint browser.
2. Right-click the constraint you want to delete and choose **Delete Constraint**.

extended_run_lib

Note: This is not a Tcl command; it is a shell script that can be run from the command line.

The extended_run_lib Tcl script enables you to run the multiple pass layout in batch mode from a command line.

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

Note:

- There is no option to save the design files from all the passes. Only the (Timing or Power) result reports from all the passes are saved.

Arguments

-root path/designer/module_name

The path to the root module located under the designer directory of the Libero project.

[-n numPasses]

Sets the number of passes to run. The default number of passes is 5.

`[-starting_seed_index numIndex]`

Indicates the specific index into the array of random seeds which is to be the starting point for the passes. Value may range from 1 to 100. If not specified, the default behavior is to continue from the last seed index that was used.

`[-compare_criteria value]`

Sets the criteria for comparing results between passes. The default value is set to frequency when the `-c` option is given or timing constraints are absent. Otherwise, the default value is set to violations.

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

`[-c clockName]`

Applies only when the clock frequency comparison criteria is used. Specifies the particular clock that is to be examined. If no clock is specified, then the slowest clock frequency in the design in a given pass is used. The clock name should match with one of the Clock Domains in the Summary section of the Timing report.

`[-analysis value]`

Applies only when the timing violations comparison criteria is used. Specifies the type of timing violations (the slack) to examine. The following table shows the acceptable values for this argument:

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

`[-slack_criteria value]`

Applies only when the timing violations comparison criteria is used. Specifies how to evaluate the timing violations (slack). The type of timing violations (slack) is determined by the `-analysis` option. The following table shows the acceptable values for this argument:

Value	Description
worst	Sets the timing violations criteria to Worst slack. For each pass obtains the most amount of negative slack (or least amount of positive slack if all constraints are met) from the timing violations report. The largest value out of all passes will determine the best pass. This is the

Value	Description
	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.

Exceptions

None

Example

```
D:/Liber0_11_3_SP1/Designer/bin/libero
script:D:/Liber0_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, RTG4, PolarFire](#)

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

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 - *_ba.sdf, *_ba.v(hd), TCL (used to run designer), designer.log (logfile)

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

simulation directory - meminit.dat, modelsim.ini files

smartgen directory - GEN files and LOG files from generated cores

stimulus directory - BTIM and VHD stimulus files

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

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

Importing Files

Anything that describes your design, or is needed to program the device, is a project source. These may include schematics, HDL files, simulation files, testbenches, etc. Import these source files.

To import a file:

1. From the **File** menu, choose **Import Files**.
2. In **Files of type**, choose the file type.
3. In **Look in**, navigate to the drive/folder where the file is located.
4. Select the file to import and click **Open**.

Note: You cannot import a Verilog File into a VHDL project and vice versa.

File Types for Import

File Type	File Extension
Behavioral and Structural VHDL; VHDL Package	*.vhd, *.vhdl
Design Block Core	*.gen
Verilog Include	*.h
Behavioral and Structural Verilog	*.v, *.sv
Stimulus	*.vhd, *.vhdl, *.v, *.sv
EDIF Netlist	*.edn
Memory file	*.mem

list_clock_groups

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

```
list_clock_groups
```

Arguments

None

Supported Families

SmartFusion2, IGLOO2, RTG4, PolarFire

Example

```
list_clock_groups
```

See Also

- [set clock groups](#)
- [remove clock groups](#)

Project Settings Dialog Box

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

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

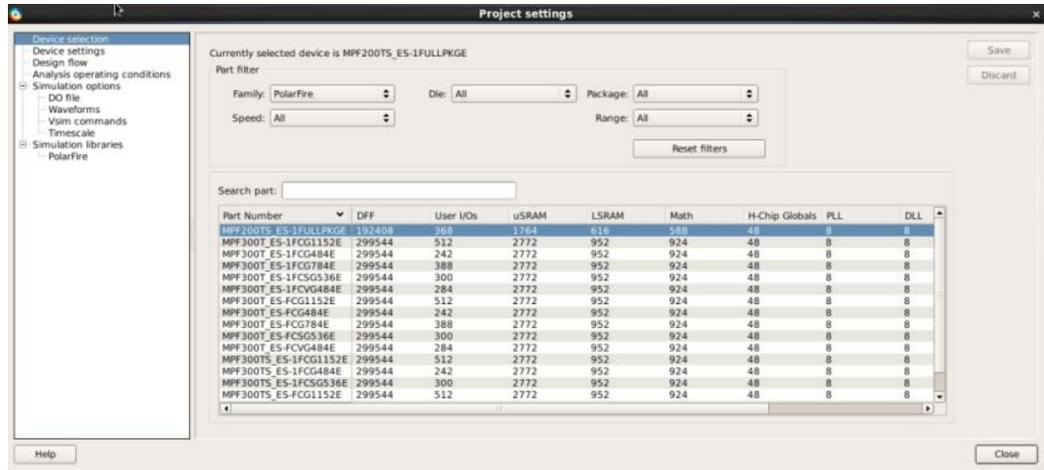


Figure 118 · Project Settings Dialog Box

Device Selection

Sets the device Family, Die, and Package for your project. See the [New Project Creation Wizard - Device Selection](#) page for a detailed description of the options.

Device Settings

Default I/O Technology - Sets all your I/Os to a default value. You can change the values for individual I/Os in the I/O Attributes Editor.

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

Design Flow

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

If your Verilog files contain System Verilog constructs, select the System Verilog option.

Note: Libero SoC supports the following Verilog and VHDL IEEE standards:

- Verilog 2005 (IEEE Standard 1364-2005)
- Verilog 2001 (IEEE Standard 1364-2001)

- Verilog 1995 (IEEE Standard 1364-1995)
- SystemVerilog 2012 (IEEE Standard 1800-2012)
- VHDL-2008 (IEEE Standard 1076-2008)
- VHDL-93 (IEEE Standard 1076-1993)

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

Enable FPGA Hardware Breakpoint Auto Instantiation

NOTE: This option is used with SmartDebug, which is not enabled in this release.

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.

Reports

Maximum number of high fanout nets to be displayed - Enter the number of high fanout nets to be displayed. The default value is 10. This means the top 10 nets with the highest fanout will appear in the <root>_compile_netlist_resource.xml Report.

Abort Flow if Errors are found in Physical Design Constraints (PDC) – Check this checkbox to abort Place and Route if the I/O or Floorplanning PDC constraint file contains errors.

Abort Flow if Errors are found in Timing Constraints (SDC) – Check this checkbox to abort Place and Route if the Timing Constraint SDC file contains errors.

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

Analysis Operating Conditions

There are no options at present for PolarFire. Only EXT range is available.

These settings are propagated to Verify Timing, Verify Power, and Backannotated Netlist for you to perform Timing/Power Analysis.

Simulation Options and Simulation Libraries

Sets your simulation options. See the [Project Settings: Simulation Options](#) topic for more information.

Project Settings: Simulation

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.

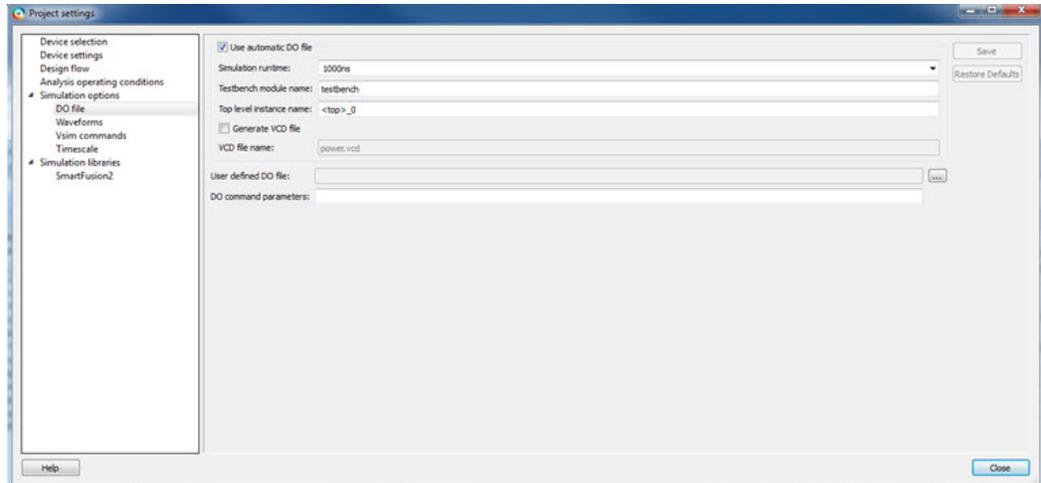


Figure 119 · Project Settings: - DO File

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 1ps.

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

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](#)

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.

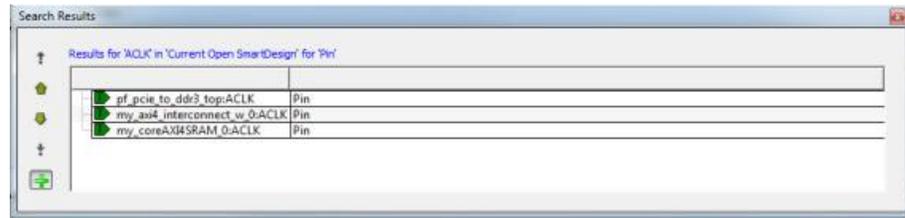


Figure 120 · Search Results

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

Type	Query Option	Function
	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.

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.

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_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 {*} ]
```

Organize Source Files Dialog Box – Synthesis

The Organize Source Files dialog box enables you to set the source file order in the Libero SoC.

Click the **Use list of files organized by User** radio button to Add/Remove source files for the selected tool.

To specify the file order:

1. In the Design Flow window under Implement Design, right-click **Synthesize** and choose **Organize Input Files > Organize Source Files**. The Organize Source Files dialog box appears.
2. Click the **Use list of files organized by User** radio button to Add/Remove source files for the selected tool.
3. Select a file and click the Add or Remove buttons as necessary. Use the Up and Down arrows to change the order of the Associated Source files.
4. Click **OK**.

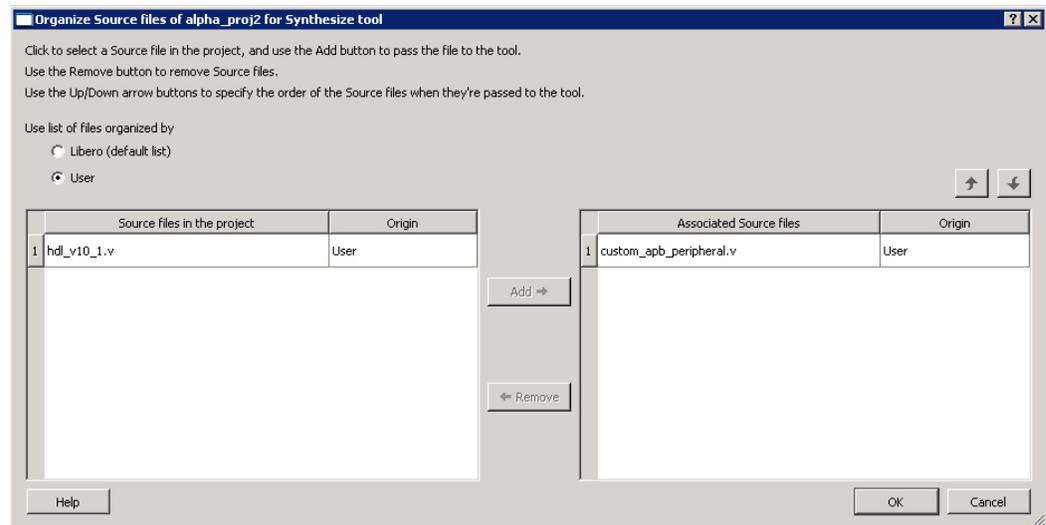


Figure 121 · Organize Source Files Dialog Box

Stimulus Hierarchy

To view the Stimulus Hierarchy, from the **View** menu choose **Windows > Stimulus Hierarchy**.

The Stimulus Hierarchy tab displays a hierarchical representation of the stimulus and simulation files in the project. The software continuously analyzes and updates files and content. The tab (see figure below) displays the structure of the modules and component stimulus files as they relate to each other.

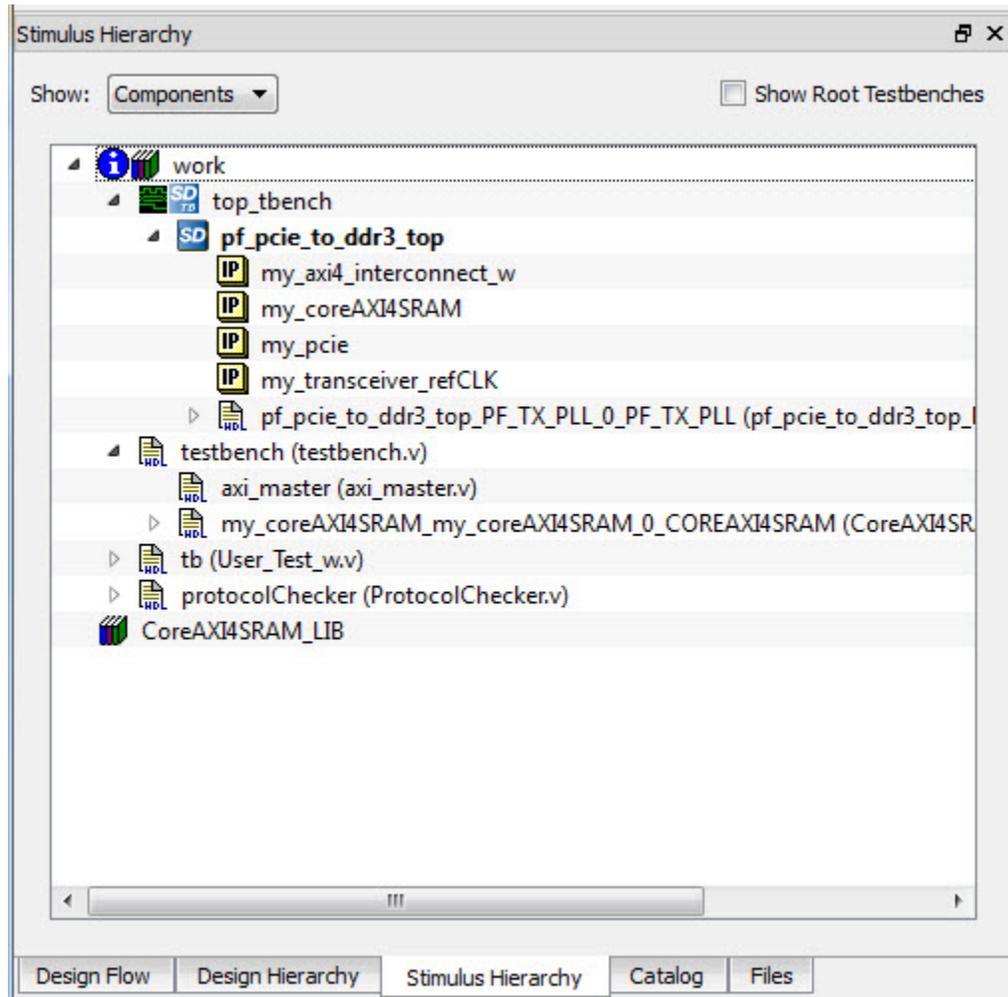


Figure 122 · Stimulus Hierarchy Dialog Box

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 3 · 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**.

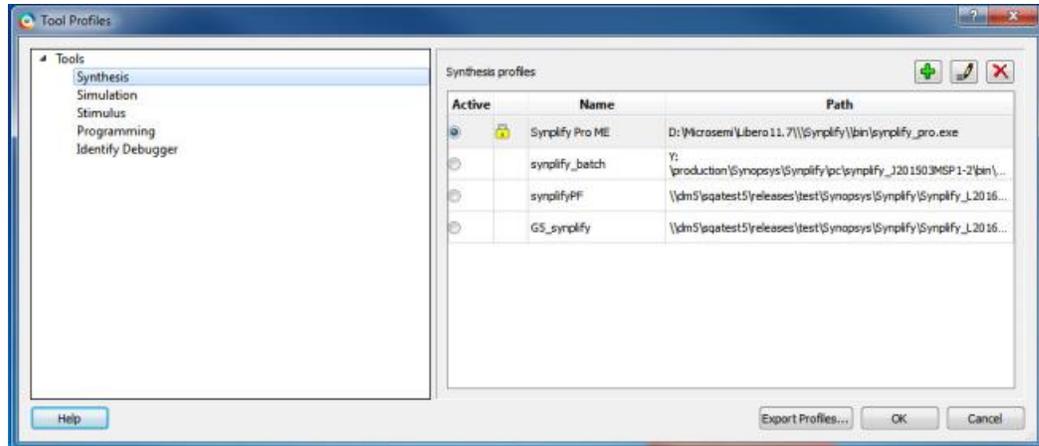


Figure 123 · Libero SoC Tool Profiles Dialog Box

The tool profile with the padlock icon indicates that it is a pre-defined tool profile (the default tool that comes with the Libero SoC Installation.)

To export the tool profile and save it for future use, click the **Export Tool Profiles** dialog box and save the tool profile file as a tool profile *.ini file. The tool profile *.ini file can be imported into a Libero SoC project (**File > Import > Others**) and select Tool Profiles (*.ini) in the File Type pull-down list.

User Preferences Dialog Box – Design Flow Preferences

This dialog box allows you to set your personal preferences for how Libero SoC manages the design flow across the projects you create.

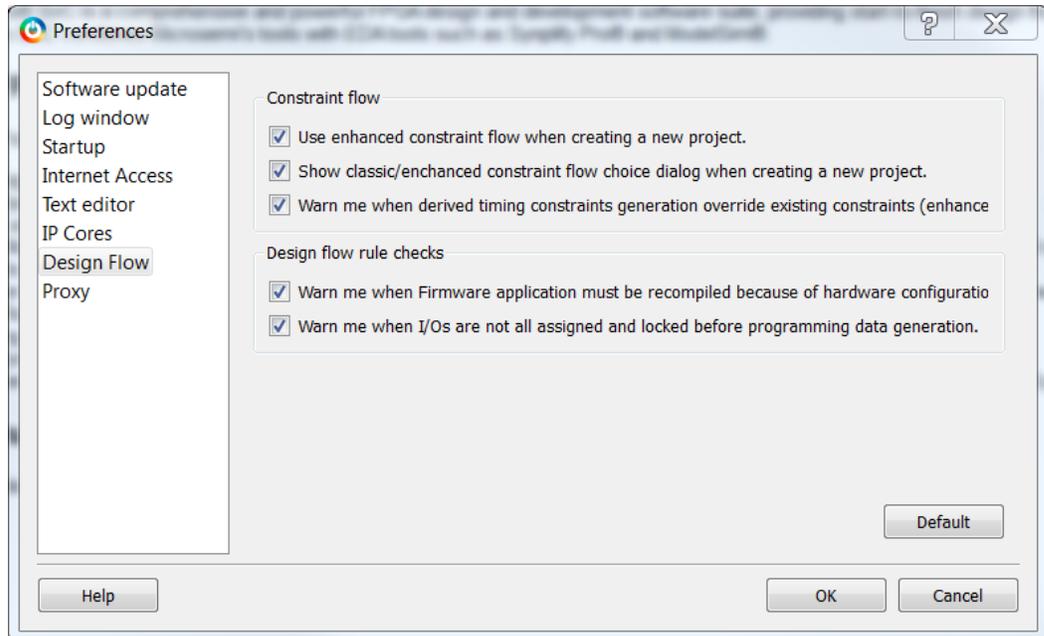


Figure 124 · Preferences Dialog Box – Design Flow Preferences

Constraint Flow

- Use enhanced constraint flow when creating a new project.

The Enhanced Constraint Flow provides a centralized graphical interface for you to better manage all your design constraint files in the design process. The Enhanced Constraint Flow is the only design flow supported for PolarFire, and cannot be turned off.

- **Show classic/enhanced constraint flow choice dialog when creating a new project.**

Since the Enhanced Constraint Flow is the only flow supported for PolarFire, this option has no effect.

- **Warn me when derived timing constraints generation override existing constraints (enhanced constraint flow).**

Libero SoC can generate/derive timing constraints for known hardware blocks and IPs such as SERDES, CCC. Check this box to have Libero SoC pop up a warning message when the generated timing constraints for these blocks override the timing constraints you set for these blocks. This box is checked by default.

Design Flow Rule Checks

- **Warn me when Firmware applications must be recompiled because of hardware configuration changes.**

Check this box if you want Libero SoC to display a warning message. This box is checked by default.

- **Warn me when I/Os are not all assigned and locked before programming data generation.**

I/Os should always be assigned and locked before programming data generation. Check this box if you want Libero SoC to display a warning message. This box is checked by default.

Note: These preferences are stored on a per-user basis across multiple projects; they are not project-specific.

Synopsys Design Constraints (SDC)

Synopsys Design Constraints (SDC) is a Tcl-based format used by Synopsys tools to specify the design intent, including the timing and area constraints for a design. Microsemi tools use a subset of the SDC format to capture supported timing constraints. Any timing constraint that you can enter using Designer tools can also be specified in an SDC file.

Use the SDC-based flow to share timing constraint information between Microsemi tools and third-party EDA tools.

Command	Action
create_clock	Creates a clock and defines its characteristics
create_generated_clock	Creates an internally generated clock and defines its characteristics
remove_clock_uncertainty	Removes a clock-to-clock uncertainty from the current timing scenario.
set_clock_latency	Defines the delay between an external clock source and the definition pin of a clock within SmartTime
set_clock_uncertainty	Defines the timing uncertainty between two clock waveforms or maximum skew

Command	Action
set_false_path	Identifies paths that are to be considered false and excluded from the timing analysis
set_input_delay	Defines the arrival time of an input relative to a clock
set_max_delay	Specifies the maximum delay for the timing paths
set_min_delay	Specifies the minimum delay for the timing paths
set_multicycle_path	Defines a path that takes multiple clock cycles
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 <i>Courier New</i> typeface.
<i>variable</i>	Variables appear in blue, italic <i>Courier New</i> typeface. You must substitute an appropriate value for the variable.
[-argument <i>value</i>]	Optional arguments begin and end with a square bracket.

Note: SDC commands and arguments are case sensitive.

Example

The following example shows syntax for the `create_clock` command and a sample command:

```
create_clock -period period_value [-waveform edge_list] source
create_clock -period 7 -waveform {2 4}{CLK1}
```

Wildcard Characters

You can use the following wildcard characters in names used in the SDC commands:

Wildcard	What it does
\	Interprets the next character literally

Wildcard	What it does
*	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 $(\text{period_value}/2)\text{ns}$.

source

Specifies the source of the clock constraint. The source can be ports or pins in the design. If you specify a clock constraint on a pin that already has a clock, the new clock replaces the existing one. Only one source is accepted. Wildcards are accepted as long as the resolution shows one port or pin.

Description

Creates a clock in the current design at the declared source and defines its period and waveform. The static timing analysis tool uses this information to propagate the waveform across the clock network to the clock pins of all sequential elements driven by this clock source.

The clock information is also used to compute the slacks in the specified clock domain that drive optimization tools such as place-and-route.

Exceptions

- None

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.

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/CLK3_PAD } \
-pll_output { FCCC_0/CCC_INST/GL2 } \
-pll_feedback { FCCC_0/CCC_INST/CLK2 } \
{ FCCC_0/CCC_INST/GL2 }
```

Microsemi Implementation Specifics

- SDC accepts either `-multiply_by` or `-divide_by` option. In Microsemi design implementation, both are accepted to accurately model the PLL behavior.
- SDC accepts defining a generated clock on many sources using a single command. In Microsemi design implementation, only one source is accepted.
- The `-duty_cycle`, `-edges` and `-edge_shift` options in the SDC `create_generated_clock` command are not supported in Microsemi design implementation.

See Also

[SDC Syntax Conventions](#)

set_clock_latency

SDC command; defines the delay between an external clock source and the definition pin of a clock within SmartTime.

```
set_clock_latency -source [-rise][-fall][-early][-late] delay clock
```

Arguments

`-source`

Specifies a clock source latency on a clock pin.

`-rise`

Specifies the edge for which this constraint will apply. If neither or both rise are passed, the same latency is applied to both edges.

-fall

Specifies the edge for which this constraint will apply. If neither or both rise are passed, the same latency is applied to both edges.

-invert

Specifies that the generated clock waveform is inverted with respect to the reference clock.

-late

Optional. Specifies that the latency is late bound on the latency. The appropriate bound is used to provide the most pessimistic timing scenario. However, if the value of "-late" is less than the value of "-early", optimistic timing takes place which could result in incorrect analysis. If neither or both "-early" and "-late" are provided, the same latency is used for both bounds, which results in the latency having no effect for single clock domain setup and hold checks.

-early

Optional. Specifies that the latency is early bound on the latency. The appropriate bound is used to provide the most pessimistic timing scenario. However, if the value of "-late" is less than the value of "-early", optimistic timing takes place which could result in incorrect analysis. If neither or both "-early" and "-late" are provided, the same latency is used for both bounds, which results in the latency having no effect for single clock domain setup and hold checks.

delay

Specifies the latency value for the constraint.

clock

Specifies the clock to which the constraint is applied. This clock must be constrained.

Description

Clock source latency defines the delay between an external clock source and the definition pin of a clock within SmartTime. It behaves much like an input delay constraint. You can specify both an "early" delay and a "late" delay for this latency, providing an uncertainty which SmartTime propagates through its calculations. Rising and falling edges of the same clock can have different latencies. If only one value is provided for the clock source latency, it is taken as the exact latency value, for both rising and falling edges.

Exceptions

None

Examples

The following example sets an early clock source latency of 0.4 on the rising edge of main_clock. It also sets a clock source latency of 1.2, for both the early and late values of the falling edge of main_clock. The late value for the clock source latency for the falling edge of main_clock remains undefined.

```
set_clock_latency -source -rise -early 0.4 { main_clock }
set_clock_latency -source -fall 1.2 { main_clock }
```

Microsemi Implementation Specifics

SDC accepts a list of clocks to -set_clock_latency. In Microsemi design implementation, only one clock pin can have its source latency specified per command.

See Also

[SDC Syntax Conventions](#)

set_clock_to_output

SDC command; defines the timing budget available inside the FPGA for an output relative to a clock.

```
set_clock_to_output delay_value -clock clock_ref [-max] [-min] output_list
```

Arguments

delay_value

Specifies the clock to output delay in nanoseconds. This time represents the amount of time available inside the FPGA between the active clock edge and the data change at the output port.

-clock *clock_ref*

Specifies the reference clock to which the specified clock to output is related. This is a mandatory argument.

-max

Specifies that *delay_value* refers to the maximum clock to output at the specified output. If you do not specify `-max` or `-min` options, the tool assumes maximum and minimum clock to output delays to be equal.

-min

Specifies that *delay_value* refers to the minimum clock to output at the specified output. If you do not specify `-max` or `-min` options, the tool assumes maximum and minimum clock to output delays to be equal.

output_list

Provides a list of output ports in the current design to which *delay_value* is assigned. If you need to specify more than one object, enclose the objects in braces ({}).

set_clock_uncertainty

SDC command; 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.

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.

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 (`{}`).

Description

The `set_external_check` command specifies the external setup and hold times on input ports relative to a clock edge. This usually represents a combinational path delay from the input port to the clock pin of a register internal to the current design. For in/out (bidirectional) ports, you can specify the path delays for both input and output modes. The tool uses external setup and external hold times for paths starting at primary inputs.

A clock is a singleton that represents the name of a defined clock constraint. This can be an object accessor that will refer to one clock. For example:

```
[get_clocks {system_clk}]
[get_clocks {sys*_clk}]
```

Examples

The following example sets an external setup check of 12 ns and an external hold check of 6 ns for port `data_in` relative to the rising edge of `CLK1`:

```
set_external_check 12 -clock [get_clocks CLK1] -setup [get_ports
data_in]
set_external_check 6 -clock [get_clocks CLK1] -hold [get_ports
data_in]
```

See Also

[SDC Syntax Conventions](#)

set_false_path

SDC command; identifies paths that are considered false and excluded from the timing analysis.

```
set_false_path [-from from_list] [-through through_list] [-to to_list]
```

Arguments

-from *from_list*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

-through *through_list*

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

-to *to_list*

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

Description

The `set_false_path` command identifies specific timing paths as being false. The false timing paths are paths that do not propagate logic level changes. This constraint removes timing requirements on these false paths so that they are not considered during the timing analysis. The path starting points are the input ports or register clock pins, and the path ending points are the register data pins or output ports. This constraint disables setup and hold checking for the specified paths.

The false path information always takes precedence over multiple cycle path information and overrides maximum delay constraints. If more than one object is specified within one `-through` option, the path can pass through any objects.

Examples

The following example specifies all paths from clock pins of the registers in clock domain `clk1` to data pins of a specific register in clock domain `clk2` as false paths:

```
set_false_path -from [get_clocks {clk1}] -to reg_2:D
```

The following example specifies all paths through the pin `U0/U1:Y` to be false:

```
set_false_path -through U0/U1:Y
```

Microsemi Implementation Specifics

SDC accepts multiple `-through` options in a single constraint to specify paths that traverse multiple points in the design. In Microsemi design implementation, only one `-through` option is accepted.

See Also

[SDC Syntax Conventions](#)

set_input_delay

SDC command; defines the arrival time of an input relative to a clock.

```
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 ({}).

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_max_delay (SDC)

SDC command; specifies the maximum delay for the timing paths.

```
set_max_delay delay_value [-from from_list] [-to to_list]
```

Arguments

delay_value

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

- If the path starting point is on a sequential device, the tool includes clock skew in the computed delay.
- If the path starting point has an input delay specified, the tool adds that delay value to the path delay.
- If the path ending point is on a sequential device, the tool includes clock skew and library setup time in the computed delay.
- If the ending point has an output delay specified, the tool adds that delay to the path delay.

-from *from_list*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

-to *to_list*

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

Description

This command specifies the required maximum delay for timing paths in the current design. The path length for any startpoint in `from_list` to any endpoint in `to_list` must be less than `delay_value`.

The tool automatically derives the individual maximum delay targets from clock waveforms and port input or output delays. For more information, refer to the [create clock](#), [set input delay](#), and [set output delay](#) commands.

The maximum delay constraint is a timing exception. This constraint overrides the default single cycle timing relationship for one or more timing paths. This constraint also overrides a multicycle path constraint.

Examples

The following example sets a maximum delay by constraining all paths from ff1a:CLK or ff1b:CLK to ff2e:D with a delay less than 5 ns:

```
set_max_delay 5 -from {ff1a:CLK ff1b:CLK} -to {ff2e:D}
```

The following example sets a maximum delay by constraining all paths to output ports whose names start by “out” with a delay less than 3.8 ns:

```
set_max_delay 3.8 -to [get_ports out*]
```

Microsemi Implementation Specifics

The `-through` option in the `set_max_delay` SDC command is not supported.

See Also

[SDC Syntax Conventions](#)

set_min_delay

SDC command; specifies the minimum delay for the timing paths.

```
set_min_delay delay_value [-from from_list] [-to to_list]
```

Arguments

delay_value

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

- If the path starting point is on a sequential device, the tool includes clock skew in the computed delay.
- If the path starting point has an input delay specified, the tool adds that delay value to the path delay.
- If the path ending point is on a sequential device, the tool includes clock skew and library setup time in the computed delay.
- If the ending point has an output delay specified, the tool adds that delay to the path delay.

`-from from_list`

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-to to_list`

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

Description

This command specifies the required minimum delay for timing paths in the current design. The path length for any startpoint in `from_list` to any endpoint in `to_list` must be less than `delay_value`.

The tool automatically derives the individual minimum delay targets from clock waveforms and port input or output delays. For more information, refer to the [create_clock](#), [set_input_delay](#), and [set_output_delay](#) commands.

The minimum delay constraint is a timing exception. This constraint overrides the default single cycle timing relationship for one or more timing paths. This constraint also overrides a multicycle path constraint.

Examples

The following example sets a minimum delay by constraining all paths from ff1a:CLK or ff1b:CLK to ff2e:D with a delay less than 5 ns:

```
set_min_delay 5 -from {ff1a:CLK ff1b:CLK} -to {ff2e:D}
```

The following example sets a minimum delay by constraining all paths to output ports whose names start by "out" with a delay less than 3.8 ns:

```
set_min_delay 3.8 -to [get_ports out*]
```

Microsemi Implementation Specifics

The `-through` option in the `set_min_delay` SDC command is not supported.

See Also

[SDC Syntax Conventions](#)

set_multicycle_path

SDC command; defines a path that takes multiple clock cycles.

```
set_multicycle_path ncycles [-setup] [-hold] [-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.

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 ({}).

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 are SDC commands. Most SDC constraint commands require one of these commands as command arguments.

Microsemi software supports the following SDC access commands:

Design Object	Access Command
Cell	get_cells
Clock	get_clocks
Net	get_nets
Port	get_ports
Pin	get_pins
Input ports	all_inputs
Output ports	all_outputs
Registers	all_registers

See Also

[About SDC Files](#)

all_inputs

[Design object access command](#); returns all the input or inout ports of the design.

```
all_inputs
```

Arguments

- None

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

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.

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.

Description

This command returns a collection of instances matching the pattern you specify. You can only use this command as part of a –from, -to, or –through argument for the following constraint exceptions: set_max_delay, set_multicycle_path, and set_false_path design constraints.

Exceptions

None

Examples

```
set_max_delay 2 -from [get_cells {reg*}] -to [get_ports {out}]
set_false_path -through [get_cells {Rblock/muxA}]
```

Microsemi Implementation Specifics

- None

See Also

[SDC Syntax Conventions](#)

get_clocks

[Design object access command](#); returns the specified clock.

```
get_clocks pattern
```

Arguments

pattern

Specifies the pattern to match to the SmartTime on which a clock constraint has been set.

Description

- If this command is used as a –from argument in maximum delay ([set_max_path_delay](#)), false path ([set_false_path](#)), and multicycle constraints ([set_multicycle_path](#)), the clock pins of all the registers related to this clock are used as path start points.
- If this command is used as a –to argument in maximum delay ([set_max_path_delay](#)), false path ([set_false_path](#)), and multicycle constraints ([set_multicycle_path](#)), the synchronous pins of all the registers related to this clock are used as path endpoints.

Exceptions

- None

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.

Exceptions

None

Example

```
create_clock -period 10 [get_pins clock_gen/reg2:Q]
```

Microsemi Implementation Specifics

- None

See Also[SDC Syntax Conventions](#)**get_nets**

[Design object access command](#); returns the named nets specified by the pattern argument.

```
get_nets pattern
```

Arguments

pattern

Specifies the pattern to match the names of the nets to return. For example, "get_nets N_255*" returns all nets starting with the characters "N_255", where "*" is a wildcard that represents any character string.

Description

This command returns a collection of nets matching the pattern you specify. You can only use this command as source objects in create clock ([create clock](#)) or create generated clock ([create generated clock](#)) constraints and as -through arguments in set false path ([set false path](#)), set minimum delay ([set_min_delay](#)), set maximum delay ([set_max_delay](#)), and set multicycle path ([set multicycle path](#)) constraints.

Exceptions

None

Examples

```
set_max_delay 2 -from [get_ports RDATA1] -through [get_nets
{net_chkp1 net_chkqi}]
set_false_path -through [get_nets {Tblk/rm/n*}]
create_clkok -name mainCLK -per 2.5 [get_nets {cknet}]
```

Microsemi Implementation Specifics

None

See Also[SDC Syntax Conventions](#)**get_ports**[Design object access command](#); returns the specified ports.

```
get_ports pattern
```

Argument*pattern*

Specifies the pattern to match the ports. This is equivalent to the macros \$in()[<pattern>] when used as –from argument and \$out()[<pattern>] when used as –to argument or \$ports()[<pattern>] when used as a –through argument.

Exceptions

None

Example

```
create_clock -period 10[get_ports CK1]
```

Microsemi Implementation Specifics

None

See Also[SDC Syntax Conventions](#)