SmartFusion2, IGLOO2, and RTG4 Block Designing with Blocks for Libero SoC v11.8 in the Classic Constraint 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
Power Matters.™

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

About Microsemi

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

5-02-00526-3/01.17

Table of Contents

SmartFusion2, IGLOO2, and RTG4 Block Flow

Block Flow is a bottom-up design methodology that enables you to use design blocks (“components” in generic terms) as building blocks for your top-level design. These building blocks may have already completed layout and been optimized for timing and power performance for a specific Microsemi device. Using these blocks as part of your top-level design can cut down design time as well as improve timing and power performance. Block advantages include:

- Focus on the timing of critical blocks and ensure the timing across the blocks meets requirements before proceeding to integrate your blocks at the top level.
- Changes in other blocks have no impact on your own block; you can re-use your block without re-optimizing for timing closure.
- The block can be re-used in multiple designs.
- Shorter verification time; you need to re-verify only the portion of the design that has changed.

Block Features

- A Block can be synthesized, simulated, and placed-and-routed the same way as a regular design.
- You can lock the placement and routing of the Block to ensure repeatable performance.
- Performance, placement and routing can be fixed absolutely; however these rules can be relaxed gradually, if necessary, to ensure that you can integrate the Block into your top level project.

Use blocks when:

- You have multiple team members working on different parts of the same design.
- The design is congested (uses 90% or more of the resources on a given die).
- You have difficulty meeting timing by doing the design in its entirety. Blocks enable you to compartmentalize the design and optimize sections before you optimize the entire design.
- You want to re-use some elements of your design.
- You want to use the identical elements multiple times in a single design.
- You want to make small changes in your design and expect to keep most of the design unchanged with guaranteed performance.

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

HDL Supported

- Verilog
- VHDL

Synthesis Tools Supported

- Synplify Pro

Nested Blocks

Nested blocks (blocks instantiated inside other blocks) are supported. When publishing, only one file will be published that contains all the required information (including the nested block).

Creating Blocks - Options and Settings

To enable Block Creation for a new project, from the **Project** menu, choose **New Project**. Check the **Enable Block Creation** checkbox.

In an existing project, from the **Project** menu, choose **Project Settings**. Click **Design Flow** and check the **Enable Block Creation** checkbox.

Synthesis Tool Settings

In Synplify Pro, the I/O Insertion option is disabled when the block is synthesized. Libero automatically disables I/O insertion for you before invoking Synplify Pro.

Compile

During Compile, Libero SoC software adds BLOCK_INTERFACE_I* instances to the block. These instances are virtual buffers added to:

- Improve timing values for the block.
- Provide you with a clear interface to floorplan
- Help with clustering constraints

The BLOCK_INTERFACE_I* instances are removed when the block is published.

Publish Options/Settings

Use the Publish Block – Configuration Options dialog box to configure the block for Publication.

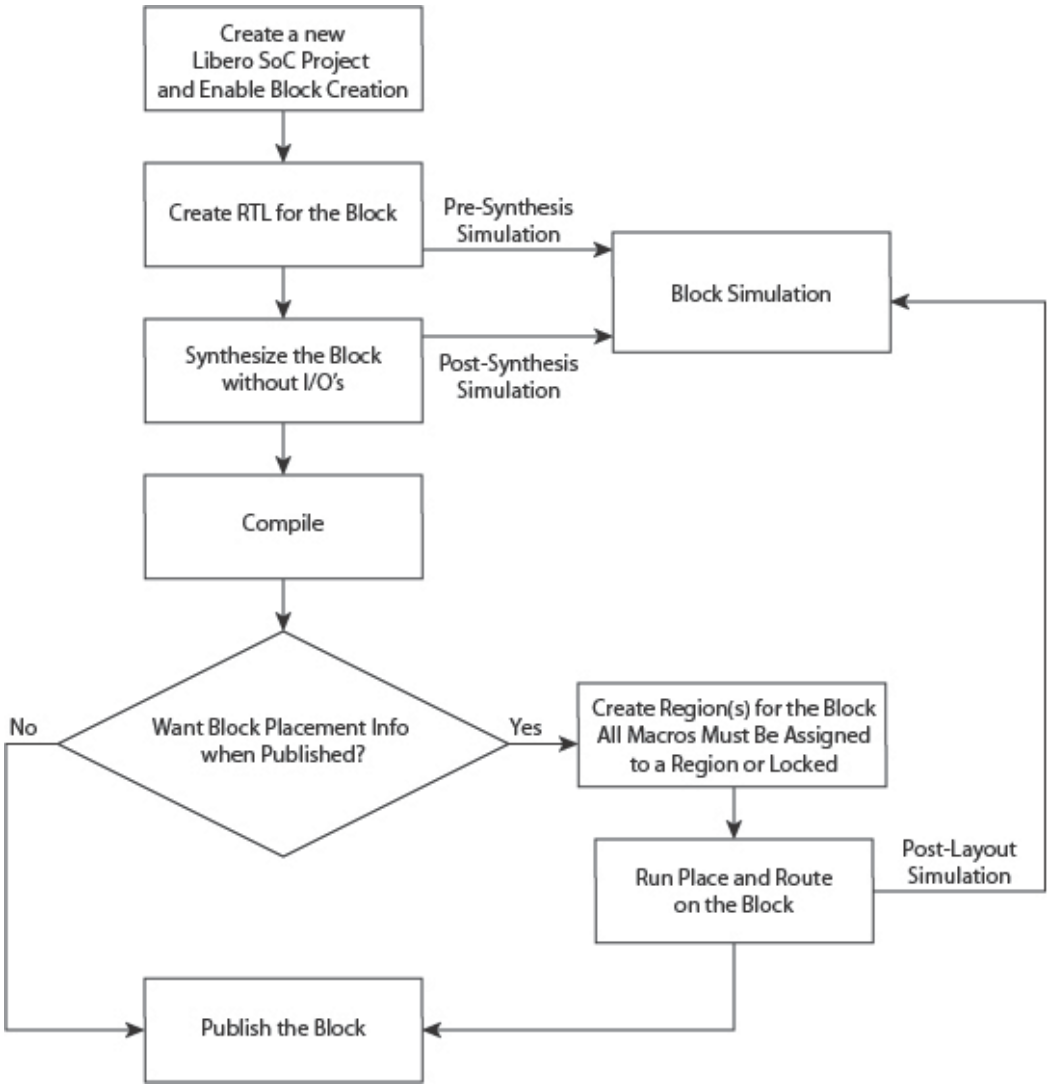

Figure 1 · Creating a Block for SmartFusion2 or IGLOO2

Publishing Blocks After Compile or Layout

You can publish your block after Compile or Layout.

Publish After Compile

If you publish a block after Compile but before Layout, a netlist is exported for the block when published. No Place and Route information or Region Constraint information is included in the block when published. A Warning message appears when you publish a block before Place and Route.

Publish After Layout

If you publish a block after Layout, the Placement, Routing and/or Region Constraint information will be published along with the netlist. You can always open the configurator and change the options to publish what you want. All macros must be locked or assigned to regions in order to publish the Placement information.

Published Content

Libero exports the <design>.cxz file to <project folder>/designer/<design_block_name>/export folder when a block is published. The <design>.cxz file is a zip file that contains the following files:

- <design_block_name>_syn.v | <design_block_name>_syn.vhd -- A timing shell file passed to synthesis tools when the top-level design is synthesized. The block is marked and treated as a black box when the top-level design is synthesized.
- <design_block_name>_sim.v | <design_block_name>_sim.vhd -- A structural HDL netlist for postsynthesis simulation of the block.
- header_report.log - A log file that contains Header Information on what and how a block is published , including the options you selected to configure the publication.
- <design_block_name>_compile.xml -- Compile Report detailing resource usage, device info, and a list of high-fanout nets.
- <design_block_name>_gp_report.xml -- Global Placement and Routing Report
- <design_block_name>_combinational_loops.xml -- Combinational Loops Report<design>.cdb – Internal proprietary file containing the optimized netlist , placement, routing or timing constraint information

The <design_block_name>.cxz file is your published block. You can move it to another folder, transfer it to other team members, etc. This is the file you import into your top-level design when you want to instantiate the block.

Guidelines for Creating Blocks

Macros/IPs Not Supported in Blocks

When creating a Block for instantiation in a top-level design, please note that the following types of macros are not allowed: MSS, SERDESIF, FDDR, FLASH_FREEZE, SYSRESET, UJTAG.

Synthesis Tool and Globals Management

The synthesis tool may promote all clocks to globals. Keep in mind the number of globals you need in your top level design and the number of globals allowed in the device (8 or 16 depending on device size) you are targeting. You may need to reduce or limit the number of globals in your block by adding row globals or plan to share globals in the top-level design with the block. To add a row global, you can add it directly to your HDL (RCLKINT).

Blocks and DRC

Regular DRC rules are applied to blocks as in the regular Libero design flow. For example, some DRC rules assume that some pins must be connected to the power nets. These rules are enforced on the blocks in the block flow just as in the regular design flow.

Blocks and Floorplanning

When creating a block, floorplanning is essential if you plan to publish placement information. Before running Layout on the block, you must floorplan the design block. You can use either Libero's MultiView Navigator (MVN) or PDC commands for floorplanning.

If you do not create a region and constrain the Block to the region (floorplanning) or lock the macros before place and route, a Warning message appears when you publish the Block. It warns you that not all macros in the Block have been constrained to regions or locked and therefore only your design netlist is exported when the Block is published.

Floorplanning with MVN

1. Create a new region. In the Design Flow tab, right-click **Floorplan Constraints** and choose **Open Interactively** to open ChipPlanner. From the **Region** menu choose **Create Inclusive/Exclusive**.
2. Assign and constrain all your block macros to this region.
    - Click and drag to create a region.
    - Select all instances except the BLOCK_INTERFACE_I* instances added by Libero.
    - Right-click and choose **Assign all Instances**.
    - From the **File** menu choose **Commit and Check**.
3. Create a separate region; it must be outside the region created in the above steps. Assign and constrain all the BLOCK_INTERFACE_I* instances (added by Libero) to this separate region. Use the prefix BLOCK_INTERFACE_I* to search for all the Block Interface instances.
   Do not use exclusive regions for the Block Interface Instances; they may limit the usage of the clusters in the top design.
4. Ensure that the new regions you created do not overlap.
5. From the **File** menu choose **Commit and Check**.

Floorplanning with PDC Commands

You can use the define_region PDC command to create a rectangular or rectilinear region, and then use the assign_region PDC command to constrain all the macros to that region. Refer to the Libero Help for the command syntax.

Floorplanning reduces the risk of placement conflicts of the blocks at the top level.

If you do not constrain your Block placement, its components may be placed anywhere on the die.

It is also important to consider the placement of all Block Interface Instances at the boundaries of Block regions. This facilitates the interconnection of the Block to the top-level design. If the Block is highly optimized (densely packed) there may be no routing channels available to connect to any internal Block Interface Instances. Placing all interfaces at Block boundaries helps you eliminate routing congestion and failure.

Block Features in MVN

Block Ports Tab - Lists all the ports in a Design Block even if they are not connected to I/Os.

Interface Instances in Design Block (Active Lists,Tools > Active List > Interface Instances) - Lists all macros connected to ports. These macros must be placed on the boundary of your Block region, since they will be connected to the <top> design.

Block content available when you instantiate a Block in the <top> design (Tools > Active List > Designer Block Content) - Lists all macros in the Block.

Block Tab - Lists all the blocks in your design.

Search Support - Enables you to find a specific block in your design.

Architecture Limitations - Managing Blocks and Globals

Architecturally, the silicon has 8 or 16 globals per device, depending on the device size. If you create a block for use in a top level design and you know that the top level design will use close to the maximum number of Globals for the device, it is good practice to minimize the number of Globals when you create the block. Examine the Global Report to see how many Globals have been used for the block. To reduce the number of Globals used in the block, you may consider clock sharing and the use of Row Globals for the block.

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

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

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

Instantiating Blocks in your Top-Level Design

You may instantiate multiple instances of the same block or multiple blocks in the top-level design.

Microsemi recommends that you create a new project for your top-level design. To do so:

1. From the **Project** menu choose **New Project**.
2. Deselect the **Enable Designer Block Creation** checkbox.
3. Choose the **Family/Die/Package** for the new project for the top-level as follows :

- If the block is a Netlist only and was not published with place and route information, choose the same **Family** as the block for the new project. Choose any Die and Package.
- If the block contains placement information, choose the same **Family** and **Die** as the block for the new project, and choose any Package.
- If the Netlist contains I/O and Placement Information, choose the same **Family**, **Die** and **Package** as the block for the new project.

Import the Block

1. From the **File** menu choose **Import > Blocks**.
2. Browse to the directory that contains your <design_block_name>.cxz file and select it.
3. Click **Open**.

<design_block_name> is imported into the top_level project. Version control is not supported for imported blocks. If you import the same block twice, the existing block is overwritten by the new one.

The files will be imported under <design>\component\work\<design_block_name>.

Review the files in the above directory to view Block Reports.

Create a Top Level Design that Uses Blocks

Use SmartDesign or HDL to create your top level design. If you use HDL you can create HDL for the top level or import a top-level HDL file.

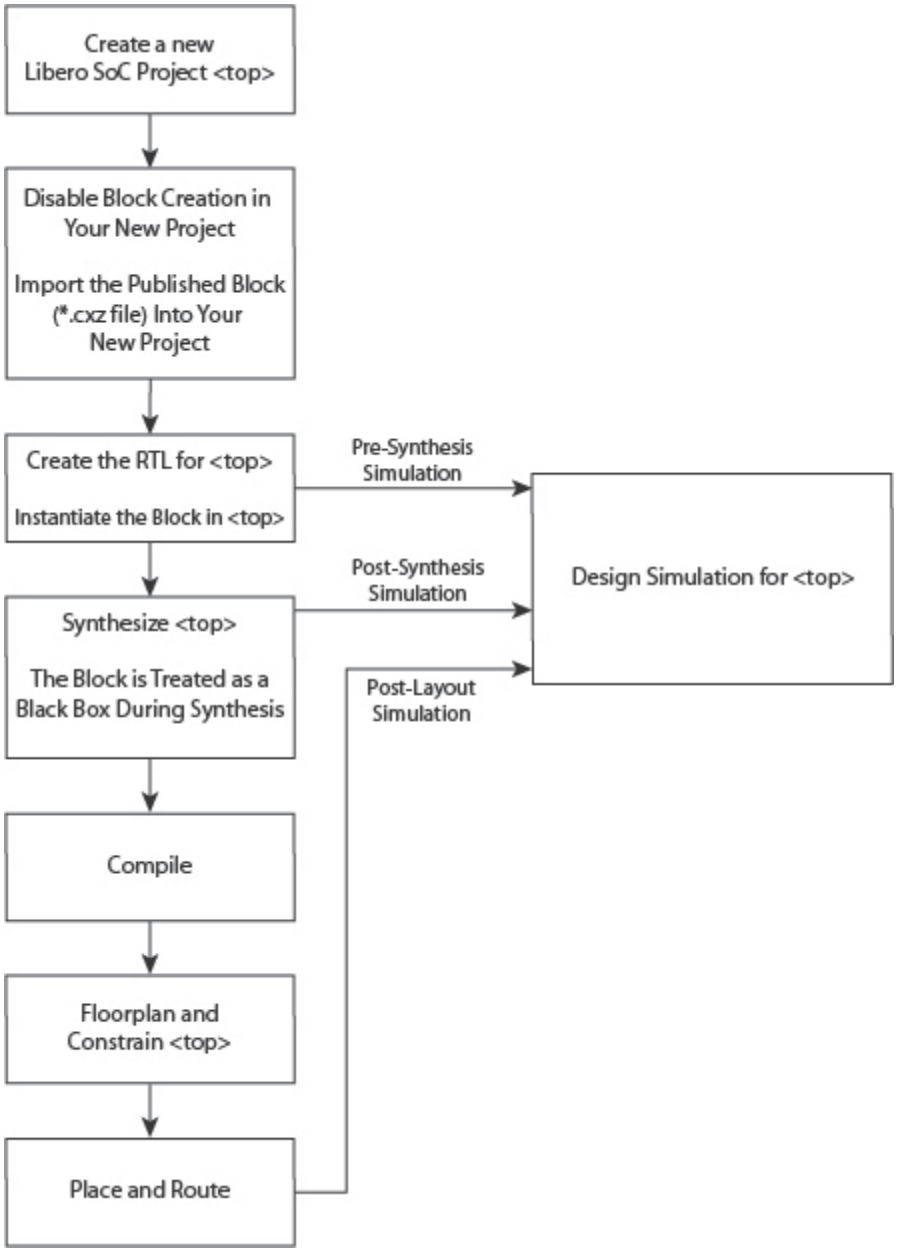

Figure 2 · Instantiating Blocks in your Top Level Design

Hierarchical Structure Resolution in Top Level Projects

If you import multiple conflicting definitions for your *.v files, Libero resolves the conflicts as shown below.

Duplicate Block Definition

If you import two versions of your block file you must choose which one you want to use. For example:

1. Import top.v and block1.v files as HDL (**File > Import HDL Source Files**) into the top level project.
2. Import <block1> (**File > Import > Blocks**).

Libero recognizes a duplicate definition of <block1>; one from the HDL and another in the imported block file. The Design Hierarchy tab shows a <block1>.cxf and <block1>.v file under Duplicate Modules; Libero uses the HDL <block1> by default.

To override the default behavior and select the Block definition, right-click the <block1>.cxf file and choose **Use This File**. When you update the behavior the Block icon appears in the Design Hierarchy.

Conflicting Definitions in top.v and Your Imported Block File

You can introduce a conflict if you import a top.v file and a block file. Libero does not support HDL definition of low level blocks inside top level HDL files and subsequent importing of block files. For example, the following will cause an error:

1. Import a top.v file (**File > Import HDL Source Files**) that contains a definition for <top> and a module definition for <block1>.
2. Import the block <block1> (**File > Import > Blocks**).

Libero passes two duplicate files to your synthesis tool because the definition for <block1> is duplicated. To continue, you must remove the definition of <block1> from top.v and then re-import it.

Resolving top.v and Block Instantiations

Libero integrates your top.v file and block file if there is no definition for the block file in top.v. For example:

1. Import your top.v (**File > Import HDL Sources Files**) that contains instantiations but no definition of <block1>.
2. Import <block1> (**File > Import> Blocks**).

Libero resolves the hierarchy for you and puts <block1> under top.v.

EDIF Netlist in the Top Level Design

If the Top Level design is in EDIF, you must convert the EDIF to HDL and then import the HDL into Libero. To convert the Top Level EDIF to HDL:

1. Write a Tcl script. For example:

```
set_device -fam SmartFusion2
read_edif -file {E:\top.edn}
write_verilog -file {E:\top.v} –skip_empty_modules 1
write_vhdl -file {E:\top.vhd}
## -skip_empty_modules 1 is to instruct the tool not to insert module ## definition for
the empty modules in the HDL created.
```

2. From the Windows Command Prompt or the Linux shell, run rwnetlist as follows (this executable is located in the same location as Libero):

```
rwnetlist --script "E:/run_export_netlist.tcl”
```

Synthesis

Libero passes the block timing to your synthesis tool when the top level is synthesized. This timing shell enables the synthesis tool to produce more accurate timing numbers for top level synthesis.

The timing shell also instructs the synthesis tool to treat the design block as a black box; this is done automatically - no action is required.

Resolving Place and Route Conflicts

To resolve Place and Route conflicts at the top-level:

- Examine the Block Compile Report. Identify the cause of the problem and manually place and constrain the placement with MultiView Navigator or with PDC commands.

- If you instantiate a block (published with placement) multiple times then placement between multiple block instances will overlap. To remove overlapping, move the block placement of one or more instances to another area using the PDC command move_block.
- The Compiler enforces Global sharing. If there is a Global driving a CLKINT in the block it will be deleted. Reduce the number of Globals at the top level by sharing Global Clock resources. Globals in the Blocks may also be re-routed (not preserved).

Resolving Place and Route Conflicts in Compile

See the Compile Options for an explanation of the Block conflict resolution options in Compile.

Block PDC Commands

Use the move_block and set_block_options commands to make changes in your Top-Level design. See the respective help topics for more information.

move_block

PDC command; moves a design block from its original, locked placement by preserving the relative placement between the instances. You can move the Block to the left, right, up, or down.

Note: If possible, routing is preserved when you move the blocks for IGLOO, SmartFusion, Fusion and ProASCI3 families.

```
move_block -inst_name instance_name -up y -down y  -left x -right x  -non_logic value
```

Arguments

`-inst_name instance_name`

Specifies the name of the instance to move. If you do not know the name of the instance, run a Compile report or look at the names shown in the Block tab of the Chip Planner.

`-up y`

Moves the block up the specified number of rows. The value must be a positive integer.

`-down y`

Moves the block down the specified number of rows. The value must be a positive integer.

`-left x`

Moves the block left the specified number of columns. The value must be a positive integer.

`-right x`

Moves the block right the specified number of columns. The value must be a positive integer.

`-non_logic value`

Specifies what to do with the non-logic part of the block, if one exists. The following table shows the acceptable values for this argument:

Value	Description
move	Move the entire block.
keep	Move only the logic portion of the block (COMB/SEQ) and keep the rest locked in the same previous location, if there is no conflict with other blocks.
unplace	Move only the logic portion of the block (COMB/SEQ) and unplace the rest of it, such as I/Os and RAM.

Supported Families

SmartFusion2, IGLOO2, RTG4, SmartFusion, IGLOO, ProASIC3, Fusion

Description

This command moves a block from its original, locked position to a new position.

You can move the entire block or just the logic part of it. You must use the -non_logic argument to specify what to do with the non-logic part of the block. You can find placement information about the block in the Block report. From the **Tools** menu in the designer software, choose **Reports > Block > Interface** to display the report that shows the location of the blocks.

The -up, -down, -left, and -right arguments enable you to specify how to move the block from its original placement. You cannot rotate the block, but the relative placement of macros within the block will be preserved and the placement will be locked. However, routing will be lost. You can either use the ChipPlanner tool or run a Block report to determine the location of the block.

The -non_logic argument enables you to move a block that includes non-logic instances, such as RAM or I/Os that are difficult to move. Once you have moved a part of a block, you can unplace the remaining parts of the block and then place them manually as necessary.

Note: Note: Microsemi recommends that you move the block left or right by increments of 12. If not, placement may fail because it violates clustering constraints. Also, Microsemi recommends that you move the block up or down by increments of three.

Exceptions

- You must import this PDC command as a source file, not as an auxiliary file.
- You must use this PDC command if you want to preserve the relative placement and routing (if possible) of a block you are instantiating many times in your design. Only one instance will be preserved by default. To preserve other instances, you must move them using this command.

Examples

The following example moves the entire block (instance name instA) 12 columns to the right and 3 rows up::

```
move_block -inst_name instA -right 12 -up 3 -non_logic move
```

The following example moves only the logic portion of the block and unplaces the rest by 24 columns to the right and 6 rows up.

```
move_block -inst_name instA –right 24 –up 6 –non_logic unplace
```

See Also

set_block_options

PDC Reference

set_block_options

PDC command; overrides the compile option for placement or routing conflicts for an instance of a block.

```
set_block_options -inst_name instance_name -placement_conflicts value -routing_conflicts value
```

Arguments

`-inst_name instance_name`

Specifies the block instance name. If you do not know the name of the instance, run a Block Report (**Design > Reports > Blocks > Interface**) or look at the names shown in the Block tab of the Chip Planner.

`-placement_conflicts value.`

Specifies what to do when the software encounters a placement conflict. The following table shows the acceptable values for this argument:

Value	Description
error	Compile errors out if any instance from a Designer block becomes unplaced or its routing is deleted. This is the default compile option.
resolve	If some instances get unplaced for any reason, the non-conflicting elements remaining are also unplaced. Basically, if there are any conflicts, nothing from the block is kept.
keep	If some instances get unplaced for any reason, the non-conflicting elements remaining are preserved but not locked. Therefore, the placer can move them into another location if necessary.
lock	If some instances get unplaced for any reason, the non-conflicting elements remaining are preserved and locked.
discard	Discards any placement from the block, even if there are no conflicts.

`-routing_conflicts value`

Specifies what to do when the software encounters a routing conflict. The following table shows the acceptable values for this argument:

Value	Description
error	Compile errors out if any route in any preserved net from a Designer block is deleted.
resolve	If a route is removed from a net for any reason, the routing for the non-conflicting nets is also deleted. Basically, if there are any conflicts, no routes from the block are kept.
keep	If a route is removed from a net for any reason, the routing for the non-conflicting nets is kept unlocked. Therefore, the router can re-route these nets.
lock	If routing is removed from a net for any reason, the routing for the non-conflicting nets is kept as locked, and the router will not change them. This is the default compile option.

Value	Description
discard	Discards any routing from the block, even if there are no conflicts.

Supported Families

SmartFusion2, IGLOO2, RTG4, SmartFusion, IGLOO, ProASIC3, Fusion

Description

This command enables you to override the compile option for placement or routing conflicts for an instance of a block.

Exceptions

You must import this PDC command as a source file, not as an auxiliary file.

If placement is discarded, the routing is automatically discarded too.

Examples

This example makes the designer software display an error if any instance from a block becomes unplaced or the routing is deleted:

```
set_block_options -inst_name instA -placement_conflicts ERROR -routing_conflicts ERROR
```

See Also

move_block

PDC Reference

Publish Block - Configuration Options

To view this dialog box you must first Enable Block Creation in the Libero SoC Project Settings or New Project Creation Wizard. After Block Creation is enabled Publish Block appears in the Design Flow window. Expand **Publish Design**, right-click **Publish Block** and choose **Configure Options**.

Configuration

Publish Placement- Check this box to publish the placement information for the Block. Note that you must assign all macros to regions or lock them in order to Publish Placement.

If checked, the published Block can only be instantiated and used in a top level design with the same family and device. If the Block contains I/Os, the published Block can only be instantiated and used in a top level design with the same family, device and package.

If unchecked, only a netlist is published for the block. The published block can be instantiated and used in a top level design for any device and package in the same device family as the block.

Publish Routing - Check this box to retain the routing information with the block when published.

Publish Region - Check this box to retain the region constraint information with the block when published.

Language

Select your Block Hardware Description Language. The default is the Preferred HDL type set in your Project Settings.

Compile - SmartFusion2, IGLOO2, and RTG4

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

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

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

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

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

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

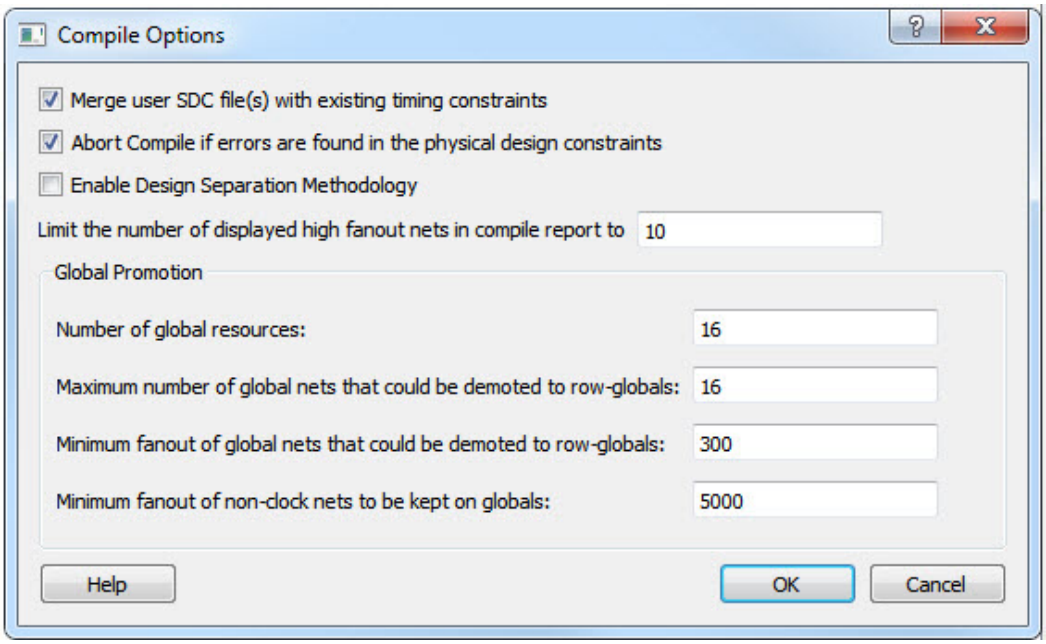

Figure 3 · Compile Options Dialog Box

Configure Options

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

Merge User SDC file(s) with Existing Timing Constraints

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

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

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

Abort Compile if errors are found in the physical design constraints

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

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

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

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

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

Enable Design Separation Methodology (SmartFusion2 and IGLOO2 only**) Checkbox** – Check this box if your design is for security and safety critical applications and you want to make your design's individual subsystems (design blocks) separate and independent (in terms of physical layout and programming) to meet your design separation requirements. When checked, Libero generates a parameter file (MSVT.param) that details design blocks present in the design and the number of signals entering and leaving a design block. Microsemi provides a separate tool, known as Microsemi Separation Verification Tool (MSVT), which checks the final design place and route result against the MSVT.param file and determines whether the design separation meets your requirements.

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

Global Promotion/Demotion Options

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

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

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

Minimum fanout of global nets that could be demoted to row-globals – Specifies the minimum fanout of global nets that could be demoted to row-global. The default is 300. If you run out of global routing resources for your design, reduce this number (to allow more globals to be demoted) or select a bigger die for your design.

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

Block Instantiation Compile Options

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

Placement

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

Resolve conflict

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

Routing

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

Resolve conflict

- **Keep non-conflicting routing**- If a nets' routing is removed for any reason, the routing for the non-conflicting nets is preserved but not locked (so that they can be rerouted). This is the default option.
- **Keep and lock non-conflicting routing**- If the routing is removed for any reason, the remaining non-conflicting nets are preserved and locked; they cannot be rerouted. This is the default option.
- **Discard routing from all blocks** – Routing information will be discarded from all blocks even if there is no conflict.