Product Version ICADVM20.1 October 2020 © 2020 Cadence Design Systems, Inc. All rights reserved. Printed in the United States of America.

Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA.

Trademarks: Trademarks and service marks of Cadence Design Systems, Inc. contained in this document are attributed to Cadence with the appropriate symbol. For queries regarding Cadence's trademarks, contact the corporate legal department at the address shown above or call 800.862.4522. All other trademarks are the property of their respective holders.

Restricted Permission: This publication is protected by copyright law and international treaties and contains trade secrets and proprietary information owned by Cadence. Unauthorized reproduction or distribution of this publication, or any portion of it, may result in civil and criminal penalties. Except as specified in this permission statement, this publication may not be copied, reproduced, modified, published, uploaded, posted, transmitted, or distributed in any way, without prior written permission from Cadence. Unless otherwise agreed to by Cadence in writing, this statement grants Cadence customers permission to print one (1) hard copy of this publication subject to the following conditions:

- 1. The publication may be used only in accordance with a written agreement between Cadence and its customer.
- 2. The publication may not be modified in any way.
- 3. Any authorized copy of the publication or portion thereof must include all original copyright, trademark, and other proprietary notices and this permission statement.
- 4. The information contained in this document cannot be used in the development of like products or software, whether for internal or external use, and shall not be used for the benefit of any other party, whether or not for consideration.

Disclaimer: Information in this publication is subject to change without notice and does not represent a commitment on the part of Cadence. Except as may be explicitly set forth in such agreement, Cadence does not make, and expressly disclaims, any representations or warranties as to the completeness, accuracy or usefulness of the information contained in this document. Cadence does not warrant that use of such information will not infringe any third party rights, nor does Cadence assume any liability for damages or costs of any kind that may result from use of such information.

Restricted Rights: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor

Contents

<u>Preface</u>	. 9
Scope	10
<u>icensing Requirements</u>	10
Related Documentation	11
What's New and KPNS	11
Installation, Environment, and Infrastructure	11
Technology Information	11
<u>Virtuoso Tools</u>	11
Additional Learning Resources	12
Video Library	12
Virtuoso Videos Book	12
Rapid Adoption Kits	12
Help and Support Facilities	13
Customer Support	13
Feedback about Documentation	13
Typographic and Syntax Conventions	14
<u>1</u>	
Design Rule Driven Editing	17
Turning on DRD Editing	18
Setting Up DRD Options	
DRD Configuration File	
DRD Operating Modes	
<u>Using Enforce Mode</u>	
<u>Using Notify Mode</u>	
Using Post Edit Mode	
Ignoring Violations inside Instances	
Severity of DRD Markers	
Selecting Constraints for Verification	
Selecting Layers for Verification	

Performing Color Checks (ICADVM20.1 Only)	38
Color Constraints Checks	
Color-Related Flow Checks	
DRD Editing and Blockage Objects	
Using DRD Editing in VLS XL, VLS GXL, and VLS EXL	
Batch-Check Mode	
<u>Verifying Selected Objects</u>	
<u>Verifying Nets</u>	
<u>2</u>	
DRD Advanced Features 4	19
DRD Targets 5	50
Setting DRD Targets	
Unsetting DRD Targets	
Bindkey Reassignment	
Targeted Enforce	
Relaxed Enforce	31
Deferred Post Edit	32
What-if Halos	32
What-if Functionality for Hierarchical Objects	34
DRD Smart Snapping 6	
Snapping to Halos and Discrete Spacing Values6	36
Snapping to a What-if Halo around a DRD Target6	39
Simultaneous Snapping between Two Orthogonal Edges	71
Layer-to-Layer Snapping 7	72
Snapping Based on the minArea Constraint7	73
Snapping Using the Quick Align Form in Virtuoso	⁷ 4
Snapping with DRD Enforce Enabled7	7 6
Toggling DRD Smart Snap during Interactive Editing7	
Incremental Violation Display 7	
Excluding Instances from Checking	'8
<u>3</u>	
DRD Interactive Compactor	⁷ 9
Overview of DRD Compactor Features	30

Prerequisites for Using the Compactor	81
Launching the DRD Compactor Options Form	
Compactor User Interface	
The DRD Compactor Options Form	83
The DRD Compactor Toolbar and Bindkeys	87
Running the Compactor	
Using the Compactor GUI	88
Using SKILL Functions and Bindkeys	89
Examples of Compaction	
<u>4</u>	
DRD Support for FinFET Devices (ICADVM20.1 Only)	105
Defining Discrete Spacing Rules	
Defining Minimum Spacing Rules	
Defining Directional and Discrete Width Rules	
Defining Directional and Discrete PR Boundary Rules	
Defining Directional and Discrete PR Boundary Width Rules	
Defining Directional and Discrete PR Boundary Spacing Rules	
Snapping Objects to Legal Width Values	
Examples of Discrete Dimension Gravity	
Defining Directional and Discrete Opposite Extension Rules	
<u>5</u>	
DRD Constraints Support	127
Supported Constraints	128
Supported Constraints and Parameters (IC6.1.8 and ICADVM20.1)	
Supported Constraints and Parameters (ICADVM20.1 Only – 95511 and 95800)	
Supported Constraints and Parameters (ICADVM20.1 Only – 95512)	
Supported Constraints and Parameters (ICADVM20.1 Only – 95511)	
Using DRD Grid Checking	
Specifying Layers in the Interconnect Section	
Supported Derived Layers	
Supported Layer-Purpose Pair and Voltage-Dependent Constraints	
Support for Layer-Purpose Pair Constraints	
Support for Voltage-Dependent Constraints	161

Using Density Constraints 164 Using Alternate Foundry Constraint Group 167
A
DRD Form Descriptions
DRD Compactor Options
<u>DRD Options</u>
<u> </u>
В
DRD Environment Variables
<u>List of Environment Variables</u>
drdBatchColorability
drdBatchColorShorts
drdBatchConsiderGrayShapes
drdBatchConsiderUnlockedShapesAsGray
drdBatchFoundryRulesOnly
drdBatchReportCoveredNonPcellGrayShapes
drdBatchScopeAreaLimit
drdBatchScopeSpecifiedArea
drdBatchUncoloredShapes
drdBatchUnlocked
drdBatchVioLimit
drdColorability
drdColorShorts
drdConsiderGrayShapes
drdConsiderUnlockedShapesAsGray
<u>drdEditAllowOrthogonalAdjust</u> 198 <u>drdEditAutoText</u> 199
drdEditAutoTurnOffHalo
drdEditAutoTurnOffHaloLimit
drdEditBatchColorRules
drdEditBatchHierDepth
drdEditColorRules
<u>drdEditCompactorActiveLayerOnly</u>
drdEditCompactorAnchorMode

<u>drdEditCompactorInterOnly</u>	
<u>drdEditCompactorMode</u>	
<u>drdEditCompactorSpacingMode</u>	209
<u>drdEditCompactorUserSpacing</u>	
<u>drdEditDensityTypeCheck</u>	211
<u>drdEditDisplayArrows</u>	
<u>drdEditDisplayDashed</u>	213
<u>drdEditDisplayDrawingColor</u>	214
<u>drdEditDisplayEdges</u>	215
drdEditDisplayHalo	216
drdEditDisplayMarkers	217
<u>drdEditDisplayMaxChars</u>	218
<u>drdEditDisplayTechRefs</u>	219
drdEditDisplayText	220
<u>drdEditDisplayTrueColor</u>	221
<u>drdEditEnableBatchCheckOnReadOnlyDesign</u>	222
<u>drdEditFontSize</u>	223
<u>drdEditHierDepth</u>	224
<u>drdEditIgnoreIntraInstanceChecks</u>	225
<u>drdEditMode</u>	226
<u>drdEditPaletteLayers</u>	227
<u>drdEditSlidingWindowSize</u>	228
<u>drdEditSmartSnapAllowedWidthSnap</u>	229
<u>drdEditSmartSnapAperture</u>	230
<u>drdEditVioLimit</u>	231
<u>drdInteractiveFoundryRulesOnly</u>	232
<u>drdPostEditAllWindows</u>	233
<u>drdPostEditEIPAllInstances</u>	234
<u>drdPostEditEIPAllWindows</u>	235
drdPostEditEIPCheckTopView	236
<u>drdPostEditIgnoreModgen</u>	237
drdPostEditReplaceAllMarkers	238
drdPostEditTimeOut	239
<u>drdPostEditVioLimit</u>	
drdUncoloredShapes	241
drdUnlocked	

drdUseBlockageForExtCheck	243
drdUseNetName	244
drdUseNewDerivedLayerEngine	245
drdValidLayersGrayCheck	246
drdVDRIncludePurposes	
maxShapesAllowedForCompaction	249
<u>C</u>	
Blockage and Boundary Objects	251
DRD Editing and Blockage Objects	251
Effective Width	251
DRD Editing and PR Boundary Objects	254
Creating and Editing Non-PR Boundary Objects	255

Preface

This guide describes how to use the Virtuoso® Design Rule Driven (DRD) Editing functionality, which provides visual and physical feedback about design constraint violations while you are interactively creating and editing design operation in Virtuoso Layout Suite tools.

This user guide is aimed at developers and designers of integrated circuits and assumes that you are familiar with:

- The Virtuoso design environment and application infrastructure mechanisms designed to support consistent operations between all Cadence tools.
- The applications used to design and develop integrated circuits in the Virtuoso design environment, notably, the Virtuoso Layout Suite and the Virtuoso Schematic Editor.
- The Virtuoso design environment technology file.

This preface contains the following topics:

- Scope
- Licensing Requirements
- Related Documentation
- Additional Learning Resources
- Customer Support
- Feedback about Documentation
- Typographic and Syntax Conventions

Scope

Unless otherwise noted, the functionality described in this guide can be used in both mature node (for example, IC6.1.8) and advanced node and methodologies (for example, ICADVM20.1) releases.

Label	Meaning
(IC6.1.8 Only)	Features supported only in mature node releases.
(ICADVM20.1 Only)	Features supported only in the ICADVM20.1 advanced nodes and advanced methodologies release.
(ICADVM20.1 Only – 95512)	Features supported only in the ICADVM20.1 release and which require the Virtuoso_Adv_Node_Opt_Lay_Std license (95512).
(ICADVM20.1 Only – 95511)	Features supported only in the ICADVM20.1 release and which require the Virtuoso_Adv_Node_Opt_Layout license (95511).
(ICADVM20.1 Only – 95511 and 95800)	Features supported only in the ICADVM20.1 release and which require the Virtuoso_Adv_Node_Opt_Layout (95511) and Virtuoso_Layout_Suite_EXL (95800) licenses.

Licensing Requirements

In IC6.1.8, DRD requires one of the following licenses:

- Virtuoso_Layout_Suite_XL
- Virtuoso_Layout_Suite_GXL
- Virtuoso_Layout_Suite_EAD

In ICADVM20.1, DRD requires either:

- A Virtuoso_Layout_Suite_XL license
- A Virtuoso_Layout_Suite_EXL license

For more detailed information on licensing in the Virtuoso design environment, see <u>Virtuoso Software Licensing and Configuration User Guide</u>.

Related Documentation

What's New and KPNS

- <u>Virtuoso Design Rule Driven Editing What's New</u>
- <u>Virtuoso Design Rule Driven Editing Known Problems and Solutions</u>

Installation, Environment, and Infrastructure

- Cadence Installation Guide
- <u>Virtuoso Design Environment User Guide</u>
- <u>Virtuoso Design Environment SKILL Reference</u>
- Cadence Application Infrastructure User Guide

Technology Information

- <u>Virtuoso Technology Data User Guide</u>
- <u>Virtuoso Technology Data ASCII Files Reference</u>
- Virtuoso Technology Data Constraints Reference
- Virtuoso Technology Data SKILL Reference

Virtuoso Tools

- Virtuoso Lavout Suite L User Guide
- Virtuoso Layout Viewer User Guide
- Virtuoso Space-based Router User Guide
- <u>Virtuoso Multi-Patterning Technology User Guide</u> (ICADVM20.1 Only)
- Virtuoso Layout Suite SKILL Reference

Additional Learning Resources

Video Library

The <u>Video Library</u> on the Cadence Online Support website provides a comprehensive list of videos on various Cadence products.

To view a list of videos related to a specific product, you can use the *Filter Results* feature available in the pane on the left. For example, click the *Virtuoso Layout Suite* product link to view a list of videos available for the product.

You can also save your product preferences in the Product Selection form, which opens when you click the *Edit* icon located next to *My Products*.

Virtuoso Videos Book

You can access certain videos directly from Cadence Help. To learn more about this feature and to access the list of available videos, see <u>Virtuoso Videos</u>.

Rapid Adoption Kits

Cadence provides a number of <u>Rapid Adoption Kits</u> that demonstrate how to use Virtuoso applications in your design flows. These kits contain design databases and instructions on how to run the design flow.

In addition, Cadence offers the following training courses that cover Design Rule Driven (DRD) editing and DRD-based interactive compactor features:

- Virtuoso Layout Design Basics
- Virtuoso Layout Pro: T4 Advanced Commands (XL)

To explore the full range of training courses provided by Cadence in your region, visit Cadence Training or write to training_enroll@cadence.com.

Note: The links in this section open in a separate web browser window when clicked in Cadence Help.

Help and Support Facilities

Virtuoso offers several built-in features to let you access help and support directly from the software.

- The Virtuoso *Help* menu provides consistent help system access across Virtuoso tools and applications. The standard Virtuoso *Help* menu lets you access the most useful help and support resources from the Cadence support and corporate websites directly from the CIW or any Virtuoso application.
- The Virtuoso Welcome Page is a self-help launch pad offering access to a host of useful knowledge resources, including quick links to content available within the Virtuoso installation as well as to other popular online content.

The Welcome Page is displayed by default when you open Cadence Help in standalone mode from a Virtuoso installation. You can also access it at any time by selecting *Help – Virtuoso Documentation Library* from any application window, or by clicking the *Home* button on the Cadence Help toolbar (provided you have not set a custom home page).

For more information, see Getting Help in Virtuoso Design Environment User Guide.

Customer Support

For assistance with Cadence products:

- Contact Cadence Customer Support
 - Cadence is committed to keeping your design teams productive by providing answers to technical questions and to any queries about the latest software updates and training needs. For more information, visit https://www.cadence.com/support.
- Log on to Cadence Online Support
 - Customers with a maintenance contract with Cadence can obtain the latest information about various tools at https://support.cadence.com.

Feedback about Documentation

You can contact Cadence Customer Support to open a service request if you:

- Find erroneous information in a product manual
- Cannot find in a product manual the information you are looking for

■ Face an issue while accessing documentation by using Cadence Help

You can also submit feedback by using the following methods:

- In the Cadence Help window, click the *Feedback* button and follow instructions.
- On the Cadence Online Support <u>Product Manuals</u> page, select the required product and submit your feedback by using the <u>Provide Feedback</u> box.

Typographic and Syntax Conventions

The following typographic and syntax conventions are used in this manual.

text	Indicates names of manuals, menu commands, buttons, and fields.
text	Indicates text that you must type as presented. Typically used to denote command, function, routine, or argument names that must be typed literally.
z_argument	Indicates text that you must replace with an appropriate argument value. The prefix (in this example, z_{-}) indicates the data type the argument can accept and must not be typed.
	Separates a choice of options.
{ }	Encloses a list of choices, separated by vertical bars, from which you must choose one.
[]	Encloses an optional argument or a list of choices separated by vertical bars, from which you may choose one.
[?argName t_arg]	
	Denotes a <i>key argument</i> . The question mark and argument name must be typed as they appear in the syntax and must be followed by the required value for that argument.
	Indicates that you can repeat the previous argument.
	Used with brackets to indicate that you can specify zero or more arguments.
	Used without brackets to indicate that you must specify at least one argument.
,	Indicates that multiple arguments must be separated by commas.

=>	Indicates the values returned by a Cadence [®] SKILL [®] language function.
/	Separates the values that can be returned by a Cadence SKILL language function.

If a command-line or SKILL expression is too long to fit within the paragraph margins of this document, the remainder of the expression is moved to the next line and indented. In code excerpts, a backslash (\setminus) indicates that the current line continues on to the next line.

1

Design Rule Driven Editing

The Design Rule Driven (DRD) editing tool provides real-time dynamic DRC feedback to facilitate correct-by-construction layout design. You can use DRD editing to significantly reduce the number of design rule check (DRC) violations during layout editing. However, the features provided through DRD are not a replacement for physical verification tools such as Pegasus or PVS.

DRD checks the layout against design rules defined as constraints. For a list of constraints supported by DRD editing, see <u>DRD Constraints Support</u>.

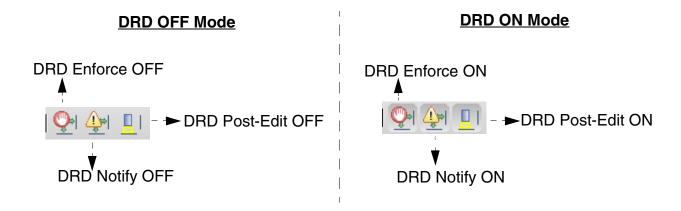
This chapter contains the following topics:

- Turning on DRD Editing
- DRD Operating Modes
- Selecting Constraints for Verification
- Selecting Layers for Verification
- Managing Interactive Display Options
- Performing Color Checks (ICADVM20.1 Only)
- DRD Editing and Blockage Objects
- Using DRD Editing in VLS XL, VLS GXL, and VLS EXL
- Batch-Check Mode
- Verifying Selected Objects
- Verifying Nets

Turning on DRD Editing

You can turn on DRD editing by using the icons available on the *Options* toolbar in Virtuoso or from the <u>DRD Options</u> form.

The figure below shows the icons for the three DRD editing modes in the OFF and ON states. By default, DRD editing is OFF.



Setting Up DRD Options

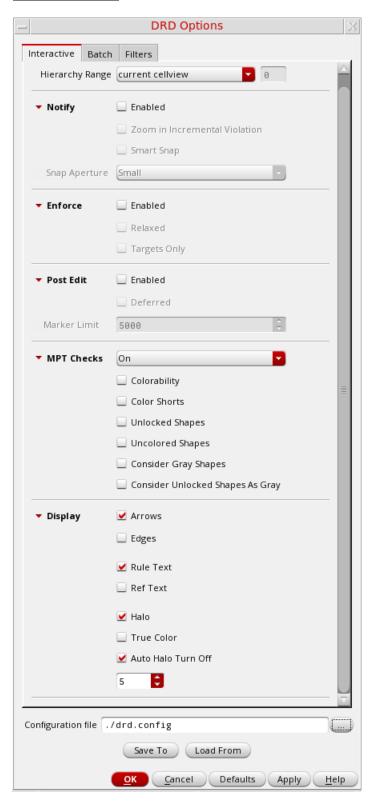
The DRD Options form allows you to set up the options that are checked by DRD during layout editing. To open the DRD Options form, do one of the following.

- Click the DRD Options icon on the DRD toolbar. You can enable or disable the DRD toolbar using one of these methods:
 - □ Choose *Window Toolbars DRD*.
 - □ Right-click anywhere in the main toolbar area and click *DRD*.
- Choose Options DRD Edit.
- Choose Verify Design Process Rules DRD Options.
- Press F7 if bindkeys are loaded.

The <u>DRD Options</u> form is displayed. The form consists of the following tabs:

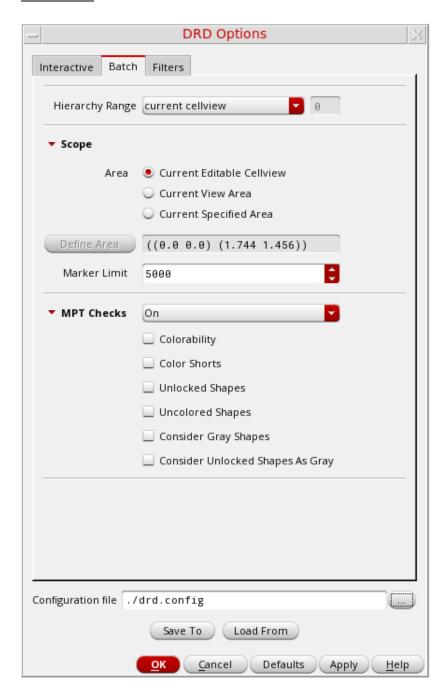
Design Rule Driven Editing

Interactive Tab



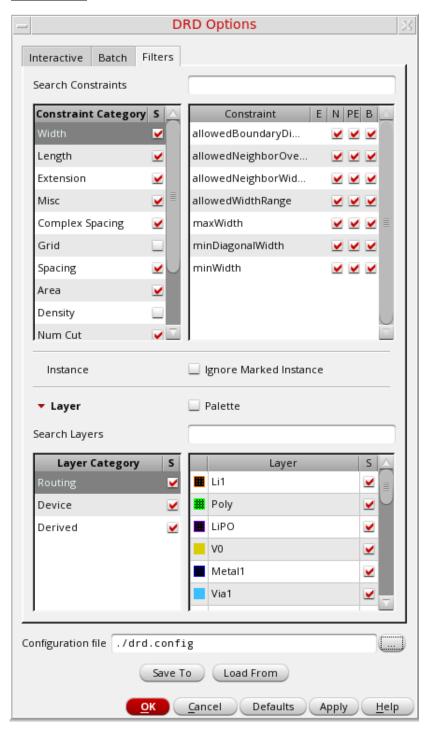
Design Rule Driven Editing

Batch Tab



Design Rule Driven Editing

Filters Tab



Design Rule Driven Editing



For a quick overview of the DRD Options form, see <u>Introduction to the New DRD</u> User Interface.

The following sections in this guide describe how you can set up the various DRD editing options.

DRD Configuration File

The DRD Options form lets you save and load its settings by using a configuration file. You can work with this file by using the fields near the lower edge of the DRD Options form.



To specify a configuration file, type its name and path in the *Configuration file* field or use the browse button to locate and select an existing file. Ensure that you have write permissions to the path and file.

To save the current settings in the DRD Options form to the specified configuration file, click the *Save To* button.

To load settings from the specified configuration file into the DRD Options form, click the *Load From* button.

DRD Operating Modes

DRD flags violations dynamically when you create or edit shapes, such as paths, pathSegs, pins, rectangles, polygons, and multi-part paths and instances, in the following three operating modes:

- Enforce stops cursor movement when the threshold for a design rule violation is reached.
- *Notify* displays visual feedback on design rule violations.
- Post Edit displays violation markers if creating or modifying an object in the layout results in design rule violations.

Design Rule Driven Editing

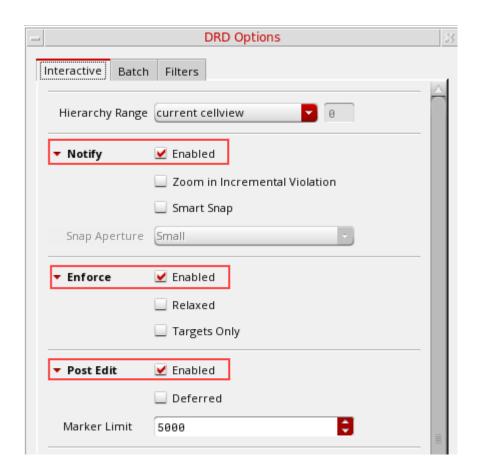
Note: For better usability, interactive checking is disabled when cursor movements are very fast or while editing designs zoomed out to a high level.

You can use DRD in the following combinations:

- Enforce and Notify
- Notify and Post Edit
- Enforce and Post Edit

To change the mode, use one of the following options:

■ The <u>DRD Options</u> form – *Interactive* tab
Select the *Enabled* check box against the three editing modes and click *OK*.



- The Ctrl+e bindkey
- The <u>leToggleDrdMode</u> SKILL function

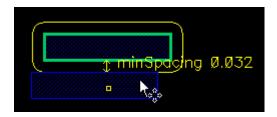
Design Rule Driven Editing

Note: The bindkey and SKILL function help toggle between *Notify* and *Enforce* and *Enforce* and *Notify* modes.

Using Enforce Mode

In *Enforce* mode, the pointer is constrained or halted when a spacing violation occurs.

The following figure shows an example of a violation created in *Enforce* mode.



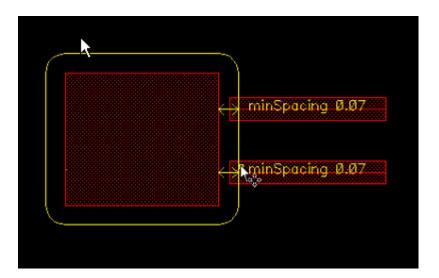
For information about the constraints supported in *Enforce* mode, see <u>DRD Constraints Support</u>.

Using Notify Mode

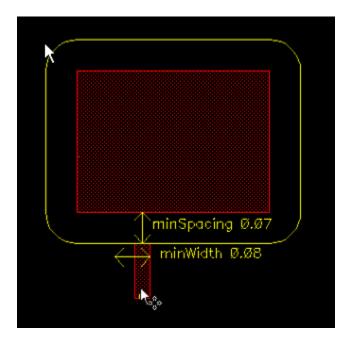
In *Notify* mode, the tool provides visual feedback (text, edges, halos, or arrows) when the specified design check rules are violated. However, the pointer is not constrained or halted, allowing you to create objects with design violations. If you create an object with design violations, the visual feedback about the violation disappears. To retain an indication of the design violation, you can turn on *Post Edit* mode. For more information, see <u>Using Post Edit Mode</u>.

Design Rule Driven Editing

The following figure shows an ex vample of a violation created in *Notify* mode.



You can enable *Enforce* and *Notify* modes together to use their combined behavior. For example, in the figure below, spacing constraints are enforced and the other constraints are notified.

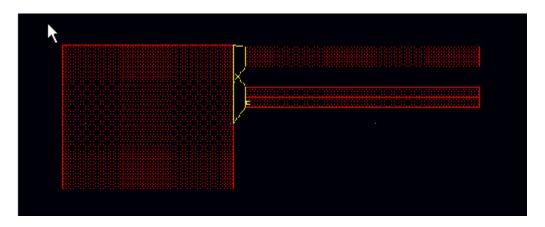


Using Post Edit Mode

In *Post Edit* mode, DRD displays violation markers if creating or modifying an object in the layout results in design rule violations.

Design Rule Driven Editing

Post edit markers are displayed when data is modified, including any modification caused by a command run in the CIW; by user SKILL functions that create, modify, or delete data; and when using batch processing tools, such as the Virtuoso Custom Placer or the Virtuoso Custom Router. The following figure shows examples of post edit markers.



Important

The *Post Edit* and *Batch* modes can be slow when too many rules are being checked in a large design. You can use Ctrl+c to interrupt the checking process. Markers found before the process is interrupted are generated.

In Post Edit mode, an incremental check is performed on an area based on the objects that are edited. For example, if you stretch a segment on the Metall layer, DRD verifies the change and updates the markers related to Metall. Existing markers that are not related to the object being edited are not deleted.

You can view, analyze, and filter post edit markers on the *DRC/DFM* tab of the Annotation Browser. The markers are persistent objects that can be saved. For more information, see <u>Annotation Browser</u> in *Virtuoso Layout Suite XL User Guide*.

You can search for, explain, and delete markers by using the commands on the *Verify – Markers* menu. For more information, see <u>Markers</u> in *Virtuoso Layout Suite XL User Guide*.

Behavior of Post Edit Mode when Editing in Place

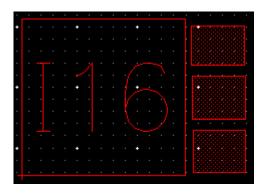
To display post edit markers when editing in place, ensure that you do the following:

- **1.** In the DRD Options form:
 - **a.**On the *Interactive* tab, choose *Hierarchy Range* as *current to bottom*.

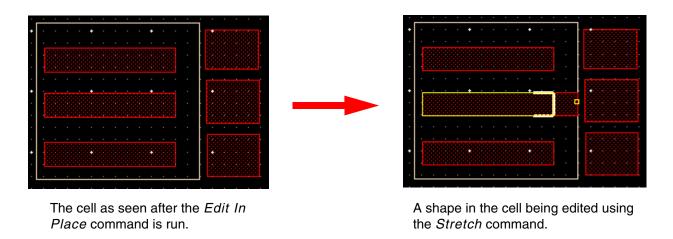
Design Rule Driven Editing

- **b.**Select the *Post Edit Enabled* check box.
- **2.** In the Annotation Browser, select *Scope* as *Top Cellview Only* or *Top Cellview To Depth*.

To understand how Post Edit mode works when editing in place, consider the following example in which an instance and three shapes are placed at the top level.



Next, select the instance and choose *Edit – Hierarchy – Edit In Place* and edit a shape inside the cell.

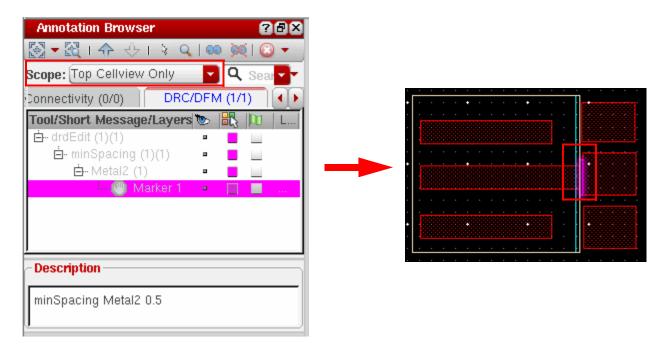


To view the post edit markers, choose *Scope* as *Top Cellview Only* in the Annotation Browser, as shown below. Select a marker in the Annotation Browser to highlight it in the canvas.

Design Rule Driven Editing

/Important

Notice that the markers displayed in the Annotation Browser are dimmed. This is because all markers are created at the top level. Consequently, you cannot select a marker by clicking it in the canvas while Edit-In-Place is active.



Related Environment Variables

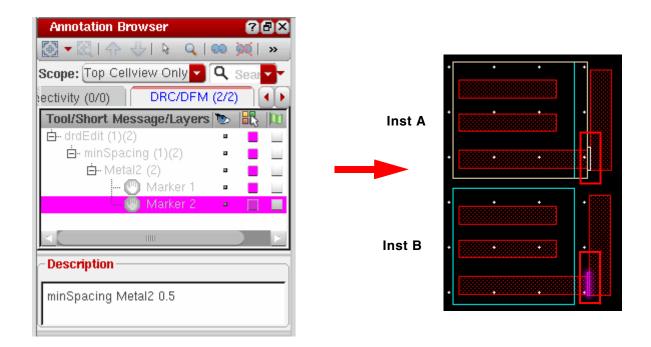
The following environment variables control the behavior of *Post Edit* mode when editing in place:

- <u>drdPostEditEIPCheckTopView</u>: Checks for DRC violations between the shape being edited and the shapes in the parent hierarchy and in the neighboring cells. The default value is t.
- <u>drdPostEditEIPAllInstances</u>: Checks the environment around all instances of a cell for DRC violations when a shape in that cell is edited in place. The default value is nil.
- <u>drdPostEditEIPAllWindows</u>: Applies the value set for drdPostEditEIPCheckTopView and drdPostEditEIPAllInstances to all open design windows. The default value is nil.

Design Rule Driven Editing

Example of Violations Created in All Instances of a Cell

The following figure illustrates the results of editing a shape in a cell with the drdPostEditEIPAllInstances environment variable set to t. You can see below two instances of the same cell, *Inst A* and *Inst B*. When a shape in *Inst A* is edited in place, it creates violations in both instances of the cell, as shown.



Ignoring Violations inside Instances

When *Hierarchy Range* is set as *current to bottom*, *current to stop level*, or *current to user level*, DRD in all modes reports DRC violations between a hierarchical object, such as an instance, and other hierarchical or top-level objects, such as wires, rectangles, and vias. However, in Enforce, Notify, and Post Edit modes, DRD does not report violations that may exist inside a hierarchical object if the shapes that cause violations are drawn on physical layers. This is because the <u>drdEditIgnoreIntraInstanceChecks</u> environment variable is by default set to t.

In Batch mode, when *Hierarchy Range* is set as *current to bottom*, *current to stop level*, or *current to user level*, DRD reports all violations that exist inside a hierarchical object, even when drdEditIgnoreIntraInstanceChecks is set to t.

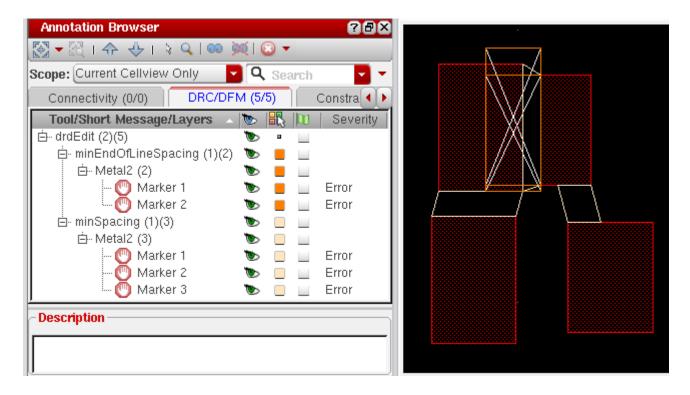
Note: DRC violations involving derived layers found inside an instance being created, moved, copied, or stretched are reported in Enforce, Notify, and Post Edit modes.

Design Rule Driven Editing

Severity of DRD Markers

The severity type for DRD markers is *Error*. The severity for DRD markers is displayed in the Annotation Browser in the *Severity* column, as shown.

Note: To display the *Severity* column, right-click a column header in the Annotation Browser and choose *Columns – Severity*.



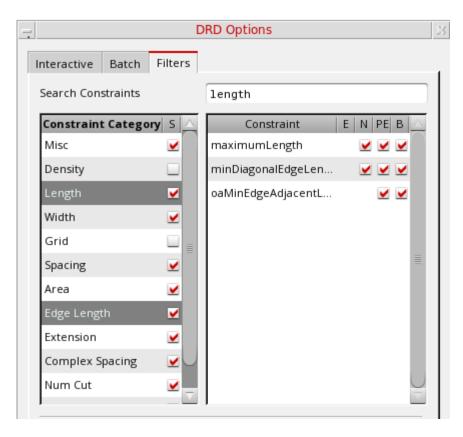
Selecting Constraints for Verification

You can use the *Filters* tab of the DRD Options form to choose the constraints that you want to check for violations. The filter feature is available at both the constraint category and individual constraint levels.

To select a constraint for design verification, do one of the following:

Type a string in the Search Constraints field.

All the constraints containing the string are listed in the *Constraint* column, and the corresponding constraint categories are highlighted in the *Constraint Category* column. However, if you click a constraint category, the string is removed from the *Search Constraints* field and only the constraints belonging to that category are displayed in the *Constraint* column.



In the *Constraint Category* column, click the category you want to select or click and hold down the Ctrl key to select more than one category.

The constraints for the selected categories are displayed in the *Constraint* column.

Design Rule Driven Editing

To control the checking of constraints for a category or a individual constraint, select or deselect the check box against it.

To enable or disable the checking of all the constraint categories at one go, click the *S* column header next to *Constraint Category*.

To enable or disable the checking of all the selected constraints at one go, click the column headers that indicate the different supported modes. The *E*, *N*, *PE*, and *B* columns indicate the Enforce, Notify, Post Edit, and Batch modes, respectively.

Selecting Layers for Verification

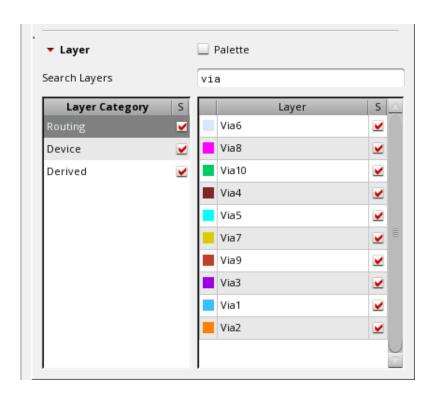
The Filters tab of the DRD Options form lets you choose the layers that you want to check for violations. When you filter the layers, only those violations that involve shapes on the selected layers are displayed. You can select the layers from the three layer categories provided, that is, *Routing*, *Device*, and *Derived*. The filter feature is available at both the layer category and individual layer levels.

Note: If you select the *Palette* check box, the layer filter options are disabled. In this case, you can control the checking of layers based on the layer visibility in Palette.

To select a layer for design verification, do one of the following:

Type a string in the Search Layers field.

All layers containing the string are listed in the *Layer* column, and the corresponding layer categories are highlighted in the *Layer Category* column. However, if you click a layer category, the string is removed from the *Search Layers* field and only the layers belonging to that category are displayed in the *Layer* column.



For optimal performance, only layers included in the fabrication process and with a corresponding drawing purpose in the technology file are displayed in the Layer filter and will be checked through the filter.

Design Rule Driven Editing

Layers are categorized as follows in DRD:

- Routing: Includes physical layers whose functions are defined and whose material type is one of these: Cut, Poly, Metal, or Contactless Metal (excluding Other material type).
- Derived: Includes derived layers in the technology file.
- Device: Includes layers that are not Routing or Derived layers and Cut layers in which the below layer is not Metal.
- □ Other material types: Includes layers with undefined function.
- In the Layer Category column, click the category you want to select or click and hold down the Ctrl key to select more than one category.

The layers for the selected categories are displayed in the *Layer* column.

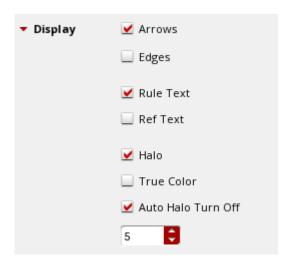
To enable or disable the checking of shapes and objects on a layer or layer category, select or deselect the check box against it.

To enable or disable the checking of all the three layer categories at one go, click the S column header next to Layer Category.

To enable or disable the checking of all the selected layers at one go, click the *S* column header next to *Layer*.

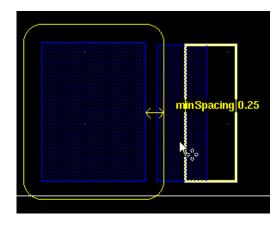
Managing Interactive Display Options

The various *Display* options available on the *Interactive* tab work with DRD Enforce and DRD Notify modes:

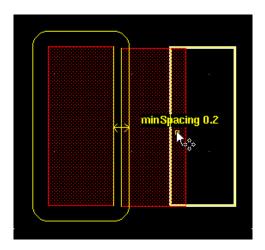


To enable an option, ensure that the corresponding check box is selected and specify the required value.

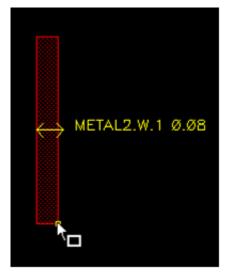
Arrows displays arrows to indicate optimum spacing between objects. The arrows are proportional to the size of the objects.



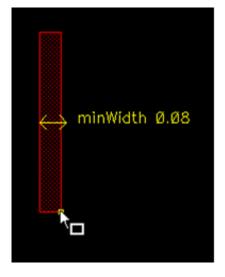
■ *Edges* highlights the edge where the violation occurs.



- Rule Text displays the name of the violated constraint at the location where the violation occurs. You can use the <u>displaySigDigits</u> environment variable to specify the number of decimal digits the constraint value must be displayed with in the rule text.
 - If the bounding boxes for rule text overlap, the rule text for the violation that is encountered first is displayed. Trailing zeros are removed to improve readability.
- Ref Text displays the ref text for a constraint from the technology file, followed by the constraint value. This text contains the rule ID from the Design Rule Manual (DRM) of the foundry, and is, therefore, more meaningful for a layout designer. When deselected, only the name of the violated constraint is displayed, as shown below.



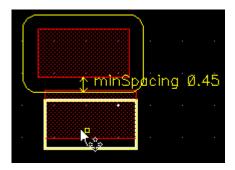
'ref Text from Techfile is on



'ref Text from Techfile is off

Design Rule Driven Editing

- *Halo* creates halos around stationary objects while creating or editing an object that violates a spacing constraint.
- *True Color* halos for each layer are drawn in the color of that layer. This makes it easier to differentiate between halos on different layers.



Default halo



True color halo

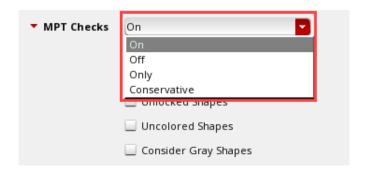
- Auto Halo Turn Off suppresses halos if the number of violations exceeds the value that you specify. Only arrows are displayed on the canvas when the defined limit is reached.
 - You can specify a value between 1 and 1000.

Performing Color Checks (ICADVM20.1 Only)

The MPT Checks options available on the Interactive and Batch tabs enable you to perform color checks on a Multi-Patterning Technology (MPT) design with DRD. These checks are divided into the following two categories:

- Color Constraints Checks
- Color-Related Flow Checks

Color Constraints Checks

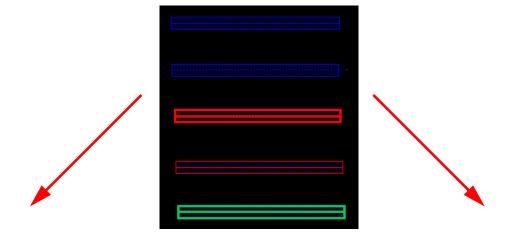


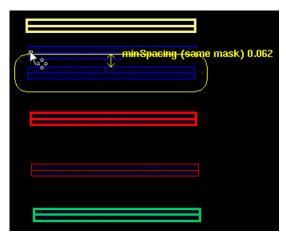
The *MPT Checks* drop-down menu lets you perform the checks for color constraints in different DRD modes. This drop-down list provides the following four options:

- On: (Default) DRD reports violations for all color-related constraints, including non-color constraints.
- Off: DRD reports violations for all available constraints except those related to color constraints. For example, DRD in this mode reports all minSpacing violations except sameMask violations.
- Only: DRD reports violations only for color-related constraints.
- Conservative: DRD checks all color rules on all shapes irrespective of their color. For example, DRD reports same-mask violations as follows:
 - Between shapes that are on the same mask, such as mask1-mask1 and mask2mask2
 - Between shapes that are on different masks, such as mask1-mask2
 - Between gray shapes and other shapes of any other color, such as gray-gray or gray-maskN, where N = 1, 2, 3, ...

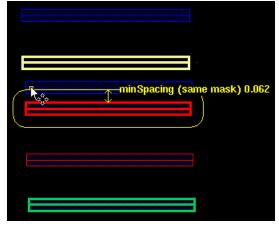
Design Rule Driven Editing

The figure below shows violations that are reported when *Conservative* is selected:

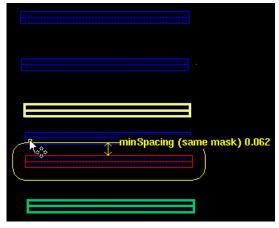




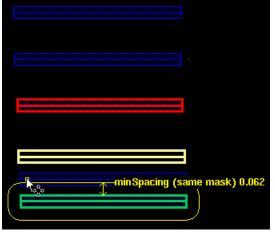
gray-gray



gray-mask1



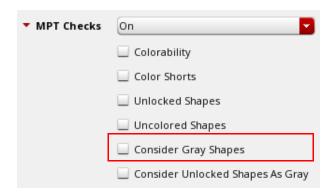
mask1-mask1



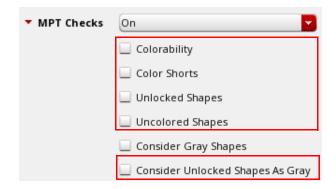
mask1-mask2

Design Rule Driven Editing

While checking color constraints, you can select the *Consider Gray Shapes* check box to report violations for all gray shapes. This option is available only in the *MPT Checks – On* and *MPT Checks – Only* modes.



Color-Related Flow Checks

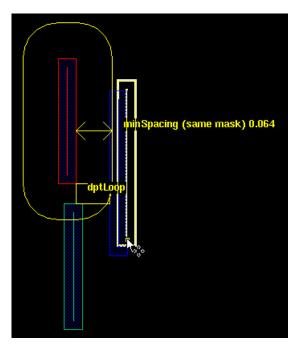


The color-related flow checks are described below. These checks are independent of the *MPT Checks* drop-down menu settings.

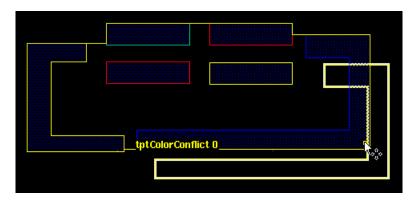
Colorability: Colorability check is performed to determine if a set of shapes is colorable without any color rule violation in an MPT environment. This check does not look for actual color of the shapes. If a set of shapes fails colorability check, those shapes cannot be colored using a certain number colors. In case of a Double Patterning Technology (DPT) environment, the set of shapes needs to be colored using two colors; in case of a Triple Patterning Technology (TPT) environment, the number of colors would be three. For more information, see multiMaskCheck in Virtuoso Technology Data Constraints Reference.

Note: Colorability checks are available only in DRD Notify, Post Edit, and Batch modes, and they can be performed only when the *Spacing* and *Complex Spacing* constraint groups are selected on the *Filters* tab of the <u>DRD Options</u> form.

The figure below shows a double patterning loop violation:



(ICADVM20.1 Only) The figure below shows a triple patterning loop violation:



 Color Shorts: Reports overlaps between shapes with different colors, irrespective of their lock status.

If minStitchOverlap is defined for a layer, DRD reports stitch violations, instead of color shorts, for shapes on that layer.

- Unlocked Shapes: Reports color-unlocked shapes on a layer.
- *Uncolored Shapes*: Reports uncolored (gray) shapes on layers that support multiple masks.
- Consider Unlocked Shapes as Gray: Considers unlocked shapes as gray.

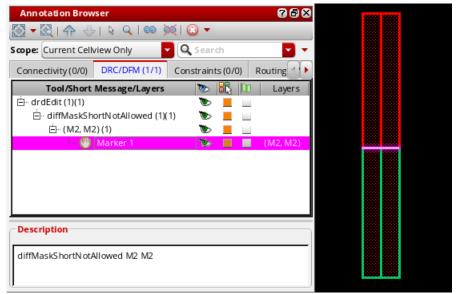
Design Rule Driven Editing

Using the Annotation Browser to View Color Check Violations

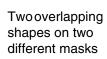
Color violations are displayed in the Annotation Browser under tool category *drdEdit*.

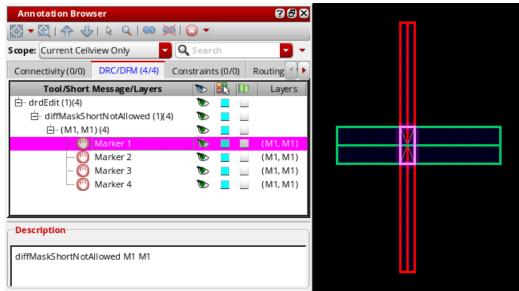
Viewing Color Shorts

The Annotation Browser lists the color shorts between shapes on different masks under diffMaskShortNotAllowed.



Two abutting shapes on two different masks

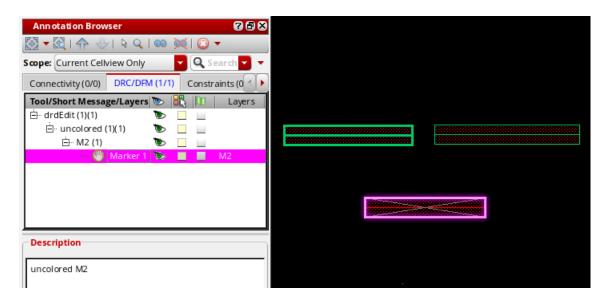




Design Rule Driven Editing

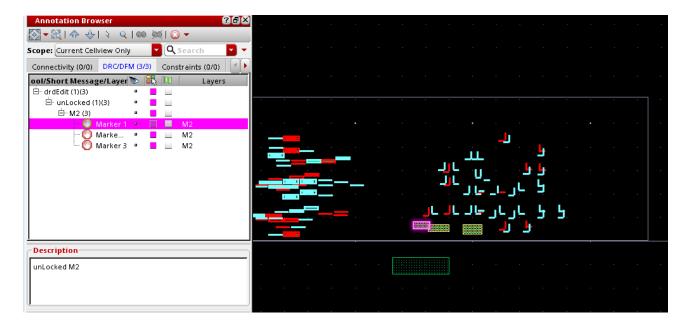
Viewing Uncolored Shapes

The Annotation Browser lists the uncolored (gray) shapes drawn on layers that support multiple masks under *uncolored*.



Viewing Color-Unlocked Shapes

The Annotation Browser lists the color-unlocked shapes on a layer under *unLocked*.



Design Rule Driven Editing

For more information about checking and fixing color violations, see <u>Virtuoso Multi-Patterning Technology User Guide</u> (ICADVM20.1 Only).

Related Environment Variables

The following environment variables can be used to set various color-related options in DRD:

- <u>drdEditColorRules</u>
- drdEditBatchColorRules
- drdColorability
- drdBatchColorability
- drdColorShorts
- drdBatchColorShorts
- drdUnlocked
- drdBatchUnlocked
- drdUncoloredShapes
- drdBatchUncoloredShapes
- drdConsiderGrayShapes
- drdBatchConsiderGrayShapes
- drdConsiderUnlockedShapesAsGray
- drdBatchConsiderUnlockedShapesAsGray

Design Rule Driven Editing

DRD Editing and Blockage Objects

A blockage object is a polygonal object that outlines illegal placement and/or routing areas for applicable types of objects in a given cellview. For example, a routing blockage is defined as an area where no interconnect objects may be placed on the specified layer.

If an interconnect object (path, via, or other basic shape) is created or edited on a layer to overlap blockage of the same layer, DRD editing violations are reported depending on the DRD editing settings (display of graphical feedback in Notify and Post Edit modes or adjustment of the interactive drag point in Enforce mode). The violation text message is "<type> blockage", where "type" is one of the blockage types: routing, placement, wiring, fill, slot, pin, and placement halo. DRD editing does not consider blockage types "feedthru" and "screen".

When a blockage object is near, contained within, or overlapping an interconnect object (path, via or other basic shape), violations are flagged only with post edit markers. DRD does not attempt to interactively flag with warnings (in Notify mode), nor attempt to adjust the drag point, nor move the non-blockage objects already placed (in Enforce mode). For example, if a path exists, and a routing blockage is created to overlap it, the violations are flagged with post edit markers by DRD editing if enabled.

When you use the *Create*, *Move*, *Copy*, *Stretch* or other edit commands on a blockage object, the markers are also only post edit if enabled. However, when you create design objects, all the three DRD modes are applied based on DRD editing settings.

Using DRD Editing in VLS XL, VLS GXL, and VLS EXL

The VLS XL, VLS GXL (IC6.1.8 Only), and VLS EXL (ICADVM20.1 Only) environments are connectivity and constraint driven, where the design is checked for short and open violations and for speciality routing constraints. DRD connectivity awareness is driven by the layout extractor. We therefore recommend that you run extraction when initially loading the design. The DRD modes use the XL extractor to assign connectivity to unassigned shapes when *Create Wire*, *Stretch*, *Reshape*, and *Point to Point* commands are run. To extract the design, choose *Connectivity – Update – Extract Layout*.

For more information about extraction, see <u>Connectivity Extraction</u> and <u>Extract Layout</u> in *Virtuoso Layout Suite XL User Guide*.

When DRD Enforce mode is enabled, assisted routing commands, such as point-to-point routing, guided routing, and finish wire, create routing after validating the commands based on the design rule checks.

Design Rule Driven Editing

For more information about assisted routing commands, see <u>Using Assisted Routing</u> <u>Commands</u> in *Virtuoso Interactive and Assisted Routing User Guide*.

Batch-Check Mode

You can run DRD in Batch mode either on the entire design or on a specified area in the design. This feature is available only in VLS XL, VLS GXL (IC6.1.8 Only), and VLS EXL (ICADVM20.1 Only) modes.

To run DRD in Batch mode, do one of the following.

■ Click the *Verify Design* icon on the DRD toolbar and select the area of check from the drop-down list. The three area options are *Current Cellview*, *Visible Area*, and *Select Area*.

Current Cellview (default) checks all the shapes in the current editable cellview.

Visible Area checks all the shapes in the current viewable area.

Select Area checks all the shapes in the specified area of the design.

When you select the *Select Area* option, specify the lower-left and upper-right coordinates for the region in this format ((xlower ylower) (xupper yupper)) in the text box provided on the DRD toolbar. Alternatively, you can define the area on the canvas.

■ Click the *DRD Options* icon on the DRD toolbar, choose *Options – DRD Edit* or choose *Verify – Design – Process Rules – DRD Options*.

The <u>DRD Options</u> form is displayed. Use the <u>Batch Tab</u> to set up the various Batch mode options.

For information about running DRD in Batch mode on read-only cellviews, see <a href="https://dread-only.com/dre

Verifying Selected Objects

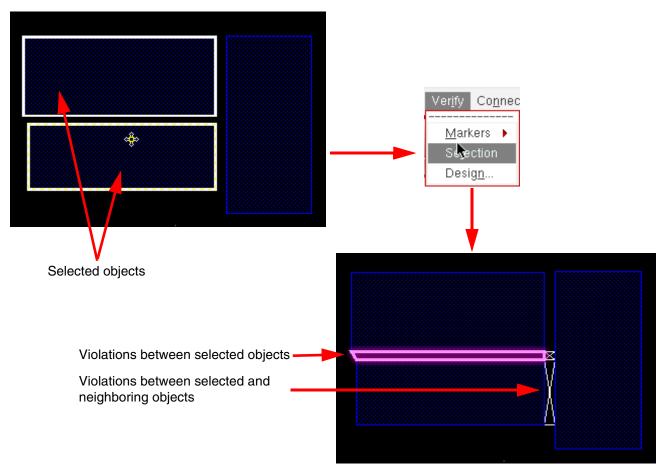
After selecting one or more objects in the layout, you can run a DRD check to verify if DRC violations exist between the selected objects and between the selected and the neighboring objects.

To verify selected objects, select one or more objects in the layout and choose *Verify – Selection*. You can verify up to 25 objects at a time.

Design Rule Driven Editing

Note: This capability is available only in VLS XL, VLS GXL (IC6.1.8 Only), and VLS EXL (ICADVM20.1 Only) modes.

The figure below shows the violations that exist between the selected objects and their neighboring objects.



Note: Choose *Edit – Undo* if you need to remove the violation markers found by *Verify – Selection*.

You can also verify objects by using the <u>drdVerifySelSet</u> SKILL function.

Verifying Nets

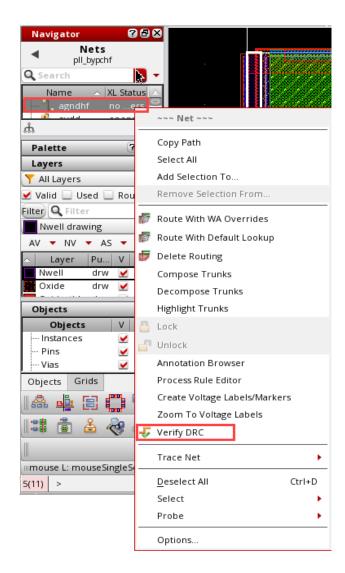
The *Verify DRC* command enables you to check for DRC violations on a net or a set of nets. All shapes for the selected nets under the current hierarchy are checked for violations. To check for violations between the shapes on the selected nets and the shapes down the hierarchy, you can change the *Hierarchy Range* appropriately on the *Batch* tab of the DRD Options form.

Design Rule Driven Editing

The *Verify DRC* command is independent of the *Scope* options specified in the DRD Options form. That is, it is run for the selected nets on the entire cellview.

To run *Verify DRC* on a net, do the following:

In the Navigator assistant, select one or more nets, right-click, and choose Verify DRC from the context menu.



You can view, analyze, and filter the violation markers of net verification on the *DRC/DFM* tab of the Annotation Browser.



For a quick overview of the feature, see Verifying Nets in DRD.

DRD Advanced Features

You can use the following DRD advanced features to create DRC-correct layouts more efficiently:

- DRD Targets
- Targeted Enforce
- Relaxed Enforce
- Deferred Post Edit
- What-if Halos
- DRD Smart Snapping
- Incremental Violation Display
- Excluding Instances from Checking

These features can be used with flat shapes located at the current level of layout hierarchy, as well as with shapes located in hierarchical and pseudo-hierarchical layout objects such as instances, vias, and groups (figGroups).

Important

The features described in this chapter are available only in Layout XL, GXL, and EXL tiers.

DRD Targets

DRD targets are designated layout objects used by advanced DRD features such as Targeted Enforce and What-if Halos. To use a layout object with any of these features, you must first set it as a DRD Target. This section explains how layout objects are set as DRD targets.

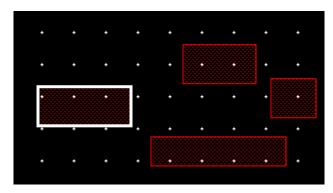
- Setting DRD Targets
- Unsetting DRD Targets

Setting DRD Targets

To set an object as a DRD target, you can either select it in the canvas and choose a *DRD Targets* menu option or use the bindkey provided. Selection works only for top-level objects; the bindkey can be used to set any object as a DRD target, including the objects below the top level in the hierarchy.

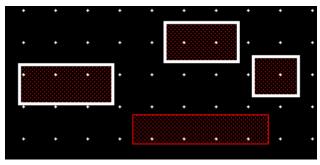
Setting DRD Targets by Selection

To set one or more objects as DRD targets, you must first select the required objects in the canvas, as shown.



Single object selected

Multiple objects selected



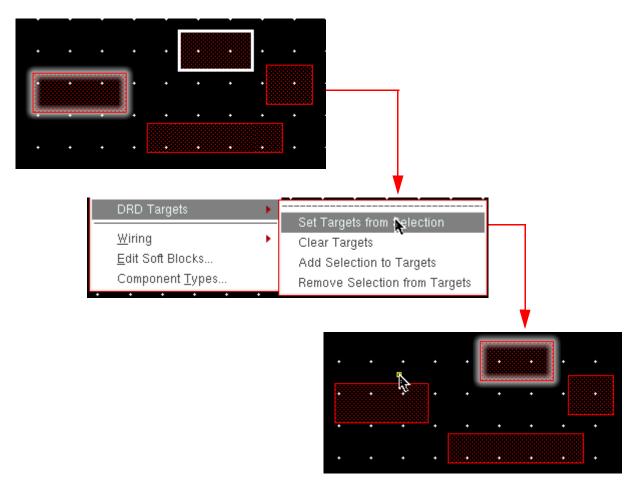
DRD Advanced Features

After you have selected the objects, you can either set the selected objects as new DRD targets or add them to an existing set of DRD targets.

■ To set only the selected objects as DRD targets, right-click the selected objects and choose DRD Targets – Set Targets from Selection from the context menu.

Alternatively, choose *Edit – DRD Targets – Set Targets from Selection*.

The selected objects are set as DRD targets, as shown:

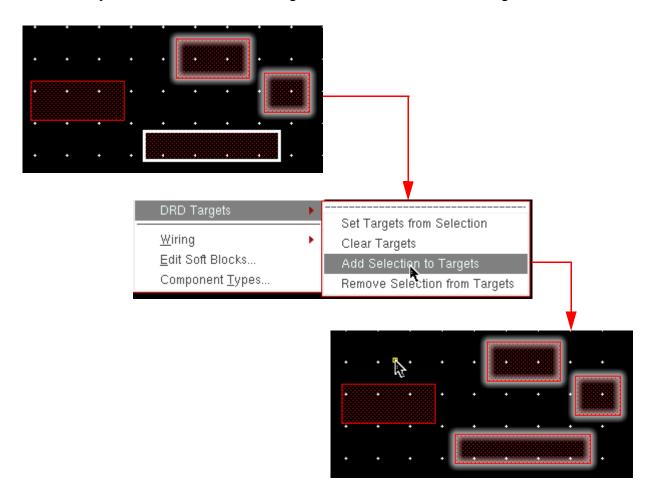


Note: The Set Targets from Selection option clears any existing DRD targets.

■ Right-click the selected objects and choose *DRD Targets – Add Selection to Targets* from the context menu if you want to add the selected objects to an existing set of DRD targets.

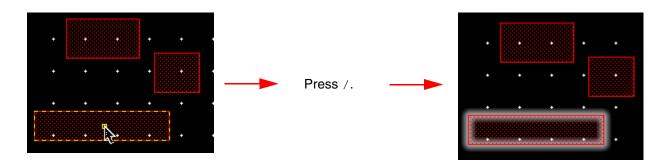
DRD Advanced Features

Alternatively, choose *Edit – DRD Targets – Add Selection to Targets*.



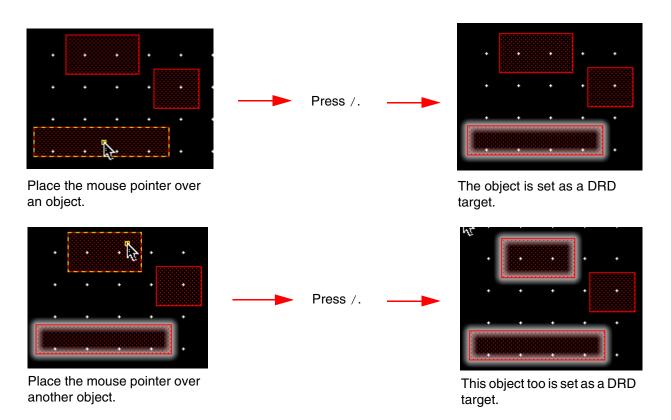
Setting DRD Targets Using the Bindkey

You can use the / (forward-slash) key on the numeric keypad to set objects as DRD targets. Place the mouse pointer on an object and press / to set it as a DRD target, as shown.



DRD Advanced Features

To set multiple objects as DRD targets with the bindkey, select them one by one, as shown.



Note: Laptop users can use the USB keyboard to activate the NumLock key and press the / key on the numeric keypad.

You can also use the bindkey to set as DRD targets the following objects: objects below the current hierarchy level and overlapping objects at any hierarchy level.

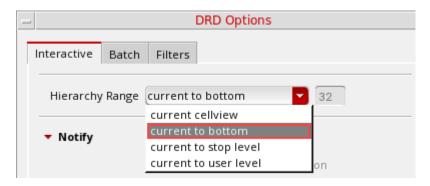
- Setting Hierarchical Objects as DRD Targets
- Setting Overlapping Objects as DRD Targets

DRD Advanced Features

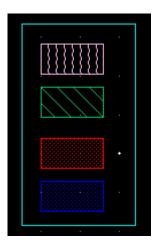
Setting Hierarchical Objects as DRD Targets

You can set objects below the current hierarchy level as DRD targets as follows:

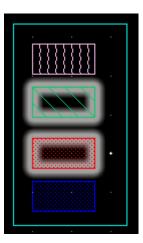
- 1. Open the DRD Options form.
- 2. On the Interactive tab, set the Hierarchy Range to current to bottom, as shown.



- 3. Place the mouse pointer over the hierarchical object that you want to set as a DRD target.
- **4.** Press / on the numeric keypad.



Objects M2 (red) and M3 (green) located below the current hierarchy level are set as DRD targets by using the / bindkey.



DRD Advanced Features

Setting Overlapping Objects as DRD Targets

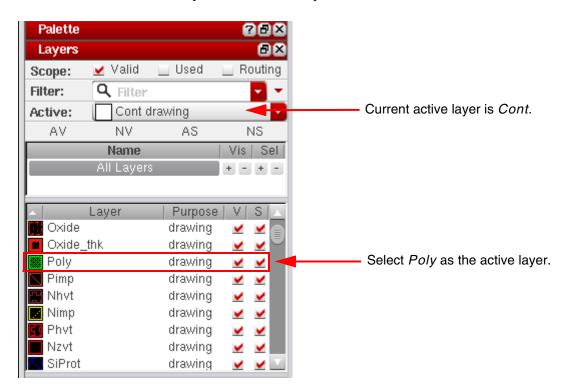
To set an overlapping object as a DRD target, set the layer containing that object as the active layer. In the following example, the objects on the *Poly* (green) and *Oxide* (hashed red) layers overlap.



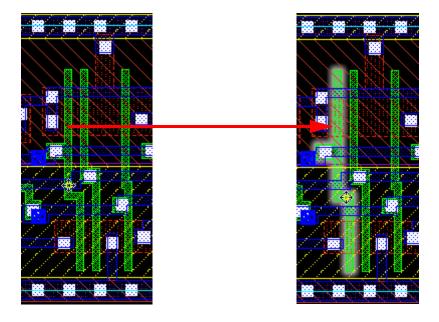
DRD Advanced Features

To set an object on the *Poly* layer as a DRD target:

1. In the Palette, select *Poly* as the active layer, as shown.



- 2. Place the mouse pointer over the object you want to set as a DRD target.
- **3.** Press / on the numeric keypad. The *Poly* gate is successfully set as a DRD target, as shown in the figure on the right.



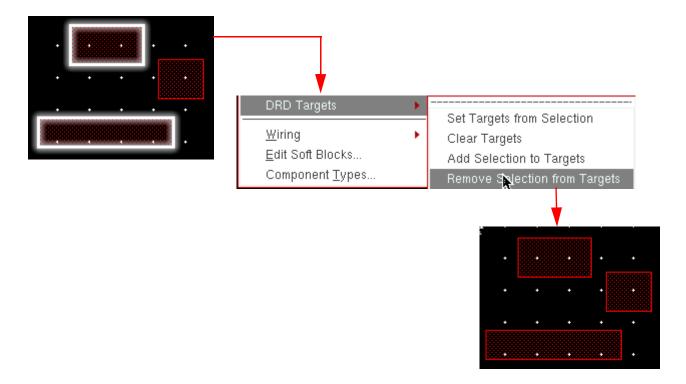
DRD Advanced Features

Unsetting DRD Targets

You can unset a DRD target by using the *DRD Targets* menu options or by using the bindkey provided.

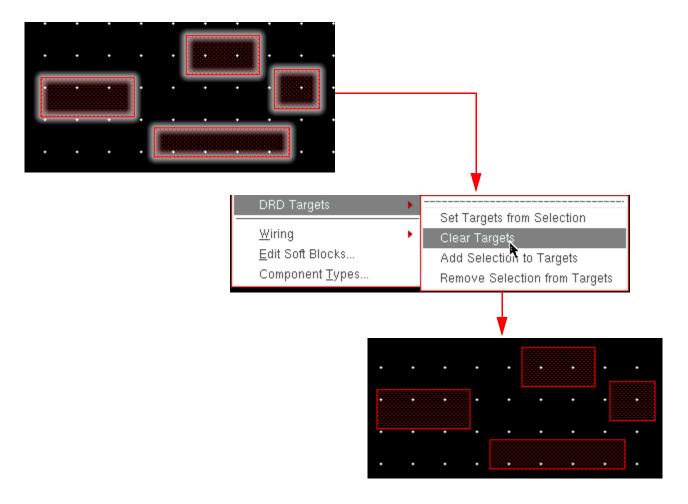
Unsetting DRD Targets by Selection

To unset specific DRD targets, right-click the DRD targets and choose *DRD Targets* – *Remove Selection from Targets* from the context menu. Alternatively, you can select those DRD targets in the canvas and choose *Edit* – *DRD Targets* – *Remove Selection from Targets*.



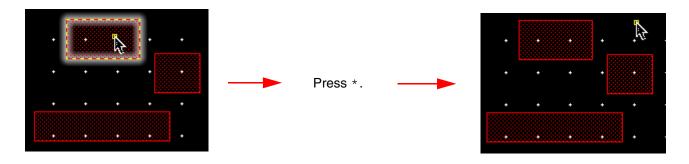
DRD Advanced Features

To unset all existing DRD targets, right-click on the canvas and choose *DRD Targets – Clear Targets* from the context menu or choose *Edit – DRD Targets – Clear Targets*, as shown.



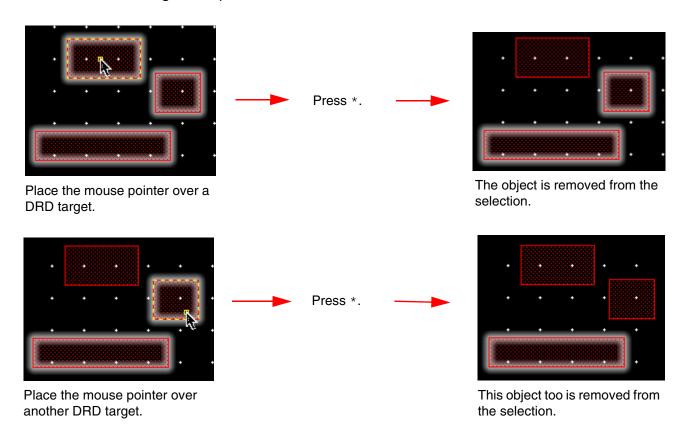
Unsetting DRD Targets Using the Bindkey

You can use the * (asterisk) key on the numeric keypad to unset DRD targets. Place the mouse pointer over an object set as a DRD target and press *. The highlight around the DRD target is removed indicating that the object is no longer set as a DRD target, as shown.



DRD Advanced Features

To unset multiple DRD targets with the bindkey, place the mouse pointer over each successive DRD target and press *, as shown.



Bindkey Reassignment

The default bindkeys for setting and unsetting DRD targets are / and *, respectively. However, if you prefer using different keys, you can reassign the <code>drdAddTargets</code> and <code>drdRemoveTargets</code> SKILL functions to your preferred keys.

In the following example, the functions for setting and unsetting DRD targets are reassigned to keys 3 and 4, respectively. These settings can be included in either a custom bindkey file or the .cdsinit file.

```
hiSetBindKeys("Layout" list(list("<Key>3" "drdAddTarget()")))
hiSetBindKeys("Layout" list(list("<Key>4" "drdRemoveTarget()")))
```

You can also customize your bindkeys by using the Bindkey Editor form.

1. In CIW, choose *Options – Bindkeys* to open the Bindkey Editor form.

DRD Advanced Features

- **2.** In the left pane, under *Application*, choose *Layout*. The bindkeys assigned for Virtuoso Layout Editor are listed on the right.
- **3.** Assign the required keys as bindkeys to the drdAddTarget and drdRemoveTarget functions.

Targeted Enforce

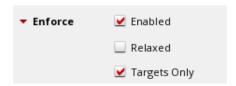
The default DRD Enforce mode enforces minimum spacing rules between the objects being edited and the surrounding objects. If you want to enforce spacing rules only on one nearby object or a subset of nearby objects, the default behavior can be restrictive. To overcome this, you can use the Targeted Enforce feature, which helps enforce minimum spacing rules for a set of objects.

To enable Targeted Enforce:

- 1. Open the DRD Options form.
- **2.** On the *Interactive* tab, select the *Enforce Enabled* check box.

The Targets Only check box is now enabled.

3. Select the *Targets Only* check box to enable targeted enforce.



4. Click OK.

You can also enable DRD Notify mode by selecting the *Notify – Enabled* check box in the DRD Options form. When DRD Notify is enabled, the unenforceable violations are shown as Notify halos.

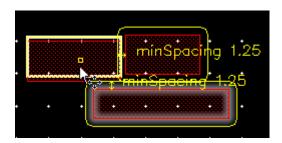
When Targeted Enforce is enabled, DRD Enforce is applied only between the objects being edited (created, copied, moved, or stretched) and the objects set as DRD targets.

To see how Targeted Enforce works:

1. In the canvas, set one or more objects as DRD targets. For more information, see <u>Setting</u> DRD Targets.

DRD Advanced Features

2. Edit an object. In the example below, an object is moved toward another object that is set as a DRD target. As shown, DRD Enforce is applied only between the moving object and the DRD target.



Relaxed Enforce

The Relaxed Enforce mode is a relaxed version of the Enforce mode. In this mode, the pointer can move into the violation region. However, when you edit an object in the violation region, the object is moved to a violation-free location.

The Relaxed Enforce mode is useful for manual device placement. For example, you want to place two PMOS devices closer with minimum oxide spacing between them. In the Enforce mode, as soon as an NWell spacing violation occurs, the pointer movement is stopped, and the pointer cannot move beyond the violation region. With the Relaxed Enforce mode ON, the pointer can move into the violation region and you can see what other violations can occur while placing the PMOS device. However, when you click the mouse button to complete the placement in the violation region, the device is moved to a violation-free location.

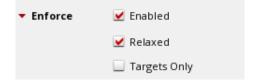
Note: Wire Editor has its own auto corrective mechanism for creating and stretching a wire. Therefore, the Relaxed Enforce feature supports only the copy and move operations for wire.

To enable Relaxed Enforce:

- 1. Open the DRD Options form.
- **2.** On the *Interactive* tab, select the *Enforce Enabled* check box.

The Relaxed check box is now enabled.

3. Select the *Relaxed* check box to enable the Relaxed Enforce mode.



DRD Advanced Features

4. Click OK.



For a quick overview of the Relaxed Enforce feature, see <u>Using the Relaxed Enforce</u> Mode in DRD.

Deferred Post Edit

The Deferred Post Edit is an advanced version of the Post Edit mode. It enables you to see the violation markers for a number of create or edit operations in one go. When you turn on the Deferred Post Edit mode, every create or edit operation is recorded but the violation markers are not displayed immediately after every operation. You can view all the violation markers that occurred during the operations as per your requirements. You can analyze and filter the recorded violation markers for the modified object on the *DRC/DFM* tab of the Annotation Browser.

To enable Deferred Post Edit:

- 1. Open the DRD Options form.
- **2.** On the *Interactive* tab, select the *Post Edit Enabled* check box.

The *Deferred* check box is now enabled.

3. Select the *Deferred* check box to enable the Deferred Post Edit mode.



4. Click OK.

You can also enable the Deferred Post Edit mode by clicking the 📳 icon on the DRD toolbar.



For a quick overview of the Deferred Post Edit mode, see <u>Using the Deferred Post Edit Mode in DRD</u>.

What-if Halos

DRD What-if halos let you visualize early in the design process the potential DRC violations between the shapes being edited, before violations occur.

DRD Advanced Features

To enable DRD What-if halos:

- **1.** Disable DRD Enforce using one of the following methods:
 - □ Click the *Enforce* button on the toolbar.
 - □ Deselect the *Enforce Enabled* check box in the DRD Options form.
- 2. Enable DRD Notify using one of the following methods:
 - Click the *Notify* button on the toolbar.
 - □ Select the *Notify Enabled* check box in the DRD Options form.
- 3. Select the *Notify Smart Snap* check box.
- 4. Set one or more objects as DRD targets. For more information, see Setting DRD Targets.
- **5.** Edit an object. The What-if halo is displayed around the DRD target as soon as you start editing the object, that is, even when no violations occur, as shown.



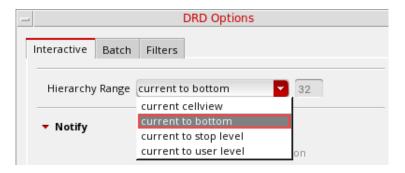
Note: When a via is set as the target, violations on cut layers are ignored and are not shown as halos. To turn on the halos on a cut layer, set the cut layer as the active layer and point to a cut layer shape. This feature helps avoid display congestion on viaArrays that have lots of spacing violation halos on the cut layer.

DRD Advanced Features

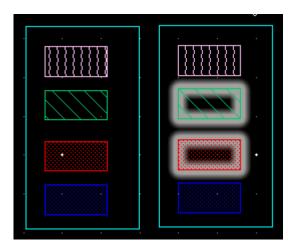
What-if Functionality for Hierarchical Objects

You can enable What-if halos for hierarchical objects as follows:

1. In the DRD Options form, ensure that the *Notify – Enabled* check box is selected and the *Hierarchy Range* is set to *current to bottom*.



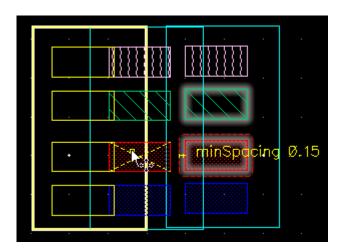
2. Use the / bindkey to set the hierarchical objects in an instance as DRD targets. The following figure shows the green M3 and red M2 objects set as DRD targets in the instance on the right.



3. Move the red M2 object in the instance on the left toward the DRD targets.

DRD Advanced Features

A What-if halo is generated around the red M2 DRD target.



4. To freeze the What-if halo, press ' (the backtick character located on the upper-left corner of your keyboard).

The frozen halo can now be used in conjunction with <u>DRD Smart Snapping</u>.

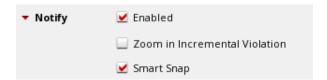
DRD Smart Snapping

The DRD Smart Snap feature lets you snap shapes to the edges or corners of DRC halos generated during DRD editing. It also lets you snap shapes to the discrete spacing values defined for the minSpanLengthSpacing constraint. This helps maintain DRC correctness while you interactively edit your design.

Note: The discrete spacing marker feature is available only for minSpanLengthSpacing.

Snapping to Halos and Discrete Spacing Values

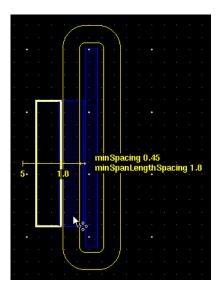
- **1.** In the DRD Options form, select the *Notify Enabled* check box.
- 2. Select the Smart Snap check box and click OK.



Note: To enable smart snapping by default, set the $\underline{drdEditVioLimit}$ environment variable to \pm .

3. Move a shape toward another shape.

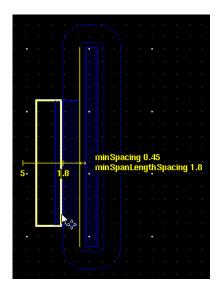
DRD Notify displays a halo around the static shape when a violation occurs between the two shapes. If minSpanLengthSpacing is defined, DRD also displays markers corresponding to the discrete spacing values, as shown below.



DRD Advanced Features

- **4.** To snap the shape to a halo or to any edge of the static shape:
 - **a.** Press ` to freeze the halos. You can also use the drdToggleSmartSnapMode SKILL function.
 - **b.** Move the object to the closest edge of a halo or the static shape.

The yellow edge shown below appears when the shape being moved is snapped to an edge.

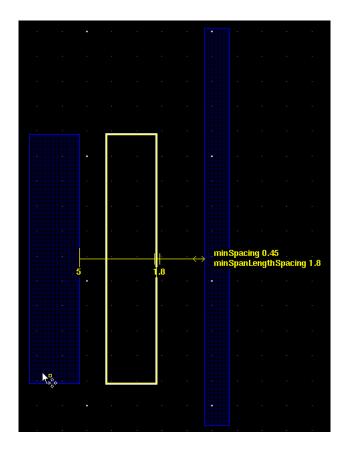


- c. Press ' again to unfreeze the halos.
- 5. To snap a shape to a discrete spacing marker:

DRD Advanced Features

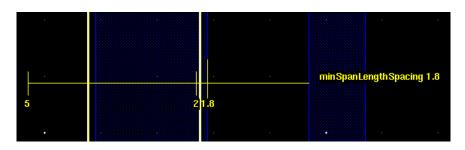
a. Press Shift+' to freeze the discrete spacing markers. You can also use the drdToggleSmartSnapModeForDiscreteSpacing SKILL function.

As a result, the halos disappear and you can see only the discrete spacing markers, as shown below.

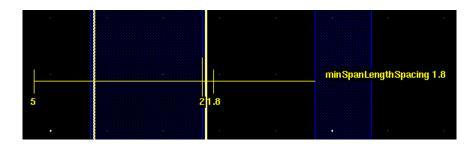


DRD Advanced Features

b. Move the shape to the closest marker. The marker appears elongated when the moving shape is snapped to it, as shown below.



Snapped to spacing value 1.8



Snapped to spacing value 2.0

c. Press Shift+' again to unfreeze the discrete spacing markers.

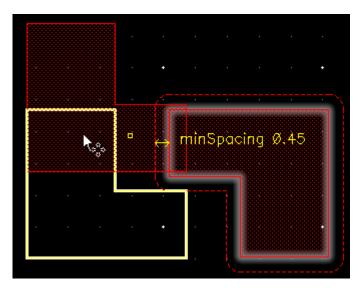
Snapping to a What-if Halo around a DRD Target

You can also snap an object to a What-if halo created around an object set as a DRD target. For more information, see What-if Halos.

- **1.** Set an object as a DRD target.
 - For more information, see **Setting DRD Targets**.
- **2.** Move an object toward the DRD target.

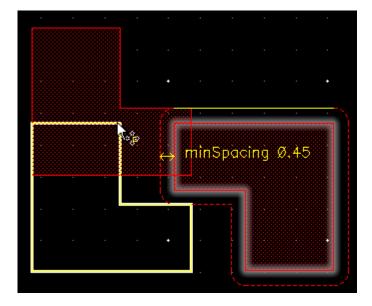
DRD Advanced Features

DRD creates a halo around the DRD target as soon as you start moving the object, irrespective of the distance between the two objects.



- 3. Press ' to freeze the halo on the static object.
- **4.** Snap the moving object to the halo or to any edge of the DRD target.

A yellow edge appears when the moving object is snapped to the edge of the frozen halo, as shown.



DRD Advanced Features

Simultaneous Snapping between Two Orthogonal Edges

To snap an object to an orthogonal edge of another object:

1. Set an object as a DRD target.

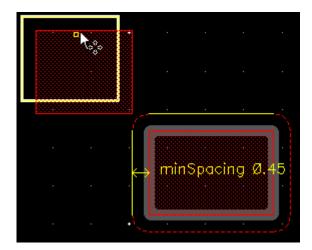
For more information, see Setting DRD Targets.

2. Move an object toward the DRD target.

DRD creates a halo around the DRD target.

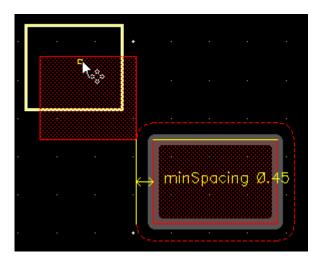
- **3.** Press ' to freeze the halo on the DRD target.
- **4.** Snap the moving object to the orthogonal edges of the frozen halo or to the orthogonal edge of the DRD target and the edge of the frozen halo.

In the example below, the object is snapped to the top horizontal edge and the left vertical edge of the frozen halo.



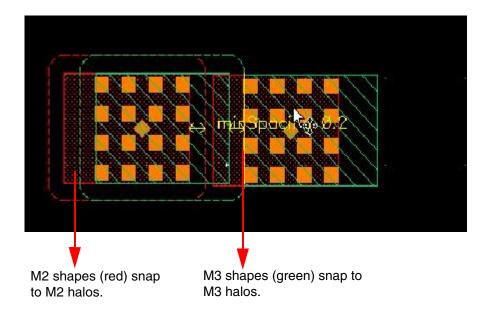
DRD Advanced Features

The following example shows the object snapping to the top horizontal edge of the DRD target and the left vertical edge of the frozen halo.



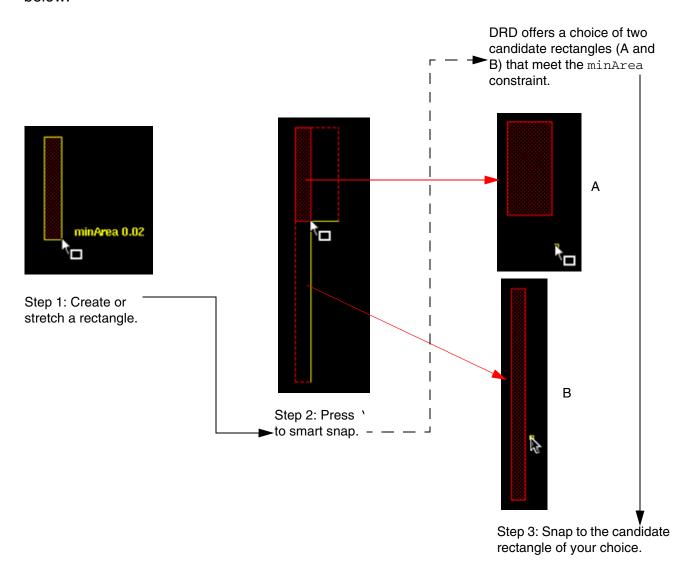
Layer-to-Layer Snapping

In layer-to-layer snapping, snapping occurs between the shapes and halos on the same layer. For example, shapes on layer M2 are snapped only to M2 halos, as shown below in the example of layer-to-layer snapping.



Snapping Based on the minArea Constraint

When creating or stretching a rectangle, DRD offers a choice of candidate rectangles that meet the minArea constraint. You can snap to a candidate rectangle of your choice as shown below.



DRD Advanced Features

Snapping Using the Quick Align Form in Virtuoso

You can snap objects to DRC halos by using the *DRD Smart Snap* option available in the Quick Align form, as shown.

Note: This option is available only when the *Smart Snap* check box is selected in the DRD Options form.

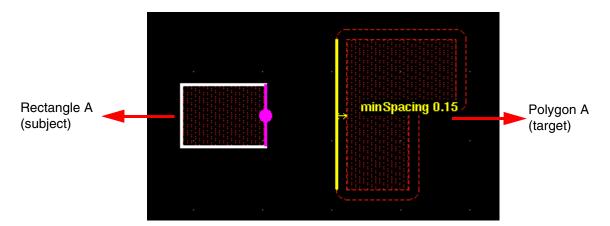


To snap an object by using the Quick Align smart snap feature:

- **1.** Choose *Edit Quick Align* or press A.
- 2. Press F3 to open the Quick Align form.
- 3. Choose the DRD Smart Snap option.
- **4.** Click the right edge of Rectangle A, the subject.
- **5.** Click the left edge of Polygon A, the target.

DRD Advanced Features

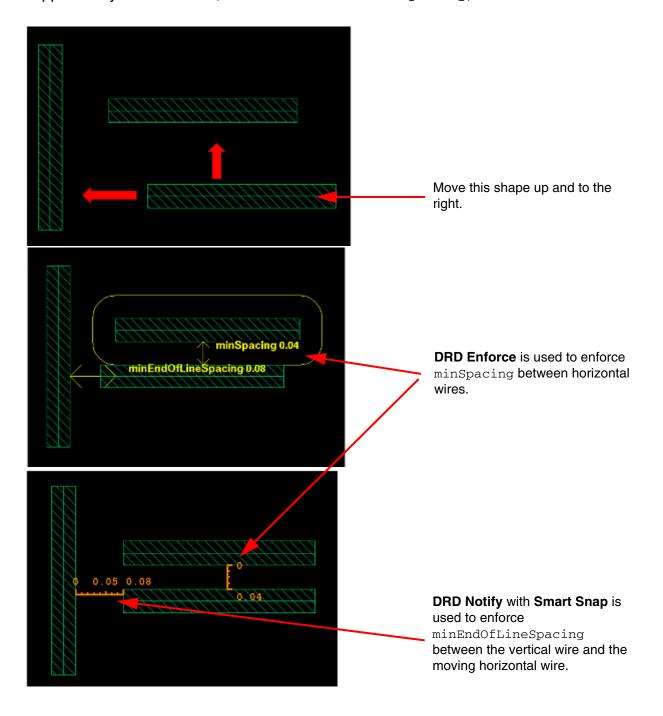
DRD Smart Snap creates halos (the dashed red lines) around Polygon A based on the spacing constraints defined in the technology file.



6. Point to the halo with which you want to align Rectangle A. Click the solid yellow line that appears to snap Rectangle A to the halo.

Snapping with DRD Enforce Enabled

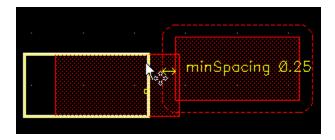
DRD Smart Snap with DRD Enforce enabled lets you use DRD Enforce to enforce constraints such as minSpacing and DRD Notify / Smart Snap to enforce constraints that are not supported by DRD Enforce, such as minEndOfLineSpacing, as shown.



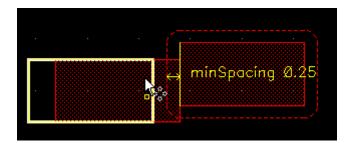
DRD Advanced Features

Toggling DRD Smart Snap during Interactive Editing

You can press 1 on the keyboard to temporarily disable smart snapping while maintaining the frozen halo in your layout, as shown.



To re-enable smart snapping, press 1 again.



Incremental Violation Display

The incremental violation display feature is available by default in the Notify mode. It lets you see and analyze the DRC violations incrementally. If you see any violations on the canvas while editing a layout with the Notify mode ON, you can enter into the incremental violation display mode by pressing the Ctrl+Shift+m bindkey. In the incremental violation display mode, the current edit operation is suspended, and you can see the violations one by one using the up and down arrow keys.

The incremental violation display feature also offers a zoomed-in view of the objects involved in violations. To do this, select the *Zoom in Incremental Violation* check box on the *Interactive* tab of the DRD Options form.

Press Esc to exit the incremental violation display mode. On exiting the mode, you do not lose the context of editing, and the pointer is displayed at the same position where you left before entering the incremental violation display mode.

DRD Advanced Features

The incremental violation display supports the *Copy*, *Create Instance*, *Create Multipart Path*, *Create P&R Boundary*, *Create Path*, *Create Pin*, *Create Rectangle*, *Create Via*, *Create Wire*, *Move*, *Reshape*, *Split*, and *Stretch* commands.



For a quick overview of the incremental violation display feature, see <u>Using the DRD</u> <u>Incremental Violation Display Feature</u>.

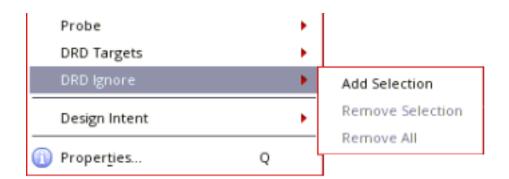
Excluding Instances from Checking

You can choose to exclude some instances from DRD checking. If the *Ignore Marked Instance* check box in the *Filters* page of the DRD Options form is selected, DRD does not perform checks on the marked instances and between the marked instances and its neighboring shapes. This is applicable for both batch and interactive modes.

The drdAddIgnore property is added to an instance marked to be excluded and the setting is saved when you save the design. It is removed when the instance is unmarked.

To mark an instance, do the following:

→ On the canvas or in the Navigator assistant, right-click the instance and choose DRD Ignore — Add Selection from the context menu.



To unmark an instance, do the following:

→ On the canvas or in the *Navigator* assistant, right-click a marked instance and choose *DRD Ignore* — *Remove Selection* from the context menu.

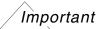
To unmark all instances, do the following:

→ Right-click a marked instance in the Navigator assistant or anywhere on the canvas and choose DRD Ignore — Remove All from the context menu.

3

DRD Interactive Compactor

This section describes the DRD interactive compactor, which you can use to compact the selected objects to the closest violation-free location according to the spacing rules defined in the Virtuoso technology file, or a custom spacing, which you can specify in the DRD Compactor Options form. You can also compact the selected objects against a specified target object (an anchor).



The compactor is available only in VLS XL and VLS GXL tiers.



For a quick overview of DRD Compactor, see <u>Using DRD Compactor</u>.

This chapter contains the following topics:

- Overview of DRD Compactor Features
- Prerequisites for Using the Compactor
- Launching the DRD Compactor Options Form
- Compactor User Interface
- Running the Compactor
- Examples of Compaction

DRD Interactive Compactor

Overview of DRD Compactor Features

The compactor supports the following features:

- Compaction of vias and orthogonal and 45-degree rectangles, polygons, and wires
 For examples of compaction of 45-degree shapes, see
 - □ Example 8: Compacting 45-degree wires in Virtuoso Partial Select mode
 - □ Example 9: Compacting 45-degree polygons in Virtuoso Partial Select mode.
- Compaction of instances

Note: The compactor does not compact objects inside an instance against each other.

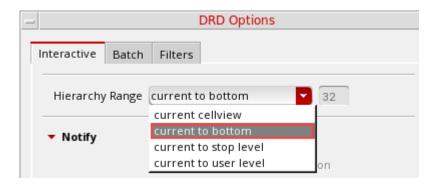
- 1-D orthogonal compaction in the left, right, top, or bottom directions using icons in the *DRD Compactor* toolbar or the default bindkeys
- Compaction with move and stretch effect, which
 - Moves shapes if Virtuoso Full Select mode is enabled as shown in <u>Example 5</u>:
 <u>Compacting objects in Virtuoso Full Select mode</u>
 - Stretches shapes if Virtuoso Partial Select mode is enabled as shown in <u>Example</u>
 6: Compacting objects in Virtuoso Partial Select mode
- Compaction according to the single- and multi-layer spacing constraints supported by DRD Enforce
- Placement of shapes next to each other after compaction if no spacing constraints are defined in the technology file and no custom spacing is specified
- Undo compaction results using the *Undo* command in Virtuoso
- Compaction in several modes as described in Compaction Mode
- Setting the compactor options in the <u>The DRD Compactor Options Form</u>
- Running the compactor in two ways:
 - Using the Compactor GUI
 - □ Using SKILL Functions and Bindkeys

Note: Compactor does not support path shapes and derived layers.

Prerequisites for Using the Compactor

Before using the compactor, do the following:

- 1. Check that your technology file includes the required spacing constraints for the layers you want to use for compaction.
- **2.** If you are compacting hierarchical objects, such as instances:
 - **a.** In the <u>DRD Options</u> form, choose *Hierarchy Range* as *current to bottom*.



b. Set the display hierarchy depth to a value greater than zero using the display stop level options.

You set the hierarchy depth using one of the following display stop level options in Virtuoso:

- The icon in the Options toolbar.
- The Display Options form, which is launched by choosing Options Display.

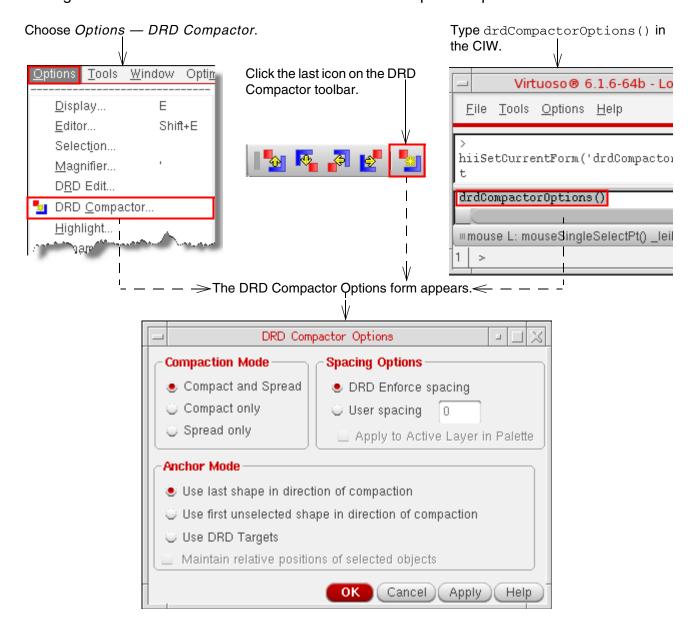


Launching the DRD Compactor Options Form

You can launch the DRD Compactor Options form using one of the following options:

- The *Options* menu in Virtuoso
- The DRD Compactor toolbar
- The <u>drdCompactorOptions</u> SKILL function

The figure below illustrates how to launch the DRD Compactor Options form.



DRD Interactive Compactor

Compactor User Interface

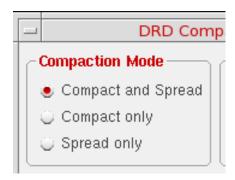
The compactor user interface comprises the following elements:

- The DRD Compactor Options Form
- The DRD Compactor Toolbar and Bindkeys

The DRD Compactor Options Form

You can choose the compaction mode, the spacing option, and the anchor mode using the DRD Compactor Options form.

Compaction Mode



The treatment of the objects being compacted differs as detailed in the table below.

Original distance of objects	Compact and Spread (default mode)	Compact only	Spread only
More than the new spacing value	Compacted against each other	Compacted against each other	Not spread
Less than the new spacing value	Spread away from each other	Not compacted	Spread away from each other



You can change the default compaction mode using the <u>drdEditCompactorMode</u> environment variable.

DRD Interactive Compactor

Spacing Options



Objects are compacted or spread according to the spacing option chosen in this group box.

■ DRD Enforce spacing (default)

Compacts and spreads according to the spacing constraints that are defined in the technology file and supported by DRD Enforce.

User spacing

Compacts and spreads according to a custom spacing value. Select the field and type the required spacing value.

For an application of this spacing option, see <u>Example 1: Compacting objects with different spacing values so that all the objects are spaced equally</u>.

Apply to Active Layer in Palette applies User spacing to only the shapes on the currently active layer.

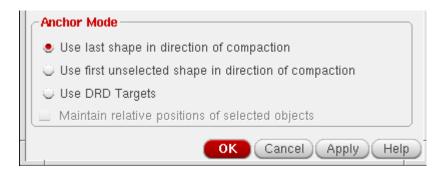


You can change the default spacing option using the drdEditCompactorSpacingMode environment variable.

You can set the default custom spacing value using the <a href="https://dream.org/drea

DRD Interactive Compactor

Anchor Mode



Anchors are the target objects in the layout against which other objects are compacted.



You can set the default anchor mode using the <u>drdEditCompactorAnchorMode</u> environment variable.

■ Use last shape in direction of compaction (default)

Compaction occurs only within a set of selected objects. The last object in the direction of compaction is used as the target for compaction. For example, if you are compacting left, the leftmost object is used as the target. The target object remains static; the remaining objects are compacted or spread against the target object.

For an application of this mode, see <u>Example 2: Compacting within a selected set of objects</u>.

■ Use first unselected shape in direction of compaction

The first unselected object found in the direction of compaction is used as the target. The selected objects are compacted or spread against the target object.

Note: If there is no unselected object in the direction of compaction within the visible area of the layout canvas, the selected objects are compacted against the edge of the layout window.

For an application of this mode, see <u>Example 3: Compacting the selected objects against</u> an <u>unselected object</u>.

■ Use DRD Targets

The selected objects are compacted against a DRD Target object. The target object can be inside or outside the visible area of the layout canvas.

DRD Interactive Compactor

/Important

If your design does not contain DRD targets, you must set an object as a target before you proceed with the compaction. For more information, see <u>Setting DRD Targets</u>.

Maintain relative positions of selected objects

Maintains the original spacing between the selected objects during compaction. For example, if you have selected three objects for compaction, the object closest to the anchor is compacted against the anchor and the original spacing between the three objects remains unchanged.

For an application of this field, see <u>Example 4: Maintaining the original spacing between the selected objects during compaction</u>.

Note: This field is enabled for selection only when one of the following anchor modes is selected:

- Use first unselected shape in direction of compaction
- □ Use DRD Targets



You can select this field by default using the <u>drdEditCompactorInterOnly</u> environment variable.

DRD Interactive Compactor

The DRD Compactor Toolbar and Bindkeys



If the DRD Compactor toolbar is not visible in the Virtuoso toolbars section, choose *Window — Toolbars — DRD Compactor* to make it visible.

You can click the icons on the toolbar to do the following:

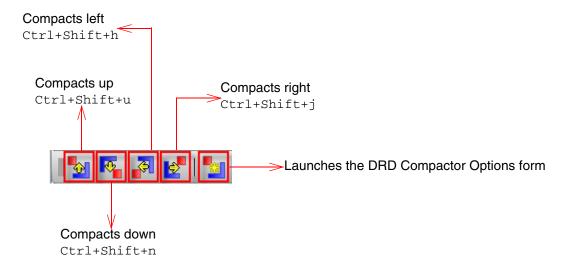
■ Launch the <u>DRD Compactor Options</u> form.

For information about launching the form, see <u>Launching the DRD Compactor Options</u> Form.

Compact objects in different directions.

As an alternative, you can also use the default compactor bindkeys.

The following figure shows the functionality of the icons, the default compactor bindkeys, and the icon-to-bindkey mapping.



You can also customize the bindkeys using the hiSetBindKey command, as shown in the following example.

```
hiSetBindKey("Layout", "<Ctrl>Left", "drdCompactSelSet(\"L\",
\"compactAndSpread\")" )
hiSetBindKey("Layout", "<Ctrl>Right", "drdCompactSelSet(\"R\",
\"compactAndSpread\")" )
hiSetBindKey("Layout", "<Ctrl>Up", "drdCompactSelSet(\"T\",
\"compactAndSpread\")" )
hiSetBindKey("Layout", "<Ctrl>Down", "drdCompactSelSet(\"B\",
\"compactAndSpread\")" )
```

DRD Interactive Compactor

In the example above, the compact and spread mode has been assigned to all the bindkeys. Each bindkey calls the drdCompactSelSet SKILL function. Ctrl+left arrow compacts left, Ctrl+right arrow compacts right, Ctrl+top arrow compacts up, and Ctrl+bottom arrow compacts down.

Running the Compactor

This section provides the generic steps to run the compactor. For different examples of compaction and the steps specific to each example, see <u>Examples of Compaction</u>.

You can run the compactor in two ways:

- Using the Compactor GUI
- Using SKILL Functions and Bindkeys

/Important

Before running the compactor, ensure that the prerequisites for using the compactor are met. For more information, see <u>Prerequisites for Using the Compactor</u>.

Using the Compactor GUI

To run the compactor using the GUI, do the following:

- **1.** On the layout canvas, select the objects that you want to compact.
- **2.** Choose *Options DRD Compactor* from the layout window menu bar.
- **3.** In the <u>DRD Compactor Options</u> form, do the following:
 - **a.**Select the Compaction Mode.
 - **b.** Select the **Spacing Options**.
 - c. Select the Anchor Mode.
 - **d.** Click *OK* after setting the compactor options as required.
- **4.** On the DRD Compactor toolbar, click the appropriate icon to compact the selected objects in the required direction.

DRD Interactive Compactor

Specifying the Number of Shapes for Compaction

You can use the <u>maxShapesAllowedForCompaction</u> environment variable to set the number of shapes that can be compacted. The default value is 500.

If the number of shapes involved in compaction is greater than the value set in this environment variable, the compactor does not compact the selected objects and displays a warning message in the CIW.

Interrupting a Compaction Job

If a compaction job takes too long to complete, you can manually interrupt it by pressing Ctrl+C. A message stating that compaction was stopped by the user is displayed in CIW.

Using SKILL Functions and Bindkeys

To run the compactor using SKILL functions and bindkeys, do the following:

- **1.** On the layout canvas, select the objects that you want to compact.
- **2.** To compact the objects, run the drdCompactSelSet SKILL function using one of the following alternatives:
 - Type the SKILL function in the CIW and specify the direction of compaction using the $s_direction$ parameter. For example, drdCompactSelSet('top) compacts the selected objects in the upward direction.

Note: Other parameters for the SKILL function are optional. If you do not set the optional parameters, the compactor uses the default compaction mode, that is, compact and spread.

Assign the SKILL function to a bindkey by specifying a direction (for more information, see <u>The DRD Compactor Toolbar and Bindkeys</u>) and press the appropriate bindkey to run the SKILL function.



You can compact objects in batch mode using the <u>drdCompactObjects</u> SKILL function.

Examples of Compaction

This section illustrates different examples of compaction. Each example contains the beforeand-after compaction scenarios, along with the scenario-specific steps to run the compactor. For the generic steps to run the compactor, see <u>Running the Compactor</u>.



Before running the compactor, ensure that the prerequisites for using the compactor are met. For more information, see <u>Prerequisites for Using the Compactor</u>.

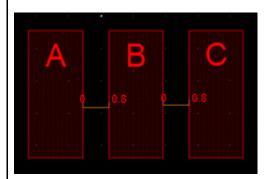
You must launch the DRD Compactor Options form and change the settings in the form, as described in each example. For more information about how to launch the form, see <u>Launching the DRD Compactor Options Form</u>. Click *OK* after setting the compactor options as required.

Example 1: Compacting objects with different spacing values so that all the objects are spaced equally

Before Compaction

Rectangles A and B are spaced at 0.5 micron; B and C are spaced at 1.5 microns.

After Compaction



The spacing between A and B, and B and C is corrected to 0.8 micron.

- **1.** Select A, B, and C, as shown in the Before Compaction figure.
- **2.** In the <u>DRD Compactor Options</u> form:
 - □ Ensure that *Compaction Mode* is set to *Compact and Spread*.
 - ☐ In *Spacing Options*, select *User spacing* and type 0.8.

3. On the DRD Compactor toolbar, click Compact selected objects left.



As shown in the After Compaction figure, the compactor does the following:

- □ Spreads rectangles A and B by 0.3 micron—from 0.5 to 0.8 micron
- □ Compacts rectangles B and C by 0.7 micron—from 1.5 to 0.8 micron

Example 2: Compacting within a selected set of objects

Before Compaction After Compaction Three of the selected rectangles need to be compacted against a selected rectangle. After Compaction Three selected rectangles are compacted against rectangle A.

- 1. Select the four rectangles, as shown in the Before Compaction figure.
- 2. In the <u>DRD Compactor Options</u> form:
 - □ Ensure that *Compaction Mode* is set to either *Compact and Spread* or *Compact only*.
 - □ In Anchor Mode, ensure that Use last shape in direction of compaction is selected.
- **3.** On the DRD Compactor toolbar, click *Compact selected objects right*.



DRD Interactive Compactor

As shown in the After Compaction figure, the compactor moves three of the selected rectangles against rectangle A because it is the last selected shape in the direction of compaction.

Example 3: Compacting the selected objects against an unselected object

After Compaction After Compaction The selected rectangles need to be compacted against an unselected rectangle. The selected rectangle A. B is compacted against the edge of the layout window.

- 1. Select the five rectangles as shown in the Before Compaction figure.
- 2. In the <u>DRD Compactor Options</u> form:
 - □ Ensure that *Compaction Mode* is set to either *Compact and Spread* or *Compact only*.
 - □ In Anchor Mode, select Use first unselected shape in direction of compaction.

DRD Interactive Compactor

3. On the DRD Compactor toolbar, click Compact selected objects left.

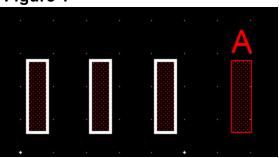


As shown in the After Compaction figure, the compactor moves the selected rectangles against the first unselected object found on the same layer, that is, rectangle A. Rectangle B is moved against the edge of the layout window because there is no unselected object in the direction of compaction to obstruct its movement.

Example 4: Maintaining the original spacing between the selected objects during compaction

Before Compaction

Figure 1



The selected rectangles need to be compacted against rectangle A.

After Compaction—Normal

Figure 2

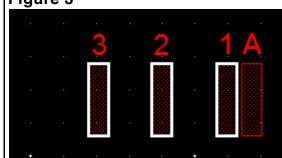


Rectangles 1, 2, and 3 are moved against rectangle A.

Note: The original spacing between 1, 2, and 3 has not been maintained.

After Compaction—Maintained Spacing

Figure 3



Rectangles 1, 2, and 3, with their original spacing intact, are moved against rectangle A.

- **1.** Select the three rectangles, as shown in Figure 1.
- **2.** In the <u>DRD Compactor Options</u> form:
 - a. Ensure that Compaction Mode is set to either Compact and Spread or Compact only.
 - **b.**In Anchor Mode, select Use first unselected shape in direction of compaction.

- c. Select Maintain relative positions of selected objects.
- 3. On the DRD Compactor toolbar, click Compact selected objects right.



As shown in Figure 3, the compactor moves rectangles 1, 2, and 3 against rectangle A, keeping the original spacing between the three rectangles intact after compaction.

Figure 2 illustrates the After Compaction result when the *Maintain relative positions* of selected objects field is not selected.

Example 5: Compacting objects in Virtuoso Full Select mode

Before Compaction After Compaction

1. On the Virtuoso Options toolbar, enable Full Select mode.



- **2.** Select the buses, as shown in the Before Compaction figure.
- **3.** In the <u>DRD Compactor Options</u> form:

The selected buses need to be compacted against bus A in Virtuoso Full Select mode.

- □ Ensure that *Compaction Mode* is set to either *Compact and Spread* or *Compact only*.
- □ In Anchor Mode, ensure that Use last shape in direction of compaction is selected.

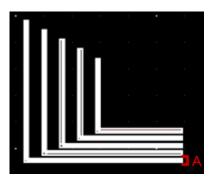
The buses are compacted against bus A.

DRD Interactive Compactor

4. On the DRD Compactor toolbar, click Compact selected objects down.



As shown below the horizontal segments of four buses are moved against bus A, which remains static.



5. Click Compact selected objects left.



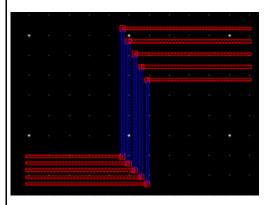
As shown in the After Compaction figure, the vertical segments of four buses are moved against bus A, which remains static.

Example 6: Compacting objects in Virtuoso Partial Select mode

Before Compaction

The buses with the connecting vias need to be compacted in Virtuoso Partial Select mode.

After Compaction

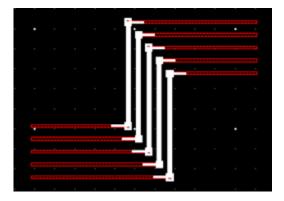


The lower horizontal and the vertical segments of the buses are compacted.

1. On the Virtuoso Options toolbar, enable Partial Select mode.



2. Select the vertical segments of the buses and the connecting vias, as shown in the following figure.



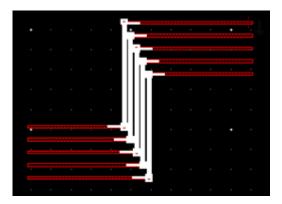
- 3. In the <u>DRD Compactor Options</u> form:
 - □ Ensure that *Compaction Mode* is set to either *Compact and Spread* or *Compact only*.
 - ☐ In Anchor Mode, ensure that Use last shape in direction of compaction is selected.

DRD Interactive Compactor

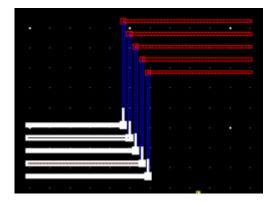
4. On the DRD Compactor toolbar, click Compact selected objects left.



As shown in the following figure, the vertical segments of the buses are compacted and the horizontal segments are stretched or shortened.



5. Select the lower horizontal segments, as shown in the following figure.



6. Click Compact selected objects down.

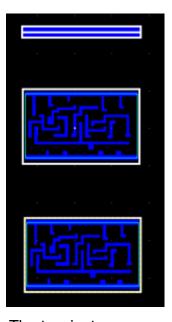


As shown in the After Compaction figure, the lower horizontal segments are compacted and the vertical segments are stretched.

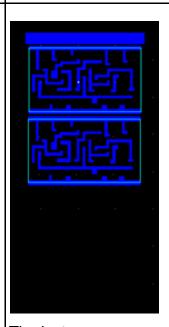
Example 7: Compacting instances

Before Compaction





The two instances need to be compacted against the wire.



The instances are compacted against the wire.

- 1. Select the instances and the wire as shown in the Before Compaction figure.
- **2.** In the <u>DRD Options</u> form, choose *Hierarchy Range* as *current to bottom*.
- **3.** Set the hierarchy depth to a value greater than zero using the display stop level icon on the Options toolbar.



- **4.** In the <u>DRD Compactor Options</u> form:
 - □ Ensure that *Compaction Mode* is set to *Compact and Spread* or *Compact only*.
 - □ Ensure that *Anchor Mode* is set to *Use last shape in direction of compaction*.
- **5.** On the DRD Compactor toolbar, click *Compact selected objects up*.



As shown above, the compactor moves the instances against the wire.

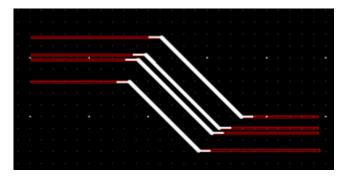
Example 8: Compacting 45-degree wires in Virtuoso Partial Select mode

Before Compaction After Compaction The 45-degree wires need to be compacted in Virtuoso Partial Select mode.

1. On the Virtuoso Options toolbar, enable Partial Select mode.



2. Select the diagonal segments of the wires, as shown in the following figure.



- 3. In the <u>DRD Compactor Options</u> form:
 - □ Ensure that *Compaction Mode* is set to either *Compact and Spread* or *Compact only*.
 - □ In *Anchor Mode*, ensure that *Use last shape in direction of compaction* is selected.
- **4.** On the DRD Compactor toolbar, click *Compact selected objects left*.



DRD Interactive Compactor

As shown in the following figure, the diagonal segments of the wires are compacted and the horizontal segments are stretched or shortened.



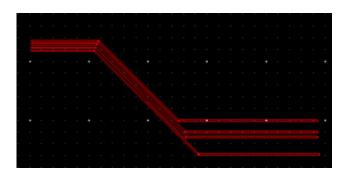
5. Select the upper horizontal segments, as shown in the following figure.



6. Click Compact selected objects up.

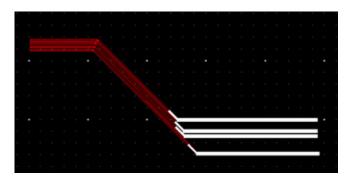


As shown below, the upper horizontal segments are compacted and the diagonal segments are stretched.



DRD Interactive Compactor

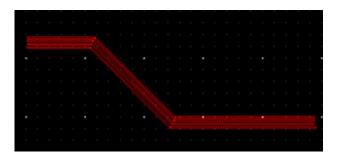
7. Select the lower horizontal segments, as shown in the following figure.



8. Click Compact selected objects down.



As shown in the following figure, the lower horizontal segments are compacted and the diagonal segments are shortened.



Example 9: Compacting 45-degree polygons in Virtuoso Partial Select mode

Before Compaction After Compaction After Compaction Polygon A is compacted.

Procedure:

1. On the Virtuoso Options toolbar, enable Partial Select mode.



- 2. Select the diagonal edges of the polygons, as shown in the Before Compaction figure.
- **3.** In the <u>DRD Compactor Options</u> form:
 - □ Ensure that *Compaction Mode* is set to either *Compact and Spread* or *Compact only*.
 - □ In Anchor Mode, ensure that Use last shape in direction of compaction is selected.
- 4. On the DRD Compactor toolbar, click Compact selected objects left.



As shown in the After Compaction figure, the highlighted edges of polygon A are stretched and the diagonal edge shrunk to enlarge the polygon.

Virtuoso Design Rule Driven Editing User Guide DRD Interactive Compactor

4

DRD Support for FinFET Devices (ICADVM20.1 Only)

DRD facilitates the creation and placement of FinFET devices that adhere to FinFET-specific spacing, width, and enclosure rules.

This chapter contains the following topics:

- Defining Discrete Spacing Rules
- Defining Minimum Spacing Rules
- <u>Defining Directional and Discrete Width Rules</u>
- Defining Directional and Discrete PR Boundary Rules
- Snapping Objects to Legal Width Values
- Defining Directional and Discrete Opposite Extension Rules



For a guick overview of DRD, see <u>DRD for Advanced Node Design</u>.

Defining Discrete Spacing Rules

The allowedSpacingRanges constraint defines a set of legal spacing rules, in the form of ranges and discrete values, between shapes on the same layer or different layers. DRD uses the following parameters of this constraint to support discrete spacing rules for FinFETs:

- stepSize: Specifies discrete incremental spacing values between shapes on the same layer.
- overLayer: Specifies discrete incremental spacing values between shapes on a layer that are located over a shape on another layer.

DRD Support for FinFET Devices (ICADVM20.1 Only)

An example of the technology file definition is shown below:

where,

- Shapes on layer Active are located over a shape on layer FB1.
- The spacing between these shapes is measured vertically and the minimum legal spacing value is 0.13 user units.
- As the spacing increases, the legal spacing values are calculated as increments of 0.048 user units, as shown below.

```
0.130

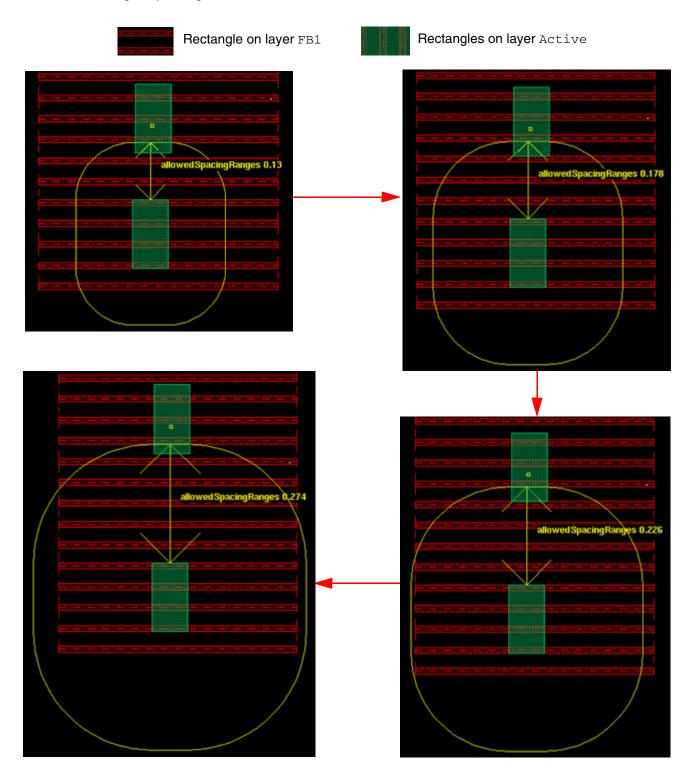
0.130 + (1 \times 0.048) = 0.178

0.130 + (2 \times 0.048) = 0.226

0.130 + (3 \times 0.048) = 0.274
```

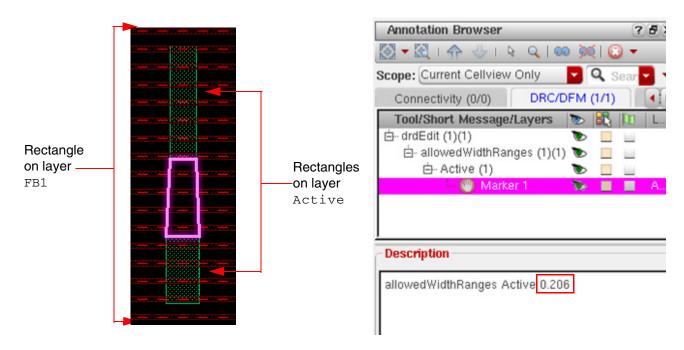
DRD Support for FinFET Devices (ICADVM20.1 Only)

The following figures illustrate this example in DRD Notify mode. As the vertical spacing between the rectangles increases, DRD displays halos and arrows (in yellow) to indicate the incremental legal spacing values.



DRD Support for FinFET Devices (ICADVM20.1 Only)

The following figure illustrates this example in DRD Post-Edit and Batch-Check modes. As shown in the Annotation Browser assistant, DRD reports an allowedSpacingRanges violation (Marker 1) because the vertical space between the rectangles does not meet the nearest legal spacing value (0.226 user units).



Defining Minimum Spacing Rules

The minSpacing constraint defines the minimum legal spacing between fin boundaries on a layer.

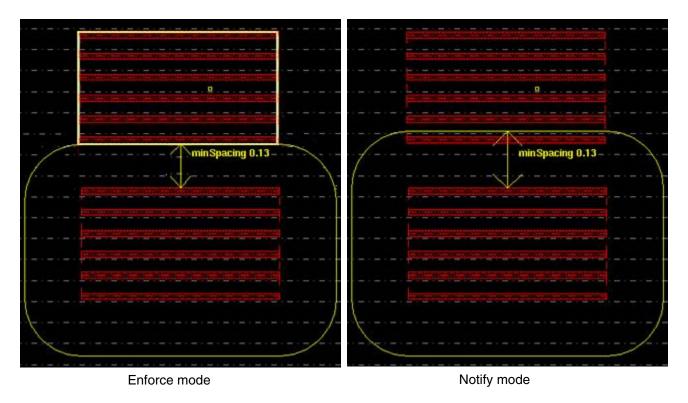
An example of the technology file definition is shown below:

```
( minSpacing "FB1" 0.130 )
```

where the fin boundaries are located on layer FB1 and the minimum legal spacing is 0.130 user units.

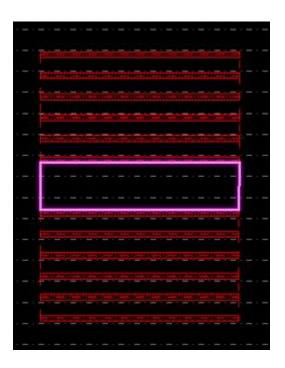
DRD Support for FinFET Devices (ICADVM20.1 Only)

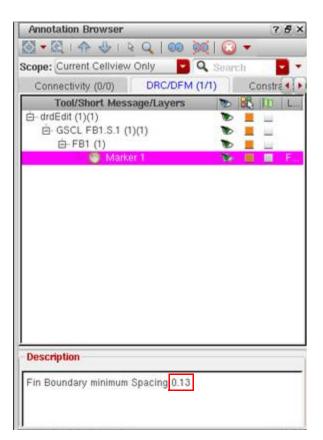
The following figures illustrate this example in DRD Enforce and Notify modes. DRD displays a halo and an arrow to indicate the minimum legal spacing value between two fin boundaries.



The following figure illustrates this example in DRD Post-Edit and Batch-Check modes. As shown in the Annotation Browser, DRD reports a minSpacing violation (Marker 1) because

the spacing between the fin boundaries does not meet the minimum legal spacing value (0.130 user units).





Defining Directional and Discrete Width Rules

The allowedWidthRanges constraint defines a set of legal width rules, in the form of ranges and discrete values, for shapes on a layer. DRD uses the following parameters of this constraint to support directional discrete width rules for FinFETs:

- stepSize: Specifies discrete incremental width values.
- measureVertical and measureHorizontal: Specifies the directions of the discrete incremental width values.

DRD Support for FinFET Devices (ICADVM20.1 Only)

An example of the technology file definition is shown below:

where,

- Shapes are located on layer Active.
- The width is measured vertically and the minimum legal width is 0.062 user units.
- As the shapes are stretched, the legal width values are calculated as increments of 0.048 user units, as shown below.

```
0.062

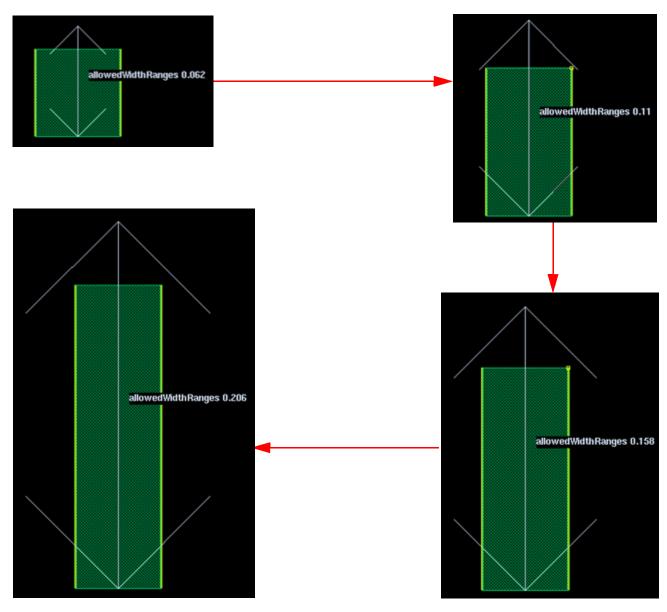
0.062 + 1 x 0.048 = 0.11

0.062 + 2 x 0.048 = 0.158

0.062 + 3 x 0.048 = 0.206
```

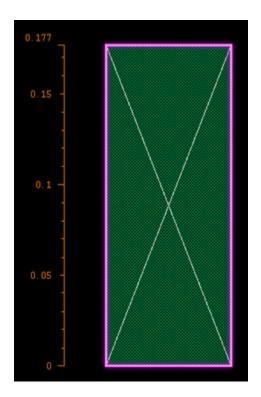
DRD Support for FinFET Devices (ICADVM20.1 Only)

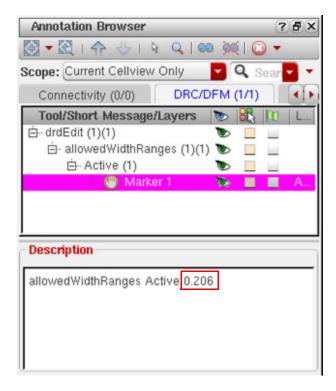
The following figures illustrate this example in DRD Notify mode. As the rectangle is stretched vertically, DRD displays arrows to indicate the incremental legal width values.



The following figure illustrates this example in DRD Post-Edit and Batch-Check modes. As shown in the Annotation Browser, DRD reports an allowedWidthRanges violation (Marker

1) because the vertical dimension of the rectangle does not conform to the nearest legal width value (0.206 user units).





Defining Directional and Discrete PR Boundary Rules

You can define PR boundary rules through the following constraints:

- allowedPRBoundaryDimensions; see <u>Defining Directional and Discrete PR</u>
 <u>Boundary Width Rules</u>
- minPRBoundaryInteriorHalo; see <u>Defining Directional and Discrete PR Boundary</u>
 <u>Spacing Rules</u>

Defining Directional and Discrete PR Boundary Width Rules

The allowedPRBoundaryDimensions constraint defines a set of legal width rules, in the form of ranges and discrete values, for PR boundaries. DRD uses the following parameters of this constraint to support discrete width rules for FinFETs:

■ stepSize: Specifies discrete incremental width values.

DRD Support for FinFET Devices (ICADVM20.1 Only)

■ vertical and horizontal: Specifies the horizontal or vertical discrete incremental width values.

An example of the technology file definition is shown below:

where,

- The PR boundary width is measured vertically and the minimum legal width is 0.048 user units.
- As the PR boundary is stretched, the legal width values are calculated as increments of 0.048 user units, as shown below.

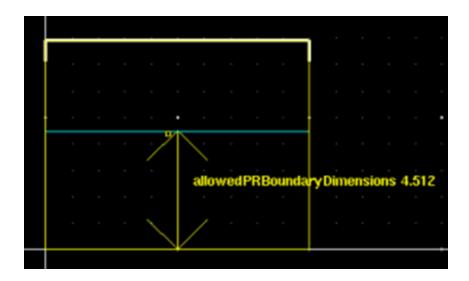
```
0.048

0.048 + 1 \times 0.048 = 0.096

0.048 + 2 \times 0.048 = 0.144

0.048 + 3 \times 0.048 = 0.192
```

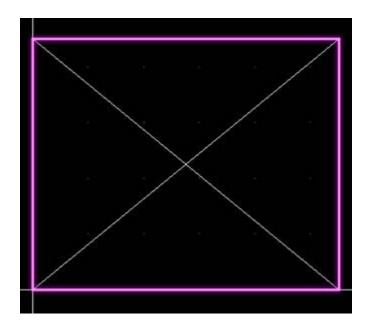
The following figure illustrates this example in DRD Notify mode. As the PR boundary is stretched vertically, DRD displays arrows to indicate the nearest incremental legal width value.

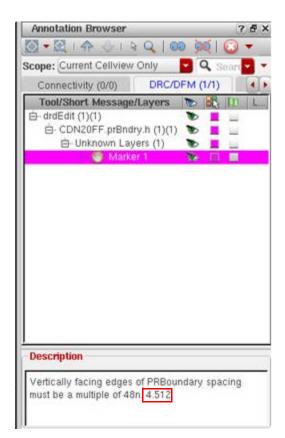


The following figure illustrates this example in DRD Post-Edit and Batch-Check modes. As shown in the Annotation Browser, DRD reports an allowedPRBoundaryDimensions

DRD Support for FinFET Devices (ICADVM20.1 Only)

violation (Marker 1) because the vertical dimension of the PR boundary does not conform to the nearest legal width value (4.512 user units).





Defining Directional and Discrete PR Boundary Spacing Rules

The minPRBoundaryInteriorHalo constraint defines a set of legal spacing values between a shape on a layer and the surrounding PR boundary. DRD uses the following parameters of this constraint to support discrete spacing rules for FinFETs:

- stepSize: Specifies discrete incremental spacing values.
- spacingDirection: Specifies the horizontal or vertical spacing direction.

An example of the technology file definition is shown below:

DRD Support for FinFET Devices (ICADVM20.1 Only)

where,

- The spacing between the edges of shapes on layer Active and the surrounding PR boundary is measured vertically and the minimum legal spacing is 0.113 user units.
- As the spacing increases, the legal spacing values are calculated as increments of 0.048 user units, as shown below.

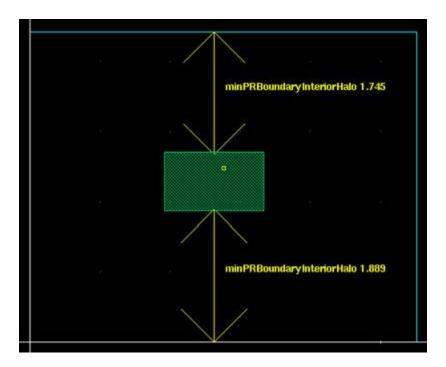
```
0.113

0.113 + 1 x 0.048 = 0.161

0.113 + 2 x 0.048 = 0.209

0.113 + 3 x 0.048 = 0.257
```

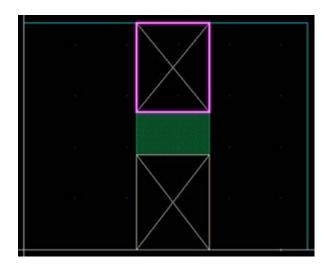
The following figure illustrates this example in DRD Notify mode. As the vertical spacing between the PR boundary and the enclosed rectangle increases, DRD displays arrows to indicate the incremental legal spacing values.

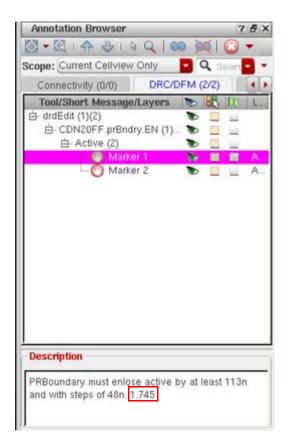


The following figure illustrates this example in DRD Post-Edit and Batch-Check modes. As shown in the Annotation Browser, DRD reports a minPRBoundaryInteriorHalo violation

DRD Support for FinFET Devices (ICADVM20.1 Only)

(Marker 1) because the vertical spacing between the PR boundary and the enclosed rectangle does not conform to the nearest legal spacing value (1.745 user units).



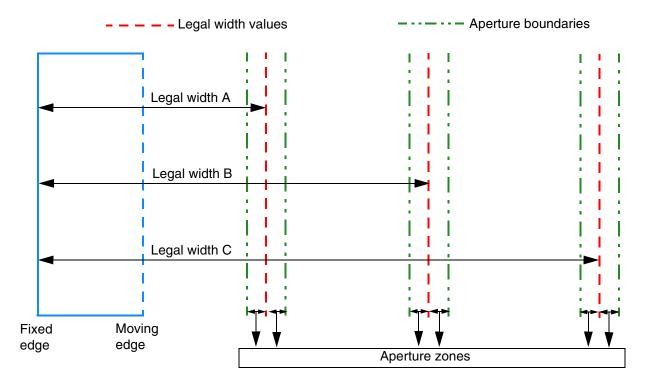


Snapping Objects to Legal Width Values

You can automatically snap specific objects to the nearest legal width value using discrete dimension gravity. This functionality supports the *Create* and *Stretch* commands in Virtuoso for the following objects:

- Rectangles and polygons; for illustrations, see Working with Shapes
- Pins; for illustrations, see Working with Pins
- Islands of shapes; for illustrations, see Working with Islands
- PR boundaries; for illustrations, see Working with PR Boundaries

The following figure explains this functionality.



Each legal width value is surrounded by aperture boundaries, which form an aperture zone. An aperture zone is the snap distance between the pointer, or the moving edge of the rectangle, and the legal width value.

As you stretch the rectangle horizontally, the rectangle automatically snaps to legal width A as soon as the pointer enters the aperture zone surrounding legal width A. If you continue to stretch the rectangle, it snaps to legal widths B, and then to C, as soon as the pointer enters their respective aperture zones.

DRD Support for FinFET Devices (ICADVM20.1 Only)

Discrete dimension gravity is turned on by default. To turn off discrete dimension gravity, set the <u>drdEditSmartSnapAllowedWidthSnap</u> environment variable to nil.

Important

Before using discrete dimension gravity, do the following to define the legal width values in your technology file:

- Add the allowedWidthRanges constraint for shapes. For more information, see Defining Directional and Discrete PR Boundary Width Rules.
- Add the allowedPRBoundaryDimensions constraint for PR boundaries. For more information, see <u>Defining Directional and Discrete PR Boundary Width Rules</u>.

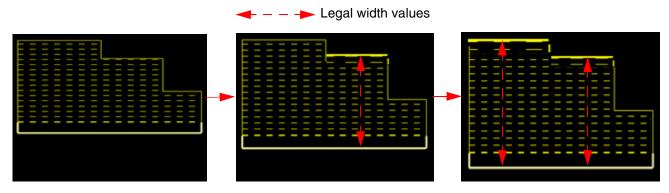
Examples of Discrete Dimension Gravity

This functionality is useful while:

- Working with Shapes
- Working with Pins
- Working with Islands
- Working with PR Boundaries

Working with Shapes

The following figure illustrates discrete dimension gravity during a single-direction polygon stretch operation:

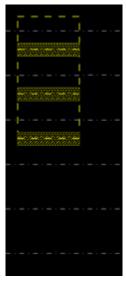


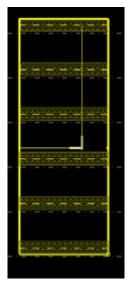
The bottom edge of the polygon is being stretched vertically

The polygon adheres to legal width values by automatically snapping to the nearest legal width values

DRD Support for FinFET Devices (ICADVM20.1 Only)

The following figure illustrates discrete dimension gravity during a two-direction rectangle stretch operation:





The original rectangle

Vertical and horizontal stretch

The rectangle abides by legal width values by automatically snapping to the nearest legal width values

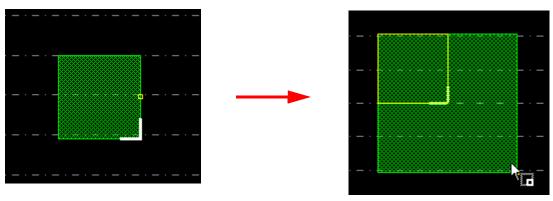
Working with Pins

When a pin is created or stretched, it automatically snaps to the nearest legal width value using discrete dimension gravity.

Note: Currently, only rectangular and polygonal pins are supported.

DRD Support for FinFET Devices (ICADVM20.1 Only)

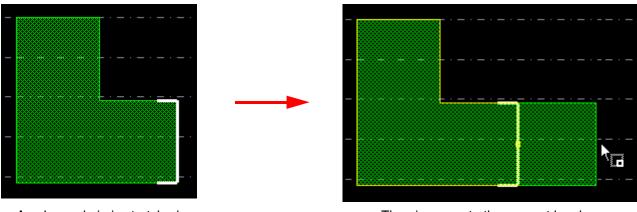
The following figure illustrates how discrete dimension gravity works for pins of type *rectangle*:



A rectangular pin is stretched both horizontally and vertically.

The pin snaps to the nearest legal width values.

The following figure illustrates how discrete dimension gravity works for pins of type *polygon*:



A polygonal pin is stretched horizontally.

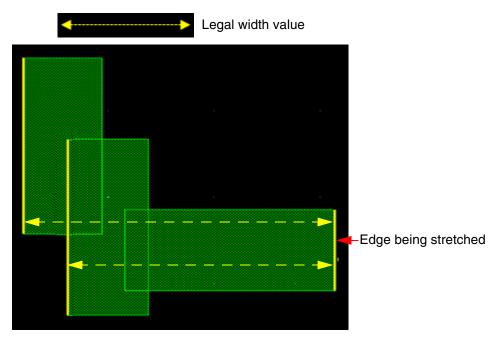
The pin snaps to the nearest legal width values.

Working with Islands

An island is formed when multiple rectangles and polygons on a layer, belonging to the same level of layout hierarchy, overlap or abut. A hierarchical island is formed by shapes on a layer that belong to different hierarchical levels.

DRD Support for FinFET Devices (ICADVM20.1 Only)

The following figure illustrates discrete dimension gravity while stretching a shape in an island of three rectangles.



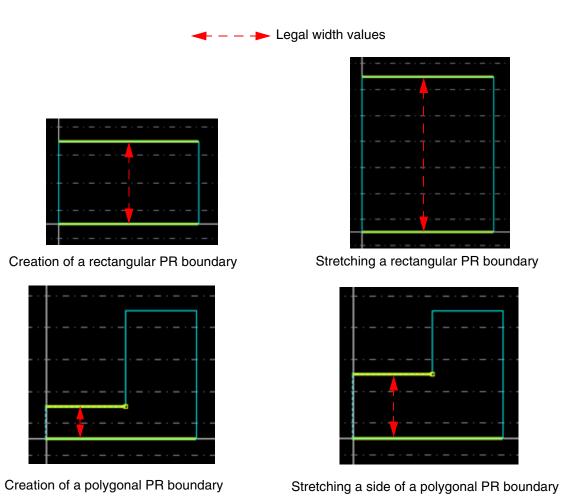
The island abides by legal width values by automatically snapping to the nearest legal width values

Discrete dimension gravity works in the same way while creating or stretching shapes in hierarchical islands.

Note: This functionality requires at least one top-level shape in a hierarchical island.

Working with PR Boundaries

The following figures illustrate discrete dimension gravity while working with PR boundaries.



The PR boundaries abide by legal width values by automatically snapping to the nearest legal width values

Defining Directional and Discrete Opposite Extension Rules

The minOppExtension constraint defines the total required enclosure between two opposite edges of overlapping shapes on two different layers. DRD uses the following parameters of this constraint to support discrete enclosure rules for FinFETs:

- stepSizePair: Specifies discrete incremental extension values of a fin boundary over the shapes on a layer.
- spacingDirection: Specifies the horizontal or vertical discrete incremental extension values of a fin boundary over the shapes on a layer.

An example of the technology file definition is shown below:

```
spacings(
    ( minOppExtension "FB1" "Active"
    'vertical
    'stepSizePair (0.048 0) (0.096 0.017)
    )
) ;spacings
```

where.

- The fin boundary is located on layer FB1 and a shape is located on layer Active.
- The minimum legal extension of the fin boundary over a shape, when measured vertically, is 0.096 user units.
- The legal vertical extension values are calculated as increments of 0.048 user units, as shown below.

```
0.096

0.096 + 1 x 0.048 = 0.144

0.096 + 2 x 0.048 = 0.192

0.096 + 3 x 0.048 = 0.240

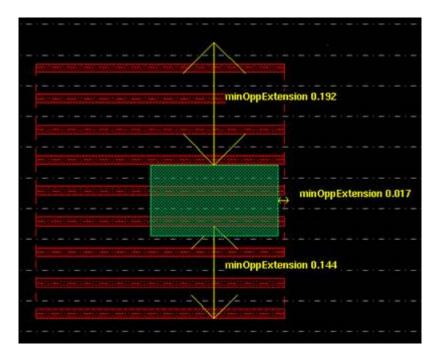
...
```

■ The minimum legal extension of the fin boundary over a shape, when measured horizontally, is 0.017 user units.

The following figures illustrate this example in DRD Notify mode. As the enclosure between the rectangles on ${\tt FB1}$ and ${\tt Active}$ increases vertically and horizontally (to the right), DRD

Virtuoso Design Rule Driven Editing User Guide DRD Support for FinFET Devices (ICADVM20.1 Only)

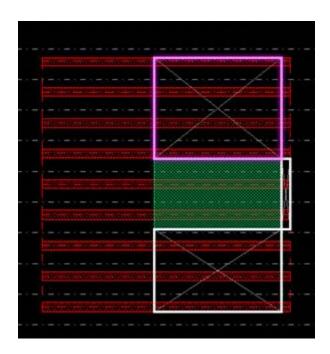
displays arrows to indicate the incremental legal extension values on three sides of the rectangle on layer Active.

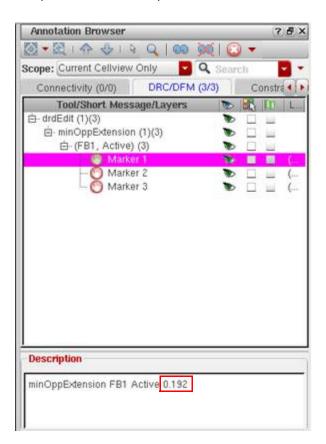


The following figure illustrates this example in DRD Post-Edit and Batch-Check modes. As shown in the Annotation Browser, DRD reports a minOppExtension violation (Marker 1)

DRD Support for FinFET Devices (ICADVM20.1 Only)

because the vertical enclosure between the rectangles on layers FB1 and Active does not conform to the nearest vertical legal enclosure value (0.192 user units).





DRD Constraints Support

This chapter contains the following topics:

- Supported Constraints
 - Supported Constraints and Parameters (IC6.1.8 and ICADVM20.1)
 - □ Supported Constraints and Parameters (ICADVM20.1 Only 95511 and 95800)
 - □ Supported Constraints and Parameters (ICADVM20.1 Only 95512)
 - □ Supported Constraints and Parameters (ICADVM20.1 Only 95511)
- Using DRD Grid Checking
- Specifying Layers in the Interconnect Section
- Supported Derived Layers
- Supported Layer-Purpose Pair and Voltage-Dependent Constraints
 - □ Support for Layer-Purpose Pair Constraints
 - Support for Voltage-Dependent Constraints
- Using Density Constraints
- Using Alternate Foundry Constraint Group

DRD Constraints Support

Supported Constraints

The tables in this section list the constraints supported by DRD. Note the following points:

- Batch-Check mode for supported constraints is available only in VLS XL, VLS GXL (IC6.1.8 Only), and VLS EXL (ICADVM20.1 Only).
- VLS L is available only in the IC6.1.8 release.
- Limited support is available for the 'or operator in DRD editing; it is used mainly to combine constraints of the type extension.

DRD Constraints Support

Supported Constraints and Parameters (IC6.1.8 and ICADVM20.1)

Category	Constraint Name (ASCII Syntax)	Tier Level Support	Support Type
Antenna	<u>antennaRatio</u>	L, XL, GXL, and EXL	notify/post- edit/batch- check
	<u>cumPerLayerAntennaRatio</u>	L, XL, GXL, and EXL	notify/post- edit/batch- check
Area	minArea	L, XL, GXL, and EXL	notify/post- edit/batch- check
	minAreaEdgeLength	L, XL, GXL, and EXL	notify/post- edit/batch- check
	<u>minHoleArea</u>	L (scalar) XL, GXL, EXL (scalar and table)	notify/post- edit/batch- check
	minIsolatedArea	L, XL, GXL, and EXL	notify/post- edit/batch- check
	minPerimeter	L, XL, GXL, and EXL	post-edit/ batch-check
	minRectArea	L, XL, GXL, and EXL	notify/post- edit/batch- check
	minSize	L, XL, GXL, and EXL	notify/post- edit/batch- check

Category	Constraint Name (ASCII Syntax)	Tier Level Support	Support Type
Complex Spacing	bothSidesViaForbiddenSpacingRange	L, XL, GXL, and EXL	notify/post- edit/batch- check
	<u>endOfLineKeepout</u>	XL, GXL, and EXL	notify/post- edit/batch- check
	minEndOfLineCutSpacing	XL, GXL, and EXL	post-edit/ batch-check
	minEndOfLineExtensionSpacing	L, XL, GXL, and EXL	post-edit/ batch-check
	minEndofLinePerpSpacing	L, XL, GXL, and EXL	post-edit/ batch-check
	minEndOfLineSpacing	L, XL, GXL, and EXL	notify/post- edit/batch- check
	minExtensionSpacing	XL, GXL, and EXL	notify/post- edit/batch- check
	minInnerVertexSpacing	L, XL, GXL, and EXL	notify/post- edit/batch- check
	minOppSpanSpacing	L, XL, GXL, and EXL	post-edit/ batch-check
	minParallelSpanSpacing	L, XL, GXL, and EXL	post-edit/ batch-check
	minProtrusionSpacing	XL, GXL, and EXL	post-edit/ batch-check

-			
Category	Constraint Name (ASCII Syntax)	Tier Level Support	Support Type
	minStubInfluenceSpacing	L, XL, GXL, and EXL	notify/post- edit/batch- check
	<u>sameMetalAlignedCuts</u>	L, XL, GXL, and EXL	notify/post- edit/batch- check
	shapeRequiredSpacing	XL, GXL, and EXL	post-edit/ batch-check
Density	maxDensity	XL, GXL, and EXL	batch-check
	maxDiffDensity	XL, GXL, and EXL	batch-check
	minDensity	XL, GXL, and EXL	batch-check

Category	Constraint Name (ASCII Syntax)	Tier Level Support	Support Type
Edge Length	maxDiagonalEdgeLength	L, XL, GXL, and EXL	notify/post- edit/batch- check
	<u>maxNumMinEdges</u>	L, XL, GXL, and EXL	post-edit/ batch-check
	minDiagonalEdgeLength	L, XL, GXL, and EXL	notify/post- edit/batch- check
	<pre>minEdgeAdjacentDistance Note: minEdgeAdjacentDistance is only meant for edgeCount param 2. It does not support edgeCount > 2.</pre>	L, XL, GXL, and EXL	notify/post- edit/batch- check
	minEdgeAdjacentLength	L, XL, GXL, and EXL	post-edit/ batch-check
	minEndOfLineAdjacentToStep	XL, GXL, and EXL	post-edit/ batch-check
	minInsideCornerEdgeLength	L, XL, GXL, and EXL	post-edit/ batch-check
	minOutsideCornerEdgeLength	L, XL, GXL, and EXL	post-edit/ batch-check
	minPerimeter	L, XL, GXL, and EXL	notify/post- edit/batch- check
	minStepEdgeLength	L, XL, GXL, and EXL	notify/post- edit/batch- check

Category	Constraint Name (ASCII Syntax)	Tier Level Support	Support Type
Extension	maxExtension	L, XL, GXL, and EXL	post-edit/ batch-check
	minCenterlineExtension	XL, GXL, and EXL	post-edit/ batch-check
	minEndOfLineEdgeExtension	L, XL, GXL, and EXL	post-edit/ batch-check
	minEndOfLineExtension	L, XL, GXL, and EXL	post-edit/ batch-check
	minExtensionDistance	L, XL, GXL, and EXL	notify/post- edit/batch- check
	minExtensionEdge	L, XL, GXL, and EXL	post-edit/ batch-check
	minInsideCornerExtension	L, XL, GXL, and EXL	post-edit/ batch-check
	minInsideCornerOverlap	L, XL, GXL, and EXL	post-edit/ batch-check
	minOppEndOfLineExtension	L, XL, GXL, and EXL	notify/post- edit/batch- check
	minOppExtension	L (scalar) XL, GXL, and EXL (scalar and table)	notify/post- edit/batch- check

Category	Constraint Name (ASCII Syntax)	Tier Level Support	Support Type
	minOverlapDistance	L, XL,	notify/post-
	Note: Minimum overlap is derived from minExtension.	GXL, and EXL	edit/batch- check
	minPRBoundaryExtension	L, XL, GXL, and EXL	notify/post- edit/batch- check
	minQuadrupleExtension	L, XL, GXL, and EXL	notify/post- edit/batch- check
	minSideExtension Note: This rule is applicable to middle-of-line (MOL) layers, and is not intended for use with cut layers.	L, XL, GXL, and EXL	notify/post- edit/batch- check
	minViaExtension	L, XL, GXL, and EXL	notify/post- edit/batch- check
	minViaJointExtension	L, XL, GXL, and EXL	notify/post- edit/batch- check
Grid	mfgGridResolution	L, XL, GXL, and EXL	post-edit/ batch-check

Category	Constraint Name (ASCII Syntax)	Tier Level Support	Support Type
Length	allowedSpanLengthRanges	L, XL, GXL, and EXL	notify/post- edit/batch- check
	maxDiffusionLength	L, XL, GXL, and EXL	notify/post- edit//batch- check
	MaxLength Note: Applies only to rectangular shapes. Non-rectangular shapes are not flagged as violations and are ignored.	L, XL, GXL, and EXL	notify/post- edit/batch- check
	minLength Note: Applies only to rectangular shapes. Non-rectangular shapes are not flagged as violations and are ignored.	L, XL, GXL, and EXL	notify/post- edit/batch- check
Misc	errorLayer	L, XL, GXL, and EXL	notify/post- edit/batch- check
NumCut	minNumCut	L, XL, GXL, and EXL	post-edit/ batch-check
	minProtrusionNumCut	L, XL, GXL, and EXL	post-edit/ batch-check
Orientation	diagonalShapesAllowed	L, XL, GXL, and EXL	notify/post- edit/batch- check
	rectShapeDir	L, XL, GXL, and EXL	notify/post- edit/batch- check

Category	Constraint Name (ASCII Syntax)	Tier Level Support	Support Type
Spacing	allowedSpacingRanges (One layer)	L, XL, GXL, and EXL	notify/post- edit/batch- check
	minCenterToCenterSpacing	L, XL, GXL, and EXL	notify/post- edit/batch- check
	minCutClassSpacing (One layer)	L, XL, GXL, and EXL	enforce/ notify/post- edit/batch- check
	minCutClassSpacing (Two layers)	L, XL, GXL, and EXL	enforce/ notify/post- edit/batch- check
	minCutRoutingSpacing (Two layers)	L, XL, GXL, and EXL	enforce/ notify/post- edit/batch- check
	minDiagonalSpacing	L (scalar), XL (scalar), GXL (scalar), and EXL (scalar)	notify/post- edit/batch- check
	minEndOfLineToConcaveCornerSpacing	L, XL, GXL, and EXL	notify/post- edit/batch- check
	minEndOfNotchSpacing	L, XL, GXL, and EXL	notify/post- edit/batch- check

Category	Constraint Name (ASCII Syntax)	Tier Level Support	Support Type
	minFillToFillSpacing	L, XL, GXL, and EXL	post-edit/ batch-check
	minFillToShapeSpacing	L, XL, GXL, and EXL	post-edit/ batch-check
	minInfluenceSpacing	L, XL, GXL, and EXL	notify/post- edit/batch- check
	minNotchSpacing	L, XL, GXL, and EXL	notify/post- edit/batch- check
	minParallelViaSpacing (One layer)	L, XL, GXL, and EXL	notify/post- edit/batch- check
	minParallelViaSpacing (Two layers plus metal)	L, XL, GXL, and EXL	notify/post- edit/batch- check
	minParallelWithinViaSpacing	XL, GXL, and EXL	post-edit/ batch-check
	minPRBoundaryInteriorHalo	L, XL, GXL, and EXL	enforce/ notify/post- edit/batch- check
	minSameMetalSharedEdgeViaSpacing	L, XL, GXL, and EXL	notify/post- edit/batch- check
	minSameNetSpacing (One layer)	L, XL, GXL, and EXL	enforce/ notify/post- edit/batch- check
	<pre>minSameNetSpacing (Two layers) Note: minSameNetSpacing is derived from minSpacing in VLS XL, GXL, and EXL.</pre>	L, XL, GXL, and EXL	enforce/ notify/post- edit/batch- check

Category	Constraint Name (ASCII Syntax)	Tier Level Support	Support Type
	minSpacing (One layer)	L, XL, GXL, and EXL	enforce/ notify/post- edit/batch- check
	minSpacing (Two layers) (width and parallel run lengths) Note: Parallel run length (PRL) checking between an orthogonal and diagonal shape is not supported. Therefore, checker uses the maximum value from the spacing table for checking minSpacing.	L, XL, GXL, and EXL	enforce/ notify/post- edit/batch- check
	minSpacingOver	L, XL, GXL, and EXL	post-edit/ batch-check
	minViaSpacing (One layer)	L, XL, GXL, and EXL	enforce/ notify/post- edit/batch- check
	minViaSpacing (Two layers)	L, XL, GXL, and EXL	enforce/ notify/post- edit/batch- check
	minVoltageSpacing (One layer)	XL, GXL, and EXL	enforce/ notify/post- edit/batch- check
	minVoltageSpacing (Two layers)	XL, GXL, and EXL	enforce/ notify/post- edit/batch- check
	viaSpacing	L, XL, GXL, and EXL	notify/post- edit/batch- check

Category	Constraint Name (ASCII Syntax)	Tier Level Support	Support Type
Via	<u>definedCutClassesOnly</u>	L, XL, GXL, and EXL	notify/post- edit/batch- check
	largeRectViaArrayAllowed	L, XL, GXL, and EXL	post-edit/ batch-check
	minLargeNeighborViaArrayCutSpacing	L, XL, GXL, and EXL	post-edit
	minLargeViaArrayCutSpacing	L, XL, GXL, and EXL	post-edit/ batch-check
	minLargeViaArraySpacing	L, XL, GXL, and EXL	post-edit/ batch-check
	minLargeViaArrayWidth	L, XL, GXL, and EXL	post-edit/ batch-check
	minNeighborViaSpacing	L, XL, GXL, and EXL	post-edit/ batch-check
	minOrthogonalViaSpacing	L, XL, GXL, and EXL	post-edit/ batch-check
	redundantViaSetback	L, XL, GXL, and EXL	post-edit/ batch-check
	stackable	L, XL, GXL, and EXL	post-edit/ batch-check
	viaStackingLimits	L, XL, GXL, and EXL	post-edit/ batch-check

Category	Constraint Name (ASCII Syntax)	Tier Level Support	Support Type
Width	allowedWidthRanges	L, XL, GXL, and EXL	notify/post- edit/batch- check
	maxWidth	L, XL, GXL, and EXL	notify/post- edit/batch- check
	minCornerToCornerDistance	L, XL, GXL, and EXL	notify/post- edit/batch- check
	minDiagonalWidth	L, XL, GXL, and EXL	notify/post- edit/batch- check
	minWidth	L, XL, GXL, and EXL	notify/post- edit/batch- check
	protrusionWidth	L, XL, GXL, and EXL	post-edit/ batch-check

DRD Constraints Support

Supported Constraints and Parameters (ICADVM20.1 Only – 95511 and 95800)



Requires the Virtuoso Advanced Node Option for Layout (95511) and Virtuoso Layout Suite EXL (95800) licenses.

Category	Constraint Name (ASCII Syntax)	Tier Level Support	Support Type
Complex Spacing	<pre>endOfLineKeepout Parameters: horizontal, vertical, eolSpacingRange, otherEolSpacingRange, jointAsEol, checkOtherWidth, equalRectWidth</pre>	XL and EXL	notify/post- edit/batch- check
	<pre>minClusterSpacing (One layer) Parameters: enclosingLayer, viaEdgeType</pre>	XL and EXL	notify/post- edit/batch- check
	<u>overlapViaGroup</u>	XL and EXL	notify/post- edit/batch- check
	viaGroup Parameters: allowBend	XL and EXL	notify/post- edit/batch- check
Extension	<pre>minEndOfLineEdgeExtension Parameters: minVoltage, maxVoltage</pre>	XL and EXL	notify/post- edit/batch- check
	<pre>minOppExtension Parameters: widthHorizontal widthVertical</pre>	XL and EXL	notify/post- edit/batch- check
	minQuadrupleExtension Parameters: lineEndGapOnlyTable	XL and EXL	notify/post- edit/batch- check
Length	<pre>minStepEdgeLength Parameters: adjEolSpacingRange, jogEdge, adjJogEdge, adjNotchEdge</pre>	XL and EXL	notify/post- edit/batch- check

Category	Constraint Name (ASCII Syntax)	Tier Level Support	Support Type
Spacing	mergedViaCornerToCornerSpacing Parameters: mergeSpacing, lengthRange, otherLengthRange tripletCenterViaSpacingX, tripletCenterViaSpacingY, tripletCenterViaLarger	XL and EXL	notify/batch- check
	minParallelWithinViaSpacing Parameters: otherWidth	XL and EXL	notify/post- edit/batch- check
	minPRBoundaryInteriorHalo Parameters: exceptLayer, exceptWidth	XL and EXL	notify/post- edit/batch- check
	<pre>minSideSpacing (One layer) Parameters: otherLayer, cutClass, cutWithin</pre>	XL and EXL	notify/post- edit/batch- check
	minViaSpacing (One layer) Parameters: otherCutClass, otherViaEdgeType, deltaVoltage, exceptGap, exceptSameMetalOverlap	XL and EXL	notify/post- edit/batch- check
	minViaSpacing (Two layers) Parameters: deltaVoltage	XL and EXL	notify/post- edit/batch- check
	requiredEndOfLineShape	XL and EXL	notify/post- edit/batch- check
Width	minCornerToCornerDistance Parameters: extensionLength, oppositeDirection	XL and EXL	notify/post- edit/batch- check

DRD Constraints Support

Supported Constraints and Parameters (ICADVM20.1 Only – 95512)

/Important

Requires Virtuoso Advanced Node Option for Layout Standard (95512) license.

Category	Constraint Name (ASCII Syntax)	Tier Level Support	Support Type
Area	<pre>minArea Parameters: insideLayers, outsideLayers, insidePurposes, outsidePurposes</pre>	XL and EXL	notify/post- edit/batch- check
Complex Spacing	<u>forbiddenCutClassSpacingRange</u>	XL and EXL	notify/post- edit/batch- check
	<u>forbiddenEdgePitchRange</u>	XL and EXL	notify/post- edit/batch- check
	forbiddenProximitySpacing	XL and EXL	notify/post- edit/batch- check
	minClusterSpacing (One layer)	XL and EXL	notify/post- edit/batch- check
	minClusterSpacing (Two layers)	XL and EXL	post-edit/ batch-check
	minCornerSpacing (One layer)	XL and EXL	notify/post- edit/batch- check
	minCornerSpacing (Two layers)	XL and EXL	notify/post- edit/batch- check

Category	Constraint Name (ASCII Syntax)	Tier Level Support	Support Type
	minEdgeLengthSpacing	XL and EXL	notify/post- edit/batch- check
	minEndOfLineExtensionSpacing Parameters: sameMask, negativePRL	XL and EXL	post-edit/ batch-check
	minEndOfLineSpacing Parameters: exceptEolWidth, exceptExactAligned, sameMask, diffMask	XL and EXL	notify/post- edit/batch- check
	minSpanLengthSpacing	XL and EXL	enforce/ notify/post- edit/batch- check
	shapeRequiredBetweenSpacing	XL and EXL	notify/post- edit/batch- check
Edge Length	minEndOfLineAdjacentToStep Parameters: adjacentLength, concaveCorner	XL and EXL	post-edit/ batch-check
	minStepEdgeLength Parameters: any, horizontalEdge, verticalEdge, allCorner, concaveCorner, convexCorner, mixedCorner, exceptExactLength, exceptExactBothLength	XL and EXL	notify/post- edit/batch- check
Extension	minDirectionalOverlap	XL and EXL	notify/post- edit/batch- check
	minEndOfLineEdgeExtension Parameter: shortEdgeOnly	XL and EXL	post-edit/ batch-check

Category	Constraint Name (ASCII Syntax)	Tier Level Support	Support Type
	minExtensionDistance Parameters: insideLayers, outsideLayers, insidePurposes, outsidePurposes	XL and EXL	notify/post- edit/batch- check
	<pre>minExtensionEdge Parameters: insideLayers, outsideLayers, insidePurposes, outsidePurposes</pre>	XL and EXL	post-edit/ batch-check
	minExtensionOnLongSide	XL and EXL	notify/post- edit/batch- check
	minExtensionToCenterLine	XL and EXL	post-edit/ batch-check
	minExtensionToCorner	XL and EXL	post-edit/ batch-check
	minInnerVertexProximityExtension	XL and EXL	post-edit/ batch-check
	minOppExtension Parameters: insideLayers, outsideLayers, insidePurposes, outsidePurposes	XL and EXL (scalar and table)	notify/post- edit/batch- check
	<u>edgeMustCoincide</u>	XL and EXL	notify/post- edit/batch- check
	<u>edgeMustOverlap</u>	XL and EXL	notify/post- edit/batch- check
	minNeighborExtension	XL and EXL	notify/post- edit/batch- check

Category	Constraint Name (ASCII Syntax)	Tier Level Support	Support Type
	<pre>minQuadrupleExtension Parameters: insideLayers, outsideLayers, insidePurposes, outsidePurposes</pre>	XL and EXL	notify/post- edit/batch- check
	minSideExtension Parameter: exceptEdgeLengthRanges	XL and EXL	notify/post- edit/batch- check
	minWireOverlap	XL and EXL	notify/post- edit/batch- check
Length	allowedLengthRanges	XL and EXL	notify/post- edit/batch- check
Misc	maxNumCorners	XL and EXL	notify/post- edit/batch- check
Orientation	<pre>rectShapeDir Parameters: exceptEdgeLength, exceptExactSize, exceptRanges, exceptViaLayer, exceptViaSize, insideLayers, insidePurposes, outsideLayers, outsidePurposes, twoSides, width</pre>	XL and EXL	notify/post- edit/batch- check

Category	Constraint Name (ASCII Syntax)	Tier Level Support	Support Type
Spacing	allowedSpacingRanges (One layer) Parameters: widthRanges, exceptOverLayer, overLayerPrl, overLayerWidthRanges, exceptNumShapes, numShapeDistance, numShapeWidthRanges, numShapeMaxWidth, stepSize, stepRange, overLayer	XL and EXL	notify/post- edit/batch- check
	minCenterLineSpacing (Two layers)	XL and EXL	notify/post- edit/batch- check
	minCutClassSpacing (One layer) Parameters: insideLayers, outsideLayers, insidePurposes, outsidePurposes	XL and EXL	enforce/ notify/post- edit/batch- check
	minCutClassSpacing (Two layers) Parameter: cutClassSizeBy Note: cutClassSizeBy specifies extension (sizing) applied on cut edges before spacing is measured.	XL and EXL	enforce/ notify/post- edit/batch- check
	minNeighboringShapesSpacing	XL and EXL	notify/post- edit/batch- check
	minEndOfLineToNotchSpacing	XL and EXL	notify/post- edit/batch- check
	minNotchSpacing Parameter: notchWidth	XL and EXL	notify/post- edit/batch- check

Category	Constraint Name (ASCII Syntax)	Tier Level Support	Support Type
	minParallelWithinViaSpacing Parameters: cutClass, longEdgeOnly, insideLayers, outsideLayers, insidePurposes, outsidePurposes	XL and EXL	post-edit/ batch-check
	minSideSpacing (One layer)	XL and EXL	enforce/ notify/post- edit/batch- check
	minSideSpacing (Two layers)	XL and EXL	notify/post- edit/batch- check
	<pre>minSpacing (One layer) Parameter: exceptEolWidth, insideLayers, outsideLayers, insidePurposes, outsidePurposes</pre>	XL and EXL	enforce/ notify/post- edit/batch- check
	minStitchOverlap	XL and EXL	notify/post- edit/batch- check
	<pre>minViaSpacing (One layer) Parameters: exceptExactAligned, overLayer, overLayerWidth, exceptSameNet, exceptSameMetal</pre>	XL and EXL	enforce/ notify/post- edit/batch- check
	<pre>viaSpacing Parameters: allCuts, cutSizeRanges, insideLayers, outsideLayers, insidePurposes, outsidePurposes</pre>	XL and EXL	notify/post- edit/batch- check

Category	Constraint Name (ASCII Syntax)	Tier Level Support	Support Type
Via	allowedCutClass	XL and EXL	notify/post- edit/batch- check
	minLargeViaArrayCutSpacing Parameter: maxNumCuts	XL and EXL	post-edit/ batch-check
Width	allowedNeighborWidthRanges (One layer)	XL and EXL	notify/post- edit/batch- check
	allowedNeighborWidthRanges (Two layers)	XL and EXL	notify/post- edit/batch- check
	allowedNeighborWidthRangesOver	XL and EXL	notify/post- edit/batch- check
	allowedPRBoundaryDimensions	XL and EXL	notify/post- edit/batch- check
	allowedWidthRanges Parameters: insideLayers, outsideLayers, insidePurposes, outsidePurposes	XL and EXL	notify/post- edit/batch- check

DRD Constraints Support

Supported Constraints and Parameters (ICADVM20.1 Only – 95511)

/Important

Requires Virtuoso Advanced Node Option for Layout (95511) license.

Category	Constraint Name (ASCII Syntax)	Tier Level Support	Support Type
Area	minRectArea	XL and EXL	notify/post-
	Parameters: colorMask, layer, overlapType		edit/batch- check
Complex	<u>endOfLineKeepout</u>	XL and EXL	notify/post-
Spacing	<pre>Parameters: mask1, mask2, mask3, diffMask, twoSides, forwardGap</pre>		edit/batch- check
	forbiddenCutClassSpacingRange	XL and EXL	notify/post-
	Parameters: cutClassPrl, allCuts, layer, layerWidth, mask1, mask2, mask3, paraLength, within, sameMask, longEdge		edit/batch- check
	maxViaArrayClusterSize	XL and EXL	notify/post- edit/batch- check
	minAdjacentFourViaSpacing	XL and EXL	notify/post- edit/batch- check
	minClusterSpacing (One layer)	XL and EXL	post-edit/
	Parameter: widthDirection (widthHorizontal widthVertical)		batch-check
	minCornerSpacing (One layer)	XL and EXL	post-edit/
	Parameters: sameMask, treatLAsJog		batch-check
	minEndOfLinePerpSpacing	XL and EXL	post-edit/
	Parameters: negativePRL	batch	batch-check

Category	Constraint Name (ASCII Syntax)	Tier Level Support	Support Type
	minEndOfLineSpacing Parameter: exceptExactEolWidth, fillConcaveCorner	XL and EXL	notify/post- edit/batch- check
	minSpanLengthSpacing Parameters: minSpanForExceptEolWidth, minSpanSpacingRanges, sameMask, insideLayers, outsideLayers, insidePurposes, outsidePurposes, eolExceptionNoPrl, diffMask	XL and EXL	enforce/ notify/post- edit/batch- check
	minViaGroupSpacing	XL and EXL	notify/post- edit/batch- check
	orthogonalSnappingLayer	XL and EXL	post-edit
	viaGroup	XL and EXL	notify/post- edit/batch- check
	<u>viaKeepoutZone</u>	XL and EXL	post-edit/ batch-checl

Category	Constraint Name (ASCII Syntax)	Tier Level Support	Support Type
Extension	minExtensionEdge	XL and EXL	post-edit/
	Parameters: edgeExtension, layer		batch-check
	minNeighborExtension	XL and EXL	notify/post-
	Parameters: mask1, mask2, mask3, mask4		edit/batch- check
	minOppExtension	XL and EXL	notify/post-
	Parameters: hollowHorizontal, hollowVertical	(scalar and table)	edit/batch- check
		XL and EXL	notify/post-
	Parameter: maxLength, trimLayer		edit/batch- check
	minVoltageExtension	XL and EXL	notify/post- edit/batch- check
Length	minEdgeLength	XL and EXL	notify/batch- check
	minStepEdgeLength	XL and EXL	notify/batch-
	Parameters: width		check

Category	Constraint Name (ASCII Syntax)	Tier Level Support	Support Type
Miscellane ous	preColoredLayers	XL and EXL	notify/post- edit/batch- check
Spacing	allowedCutClass Parameters: minVoltage, maxVoltage	XL and EXL	notify/batch- check
	allowedSpacingRanges (One layer) Parameters: exceptWidthEdge	XL and EXL	notify/batch- check
	mergedViaCornerToCornerSpacing	XL and EXL	notify/batch- check
	minCornerVoltageSpacing	XL and EXL	notify/post- edit/batch- check
	minCutEdgeSpacing (Two layers)	XL and EXL	notify/post- edit/batch- check
	minNestedViaSpacing	XL and EXL	notify/post- edit/batch- check
	<pre>minNotchSpacing Parameters: excludeSpacing, horizontal vertical</pre>	XL and EXL	notify/post- edit/batch- check
	minSideSpacing (One layer) Parameters: facingEdges, sameMask, cornerEuclidian, prlRange, lengthRanges, otherLengthRanges, insideLayers, outsideLayers, insidePurposes, outsidePurposes, widthDirection (widthHorizontal widthVertical), facingEdgeDirection, otherFacingEdgeRange	XL and EXL	enforce/ notify/post- edit/batch- check

Category	Constraint Name (ASCII Syntax)	Tier Level Support	Support Type
	minSideSpacing (Two layers) Parameters: cornerEuclidian, insideLayers, outsideLayers, insidePurposes, outsidePurposes, otherColorMask ('otherMask1 'otherMask2 'otherMask3), exceptOverlap, deltaVoltage, prlRange	XL and EXL	notify/post- edit/batch- check
	<pre>minSpacingOver Parameter: colorMask ('mask1 'mask2 'mask3)</pre>	XL and EXL	post-edit/ batch-check
	minStitchSize	XL and EXL	notify/post- edit/batch- check
	minViaSpacing (One layer) Parameters: diffMask, above, below, bothAboveBelow, shortSideToShortSide shortSideToLongSide shortSideToAnySide longSideToLongSide longSideToAnySide anySideToAnySide, prlRange, viasOnSameNet viasOnSameMetal viasOnSameVia, exceptViasOnSameMetal exceptViasOnSameWetal exceptViasOnSameWetal exceptViasOnSameVia	XL and EXL	enforce/ notify/post- edit/batch- check
	minViaSpacing (Two layers) Parameters: exceptSameNet, exceptSameMetal	XL and EXL	enforce/ notify/post- edit/batch- check
	multiMaskCheck	XL and EXL	notify/post- edit/batch- check

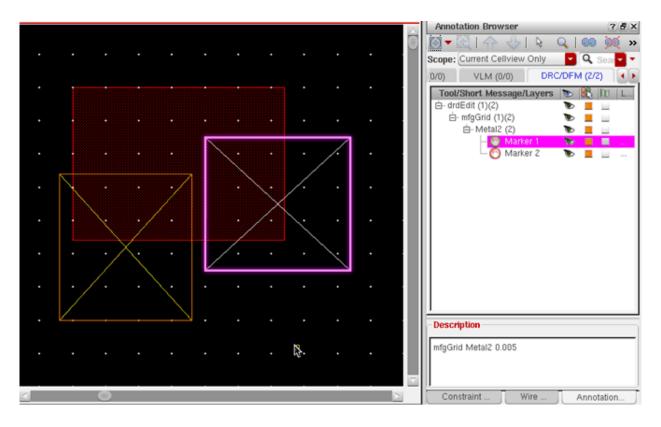
Category	Constraint Name (ASCII Syntax)	Tier Level Support	Support Type
	orthogonalWSPGrid	XL and EXL	notify/post- edit/batch- check
	trimMinSpacing (One layer)	XL and EXL	notify/post- edit/batch- check
	trimShape	XL and EXL	notify/post- edit/batch- check
	<u>vertexInsideForbidden</u>	XL and EXL	notify/post- edit/batch- check
	<pre>viaSpacing Parameters: sameMask, exceptWithin, edgeExtension, layer, extensionDirection</pre>	XL and EXL	notify/post- edit/batch- check

Using DRD Grid Checking

You can use the DRD grid checking capability to verify if any shapes in the layout are off the manufacturing grid specified in the mfgGridResolution section of the technology file. The DRD grid checking capability is available in Post-Edit and Batch-Check modes.

To enable DRD grid checking, select the *Grid* check box in Post-Edit mode of the DRD Options form. For batch checking, select the *Grid* check box in the Batch-Check mode of the DRD Options form or the *Grid* check box in the *Check For* group box in the Batch Checker form.

After running the grid check, violations found in the design are listed in the Annotation Browser and the corresponding violation markers are created in the layout canvas. In the figure below, you can see that the right vertical edge and the bottom horizontal edge of the *Metal2* rectangle are off the grid; marker details are displayed in the Annotation Browser.



Specifying Layers in the Interconnect Section

In VLS XL, VLS GXL (IC6.1.8 Only), and VLS EXL (ICADVM20.1 Only), layers that are not specified as validLayers in the interconnect section of a constraint group receive only

DRD Constraints Support

basic checks (checks available in Virtuoso Layout Suite L); none of the following advanced constraints are applied:

maxNumMinEdges
minCenterToCenterSpacing
minEdgeAdjacentLength
minEndOfLineSpacing
minNumCut
minOppExtension (table)
minParallelViaSpacing
minProtrusionNumCut
minRectArea
minStepEdgeLength
minstubInfluenceSpacing
redundantViaSetback
viaSpacing
viaStackingLimits

Supported Derived Layers

Derived layer support in DRD helps check for DRC violations on layers derived from Boolean operations and on layers derived by sizing other layers.

Supported derived layer operators include:

- 'and
- 'or
- 'not
- 'xor
- 'select
- 'avoiding

DRD Constraints Support

- 'butting
- 'touching
- 'straddling
- 'inside
- 'outside
- 'color (ICADVM20.1 Only)

DRD Constraints Support

Sized layers can be created using the following operators:

- 'grow
- 'growVertical
- 'growHorizontal
- 'shrink
- 'shrinkVertical
- 'shrinkHorizontal

Derived layers with restricted area are created using the 'area operator.

The derived layer definitions are added to the techDerivedLayers section of the layerDefinitions section of the technology file. In general, layer purposes are not supported in derived layer definitions, except with the 'select operator. Derived layer numbers start from 10000 to prevent conflict with the layer numbers assigned to "regular" layers.

For more information about these operators, see <u>Virtuoso Technology Data ASCII Files</u> Reference.

Supported Layer-Purpose Pair and Voltage-Dependent Constraints

Support for Layer-Purpose Pair Constraints

A few constraints can also be applied to *user-defined* layer-purpose pairs that are defined in the techPurposes section of the technology file. When applied to user-defined layer-purpose pairs, these constraints are checked only if they are specified in the foundry constraint group.

Table 5-1 Supported Layer-Purpose Pair Constraints

Constraint	ASCII Syntax	Tier Level Support	Support Type
Minimum spacing	minSpacing (One layer)	L, XL, GXL, and EXL	enforce/notify/post- edit
Minimum clearance	minSpacing (Two layers)	L, XL, GXL, and EXL	enforce/notify/post- edit
Minimum width	minWidth	L, XL, GXL, and EXL	notify/post-edit

To determine minimum spacing, clearance, or width for an object, the rules are applied in the following order of precedence:

- The object's layer-purpose pair rule
- The object's parent layer-purpose pair rule
- The object's layer rule

Example

This example sets the following conditions:

- The minimum width for ("Metall" "HV") shapes is 0.14, and for all other Metall shapes, it is 0.12.
- The minimum spacing for ("Metall" "LV") shapes is 0.4.
- The minimum spacing for ("Metall" "logic") shapes is 0.5.

DRD Constraints Support

- The minimum spacing for ("Metall" "HV") shapes is 1.0.
- The minimum clearance between a ("Metall" "HV") shape and a ("Metall" "LV") shape is 0.5.
- The minimum clearance between a ("Metall" "HV") shape and a ("Metall" "logic") shape is 0.7.
- The minimum clearance between a ("Metall" "HV") shape and a V1 shape is 0.8.

The user-defined logic, HV, and LV purposes are specified in the techPurposes section of the technology file.

In the foundry constraint group, the minimum width, spacing, and clearance constraints are specified as follows:

```
spacings (
               "Metal1"
                                                  0.12)
  ( minWidth
  ( minWidth
               ("Metal1" "HV")
                                                  0.14)
  ( minSpacing ("Metal1" "LV")
                                                  0.4
  ( minSpacing ("Metall" "logic")
                                                  0.5
  ( minSpacing ("Metall" "HV")
                                                  1.0
  (minSpacing ("Metall" "HV") ("Metall" "LV")
                                                  0.5
  (minSpacing ("Metall" "HV") ("Metall" "logic") 0.7
) ; spacings
spacings (
  (minSpacing ("Metall" "HV") "V1"
                                                  0.8)
) ; spacings
```

Support for Voltage-Dependent Constraints

In process nodes of 45nm and below, voltage-specific spacing rules are defined by the foundry. These constraints, listed in <u>Table 5-2</u> on page 162, are supported in VLS XL, VLS GXL (IC6.1.8 Only), and VLS EXL (ICADVM20.1 Only), and can be layer-purpose-pair-based or net-based as described in the following sections:

- Layer-Purpose Pair-based VDR Support
- Net-based VDR Support

DRD Constraints Support

Table 5-2 Supported Voltage Dependent Rule Constraints

Constraint	ASCII Syntax	Tier Level Support	Support Type
Minimum voltage spacing	minVoltageSpacing (one-layer)	XL, GXL, and EXL	enforce/notify/post-edit
Minimum voltage clearance	minVoltageSpacing (two-layer)	XL, GXL, and EXL	enforce/notify/post-edit

Layer-Purpose Pair-based VDR Support

Voltage-dependent rules can be applied to *user-defined* layer-purpose pairs that are defined in the techPurposes section of the technology file. The voltage-dependent minimum spacing (minVoltageSpacing (One layer)) and clearance (minVoltageSpacing (Two layers)) constraints are set by layer. A voltage swing must be specified for each layer-purpose pair on which the constraint is defined. When applied to user-defined layer-purpose pairs, these constraints are checked only if they are specified in the foundry constraint group.

Example

This example sets the following conditions:

- ("Metal1" "HV") shapes have a voltage swing between 0.0 and 3.3.
- ("Metal1" "LV") shapes have a voltage swing between 0.0 and 1.5.
- The minimum spacing between Metall shapes whose maximum voltage swing is less than 1.5 is 0.06.
- The minimum spacing between Metall shapes whose maximum voltage swing is greater than or equal to 1.5 and less than 3.3 is 0.08.
- The minimum spacing between Metall shapes whose maximum voltage swing is greater than or equal to 3.3 is 0.1.

DRD Constraints Support

The user-defined HV and LV purposes are specified in the techPurposes section of the technology file with their associated voltage ranges.

```
techPurposes(
;(PurposeName Purpose# Abbreviation)
;(-----)
;User-defined Purposes:
(HV 13 HV 'sigType "digital" 'parent "drawing" voltageRange (0.0 3.3))
(LV 14 LV 'sigType "digital" 'parent "drawing" voltageRange (0.0 1.5))
)
```

In the foundry constraint group, the voltage-dependent minimum spacing between two shapes on the Metall layer (minVoltageSpacing) is described as:

```
spacingTables(
  ( minVoltageSpacing "Metall"
    ( "voltage" nil nil )
    ( 0.0 0.06
      1.5 0.08
      3.3 0.10
    )
  ); spacingTables
```

To determine the minimum spacing from the spacing table, the voltage swing between shapes is calculated as follows:

```
V_{swing} = max(maximum \ voltage \ of \ each \ shape) - min(minimum \ voltage \ of \ each \ shape)
```

This voltage swing is compared with the "voltage" index in the spacing table to determine the first value that is less than the calculated voltage swing. The corresponding minimum spacing applies. In this example:

- For a voltage swing \geq 0.0 and < 1.5, the minimum spacing is 0.06.
- For a voltage swing >= 1.5 and < 3.3, the minimum spacing is 0.08.
- For a voltage swing >= 3.3, the minimum spacing is 0.1.

Net-based VDR Support

The minimum and maximum voltages on nets can be set from the *Property Editor* assistant. For more information about the prerequisites for VDR flows and an explanation of how DRD checks voltage-dependent spacing violations, see <u>Virtuoso Voltage Dependent Rules Flow Guide</u>.

DRD Constraints Support

Using Density Constraints

Density is calculated as a percentage of the design area that metal on a layer occupies. It is calculated as follows:

Density = Metal area / Total area

The density check performed by DRD in Batch-Check mode relies on the minDensity and maxDensity constraints defined in the technology file, similar to the examples given below for Metall:

```
( minDensity "Metall"
  (( "step" nil nil ))
  (
    60.0 20.0
  )'ref "METAL1.D.1" 'description "Minimum Metall Density"
)
( maxDensity "Metall"
  (( "step" nil nil ))
  (
    60.0 65.0
  ) 'ref "METAL1.D.1" 'description "Maximum Metall Density"
)
```

The second value (20.0) in the minDensity constraint listed above is the minimum density required for Metall. The second value (65.0) in the maxDensity constraint represents the maximum density allowed for Metall.

When you run DRD with the Density check option, the values for minDensity and maxDensity constraints are listed in the *Min Density* and *Max Density* columns, respectively, in the report displayed in CIW. Density is calculated for each specified window and step size and these results too are displayed in CIW, as shown below:

000	Window	Window	Min	Max	Violatio	n Count	Density	Percentage	Statist	ics
Layer	Size	Step	Density	Density	Min	Max	Lowest High	est Median	Ave	Std Dev
========	=======	=======	======	======	======	======	=======================================		======	======
Metal1	120.00	60.00	20.00%	65.00%	0	4	100.00% 100.)0% 100.00%	100.00%	0.00%

Note: To run density checks of only one type, minimum or maximum, set the <u>drdEditDensityTypeCheck</u> environment variable to min or max, respectively.

DRD Constraints Support

The *Density Percentage Statistics* figures provide the lowest and the highest area occupied on a metal layer as a percentage of the design. To better understand the significance of the various statistical terms, consider the following example:

	Window	Window	Min	Max	Violatio	n Count
Layer	Size	Step	Density	Density	Min	Max
metal1	8.00	4.00	0.00%	100.00%	0	0
	Window	Window	Min	Max	Violatio	n Count
Layer	Size	Step	Density	Density	Min	Max
metal2	8.00	4.00	0.00%	100.00%	0	0
Verify Proc	ess Rules	Summary: 0	Marker(s) created.	,	

Der	nsity Per	Statistics		
Lowest	Highest	Median	Ave	Std Dev
	======		======	======
50.00%	75.00%	75.00%	62.50%	5.00%
Der	nsity Per	Statistics		
Lowest	Highest	Median	Ave	Std Dev
0.00%	0.00%	0.00%	0.00%	0.00%

Average (Ave)

In the example above, the lowest density value for metal1 is 50%, and the highest density value for metal1 is 75%. Therefore, the average density value is 62.5%.

Average (Ave) =
$$(Lowest + Highest) / 2 = 62.5$$

Median

The median value is obtained after sorting all density values in the ascending order. If the number of density check windows is odd, the median is equal to the value that lies in the 'middle' of the sorted list, and if the number of density check windows is even, the median is calculated as an average of the two 'middle' values.

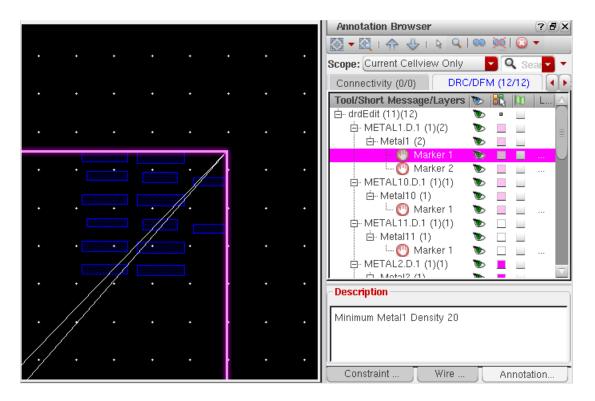
Standard Deviation (Std Dev)

The standard deviation measures how close the density value for each density check window is to the mean value. A small standard deviation value of 5% indicates that the values are concentrated close to the mean value, which in this case is 62.5%.

DRD Constraints Support

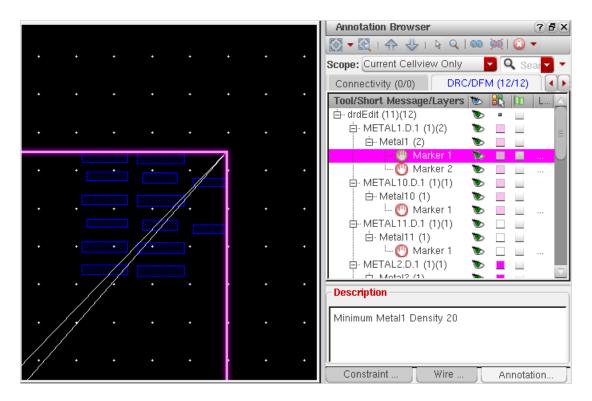
Viewing Density Violations in Annotation Browser

When a minDensity constraint is violated, DRD batch checker (Verify - Design) reports it as follows:



DRD Constraints Support

When a maxDensity constraint is violated, DRD batch checker (*Verify – Design*) reports it as follows:



Using Alternate Foundry Constraint Group

If an alternate foundry constraint group (AFCG) is defined in the .cdsenv file, DRD replaces the foundry constraint group and uses the specified AFCG instead.

You can specify an AFCG by setting the following environment variable in your . cdsenv file:

cdba.layout AlternateFoundryCG string "AFCG_name"

A

DRD Form Descriptions

This section lists the Virtuoso® Layout Suite Design Rule Driven editing options forms.

Note: Many of the options described in this section have a corresponding environment variable noted next to the description. For more information, see <u>DRD Environment Variables</u>.

DRD Compactor Options on page 170

DRD Options on page 172

DRD Form Descriptions

DRD Compactor Options

Use the **DRD Compactor Options** form to specify how objects in the canvas are compacted and spread by DRD Compactor, the spacing on which changes are based, and the reference objects used to anchor the compaction or spreading. For more information about this feature, see <u>DRD Interactive Compactor</u>.

Compaction Mode lets you specify how compaction and spreading is performed in relation to the specified spacing options.

Environment variable: drdEditCompactorMode

Compact and Spread compacts objects that were originally spaced more than the spacing value apart and spreads objects that were originally spaced less than the new spacing value apart.

Compact only compacts objects that were originally spaced more than the new spacing value apart, but does not spread any objects.

Spread only spreads objects that were originally spaced less than the new spacing value apart, but does not perform any compaction.

Spacing Options let you choose the spacing on which compaction and spreading is based. Objects spaced more than the specified value apart are compacted; those spaced less than the specified value apart are spread.

Environment variable: drdEditCompactorSpacingMode

DRD Enforce spacing compacts and spreads objects based on the spacing constraints supported by DRD Enforce.

User spacing compacts and spreads based on the value you type in to the field. Environment variable: drdEditCompactorUserSpacing

Apply to Active Layer in Palette applies *User spacing* to only the shapes on the layer currently defined as active in the Palette.

Environment variable: drdEditCompactorActiveLayerOnly

Anchor Mode specifies target objects in the layout against which other objects are compacted or spread.

Environment variable: drdEditCompactorAnchorMode

Use last shape in direction of compaction compacts or spreads selected objects by using as the target the last object in the direction of compaction. For example, if you are compacting left, the leftmost object remains static and the remaining objects are compacted (or spread) in relation to that object.

DRD Form Descriptions

Use first unselected shape in direction of compaction compacts or spreads selected objects by using as the target the first unselected object found in the direction of compaction. If there is no unselected object in the direction of compaction within the visible area of the layout canvas, the selected objects are compacted against the edge of the canvas.

Use DRD Targets compacts or spreads selected objects against a DRD Target object, which can be inside or outside the visible area of the layout canvas.

Maintain relative positions of selected objects maintains the original spacing between selected objects during compaction. For example, if you select three objects for compaction, the object closest to the anchor is compacted against the anchor and the original spacing between the three objects remains unchanged. Environment variable: drdEditCompactorInterOnly

DRD Form Descriptions

DRD Options

The form comprises the following tabs:

- Interactive Tab
- Batch Tab
- Filters Tab

Interactive Tab

Hierarchy Range sets how many levels of the hierarchy are to be considered during DRD editing.

current cellview the current cellview is considered. This option is selected by default in the drop-down list.

current to bottom the current level of hierarchy to the level specified in the *Open to Stop Level* (Display Levels Stop) field in the Display Options form is considered.

current to stop level the current level in the hierarchy to the level specified in the *Display Levels Stop* field in the Display Options form is considered.

current to user level the current level in the hierarchy to the level that you specify. You can specify a value between 0 and 32.

Environment variables: <u>drdEditBatchHierDepth</u>, <u>drdEditHierDepth</u>

Notify provides visual feedback in the form of halos and arrows when design rule violations occur.

Enabled enables the Notify editing mode.

Zoom in Incremental Violation enables a zoomed-in view of the objects involved in violations during incremental violation display.

Smart Snap enables DRD Smart Snap mode, which lets you snap objects to the edges or corners of DRC halos generated during DRD editing. This option is available only when the *Notify – Enabled* option is enabled. For more information, see <u>DRD Smart Snapping</u>.

Enforce stops pointer movement momentarily when a violation threshold is reached. The halo disappears as soon as the threshold is crossed.

Enabled enables the Enforce editing mode.

DRD Form Descriptions

Relaxed enables the Relaxed Enforce mode. For more information, see <u>Relaxed</u> Enforce.

Targets Only enables the Targeted Enforce mode. For more information, see <u>Targeted</u> Enforce.

Post Edit generates markers if a layout editing operation results in design rule violations.

Enabled enables the Post Edit editing mode.

Deferred enables the Deferred Post Edit mode. For more information, see <u>Deferred Post Edit</u>.

Marker Limit Specifies the maximum number of post-edit markers to be created per call. The default is 5000.

Environment variable: drdPostEditVioLimit

(ICADVM20.1 Only – 95512) MPT Checks help check the design for conformance to the rules defined in the technology file.

On reports violations for all selected rules, including color-related violations.

Off reports all violations except those related to color.

Only reports only color-related violations.

Conservative reports same-mask spacing violations.

Environment variable: drdEditColorRules

Colorability reports double or triple patterning violations for shapes on a layer.

Environment variable: <u>drdColorability</u>

Color Shorts reports overlaps between shapes with different colors, irrespective of their lock status.

Environment variable: drdColorShorts

Unlocked Shapes reports color-unlocked shapes on a layer.

Environment variable: drdUnlocked

Uncolored Shapes reports uncolored (gray) shapes on layers that support multiple masks.

Environment variable: drdUncoloredShapes

DRD Form Descriptions

Consider Gray Shapes considers gray shapes as colored and does not report gray shapes as violations while performing color checks on a design.

Environment variable: drdConsiderGrayShapes

Consider Unlocked Shapes As Gray considers unlocked shapes as gray while performing color checks on a design.

Environment variable: <u>drdConsiderUnlockedShapesAsGrav</u>

Display helps specify the display elements that should appear when a constraint is violated.

Arrows displays arrows to indicate the optimum spacing between objects.

Edges highlights the edges where the violation occurs.

Environment variable: drdEditDisplayEdges

Rule Text displays a description of the violated constraint in text format.

Environment variable: drdEditDisplayText

Ref Text displays the reference text for the violated constraint from the technology file, followed by the constraint value. When deselected, only the name of the violated constraint is displayed.

Environment variable: drdEditDisplayTechRefs

Halo creates halos around stationary objects when creating or editing an object that violates a spacing constraint.

Environment variable: <u>drdEditDisplavHalo</u>

True Color displays the halo in true color, that is, the color assigned to a layer.

Environment variable: drdEditDisplayTrueColor

Auto Halo Turn Off suppresses halos if the number of violations exceeds the value specified by the user. Only arrows are displayed on the canvas when the user-defined limit is reached.

Environment variables: <u>drdEditAutoTurnOffHalo</u>, <u>drdEditAutoTurnOffHaloLimit</u>

Batch Tab

Hierarchy Range sets how many levels of the hierarchy are to be considered during DRD editing in batch mode.

DRD Form Descriptions

current cellview the current cellview is considered. This option is selected by default in the drop-down list.

current to bottom the current level of hierarchy to the level specified in the *Open to Stop Level* (Display Levels Stop) field in the Display Options form is considered.

current to stop level the current level in the hierarchy to the level specified in the *Display Levels Stop* field in the Display Options form is considered.

current to user level the current level in the hierarchy to the level that you specify.

Environment variable: drdEditBatchHierDepth

Scope helps define the scope of checking in DRD batch mode.

Area specifies the area of check in DRD batch mode.

Environment variable: drdBatchScopeAreaLimit

Current Editable Cellview checks all the shapes in the current editable cellview.

Current View Area checks all the shapes in the current viewable area.

Current Specified Area checks all the shapes in the specified area of the design.

Define Area lets you specify the region in which you want to run DRD in batch mode. Specify the lower-left and upper-right coordinates for the region in this format ((xlower ylower) (xupper yupper)). This option is available only when the *Current Specified Area* option is selected.

Environment variable: <u>drdBatchScopeSpecifiedArea</u>

Marker Limit Specifies the maximum number of post-edit markers to be created per call in batch mode. The default is 5000.

Environment variable: drdBatchVioLimit

(ICADVM20.1 Only – 95512) MPT Checks help check the design for conformance to the rules defined in the technology file.

On reports violations for all selected rules, including color-related violations.

Off reports all violations except those related to color.

Only reports only color-related violations.

Conservative reports same-mask spacing violations.

Environment variable: drdEditBatchColorRules

DRD Form Descriptions

Colorability reports double or triple patterning violations for shapes on a layer.

Environment variable: drdBatchColorability

Color Shorts reports overlaps between shapes with different colors, irrespective of their lock status.

Environment variable: drdBatchColorShorts

Unlocked Shapes reports color-unlocked shapes on a layer.

Environment variable: drdBatchUnlocked

Uncolored Shapes reports uncolored (gray) shapes on layers that support multiple masks.

Environment variable: drdBatchUncoloredShapes

Consider Gray Shapes considers gray shapes as colored and does not report gray shapes as violations while performing color checks on a design.

Environment variable: drdBatchConsiderGrayShapes

Consider Unlocked Shapes As Gray considers unlocked shapes as gray while performing color checks on a design.

Environment variable: drdBatchConsiderUnlockedShapesAsGray

Filters Tab

Search Constraints lets you specify strings to filter the constraints for which you want DRD to display violations on the canvas.

Constraint Category lists the constraint categories.

Point to a constraint category to view a list of all the constraints that DRD supports in that category along with the supported modes for the constraints.

Constraint lists the constraints filtered either by using a string or selecting one or more constraint categories.

E (Enforce) indicates whether the filtered constraints are supported in Enforce mode.

N (Notify) indicates whether the filtered constraints are supported in Notify mode.

PE (Post Edit) indicates whether the filtered constraints are supported in Post Edit mode.

B (Batch) indicates whether the filtered constraints are supported in Batch mode.

DRD Form Descriptions

Ignore Marked Instance excludes the instances from DRD checking that are marked as ignored through the *DRD Add Ignore* option in the *Navigator* assistant.

Layer contains the filtering options for layers for which you want DRD to display violations on the canvas.

Palette monitors only those layers for violations that are defined in the *Layers* panel of the *Palette* assistant.

Environment variable: drdEditPaletteLayers

Search Layers lets you specify strings to filter the layers.

Layer Category lists the three layer categories, *Routing*, *Device*, and *Derived*.

Layer lists the layers filtered either by using a string or selecting one or more layer categories.

S (Selectability) determines whether the shapes and objects on a selected layer are to be checked by DRD.

The following fields are common to all the tabs:

Configuration file lets you specify the configuration file. The browse button next to it lets you locate and select the existing configuration file and specify the location at which to save the new configuration file.

Save To lets you save the current settings from the DRD Options form to the specified configuration file.

Load From lets you load settings from the specified configuration file into the DRD Options form.

Virtuoso Design Rule Driven Editing User Guide DRD Form Descriptions

В

DRD Environment Variables

This appendix contains the environment variable names, types, and values supported by DRD.

Note: Only the environment variables documented in this chapter are supported for public use. All other DRD environment variables, regardless of their name or prefix, and undocumented aspects of the environment variables described below, are private and are subject to change at any time.

List of Environment Variables

- drdBatchColorability
- drdBatchColorShorts
- drdBatchConsiderGrayShapes
- drdBatchConsiderUnlockedShapesAsGray
- drdBatchFoundryRulesOnly
- drdBatchReportCoveredNonPcellGrayShapes
- drdBatchScopeAreaLimit
- drdBatchScopeSpecifiedArea
- drdBatchUncoloredShapes
- drdBatchUnlocked
- drdBatchVioLimit
- drdColorability
- drdColorShorts
- drdConsiderGrayShapes

DRD Environment Variables

- drdConsiderUnlockedShapesAsGray
- drdEditAllowOrthogonalAdjust
- drdEditAutoText
- drdEditAutoTurnOffHalo
- drdEditAutoTurnOffHaloLimit
- drdEditBatchColorRules
- drdEditBatchHierDepth
- <u>drdEditColorRules</u>
- drdEditCompactorActiveLayerOnly
- drdEditCompactorAnchorMode
- drdEditCompactorInterOnly
- drdEditCompactorMode
- <u>drdEditCompactorSpacingMode</u>
- drdEditCompactorUserSpacing
- drdEditDensityTypeCheck
- drdEditDensityTypeCheck
- drdEditDisplayArrows
- drdEditDisplayDashed
- drdEditDisplayDrawingColor
- drdEditDisplayEdges
- drdEditDisplayHalo
- drdEditDisplayMarkers
- drdEditDisplayMaxChars
- drdEditDisplayTechRefs
- drdEditDisplayText
- <u>drdEditDisplayTrueColor</u>
- drdEditEnableBatchCheckOnReadOnlyDesign

DRD Environment Variables

- drdEditFontSize
- drdEditFontSize
- drdEditHierDepth
- <u>drdEditIgnoreIntraInstanceChecks</u>
- drdEditMode
- drdEditMode
- drdEditPaletteLayers
- <u>drdEditSlidingWindowSize</u>
- drdEditSlidingWindowSize
- drdEditSmartSnapAllowedWidthSnap
- drdEditSmartSnapAperture
- drdEditVioLimit
- <u>drdEditVioLimit</u>
- drdInteractiveFoundryRulesOnly
- drdPostEditAllWindows
- drdPostEditEIPAllInstances
- drdPostEditEIPAllWindows
- drdPostEditEIPCheckTopView
- drdPostEditIgnoreModgen
- drdPostEditReplaceAllMarkers
- drdPostEditTimeOut
- <u>drdPostEditVioLimit</u>
- drdUncoloredShapes
- drdUnlocked
- drdUseBlockageForExtCheck
- drdUseNetName
- drdUseNewDerivedLayerEngine

DRD Environment Variables

- drdValidLayersGrayCheck
- drdVDRIncludePurposes
- maxShapesAllowedForCompaction

DRD Environment Variables

drdBatchColorability

```
layout drdBatchColorability boolean { t | nil }
```

Description

(ICADVM20.1 Only) Enables you to specify whether or not to perform colorability check on the design in Batch mode. If set to true (t), DRD checks if a set of shapes is colorable without any color rule violation. The default value is nil.

GUI Equivalent

Command Options – DRD Edit, or Verify – Design – Process Rules –

DRD Options

Field Batch (tab) – MPT Checks – Colorability (DRD Options)

Examples

```
envGetVal("layout" "drdBatchColorability")
envSetVal("layout" "drdBatchColorability" 'boolean t)
envSetVal("layout" "drdBatchColorability" 'boolean nil)
```

Related Topics

Batch-Check Mode

Batch Tab

DRD Environment Variables

drdBatchColorShorts

```
layout drdBatchColorShorts boolean { t | nil }
```

Description

(ICADVM20.1 Only) Enables you to specify whether or not to check the design for color shorts violations in Batch mode. If set to true (t), DRD reports overlaps between shapes with different colors. The default value is nil.

GUI Equivalent

Command Options – DRD Edit, or Verify – Design – Process Rules –

DRD Options

Field Batch (tab) – MPT Checks – Color Shorts (DRD Options)

Examples

```
envGetVal("layout" "drdBatchColorShorts")
envSetVal("layout" "drdBatchColorShorts" 'boolean t)
envSetVal("layout" "drdBatchColorShorts" 'boolean nil)
```

Related Topics

Batch Tab

Batch-Check Mode

DRD Environment Variables

drdBatchConsiderGrayShapes

```
layout drdBatchConsiderGrayShapes boolean { t | nil }
```

Description

(ICADVM20.1 Only) Enables you to specify whether or not you want gray shapes to be considered as colored in Batch mode. If set to true (t), DRD does not report gray shapes as violations while performing color checks on a design for uncolored shapes. The default value is nil.

GUI Equivalent

Command Options – DRD Edit, or Verify – Design – Process Rules –

DRD Options

Field Batch (tab) – MPT Checks – Consider Gray Shapes (DRD

Options)

Examples

```
envGetVal("layout" "drdBatchConsiderGrayShapes")
envSetVal("layout" "drdBatchConsiderGrayShapes" 'boolean t)
envSetVal("layout" "drdBatchConsiderGrayShapes" 'boolean nil)
```

Related Topics

Batch Tab

Batch-Check Mode

DRD Environment Variables

drdBatchConsiderUnlockedShapesAsGray

layout drdBatchConsiderUnlockedShapesAsGray boolean { t | nil }

Description

(ICADVM20.1 Only) Enables you to specify whether or not you want unlocked shapes to be considered as gray in Batch mode. If set to true (t), DRD considers unlocked shapes as gray while performing color checks on a design. The default value is nil.

GUI Equivalent

Command Options – DRD Edit, or Verify – Design – Process Rules –

DRD Options

Field Batch (tab) – Consider Unlocked Shapes As Gray (DRD

Options)

Examples

```
envGetVal("layout" "drdBatchConsiderUnlockedShapesAsGray")
envSetVal("layout" "drdBatchConsiderUnlockedShapesAsGray" 'boolean t)
envSetVal("layout" "drdBatchConsiderUnlockedShapesAsGray" 'boolean nil)
```

Related Topics

Batch Tab

Batch-Check Mode

DRD Environment Variables

drdBatchFoundryRulesOnly

```
layout drdBatchFoundryRulesOnly boolean { t | nil }
```

Description

When set to t, DRD in Batch mode checks a design against only those rules that are provided by the foundry or manufacturer. The default value is nil.

GUI Equivalent

None

Examples

```
envGetVal("layout" "drdBatchFoundryRulesOnly")
envSetVal("layout" "drdBatchFoundryRulesOnly" 'boolean t)
envSetVal("layout" "drdBatchFoundryRulesOnly" 'boolean nil)
```

Related Topics

None

DRD Environment Variables

drdBatchReportCoveredNonPcellGrayShapes

layout drdBatchReportCoveredNonPcellGrayShapes boolean { t | nil }

Description

(ICADVM20.1 Only) Enables you to specify whether or not to report uncolored shapes inside non-Pcell instances in Batch mode even if these are covered with colored and locked shapes. If set to t, DRD reports such shapes. The default value is nil. This variable takes effect only when $\underline{drdBatchUncoloredShapes}$ and $\underline{drdBatchUnlocked}$ are set to t.

GUI Equivalent

None

Examples

```
envGetVal("layout" "drdBatchReportCoveredNonPcellGrayShapes")
envSetVal("layout" "drdBatchReportCoveredNonPcellGrayShapes" 'boolean t)
envSetVal("layout" "drdBatchReportCoveredNonPcellGrayShapes" 'boolean nil)
```

Related Topics

Batch Tab

Batch-Check Mode

DRD Environment Variables

drdBatchScopeAreaLimit

layout drdBatchScopeAreaLimit int integer_number

Description

Sets the area of check in DRD Batch mode to *Current Editable Cellview* (0), *Current View Area* (1), or *Current Specified Area* (2). The default value is 0.

GUI Equivalent

Command Options – DRD Edit, or Verify – Design – Process Rules –

DRD Options

Field Batch (tab) – Scope – Area (DRD Options)

Examples

```
envGetVal("layout" "drdBatchScopeAreaLimit")
envSetVal("layout" "drdBatchScopeAreaLimit" 'int 0)
envSetVal("layout" "drdBatchScopeAreaLimit" 'int 1)
envSetVal("layout" "drdBatchScopeAreaLimit" 'int 2)
```

Related Topics

Batch Tab

Batch-Check Mode

DRD Environment Variables

drdBatchScopeSpecifiedArea

layout drdBatchScopeSpecifiedArea string "((0.0 0.0) (0.0 0.0))"

Description

Specifies the area of the design on which you want to run DRD in Batch mode. The default value is $((0.0 \ 0.0) \ (0.0 \ 0.0))$.

GUI Equivalent

Command Options – DRD Edit, or Verify – Design – Process Rules –

DRD Options

Field Batch (tab) – Scope – Define Area (DRD Options)

Examples

```
envGetVal("layout" "drdBatchScopeSpecifiedArea")
envSetVal("layout" "drdBatchScopeSpecifiedArea" 'string "((0.0 0.0) (0.0 0.0))")
envSetVal("layout" "drdBatchScopeSpecifiedArea" 'string "((0.0 -0.048) (1.938 3.743))")
```

Related Topics

Batch Tab

Batch-Check Mode

DRD Environment Variables

drdBatchUncoloredShapes

```
layout drdBatchUncoloredShapes boolean { t | nil }
```

Description

(ICADVM20.1 Only) Enables you to specify whether or not to check the design for uncolored shape violations in Batch mode. If set to true (t), DRD reports uncolored (gray) shapes on layers that support multiple masks. The default value is nil.

GUI Equivalent

Command Options – DRD Edit, or Verify – Design – Process Rules –

DRD Options

Field Batch (tab) – MPT Checks – Uncolored Shapes (DRD)

Options)

Examples

```
envGetVal("layout" "drdBatchUncoloredShapes")
envSetVal("layout" "drdBatchUncoloredShapes" 'boolean t)
envSetVal("layout" "drdBatchUncoloredShapes" 'boolean nil)
```

Related Topics

Batch Tab

Batch-Check Mode

DRD Environment Variables

drdBatchUnlocked

```
layout drdBatchUnlocked boolean { t | nil }
```

Description

(ICADVM20.1 Only) Enables you to specify whether or not to check the design for color-unlocked shape violations in Batch mode. If set to true (t), DRD reports color-unlocked shapes on a layer. The default value is nil.

GUI Equivalent

Command Options – DRD Edit, or Verify – Design – Process Rules –

DRD Options

Field Batch (tab) – MPT Checks – Unlocked Shapes (DRD)

Options)

Examples

```
envGetVal("layout" "drdBatchUnlocked")
envSetVal("layout" "drdBatchUnlocked" 'boolean t)
envSetVal("layout" "drdBatchUnlocked" 'boolean nil)
```

Related Topics

Batch Tab

Batch-Check Mode

DRD Environment Variables

drdBatchVioLimit

layout drdBatchVioLimit int integer_number

Description

Sets the maximum number of violation markers to be created per call when DRD is run in Batch mode. The default value is 5000.

GUI Equivalent

Command Options – DRD Edit, or Verify – Design – Process Rules –

DRD Options

Field Batch (tab) – Scope – Marker Limit (DRD Options)

Examples

```
envGetVal("layout" "drdBatchVioLimit")
envSetVal("layout" "drdBatchVioLimit" 'int 100)
```

Related Topics

Batch Tab

Batch-Check Mode

DRD Environment Variables

drdColorability

```
layout drdColorability boolean { t | nil }
```

Description

(ICADVM20.1 Only) Enables you to specify whether or not to perform colorability check on the design. If set to true (t), DRD checks if a set of shapes is colorable without any color rule violation. The default value is nil.

GUI Equivalent

Command Options – DRD Edit, or Verify – Design – Process Rules –

DRD Options

Field Interactive (tab) – MPT Checks – Colorability (DRD Options)

Examples

```
envGetVal("layout" "drdColorability")
envSetVal("layout" "drdColorability" 'boolean t)
envSetVal("layout" "drdColorability" 'boolean nil)
```

Related Topics

Interactive Tab

DRD Environment Variables

drdColorShorts

```
layout drdColorShorts boolean { t | nil }
```

Description

(ICADVM20.1 Only) Enables you to specify whether or not to check the design for color shorts violations. If set to true (t), DRD reports overlaps between shapes with different colors. The default value is nil.

GUI Equivalent

Command Options – DRD Edit, or Verify – Design – Process Rules –

DRD Options

Field Interactive (tab) – MPT Checks – Color Shorts (DRD

Options)

Examples

```
envGetVal("layout" "drdColorShorts")
envSetVal("layout" "drdColorShorts" 'boolean t)
envSetVal("layout" "drdColorShorts" 'boolean nil)
```

Related Topics

Interactive Tab

DRD Environment Variables

drdConsiderGrayShapes

layout drdConsiderGrayShapes boolean { t | nil }

Description

(ICADVM20.1 Only) Enables you to specify whether or not you want gray shapes to be considered as colored. If set to true (t), DRD does not report gray shapes as violations while performing color checks on a design for uncolored shapes. The default value is nil.

GUI Equivalent

Command Options – DRD Edit, or Verify – Design – Process Rules –

DRD Options

Field Interactive (tab) – Consider gray shapes (DRD Options)

Examples

```
envGetVal("layout" "drdConsiderGrayShapes")
envSetVal("layout" "drdConsiderGrayShapes" 'boolean t)
envSetVal("layout" "drdConsiderGrayShapes" 'boolean nil)
```

Related Topics

Interactive Tab

DRD Environment Variables

drdConsiderUnlockedShapesAsGray

layout drdConsiderUnlockedShapesAsGray boolean { t | nil }

Description

(ICADVM20.1 Only) Enables you to specify whether or not you want unlocked shapes to be considered as gray. If set to true (t), DRD considers unlocked shapes as gray while performing color checks on a design. The default value is nil.

GUI Equivalent

Command Options – DRD Edit, or Verify – Design – Process Rules –

DRD Options

Field Interactive (tab) – Consider Unlocked Shapes As Gray

shapes (DRD Options)

Examples

```
envGetVal("layout" "drdConsiderUnlockedShapesAsGray")
envSetVal("layout" "drdConsiderUnlockedShapesAsGray" 'boolean t)
envSetVal("layout" "drdConsiderUnlockedShapesAsGray" 'boolean nil)
```

Related Topics

Interactive Tab

DRD Environment Variables

drdEditAllowOrthogonalAdjust

```
layout drdEditAllowOrthogonalAdjust boolean { t | nil }
```

Description

Allows DRD Enforce to move a shape in an orthogonal direction to avoid spacing errors. When set to nil, DRD enforces spacing errors only by stopping the cursor point.

GUI Equivalent

None

Examples

```
envGetVal("layout" "drdEditAllowOrthogonalAdjust")
envSetVal("layout" "drdEditAllowOrthogonalAdjust" 'boolean t)
envSetVal("layout" "drdEditAllowOrthogonalAdjust" 'boolean nil)
```

Related Topics

None

DRD Environment Variables

drdEditAutoText

```
layout drdEditAutoText boolean { t | nil }
```

Description

Suppresses any overlapping design rule violation text when set to t; displays all the text when set to nil. The default is t.

GUI Equivalent

None

Examples

```
envGetVal("layout" "drdEditAutoText")
envSetVal("layout" "drdEditAutoText" 'boolean t)
envSetVal("layout" "drdEditAutoText" 'boolean nil)
```

Related Topics

None

DRD Environment Variables

drdEditAutoTurnOffHalo

```
layout drdEditAutoTurnOffHalo boolean { t | nil }
```

Description

Enables you to specify whether or not to suppress the display of halos if the number of violations increases beyond a threshold. The default value is t.

GUI Equivalent

Command Options – DRD Edit, or Verify – Design – Process Rules –

DRD Options

Field Interactive (tab) – Display – Auto Halo Turn Off (DRD)

Options)

Examples

```
envGetVal("layout" "drdEditAutoTurnOffHalo")
envSetVal("layout" "drdEditAutoTurnOffHalo" 'boolean t)
envSetVal("layout" "drdEditAutoTurnOffHalo" 'boolean nil)
```

Related Topics

Interactive Tab

DRD Environment Variables

drdEditAutoTurnOffHaloLimit

layout drdEditAutoTurnOffHaloLimit int any_integer_between_1_and_1000

Description

Enables you to specify the threshold halo limit after which DRD automatically suppress the display of halos. The default value is 5.

GUI Equivalent

Command Options – DRD Edit, or Verify – Design – Process Rules –

DRD Options

Field Interactive (tab) – Display – Auto Halo Turn Off (select the

halo limit, between 1 and 1000, from the drop-down list) (DRD

Options)

Examples

```
envGetVal("layout" "drdEditAutoTurnOffHaloLimit")
envSetVal("layout" "drdEditAutoTurnOffHaloLimit" 'int 3)
envSetVal("layout" "drdEditAutoTurnOffHaloLimit" 'int 12)
```

Related Topics

Interactive Tab

DRD Environment Variables

drdEditBatchColorRules

layout drdEditBatchColorRules int integer_number

Description

(ICADVM20.1 Only) Sets the *MPT Checks* option in DRD Batch mode to On (0), Off (1), Only (2), or Conservative (3). With the MPT Checks options, you can check a design for conformance to the color rules defined in the technology file. The default value is 0.

GUI Equivalent

Command Options – DRD Edit, or Verify – Design – Process Rules –

DRD Options

Field Batch (tab) – MPT Checks (DRD Options)

Examples

```
envGetVal("layout" "drdEditBatchColorRules")
envSetVal("layout" "drdEditBatchColorRules" 'int 0)
envSetVal("layout" "drdEditBatchColorRules" 'int 1)
envSetVal("layout" "drdEditBatchColorRules" 'int 2)
envSetVal("layout" "drdEditBatchColorRules" 'int 3)
```

Related Topics

Batch Tab

Batch-Check Mode

DRD Environment Variables

drdEditBatchHierDepth

layout drdEditBatchHierDepth int any_integer_between_0_and_32

Description

Sets the number of hierarchy levels to be checked for violations in Batch mode when the hierarchy range is set to *current to user level*. You can specify a value in the range 0-32.

The default value is 0, which means that only the current level is checked. When the value is set to 32, the current level and all the levels below are checked.

GUI Equivalent

Command Options – DRD Edit, or Verify – Design – Process Rules –

DRD Options

Field Batch (tab) - Hierarchy Range (select current to user level

from the drop-down list and enter a value in the field provided)

(DRD Options)

Examples

```
envGetVal("layout" "drdEditBatchHierDepth")
envSetVal("layout" "drdEditBatchHierDepth" 'int 0)
envSetVal("layout" "drdEditBatchHierDepth" 'int 32)
```

Related Topics

Batch Tab

Batch-Check Mode

DRD Environment Variables

drdEditColorRules

layout drdEditColorRules int integer_number

Description

(ICADVM20.1 Only) Sets the *MPT Checks* option to *On* (0), *Off* (1), *Only* (2), or *Conservative* (3). With the MPT Checks options, you can check a design for conformance to the color rules defined in the technology file. The default value is 0.

GUI Equivalent

Command Options – DRD Edit, or Verify – Design – Process Rules –

DRD Options

Field Interactive (tab) – MPT Checks (DRD Options)

Examples

```
envGetVal("layout" "drdEditColorRules")
envSetVal("layout" "drdEditColorRules" 'int 0)
envSetVal("layout" "drdEditColorRules" 'int 1)
envSetVal("layout" "drdEditColorRules" 'int 2)
envSetVal("layout" "drdEditColorRules" 'int 3)
```

Related Topics

Interactive Tab

DRD Environment Variables

drdEditCompactorActiveLayerOnly

```
layout drdEditCompactorActiveLayerOnly boolean { t | nil }
```

Description

Applies the user spacing specified in the DRD Compactor Options form to only those shapes that are on the layer currently set as active in the Palette.

The default value is nil, which means that the user spacing is applied to shapes on all layers.

GUI Equivalent

None

Examples

```
envGetVal("layout" "drdEditCompactorActiveLayerOnly")
envSetVal("layout" "drdEditCompactorActiveLayerOnly" 'boolean t)
envSetVal("layout" "drdEditCompactorActiveLayerOnly" 'boolean nil)
```

Related Topics

DRD Environment Variables

drdEditCompactorAnchorMode

```
layout drdEditCompactorAnchorMode cyclic { "automatic" | "external" | "target" }
```

Description

Sets the default anchor mode for compaction.

- automatic corresponds to *Use last shape in direction of compaction*.
- external corresponds to *Use first unselected shape in direction of compaction*.
- target corresponds to *Use DRD Targets*.

The default value is automatic.

This environment variable corresponds to the *Anchor Mode* group box on the DRD Compactor Options form.

GUI Equivalent

Command Options – DRD Compactor

Field Anchor Mode (DRD Compactor Options)

Examples

```
envGetVal("layout" "drdEditCompactorAnchorMode")
envSetVal("layout" "drdEditCompactorAnchorMode" 'cyclic "automatic")
envSetVal("layout" "drdEditCompactorAnchorMode" 'cyclic "external")
envSetVal("layout" "drdEditCompactorAnchorMode" 'cyclic "target")
```

Related Topics

DRD Environment Variables

drdEditCompactorInterOnly

```
layout drdEditCompactorInterOnly boolean { t | nil }
```

Description

Sets the state of the *Maintain relative positions of selected objects* check box. The default is nil.

Note: This field is enabled for selection only when one of the following anchor modes is selected:

- Use first unselected shape in direction of compaction
- Use DRD Targets

GUI Equivalent

Command Options – DRD Compactor

Field Anchor Mode (DRD Compactor Options)

Examples

```
envGetVal("layout" "drdEditCompactorInterOnly")
envSetVal("layout" "drdEditCompactorInterOnly" 'boolean t)
envSetVal("layout" "drdEditCompactorInterOnly" 'boolean nil)
```

Related Topics

DRD Environment Variables

drdEditCompactorMode

```
layout drdEditCompactorMode cyclic { "both" | "compact" | "spread" }
```

Description

Sets the default compaction mode.

- both corresponds to Compact and Spread.
- compact corresponds to Compact only.
- spread corresponds to Spread only.

The default is "both".

GUI Equivalent

Command Options – DRD Compactor

Field Compaction Mode (DRD Compactor Options)

Examples

```
envGetVal("layout" "drdEditCompactorMode")
envSetVal("layout" "drdEditCompactorMode" 'cyclic "both")
envSetVal("layout" "drdEditCompactorMode" 'cyclic "compact")
envSetVal("layout" "drdEditCompactorMode" 'cyclic "spread")
```

Related Topics

Compactor User Interface

208

DRD Environment Variables

drdEditCompactorSpacingMode

```
layout drdEditCompactorSpacingMode cyclic { "minSpacing" | "userSpacing" }
```

Description

Sets the default spacing option for compaction.

- minSpacing corresponds to DRD Enforce spacing.
- userSpacing corresponds to *User spacing*. You can set the user spacing value by using <u>drdEditCompactorUserSpacing</u>.

The default is "minSpacing".

GUI Equivalent

Command Options – DRD Compactor

Field Spacing Options (DRD Compactor Options)

Examples

```
envGetVal("layout" "drdEditCompactorSpacingMode")
envSetVal("layout" "drdEditCompactorSpacingMode" 'cyclic "minSpacing")
envSetVal("layout" "drdEditCompactorSpacingMode" 'cyclic "userSpacing")
```

Related Topics

DRD Environment Variables

drdEditCompactorUserSpacing

layout drdEditCompactorUserSpacing float float_number

Description

Defines the custom spacing value used by DRD Compactor when compacting objects. The default is 0.0.

GUI Equivalent

Command Options – DRD Compactor

Field User spacing (DRD Compactor Options)

Examples

```
envGetVal("layout" "drdEditCompactorUserSpacing")
envSetVal("layout" "drdEditCompactorUserSpacing" 'float 0.06)
```

Related Topics

Compactor User Interface

<u>drdCompactSelSet</u>

DRD Environment Variables

drd Edit Density Type Check

```
layout drdEditDensityTypeCheck cyclic { "all" | "min" | "max" }
```

Description

Determines the type of density check to perform.

- all performs both minimum and maximum density checks.
- min performs only minimum density checks.
- max performs only maximum density checks.

The default is all.

GUI Equivalent

None

Examples

```
envGetVal("layout" "drdEditDensityTypeCheck")
envSetVal("layout" "drdEditDensityTypeCheck" 'cyclic "all")
envSetVal("layout" "drdEditDensityTypeCheck" 'cyclic "min")
envSetVal("layout" "drdEditDensityTypeCheck" 'cyclic "max")
```

Related Topics

Using Density Constraints

DRD Environment Variables

drdEditDisplayArrows

```
layout drdEditDisplayArrows boolean { t | nil }
```

Description

Sets the state of the *Arrows* check box. When set to t, displays arrows to indicate optimum spacing between objects. The default value is t.

GUI Equivalent

Command Options – DRD Edit, or Verify – Design – Process Rules –

DRD Options

Field Interactive (tab) – Display – Arrows (DRD Options)

Examples

```
envGetVal("layout" "drdEditDisplayArrows")
envSetVal("layout" "drdEditDisplayArrows" 'boolean t)
envSetVal("layout" "drdEditDisplayArrows" 'boolean nil)
```

Related Topics

Interactive Tab

DRD Environment Variables

drdEditDisplayDashed

```
layout drdEditDisplayDashed boolean { t | nil }
```

Description

Sets the state of the *Dashed* check box. When set to t, displays halos using dashed lines. The default is nil.

GUI Equivalent

None

Examples

```
envGetVal("layout" "drdEditDisplayDashed")
envSetVal("layout" "drdEditDisplayDashed" 'boolean t)
envSetVal("layout" "drdEditDisplayDashed" 'boolean nil)
```

Related Topics

Managing Interactive Display Options

<u>drdEditDisplayHalo</u>

DRD Environment Variables

drdEditDisplayDrawingColor

layout drdEditDisplayDrawingColor string "layer_name"

Description

Specifies the layer to be used to display halos, arrows, and text. The default value is y0.

layer_name is the name of the layer.

GUI Equivalent

Command Options – DRD Edit, or Verify – Design – Process Rules –

DRD Options

Field Drawing Color (DRD Options)

Examples

```
envGetVal("layout" "drdEditDisplayDrawingColor")
envSetVal("layout" "drdEditDisplayDrawingColor" 'string "y2")
```

Related Topics

DRD Environment Variables

drdEditDisplayEdges

```
layout drdEditDisplayEdges boolean { t | nil }
```

Description

Sets the state of the Edges check box. When set to t, highlights the edge where the violation occurs. The default value is nil.

GUI Equivalent

Command Options – DRD Edit, or Verify – Design – Process Rules –

DRD Options

Field Interactive (tab) – Display – Edges (DRD Options)

Examples

```
envGetVal("layout" "drdEditDisplayEdges")
envSetVal("layout" "drdEditDisplayEdges" 'boolean t)
envSetVal("layout" "drdEditDisplayEdges" 'boolean nil)
```

Related Topics

Interactive Tab

DRD Environment Variables

drdEditDisplayHalo

```
layout drdEditDisplayHalo boolean { t | nil }
```

Description

Sets the state of the Halo check box. When set to t, displays halos around stationary objects while creating or editing an object that violates a spacing constraint. The default value is nil.

GUI Equivalent

Command Options – DRD Edit, or Verify – Design – Process Rules –

DRD Options

Field Interactive (tab) – Display – Halo (DRD Options)

Examples

```
envGetVal("layout" "drdEditDisplayHalo")
envSetVal("layout" "drdEditDisplayHalo" 'boolean t)
envSetVal("layout" "drdEditDisplayHalo" 'boolean nil)
```

Related Topics

<u>drdEditDisplayTrueColor</u>

Interactive Tab

DRD Environment Variables

drdEditDisplayMarkers

```
layout drdEditDisplayMarkers boolean { t | nil }
```

Description

When set to t, the Post Edit mode is enabled. The default value is nil.

GUI Equivalent

Command Options – DRD Edit, or Verify – Design – Process Rules –

DRD Options

Field Interactive (tab) – Post Edit – Enabled (DRD Options)

Examples

```
envGetVal("layout" "drdEditDisplayMarkers")
envSetVal("layout" "drdEditDisplayMarkers" 'boolean t)
envSetVal("layout" "drdEditDisplayMarkers" 'boolean nil)
```

Related Topics

DRD Environment Variables

drdEditDisplayMaxChars

layout drdEditDisplayMaxChars int integer_number

Description

Specifies the maximum number of characters to display when *Ref Text* is selected. The default is 0, which displays the complete reference text.

The value -1 displays the reference text found before the first delimiter (", " or "; ").



The value set by this environment variable is used only if the drdEditDisplayTechRefs environment variable is set to t.

GUI Equivalent

None

Examples

```
envGetVal("layout" "drdEditDisplayMaxChars")
envSetVal("layout" "drdEditDisplayMaxChars" 'int -1)
envSetVal("layout" "drdEditDisplayMaxChars" 'int 0)
envSetVal("layout" "drdEditDisplayMaxChars" 'int 14)
```

Related Topics

Managing Interactive Display Options

<u>drdEditDisplayTechRefs</u>

<u>drdEditDisplayText</u>

<u>drdEditFontSize</u>

DRD Environment Variables

drdEditDisplayTechRefs

```
layout drdEditDisplayTechRefs boolean { t | nil }
```

Description

Sets the state of the *Ref Text* check box. When set to t, displays the reference text for a constraint from the technology file, followed by the constraint value. Otherwise, displays only the name and value of the violated constraint.

The default value is nil.

GUI Equivalent

Command Options – DRD Edit, or Verify – Design – Process Rules –

DRD Options

Field Interactive (tab) – Display – Ref Text (DRD Options)

Examples

```
envGetVal("layout" "drdEditDisplayTechRefs")
envSetVal("layout" "drdEditDisplayTechRefs" 'boolean t)
envSetVal("layout" "drdEditDisplayTechRefs" 'boolean nil)
```

Related Topics

<u>drdEditDisplayText</u>

Interactive Tab

Managing Interactive Display Options

DRD Environment Variables

drdEditDisplayText

```
layout drdEditDisplayText boolean { t | nil }
```

Description

Sets the state of the *Rule Text* check box. When set to t, displays the name of the violated constraint at the location where the violation occurs. The default value is t.

GUI Equivalent

Command Options – DRD Edit, or Verify – Design – Process Rules –

DRD Options

Field Interactive (tab) – Display – Rule Text (DRD Options)

Examples

```
envGetVal("layout" "drdEditDisplayText")
envSetVal("layout" "drdEditDisplayText" 'boolean t)
envSetVal("layout" "drdEditDisplayText" 'boolean nil)
```

Related Topics

<u>drdEditDisplayTechRefs</u>

Interactive Tab

Managing Interactive Display Options

DRD Environment Variables

drdEditDisplayTrueColor

```
layout drdEditDisplayTrueColor boolean { t | nil }
```

Description

Sets the state of the *True Color* check box. When set to t, displays halos in true color. The default value is nil.

GUI Equivalent

Command Options – DRD Edit, or Verify – Design – Process Rules –

DRD Options

Field Interactive (tab) – Display – True Color (DRD Options)

Examples

```
envGetVal("layout" "drdEditDisplayTrueColor")
envSetVal("layout" "drdEditDisplayTrueColor" 'boolean t)
envSetVal("layout" "drdEditDisplayTrueColor" 'boolean nil)
```

Related Topics

<u>drdEditDisplayHalo</u>

Interactive Tab

Managing Interactive Display Options

DRD Environment Variables

drdEditEnableBatchCheckOnReadOnlyDesign

layout drdEditEnableBatchCheckOnReadOnlyDesign boolean { t | nil }

Description

Enables batch check for read-only designs if set to t. As a result, markers are created in the canvas when the design is opened in read-only mode. The default is nil.

GUI Equivalent

None

Examples

```
envGetVal("layout" "drdEditEnableBatchCheckOnReadOnlyDesign")
envSetVal("layout" "drdEditEnableBatchCheckOnReadOnlyDesign" 'boolean t)
envSetVal("layout" "drdEditEnableBatchCheckOnReadOnlyDesign" 'boolean nil)
```

Related Topics

Batch-Check Mode

drdBatchCheck

DRD Environment Variables

drdEditFontSize

layout drdEditFontSize int integer_number

Description

Sets the size of the reference text displayed in the layout canvas. You can specify a value in the range 4-50. The minimum is 4 and the maximum is 50. The default is 12.

GUI Equivalent

None

Examples

```
envGetVal("layout" "drdEditFontSize")
envSetVal("layout" "drdEditFontSize" 'int 14)
```

Related Topics

Managing Interactive Display Options

<u>drdEditDisplayMaxChars</u>

drdEditDisplayTechRefs

<u>drdEditDisplayText</u>

DRD Environment Variables

drdEditHierDepth

layout drdEditHierDepth int any_integer_between_0_and_32

Description

Sets the number of hierarchy levels to be checked for violations when the hierarchy range is set to *current to user level*. You can specify a value in the range 0-32.

The default value is 0, which means that only the current level is checked. When the value is set to 32, the current level and all levels below are checked.

GUI Equivalent

Command Options – DRD Edit, or Verify – Design – Process Rules –

DRD Options

Field Interactive (tab) - Hierarchy Range (select current to user

level from the drop-down list and enter a value in the field

provided) (DRD Options)

Examples

```
envGetVal("layout" "drdEditHierDepth")
envSetVal("layout" "drdEditHierDepth" 'int 0)
envSetVal("layout" "drdEditHierDepth" 'int 32)
```

Related Topics

Interactive Tab

DRD Environment Variables

drdEditIgnoreIntraInstanceChecks

layout drdEditIgnoreIntraInstanceChecks boolean { t | nil }

Description

Causes intra-instance checks to be ignored in Enforce, Notify, and Post Edit modes when an instance is created or when an edit command, such as move, copy, or stretch, is run. The default value is t.

To enable intra-instance checks, set the environment variable to nil.

GUI Equivalent

None

Examples

```
envGetVal("layout" "drdEditIgnoreIntraInstanceChecks")
envSetVal("layout" "drdEditIgnoreIntraInstanceChecks" 'boolean t)
envSetVal("layout" "drdEditIgnoreIntraInstanceChecks" 'boolean nil)
```

Related Topics

Ignoring Violations inside Instances

DRD Environment Variables

drdEditMode

```
layout drdEditMode cyclic { "off" | "notify" | "enforce" | "enforce&notify" }
```

Description

Enables DRD editing in the specified mode.

- off turns off DRD editing.
- notify enables DRD editing in Notify mode.
- enforce enables DRD editing in Enforce mode.
- enforce¬ify enables DRD editing in both Notify and Enforce modes.

GUI Equivalent

Command Options – DRD Edit, or Verify – Design – Process Rules –

DRD Options

Field DRD Mode (DRD Options)

Examples

```
envGetVal("layout" "drdEditMode")
envSetVal("layout" "drdEditMode" 'cyclic "off")
envSetVal("layout" "drdEditMode" 'cyclic "notify")
envSetVal("layout" "drdEditMode" 'cyclic "enforce")
envSetVal("layout" "drdEditMode" 'cyclic "enforce&notify")
```

Related Topics

DRD Environment Variables

drdEditPaletteLayers

```
layout drdEditPaletteLayers boolean { t | nil }
```

Description

If set to t, DRD monitors only those layers for violations that are defined in the *Layers* panel of the *Palette* assistant. The default value is nil.

GUI Equivalent

Command Options – DRD Edit, or Verify – Design – Process Rules –

DRD Options

Field Filters (tab) – Layer – Palette (DRD Options)

Examples

```
envGetVal("layout" "drdEditPaletteLayers")
envSetVal("layout" "drdEditPaletteLayers" 'boolean t)
envSetVal("layout" "drdEditPaletteLayers" 'boolean nil)
```

Related Topics

Filters Tab

Selecting Layers for Verification

DRD Environment Variables

drdEditSlidingWindowSize

layout drdEditSlidingWindowSize float float_number

Description

Checks for violations within a window with the specified size as you move the mouse pointer across the canvas while creating a wire or a pathSeg. The default window size is 1.0 micron.

To disable the environment variable, set it to 0.0.

GUI Equivalent

None

Examples

```
envGetVal("layout" "drdEditSlidingWindowSize")
envSetVal("layout" "drdEditSlidingWindowSize" 'float 1.5)
```

Related Topics

None

DRD Environment Variables

drdEditSmartSnapAllowedWidthSnap

```
layout drdEditSmartSnapAllowedWidthSnap boolean { t | nil }
```

Description

(ICADVM20.1 Only) Turns on or off discrete dimension gravity. The default is t, which implies that discrete dimension gravity is turned on by default.

GUI Equivalent

None

Examples

```
envGetVal("layout" "drdEditSmartSnapAllowedWidthSnap")
envSetVal("layout" "drdEditSmartSnapAllowedWidthSnap" 'boolean t)
envSetVal("layout" "drdEditSmartSnapAllowedWidthSnap" 'boolean nil)
```

Related Topics

Snapping Objects to Legal Width Values

DRD Environment Variables

drdEditSmartSnapAperture

layout drdEditSmartSnapAperture float float_number

Description

GUI Equivalent

None

Examples

```
envGetVal("layout" "drdEditSmartSnapAperture")
envSetVal("layout" "drdEditSmartSnapAperture" 'float 0.2)
```

DRD Environment Variables

drdEditVioLimit

layout drdEditVioLimit int integer_number

Description

Sets the number of violations displayed at one time when Enforce or Notify mode is enabled. The default is 10.

GUI Equivalent

None

Examples

```
envGetVal("layout" "drdEditVioLimit")
envSetVal("layout" "drdEditVioLimit" 'int 1)
```

Related Topics

DRD Environment Variables

drdInteractiveFoundryRulesOnly

```
layout drdInteractiveFoundryRulesOnly boolean { t | nil }
```

Description

When set to t, DRD in interactive mode checks a design against only those rules that are provided by the foundry or manufacturer. The default value is nil.

GUI Equivalent

None

Examples

```
envGetVal("layout" "drdInteractiveFoundryRulesOnly")
envSetVal("layout" "drdInteractiveFoundryRulesOnly" 'boolean t)
envSetVal("layout" "drdInteractiveFoundryRulesOnly" 'boolean nil)
```

Related Topics

None

DRD Environment Variables

drdPostEditAllWindows

```
layout drdPostEditAllWindows boolean { t | nil }
```

Description

Runs DRD checks in Post Edit mode in all open windows if set to t; otherwise, in only the current window. The default value is nil.

GUI Equivalent

None

Examples

```
envGetVal("layout" "drdPostEditAllWindows")
envSetVal("layout" "drdPostEditAllWindows" 'boolean t)
envSetVal("layout" "drdPostEditAllWindows" 'boolean nil)
```

Related Topics

DRD Operating Modes

233

DRD Environment Variables

drdPostEditEIPAIIInstances

```
layout drdPostEditEIPAllInstances boolean { t | nil }
```

Description

Checks the environment around all instances of a cell for DRC violations when a shape in that cell is edited in place. The default value is nil.

GUI Equivalent

None

Examples

```
envGetVal("layout" "drdPostEditEIPAllInstances")
envSetVal("layout" "drdPostEditEIPAllInstances" 'boolean t)
envSetVal("layout" "drdPostEditEIPAllInstances" 'boolean nil)
```

Related Topics

DRD Environment Variables

drdPostEditEIPAllWindows

```
layout drdPostEditEIPAllWindows boolean { t | nil }
```

Description

Applies the value of drdPostEditEIPCheckTopView and drdPostEditEIPAllInstances to all open design windows. The default value is nil.

GUI Equivalent

None

Examples

```
envGetVal("layout" "drdPostEditEIPAllWindows")
envSetVal("layout" "drdPostEditEIPAllWindows" 'boolean t)
envSetVal("layout" "drdPostEditEIPAllWindows" 'boolean nil)
```

Related Topics

DRD Operating Modes

<u>drdPostEditEIPAllInstances</u>

<u>drdPostEditEIPCheckTopView</u>

DRD Environment Variables

drdPostEditEIPCheckTopView

```
layout drdPostEditEIPCheckTopView boolean { t | nil }
```

Description

Checks for DRC violations between the shape being edited and the shapes in the parent hierarchy and the shapes in the neighboring cells. The default value is t.

GUI Equivalent

None

Examples

```
envGetVal("layout" "drdPostEditEIPCheckTopView")
envSetVal("layout" "drdPostEditEIPCheckTopView" 'boolean t)
envSetVal("layout" "drdPostEditEIPCheckTopView" 'boolean nil)
```

Related Topics

<u>drdPostEditEIPAllInstances</u>

drdPostEditEIPAllWindows

DRD Environment Variables

drdPostEditIgnoreModgen

```
layout drdPostEditIgnoreModgen boolean { t | nil }
```

Description

Disables DRD Post Edit mode in the Modgen environment when set to t. This prevents violation markers from being created when an object is edited in the Modgen Editor window. The default value is nil.

GUI Equivalent

None

Examples

```
envGetVal("layout" "drdPostEditIgnoreModgen")
envSetVal("layout" "drdPostEditIgnoreModgen" 'boolean t)
envSetVal("layout" "drdPostEditIgnoreModgen" 'boolean nil)
```

Related Topics

DRD Environment Variables

drdPostEditReplaceAllMarkers

```
layout drdPostEditReplaceAllMarkers boolean { t | nil }
```

Description

Deletes all existing markers before new markers are generated in DRD Post Edit mode, if set to t. The default value is nil.

GUI Equivalent

None

Examples

```
envGetVal("layout" "drdPostEditReplaceAllMarkers")
envSetVal("layout" "drdPostEditReplaceAllMarkers" 'boolean t)
envSetVal("layout" "drdPostEditReplaceAllMarkers" 'boolean nil)
```

Related Topics

DRD Environment Variables

drdPostEditTimeOut

layout drdPostEditTimeout float float_number

Description

Sets the timeout threshold for Post Edit mode. The default is 1.0 second.

GUI Equivalent

Command Options – DRD Edit, or Verify – Design – Process Rules –

DRD Options

Field Timeout (secs) (DRD Options)

Examples

```
envGetVal("layout" "drdPostEditTimeout")
envSetVal("layout" "drdPostEditTimeout" 'float 9.0)
```

Related Topics

DRD Environment Variables

drdPostEditVioLimit

layout drdPostEditVioLimit int integer_number

Description

Sets the number of post edit markers to be created per call. The default value is 5000.

GUI Equivalent

Command Options – DRD Edit, or Verify – Design – Process Rules –

DRD Options

Field Interactive (tab) – Post Edit – Marker Limit (DRD Options)

Examples

```
envGetVal("layout" "drdPostEditVioLimit")
envSetVal("layout" "drdPostEditVioLimit" 'int 1)
envSetVal("layout" "drdPostEditVioLimit" 'int 5000)
```

Related Topics

DRD Operating Modes

Interactive Tab

DRD Environment Variables

drdUncoloredShapes

```
layout drdUncoloredShapes boolean { t | nil }
```

Description

(ICADVM20.1 Only) Enables you to specify whether or not to check the design for uncolored shape violations. If set to true (t), DRD reports uncolored (gray) shapes on layers that support multiple masks. The default value is nil.

GUI Equivalent

Command Options – DRD Edit, or Verify – Design – Process Rules –

DRD Options

Field Interactive (tab) – MPT Checks – Uncolored Shapes (DRD

Options)

Examples

```
envGetVal("layout" "drdUncoloredShapes")
envSetVal("layout" "drdUncoloredShapes" 'boolean t)
envSetVal("layout" "drdUncoloredShapes" 'boolean nil)
```

Related Topics

Interactive Tab

Performing Color Checks (ICADVM20.1 Only)

DRD Environment Variables

drdUnlocked

```
layout drdUnlocked boolean { t | nil }
```

Description

(ICADVM20.1 Only) Enables you to specify whether or not to check the design for color-unlocked shape violations. If set to true (t), DRD reports color-unlocked shapes on a layer. The default value is nil.

GUI Equivalent

Command Options – DRD Edit, or Verify – Design – Process Rules –

DRD Options

Field Interactive (tab) – MPT Checks – Unlocked Shapes (DRD

Options)

Examples

```
envGetVal("layout" "drdUnlocked")
envSetVal("layout" "drdUnlocked" 'boolean t)
envSetVal("layout" "drdUnlocked" 'boolean nil)
```

Related Topics

Interactive Tab

Performing Color Checks (ICADVM20.1 Only)

DRD Environment Variables

drdUseBlockageForExtCheck

```
layout drdUseBlockageForExtCheck boolean { t | nil }
```

Description

Causes blockages to be included while via extensions are checked for end-of-line spacing. The default value is nil.

GUI Equivalent

None

Examples

```
envGetVal("layout" "drdUseBlockageForExtCheck")
envSetVal("layout" "drdUseBlockageForExtCheck" 'boolean t)
envSetVal("layout" "drdUseBlockageForExtCheck" 'boolean nil)
leSetEnv("drdUseBlockageForExtCheck" t)
```

Related Topics

DRD Editing and Blockage Objects

DRD Environment Variables

drdUseNetName

layoutXL drdUseNetName boolean { t | nil }

Description

See drdUseNetName in the Virtuoso Layout Suite XL User Guide.

DRD Environment Variables

drdUseNewDerivedLayerEngine

```
layout drdUseNewDerivedLayerEngine boolean { t | nil }
```

Description

Determines the derived layer engine that DRD uses. The default value is t, which means that DRD, by default, uses the newly introduced derived layer engine. This new engine supports a set of derived layer operators that enable you to check for violations on layers derived from Boolean operations and on layers derived by sizing other layers.

GUI Equivalent

None

Examples

```
envGetVal("layout" "drdUseNewDerivedLayerEngine")
envSetVal("layout" "drdUseNewDerivedLayerEngine" 'boolean t)
envSetVal("layout" "drdUseNewDerivedLayerEngine" 'boolean nil)
```

Related Topics

Supported Derived Layers

DRD Environment Variables

drdValidLayersGrayCheck

```
layout drdValidLayersGrayCheck boolean { t | nil }
```

Description

Indicates that only the colorable layers specified in the <u>validLayers</u> constraint are checked for uncolored and unlocked shapes.

When set to nil, all colorable layers are checked for uncolored and unlocked shapes.

GUI Equivalent

None

Examples

```
envGetVal("layout" "drdValidLayersGrayCheck")
envSetVal("layout" "drdValidLayersGrayCheck" 'boolean t)
envSetVal("layout" "drdValidLayersGrayCheck" 'boolean nil)
```

DRD Environment Variables

drdVDRIncludePurposes

layout drdVDRIncludePurposes string { "" | "fill" }

Description

(ICADVM20.1 Only) Determines whether the shapes with purpose fill are included or excluded from minVoltageSpacing checks by DRD. The default value of this environment variable is an empty string (""), which implies that shapes with purpose fill are excluded.

Set the environment variable to fill if you want to include purpose fill in minVoltageSpacing checks.

Note: Shapes drawn on a purpose whose 'parent purpose' is set to fill in the technology file also exhibit the behavior described above; that is, such shapes are excluded from minVoltageSpacing checks if this environment is set to an empty string (" "). The shapes are included in minVoltageSpacing checks if the environment is set to fill.

GUI Equivalent

None

Example

Consider the following purposes:

- Purpose purpA has 'parentPurpose = fill.
- Purpose purpB has 'parentPurpose = fill.
- Purpose purpC has 'parentPurpose = drawing.

Scenario 1

layout drdVDRIncludePurposes 'string "fill"

- DRD reports minVoltageSpacing violations for purpose fill, and also for purposes purpA and purpB because the 'parent purpose' for these purposes is fill.
- DRD reports minVoltageSpacing violations for purpose purpC because its 'parent purpose' is drawing—a purpose other than fill.

DRD Environment Variables

Scenario 2

layout drdVDRIncludePurposes 'string ""

- DRD does not report minVoltageSpacing violations for purposes fill, purpA, purpB.
- DRD reports minVoltageSpacing violations for purpC.

Related Topics

Supported Layer-Purpose Pair and Voltage-Dependent Constraints

DRD Environment Variables

maxShapesAllowedForCompaction

layout maxShapesAllowedForCompaction int integer_number

Description

Sets the number of shapes that can be compacted. If the number of shapes involved in compaction is greater than the value set by this environment variable, the compactor does not compact the selected objects and displays a warning message in the CIW.

The default value is 500.

GUI Equivalent

None

Examples

```
envGetVal("layout" "maxShapesAllowedForCompaction")
envSetVal("layout" "maxShapesAllowedForCompaction" 'int 200)
```

Related Topics

DRD Interactive Compactor

Virtuoso Design Rule Driven Editing User Guide DRD Environment Variables

C

Blockage and Boundary Objects

DRD Editing and Blockage Objects

In DRD editing, objects are blocked as summarized in the following table:

Blockage Type	Rule	Blocked Objects for DRD Editing
fill	No fill may be placed in the blockage area on the blockage layer.	Shapes on blockage layer with purpose fillBlock
pin	No shape pins may be placed in the blockage area on the blockage layer. No instance pins that use the blockage layer may be placed in this blockage area.	Pins
placement (area)	No instances or instance halos may be placed in the blockage area.	Instances
routing (layer)	No interconnect shapes may be placed in the blockage area on the blockage layer.	Top-level shapes (rectangles, polygons, path multipart paths, wires) and vias
slot	No slotting may be placed in the blockage area on the blockage layer.	Shapes on blockage layer with purpose slotBlock

Effective Width

Effective width is an attribute that you can assign to blockages. If defined, it is used as the width of the blockage when looking up the minSpacing value. DRD warns of a violation in Notify mode and prevents placement in Enforce mode if spacing between a blockage and an

Blockage and Boundary Objects

object is less than the minSpacing value defined for the layer. It is not necessary to have the effective width defined for all blockages. If the effective width of a blockage is not defined, the exact width of the blockage is used to look up the minSpacing value.

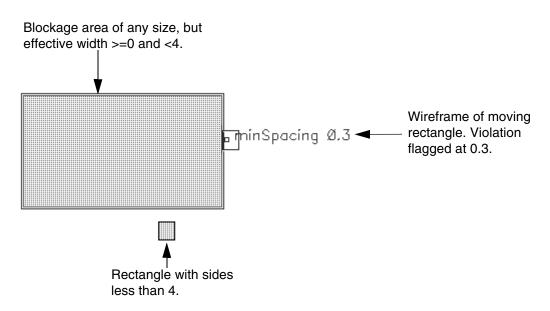
Using Effective Width

You can alter the effective width of a blockage object to flag or prevent spacing violations when using DRD while creating or editing certain shapes. The effective width applies to the entire blockage. The larger of the two values, the blockage effective width or the side of the non-blockage object, is used to index into the spacing table to determine the minSpacing value that applies.

In the following example, each side of the moving rectangle is less than 4. Different spacing values apply as the effective width of the blockage changes.

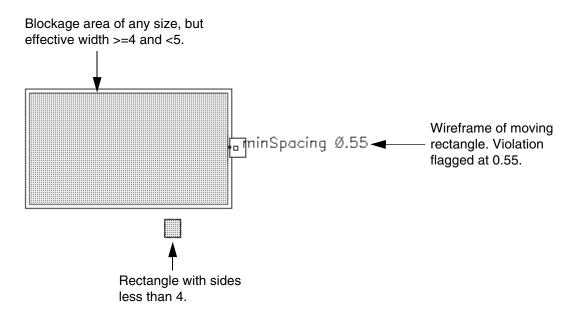
Suppose that the technology file contains the following spacing values for Metal3:

When the effective width of the blockage is set to a value greater than or equal to 0 and less than 4, the spacing violation is flagged at 0.3, as shown below:

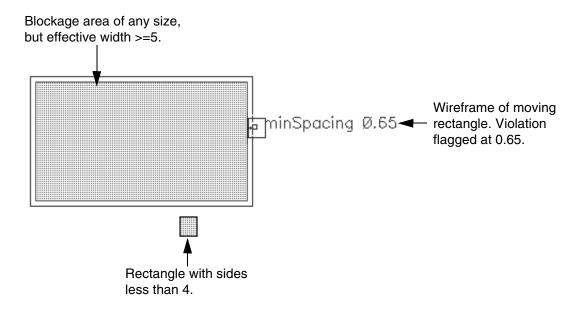


Blockage and Boundary Objects

When the effective width is greater than or equal to 4 and less than 5, the spacing violation is flagged at 0.55, as shown below:



When the effective width is changed from 5.0 to any number larger than 5.0, the spacing violation is flagged at 0.65, as shown below:



Blockage and Boundary Objects

DRD Editing and PR Boundary Objects

A PR boundary is a polygonal object that defines the legal placement and routing area in a design.

DRD provides interactive feedback when PR boundary objects are created and edited. The PR boundary spacing and extension rules defined in the technology file dictate that objects enclosed within the PR boundary object on the specified layer must be at the specified distance from it, or may extend beyond it by some minimum distance. An optional parameter 'coincidentAllowed specifies whether the spacing rules allow an object on the specified layer to coincide with the PR boundary object.

The following are examples of PR boundary spacing rules:

■ Any metall object enclosed within a PR boundary object must be at least 0.5 user units away from it.

```
( minPRBoundaryInteriorHalo "metal1" 0.5 )
```

Any metal2 object must either be enclosed by the PR boundary object, separated from it by at least 0.3 user units, or can coincide with an edge of the PR boundary object.

```
( minPRBoundaryInteriorHalo "metal2" 0.3 'coincidentAllowed )
```

Any metal2 object that extends beyond the PR boundary must have an extension of at least 1.0 user units over the PR boundary.

```
( minPRBoundaryExtension "metal2" 1.0 )
```

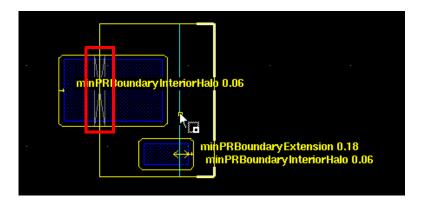
If objects already exist when the PR boundary object is created, DRD does not attempt to interactively flag errors with warnings (in Notify mode), nor does it attempt to adjust the drag point or move the objects already placed (in Enforce mode). However, if Post-Edit mode is enabled, violations, if any, are flagged after the PR boundary object has been placed. For example, if a path exists and a PR boundary is created to overlap it, the violations are flagged with post-edit markers, if DRD Post-Edit mode is enabled.

However, if a path that violates the boundary rules is subsequently created, the path is interactively checked and errors are flagged with markers, depending on the DRD mode that is currently active.

When you use the *Create*, *Move*, *Stretch* or any other edit command on a PR boundary object, violations are flagged as shown below. You can also see in this example the post-edit

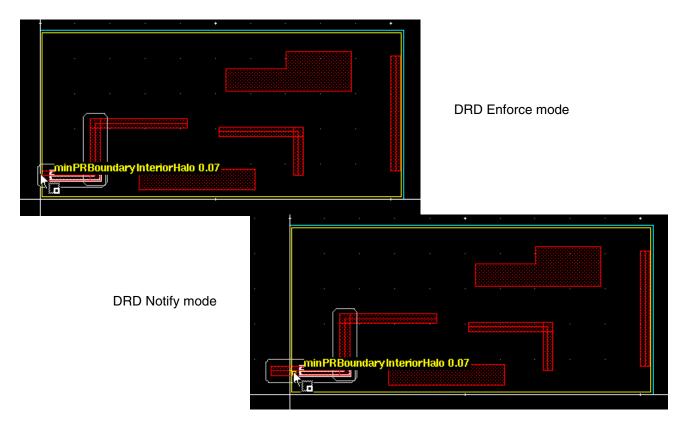
Blockage and Boundary Objects

marker that was created when the PR boundary object was placed on an already existing rectangular shape.



Creating and Editing Non-PR Boundary Objects

When non-PR boundary objects are created or edited, DRD displays warnings and enforces drag point adjustment when a PR boundary object is encountered. For example, if a path segment is stretched too close to the inside edge of a PR boundary object, DRD displays warnings, and, in Enforce mode, adjusts the drag point so that the object does not violate the spacing or overlap rule.



Blockage and Boundary Objects

The following objects are supported by DRD with respect to PR boundary objects:

- Rectangles
- Polygons
- Paths and path segments
- Multipart paths
- Instances
- Vias

When you use the *Edit In Place* or *Descend* command to edit objects within the hierarchy, DRD checks are performed against the PR boundary at the same hierarchical level as the object being edited, if a PR boundary exists at that level. Checking and enforcement is done through the hierarchy up to the depth specified in the <u>DRD Options</u> form.