Libero® SoC v2021.2

Microchip Separation Verification Tool User Guide

Introduction

The Microchip Design Separation methodology provides a way to create the independent critical subsystems required to implement security- and safety-critical applications on a single FPGA. Microchip Separation Verification Tool (MSVT) is a stand-alone tool provided with your Libero® installation. It is used to verify that your design meets the requirements of the design separation criteria. For more information, see *the Microchip Design Separation Methodology Guide*.

Table of Contents

1. Overview

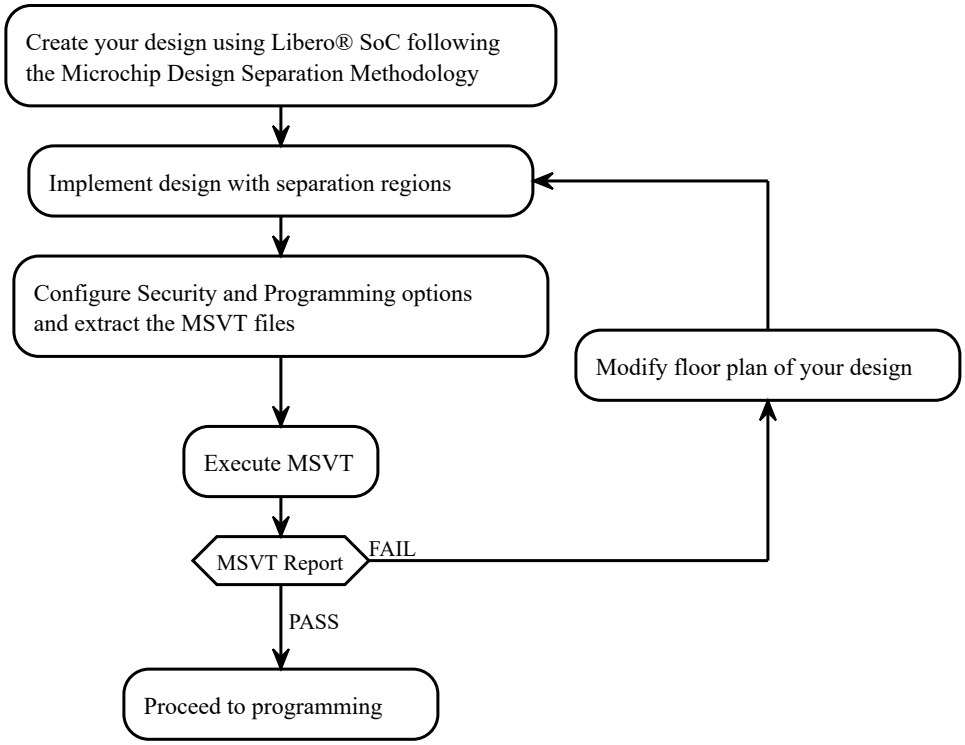
MSVT can work on any placed and routed design that has a block requiring a separation from all elements external to the block. The tool works iteratively on every block to be verified. Internal signals and Inter-region signal (IRS) are verified separately. The tool checks whether the separation criteria is satisfied for each block and corresponding set of IRS signals.

Your design must adhere to the following criteria to implement a security- and safety-critical system:

- Each block of your design must be assigned to a separation region.
- There must be a minimum gap of unused logic clusters between separation regions, depending on your design requirement.
- Each set of inter-block interface signals must be defined as an IRS region.

The following flowchart lists the design separation methodology steps.

Figure 1-1. Design Separation Methodology Steps

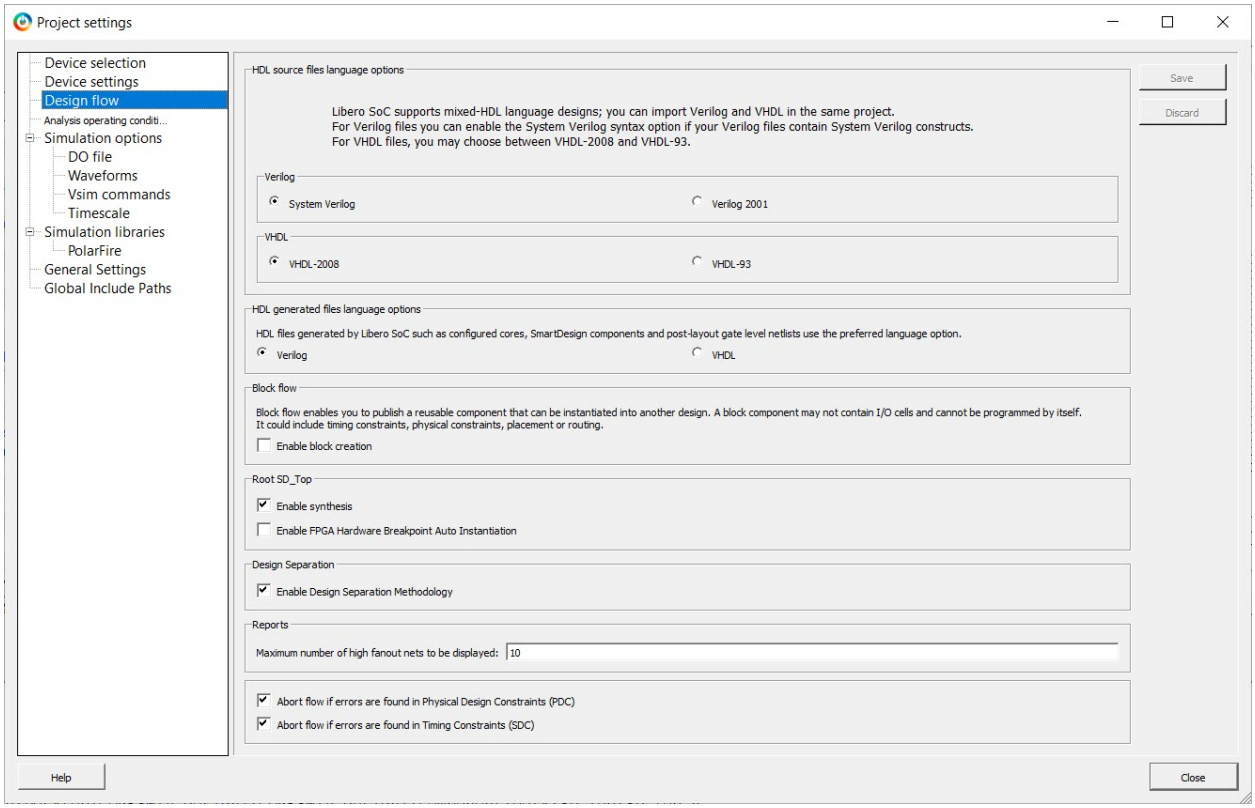


2. Creating a Design

A complete design comprises sub-systems published in terms of block elements. Each block element is instantiated in a top-level module. The top-level module is floorplanned into separation regions for each sub-system block and overlapping IRS region connecting the blocks. These IRS regions must be isolated physically from other IRS regions. Select the **Enable Design Separation Methodology** check box in the Design flow settings on the Project settings page as shown in following figure. The design is then run through a layout, followed by the timing closure of your design.

For more information about creating your design, see the *Microchip Design Separation Methodology Guide*.

Figure 2-1. Enabling Design Separation Methodology Before Compile



3. Extracting MSVT Files

The information used by the MSVT is exported while generating the programming file. The tool takes as input the design database and a parameter file that is generated once per project. The parameter file describes the isolation regions in the design as well as the inter-region signals between isolation regions. The parameter file is exported to the following location:

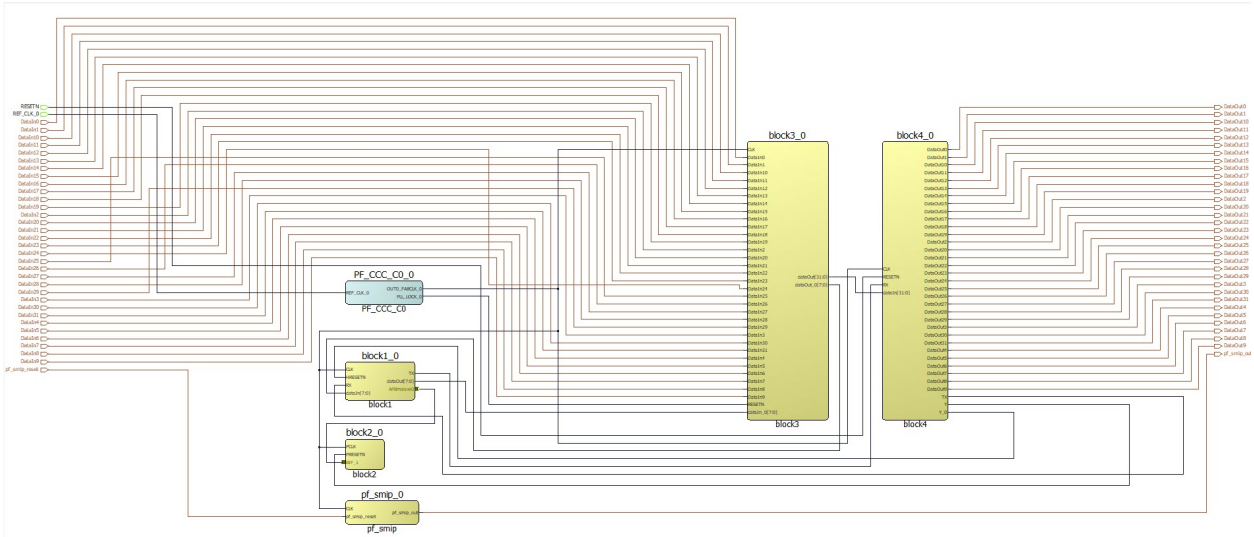
```
<project_path>/designer/<Top_Level_Module>/msvt.param
```

3.1 MSVT.param File

`msvt.param` is an auto-generated file that contains parameters required by MSVT to verify design separation. You can modify the contents of this file according to your requirements.

The following is a Top-Level view of an example design.

Figure 3-1. Top-Level Smart Design Showing Different Design Blocks



The following is an example of a `msvt.param` file. The contents of the `msvt.param` file depend on your design.

Example 3-1. msvt.param File

```
//*********************************************************************
//
//  This is input parameters file for MSVT Check program
//
//*********************************************************************

DEVICE = MPF300TS
DESIGN = SD_Top.msvt
VERIFY_BLOCKS = block4_0 block2_0 block3_0 block1_0 pf_smip_0 // empty list
means all blocks in design will be verified
REQUIRED_SEPARATION = 1
MAX_VIOLATIONS_PER_REPORT_SECTION = 1
IRS block4_0 block2_0 = block4_0_Y
IRS block2_0 block4_0 =
IRS block4_0 block3_0 =
IRS block3_0 block4_0 = block3_0_dataOut[31] block3_0_dataOut[30]
block3_0_dataOut[29]
        block3_0_dataOut[28] block3_0_dataOut[27] block3_0_dataOut[26]
block3_0_dataOut[25]
        block3_0_dataOut[24] block3_0_dataOut[23] block3_0_dataOut[22]
block3_0_dataOut[21]
        block3_0_dataOut[20] block3_0_dataOut[19] block3_0_dataOut[18]
block3_0_dataOut[17]
```

```
        block3_0_dataOut[16] block3_0_dataOut[15] block3_0_dataOut[14]
block3_0_dataOut[13]
        block3_0_dataOut[12] block3_0_dataOut[11] block3_0_dataOut[10]
block3_0_dataOut[9]
        block3_0_dataOut[8] block3_0_dataOut[7] block3_0_dataOut[6]
block3_0_dataOut[5]
        block3_0_dataOut[4] block3_0_dataOut[3] block3_0_dataOut[2]
block3_0_dataOut[1]
        block3_0_dataOut[0]
IRS block4_0 block1_0 = block4_0_TX block4_0_Y_0
IRS block1_0 block4_0 = block1_0_TX
IRS block4_0 pf_smip_0 =
IRS pf_smip_0 block4_0 =
IRS block2_0 block3_0 =
IRS block3_0 block2_0 =
IRS block2_0 block1_0 = block1_0_APBmslave0_PRDATA[31]
block1_0_APBmslave0_PRDATA[30]
        block1_0_APBmslave0_PRDATA[29] block1_0_APBmslave0_PRDATA[28]
block1_0_APBmslave0_PRDATA[27]
        block1_0_APBmslave0_PRDATA[26] block1_0_APBmslave0_PRDATA[25]
block1_0_APBmslave0_PRDATA[24]
        block1_0_APBmslave0_PRDATA[23] block1_0_APBmslave0_PRDATA[22]
block1_0_APBmslave0_PRDATA[21]
        block1_0_APBmslave0_PRDATA[20] block1_0_APBmslave0_PRDATA[19]
block1_0_APBmslave0_PRDATA[18]
        block1_0_APBmslave0_PRDATA[17] block1_0_APBmslave0_PRDATA[16]
block1_0_APBmslave0_PRDATA[15]
        block1_0_APBmslave0_PRDATA[14] block1_0_APBmslave0_PRDATA[13]
block1_0_APBmslave0_PRDATA[12]
        block1_0_APBmslave0_PRDATA[11] block1_0_APBmslave0_PRDATA[10]
block1_0_APBmslave0_PRDATA[9]
        block1_0_APBmslave0_PRDATA[8] block1_0_APBmslave0_PRDATA[7]
block1_0_APBmslave0_PRDATA[6]
        block1_0_APBmslave0_PRDATA[5] block1_0_APBmslave0_PRDATA[4]
block1_0_APBmslave0_PRDATA[3]
        block1_0_APBmslave0_PRDATA[2] block1_0_APBmslave0_PRDATA[1]
block1_0_APBmslave0_PRDATA[0]
        block1_0_APBmslave0_PREADY
IRS block1_0 block2_0 = block1_0_APBmslave0_PADDR[11]
block1_0_APBmslave0_PADDR[10]
        block1_0_APBmslave0_PADDR[9] block1_0_APBmslave0_PADDR[8]
block1_0_APBmslave0_PADDR[7]
        block1_0_APBmslave0_PADDR[6] block1_0_APBmslave0_PADDR[5]
block1_0_APBmslave0_PADDR[4]
        block1_0_APBmslave0_PADDR[3] block1_0_APBmslave0_PADDR[2]
block1_0_APBmslave0_PADDR[1]
        block1_0_APBmslave0_PADDR[0] block1_0_APBmslave0_PWDATA[31]
block1_0_APBmslave0_PWDATA[30]
        block1_0_APBmslave0_PWDATA[29] block1_0_APBmslave0_PWDATA[28]
block1_0_APBmslave0_PWDATA[27]
        block1_0_APBmslave0_PWDATA[26] block1_0_APBmslave0_PWDATA[25]
block1_0_APBmslave0_PWDATA[24]
        block1_0_APBmslave0_PWDATA[23] block1_0_APBmslave0_PWDATA[22]
block1_0_APBmslave0_PWDATA[21]
        block1_0_APBmslave0_PWDATA[20] block1_0_APBmslave0_PWDATA[19]
block1_0_APBmslave0_PWDATA[18]
        block1_0_APBmslave0_PWDATA[17] block1_0_APBmslave0_PWDATA[16]
block1_0_APBmslave0_PWDATA[15]
        block1_0_APBmslave0_PWDATA[14] block1_0_APBmslave0_PWDATA[13]
block1_0_APBmslave0_PWDATA[12]
        block1_0_APBmslave0_PWDATA[11] block1_0_APBmslave0_PWDATA[10]
block1_0_APBmslave0_PWDATA[9]
        block1_0_APBmslave0_PWDATA[8] block1_0_APBmslave0_PWDATA[7]
block1_0_APBmslave0_PWDATA[6]
        block1_0_APBmslave0_PWDATA[5] block1_0_APBmslave0_PWDATA[4]
block1_0_APBmslave0_PWDATA[3]
        block1_0_APBmslave0_PWDATA[2] block1_0_APBmslave0_PWDATA[1]
block1_0_APBmslave0_PWDATA[0]
        block1_0_APBmslave0_PENABLE block1_0_APBmslave0_PSELx
block1_0_APBmslave0_PWRITE
IRS block2_0 pf_smip_0 =
IRS pf_smip_0 block2_0 =
IRS block3_0 block1_0 = block3_0_dataOut_0[7] block3_0_dataOut_0[6]
block3_0_dataOut_0[5]
        block3_0_dataOut_0[4] block3_0_dataOut_0[3] block3_0_dataOut_0[2]
block3_0_dataOut_0[1]
```

```
        block3_0_dataOut_0[0]
IRS block1_0 block3_0 = block1_0_dataOut[7] block1_0_dataOut[6]
block1_0_dataOut[5]
        block1_0_dataOut[4] block1_0_dataOut[3] block1_0_dataOut[2]
block1_0_dataOut[1]
        block1_0_dataOut[0]
IRS block3_0 pf_smip_0 =
IRS pf_smip_0 block3_0 =
IRS block1_0 pf_smip_0 =
IRS pf_smip_0 block1_0 =
REGIONS_VERBOSITY = 0
```

Table 3-1. Parameters in the msvt.param File

Parameter	Description
DEVICE	Name of the Microchip FPGA device implementing the design.
DESIGN	Location of the files required for MSVT. By default, it will point to the auto-generated `msvt.dtf` folder.
VERIFY BLOCKS	Contains a list of blocks to be verified. By default, all the block names present in the design are listed. You can modify this list and include a subset of blocks to be audited by MSVT.
REQUIRED_SEPARATION	Required separation parameter per the guideline requirements. The default value is 1.
MAX_VIOLATIONS_PER_REPORT_SECTION	Controls the number of violations that are to be reported in each section. The default value is 1.
IRS	Contains a list of IRS signal names present in the design. Each IRS statement is comprised of a pair of separated blocks followed by a list of the IRS signal names between them.
REGIONS_VERBOSITY	Controls reporting of each routing region and the assigned instances. The default value is 0.

You can modify the parameters in the `msvt.param` file to refine your verification criteria. We recommend modifying the `REQUIRED_SEPARATION` parameter according to your system requirements before executing MSVT. You can also specify the blocks you want to verify, the names of each IRS signal, and add a limit to the max number of violations to be reported.

4. Using the MSVT Tool

The MSVT tool prints a comprehensive report about each block and corresponding IRS region being verified. If any block or IRS signals do not satisfy the minimum separation criteria, the tool reports the details of the affected instances. More information about each section of the report is described in the following sections.

An MSVT failure indicates that the design has not met the design separation criteria and that one or more sub-blocks (or signals) are not independent from the rest of the system. In this case, you must:

- Identify the instances that cause violations in the MSVT output and modify the design floor-plan accordingly.
- Recompile the design to generate a new placed and routed netlist.
- Use the MSVT tool to verify the modified design.

4.1 Using the msvt_check Command

`msvt_check` is a command-line based standalone tool that verifies designs developed for the SmartFusion®2 and IGLOO®2 families. The tool is available in the `<Libero_path>/bin64` folder.

Note: To verify designs developed for the PolarFire® family of devices, use the `msvt_check_pf` command.

Syntax

```
<Libero_path>/bin64/msvt_check -p <project_path>/designer/<Top_Level_Module>/msvt.param [-o msvt_check.log]
```

Arguments

The following table lists arguments you can use with the `msvt_check` command.

Table 4-1. msvt_check Arguments

Argument	Description
`-p <msvt.param file path>`	Path to the `msvt.param` file. The `msvt.param` file is generated using Libero. This argument is required.
`-o <filename>`	Prints a comprehensive report to the specified file. The report prints at the command prompt, if the `-o` argument is omitted. This argument is optional.

Returns

The following table lists the return values when the `msvt_check` command is used.

Table 4-2. msvt_check Return Values

Return Value	Description
MSVT Check Failed	Design has not met one or more of the separation criteria.
MSVT Check Succeeded	Design has met all separation criteria.

4.2 Using the msvt_check_pf Command

`msvt_check_pf` is a command-line based standalone tool that verifies the designs developed for the PolarFire® family. The tool is available in the `<Libero_path>/bin64` folder.

Note: To verify designs developed for the SmartFusion®2 and IGLOO®2 family devices, use the `msvt_check` command.

Syntax

```
<Libero_path>/bin64/msvt_check_pf -p <project_path>/designer/<Top_Level_Module>/msvt.param [-
o msvt_check.log]
```

Arguments

The following table lists arguments you can use with the `msvt_check_pf` command.

Table 4-3. msvt_check_pf Arguments

Argument	Description
`-p <msvt.param file path>`	Path to the `msvt.param` file. The `msvt.param` file is generated using Libero. This argument is required.
`-o <filename>`	Prints a comprehensive report to the specified file. The report prints at the command prompt if the `-o` argument is omitted. This argument is optional.

Returns

The following table lists the return values when the `msvt_check_pf` command is used.

Table 4-4. msvt_check_pf Return Values

Return Value	Description
MSVT Check Failed	Design has not met one or more of the separation criteria.
MSVT Check Succeeded	Design has met all separation criteria.

4.3 MSVT Report

When MSVT executes successfully, it generates a comprehensive report with details about each block and the IRS regions between each block.

The following example is an MSVT-generated report for a design that satisfies the design separation criteria as specified in the `msvt.param` discussed in the 3.1 MSVT.param File section.

Example 4-1. MSVT Output Report

```
MSVT Check
Design: SD_Top.msvt                    Started: Tue Dec 22 04:25:38 2020


Checking IRS connectivity against parameter file
================================================

The following instances do not belong to any routing region:
====================================================================
   PF_CCC_C0_0/PF_CCC_C0_0/pll_inst_0
   REF_CLK_0_ibuf/U_IOIN


The following IRS nets are not constrained by any routing region:
====================================================================
   block4_0_TX
   block1_0_TX

Analyzing  floorplan ...
========================

   block4_0 and block2_0 : Minimal floorplan separation = 9 clusters.
      block4_0 at cluster (144,62)
      block2_0 at cluster (144,52)
   block4_0 and block2_0 : Minimal placement separation = 21 clusters.
      (2148,156) containing cell block4_0/BUFD_1/U0
      (2148,225) containing cell block2_0/BUFD_0/U0
```

```
   block4_0 and block3_0 : Minimal floorplan separation = 11 clusters.
      block4_0 at cluster (99,27)
      block3_0 at cluster (87,27)
   block4_0 and block3_0 : Minimal placement separation = 11 clusters.
      (1211,82) containing cell block4_0/CoreGPIO_C4_0/CoreGPIO_C4_0/
inData_s2[6]
      (1057,81) containing cell block3_0/APB_dp_fp_1/U0/i_post_norm_mul/
s_shl2_RNIS34841[4]

   block4_0 and block1_0 : Minimal floorplan separation = 11 clusters.
      block4_0 at cluster (99,64)
      block1_0 at cluster (99,52)
   block4_0 and block1_0 : Minimal placement separation = 13 clusters.
      (1368,156) containing cell block4_0/BUFD_0/U0
      (1368,201) containing cell block1_0/BUFD_0/U0

   block4_0 and pf_smip_0 : Minimal floorplan separation = 37 clusters.
      block4_0 at cluster (99,0)
      pf_smip_0 at cluster (61,0)
   block4_0 and pf_smip_0 : Minimal placement separation = 38 clusters.
      (1219,2) containing cell block4_0/block4_IO_0/OUTBUF_31/U_IOTRI
      (746,2) containing cell pf_smip_0/PF_IO_C1_0/PF_IO_C1_0/I_IOD_0

   block4_0 and 'others' : Minimal floorplan separation =  overlapping.
      block4_0 at cluster (99,0)
      'others' at cluster (99,0)
   block4_0 and 'others' : Minimal placement separation = 0 clusters.
      (1219,2) containing cell block4_0/block4_IO_0/OUTBUF_31/U_IOTRI
      (1202,2) containing cell RESETN_ibuf/U_IOIN

   block2_0 and block3_0 : Minimal floorplan separation =  diagonal.
   block2_0 and block3_0 : Minimal placement separation =  diagonal.

   block2_0 and block1_0 : Minimal floorplan separation = 9 clusters.
      block2_0 at cluster (144,93)
      block1_0 at cluster (134,93)
   block2_0 and block1_0 : Minimal placement separation = 9 clusters.
      (1743,282) containing cell block2_0/BUFD_53/U0
      (1620,282) containing cell block1_0/BUFD_87/U0

   block2_0 and pf_smip_0 : Minimal floorplan separation =  diagonal.
   block2_0 and pf_smip_0 : Minimal placement separation =  diagonal.

   block2_0 and 'others' : Minimal floorplan separation = 9 clusters.
      block2_0 at cluster (144,62)
      'others' at cluster (144,52)
   block2_0 and 'others' : Minimal placement separation =  diagonal.

   block3_0 and block1_0 : Minimal floorplan separation = 10 clusters.
      block3_0 at cluster (38,64)
      block1_0 at cluster (38,53)
   block3_0 and block1_0 : Minimal placement separation = 22 clusters.
      (842,124) containing cell block3_0/CoreGPIO_C2_0/CoreGPIO_C2_0/dataOut[7]
      (842,196) containing cell block1_0/CoreGPIO_C0_0/CoreGPIO_C0_0/
inData_s1[7]

   block3_0 and pf_smip_0 : Minimal floorplan separation = 4 clusters.
      block3_0 at cluster (66,0)
      pf_smip_0 at cluster (61,0)
   block3_0 and pf_smip_0 : Minimal placement separation = 4 clusters.
      (811,2) containing cell block3_0/Block3_IO_0/INBUF_17/U_IOIN
      (746,2) containing cell pf_smip_0/PF_IO_C1_0/PF_IO_C1_0/I_IOD_0

   block3_0 and 'others' : Minimal floorplan separation = 11 clusters.
      block3_0 at cluster (99,0)
      'others' at cluster (87,0)
   block3_0 and 'others' : Minimal placement separation = 15 clusters.
      (1010,2) containing cell block3_0/Block3_IO_0/INBUF_19/U_IOIN
      (1202,2) containing cell RESETN_ibuf/U_IOIN

   block1_0 and pf_smip_0 : Minimal floorplan separation = 60 clusters.
      block1_0 at cluster (38,64)
      pf_smip_0 at cluster (38,3)
   block1_0 and pf_smip_0 : Minimal placement separation =  diagonal.
```

```
   block1_0 and 'others' : Minimal floorplan separation = 11 clusters.
      block1_0 at cluster (99,64)
      'others' at cluster (99,52)
   block1_0 and 'others' : Minimal placement separation = 66 clusters.
      (1204,204) containing cell block1_0/MIV_RV32IMC_C0_0/MIV_RV32IMC_C0_0/
u_opsrv_0/u_core_0/u_lsu_0/un1_lsu_expipe_req_op_2
      (1202,2) containing cell RESETN_ibuf/U_IOIN

   pf_smip_0 and 'others' : Minimal floorplan separation = 37 clusters.
      pf_smip_0 at cluster (61,0)
      'others' at cluster (99,0)
   pf_smip_0 and 'others' : Minimal placement separation = 37 clusters.
      (746,2) containing cell pf_smip_0/PF_IO_C1_0/PF_IO_C1_0/I_IOD_0
      (1202,2) containing cell RESETN_ibuf/U_IOIN


Checking internal nets for block block4_0 ...
=============================================================================

Checking IRS nets for block block4_0 ...
=============================================================================

Propagating IRS nets outgoing from block4_0 to block2_0
=============================================================================

Propagating IRS nets outgoing from block4_0 to block1_0
=============================================================================

Checking internal nets for block block2_0 ...
=============================================================================

Checking IRS nets for block block2_0 ...
=============================================================================

Propagating IRS nets outgoing from block2_0 to block1_0
=============================================================================

Checking internal nets for block block3_0 ...
=============================================================================

Checking IRS nets for block block3_0 ...
=============================================================================

Propagating IRS nets outgoing from block3_0 to block4_0
=============================================================================

Propagating IRS nets outgoing from block3_0 to block1_0
=============================================================================

Checking internal nets for block block1_0 ...
=============================================================================

Checking IRS nets for block block1_0 ...
=============================================================================

Propagating IRS nets outgoing from block1_0 to block4_0
=============================================================================

Propagating IRS nets outgoing from block1_0 to block2_0
=============================================================================

Propagating IRS nets outgoing from block1_0 to block3_0
=============================================================================

Checking internal nets for block pf_smip_0 ...
=============================================================================

Checking IRS nets for block pf_smip_0 ...
=============================================================================


Design has met 2 switches separation requirement
```

```
MSVT Check succeeded.
Number of errors: 0
```

4.4 MSVT Report Sections

This section describes the MSVT report sections and includes examples.

4.4.1 Checking IRS Connectivity Against a Parameter File

MSVT checks that all inter-region signals are specified as IRS statements in the `msvt.param` file, and that the specified IRS connections are consistent with the design netlist. A missing IRS net or invalid connection is counted as an error and listed in the `Checking IRS connectivity against parameter file` section.

Sample msvt.param File

The following example is a `msvt.param` file that has missing IRS signals.

Example 4-2. msvt.param File Snippet

```
VERIFY_BLOCKS = Block_Cdr_0 COretse_Block1_0 // empty list means all blocks in
design will be verified
REQUIRED_SEPARATION = 2
MAX_VIOLATIONS_PER_REPORT_SECTION = 1
IRS Block_Cdr_0 COretse_Block1_0 = Block_Cdr_0_APBmslave0_PADDR[9]
Block_Cdr_0_APBmslave0_PADDR[8]
        Block_Cdr_0_APBmslave0_PADDR[7] Block_Cdr_0_APBmslave0_PADDR[6]
Block_Cdr_0_APBmslave0_PADDR[5]
        Block_Cdr_0_APBmslave0_PADDR[4] Block_Cdr_0_APBmslave0_PADDR[3]
Block_Cdr_0_APBmslave0_PADDR[2]
        Block_Cdr_0_APBmslave0_PENABLE Block_Cdr_0_to_CORETSE_Preset
Block_Cdr_0_APBmslave0_PSELx
        Block_Cdr_0_APBmslave1_PSELx Block_Cdr_0_APBmslave2_PSELx
Block_Cdr_0_APBmslave0_PWDATA[31]
        Block_Cdr_0_APBmslave0_PWDATA[30] Block_Cdr_0_APBmslave0_PWDATA[29]
Block_Cdr_0_APBmslave0_PWDATA[28]
        Block_Cdr_0_APBmslave0_PWDATA[27] Block_Cdr_0_APBmslave0_PWDATA[26]
Block_Cdr_0_APBmslave0_PWDATA[25]
        Block_Cdr_0_APBmslave0_PWDATA[24] Block_Cdr_0_APBmslave0_PWDATA[23]
Block_Cdr_0_APBmslave0_PWDATA[22]
        Block_Cdr_0_APBmslave0_PWDATA[21] Block_Cdr_0_APBmslave0_PWDATA[20]
Block_Cdr_0_APBmslave0_PWDATA[19]
        Block_Cdr_0_APBmslave0_PWDATA[18] Block_Cdr_0_APBmslave0_PWDATA[17]
Block_Cdr_0_APBmslave0_PWDATA[16]
        Block_Cdr_0_APBmslave0_PWDATA[15] Block_Cdr_0_APBmslave0_PWDATA[14]
Block_Cdr_0_APBmslave0_PWDATA[13]
        Block_Cdr_0_APBmslave0_PWDATA[12] Block_Cdr_0_APBmslave0_PWDATA[11]
Block_Cdr_0_APBmslave0_PWDATA[10]
        Block_Cdr_0_APBmslave0_PWDATA[9] Block_Cdr_0_APBmslave0_PWDATA[8]
Block_Cdr_0_APBmslave0_PWDATA[7]
        Block_Cdr_0_APBmslave0_PWDATA[6] Block_Cdr_0_APBmslave0_PWDATA[5]
Block_Cdr_0_APBmslave0_PWDATA[4]
        Block_Cdr_0_APBmslave0_PWDATA[3] Block_Cdr_0_APBmslave0_PWDATA[2]
Block_Cdr_0_APBmslave0_PWDATA[1]
        Block_Cdr_0_APBmslave0_PWDATA[0] Block_Cdr_0_APBmslave0_PWRITE
Block_Cdr_0_RX_DATA[9]
        Block_Cdr_0_RX_DATA[8] Block_Cdr_0_RX_DATA[7] Block_Cdr_0_RX_DATA[6]
Block_Cdr_0_RX_DATA[5]
        Block_Cdr_0_RX_DATA[4] Block_Cdr_0_RX_DATA[3] Block_Cdr_0_RX_DATA[2]
Block_Cdr_0_RX_DATA[1]
        Block_Cdr_0_RX_DATA[0] Block_Cdr_0_RX_CLK_R Block_Cdr_0/
PF_IOD_CDR_CCC_C0_0/PF_CLK_DIV_0/N_1_inferred_clock_RNI51A9/U0_Y
IRS COretse_Block1_0 Block_Cdr_0 = Block_Cdr_0_APBmslave0_PRDATA[31]
Block_Cdr_0_APBmslave0_PRDATA[30]
        Block_Cdr_0_APBmslave0_PRDATA[29] Block_Cdr_0_APBmslave0_PRDATA[28]
Block_Cdr_0_APBmslave0_PRDATA[27]
        Block_Cdr_0_APBmslave0_PRDATA[26] Block_Cdr_0_APBmslave0_PRDATA[25]
Block_Cdr_0_APBmslave0_PRDATA[24]
        Block_Cdr_0_APBmslave0_PRDATA[23] Block_Cdr_0_APBmslave0_PRDATA[22]
Block_Cdr_0_APBmslave0_PRDATA[21]
        Block_Cdr_0_APBmslave0_PRDATA[20] Block_Cdr_0_APBmslave0_PRDATA[19]
```

```
Block_Cdr_0_APBmslave0_PRDATA[18]
        Block_Cdr_0_APBmslave0_PRDATA[17] Block_Cdr_0_APBmslave0_PRDATA[16]
Block_Cdr_0_APBmslave0_PRDATA[15]
        Block_Cdr_0_APBmslave0_PRDATA[14] Block_Cdr_0_APBmslave0_PRDATA[13]
Block_Cdr_0_APBmslave0_PRDATA[12]
        Block_Cdr_0_APBmslave0_PRDATA[11] Block_Cdr_0_APBmslave0_PRDATA[10]
Block_Cdr_0_APBmslave0_PRDATA[9]
        Block_Cdr_0_APBmslave0_PRDATA[8] Block_Cdr_0_APBmslave0_PRDATA[7]
Block_Cdr_0_APBmslave0_PRDATA[6]
        Block_Cdr_0_APBmslave0_PRDATA[5] Block_Cdr_0_APBmslave0_PRDATA[4]
Block_Cdr_0_APBmslave0_PRDATA[3]
        Block_Cdr_0_APBmslave0_PRDATA[2] Block_Cdr_0_APBmslave0_PRDATA[1]
Block_Cdr_0_APBmslave0_PRDATA[0]
        Block_Cdr_0_APBmslave1_PRDATA[15] Block_Cdr_0_APBmslave1_PRDATA[14]
Block_Cdr_0_APBmslave1_PRDATA[13]
        Block_Cdr_0_APBmslave1_PRDATA[12] Block_Cdr_0_APBmslave1_PRDATA[11]
Block_Cdr_0_APBmslave1_PRDATA[10]
        Block_Cdr_0_APBmslave1_PRDATA[9] Block_Cdr_0_APBmslave1_PRDATA[8]
Block_Cdr_0_APBmslave1_PRDATA[7]
        Block_Cdr_0_APBmslave1_PRDATA[6] Block_Cdr_0_APBmslave1_PRDATA[5]
Block_Cdr_0_APBmslave1_PRDATA[4]
        Block_Cdr_0_APBmslave1_PRDATA[3] Block_Cdr_0_APBmslave1_PRDATA[2]
Block_Cdr_0_APBmslave1_PRDATA[1]
        Block_Cdr_0_APBmslave1_PRDATA[0] Block_Cdr_0_APBmslave2_PRDATA[7]
Block_Cdr_0_APBmslave2_PRDATA[6]
        Block_Cdr_0_APBmslave2_PRDATA[5] Block_Cdr_0_APBmslave2_PRDATA[4]
Block_Cdr_0_APBmslave2_PRDATA[3]
        Block_Cdr_0_APBmslave2_PRDATA[2] Block_Cdr_0_APBmslave2_PRDATA[1]
Block_Cdr_0_APBmslave2_PRDATA[0]
        COretse_Block1_0_TCG[9] COretse_Block1_0_TCG[8] COretse_Block1_0_TCG[7]
        COretse_Block1_0_TCG[6] COretse_Block1_0_TCG[5] COretse_Block1_0_TCG[4]
        COretse_Block1_0_TCG[3] COretse_Block1_0_TCG[2] COretse_Block1_0_TCG[1]
        COretse_Block1
```

MSVT Output Report Section

When executing MSVT with the example parameter file as an input, MSVT fails and reports the following errors, as shown in the following example:

- PHY_RST_c
- coma_mode_c
- LINK_OK_c
- PHY_RST_0_c
- RD_BC_ERROR_c
- SPISCLKO_c
- SPISDO_c
- SPISS_c
- TX_c
- coma_mode_0_c

The top-level design must contain only blocks, or at least very few other instances. The errors in this example are most likely caused by ‘others’ on top level. The nets connecting blocks to ‘others’ are IRS to the blocks, but not listed in the `msvt.param` file.

Example 4-3. Checking IRS Connectivity Against a Parameter File

```
-----------------------------------------------------
Checking IRS connectivity against parameter file
=====================================================
   Error: IRS net PHY_RST_c is not listed in param file
   Error: IRS net coma_mode_c is not listed in param file
   Error: IRS net LINK_OK_c is not listed in param file
   Error: IRS net PHY_RST_0_c is not listed in param file
   Error: IRS net RD_BC_ERROR_c is not listed in param file
   Error: IRS net SPISCLKO_c is not listed in param file
   Error: IRS net SPISDO_c is not listed in param file
   Error: IRS net SPISS_c is not listed in param file
```

```
   Error: IRS net TX_c is not listed in param file
   Error: IRS net coma_mode_0_c is not listed in param file
-----------------------------------------------------------------
```

4.4.2 Checking a Block Instance Assignment to a Routing Region

MSVT checks the assignment of all instances of each block in the design to a routing region. The regions must be defined with `-route true` to constrain routing. If a block instance is not assigned to a routing region, MSVT lists the instance name in the `Following instances do not belong to any routing region` section of the report. If all instances are assigned to a separation region, the MSVT report omits this section.

Note: This is an informational section meant to identify instances that are not assigned to any region. The identified instances will not be considered as errors.

Sample Design Constraints File

The following sample physical constraints code constrains a PolarFire design within separation routing regions.

Tip:
For more information about PDC constraints, see the *PDC Commands User Guide*.

Example 4-4. PDC Constraints of a Sample Design for PolarFire

```
define_region -region_name Block1region -type exclusive -color 2143338688 -
route true -push_place false -x1 456 -y1 195 -x2 1631 -y2 371
define_region -region_name Block2region -type exclusive -color 2143338688 -
route true -push_place false -x1 1752 -y1 189 -x2 2435 -y2 377
define_region -region_name Block3region -type exclusive -color 2143338688 -
route true -push_place false -x1 0 -y1 0 -x2 335 -y2 41 -x1 0 -y1 42 -x2 1067
-y2 161 -x1 804 -y1 0 -x2 1067 -y2 41
define_region -region_name Block4region -type exclusive -color 2143338688 -
route true -push_place false -x1 1200 -y1 0 -x2 2351 -y2 158
define_region -region_name SMIPregion -type exclusive -color 2143338688 -route
true -push_place false -x1 384 -y1 0 -x2 755 -y2 11
define_region -region_name IBR2_4 -type inclusive -color 2147442270 -route
true -push_place false -x1 2148 -y1 105 -x2 2327 -y2 266
define_region -region_name IBR3_4 -type inclusive -color 2147442270 -route
true -push_place false -x1 888 -y1 45 -x2 1463 -y2 98
define_region -region_name IBR1_4 -type exclusive -color 2143338688 -route
true -push_place false -x1 1356 -y1 126 -x2 1499 -y2 245
define_region -region_name IBR1_3 -type inclusive -color 2147442270 -route
true -push_place false -x1 636 -y1 102 -x2 851 -y2 239
define_region -region_name IBR1_2 -type inclusive -color 2147442270 -route
true -push_place false -x1 1584 -y1 282 -x2 2027 -y2 362
assign_region -region_name Block1region -inst_name block1_0
assign_region -region_name Block2region -inst_name block2_0
assign_region -region_name Block3region -inst_name block3_0
assign_region -region_name Block4region -inst_name RESETN_ibuf
assign_region -region_name Block4region -inst_name block4_0
assign_region -region_name SMIPregion -inst_name pf_smip_0
assign_net_macros -region_name IBR2_4 -net_name block4_0_Y -include_driver true
assign_net_macros -region_name IBR3_4 -net_name block3_0_dataOut\[31\] -
include_driver true
assign_net_macros -region_name IBR3_4 -net_name block3_0_dataOut\[30\] -
include_driver true
assign_net_macros -region_name IBR3_4 -net_name block3_0_dataOut\[29\] -
include_driver true
assign_net_macros -region_name IBR3_4 -net_name block3_0_dataOut\[28\] -
include_driver true
```

Example 4-5. PDC Constraints of a Sample Design for SmartFusion2 and IGLOO2

```
-------------------------------------------------------------------------------
-------------------------------------------------------
define_region -name UserRegion0 -type inclusive -color 8388736 -route YES -
push_place YES 0 93 143 146
define_region -name UserRegion2 -type inclusive -color 12639424 -route YES -
push_place YES 240 0 443 62
define_region -name UserRegion3 -type inclusive -color 15780518 -route YES -
push_place YES 0 0 143 65
define_region -name UserRegion4 -type inclusive -color 16735838 -route YES -
push_place YES 108 111 263 131
define_region -name UserRegion5 -type inclusive -color 975928 -route YES -
push_place YES 324 21 371 119
define_region -name UserRegion6 -type inclusive -color 65535 -route YES -
push_place YES 96 9 263 47
define_region -name UserRegion1 -type inclusive -color 32896 -route YES -
push_place YES 240 96 443 146 276 201 359 206 324 138 347 206
assign_region UserRegion0 U_ST1/DCT8AAN1_0*
assign_region UserRegion0 U_ST1/INBUF_2*
assign_region UserRegion0 U_ST1/INBUF_3*
assign_region UserRegion0 U_ST1/IO_0*
assign_region UserRegion0 U_ST1/CFG0_GND_*
assign_region UserRegion1 U_B1/DCT_BUF_10_0*
assign_region UserRegion1 U_B1/INBUF_2*
assign_region UserRegion2 U_ST2/DCT8AAN2_0*
assign_region UserRegion2 U_ST2/INBUF_2*
assign_region UserRegion2 U_ST2/CFG0_GND*
assign_region UserRegion3 U_B2/DCT_BUF_12_0*
assign_region UserRegion3 U_B2/INBUF_2*
assign_region UserRegion3 U_B2/IO_0*
assign_region UserRegion3 U_B2/OUTBUF_0*
assign_net_macros UserRegion4 data1<* -include_driver YES
assign_net_macros UserRegion4 rdy1
assign_net_macros UserRegion5 data1r<* -include_driver YES
assign_net_macros UserRegion5 rdy2
assign_net_macros UserRegion6 data2<* -include_driver YES
assign_net_macros UserRegion6 rdy3 -include_driver YES
-------------------------------------------------------------------------------
-------------------------------------------------------
```

MSVT Output Report

MSVT reports all instances of the design that are not included in any routing regions, as shown in the following example.

Example 4-6. Instances That Do Not Belong to Any Routing Region

```
-------------------------------------------------------------------------------
-------------------------------------------------------
The following instances do not belong to any routing region:
=============================================================
   PF_CCC_C0_0/PF_CCC_C0_0/pll_inst_0
   REF_CLK_0_ibuf/U_IOIN
-------------------------------------------------------------------------------
-------------------------------------------------------
```

4.4.3 Checking an IRS Net Assignment to a Routing Region

MSVT checks whether all IRS nets of a design are assigned to a routing region. MSVT lists all instances of IRS nets that are not assigned to a routing region in the `Following IRS nets are not constrained by any routing region` section of the report. If all of the IRS nets are assigned to an IRS routing region, the MSVT report omits this section.

Note: This is an informational section meant to identify nets that are not assigned to IRS region and does not cause MSVT to fail.

MSVT Output Report

MSVT reports the nets that are constrained by routing region, as shown in the following example.

Example 4-7. IRS Nets That are Not Constrained by Any Routing Region

```
The following IRS nets are not constrained by any routing region:
=================================================================
block4_0_TX
block1_0_TX
```

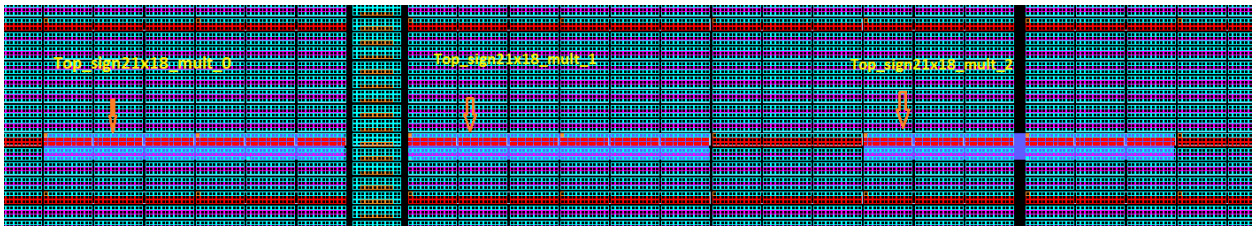
4.4.4 Checking Cascaded Math Block Instance Adherence to the Separation Criteria

MSVT checks if all of the cascaded Math block instances adhere to the separation criteria. If there are any cascaded Math block instances that violate the separation criteria, then they are listed in the `Input signals of the following Math block instances can be observed by failure of config switches in cascade chain` section of the report.

Design Example

The following is a chip planner snap shot of a design in which `Top_sign21x18_mult_0`, `Top_sign21x18_mult_1` and `Top_sign21x18_mult_2` are three Math blocks. `Top_sign21x18_mult_0` is adjacent to `Top_sign21x18_mult_1`, whereas `Top_sign21x18_mult_2` is separated by `Top_sign21x18_mult_1` with one `MATH` cluster.

Figure 4-1. Chip Planner View of a Sample Design



MSVT Output Report

Because `Top_sign21x18_mult_0` is adjacent to `Top_sign21x18_mult_1`, MSVT fails and reports an error for these `MACC` instances, as shown in the following. However, as`Top_sign21x18_mult_2` is separated from `Top_sign21x18_mult_1` by at least one `MATH` cluster, it is not identified as an error by MSVT, as shown in following report section example.

Example 4-8. Input signals of the following Math block instances can be observed by failure of the config switches in the cascade chain

```
------------------------------------------------------------------------------
------------------------------------------------------------
Input signals of the following Math block instances can be observed by failure
of the config switches in the cascade chain:
==============================================================================
================================== Block1_rom_0/macc_rom_top/MACC_PA_BC_ROM_5/
MACC_PHYS_0/INST_MACC_IP of block Block1_rom_0 can be observed by Math block
instance Block0_0/block0_level_0/MACC_PA_8/MACC_PHYS_INST/INST_MACC_IP of
block Block0_0
------------------------------------------------------------------------------
------------------------------------------------------------
```

4.4.5 Analyzing a Floorplan

MSVT extracts the separation between each pair of blocks in the design, which may help identify violations in the floorplan. For each pair of blocks in a design, the following information is generated in the `Analyzing floorplan` section of the report. This information can be used to identify, which blocks lack sufficient cluster separation between them that can lead to an MSVT failure.

- **Floorplan Separation**: Shows the minimum separation between the respective routing regions in cluster units. If there is no overlap between the X and Y dimensions of the two regions, this is indicated as “diagonal.”
- **Placement Separation**: Shows the minimum separation between the actual placements of instances in each block in cluster units. If there is no overlap between the X and Y dimensions of the placements of instances of the two blocks, this is indicated as “diagonal.”

Note: The coordinates shown for blocks is presented in terms of clusters.

Design Example
For example, consider blocks `block4_0` and `block2_0` separated in MVN, as shown in the following figure.

Figure 4-2. Chip Planner View of Design

MSVT Output Report
In the sample report section shown in the following, the floorplan separation indicates that the regions to which blocks `block4_0` and `block2_0` are assigned are separated by 9 clusters.

Placement separation between instances of blocks `block4_0` and `block2_0` is placed in 21 clusters. `block4_0` at (144,62) suggests that the `block4_0` region spans from cluster 144 in the X-direction and cluster 62 in the Y-direction.

Example 4-9. Analyzing Floorplan

```
--------------------------------------------------------------------------------
------------------------------------------------------------
Analyzing  floorplan ...
========================

   block4_0 and block2_0 : Minimal floorplan separation = 9 clusters.
      block4_0 at cluster (144,62)
      block2_0 at cluster (144,52)
   block4_0 and block2_0 : Minimal placement separation = 21 clusters.
      (2148,156) containing cell block4_0/BUFD_1/U0
      (2148,225) containing cell block2_0/BUFD_0/U0

   block4_0 and block3_0 : Minimal floorplan separation = 11 clusters.
      block4_0 at cluster (99,27)
      block3_0 at cluster (87,27)
   block4_0 and block3_0 : Minimal placement separation = 11 clusters.
      (1211,82) containing cell block4_0/CoreGPIO_C4_0/CoreGPIO_C4_0/
inData_s2[6]
      (1057,81) containing cell block3_0/APB_dp_fp_1/U0/i_post_norm_mul/
s_shl2_RNIS34841[4]
--------------------------------------------------------------------------------
------------------------------------------------------------
```

4.4.6 Checking Internal Nets for a Given Block

MSVT checks the separation of internal nets corresponding to a given block from external nets as per the specified separation criteria. If any of the nets of the design fails to satisfy separation criteria, then information related to the violating net is listed in the `Checking internal nets for block <block name>` section of the report and MSVT treats this as an error. This section is empty if your design does not have any net violating separation criteria.

MSVT Output Report

The following example shows that the internal net `block4_0/MIV_RV32IMC_C2_0/MIV_RV32IMC_C2_0/u_opsrv_0/un2_apb_mstr_int_sel` of the `block4_0` block is failing separation criteria and the net can access untrusted net `block3_0/APB_dp_fp_1/U0/i_post_norm_mul/s_frac2a_3_157` through net `block3_0/APB_dp_fp_1/U0/i_post_norm_mul/m16_4_03_0` through switches present at the specified coordinates.

Example 4-10. Checking internal nets for block block4_0

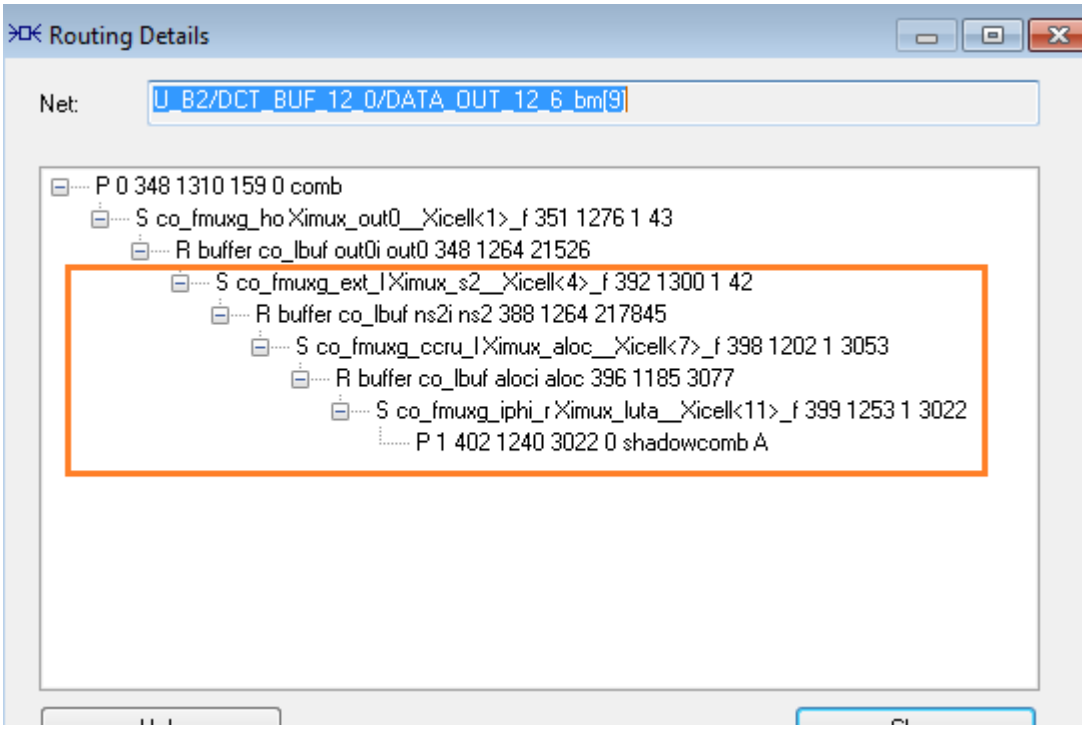
```
------------------------------------------------------------------------------
------------------------------------------------------------
Checking internal nets for block block4_0 ...
============================================================================

   Net block4_0/MIV_RV32IMC_C2_0/MIV_RV32IMC_C2_0/u_opsrv_0/
un2_apb_mstr_int_sel
      can be observed by cell block3_0/APB_dp_fp_1/U0/i_post_norm_mul/
s_frac2a_3_157
      through net block3_0/APB_dp_fp_1/U0/i_post_norm_mul/m16_4_03_0
      with cluster separation (13,0) due to 2 failing switches:
      When a routed Mux at (1185,81) fails
      The path is also driven by  constant '1'  signal
         Through an existing routed Buffer at (1185,81)
         Through an existing routed Buffer at (1113,81)
      When a routed Mux at (1053,81) fails
      The path is also driven by  block3_0/APB_dp_fp_1/U0/i_post_norm_mul/
m16_4_03_0 signal
         Through an existing routed Buffer at (1053,81)
         Through an existing routed Mux at (1055,96)
         Through an existing routed Buffer at (1055,96)
         Through an existing routed Mux at (1024,96)
         Through an existing routed Mux at (1024,96)
         Through an existing routed Buffer at (1024,96)
         Through an existing routed Mux at (1026,96)
------------------------------------------------------------------------------
------------------------------------------------------------
```

Routing Details

You can view the routing details of the untrusted net (“U_B2/DCT_BUF_12_0/DATA_OUT_12_6_bm[9]“ in the preceding example) by viewing routing details from ChipPlanner till the point where the switch is in the ON state.

Figure 4-3. Routing Details of Corresponding Net from Chip Planner



4.4.7 Checking IRS Nets for a Given Block

MSVT checks for the separation of IRS nets corresponding to a given block from external nets, per the specified separation criteria. If any of the nets of the design fail to satisfy the separation criteria, then information related to the violating net is listed in the `Checking IRS nets for block <block name>` section of the report and MSVT treats this as an error. This section is empty if your design does not have any nets violating separation criteria.

MSVT Output Report

The following is an example of this section in which the IRS net `PHY_RST_c` of the block `Block_Cdr_0` is being observed by an external net corresponding to another untrusted logic

Example 4-11. Checking IRS nets for block Block_Cdr_0

```
------------------------------------------------------------------------------
------------------------------------------------------------
Checking IRS nets for block Block_Cdr_0 ...
============================================================
The following outgoing IRS nets have 0 switches separation since they are
connected directly to at least one untrusted logic:
   PHY_RST_c (cell PHY_RST_obuf/U_IOTRI)
   coma_mode_c (cell coma_mode_obuf/U_IOTRI)
------------------------------------------------------------------------------
------------------------------------------------------------
```

4.4.8 Propagating IRS Nets Outgoing from <Block1> to <Block2>

MSVT checks the separation of the IRS nets corresponding to a given block from external nets, per the specified separation criteria. If any of the nets of the design fail to satisfy separation criteria, then information related to the violating net is listed in the `Propagating IRS nets outgoing from <block name> to <block name>` section of the report and MSVT treats this as an error. This section is empty if your design does not have any net violating separation criteria.

MSVT Output Report

The following example shows that the IRS net block4_0_TX connecting block4_0and block1_0 instances is failing to meet separation criteria. This net can be observed by an external net block3_0/APB_dp_fp_1/U0/i_post_norm_mul/un3_s_ine_o_1_0_21_Z of block block3_0.

Example 4-12. Propagating IRS nets outgoing from block4_0 to block1_0

```
------------------------------------------------------------------------------
------------------------------------------------------------
Propagating IRS nets outgoing from block4_0 to block1_0
============================================================================

   Net block4_0_TX
      can be observed by cell block3_0/APB_dp_fp_1/U0/i_post_norm_mul/
un3_s_ine_o_1_0_29
      through net block3_0/APB_dp_fp_1/U0/i_post_norm_mul/un3_s_ine_o_1_0_21_Z
      with cluster separation (13,0) due to 3 failing switches:
      When a routed Mux at (1269,93) fails
      The path is also driven by  constant '1'  signal
         Through an existing routed Buffer at (1269,93)
         Through an existing routed Buffer at (1197,93)
      When a routed Mux at (1137,93) fails
      The path is also driven by  constant '1'  signal
         Through an existing routed Buffer at (1137,93)
         Through an existing routed Buffer at (1065,93)
      When a routed Mux at (1000,93) fails
      The path is also driven by  block3_0/APB_dp_fp_1/U0/i_post_norm_mul/
un3_s_ine_o_1_0_21_Z signal
         Through an existing routed Buffer at (1000,93)
         Through an existing routed Mux at (990,93)
         Through an existing routed Mux at (990,93)
         Through an existing routed Buffer at (990,93)
         Through an existing routed Mux at (990,93)
         Through an existing routed Mux at (990,93)
------------------------------------------------------------------------------
------------------------------------------------------------
```

The example report indicates that an IRS signal through the net block4_0_TX between blocks block4_0 and block1_0 is not separated by the number of required switches specified in the DESIGN_SEPARATION parameter from another net un3_s_ine_o_1_0_21_Z, which is a part of the block3_0 block.

If parameter REGIONS_VERBOSITY is set to ‘1’, then MSVT outputs the following additional information related to the floorplan. These sections provide additional information related to the design and cannot be considered as errors.

- 4.4.8.1 Region Constraints Associated with Block <Block_Name>
- 4.4.8.2 Empty Regions in the MSVT Output Report

4.4.8.1 Region Constraints Associated with Block <Block_Name>

MSVT describes in detail all the block instances associated with a given separation region. If the separation region is rectilinear, each sub-rectangular region is analyzed for block instances.

MSVT Output Report

The following example shows a section where block4_0 and block2_0 are part of a rectilinear separation region. The report shows two region constraints associated with block4_0, each with coordinates of two sub-rectangular regions corresponding to the rectilinear region.

Example 4-13. MSVT Output Report Section

```
------------------------------------------------------------------------------
------------------------------------------------------------
The following region constraints are associated with blocks: block4_0 block2_0
==============================================================================
( INCLUSIVE REGION
        ( RECT 2148 105 2328 267)
        ( CELLS
                block2_0/BUFD_0/U0
                block4_0/BUFD_1/U0
```

```
        )
)

The following region constraints are associated with blocks: block4_0 block3_0
================================================================================
( INCLUSIVE REGION
        ( RECT 888 45 1464 99)
        ( CELLS
                block3_0/CoreGPIO_C1_0/CoreGPIO_C1_0/dataOut_Z[0]
                block3_0/CoreGPIO_C1_0/CoreGPIO_C1_0/dataOut_Z[1]
                block3_0/CoreGPIO_C1_0/CoreGPIO_C1_0/dataOut_Z[2]
                .
                .
                .
                .
                block4_0/CoreGPIO_C4_0/CoreGPIO_C4_0/inData_s1[30]
                block4_0/CoreGPIO_C4_0/CoreGPIO_C4_0/inData_s1[31]
        )
)

.
.
.
.
The following region constraints are associated with blocks: block2_0 block1_0
================================================================================
( INCLUSIVE REGION
        ( RECT 1584 282 2028 363)
        ( CELLS
                block1_0/BUFD_1/U0
                block1_0/BUFD_2/U0
                block1_0/BUFD_3/U0
                block1_0/BUFD_4/U0
                .
                .
                .
                .
                block2_0/BUFD_67/U0
                block2_0/BUFD_68/U0
        )
)

The following region constraints are associated with blocks: block3_0 block1_0
================================================================================
( INCLUSIVE REGION
        ( RECT 636 102 852 240)
        ( CELLS
                block1_0/BUFD_22/U0
                block1_0/BUFD_36/U0
                block1_0/BUFD_37/U0
                block1_0/BUFD_38/U0
                .
                .
                .
                .
                block3_0/CoreGPIO_C2_0/CoreGPIO_C2_0/dataOut[6]
                block3_0/CoreGPIO_C2_0/CoreGPIO_C2_0/dataOut[7]
        )
)
--------------------------------------------------------------------------------
------------------------------------------------------------
```

4.4.8.2 Empty Regions in the MSVT Output Report

This section of the MSVT output report shows the empty regions. These regions should also be defined as Routing Regions. This section is generated only if there are such empty routing regions in the design.

MSVT Output Report

The following is a Chip Planner view of the input design having three empty Routing Regions defined.

Figure 4-4. Chip Planner View of Example Design

Example 4-14. MSVT Output Report Section

```
--------------------------------------------------------------------------------
------------------------------------------------------------
The following are empty region(s):
================================================================================
( EXCLUSIVE REGION
    ( RECT 1476 6 1584 84)
)
--------------------------------------------------------------------------------
------------------------------------------------------------
```

5. Report Conclusion

When MSVT completes successfully, the report shows the following final message:

```
Design has met 2 switches separation requirement
MSVT Check succeeded.
Number of errors: 0
```

Note: In the message above, `number of errors` shows the total number of errors that MSVT reported in the output report.

The report shows the following final message for MSVT separation criteria failure:

```
Design failed for 2 switches separation requirement
MSVT Check failed.
Number of errors: 7
```

You can now program your FPGA with the generated programming file.

6. Reference Documents

To implement a design using Design Separation Methodology, see the following documents:

- Microchip Design Separation Methodology
- SmartFusion2 and IGLOO2 Block Flow User's Guide
- SmartFusion2 and IGLOO2 SmartTime, I/O Editor, and Chip Planner User's Guide

7. Revision History

Revision	Date	Description
B	08/2021	This document is released with Libero SoC Design Suite v2021.2 without changes from v2021.1.
A	04/2021	Initial Revision

8. Microchip FPGA Technical 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. This section provides information about contacting Microchip FPGA Products Group and using these support services.

8.1 Customer Service

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

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

8.2 Customer Technical Support

Microchip FPGA Products Group staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions about Microchip FPGA Products. The Customer Technical Support Center spends a great deal of time creating application notes, answers to common design cycle questions, documentation of known issues, and various FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

You can communicate your technical questions through our Web portal and receive answers back by email, fax, or phone. Also, if you have design problems, you can upload your design files to receive assistance. We constantly monitor the cases created from the web portal throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

Technical support can be reached at soc.microsemi.com/Portal/Default.aspx.

For technical support on RH and RT FPGAs that are regulated by International Traffic in Arms Regulations (ITAR), log in at soc.microsemi.com/Portal/Default.aspx, go to the **My Cases** tab, and select **Yes** in the ITAR drop-down list when creating a new case. For a complete list of ITAR-regulated Microchip FPGAs, visit the ITAR web page.

You can track technical cases online by going to My Cases.

8.3 Website

You can browse a variety of technical and non-technical information on the Microchip FPGA Products Group home page, at www.microsemi.com/soc.

8.4 Outside the U.S.

Customers needing assistance outside the US time zones can either contact technical support at (https://soc.microsemi.com/Portal/Default.aspx) or contact a local sales office.

Visit About Us for sales office listings and corporate contacts.

The Microchip Website

Microchip provides online support via our website at www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: www.microchip.com/support

Microchip Devices Code Protection Feature

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

- 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 and under normal conditions.
- There are dishonest and possibly illegal methods being used in attempts to breach the code protection features of the Microchip devices. We believe that these methods require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Attempts to breach these code protection features, most likely, cannot be accomplished without violating Microchip's intellectual property rights.
- Microchip is willing to work with any customer who is concerned about the integrity of its code.
- 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. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Trademarks

ISBN: 978-1-5224-8520-9

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: www.microchip.com/support Web Address: www.microchip.com **Atlanta** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **Austin, TX** Tel: 512-257-3370 **Boston** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **Chicago** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **Dallas** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **Detroit** Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983 **Indianapolis** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4485-5910 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-72400 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-72884388 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820