Libero SoC Block Flow User Guide

Libero® SoC v2024.2

Introduction (Ask a Question)

PolarFire® FPGA Block Flow is a bottom-up design methodology that enables you to use design blocks (components) as building blocks for your top-level designs. These building blocks might have already completed layout, and been optimized for timing and power performance for a specific Microchip device. Using these blocks in the top-level designs can reduce design time and improve timing and power performance.

Block offers unique advantages that allow you to:

- Focus on the timing of primary blocks and ensure that timing requirements are met across these blocks before integrating with top-level blocks.
- Re-use the block without reoptimizing for timing closure to ensure that changes in other blocks do not impact your block.
- Re-use a block in multiple designs.
- Reduce the verification time by re-verifying the updated part of the design only.

Supported Device Families

The following table lists the family of devices that Libero® SoC supports. This guide covers all these device families. However, some information in this guide might apply to certain device families only. In this case, such information is clearly identified.

Table 1. Device Families Supported by Libero SoC

Device Family	Description
PolarFire®	PolarFire FPGAs deliver the industry's lowest power at mid-range densities with exceptional security and reliability.
PolarFire SoC	PolarFire SoC is the first SoC FPGA with a deterministic, coherent RISC-V CPU cluster, and a deterministic L2 memory subsystem enabling Linux and real-time applications.
SmartFusion®2	SmartFusion2 addresses fundamental requirements for advanced security, high reliability, and low power in critical industrial, military, aviation, communications, and medical applications.
IGLOO®2	IGLOO2 is a low-power mixed-signal programmable solution.
RTG4™	RTG4 is Microchip's family of radiation-tolerant FPGAs.

© 2024 Microchip Technology Inc. and its subsidiaries

Table of Contents

© 2024 Microchip Technology Inc. and its subsidiaries

© 2024 Microchip Technology Inc. and its subsidiaries

1. Overview (Ask a Question)

The following topics provide an overview of Libero SoC Block Flow.

1.1 Block Features (Ask a Question)

You can synthesize, simulate, and place and route a block as a regular design. You can lock the place and route of the block to ensure repeatable performance. Performance and place and route of a block can be fixed. However, these rules can be relaxed, if necessary, to ensure that you can integrate the block into the top-level project.

Use blocks when you:

- Have multiple team members working on different parts of the same design.
- Have a crowded design that uses 90 percent or more of the resources on a given die.
- Have difficulty in meeting timing as working on the entire design. Blocks allow you to compartmentalize and optimize design sections before optimizing the entire design.
- Want to re-use elements of the design.
- Want to make minor changes to the design and expect to keep the remaining design unchanged with ensured performance.

Blocks are family- and die-specific, and cannot be used with all families. If your block has I/Os, it is also package-specific.

1.2 Supported Hardware Description Languages (Ask a Question)

PolarFire Block Flow supports Verilog and VHDL Hardware Description Languages (HDLs).

1.3 Supported Synthesis Tools (Ask a Question)

PolarFire Block Flow supports Synplify Pro®.

1.4 Nested Blocks (Ask a Question)

PolarFire Block Flow supports nested blocks (that is, blocks instantiated inside other blocks). When published, only one file gets published. The published file contains all the required information, including the nested block.

© 2024 Microchip Technology Inc. and its subsidiaries

2. Creating Blocks, Options, and Settings (Ask a Question)

PolarFire Block Flow allows you to publish a reusable component that can be instantiated into another design. A block component includes timing constraints, physical constraints, placement, and routing details. To enable block creation option:

1. Launch the Libero SoC tool.
2. For a new project:
   a) Select **Project** > **New Project**. The **New Project** window appears.
   b) Specify the project details and select the **Enable block creation** check box.
   c) Click **Next** and continue with creating a new project.
3. For an existing project:
   a) Select **Project** > **Project Settings**. The **Project settings** window appears.
   b) Click **Design flow** on the left pane.
   c) Select the **Enable block creation** check box available on the right panel.
   d) Click **Save** to update the project settings.

   Important: To change the block flow settings, you must clean and rerun all your design flow tools. It is recommended that you back up the files you want to preserve.

   e) Click **Close** to close the Project settings window.

Important: A block flow component may not contain I/O cells and cannot be programmed by itself.

2.1 Synthesis Tool Settings (Ask a Question)

Libero disables I/O insertion automatically before invoking the Synplify Pro tool. As a result, the **I/O Insertion** option in the tool is disabled when the block is synthesized.

2.2 Synthesis (Ask a Question)

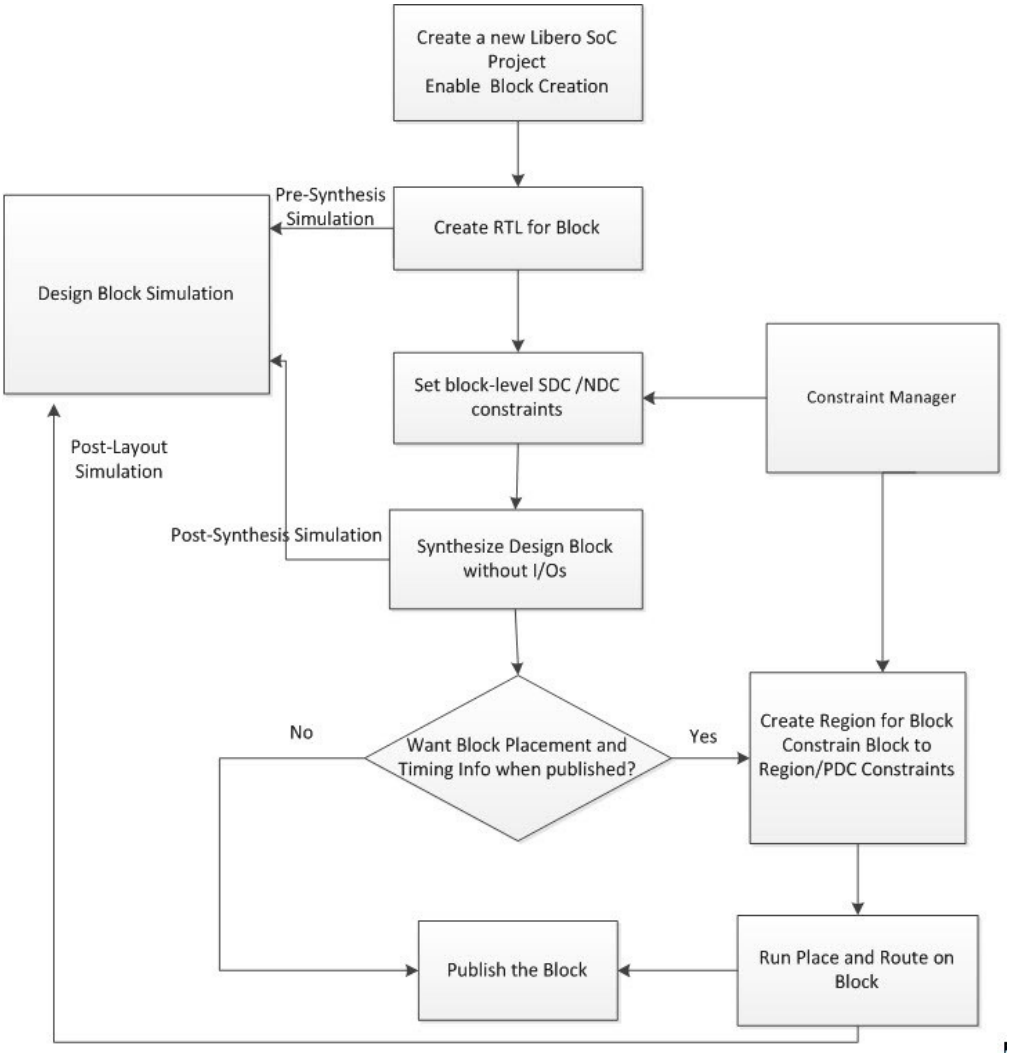
During Synthesis, the Libero SoC software adds instances of `BLOCK_INTERFACE_I*` to the block. These instances are virtual buffers added to:

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

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

© 2024 Microchip Technology Inc. and its subsidiaries

Figure 2-1. Creating and Publishing Design Blocks



2.3 Publish Options/Settings (Ask a Question)

Use the Publish Block – Configuration Options dialog box to configure and publish the block.

© 2024 Microchip Technology Inc. and its subsidiaries

3. Publishing Blocks (Ask a Question)

You can publish a block after Synthesis or Layout.

3.1 Publish Block - Configuration Options (Ask a Question)

To view this dialog box, you must check **Enable block creation** option in the **Project > Project Settings... > Design Flow** dialog box or in the **New Project Creation Wizard**. Once enabled, **Publish Block** option appears in the **Design Flow** window.

To export the block, expand **Publish Design**, right-click **Publish Block**, and select **Export**.

Publish Block Configuration

- **Publish Placement**- Select to publish the placement information for the Block. Note that you must assign all macros to regions or lock them 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** - Select to retain the routing information with the block when published.
- **Publish Region** - Select to retain the region constraint information with the block when published.

Language

Select Block HDL (Verilog or VHDL) to Preferred HDL type set specified in the **Project Settings** dialog box by default.

Important: Blocks must always be published in Verilog, not VHDL.

3.2 Publishing Post Synthesis (Ask a Question)

If you publish a block after Synthesis but before Layout, a netlist is exported for the block. Place and route information or Region Constraint information is not included in the published block. A warning message is displayed when you publish a block prior to Place and Route.

3.3 Publishing Post Layout (Ask a Question)

If you publish a block after Layout, the placement, routing, and/or region constraint information is published with the netlist. In the configurator, you can select the information you want to publish. To publish the placement and routing information, all macros must be locked or assigned to respective regions.

3.4 Publishing Content (Ask a Question)

When a block is published, Libero exports the `<design>.cxz` file to the `<project folder>/designer/<design_block_name>/export` folder. The `<design>.cxz` file is a zip file that contains the files listed in the table:

Table 3-1. Contents of the design.cxz Zip File

File	Description
`<design_block_name>_syn.v` | `<design_block_name>_syn.vhd`	Timing shell file passed to the 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.

© 2024 Microchip Technology Inc. and its subsidiaries

.....continued

File	Description
<design_block_name>_sim.v | <design_block_name>_sim.vhd	Structural HDL netlist for post-synthesis simulation of the block.
header_report.log	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_netlist_resources.xml	Compile report that details resource usage, device information, and a list of high-fanout nets.
<design_block_name>_gp_report.xml	Global Place and Route report.
<design_block_name>_compile_netlist_combinational_loops.xml	Combinational Loops report.
<design>.cdb	Internal proprietary file contains the optimized netlist, Place and Route, or the timing constraint information.
<design_block_name>.sdc	Contains the SDC constraints for the block to be used for Timing Verifications.
<design_block_name>.cxz	The published block. You can move it to another folder, transfer it to other team members, and so on. This file is imported into the top-level design when you want to instantiate the block.

© 2024 Microchip Technology Inc. and its subsidiaries

4. Guidelines for Creating Blocks (Ask a Question)

Following are the guidelines that need to be followed when creating blocks.

4.1 Peripheral and Embedded Hard Blocks (Ask a Question)

Various features of the architecture are located within the periphery but are not available within the FPGA array. These features can be incorporated within a block design but must be carefully enclosed to all the feature's I/Os within the block. Following are the list of features for PolarFire design:

- DRI
- APBM
- SCB
- ENFORCE
- DEBUG
- TVS
- OSC_RC200MHZ
- PF_SPI
- SC_STATUS
- UJTAG_SEC
- SYS_SERVICES
- VOLTAGEDETECT
- OSC_RC2MHZ
- INIT
- TAMPER
- PCIE
- XCVR_PIPE_AXI0
- XCVR_PIPE_AXI1

Note: While checking these interfaces in Chip Planner, flyline connections could converge at a single co-ordination. A large bus structure spread across an area is seen in the Chip Planner when the actual routing is viewed. If any I/O has an external name, off the chip, then it must be the same port name as in the top-level of the design.

4.2 Managing Place and Route and Globals (Ask a Question)

This option is available only when the **Block Creation** option is enabled (**Project > Project Settings > Design Flow > Enable Block Creation**).

The Place-and-Route tool has an option to limit the number of row globals used for block creation.

To use this option:

1. From the `Design Flow` window, right-click **Place and Route** and choose **Configure Options**. The Layout Options dialog box shows the default number of row-global resources for the technology family.
2. Enter a value to restrict the number of row-global resources available in every half-row of the device. During Place and Route of the block, the tool will not exceed this capacity on any half-row. The default value is the maximum number of row globals. If you enter a value lower than the maximum capacity (the default), the layout of the block will be able to integrate with the rest of the design if the remaining row-global capacity is consumed.

© 2024 Microchip Technology Inc. and its subsidiaries

4.3 Blocks and DRC (Ask a Question)

Regular Design Rules Check (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.

4.4 Blocks and Floorplanning (Ask a Question)

When creating a block, floorplanning is essential if you plan to publish the placement information. Before running Layout on the block, you must have the floorplan of the design block. You can use Chip Planner 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. The message warns you that not all macros in the block have been constrained to regions or locked. Therefore, only your design netlist is exported when the block is published.

4.4.1 Floorplanning with PDC Commands (Ask a Question)

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.

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.

4.4.2 Floorplanning with Chip Planner (Ask a Question)

For information about using the Chip Planner for floorplanning, see the Chip Planner User Guide for Libero SoC v2021.1 for all the families.

4.5 Architectural Limitations (Ask a Question)

The silicon architecture has a limited number of globals per device. If you create a block in a top-level design, it is recommended to minimize the number of globals when you create the block as the top-level design uses the maximum number of globals for the device, in general.

The Global Report shows the number of globals that are used for the block. To reduce the number of globals used in the block, consider clock sharing and using row globals for the block.

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

For example, to add a `CLKINT` after a CLK, use the following command:

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


© 2024 Microchip Technology Inc. and its subsidiaries

5. Instantiating Blocks in the Top-Level Design (Ask a Question)

You can instantiate multiple instances of the same block or multiple blocks in the top-level design. Microchip recommends that you create a new project for your top-level design using the following steps:

1. From the **Project** menu, choose **New Project**.
2. Uncheck the **Enable Designer Block Creation** check box.
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.

5.1 Importing a Block (Ask a Question)

To import a block:

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

The `<design_block_name>` file is imported into the `top_level` project. Version control is not supported for imported blocks. If you import the same block twice, the new block overwrites the existing block.

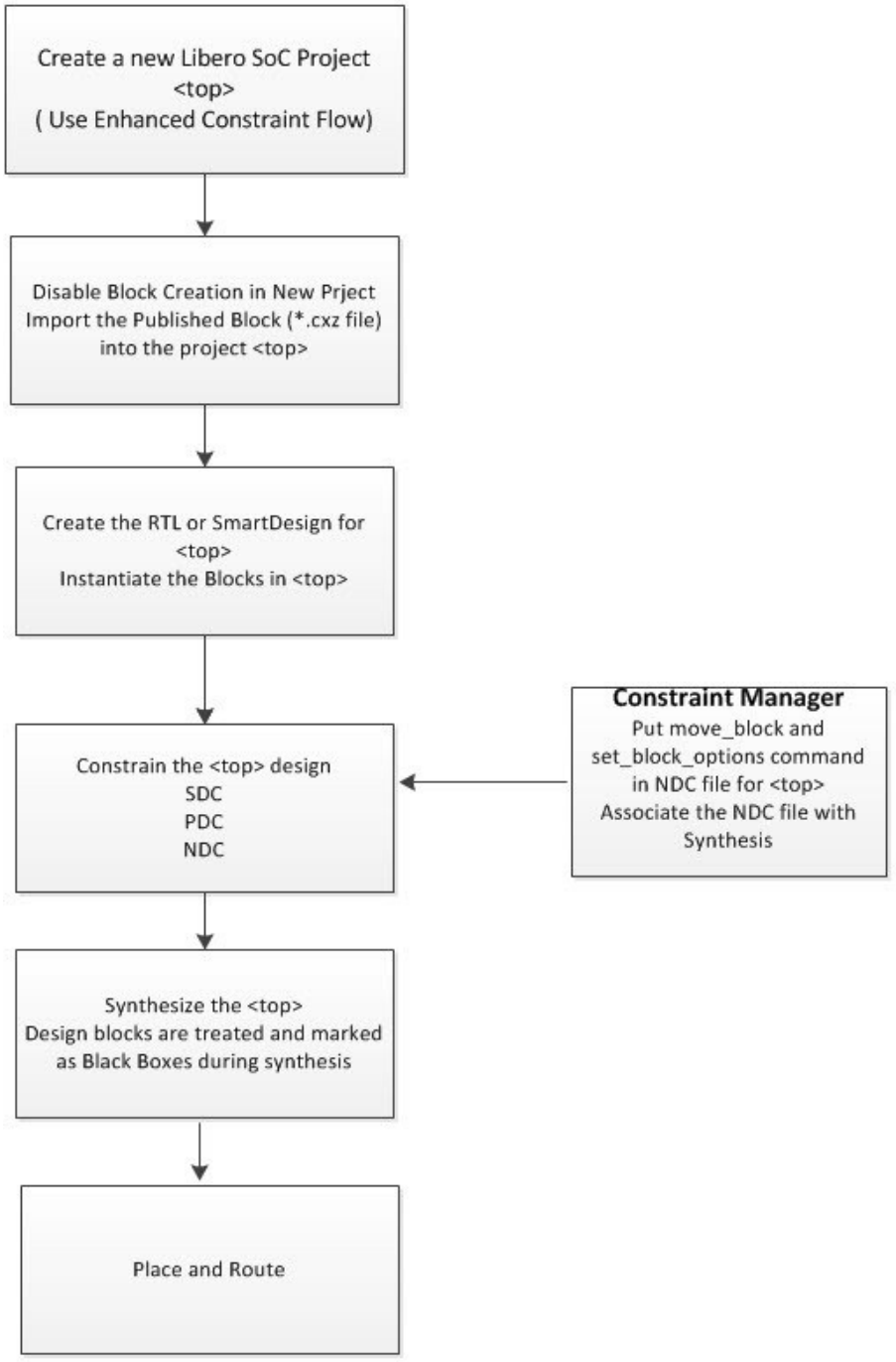
Files are imported under the `<design>\component\work\<design_block_name>` folder. See the Block reports in the directory.

5.2 Create a Top-Level Design that Uses Blocks (Ask a Question)

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

© 2024 Microchip Technology Inc. and its subsidiaries

Figure 5-1. Instantiating Blocks in the Top-Level Design



5.3 Constraints Management (Ask a Question)

When a block with PDC constraints are imported into the top-level design, the block's PDC constraints are captured and stored in two files:

- `<top_level_module>.block.io.pdc` for the I/O PDC constraints.
- `<top_level_module>.block.fp.pdc` for the floorplanning PDC constraints.

The `<top_level_module>.block.io.pdc` is displayed in the `I/O Attributes` tab of the Constraint Manager on top of any other IO PDC files.

© 2024 Microchip Technology Inc. and its subsidiaries

The `<top_level_module>.block.fp.pdc` is displayed in the `Floor Planning` tab of the Constraint Manager on top of any other floorplanning PDC files.

Note: If you need to make any changes to the PDC files, go to the project where the blocks are created and published and make required changes. Make the floorplanning modifications and publish the block. Re-import the block into the top level. You may need to remove any duplicate blocks, at the top level after the re-import.

© 2024 Microchip Technology Inc. and its subsidiaries

6. Hierarchical Structure Resolution in Top-Level Projects (Ask a Question)

If you import multiple conflicting definitions for your `*.v` files, following are the conflicts resolved by Libero.

6.1 Duplicate Block Definition (Ask a Question)

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

6.2 Conflicting Definitions in top.v and Imported Block File (Ask a Question)

You can introduce a conflict if you import `top.v` and block files. 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 the Synthesis tool because the definition for `<block1>` is duplicated. To continue, you must remove the `<block1>` definition from the `top.v` file and then re-import it.

6.3 Resolving top.v and Block Instantiations (Ask a Question)

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

1. Import the `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 and moves <block1> under `top.v`.

© 2024 Microchip Technology Inc. and its subsidiaries

7. Synthesis (Ask a Question)

Libero passes the block timing to the Synthesis tool when the top-level design 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.

Use the Synthesis tool options (**Design Flow > Synthesize > Configure Options**) to resolve Place and Route conflicts.

© 2024 Microchip Technology Inc. and its subsidiaries

8. Resolving Place and Route Conflicts (Ask a Question)

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

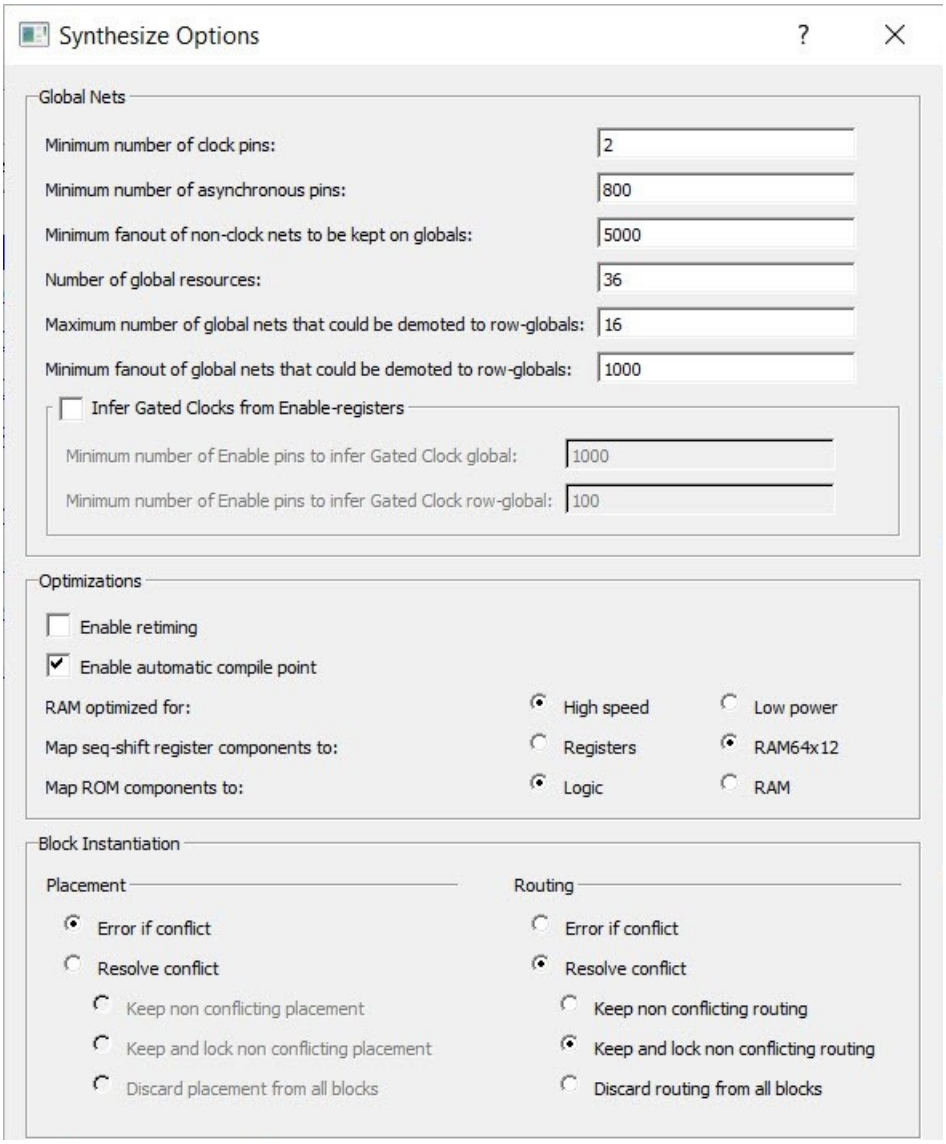
- Examine the `<design_block_name>_compile_netlist_resources.xml` report. Identify the cause of the problem and manually place and constrain the placement with Chip Planner or with PDC commands. See Chip Planner User Guide.
- If you instantiate a block (published with placement) multiple times, then 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`move_block` command. Put the `move_block` command inside the NDC file and associate the NDC file with Synthesis (**Constraint Manager > Netlist Attributes**).
- The software enforces Global sharing. If there is a Global driving 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).

8.1 Synthesis Options to Resolve Place and Route Conflicts (Ask a Question)

If there are multiple blocks instantiated in the top-level design, users can use the synthesis options to resolve the conflicts. These options appear only if there are blocks in the design. Use the synthesis options (**Design Flow** > **Synthesize** > **Configure Options**) to resolve place and route conflicts.

© 2024 Microchip Technology Inc. and its subsidiaries

Figure 8-1. Synthesis Options Dialog Box



8.1.1 Placement (Ask a Question)

The following table lists the various Block Instantiation options for placement.

Table 8-1. Block Instantiation - Placement Options

Option	Description
Error if conflict	The Layout tool 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.

© 2024 Microchip Technology Inc. and its subsidiaries

8.1.2 Routing (Ask a Question)

The following table lists the various Block Instantiation options for routing.

Table 8-2. Block Instantiation - Routing Options

Option	Description
Error if conflict	The Layout tool 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.

© 2024 Microchip Technology Inc. and its subsidiaries

9. Block PDC Commands (Ask a Question)

move_block and set_block_options are two PDC commands available for working with design blocks at the top-level design.

Use the move_block and set_block_options commands to make changes in the top-level design.

In the top-level design, put the move_block and set_block_options commands in an NDC file (**Design Flow Window** > **Manage Constraints** > **Open Manage Constraints View** > **Netlist Attributes** > **New** > **Create New Compile Netlist Constraints NDC**) and associate the NDC file with Synthesis.

9.1 move_block (Ask a Question)

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 Chip Planner 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 move a part of the block, you can move the remaining parts of the block and then place them manually as necessary.

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

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

Arguments

Parameter	Description
-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. For more information, see Chip Planner User Guide.
-up y	Moves the block to the specified number of rows up. The value must be a positive integer.
-down y	Moves the block to the specified number of rows down. The value must be a positive integer.
-left x	Moves the block left the specified number of rows. The value must be a positive integer.
-right x	Moves the block right the specified number of rows. The value must be a positive integer.

© 2024 Microchip Technology Inc. and its subsidiaries

.....continued	
Parameter	**Description**
`-non_logic value`	Specifies what to do with the non-logic part of the block, if one exists. The acceptable values for this argument are: • `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.

Exceptions

- You must import the PDC command as an NDC file.
- You must use the PDC command if you want to preserve the relative placement of a block you are instantiating many times in the design. Only one instance will be preserved by default. To preserve other instances, you must move them using this command.

Example

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
```

Supported Families

Supported Families	Supported Versions
PolarFire® FPGAs	v12.4+

See Also

- set_block_options

9.2 set_block_options (Ask a Question)

Description

This command enables you to override 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

Parameter	Description
`-inst_name instance_name`	Specifies the name of the instance to move. If you do not know the name of the instance, run a Block report (**Design** > **Reports** > **Block** > **Interface**) or look at the names shown in the Block tab. For more information, see Chip Planner User Guide.

© 2024 Microchip Technology Inc. and its subsidiaries

.....continued	
Parameter	**Description**
`-placement_conflicts value`	Specifies what to do when the software encounters a placement conflict. The acceptable values for this argument are: • `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 acceptable values for this argument are: • `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. • `discard` - Discards any routing from the block, even if there are no conflicts.

Exceptions

- You must import this PDC command as an NDC file.
- If placement is discarded, the routing is automatically discarded too.

Example

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 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
```

Supported Families

Supported Families	Supported Versions
PolarFire® FPGAs	v12.4+

See Also

- move_block

© 2024 Microchip Technology Inc. and its subsidiaries

10. Revision History (Ask a Question)

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	Date	Description
D	12/2024	The following is a summary of the changes made in this revision. • Updated the title of the document from PolarFire® Block Flow User Guide to Libero SoC Block Flow User Guide. • Added the Supported Device Families in Introduction. • Updated the Publish Block - Configuration Options section. • Editorial updates throughout the document.
C	12/2021	This document is released with Libero SoC Design Suite v2021.3 without changes from v2021.2.
B	08/2021	This document is released with Libero SoC Design Suite v2021.2 without changes from v2021.1.
A	04/2021	Initial Revision.

© 2024 Microchip Technology Inc. and its subsidiaries

Microchip Information

Trademarks

The “Microchip” name and logo, the “M” logo, and other names, logos, and brands are registered and unregistered trademarks of Microchip Technology Incorporated or its affiliates and/or subsidiaries in the United States and/or other countries (“Microchip Trademarks”). Information regarding Microchip Trademarks can be found at https://www.microchip.com/en-us/about/legal-information/microchip-trademarks.

ISBN: 979-8-3371-0124-8

Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

THIS INFORMATION IS PROVIDED BY MICROCHIP “AS IS”. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP’S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION.

Use of Microchip devices in life support and/or safety applications is entirely at the buyer’s risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip product is strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is “unbreakable”. Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.

© 2024 Microchip Technology Inc. and its subsidiaries

Microchip FPGA Support

Microchip FPGA products group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, and worldwide sales offices. Customers are suggested to visit Microchip online resources prior to contacting support as it is very likely that their queries have been already answered.

Contact Technical Support Center through the website at www.microchip.com/support. Mention the FPGA Device Part number, select appropriate case category, and upload design files while creating a technical support case.

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

- From North America, call **800.262.1060**
- From the rest of the world, call **650.318.4460**
- Fax, from anywhere in the world, **650.318.8044**

© 2024 Microchip Technology Inc. and its subsidiaries