Libero SoC

Migrating an Existing Project Created with Classic Constraint Flow to Enhanced Constraint Flow

03/2019

a Microchip company

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

512000857-1/03.19

Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

Revision 1.0

Revision 1.0 is the first publication of this document.

Contents

Introduction

Libero SoC v12.0 only supports the Enhanced Constraint Flow for SmartFusion2, IGLOO2 and RTG4 device families. To open a design created with the Classic Constraint Flow in a Libero SoC v11.9 or earlier release, you must manually port your design to the Enhanced Constraint Flow.

This document describes the step by step procedure to migrate a SmartFusion2, IGLOO2 and RTG4 project created with the Classic Constraint Flow to the Enhanced Constraint Flow.

To migrate a design created in the Classic Constraint Flow in a Libero SoC v11.9 or an earlier release to the Enhanced Constraint Flow in Libero SoC v11.9 or v12.0 release, you must follow the below steps:

1. Create a new project and import/upgrade/re-generate the design source files.
2. Import and organize the constraint files (see UG0691: Libero SoC Design Flow User Guide).
3. Configure and run the Synthesis tool.
4. Configure and run the Place and Route tool.
5. Configure and run the Verify Timing tool.

There is no change in the way the remaining tools work; however, you will need to regenerate any reports, bitstreams, etc., as the project is new.

1 Create a New Project and Import/Upgrade/Regenerate the Design Source Files

Using Libero SoC v11.9 or v12.0, create a new project with the same Family, Die, Package, Speed, Die Voltage and Part Range as the Classic Constraint Flow project to be migrated. You must ensure that all the fields in the project settings dialog box are same. If you are using Libero SoC v11.9, ensure that the “Use enhanced constraint flow when creating a new project” option is enabled in the **Project → Preferences → Design Flow dialog box** before creating the project to create the new project in the Enhanced Constraint Flow.

For the remainder of this document, the original (CCF) project will be referred to as “the CCF project”, and the new project will be referred to as “the ECF project”.

1.1 Import Design Source Files

You must import all the design source files to re-create the project.

Note: You cannot import blocks created using a CCF project to the ECF project. If you have a block published in the Classic Constraint Flow, you must recreate the block in an Enhanced Constraint Flow project and republish the block.

1.1.1 Import HDL Source files (.v, .vhd, .vhdl, .sv, .vm, .vh, .svh, .h)

Import/link all the HDL source files/folders in your design. To import the HDL source files/folders to the ECF project, click **File → Import → HDL Sources Files/Folders** and browse to the **<ccf_project>/hdl** folder, select all the hdl files/folders and click **Open**. Similarly, you can use the **File → Link Files** option to link the HDL source files/folders used. When importing VHDL files, you must make sure to create and organize the VHDL libraries.

1.2 Import Stimulus Files (.v, .vhd, .vhdl, .sv, .svh, .vh, .h)

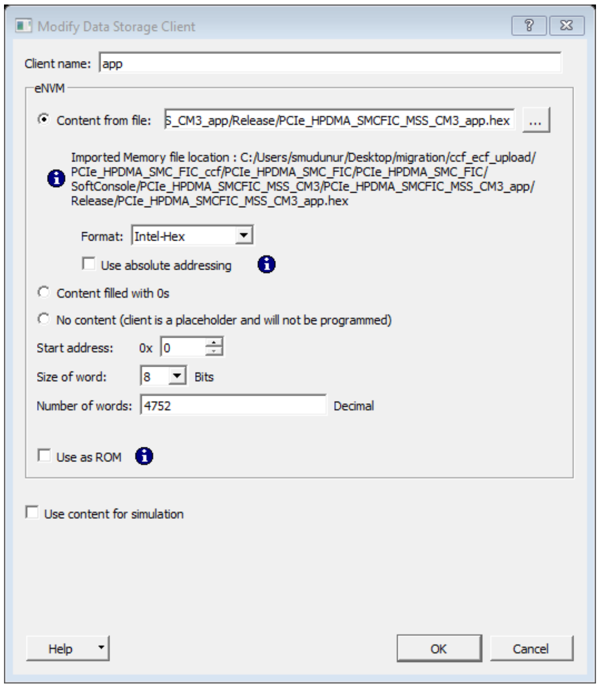
Import all Stimulus files in your design. To import the HDL Stimulus files to the ECF project, click **File → Import → HDL Stimulus Files** and browse to the **<ccf_project>/stimulus** folder, select all the stimulus files and click **Open**.

1.2.1 Import Components (.cxf files)

Note: You must import all Component files from the CCF project to the ECF project.

To import a Component to the ECF project, you must import its corresponding .cxf file from the CCF project. Components include SmartDesign, SystemBuilder, MSS and Configured cores. To import a .cxf file, click **File → Import → Components** and browse to the **<ccf_project>/Component/work** folder, select the .cxf file and click **Open**. Repeat this step for each Component in the CCF project.

If your design uses an on-FPGA nonvolatile memory (eNVM or uPROM), you may need to adjust the path to the memory files associated with various clients. To adjust the path to the memory files, open the eNVM/uPROM configurator, select the Data Storage Client and choose **Modify Client** from the right-click menu to open the Modify Data Storage Client dialog box. Adjust the path of the memory file by selecting the Browse button next to the **Content from file** option, as shown in the following figure.



Modify Data Storage Client

Client name: app

eNVM

(●) Content from file: 5_CM3_app/Release/PCIe_HPDMA_SMCFIC_MSS_CM3_app.hex

Imported Memory file location : C:/Users/smudunur/Desktop/migration/ccf_ecf_upload/PCIe_HPDMA_SMC_FIC_ccf/PCIe_HPDMA_SMC_FIC/PCIe_HPDMA_SMC_FIC/SoftConsole/PCIe_HPDMA_SMCFIC_MSS_CM3/PCIe_HPDMA_SMCFIC_MSS_CM3_app/Release/PCIe_HPDMA_SMCFIC_MSS_CM3_app.hex

Format: Intel-Hex

☐ Use absolute addressing

( ) Content filled with 0s

( ) No content (client is a placeholder and will not be programmed)

Start address: 0x 0

Size of word: 8 Bits

Number of words: 4752 Decimal

☐ Use as ROM

☐ Use content for simulation

Figure 1. Modify Data Storage Client Dialog Box

Note: If the CCF project uses HDL+ cores (cores created from HDL, shown as HDL+ in the Design Hierarchy) instantiated in SmartDesign Components, you must import the HDL+ cores to successfully re-create the SmartDesign Components in the ECF project. To import the HDL+ cores to the ECF project, close the project and copy the **User** folder from **<ccf_project>/Component/** to **<ecf_project>/Component/.** Reopen the ECF project to see the HDL+ cores.

1.2.2 Upgrade Cores to the Latest Versions

After importing cores, you can upgrade them to the latest versions as available. To upgrade cores to the latest versions, follow these steps:

1. Download the latest version of the core into your vault.
2. Upgrade each configured core in your design to the latest version by right-clicking the core Component in the design hierarchy and choosing '**Replace Component Version…**'.
3. Regenerate the design.
4. Rerun the Derive Constraints step.
5. Rerun the tool flow.

Note: To upgrade the cores that are directly instantiated in SmartDesign components, right-click the core instance in the SmartDesign component and choose '**Replace Version**'.

1.2.3 Regenerate all Components

You must regenerate all Components after importing the Component files and upgrading the cores.

If you have System Builder Components in your design, you must open the System Builder Component's GUI (by double-clicking on the Component in the Design Hierarchy), go through all the pages, review the page configurations, including the file paths to eNVM/uPROM clients in the Memories page, and click **Finish** to regenerate the System Builder Component.

During regeneration, the Components containing CCC, SERDES, FDDR, OSC and MSS cores will generate Component level SDC files which will be used later to generate the derived constraints in the Enhanced Constraint Flow.

Note: If you are migrating to Libero SoC v12.0, you must build the design hierarchy after importing all HDL sources and importing and generating all Components. To do so, click ⚠ Build Hierarchy in the Design Hierarchy tab and set the top module as the root.

2 Import and Organize Constraint files

The Constraint Manager in Enhanced Constraint Flow is used to create, import, link, check, delete, edit design constraints and associate the constraint files to design implementation tools. The Constraint Manager allows you to organize constraints for SynplifyPro Synthesis, Libero SoC Place-and-Route and SmartTime Timing Analysis. For timing constraints, this strategy replaces the timing analysis 'scenario' feature available in SmartTime in the Classic Constraint Flow. For all constraints, the Constraints Manager replaces the "Associate for Compile/Place and Route" options in the Classic Constraint Flow.

2.1 I/O and Floorplanning Constraints

2.1.1 Importing PDC Files

Import/link the existing I/O and Floorplan PDC files containing I/O and Floorplan assignments and attributes using the corresponding tabs in the Constraint Manager window. You must associate the PDC files with the Place and Route tool by enabling the corresponding checkbox available for each PDC file imported/linked.

You may need to modify PDC files as follows:

- If you had constraints for I/O register combining options in the original CCF project's PDC files, you must remove them, create a Compile Netlist constraint file (NDC) and specify the I/O register combining constraints using the new set_ioff command in the NDC file.
- set_preserve and set_mitigation PDC commands are now NDC commands in the Enhanced Constraint Flow and must be removed from the I/O PDC file before importing the I/O PDC file. Create an Compile Netlist Constraint (NDC) file and add the set_preserve and set_mitigation commands there.
- If you had reserved pins for device migration using the Interactive I/O Editor in the CCF project, you must re-specify these constraints under the I/O Settings group available in the Constraint Manager I/O Attributes tab.
- If you had reserved pins for probes or programming set in **Project → Project Settings → Device Settings** configuration page in the CCF project, you must re-specify these constraints under the I/O Settings group available in the Constraint Manager I/O Attributes tab.

The following table shows how the constraints set in I/O PDC files in the CCF project must be modified in the ECF project.

Constraints in I/O PDC File in Classic Constraint Flow	**Constraints in Enhanced Constraint Flow**	
	I/O PDC file	**NDC file**
set_io CLK0_PAD \ -pinname 137 \ -fixed yes \ -LPE Wake_On_Change \ -DIRECTION INPUT	set_io CLK0_PAD \ -pinname 137 \ -fixed yes \ -LPE Wake_On_Change \ -DIRECTION INPUT	
set_io D \ -REGISTER yes \ -IN_DELAY 0	set_io D \ -IN_DELAY 0	set_ioff {D} \ -in_reg yes \ -out_reg no \ -en_reg no

set_io CLK0_PAD_0 \ -pinname 23 \ -fixed yes \ -FF_IO_STATE LAST_VALUE \ -IN_DELAY 0 \ -LPE Wake_On_Change \ -RES_PULL Up \ -SCHMITT_TRIGGER On \ -DIRECTION INPUT	set_io CLK0_PAD_0 \ -pinname 23 \ -fixed yes \ -FF_IO_STATE LAST_VALUE \ -IN_DELAY 0 \ -LPE Wake_On_Change \ -RES_PULL Up \ -SCHMITT_TRIGGER On \ -DIRECTION INPUT	
set_preserve RCLKINT_0		set_preserve RCLKINT_0
set_mitigation \ -inst_name CLKINT_0 \ -mitigated yes		set_mitigation \ -inst_name CLKINT_0 \ -mitigated yes

2.2 Timing Constraints

2.2.1 Derived Constraints

Similar to constraints that were automatically generated for Components such as CCC, MSS, Chip Oscillators, and IP Cores such as CoreResetP in the Classic Constraint Flow, Libero has generalized the concept of Component/core generated constraints with the new ‘Derived Constraints’ feature available from the Constraint Manager’s Timing tab in the Enhanced Constraint Flow.

After executing Section 2, execute the ‘Derive Constraints’ action. This instantiates Component-level timing constraints (for Components that represent silicon features) into the design. It generates a *<root module name>_derived_constraints.sdc* file that can be managed with the rest of the user SDC files to provide a complete set of SDC constraints for the design.

2.2.2 User Constraints

In the ECF project, import all SDC files from the **<ccf_project>/constraint** folder by clicking the **Import** button in the Constraint Manager Timing tab.

If you had timing constraints set in interactive mode in the SmartTime Constraints Editor in the CCF project, they are not captured in the user SDC files. Export the SDC file **(File → Export → Timing Constraints (SDC))** to capture such constraints from the previous Libero SoC release supporting Classic Constraint Flow and import that SDC file into the ECF project.

Each user defined SDC constraint must be modified as follows:

- If a constraint in a user SDC file is included in the Derived Constraints file, it must be removed from the user SDC file.
- A significant change from the Classic Constraint Flow to the Enhanced Constraint Flow is that user SDC constraints must be defined at the RTL level. This means that any reference to a design object (net, pin, instance, or port) must use pre-synthesis names and hierarchies. This is a change compared to the Classic Constraint Flow, where SDC constraints used post-layout names and hierarchies. As a result, all constraints referring to post-layout object names must be changed as follows:
    - The pin separator is now ‘/’ (it was ‘:’ in the Classic Constraint Flow).
    - Instance names at hierarchy levels below the RTL instance names must be removed.

The following table shows examples of constraints set in the Classic Constraint Flow and their equivalents in the Enhanced Constraint Flow (modified).

Timing Constraint in Classic Constraint Flow	Timing Constraint in Enhanced Constraint Flow
create_clock -name { PCIe_HPDMA_SMCFIC_0.CCC_0.GL0_net } -period 13.330 -waveform { 0.000 6.660 } [get_pins {PCIe_HPDMA_SMCFIC_0/CCC_0/CCC_INST/INST_CCC_IP:GL0 }]	create_clock -name { PCIe_HPDMA_SMCFIC_0.CCC_0.GL0_net } -period 13.330 -waveform { 0.000 6.660 } [get_pins { PCIe_HPDMA_SMCFIC_0/CCC_0/CCC_INST/GL0 }]
create_clock -name {PCIe_HPDMA_SMCFIC_CCC_0_FCCC|GL3_net_inferred_clock } -period 10.000 -waveform { 0.000 5.000 } [get_pins { PCIe_HPDMA_SMCFIC_0/CCC_0/CCC_INST/INST_CCC_IP:GL3 }]	create_clock -name { PCIe_HPDMA_SMCFIC_CCC_0_FCCC|GL3_net_inferred_clock } -period 10.000 -waveform { 0.000 5.000 } [get_pins { PCIe_HPDMA_SMCFIC_0/CCC_0/CCC_INST/GL3 }]
create_clock -name { FIC_2_APB_M_PCLK } -period 53.333 -waveform { 0.000 26.666 } [get_pins { PCIe_HPDMA_SMCFIC_0/PCIe_HPDMA_SMCFIC_MSS_0/MSS_ADLIB_INST /INST_MSS_120_IP:CLK_CONFIG_APB}]	create_clock -name { FIC_2_APB_M_PCLK } -period 53.333 -waveform { 0.000 26.666 } [get_pins { PCIe_HPDMA_SMCFIC_0/PCIe_HPDMA_SMCFIC_MSS_0/MSS_ADLIB_INST/CLK_CONFIG_APB }]

2.2.3 Organizing Constraints

As you migrate your design to the Enhanced Constraint Flow, make sure to properly organize your SDC constraints to match your needs. Use the Constraints Manager UI to decide which constraints files to apply to Synthesis, Place and Route, and Timing Verification, respectively.

2.3 Netlist Attributes Constraints

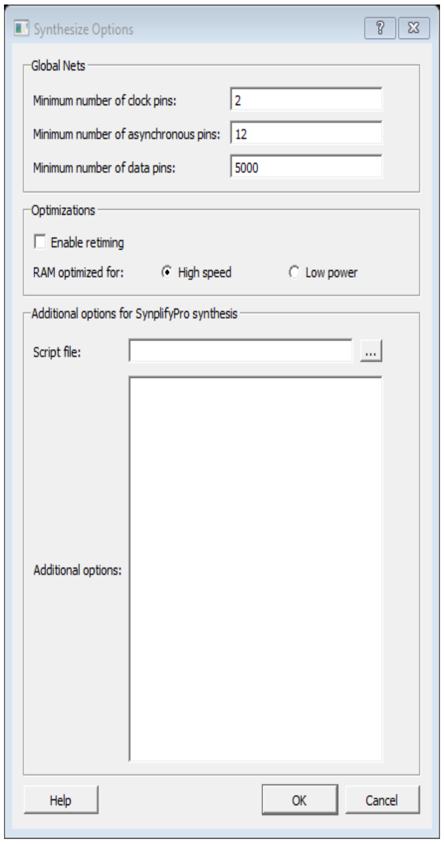
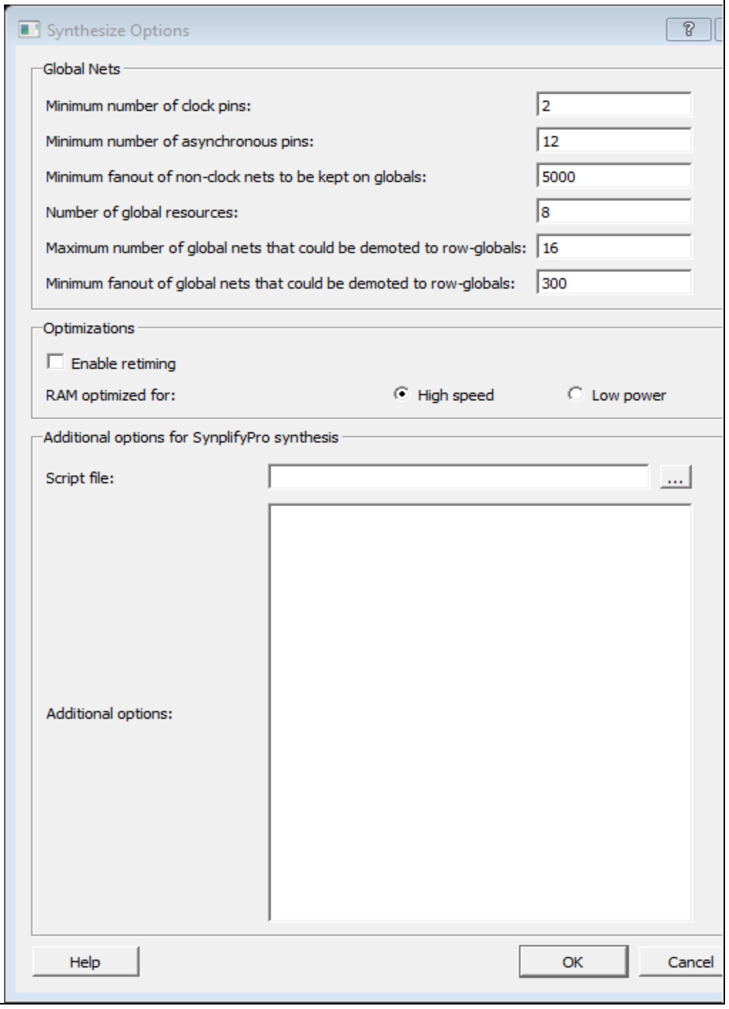
2.3.1 Synthesis Netlist Attributes

Synthesis netlist attributes are typically used to set Synthesis optimization directives for design objects (instances, nets, pins, and ports) at the RTL level. The Enhanced Constraint Flow allows users to manage Synopsys FDC constraints within the Libero environment. If there were any FDC constraints for the original design, you may now create/import/link FDC files in the Constraint Manager's Netlist Attributes tab. Note that FDC files should not contain timing constraints; see section 2.2 Timing Constraints to manage timing constraints for Synthesis.

3 Configure and Run the Synthesis Tool

In the Enhanced Constraint Flow, the synthesis and post synthesis compile netlist steps have been combined to simplify the flow. You must configure the Synthesis step in the Enhanced Constraint Flow by combining both the Synthesis and Compile configuration options used in the Classic Constraint Flow.

Note: Some Compile options (such as Enable Design Separation Methodology, Abort Compile if errors are found in the physical design constraints, Limit the number of displayed high fanout nets in compile report to option) set in the Compile Options dialog box (**Design flow → Compile → Configure Options**) in the Classic Constraint Flow have been moved to the **Project → Project Settings → Design Flow** configuration page in the Enhanced Constraint Flow.

Classic Constraint Flow	Enhanced Constraint Flow
	

Classic Constraint Flow – Synthesize Options

Global Nets
Minimum number of clock pins: 2
Minimum number of asynchronous pins: 12
Minimum number of data pins: 5000

Optimizations
☐ Enable retiming
RAM optimized for: [✓ High speed] Low power

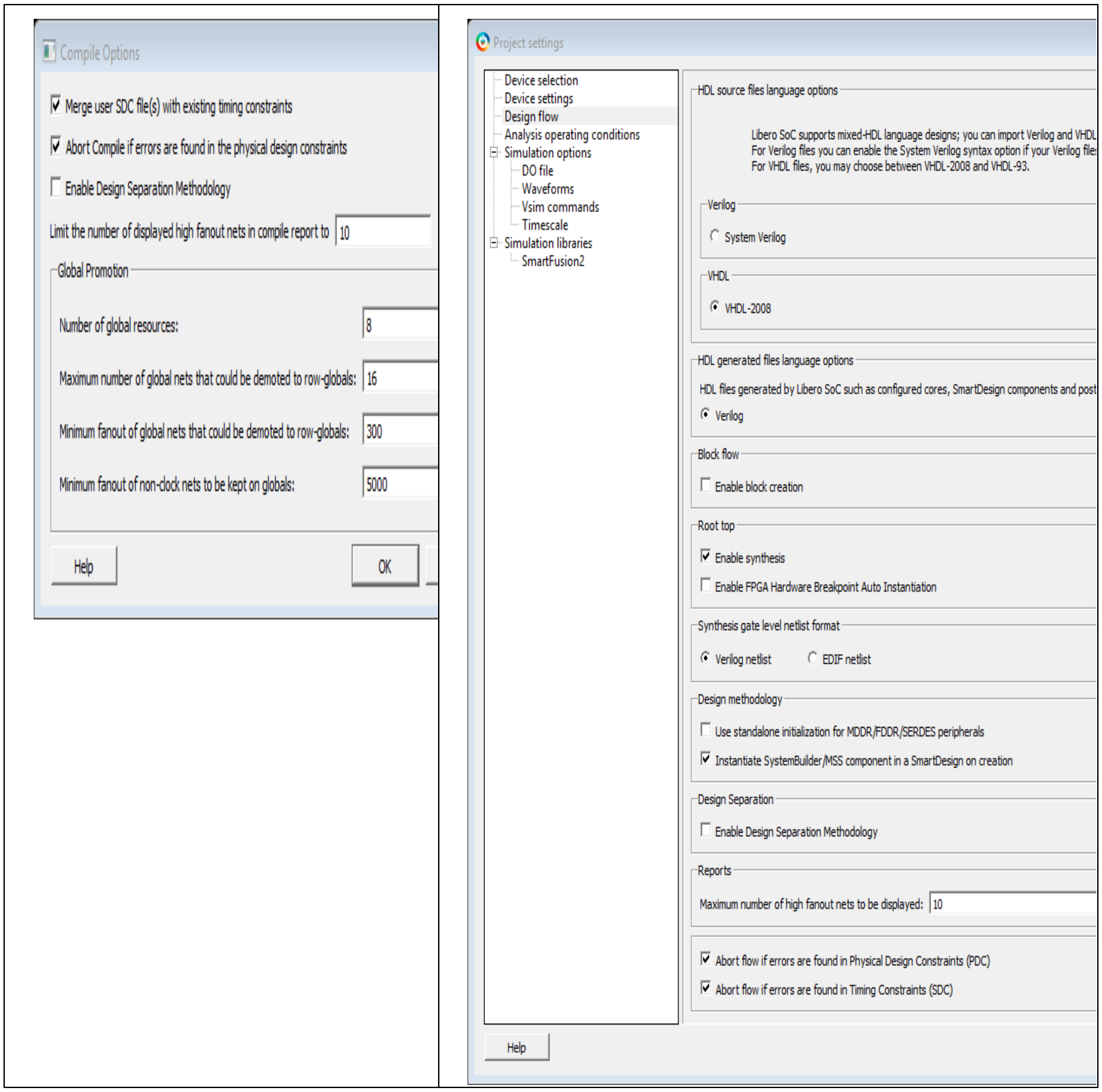
Additional options for SynplifyPro synthesis
Script file:
Additional options:

Enhanced Constraint Flow – Synthesize Options

Global Nets
Minimum number of clock pins: 2
Minimum number of asynchronous pins: 12
Minimum fanout of non-clock nets to be kept on globals: 5000
Number of global resources: 8
Maximum number of global nets that could be demoted to row-globals: 16
Minimum fanout of global nets that could be demoted to row-globals: 300

Optimizations
☐ Enable retiming
RAM optimized for: [✓ High speed] Low power

Additional options for SynplifyPro synthesis
Script file:
Additional options:



Compile Options

[✓ Merge user SDC file(s) with existing timing constraints]

[✓ Abort Compile if errors are found in the physical design constraints]

Enable Design Separation Methodology

Limit the number of displayed high fanout nets in compile report to: 10

Global Promotion

Number of global resources: 8

Maximum number of global nets that could be demoted to row-globals: 16

Minimum fanout of global nets that could be demoted to row-globals: 300

Minimum fanout of non-clock nets to be kept on globals: 5000

Project settings

HDL source files language options

Libero SoC supports mixed-HDL language designs; you can import Verilog and VHDL
For Verilog files you can enable the System Verilog syntax option if your Verilog file
For VHDL files, you may choose between VHDL-2008 and VHDL-93.

Verilog

System Verilog

VHDL

[✓ VHDL-2008]

HDL generated files language options

HDL files generated by Libero SoC such as configured cores, SmartDesign components and post

[✓ Verilog]

Block flow

Enable block creation

Root top

[✓ Enable synthesis]

Enable FPGA Hardware Breakpoint Auto Instantiation

Synthesis gate level netlist format

[✓ Verilog netlist] EDIF netlist

Design methodology

Use standalone initialization for MDDR/FDDR/SERDES peripherals

[✓ Instantiate SystemBuilder/MSS component in a SmartDesign on creation]

Design Separation

Enable Design Separation Methodology

Reports

Maximum number of high fanout nets to be displayed: 10

[✓ Abort flow if errors are found in Physical Design Constraints (PDC)]

[✓ Abort flow if errors are found in Timing Constraints (SDC)]

Figure 2: Comparison of Synthesis and Compile Options in both the Classic Constraint Flow and the Enhanced Constraint Flow

Running Synthesis in the Enhanced Constraint Flow invokes Synopsys' Synplify Pro to generate the synthesized netlist, and then runs the post synthesis Compile Netlist step. This post synthesis compile netlist step does the following:

- Validates the post synthesis netlist.
- Generates various reports including an accurate resource report. Note that a hierarchical resource report is available in the Enhanced Constraint Flow, to assess resource count at any level in the design hierarchy.

Note: After Synthesis, verify the compile report to ensure that the resource utilization is preserved in the Enhanced Constraint Flow. A significantly different resource utilization may indicate missing or incorrect migration steps.

4 Configure and Run the Place and Route Tool

You can configure the Place and Route tool in the Enhanced Constraint Flow using the same configuration options as in the Classic Constraint Flow. The Place and Route tool use the constraints files associated with the Place and Route tool set in the different tabs in the Constraint Manager window.

5 Configure and Run the Verify Timing Tool

You can configure the Verify Timing tool in the Enhanced Constraint Flow using the same configuration options as in the Classic Constraint Flow. The Verify Timing tool and the SmartTime GUI use the timing constraints files associated with the Timing Verification tool in the Constraint Manager window.

6 Appendix: Migrating an Example Project from Libero SoC v11.9 to Libero SoC v12.0

The following steps for migration also apply for CCF projects created with Libero SoC v11.8 as well as with Libero SoC v11.9 and their service pack (SP) releases.

6.1 Prerequisites

Before migration, do the following:

1. Download the original project from https://coredocs.s3.amazonaws.com/Libero/12_0_0/support_files/IGLOO2_Oversampling_119_ccf.7z and save as (for example) "IGLOO2_Oversampling_119_ccf".
2. Install Libero SoC v11.9 SP2 software.
3. Install Libero SoC v12.0 software.

6.2 Example Project Migration

To migrate the IGLOO2 High Speed SERDES design created using the Classic Constraint Flow in Libero SoC v11.9 to the Enhanced Constraint Flow in Libero SoC v12.0, do the following:

Step 1: Create a new project and import/upgrade/re-generate the design source files

1. Create a new project with the following device settings:
   a. Project Name: IGLOO2_Oversampling_120_ecf
   b. Family: IGLOO2
   c. Die: M2GL010T
   d. Package: 484 FBGA
   e. Speed: -1
   f. Part Range: COM
   g. Voltage: 2.5
2. Import the HDL source files/folders to the ECF project:
   a. Click **File** → **Import** → **HDL Sources Files/Folders** and browse to the **IGLOO2_Oversampling_119_ccf/hdl** folder of the CCF project
   b. Select all the hdl files and click **Open**.
3. Import the Component files:
   a. Click **File** → **Import** → **Components** and browse to the **IGLOO2_Oversampling_119_ccf/Component/work** folder
   b. Select the .cxf file located in each Component folder and click **Open.**
4. For this example, ensure that you import the following .cxf files:
   a. IGLOO2_Oversampling.cxf
   b. IGLOO2_Oversampling_top.cxf
   c. Receiver.cxf
   d. Transmitter.cxf
   e. UART_INTERFACE.cxf

   Note: When you import a System Builder component, the underlying MSS/HPMS component need not be imported. In this example IGLOO2_Oversampling_HPMS.cxf need not be imported.

5. Open the System Builder Component (IGLOO2_Oversampling in the Design Hierarchy), click through all the pages and click **Finish** at the end to regenerate the System Builder Component.
6. Open and regenerate the Transmitter SmartDesign Component.
7. Open and regenerate the Receiver SmartDesign Component.
8. Open and regenerate the UART_INTERFACE SmartDesign Component.
9. Open and regenerate the IGLOO2_Oversampling_top SmartDesign Component.
10. Build the Design Hierarchy and set the SmartDesign Component **IGLOO2_Oversampling_top** as the root module: In the Design Hierarchy tab, right-click "**IGLOO2_Oversampling_top**" and click"**Set as root"**.

Step 2: Import and organize the constraint files

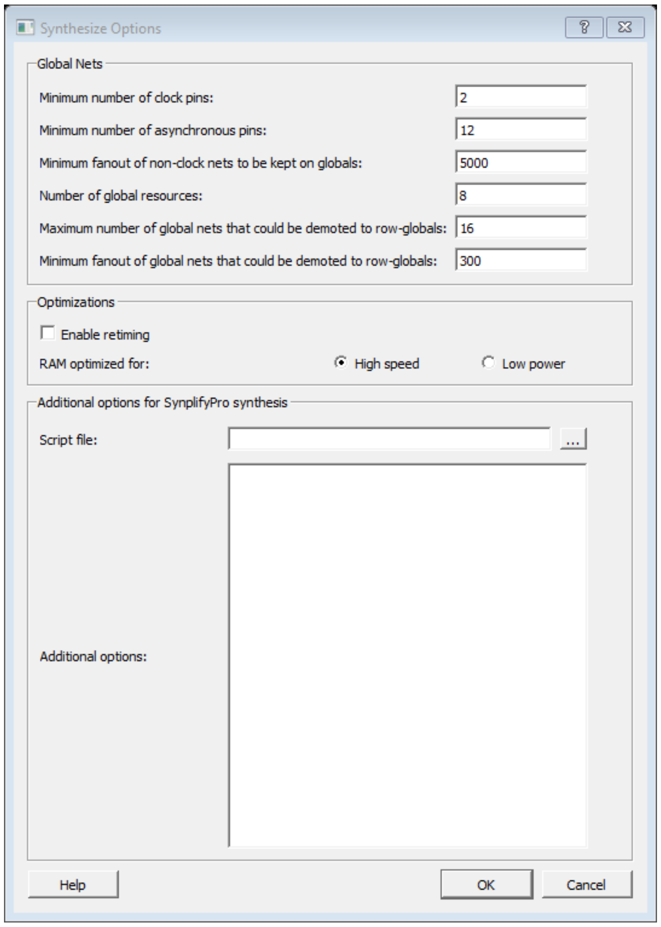
1. Import the I/O constraint file (io.pdc) by navigating to **IGLOO2_Oversampling_119_ccf/constraint/io** folder of the CCF project by selecting **File → Import →I/O Constraint (PDC) Files**.
2. The timing constraints set in the SmartTime tool will not appear in the user sdc file in the Classic Constraint Flow. To capture these timing constraints, open the CCF project using Libero SoC v11.9 and export the sdc file as export_sdc.sdc by selecting **File→Export→SDC**. Then import the export_sdc.sdc file to the ECF project by selecting **File→Import→Timing Constraint (SDC) Files**.
3. Open the Constraint Manager's Timing tab and click **Derive Constraints** to generate the timing derived constraints.
4. Open the two sdc files present in the Constraint Manager's Timing tab and remove the constraints which are already present in IGLOO2_Oversampling_top_derived_constraints.sdc from the export_sdc.sdc file.
5. Change the remaining constraints in export_sdc.sdc as follows:

From	To
set_clock_groups -name internal_groupname_1 -asynchronous -group {IGLOO2_Oversampling_top_FCCC_0_FCCC|GL0_net_inferred_clock}	set_clock_groups -name internal_groupname_1 -asynchronous -group {FCCC_0/GL0}
set_clock_groups -name internal_groupname_2 -asynchronous -group {IGLOO2_Oversampling_top_FCCC_0_FCCC|GL1_net_inferred_clock}	set_clock_groups -name internal_groupname_2 -asynchronous -group {FCCC_0/GL1}
set_clock_groups -name internal_groupname_3 -asynchronous -group {IGLOO2_Oversampling_CCC_0_FCCC|GL0_net_inferred_clock}	set_clock_groups -name internal_groupname_3 -asynchronous -group {IGLOO2_Oversampling_0/CCC_0/GL0}
set_clock_groups -name internal_groupname_4 -asynchronous -group {IGLOO2_Oversampling_HPMS|FIC_2_APB_M_PCLK_inferred_clock}	set_clock_groups -name internal_groupname_4 -asynchronous -group {IGLOO2_Oversampling_0/IGLOO2_Oversampling_HPMS_0/CLK_CONFIG_APB}
set_clock_groups -name internal_groupname_5 -asynchronous -group {IGLOO2_Oversampling_top_SERDES_IF_0_SERDES_IF|EPCS_1_TX_CLK_inferred_clock}	set_clock_groups -name internal_groupname_5 -asynchronous -group {SERDES_IF_0/SERDESIF_INST/EPCS_TXCLK_1 }
set_clock_groups -name internal_groupname_6 -asynchronous -group {IGLOO2_Oversampling_top_SERDES_IF_0_SERDES_IF|EPCS_1_RX_CLK_inferred_clock}	set_clock_groups -name internal_groupname_6 -asynchronous -group {SERDES_IF_0/SERDESIF_INST/EPCS_RXCLK_1 }

6. To associate the timing constraints files with Synthesis, Place and Route and Timing Verification tools, check all the boxes for the two sdc files in the Timing tab of the Constraint Manager.
7. To associate the IO constraint file with the Place and Route tool, check the box next to the pdc file in the IO Attributes tab of the Constraint Manager.

Step 3: Configure and run the Synthesis tool

Configure the Synthesis tool with the same options set in the Synthesis and Compile tool in the Classic Constraint Flow (as shown below) and run Synthesis.


Synthesize Options

Global Nets

Minimum number of clock pins: 2
Minimum number of asynchronous pins: 12
Minimum fanout of non-clock nets to be kept on globals: 5000
Number of global resources: 8
Maximum number of global nets that could be demoted to row-globals: 16
Minimum fanout of global nets that could be demoted to row-globals: 300

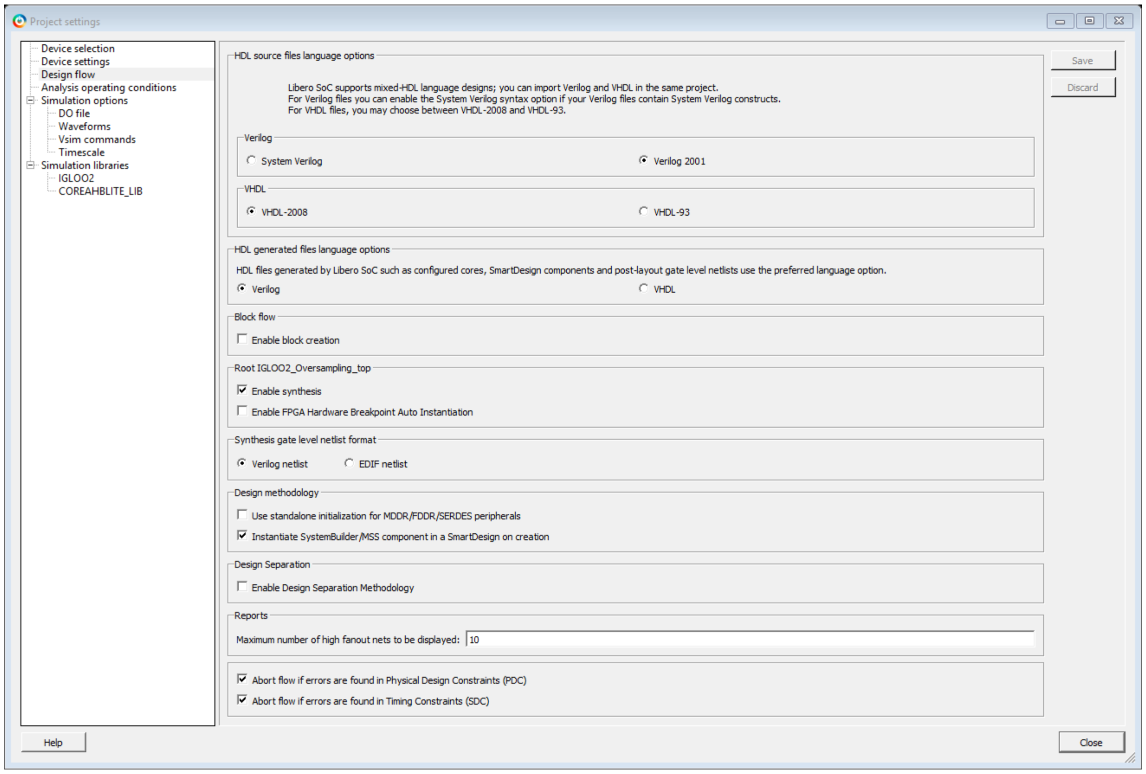
Optimizations

☐ Enable retiming
RAM optimized for: [✓ High speed] Low power

Additional options for SynplifyPro synthesis

Script file:
Additional options:

Figure 3: Synthesize Options Dialog Box



HDL source files language options

Libero SoC supports mixed-HDL language designs; you can import Verilog and VHDL in the same project.
For Verilog files you can enable the System Verilog syntax option if your Verilog files contain System Verilog constructs.
For VHDL files, you may choose between VHDL-2008 and VHDL-93.

Verilog: System Verilog [✓ Verilog 2001]

VHDL: [✓ VHDL-2008] VHDL-93

HDL generated files language options

HDL files generated by Libero SoC such as configured cores, SmartDesign components and post-layout gate level netlists use the preferred language option.

[✓ Verilog] VHDL

Block flow

- [ ] Enable block creation

Root IGLOO2_Oversampling_top

- [x] Enable synthesis
- [ ] Enable FPGA Hardware Breakpoint Auto Instantiation

Synthesis gate level netlist format

[✓ Verilog netlist] EDIF netlist

Design methodology

- [ ] Use standalone initialization for MDDR/FDDR/SERDES peripherals
- [x] Instantiate SystemBuilder/MSS component in a SmartDesign on creation

Design Separation

- [ ] Enable Design Separation Methodology

Reports

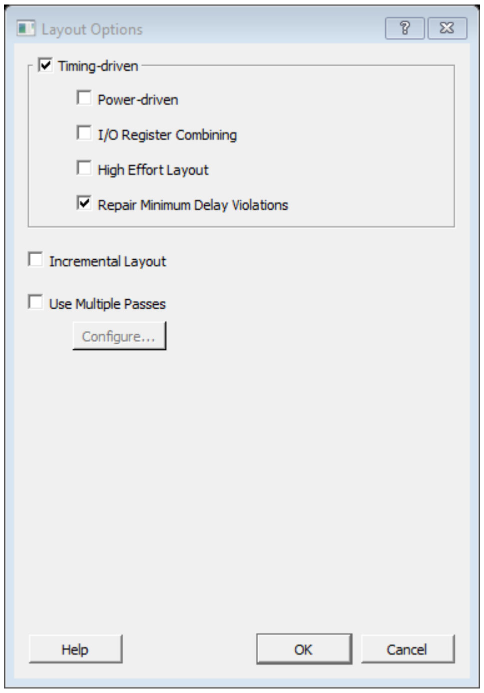
Maximum number of high fanout nets to be displayed: 10

- [x] Abort flow if errors are found in Physical Design Constraints (PDC)
- [x] Abort flow if errors are found in Timing Constraints (SDC)

Figure 4: Project Settings-Design Flow Dialog Box

Step 4: Configure and run the Place and route tool

Configure the Place and Route tool as shown below and run the tool.

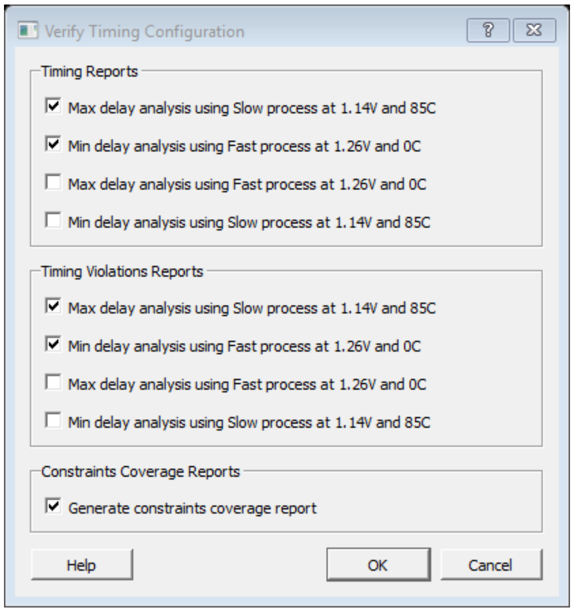


- [x] Timing-driven
  - [ ] Power-driven
  - [ ] I/O Register Combining
  - [ ] High Effort Layout
  - [x] Repair Minimum Delay Violations
- [ ] Incremental Layout
- [ ] Use Multiple Passes

Figure 5: Layout Options Dialog Box

Step 5: Configure and run the Verify Timing tool

Configure the Verify Timing tool as shown below and run the tool.



Timing Reports

- [✓] Max delay analysis using Slow process at 1.14V and 85C
- [✓] Min delay analysis using Fast process at 1.26V and 0C
- [ ] Max delay analysis using Fast process at 1.26V and 0C
- [ ] Min delay analysis using Slow process at 1.14V and 85C

Timing Violations Reports

- [✓] Max delay analysis using Slow process at 1.14V and 85C
- [✓] Min delay analysis using Fast process at 1.26V and 0C
- [ ] Max delay analysis using Fast process at 1.26V and 0C
- [ ] Min delay analysis using Slow process at 1.14V and 85C

Constraints Coverage Reports

- [✓] Generate constraints coverage report

Figure 6: Verify Timing Configuration Options Dialog Box

For reference, the post-migration design can be accessed here:

https://coredocs.s3.amazonaws.com/Libero/12_0_0/support_files/IGLOO2_Oversampling_120_ecf.7z