cadence®

Virtuoso Design Rule Driven Editing User Guide

Product Version IC23.1
September 2023

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

Contents

1
Design Rule Driven Editing 9
Turning on DRD Editing 10
Setting Up DRC Options 10
DRD Configuration File 15
Specifying Options on the DRC Toolbar 16
DRD Operating Modes 16
Using Enforce Mode 17
Using Notify Mode 18
Using Post Edit Mode 19
Ignoring Violations inside Instances 23
Severity of DRD Markers 23
Selecting Constraints for Verification 25
Selecting Layers for Verification 27
Managing Interactive Display Options 29
Performing Color Checks 34
Color Constraints Checks 34
Color-Related Flow Checks 36
DRD Editing and Blockage Objects 42
Using DRD Editing in VLS XL and VLS EXL 42
Batch-Check Mode 43
Verifying Selected Objects 43
Verifying Nets 44

2
DRD Advanced Features 47
DRD Targets 48
Setting DRD Targets 48
Unsetting DRD Targets 55
Bindkey Reassignment 57
Targeted Enforce 58

Relaxed Enforce . . . 59
Deferred Post Edit . . . 60
What-if Halos . . . 60
What-if Functionality for Hierarchical Objects . . . 62
DRD Smart Snapping . . . 64
Snapping to Halos and Discrete Spacing Values . . . 64
Snapping to a What-if Halo around a DRD Target . . . 67
Simultaneous Snapping between Two Orthogonal Edges . . . 69
Layer-to-Layer Snapping . . . 70
Snapping Based on the minArea Constraint . . . 71
Snapping Using the Quick Align Form in Virtuoso . . . 72
Snapping with DRD Enforce Enabled . . . 74
Toggling DRD Smart Snap during Interactive Editing . . . 75
Incremental Violation Display . . . 75
Excluding Instances from Checking . . . 76

3
DRD Support for FinFET Devices . . . 77
Defining Discrete Spacing Rules . . . 77
Defining Minimum Spacing Rules . . . 80
Defining Directional and Discrete Width Rules . . . 82
Defining Directional and Discrete PR Boundary Rules . . . 85
Defining Directional and Discrete PR Boundary Width Rules . . . 85
Defining Directional and Discrete PR Boundary Spacing Rules . . . 87
Snapping Objects to Legal Width Values . . . 90
Examples of Discrete Dimension Gravity . . . 91
Defining Directional and Discrete Opposite Extension Rules . . . 96

4
DRD Constraints Support . . . 99
Supported Constraints . . . 100
Supported Constraints and Parameters . . . 101
Supported Constraints and Parameters (Virtuoso Layout Suite EXL) . . . 112
Supported Constraints and Parameters (Virtuoso Advanced Node for Layout Standard) 115

Supported Constraints and Parameters (Virtuoso Advanced Node for Layout Only) 122
Using DRD Grid Checking . . . 129
Specifying Layers in the Interconnect Section . . . 130
Supported Derived Layers . . . 130
Supported Layer-Purpose Pair and Voltage-Dependent Constraints . . . 133
Support for Layer-Purpose Pair Constraints . . . 133
Support for Voltage-Dependent Constraints . . . 134
Using Density Constraints . . . 137
Using Alternate Foundry Constraint Group . . . 140

A
DRD Form Descriptions . . . 141
DRC Options Form . . . 142
Interactive . . . 142
Batch . . . 146
Filters . . . 149
Sign-Off Settings . . . 151
Signoff Fill Form . . . 152

B
DRD Environment Variables . . . 155
List of Environment Variables . . . 155
drdBatchColorability . . . 158
drdBatchColorShorts . . . 159
drdBatchConsiderGrayShapes . . . 160
drdBatchConsiderUnlockedShapesAsGray . . . 161
drdBatchFoundryRulesOnly . . . 162
drdBatchReportCoveredNonPcellGrayShapes . . . 163
drdBatchScopeAreaLimit . . . 164
drdBatchScopeSpecifiedArea . . . 165
drdBatchUncoloredShapes . . . 166
drdBatchUnlocked . . . 167
drdBatchVioLimit . . . 168
drdColorability . . . 169
drdColorShorts . . . 170

drdConsiderGrayShapes . . . 171
drdConsiderUnlockedShapesAsGray . . . 172
drdEditAllowOrthogonalAdjust . . . 173
drdEditAutoText . . . 174
drdEditAutoTurnOffHalo . . . 175
drdEditAutoTurnOffHaloLimit . . . 176
drdEditBatchColorRules . . . 177
drdEditBatchHierDepth . . . 178
drdEditColorRules . . . 179
drdEditDensityTypeCheck . . . 180
drdEditDisplayArrows . . . 181
drdEditDisplayDashed . . . 182
drdEditDisplayDrawingColor . . . 183
drdEditDisplayEdges . . . 184
drdEditDisplayHalo . . . 185
drdEditDisplayMarkers . . . 186
drdEditDisplayMaxChars . . . 187
drdEditDisplayTechRefs . . . 188
drdEditDisplayText . . . 189
drdEditDisplayTrueColor . . . 190
drdEditEnableBatchCheckOnReadOnlyDesign . . . 191
drdEditFontSize . . . 192
drdEditHierDepth . . . 193
drdEditIgnoreIntraInstanceChecks . . . 194
drdEditMode . . . 195
drdEditPaletteLayers . . . 196
drdEditSlidingWindowSize . . . 197
drdEditSmartSnapAllowedWidthSnap . . . 198
drdEditSmartSnapAperture . . . 199
drdEditVioLimit . . . 200
drdInteractiveFoundryRulesOnly . . . 201
drdMobilityMode . . . 202
drdPostEditAllWindows . . . 203
drdPostEditEIPAllInstances . . . 204
drdPostEditEIPAllWindows . . . 205
drdPostEditEIPCheckTopView . . . 206

drdPostEditIgnoreModgen 207
drdPostEditReplaceAllMarkers 208
drdPostEditTimeOut 209
drdPostEditVioLimit 210
drdUncoloredShapes 211
drdUnlocked 212
drdUseBlockageForExtCheck 213
drdUseNetName 214
drdUseNewDerivedLayerEngine 215
drdValidLayersGrayCheck 216
drdVDRIncludePurposes 217
threads 219

C
Blockage and Boundary Objects 221
DRD Editing and Blockage Objects 221
Effective Width 221
DRD Editing and PR Boundary Objects 224
Creating and Editing Non-PR Boundary Objects 225

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 DRD Constraints Support.

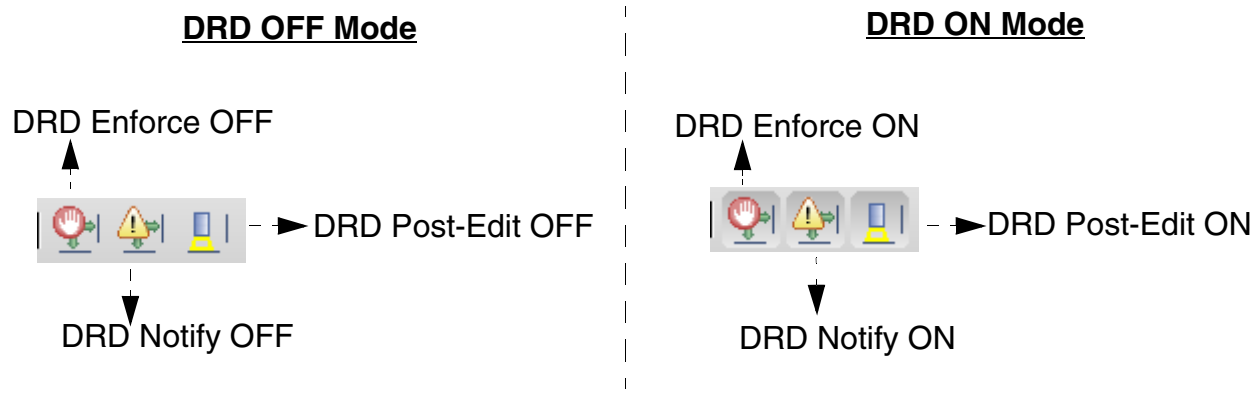
This chapter contains the following topics:

- Turning on DRD Editing
- DRD Operating Modes
- Selecting Constraints for Verification
- Selecting Layers for Verification
- Managing Interactive Display Options
- Performing Color Checks
- DRD Editing and Blockage Objects
- Using DRD Editing in VLS XL 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 DRC Options Form 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.

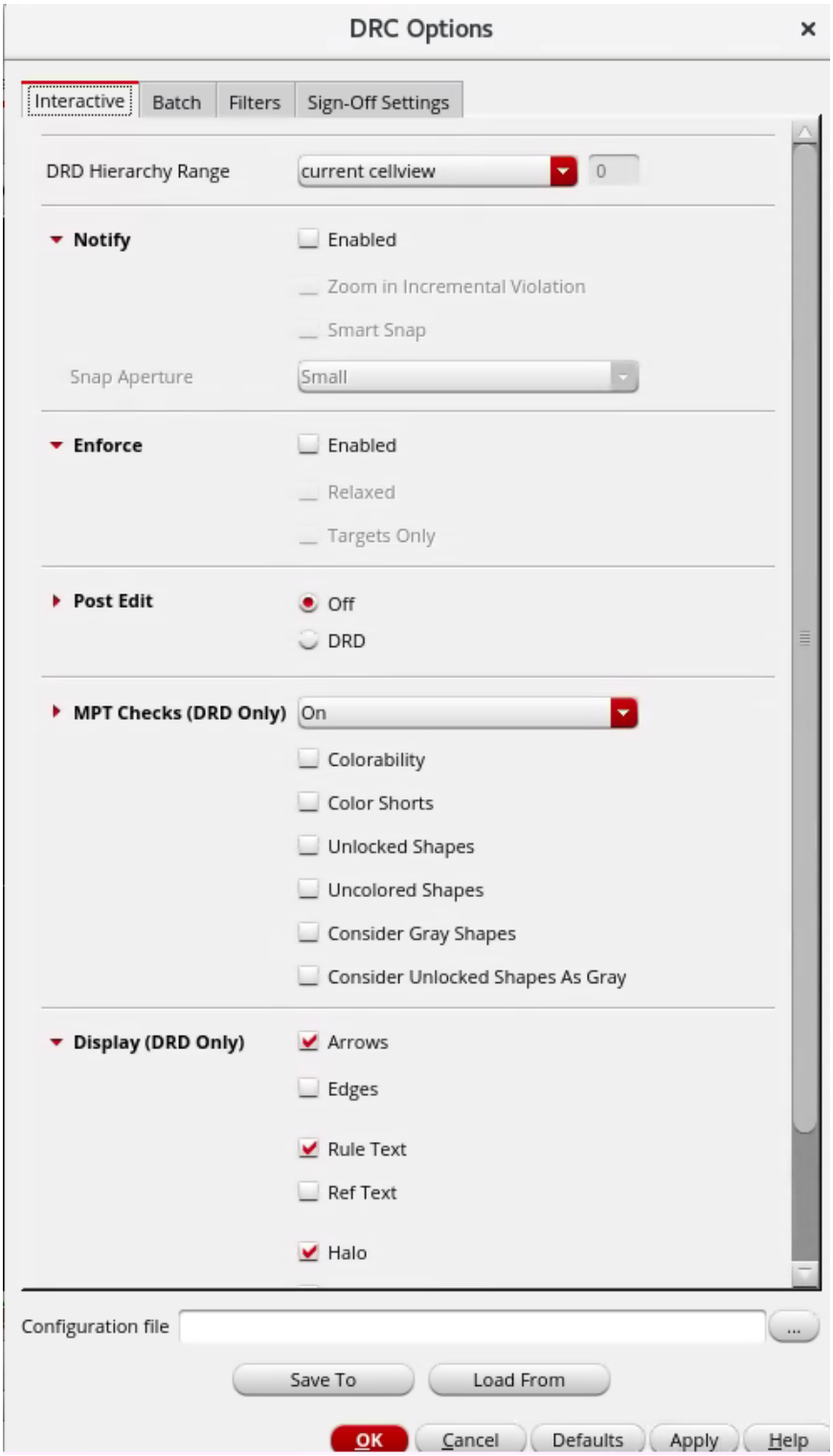


Setting Up DRC Options

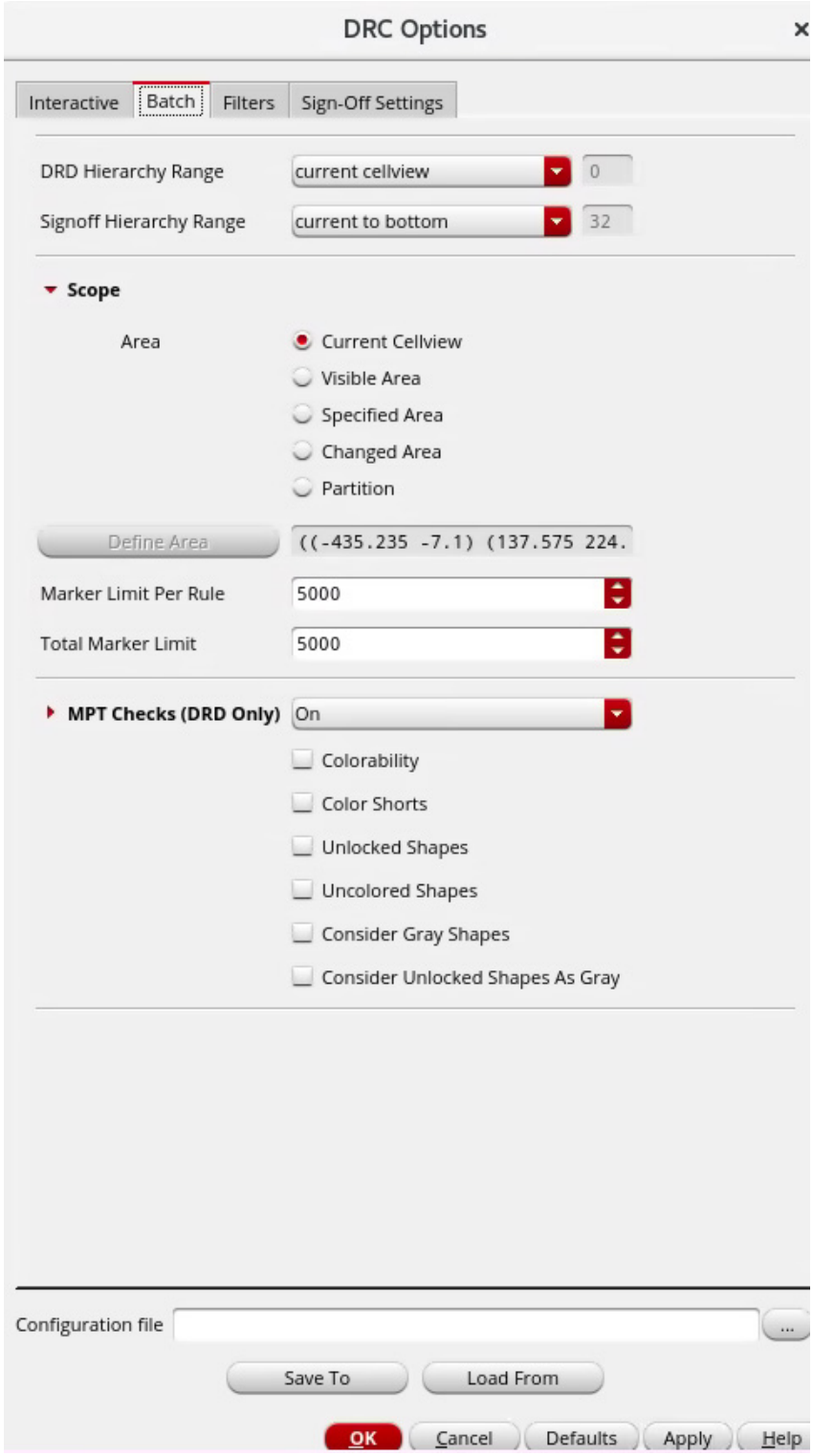
The DRC Options form allows you to set up the options that are checked by DRD during layout editing. To open the DRC Options form, do one of the following.

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

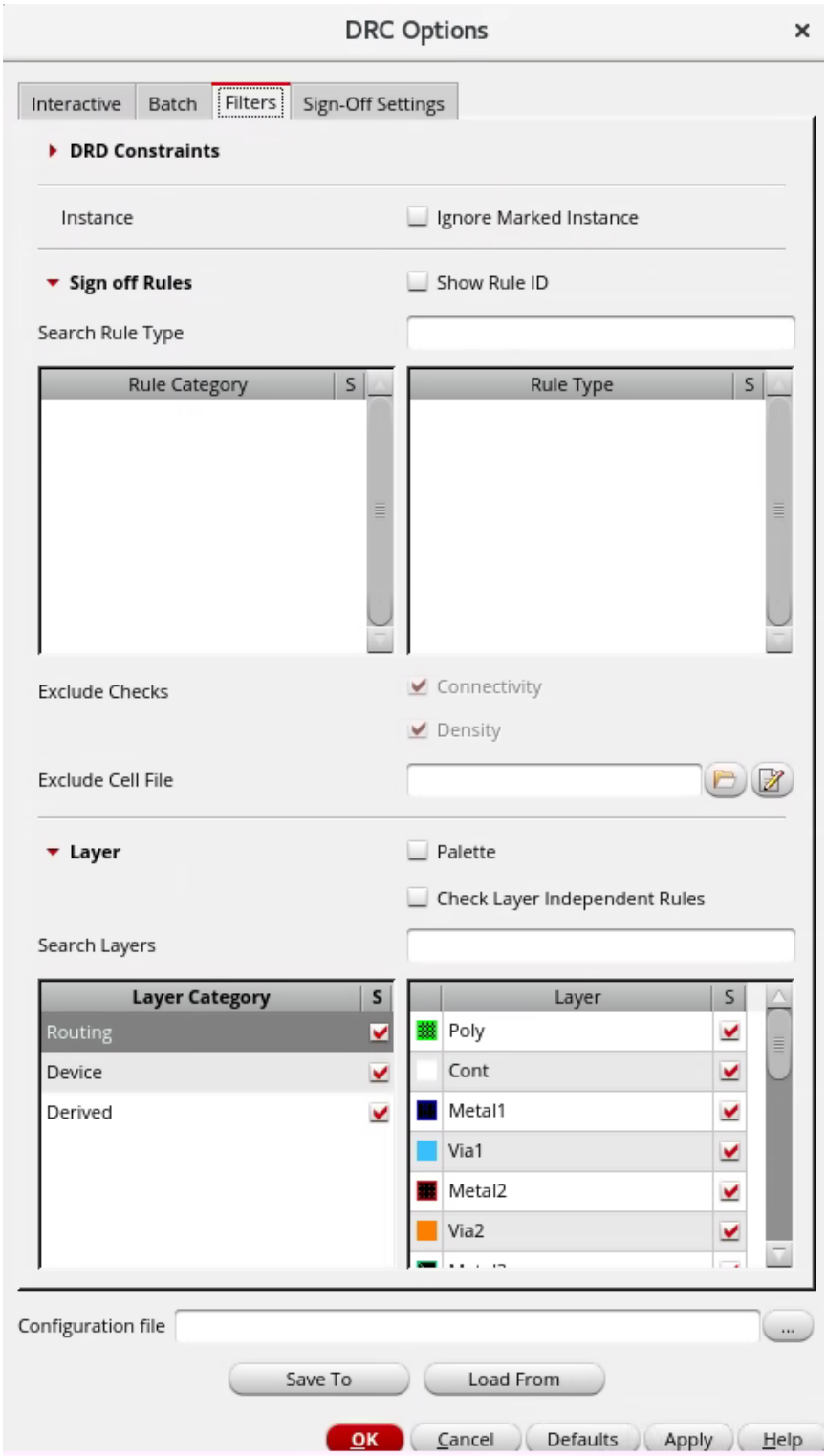
The DRC Options Form form is displayed. The form consists of the following tabs:
Interactive



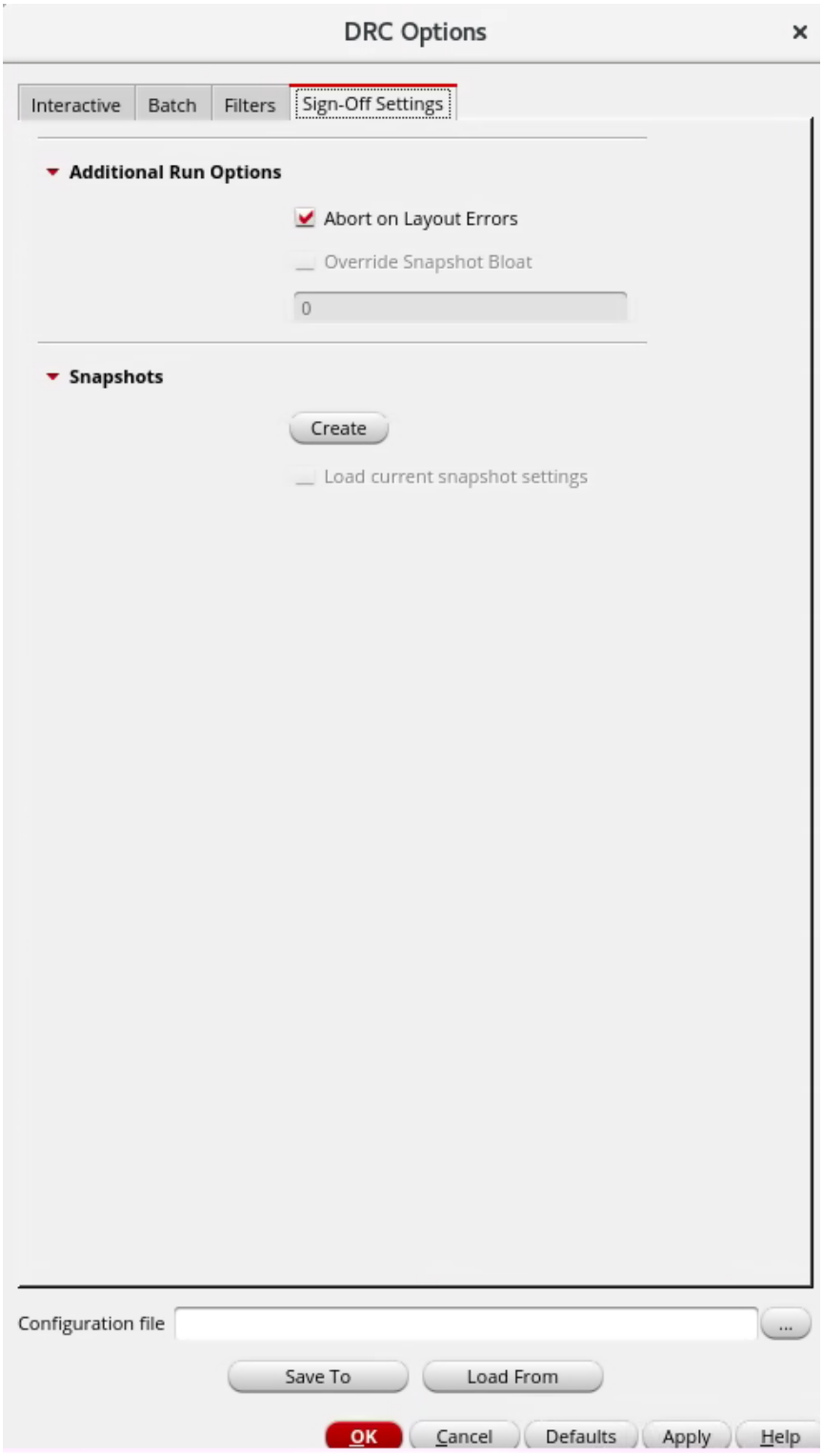
Batch



Filters



Sign-Off Settings



DRD Configuration File

The DRC 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 DRC Options form.

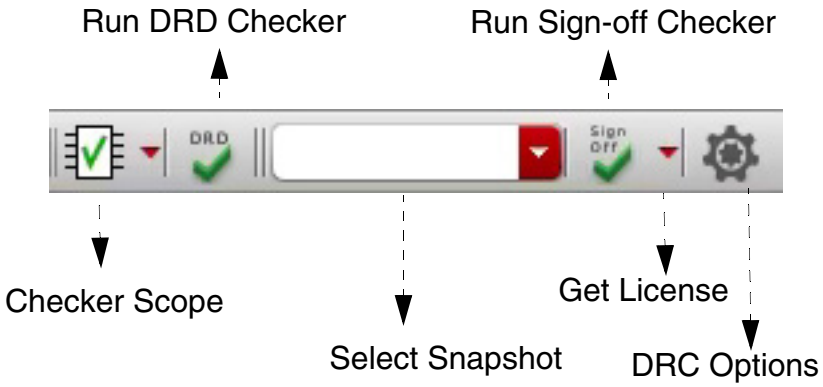
To specify a configuration file, type its name and path in the *Configuration file* field or use the browse button to locate and select an existing file. Ensure that you have write permissions to the path and file.

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

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

Specifying Options on the DRC Toolbar

You can access the DRC options using the icons available on the *DRC* toolbar in Virtuoso.



You can use the *DRC* toolbar to select the scope of DRC checking, to run the DRD checker, to select a snapshot to run the sign-off DRC, to run sign-off checker, and to open the DRC Options form to update the settings.

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.

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

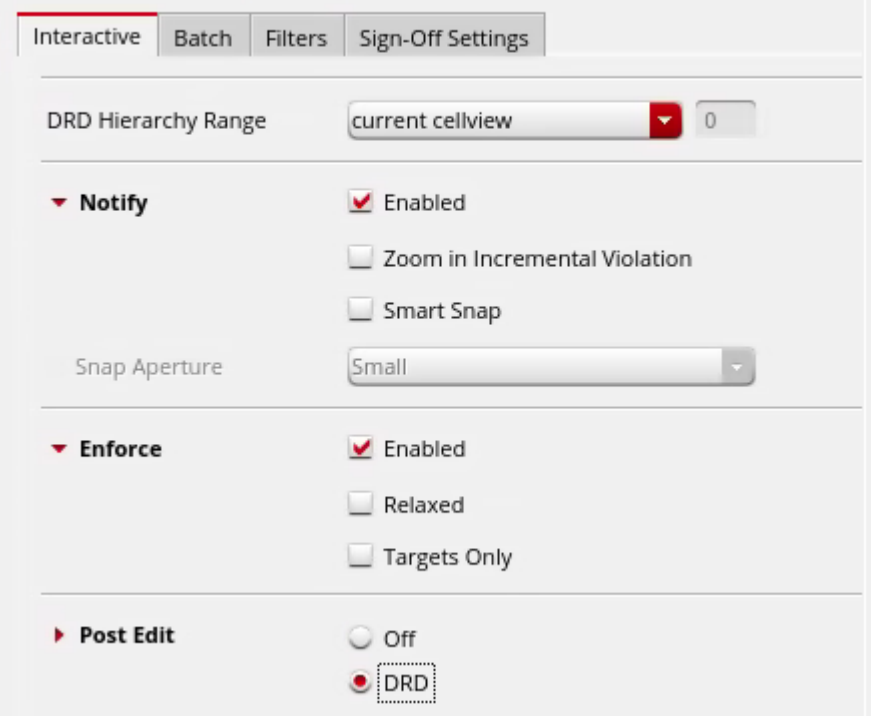
You can use DRD in the following combinations:

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

- *Enforce* and *Post Edit*

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

- The DRC Options Form form – *Interactive* tab



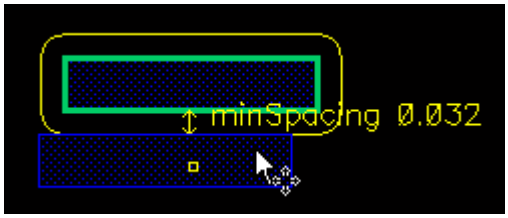
- The `Ctrl+e` bindkey
- The `leToggleDrdMode` SKILL function

  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.

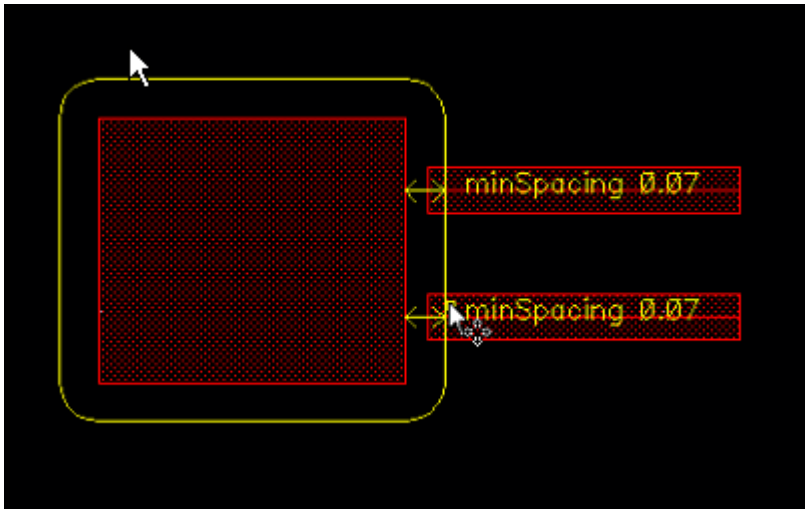


For information about the constraints supported in *Enforce* mode, see DRD Constraints Support.

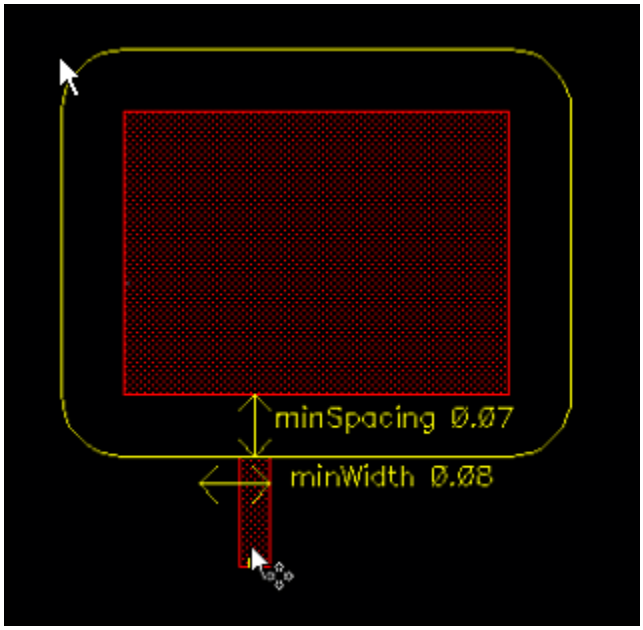
Using Notify Mode

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. See,Using Post Edit Mode.

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



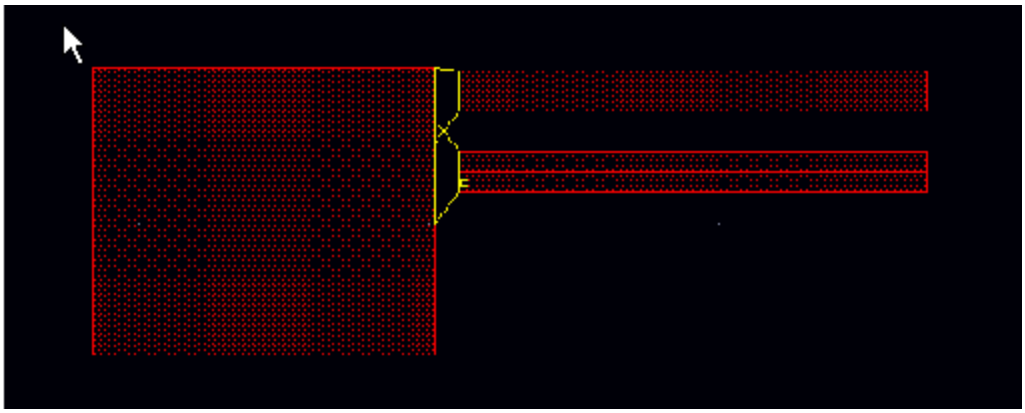
You can enable *Enforce* and *Notify* modes together to use their combined behavior. For example, in the figure below, spacing constraints are enforced and the other constraints are notified.



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.

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 `Metal1` layer, DRD verifies the change and updates the markers related to `Metal1`. 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. See, Annotation Browser in *Virtuoso Layout Suite XL User Guide*.

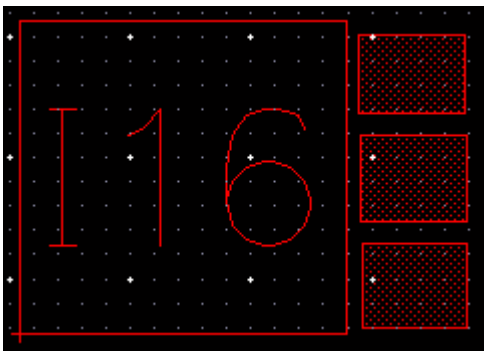
You can search for, explain, and delete markers by using the commands on the *Verify – Markers* menu. See Markers in *Virtuoso Layout Suite XL User Guide*.

Behavior of Post Edit Mode when Editing in Place

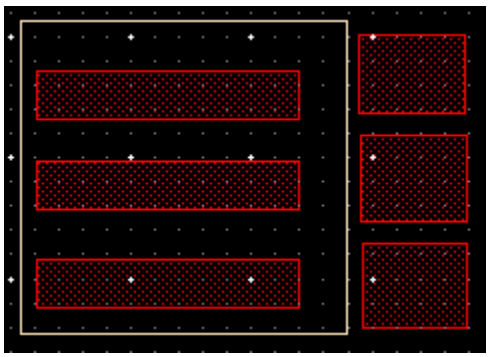
To display post edit markers when editing in place, ensure that you do the following:

1. In the DRC Options form:
   a. On the *Interactive* tab, choose *Hierarchy Range* as *current to bottom*.
   b. Select the *Post Edit – Enabled* check box.
2. In the Annotation Browser, select *Scope* as *Top Cellview Only* or T*op Cellview To Depth*.

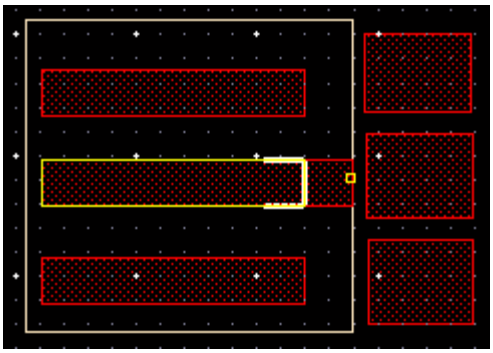
To understand how Post Edit mode works when editing in place, consider the following example in which an instance and three shapes are placed at the top level.



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



The cell as seen after the *Edit In Place* command is run.

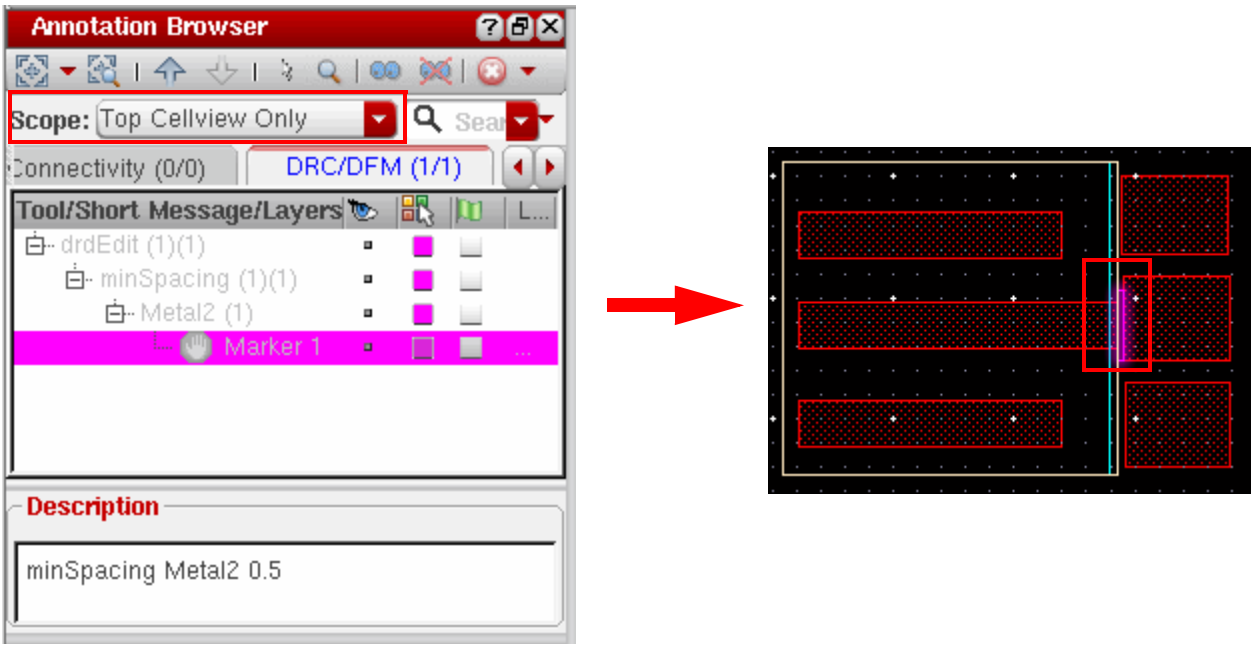


A shape in the cell being edited using the *Stretch* command.

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.

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.



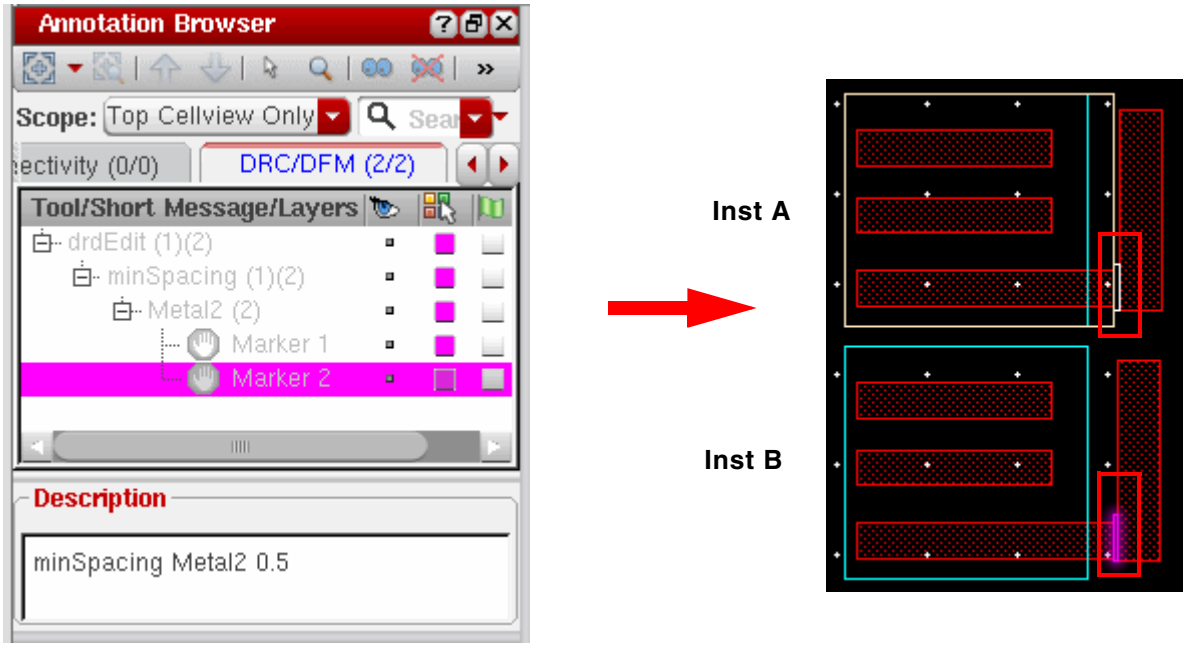
Related Environment Variables

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

- drdPostEditEIPCheckTopView: 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`.
- drdPostEditEIPAllInstances: 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`.
- drdPostEditEIPAllWindows: Applies the value set for `drdPostEditEIPCheckTopView` and `drdPostEditEIPAllInstances` to all open design windows. The default value is `nil`.

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 drdEditIgnoreIntraInstanceChecks environment variable is by default set to `t`.

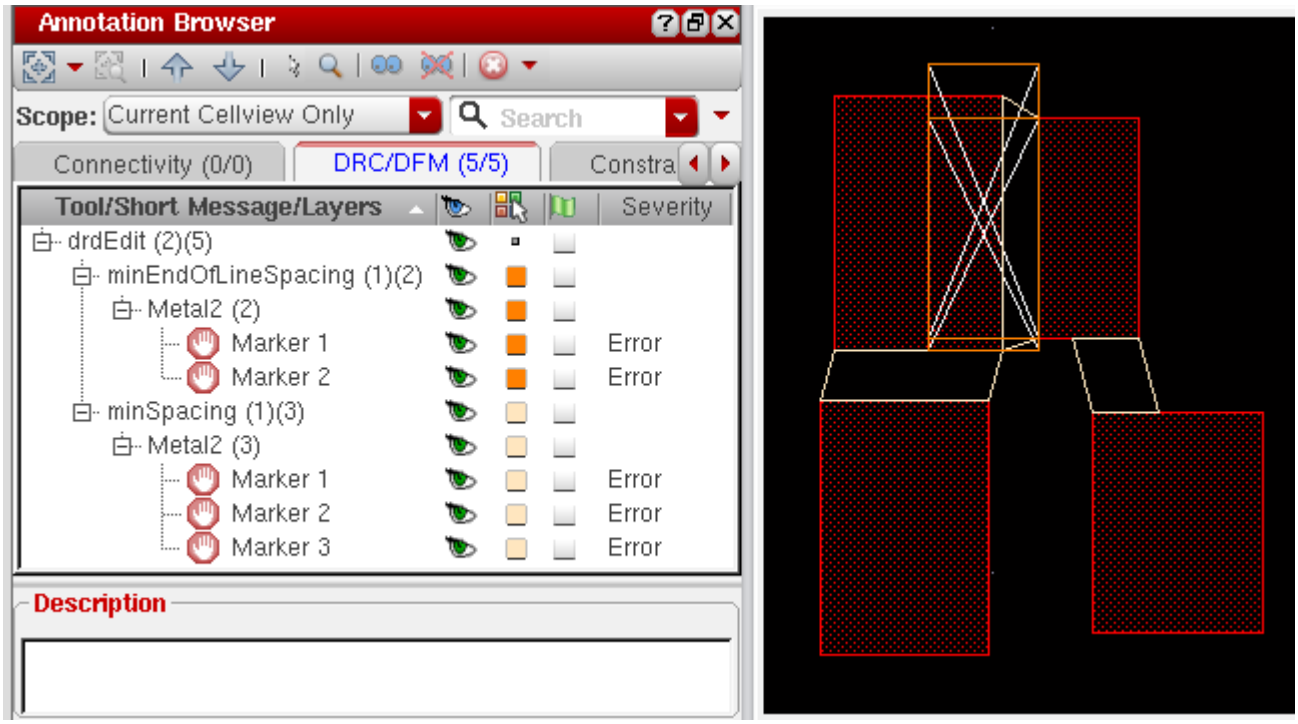
In Batch mode, when *Hierarchy Range* is set as *current to bottom*, *current to stop level*, or *current to user level*, DRD reports all violations that exist inside a hierarchical object, even when `drdEditIgnoreIntraInstanceChecks` is set to `t`.

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.

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*.



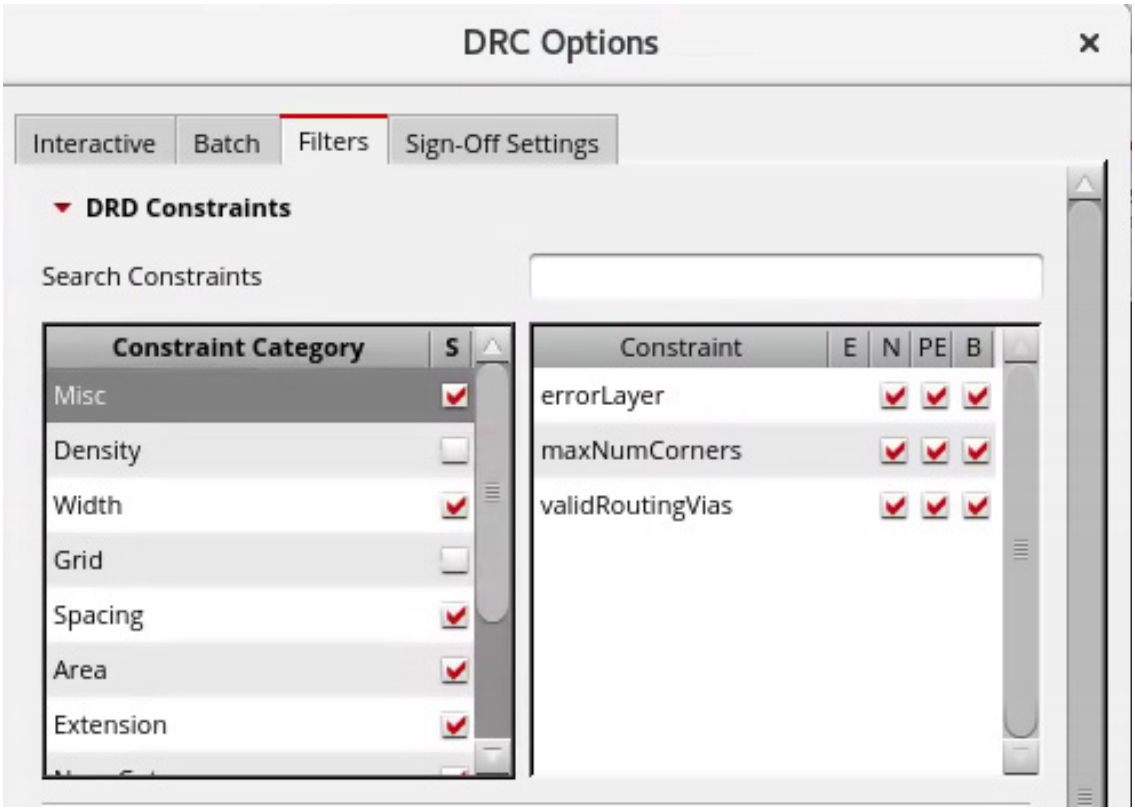
Selecting Constraints for Verification

You can use the *Filters* tab of the DRC 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.

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

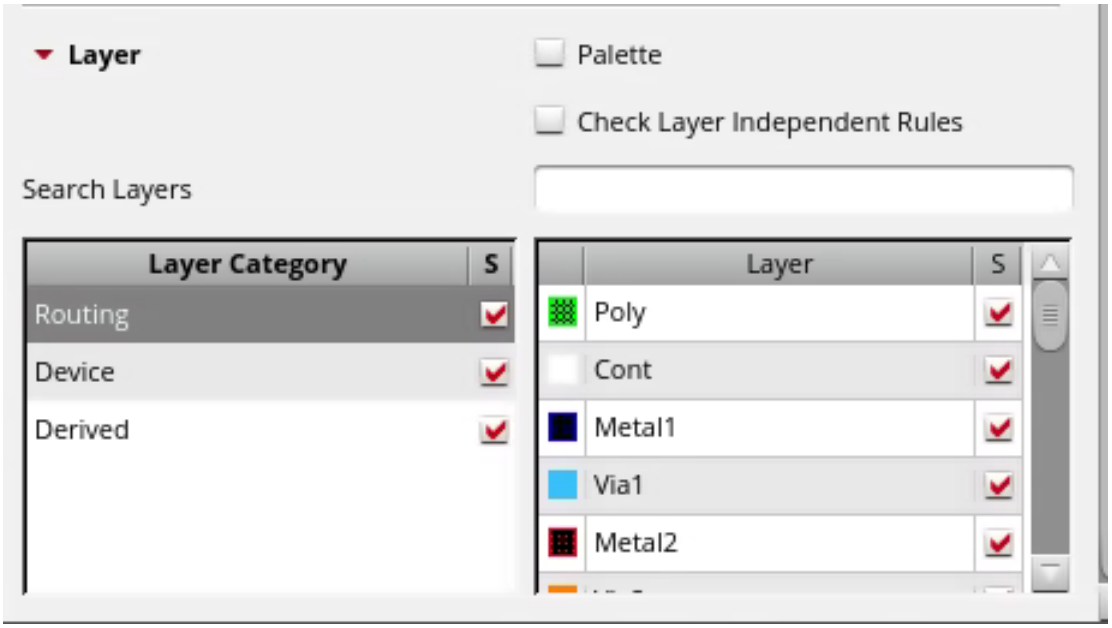
The Filters tab of the DRC Options form lets you choose the layers that you want to check for violations. When you filter the layers, only those violations that involve shapes on the selected layers are displayed. You can select the layers from the three layer categories provided, that is, *Routing*, *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.

  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).
- Device: Includes layers that are not Routing or Derived layers and Cut layers in which the below layer is not Metal.
- Derived: Includes derived layers in the technology file.
- 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.

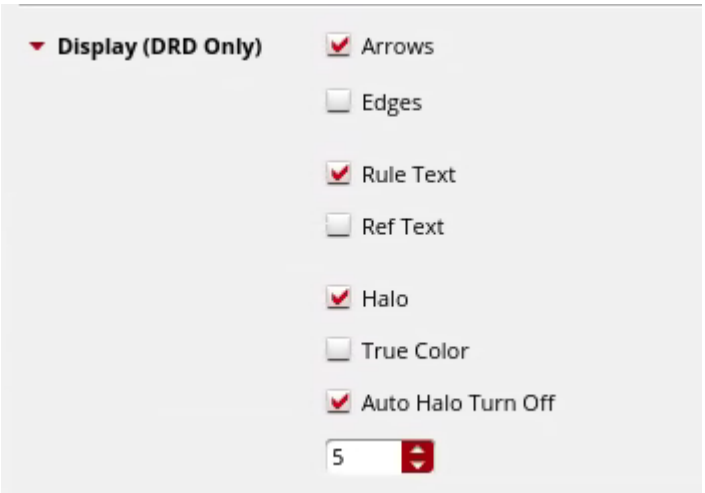
To enable or disable the checking of shapes and objects on a layer or layer category, select or deselect the check box against it.

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*.

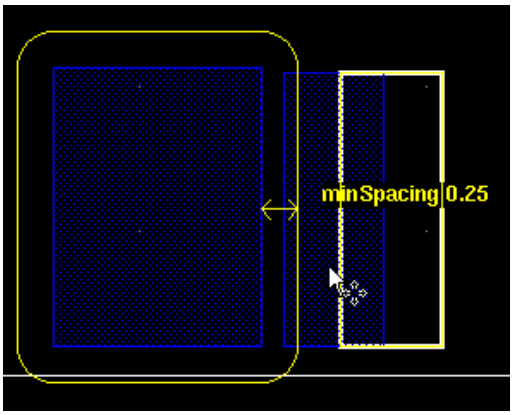
Managing Interactive Display Options

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

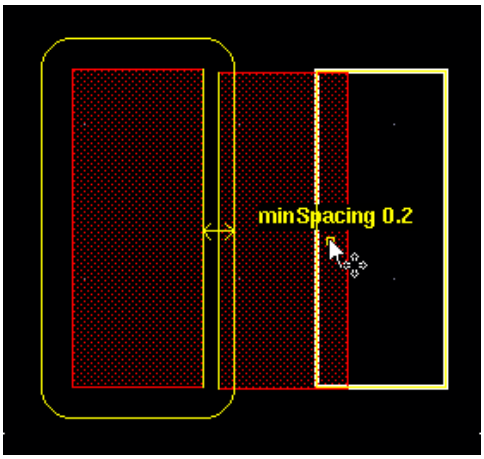


To enable an option, ensure that the corresponding check box is selected and specify the required value.

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



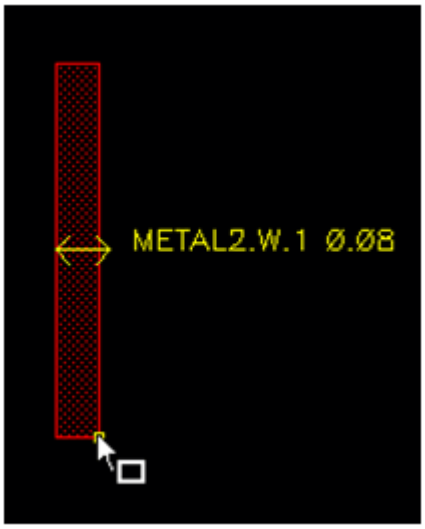
- *Edges* highlights the edge where the violation occurs.



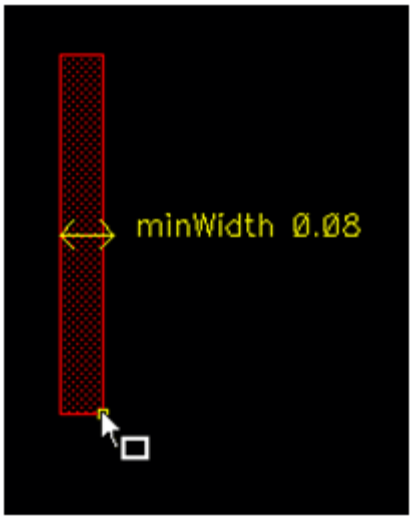
- *Rule Text* displays the name of the violated constraint at the location where the violation occurs. You can use the `displaySigDigits` environment variable to specify the number of decimal digits the constraint value must be displayed with in the rule text.

  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.

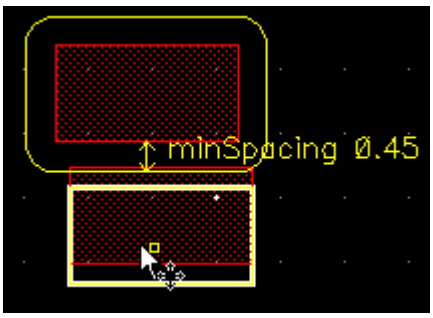


'ref Text from Techfile is on

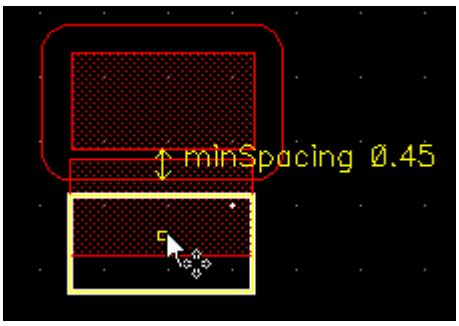


'ref Text from Techfile is off

- *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



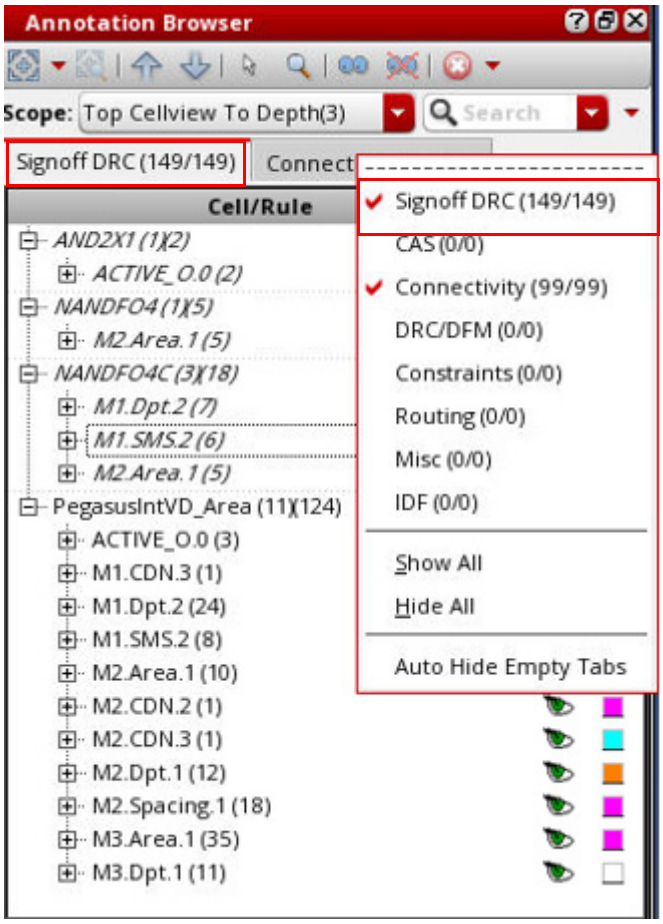
True color halo

- *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`.

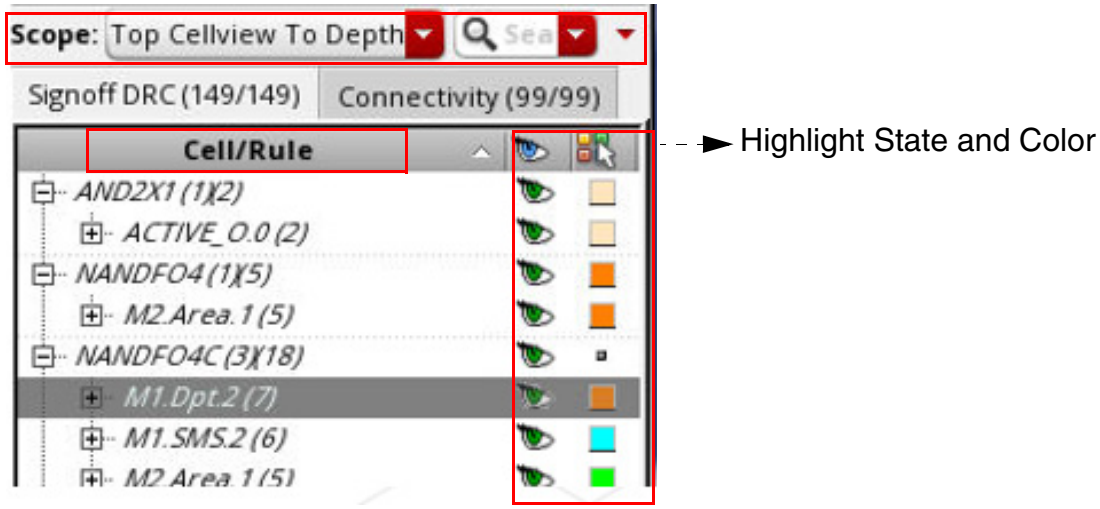
Viewing Sign-off Violations in the Annotation Browser

Sign-off violations are displayed in the Annotation Browser in the *Signoff DRC* tab of the Annotation Browser.



You can view the cells and their violations in the Annotation Browser. You can select the scope to display markers by selecting the options in the *Scope* field. You can also zoom to the

selected marker. In the browser pane, you can specify the highlight state and color used to draw markers in the design.

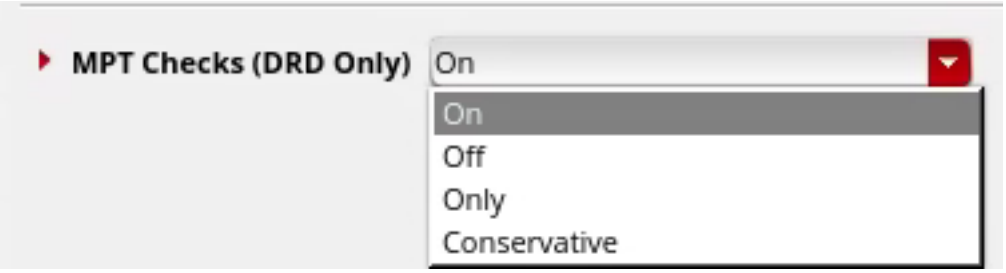


Performing Color Checks

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

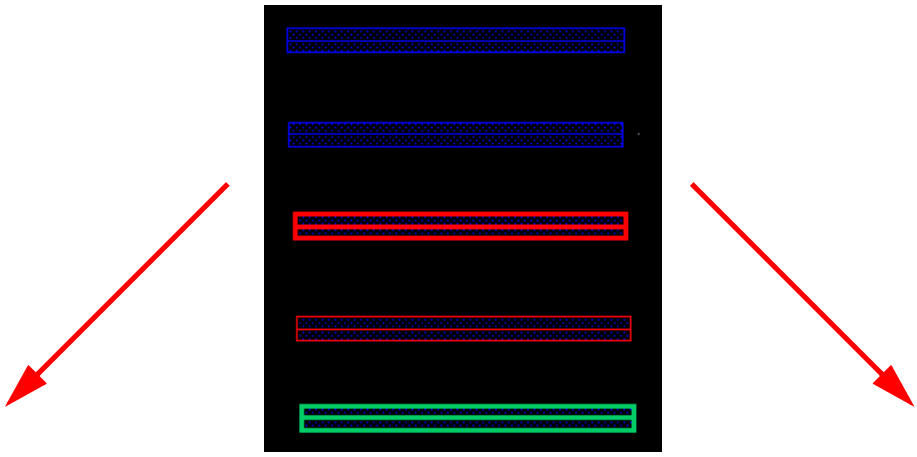
Color Constraints Checks

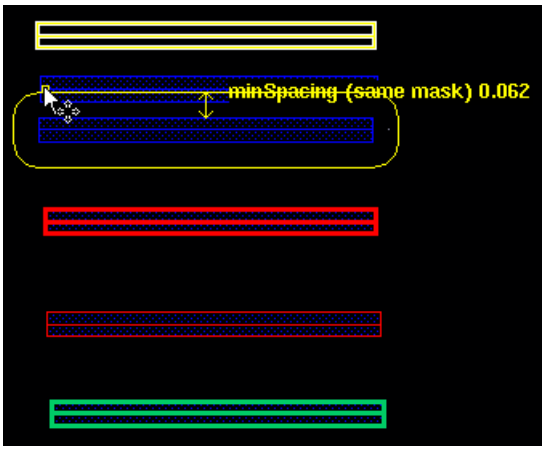


The *MPT Checks* drop-down menu lets you perform the checks for color constraints in different DRD modes. This drop-down list provides the following four options:

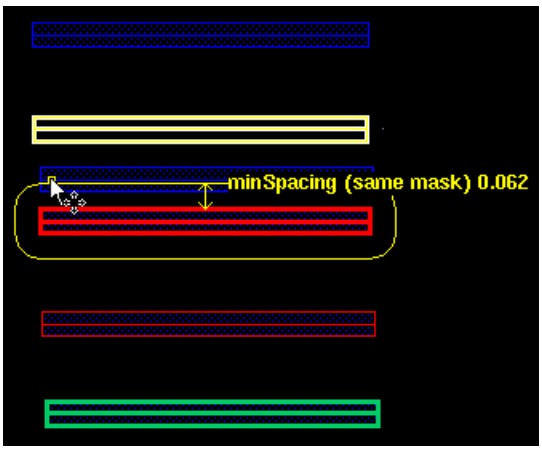
- *On*: (Default) DRD reports violations for all color-related constraints, including non-color constraints.
- *Off*: DRD reports violations for all available constraints except those related to color constraints. For example, DRD in this mode reports all `minSpacing` violations except `sameMask` violations.
- *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 mask2-mask2
  - 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-mask*N*, where *N* = 1, 2, 3, …

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

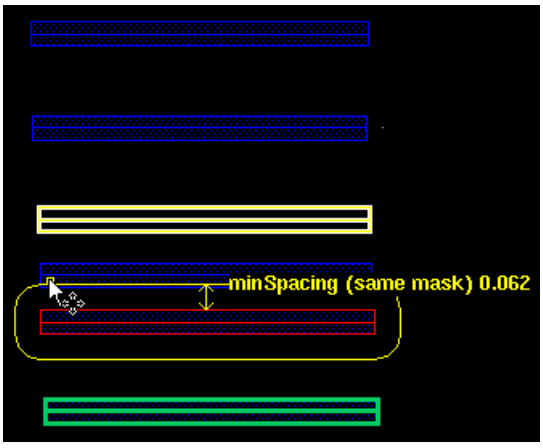




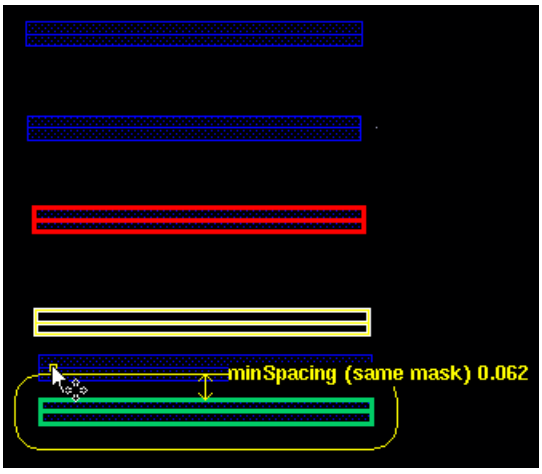
gray-gray



gray-mask1

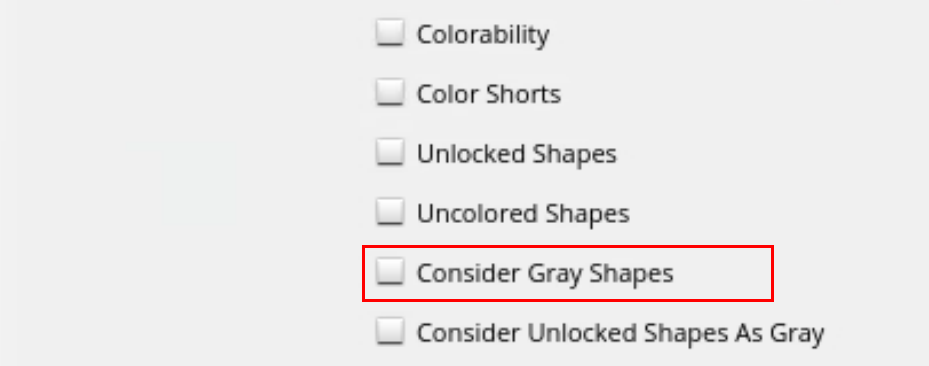


mask1-mask1

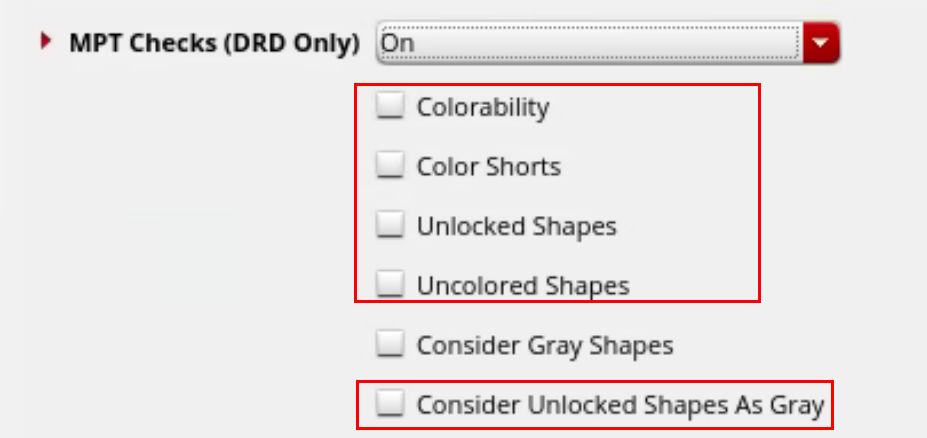


mask1-mask2

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



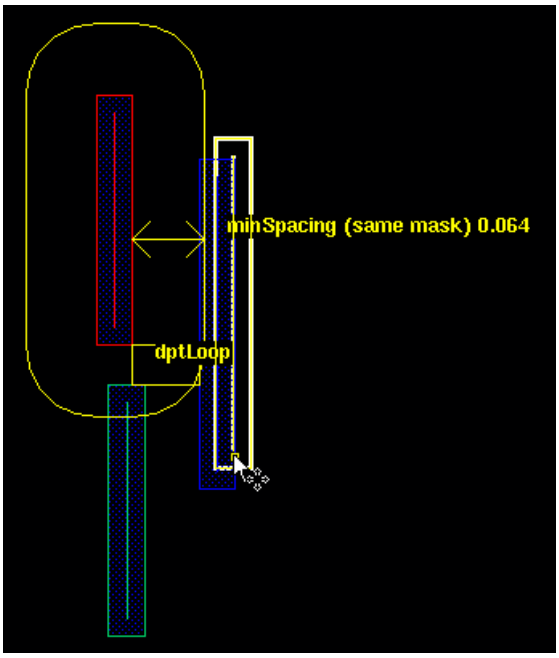
The color-related flow checks are described below. These checks are independent of the *MPT Checks* drop-down menu settings.

- *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. See `multiMaskCheck` in *Virtuoso Technology Data Constraints Reference*.

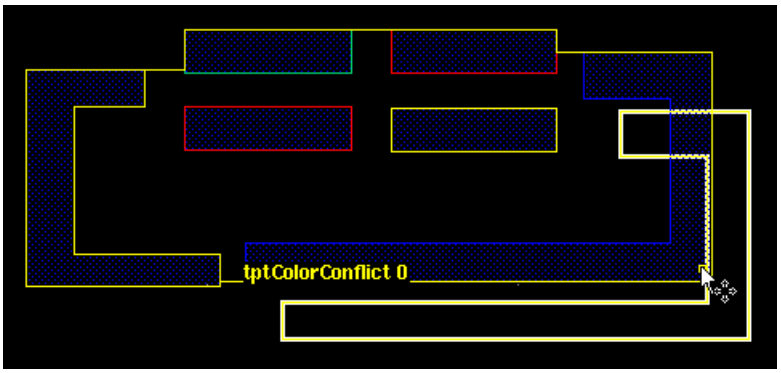
  Note: Colorability checks are available only in DRD Notify, Post Edit, and Batch modes,

and they can be performed only when the *Spacing* and *Complex Spacing* constraint groups are selected on the *Filters* tab of the DRC Options Form form.

The figure below shows a double patterning loop violation:



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.

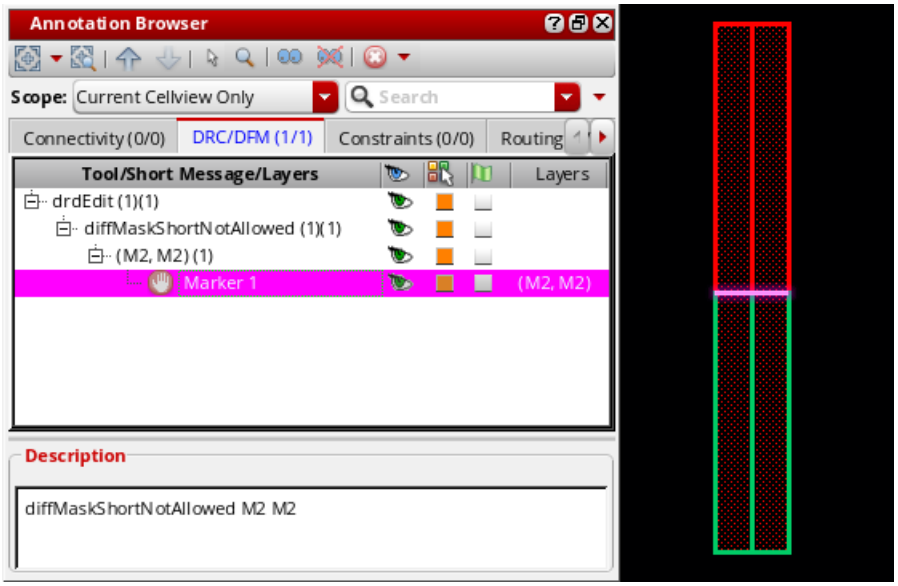
- *Uncolored Shapes*: Reports uncolored (gray) shapes on layers that support multiple masks.
- *Consider Unlocked Shapes as Gray*: Considers unlocked shapes as gray.

Using the Annotation Browser to View Color Check Violations

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

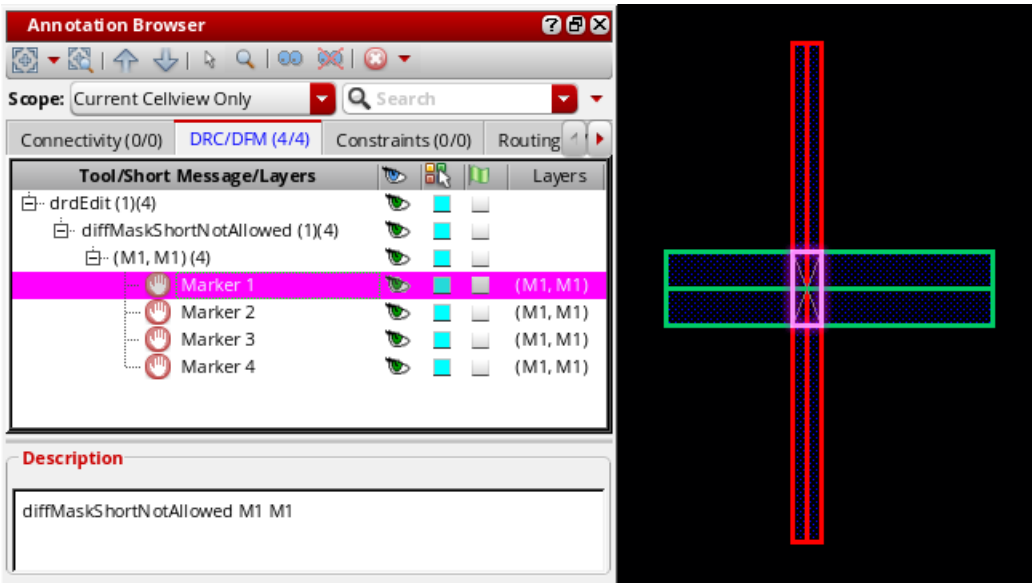
Viewing Color Shorts

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



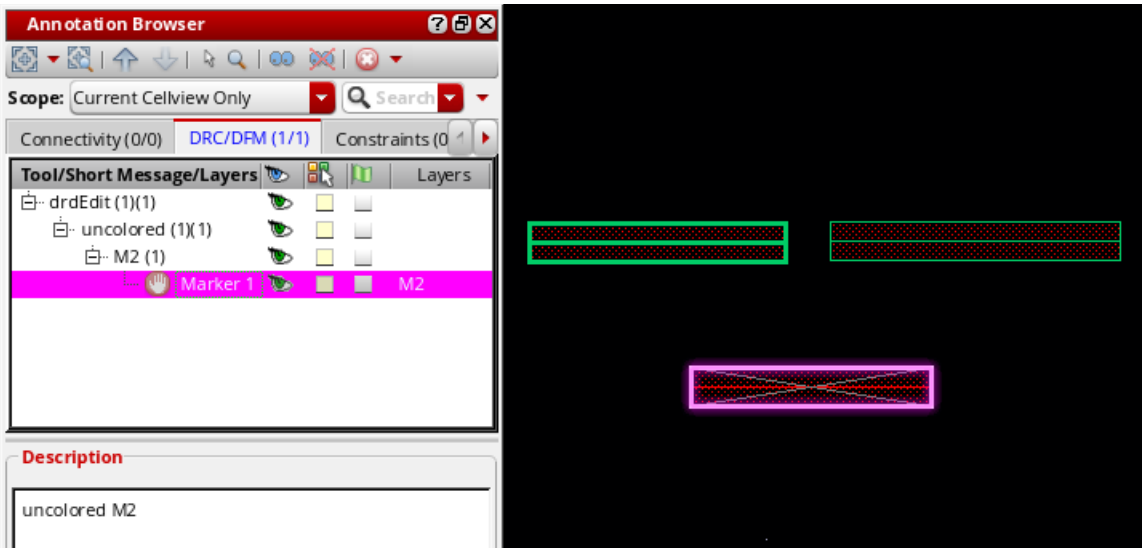
Two abutting shapes on two different masks

Two overlapping shapes on two different masks



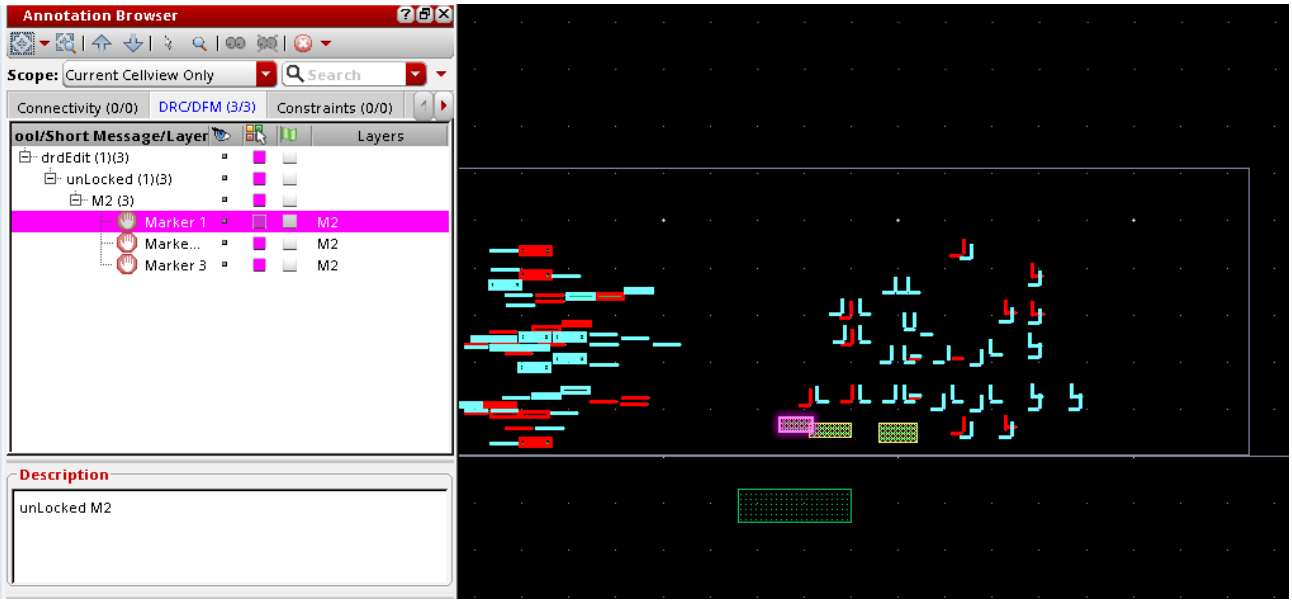
Viewing Uncolored Shapes

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*.



See *Virtuoso Multi-Patterning Technology User Guide*.

Related Environment Variables

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

- drdEditColorRules
- drdEditBatchColorRules
- drdColorability
- drdBatchColorability
- drdColorShorts
- drdBatchColorShorts
- drdUnlocked
- drdBatchUnlocked
- drdUncoloredShapes
- drdBatchUncoloredShapes
- drdConsiderGrayShapes
- drdBatchConsiderGrayShapes
- drdConsiderUnlockedShapesAsGray
- drdBatchConsiderUnlockedShapesAsGray

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 and VLS EXL

The VLS EXL 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*.

See Connectivity Extraction and Extract Layout 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.

See Using Assisted Routing Commands 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 EXL modes.

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

- Select the area of check from the drop-down list. The three area options are *Current Cellview*, *Visible Area*, *Specified Area*, *Changed Area*, and *Partition*.

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

  The DRC Options Form form is displayed. Use the Batch to set up the various Batch mode options.

For information about running DRD in Batch mode on read-only cellviews, see drdBatchCheck and drdEditEnableBatchCheckOnReadOnlyDesign.

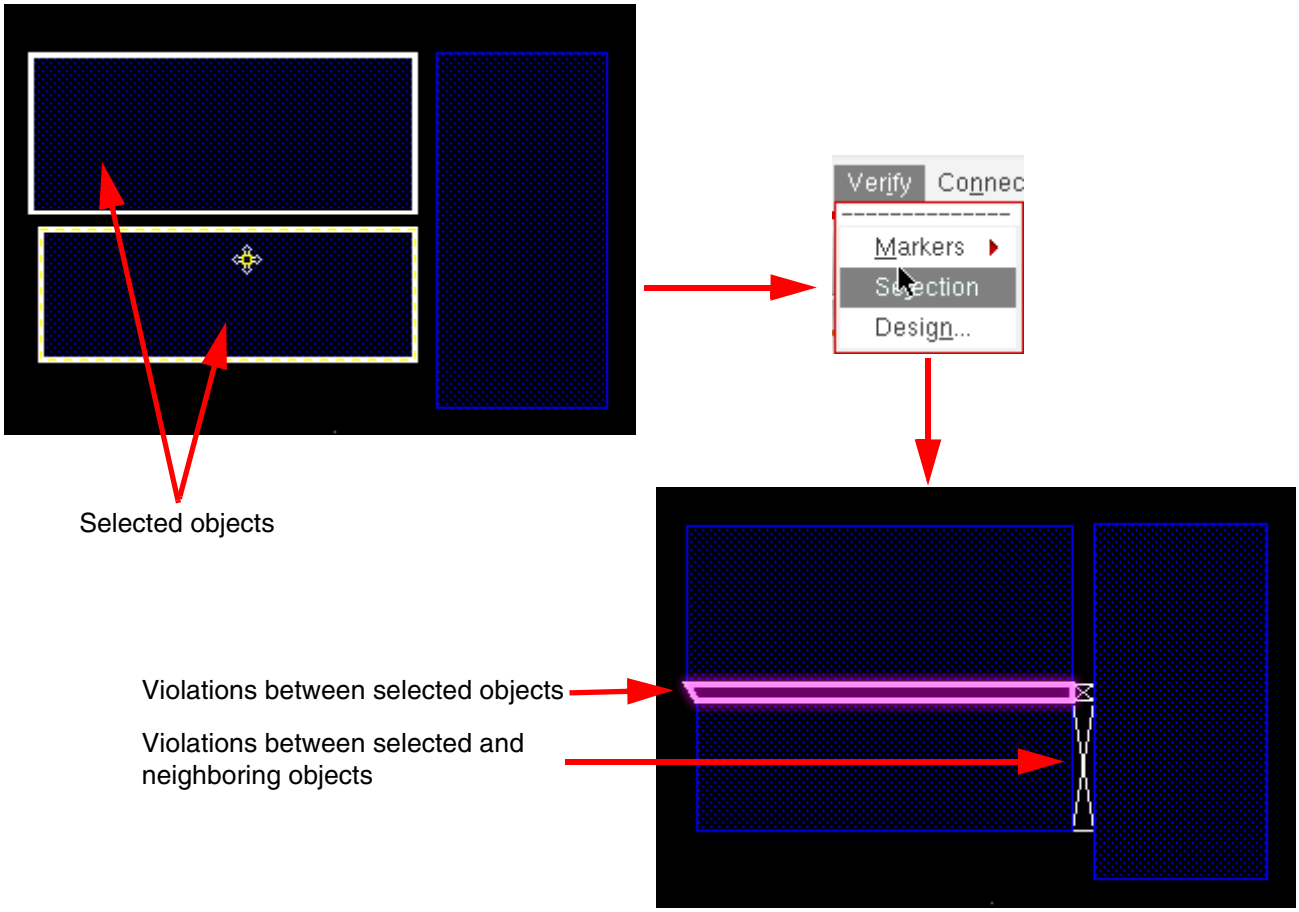
Verifying Selected Objects

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.

Note: This capability is available only in VLS EXL 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 `drdVerifySelSet` SKILL function.

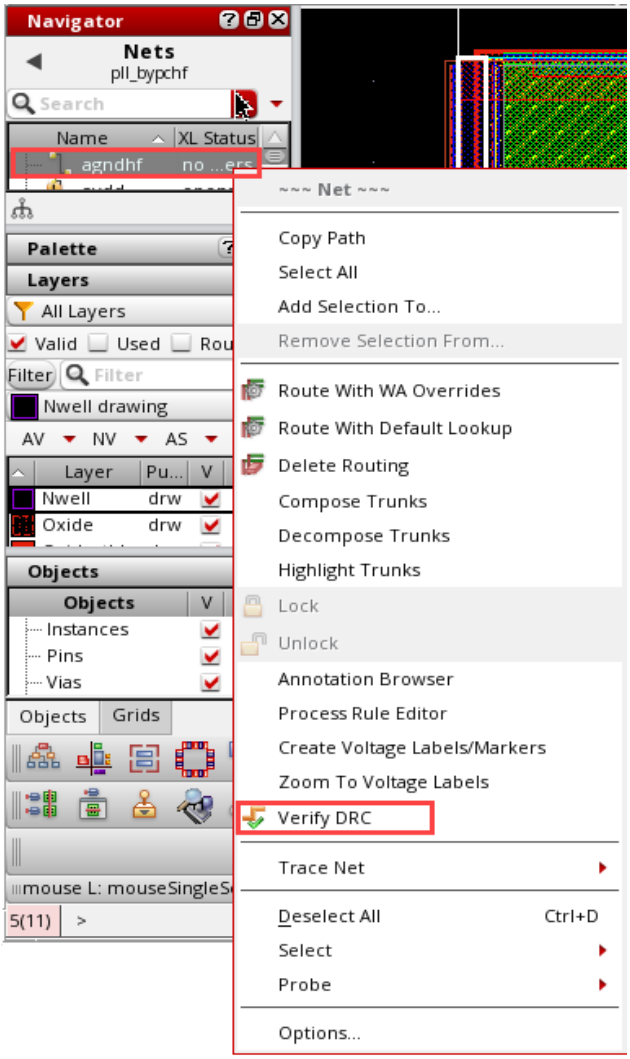
Verifying Nets

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 DRC Options form.

The *Verify DRC* command is independent of the *Scope* options specified in the DRC 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.

Video

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

2

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.

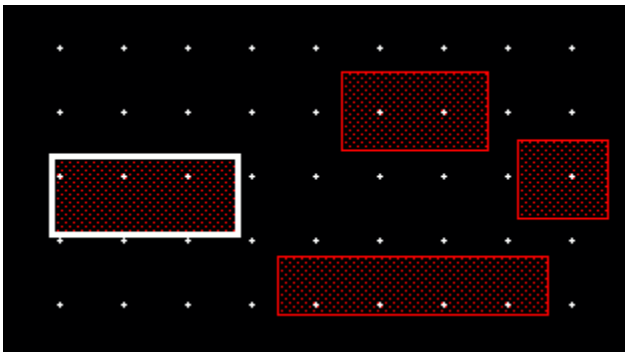
- Setting DRD Targets
- Unsetting DRD Targets

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.

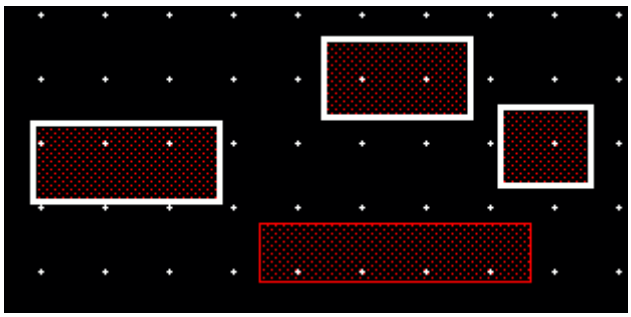
Setting DRD Targets by Selection

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

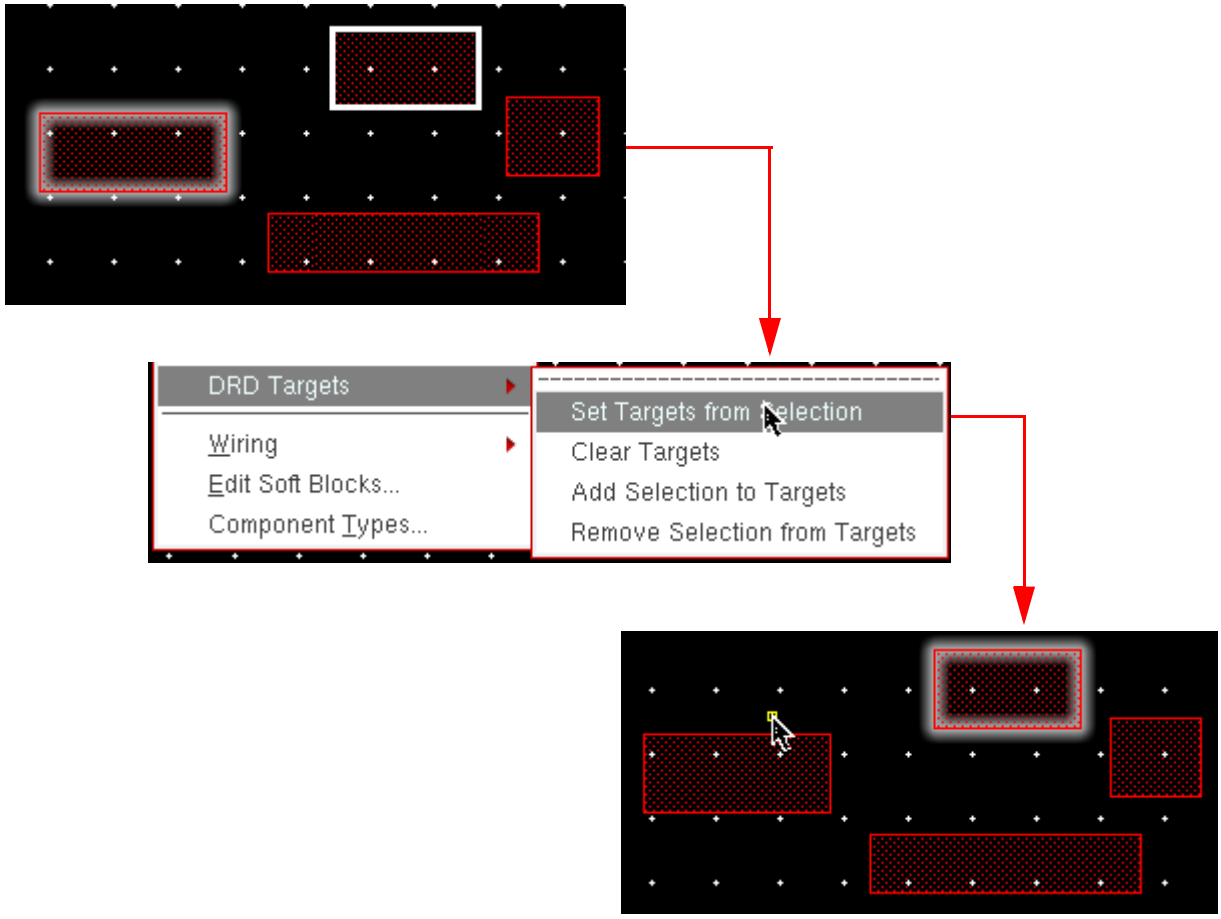


After you have selected the objects, you can either set the selected objects as new DRD targets or add them to an existing set of DRD targets.

- 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*.

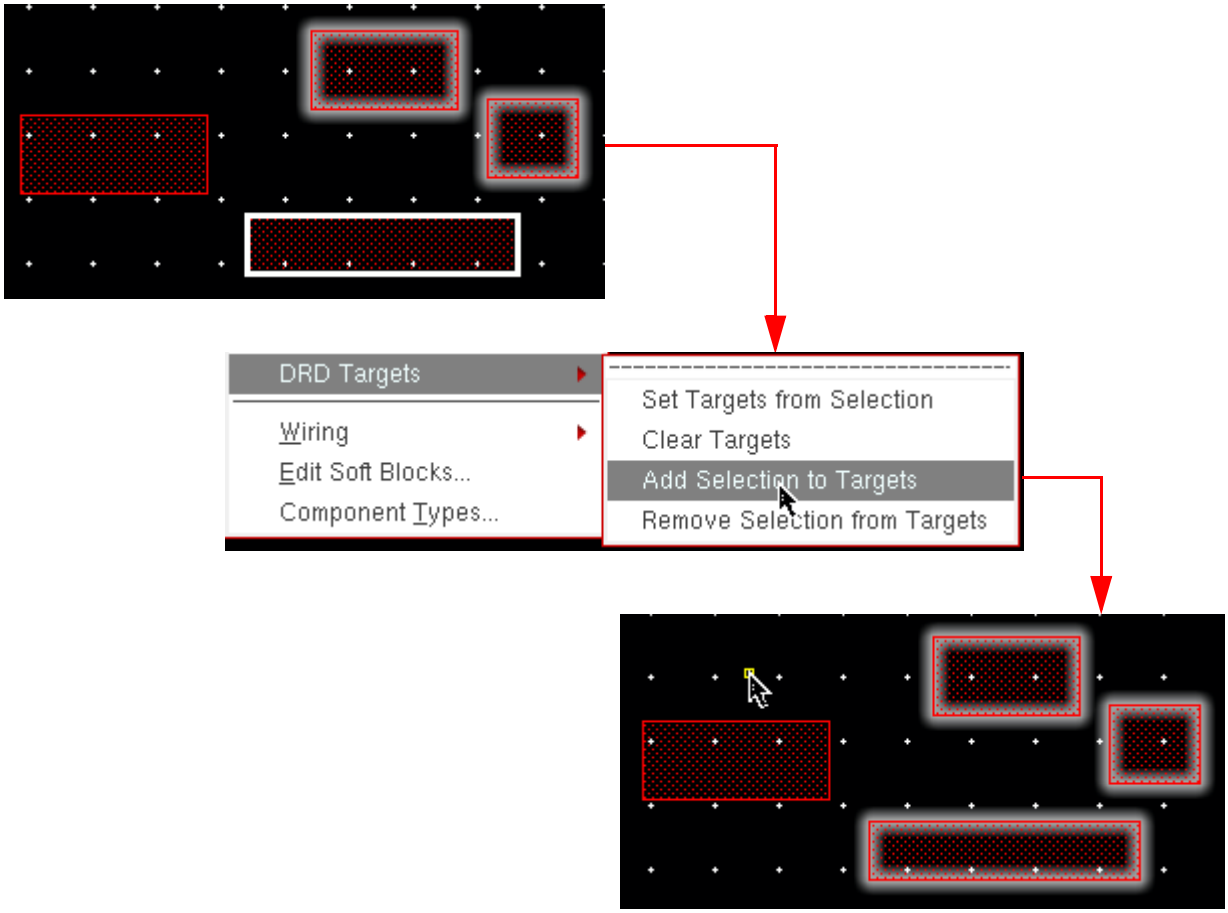
  The selected objects are set as DRD targets, as shown:



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

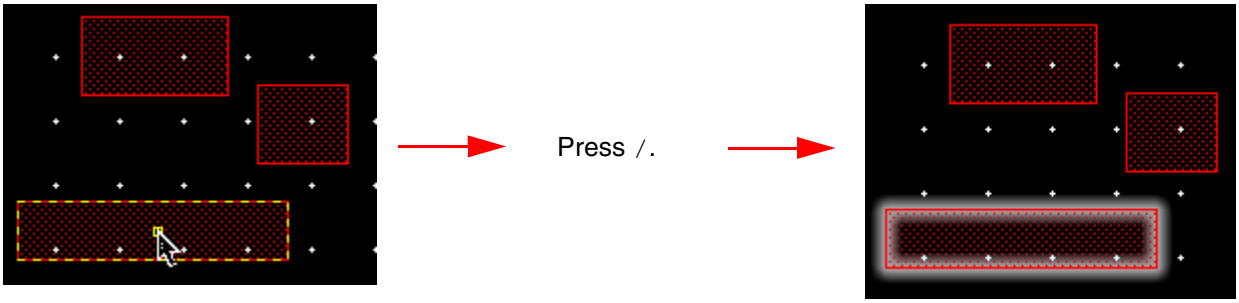
- Right-click the selected objects and choose *DRD Targets – Add Selection to Targets* from the context menu if you want to add the selected objects to an existing set of DRD targets.

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

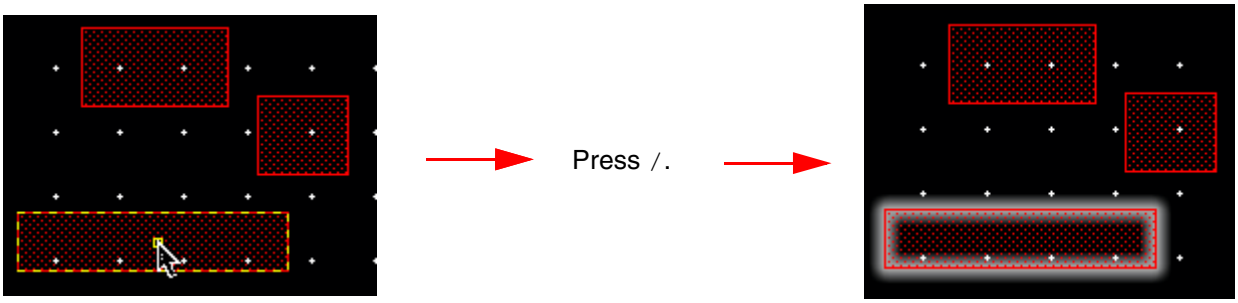


Setting DRD Targets Using the Bindkey

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.

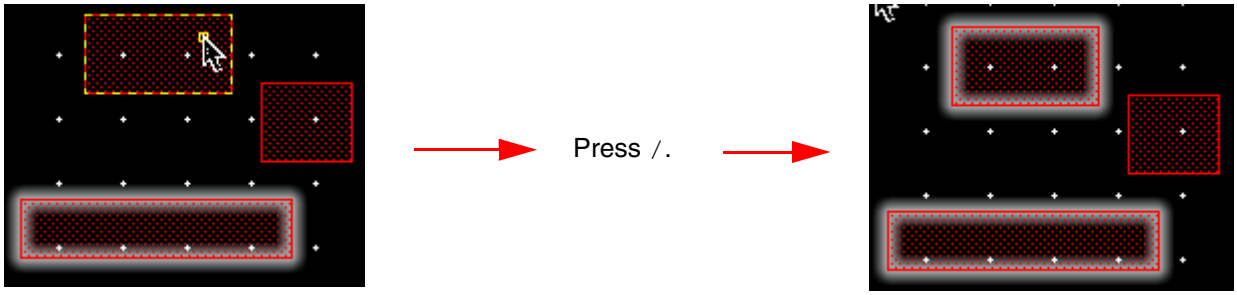


To set multiple objects as DRD targets with the bindkey, select them one by one, as shown.



Place the mouse pointer over an object.

The object is set as a DRD target.



Place the mouse pointer over another object.

This object too is set as a DRD target.

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

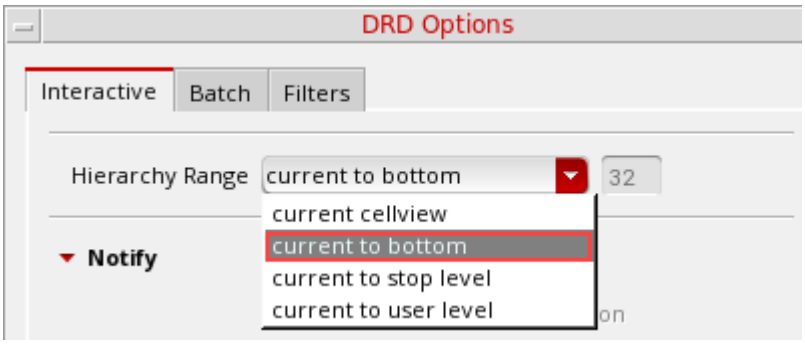
You can also use the bindkey to set as DRD targets the following objects: objects below the current hierarchy level and overlapping objects at any hierarchy level.

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

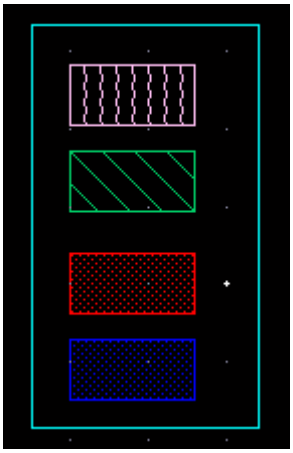
Setting Hierarchical Objects as DRD Targets

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

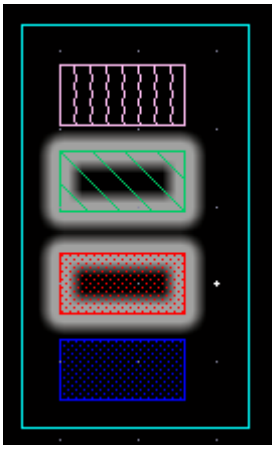
1. Open the DRC Options form.
2. On the *Interactive* tab, set the *Hierarchy Range* to *current to bottom*, as shown.



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

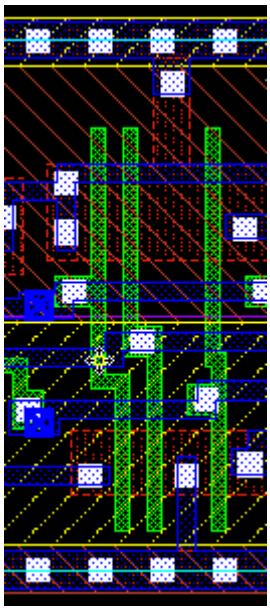


Objects M2 (red) and M3 (green) located below the current hierarchy level are set as DRD targets by using the / bindkey.



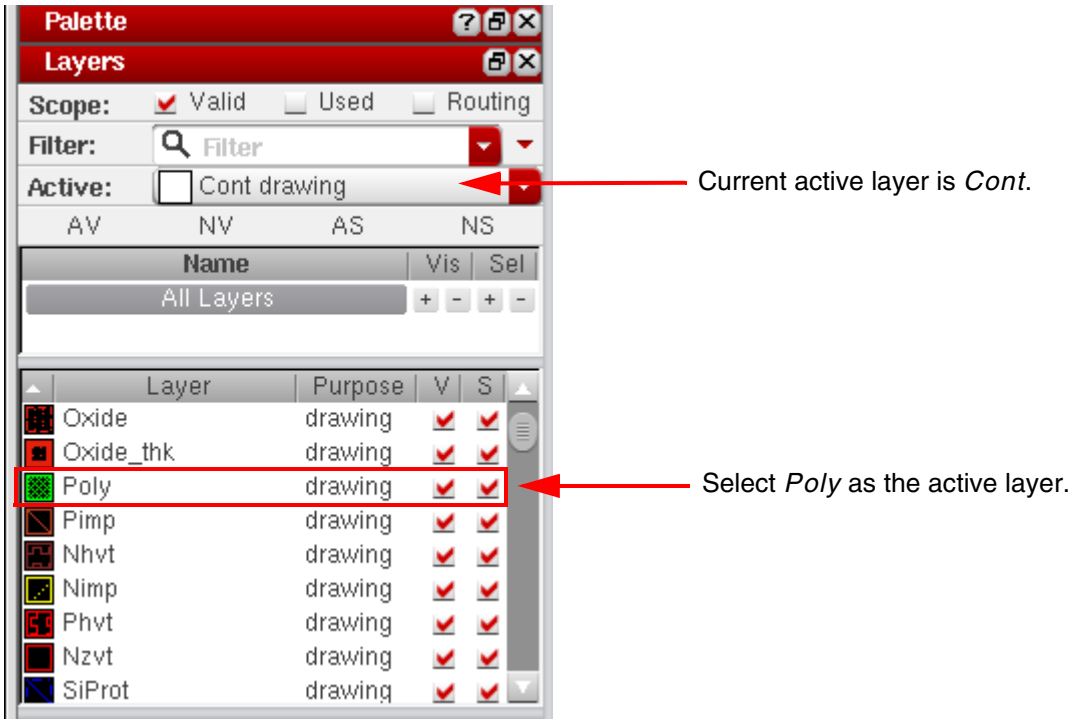
Setting Overlapping Objects as DRD Targets

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.

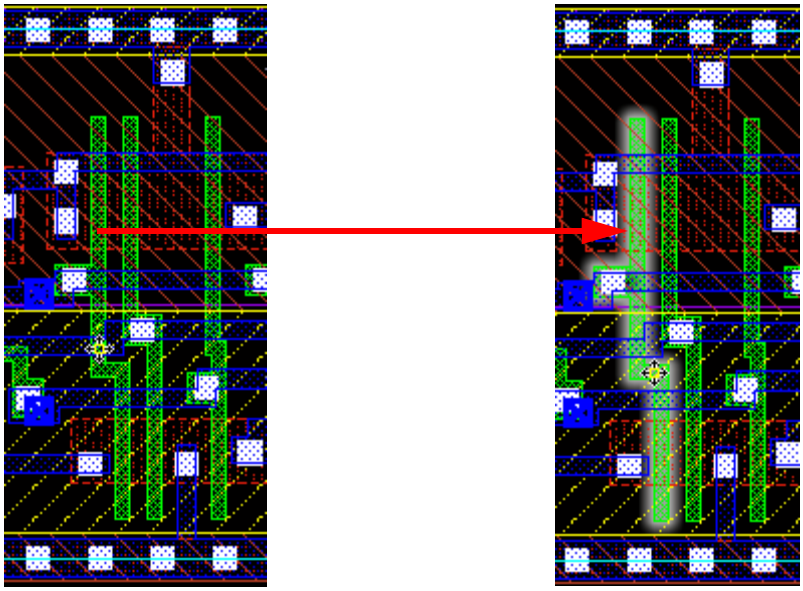


To set an object on the *Poly* layer as a DRD target:

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.

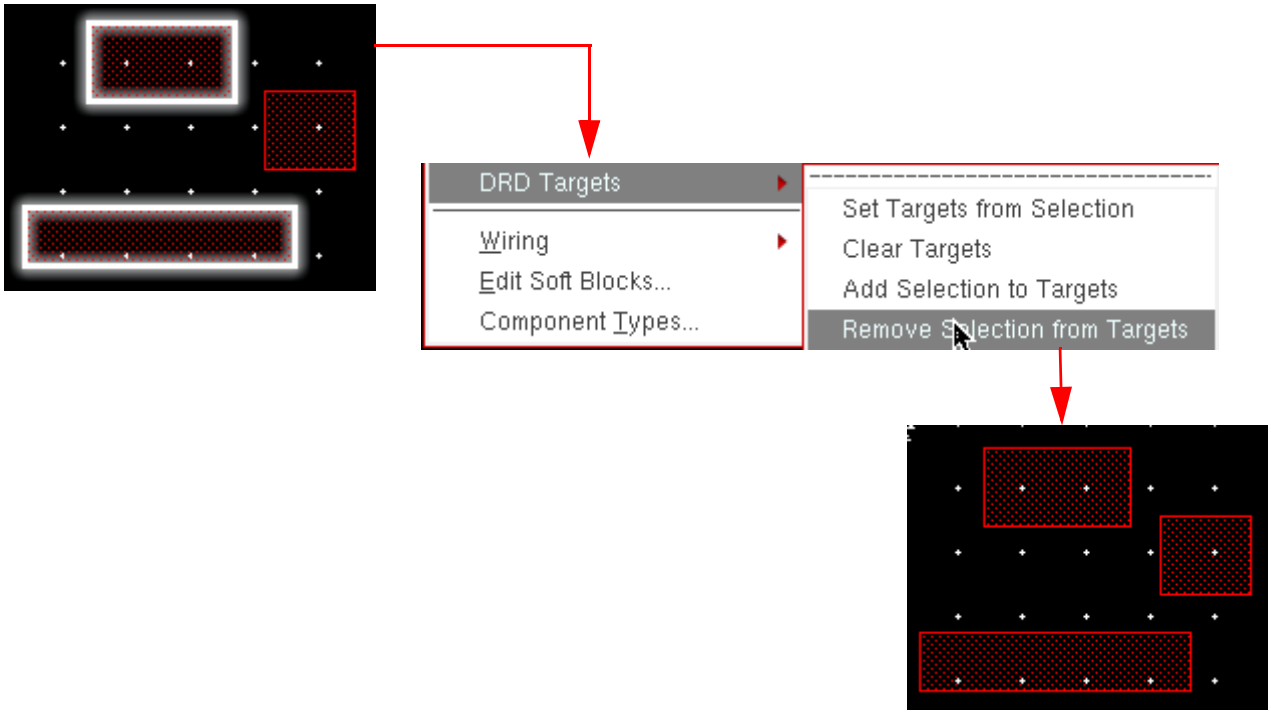


Unsetting DRD Targets

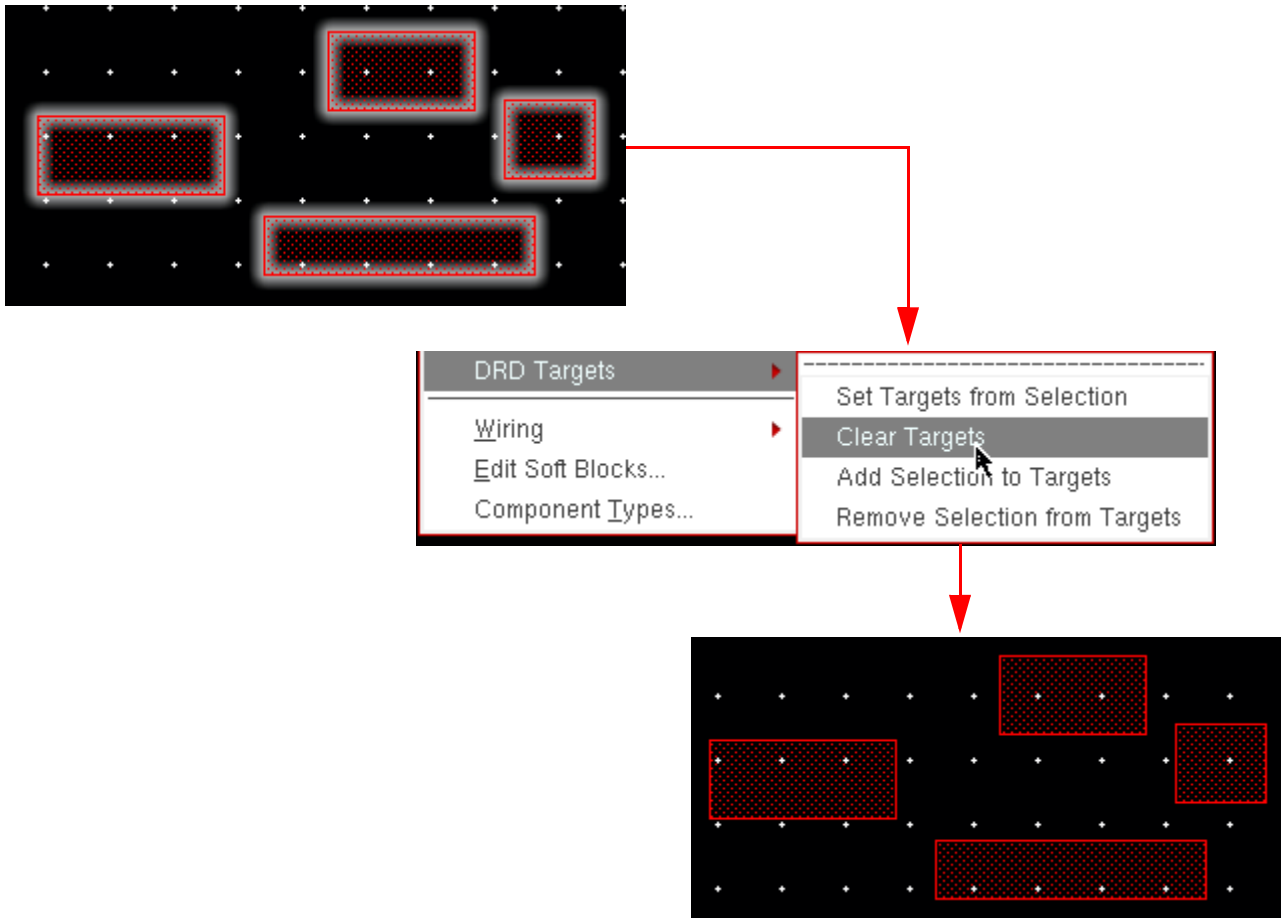
You can unset a DRD target by using the *DRD Targets* menu options or by using the bindkey provided.

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*.

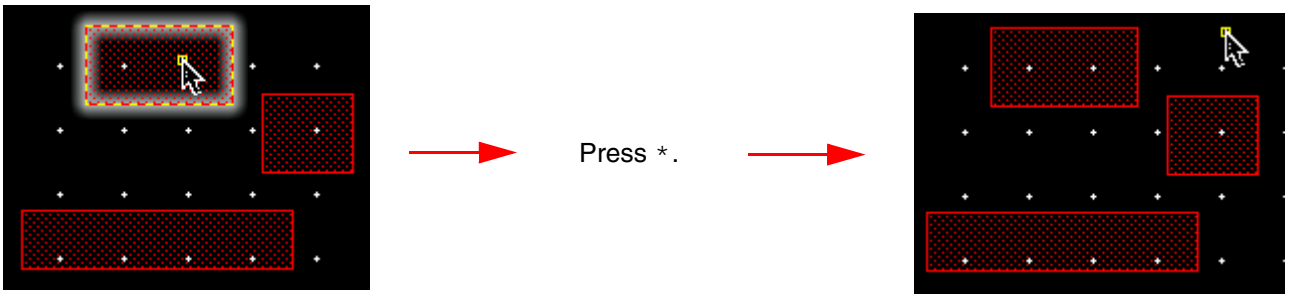


To unset all existing DRD targets, right-click on the canvas and choose *DRD Targets – Clear Targets* from the context menu or choose *Edit – DRD Targets – Clear Targets*, as shown.

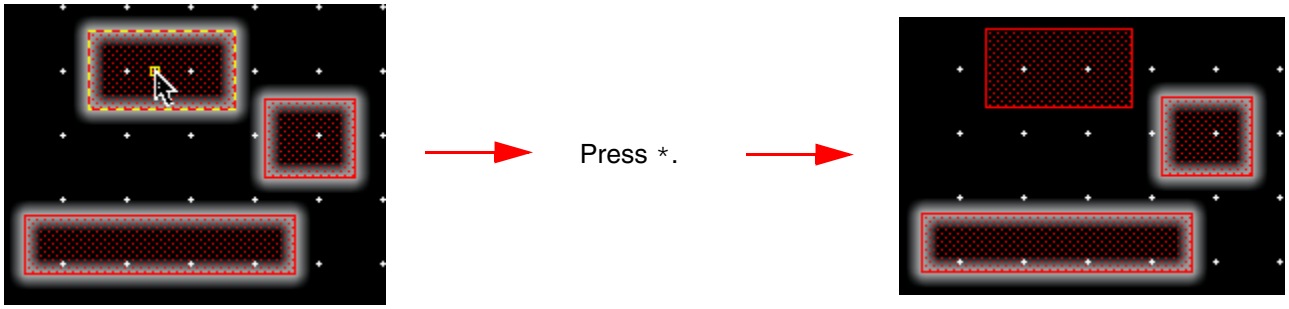


Unsetting DRD Targets Using the Bindkey

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.

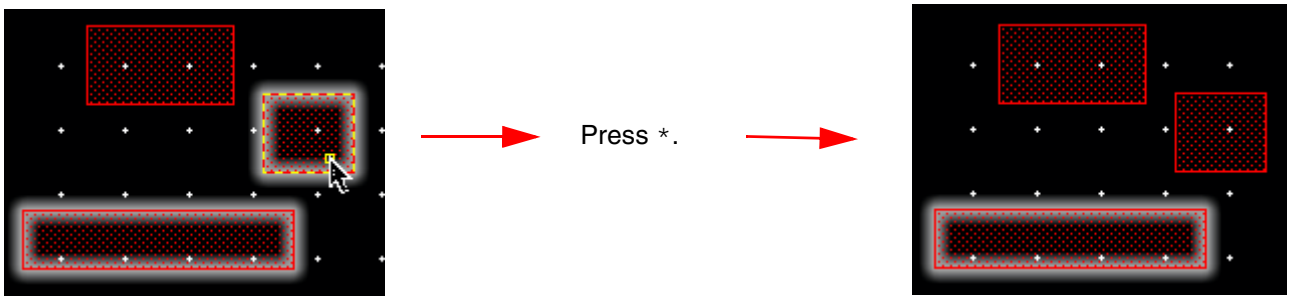


To unset multiple DRD targets with the bindkey, place the mouse pointer over each successive DRD target and press `*`, as shown.



Place the mouse pointer over a DRD target.

The object is removed from the selection.



Place the mouse pointer over another DRD target.

This object too is removed from the selection.

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 `drdAddTargets` and `drdRemoveTargets` 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.

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.

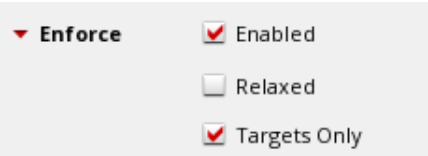
Targeted Enforce

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 DRC 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*.

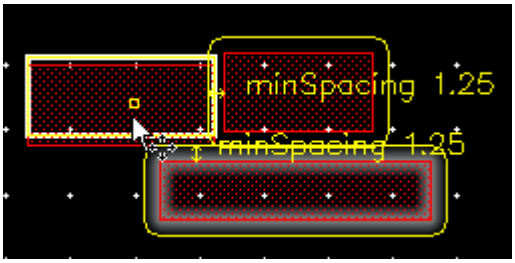
You can also enable DRD Notify mode by selecting the *Notify – Enabled* check box in the DRC Options form. When DRD Notify is enabled, the unenforceable violations are shown as Notify halos.

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. See Setting DRD Targets.

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. When you edit an object in the violation region, the object is moved to a violation-free location.

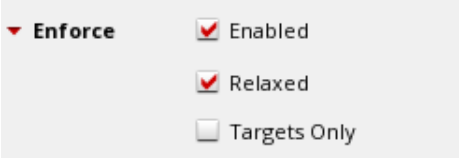
The Relaxed Enforce mode is useful for manual device placement. For example, you want to place two PMOS devices closer with minimum oxide spacing between them. In the Enforce mode, as soon as an Nwell spacing violation occurs, the pointer movement is stopped, and the pointer cannot move beyond the violation 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. The Relaxed Enforce feature supports only the copy and move operations for wire.

To enable Relaxed Enforce:

1. Open the DRC 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.



4. Click *OK*.

Video

For a video demonstration of the Relaxed Enforce feature, see Using the Relaxed Enforce Mode in DRD.

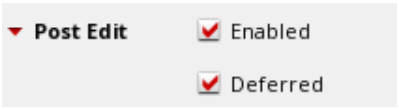
Deferred Post Edit

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 DRC 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.

Video

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

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.

To enable DRD What-if halos:

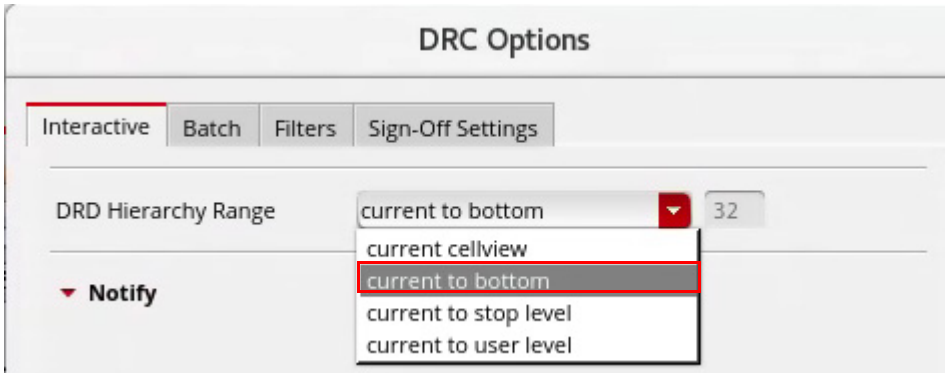
1. Disable DRD Enforce using one of the following methods:
   - Click the *Enforce* button on the toolbar.
   - Deselect the *Enforce – Enabled* check box in the DRC Options form.
2. Enable DRD Notify using one of the following methods:
   - Click the *Notify* button on the toolbar.
   - Select the *Notify – Enabled* check box in the DRC Options form.
3. Select the *Notify – Smart Snap* check box.
4. Set one or more objects as DRD targets. 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.

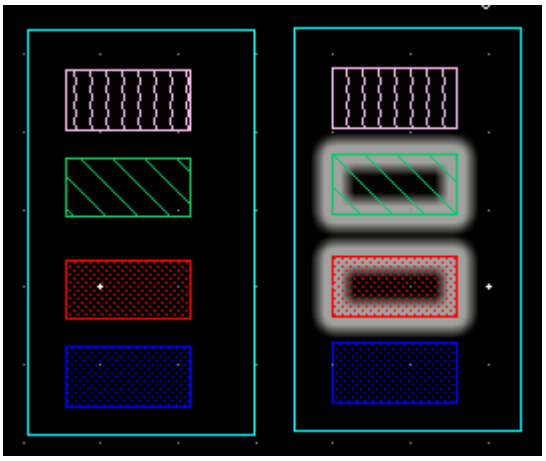
What-if Functionality for Hierarchical Objects

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

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

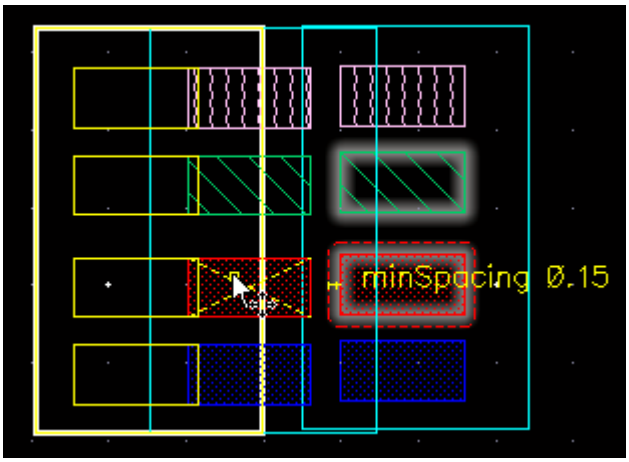


2. Use the / bindkey to set the hierarchical objects in an instance as DRD targets. The following figure shows the green M3 and red M2 objects set as DRD targets in the instance on the right.



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

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 DRD Smart Snapping.

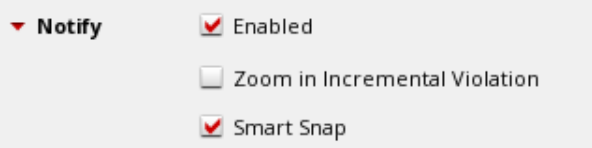
DRD Smart Snapping

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

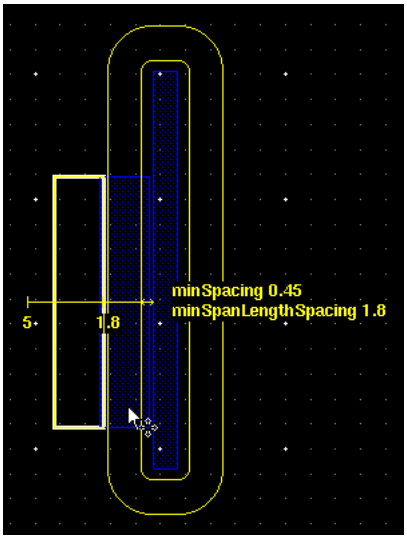
1. In the DRC Options form, select the *Notify – Enabled* check box.
2. Select the *Smart Snap* check box and click *OK*.



Note: To enable smart snapping by default, set the drdEditVioLimit environment variable to `t`.

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.

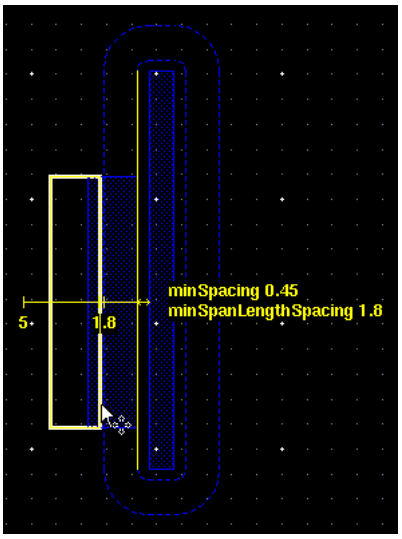


4. To snap the shape to a halo or to any edge of the static shape:

   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.

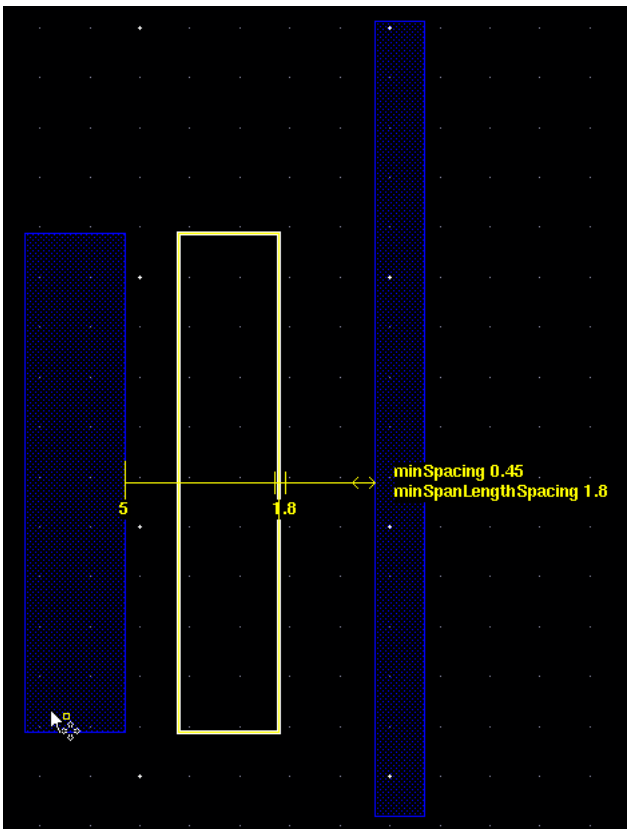


   c. Press ` again to unfreeze the halos.

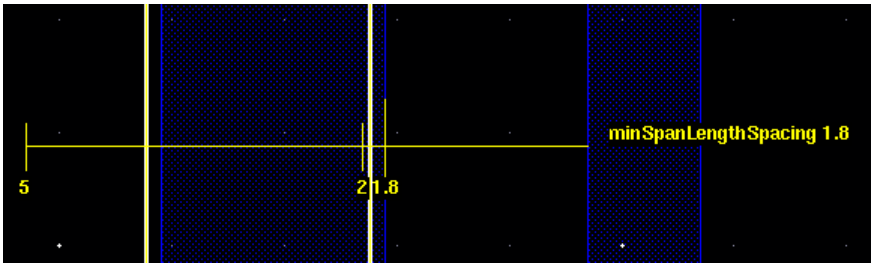
5. To snap a shape to a discrete spacing marker:

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

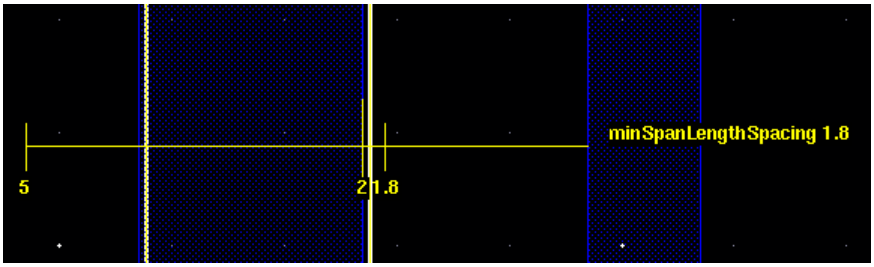
As a result, the halos disappear and you can see only the discrete spacing markers, as shown below.



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

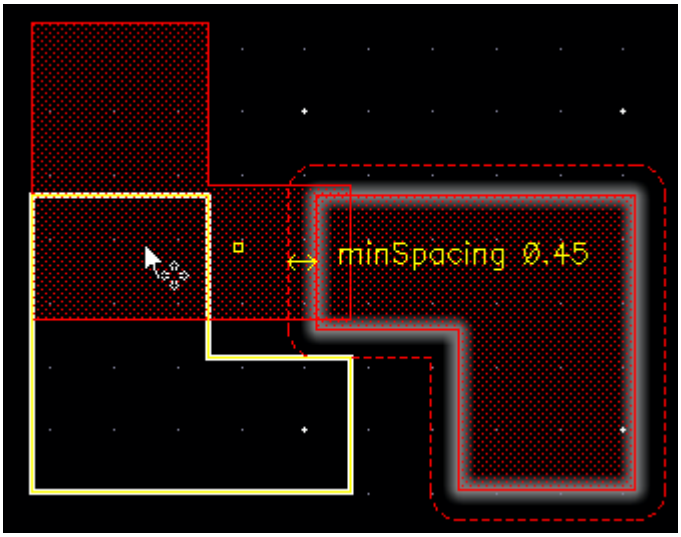
You can also snap an object to a What-if halo created around an object set as a DRD target. See What-if Halos.

1. Set an object as a DRD target.

   See Setting DRD Targets.

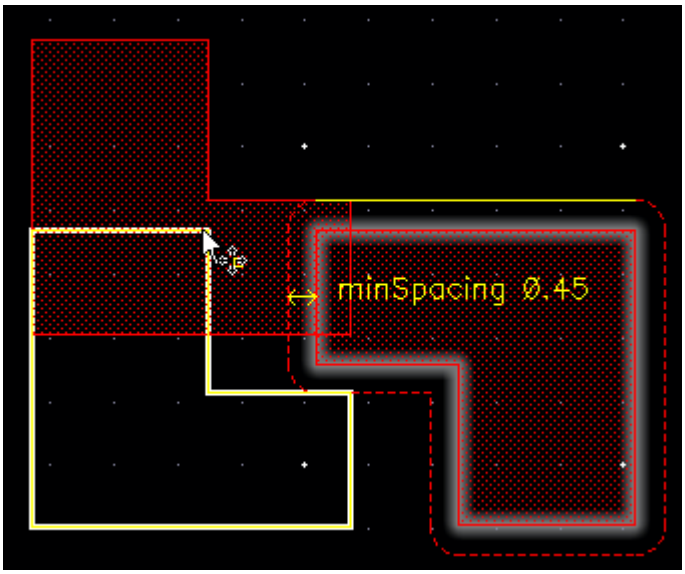
2. Move an object toward the DRD target.

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.



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.

   See Setting DRD Targets.

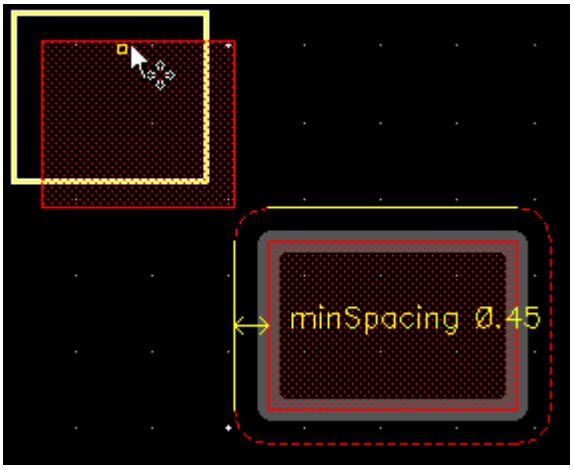
2. Move an object toward the DRD target.

   DRD creates a halo around the DRD target.

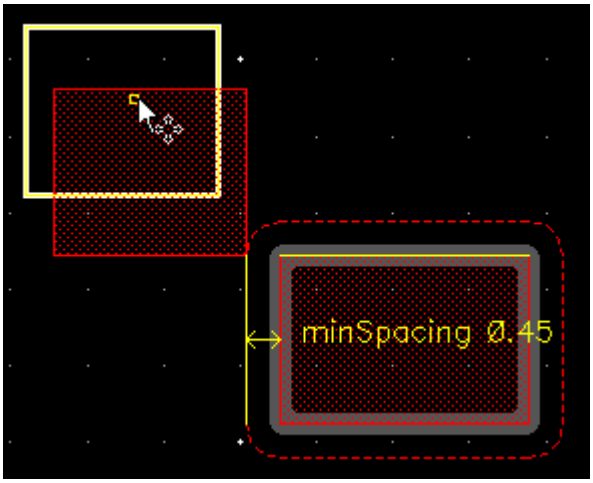
3. Press ` to freeze the halo on the DRD target.

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.

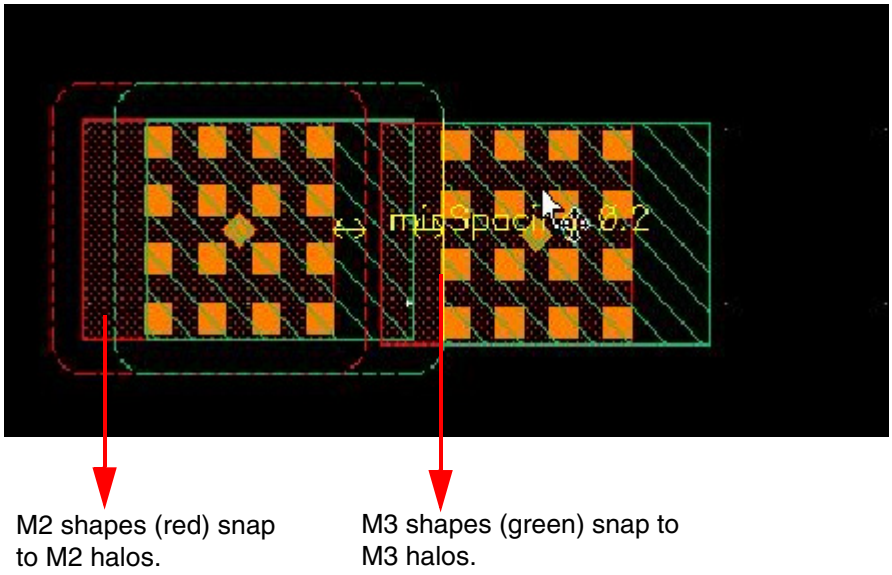


The following example shows the object snapping to the top horizontal edge of the DRD target and the left vertical edge of the frozen halo.



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.

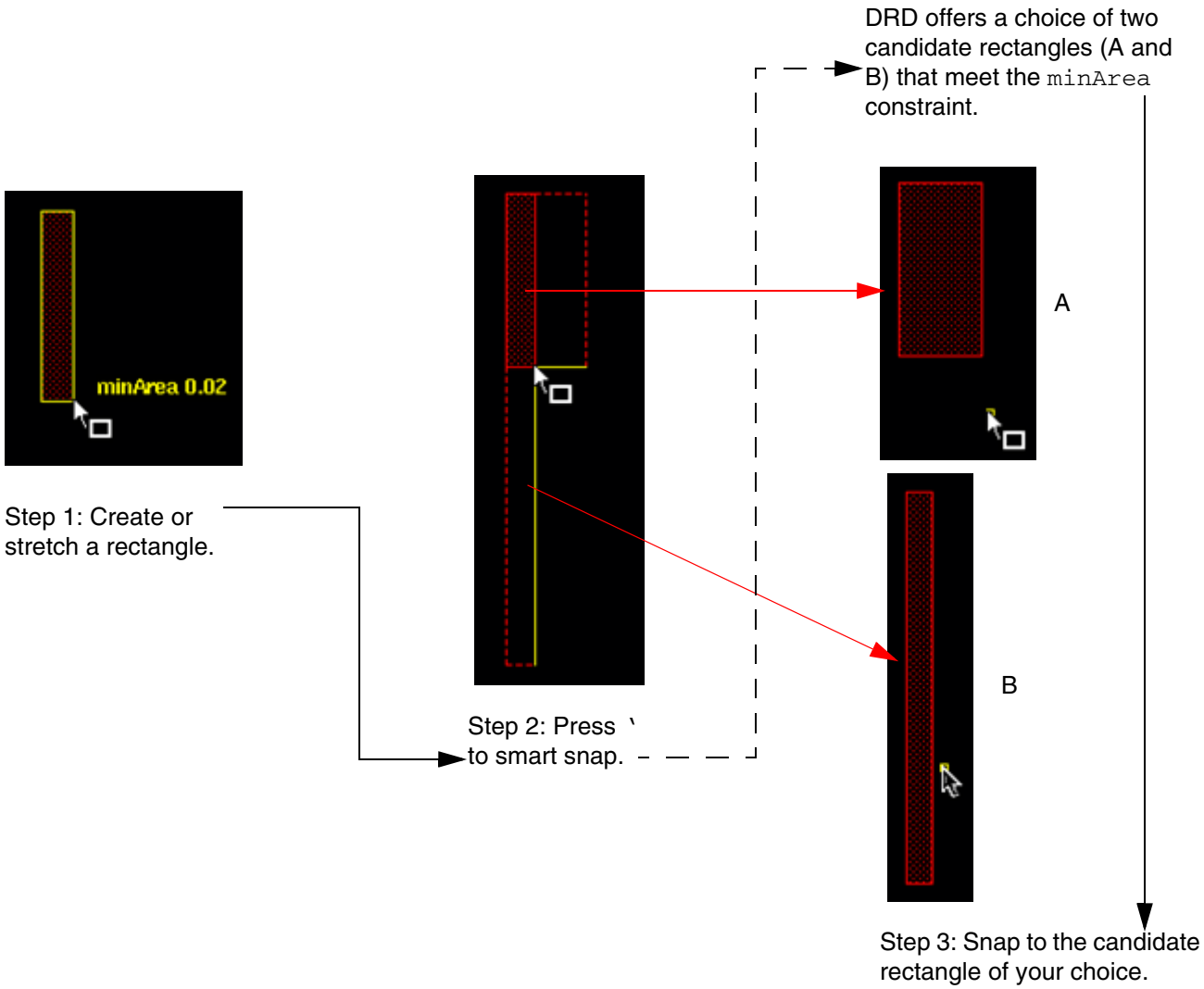


M2 shapes (red) snap to M2 halos.

M3 shapes (green) snap to M3 halos.

Snapping Based on the minArea Constraint

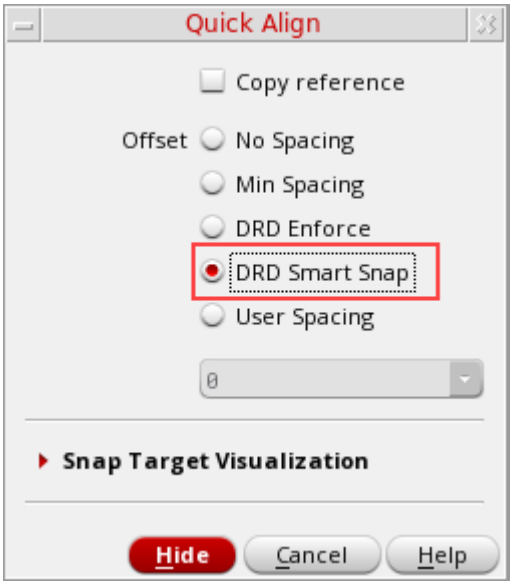
When creating or stretching a rectangle, DRD offers a choice of candidate rectangles that meet the `minArea` constraint. You can snap to a candidate rectangle of your choice as shown below.



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.

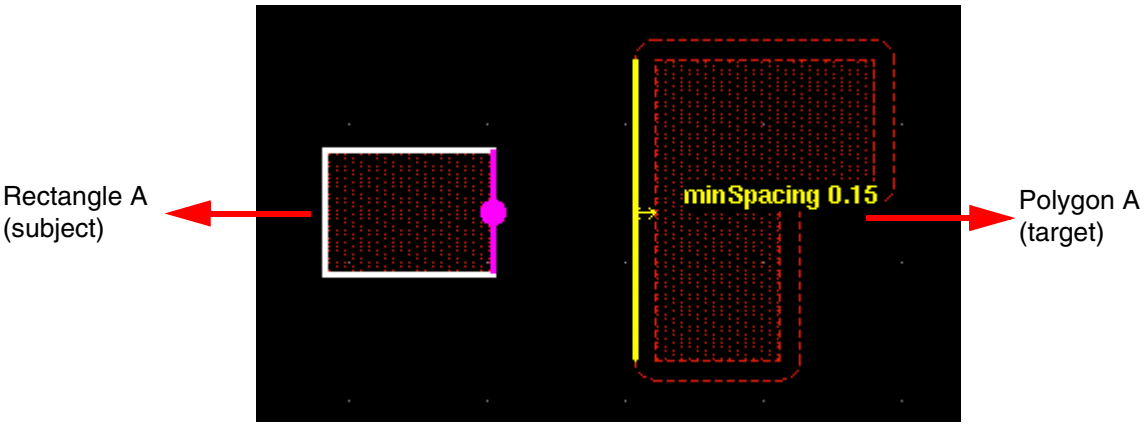
Note: This option is available only when the *Smart Snap* check box is selected in the DRC Options form.



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.

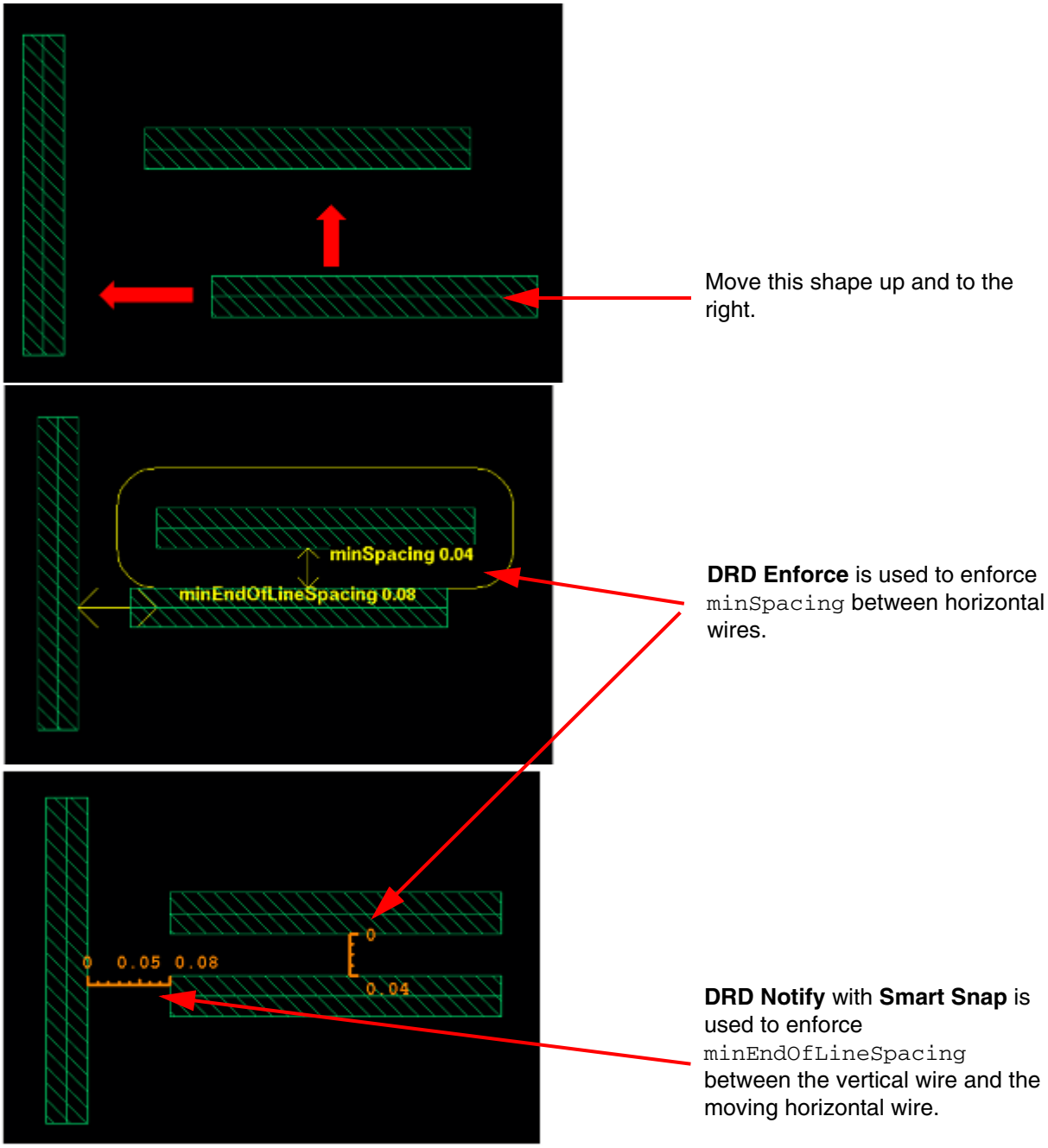
DRD Smart Snap creates halos (the dashed red lines) around `Polygon A` based on the spacing constraints defined in the technology file.



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.



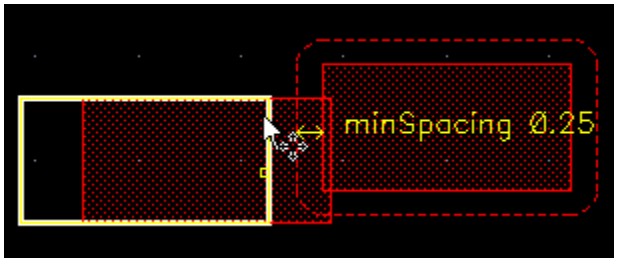
Move this shape up and to the right.

DRD Enforce is used to enforce `minSpacing` between horizontal wires.

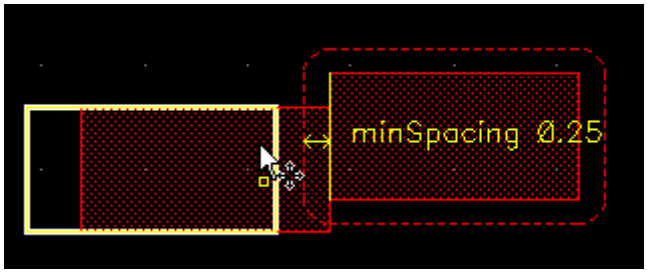
DRD Notify with **Smart Snap** is used to enforce `minEndOfLineSpacing` between the vertical wire and the moving horizontal wire.

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 DRC 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.

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.

Video

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

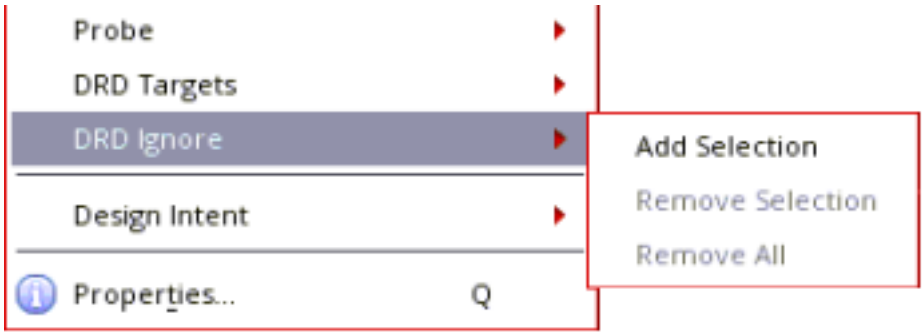
Excluding Instances from Checking

You can choose to exclude some instances from DRD checking. If the *Ignore Marked Instance* check box in the *Filters* page of the DRC 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 Support for FinFET Devices

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
- Defining Directional and Discrete Width Rules
- Defining Directional and Discrete PR Boundary Rules
- Snapping Objects to Legal Width Values
- Defining Directional and Discrete Opposite Extension Rules

Video

For a quick overview of DRD, see DRD for Advanced Node Design.

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.

An example of the technology file definition is shown below:

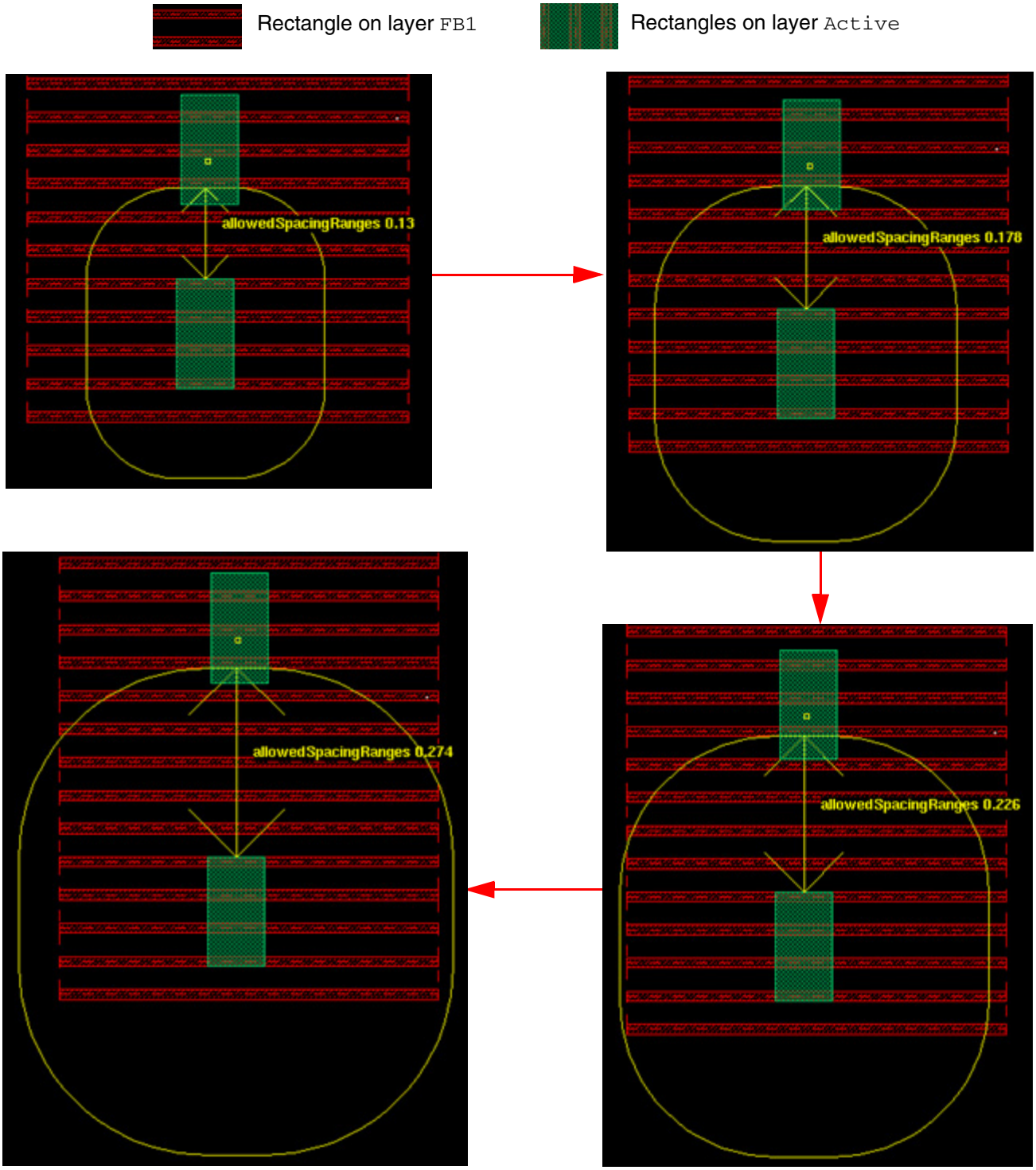
```
spacings(
    ( allowedSpacingRanges "Active"
        'vertical
        'stepSize 0.048
        'overLayer "FB1"
        (">=0.130")
    )
) ;spacings
```

where,

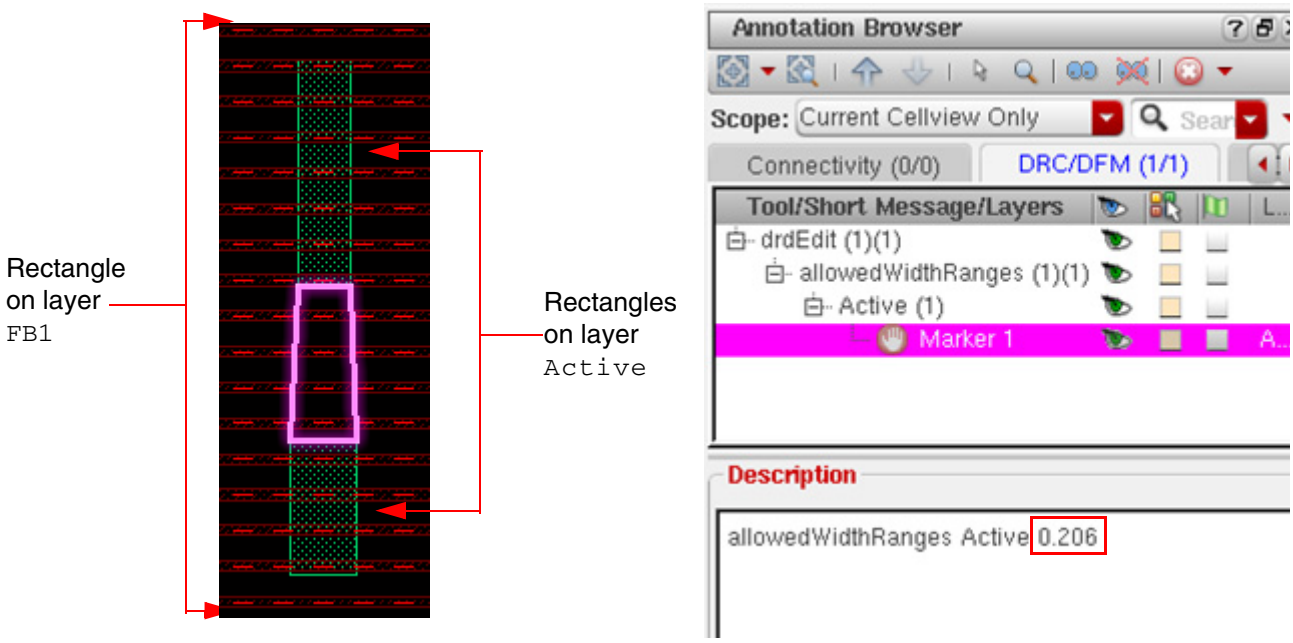
- Shapes on layer `Active` are located over a shape on layer `FB1`.
- The spacing between these shapes is measured vertically and the minimum legal spacing value is 0.13 user units.
- As the spacing increases, the legal spacing values are calculated as increments of 0.048 user units, as shown below.

```
0.130
0.130 + (1 x 0.048) = 0.178
0.130 + (2 x 0.048) = 0.226
0.130 + (3 x 0.048) = 0.274
…
```

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.



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

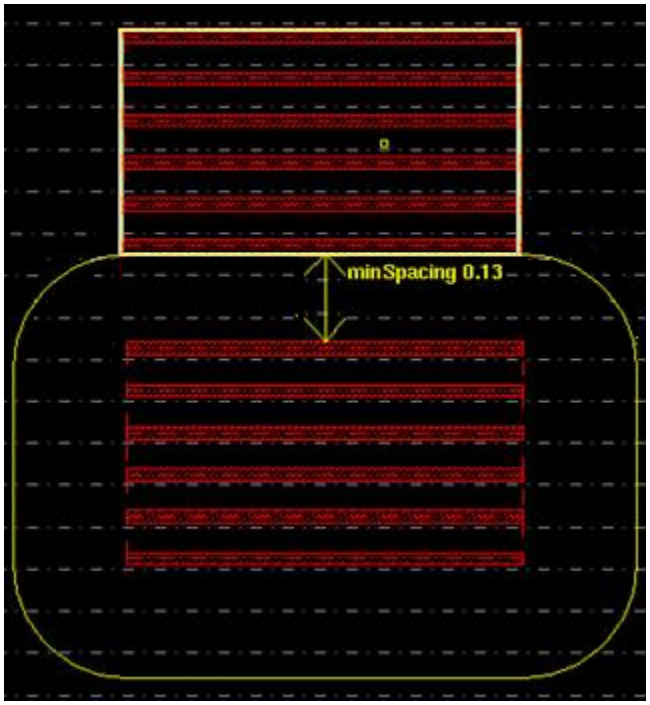
The `minSpacing` constraint defines the minimum legal spacing between fin boundaries on a layer.

An example of the technology file definition is shown below:

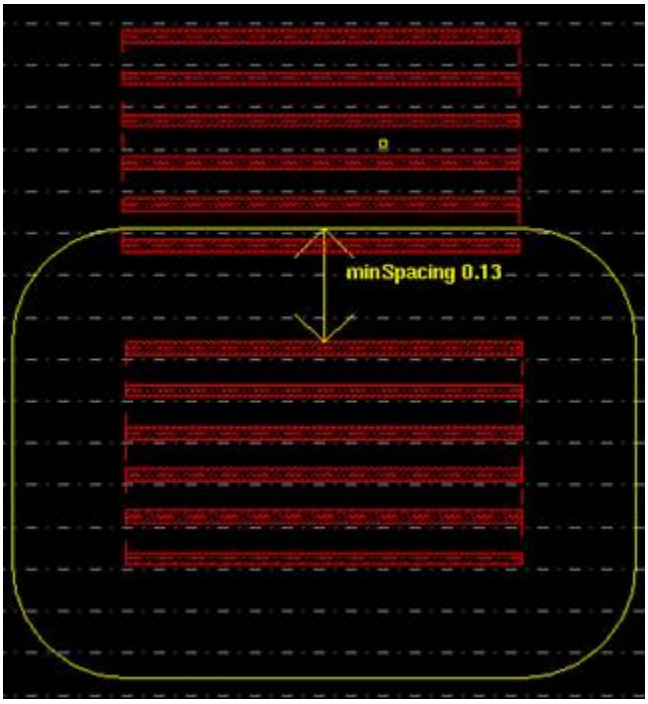
```
( minSpacing "FB1" 0.130 )
```

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

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.



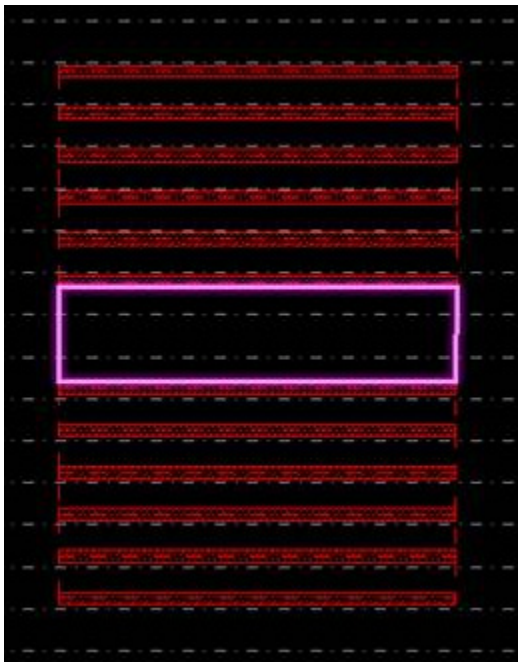
Enforce mode

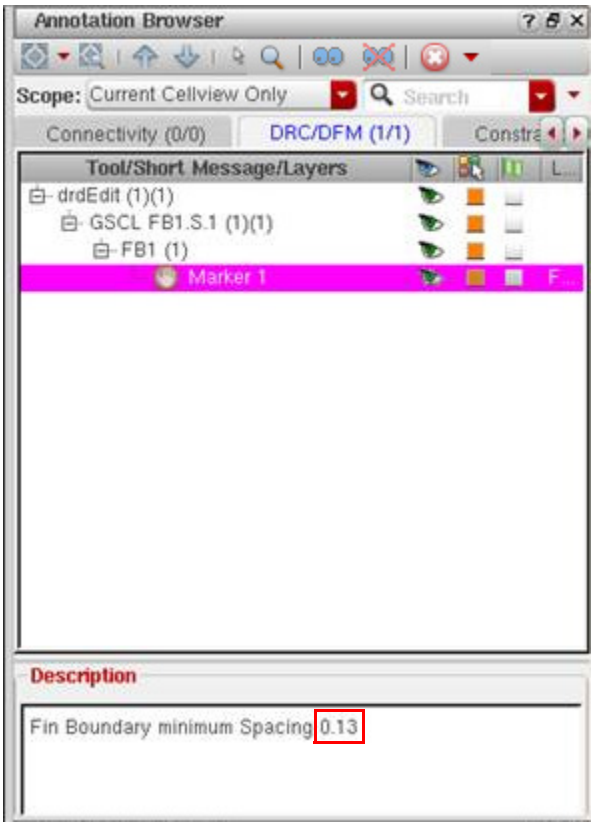


Notify mode

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.

An example of the technology file definition is shown below:

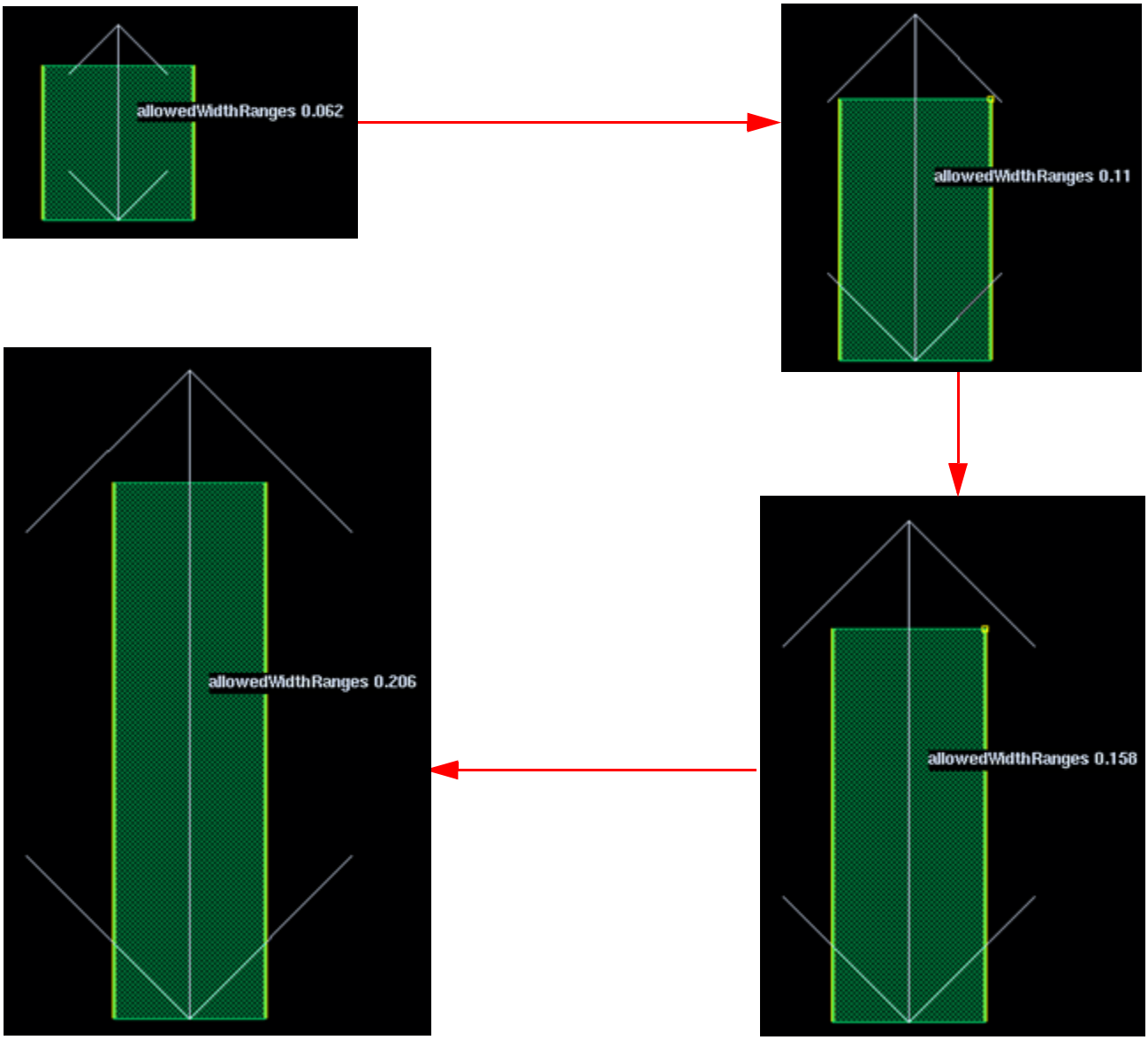
```
spacings(
    ( allowedSpacingRanges "Active"
        'measureVertical
        'stepSize 0.048
        (">=0.062")
    )
) ;spacings
```

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.

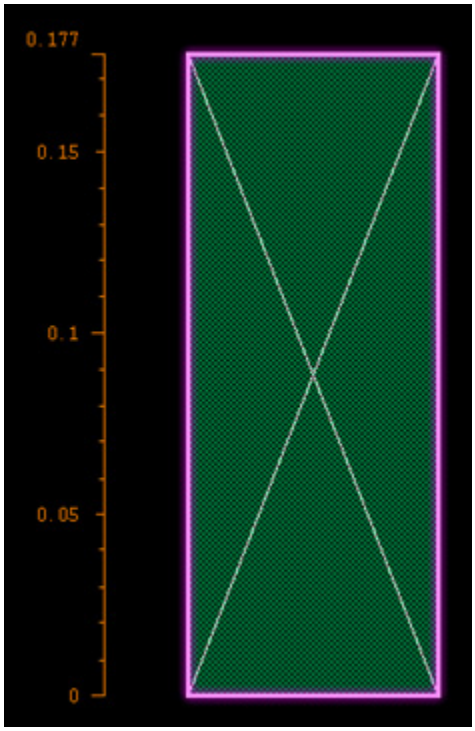
  ```
  0.062
  0.062 + 1 x 0.048 = 0.11
  0.062 + 2 x 0.048 = 0.158
  0.062 + 3 x 0.048 = 0.206
  …
  ```

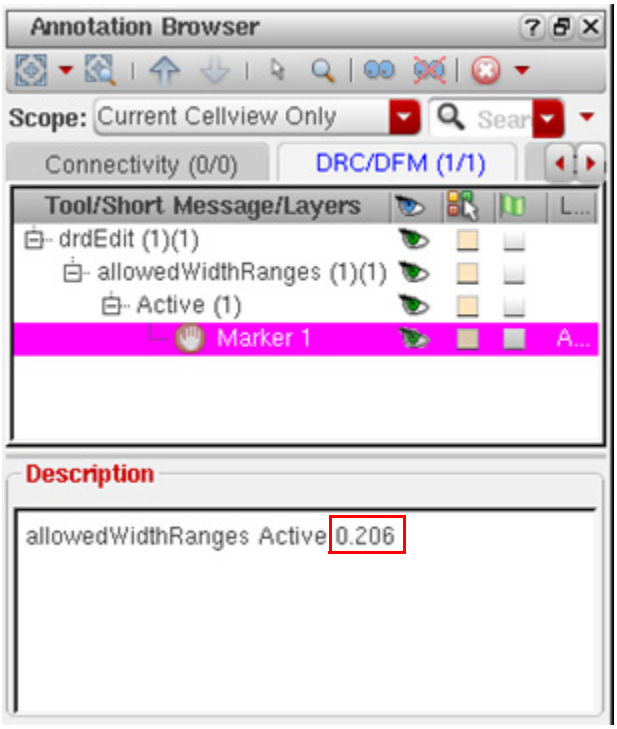
The following figures illustrate this example in DRD Notify mode. As the rectangle is stretched vertically, DRD displays arrows to indicate the incremental legal width values.



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 Defining Directional and Discrete PR Boundary Width Rules
- `minPRBoundaryInteriorHalo`; see Defining Directional and Discrete PR Boundary Spacing Rules

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.

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

An example of the technology file definition is shown below:

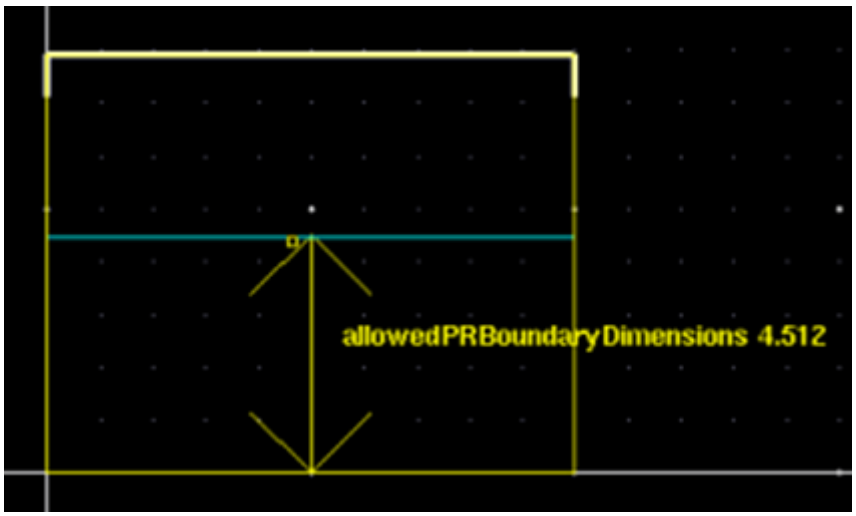
```
spacings(
    ( allowedPRBoundaryDimensions
        'vertical
        'stepSize 0.048
        (">=0.048")
        'ref "prBndry.h"
    )
) ;spacings
```

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.

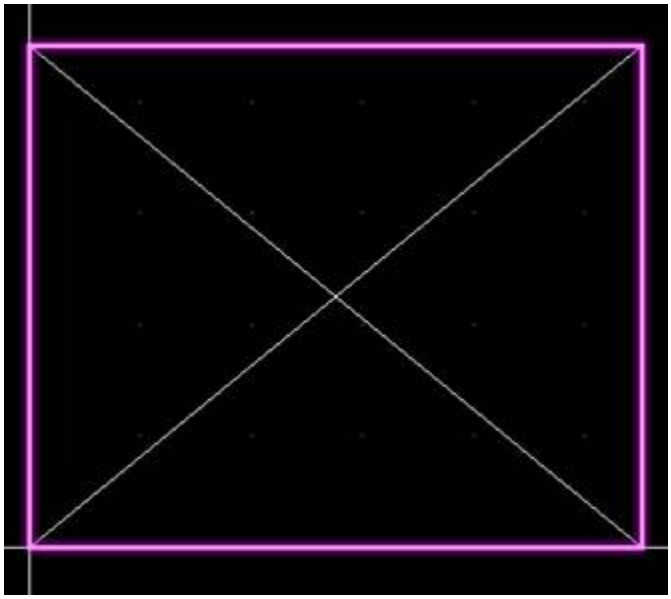
```
0.048
0.048 + 1 x 0.048 =  0.096
0.048 + 2 x 0.048 =  0.144
0.048 + 3 x 0.048 =  0.192
…
```

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`

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:

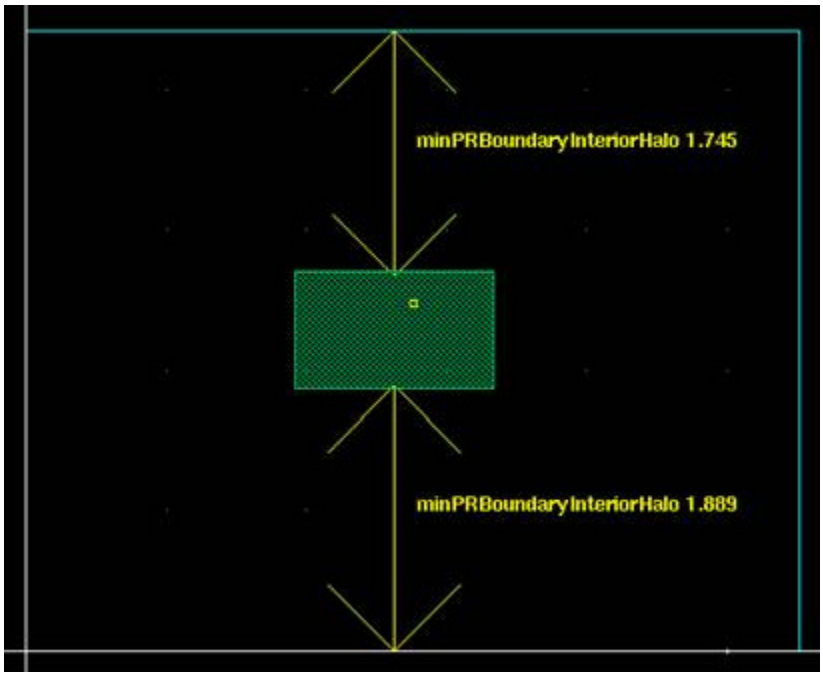
```
spacings(
    ( minPRBoundaryInteriorHalo "Active"
        'vertical
        'stepSize 0.048  0.113
        'ref "prBndry.EN"
    )
) ;spacings
```

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.

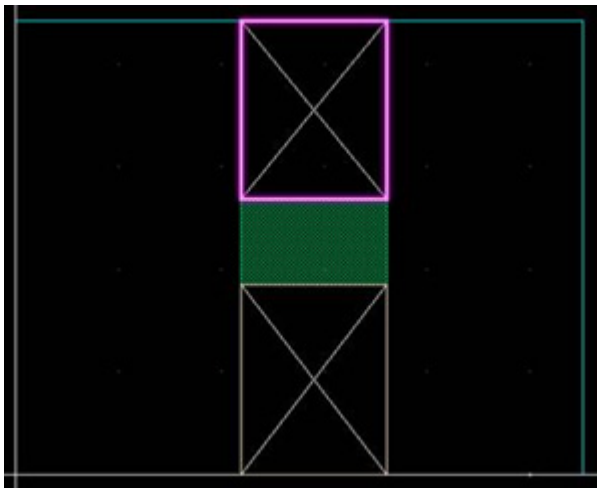
```
0.113
0.113 + 1 x 0.048 = 0.161
0.113 + 2 x 0.048 = 0.209
0.113 + 3 x 0.048 = 0.257
…
```

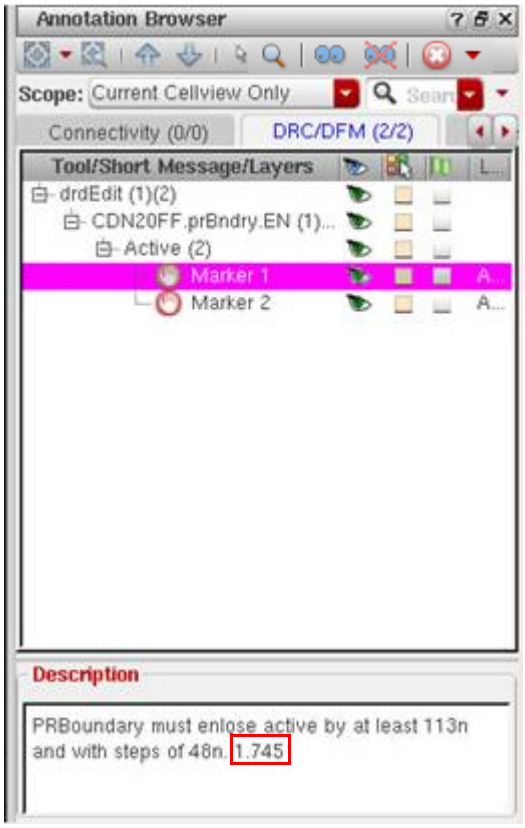
The following figure illustrates this example in DRD Notify mode. As the vertical spacing between the PR boundary and the enclosed rectangle increases, DRD displays arrows to indicate the incremental legal spacing values.



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

(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).



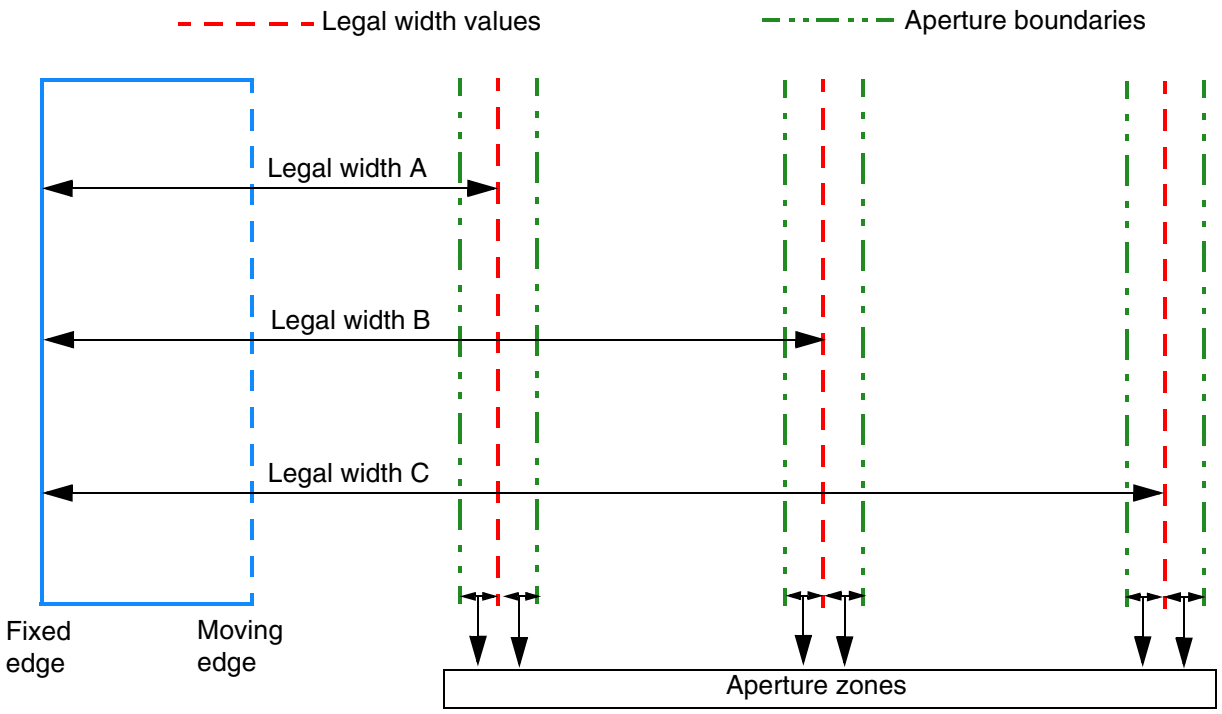


Snapping Objects to Legal Width Values

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.

Discrete dimension gravity is turned on by default. To turn off discrete dimension gravity, set the drdEditSmartSnapAllowedWidthSnap 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. See Defining Directional and Discrete PR Boundary Width Rules.
- Add the `allowedPRBoundaryDimensions` constraint for PR boundaries. See Defining Directional and Discrete PR Boundary Width Rules.

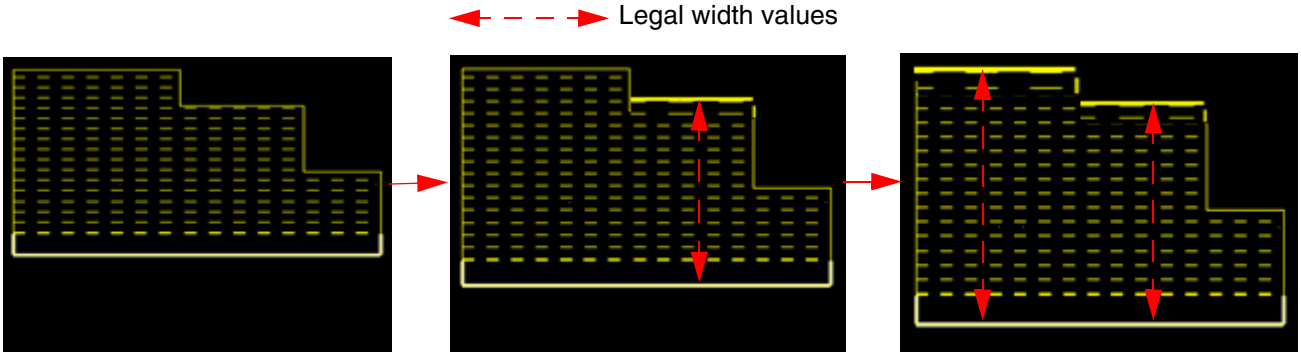
Examples of Discrete Dimension Gravity

This functionality is useful while:

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

Working with Shapes

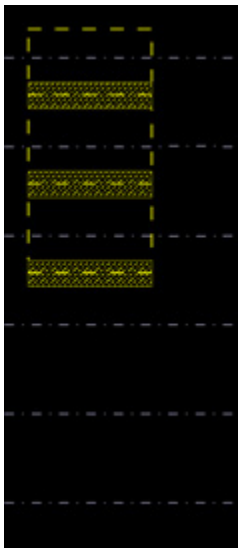
The following figure illustrates discrete dimension gravity during a single-direction polygon stretch operation:



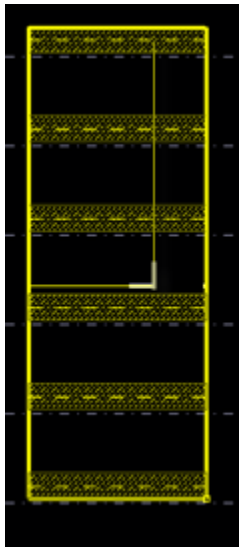
The bottom edge of the polygon is being stretched vertically

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

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



The original rectangle



Vertical and horizontal stretch

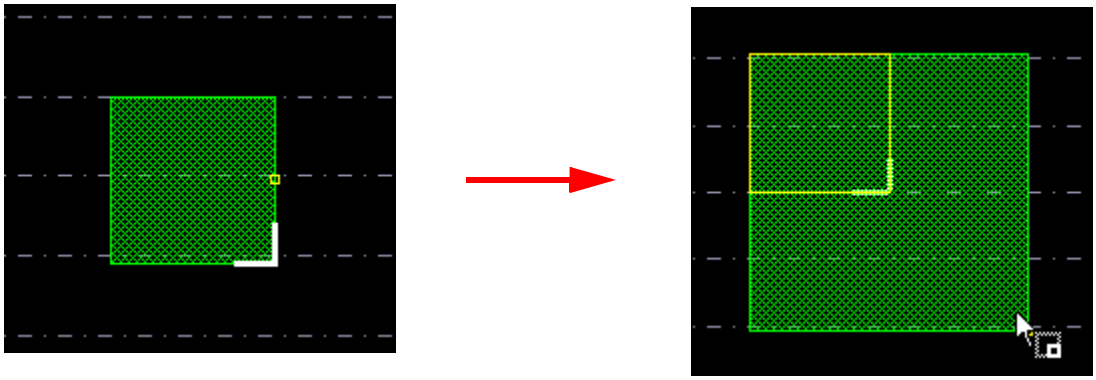
The rectangle abides by legal width values by automatically snapping to the nearest legal width values

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.

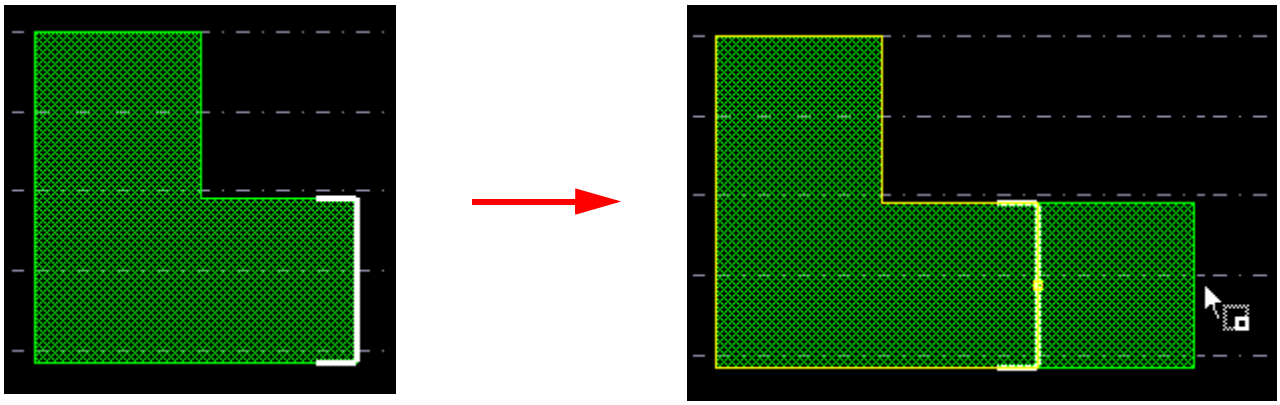
The following figure illustrates how discrete dimension gravity works for pins of type *rectangle*:



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*:



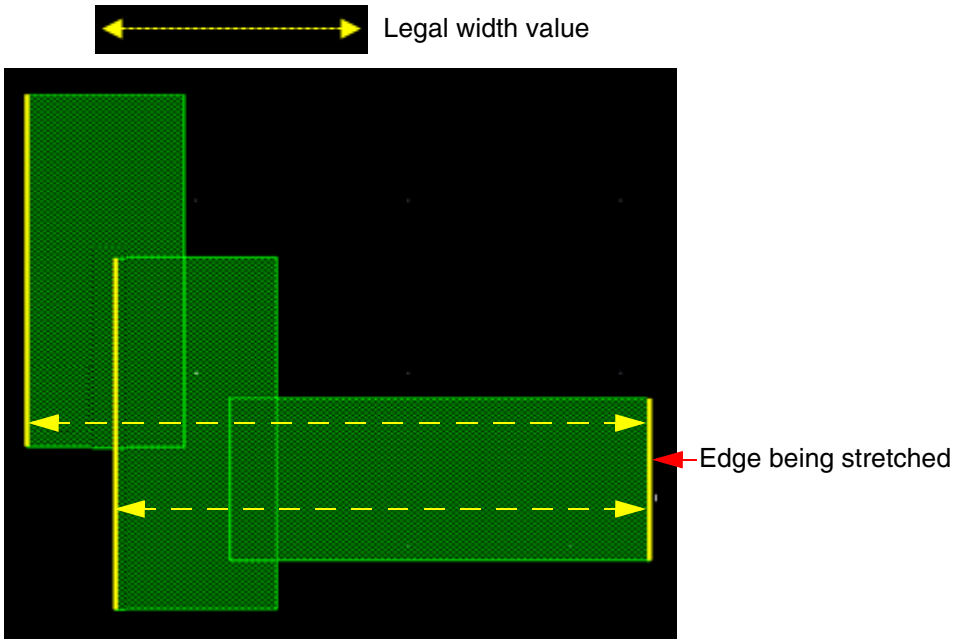
A polygonal pin is stretched horizontally.

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.

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



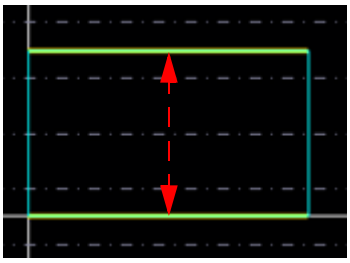
The island abides by legal width values by automatically snapping to the nearest legal width values

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

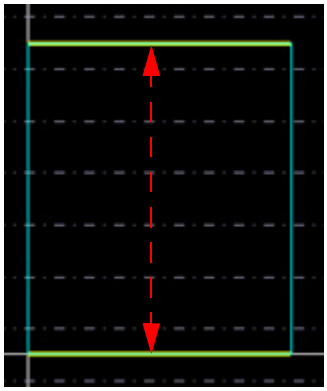
Note: This functionality requires at least one top-level shape in a hierarchical island.

Working with PR Boundaries

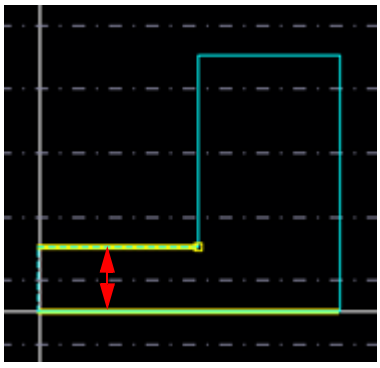
The following figures illustrate discrete dimension gravity while working with PR boundaries.



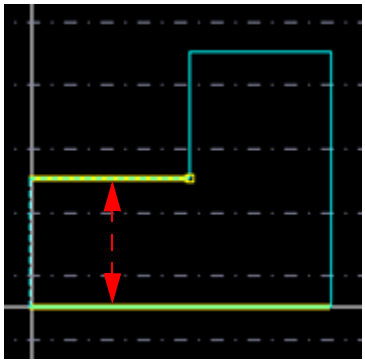
Creation of a rectangular PR boundary



Stretching a rectangular PR boundary



Creation of a polygonal PR boundary



Stretching a side of a polygonal PR boundary

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,

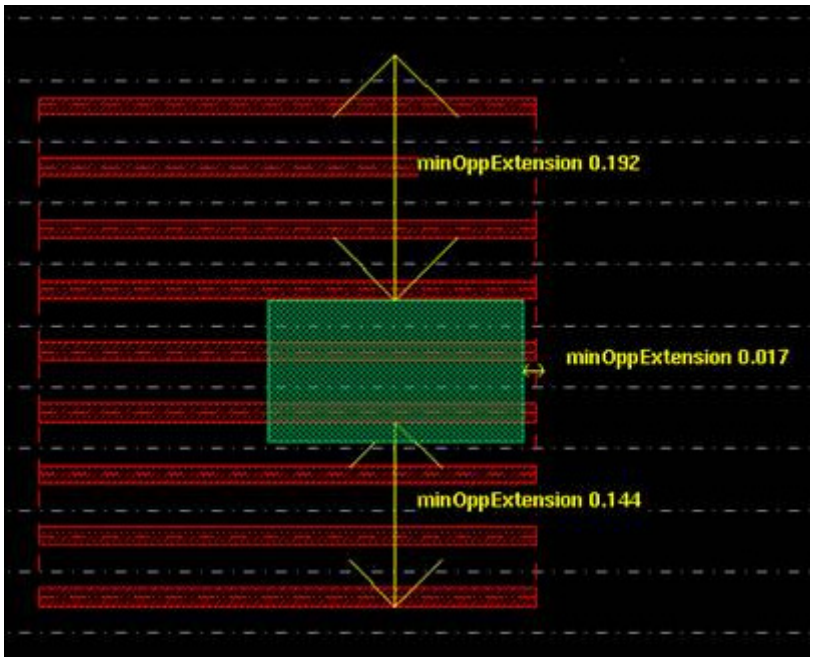
- The fin boundary is located on layer `FB1` and a shape is located on layer `Active`.
- The minimum legal extension of the fin boundary over a shape, when measured vertically, is 0.096 user units.
- The legal vertical extension values are calculated as increments of 0.048 user units, as shown below.

  ```
  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.

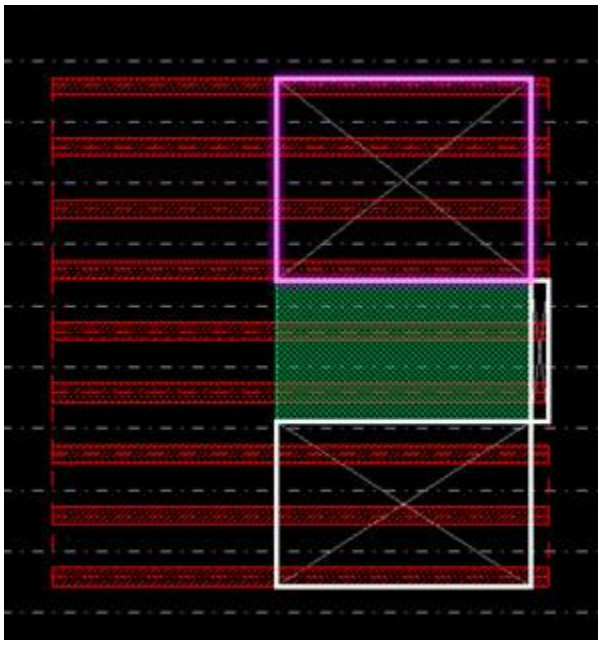
The following figures illustrate this example in DRD Notify mode. As the enclosure between the rectangles on `FB1` and `Active` increases vertically and horizontally (to the right), DRD

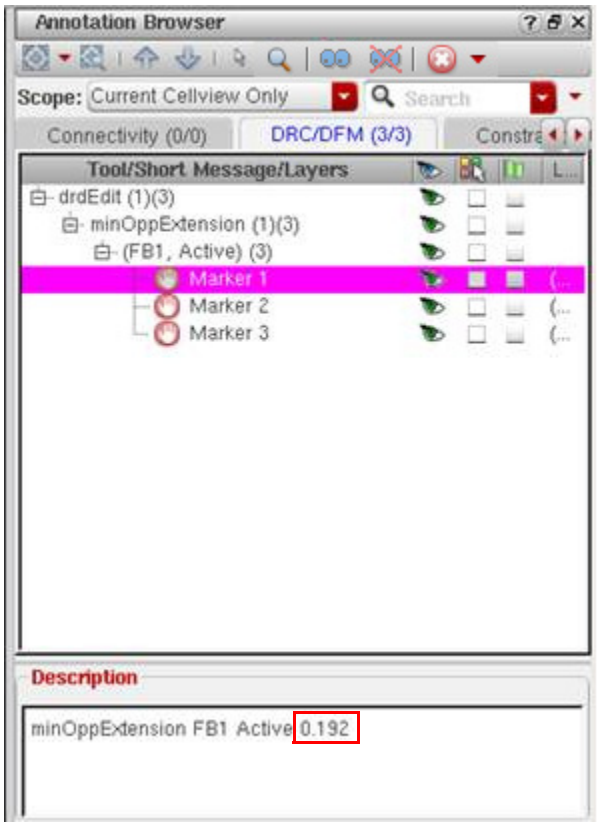
displays arrows to indicate the incremental legal extension values on three sides of the rectangle on layer `Active`.



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)

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).





4

DRD Constraints Support

This chapter contains the following topics:

- Supported Constraints
  - Supported Constraints and Parameters
  - Supported Constraints and Parameters (Virtuoso Layout Suite EXL)
  - Supported Constraints and Parameters (Virtuoso Advanced Node for Layout Standard)
  - Supported Constraints and Parameters (Virtuoso Advanced Node for Layout Only)
- 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

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 EXL.
- Limited support is available for the `'or` operator in DRD editing; it is used mainly to combine constraints of the type extension.

Supported Constraints and Parameters

Category	Constraint Name (ASCII Syntax)	Tier Level Support	Support Type
Antenna	antennaRatio	L, XL, GXL, and EXL	notify/post-edit/batch-check
	cumPerLayerAntennaRatio	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
	minHoleArea	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	endOfLineKeepout	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
	sameMetalAlignedCuts	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
	maxNumMinEdges	L, XL, GXL, and EXL	post-edit/ batch-check
	minDiagonalEdgeLength	L, XL, GXL, and EXL	notify/post-edit/batch-check
	minEdgeAdjacentDistance **Note:** minEdgeAdjacentDistance is only meant for edgeCount param 2. It does not support edgeCount > 2.	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 **Note:** Minimum overlap is derived from minExtension.	L, XL, GXL, and EXL	notify/post-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
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

Category	Constraint Name (ASCII Syntax)	Tier Level Support	Support Type
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
	minSameNetSpacing (Two layers) **Note:** minSameNetSpacing is derived from minSpacing in VLS XL, GXL, and EXL.	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	definedCutClassesOnly	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

Supported Constraints and Parameters (Virtuoso Layout Suite EXL)

Important

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	`endOfLineKeepout` Parameters: `horizontal`, `vertical`, `eolSpacingRange`, `otherEolSpacingRange`, `jointAsEol`, `checkOtherWidth`, `equalRectWidth`	XL and EXL	notify/post-edit/batch-check
	`minClusterSpacing (One layer)` Parameters: `enclosingLayer`, `viaEdgeType`	XL and EXL	notify/post-edit/batch-check
	`overlapViaGroup`	XL and EXL	notify/post-edit/batch-check
	`viaGroup` Parameters: `allowBend`	XL and EXL	notify/post-edit/batch-check
Extension	`minEndOfLineEdgeExtension` Parameters: `minVoltage`, `maxVoltage`	XL and EXL	notify/post-edit/batch-check
	`minOppExtension` Parameters: `widthHorizontal | widthVertical`	XL and EXL	notify/post-edit/batch-check
	`minQuadrupleExtension` Parameters: `lineEndGapOnlyTable`	XL and EXL	notify/post-edit/batch-check
Length	`minStepEdgeLength` Parameters: `adjEolSpacingRange`, `jogEdge`, `adjJogEdge`, `adjNotchEdge`	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, maxNeighbors, neighborSpacing	XL and EXL	notify/batch-check
	minCutRoutingSpacing (Two layers) Parameters: toDiffConcaveCorner	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
	minSideSpacing (One layer) Parameters: otherLayer, cutClass, cutWithin	XL and EXL	notify/post-edit/batch-check
	minViaSpacing (One layer) Parameters: otherCutClass, otherViaEdgeType, deltaVoltage, exceptGap, exceptSameMetalOverlap, orthogonalEnclosingMetal, fixedCutClassList, otherFixedCutClassList, horizontalSpacingRange, verticalSpacingRange, farEnds	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

Category	Constraint Name (ASCII Syntax)	Tier Level Support	Support Type
Width	allowedWidthRanges Parameters: exceptOverlap	XL and EXL	notify/post-edit/batch-check
	minCornerToCornerDistance Parameters: extensionLength, oppositeDirection	XL and EXL	notify/post-edit/batch-check

Supported Constraints and Parameters (Virtuoso Advanced Node for Layout Standard)

Important

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

Category	Constraint Name (ASCII Syntax)	Tier Level Support	Support Type
Area	`minArea` Parameters: `insideLayers`, `outsideLayers`, `insidePurposes`, `outsidePurposes`	XL and EXL	notify/post-edit/batch-check
Complex Spacing	`forbiddenCutClassSpacingRange`	XL and EXL	notify/post-edit/batch-check
	`forbiddenEdgePitchRange`	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
	minExtensionEdge Parameters: insideLayers, outsideLayers, insidePurposes, outsidePurposes	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, alignedLayer, alignedHorizontal | alignedVertical	XL and EXL (scalar and table)	notify/post-edit/batch-check
	edgeMustCoincide	XL and EXL	notify/post-edit/batch-check
	edgeMustOverlap	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
	minQuadrupleExtension Parameters: insideLayers, outsideLayers, insidePurposes, outsidePurposes	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	rectShapeDir Parameters: exceptEdgeLength, exceptExactSize, exceptRanges, exceptViaLayer, exceptViaSize, insideLayers, insidePurposes, outsideLayers, outsidePurposes, twoSides, width	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
	minSpacing (One layer) Parameter: exceptEolWidth, insideLayers, outsideLayers, insidePurposes, outsidePurposes	XL and EXL	enforce/ notify/post-edit/batch-check
	minStitchOverlap	XL and EXL	notify/post-edit/batch-check
	minViaSpacing (One layer) Parameters: exceptExactAligned, overLayer, overLayerWidth, exceptSameNet, exceptSameMetal	XL and EXL	enforce/ notify/post-edit/batch-check
	viaSpacing Parameters: allCuts, cutSizeRanges, insideLayers, outsideLayers, insidePurposes, outsidePurposes	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

Supported Constraints and Parameters (Virtuoso Advanced Node for Layout Only)

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

Category	Constraint Name (ASCII Syntax)	Tier Level Support	Support Type
Area	`minRectArea` Parameters: `colorMask`, `layer`, `overlapType`	XL and EXL	notify/post-edit/batch-check

Category	Constraint Name (ASCII Syntax)	Tier Level Support	Support Type
Complex Spacing	bothSidesViaForbiddenSpacingRange	XL and EXL	notify/post-edit/batch-check
	endOfLineKeepout Parameters: mask1, mask2, mask3, diffMask, twoSides, forwardGap	XL and EXL	notify/post-edit/batch-check
	forbiddenCutClassSpacingRange Parameters: cutClassPrl, allCuts, layer, layerWidth, mask1, mask2, mask3, paraLength, within, sameMask, longEdge	XL and EXL	notify/post-edit/batch-check
	maxViaArrayClusterSize	XL and EXL	notify/post-edit/batch-check
	minAdjacentFourViaSpacing	XL and EXL	notify/post-edit/batch-check
	minClusterSpacing (One layer) Parameter: widthDirection (widthHorizontal | widthVertical)	XL and EXL	post-edit/ batch-check
	minCornerSpacing (One layer) Parameters: sameMask, treatLAsJog	XL and EXL	post-edit/ batch-check
	minEndOfLinePerpSpacing Parameters: negativePRL	XL and EXL	post-edit/ 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
	sameMetalAlignedCuts Parameters: centerToCenterSpacing | edgeToEdgeSpacing, horizontal | vertical, sameViaConnectivity	XL and EXL	notify/post-edit/batch-check
	viaGroup	XL and EXL	notify/post-edit/batch-check
	viaKeepoutZone	XL and EXL	post-edit/ batch-check

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

Category	Constraint Name (ASCII Syntax)	Tier Level Support	Support Type
Miscellaneous	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
	minNotchSpacing Parameters: excludeSpacing, horizontal | vertical	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
	minSpacingOver Parameter: colorMask ('mask1 | 'mask2 | 'mask3)	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, exceptViasOnSameNet | exceptViasOnSameMetal | 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

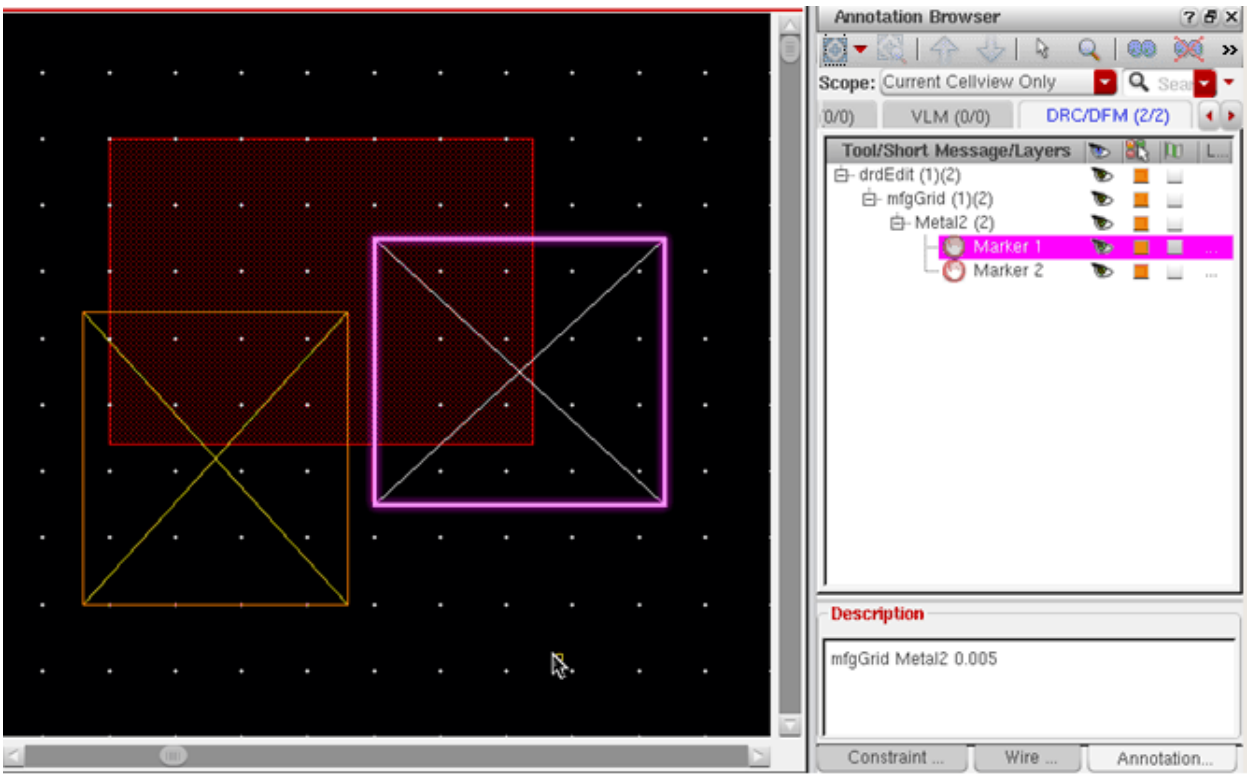
Category	Constraint Name (ASCII Syntax)	Tier Level Support	Support Type
	orthogonalWSPGrid	XL and EXL	notify/post-edit/batch-check
	trimMinSpacing (One layer)	XL and EXL	notify/post-edit/batch-check
	trimShape	XL and EXL	notify/post-edit/batch-check
	vertexInsideForbidden	XL and EXL	notify/post-edit/batch-check
	viaSpacing Parameters: sameMask, exceptWithin, edgeExtension, layer, extensionDirection	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 DRC Options form. For batch checking, select the *Grid* check box in the Batch-Check mode of the DRC 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 EXL, layers that are not specified as `validLayers` in the interconnect section of a constraint group receive only 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`
- `'butting`
- `'touching`
- `'straddling`
- `'inside`
- `'outside`
- `'color`

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 *Virtuoso Technology Data ASCII Files 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 4-1 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 `("Metal1" "HV")` shapes is 0.14, and for all other `Metal1` shapes, it is 0.12.
- The minimum spacing for `("Metal1" "LV")` shapes is 0.4.
- The minimum spacing for `("Metal1" "logic")` shapes is 0.5.

- The minimum spacing for `("Metal1" "HV")` shapes is 1.0.
- The minimum clearance between a `("Metal1" "HV")` shape and a `("Metal1" "LV")` shape is 0.5.
- The minimum clearance between a `("Metal1" "HV")` shape and a `("Metal1" "logic")` shape is 0.7.
- The minimum clearance between a `("Metal1" "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.

```
techPurposes(
;( PurposeName Purpose#   Abbreviation   [Attributes] )
;( ----------- --------   ------------   ------------ )
;User-Defined Purposes:
 ( logic       2000       LGC       )
 ( HV          2010       HV        'sigType "digital" 'parent "drawing")
 ( LV          2020       LV        'sigType "digital" 'parent "drawing")
) ;techPurposes
```

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

```
spacings(
  ( minWidth    "Metal1"                          0.12 )
  ( minWidth   ("Metal1" "HV")                    0.14 )
  ( minSpacing ("Metal1" "LV")                    0.4  )
  ( minSpacing ("Metal1" "logic")                 0.5  )
  ( minSpacing ("Metal1" "HV")                    1.0  )
  ( minSpacing ("Metal1" "HV") ("Metal1" "LV")    0.5  )
  ( minSpacing ("Metal1" "HV") ("Metal1" "logic") 0.7  )
) ;spacings
spacings(
  ( minSpacing ("Metal1" "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 Table 4-2 on page 135, are supported in VLS EXL, 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

Table 4-2 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 `Metal1` shapes whose maximum voltage swing is less than 1.5 is 0.06.
- The minimum spacing between `Metal1` 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 `Metal1` shapes whose maximum voltage swing is greater than or equal to 3.3 is 0.1.

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 `Metal1` layer (`minVoltageSpacing`) is described as:

```
spacingTables(
  ( minVoltageSpacing "Metal1"
   ( "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 >= 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 *Virtuoso Voltage Dependent Rules Flow Guide*.

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 `Metal1`:

```
( minDensity "Metal1"
  (( "step" nil nil ))
  (
    60.0 20.0
  )'ref "METAL1.D.1" 'description "Minimum Metal1 Density"
)
( maxDensity "Metal1"
  (( "step" nil nil ))
  (
    60.0  65.0
  ) 'ref "METAL1.D.1" 'description "Maximum Metal1 Density"
)
```

The second value (`20.0`) in the `minDensity` constraint listed above is the minimum density required for `Metal1`. The second value (`65.0`) in the `maxDensity` constraint represents the maximum density allowed for `Metal1`.

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:

```
___
             Window    Window     Min      Max    Violation Count       Density Percentage Statistics
Layer         Size      Step    Density  Density    Min     Max     Lowest Highest Median    Ave   Std Dev
==========  ========  ========  =======  =======  ======= =======  ======= ======= ======= ======= =======
Metal1       120.00     60.00   20.00%   65.00%        0       4  100.00% 100.00% 100.00% 100.00%   0.00%
```

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

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:

Layer	Window Size	Window Step	Min Density	Max Density	Violation Count Min	Violation Count Max	Density Percentage Statistics Lowest	Highest	Median	Ave	Std Dev
metal1	8.00	4.00	0.00%	100.00%	0	0	50.00%	75.00%	75.00%	62.50%	5.00%
Layer	Window Size	Window Step	Min Density	Max Density	Violation Count Min	Violation Count Max	Density Percentage Statistics Lowest	Highest	Median	Ave	Std Dev
metal2	8.00	4.00	0.00%	100.00%	0	0	0.00%	0.00%	0.00%	0.00%	0.00%

Verify Process Rules Summary: 0 Marker(s) created.

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

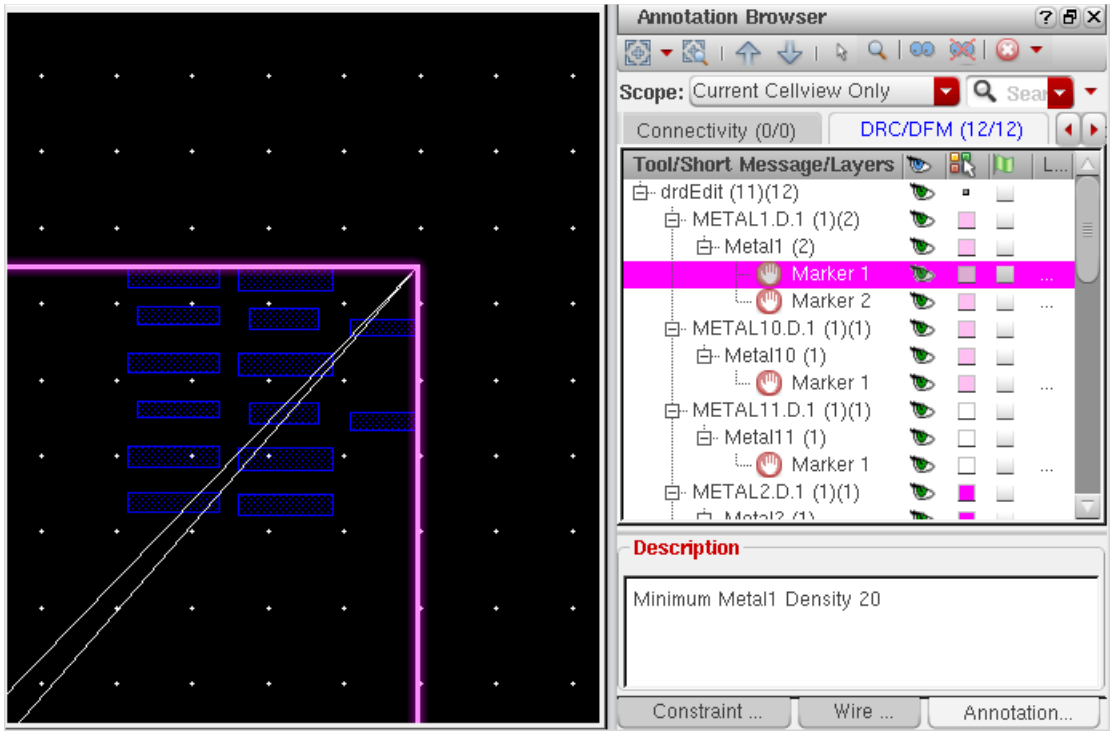
The median value is obtained after sorting all density values in the ascending order. If the number of density check windows is odd, the median is equal to the value that lies in the 'middle' of the sorted list, and if the number of density check windows is even, the median is calculated as an average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

Standard Deviation (Std Dev)

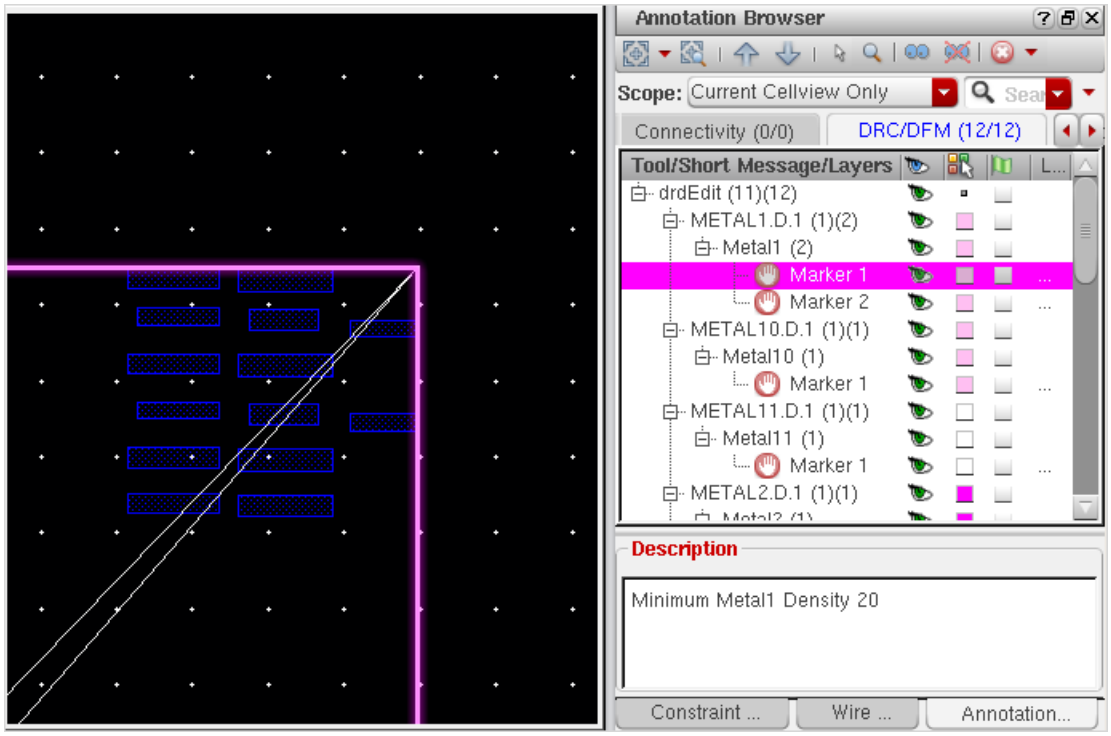
The standard deviation measures how close the density value for each density check window is to the mean value. A small standard deviation value of 5% indicates that the values are concentrated close to the mean value, which in this case is 62.5%.

Viewing Density Violations in Annotation Browser

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



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.

- DRC Options Form
- Signoff Fill Form

DRC Options Form

The DRC Options form lets you specify the In-Design DRC verification options in Virtuoso. The form contains the following tabs.

Tab	Description
Interactive	Lets you specify the settings for DRD operating modes.
Batch	Lets you define the range and scope for running DRC violation checks.
Filters	Lets you define the constraints that you want to check for violations.
Sign-Off Settings	Lets you specify settings for fill shapes.

Interactive

The following table describes the fields available on the *Interactive* tab of the DRC Options form.

Field	Description
DRD Hierarchy Range	Sets the number of levels of hierarchy that are to be considered during DRD editing. The options are: ■ *current cellview*: The current cellview is considered. This is the default option. ■ *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: drdEditHierDepth

Field	Description
Notify	Provides visual feedback in the form of halos and arrows when design rule violations occur.
Enabled	Enables notify editing mode.
Zoom in Incremental Violation	Displays a zoomed-in view of the objects for which violations are reported 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. See DRD Smart Snapping.
Snap Aperture	Sets the aperture size to be used to snap the shape to the nearest valid width.
Enforce	Stops the 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.
Relaxed	Enables the relaxed enforce mode. SeeRelaxed Enforce.
Targets Only	Enables the targeted enforce mode. See Targeted Enforce.
Post Edit	Generates markers if a layout editing operation results in design rule violations.
Off	Disables the Post Edit DRD editing mode.
DRD	Enables the Post Edit DRD editing mode.

Field	Description
MPT Checks (DRD Only)	Checks the design for conformance to the rules defined in the technology file. The options are: ■ *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: drdColorability
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
Consider Gray Shapes	Considers gray shapes as colored and does not report these 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: drdConsiderUnlockedShapesAsGray
Display (DRD Only)	Specifies the display elements that should appear when a constraint is violated.

Field	Description
Arrows	Displays arrows to indicate the optimum spacing between objects. Environment variable: drdEditDisplayArrows
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: drdEditDisplayHalo
True Color	Displays halos in true colors which is the color assigned to a layer. Environment variable: drdEditDisplayTrueColor
Auto Halo Turn Off	Suppresses halos if the number of violations exceeds the specified value. Only arrows are displayed on the canvas when the specified limit is reached. Environment variables: drdEditAutoTurnOffHalo, drdEditAutoTurnOffHaloLimit

Batch

The following table describes the fields available on the *Batch* tab of the DRC Options form.

Field	Description
DRD Hierarchy Range	Sets the number of levels of hierarchy that are to be considered during DRD editing in batch mode. The options are: ■ *current cellview*: The current cellview is considered. This option is selected by default. ■ *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
Signoff Hierarchy Range	Sets the number of levels of the signoff hierarchy are to be considered during DRD editing in batch mode. The options are: ■ *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.

Field	Description
Scope	Defines the scope of checking in DRD batch mode.
Area	Specifies the area of check in DRD batch mode. The options are: ■ *Current Cellview*: Checks all the shapes in the current editable cellview. ■ *Visible Area*: Checks all the shapes in the visible area. ■ *Specified Area***:** Checks all the shapes in the specified area of the design. Enables the *Define Area* button. ■ *Changed Area***:** Checks all the shapes in the changed area. ■ *Partition*: Checks all the shapes in the partition. Environment variable: drdBatchScopeAreaLimit, drdBatchScopeSpecifiedArea
Define Area	Lets you specify the region in which you want to run DRD checks 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 *Specified Area* option is selected.
Marker Limit Per Rule	Specifies the maximum number of post-edit markers that can be created per rule in batch mode. The default is `5000`. Environment variable: drdBatchVioLimit
Total Marker Limit	Specifies the total limit of post-edit markers that can be created in batch mode.

Field	Description
MPT Checks (DRD Only)	Checks the design for conformance to the rules defined in the technology file. The options are: ■ *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
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

The following table describes the fields available on the *Filters* tab of the DRC Options form.

Field	Description
DRD Constraints	Specifies the DRD constraints for which you want to display violations on the canvas.
Search Constraints	Specifies the 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 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.
Instance	Specifies the instances to be excluded from DRD checking.
Ignore Marked Instance	Excludes the instances that are marked as ignored through the *DRD Add Ignore* option in the *Navigator* assistant.
Sign off Rules	Specifies the Sign off rules for which you want to display violations on the canvas.
Show Rule ID	Lets you specify the sign off rule ID instead of the rule type.
Search Rule Type / Search Rule ID	Specifies the strings to filter the sign off rules for which you want DRD to display violations on the canvas.

Field	Description
Rule Category	Lists the sign off rule categories. Point to a rule category to view a list of all the rule types that DRD supports in that category.
Rule Type / Rule ID	Lists the rule type or rule ID filtered by using a string or selecting one or more rule categories.
Exclude Checks	Excludes the following checks: ■ *Connectivity*: Excludes connectivity checks when running DRD violations. ■ *Density*: Excludes density checks.
Exclude Cell File	Specifies the path to the file that lists the cellviews to be excluded when running violation checks. You can also use the editor to create a new file or edit an existing file that lists the cellviews to be excluded.
Layer	Specifies 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.
Check Layer Independent Rules	Checks rules that are independent of layers.
Search Layers	Specifies strings to filter the layers.
Layer Category	Lists the three layer categories, *Routing*, *Device*, and *Derived*.
Layer	Lists the layers filtered 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.

Sign-Off Settings

The following table describes the fields available on the *Sign-Off Settings* tab of the DRC Options form.

Field	Description
Additional Run Options	Specifies the additional options when running DRD checks.
Abort on Layout Errors	Aborts DRD checks when errors are reported.
Override Snapshot Bloat	Overrides the bloat value specified in the selected snapshot.
Snapshots	Lets you create snapshots.
Create	Creates a new snapshot.
Load current snapshot settings	Loads the current snapshot settings.

The following fields and action buttons are common to all the tabs:

Field	Description
Configuration file	Specifies the configuration file. The browse button lets you locate and select the existing configuration file and specify the location at which to save the new configuration file.
Save To	Saves the current settings from the DRC Options form to the specified configuration file.
Load From	Loads settings from the specified configuration file into the DRC Options form.

Related Topics

Setting Up DRC Options

Specifying Options on the DRC Toolbar

Signoff Fill Form

Generates fill shapes in the design.

Field	Description
Input	Specifies the inputs to generate fill shapes.
Snapshot	Specifies the snapshot.
Layer Map	Specifies the layer map file.
Output	Specifies the output fill settings.
Fill Instance Name	Specifies the fill instance name.
Library	Specifies the library name.
Cell	Specifies the cell name.
View	Specifies the view name.
Create Hierarchical Fill	Create a hierarchical fill.
Clear Existing Fill	Clears an existing fill.
Scope	Defines the scope of fill generation.
Area	Specifies the area used to generate the fill. The options are: ■ *Current Cellview*: Fill is generated in the current editable cellview. ■ *Specified Area*: Fill is generated in the specified area. If you choose this option, the *Define Area* button is enabled.
Define Area	Lets you specify the region in which you want to generate the fill. 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 *Specified Area* option is selected.
Use Function	Lets you use a function to define the scope of fill generation.
Advanced Options	Specifies the advanced options for generating a fill.
Symmetry	Specifies the symmetry options when generating a fill. The options are *None*, *X-Axis*, *Y-Axis*, and *Point*.
Mirrored	Specifies whether the fill is mirrored.

Field	Description
Specify Offset	Specifies the offset of the fill.
X-Offset	Specifies the value of the offset on the x-axis.
Y-Offset	Specifies the value of the offset on the y-axis.
Consider	Specifies the layers to be considered when generating sign-off fill. The options are: *All Layers*, *Active Layers*, *Visible Layers*, and *Custom Layers*.
Exclude Cell File	Specifies the file that lists the cellviews to be excluded when generating the sign-off fill. You can also use the editor to create a new file or edit an existing file that lists the cellviews to be excluded.
Snapshots	Lets you create snapshots.
Create	Creates a new snapshot.
Load current snapshot settings	Loads the current snapshot settings.

Related Topics

Setting Up DRC Options

Specifying Options on the DRC Toolbar

B

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

- drdConsiderUnlockedShapesAsGray
- drdEditAllowOrthogonalAdjust
- drdEditAutoText
- drdEditAutoTurnOffHalo
- drdEditAutoTurnOffHaloLimit
- drdEditBatchColorRules
- drdEditBatchHierDepth
- drdEditColorRules
- drdEditDensityTypeCheck
- drdEditDensityTypeCheck
- drdEditDisplayArrows
- drdEditDisplayDashed
- drdEditDisplayDrawingColor
- drdEditDisplayEdges
- drdEditDisplayHalo
- drdEditDisplayMarkers
- drdEditDisplayMaxChars
- drdEditDisplayTechRefs
- drdEditDisplayText
- drdEditDisplayTrueColor
- drdEditEnableBatchCheckOnReadOnlyDesign
- drdEditFontSize
- drdEditFontSize
- drdEditHierDepth
- drdEditIgnoreIntraInstanceChecks
- drdEditMode
- drdEditMode

- drdEditPaletteLayers
- drdEditSlidingWindowSize
- drdEditSlidingWindowSize
- drdEditSmartSnapAllowedWidthSnap
- drdEditSmartSnapAperture
- drdEditVioLimit
- drdEditVioLimit
- drdInteractiveFoundryRulesOnly
- drdMobilityMode
- drdPostEditAllWindows
- drdPostEditEIPAllInstances
- drdPostEditEIPAllWindows
- drdPostEditEIPCheckTopView
- drdPostEditIgnoreModgen
- drdPostEditReplaceAllMarkers
- drdPostEditTimeOut
- drdPostEditVioLimit
- drdUncoloredShapes
- drdUnlocked
- drdUseBlockageForExtCheck
- drdUseNetName
- drdUseNewDerivedLayerEngine
- drdValidLayersGrayCheck
- drdVDRIncludePurposes
- threads

drdBatchColorability

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

Description

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 – DRC Options*
Field	*Batch* (tab) – *MPT Checks – Colorability* (DRC Options Form)

Examples

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

Related Topics

Batch-Check Mode

Batch

Performing Color Checks

drdBatchColorShorts

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

Description

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 – DRC Options*
Field	*Batch* (tab) – *MPT Checks – Color Shorts* (DRC Options Form)

Examples

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

Related Topics

Batch

Batch-Check Mode

Performing Color Checks

drdBatchConsiderGrayShapes

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

Description

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 – DRC Options*
Field	*Batch* (tab) – *MPT Checks – Consider Gray Shapes* (DRC Options Form)

Examples

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

Related Topics

Batch

Batch-Check Mode

Performing Color Checks

drdBatchConsiderUnlockedShapesAsGray

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

Description

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 – DRC*, or *Verify – Design – Process Rules – DRC Options*
Field	*Batch* (tab) – *Consider Unlocked Shapes As Gray* (DRC Options Form)

Examples

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

Related Topics

Interactive

Batch-Check Mode

Performing Color Checks

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

drdBatchReportCoveredNonPcellGrayShapes

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

Description

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 drdBatchUncoloredShapes and 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

Batch-Check Mode

Performing Color Checks

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 – DRC*, or *Verify – Design – Process Rules – DRC Options*
Field	*Batch* (tab) – *Scope – Area* (DRC Options Form)

Examples

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

Related Topics

Batch

Batch-Check Mode

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 – DRC*, or *Verify – Design – Process Rules – DRC Options*
Field	*Batch* (tab) – *Scope – Define Area* (DRC Options Form)

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

Batch-Check Mode

drdBatchUncoloredShapes

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

Description

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 – DRC*, or *Verify – Design – Process Rules – DRC Options*
Field	*Batch* (tab) – *MPT Checks – Uncolored Shapes* (DRC Options Form)

Examples

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

Related Topics

Batch

Batch-Check Mode

Performing Color Checks

drdBatchUnlocked

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

Description

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 – DRC* or *Verify – Design – Process Rules – DRC Options*
Field	*Batch* (tab) – *MPT Checks – Unlocked Shapes* (DRC Options Form)

Examples

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

Related Topics

Batch

Batch-Check Mode

Performing Color Checks

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 – DRC* or *Verify – Design – Process Rules – DRC Options*
Field	*Batch* (tab) – *Scope – Marker Limit* (DRC Options Form)

Examples

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

Related Topics

Batch

Batch-Check Mode

drdColorability

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

Description

DRD checks if a set of shapes can be colored without causing any color rule violations. The default value is `nil`.

GUI Equivalent

Command	*Options – DRC*, or *Verify – Design – Process Rules – DRC Options*
Field	*Interactive* (tab) – *MPT Checks – Colorability* (DRC Options Form)

Examples

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

Related Topics

Interactive

Performing Color Checks

drdColorShorts

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

Description

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 – DRC*, or *Verify – Design – Process Rules – DRC Options*
Field	*Interactive* (tab) – *MPT Checks – Color Shorts* (DRC Options Form)

Examples

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

Related Topics

Interactive

Performing Color Checks

drdConsiderGrayShapes

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

Description

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 – DRC*, or *Verify – Design – Process Rules – DRC Options*
Field	*Interactive* (tab) – *Consider Gray shapes* (DRC Options Form)

Examples

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

Related Topics

Interactive

Performing Color Checks

drdConsiderUnlockedShapesAsGray

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

Description

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 – DRC*, or *Verify – Design – Process Rules – DRC Options*
Field	*Interactive* (tab) – *Consider Unlocked Shapes As Gray* (DRC Options Form)

Examples

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

Related Topics

Interactive

Performing Color Checks

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

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

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 – DRC*, or *Verify – Design – Process Rules – DRC Options*
Field	*Interactive* (tab) – *Display – Auto Halo Turn Off* (DRC Options Form)

Examples

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

Related Topics

Interactive

Managing Interactive Display Options

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 – DRC*, or *Verify – Design – Process Rules – DRC Options*
Field	*Interactive* (tab) – *Display – Auto Halo Turn Off* (Halo limit value between `1` and `1000`) (DRC Options Form)

Examples

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

Related Topics

Interactive

Managing Interactive Display Options

drdEditBatchColorRules

```
layout drdEditBatchColorRules int integer_number
```

Description

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 – DRC*, or *Verify – Design – Process Rules – DRC Options*
Field	*Batch* (tab) – *MPT Checks* (DRC Options Form)

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

Batch-Check Mode

Performing Color Checks

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 – DRC*, or *Verify – Design – Process Rules – DRC Options*
Field	*Batch* (tab) – *Hierarchy Range* (select *current to user level* from the drop-down list and enter a value in the field provided) (DRC Options Form)

Examples

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

Related Topics

Batch

Batch-Check Mode

drdEditColorRules

```
layout drdEditColorRules int integer_number
```

Description

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

GUI Equivalent

Command	*Options – DRC*
	Verify – Design – Process Rules – DRC Options
Field	*Interactive* (tab) – MPT Checks (DRC Options Form)

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

Performing Color Checks

drdEditDensityTypeCheck

```
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

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 – DRC*, or *Verify – Design – Process Rules – DRC Options*
Field	*Interactive* (tab) – *Display – Arrows* (DRC Options Form)

Examples

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

Related Topics

Interactive

Managing Interactive Display Options

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

drdEditDisplayHalo

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 – DRC*, or *Verify – Design – Process Rules – DRC Options*
Field	*Drawing Color* (DRC Options Form)

Examples

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

Related Topics

Managing Interactive Display Options

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 – DRC*, or *Verify – Design – Process Rules – DRC Options*
Field	*Interactive* (tab) – *Display – Edges* (DRC Options Form)

Examples

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

Related Topics

Interactive

Managing Interactive Display Options

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 – DRC*, or *Verify – Design – Process Rules – DRC Options*
Field	*Interactive* (tab) – *Display – Halo* (DRC Options Form)

Examples

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

Related Topics

drdEditDisplayTrueColor

Interactive

Managing Interactive Display Options

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 – DRC*, or *Verify – Design – Process Rules – DRC Options*
Field	*Interactive* (tab) – *Post Edit – Enabled* (DRC Options Form)

Examples

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

Related Topics

DRD Operating Modes

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 `";"`).

Important

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

drdEditDisplayTechRefs

drdEditDisplayText

drdEditFontSize

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 – DRC*, or *Verify – Design – Process Rules – DRC Options*
Field	*Interactive* (tab) – *Display – Ref Text* (DRC Options Form)

Examples

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

Related Topics

drdEditDisplayText

Interactive

Managing Interactive Display Options

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 – DRC*, or *Verify – Design – Process Rules – DRC Options*
Field	*Interactive* (tab) – *Display – Rule Text* (DRC Options Form)

Examples

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

Related Topics

drdEditDisplayTechRefs

Interactive

Managing Interactive Display Options

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 – DRC*, or *Verify – Design – Process Rules – DRC Options*
Field	*Interactive* (tab) – *Display – True Color* (DRC Options Form)

Examples

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

Related Topics

drdEditDisplayHalo

Interactive

Managing Interactive Display Options

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`

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

drdEditDisplayMaxChars

drdEditDisplayTechRefs

drdEditDisplayText

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 – DRC*, or *Verify – Design – Process Rules – DRC Options*
Field	*Interactive* (tab) – *Hierarchy Range* (select *current to user level* from the drop-down list and enter a value in the field provided) (DRC Options Form)

Examples

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

Related Topics

Interactive

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

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&notify` enables DRD editing in both Notify and Enforce modes.

GUI Equivalent

Command	*Options – DRC*, or *Verify – Design – Process Rules – DRC Options*
Field	*DRD Mode* (DRC Options Form)

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 Operating Modes

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 – DRC*, or *Verify – Design – Process Rules – DRC Options*
Field	*Filters* (tab) – *Layer – Palette* (DRC Options Form)

Examples

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

Related Topics

Filters

Selecting Layers for Verification

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

drdEditSmartSnapAllowedWidthSnap

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

Description

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

drdEditSmartSnapAperture

```
layout drdEditSmartSnapAperture float float_number
```

Description

Sets the aperture size to be used to snap the shape to the nearest valid width when drdEditSmartSnapAllowedWidthSnap is enabled. The default value is 0.01.

GUI Equivalent

None

Examples

```
envGetVal("layout" "drdEditSmartSnapAperture")
envSetVal("layout" "drdEditSmartSnapAperture" 'float 0.2)
```

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 Operating Modes

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

drdMobilityMode

```
layout drdMobilityMode cyclic { "off" | "low" | "medium" | "high"}
```

Description

Controls the DRD mobility feature, which defines how DRD notification functions based on the movement of the pointer on the canvas.

The default is `high`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "drdEditMobilityMode")
envSetVal("layout" "drdEditMobilityMode" 'cyclic "off")
envSetVal("layout" "drdEditMobilityMode" 'cyclic "low")
envSetVal("layout" "drdEditMobilityMode" 'cyclic "medium")
envSetVal("layout" "drdEditMobilityMode" 'cyclic "high")
```

Related Topics

None

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

drdPostEditEIPAllInstances

```
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 Operating Modes

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

drdPostEditEIPAllInstances

drdPostEditEIPCheckTopView

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

drdPostEditEIPAllInstances

drdPostEditEIPAllWindows

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 Operating Modes

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 Operating Modes

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 – DRC*, or *Verify – Design – Process Rules – DRC Options*
Field	*Timeout (secs)* (DRC Options Form)

Examples

```
envGetVal("layout" "drdPostEditTimeout")
envSetVal("layout" "drdPostEditTimeout" 'float 9.0)
```

Related Topics

DRD Operating Modes

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 – DRC*, or *Verify – Design – Process Rules – DRC Options*
Field	*Interactive* (tab) – *Post Edit – Marker Limit* (DRC Options Form)

Examples

```
envGetVal("layout" "drdPostEditVioLimit")
envSetVal("layout" "drdPostEditVioLimit" 'int 1)
envSetVal("layout" "drdPostEditVioLimit" 'int 5000)
```

Related Topics

DRD Operating Modes

Interactive

drdUncoloredShapes

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

Description

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 – DRC*, or *Verify – Design – Process Rules – DRC Options*
Field	*Interactive* (tab) – *MPT Checks – Uncolored Shapes* (DRC Options Form)

Examples

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

Related Topics

Interactive

Performing Color Checks

drdUnlocked

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

Description

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 – DRC*, or *Verify – Design – Process Rules – DRC Options*
Field	*Interactive* (tab) – *MPT Checks – Unlocked Shapes* (DRC Options Form)

Examples

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

Related Topics

Interactive

Performing Color Checks

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

drdUseNetName

```
layoutXL drdUseNetName boolean { t | nil }
```

Description

See drdUseNetName in the *Virtuoso Layout Suite XL User Guide*.

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

drdValidLayersGrayCheck

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

Description

Indicates that only the colorable layers specified in the `validLayers` 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)
```

drdVDRIncludePurposes

```
layout drdVDRIncludePurposes string { "" | "fill" }
```

Description

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`.

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

threads

`levEnv.drdEdit threads float` *`number_of_threads`*

Description

Specifies the number of threads for DRD editing.

- `0`: Use maximum twelve threads
- `1`: Disable multi-threading
- `Greater than 1`: Use the number of threads specified by the user

The default value is `0`.

GUI Equivalent

None

Examples

```
envGetVal("levEnv.drdEdit" "threads")
envSetVal("levEnv.drdEdit" "threads" 'float 2)
```

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

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

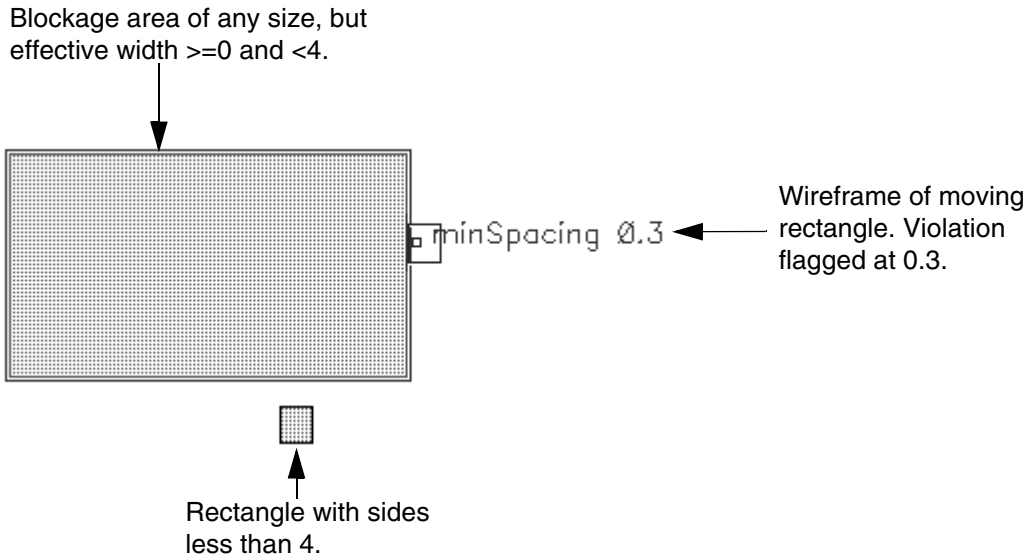
You can alter the effective width of a blockage object to flag or prevent spacing violations when using DRD while creating or editing certain shapes. The effective width applies to the entire blockage. The larger of the two values, the blockage effective width or the side of the non-blockage object, 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.

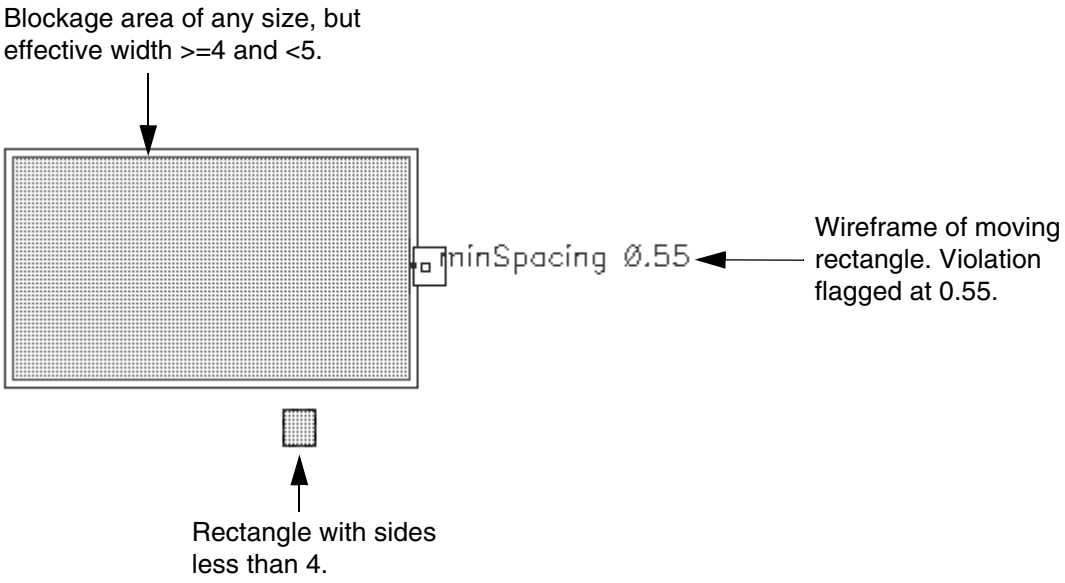
Suppose that the technology file contains the following spacing values for `Metal3`:

```
( minSpacing "Metal3"
    (( "width" nil nil ))
    (
      0  0.3
      4  0.55
      5  0.65
    )
)
```

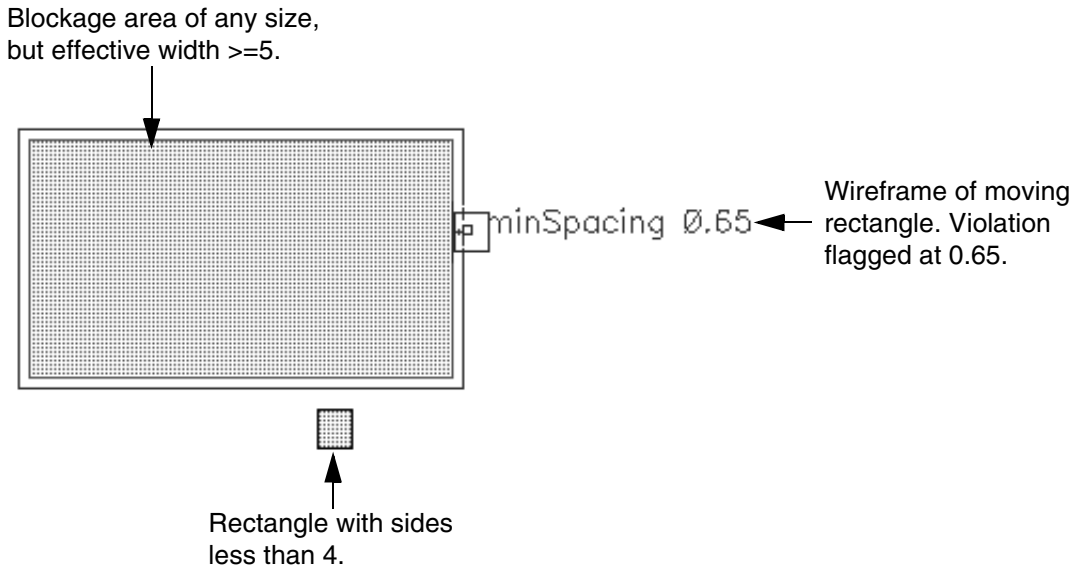
When the effective width of the blockage is set to a value greater than or equal to 0 and less than 4, the spacing violation is flagged at 0.3, as shown below:



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:



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 `metal1` 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.

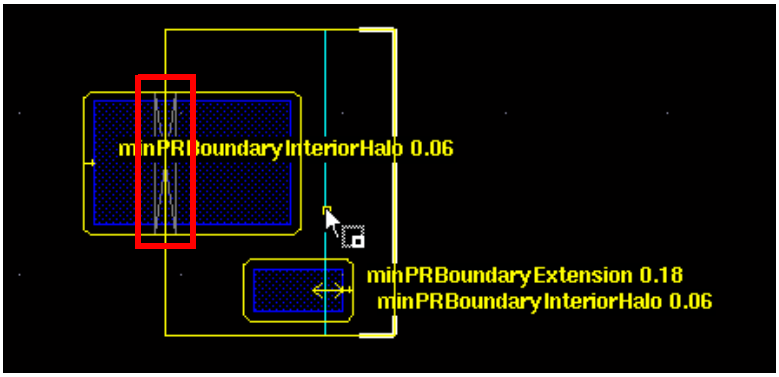
  ```
  ( minPRBoundaryExtension "metal2" 1.0 )
  ```

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

marker that was created when the PR boundary object was placed on an already existing rectangular shape.

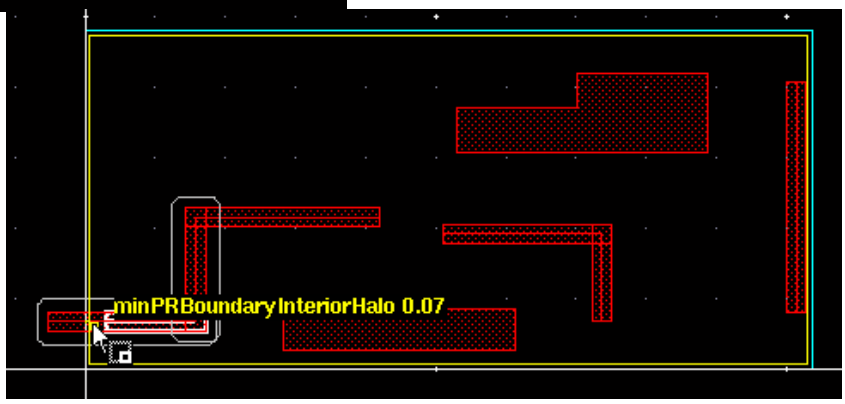


Creating and Editing Non-PR Boundary Objects

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.

DRD Enforce mode

DRD Notify mode



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 DRC Options Form form.